ISR / ROBOTIK 2010
In Proceedings of Joint 41th International Symposium on Robotics and 6th German Conference on Robotics, Munich, June 2010.

Evaluating the Efficiency of Frontier-based Exploration Strategies

Dirk Holz[1], Nicola Basilico[2], Francesco Amigoni[2] and Sven Behnke[1]
[1] University of Bonn, Department of Computer Science VI, Bonn, Germany
[2] Politecnico di Milano, Dipartimento di Elettronica e Informazione, Milano, Italy

Abstract

Exploration and mapping are fundamental prerequisites for autonomous robots operating in initially unknown environments. In this paper, we evaluate simple yet efficient frontier-based exploration strategies. Furthermore, we discuss improvements to the classic frontier-based exploration strategy by Yamauchi *et al.* that further shorten the resulting exploration paths and present results from a comparative evaluation with a reference exploration strategy taken from the literature.

1 Introduction and Related Work

Exploring and mapping environments are fundamental prerequisites for autonomous mobile robots operating in initially unknown or dynamic environments. Exploration is related to well-known problems from the field of computational geometry, namely art gallery, illumination and shortest watchmen route problems. Since the original art gallery problem is NP-complete [1] and requires complete knowledge about the environment, exploring an unknown environment is usually performed in a reactive or greedy fashion. Instead of planning in advance all locations where the robot needs to acquire sensory information (or where guards need to be placed), a greedy exploration strategy solely plans one step ahead by determining a next best view that provides new information about the environment while minimizing some objective function. Over the last decades different exploration strategies have been proposed [2, 3, 4, 5]. However, there are only a few comparative evaluations of different strategies such as [6] and [7]. This paper focuses on the efficiency of frontier-based exploration strategies [8], presents a strategy incorporating a map segmentation algorithm for exploring the environment room-wise, and compares the resulting strategies with a state-of-the-art decision-theoretic exploration strategy from [9].

Frontier-based exploration strategies usually operate on grid maps [10]. In contrast to continuous geometric feature maps such as point or line maps, these maps distinguish between free and previously unexplored regions (white and gray pixels in **Figure 1**). The idea of frontier-based exploration strategies is to guide the robot to the frontiers or boundaries between cells known to be free (i.e., not occupied by obstacles) and cells for which no information is available (i.e., for which the robot has not yet acquired sensory information). Here, the problem of determining the next best view reduces to determining the frontier being closest (w.r.t. path length) to the robot. This is equivalent to say that the amount of information that is expected to be acquired is assumed to be the same for all frontiers.

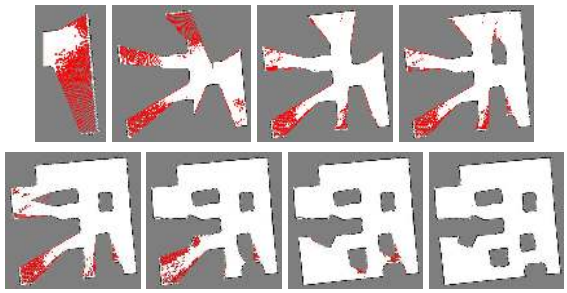


Figure 1: From left to right and top to bottom: frontier cells in the progress of exploration and incremental map construction (free space: white, unknown regions: gray, frontiers: red/dark gray).

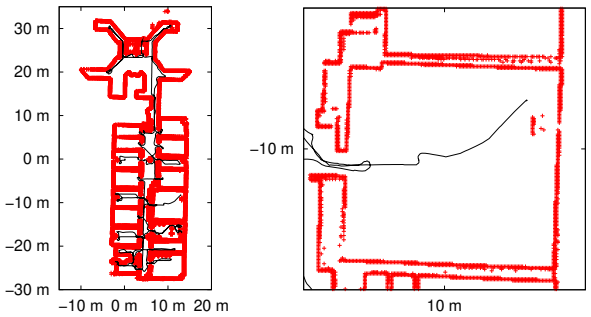
That the trajectories of a mobile robot exploring closest frontiers are kept reasonably short and that there are upper and lower bounds on the trajectory length have been shown by Koenig *et al.* [11]. In contrast, the number of poses that need to be reached in a closest-frontier exploration is usually larger compared to other strategies [5]. However, this is only problematic when not continuously acquiring information and updating the map. Here, we assume a continuous perception of the environment, using 2D laser range scans, and continuously updating the robot's environment model with an efficient on-line SLAM algorithm [12]. Still there are two drawbacks of closest-frontier exploration compared to other strategies:

(1) due to the continuous map updates, the currently approached unknown region might get fully explored during navigation before reaching the destination frontier, in which case the robot could start to explore the next unknown region, and

(2) the closest frontier can lie outside of the room that is

currently explored. This situation may cause the robot to explore the same room two times with unnecessary long trajectories (see **Figure 2**).

