Low-cost platform for landmark-based positioning and navigation

Dipl.-Ing. Sabine Hofmann, M.Sc. Daniel Eggert, Dipl.-Inf. Colin Kuntzsch,
Prof. Dr.-Ing. Monika Sester
Institute of Cartography and Geoinformatics, Leibniz University Hannover, Germany

Abstract
This paper presents a platform based on low-cost components for landmark-based navigation intended for research and teaching purposes. The proposed platform includes a LEGO Mindstorms kit, an Android-based Smartphone as well as a compact laser scanner Hokuyo URG-04LX.

The LEGO Mindstorms kit builds the robot's chassis and provides an interface to the servo motors for movement. As a range sensor, the compact laser scanner gathers range information about the robot's environment. Finally, the Smartphone represents the central data processing and control unit bringing together sensors and actuators. The data processing involves landmark detection from the range data provided by the laser scanner and map matching of the identified landmarks. The required landmark map data structure is not based on single landmarks, but on unique constellations formed by the landmarks.

Keywords
Landmark map, Positioning, Laser scanning, Navigation, Robotics

1 INTRODUCTION

Driving autonomously requires highly accurate and reliable positioning. In order to meet the requirements of different applications, varying solutions might be suitable. In most outdoor applications, global satellite navigation systems are used. In indoor scenarios or where high positioning accuracies are required other solutions are needed. Therefore, alternative positioning systems to GPS are required, especially to increase the accuracy and to have a complementary data source in areas where GPS is not available. A possible alternative is the use of landmark maps with natural or artificial landmarks.

The platform for positioning and navigation described in this paper was specifically designed for research and teaching purposes. The idea is to simulate a real-world scenario, such as autonomous driving in an urban scene, on an easy-to-manage scale. Therefore, the used landmark map is designed for an indoor scenario where both spatial extent and number of features are limited.

The paper is structured as follows: In Chapter 3 the map generation process as well as feature detection, localization, routing and navigation are described. Details on the required hardware components for this approach and our prototype are provided in Chapter 4. This is followed in Chapter 5 by an overview of practical problems encountered during design and implementation of the prototype.

2 RELATED WORK

There are currently many map-based approaches such as occupancy grid maps as well as symbolic representations such as line maps or landmark-based maps (Burgard & Hebert 2008). These maps can be generated by the robot itself while driving, using simultaneous localization and mapping (SLAM). One important advantage is that no map needs to be produced in advance. But if landmarks should be used for positioning, appropriate landmarks have to be extracted and proofed, before being saved as a

map object (Thrun et al. 2006). Consequently, the accuracy of the produced map directly influences the positioning accuracy of the robot.

Another solution is to provide map data to the robot, which may contain only useable landmarks for positioning tasks. These maps have to be produced in advance, using for example highly accurate laser scanners or cameras, total stations, CAD plans (indoor), cadastral maps or aerial images (outdoor). Using existing map information has the advantage to provide scene interpretation and to supply knowledge of areas which appear outside the robot's field of view, which can be important for path planning. For driver assistance systems, the use of navigational maps was investigated by Blervaque (2008).

Using pole patterns for localization is currently investigated for localization in large outdoor scenarios. Weiss et al. (2005) combined GPS, odometry and a laser scanner to estimate a vehicle's position. A good overview of point matching algorithms is given by Chui and Rangarajan (2003). Brenner (2009a) describes the use of highly accurate full 3D laser scans and the localization based on local pole patterns which are used as descriptors within the map. Using pole patterns as map and an automotive laser scanner for localization gives promising results (Brenner 2009b).

3 THEORY

In the following sections all steps from map production, feature detection, localization, route planning and finally navigation to a target position are described in detail.

3.1 Landmark Map Generation

In general, for various purposes different types of maps are designed and used, e.g. maps for autonomous vehicles and maps for pedestrian navigation have to meet different requirements with regard to accuracy or feature content. When designing a map for robot indoor navigation one needs to take into account several requirements, such as the choice of objects depending on the used sensors or the uniqueness of feature patterns (e.g. triangles).

