IEEE International Conference on Robotics and Automation, New Orleans, LA, April 26-May 1, 2004, pp. 3863-3869.

A Method for Mobile Robot Navigation on Rough Terrain

Cang Ye, Member, IEEE and Johann Borenstein, Member, IEEE
The University of Michigan
yecang@umich.edu and johannb@umich.edu

Abstract—This paper presents a new obstacle negotiation method for mobile robot navigation on rough terrain. The proposed method first transforms a local terrain map surrounding the robot's momentary position into a grid-type traversability map by extracting the slope and roughness of a terrain patch through least-squares plane-fitting. It then computes so-called "*polar obstacle densities*" for the cells in the traversability map and transforms the traversability map into a *traversability field histogram,* from which the velocity and the steering command for the robot are determined. Simulation results show that the algorithm is able to navigate the robot to the target with a finite-length path.

Keywords—obstacle negotiation, obstacle avoidance, terrain traversability analysis, traversability map, vector field histogram, traversability field histogram, motion-context.

I. Introduction

Autonomous navigation of mobile robots on rough terrain requires the ability to decide if the terrain and/or obstacle ahead are traversable or if they have to be circumnavigated. This, together with the ability to generate the appropriate steering and speed commands for the robot, is termed Obstacle Negotiation (ON). There are two closely related issues in ON—terrain mapping and path planning [1]. This paper relates to path planning and it assumes that a terrain map is available from our recently developed terrain mapping method [2, 3].

Much work has been done on path planning for rough terrain, which addresses two problems: Terrain Traversability Analysis (TTA) and path generation. Langer et al [4] 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 [5] 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.

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 and Howard [7] proposed a new 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.

In this paper, we concern ourselves only with local obstacle negotiation. Local or reflexive obstacle avoidance is often seen as a *behavior*, and behavior-based navigation methods have been employed in a number of research projects [4, 7]. For instance, the system described in [7] 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 for decision-making. 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 command for the robot. This method is similar to the flat ground navigator in [8].

Some researchers employed the so-called "arcs approach" [4, 6, 9]. In the arcs approach the algorithm generates a number of candidate arcs and then either votes for the arc with the largest clearance [4] or it calculates the costs along each arc and selects the one with the lowest cost [6, 9]. The robot is then 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 [5] 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 [10]. However, the application of potential field methods to obstacle avoidance may result in oscillatory

motion [11] or cause the robot to get trapped in a local minimum. The Vector Field Histogram (VFH) method, developed by one of the co-authors [12] 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.

In this paper, we modify and extend the capabilities of the VFH algorithm and apply the modified algorithm to navigation on rough terrain. 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 in greater detail our Terrain Traversability Analysis algorithm and our Path Planning algorithm, respectively. In Section V, we present simulation results for the overall ON algorithm. Section VI concludes the paper.

II. Overview of the Proposed Traversability Field Histogram Method

Figure 1 shows a block diagram of our ON system. The system consists of four main modules: Terrain Mapping, Terrain Traversability Analysis (TTA), Path Planning, and Motion Control. In this section we give a brief overview of our system, before we focus on the main subject of this paper, the TTA and the related path planning algorithm.

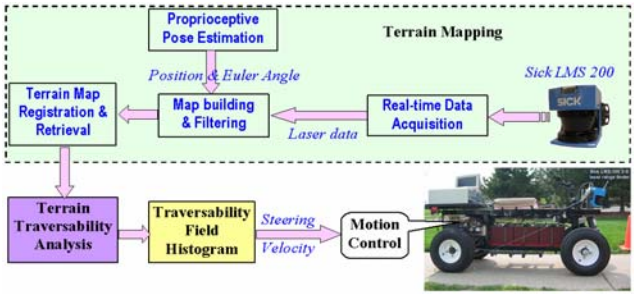


Fig. 1 Diagram of the obstacle negotiation system: the module within the dashed lines represents the Terrain Mapping module

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

The task of the TTA module is to analyze the terrain map cell by cell and to generate a new 2-D grid-type map, called the "traversability map." Each cell in that map holds a value that expresses the degree of difficulty for the robot to move across that cell. This value is called the Traversability Index (TI).

The Path Planning module is a local path planner. It analyzes the traversability map and generates steering and velocity commands to avoid cells with high TIs. TTA and path planning are the main subjects of this paper. The method described in the following Sections has been tested with a few real data sets (terrain maps) and many more simulated data sets. The real data was collected with the system shown in Fig. 1.

