Robust and Computationally Efficient Navigation in Domestic Environments

Dirk Holz[1,2], Gerhard K. Kraetzschmar[1], and Erich Rome[2]

[1] Bonn-Rhein-Sieg University of Applied Sciences, Computer Science Department
dirk.holz@ieee.org, gerhard.kraetzschmar@h-brs.de
[2] Fraunhofer Institute for Intelligent Analysis and Information Systems (IAIS)
erich.rome@iais.fraunhofer.de

Abstract. Presented in this paper is a complete system for robust autonomous navigation in cluttered and dynamic environments. It consists of computationally efficient approaches to the problems of simultaneous localization and mapping, path planning, and motion control, all based on a memory-efficient environment representation. These components have been implemented and integrated with additional components for human-robot interaction and object manipulation on a mobile manipulation platform for service robot applications. The resulting system performed very successfully in the 2008 RoboCup@Home competition.

1 Introduction

Autonomous service robots that assist in housekeeping, serve as butlers, guide visitors through exhibitions in museums and trade fairs, or provide care to elderly and disabled people could substantially ease everyday life for many people and present an enormous economic potential [7,17,19]. Robots for all these applications face, however, the challenging task of operating in real-world indoor and domestic environments, such as those addressed by the RoboCup@Home league. Domestic environments tend to be cluttered, dynamic, and are populated by humans and domestic animals. In order to adequately react to sudden dynamic changes and avoid collisions, these robots need to be able to constantly acquire and process in real-time information about their environment. Furthermore, in order to act in a goal-directed manner, plan actions and navigate effectively, a robot needs an internal representation of its environment. Nature and complexity of these representations highly depend on the robot's task and application space.

For a more concrete example, consider a domestic service robot that is given the task to serve a cold drink from the refrigerator to a guest in the living room. Aside of the activities like interacting with the host and the guest, grasping objects like a can of soft drink, or other manipulation tasks, the robot needs to solve several problems related to navigation: If the environment is initially unknown, the robot must i) *explore the environment* and ii) *build a map*. Both during this exploration and map building phase and during everyday operation later on, the robot needs to iii) *localize itself* and iv) *localize task-relevant objects* (such as the refrigerator) within its environment representation. As self-localization requires a map of the environment, while mapping requires the ability to self-localize, these two problems need to be considered jointly as simultaneous localization and mapping (SLAM). SLAM has not only been a substantial research focus in the robotics community over the last decades but is also regarded as a major precondition of truly autonomous robots [20]. For actually moving to certain locations in the environment, the robot needs to iv) *plan obstacle-free paths* and v) *follow planned paths*. Due to the fact that it operates in a dynamic environment, the robot must also constantly acquire information about the environment during navigation, and use it to vi) *update the map* and vii) *avoid collisions*.

All of the above problems have been well researched in robotics, at least in isolation. For each of these problems a large variety of sophisticated algorithms have been proposed. They coexist legitimately, since they are designed or especially appropriate for a specific purpose. However, despite the huge body of literature available, the problem of robust and computationally efficient navigation in domestic environments cannot be considered solved yet. The first issue is *robustness*. Especially in RoboCup@Home, there is only a short preparation time and only five to ten minutes to solve a complex task. Hence, algorithms need to be robust and the overall system has to act reliably. Advancing robustness, however, often comes with increasing complexity that affects the *real-time applicability* of the algorithm and the overall system which is the second issue. *Scalability* is another issue since the computational complexity of many sophisticated approaches e.g. in SLAM either directly results in prohibitive memory and runtime requirements if applied to realistically-sized or large real-world environments, or at least cannot be used online in a reasonable fast cycle time. The forth issue is *integration*. The aforementioned problems are strongly interwoven as, for example, the choice of the environment representation affects the choice of localization and path-planning algorithms. Identified best-in-class solutions may have different underlying assumptions hindering integration or necessitating possibly complex transformations from one representation into another. Efficiency problems may occur especially if such transformations cannot be done once and offline, but need to be done constantly or in regular intervals due to environmental dynamics. Furthermore, if published implementations are available at all, they are often not modular and easily re-usable as they depend on a specific architecture, development framework or inter-module communication.