For our prototype, we decided on using pole-like objects in our landmark map. In an outdoor environment such pole objects could be tree trunks, poles of traffic lights, sign posts or lamp posts. Using poles (i.e. upright structures with homogenous shape, diameter and position at all heights) as landmarks has many advantages: they can be naturally found in almost all environments and can be described with simple geometric models detectable by different types of sensors. Full 3D laser scanners may show the curvature on the surface of the poles, depending on the angular resolution and point density which can be used to distinguish between poles and other objects. In lower resolution the linear upright structure can be found by using a geometric model or depth-jumps, which appear on both sides of the pole. When scanning only in a few horizontal planes, the pole structure will lead to a stack of horizontal slices. In a single horizontal plane (Figure 1a), where the height of the scanners on different vehicles could vary, the same poles can be detected at different heights. Therefore, their extraction from a laser scan-profile is straightforward (see Chapter 3.2). When using other types of sensors, such as radar or cameras, other features should be considered.

Another advantage of using poles is that their 3D structure can be easily encoded into a 2D map, using a representative horizontal section. This way, we represent pole-like landmarks using only 2D center coordinates and landmark radius (Figure 1b). Multiple landmark center points can form simple patterns, e.g. geometric shapes like triangles. Self-positioning is now possible through map-matching a number of measured triangles with the triangles implicitly stored within the landmark database (see Figure 1b). As matching single triangles may lead to ambiguities among several similar-shaped triangles, we consider all visible triangles, looking for unique landmark constellations (see Chapter 3.3). In order to increase the accuracy of our self-positing approach, a certain minimum density of known landmarks is required, which depends on the used laser scanner's range.

3.2 Landmark Detection

In order to meet the requirements of a low-cost platform, we use a compact 2D laser scanner, scanning in one horizontal plane (Figure 1a). Therefore, the landmark detection algorithm has to be adapted to the used sensor. In our approach, poles are extracted from depth-jumps within the laser scan profile. With a range sensor, depth-jumps will be detected if the measured range gap at two consecutive measurements is higher than a defined threshold (Figure 1c). For poles, depth-jumps can be found on either side of the scanned object. In order to reduce the influence of outliers, the radius of pole objects, which is known from the map, is used. Therefore, objects which appear to be smaller or wider in radius than the known landmarks are not used for localization.

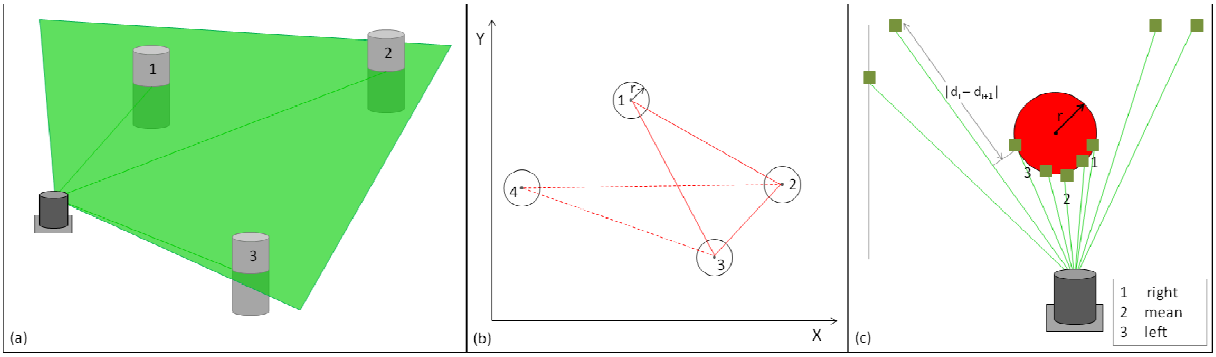

Figure 1: Laser scanner scanning in one plane (a), 2D point pattern from 3D poles (b), Pole detection based on depth-jumps (c).

As only the surface of the objects is scanned, the distance to the centre of any pole has to be computed based on the measurements and the knowledge of the pole's diameter, which is provided by the map. As directional measurement, the mean laser beam in-between the rightmost and leftmost laser beams of a detected pole's surface is used. As distance to the pole's centre, the radius of the pole is added to the range measurement at the mean laser beam (Figure 1c).

3.3 Landmark-based Positioning

The positioning process is separated into two parts. Firstly, corresponding poles between all extracted objects and reference landmarks have to be found. Therefore a map matching approach is used. Secondly, a least squares adjustment is performed to estimate the laser scanner's position (which can then be used to determine the robot's position from the fixed scanner location on top of the mobile robot platform).

In our map matching approach, we do not use single landmarks, but unique constellations formed by the landmarks. The correlation step is done by finding corresponding triangles in both the landmark map and the pole configuration from the current laser scan profile. Additional detected poles are then used for verification.

