Behavior maps for online planning of obstacle negotiation and climbing on rough terrain

Christian Dornhege and Alexander Kleiner
Institut für Informatik
University of Freiburg
79110 Freiburg, Germany
{*dornhege, kleiner*}*@informatik.uni-freiburg.de*

***Abstract*— To autonomously navigate on rough terrain is a challenging problem for mobile robots, requiring the ability to decide whether parts of the environment can be traversed or have to be bypassed, which is commonly known as Obstacle Negotiation (ON). In this paper, we introduce a planning framework that extends ON to the general case, where different types of terrain classes directly map to specific robot skills, such as climbing stairs and ramps. This extension is based on a new concept called *behavior maps*, which is utilized for the planning and execution of complex skills. Behavior maps are directly generated from elevation maps, i.e. two-dimensional grids storing in each cell the corresponding height of the terrain surface, and a set of skill descriptions. Results from extensive experiments are presented, showing that the method enables the robot to explore successfully rough terrain in real-time, while selecting the optimal trajectory in terms of costs for navigation and skill execution.**

I. INTRODUCTION

To autonomously navigate on rough terrain is a challenging problem for mobile robots, requiring the ability to decide which parts of the environment can be traversed or have to be bypassed, which is commonly known as Obstacle Negotiation (ON) [15]. While this problem has been mainly addressed for semi-rough terrain, e.g. within outdoor [9], [11], [13], and indoor scenarios [17], rough terrain, containing steep structure elements, demands robots to be explicitly aware of their context, thus allowing to execute specific skills with respect to the situation.

In this paper, we introduce a planning framework that extends ON to the general case, where different types of terrain classes directly map to specific robot skills, such as climbing stairs and ramps. The framework is based on a new concept called *behavior maps*, which is utilized for the planning and execution of complex skills on rough terrain. Behavior maps are directly generated from elevation maps, i.e. two-dimensional grids storing in each cell the corresponding height of the terrain surface [3], [8], and a set of skill descriptions. Skill descriptions contain a set of fuzzy rules for the classification of structures they can handle, a set of spacial constraints encoding preconditions required for their execution, a cost function utilized for A* planning on the map, and the skill routine to be executed. According to these skill descriptions, elevation maps are segmented into different regions, whereas each region corresponds to a skill that can be executed therein. We utilize Markov Random Field (MRF) models, which are automatically constructed from the set of fuzzy rules, for detecting structure elements on the elevation map. Furthermore, behavior maps are augmented with preconditions, such as starting locations and angles, which are automatically generated from the sets of spacial constraints. The resulting 2D representation encodes context information of the environment, and can efficiently be utilized for the planning and execution of skills. The final system consists of four main modules, which are all executed in real-time during navigation: elevation mapping, terrain classification, skill planning, and motion control.

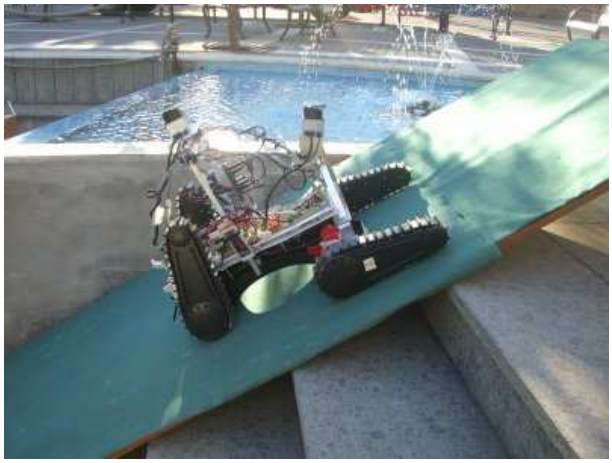


Fig. 1. The *Lurker* robot climbing up a ramp during the Rescue Robotics Camp 2006 in Rome.

We conducted extensive experiments within an indoor environment containing pallets and ramps, which was designed accordingly to the testing arenas build by NIST for evaluating robots for USAR (Urban Search And Rescue) [7]. Results from these experiments show that the robot was able to successfully explore the environment in real-time, while selecting the optimal trajectory in terms of costs for navigation and skill execution.

