Dynamic Frontier Based Exploration with a Mobile Indoor Robot.

Jens Wettach, Karsten Berns
Robotics Research Lab, Gottlieb-Daimler Straße, 67663 Kaiserslautern, Germany

Abstract

Any indoor service robot needs a kind of map as knowledge of its working space in order to perform navigation tasks efficiently. Ideally the robot should be able to generate this map autonomously when it is placed in a priori unknown environment. Thus it has to explore its surroundings completely in a reasonable amount of time and produce a map as accurate and complete as possible. This paper presents an integrated dynamic exploration approach that combines existing solutions of mapping, localization, obstacle avoidance and navigation to realize an efficient exploration strategy.

1 Introduction

Frontier based exploration is a well established method for automatic mapping of indoor environments. The key idea is that at any point of time during an exploration process the working space is divided into known, i.e. already mapped areas and unknown regions. And the most promising locations for gaining new information are the frontiers between these two spaces. One advantage of this technique is the straightforward decision where to move to perform the next environmental sensing. Besides it guarantees that in case of a 360° sensing operation there is enough overlap between the current global map and the extracted local map which improves relocalization and reduces map errors. Finally assuming the absence of dynamic obstacles the known regions can be passed without the risk of accidents. Therefore most of the existing approaches restrict navigation towards frontiers to the known space.

In the following an innovative integrated exploration system is presented that improves established implementations in several regards. First a SLAM component provides pose corrections during robot motions and thus enables a continuous map update during approaching a frontier position. Second a network of adaptive behaviors uses distance data from three sensor systems for dynamic obstacle avoidance in $\mathbb{R}^3$. Third an elastic band analyser permanently guarantees that the sensing position is still reachable via the chosen path, otherwise either a new path is calculated or an immediate sensing operation is initiated. This way the robot can safely leave the known area while approaching a frontier. Online map update and path validation make the exploration process dynamic: the mapping process is accelerated as new information is not only integrated at distinct frontier positions and navigation detours due to the known space restriction are avoided.

This paper is organized as follows: section 2 lists existing implementations of the frontier based exploration idiom and emphasizes the benefits of the presented approach. Section 3 describes the employed robot and sensor systems. Section 4 introduces the techniques for online 3D collision avoidance. Mapping and localization are discussed in section 5. Navigation and exploration are presented in section 6. The paper concludes with a validation of the developed integrated exploration system in simulated and real life experiments (section 7) and with an outlook on future work (section 8).

2 State of the Art

The main challenge of autonomous exploration is the following: given a partial knowledge of the world (incomplete map) where should the robot move next to increase this knowledge and improve the map. That is new information has to be integrated without destroying the existing information, e.g. due to localization errors. According to [1] an integrated exploration process consists of three sub-tasks: mapping, localization and motion control. This last task comprises deciding where to go next and then navigating towards this next best viewing (NBV) position. [2] describes the general NBV process as sequence of NBV selection – navigation – partial map acquisition – integration of the partial map into the global one. A well established method for determining the NBV is the frontier based exploration approach introduced by [3] and then picked up by several researchers [4], [5], [6], [7], among others. The basic idea is to choose the NBV at the frontier between known and unknown space as this maximizes the amount of newly gathered information at each exploration step. [3] uses a probabilistic occupancy grid map that is filled with omnidirectional distance data from sonar sensors and a 2D laser scanner at distinct frontier positions. The NBV is calculated at the nearest unvisited frontier and the shortest collision-free path is calculated by a standard path planner. Dynamic obstacles are cleared by reactive behaviors. If the goal position cannot be reached it is marked as inaccessible and an immediate sensing operation is performed. [4] extracts polygonal boundaries between known and unknown space from omnidirectional 2D laser scans at each NBV position. This way the already mapped safe region is consecutively expanded and collisions are avoided by restricting the robot motions to this area. The sensing positions are determined by weighting path length and expected information gain, i.e. unknown space that will be scanned which is estimated by ray tracing. Relocalization
is performed by aligning the partial map extracted from each scan with the global map. A similar approach has been reported by [6] which uses a line based map at distinct planes parallel to the ground to approximate a full 3D mapping system. [5] builds a topological graph with edges as sets of points that are equidistant to two or more obstacles and nodes as meeting points of these edges or end points, i.e. obstacles. The edges are online extracted from 2D distance data and the nodes are salient places, equidistant to at least three neighboring obstacles. Thus the working space is modeled in a generalized Voronoi graph and exploration is triggered by unexplored edges. Similarly [7] constructs a topological graph where each node is a place where a 360° monocular panorama image is taken and stored along with extracted SIFT features. The nodes are built at regular intervals during navigation and are connected by straight line paths as edges. Exploration is triggered by extracting the boundary between floor/ground obstacles and non-floor pixels in the camera image that mark the horizon towards navigable passages. The frontier with minimum orientation offset is used as NBV to minimize localization errors. Although using different types of sensors, feature extraction and mapping approaches these systems have two common properties:

- the NBV is taken at a frontier that is directly accessible through the already known space
- sensing operations, relocalization and map updates are only performed at these distinct sensing positions

In contrast to this the system presented in the following contains several improvements towards an dynamic integrated exploration process. A grid map based SLAM component guarantees continuous pose correction while the robot is moving. This enables limited online map updates along the path towards the NBV where an omnidirectional sensing operation is performed. As obstacles are recorded in $\mathbb{R}^3$ the path towards the NBV is continuously evaluated and adapted by an elastic band path planner. In combination with reactive obstacle avoidance behaviors this facilitates traversing unknown space towards the NBV and thus speeds up the exploration process.

### 3 Robot and Sensor Systems

The exploration system has been developed on MARVIN (see figure 1), a mobile indoor robot equipped with differential drive, two planar laser scanners for 2D obstacle avoidance, a belt of 20 ultrasonic sensors for detecting raised objects and a rotating laser scanner for mapping and detecting obstacles in 3D. Figure 2 shows the schematic sensor configuration in top (left) and side view (right). The space in front of and behind the robot is covered by a SICK S3000 and a LMS 200 mounted parallel to the floor 10 cm above ground. The provided distance data is evaluated with a 180° fov (red semicircles) which yields blind areas (yellow) at the left and right side of the robot. In order to avoid hitting obstacles in these areas any object that “leaves” the sensor area due to robot motions and thus enters the blind area is recorded in a linked list. As soon as any object in this list re-enters the opposite sensor area it is removed from the list. This way a short term obstacle memory is realized.

![Figure 1: Sensor systems for navigation and mapping on MARVIN](image)

The planar sensor field of the scanners does not cover the whole 3D space, e.g. they only detect table-legs but not tabletops. Therefore a belt of 10 ultrasonic sensors (small black boxes) with an opening angle of 60° has been mounted 48 cm above ground at the front and rear side of the robot respectively. Each of them is inclined 15° upwards so that there is an opening angle of 45° towards the ceiling. They are mounted at the outer frame of the robot relative to a virtual common center ($S_1, S_2$). The benefit compared to the planar scanners is the detection of protruding raised objects as tables and chairs at a continuous rate of 10 Hz. Due to the inherent sensor noise (false reflections, multipath readings) only the current readings are used. The danger of not using an obstacle memory is to approach e.g. tabletops too close so that the sensors detect the bottom instead of the front side. Therefore the obstacle avoidance system has to intervene before the robot falls into such a trap.

![Figure 2: Schematic configuration of sensors on MARVIN](image)

As a third sensor system a swiveling SICK S300 scanner provides accurate and dense 3D distance information. It is mounted 1.03 m above ground and continuously swiveled around a vertical axis between -65°...65° with 1 Hz. In order to avoid sensing parts of the robot only the angular range of -84°...107° of the swiveled 2D scan plane is used. The sensor data is classified as ground, obstacle and
ceiling data and stored in a local grid map as short term memory (see [11] for the original version of this approach). The benefit of this sensor is an exact detection of any 3D object. The main drawback is its low sensing speed of 1 s for 130° horizontal scan range and the restriction of sensing only the space in front of the robot. Thus objects that have been detected once and added to the grid map may be updated or removed from this memory only when the corresponding space is swept by the scanner again. Depending on the robot motion this can take a long time especially for objects behind the robot. Hence this memory is not as up-to-date as the one of the planar scanners.