The remainder of this paper is organized as follows: Section 2 presents the principle of frontier-based exploration and the proposed extensions to address problems (1) and (2). Results from a comparative evaluation are summarized in Section 3. They show that the achievable results of frontier-based exploration do not rank behind those of more sophisticated strategies and that both extensions further shorten the overall length of the path taken by the robot.



(a) Trajectory (b) Detail view

Figure 2: Unnecessary long trajectories in unsegmented maps. Shown are the path of the robot taken for exploring an environment with multiple rooms (a) and a room that needs to be entered twice in order to explore remaining frontiers (b).

2 Strategy

The idea of frontier-based exploration strategies is to detect borders between already modeled regions of the environment and those regions where the robot has not yet acquired information about environmental structures. Being more precise, the robot searches for regions that are *traversable* in the so-far built map and adjacent to unmodeled regions and holes in the map. Closest frontier strategies, in particular, evaluate the length of the paths from the robot's current location to all determined frontiers and guide the robot to the one being closest.
The strategy of exploring closest frontiers, as introduced by Yamauchi *et al.* in [8], can be briefly summarized as shown in **Figure 3**.

2.1 SLAM and Map Representation

For actually building a map, we use a SLAM (Simultaneous Localization And Mapping) approach that is based on incrementally registering raw 2D laser range scans [12]. The map is represented as an unordered point cloud. Duplicate storage of measurements is avoided by adding to the map only points that provide new information, i.e., that have some minimum distance to points already being stored in the map. In addition, we maintain a probabilistic reflection grid map that counts, for each cell, the number $\#hits$ of range beams being reflected by an object in the corresponding region and the number $\#misses$ of range beams passing through the cell without getting reflected. The reflection probability is then

$$p(c^{[xy]}) = \frac{\#hits}{\#hits + \#misses}.$$

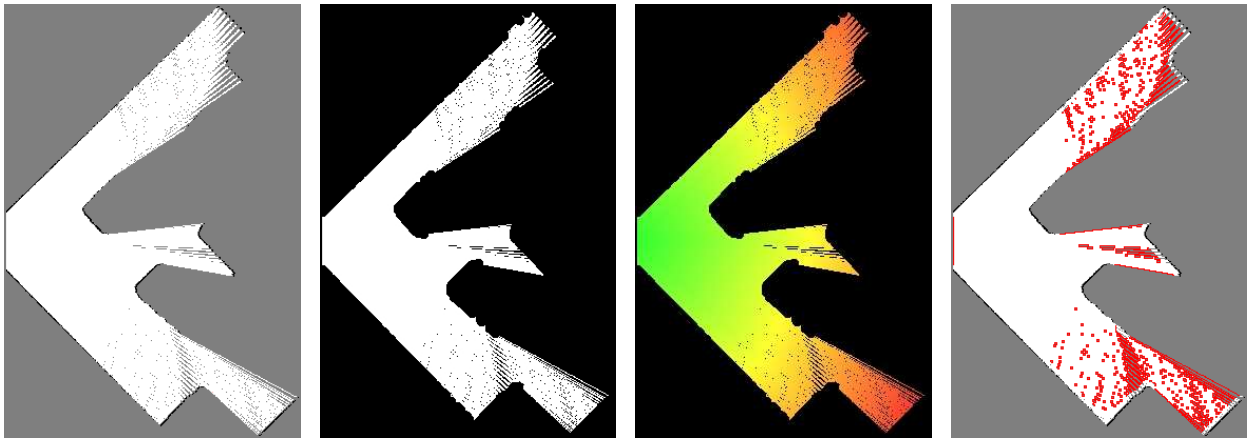
$p(c^{[xy]}) = 0.5$ is the prior probability that represents that the cell $c^{[xy]}$ is unknown.

1. Determine the set T of traversable cells, i.e., compute a traversability map.
2. Determine the set R of reachable cells, i.e., compute a reachability map by conducting a path search without goal specification.
3. Determine the candidate set C of cells that are reachable and traversable, i.e.,
$$C = \{c^{[xy]} \mid c^{[xy]} \in T \cap R\}. \quad (1)$$
4. Determine the set of frontier cells F by checking for every cell in the candidate set C if it is adjacent to a cell with unknown reflection probability:
$$\begin{aligned} F \quad = \quad & \{c^{[xy]} \mid c^{[xy]} \in C, \\ & \exists c^{[(x+m)(y+n)]} : p(c^{[(x+m)(y+n)]}) = 0.5, \\ & m \in [-1,1], n \in [-1,1]\} \end{aligned} \quad (2)$$
5. Determine the next best view $\mathbf{n} = (n^x \; n^y)^T$ as being the frontier cell lying closest to the robot's current position $\mathbf{r} = (r^x \; r^y)^T$:
$$\mathbf{n} = \underset{c^{[xy]} \in F}{\arg\min}\, L\left((x \; y)^T, \mathbf{r}\right), \quad (3)$$
where $L(\mathbf{p}, \mathbf{r})$ is the length of the shortest path from $\mathbf{p}$ to $\mathbf{r}$.

