Exploration Transform: A stable exploring algorithm for robots in rescue environments

Stephan Wirth
University of Koblenz and Landau
Universitätsstr. 1
56070 Koblenz, Germany
stwirth@uni-koblenz.de

Johannes Pellenz
University of Koblenz and Landau
Universitätsstr. 1
56070 Koblenz, Germany
pellenz@uni-koblenz.de

Abstract — Autonomous robots in rescue environments have to fulfill several task at the same time: They have to localize themselves, build maps, detect victims and decide where to go next for further exploration. In this contribution we present an approach that provides a robust solution for the exploration task: Based on the knowledge of the environment that the robot has already acquired, the algorithm calculates a path to the next interesting "frontier". Our comprehensive approach takes into account the distance to the next frontier and the difficulty of the path for the robot. Those difficulties can result from narrow passages but also from wide, open spaces where the sensors cannot detect any landmark. For the native exploration task, the algorithm is fed with occupancy grids. For the search task, it can also process maps that encode additional information, e. g. places that have not been searched by other sensors yet.

The Exploration Transform was successfully implemented on our mobile system Robbie and was used during the RoboCup German Open 2007 and the RoboCup World Championship 2007. Using this approach, our Team "resko" achieved the "Best in Class Autonomy Award" in the Rescue Robot League in both competitions.

Keywords: *Robotic, RoboCup Rescue, autonomous navigation, path planning, exploration, USAR*

I. Introduction

Robots need to know about the structure of their environment to be able to fulfill complex tasks [7]. Anyway, in rescue situations resulting from earthquakes, this knowledge is not available a priori: Even if floor plans of a collapsed building are accessible, they are useless because it is likely that the furniture and even walls have been moved due to the impact. Therefore, the robots have to build their own maps while localizing themselves. This problem is widely known as the SLAM problem (Simultaneous Localization and Mapping) [3], [5]. Beyond that, autonomous robots must also decide where to go next to find out more about the environment (see [2]). This problem can be split into two questions:

1) Select a target: *Where should the robot go next to extend the map or search for victims?*
2) Choose a path: *Which way should the robot take to reach this target?*

The approach that we present in this paper uses an occupancy grid [1], that our mobile system Robbie builds automatically while driving through an unknown environment. The grid consists of cells which store the probability that the cell is occupied. To generate this map, the algorithm uses the robot's odometry data and the sensor readings of a Hokuyo URG-04LX laser range finder. The map building process uses a particle filter to match the current scan onto the occupancy grid. This approach was successfully tested during the RoboCup World Championship 2006, but relied on a remotely controlled robot. To make the system fully autonomous, the exploration behavior described in the following sections was implemented.

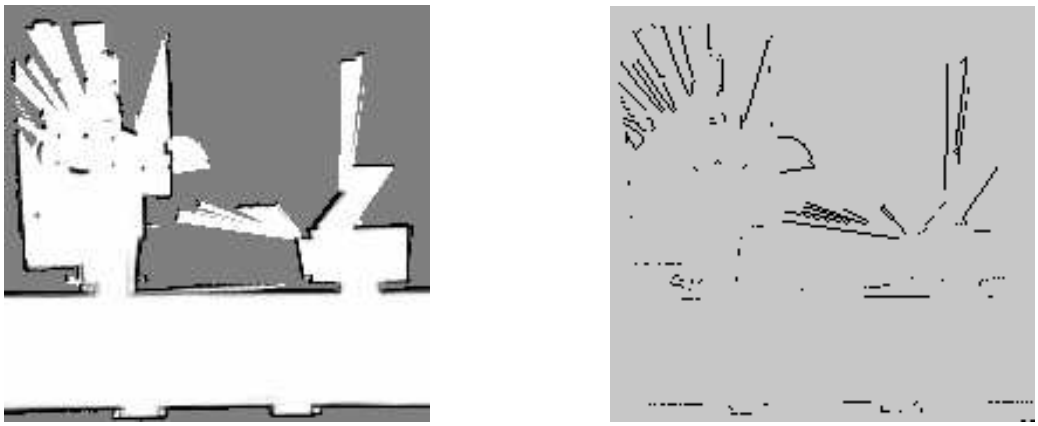
The paper is organized as follows: In section II, the related work is presented. The different approaches for exploration and path planning are explained. In section III the Exploration Transform is developed as a combination of the solution discussed in section II. It is extended to encourage the robot to stay into the sight of walls, so the sensors can always detect some landmarks. This behavior is known as "coastal navigation" [6]. The complete algorithm was implemented on our rescue robot Robbie. Section IV describes the results of the experiments we conducted in different test arenas, including the rescue arenas of the RoboCup German Open 2007 and the RoboCup World Championship 2007.