The coordinates of all extracted poles are computed in a local robot coordinate system. With these coordinates, the side lengths of all possible triangles formed by the extracted poles are computed. The map contains the reference poles and thus implicitly the reference triangles. In order to find corresponding triangles, the side lengths of local and reference triangles are compared pair-wise. For comparison of possible triangle-matches between triangles t_{map} within the landmark map and t_{scan} within the current laser scan profile, we define a similarity score as

$$score(t_{map}, t_{scan}) = \left(\overline{a}(t_{map}) - \overline{a}(t_{scan})\right)^2 + \left(\overline{b}(t_{map}) - \overline{b}(t_{scan})\right)^2 + \left(\overline{c}(t_{map}) - \overline{c}(t_{scan})\right)^2$$

where $\overline{a}$ denotes the length of the shortest edge, $\overline{b}$ the length of the longest edge and $\overline{c}$ the length of the remaining edge within the respective triangles.
Depending on the number of visible poles, three cases have to be considered:

1) Three poles: Similarity scores for all reference triangles and the single triangle formed by the visible poles are computed. The reference poles with the lowest squared position deviation are used as corresponding poles.
2) More than three poles: A single local triangle is chosen. The reference triangle with the best similarity score is used to compute an estimate for the robot's position. Based on this preliminary estimation, all additional visible poles are matched with the landmark map. The redundancy of additional poles allows for identification of objects not contained within the landmark map, and vice versa of landmarks known from the landmark map, that are missing within the current observation. Selection of the local triangle and matching all visible poles to the landmark map is repeated for all possible triangles. The solution with the maximum number of successfully matched poles is then accepted.
3) Less than three poles: No map matching is possible, i.e. no position can be estimated from less than three visible pole objects. This leads to requirements in regards to the density of the landmark map.

After the map matching process is completed successfully, the second part, position estimation, is done by resection using the coordinates of the map features and the measurements of the laser scanner (distance and direction) to all visible successfully matched poles. As a result of the adjustment one receives the robot's current position and orientation (heading) as well as belonging accuracies.

The error equations

$$d_i + v_{di} = \sqrt{(X_i - X_R)^2 + (Y_i - Y_R)^2}$$
$$\varphi_i + v_{\varphi i} = \tan^{-1}\left(\frac{Y_i - Y_R}{X_i - X_R}\right) - \gamma_0$$

where d_i, ϕ_i ... distance and direction measurement to pole i,
X_i, Y_i ... coordinates of reference pole i,
X_R, Y_R ... coordinates of robot position,
γ_0 ... heading,

result in the design matrix, which contains all geometric information.

In the least squares adjustment, the positional error of the reference map features must be considered, as the reference positions are not error-free. Therefore, for any visible pole the number of rows of the design matrix increases by four. The stochastic information of the system is considered in a weight matrix, were the measurements are assumed to be uncorrelated.

3.4 Routing

The Routing is based on the Dijkstra Shortest Path (short: DSP) algorithm (Dijkstra 1959). Since DSP determines the shortest path between two nodes within a graph, the landmark map described above has to be converted into such a graph. This conversion (Figure 2) includes the following steps:

1) Landmark Map Rasterization (with fixed cell size)
2) Tagging the cells (accessible or not) based on known obstacles (in our case landmarks only)
3) Build the graph based on accessible cells
    a. where the center of each cell represents a node in the graph and
    b. each node is connected by an edge to each accessible neighbor cell (8-neighborhood)

Furthermore, DSP depends not only on a simple graph, but on a graph with weighted edges. The common edge weight reflects the length of the edge, but also other aspects may impact the weight. Since we are going to use the determined path for autonomous navigation purposes, we also include the riskiness of an edge into its weight. A risky edge is an edge right next to cells tagged as not

accessible. That leads to longer but less “dangerous” paths in terms of possible collisions with known obstacles as shown in Figure 3.