Figure 3: Strategy of exploring closest frontiers.

2.2 Determining and Evaluating Frontiers

For determining the set of candidate poses C and the frontier cell lying closest to the robot's current position we examine all cells in the probabilistic reflection map that are both traversable and reachable. In order to guarantee safe navigation and that the robot can approach the selected next best view position, we define traversable cells in a way that the robot should not traverse regions that are occupied or cells where no information is available. That is, candidate cells $c^{[xy]}$ have a probability of being reflective of $p(c^{[xy]}) \leq p_{\text{free}}$ (in our implementation



(a) Reflection map (b) Traversability map (c) Reachability map (d) Frontier cells

Figure 4: Determining frontiers in a probabilistic reflection map. Shown is a partial reflection map (a) together with the computed traversability and reachability maps (b+c) used for determining frontier cells (d). Frontier cells are marked red. The color coding in the reachability map corresponds to the length of the shortest path to that cell.

$p_{\text{free}} = 0.25$). Furthermore, we want the robot to keep a minimum distance to surrounding objects. We, therefore, neglect candidates whose distance to the closest obstacle region falls below some threshold d_{max} (in our implementation, $d_{\text{max}} = 30\,\text{cm}$).

We determine the set of actually reachable cells and evaluate the cost involved in traveling there by performing a path search from the robot's current location without specifying a goal pose. By this means, the internally used $A^\star$ path planner turns into Dijkstra's algorithm. Without the goal specification, it fully explores the reachable workspace and constructs a complete *reachability map* (see **Figure 4**). This map stores for every cell both the cost involved when traveling to it (i.e., the length of the shortest path) as well as the preceding cell along the shortest path. That is, once the set F of frontier cells has been determined, we can simply lookup the distances from $\mathbf{r}$ (the robot's current position) to all candidates $c^{[xy]} \in F$ and select the one being closest. The shortest obstacle-free path for navigating to the selected candidate can be recursively looked up in the reachability map.

As can be seen in **Figure 1**, always selecting the closest frontier as the next best view (NBV) already yields a reasonable exploration behavior. However, the following two extensions can further shorten the paths traveled by the robot.

2.3 Repetitive Re-Checking

During navigation to a target pose (i.e., the selected NBV), we continuously apply our SLAM approach from [12] in order to localize the robot and integrate newly acquired information into the so far built model. As a result, the robot might have fully explored an unknown region before actually reaching the corresponding frontier. Hence, continuing to travel to the selected NBV does not yield any new information.

We address this problem by repetitively re-checking whether or not the currently approached frontier cell is still adjacent to a cell with unknown reflectivity. As soon as the currently approached cell is no longer a valid frontier, the next closest frontier is selected using the exploration strategy. Since this check is a constant time lookup, i.e., in $O(1)$, it can be integrated into the original strategy without increasing the computational complexity.

Figure 5 shows a typical example of a trajectory obtained when using repetitive re-checking. It can be seen that the robot's trajectory is considerably shortened especially when approaching frontiers in the vicinity of corners.