II. Related work

A. Exploration

In [8], Brian Yamauchi proposes a frontier-based approach to find the next exploration target. Occupancy grids are used as an input for the algorithm. The key idea is as follows: In order to get new information, go to a frontier that separates known from unknown regions. Such a frontier is a cell in the occupancy grid that is marked as *free* but has a neighboring cell that is marked *unknown*. A segment of adjacent frontier cells is considered as a potential target if it is large enough so that the robot could pass it. If more than one potential target is detected in the occupancy grid, then the closest target is selected. An example of a floor plan and the extracted frontiers is given in Fig. 1.

González-Baños and Latombe propose another approach to select the next target point: In [2] they present an algorithm that finds the point in the map from which the sensors can be used optimal to extend the map ("next-best-view problem"). To find this point, candidate target points around a frontier are generated. For each candidate point, the size of the area



(a) Metric map (b) Extracted frontiers

Fig. 1. A metric map (stored as an occupancy grid) and the extracted frontiers of the map. The frontiers form lines that separate free from unknown areas.

is calculated which could be newly measured from this point. The point that yields a view to the largest area is chosen.

B. Distance Transform

To find a way from an arbitrary starting point to a fixed target Javis and Byrne [4] propose the distance transform. The distance transform of an occupancy grid calculates for each free cell the cost to reach the target cell. The cost between two cells c_i and c_j (with no obstacle between them) can be the city block distance, the chessboard distance or the euclidian distance. After the distance transform has been applied for each cell of the grid, the shortest path from any cell to the target cell can be extracted by simply following the steepest gradient.

Fig. 2(a) shows the distance transform using the chessboard distance.

C. Obstacle transform and path transform

The distance transform always chooses the shortest way between two cells. This way is not always suitable for robots, because it often touches walls and goes through narrow passages. Alexander Zelinsky's path transform ([9], [10]) adds a security component to the distance transform: An *obstacle transform* Ω calculates for each cell the distance to the closest obstacle. Fig. 2(b) shows the obstacle transform for a simple floor plan.

The *path transform* Φ of a cell c to reach the target cell c_g is defined as follows:

$$\Phi(c, c_\mathrm{g}) = \min_{C \in \chi_c^{c_\mathrm{g}}} \left(l(C) + \alpha \sum_{c_i \in C} c_\mathrm{danger}(c_i) \right) \qquad (1)$$

with $\chi_c^{c_\mathrm{g}}$ the set of all possible paths from c to c_g, $l(C)$ the length of the path C, $c_\mathrm{danger}(c_i)$ the cost function for the "discomfort" of entering cell c_i, and α a weighting factor ≥ 0.

The length $l(C)$ of the path C can be calculated incrementally:

$$l(C) = l(c_0, \ldots, c_n) = \sum_{i=0}^{n-1} d(c_i, c_{i+1}) \qquad (2)$$

where d denotes the distance between two cells (e. g. the chessboard distance).

The "discomfort" cost c_danger of a cell is calculated based on the obstacle transform of this cell. For distances to the wall that are smaller than half the size of the robot the cost should be very high. Zelinsky's choice for such a cost function (see [11]) is given in (3).

$$c_\mathrm{danger}(c_i) = \begin{cases} (X - \Omega(c_i))^3, & \text{if} \quad \Omega(c_i) \leq X \\ 0, & \text{else} \end{cases} \qquad (3)$$

The constant X determines the minimum distance to obstacles and depends on the size of the robot and the accuracy of the sensors and the map. The weight α in (1) determines how far the path stays away from obstacles.

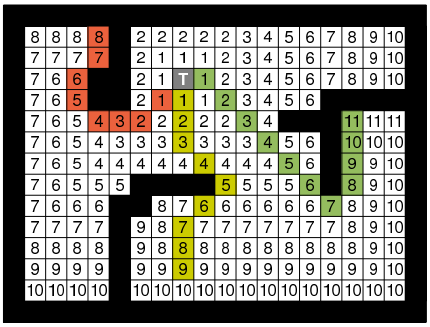
Again, to go to the target cell from any free cell, it is sufficient to follow the steepest gradient.

