

Autonomous Robot Navigation in Highly Populated Pedestrian Zones

Rainer Kümmerle

Department of Computer Science
University of Freiburg
79110 Freiburg, Germany
kuemmerl@informatik.uni-freiburg.de

Michael Ruhnke

Department of Computer Science
University of Freiburg
79110 Freiburg, Germany
ruhnke@informatik.uni-freiburg.de

Bastian Steder

Department of Computer Science
University of Freiburg
79110 Freiburg, Germany
steder@informatik.uni-freiburg.de

Cyrill Stachniss

Department of Computer Science
University of Freiburg
79110 Freiburg, Germany
stachnis@informatik.uni-freiburg.de

Wolfram Burgard

Department of Computer Science
University of Freiburg
79110 Freiburg, Germany
burgard@informatik.uni-freiburg.de

Abstract

In the past, there has been a tremendous progress in the area of autonomous robot navigation and a large variety of robots have been developed who demonstrated robust navigation capabilities indoors, in non-urban outdoor environments, or on roads and relatively few approaches focus on navigation in urban environments such as city centers. Urban areas, however, introduce numerous challenges for autonomous robots as they are rather unstructured and dynamic. In this paper, we present a navigation system for mobile robots designed to operate in crowded city environments and pedestrian zones. We describe the different components of this system including a SLAM module for dealing with huge maps of city centers, a planning component for inferring feasible paths taking also into account the traversability and type of terrain, a module for accurate localization in dynamic environments, and means for calibrating and monitoring the platform. Our navigation system has been implemented and tested in several large-scale field tests, in which a real robot autonomously navigated over several kilometers in a complex urban environment. This also included a public demonstration, during which the robot autonomously traveled along a more than three kilometer long route through the city center of Freiburg, Germany.

1 Introduction

Navigation is a central capability of mobile robots and substantial progress has been made in the area of autonomous navigation in the past. The majority of navigation systems developed thus far, however, focuses on navigation in indoor environments, through rough outdoor terrain, or based on road usage. Only few systems have been designed for robot navigation in populated urban environments such as, for example, the autonomous city explorer (Bauer et al., 2009). Robots that are able to successfully navigate in urban environments and pedestrian zones have to cope with a series of challenges including complex three-dimensional settings and highly dynamic scenes paired with unreliable GPS information.

In this paper, we describe a navigation system that enables mobile robots to autonomously navigate through urban city-center. Our system builds upon and extends existing technology for autonomous navigation. In particular, it contains a SLAM approach for learning accurate maps of urban city areas, a dedicated map data structure for dealing with large-scale maps, a variant of Monte-Carlo localization that utilizes this data structure, and a dedicated method for terrain analysis that deals with vegetation, dynamic objects, and negative obstacles. We furthermore describe how these individual components are integrated and how the sensors are calibrated. In addition, we present the result of a large-scale experiment, during which the robot Obelix navigated autonomously from the campus of the Faculty of Engineering of the University of Freiburg to the city center of Freiburg during a busy day in August 2012. During that trial, the robot had to master a distance of more than 3 km. The trajectory taken by the robot and two pictures taken during its experiment are depicted in Fig. 1.

The aim of this paper is to not only describe the relevant components but also to highlight the capabilities that can be achieved with such a system. We motivate our design decisions, discuss critical aspects as well as limitations of the current setup. This paper extends our previous work (Kümmerle et al., 2013) as it contains more in-depth explanations and clarifying figures, details on sensor calibration and monitoring of the running modules, and an extended experimental evaluation.

2 Related Work

The problem of autonomous navigation in populated exhibition areas has been studied intensively in the past. Among the early systems are the robots RHINO (Burgard et al., 1998) and Minerva (Thrun et al., 1999), which operated as interactive mobile tour-guides in crowded museums. An extension of this tour-guide concept to interactive multi-robot systems is the RoboX system developed by Siegwart et al. (2003) for the Expo'02 Swiss National Exhibition and the TOURBOT system by Trahanias et al. (2005). Gross et al. (2009) installed a robot as a shopping assistant that provided wayfinding assistance in home improvement stores. Although these systems are able to robustly navigate in heavily crowded environments, they assume that the robots operate in a relatively confined planar area and furthermore are restricted to two-dimensional representations of their environment.

Relatively few robotic systems have been developed for autonomous navigation in city centers. The concept closest to the one described in this paper probably is the one of the Autonomous City Explorer developed by Bauer et al. (2009). In contrast to our system, which operates completely autonomously, the Autonomous City Explorer project mainly focuses on human-robot interaction. The robot communicates with pedestrians to get the direction for where to move next. It builds local maps and does not autonomously plan its path from its current position to the overall goal location. Mirnig et al. (2012) investigated methods to further improve the communication between a human and a robot asking for directions. Other researchers developed robots that monitor pollutions in city environments and are designed to improve trash collection (Reggente et al., 2010). A further related approach has been developed in the context of the URUS project (Sanfeliu et al., 2010), which considered urban navigation but focused more on networking and collaborative actions as well as the integration with surveillance cameras and portable devices. Another result of the URUS project (Trulls et al., 2011) considers urban navigation in pedestrian-only environments. Compared to our



Figure 1: Example trajectory traveled by our robot navigating through Freiburg, Germany. It includes a pedestrian zone with a large number of people. The map data on the left is from OpenStreetMap (© OpenStreetMap contributors).

city-scale navigation system, their approach operates at a smaller scale in terms of the dimension of the map and the area covered while navigating autonomously. Due to the pedestrian-only scenario, it does not in particular address navigating on sidewalks and crossing streets. Additionally, the system developed by Morales et al. (2009) allows a robot to navigate over pedestrian walkways. In contrast to our approach, their system relies on a list of pre-defined waypoints which the robot follows to reach the desired goal location.

Also, the problem of autonomous navigation with robotic cars has been studied intensively. For example, there has been the DARPA Grand Challenge (Seetharaman et al. (2006)) during which autonomous vehicles showed an impressive capability to navigate successfully over large distances through desert areas (Cremean et al., 2006; Thrun et al., 2006; Urmson, 2005). Throughout the later DARPA urban challenge, several car systems have been presented that were able to autonomously navigate through dynamic city street networks with complex car traffic scenarios and under consideration of road traffic navigation rules (Urmson et al., 2008; Montemerlo et al., 2008; Rauskolb et al., 2008). Such competitions or demonstrations like the European Land Robot Trial (ELROB) have the potential to accelerate robotics research and to make the general public aware of the current state of the art (Schneider and Wildermuth, 2011). Recently, commercial self-driving cars (Google Inc., 2012) have been developed and legalized to perform autonomous navigation with cars. In contrast to such approaches, which focused on car navigation, the system described in this paper has been developed to enable mobile robots to perform autonomous navigation in densely populated urban environments with many types of dynamic objects like pedestrians, cyclists, or pets.

A long-term experiment about the robustness of an indoor navigation system was recently presented by Marder-Eppstein et al. (2010). Their system used a tilting a laser range finder for navigation and for obstacle detection. In contrast to this system, our approach also has a component for tracking moving obstacles to explicitly deal with the dynamic objects in highly populated environments. It furthermore includes a terrain analysis component that is able to deal with a larger variety of terrain.

While the approaches discussed so far focus on ground-vehicles, navigation with flying robots can be realized by applying the same paradigms but require specific adaptations to the demands of flying robots (Bachrach et al., 2011; Grzonka et al., 2012). An essential element of navigation with both, ground-based and flying robots, is estimating the pose of the vehicle, for example, by localizing the robot against a map by means of a particle filter (Dellaert et al., 1998). Navigating in an environment that is as large as a city creates high demands on the mapping and localization algorithms. Newman et al. (2009) presented an approach that combines vision and laser data along with topological and metric representations to estimate maps of urban environments. Although their robot is navigating outdoors in such scenarios, it cannot rely on external systems such as GPS because the corresponding signals might be partially or completely blocked. Therefore, the robot has to estimate its pose with the data of the onboard sensors and a map (Georgiev and Allen, 2004). Furthermore, the robot may perform specific actions to resolve ambiguities and consequently converge faster to a single hypothesis (Fox et al., 1998; Murtra et al., 2008).

Instead of estimating a map, which is subsequently used for localizing the robot, the data recorded during following a route can be utilized more directly. Furgale and Barfoot (2010) presented a teach and repeat system based on stereo vision. In a first step, the robot is manually driven along a route and creates sub-maps by visual odometry. Within the second step, the retraversal of the learned route, a localization algorithm allows the robot to follow the taught route also in urban settings. Royer et al. (2005) suggested a similar system based on monocular vision. Compared to our system, both approaches rely on a previously taught path to reach a specific goal location.

3 The Robot Obelix used for the Evaluation

The robot used to carry out the field experiments is a custom made platform developed within the EC-funded project EUROPA (2009), which is an acronym for the European Robotic Pedestrian Assistant. The robot has a differential drive that allows it to move with a maximum velocity of 1 m/s. Using flexibly mounted caster

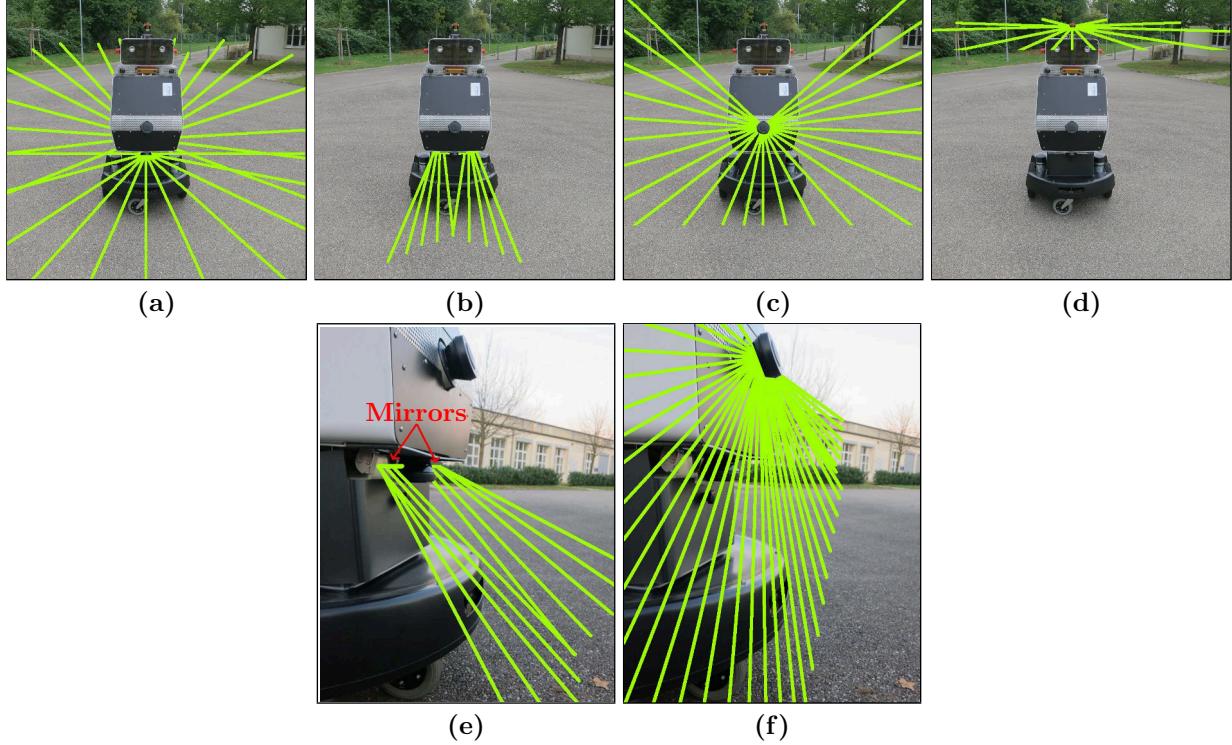


Figure 2: The laser sensor setup of our robot. The scan patterns of different lasers are visualized by the green lines. (a): The robot is equipped with two horizontal SICK lasers, one 270° scanner to the back and one 270° scanner to the front, whereas only the middle 180° beams are used horizontally. The remaining beams are reflected onto the ground by mirrors, building a cone in front of the robot, as visible in (b)&(e). (c)&(f) show a 270° SICK scanner in the front of the robot that is tilted downwards to scan the ground in front of the robot. (d) shows a small 270° Hokuyo scanner on a tilting servo on the head of the robot. The beams shown in (b) and (c) are mainly used to perform the traversability analysis, whereas the beams shown in (a) and (d) are mostly used for planning, localization and mapping purposes.

wheels in the front and the back, the robot is able to climb steps up to a maximum height of approximately 3 cm. Whereas this is sufficient to negotiate a lowered pedestrian sidewalk, it has not been designed to go up and down normal curbs so that the robot needs to avoid such larger steps. The footprint of the robot is 0.9 m × 0.75 m and the robot is around 1.6 m tall. Additionally, a touchscreen monitor is mounted in the back of the robot. The screen allows an experienced user to inspect the state of the navigation system. During our demonstrations the screen features information about the planned route of the robot as well as dialog windows to interact with users, for example, the robot displays a dialog to ask for permission before crossing a street.