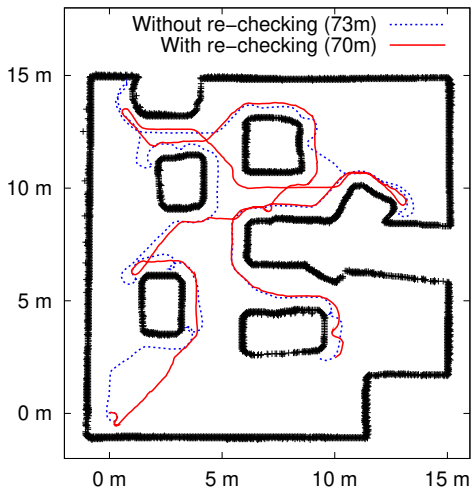
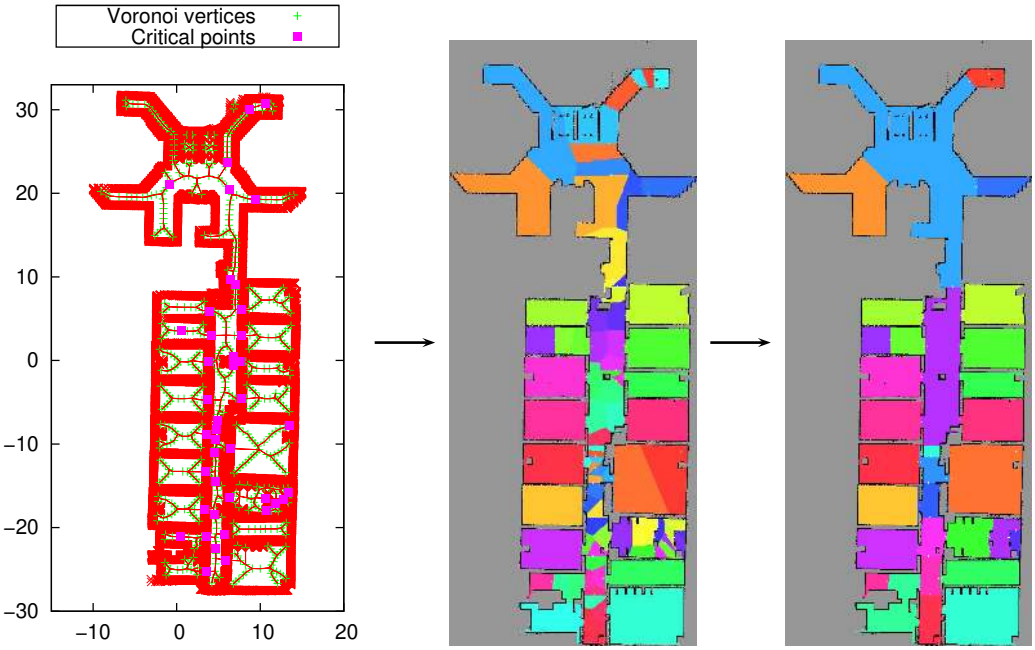


Figure 5: Achievable results of repetitive re-checking. With repetitive re-checking the length of the robot's trajectory is reduced from (approx.) $73\,\text{m}$ to $70\,\text{m}$.



(a) Critical points (b) Segmentation (c) Refined segmentation

Figure 6: Segmenting a map modeling the 5-th floor of the AVZ building at the University of Osnabrück. The constructed Voronoi diagram and the critical points are shown in (a). The resulting segmentation of free space into differently colored segments is shown in (b), its refinement in (c).

2.4 Map Segmentation

Another problem with the classical strategy is that a single room might get visited multiple times if successively selected NBVs lie in different rooms. In this case, the robot stops exploring a room although it is not yet fully modeled and enters the next room.

To reduce the number of multiple visits, we segment the partial map built so far into individual rooms and prefer frontier cells lying in the same segment as the robot's current location. For segmenting the grid map we use an approach based on the work by Thrun [13] that splits map regions at local minima (critical points) in the Voronoi diagram of the map's free space. Thrun used this segmentation algorithm for extracting the topology of an environment from grid maps. This algorithm, however, can produce a vast amount of segments, especially in longer corridors.

To address this issue, the algorithm was improved by Wurm *et al.* in [14] who used the segmentation to coordinate a team of multiple exploring autonomous robots. They restricted critical points to be of degree 2 and adjacent to a junction node in the Voronoi diagram. This modification considerably decreases the number of segments. However, real-world experiments conducted in the context of this paper have revealed that this is too restrictive. Especially at doorways the above constraint is often not met, in which case rooms are not well segmented. That is, two rooms form a single segment if the local minima in between is not directly adjacent to a junction node.

We define critical points to be local minima with respect to the distances to the closest Voronoi site [13, 14], nodes of degree 2, and to be itself adjacent to a junction node or adjacent to another node that is adjacent to a junction node. The latter modification relaxes the aforementioned constraint of [14] and better partitions the free space at doorways into separate segments.

The actual segmentation is conducted as follows. Using the critical points we split previously unassigned map regions into two parts. Instead of using the direct connection between the closest occupied cells at the critical points as in [13, 14], we assign cells to segments with respect to their distances to critical points. That is, we form clusters of cells being closest to a common split point. This can be achieved efficiently by computing an Euclidean distance transform (EDT) for the critical points [15]. For the actual assignment we compute and store both the distance to the closest critical point (as for the EDT) and the closest critical point itself; thus computing a nearest neighbor transform. A typical result of this initial assignment on maps of office-like environments is shown in **Figure 6(b)**.

As it can be seen, this initial assignment causes a vast amount of small segments, two for each split point. In order to obtain a clean segmentation, we construct a graph of splits from the initial assignments. In an iterative refinement step we then merge segments that are adjacent to each other but not split by the same critical point. A typical result of applying the overall segmentation algorithm is shown in **Figure 6(c)**. Except for a small number of

rooms, the shown map modeling the 5th floor of the AVZ building at the University of Osnabrück is well segmented. The overall segmentation algorithm is summarized in **Figure 7**.