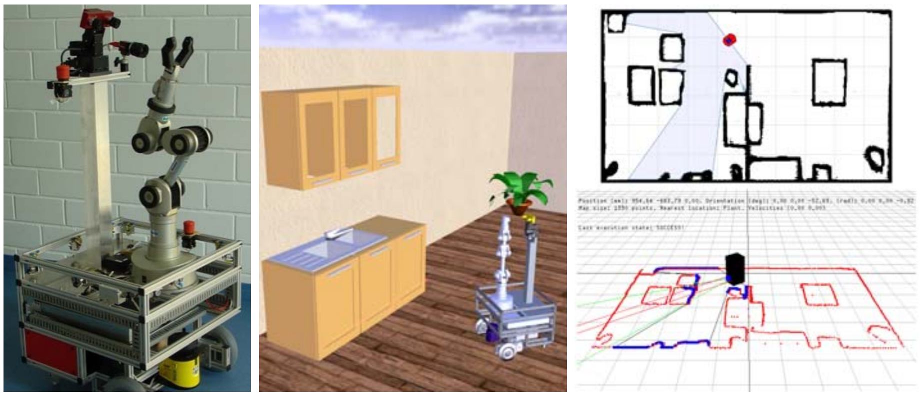
Instead of proposing yet another toolkit for navigational purposes, the goal of our work is to design and implement a (complete) set of algorithms for autonomously performing SLAM, planning paths, and controlling the motion of a mobile service robot, i.e. an approach addressing the aforementioned problems ii) to vi) which is robust, efficient and scalable. Exploration and collision avoidance are not addressed in the context of this paper. The algorithms are implemented in a modular and reusable way. Dependencies on external libraries are kept at a minimum. Primary design goals are robustness, simplicity and real-time applicability of the algorithms and the overall system.

The remainder of this paper is organized as follows: Section 2 provides a brief overview on the robot platform used for implementation and in the RoboCup@Home competitions. Section 3 introduces *sparse point maps* as a space efficient environment representation together with the proposed SLAM algorithm. Path-planning based on this representation and the used motion controllers are presented in Section 4 and Section 5, respectively. Finally, Section 6 contains some concluding remarks and an outlook on future work.

2 Base System

For evaluating the performance and robustness of the algorithms presented in this paper, the mobile service robot *Johnny Jackanapes* was used (see Figure 1), which is based on a modular mobile robot platform called *VolksBot* [21]. VolksBot has been designed specifically for rapid prototyping and robot applications in education, research and industry. The customized variant used has an integrated manipulator, a Neuronics Katana 6M180 robot arm equipped with six motors providing five degrees of freedom w.r.t. the gripper's position and orientation in its reachable workspace. It is mounted in a way to provide good reachability and maneuverability. The overall platform size is (51×51×120)cm (W×L×H) and its weight is 60 kg. The drive unit used for locomotion uses a differential drive with two actively driven wheels, powered by two 150 W motors, and two caster wheels to enhance rotating and stability under load. The robot's maximum velocity is 2 m/s.

For perceiving environmental structures, a SICK S300 2D laser scanner is used. The size of the apex angle limiting the scan plane is 270°, with an angular resolution of 0.5°. For accessing other sensors and robot platforms as well



(a) (b) (c)

Fig. 1. (a) Robot platform "Johnny Jackanapes". (b) Simulation in Microsoft Robotics Studio. (c) Simulation using Player/Stage (top) and view on the remote inspection/debug application (bottom).