III. Exploration Transform

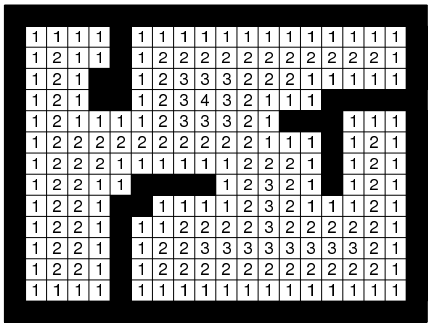
By combining the Yamauchi's frontier based exploration with Zelinsky's path transform an elegant solution for the exploration problem can be achieved: The path transform is extended in a way that not the cost of a path to a certain target cell is calculated, but the cost of a path that goes to a close frontier. The path is not necessarily the shortest and the frontier not necessarily the closest, since the cost is determined by the distance *and* the safety of the path. The overall formula of the Exploration Transform is given in (4).

$$\Psi(c) = \min_{c_\mathrm{g} \in F} \left(\min_{C \in \chi_c^{c_\mathrm{g}}} \left(l(C) + \alpha \sum_{c_i \in C} c_\mathrm{danger}(c_i) \right) \right) \qquad (4)$$

with F the set of all frontier cells, $\chi_c^{c_\mathrm{g}}$ the set of all paths from c to c_g, $l(C)$ the length of the path C, $c_\mathrm{danger}(c_i)$ the cost function for the "discomfort" of entering cell c_i, and α a weighting factor ≥ 0.



(a) Distance transform: The target cell is marked with a T. To find the shortest way from any cell to the target, the steepest gradient has to be followed.



(b) Obstacle transform: Each cell stores the distance to the closest obstacle.

Fig. 2. Distance transform and obstacle transform

The Exploration Transform has the favorable property that by construction no local minima can occur. Therefore, from each cell a path to a close frontier can directly be extracted.

Compared to the path transform, the Exploration Transform performs a search over all possible frontier cells. In the case that the set F contains only one element c, the Exploration Transform is identical to the path transform to the target cell c.

The computation of (4) for each free cell in the occupancy grid is done by a specialized flood-fill algorithm. First, all frontier cells are chosen as seeds and all adjacent cells are added to a queue of cells whose Exploration Transform value has to be computed. The Exploration Transform value of a cell is then computed as the minimum value of its eight neighbouring cells plus the movement cost from this neighbour to the current cell plus the weighted "discomfort cost" for the current cell. Whenever an Exploration Transform value of a cell changes, it's neighbours might change their values too so they are added to the end of the computation queue. The Exploration Transform converges to its final state when the queue runs out of elements. This way each free cell of the occupancy grid is touched at least once. Though, experiments showed that computation time for the Exploration Transform can not be seen as a drawback, for an occupancy map of 500 $\times$ 500 pixels, the computation takes less than 500 ms on a standard Pentium 1,3 GHz processor.

Fig. 3 shows an occupancy grid and two corresponding Exploration Transforms. Areas in Fig. 3(b) and 3(c) close to frontiers are dark, which shows a short distance to a frontier. Areas that are further away from any frontier are marked light, showing a higher cost. The weighting factor α of the Exploration Transform determines how much a safer path is preferred over a shorter one. Fig. 3(b) and 3(c) show different paths for different settings of α.

A. Coastal navigation

Equation 3 forces the robot to stay away from obstacles, no matter how far the robots gets away from them. This bears the risk that the sensors (with their limited range) cannot see any landmarks. This should be avoided, because the robot needs landmarks for localizing itself and for extending the map. Therefore, the cost function is extended in a way that (a) the robot is encouraged to stay away from obstacles at a distance of d_{opt} and (b) never gets closer than d_{min}. The new *discomfort cost* c_{danger} of a cell c with distance d to the next obstacle is calculated as follows:

$$c_{\text{danger}}(c) = \begin{cases} \infty, & \text{if } d < d_{\text{min}} \\ (d_{\text{opt}} - d)^2, & \text{else.} \end{cases} \tag{5}$$

Fig. 4(a) and 4(b) illustrate the discomfort costs of a sample map with different parameters.