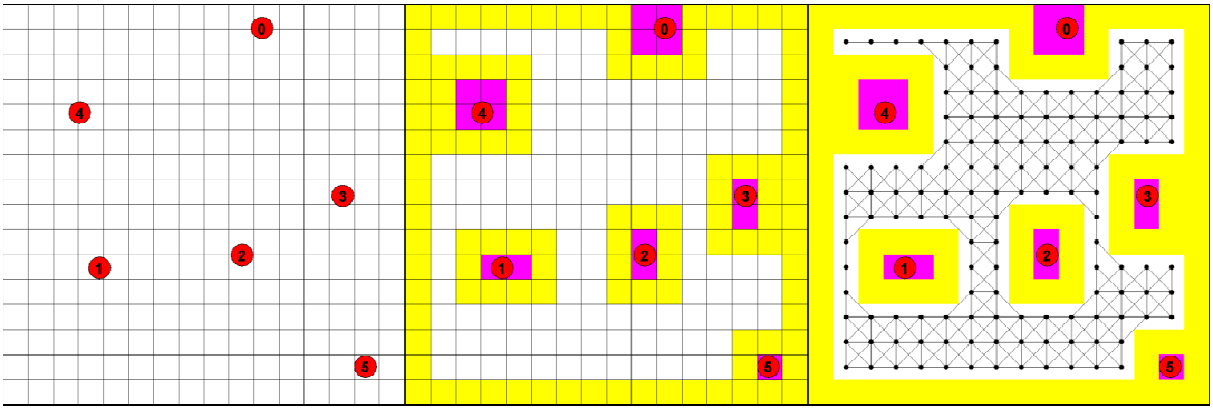

Figure 2: Conversion from Landmark Map to a Graph used by Dijkstra Shortest Path.

Depending on what kind of navigation is used for determination of the route, short route segments may raise problems due to positioning imprecision. In order to address those problems the determined path is altered. Based on the DSP route and the knowledge of what cell is accessible, the route is generalized with the goal of longer route segments. The simplest way of getting longer route segments is to join consecutive segments with no direction changes. This results in long segments for the majority of the route segments. Nonetheless, paths with many direction changes still contain very short segments. Therefore, we simplify the path in another fashion. As mentioned the route is altered not only in respect to the given graph presentation of the landmark map but also with the knowledge of the underlying tagged cell structure. In regard to that, two nodes are directly connected, if the connecting segment does not intersect with cells tagged as not accessible. This results in a virtual line-of-sight route with a minimum of short route segments. As shown in Figure 3, the described simplification results in only two long route segments. By comparison, the original DSP route consists of 30 segments, while the simplification of joining consecutive segments with no direction changes still results in eight segments.

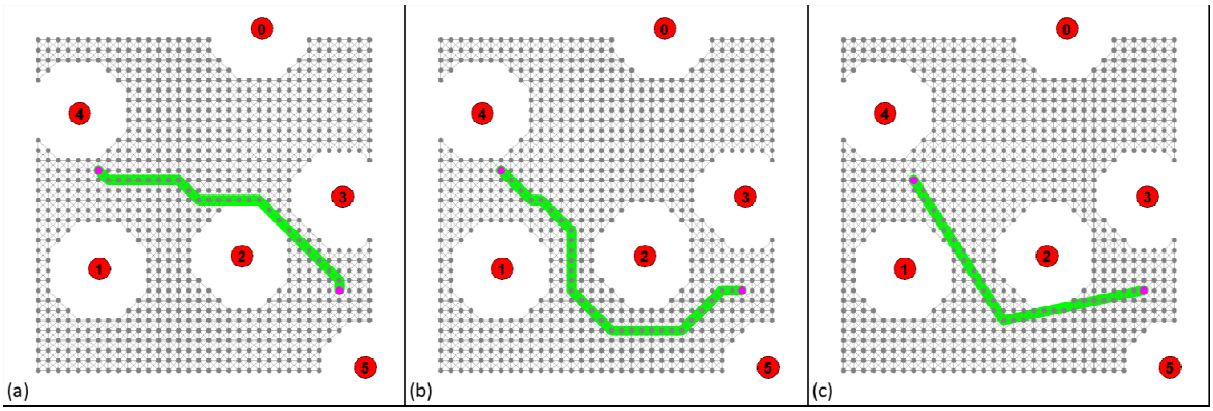

Figure 3: Shortest path in respect to distance (a), riskiness (b) and long route segments (c).

3.5 Navigation

Our Navigation solution is mainly based on navigating along each straight route segment from the robot’s current position to the desired segment destination. This task is divided into two steps. The first step is to turn towards the current destination, so the target position is right in front of the robot. The second step is to drive along the straight path until the target position is reached.