Former work mainly focused on behavior-based approaches for robot navigation on rough terrain. For example, Herbert and colleagues developed a tracked robot for scrambling over rubble, whereas autonomous

navigation was based on visual servoing and stereo vision-based obstacle avoidance [5]. Ye and Borenstein extracted traversability field histograms from elevation maps for obstacle avoidance on a Segway platform [16]. While purely behavior-based approaches are successful on moderately rugged 3-D terrain, they might fail within densely structured environments due to a limited lookahead needed for concluding efficient sequences of skills.

Methods for terrain classification [9], [11], [13] and terrain trafficability characterization [10], have been extensively studied in the past. These methods mainly focus on the question whether a terrain feature can be traversed or has to be bypassed, e.g. they classify the terrain into classes such as road, grass, and rock.

The remainder of this paper is structured as follows. In Section II we describe the utilized hardware platform, in Section III we explain the construction of behavior maps, and in Section IV planning on these maps is described. Finally, we provide results from robot experiments in Section V and conclude in Section VI.

II. HARDWARE PLATFORM

Figure 1 shows the tracked *Lurker* robot, which is based on the *Tarantula* R/C toy. Although based on a toy, this robot is capable of climbing difficult obstacles, such as stairs, ramps, and random stepfields [7]. This is possible due to its tracks, which can operate independently on each side, and the "Flippers" (i.e. the four arms of the robot), which can freely be rotated at 360°. The base has been modified in order to enable autonomous operation. First, we added a 360° freely turnable potentiometer to each of the two axes for measuring the angular position of the flippers. Second, we added 10 touch sensors to each flipper, allowing the robot to measure physical pressure when touching the ground or an object.

Furthermore, the robot is equipped with a 3-DOF Inertial Measurement Unit (IMU) from *Xsens*, allowing drift-free measurements of the three Euler angles *yaw*, *roll*, and *pitch*, and two *Hokuyo URG-X004* Laser Range Finders (LRFs), one for scan matching, and one for elevation mapping, which can be tilted in the pitch angle within 90°.

III. BUILDING BEHAVIOR MAPS

In this section we describe the process of building behavior maps, which can be applied online during elevation mapping. Elevation maps are generated from the successive integration of the robot's pose, and data measured by a downwards tilted Laser Range Finder (LRF). For a more detailed description of building elevation maps the reader is referred to [8].

Compared to conventional world representations, as for example occupancy maps, behavior maps contain context information for negotiating different kinds of structures, such as ramps and stairs. Within the proposed planning framework they serve as a basis for skill planning and execution, and are automatically generated from elevation maps and a set of skill descriptions. For this purpose skills are implementing a set of *fuzzy rules* for the classification of structures they can handle, a set of *spacial constraints* encoding preconditions required for their execution, a *cost function* utilized for A* planning and the *skill routine* to be executed. Figure 2 summarizes the process of constructing behavior maps, and their application for planning and skill execution. Note that the fuzzy rule sets are utilized for both pre-classification and a MRF-based (Markov Random Field) classification. The pre-classification allows real-time extraction of trivially decidable regions, such as floors and walls, whereas the MRF-based classification is executed delayed in the background in order to detect undecided regions, such as ramps and stairs, more reliably. In the following, all components of skill descriptions are discussed in more detail.