B. Merging the occupancy grid with other sensor data

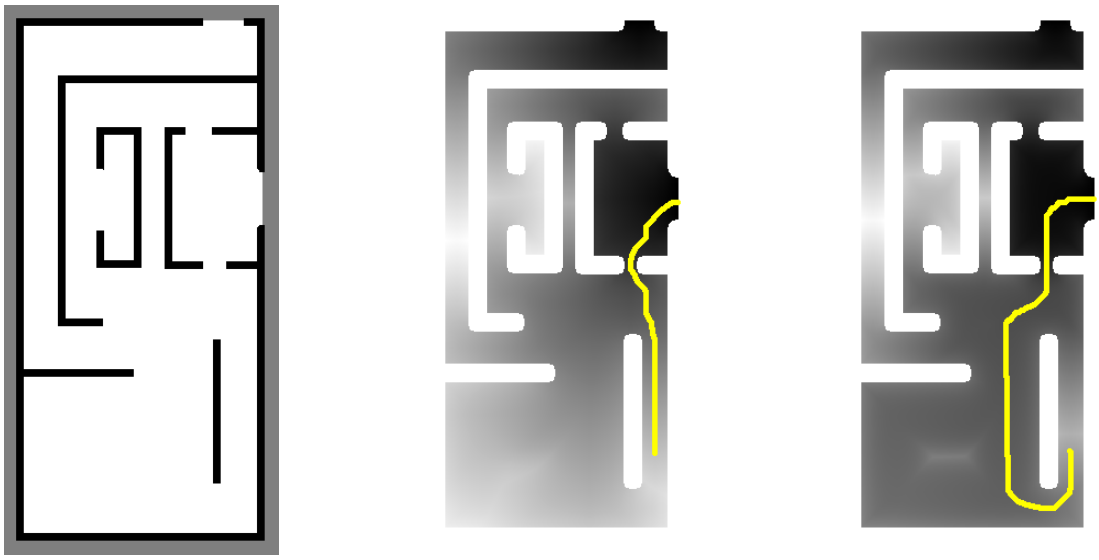
As described so far, the Exploration Transform works on the occupancy grid only. This is sufficient if the tasks of the robot are solely exploration and mapping of the environment. But if the robot has to fulfill another task at the same time, the exploration strategy has to be refined. In our case, while exploring, the robot searches for victims using a thermal sensor with a field of view of 180° and a range of only 2 meters (the laser range finder has a range of 4 meters). To keep track of the areas that the thermal sensor has scanned *and* that are mapped by the laser scanner, a second grid called the the *navigation grid*, is calculated as follows:

$$\text{navGrid}(c_i) = \begin{cases} \text{free}, & \text{if occGrid}(c_i) = \text{ free } \wedge \\ & \text{ScannedbyThermalSensor } (c_i) \\ \text{occupied}, & \text{if occGrid}(c_i) = \text{occupied} \\ \text{unknown}, & \text{else.} \end{cases} \tag{6}$$

Using the navigation grid for the Exploration Transform instead of the occupancy grid, paths to areas that yet were not scanned by the thermal sensor are computed and the robot is able to scan systematically the whole environment by following these paths.

C. Path optimization

The result of the path planning is a list of adjacent cells in the occupancy grid. Anyway, only a subset of these cells is required to describe a safe way from the starting point to

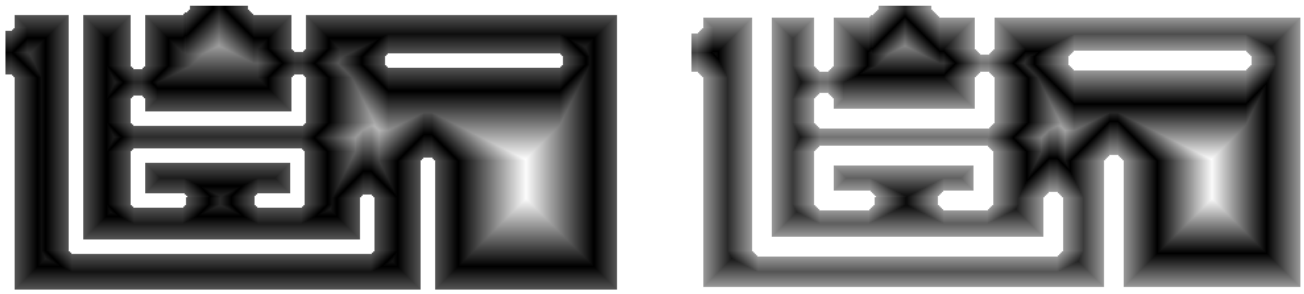


(a) Occupancy grid

(b) Exploration Transform with short path

(c) Exploration Transform with safe path

Fig. 3. Occupancy grid (a) and the corresponding results of the Exploration Transformation for different values of α (b) and (c). Dark cells in (b) and (c) indicate a short distance to the next safe frontier, white cells indicate a long way. In (b) a shorter and in (c) a safer path is chosen.