4 HARDWARE PLATFORM

In this section we provide details on the physical components of our prototype from the conceptual perspective, defining requirements for the useable hardware components. We then continue to briefly describe specific hardware choices for selected system components that introduce additional capabilities and constraints influencing the performance of the system.

4.1 System overview

Our self-positioning approach requires a moving vehicle carrying a 2D laser scanner and a processing unit dealing with laser scanner result interpretation, matching of detected landmarks with known landmark constellations, self-positioning and routing. The laser scanner needs to be capable of measuring along the horizontal axis parallel to the floor plane. Additional requirement is a database of known detectable objects within this plane whose position in a global geo reference system is known beforehand. This database must be accessible by the vehicle's processing unit (e.g. stored within the processing unit or accessible from a remote data store).

In order for the self-positioning approach to work, it is necessary that at least three objects are detectable by the laser scanner at all times. Larger numbers of visible objects allow for redundancy within position estimation and thus for detection of unknown objects and measurement error reduction. This requires the object database to provide a sufficiently high density of objects in relation to the laser scanner's range along the vehicle's route (or vice versa a sufficient scanner range for a given object density).

4.2 Hardware specifics

The developed prototype consists of three parts. First, the chassis-platform basically built from the LEGO Mindstorms NXT 2.0 Set, including the NXT Brick microcontroller and two servo motors. Second a Hokuyo URG-04LX compact laser scanner mounted on top of the chassis, providing range information for the landmark detection. And finally, a Smartphone mounted at the front as central data processing and control unit. For demonstration purposes, the object database is stored within the processing unit, i.e. stored on the Smartphone. It is complete in regards to the objects contained within our experiment setup.

4.3 Communication interfaces between system components

As the prototype has been developed for teaching and research purposes (e.g. demonstration of the self-positioning approach), we created additional input and output interfaces to access the mobile processing unit, e.g. to be able to manually change system parameters on-the-fly or to inspect intermediate system states. This way, additional control and visualization components can be run on external processing units (e.g. students' computers), accessing intermediary results and interfaces for manual control on the mobile processing unit, while not being required for the operability of the self-positioning/navigation task.

A modular architecture allows exchange or modification of the system components, as well as extension with additional modules using the provided interfaces (see Figure 4), as long as the predefined (unidirectional) interfaces between pairs of system components are implemented:

1) The interface between the laser scanner and the on-board smart phone translates a low level data stream from the sensor into a suitable representation for further processing (e.g. depth jump detection). Knowledge of the sensor-specific data format and encoding is required for the interpretation of the measurements (exchange of the sensor hardware thus requires re-implementation of the interface). The current prototype implementation for example translates arrays of distance measures, transmitted in the sequence of measurement (increasing angle in clockwise direction), into 2D positions of measured points relative to the laser scanner position and furthermore provides parameter-controlled algorithms for depth-jump detection.

2) The interface between the Smartphone and the robot motor controller needs to translate high-level navigation commands into low-level actuator-instructions (in our case setting the rotation speeds and directions for two servo motors). For the prototype, we implemented navigation commands for moving forward/backward on a straight line and turning left/right while standing still. This set of commands is sufficient for navigation in our scenario, while being exchangeable with any other robot movement specification.
3) In terms of system architecture, the on-board Smartphone acts as a data server, allowing an arbitrary number of clients to register onto the streams of laser scanner data, depth-jump detection results and routing results.
4) Additionally, we provide an interface for direct access to the Smartphone, to allow direct access of the high-level robot commands (e.g. manual or by means of alternative routing mechanisms).