The main sensor input is provided by laser range finders. Two SICK LMS 151 are mounted horizontally in the front and in the back of the robot (see Fig. 2a). The horizontal field of view of the front laser is restricted to 180°. The remaining beams are reflected by mirrors to observe the ground surface in front of the robot (see Fig. 2b). Additionally, another scanner is tilted 70° downwards to detect obstacles and to identify the terrain the robot drives upon (see Fig. 2c). A Hokuyo UTM-30LX mounted on top of the head of the robot is used for mapping and localization (see Fig. 2d), whereas the data of an XSens IMU is integrated to align the UTM horizontally by controlling a servo accordingly. The robot is furthermore equipped with a Trimble GPS Pathfinder Pro to provide prior information about its position during mapping tasks. While the robot also has stereo cameras onboard, their data is not used for the described navigation tasks. Rather the images are only used for the sake of visualization in this paper. The entire navigation system described in this paper runs on one standard quad core i7 laptop operating at 2.3 GHz.

4 System Overview

In order to autonomously navigate in an environment, our system requires to have a map of the area. This might seem like a drawback, but mapping an environment can be done in a considerably small amount of time. For example, it took us around 3 hours to map a 7.4 km long trajectory by controlling the robot with a joystick. Furthermore, this only has to be done once, as the main structures of an urban area do not change quickly. Small modifications to the environment, like billboards or shelves placed in front of shops, can be handled by our system in a robust manner. In the following, we describe how we obtain the map of the environment by means of a SLAM algorithm as well as the most important components of the autonomous navigation system, such as the algorithms for localization, path planning, calibration, and obstacle detection, which enable our robot to operate in large scale city centers.

4.1 Calibration

As described in Section 3, our robot is equipped with multiple laser scanners. To enable the system to combine the measurements from the different sensors in a consistent manner, an accurate calibration for the positions of the sensors on the robot is required. To this end, we in a first step extended our simultaneous calibration method (Kümmerle et al., 2012) to handle several laser range finders. The objective of the calibration is to estimate the positions $\mathbf{d} = (\mathbf{d}^{[1]}, \dots, \mathbf{d}^{[K]})^\top$ of the range finders, where $K = 3$ for our robot.

Let \mathbf{x}_i be the pose of the robot at time i . The error function $\mathbf{c}_{ij}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{d}^{[k]}, \mathbf{z}_{ij}^{[k]})$ measures how well the parameter blocks \mathbf{x}_i , \mathbf{x}_j , and $\mathbf{d}^{[k]}$ satisfy the measurement $\mathbf{z}_{ij}^{[k]}$, which is obtained as the scan-matching result of the range data acquired by the respective range finder at time i and j . If the three parameters perfectly satisfy the error function, then its value is $\mathbf{0}$. Here, we assume that the laser is mounted without inclination, which is the ideal condition. The error function $\mathbf{c}_{ij}(\cdot)$ has the following form:

$$\mathbf{c}_{ij}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{d}^{[k]}, \mathbf{z}_{ij}^{[k]}) = ((\mathbf{x}_j \oplus \mathbf{d}^{[k]}) \ominus (\mathbf{x}_i \oplus \mathbf{d}^{[k]})) \ominus \mathbf{z}_{ij}^{[k]}, \quad (1)$$

where \oplus corresponds to the standard motion composition operator (Smith et al., 1990) and \ominus denotes its inverse. We obtain the calibration result along with the positions of the robot by solving the following minimization:

$$(\mathbf{x}, \mathbf{d})^* = \operatorname{argmin}_{\mathbf{x}, \mathbf{d}} \sum_{k=1}^K \sum_{ij \in \mathcal{C}^{[k]}} \|\mathbf{c}_{ij}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{d}^{[k]}, \mathbf{z}_{ij}^{[k]})\|_{\Sigma_{ij}^{[k]}}, \quad (2)$$

where $\|\mathbf{c}\|_\Sigma \stackrel{\text{def.}}{=} \mathbf{c}^\top \Sigma^{-1} \mathbf{c}$ computes the Mahalanobis distance of its argument, $\Sigma_{ij}^{[k]}$ represents the uncertainty of the scan matching result, and $\mathcal{C}^{[k]}$ denotes the set of measurements that are available for a specific range finder. We solve Eq. (2), using the Levenberg-Marquardt algorithm as it is implemented in our g2o toolkit (Kümmerle et al., 2011).

To carry out the calibration procedure outlined above, we record a data set in an indoor environment to facilitate the scan-matching process. The trajectory taken by the robot has to consist of translations as well as rotation, which we can guarantee easily by controlling the robot appropriately. Otherwise, we would not be able to recover the positions of the onboard sensors (Brookshire and Teller, 2011).

The method described above is not applicable for the downwards facing laser and the mirrored beams (see Fig. 2b and c), since scan matching cannot estimate the ego-motion of the robot by considering those measurements. However, they observe the same region of the environment when the robot is moving. We therefore developed a module which exploits this to calibrate the downward facing laser and the mirrors to obtain consistent 3D measurements while driving with the robot. The calibration procedure works as follows: First, the robot is driven through a small area with significant 3D structure that is visible in both, the downwards facing laser and the mirrored beams. We typically used a straight trajectory of about 3 m.



Figure 3: (a) shows an example environment that we used for our calibration procedure. We created a narrow corridor with several objects on both sides, for the robot to drive through. These objects provide structure that is visible both in the downwards facing laser scanner, as well as the laser beams reflected by the mirrors. (b) shows the corresponding point clouds, whereas the black points are from the downwards facing scanner and the blue and red points are from the beams reflected by the left and the right mirror respectively. These point clouds were created with the already calibrated parameters.

Fig. 3a depicts an example environment that we used for the calibration procedure. After this data is collected, the point clouds of the different sources — the downwards facing laser, the beams hitting the left mirror, and the beams hitting the right mirror — are accumulated given the odometry of the robot and the current estimate for the demanded calibration parameters. Fig. 3b visualizes the resulting point cloud. The user is then able to manually change the parameters in a graphical user interface to provide a reasonable initial guess. Given this guess, the system starts a greedy sampling based method to improve the overlap of the different point clouds. That means the system changes the parameters of either one of the lasers or one of the mirrors, recalculates the corresponding point cloud and determines the overlap with the other clouds. If the overlap was improved, the calibration parameters are replaced with this solution. Else it is discarded. This process is then repeated. We found that this leads to a much more consistent calibration than what we achieved by hand-tuning the parameters and takes less time.

4.2 Mapping

We apply a graph-based SLAM formulation (Dellaert and Kaess, 2006; Grisetti et al., 2010) to estimate the maximum-likelihood (ML) configuration. In this formulation, the vector $\mathbf{x} = (\mathbf{x}_1, \dots, \mathbf{x}_n)^\top$ describes the poses of the robot where each element corresponds to the pose of the robot at a certain time. \mathbf{z}_{ij} and Σ_{ij} are respectively the mean and the covariance matrix of an observation describing the motion of the robot between the time indices i and j , whereas we assume Gaussian noise. Let $\mathbf{e}_{ij}(\mathbf{x})$ be an error function which computes the difference between the observation \mathbf{z}_{ij} and the expected value given the current state of node i and j . Note that the function $\mathbf{e}_{ij}(\mathbf{x})$ is closely related to $\mathbf{c}_{ij}(\cdot)$ (see Section 4.1) but it directly employs the previously found calibration result as known parameters. Additionally, let $\mathbf{e}_i(\mathbf{x})$ be an error function which relates the state of node i to its prior $\hat{\mathbf{x}}_i$ having the covariance matrix Σ_i . The function $\mathbf{e}_i(\mathbf{x})$ computing the difference is defined as

$$\mathbf{e}_i(\mathbf{x}) = \mathbf{x}_i \ominus \hat{\mathbf{x}}_i. \quad (3)$$

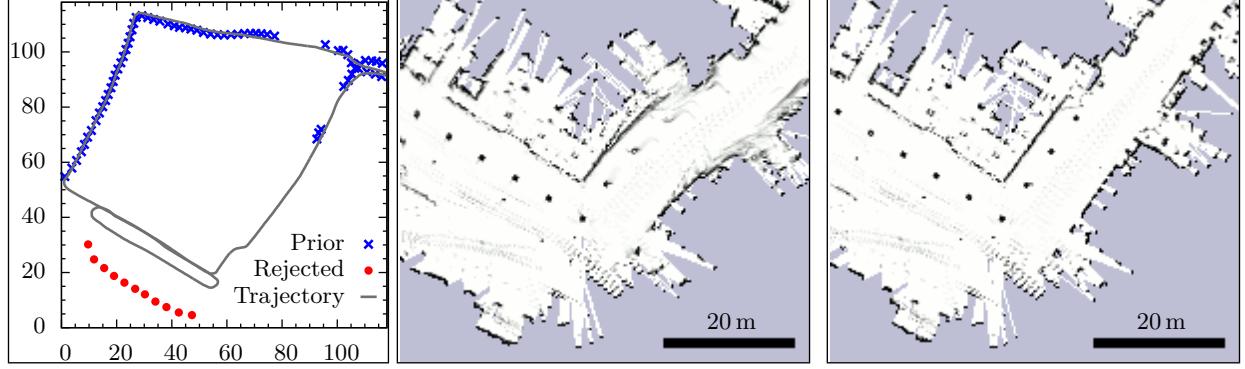


Figure 4: Influence of outliers in the set of prior measurements. Left: Our method rejects prior measurements having a large error. Middle: The map as it is estimated by taking into account all prior measurements. The outliers in the prior information lead to artifacts in the map. Right: Our method achieves a good estimate for the map by rejecting priors which are likely to be outliers. Compared to the image in the middle the walls are straight and the map better captures what the environment looks like.

Assuming the measurements are independent, we obtain the ML configuration of the robot’s trajectory as

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_{ij \in \mathcal{C}} \|\mathbf{e}_{ij}(\mathbf{x})\|_{\Sigma_{ij}} + \sum_{i \in \mathcal{P}} \|\mathbf{e}_i(\mathbf{x})\|_{\Sigma_i}, \quad (4)$$

where \mathcal{C} and \mathcal{P} are a set of constraints and priors, respectively. Eq. (4) corresponds to the negative log-likelihood of the joint probability of all measurements, see, for example, Dellaert and Kaess (2006) or Grisetti et al. (2010) for the details. Hence, the minimization in Eq. (4) complies with maximizing the likelihood of all measurements. Again, we employ our g2o toolkit (Kümmerle et al., 2011) for solving Eq. (4), which iteratively linearizes and solves the linear approximation until a convergence criterion is matched.

The laser-based front-end generating the set of constraints \mathcal{C} is an extension of the approach proposed by Olson (2008). We apply a correlative scan-matcher to estimate the motion of the robot between successive time indices. Furthermore, the front-end obtains loop closures by matching the current scan against all scans which are within the three-sigma uncertainty ellipsoid. If a loop closure hypothesis is found, we additionally perform a step to reject outliers. To this end, a pairwise consistency test forms rigid loops by combining two hypotheses along with the odometry of the robot. Each such loop should result in an identity matrix. By determining the difference between chaining the constraints of the loop and the identity transformation, we obtain a measure for the consistency for each pair of two hypotheses. For a set of hypotheses we want to identify the subset that is maximally self-consistent to robustly handle outliers. One way to identify this subset of hypotheses is to apply spectral clustering, as suggested by Olson (2009). In addition to the loop closures, we integrate a set of priors \mathcal{P} given by the GPS sensor. As GPS signals may be corrupted by multi-path effects, we apply an outlier rejection method to remove those measurements. Instead of directly solving Eq. (4), we consider a robust cost function – namely the Pseudo Huber cost function (Hartley and Zisserman, 2004) – for the prior measurements. After solving Eq. (4), we remove 2% of the prior edges having the largest residual. We repeat this process five times. Thus, we keep approximately 90% of the original prior information. Using this approach, some good GPS measurements might be rejected. However, we found in our practical experiments that the effect of outliers in the prior measurements may be severe (see Fig. 4). Including the prior information has several advantages. First, it improves the accuracy of the obtained maps (Kümmerle et al., 2011). Second, if the robot extends its map, coordinates are easy to transform between different maps, because the maps share a common global coordinate frame.

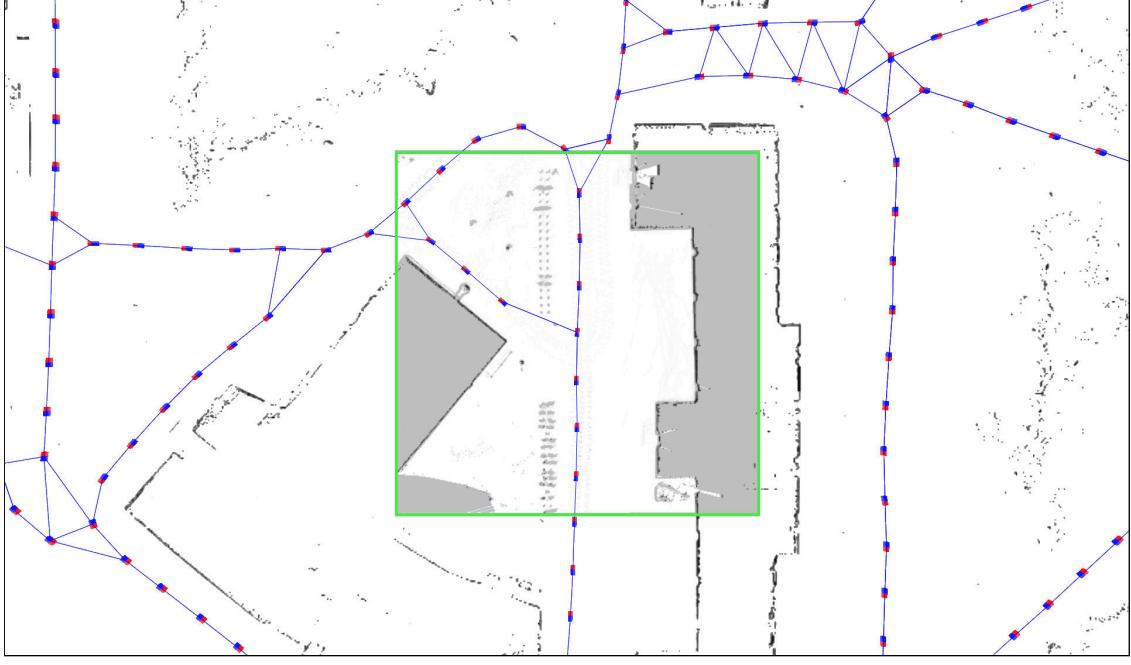


Figure 5: An example map of our campus. The graph nodes are colored red and blue, the edges are colored blue. The green marked region corresponds to a $40\text{ m} \times 40\text{ m}$ local map. Since we load only small parts of the environment into the main memory we use pre-computed low resolution thumbnails for visualization of the local maps that are not available.