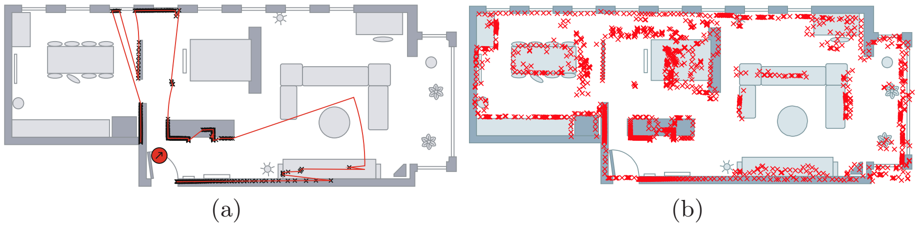


Fig. 2. (a) Single laser scan taken from the data set used in [23]. (b) Map constructed from all laser scans in the same data set. The couch table did not intersect the scan plane and is neither perceived nor modeled in the map.

as simulation environments, wrappers and interfaces have been implemented to interact with Microsoft Robotics Studio, Player/Stage and CARMEN (the Carnagie Mellon Navigation Toolkit). However, the drawback of 2D laser range finders for the purposes of collision avoidance and mapping is that objects not intersecting the 2D scan plane cannot be perceived by the robot. See e.g. the couch table that does not cause reflections in the 2D laser scan in Figure 2(a) and is thus not modeled in the point map in Figure 2(b). While in some indoor robot applications this drawback can be neglected, it plays an important role in a human's everyday environment, where typically many objects do not intersect the measurement plane, but still pose a threat to the robot. Examples include open drawers or small objects lying on the ground. In such environments, 3D information becomes crucial. Although we currently do not use a 3D sensor on the robot platform, like e.g. a 3D laser scanner or a time-of-flight camera, the proposed SLAM algorithm is already applicable to both 2D and 3D information.

3 Simultaneous Localization and Mapping

Performing SLAM to build maps and localizing in preliminary built maps are major preconditions for the autonomous operation of mobile robots in changing or preliminary unknown environments. Approaches addressing mapping and localization differ, amongst others, in formulating the problem, the means to cope with the addressed problem and in representing the environment. Occupancy grids [14] are a popular metric map representation for navigation which can be built from various kinds of simple range sensors like sonars and laser range finders. These sensors deliver information that there is some kind of obstacle in a certain distance. Occupancy grids provide a discretized representation of this kind of occupancy information. Furthermore, they distinguish unoccupied and not yet visited areas compared to feature-based representations that only store certain features perceived in the environment or geometric primitives modeling environmental structures. However, occupancy grid maps typically require a large amount of memory and can be computationally expensive to handle. On the other hand, feature-based approaches require robust feature extraction mechanisms which may be computationally expensive.

When addressing SLAM in terms of range image registration, raw measurements (i.e. point clouds) acquired with a laser scanner can directly be used to model environmental structures and to localize a mobile robot by using a matching algorithm. Hence, there is no need for applying additional feature extraction mechanisms. The problem of registering point clouds can be formulated as follows. Given two point clouds M, called model set, and D, called data set, find a transformation $\mathbf{T}$ that minimizes the alignment error between the two sets and correctly maps D onto M. The essential problems derived from this formulation are a) how to define the error function and b) how to minimize this error.

3.1 The ICP Algorithm

A widely used solution to the registration problem is the Iterative Closest Point (ICP) algorithm by Besl and McKay [1], which determines $\mathbf{T}$ in an iterative way. In each iteration step, the ICP algorithm determines pairs of corresponding points from D and M using a nearest-neighbor search. These correspondences are used to quantify and minimize the alignment error:

$$E(\mathbf{R},\mathbf{t}) = \sum_{i=1}^{|M|}\sum_{j=1}^{|D|} w_{i,j} \|\mathbf{m}_i - (\mathbf{R}\mathbf{d}_j + \mathbf{t})\|^2, \; w_{i,j} = \begin{cases} 1, & \mathbf{m}_i \text{ corresponds to } \mathbf{d}_j \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

$$\mathbf{T} = \begin{pmatrix} \mathbf{R}_{ICP} & \mathbf{t}_{ICP} \\ 0\,0\,0 & 1 \end{pmatrix} \text{ with } (\mathbf{R}_{ICP}, \mathbf{t}_{ICP}) = \underset{\mathbf{R},\mathbf{t}}{\arg\min}\, E(\mathbf{R},\mathbf{t}) \tag{2}$$