4 3D Obstacle Avoidance

Obstacle avoidance is realized by a network of competing behaviors: EVasion (EV), Keeping Distance to obstacles at Left and Right side of the robot (KDL, KDR), collision avoidance by slow-down (Anti Collision Front – ACF, Anti Collision Rear – ACB), instantiated for each sensor class (2D scanners, ultrasonic sensors, 3D scanner). They are implemented via the iB2C framework (see [12]).

Figure 3: Sector maps of planar laser scanners

Input of each behavior module is not the raw sensor data, but an abstraction as sector maps. In these maps the distance data that belongs to one sector is collected and represented by the most prominent one, i.e. the one with minimum distance to the robot. Sector maps may either consist of polar or Cartesian sectors. Figure 3 shows the sector maps that are filled with data from the two planar laser scanners. The Cartesian maps consist of 10 parallel stripes that cover the space in front of and behind the robot. The global polar sector map consists of 50 radial sectors that cover the complete space around the robot. They are filled with distance data from the two scanners and from the obstacle memory. Each sector contains the distance between the nearest object and the robot frame as well as between this object and the map origin. This way a sector map represents a virtual distance sensor. For the 3D scanner a similar set of sector maps is created and filled with data from the local obstacle grid map. The data of the ultrasonic sensors is only used to fill two polar sector maps centered at the virtual sensor centers S1, S2 (see figure 2). Each sector covers exactly one sensor cone, i.e. it is directly filled with the corresponding sensor value. Table 1 shows the combination of control behavior and used sector map.

<table>
<thead>
<tr>
<th>2D scanner</th>
<th>3D scanner</th>
<th>ultrasound</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cartf</td>
<td>Cartg</td>
<td>poli</td>
</tr>
<tr>
<td>Cartf</td>
<td>Cartg</td>
<td>poli</td>
</tr>
<tr>
<td>poli</td>
<td>poli</td>
<td>poli</td>
</tr>
</tbody>
</table>

Table 1: Combination of robot motion control behaviors and used sector maps (Cartesian, polar, front/rear/global)

Correspondingly for each of these 5 functional units there is one group with three behavior modules, each operating on another sector map with a different set of parameters that define the reaction on the distance values in the distinct sectors. Hence the vector of output values of the behaviors within each group has to be fused into one vector of the group. As an example, figure 4 shows the fusion for the ACF group. Each behavior generates a control value \( c \in [0, 1] \), in this case to slow down the forward motion of the robot. This value is accompanied by two values that describe the internal state of the behavior: activity \( a \) – how strong this behavior wants to influence the control system and target rating \( r \) – how satisfied the behavior is with the current system state. For more information on the iB2C terminology and principles please see [12].

The fusion behavior combines the control values either by weighting them according to the respective activity or - as in this case - by taking the control value of the behavior with the highest activity (see equation 1). The ACB group has the same layout and generates a control value to decelerate the backward motion.

\[
c = c_m \quad \text{where} \quad a_m = \max_{i \in M}(a_i) \quad \text{and} \quad M = \{AC_{US}, AC_{2D}, AC_{3D}\}
\]

(1)

In contrast to this, the KDL, KDR and EV groups calculate control values for the turning motions by preferring one direction and by restricting the opposite direction. Consequently the control values for turning left and/or right from these three groups have to be fused again on the next level, i.e. outside of these groups, this time by calculating a weighted sum according to equation 2.

\[
c = \frac{\sum_{i \in M} a_i \cdot c_i}{\sum_{i \in M} a_i} \quad \text{where} \quad M = \{KDL, KDR, EV\}
\]

(2)

