A Navigation System for Robots Operating in Crowded Urban Environments

Rainer Kümmerle Michael Ruhnke Bastian Steder Cyrill Stachniss Wolfram Burgard

***Abstract*— Over the past years, there has been a tremendous progress in the area of robot navigation. Most of the systems developed thus far, however, are restricted to indoor scenarios, non-urban outdoor environments, or road usage with cars. Urban areas introduce numerous challenges to autonomous mobile robots as they are highly complex and in addition to that dynamic. In this paper, we present a navigation system for pedestrian-like autonomous navigation with mobile robots in city environments. We describe different components including a SLAM system for dealing with huge maps of city centers, a planning approach for inferring feasible paths taking also into account the traversability and type of terrain, and a method for accurate localization in dynamic environments. The navigation system has been implemented and tested in several large-scale field tests in which the robot Obelix managed to autonomously navigate from our university campus over a 3.3 km long route to the city center of Freiburg.**

I. Introduction

Navigation is a central capability of mobile robots and substantial progress has been made in the area of autonomous navigation over the past years. 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 pedestrian zones, for example, the autonomous city explorer [1]. 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 city-center scenes. Our system builds upon and extends existing technology for autonomous navigation. In particular, it contains a SLAM system 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 approach for terrain analysis that deals with vegetation, dynamic objects, and negative obstacles. We furthermore describe how these individual components are integrated. Additionally, we will present the result of a large-scale experiment during which the robot Obelix traveled autonomously from our university campus to the city center of Freiburg during a busy day in August 2012. During that trial, the robot had to master a distance of over 3km. The trajectory taken by the robot and two pictures taken during its run are depicted in Fig. 1.

This work has been supported by the EC under FP7-231888-EUROPA. All authors are with the Dept. of Comp. Science, University of Freiburg.

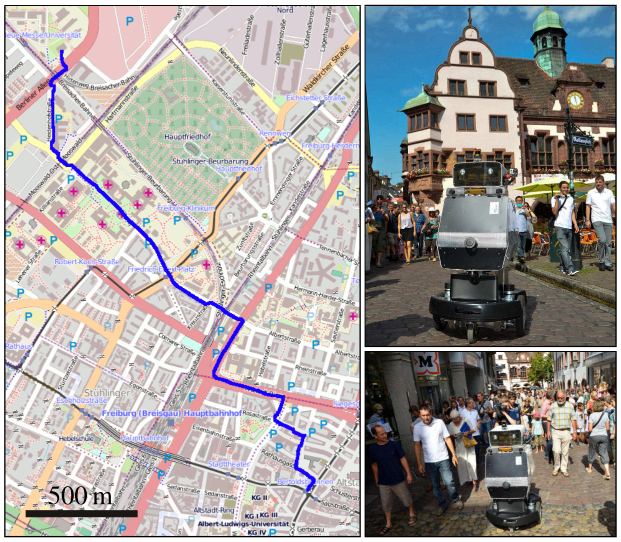

Fig. 1. Example trajectory traveled by our robot navigating in an urban environment that also includes a pedestrian zone with a large number of people. Map data on the left from OpenStreetMap (© OpenStreetMap contributors).

Thus, the aim of this paper is to not only describe the relevant components but also to highlight the capabilities that can be achieved with a system like that. We try to motivate our design decisions, critical aspects, as well as limitations of the current setup.

II. Related Work

The problem of autonomous navigation in populated areas has been studied intensively in the past. One of the pioneering systems were the robots RHINO [2] and Minerva [3] which operated as interactive mobile tour-guides in crowded museums. An extension of this tour-guide concept to interactive multi-robot systems was the RoboX system developed by Siegwart *et al.* [4] for the Expo'02 Swiss National Exhibition. Gross *et al.* [5] installed a robot as a shopping assistant that provided wayfinding assistance in home improvement stores. Although these systems were able to robustly navigate in heavily crowded environments, they were restricted to two-dimensional representations of the environment and assumed that the robots operated in a relatively confined planar area.