In the exploration strategy it is then checked for every frontier cell if it belongs to the same segment as the one currently being explored by the robot. If no such cell exists, the robot selects the closest frontier $c^{[xy]}$ from the set F of all frontier cells. If, however, the set $F_{\mathbf{r}} \subseteq F$ of frontier cells in the robot's segment is not empty, the robot selects the closest frontier cell from $F_{\mathbf{r}}$. As shown in the following section, this straightforward extension can considerably decrease the length of the path traveled by the robot.

1. Construct the Voronoi diagram G, e.g., by means of Fortune's Sweep Line Algorithm [16] on extracted occupied cells or by means of the distance transform and skeletonization [14]:
$$G = (V, E).$$
2. Extract the set of critical points in the Voronoi diagram
$$C_c = \{v_c \mid v_c \in V\},$$
where v_c is a local minimum, has degree 2 and is adjacent to a junction node of degree 3 or a node of another degree that is adjacent to a junction node.
3. Create for the two neighboring nodes v_{n_1} and v_{n_2} to segments s_1 and s_2 and add them to the list of segments S.
4. Determine for all free cells c in the map the closest segment $s \in S$. Assign s to c.
5. Determine all transitions (s_1, s_2) in the segmented map and add them the adjacency list A.
6. Merge two segments s_1 and s_2 with $(s_1, s_2) \in A$ if s_1 and s_2 do not originate from the same split point or terminate if no such (s_1, s_2) exists.
Repeat step 6.

Figure 7: Map segmentation algorithm.

3 Results

This section will present our experimental setup as well as the results obtained from a comparative evaluation of different exploration strategies.

3.1 Experimental Setup and Robot Platform

In order to evaluate the efficiency of closest frontier exploration as well as of the proposed extensions, we have integrated different exploration strategies into our robot control architecture [12, 17] that can be simulated using both Player/Stage and Microsoft Robotics Developer Studio. The system consists of a differential-drive robot platform and a SICK LMS 200 laser range scanner. The scanner has a field of view of $180°$ and an angular resolution of $1°$. For following planned paths to target locations we use the non-linear motion controllers from [17]. Localization of the robot during navigation and map updates are conducted using the SLAM approach presented in [12].

In a comparative evaluation, we have applied the different exploration strategies in three different environments, a larger open-space and two office-like environments consisting of multiple rooms and corridors. The only criterion considered in the evaluation is the overall distance traveled by the robot in order to reach a full coverage of the environment. Regardless of the actually used exploration strategy, frontier cells are always determined. The non-existence of frontier-cells, i.e., there is not any single reachable cell being adjacent to an unknown cell, is used as a common termination criterion. Since we update the map continuously during navigation to a target pose, the total number of sensing locations is not considered in the evaluation.

3.2 Reference Strategy

As a reference strategy, we selected the decision-theoretic strategy of González-Baños and Latombe [9]. It draws sample poses $\mathbf{p}$ from the free space in the so far built map and evaluates their utility $u(\mathbf{p})$ in the context of exploration.

Two criteria are considered in the evaluation – the traveling cost $L(\mathbf{p})$ for reaching $\mathbf{p}$ and the expected information gain $I(\mathbf{p})$ when performing a sensing action at $\mathbf{p}$:

$$u(\mathbf{p}) = I(\mathbf{p})e^{-\lambda L(\mathbf{p})}.$$

As suggested in [9], we use $\lambda = 0.2$ to balance information gain and traveling cost. Whereas the traveling cost is estimated in the same way as for the frontier cells, the information gain is approximated as the expected relative change in map entropy. That is, we simulate range scans and corresponding map updates at all candidate poses $\mathbf{p}$. The information gain $I(\mathbf{p})$ is approximated as the difference between the map's entropy before ($H_{p(x)}$) and after ($\hat{H}_{p(x)}$) the simulated update:

$$I(\mathbf{p}) = \hat{H}_{p(x)} - H_{p(x)}.$$

Since the used probabilistic reflection maps, in principle, represent two probabilities (being occupied *and* being

free), we estimate the map entropy by means of

$$H = -\sum_{c^{[xy]}} \Bigg[\underbrace{p(c^{[xy]}) \log p(c^{[xy]})}_{\widehat{=} H_{p(occupied)}} + \underbrace{(1 - p(c^{[xy]})) \log(1 - p(c^{[xy]}))}_{\widehat{=} H_{p(free)}} \Bigg] .$$

The samples itself are generated to lie in the vicinity of frontiers as in [9]. As a baseline for comparison, we report also results obtained with a Random Frontier (RF) selection strategy, that chooses the next observation location according to a uniform probability distribution over the current set of frontier cells.

3.3 Exploring Open Spaces

Figure 8(a) shows the results of simulating an exploring robot in a larger open space. This Player/Stage example environment consists of a single polygon with holes and spans a region of approximately $256\,\mathrm{m}^2$. Plotted are both the mean length and standard variation of the path traveled by the robot and measured over 10 runs (see also **Table 1**).