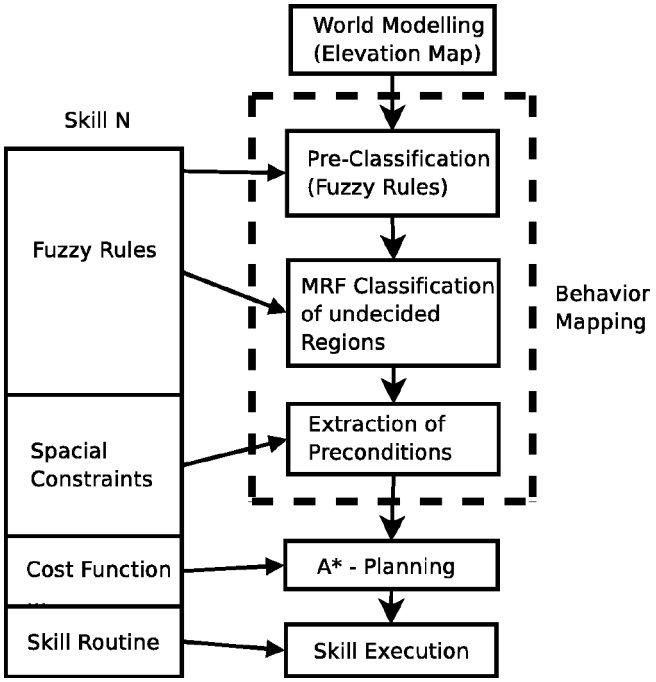


Fig. 2. The planning framework based on behavior maps, which are generated from elevation maps and skill descriptions.

A. Fuzzy rules for structure detection

Depending on the structure element to be recognized, both classification methods need to have a set of representative features that can differentiate the structure element from the environment. These features are generated for each cell of the elevation map by combinations of fuzzy rules that project parameters, as for example, the height difference between cells, into the $[0, 1]$ interval. Fuzzy rules are generated by the functions $SUp(x, a, b)$ (Equation 1) and $SDown(x, a, b)$ (Equation 2), whereas

a and b denote the desired range of the parameter.

$$SUp(x,a,b) = \begin{cases} 0 & \text{if } x < a \\ \frac{x-a}{b-a} & \text{if } a \leq x \leq b \\ 1 & \text{if } x > b \end{cases} \quad (1)$$

$$SDown(x,a,b) = 1 - SUp(x,a,b) \quad (2)$$

For example, the features *Flat Surface*, *Ramp Angle* and *Steep Angle* are build from the parameters δh_i, denoting the maximum height difference around a cell, and α_i, denoting the angle between the normal vector $\mathbf{n_i}$ and the upwards vector $(0,1,0)^T$, as shown by Equation 3 and Equation 4, respectively.

$$\delta h_i = \max_{j \text{ is neighbor to } i} |h_i - h_j| \quad (3)$$

$$\alpha_i = (0,1,0)^T \cdot \mathbf{n_i} = n_{i_y} \quad (4)$$

Given these parameters, the features are defined, for example, by:

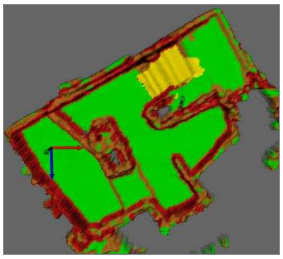
- Flat Surface $= SDown(\delta h_i, 15, 40)$
- Ramp Angle $= SUp(\alpha_i, 3, 25) \cdot SDown(\alpha_i, 25, 40)$
- Steep Angle $= SUp(\alpha_i, 25, 90)$

Each time the elevation map is updated, the pre-classification procedure applies these fuzzy rules on new height estimates in order to classify them into regions, such as *flat ground*, *wall*, and *undecidable region*. The procedure is executed in real-time, and is the first step to classify cells into traversable and non-traversable terrain. However, in order to be able to discriminate between different obstacle types, e.g. to detect a ramp, more sophisticated techniques, such as Markov Random Field (MRF) models, are required. Since MRFs do not linearly scale with the size of the map, we apply them exclusively on undecided regions which have not been classified by the pre-classification procedure. In order to extract those regions from the map, we utilize a color thresholding technique borrowed from computer vision [4]. This is carried out by generating a color image from the pre-classified elevation map, whereas each color corresponds to an obstacle class. The image is then processed by the thresholding technique, resulting in a segmentation of these areas into connected regions, which can separately be processed by MRFs. An example of the result of color thresholding applied on an elevation map is shown in Figure 3.

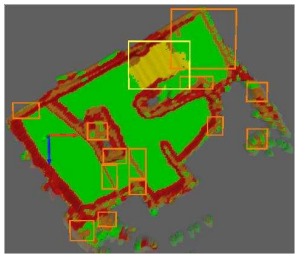
B. Markov Random Fields

To recognize complex structures from noisy data is generally a challenging task which can hardly be solved by heuristics applied on single cells of the elevation map. Hence, we utilize Markov Random Fields (MRFs) for modeling the neighborhood relations between them.