(a) $d_{\min} = 3$, $d_{\text{opt}} = 20$

(b) $d_{\min} = 5$, $d_{\text{opt}} = 30$

Fig. 4. "Discomfort" costs for different parameters. In (a) the robot would stay close to walls and also uses narrow passages, in (b) the robot would stay further away from walls and avoids to use narrow ways.

the frontier: In regions with nearby obstacles, more points are chosen than in regions with no obstacles. The result of the obstacle transform is used to determine the required distance between two waypoints of the path. Fig. 5 shows a path that consists of the original 212 cells and the reduced path of only 10 waypoints.

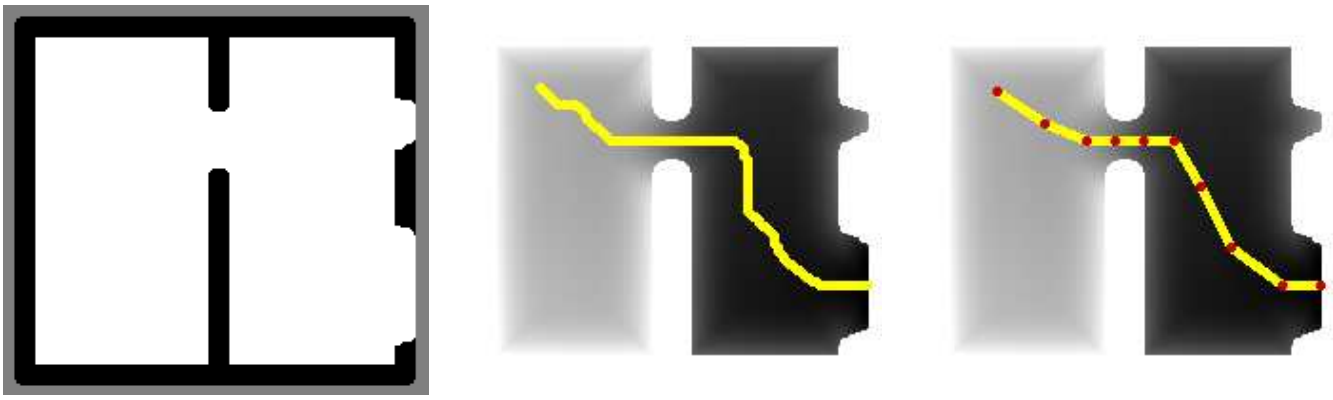
IV. Experiments

We tested the approach on our mobile system Robbie, which is able to generate a map online using a laser range finder with a range of 4 meters. The constantly growing map was used as the input of the Exploration Transform. The final maps (after the exploration) are shown in Fig. 6. In 6(a), the robot first follows the large (and therefore safe) frontier on the right side, and avoids to enter the door in the lower end. It keeps a distance of about 0.8 m from the walls, which was set as the optimal distance to obstacles. The other maps also show that the robot prefers larger passages. Overall, the robot comes up with a nosy, but secure behavior.

The algorithm relies on correct maps of the environment (which might be incomplete). Anyway, if a path is planned on a corrupted map, the robot stops in front of an obstacle, and updates the map. After the update process, the robot recalculates the path and continues driving.