III. Terrain Traversability Analysis (TTA)

1. T-Transformation and 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 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 $\mathbf{n}$ 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

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

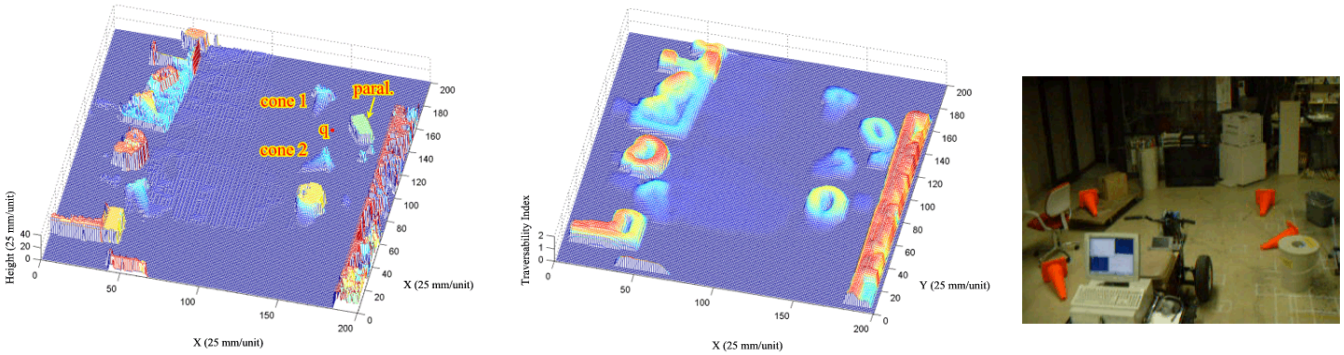
where d_i is the distance between the i^{th} data and the fitted plane. Finally, the TI of cell (i, j) is calculated by

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

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 $\mathbf{E}$ is transformed into a traversability map $\mathbf{T} = \{\tau_{i,j}\}$. We denote this transformation $\mathbf{T} = T(\mathbf{E})$ and call it T-transform. Fig. 2a 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. 2c. Fig. 2b shows the traversability map obtained by the TTA algorithm.

In a grid-type terrain map, an obstacle's boundary comprises line segments in X-, Y, or diagonal direction. The T-transform grows the boundary aligned with X- or Y-direction by L cells and it grows the boundary aligned diagonally by $\sqrt{2}L$. This automatically takes into account the robot's dimensions during



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

Fig. 2 Traversability analysis on the elevation map built by the terrain mapping system: the TTA algorithm using a patch size of 9 cells × 9 cells

navigation and it allows us to treat the robot as a point in the path-planning algorithm described in the following section.

IV. Path Planning Using the Traversability Field Histogram

The VFH method was proposed in [12] for fast obstacle avoidance. In this paper, we employ the VFH concept and propose a new local path planning method, the Motion-context Guided Traversability Field Histogram (MGTFH) algorithm for obstacle negotiation (ON). The MGTFH method extends the VFH's capability from obstacle avoidance on flat surfaces to ON on 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.

The MGTFH 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 a result of this choice, S^* covers a physical region of 9.2 × 9.2 m (37×250 mm = 9.2 m).

Next, the MGTFH algorithm performs the TTA over the cells in S^* and transforms the terrain map into a traversability map. Then the MGTFH algorithm computes the motion command based on the traversability map inside S^*. The computation consists of two stages: (1) data reduction to a one-dimensional polar histogram, and (2) generation of a motion command. These two stages are explained next.

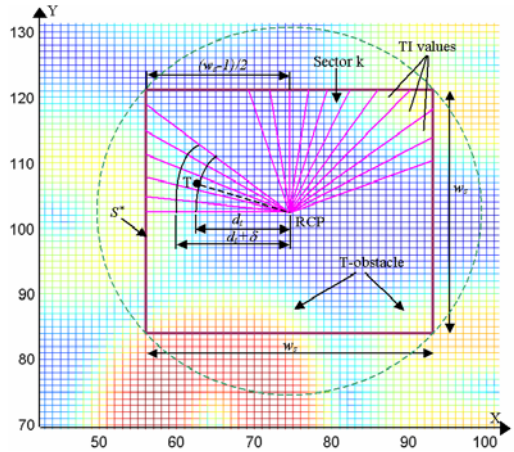


Fig. 3 Transformation of a traversability map into a polar histogram: X- and Y-axes represent the world coordinates.