A MRF is defined by an undirected graph $\mathcal{G} = (\mathrm{Y}, \mathcal{E})$, where Y is a set of discrete variables $\mathrm{Y} = \{Y_1, \ldots, Y_N\}$, and $\mathcal{E}$ is a set of edges between them. Each variable $Y_i \in \{1, \ldots, K\}$ can take on one of K possible states. Hence, $\mathcal{G}$ describes a joint distribution over $\{1, \ldots, K\}^N$.



The input map



Extracted regions of interest

Fig. 3. Extraction of regions from a pre-classified map, which can be further classified by MRFs, whereas brown rectangles indicate *obstacle* regions, and yellow rectangles indicate *ramp* regions.

According to the approach of Anguelov and his colleagues [2], we utilize *pairwise* Markov networks, where a potential $\phi(y_i)$ is associated to each node and a potential $\phi(y_i, y_j)$, to each undirected edge $\mathcal{E} = \{(ij)\}(i < j)$ between two nodes. Consequently, the pairwise MRF model represents the joint distribution by:

$$P_\phi(\mathrm{y}) = \frac{1}{Z} \prod_{i=1}^{N} \phi_i(y_i) \prod_{(ij)\in\mathcal{E}} \phi_{ij}(y_i, y_j) \quad (5)$$

where Z denotes a normalization constant, given by $Z = \sum_{y'} \prod_{i=1}^{N} \phi_i(y'_i) \prod_{(ij)\in\mathcal{E}} \phi_{ij}(y'_i, y'_j)$.

A specific assignment of values to Y is denoted by y and represented by the set $\{y_i^k\}$ of $K \cdot N$ indicator variables, for which $y_i^k = I(y_i = k)$. In order to foster the associativity of the model, we reward instantiations that have neighboring nodes, which are labeled by the same class. This is enforced by requiring $\phi_{ij}(k,l) = \lambda_{ij}^k$, where $\lambda_{ij}^k > 1$, for all $k = l$, and $\phi_{ij}(k,l) = 1$, otherwise [12]. Inference is carried out by solving the *maximum a-posterior (MAP)* inference problem, i.e. to find $\arg\max_{\mathrm{y}} P_\phi(\mathrm{y})$.

For our specific problem, we define the node potentials $\phi(y_i)$ by a vector of features, whereas each feature consists of fuzzy rules as defined in Section III-A. Likewise we define the edge potentials $\phi(y_i, y_j)$ by a vector of features that reflect the spacial relations between surface cells. For this purpose we consider the height difference $\delta h = |h_i - h_j|$ between neighboring cells as a parameter for defining features expressing relations between them, as for example:

- Flat Transition $= SDown(\delta h, 15, 40)$
- Obstacle Transition $= SUp(\delta h, 25, 80) \cdot SDown(\delta h, 200, 350)$
- Wall Difference $= SUp(\delta h, 200, 300)$

Given the node and edge potentials, for each extracted region of interest one MRF graph is dynamically constructed, whereas each node connects to the four closest neighbors in its vicinity.

For the sake of simplicity, we represent potentials by a log-linear combination $\log \phi_i(k) = \mathrm{w}_n^k \cdot \mathrm{x}_i$ and $\log \phi_{ij}(k,k) = \mathrm{w}_e^k \cdot \mathrm{x}_{ij}$, where x_i denotes the node feature vector, x_{ij} the edge feature vector, and w_n^k and w_e^k the

row vectors according to the dimension of node features and edge features, respectively. Consequently, we can denote the MAP inference problem $\arg\max_{\mathrm{y}} P_\phi(\mathrm{y})$ by:

$$\arg\max_{\mathrm{y}} \sum_{i=1}^{N}\sum_{k=1}^{K}(\mathrm{w}_n^k \cdot \mathrm{x}_i) y_i^k + \sum_{(ij)\in\mathcal{E}}\sum_{k=1}^{K}(\mathrm{w}_e^k \cdot \mathrm{x}_{ij}) y_i^k y_j^k. \quad (6)$$