4.3 Map Data Structure

Obtaining a 2D map given the graph-based SLAM solution and the laser data is typically done in a straightforward manner, for example, by computing an occupancy grid. However, storing one monolithic occupancy grid for a large-scale environment may lead to a large memory footprint. For example, a $2\text{ km} \times 2\text{ km}$ area at a resolution of 0.1 m and 4 bytes per cell requires around 1.5 GB of main memory. Instead of computing one large map, we use the information stored in the graph to render maps locally and close to the robot’s position. Therefore we select a subset of graph nodes from the SLAM solution which has a minimum distance of 5 m between all neighboring nodes in the subset. Furthermore we transfer the connectivity information of the SLAM pose graph to the constructed graph. This sparse graph represents the topology of our map and every node can store local maps (see Fig. 5). A similar approach was recently described by Konolige et al. (2011).

We generate the local occupancy map as follows. For a node in our map data structure, we apply Dijkstra’s algorithm to compute the distance to the nodes of the SLAM graph. This allows us to only consider observations that have been obtained by the robot in the local neighborhood of a given location. We compute the set of nodes to be used to build the local map as

$$V_{map} = \{\mathbf{x}_i \in \mathbf{x} \mid \text{dijkstra}(\mathbf{x}_i, \mathbf{x}_{base}) < \delta\}, \quad (5)$$

where V_{map} is the set of observations that will be used for obtaining the local map, \mathbf{x}_{base} the closest node to the robot’s current position, $\text{dijkstra}(\mathbf{x}_i, \mathbf{x}_{base})$ the distance between the two nodes according to Dijkstra’s algorithm, and δ the maximal allowed distance for a node to be used in the map rendering process. As hard-disk space is rather cheap and its usage does not affect the performance of other processes, we store each local map on the disk after the first access to it by the system.

The localization and path-planning algorithms described in the following sections all operate on these local

maps. The map is expressed in the local frame of \mathbf{x}_{base} and we currently use a local map of 40 m by 40 m.

4.4 Localization

To estimate the pose \mathbf{x} of the robot given a map, we maintain a probability density $p(\mathbf{x}_t \mid \mathbf{z}_{1:t}, \mathbf{u}_{0:t-1})$ of the location \mathbf{x}_t of the robot at time t given all observations $\mathbf{z}_{1:t}$ and all control inputs $\mathbf{u}_{0:t-1}$.

Our implementation employs a sample-based approach which is commonly known as Monte Carlo localization (MCL) (Dellaert et al., 1998). MCL is a variant of particle filtering (Doucet et al., 2001) where each particle corresponds to a possible robot pose and has an assigned weight $w^{[i]}$. In the prediction step, we draw for each sample a new particle according to the prediction model $p(\mathbf{x}_t \mid \mathbf{u}_{t-1}, \mathbf{x}_{t-1})$. Based on the sensor model $p(\mathbf{z}_t \mid \mathbf{x}_t)$ each particle within the correction step gets assigned a new weight. To focus the finite number of particles in the regions of high likelihood, we need to re-sample the particles. A particle is drawn with a probability proportional to its weight. However, re-sampling may drop good particles. To this end, the decision when to re-sample is based on the number of effective particles (Grisetti et al., 2005; Liu, 1996). Our current implementation uses 1,000 particles.

A crucial question in the context of localization is the design of the observation model that calculates the likelihood $p(\mathbf{z} \mid \mathbf{x})$ of a sensor measurement \mathbf{z} given the robot is located at the position \mathbf{x} . We employ the so-called endpoint model or likelihood fields (Thrun et al., 2005). Let z'_k be the k^{th} range measurement of \mathbf{z} re-projected into the map given the robot pose \mathbf{x} . Assuming that the beams are independent and the noise is Gaussian, the endpoint model evaluates the likelihood $p(\mathbf{z} \mid \mathbf{x})$ as

$$p(\mathbf{z} \mid \mathbf{x}) \propto \prod_i \exp\left(-\frac{\|z'_i - d'_i\|^2}{2\sigma^2}\right), \quad (6)$$

where d'_i is the point in the map which is closest to z'_i . As described above and in contrast to most existing localization approaches, our system does not employ a single grid map to estimate the pose of the robot. Given our graph-based structure, we need to determine a vertex \mathbf{x}_{base} whose map should be taken into account for evaluating $p(\mathbf{z} \mid \mathbf{x})$. We determine the base node \mathbf{x}_{base} as the pose-graph vertex that minimizes the distance to \mathbf{x} and furthermore guarantees that the current location of the robot was observed in the map. This visibility constraint is important to maximize the overlap between the map and the current observation. Without this constraint, the closest vertex might be outside a building while the robot is actually inside of it.

Another important design-related question for a robust localization in crowded urban environments is where to mount a laser scanner. Fig. 6 shows an exemplary scene in downtown Freiburg. The interesting aspect is the amount of measurements (shown as red points) that match to the map. Fig. 6a depicts the measurements of a laser mounted at a height of 0.55 m. In comparison, Fig. 6b visualizes a corresponding measurement of a laser mounted at a height of 1.55 m. The higher mounting position of the laser allows the robot to sense more of the static parts of the environment. This improves the localization since the robot is able to measure above small obstacles, e.g., chairs, shopping bags, bikes, and children.

A challenging problem is the pose initialization after starting the system also referred to as global localization. Since we have a GPS sensor on the robot and a GPS aligned map, we use GPS measurements as initial guess to localize the robot. In that way, we avoid to distribute a massive amount of particles on a map that covers several kilometers and save computational resources. Furthermore, we exploit the compass of the Inertial Measurement Unit (IMU) to constrain the heading of the particles. Based on the GPS and IMU measurements, we sample a pose for each particle from a Gaussian distribution around the initial guess. As a standard deviation we apply 5 m for the position and 10° for the heading. In case the automatic pose initialization fails because there is no GPS signal, e.g., if the robot is indoors, the user has the possibility to provide an initial guess manually.

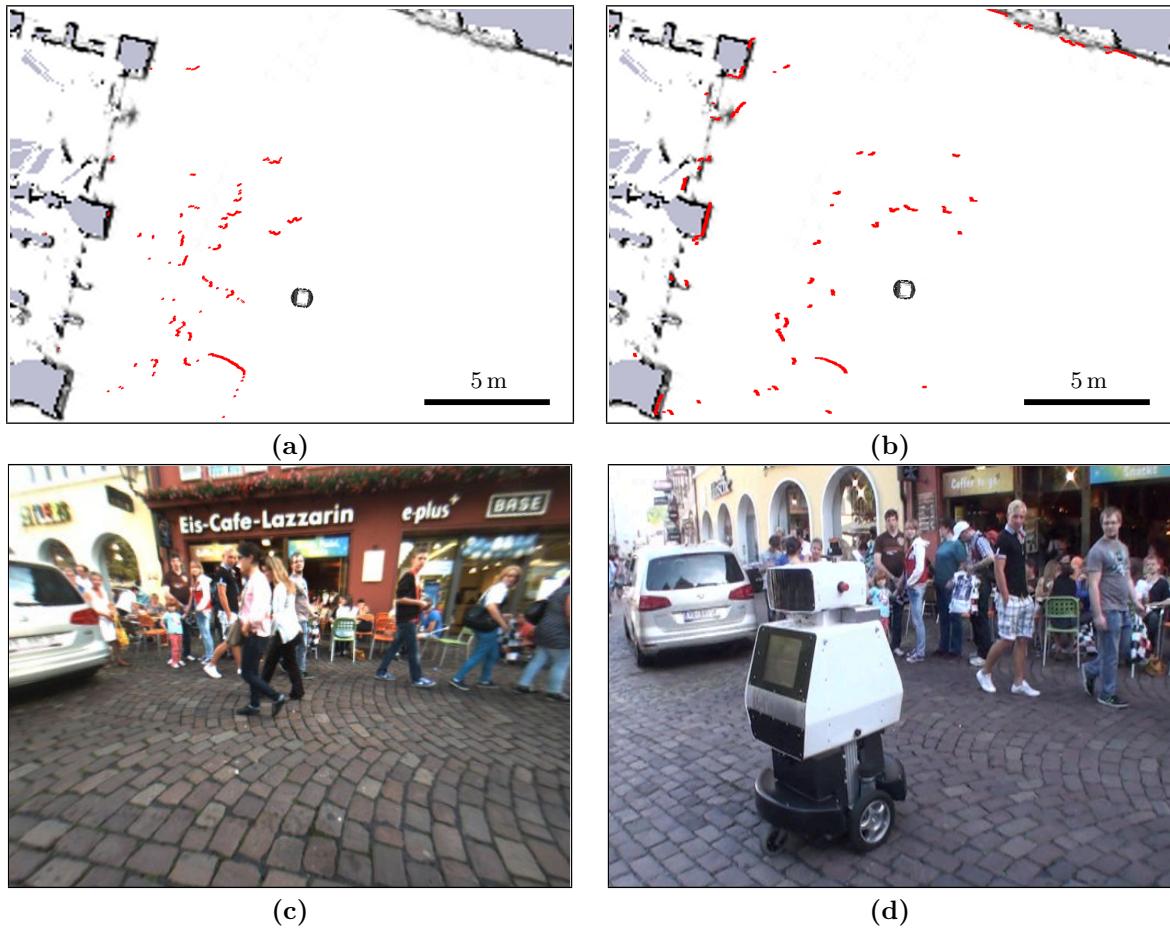


Figure 6: Example of a crowded scene in downtown Freiburg. (a) An observation of a laser mounted at a height of 0.55 m. In comparison (b) shows a corresponding observation of a laser acquired mounted at a height of 1.55 m. The image in (c) shows the scene from the viewpoint of the robot and (d) from behind the robot. The additional height of the head laser allows the robot to sense more of the static parts of the environment.

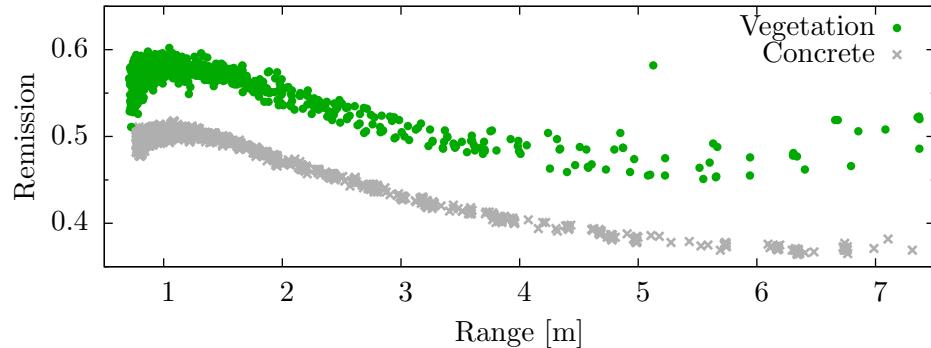


Figure 7: Range and remission data collected by the robot observing either a concrete surface or vegetation.

4.5 Traversability Analysis

The correct identification of obstacles is a critical component for autonomous navigation with a robot. Given our robotic platform, we need to identify obstacles having a height just above 3 cm. Such obstacles are commonly described as positive obstacles, as they stick out of the ground surface the robot travels upon. In contrast to that, negative obstacles are dips above the maximum traversable height of 3 cm and such obstacles should also be avoided by the robot. In the following, we describe the module which detects positive and negative obstacles while at the same time allowing the robot to drive over manhole covers and grids which might be falsely classified as negative obstacles. Furthermore, while navigating in urban areas the robot may encounter other undesirable surfaces, such as lawn. Here, considering only the range data is not sufficient, as the surface appears to be smooth and drivable. Since our platform cannot safely traverse grass areas, where it might easily get stuck due to the small caster wheels, we also have to identify such areas to allow the robot to avoid them and thus to reduce the risk of getting stuck while trying to reach a desired location.

4.5.1 Vegetation Detection

In our implementation, we detect flat vegetation, such as grass, which cannot be reliably identified using only range measurements, by considering the remission values returned by the laser scanner along with the range (Wurm et al., 2009, 2012). We exploit the fact that living plants show a different characteristic with respect to the reflected intensity than the concrete surface found on streets.