The benefit of this approach is an increase of system dependability: even if one or two sensor systems fail a reasonable motion control is guaranteed based on the distance values of the remaining sensors. However this behavior may not be optimal in any case, e.g. tabletops are not detected using only the planar scanners. On the other hand if all systems are running – which is the default case – the chance of overlooking any obstacle is negligible: assuming only forward motions at a limited velocity guarantees that the complete 3D space in motion direction is covered dense enough by the rotating scanner.
more than 0.5 m or turned more than 17° while the robot is moving. Whenever the robot has moved the local sensor grid map is evaluated at regular intervals. The data of the 3D scanner is used in two different ways: smaller size is too expensive for online updates. This could prevent the robot from entering narrow passages, a small adaptation of the occupancy attributes of each cell in the sensor grid map is “copied” into the global map by evaluating distance data from the three laser scanners. Measurements of the ultrasonic sensors are not accurate enough for a reasonable integration in the map. For the two planar scanners ray tracing is performed, i.e. for each laser beam within one scan (361 beams/distance values for 180° scan range at 0.5° resolution) the grid cells that are crossed by the beam are marked as free and the one where it hits an object is marked as occupied. The occupancy information is stored in a counter \( c \in [-127, 127] \) per cell which is increased by 1 for each new “occupied” event and decreased by 1 for each new “free” event. Correspondingly the occupancy state of a cell is given in equation 3.

\[
\text{cell} = \begin{cases} 
\text{free} & \text{if } c < 0 \\
\text{unknown} & \text{if } c = 0 \\
\text{occupied} & \text{if } c > 0
\end{cases}
\]

This way a kind of probability is added to the occupancy values, i.e. the higher the absolute value of the counter is, the higher is the confidence about the cell state. Cells which are swept by the robot are set to the lowest counter value as they are surely traversable. The cell size is set to 100 \times 100 \text{mm}^2 which is a compromise between accuracy and effort of computation and storage. A bigger cell size could prevent the robot from entering narrow passages, a smaller size is too expensive for online updates.

The data of the 3D scanner is used in two different ways to update the grid map. First the information stored in the local sensor grid map is evaluated at regular intervals while the robot is moving. Whenever the robot has moved more than 0.5 m or turned more than 17° the complete sensor grid map is “copied” into the global map by evaluating the set of occupancy attributes of each cell in the sensor map (see [11]). Second at each NBV position a raw 3D point cloud is collected during a 360° panorama scan. Here the scanner rotation is stopped and the distance data is recorded while the robot performs one turn around its z axis. All 3D distance samples that may interfere with the robot set the counter of the corresponding grid cell to the maximum value. Thus the occupancy belief does not depend on the number of data samples that fall into the respective grid cell but is set to “sure” by at least one sample in this cell. This is motivated by the accuracy of laser based distance data and the data acquisition while the robot is standing still.

Of course each update of the global map requires that the distance data acquired at the robot is transferred without errors into the global reference frame. In case of the planar scanners the coordinate transformation of distance values consists of a transformation from the fixed sensor center to the robot coordinate system (see figure 2) and then – using the robot pose – to the global reference frame. In case of the 3D scanner there is an additional transformation between the rotating sensor center and the mount position of the rotation unit. For this the exact rotation angle is needed. It is measured by an incremental encoder attached to the shaft of the driving motor. The measurement error is neglectable as the motor is mechanically connected by gears to the rotation axis of the scanner and the delay between the measurements of the rotation angle and the distance values is especially taken into account (see [13]).

Summarizing the evident source of error in this transformation chain is the robot pose. It is estimated based on odometry using incremental encoders attached to the driving wheels. As pose errors are accumulated quickly during this path integration a standard grid map based SLAM algorithm has been integrated in order to correct the odometry pose estimate at regular intervals. For this purpose an implementation of the Distributed Particle SLAM ([9], [10]) approach that is freely available for research work at http://www.cs.duke.edu/~parr/dpslam is applied in the developed mapping and exploration system. The benefits of this approach are its efficiency, reliability and straightforward integration into the developed mapping system.