Equation 6 can be solved as a linear optimization problem by replacing the quadratic term $y_i^k y_j^k$ with the variable y_{ij}^k and adding the linear constraints $y_{ij}^k \leq y_i^k$ and $y_{ij}^k \leq y_j^k$. Hence, the linear programming formulation of the inference problem can be written as:

$$\begin{aligned} \max \sum_{i=1}^{N}\sum_{k=1}^{K}(\mathrm{w}_n^k \cdot \mathrm{x}_i) y_i^k + \sum_{(ij)\in\mathcal{E}}\sum_{k=1}^{K}(\mathrm{w}_e^k \cdot \mathrm{x}_{ij}) y_{ij}^k \quad (7) \\ \text{s.t. } y_i^k \geq 0, \quad \forall i,k; \qquad \sum_k y_i^k = 1, \quad \forall i; \\ y_{ij}^k \leq y_i^k, \quad y_{ij}^k \leq y_j^k, \quad \forall ij \in \mathcal{E}, k, \end{aligned}$$

which can, for example, be solved by the *Simplex* method. For a more detailed description, we refer to the work of Anguelov and colleagues [2]. Furthermore, it is necessary to learn the weight vectors for the node and edge potential from data, which we carried out by utilizing the *maximum margin* approach recommended by Taskar and colleagues [12].

C. Encoding skill preconditions

Due to the fact that obstacles might not be traversable from every point and every direction as flat areas are, we need to detect transitions that lead onto the obstacle. It is also necessary to determine preconditions, namely starting point and starting direction, that enable a skill to successfully overcome the transition.

For each skill the user has to define, what kind of transition the skill can traverse. This is done using qualitative relations that are formally given by a subset of the relations provided by *Allen's Interval Calculus* [1]. For example, to describe a ramp, one would create the following set of rules: $\{F$ meets R, F starts T, R ends $T\}$. Figure 4(b) shows a floor-ramp transition that does not contain any other intervals besides floor (F) and ramp (R), while T stands for the interval containing all intervals considered.

The clusters generated in Section III-A contain cells belonging to a specific class. To extract preconditions from these cells, the following steps are performed: First, cells that are potentially part of a transition are selected. Second, straight lines from the selected cells are extracted, whereas starting and ending cells of these lines are determined according to the qualitative relations describe above. In the following, these lines are termed *transition edges*. Third, preconditions, i.e. the exact starting locations for the skill execution, are computed.

For the extraction of suitable cells we select every cell that fulfills one of the *meets* rules defined by the qualitative relations. By this, cells that form a frontier from one classified region to another are selected, analogous to the definition of frontier cells [14] that define the frontier between unexplored and obstacle-free terrain within occupancy grids.

We assume that the transitions can be considered to be straight lines and apply the Hough transform [6] on the selected points to detect these lines. The Hough transform represents a line in normal form (see Equation 8), so that each point (x, y) lies on the straight that has an angle θ to the x-axis and distance ρ to the origin.

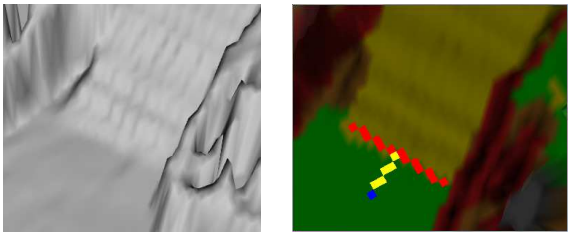
$$x \cdot \cos\theta + y \cdot \sin\theta = \rho \quad (8)$$

The transformation uses an accumulator in (θ, ρ) space to count for each point (x, y) each corresponding point in (θ, ρ). Consequently, the position (θ^*, ρ^*) with the highest count represents the line that goes through most points and thus is returned as the best possible line through the point set.

In order to check whether straight lines are transition edges, each cell of the line has to be considered. This is accomplished by examining the perpendicular line of each cell and validating if the qualitative relations, defined above, hold along the class intervals on this line. If valid cells are found over an extent of at least the robot's width, a transition edge has been detected.