Although the samples for the reference strategy [9] are sampled only in the vicinity of frontiers, increasing the number n of samples decreases the overall path length. In all three environments, $n > 500$ samples did not show considerable improvements.

It can also be seen that the traveled paths when exploring closest frontiers are considerably shorter than those resulting from applying the reference strategy. Repetitive rechecking further improves the result of closest frontier exploration as it effectively hinders the robot from traveling into already explored map regions. However, segmenting the so far built map and preferring frontiers in the currently explored segment does not further improve the achievable results in this scenario. This is mainly caused by the fact that the environment does not contain any room causing the segmentation to be rather random than reasonable. A considerable improvement can only be achieved in office-like or domestic environments that are composed of multiple rooms.

Strategy	Path length (m)
Random Frontier	102.84 ± 34.50
Closest Frontier	73.71 ± 1.00
Closest Frontier + RR	70.45 ± 0.97
Closest Frontier + RR + Seg.	72.89 ± 5.77
Decision Theoretic (n=100)	100.01 ± 17.64
Decision Theoretic (n=500)	98.27 ± 11.76

Table 1: Measured path lengths for exploring the large open space. See **Figure 8(a)**.

3.4 Exploring Office-Like and Domestic Environments

Figure 8(b) and **Figure 8(c)** show the results of simulating an exploring robot in larger office-like environments. Both environments can be segmented into individual rooms. Typical segmentations are shown to the right of the measured path lengths. In these scenarios, segmenting the map and preferring frontier cells in the currently explored segment considerably decreases the lengths of the paths traveled by the robot.

Table 2 summarizes the measured path lengths for exploring the 5th floor of the AVZ building at the University of Osnabrück. Here, map segmentation only provides a minor improvement, which is primarily caused by the fact that, even without map segmentation, the robot only rarely visits the same room twice.

Strategy	Path length (m)
Random Frontier	601.51 ± 141.93
Closest Frontier	382.07 ± 3.99
Closest Frontier + RR	361.88 ± 3.01
Closest Frontier + RR + Seg.	359.77 ± 12.91
Decision Theoretic (n=500)	447.68 ± 32.78

Table 2: Measured path lengths for exploring the AVZ building. See **Figure 8(b)**.

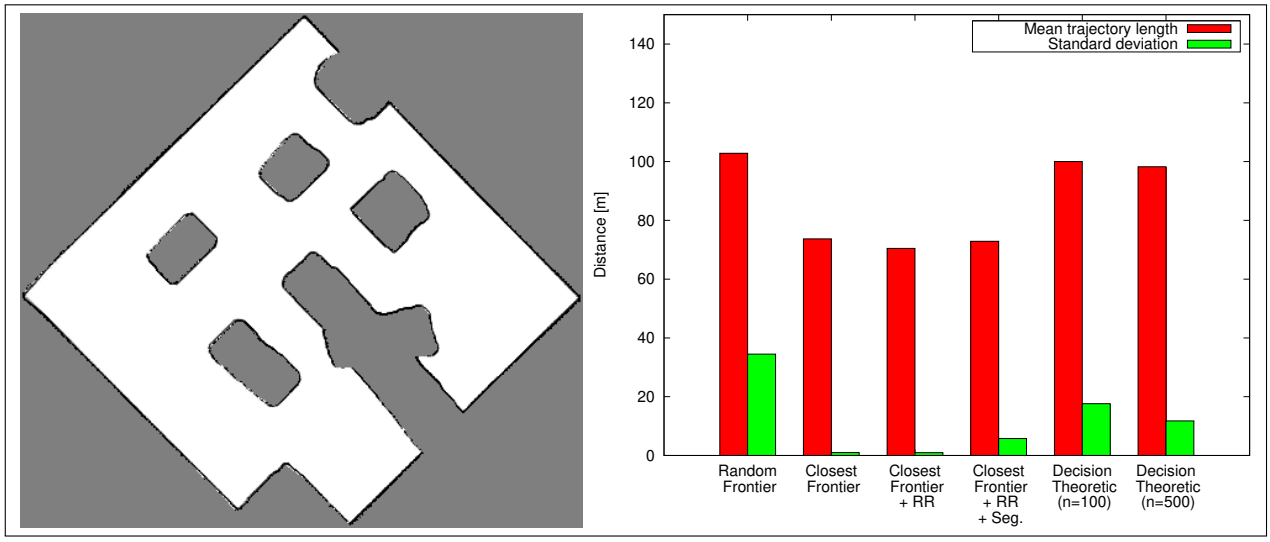
Table 3 summarizes the results for the "Hospital" environment that contains considerably more individual rooms. Here, map segmentation considerably shortens the path travelled by the robot.