The DP-SLAM system uses the odometry pose \( p_{\text{odo}}(t) = (x_{\text{odo}}(t), y_{\text{odo}}(t), \theta_{\text{odo}}(t)) \) and a corresponding distance data set \( d(t) \) from the front planar scanner to fill a grid map of its own as environmental representation. The belief about this model and the robot position is represented by a set of particles. In contrast to standard particle SLAM there is not one map per particle, but only one single map where each grid cell stores the history of particles that made changes to the occupancy value of the cell in a balanced ancestor tree. This core idea of DP-SLAM constitutes the computational and storage efficiency. As a result each particle of the most recent generation carries a probability and associated robot pose belief \( p_{\text{bel}}(t) = (x_{\text{bel}}(t), y_{\text{bel}}(t), \theta_{\text{bel}}(t)) \). To increase accuracy the grid size is set to 25 \times 25 \text{mm}^2 according to the specified error of distance measurements. To save computation time and thus increase the update rate only 180 distance samples with 1° angular resolution are used.

The DP-SLAM implementation has been integrated with small adaptations concerning its interface and the used parameters describing the properties of the differential drive, i.e. typical error of encoder sensors and wheel slippage. It runs in a thread parallel to the main mapping/exploration program as fast as possible on a dual core processor. Fig-

![Figure 4: Fusion of slow down values from AC modules](http://www.cs.duke.edu/~parr/dpslam)
Figure 5 illustrates the data flow in the control system. The DP-SLAM module transforms $p_{odo}(t_i)$, $\vec{d}(t_i)$ as input into $p_{belief}(t_i)$, i.e. most probable robot pose at time $t_i$, as output.

![Flow of robot poses and distance data between hardware, localization and slam modules](image)

For this purpose the difference $\Delta e_r(t_j)$, $\Delta e_l(t_j)$ of continuous readings from the incremental left/right wheel encoders are transformed into a robot pose estimate $p_{odo}(t)$ and put into two fifo queues in the main thread of the control program. The odometry based localization module just stores the history of $p_{odo}(t_i) = p(t_i)$ whereas the slam based localization module uses $p_{belief}(t_i)$ to correct these pose estimates $p_{odo}(t_i) \mapsto p(t_i)$. Storing both pose lists is necessary because the DP-SLAM algorithm needs the estimated path $[p_{odo}(t_i) - p_{odo}(t_{i-1})] = (\Delta x_{odo}, \Delta y_{odo}, \Delta \theta_{odo})(t_i)$ that has been traveled since the last slam update as input. That means this input must not be influenced by the correction $p_{belief}(t_i)$. On the other hand the mapping and exploration system needs the most accurate pose estimate available.

![Sequence diagram of information flow between hardware, localization and slam modules](image)

As the DP-SLAM module runs decoupled and a slam update cycle takes longer than odometry updates, the correction via $p_{belief}(t_i)$ is propagated to all odometry poses that have been collected by the slam based localization module since $t_i$.

These time relationships are illustrated in figure 6. Encoder values $\Delta e_{r,l}(t_i)$ are passed at regular intervals to both localization modules. The slam unit is initialized by a data set $p(t_0), \vec{d}(t_0)$. As soon as the next data set $p(t_1), \vec{d}(t_1)$ is available the first slam update cycle is started. In the meanwhile pose updates up to $p(t_i)$ are stored in the fifos of the localization modules by the main execution thread. When the slam update has been finished the most probable pose estimate $p_{belief}(t_1)$ is passed to the slam localization module. This triggers a correction of the correlated odometry pose $p(t_1) \mapsto \tilde{p}(t_1)$. Of course this correction affects all more recent odometry poses that have been added to the fifo during the DP-SLAM update cycle: $p(t_2) \mapsto \tilde{p}(t_2), \ldots, p(t_i) \mapsto \tilde{p}(t_i)$. That means the most recent accurate pose estimate provided to the mapping and exploration system is $\tilde{p}(t_i)$. After this update the next slam update cycle starts with a new input data set $p(t_i), \vec{d}(t_i) -$ the newest values available at this point of time. As one slam cycle takes significantly longer than the odometry updates, only the $i$-th odometry pose is directly corrected by the slam pose belief. Thus the propagation of correction values assures that all poses in the slam localization fifo are as accurate as possible.