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 Munich City Explorer developed by Bauer *et al.* [1]. In contrast to our system, which operates completely autonomously and does not require human intervention, the city exploration system relies on interaction with humans to get the direction where to move next. The city explorer only builds local maps and does not autonomously plan its path from its position to the overall goal location. A further related approach has been developed in the context of the URUS project [6], which considered urban navigation but focused more on networking and collaborative actions as well as the integration with surveillance cameras and portable devices.

Also, the problem of autonomous navigation with robotic cars has been studied intensively. For example, there has been the DARPA Grand Challenge during which autonomous vehicles showed the ability to navigate successfully over large distances through desert areas [7], [8], [9]. During the DARPA urban challenge, several car systems have been presented that are able to autonomously navigate through dynamic city street networks with complex car traffic scenarios and under consideration of road traffic navigation rules [10], [11]. Recently, commercial self-driving cars [12] have been developed and legalized to perform autonomous navigation with cars. In contrast to these methods, which focused on car navigation, the system described in this paper has been developed to enable mobile robots to perform pedestrian-like autonomous navigation in 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.* [13]. Here, the accurate and efficient obstacle detection operating on the data obtained by tilting a laser range finder has been realized. In contrast to this system, our approach has a component for tracking moving obstacles to explicitly deal with the dynamic objects in highly populated environments and also includes a terrain analysis component that is able to deal with a larger variety of terrain.

III. 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 [14], 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 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 $\times$ 0.75 m and the robot is around 1.6 m tall.

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. 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. Additionally, another scanner is tilted 70° downwards to detect obstacles and to identify the terrain the robot drives upon. Fig. 2 visualizes the setup of the non-horizontal laser beams. A Hokuyo UTM-30LX mounted on top of the head of the robot is used for mapping and localization, 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 Trimple 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.

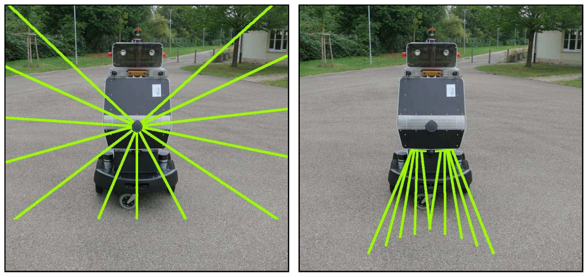
Fig. 2. One laser is mounted downwards (left) to sense the surface in front of the robot to decide whether it is safe to navigate over a particular area. A second horizontally mounted laser is combined with mirrors, which reflect a portion of its beams towards the ground (right). The data from those two lasers is used to find obstacles that are not visible in the horizontal scans.

IV. 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 huge 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 shelfs 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, and obstacle detection, which enable our robot to operate in large scale city centers. The entire navigation system described in this section runs on one standard quad core i7 laptop operating at 2.3 GHz.

A. Mapping

We apply a graph-based SLAM formulation to estimate the maximum-likelihood (ML) configuration. Let $\mathbf{x} = (\mathbf{x}_1, \ \dots \ , \mathbf{x}_n)^T$ be a vector where each element describes 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. Additionally, let $\mathbf{e}_i(\mathbf{x})$ be an

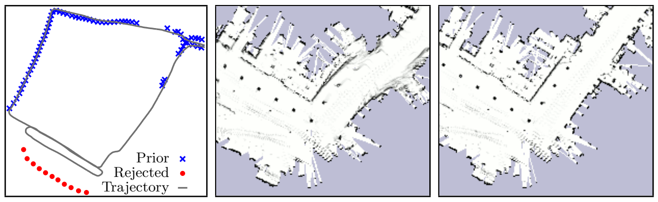

Fig. 3. Influence of outliers in the set of prior measurements. Left: Our method rejects prior measurements having a large error. Middle: The map as it is estimated by taking into account all prior measurements. Right: Our method achieves a good estimate for the map by rejecting priors which are likely to be outliers.

error function which relates the state of node i to its prior $\mathbf{z}_i$ having the covariance matrix Σ_i.