Strategy	Path length (m)
Random Frontier	669.17 ± 186.41
Closest Frontier	281.07 ± 3.82
Closest Frontier + RR	253.09 ± 3.12
Closest Frontier + RR + Seg.	221.37 ± 19.97
Decision Theoretic (n=500)	312.68 ± 38.11

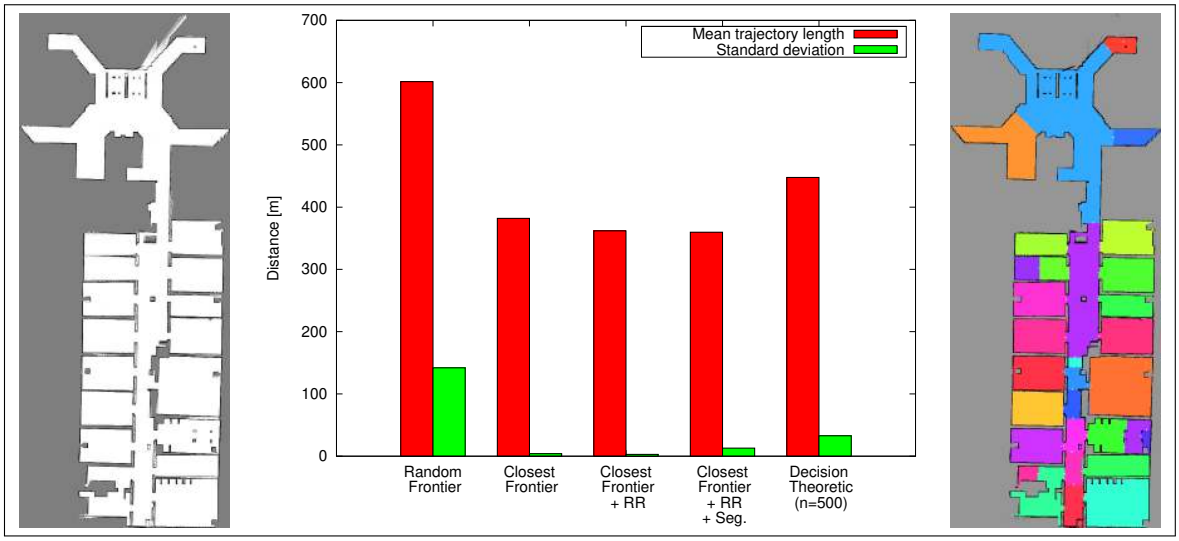
Table 3: Measured path lengths for exploring the Player/Stage environment "Hospital". See **Figure 8(c)**.

4 Conclusions and Future Work

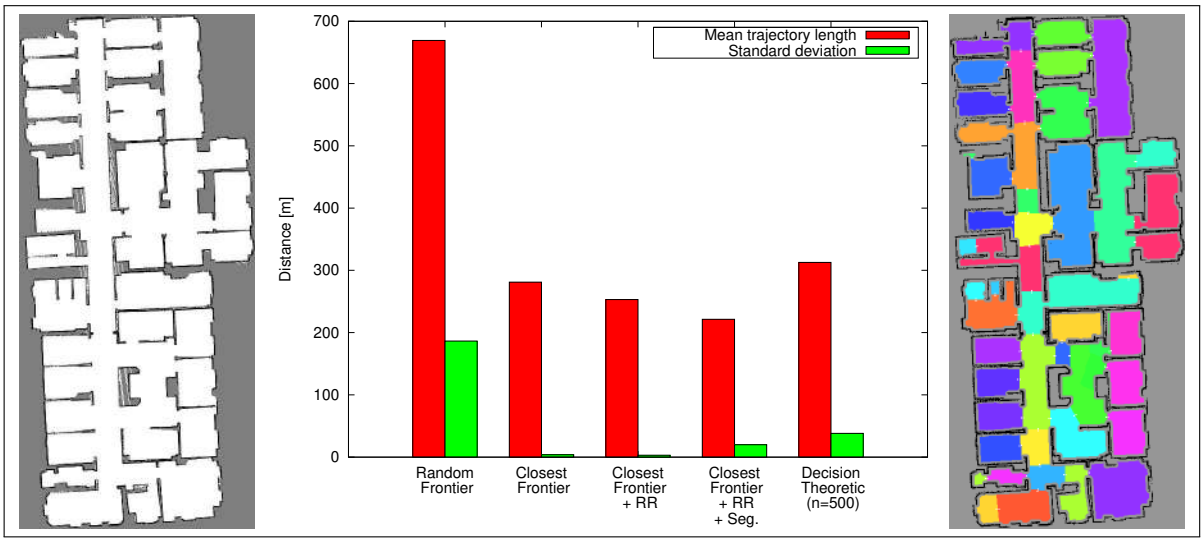
Although exploring closest frontiers is a rather simple and naïve strategy, the resulting paths are reasonably short and do not rank behind those acquired using more sophisticated strategies. Repetitive re-checking is an obvious and intuitive improvement to plain frontier-based exploration strategies. It considerably decreases the length of the traveled trajectories in all experiments conducted in the context of this paper. As it is computationally very efficient, it can easily be integrated into existing frontier-based strategies. It is, however, a matter of future work to adapt this concept to decision-theoretic strategies, e.g., by re-evaluating the utility of the currently approached candidate and, where