1. *Creating the Polar Histogram*

Figure 3 is the 2-D representation of a traversability map, where the TI value is rendered in color. Cells with nonzero TI values in the traversability map generate 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" and we term the polar histogram **H** "traversability field histogram." Since we are extending the VFH method [12] to handle ON problems, we adopt the same terms, "obstacle vector" and "polar obstacle density" as in [12]. The exact definition of these terms follows.

The contents of each cell in the traversability map is treated as an obstacle vector [12] whose direction is calculated by

$$\beta_{i,j} = \tan^{-1}\frac{y_j - y_o}{x_i - x_o} \tag{4}$$

and the magnitude is given by

$$m_{i,j} = \tau_{i,j}^2 (a - b d_{i,j}) \tag{5}$$

where

$\beta_{i,j}$ is the direction from cell (i, j) in S^* to the RCP,
x_i, y_j are the coordinates of cell (i, j),
x_o, y_o are the present coordinates of the RCP,
$m_{i,j}$ is the magnitude of the obstacle vector at cell (i, j).
$\tau_{i,j}$ is the TI value of cell (i, j),
a, b are positive constants, and
$d_{i,j}$ is the distance between cell (i, j) and the RCP,

In equation (5), $\tau_{i,j}$ is squared such that in the polar histogram the impact of a cell with a small/large TI value is diminished/magnified. Since $m_{i,j}$ is proportional to $-d$, cells with non-zero TIs cause larger vector magnitudes when they are closer to the robot and smaller ones when they are farther away. a and b are chosen such that $a - bd_{\max} = 0$, where $d_{\max} = \sqrt{2}(w_s - 1)/2$ is the distance between the farthest cell

of S^* and the RCP. This arrangement guarantees $m_{i,j} = 0$ for the four farthest cells (the four vertices) of S^*.

As shown in Fig. 3, S^* is divided into n sectors, each of which has an angular resolution of α (α is chosen such that $360/\alpha$ is an integer). Each sector k, for $k = 0, \cdots, n-1$, corresponds to a discrete angle $\rho = k\alpha$ in **H**. The cells (i, j) in S^* are assigned to the k^{th} sector according to

$$k = \text{int}(\beta_{i,j}/\alpha) \tag{6}$$

For each sector k, the polar obstacle density h_k is then calculated by

$$h_k = \sum_{i,j} m_{i,j} \tag{7}$$

In our implementation α = 5°, therefore, there are 72 sectors. Polar obstacle densities produced by this equation are discrete, hence **H** is discontinuous. This may lead to drastic change in motion since **H** is used to determine the robot's steering and velocity. In order to alleviate this problem, we use the following function to smooth the polar obstacle density:

$$h_k' = \frac{\sum_{l=-p}^{p} h_{k-l}}{2p+1} \tag{8}$$

where h_k' is the Smoothed Polar Obstacle Density (POD). The parameter p determines how much the Polar Histogram is smoothed, and we found that p=3 produces the best results.

The POD of each sector represents the level of difficulty of moving in the corresponding direction. Figure 4 shows the polar histogram **H** for the momentary situation of Fig. 2a, where the RCP is located at cell (150, 144) (labeled 'q' in Fig. 2a). The parallelepiped object produced large PODs in sectors

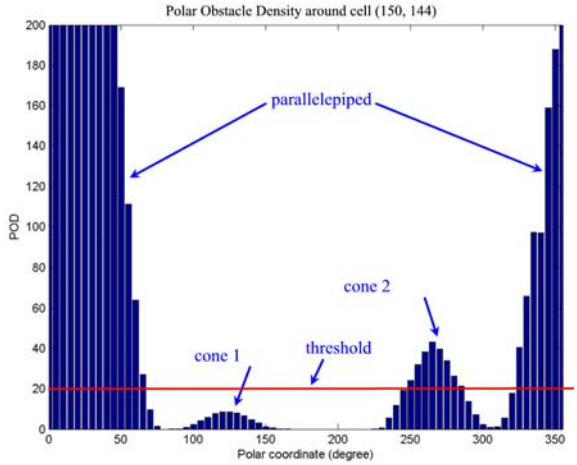


Fig. 4 This smoothed polar histogram is was created when the RCP was at point '*q*' in Fig. 2a

1-14 and sectors 64-72 (i.e., -40°–70°) because it is high in elevation. Cone 1 generated small PODs in sectors 19-30 (i.e., 95°–150°) since its low-profile tip faced the robot, whereas cone 2 produced relatively larger PODs in sector 47-60 (i.e., 235°–300°) because its base (with its higher elevation) faced the robot. This exemplifies how H correctly reflects the overall traversability of each sector in region S^*.