Finding the nearest neighbors and determining the correspondences is the computationally most expensive step in the ICP algorithm ($O(|D|\,|M|)$ for a brute-force implementation), since for every point $\mathbf{d}_j \in D$ the closest point $\mathbf{m}_i \in M$ needs to be determined. Here, we use an approximate *kd*-tree search [15], which reduces the complexity of the algorithm to $O(|D| \log |M|)$.

To estimate the rigid transformation $\mathbf{T}$, consisting of a rotation $\mathbf{R}$ and a translation $\mathbf{t}$, that minimizes Eq. (1) there are closed form solutions in both the two- and three-dimensional case (see [13] for a comparison). Extensions to the ICP algorithm for e.g. dealing with partial overlap of D and M or false correspondences as well as weighting and rejecting correspondence pairs can be found in [18]. The primary extension used here is to reject pairs for which the point-to-point distance exceeds a certain threshold. This threshold exponentially decays during the registration process. While initially permitting larger distances between corresponding points guarantees fast convergence of $E(\mathbf{R},\mathbf{t})$, smaller distances in later iteration steps allow fine-tuning the registration result.

3.2 Incremental Registration Using the ICP Algorithm

For registering multiple range scans and constructing a consistent map that models environmental surfaces, an incremental registration procedure is used. The first laser scan D_0 is used as the initial environment model M_0. Thus, the local coordinate frame of D_0 forms the coordinate frame for the overall map. All subsequent scans $D_i, i > 0$ are matched against M_{i-1}. The resulting transformation $\mathbf{T}_i$ is used to correct the position of all points contained in D_i, yielding the transformed point set $\check{D}_i = \{\check{\mathbf{d}}_{i,j} | \check{\mathbf{d}}_{i,j} = \mathbf{R}\mathbf{d}_{i,j} + \mathbf{t}\}$. As an initial estimate $\hat{\mathbf{T}}_i$ for $\mathbf{T}_i$ in this incremental registration we use the transformation from the last registration, i.e. $\hat{\mathbf{T}}_i = \mathbf{T}_{i-1}$. This speeds up the convergence in the ICP algorithm and drastically reduces the probability of converging to a local minimum possibly resulting in an incorrect registration result. If odometry information is available, the estimate $\hat{\mathbf{T}}_i$ is further corrected taking into account the estimated pose shift between the acquisition of D_{i-1} and D_i. Furthermore, we only register a new range scan D_i if the robot traversed more than e.g. 50 cm or turned more than e.g. 25° – a practice being quite common in recent SLAM algorithms.

To account for possibly new information in D_i, the transformed points are than added to M_{i-1}. That is, after matching range image D_i, the model set M_{i-1} computed so far is updated in step i to:

$$M_i = M_{i-1} \cup \{\check{\mathbf{d}}_{i,j} \mid \check{\mathbf{d}}_{i,j} \in \check{D}_i\}. \tag{3}$$

Thus, a model M_N, constructed by incrementally registering N range images, contains all points measured in the environment, i.e.

$$M_N = \bigcup_{i=[0,N]} \{\check{\mathbf{d}}_{i,j} \mid \check{\mathbf{d}}_{i,j} \in \check{D}_i\}. \tag{4}$$

3.3 Sparse Point Maps

The main problem of the incremental registration approach is its scalability with respect to the size of the environment and the number of range images taken. To fully cover a large environment, a lot of range images might be needed. When registering and adding all acquired range images, the model set M can get quite large, e.g. several million points for 3D scans taken in a large outdoor environment [16,22]. However, when acquiring range images in parts of the environment which are already mapped, lots of points would be added to M without providing new information about the environment. This is exploited by the following improvement to our SLAM approach, which makes the point clouds sparse.