(a) Player/Stage "Cave" environment (provided by R. Vaughan)



(b) AVZ building at University of Osnabrück (provided by K. Lingemann, A. Nüchter and J. Hertzberg)



(c) Player/Stage "Hospital" environment (provided by R. Vaughan)

Figure 8: Evaluation results in simulated environments. Shown are measured mean values and standard deviations for the paths travelled by the robot in three different environments (used abbreviations: RR = repetitive re-checking, Seg. = map segmentation).

applicable, interrupt the travel.
In office-like and domestic environments, i.e., environments that are composed of several rooms, map segmentation and preferring frontier cells in the currently explored segment force a room-wise exploration. This further improves the achievable results. However, in wide open spaces, i.e., environments that cannot be reasonably segmented, map segmentation can even slightly increase the trajectory length. It is a matter of future work to decide, on the fly and depending on the currently explored environment, whether or not to use this extension.
Furthermore, we plan to make available a framework for benchmarking exploration strategies and to continue our efforts towards good experimental methodologies for comparing exploration strategies.

References

[1] A. Aggarwal, "The art gallery theorem: its variations, applications and algorithmic aspects," Ph. D. Thesis, John Hopkins University, 1984.

[2] F. Amigoni and A. Gallo, "A Multi-Objective Exploration Strategy for Mobile Robots," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2005, pp. 3861–3866.

[3] W. Burgard, M. Moors, C. Stachniss, and F. Schneider, "Coordinated Multi-Robot Exploration," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 376–386, 2005.

[4] R. Sim and N. Roy, "Global A-Optimal Robot Exploration in SLAM," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, April 2005, pp. 661–666.

[5] C. Stachniss and W. Burgard, "Exploring unknown environments with mobile robots using coverage maps," in *Proceedings of the International Conference on Artificial Intelligence (IJCAI)*, 2003, pp. 1127–1134.

[6] D. Lee and M. Recce, "Quantitative evaluation of the exploration strategies of a mobile robot," *International Journal of Robotics Research*, vol. 16, no. 4, pp. 413–447, 1997.

[7] F. Amigoni, "Experimental Evaluation of Some Exploration Strategies for Mobile Robots," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2008, pp. 2818–2823.

[8] B. Yamauchi, "A Frontier Based Approach for Autonomous Exploration," in *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, 1997, pp. 146–151.

[9] H. H. González-Baños and J. Latombe, "Navigation Strategies for Exploring Indoor Environments," *International Journal of Robotics Research*, vol. 21, pp. 829–848, 2002.

[10] H. Moravec and A. E. Elfes, "High Resolution Maps from Wide Angle Sonar," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1985, pp. 116–121.

[11] S. Koenig, C. Tovey, and W. Halliburton, "Greedy Mapping of Terrain," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2001, pp. 3594–3599.

[12] D. Holz and S. Behnke, "Sancta simplicitas – on the efficiency and achievable results of SLAM using ICP-Based Incremental Registration," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

[13] S. Thrun, "Learning Metric-Topological Maps for Indoor Mobile Robot Navigation," *Artificial Intelligence*, vol. 99, no. 1, pp. 21–71, 1998.

[14] K. M. Wurm, C. Stachniss, and W. Burgard, "Coordinated Multi-Robot Exploration using a Segmentation of the Environment," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2008, pp. 1160–1165.

[15] R. Fabbri, L. da F. Costa, J. C. Torelli, and O. M. Bruno, "2d euclidean distance transform algorithms: A comparative survey," *ACM Computing Surveys*, vol. 40, no. 1, pp. 2:1–2:44, 2008.

[16] S. J. Fortune, "Voronoi Diagrams and Delaunay Triangulations," E. Goodman and J. O'Rourke, Eds. CRC Press, New York, 1998, pp. 377–388.

[17] D. Holz, G. K. Kraetzschmar, and E. Rome, "Robust and Computationally Efficient Navigation in Domestic Environments," in *RoboCup 2009: Robot Soccer World Cup XIII*, ser. Lecture Notes in Computer Science, J. Baltes, M. Lagoudakis, T. Naruse, and S. Shiry, Eds. Germany: Springer, 2009, vol. 5949/2010, pp. 104–115.