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

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_{ij \in \mathcal{C}} \|\mathbf{e}_{ij}(\mathbf{x})\|_{\Sigma_{ij}} + \sum_{i \in \mathcal{P}} \|\mathbf{e}_i(\mathbf{x})\|_{\Sigma_i}, \quad (1)$$

where $\|\mathbf{e}\|_\Sigma \overset{\text{def.}}{=} \mathbf{e}^T \Sigma^{-1} \mathbf{e}$ computes the Mahalanobis distance of its argument, and $\mathcal{C}$ and $\mathcal{P}$ are a set of constraints and priors. We employ our g2o toolkit [15] for solving Eq. (1), 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 [16]. It applies a correlative scan-matcher to estimate the motion of the robot between successive time indices. Furthermore, it obtains loop closures by matching the current scan against all scans which are within the three-sigma uncertainty ellipsoid. It filters false-positives by spectral clustering. The GPS sensor provides the set of priors $\mathcal{P}$. 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. (1), we consider a robust cost function – namely the Pseudo Huber cost function [17] – for the prior measurements. After optimization, 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. 3). Including the prior information has several advantages. First, it improves the accuracy of the obtained maps [18]. Second, if the robot extends its map, coordinates are easy to transform between different maps, because the maps share a common global coordinate frame.

B. Map Data Structure

Obtaining a 2D map given the graph-based SLAM solution and the laser data is typically done in a straight-forward 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 by 2 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. A similar approach was recently described by Konolige *et al.* [19].

We generate the local map as follows. We apply Dijkstra's algorithm to compute the distance between the nodes in the graph. This allows us to only consider observations that have been obtained by the robot in the local neighborhood of its current location. We compute the set of nodes to be used to build the local map as

$$V_{map} = \{\mathbf{x}_i \in \mathbf{x} \mid \mathit{dijkstra}(\mathbf{x}_i, \mathbf{x}_{base}) < \delta\}, \quad (2)$$

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, $\mathit{dijkstra}(\mathbf{x}_i, \mathbf{x}_{base})$ 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 $\times$ 40 m.

C. 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) [20]. MCL is a variant of particle filtering [21] 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 [22]. 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 [23]. 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

$$p(\mathbf{z} \mid \mathbf{x}) \propto \prod_i \exp\left(-\frac{\|z'_i - d'_i\|^2}{2\sigma^2}\right), \quad (3)$$

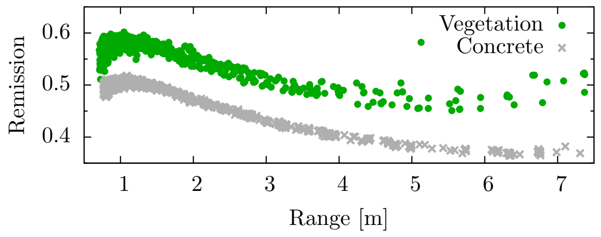

Fig. 4. Range and remission data collected by the robot observing either a concrete surface or vegetation.

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.

D. 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.

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 [24]. 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.* [24], 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. 4 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. 5, still sufficient for identifying regions covered by vegetation that should be avoided by the robot.

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. We first accumulate the 2D laser readings in 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. Then, we filter out those obstacles that appear to be dynamic but that were occluded in the older map and are therefore probably false positives. 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 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. 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 III). 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. 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,