The key idea of *sparse point maps* is to avoid duplicate storage of points, and thereby minimize the amount of memory used by the map, by conducting an additional correspondence search. That is, to neglect points that correspond to the same point in the real physical environment as a point already stored in the map. Correspondence is, thereby, defined just like in the ICP algorithm, i.e. a point $\check{\mathbf{d}}_{i,j} \in \check{D}_i$ is not added to M_{i-1}, if the point-to-point distance to its closest point $\mathbf{m}_{i-1,k} \in M_{i-1}$ is smaller than a minimum allowable distance ϵ_D.

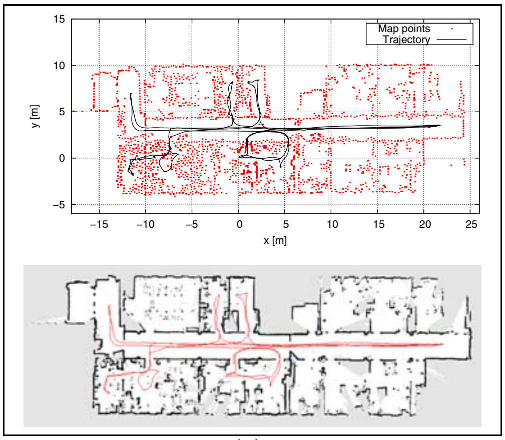
$$M_i = M_{i-1} \cup \{\check{\mathbf{d}}_{i,j} \mid \check{\mathbf{d}}_{i,j} \in D_i, \nexists \mathbf{m}_{i-1,k} \in M_{i-1} : \|\check{\mathbf{d}}_{i,j} - \mathbf{m}_{i-1,k}\| < \epsilon_\mathrm{D}\} \tag{5}$$

The threshold ϵ_D spans regions in the model in which the number of points is limited to 1, thereby providing an upper bound on the point density in a sparse point map M. Choosing a value of ϵ_D according to the accuracy of the range sensor used will exactly neglect duplicate storage of one and the same point assuming correct alignment of range images. Choosing, however, a larger value allows to reduce the number of points stored in the map. Although some details of the environment might not get modeled, a map constructed in this manner still provides a coarse-grained model of the environment as can be seen in Figure 2(b). In the actual implementation, the additional correspondence search is carried out on the kd-tree built for the ICP algorithm using ϵ_D as the distance threshold in the pair rejection step. However, here the rejected pairs are used to determine the points in $\check{D}_i$ that need to be added to M_{i-1}.

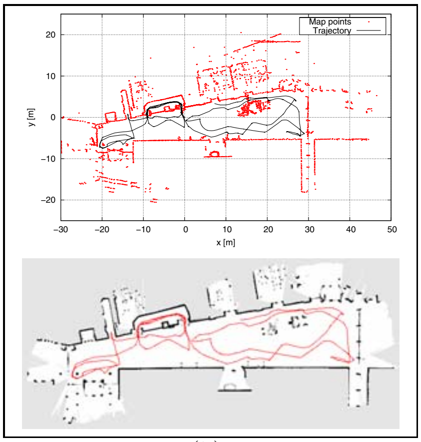
3.4 Examples and Results

The proposed incremental registration approach is so computationally efficient that it can be applied continuously during robot operation, thereby quickly reflecting changes in the environment. The runtime of the algorithm for registering a 2D laser scan lies in the range of milliseconds and increases only slightly ($\log |M|$) for growing map sizes. Figure 3 illustrates that the maps and trajectories resulting from the application of the proposed SLAM procedure are not inferior compared to those resulting from other state-of-the-art SLAM algorithms, like e.g. Rao-Blackwellized Particle Filters [6].