In contrast to Wurm et al. (2009), we detect vegetation with a fixed downward looking laser instead of a tilting laser. This results in an easier classification problem, as the range of a beam hitting the presumably flat ground surface correlates with the incidence angle. Fig. 7 visualizes the data obtained with our platform. As can be seen from the image, the two classes can be separated by a non-linear function. We choose to fit a function to the lower boundary of the vegetation measurements which allows us to identify measurements which are likely to be vegetation with a high efficiency. The resulting classification accuracy is slightly worse compared to the original approach but faster and, as can be seen in Fig. 9, still sufficient for identifying regions covered by vegetation that should be avoided by the robot. This classification allows us to automatically annotate region in the map that should not be traversed by the robot due to the potential risk of getting stuck. An example for such a map along with an aerial image can be seen in Fig. 8. As can be seen from the image, the classification is accurate and allows the robot to avoid the potentially dangerous areas.

4.5.2 Tracking Dynamic Obstacles

To detect moving obstacles in the vicinity of the robot, like pedestrians or bicyclists, we employ a blob tracker based on the 2D range scanner data and a common coordinate frame defined by the wheel odometry. This tracker works by first accumulating the 2D laser readings into a 2D grid map for a short time window (about 100 ms in our current implementation). In addition to that, we keep a history of these maps for a larger amount of time (about 1 s). To find the dynamic elements in the map, we compare the current map with the oldest in the history and mark the obstacles that only appear in the newer map as dynamic.

While the robot is moving, new areas become visible that were occluded before. This can lead to false positives in the detection of dynamic parts in the environment. Hence, we perform an additional step to filter these out by performing a check on every candidate dynamic obstacle if it was occluded in an earlier time step. To this end, we inspect the history of preceding scans and analyze the line of sight to the obstacle in question. If an obstacle was present that blocked the line of sight, we do not consider this part of the environment to be dynamic.

In the next step, we cluster the dynamic obstacles into blobs using a region growing approach. Then, we find corresponding blobs in the preceding map using a nearest neighbor approach (rejecting neighbors above

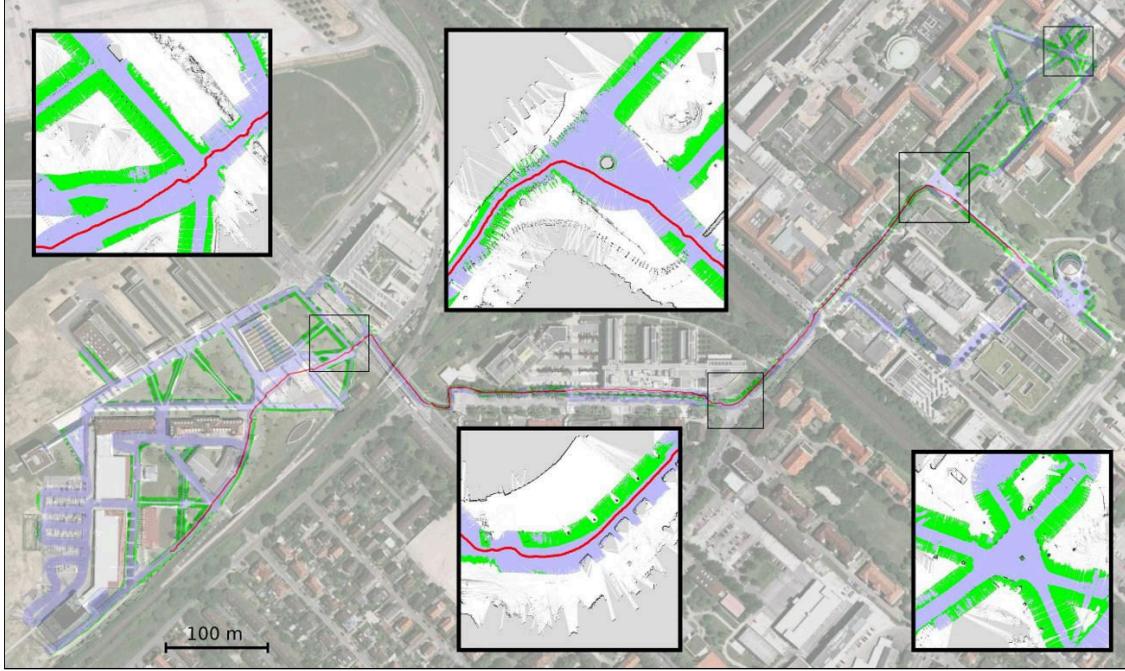


Figure 8: A map of the environment in which areas covered with vegetation are annotated. Areas that have been classified as vegetation are drawn in green, while concrete is drawn in blue. The trajectory drawn in red corresponds to a path that the robot travelled autonomously within one of the field tests.

a predefined distance). Based on the mean positions of the corresponding blobs, we estimate velocities and bounding boxes that are aligned to the movement direction.

While this method is relatively simple (and occasionally creates false positives and sometimes wrongly merges multiple moving objects into one), it proved to be highly effective for the city navigation task. It can be calculated in a highly efficient manner and provides a sufficient movement prediction for avoidance purposes, as can be seen in Fig. 9.

4.5.3 Detection of 3D Obstacles

Unfortunately, not all obstacles that might block the robot’s path are visible in the horizontal laser scans. For this reason, we implemented a module that analyzes the scan lines captured by the downwards facing laser and the mirrored laser beams in front of the robot (see Section 3). These lasers provide 3D information about the environment when the robot is moving.

In a first step, we perform a filtering on the raw scans to get rid of some false measurements. This especially targets at spurious points typically returned at the borders of objects in the form of interpolated point positions between the foreground and the background. These points might create false obstacles. To detect them, we check for sudden changes in depth which are noticeable as very small viewing angles from one point in the 2D scan to its immediate neighbors. Those border areas are erased from the scans before performing the obstacle detection procedure.

The main part of the obstacle detection process is done by analyzing only single scan lines, instead of the point cloud which is accumulated during driving. A similar approach was considered by Pfaff et al. (2007) for navigating with a car in cities. To decide whether points in these scan lines measure flat ground or an obstacle the robot cannot traverse, we analyze how straight the scan lines are and if there are significant discontinuities in there, since a flat ground would lead to straight scan lines. To be robust to noise, we

use polynomial approximations for small segments of the scan lines and analyze their incline. Every point that lies in a segment which has an incline above a maximum value (10°) and a height difference above a maximum allowed step height (3 cm) is reported as a potential obstacle. Note that these parameters arise from the capabilities of the platform.

This heuristic proved to be effective and has the advantage of being efficient for single 2D scans, without the need of integration over a longer period of time. It also does not require information about position changes of the robot, which would be a source of considerable noise. In addition to that, there are no strong requirements regarding the external calibration parameters of the lasers.

Unfortunately, there are rare cases where this procedure fails. The main source of failures are false positives on manhole covers or gutters in the ground. Example images can be seen in Fig. 10 (top). Since some laser beams go through the structure and some not, they appear to be negative obstacles. We implemented a heuristic to detect those cases by identifying areas with small holes. For this purpose, we extended the method described above and build an elevation map from the accumulated scan points while driving. An elevation map (Bares et al., 1989; Hebert et al., 1989) is a 2D grid map, where every cell contains the average z-value of the points falling into it. For every scan point, we check if it lies substantially below the estimated elevation of the corresponding neighboring cells, both in x- and y-direction. If this is the case, this indicates a small hole, since most of the measurements in the area were on ground level but a few went through the hole and lead to much deeper measurements. Obstacles close to such holes are ignored, if they are below a certain height (10 cm). This approach proved to provide the desired reliability for different settings in which the naive approach would have reported non-traversable negative obstacles (see Fig. 10, bottom image, for an example).

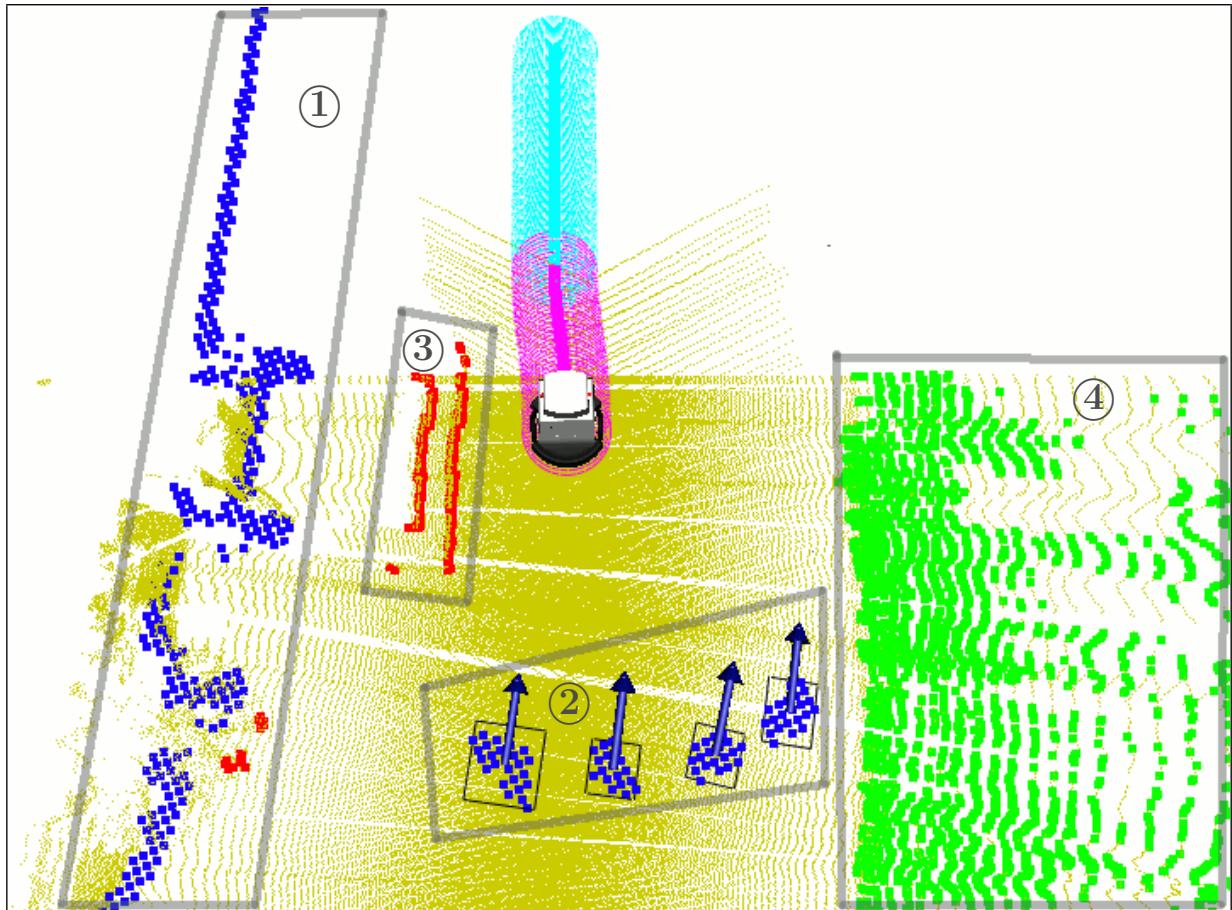
For every positive obstacle detected by the approach above, we check if this obstacle also has a corresponding obstacle in its vicinity in the 2D scans from the horizontal lasers. If not, the corresponding 3D position is reported as a 3D obstacle. If yes, it is considered to belong to the 2D obstacle and only stored for a short amount of time. The reason for this is that our sensor setup does typically not allow us to reobserve a 3D obstacle in regular intervals, since it is just seen once while driving by. Therefore, we have to keep the 3D obstacles in the map for an indefinite amount of time. In contrast, obstacles observed in the 2D scanners can be reobserved and therefore do not have to be kept for a long time. This procedure prevents most dynamic objects (those that are also visible in 2D) from trapping the robot because it does not notice their disappearance. An example regarding the different obstacle types can be seen in Fig. 9.

4.5.4 Vibration-Based Ground Evaluation

While the approach described above allows the robot to identify objects that need to be avoided, the ground surface itself needs to be taken into account while driving autonomously. Cobble stone pavement, which can typically be found in the centers of old European cities, leads to a substantial vibration and shaking of the platform. Hence, we consider the measurements provided by the IMU to control the maximum velocity of the platform based on the current vibration. We found that the values provided by the gyroscopes of the IMU give the most reliable measure for the strength of the vibration, by indicating how much the robot is shaking back and forth while driving. The accelerometer signal on the other hand proved to be much noisier and less reliable. For this purpose we calculate the mean angular velocities from the gyros for the x- and y-axis (corresponding to changes in roll and pitch) over a small amount of time (50 ms). If one of these values exceeds a certain limit, the maximum allowed velocity of the platform is gradually decreased. As the accuracy of the laser sensors is not sufficient to classify the smoothness of the surface, the robot has no means to identify whether the surface allows driving fast again without inducing vibrations. Hence, we greedily increase the maximum velocity again after a short delay and repeat the entire process. See Fig. 11 for an example.



(a)



(b)

Figure 9: (a): This photo shows an example scene. (b): This image visualizes the different kinds of detected obstacles in the scene shown in (a), whereas four people are walking behind the robot. Blue points mark obstacles that are visible in the horizontal 2D laser scanners (areas ① and ②). Red points mark 3D obstacles that are visible in the downwards facing laser beams, but not in the 2D laser beams (mainly area ③). Green points mark the detected vegetation/grass (area ④). The black boxes with the arrows mark detected dynamic obstacles (area ②). The remaining yellow dots visualize the accumulated point cloud from the laser measurements.