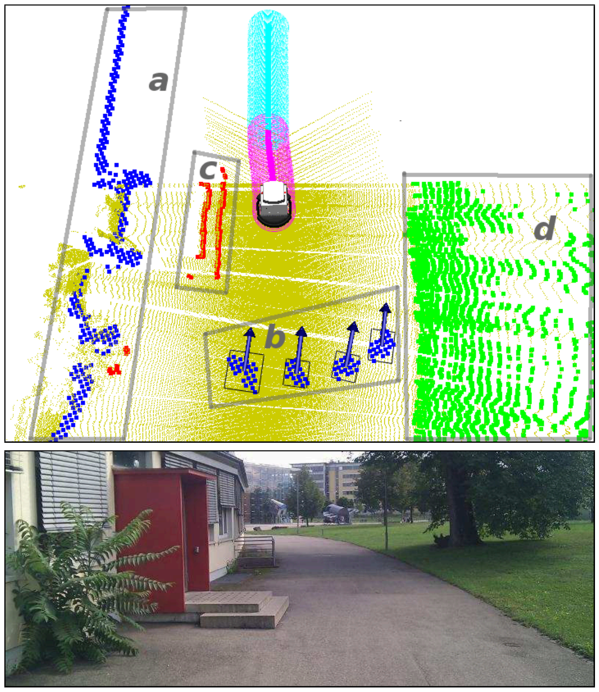

Fig. 5. Visualization of the different kinds of detected obstacles (top image). Blue points mark obstacles that are visible in the horizontal 2D laser scanners (areas a and b). Red points mark 3D obstacles that are visible in the downwards facing laser beams, but not in the 2D laser beams (mainly area c). Green points mark the detected vegetation/grass (area d). The black boxes with the arrows mark detected dynamic obstacles (area b). The remaining small yellow dots visualize the accumulated point cloud from the laser measurements. The scene depicts the robot and its planned trajectory in an environment with a lawn on the right, a building with a two-step staircase on the left (see bottom image) and four people moving behind it.

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 very effective and has the advantage of being very 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. 6 (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 a height map from the accumulated scan points while driving. For every scan point, we check if it lies substantially below the estimated height in both directions. This indicates a small hole. 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. 6, bottom image, for an example).

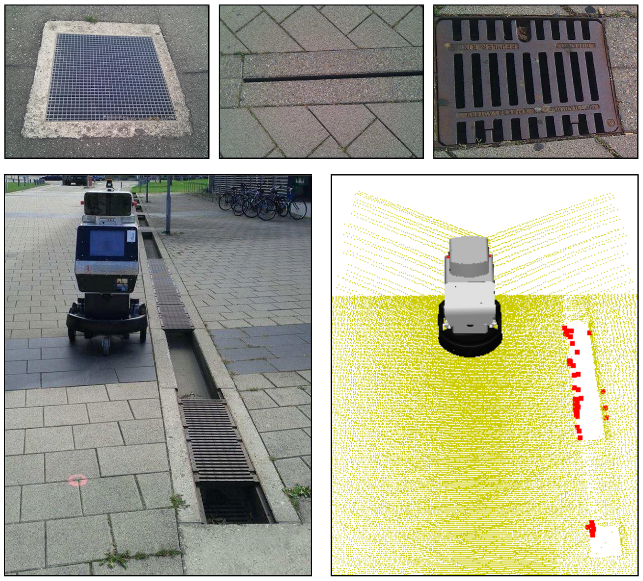

Fig. 6. Top: Traversable structures that might be detected as negative obstacles by a naive method, because some laser beams can go through them. Bottom: Example case for the obstacle detection module. While the small canals on the robot's right side are classified as negative obstacles, the gutters are identified as traversable even though there are laser measurements going through the holes.

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. On the other hand, 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. 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 speed of the platform based on the current vibration. If the vibration 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.

E. 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. On the highest level, only the topology of the environment is considered, i.e., the graph connecting 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.* [25]. This low-level planner actually computes the velocity commands sent to the robot. Note that by using this hierarchy, we loose the optimality of the computed paths. However, as reported by Konolige *et al.* [19], 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. This can be pre-computed 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. The topology graph exhibits a denser connectivity as can be seen in Fig. 7.

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.

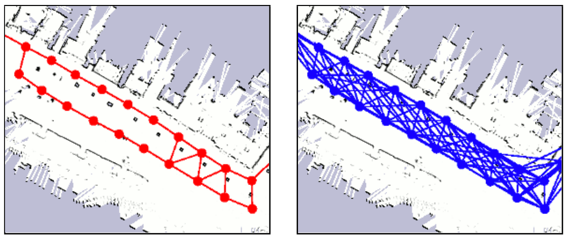

Fig. 7. Left: Partial view of the pose-graph with its constraints used for estimating the poses. Right: The same view of the topology graph generated by the planner shows that this graph typically features a denser connectivity.

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.

V. 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 7,405 m long trajectory. The map covers the area between the Technical Faculty 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. 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, the robot once got stuck in front of a little bump and one further time was manually stopped by us because of an obstacle that we believed not being perceivable by the robot.