Having larger loops in the robot's trajectory, however, would require post-processing such as global relaxation using e.g. Lu-Milios-like approaches [2] or graph-based optimization [5]. Still, for the kind of environments addressed here,

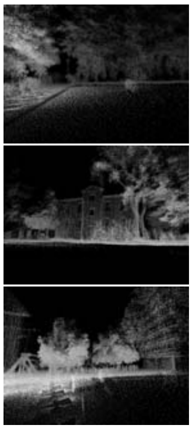


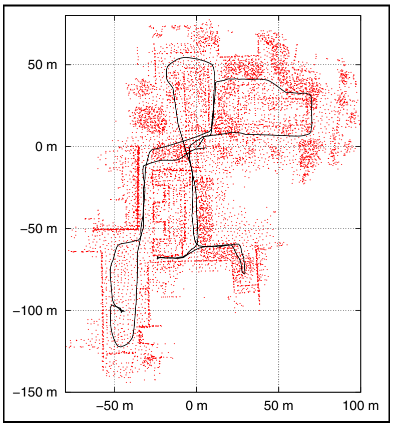
(a)



(b)

Fig. 3. Example on applying the proposed SLAM procedure on typical robot data sets (here two data sets from Cyrill Stachniss and Giorgio Grisetti). The resulting maps (a: 3092 out of 1 123 560 points, b: 2364 out of 1 975 680 points, $\epsilon_D = 15\,\mathrm{cm}$) and trajectories are shown in the upper plot. Maps provided with the data sets are shown at the bottom.


(a)



(b)

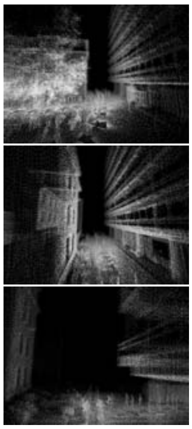

(c)

Fig. 4. Example of applying the proposed SLAM procedure on 3D data sets (here a data set from Oliver Wulf). Shown are a topview (b) with $\epsilon_D = 2\,\text{m}$ and detail views (a+c) of a model with $\epsilon_D = 20\,\text{cm}$. Map sizes are 10 060 points ($\epsilon_D = 2\,\text{m}$) and 550 979 points ($D_{min} = 20\,\text{cm}$) out of approx. 10^7 points.

e.g. apartments, the proposed stand-alone single-hypothesis approach seems sufficient. Furthermore, the approach can be integrated into a particle filter framework for multihypotheses SLAM.

An example of matching 3D laser scans to construct a 3D model of the environment and to localize the robot with all six degrees of freedom in space is shown in Figure 4.

4 Path Planning

Grid maps already have an internal structure that can be used directly for path-planning purposes. The sparse point maps used here, however, lack this ability. Instead, path-planning is addressed as a graph-search problem in the Voronoi diagram of the map points. Planning on the Voronoi diagram may not result in the shortest path, but when traveling along a path planned, the robot will always maintain a maximum distance to the obstacles represented in the map.

The Voronoi diagram is constructed using Fortune's Sweep-Line Algorithm [3]. The runtime complexity of this algorithm is $O(n \log n)$ with space complexity $O(n)$ for n points. A typical result of applying this algorithm to sparse point maps is shown in Figure 5. However, as shown in Figure 5(b), the Voronoi diagram constructed contains edges that lie outside of the modeled environment. Other edges cannot be traversed by the robot as the distance to the nearest obstacles is too short. Therefore, we prune the Voronoi diagram, first by removing all edges lying outside of or intersecting the convex hull for the map points, and