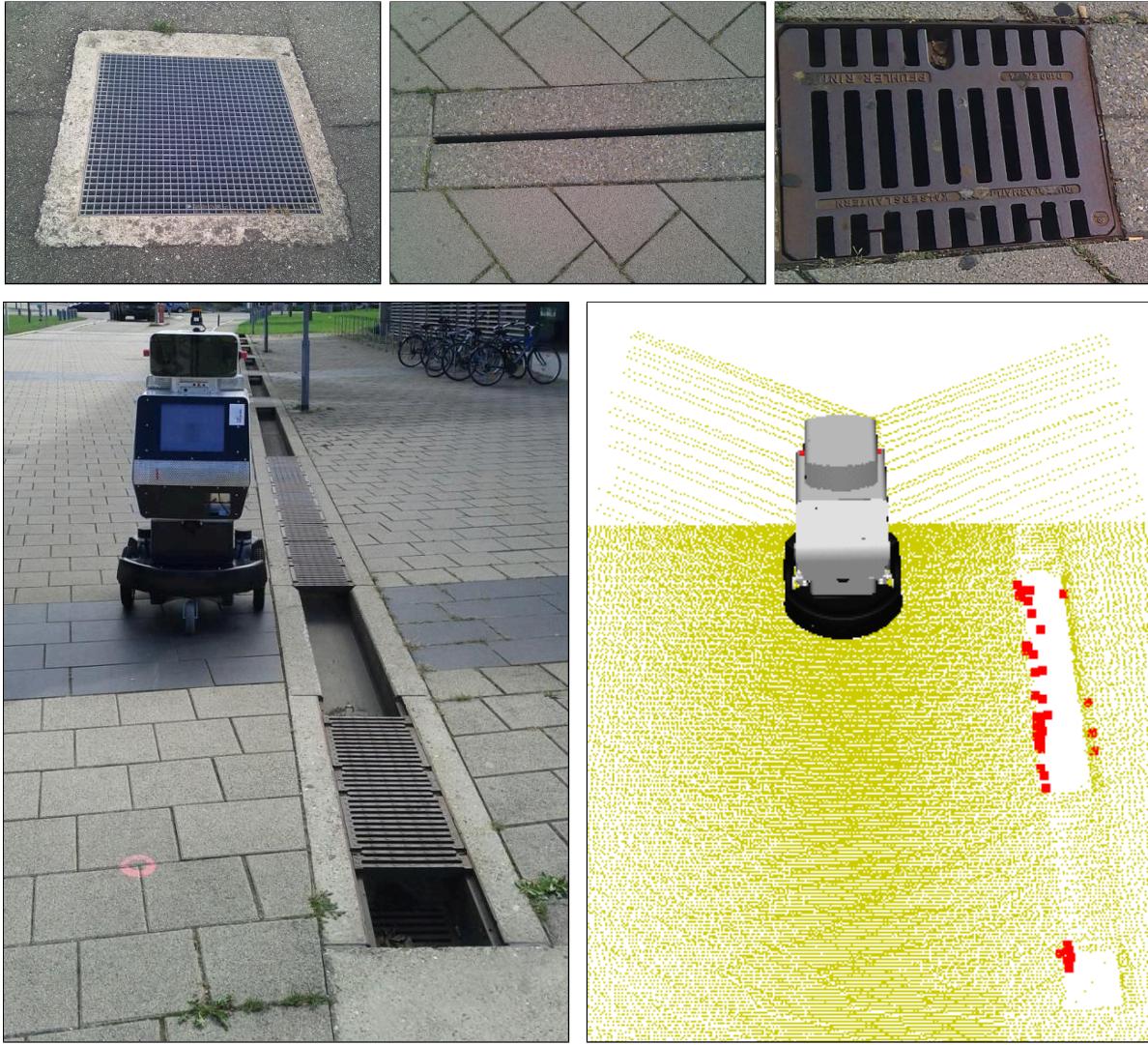


Figure 10: Top: Traversable structures that might be detected as negative obstacles by a naive method, because some laser beams can go through them. Bottom: Example case for the obstacle detection module. While the small canals on the robot's right side are classified as negative obstacles, the gutters are identified as traversable even though there are laser measurements going through the holes.

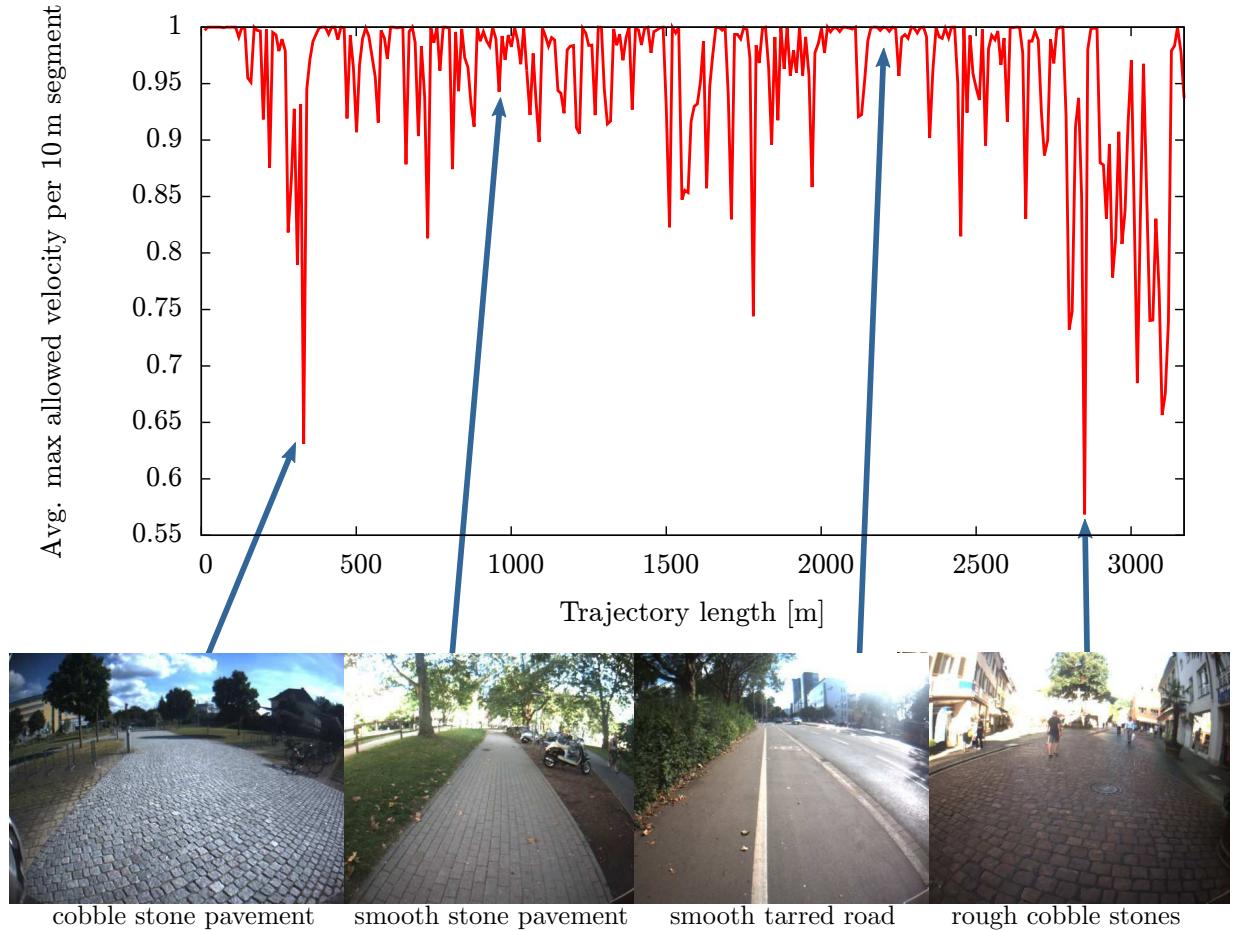


Figure 11: The plot at the top of this figure shows the maximum allowed velocity of the robot during a 3.2 km long autonomous run. This maximum velocity is determined based on the vibration measured while driving and therefore depends mostly on the ground surface. The pictures at the bottom show example images captured at the corresponding places of the trajectory. The left-most and right-most images are examples of rough surfaces that lead to a reduced maximum velocity, whereas the two middle images show examples of smooth surfaces that allow a high maximum velocity.

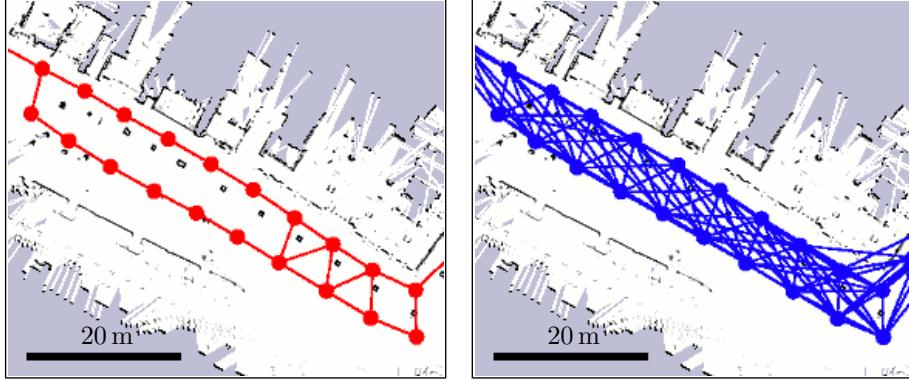


Figure 12: Left: Partial view of the pose-graph with its constraints used for estimating the poses. Right: The same view of the topology graph generated by the planner shows that this graph typically features a denser connectivity.

4.6 Planner

Our planner considers different levels of abstraction to compute a feasible path for the robot towards a goal location. The architecture consists of three levels. The highest level only considers the topology of the environment, i.e., the graph connecting the local maps. The intermediate level employs Dijkstra’s algorithm on the local maps to calculate way-points which serve as input for the low-level planner developed by Rufli et al. (2009). This low-level planner actually computes the velocity commands sent to the robot and deals with the dynamic objects that are present in the vicinity of the robot. The latter uses the information from the blob tracker (see Section 4.5.2) and projects this movement into the future to predict where the obstacle will most probably be at a certain time in the planned trajectory. Note that by using this hierarchy of planners, we loose the optimality of the computed paths. However, as reported by Konolige et al. (2011), who developed a similar planning framework, the resulting paths are only approximately 10 % longer while the time needed to obtain them can actually be several orders of magnitude shorter.

Given the pose estimates of the SLAM module, our planner constructs a topology \mathcal{T} represented by a graph. This graph is constructed as follows: Each node \mathbf{x}_i of the graph is labeled with its absolute coordinates in the world. Furthermore, each node comes with its local traversability map describing the drivable environment in the neighborhood of \mathbf{x}_i which serves as the background information for the planner. Additionally, each cell in the map encodes the cost of driving from \mathbf{x}_i to the cell. In our implementation we assign the cost for traversing a cell as a weighted sum of the travelled distance and free space around the cell. The idea is to penalize driving close to obstacles. This cost can be pre-computed for each cell of the map efficiently by a single execution of Dijkstra’s algorithm starting from \mathbf{x}_i . We refer to this as the reachability information of the map.

Two nodes are connected by an edge if there is a path from one node to the other given the information stored in their local maps. The edge is labeled with the cost for traversing the path which is determined by planning on the local maps. If such a path cannot be found, we assign a cost of infinity to this edge. Otherwise, we assign to the edge the cost returned by the intermediate level planner which is typically proportional to the length of the path. Yet, in contrast to the straight-line distance, the cost better reflects the local characteristics of the environment. By this procedure, which is carried out once as a pre-processing step, the planner will consider the real costs for the robot to traverse the edge instead of only considering the Euclidean distance. Note that the set of edges contained in the topology graph \mathcal{T} in general differs from the set of constraints \mathcal{C} generated by the SLAM module (see Eq. (4)). The topology graph exhibits a denser connectivity as can be seen in Fig. 12 because there exist more paths from one map to the other than links found by the place recognition part of the mapping algorithm.

While driving autonomously, the robot may encounter unforeseen obstacles, e.g., a passage might be blocked by a construction site or parked cars. Our planner handles such situations by identifying the edges in the topology which are not traversable in the current situation. Those edges are temporarily marked with infinite costs which allows the hierarchical planner framework to determine another path to the goal location.

Planning a path from the current location of the robot towards a desired goal location works as follows. First, we need to identify the nodes or maps in \mathcal{T} which belong to the current position of the robot and the goal. To this end, we refer to the reachability information of the maps. We select the maps with the shortest path from the center of the map to the robot and the goal, respectively. Given the robot node and the goal node, the high level planner carries out an A* search on \mathcal{T} . Since the cost for traversing an edge corresponds to the real cost of the robot to traverse the edge, this search provides a fast approximation of an A* on the complete grid map but is orders of magnitude faster. The result is a list of way-points towards the goal. However, following this list closely may lead to sub-optimal paths. Hence, we perform the Dijkstra algorithm in the local map starting from the current location of the robot and select as intermediate goal for the low-level planner the farthest way-point that is still reachable. Note that the local map containing the current position of the robot is augmented online with the static obstacles found by the obstacle detection.