6 Navigation Based on Elastic Bands and Online Map Update

Figure 7 summarizes the steps executed during one exploration cycle. First the map is updated by a 360° panorama scan with the 3D scanner (see section 5). Then the best pose for performing the next panorama scan, i.e. the next best view (NBV), is calculated.

![Exploration steps](image)

This step comprises extracting frontiers between known and unknown space, calculating of NBV candidates along frontiers and computing the cost of traveling to these candidates. As best pose the one with minimal path cost is selected in Greedy manner. Finally the robot starts moving.
towards this goal pose along the calculated path. While the robot is moving a continuous free space analysis is performed by transforming the path into an elastic band that takes the robot dimensions into account. As this band is adapted to all objects added online to the map it stays up to date and assures that the goal pose is still reachable on the chosen path. Besides it smooths the path incrementally and thus reduces travel time. As soon as the elastic band is tied up a new path is planned and the goal access is restarted. When the current NBV has either been reached successfully or been abandoned after a certain number of unsuccessful replanning operations a new exploration cycle is triggered.

**Frontier Extraction** At each point of time the grid map consists of cells that are either free, occupied or unknown. A frontier cell is a known cell (free or occupied) adjacent to at least one unknown cell. Thus the frontier cells can be calculated by linear processing of all grid cells while looking at the respective neighbor cells. A frontier is a sequence of connected frontier cells that separate known from unknown space. The longer a frontier is the higher is the expected information gain, i. e. unknown space that may be acquired by a panorama scan at the center of this frontier. Therefore the pure knowledge about single frontier cells is not sufficient.

![Figure 8: Frontier extraction and NBV calculation in a simulated lab scenario (without slam and wheel slippage). Left: Grid map with occupied (red circles), free (yellow/orange circles) and frontier cells (black crosshairs) Right: Grayscale image of grid map (upper) and extracted contours (lower)](http://sourceforge.net/projects/opencvlibrary)

To compute the sequences of frontier cells efficiently the grid map is transformed into a grayscale image and a contour analysis is performed using the image processing library OpenCV¹. In this image each pixel represents one grid cell. The image is initially set to black and then free cells are marked gray and occupied cells white. The contour processor returns a list of pixel sequences each representing a connected contour of an image area. Each sequence in this list is processed sequentially and split into clusters of free and occupied cells. If occupied cells were not added to the image the contours might incorrectly mark frontiers directly adjacent to known obstacles, e. g. room walls. But as these features are added to the image, a closed contour of a partially known room consists of connected subsets of occupied and free cells, representing walls and border of free space. Therefore such a contour is split into several frontiers.

When all frontiers have been extracted from the contour list, they are sorted according to decreasing length, i. e. number of grid cells. Finally the center cells of the 10 longest frontiers are selected as NBV candidates. Figure 8 shows the result of the frontier extraction after an initial panorama scan in a simulated lab environment. The robot is part of the circular shaped area in the middle right section of the map. The cells below the robot are marked as free. They are surrounded by a circular band of unknown cells which lie in the blind area of the 3D scanner. The free space is one connected area that surrounds the robot. It is disrupted by obstacles and unknown space that is shadowed by obstacles. The grayscale image highlights these holes and the contour image shows the resulting scatter of contours. Most of them are filtered out due to insufficient size during frontier extraction. Thus the analysis results in four big frontiers which are marked by a blue line that connects the robot pose with the corresponding center cell as NBV candidate. The one which is connected to the robot by the purple navigation path has been selected as best one based on minimal travel cost.