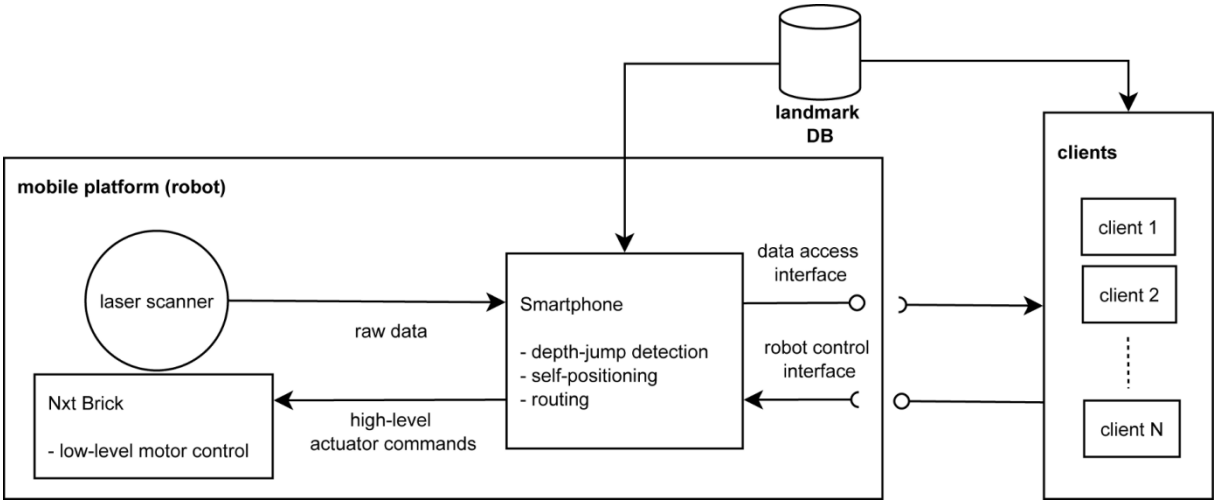

Figure 4: Conceptual view of system components and interfaces.

4.4 Prototype

For demonstration purposes we constructed a wooden square-shaped box with dimensions 2m x 2m. It serves the purposes both to provide a physical boundary for the laser scanner to not detect objects outside the experimental setup as well as a reference frame for definition of a local cartesian coordinate system. Within this local coordinate system we set up a configuration of cylindric obstacles with known positions (both physically placed within our experimental setup as well as stored within our object database). In first experiments we were able to successfully simulate autonomous navigation of the robot within the setup without external positioning assistance or active collision detection.

4.4.1 Visualization client specifics

In order to provide a graphical interface to the system's internal state for demonstration of the self-positioning approach, we created an example for a possible client application directly accessing the on-board Smartphone in both directions. It allows a live-view on the produced data, i.e. laser scan profiles, detected/matched landmarks, current routing strategy and manual initialization/termination of the navigation task. For better visualization, we installed a static web-cam filming the prototype setup. After manually registering 2D world coordinates to points within the image of the 3D scene, we are then able to directly draw 2D information on top of the web-cam image using a homography transformation (an example for the chosen visualization is shown in Figure 5).

4.4.2 Landmark setup specifics

Creation of a suitable landmark-setup for our demonstration prototype proved less trivial than expected, as we needed to consider technical and practical requirements and limitations: depth-jump detection requires a minimum distance between pairs of individual landmarks and a minimum distance

of poles from the outer boundary. The self-positioning task needs a sufficient number of landmarks producing unique triangles to allow for redundancy for error detection. On the other hand the setup is constrained by a limited amount of space available for the setup, further complicated by our desire to leave enough space between pairs of individual landmarks for proper navigation. Finally, we developed a semi-automatic setup generator that considers all of those requirements. Landmark data is currently stored locally within the Smartphone and all clients. In future versions of the prototype we are planning to centralize the landmark database in order to realize modular design, with the landmark database being a centralized component.