4.7 Watchdog

For a real-world system it proved to be useful to have a module that monitors the multitude of the other modules for errors and can restart them if something goes wrong. For this purposes, all our modules perform frequent consistency checks. For example, the hardware drivers check if they still receive data from the sensors as well as the platform and broadcasts status messages regularly. The watchdog module receives regular updates (typically once a second) from every module. If a module reports a failure or if its alive-messages time out, the watchdog module kills the process and restarts it. Modules that rely on a history, like the localization, store their current state regularly in a permanent manner, so that they can directly resume their work from a recent state when being restarted. Should the watchdog not be able to fix a problem by restarting a module, meaning it repeatedly fails after the restart, the watchdog will report the failure and, if needed, can enforce an emergency stop. The watchdog communicates with the operator over a graphical user interface, whose information is also available remotely, and plays sounds when problems appear.

This module does not have an explicit scientific background but serves as a handy solution for robots in the real world. It would be unnecessary in a system where every module and each of its third party dependencies work perfectly. But in practice, the watchdog module proved to be essential for long-term autonomous navigation, especially regarding the robust operation of certain hardware components. Since we logged whenever a module was reseted during our tests, we were able to carry out long-term experiments and to analyze the log-files afterwards to identify the faulty software parts.

5 Evaluation

In this section, we describe a set of experiments in which we evaluated the system described in this paper. The map used to carry out the experiments was obtained by driving the robot along a 7.4 km long trajectory. The map covers the area between the Faculty of Engineering of the University of Freiburg and the city center of Freiburg. Using this map, we carried out a series of experiments. Among several smaller tests, we performed six extensive navigation experiments during which we let the robot navigate from our campus to the city center and back. Figure 13 depicts an example along with images taken along the trajectory. In these experiments, the robot traveled an overall distance of around 20 km and for three times required manual intervention. In addition to a localization failure discussed below in more detail, the robot once got stuck in front of a little bump and one further time was manually stopped because of an obstacle that we believed not being perceivable by the robot.



Figure 13: This figure shows the driven trajectory of our robot on an aerial image (aerial image © Google). On the side are photos of exemplary locations on the trajectory, taken at the places indicated by the arrows.

Note that the final experiment was announced widely to give the public and the press the opportunity to see whether state-of-the-art robotics navigation technology can lead to a mobile robot that can navigate autonomously through an urban environment. The event itself attracted journalists from both TV and newspapers and lead to a nationwide and international coverage in top-media. Video material for the event itself and one of the test runs can be found on the web¹ and is also available as supplementary material to this paper. During all of our experiments at least one human operator accompanied the robot. The task of the operator was to control the robot and to stop the robot by triggering the wireless emergency stop in case of a dangerous situation. Additionally, the robot was not allowed to cross streets without asking for permission. To this end, the operator had to provide the information that it is safe to cross the street by pressing a button on the screen of the robot.

5.1 Localization

Whenever a robot navigates within an urban environment, the measurements obtained by the sensors of the robot are affected by the people surrounding the robot. As the localization algorithm is one of the core components of our system, we analyzed the occlusions in the range data caused by people partially blocking the view of the robot.

Fig. 14 depicts the fraction of valid range readings, i.e., readings smaller than the maximum range of the laser scanner, and the number of beams that match to the map for one of the large experiments mentioned above. Here, we regard a range reading as matching to the map if the distance between the measurement and the closest point in the map is below 0.2 m. The plot depicts several interesting aspects. A small fraction of valid beams indicates that the robot is navigating within open regions where only a small amount of structure is available to the robot for localizing itself. This can be seen in the examples that correspond to 4 min and

¹<http://europa.informatik.uni-freiburg.de/videosJFR.html>

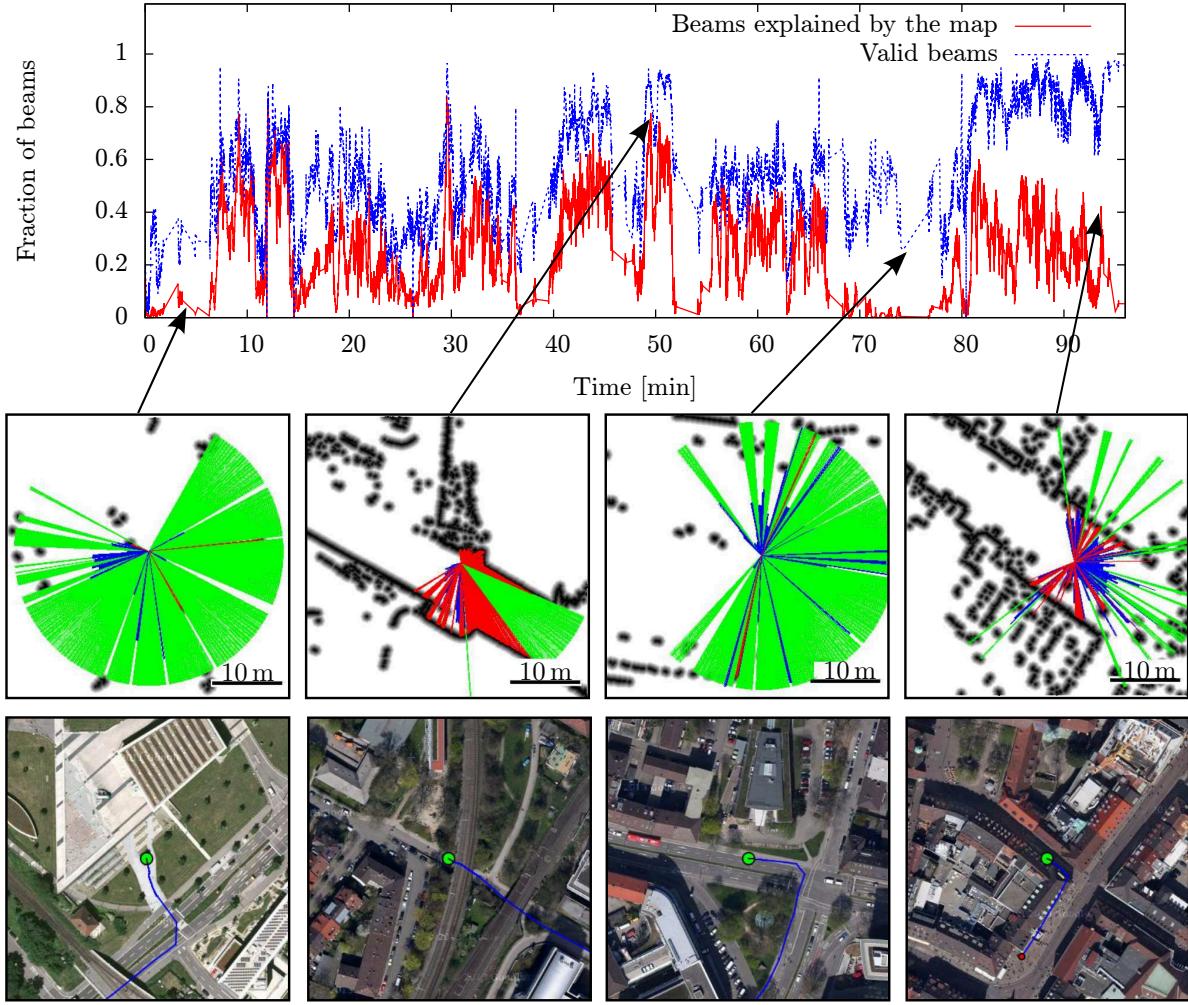


Figure 14: This plot shows the fraction of valid beams returned by the range scanner and the fraction of beams that can be explained by the map of the environment. We regard a beam as valid if it reports a range reading shorter than the maximum range of the scanner. We classified a range reading as being explained by the map if the distance between the end point of the range reading and the closest occupied cell in the map is below 0.2 m. Furthermore, we show four different examples along the trajectory with colored beams on a distance map. Green beams correspond to maximum range readings. For a better visibility we cropped the beams after 19 m. Blue beams show valid range measurements that are not well explained by the map and red beams indicate measurements that can be explained by the map. The first example shows the robot in an open space where only a few trees are perceptible by the robot. In the second example the laser scan is well explained. The third example shows a situation a few minutes before the localization failure. Only a small fraction of the beams is explained by the map due to the open terrain and the presence of humans around the robot. The remaining observable structure is mainly parallel to the driving direction and does not constrain the position estimate in that direction. This lead to the localization failure after around 78 minutes. The last example shows a densely crowded scene in the Freiburg downtown area. Due to the height of the laser, major parts of the surrounding buildings are visible and enable our robot to successfully localize itself (aerial image © Google).

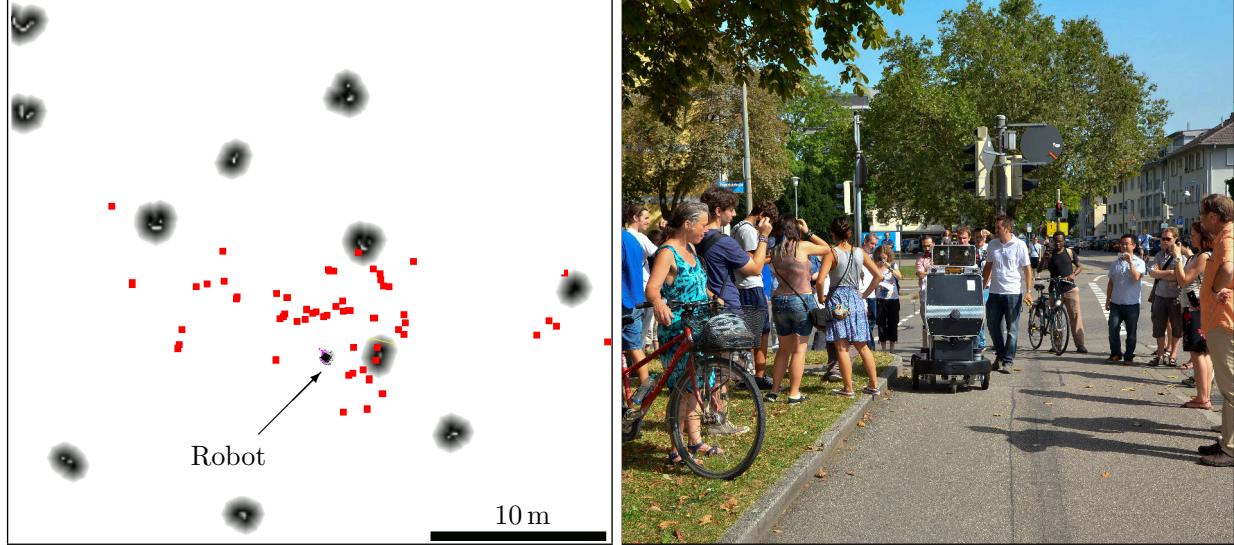


Figure 15: Background information for the localization failure during an autonomous run to Freiburg downtown. The 2D distance map is shown on the left. As can be seen, there are only few localization features (shown in black) around — mostly tree trunks — and nearly all laser observations (red points) mismatch the provided model. The picture on the right shows that the robot is almost completely surrounded by people.

75 min where most of the laser scans reports maximum range (green). Furthermore, the difference between the number of valid beams and the number of beams that match to the map indicates that the view of the robot was partially blocked. In open areas with only few landmarks it becomes critical if most of the visible landmarks are occluded for a longer time. In contrast, the example around 50 min shows a situation where most of the laser scan is well explained and only few pedestrians are around the robot. The last example shows a densely crowded scene in the Freiburg downtown area. Due to the height of the laser, major parts of the surrounding buildings are visible and enable our robot to successfully localize itself.

In this experiment, the autonomous run was interrupted twice. In the first incident, the robot’s wireless emergency stop button was pressed unintentionally, thereby being a human mistake. In the second case, a localization error occurred after around 78 min. As can be seen in Fig. 14 between minutes 70 and 78 the robot traveled 200 m in an area with a very small amount of features while being surrounded by many people, as depicted in Fig. 15. This mixture of very few relevant features in the map (shown on the left hand side in Fig. 15) and the fact that the robot was driving for an extensive distance while receiving mostly spurious measurements lead to an error in the position estimate of around 2 m. This caused problems in negotiating a sidewalk after crossing the street. It made the robot stop and required us to re-localize the robot.

In other instances, sharing the same characteristics, for example, around minute 37 and around minute 52 the robot only drives substantially lower distances of 100 m and 50 m without meaningful sensor measurements. In both situations, the system is able to overcome the problem because it receives relevant information early enough again.

We also analyzed a similar trajectory of the robot carried out during night time (see Fig. 16). At night, typically a smaller number of people are around and less occlusions happen to the measurements. Hence, the offset between the number of valid beams and the number of beams matching to the map is small all the time. In this experiment, the robot successfully reached its goal location without any problems and along a slightly different path of 3.5 km length. Visual inspection of the trajectory on the map revealed that the localization pose is much more accurate than the GPS solutions we had available. The trajectory was estimated correctly throughout the whole experiment.