2. *Steering Control*

When the robot travels across feature-rich terrain, the polar histogram **H** changes from instance to instance. The objective of the ON algorithm is to steer the robot into a direction with lower PODs, while still maintaining a direction that is close to the target direction. As can be seen from Fig. 4, a polar histogram typically has "hills" (sectors with high PODs) and "valleys" (sectors with low PODs). A "candidate valley" is a cluster of consecutive sectors with PODs 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 "*wining valley*." In the basic VFH algorithm, 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

$$k_h = \begin{cases} k_w^R + \min(s_w, s_{\max})/2 & \text{if } |k_t - k_w^R| \leq |k_t - k_w^L| \\ k_w^L - \min(s_w, s_{\max})/2 & \text{otherwise} \end{cases} \tag{9}$$

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 [11] 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 suggests. Simply selecting the candidate valley closest to the target direction as the winning valley (we call this the *closest-valley-wins* scheme) may result in so-called "trap" situations. When the robot is in motion, the POD of each sector changes all the time. This may switches 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

$$\mu = sign(k_h - k_t), \tag{10}$$

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.

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

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 in clockwise (cw) direction because the avoidance maneuver was initiated by a left-turn. . 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 POD 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.

3. 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 POD 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 POD value in the momentary direction of travel:

$$v' = v_{\max}(1 - \frac{\min(h_c', h_m)}{h_m}) \tag{11}$$

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

$$v'' = \frac{v' \min(d_t, d_m)}{d_m} \tag{12}$$

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

$$V = \frac{v'' l_{xyz}}{l_{xy}}, \tag{13}$$

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 μ=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.

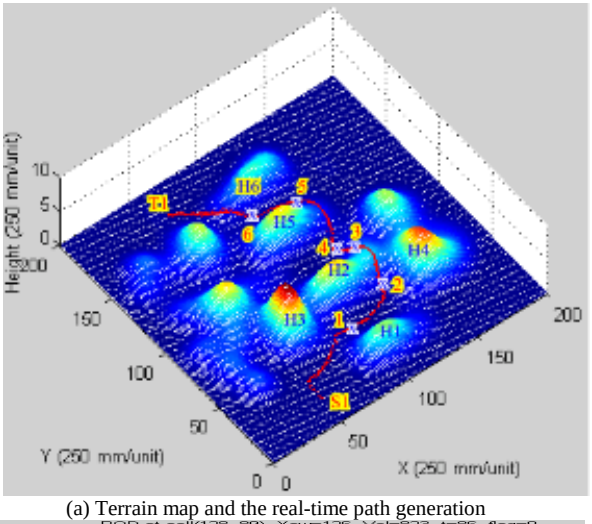
V. SIMULATION RESULTS

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. As noted before, the robot's size determines the grid size of the terrain map as follows: the boundary of a terrain patch, which has a size of 9×9 cells must fully envelop the

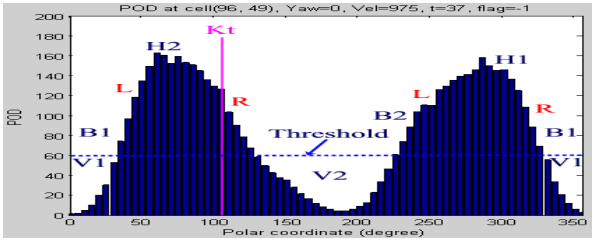
robot at any orientation. For our Gorilla platform this translates into a cell size of 250×250 mm.

In this section, we present results obtained with our ON simulator. Figure 5a 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.