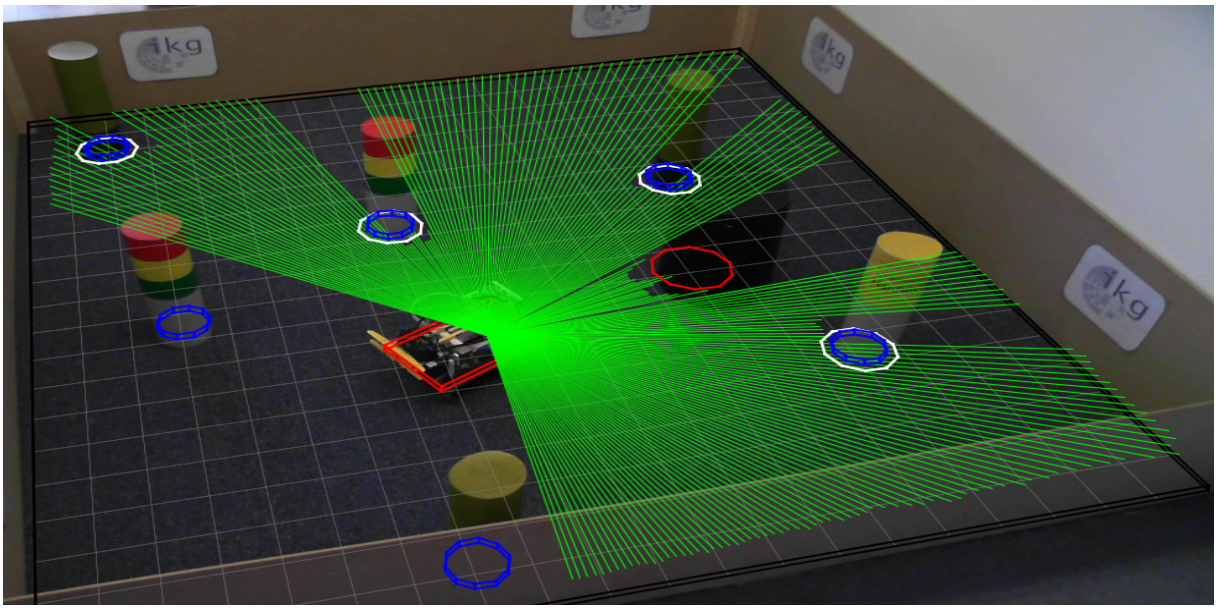

Figure 5: Prototype setup: green = current laser scan measurements, blue = known landmarks, white = successfully matched landmarks detected from laser scan profile, red = depth-jump not corresponding to a known landmark. Screenshot taken from visualization client implementation.

5 PRACTICAL PROBLEMS AND SOLUTIONS

In this chapter we present some interesting problems encountered during design and implementation and how we faced them.

5.1 Module design

As mentioned above, the described platform was developed for teaching and research purposes, so all modules like landmark detection, landmark based positioning, routing and even navigation are designed to be extensible and exchangeable. The general design of each module is shown in Figure 6. All modules work asynchronously to each other, so new available data can be processed by a module while older data can still be used by processing intensive tasks/modules. Each module has to register itself to the module providing the needed input-data, e.g. the positioning module has to register for newly detected landmarks at the landmark detection module. As soon as the landmark detection module finished processing the laser scanner data, it triggers all registered modules (like the positioning module) and provides the most current detected landmarks on demand. The trigger mechanism makes use of binary semaphores, which resume threads from idle state in case new input data is available.

Since the single modules are only coupled over the provided output and needed input data, each module can be easily altered or even exchanged as long as it sticks to the provided output data and the shown module registration and trigger mechanism. So, if a new landmark detection scheme is under investigation, none of the other modules have to be altered, which makes it well suitable for research and teaching purposes.

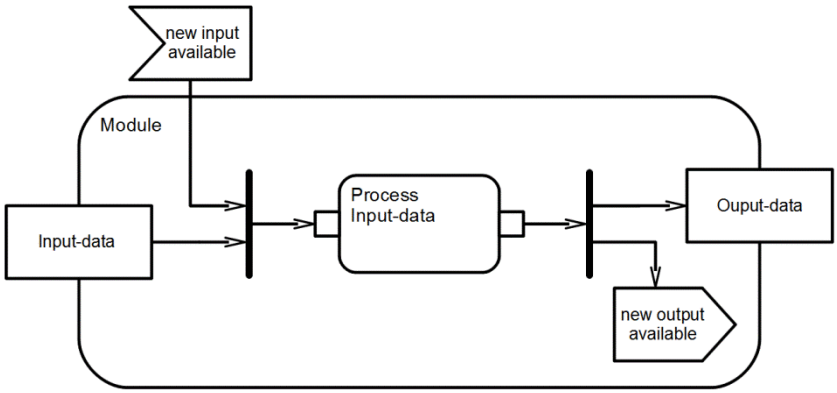

Figure 6: General module design (UML activity diagram).

5.2 Navigation Module Implementation

The navigation implementation navigates the robot based on the given route and the current position along each route segment. As described in Chapter 3.5 this involves two steps, turning to segment destination and driving to segment destination. Since the robot uses only one laser scanner which also has a blind angle of 120 degrees, the navigation module has to deal with the possibility of not being able to determine its current position or heading. In case of the turn step, the robot may not have a heading solution for the destination angle, so it will not stop turning at the desired target heading. In case of the driving step the robot may not have a position solution for its current position because less than three landmarks are visible to the laser scanner. Both problems can be faced by introducing a reverse navigation mode, where the robot simply drives backwards. Another possibility is to mount the laser scanner pivoting, which however requires a more sophisticated cable management. Moreover, a second rear facing laser scanner could be mounted onto the robot, which however conflicts with a low-cost solution.