Finally, the preconditions (x_s, y_s, ϕ_s) for the skill execution are calculated by projecting a starting point (x_s, y_s) perpendicular from the center of the transition edge with a fixed offset in front of the obstacle. The starting direction ϕ is set perpendicular to the transition edge pointing towards the obstacle. Additionally, the signed height difference h_s between the transition edge and the facing end of the obstacle, i.e. the height difference the robot has to overcome, is determined as a parameter for the skill execution. This parameter indicates, for example, whether a transition goes upwards or downwards, and thus influences the execution of the skill.

Each map cell surrounding the transition edge will be associated with this transition edge, its preconditions and skill function, thus forming the *behavior map*.



(a) (b)

Fig. 4. Visualization of the process of determining skill preconditions: (a) Elevation map with ramp structure. (b) Classified elevation map with the detected classes *floor* (green), *ramp* (yellow), and *walls* (dark red), detected transition edge (red), and detected preconditions, which are the starting point (blue) and staring direction (yellow).

For an example consider the analysis of a ramp in Figure 4. Figure 4(a) shows a ramp structure

on the elevation map. Figure 4(b) shows the classified elevation map with the detected classes *floor* (green), *ramp* (yellow), and *walls* (dark red). Constraints have been tested for each cell along a straight line through the region, resulting in the transition edge, marked in red. Note that along the transition edge the $\{F$ meets R, F starts T, R ends $T\}$ rules are always valid. The detected preconditions are the starting point (x_s, y_s), marked as blue, and the starting direction ϕ_s pointing upwards the ramp, marked as yellow.

IV. PLANNING AND SKILL EXECUTION

By utilizing behavior maps for planning and skill execution we can employ two dimensional A* search for path planning. This is due to the fact that the behavior map itself contains the information of how to overcome an obstacle and thus there is no need to explicitly plan complex robot skills in three dimensions. To facilitate A* planning, an occupancy map is built based on the behavior map by setting the occupancy values of traversable cells to 0.0, respectively to 1.0 for non-traversable cells. Note that traversability from *floor* cells to non-floor cells is just given if there is a *transition edge* at this position. The occupancy map is then blurred with a Gaussian kernel guiding the planning away from non-traversable terrain.

Applying the A* search on the map might lead to plans that traverse obstacles. Therefore, we need to take care of the fact that the costs for negotiating obstacles usually differ from those given solely by an occupancy map. To represent the higher amount of time spent during the skill execution on obstacles, and the increased risk of failure, a cost factor that leads to higher costs and thus to better suited planning results has to be provided by each skill.

The heuristic used for guiding the A* search is the Euclidean distance $h = \sqrt{\delta x^2 + \delta y^2}$, which is commonly employed. It is obvious that due to the increasing costs when traversing obstacles this heuristic is far from being optimal, but we refrain from changing the heuristic to guarantee the optimality of planning. Practical experiments have shown that time needed for A* planning constitutes no problem to the overall system.

The target selection process uses the concept of frontier cells [14], where a frontier cell is defined as being traversable and having, at least, one uninitialized neighbor. This ensures that the robot will explore unknown areas. We calculate the utility (Equation 9) based on the robots angle to the cell $\delta\theta$ and choose the frontier cell with maximum utility, which prevents frequent turning, i.e. oscillation.

$$u = (SDown(\delta\theta, 0, 180))^2 \tag{9}$$

We want to prevent the robot from choosing plans that lead over obstacles, if there are still targets reachable by ground exploration. Therefore, the planner accepts costs for ground exploration until a certain threshold, before permitting planning over obstacles.

When executing the actual plan, the skill subroutine that has to be executed by the controller is determined from the robot's current position on the behavior map. A *transition edge* stores the precondition that has to be fulfilled first before executing the skill. Whenever the controller is queried to execute a new skill routine, it first ensures this precondition by, for example, turning towards the required starting direction before calling the skill itself.

The skills (besides *Ground Exploration*) implemented in our system use as sensory input the robot's pitch and roll from the IMU, the angles of the flippers, and touch point sensors in the flippers that can give feedback about where the flipper has contact with an obstacle. Based on features created from these sensors, a state machine is executed that can run different actions in parallel (e.g. driving forward while moving the rear flippers down). Currently our robots have the ability to lift up and drive down from a *pallet*-like obstacle, drive a *ramp* and climb up *stairs*.