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

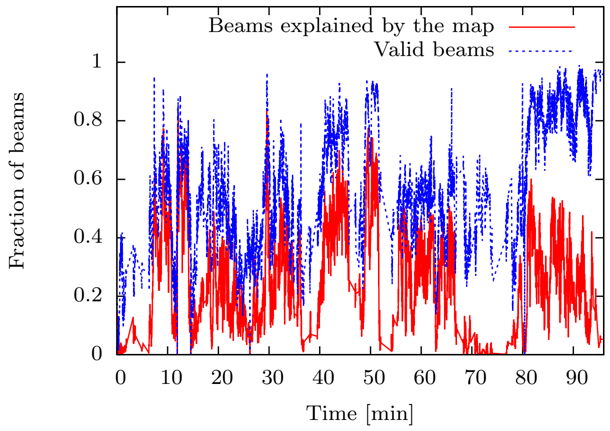

Fig. 8. This plot shows the fraction of valid beams returned by the range scanner and the fraction of beams that can be explained by the map of the environment. The robot entered a crowded area in the city center after around 80 minutes. In this period, the localization algorithm can only consider approximately 50 % of the valid readings for localization.

and international coverage in top-media. The multimedia attachment documents parts of this experimental run. More material can be found on the web[1].

A. 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. 8 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. 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. For example, after 80 minutes the robot navigates through a very crowded area. This leads to a large fraction of measurement that cannot be explained by considering the map.

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 minutes. As can be seen in Fig. 8 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. 9. This mixture of very few relevant features in the map (shown on the left hand side in Fig. 9) and the fact that the robot was driving for an extensive

[1]http://europa.informatik.uni-freiburg.de/videosdowntownDemo.html

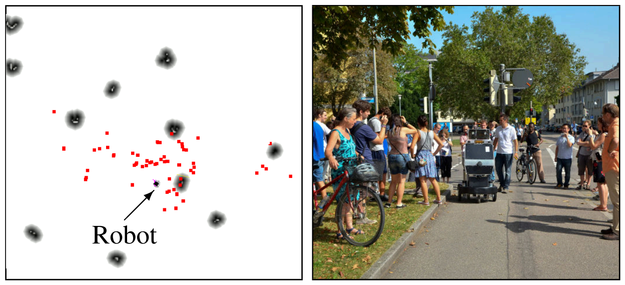

Fig. 9. Background information for the localization failure during an autonomous run to Freiburg downtown. The 2D distance map is shown on the left. As can be seen, there are only few localization features around (mostly stems of trees) and nearly all laser observations mismatch the provided model. The picture on the right shows that the robot is almost completely surrounded by people.

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 100 and 50 meters without meaningful sensations. 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. At night, typically a way smaller number of people is around and less occlusions happen to the measurements. Hence, the offset between the number of valid beams and the number of beams matching 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. A visual inspection of the localization result revealed that the position of the robot was correctly estimated at all times.

VI. Discussion

As mentioned above, the navigation system described in this paper has been implemented for and on the robot Obelix characterized in Section III. It is well-known that the design of a platform typically has a substantial influence on the algorithms needed for accomplishing the desired task. Given the navigation task Obelix had to carry out, 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.

The experimental evaluation additionally indicated several desiderata for sensor devices and perception processes. The

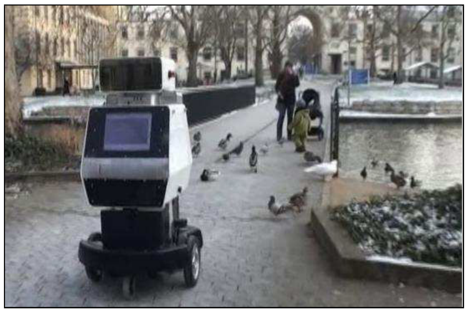

Fig. 10. Dynamic 3D obstacles which pose substantial challenges for the navigation system.

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. Appropriately dealing with such situations would require enormously farsighted sensors such as radar or similar. Additionally, simply looking at traffic lights at pedestrian crossings will not solve the problem, because the robot might want to verify as to whether a car really stops before it starts moving. For example, a police car in action might not expect the robot to actually start moving when it approaches that crossing. In such a case, additional sensations such as audio and vision might be required. 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.