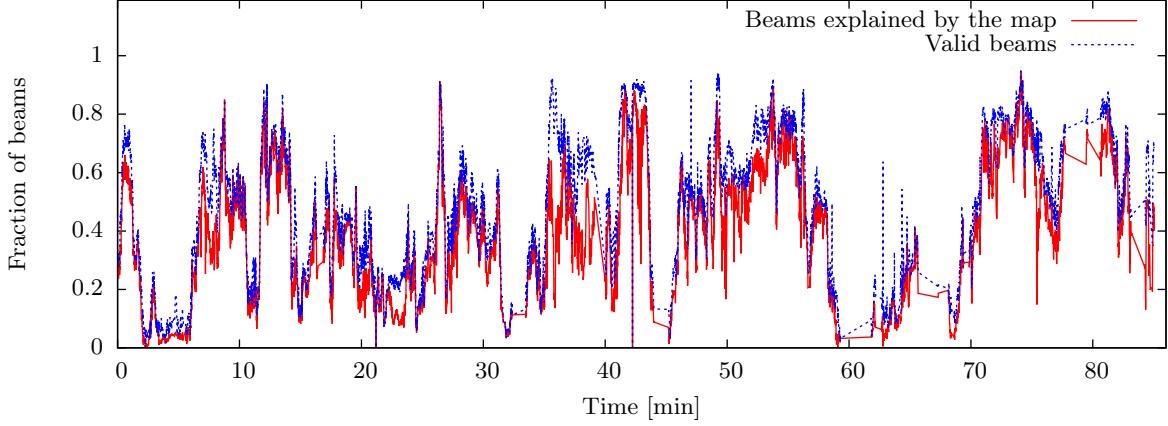


Figure 16: In this figure we present a similar plot to Fig. 14 for a dataset acquired during an autonomous run in the night. The environment is almost static and therefore most valid beams are explained by the model. The robot traversed the same region as shown in the previous Figure around minute 60.



Figure 17: This figure shows an exemplary marking on the ground on an open space close to building 101 on our campus (left) and the robot standing on the marked position (right). We were able to manually drive the robot on the marking with a precision of approximately 1.5 cm.

The global accuracy of the localization module is difficult to evaluate since city scale ground truth positions for a mobile robot are hard to acquire. Note that GPS in general is not accurate enough in urban canyons and without additional ground stations. Therefore, we decided to evaluate the localization accuracy locally with respect to two specific real-world positions. While the first position is nearby building 079 on our campus where the robot can observe a large portion of the building around, the second position on the open area in front of building 101 result in only a small portion of usable range measurements. We drove the robot ten times onto each marked position and evaluated the localization estimates. Figure 17 shows an exemplary marking on the ground and the robot standing on the marked position. After each visit of the goal location we drove at least 15 m before we approached the location for another pass.

Using the recorded estimates we computed the mean position and the corresponding standard deviation for the two locations on our campus. The standard deviation is an indicator for the localization accuracy but also suffers from small position errors since we cannot drive manually to a target location with millimeter accuracy. Figure 18 shows the collected localization estimates for both locations along with the 3σ confidence ellipse in blue and the robot footprint in green.

For the location next to building 079 the translational standard deviation was 0.035 m with a rotational standard deviation of 0.82° . For the other location near building 101 the standard deviations were 0.027 m

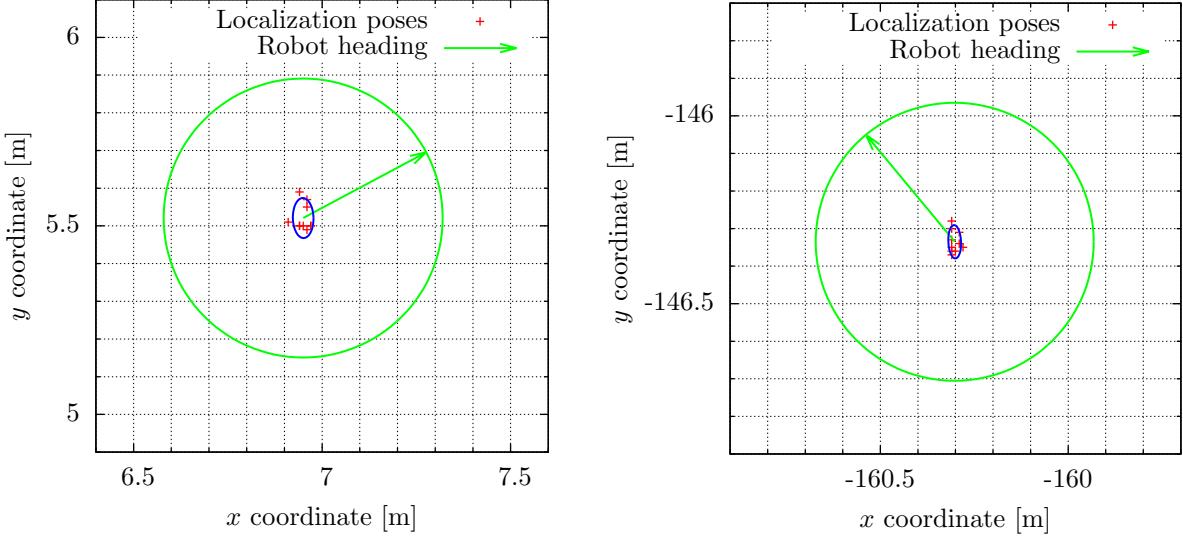


Figure 18: In this figure, we show the localization estimates for several stops at the same location for two different places on our campus. The left plot shows the localization estimates for the reference pose near building 079 and the right image illustrates the corresponding plot for the reference pose near building 101. The green circles show the approximate footprint of the robot and the blue ellipses show the 3σ confidence ellipse around the mean position. The plotted grid has a cell size of 10 cm.

for the translational error and 0.87° for rotational error, respectively. The resulting localization accuracy is more than sufficient to carry out the described navigation experiments. Some of the paths the robot regularly took during our experiments were only 20 cm wider than the width of the robot, thereby enforcing a minimum accuracy for the localization pose of 10 cm.

5.2 Global Localization

In this section, we evaluate the success rate of our GPS-based MCL initialization scheme. Therefore, we recorded one dataset in which we had two different GPS devices mounted on the robot. In addition to the Trimble GPS Pathfinder Pro GPS device we mounted a low cost consumer GPS, i.e., a standard SiRF III, as baseline. For both devices, we selected GPS measurements every 15 s on the dataset when available. For every location we initialized the MCL algorithm 10 times with the corresponding coordinates and let it run until either the filter converged or the time limit of 15 s was met. As ground truth, we considered the poses of the localization module running while acquiring the data and considered every pose estimate closer than 1 m as success. The chosen ground truth is not perfect but a small difference to the ground truth indicates that the particle filter converged close to the true position of the robot. In order to provide insights in possible failure cases, we also evaluated the GPS accuracy with respect to the ground truth and the number of beams that match to the map on the ground truth trajectory. Throughout all our experiments, we used 1,000 particles and a standard deviation of 5 m for both GPS devices.

Fig. 19 shows the resulting quantities for both GPS devices. As can be seen in the upper plot, the pose estimates of the low-cost GPS (green) can be up to 50 m away from the true pose. In comparison the second GPS device results in errors below 20 m. The localization success per time slot is plotted in red and the number of beams that match to the map is plotted in blue. The overall localization success was 53.3 % for the low-cost GPS and 83.9 % for the highly accurate GPS device. Of course the accuracy of the GPS measurements has a major impact on the localization success but also the amount of information available at the true pose in the map. As can be seen in the lower plot in Fig. 19, between minutes 5 and 10 the GPS measurement is around 1 m close to the true pose but the localization success is 0 %. This is mainly caused by the fact that there is not much laser based information available, as can be seen in the blue plot.

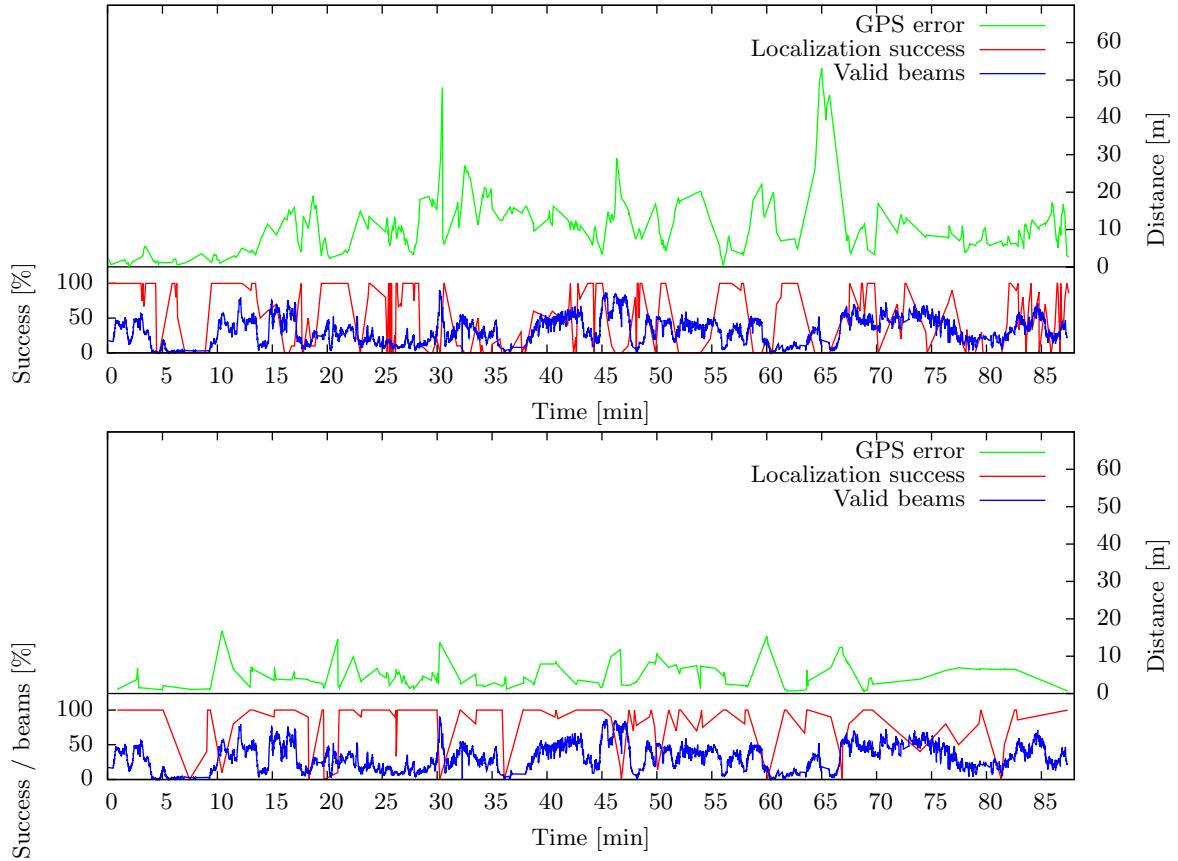


Figure 19: The upper plot shows the results for initializing the localization algorithm based on the low-cost GPS device. As can be seen, the pose estimates of the low-cost GPS (green) can be up to 50 m away from the true pose. The localization success per time slot is plotted in red and the number of beams that match to the map is plotted in blue. The lower plot shows the results for the highly accurate GPS device. The maximum GPS measurement error for this device is below 20 m.

Overall, a successful initialization for the localization algorithm highly depends on the accuracy of the GPS measurement as initial guess and how much information the laser provides in a region.

6 Discussion and Lessons Learned

The navigation system described in this paper has been implemented for and on the robot Obelix characterized in Section 3. The design of a platform typically has a substantial influence on the algorithms needed for accomplishing the desired task. Given the navigation task of Obelix, his structure definitely influenced the design of certain software components. For example, its almost circular footprint makes the planning of paths easier, as only a two-dimensional path needs to be computed (see Fig. 1). Additionally, the specific mounting of the range scanners, that resulted in the fact that three-dimensional structures could only be sensed when the robot moves, has an influence on collision avoidance routines. We are still convinced that these platform-specific design choices are not critical and that the mixture of components we realized is relevant for accomplishing this challenging navigation task and is sufficiently generic to be easily transferable to other robotic platforms, such as robotic wheelchairs or transportation vehicles in cities.

6.1 Lessons Learned

- Pedestrians, in particular children, in our experiments sometimes intentionally blocked the path of the robot. While this did not cause any collisions, the robot got stuck in such situations because it is not as agile as a human. According to our experience with such cases, a planning framework should actively negotiate a path with the humans around in highly crowded scenarios instead of solely focusing on collision free trajectories.
- Besides tools for automatic data logging and debugging, we found it handy to have the option to selectively de-activate various aspects of the autonomous system for testing and to easily change parameters on-the-fly. One typical example is the adjustment of the maximum allowed velocity during experiments. Our tools also allowed us to manually control the robot while inspecting the results of different aspects of the system, e.g., the tracking of dynamic objects and their impact on the planned path.
- In our system, the modules that have to fuse data over time operate in a local frame, which is given by the odometry of the robot. Clearly, this frame drifts over time but it is locally smooth, which cannot be guaranteed for position estimates in a global frame, such as the estimate of the localization algorithm. The usage of a local frame allowed us to estimate a consistent map that fuses multiple measurements (see also Moore et al. (2009)).
- Analyzing single range readings for obstacles and fusing these individually detected obstacles in a local frame proved to be very efficient. Furthermore, compared to analyzing point clouds, which are accumulated over time, it reduces the risk of detecting non-existing obstacles that may be caused by inaccurate calibration or imprecise position estimates of the robot.
- We found that specifying goal locations such as the stops of a guided tour as global coordinates (latitude/longitude) should be preferred compared to storing those location in local map coordinates. Since our mapping algorithm incorporates the GPS prior, we are able to project between map coordinates and the GPS frame. Without considering the GPS measurements, the coordinates in a map frame depend on the initial location of the robot. Consequently, the local map coordinates are subject to change while the GPS frame remains fixed.
- Throughout the project, we developed a vast set of visualization tools. These tools and the fact that we recorded all the data (sensor data and messages that got exchanged between the modules) allowed us to carefully analyze the behavior of the algorithms in practice.