In addition to the laser scanner's view angle constraints the imprecision of the positioning has to be taken into account in case of navigation. Since it is almost impossible to stop the robot at the exact target heading or at the exact target position some thresholds have to be defined. In our case, we set the angle tolerance threshold to ±10 degrees, meaning that the target heading is considered hit even if the determined heading is 10 degrees below or above the target heading. That heading offset has to be faced in the subsequent driving step. This can be considered as a typical regulation problem. Our solution is to let the robot trail along the route segment by setting different speeds to the driving chains. The second defined threshold is the position tolerance threshold. In our case we set it to 0.1 m, which was derived from the robot's size (0.3 x 0.2 meters) and the cell size used in the routing module (0.05m).

5.3 Real-time Self-positioning

As stated in the previous chapter, the navigation task includes both a turning step as well as a driving step. In the case the NXT servo motors actuators were optimal calibrated both navigation steps might be trivial. It is, however, impossible to perfectly calibrate the servo motors, especially since it relies on the available battery voltage, the used type of drive and of course the current underground. Furthermore, the latency between sending a command and the corresponding actuators reaction leads to small deviations of the actual route from the supposed route.

The scanning frequency of the used Hokuyo laser scanner is 10 Hz, which results in 10 range measurement sets per second. The sparse time-consuming landmark detection and map matching implementations enable us to derive a position solution from each of the provided range measurement sets. Considering the fairly slow speed of the robot, the self-positioning can be considered as a real-time self-positioning solution. This fact alleviates the navigation task, dealing with positioning inaccuracies as well as the shortcomings at the actuator translations described above.

To counter those effects our approach completely omits any actuator calibration and faces this shortcoming with a continuous real-time self-positioning. Thus, with regards to the turning command we let the motors turn until the desired heading is reached. In the same way, driving forward is a continuous process which repeatedly checks, whether the destination has been reached (as opposed to driving forward for a specified distance or time). Finally, the described routing task is not only performed once, but repeatedly whenever a significant deviation from the previous route is detected. This leads to a continuous self-adjustment of the navigation task through continuous comparison of supposed and actual robot position.

6 CONCLUSIONS AND OUTLOOK

This paper proposed a prototype of a low-cost platform for positioning and navigation purposes in an experimental indoor setup. All steps, from map generation to autonomous route planning and navigation were shown. This was followed by a detailed description of the prototype and the software modules. The modular design of the software allows the use of the prototype in teaching and research.

Currently we are working on the analysis of achievable accuracies in different map scenarios. Several experiments and practical tests are carried out to demonstrate a stable run. Furthermore, extensions to the modules such as obstacle detection, the detection of missing poles and an incremental update of the map using the robot's measurements are planned.

REFERENCES

BRENNER, C.: *Global Localization of Vehicles using Local Pole Patterns*. Proc. DAGM 2009, 31st Annual Symposium of the German Association for Pattern Recognition, Springer LNCS 5748, p. 61-70, 2009a.

BRENNER, C.: *Extraction of Features from Mobile Laser Scanning Data for Future Driver Assistance Systems*, Advances in GIScience: Proceedings of 12th AGILE Conference on GIScience, Lecture Notes in Geoinformation and Cartography, Springer, Berlin, p. 25-42, 2009b.

BURGARD, W., HEBERT, M.: *Springer Handbook of Robotics*. Chapter *World Modeling*, p. 853-869, Springer, 2008.

DIJKSTRA, E. W.: *A note on two problems in connexion with graphs*, Numerische Mathematik, Volume 1, Number 1, p. 269-271, 1959.

THRUN, S., BURGARD, W., FOX, D.: *Probabilistic Robotics*. The MIT Press, Cambridge, Mass., 2005.

WEISS, T., KAEMPCHEN, N., DIETMAYER, K.: *Precise ego localization in urban areas using laserscanner and high accuracy feature maps.* Proc. 2005 IEEE Intelligent Vehicles Symposium, Las Vegas, USA, p. 284-289, 2005.

Contact:
Dipl.-Ing. Sabine Hofmann,
Institute of Cartography and Geoinformatics, Leibniz University Hannover
Appelstr. 9A, D-30167 Hannover, Germany
phone: +49 (0511) 762-3723
fax: +49 (0511) 762-2780
Email: Sabine.Hofmann@ikg.uni-hannover.de