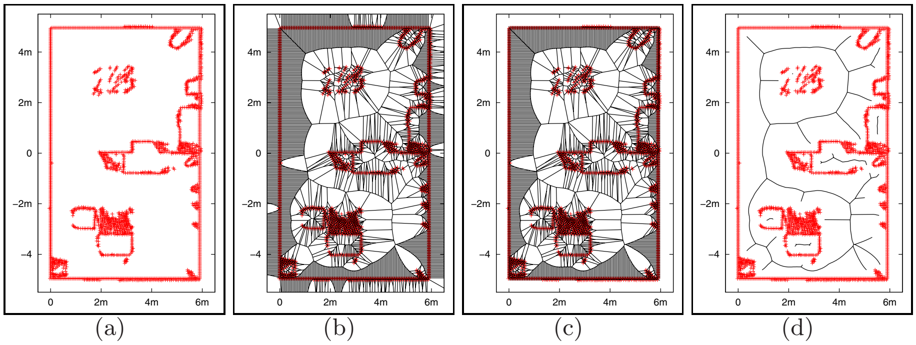


Fig. 5. Voronoi diagram construction and pruning based on a sparse point map acquired during the RoboCup GermanOpen in Hannover

second by removing all edges whose distance to neighboring points is smaller than half of the robot's width plus some safety distance (e.g. 5 cm). The latter pruning step can directly be integrated into Fortune's algorithm not affecting its complexity. The convex hull is computed by Graham's Scan Algorithm [4], which has a runtime complexity of $O(n \log n)$ for n map points. The results of both pruning steps are shown in Figure 5(c) and Figure 5(d).

Path planning is performed on the resulting graphs using $A^\star$ search [8]. The Euclidean distance to the target position ($\mathbf{x}_{goal}$) is used as an admissible heuristic in the cost function. Therefore, $A^\star$ is optimal and guaranteed to find the shortest path, if a solution exists. The overall cost function for a path from the start position through a node n to the goal is thereby defined as:

$$f(n) = g(n) + h(n) = \left(\sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{x}_{i-1}\| \right) + \|\mathbf{x}_{\text{goal}} - \mathbf{x}_n\| \tag{6}$$

where $\mathbf{x}_0 = \mathbf{x}_{\text{start}}$ and the sequence $< \mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_n >$ represents the shortest path between $\mathbf{x}_{\text{start}}$ and $\mathbf{x}_n$. As $A^\star$ can only plan paths between nodes in the graph, representatives for the true start and goal poses need to be found. The algorithm simply chooses the closest nodes in the graph and in cases where multiple nodes have similar distances to the true poses, we prefer the nodes in the direction of the other true pose. A result from applying this path-planning procedure is shown in Figure 6(a). Also shown in the map is a part of a topological layer on top of the map, storing a vector of learned and predefined objects with positions, orientations, shapes and names used to communicate with a human user.

5 Motion Control

To actually follow a planned path and reach a target location and orientation, we subsequently apply two non-linear motion controllers which have been especially

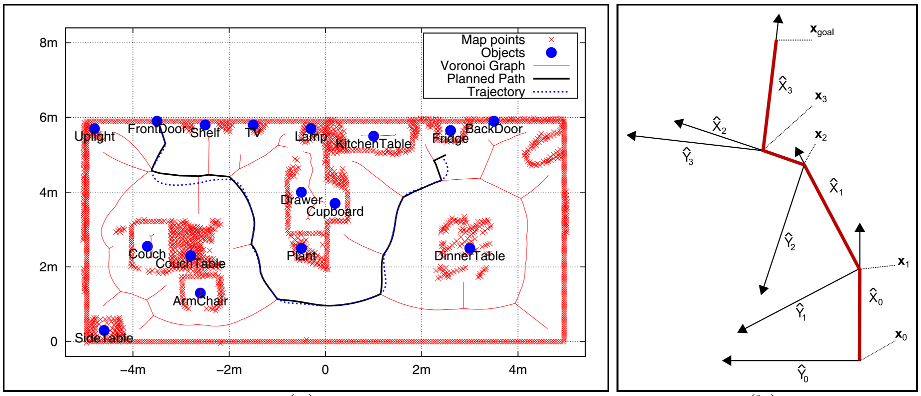


Fig. 6. (a) Path planning and following in a map of the RoboCup@Home arena at the GermanOpen 2008. Shown are the path graph (thin red lines), the planned path (thicker black lines) and the trajectory of the robot (dotted blue lines). (b) Local coordinate frames (thin black axes) in the path-tracking controller for an example path (thicker red lines).

designed for motion control of non-holonomic vehicles. The first motion controller by Indiveri and Corradini [11] is for tracking linear paths and is applied until the robot reaches the immediate vicinity of the target location. Then a second controller is used, which controls both linear velocity v and angular velocity ω of the robot, to reach the target pose while traversing a smooth trajectory [10]. The latter motion controller has previously been successfully used in an affordance-based robot control architecture in the EU FP6 project MACS [12].