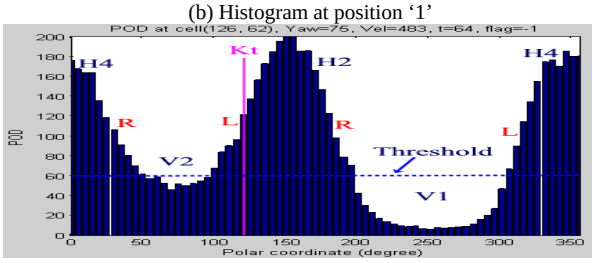
Figures 5b-g show snapshots of the polar histograms for points 1-6 on the robot's path. The robot first steered to the passage between hills 'H1' and 'H7' and moved to point '1'. The polar histogram at this point is displayed in Fig. 5b. Since the motion-context was –1, the left border of valley 'V1' 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



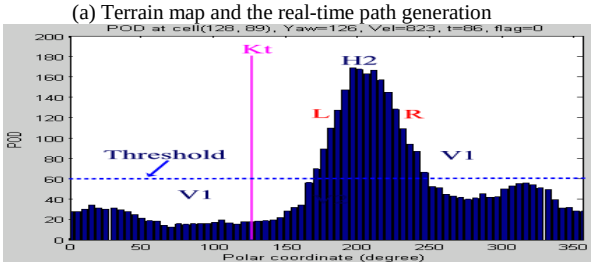
(a) Terrain map and the real-time path generation



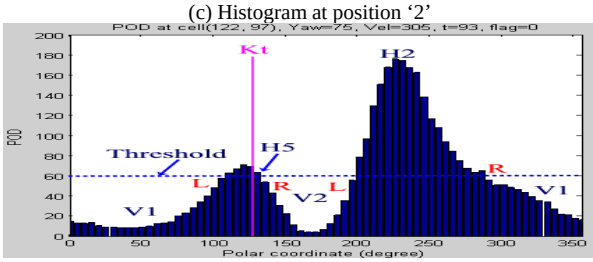
(b) Histogram at position '1'



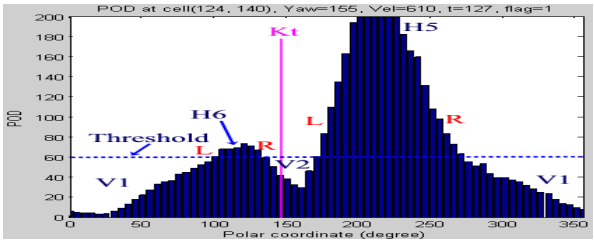
(c) Histogram at position '2'



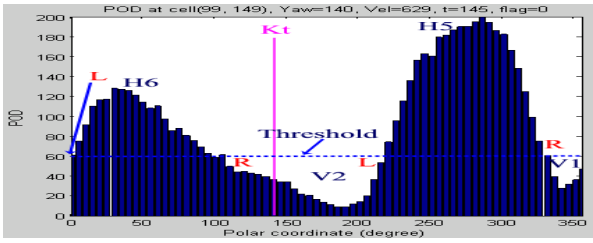
(d) Histogram at position '3'



(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.

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. 5c. 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. 5e) 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. 5f), 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. 5f). 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 achieved 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. 5a as an example, the roll and pitch angles of the robot using the ON algorithm are within [-6, 6.5] and [-8, 9], respectively.

Finally, our complexity analysis on the MGTFH algorithm reveals that it requires ~171,568 operations to determine the motion commands. It takes ~969 μs on our AMD Athlon XP 1800+ processor-based PC. This makes our algorithm suitable for real-time implementation.

VI. Discussion and Conclusions

In conclusion, we have presented a novel obstacle negotiation method for mobile robots traversing rough terrain. In the proposed method, we introduce the new concept of the Traversability Field. This approach allows us to convert the ON negotiation problem into the framework of the Vector Field Histogram method, which was developed in earlier work for obstacle avoidance on flat terrain.

The algorithm first employs an analytical terrain traversability analysis method to transform the local terrain map into a traversability map. It then reduces the amount of data in the 2-dimensional traversability map by transforming it into a 1-dimensional traversability field histogram, from which the velocity and the steering command are determined. To overcome trap situations and potential fluctuation in heading with the *closest-valley-wins* scheme, we proposed the motion-context based algorithm. Simulation runs show that the proposed method reliably guides the mobile robot across moderate terrain, while avoiding high profiled terrain features and/or obstacles. The algorithm is practical for real-time implementation and is currently being tested on a real robot, the Segway Robotics Mobility Platform (RMP).

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

1 A. Kelly and A. Stentz, “Rough terrain autonomous mobility,” *Autonomous Robots*, vol. 5. no.2, pp. 129-198, 1998.

2 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.

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

4 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

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

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 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, 2002.

8 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.

9 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.

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

11 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.

12 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.