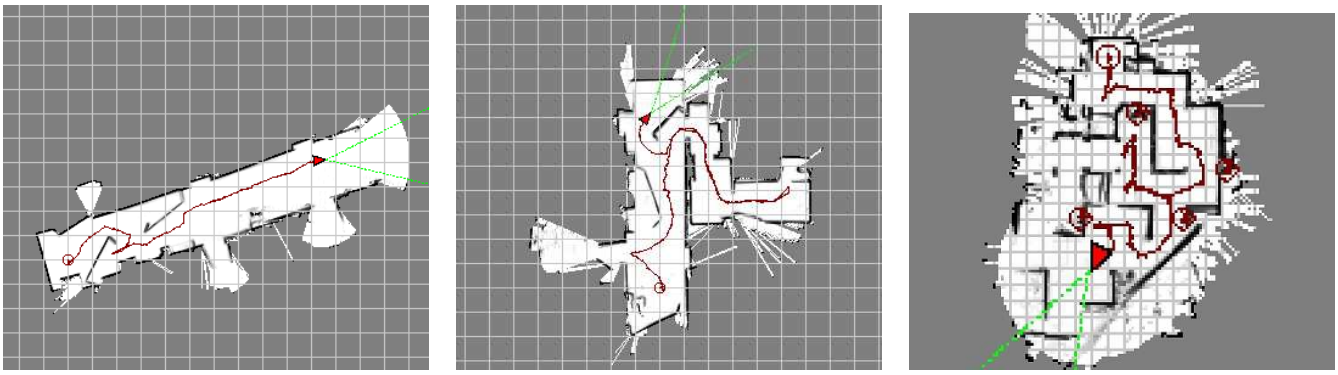
V. Conclusion

In this contribution we presented a novel approach for path planning of autonomous robots in rescue environments. The approach combines the frontier based exploration with the path transform and extends it with coastal navigation and the possibility to work on multi-sensor data. The algorithm is fast enough so that it can be used on our mobile system Robbie for online exploration. The algorithm has properties that make it very usefull for rescue robots:



(a) Floor plan (b) All adjacent free cells on the path (c) Optimized path

Fig. 5. Path optimization: (a) shows the original floor plan, (b) the extracted path from the top left to the lower right, consisting of 212 free cells. In (c), the path is reduced to only 10 waypoints.



(a) Hallway at the University of Koblenz and Landau (b) Hallway and offices at the University of Koblenz and Landau (c) Arena at the RoboCup German Open 2007

Fig. 6. Metric maps, generated during autonomous missions. The distance between the white lines is 1 m, the grid size of the occupancy grid was 0.05×0.05 m in (a) and (b), 0.1×0.1 m in (c). The line that ends with an arrow marks the path the robot followed during exploration. It shows that the robot looked "nosy" behind the barriers.

- The range of different sensors can be handled. In our case, the limited range of a thermal sensor (about 2 m, compared to 4 m of the laser range finder) limits the range of the exploration behavior.
- The algorithm works on noisy and incomplete maps.
- If a path exists from the current robot position to a frontier, the algorithm always returns a safe way with a minimal set of waypoints.

The approach was used by the Team "resko" of the University of Koblenz and Landau during the RoboCup German Open 2007 in Hannover (Germany) and during the RoboCup World Championship 2007 in Atlanta (GA, USA). The team achieved the "Best in Class Autonomy Award" in both competitions.

REFERENCES

[1] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 6 1989.

[2] Héctor H. Gonzáles-Baños and Jean-Claude Latombe. Navigation strategies for exploring indoor environments. *The International Journal of Robotics Research*, 2002.

[3] J. Gutmann and K. Konolige. Incremental mapping of large cyclic environments. In *Proceedingsof the IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, pages 318–325, Monterey, California, 1999.

[4] R. A. Jarvis and J. C. Byrne. Robot navigation: Touching, seeing an knowing. In *Proc. Australian Conf. on Artificial Intelligence*, Melbourne, Australia, 1986.

[5] J. M. M. Montiel, Jose A. Castellanos, J. Neira, and J.D. Tardós. The spmap: A probabilistic framework for simultaneous localization and map building. *IEEE Transactions on Robotics and Automation*, 15(5):948–952, 1999.

[6] Nicholas Roy, Wolfram Burgard, Dieter Fox, and Sebastian Thrun. Coastal navigation: Mobile robot navigation with uncertainty in dynamic environments. In *ICRA*, pages 35–40, 1999.

[7] Sebastian Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71, 1998.

[8] B. Yamauchi. A frontier-based approach for autonomous exploration. *1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, page 146 ff., 1997.

[9] Alexander Zelinsky. Robot navigation with learning. *Australian Computer Journal*, 20(2):85–93, 5 1988.

[10] Alexander Zelinsky. *Environment Exploration and Path Planning Algorithms for a Mobile Robot using Sonar*. PhD thesis, Wollongong University, Australia, 1991.

[11] Alexander Zelinsky. Using path transforms to guide the search for findpath in 2D. *I. J. Robotic Res*, 13(4):315–325, 1994.