V. EXPERIMENTAL RESULTS

In order to evaluate the planning framework, a fully autonomous exploration run has been conducted. Results from this run can be seen in Figure 5. The test run

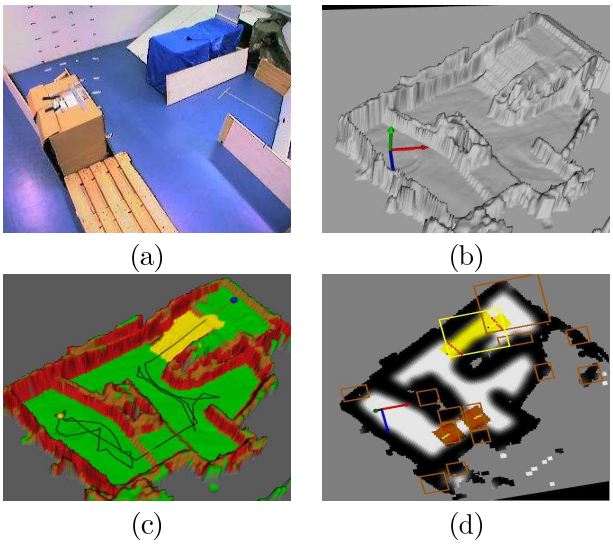


Fig. 5. (a) Test arena for the exploration task. (b) The created elevation map. (c) Classified map containing the trajectory of the exploration. (d) The behavior map showing the detected clusters of interest for obstacles (brown rectangles) and ramp (yellow rectangles) and recognized obstacle transitions (red). The robot started exploring the area at the left, crossed the pallet, continued to explore the center area and finally drove up the ramp.

demonstrates the system's ability of online planning and obstacle negotiation during an exploration task on rough terrain. Figure 5(d) shows the exploration trajectory, covering the complete area, whereas each obstacle has been crossed only once. The robot decided to explore each segment of the map before climbing an obstacle to reach another area. During this exploration run the robot traversed a trajectory with a length of 23.00m

within 9 minutes and 55 seconds. Main factors to the overall run time on a map of 60,000 cells were: the hough transform ($63.94ms \pm 0.78ms$), the occupancy map blur ($37.42ms \pm 0.98ms$) and the A* planning ($32.26ms \pm 4.76ms$) on a AMD Athlon X2 3800+. Additionally, the MRF algorithm took $26.31s \pm 27.88s$ to classify a region. This however did not limit the system's real-time ability since the robot can rely on the pre-classification for planning and continue the exploration while MRF classification is computed in the background.

A basis for climbing and obstacle negotiation based on behavior maps is the robot's capability to execute different kinds of skills. In Figure 6 we present a detailed view showing the robot traversing stairs autonomously. To achieve this task, the robot lifts itself up by pushing down with the front flippers after aligning to the stairs. Then, the robot drives forward while perceiving feedback from touch-point sensors within the flippers if contacting the stair. This enables the robot to become aware of its state and to adjust the flipper angles accordingly. Finally, when reaching the end of the stairs, the arms are moved back into driving position. A video

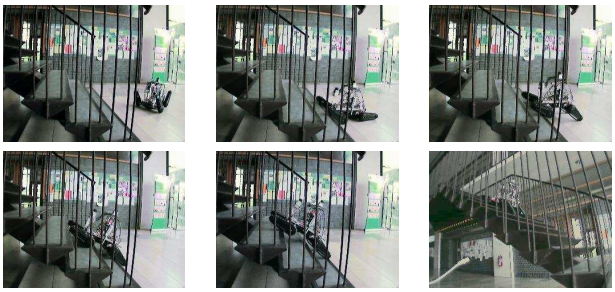


Fig. 6. Autonomous climbing of stairs.

showing the test run and the autonomous stair climbing is available at `www.informatik.uni-freiburg.de/~rescue/robots/video/behaviour_maps.mpg`.

VI. CONCLUSIONS AND FUTURE WORKS

In this paper we introduced an extension to obstacle negotiation that allows the robot to execute specific skills with respect to the context. The proposed planning framework is based on a new concept called behavior maps, which allows robots to deliberatively plan and execute complex skills on rough terrain. The advantage of the described framework, i.e. the abstract definition of skills, is that it can be extended by new skills, and even be adapted to other robot types.