We furthermore realized that other aspects are pretty challenging, as, for example, curly leafs on the ground look similar to little rocks. Whereas the robot can easily drive over leafs, rocks can actually have a substantial effect on the platform itself. Furthermore, pets or other animals like pigeons or ducks need to be modeled appropriately to effectively navigate in their vicinity (see Fig. 10).

VII. Conclusions

In this paper, we presented a navigation system that enables a mobile robot to autonomously navigate through city centers. To accomplish its task, 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 system has been implemented and demonstrated in a large-scale 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 potential hazards.

References

[1] A. Bauer, K. Klasing, G. Lidoris, Q. Mühlbauer, F. Rohrmüller, S. Sosnowski, T. Xu, K. Khnlenz, D. Wollherr, and M. Buss, "The autonomous city explorer: Towards natural human-robot interaction in urban environments," *International Journal of Social Robotics*, vol. 1, pp. 127–140, 2009.

[2] W. Burgard, A. B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun, "The interactive museum tour-guide robot," in *Proc. of the National Conference on Artificial Intelligence (AAAI)*, 1998.

[3] S. Thrun, M. Bennewitz, W. Burgard, A. Cremers, F. Dellaert, D. Fox, D. Hähnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz, "MINERVA: A second generation mobile tour-guide robot," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1999.

[4] R. Siegwart *et al.*, "RoboX at Expo.02: A large-scale installation of personal robots," *Journal of Robotics & Autonomous Systems*, vol. 42, no. 3-4, 2003.

[5] H.-M. Gross, H. Boehme, C. Schroeter, S. Mueller, A. Koenig, E. Einhorn, C. Martin, M. Merten, and A. Bley, "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)*, 2009.

[6] A. Sanfeliu, "URUS project: Communication systems," in *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, 2009, workshop on Network Robots Systems.

[7] L. Cremean *et al.*, "Alice: An information-rich autonomous vehicle for high-speed desert navigation," *Journal on Field Robotics*, 2006.

[8] S. Thrun *et al.*, "Winning the darpa grand challenge," *Journal on Field Robotics*, 2006.

[9] C. Urmson, "Navigation regimes for off-road autonomy," Ph.D. dissertation, Robotics Institute, Carnegie Mellon University, 2005.

[10] C. Urmson *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," *Journal on Field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.

[11] M. Montemerlo *et al.*, "Junior: The stanford entry in the urban challenge," *Journal on Field Robotics*, vol. 25, no. 9, pp. 569–597, 2008.

[12] "Google self-driving car project," http://googleblog.blogspot.com, 2012.

[13] E. Marder-Eppstein, E. Berger, T. Foote, B. P. Gerkey, and K. Konolige, "The office marathon: Robust navigation in an indoor office environment," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2010.

[14] "The European robotic pedestrian assistant," http://europa.informatik.uni-freiburg.de, 2009.

[15] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A general framework for graph optimization," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011.

[16] E. Olson, "Robust and efficient robotic mapping," Ph.D. dissertation, MIT, Cambridge, MA, USA, June 2008.

[17] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004.

[18] R. Kümmerle, B. Steder, C. Dornhege, A. Kleiner, G. Grisetti, and W. Burgard, "Large scale graph-based SLAM using aerial images as prior information," *Autonomous Robots*, vol. 30, no. 1, pp. 25–39, 2011.

[19] K. Konolige, E. Marder-Eppstein, and B. Marthi, "Navigation in hybrid metric-topological maps," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011.

[20] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte carlo localization for mobile robots," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Leuven, Belgium, 1998.

[21] A. Doucet, N. de Freitas, and N. Gordan, Eds., *Sequential Monte-Carlo Methods in Practice*. Springer Verlag, 2001.

[22] G. Grisetti, C. Stachniss, and W. Burgard, "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)*, 2005.

[23] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2005.

[24] K. Wurm, R. Kümmerle, C. Stachniss, and W. Burgard, "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)*, 2009.

[25] M. Rufli, D. Ferguson, and R. Siegwart, "Smooth path planning in constrained environments," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2009.