**NBV Calculation** Calculating the cost of traveling from the current robot pose to a NBV candidate is performed by planning a navigation path between these poses. The path planner executes a standard A* algorithm on the grid map. The core operation of this algorithm is to calculate the cost $C(c_S, c_G)$ for moving from a start cell $c_S$ to a goal cell $c_G$ according to equation 4.

$$C(c_S, c_G) = \begin{cases} \frac{ncf^2}{nc} & \text{if } c_G \text{ is occupied} \\ \frac{ncf}{nc} & \text{if } c_G \text{ is directly adjacent to an obstacle} \\ \frac{\|c_S - c_G\| + ncf}{d(c_G) - 1 + n cf e} & \text{if } c_G \text{ is near an obstacle} \\ \|c_S - c_G\| & \text{otherwise} \end{cases}$$

In this function $\|c_S - c_G\|$ is the Euclidean distance between the center of the two grid cells and $d(c_G)$ is the distance in cell units between $c_G$ and the nearest obstacle, i. e. number of cells between $c_G$ and the nearest occupied cell. $ncf$ is the neighbor cost factor and $ncfe$ is the neighbor cost function exponent, two parameters that influence the cost of traveling to $c_G$ that is occupied or near to an occupied cell. This neighborhood is determined via an adjustable neighbor range parameter $nr$. All cells that lie in a circle with radius $nr$ around an obstacle are marked as neighbor cells. This approach is needed to take the robot dimensions into account since the robot pose is represented by the cell occupied by the center of the robot and the size of a grid cell is less than the robot dimensions ($100 \times 100 \text{ mm}^2$ vs. circle with radius 400 mm). Thus

¹ [http://sourceforge.net/projects/opencvlibrary](http://sourceforge.net/projects/opencvlibrary)
a navigation path is only valid if the robot never enters a neighbor range of 4 cell units.

The $n_{cf}$ parameter is used to prevent the path planner from entering the neighbor range and, in combination with $n_{cf)e}$, to prefer cells that are far away from obstacles in order to avoid slow robot motions. Of course detours have to be minimized simultaneously to not increase the travel distance unnecessarily. This trade-off is described by the actual parameter setup. It has been set to $nr = 8$, $n_{cf} = 60000$ and $n_{cf)e} = 4$ as result of a series of test runs. The cost function $C(v_{G}, v_{O})$ has been designed to provide a path that runs in a smooth curve around obstacles while assuring that a valid path is found when there is one. To guarantee that no valid path is rejected due to an invalid neighbor range the path may enter this range, but never cross obstacle cells. A complete free space analysis is performed via an elastic band algorithm that uses the planned path to the NBV as seed.

**Heading for NBV** When the NBV candidate with minimal path cost has been selected the robot starts moving on the computed path. For this purpose the path is translated into an elastic band of connected bubbles that cover the free space along the path. This procedure is based on [14] and described in detail in [15] (section 7.2).

Its benefit is threefold: first, the size of the smallest bubble allows to decide whether there is enough free space along the path to reach the NBV, i.e., when at least one bubble is smaller than the robot for a certain amount of time the path gets invalid and a replanning is triggered; second, it smoothes the path as bubbles are repulsed from obstacles and simultaneously contracted along the band; third, it is updated during robot motion and thus takes features that are added online to the map instantly into account.

Naturally during exploration the environment is only partially known so the elastic band can only guess the free space that is beyond the current sensor range. But since it is updated online during robot motion it adapts the navigation path to the map knowledge as fast as possible. As the robot always moves towards the center of the nearest bubble which lies in its current sensor range it does only get close to obstacles in narrow passages so slow-downs are minimized. When a static obstacle ties up the band the robot is stopped and a new path is planned. This procedure is only repeated for a certain number of operations so that a NBV which is not reachable in a certain amount of time is discarded. Then a new panorama scan is performed and a new NBV is calculated based on the updated map. Hence selecting the same unreachable target is impossible.

Summarizing the combination of 3D obstacle acquisition, online map update, dependable safety behaviors and the free space analysis via elastic band enables to the robot to safely cross regions that are unknown during NBV selection. The path planner and the elastic band analyzer handle unknown cells as free ones so there is no computational overhead due to restricting the navigation path to known areas. Besides crossing unknown regions on the way to the NBV speeds up exploration as new information is added online to the map, not only during the panorama scans. This makes the whole process really dynamic.