Our results have shown that the final system implements a truly autonomous robot that builds an elevation map of the environment, classifies this map, and plans and performs sequences of skills, all in real-time and online during execution. This is a much harder problem than processing data offline, e.g. to classify structures from a pre-computed elevation map, since the incremental states of the elevation map only partially represent the environment due to the limited field of view of the robot.

In future work, we will extend the proposed approach towards a multi-level representation of behavior maps, allowing the robot, for example, to autonomously explore multistory buildings. Furthermore, we intend to apply learning methods within skills by exploiting the context information given by the behavior map.

VII. ACKNOWLEDGMENTS

The authors gratefully acknowledge the work done by all members of the *RescueRobots Freiburg team*, particularly Rainer Kümmerle and Bastian Steder.

REFERENCES

[1] J. F. Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843, 1983.

[2] D. Anguelov, B. Taskar, V. Chatalbashev, D. Koller, D. Gupta, G. Heitz, and A.Y. Ng. Discriminative learning of markov random fields for segmentation of 3D range data. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, San Diego, California, June 2005.

[3] J. Bares, M. Hebert, T. Kanade, E. Krotkov, T. Mitchell, R. Simmons, and W. R. L. Whittaker. Ambler: An autonomous rover for planetary exploration. 22(6):18–22, 1989.

[4] J. Bruce, T. Balch, and M. Veloso. Fast and inexpensive color image segmentation for interactive robots. In *Proceedings of IROS-2000*, Japan, October 2000.

[5] M. Hebert, R. MacLachlan, and P. Chang. Experiments with driving modes for urban robots. In *Proceedings of SPIE*, 1999.

[6] P.V.C. Hough. Methods and means for recognizing complex patterns. In *U.S. Patent 069654*, 1962.

[7] A. Jacoff, E. Messina, and J. Evans. Experiences in deploying test arenas for autonomous mobile robots. In *Proc. of the PerMIS Workshop*, Mexico, 2001.

[8] A. Kleiner and C. Dornhege. Real-time localization and elevation mapping within urban search and rescue scenarios. Technical Report 231, University of Freiburg, 2007.

[9] R. Manduchi, A. Castano, A. Talukder, and Larry Matthies. Obstacle detection and terrain classification for autonomous off-road navigation. *Auton. Robots*, 18(1):81–102, 2005.

[10] L. Ojeda, J. Borenstein, and G. Witus. Terrain trafficability characterization with a mobile robot. In G. R. Gerhart, C. M. Shoemaker, and D. W. Gage, editors, *Unmanned Ground Vehicle Technology VII*, volume 5804, pages 235–243, 2005.

[11] A. Talukder, R. Manduchi, R. Castano, K. Owens, L. Matthies, A. Castano, and R. Hogg. Autonomous terrain characterisation and modelling for dynamic control of unmanned vehicles. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 708–713, 2002.

[12] B. Taskar, V. Chatalbashev, and D. Koller. Learning associative markov networks. In *Proc. of the 21th ICML*, 2004.

[13] N. Vandapel, D. Huber, A. Kapuria, and M. Hebert. Natural terrain classification using 3-d ladar data. In *IEEE International Conference on Robotics and Automation*, volume 5, pages 5117 – 5122, April 2004.

[14] B. Yamauchi. A frontier-based approach for autonomous exploration. In *CIRA*, 1997.

[15] C. Ye and J. Borenstein. A new terrain mapping method for mobile robot obstacle negotiation. In *Proc. of the UGV Technology Conf. at the 2003 SPIE AeroSense Symposium*, pages 21–25, Orlando, USA, 2003.

[16] C. Ye and J. Borenstein. Obstacle avoidance for the segway robotic mobility platform. In *Proc. of the American Nuclear Society Int. Conf. on Robotics and Remote Systems for Hazardous Environments*, pages 107–114, Gainesville, USA, 2004.

[17] C. Ye and J. Borenstein. T-transformation: a new traversability analysis method for terrain navigation. In *Proc. of the SPIE Defense and Security Symposium*, Orlando, USA, 2004.