- We furthermore realized that other aspects are rather challenging, as, for example, curly leafs on the ground look similar to little rocks in the range data. Whereas the robot can easily drive over leafs, rocks can actually have a substantial effect on the platform itself. This is a general dilemma which cannot be solved alone with perception algorithms. Therefore we decided to drive around all potential obstacles. In future systems this could be addressed by designing a robot platform that can measure forces on the bumpers and that is allowed to use a small force to overcome obstacles.
- In an early test of our system, we encountered a failure before the robot had to cross a traffic light. The robot was driving on a steep downward facing ramp leading from the sidewalk onto the street. This lead to laser beams hitting the street surface approximately 4m in front of the robot that were wrongly classified as an obstacle. As a consequence, we revised our system to incorporate the attitude of the robot to identify measurements that are likely to be caused by the laser beams observing the ground.

6.2 Limitations of our Approach

- The most critical aspect of the entire navigation task was the crossing of roads or all situations in which the robot potentially had to interact with fast-driving cars. The two dimensional laser sensors do not allow to reliably sense the cars approaching the robot in all situations. In our demonstration, we solved this problem by having the robot ask for permission to cross streets or other safety-relevant areas, which we marked manually in the robot's map.
- The obstacle detection operates on rigidly mounted laser range finders (see Fig. 2). While this has the advantage that no moving parts are involved, the rigid mounting of the sensors only allows the robot to observe objects such as curbs only at a range of approximately 1.5 m. When navigating in a previously unknown region, the robot has to frequently update its plan to move alongside a curb.
- Dynamic objects that are not visible in the horizontal range data, such as pets or other animals like pigeons or ducks (see Fig. 20) currently lead to a sub-optimal path execution, because they can only be observed in the close vicinity by the robot.
- The sensor setup has blind spots, for example, overhanging obstacles above a certain height in front of the robot are not visible. The only situation we encountered, where this was critical was a bike handlebar in a unfortunate position. Most likely an update of the platform should incorporate a revised sensor hardware.

7 Conclusions

In this paper, we presented a navigation system that enables a mobile robot to autonomously navigate through city centers. This navigation system uses an extended SLAM routine that deals with the outliers generated by the partially GPS-denied environments, a localization routine that utilizes a special data structure for large-scale maps, dedicated terrain analysis methods for also dealing with negative obstacles, and a trajectory planning system that considers dynamic objects. The navigation system has been implemented and demonstrated in a large-scale public field test, during which the robot Obelix autonomously navigated over a path of more than three kilometers through the crowded city center of Freiburg thereby negotiating with several substantial hazards. We believe that the individual techniques provided by the navigation system of Obelix are fundamental pre-requisites for mobile robots offering services to people in urban city centers including transportation, guidance, and surveillance.

Acknowledgments

This work has been supported by the European Commission under FP7-231888-EUROPA and FP7-610603-EUROPA2. We would like to thank all of our project partners for their valuable contributions and also the



Figure 20: Dynamic 3D obstacles which pose substantial challenges for the navigation system.

project officer Mariusz Bałdyga.

References

- Bachrach, A., Prentice, S., He, R., and Roy, N. (2011). RANGE - robust autonomous navigation in GPS-denied environments. *Journal of Field Robotics*, 28(5):644–666.
- Bares, J., Hebert, M., Kanade, T., Krotkov, E., Mitchell, T., Simmons, R., and Whittaker, W. R. L. (1989). Ambler: An autonomous rover for planetary exploration. *IEEE Computer Society Press*, 22(6):18–22.
- Bauer, A., Klasing, K., Lidoris, G., Mühlbauer, Q., Rohrmüller, F., Sosnowski, S., Xu, T., Khnlenz, K., Wollherr, D., and Buss, M. (2009). The autonomous city explorer: Towards natural human-robot interaction in urban environments. *International Journal of Social Robotics*, 1:127–140.
- Brookshire, J. and Teller, S. (2011). Automatic calibration of multiple coplanar sensors. In *Proc. of Robotics: Science and Systems (RSS)*.
- Burgard, W., Cremers, A. B., Fox, D., Hähnel, D., Lakemeyer, G., Schulz, D., Steiner, W., and Thrun, S. (1998). The interactive museum tour-guide robot. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*.
- Cremean, L. et al. (2006). Alice: An information-rich autonomous vehicle for high-speed desert navigation. *Journal of Field Robotics*.
- Dellaert, F., Fox, D., Burgard, W., and Thrun, S. (1998). Monte carlo localization for mobile robots. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Leuven, Belgium.
- Dellaert, F. and Kaess, M. (2006). Square Root SAM: Simultaneous localization and mapping via square root information smoothing. *Int. Journal of Robotics Research*, 25(12):1181–1204.
- Doucet, A., de Freitas, N., and Gordon, N., editors (2001). *Sequential Monte-Carlo Methods in Practice*. Springer Verlag.
- EUROPA (2009). The European robotic pedestrian assistant. <http://europa.informatik.uni-freiburg.de>.
- Fox, D., Burgard, W., and Thrun, S. (1998). Active markov localization for mobile robots. *Journal of Robotics & Autonomous Systems*, 25:195–207.
- Furgale, P. and Barfoot, T. D. (2010). Visual teach and repeat for long-range rover autonomy. *Journal of Field Robotics*, 27(5):534–560.

- Georgiev, A. and Allen, P. (2004). Localization methods for a mobile robot in urban environments. *IEEE Trans. on Robotics*, 20(5):851–864.
- Google Inc. (2012). Google self-driving car project. <http://googleblog.blogspot.com>.
- Grisetti, G., Kümmerle, R., Stachniss, C., and Burgard, W. (2010). A tutorial on graph-based SLAM. *Intelligent Transportation Systems Magazine, IEEE*, 2(4):31–43.
- Grisetti, G., Stachniss, C., and Burgard, W. (2005). Improving grid-based SLAM with rao-blackwellized particle filters by adaptive proposals and selective resampling. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*.
- Gross, H.-M., Boehme, H., Schroeter, C., Mueller, S., Koenig, A., Einhorn, E., Martin, C., Merten, M., and Bley, A. (2009). TOOMAS: Interactive shopping guide robots in everyday use - final implementation and experiences from long-term field trials. In *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*.
- Grzonka, S., Grisetti, G., and Burgard, W. (2012). A fully autonomous indoor quadrotor. *IEEE Trans. on Robotics*, 8(1):90–100.
- Hartley, R. I. and Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition.
- Hebert, M., Caillas, C., Krotkov, E., Kweon, I., and Kanade, T. (1989). Terrain mapping for a roving planetary explorer. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 997–1002.
- Konolige, K., Marder-Eppstein, E., and Marthi, B. (2011). Navigation in hybrid metric-topological maps. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*.
- Kümmerle, R., Grisetti, G., and Burgard, W. (2012). Simultaneous parameter calibration, localization, and mapping. *Advanced Robotics*, 26(17):2021–2041.
- Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K., and Burgard, W. (2011). g2o: A general framework for graph optimization. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*.
- Kümmerle, R., Ruhnke, M., Steder, B., Stachniss, C., and Burgard, W. (2013). A navigation system for robots operating in crowded urban environments. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*.
- Kümmerle, R., Steder, B., Dornhege, C., Kleiner, A., Grisetti, G., and Burgard, W. (2011). Large scale graph-based SLAM using aerial images as prior information. *Autonomous Robots*, 30(1):25–39.
- Liu, J. S. (1996). Metropolized independent sampling with comparisons to rejection sampling and importance sampling. *Statist. Comput.*, 6:113–119.
- Marder-Eppstein, E., Berger, E., Foote, T., Gerkey, B. P., and Konolige, K. (2010). The office marathon: Robust navigation in an indoor office environment. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*.
- Mirnig, N., Gonsior, B., Sosnowski, S., Landsiedel, C., Wollherr, D., Weiss, A., and Tscheiligi, M. (2012). Feedback guidelines for multimodal human-robot interaction: How should a robot give feedback when asking for directions? In *IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*.
- Montemerlo, M. et al. (2008). Junior: The stanford entry in the urban challenge. *Journal of Field Robotics*, 25(9):569–597.
- Moore, D., Huang, A. S., Walter, M., Olson, E., Fletcher, L., Leonard, J., and Teller, S. (2009). Simultaneous local and global state estimation for robotic navigation. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*.
- Morales, Y., Carballo, A., Takeuchi, E., Aburadani, A., and Tsubouchi, T. (2009). Autonomous robot navigation in outdoor pedestrian walkways. *Journal of Field Robotics*, 26(8):609–635.
- Murtra, A. C., Tur, J. M. M., and Sanfeliu, A. (2008). Action evaluation for mobile robot global localization in cooperative environments. *Journal of Robotics & Autonomous Systems*, 56(10):807 – 818.

- Newman, P., Sibley, G., Smith, M., Cummins, M., Harrison, A., Mei, C., Posner, I., Shade, R., Schroeter, D., Murphy, L., et al. (2009). Navigating, recognizing and describing urban spaces with vision and lasers. *Int. Journal of Robotics Research*, 28(11-12):1406–1433.
- Olson, E. (2008). *Robust and Efficient Robotic Mapping*. PhD thesis, MIT, Cambridge, MA, USA.
- Olson, E. (2009). Recognizing places using spectrally clustered local matches. *Robotics and Autonomous Systems*, 57(12):1157–1172.
- Pfaff, P., Triebel, R., Stachniss, C., Lamon, P., Burgard, W., and Siegwart, R. (2007). Towards mapping of cities. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Rome, Italy.
- Rauskolb, F. W., Berger, K., Lipski, C., Magnor, M., Cornelisen, K., Effertz, J., Form, T., Graefe, F., Ohl, S., Schumacher, W., et al. (2008). Caroline: An autonomously driving vehicle for urban environments. *Journal of Field Robotics*, 25(9):674–724.
- Reggente, M., Mondini, A., Ferri, G., Mazzolai, B., Manzi, A., Gabelletti, M., Dario, P., and Lilienthal, A. J. (2010). The dustbot system: Using mobile robots to monitor pollution in pedestrian area. *Chemical Engineering Transactions*, 23:273–278.
- Royer, E., Bom, J., Dhome, M., Thuilot, B., Lhuillier, M., and Marmoiton, F. (2005). Outdoor autonomous navigation using monocular vision. In *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*.
- Rufli, M., Ferguson, D., and Siegwart, R. (2009). Smooth path planning in constrained environments. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*.
- Sanfeliu, A., Andrade-Cetto, J., Barbosa, M., Bowden, R., Capitán, J., Corominas, A., Gilbert, A., Illingworth, J., Merino, L., Mirats, J. M., et al. (2010). Decentralized sensor fusion for ubiquitous networking robotics in urban areas. *Sensors*, 10(3):2274–2314.
- Schneider, F. and Wildermuth, D. (2011). Results of the european land robot trial and their usability for benchmarking outdoor robot systems. *Towards Autonomous Robotic Systems*, pages 408–409.
- Seetharaman, G., Lakhota, A., and Blasch, E. (2006). Unmanned vehicles come of age: The darpa grand challenge. *Computer*, 39(12):26–29.
- Siegwart, R. et al. (2003). RoboX at Expo.02: A large-scale installation of personal robots. *Journal of Robotics & Autonomous Systems*, 42(3-4).
- Smith, R., Self, M., and Cheeseman, P. (1990). Estimating uncertain spatial realtionships in robotics. In Cox, I. and Wilfong, G., editors, *Autonomous Robot Vehicles*, pages 167–193. Springer Verlag.
- Thrun, S., Bennewitz, M., Burgard, W., Cremers, A., Dellaert, F., Fox, D., Hähnel, D., Rosenberg, C., Roy, N., Schulte, J., and Schulz, D. (1999). MINERVA: A second generation mobile tour-guide robot. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*.
- Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic Robotics*. MIT Press.
- Thrun, S. et al. (2006). Winning the darpa grand challenge. *Journal of Field Robotics*.
- Trahanias, P., Burgard, W., Argyros, A., Hähnel, D., Baltzakis, H., Pfaff, P., and Stachniss, C. (2005). TOURBOT and WebFAIR: Web-operated mobile robots for tele-presence in populated exhibitions. *IEEE Robotics & Automation Magazine*, 12(2):77–89.
- Trulls, E., Corominas Murtra, A., Pérez-Ibarz, J., Ferrer, G., Vasquez, D., Mirats-Tur, J. M., and Sanfeliu, A. (2011). Autonomous navigation for mobile service robots in urban pedestrian environments. *Journal of Field Robotics*, 28(3):329–354.
- Urmson, C. (2005). *Navigation Regimes for Off-Road Autonomy*. PhD thesis, Robotics Institute, Carnegie Mellon University.
- Urmson, C. et al. (2008). Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466.
- Wurm, K. M., Kretzschmar, H., Kümmeler, R., Stachniss, C., and Burgard, W. (2012). Identifying vegetation from laser data in structured outdoor environments. *Robotics and Autonomous Systems*. In Press.

Wurm, K. M., Kümmerle, R., Stachniss, C., and Burgard, W. (2009). Improving robot navigation in structured outdoor environments by identifying vegetation from laser data. In *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*.