### 7 Experiments

The exploration process has been validated on MARVIN in a typical office environment. First an experiment has been performed in a 3D simulation of this workspace based on SimVis3D (see [8]). The test scenario consists of a big lab and 26 office rooms connected to a common hallway (ca. 880 m² in total). Only the lab is equipped with furniture. Doors are omitted so that the robot can enter all rooms. Dynamic obstacles are not present, too. Errors of the distance sensors are simulated by adding Gaussian noise to the data. Wheel slippage is simulated by a 1.16% error of the encoder tics per wheel rotation.

![Figure 9: Recorded simulation map after 77 exploration cycles. 7 rooms are not completely covered (arrows).](image)

The exploration is started by a panorama scan from the middle of the lab, see figure 8. But now the slam component is active to correct the sensor noise and wheel slippage. Consequently the recorded map is not as exact as in this figure. As the robot may be located by the slam estimate up to 30 cm away from its true pose the walls for instance may cover up to 5 (instead of 2) parallel lines of grid cells. Besides there is a counter clockwise rotational offset of about 14° during the initial panorama scan as the slam subsystem is started from scratch. So the geographical orientation of the map is marked by a compass rose.

*Figure 9* shows the map after the exploration process did not find any new NBV. It took 77 exploration cycles and 176 minutes compared to an optimal predefined NBV setting with 46 positions/132 minutes. Despite the inaccuracies of the robot location the rectangular structure of the environment is mapped quite well and the workspace is almost completely covered. There are still 7 frontiers (marked by arrows), that are extracted as valid contours but not transformed to NBV candidates as small contours are filtered out. Hence the corresponding parameter settings have to be adapted.

*Figure 10* shows the result of exploring the real envi-
vironment with MARVIN. The sequence of NBV positions shows that the robot quickly left the lab and entered the hallway due to the high amount of clutter (tables, PCs, other robots). The narrow door passage and limited number of replanning per NBV resulted in the clustering of pose 2 and 3. Up to NBV 14 the exploration is straightforward as most of the doors were closed. But then the closed leaf of a door covering half of the corridor forced the robot to turn back to the other end of the hallway at pose 15.

![Figure 10: Recorded real map after 28 cycles with numbered NBV positions.](image)

Then the scoring of frontiers based on their length and accessibility led the robot to continue the exploration near pose 14. This time the door had been opened completely to avoid further oscillations. After pose 22 the north part of the corridor had been completely explored and the robot turned back. At pose 23 some people blocked the passage so the robot preferred moving to the west end (pose 24). Finally it managed to explore the east part of the corridor up to NBV 28. At pose 27 it also tried to map a room but failed to cover it completely as it was blocked by people and furniture. The whole process took about 88 minutes. The experiments show that the combination of 3D obstacle avoidance, slam, online map update and elastic band navigation creates a feasible, almost complete map of a typical indoor scenario without restricting the robot to only move within known space.

8 Conclusion

In this paper an enhancement of the well established frontier based exploration approach has been presented. By using a dependable 3D obstacle avoidance strategy and an off-the-shelf slam component for robot localization two main restrictions of state of the art exploration can be alleviated: the robot is able to leave the safe region of already explored space while traveling to the next best sensing pose and map features can also be added online during this trip. The first item saves computation time as planning and online validation of navigation paths has not to decide between free and unknown space. Both items make the exploration process really dynamic as the map is not only updated at singular NBV positions and the robot may move through unknown space during NBV access. However the evaluation of the frontiers renders the exploration decision where to move next straightforward and serves as termination criterion.

The developed approach has been tested successfully both in simulation and reality. Suboptimal parameters for evaluation frontiers and path cost prevented a complete map coverage and caused NBV oscillations. To correct this behavior is part of future work. Besides the NBV selection can be fine tuned by weighting frontier length as estimate of information gain and path cost. This may provide further potential for exploration efficiency. Finally the travel time and length has to be compared to a standard frontier based exploration in order to assess the proposed speed up.

References