For the application of the path following controller, planned paths are represented as a chain of local coordinate frames, as shown in Figure 6(b). Transforming the robot's pose into the local coordinate frame of the currently traversed graph edge allows for directly applying Indiveri's steering control law

$$\omega = -hvy\frac{\sin\theta}{\theta} - \gamma\theta \quad : \quad h, \gamma > 0 \tag{7}$$

where the controller gains h and γ are calculated depending on the current situation, i.e. whether the robot is regaining or maintaining the currently tracked path segment. Furthermore, transforming the latest laser scan into the local frame allows for checking whether the current segment is obstacle-free and can be traversed. If the segment is block, the corresponding edge is marked as being not traversable and the path is re-planned. In addition, the robot performs reactive collision avoidance [9]. The x-axes $\hat{X}_i$ of the local frames are formed by the path segments, whereas y, θ form the error signal of the controller (position and orientation deviation). The linear velocity v can be freely chosen and adapted. For details on both motion controllers it is referred to [10] and [11].

6 Concluding Remarks

We presented a complete system for autonomous navigation, including algorithms for SLAM, path planning and motion control. Using the ICP algorithm in an incremental registration procedure and sparse point maps, simulated and real robots were able to construct memory-efficient environment representations online. Path-planning on the resulting point maps has been done using $A^\star$ and fast algorithms for computing Voronoi diagrams and convex hulls for obtaining a pruned path graph. Using non-linear motion controllers for non-holonomic systems, simulated and real robots were able to robustly follow planned paths and reach target poses while localizing in and updating the sparse point map. All algorithms are highly efficient and run within the main control loop of the mobile robot platform (50-100 Hz).

Future work will focus on the development of efficient exploration and inspection strategies based on and consistent with the proposed algorithms. Extensions to this system for 3D collision avoidance and filtering out dynamics from raw range data can be found in [9]. The proposed algorithms as well as further details will be made publicly available through the RoboCup@Home Wiki[1]. Videos showing the proposed system in action and the performace of the presented SLAM algorithm are available at `http://www.b-it-bots.de/media`.

Acknowledgments

The continued support by the b-it Foundation is gratefully acknowledged. Our thanks go to Cyrill Stachniss, Giorgio Grisetti and Oliver Wulf for providing the data sets and Zoran Zivkovic for providing the sketch of the apartment environment used in Figures 2(a) and 2(b) as well as the corresponding data set.

References

1. Besl, P.J., McKay, N.D.: A Method for Registration of 3-D Shapes. IEEE Transactions on Pattern Analysis and Machine Intelligence 14(2), 239–256 (1992)
2. Borrmann, D., Elseberg, J., Lingemann, K., Nüchter, A., Hertzberg, J.: Globally Consistent 3D Mapping with Scan Matching. Journal of Robotics and Autonomous Systems 56(2), 130–142 (2008)
3. Fortune, S.J.: A Sweepline Algorithm for Voronoi Diagrams. Algorithmica, 153–174 (1987)
4. Graham, R.L.: An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set. Information Processing Letters 1(4), 132–133 (1972)
5. Grisetti, G., Lordi Rizzini, D., Stachniss, C., Olson, E., Burgard, W.: Online Constraint Network Optimization for Efficient Maximum Likelihood Map Learning. In: Proceedings of the IEEE International Conference on Robotics and Automation (2008)

[1] RoboCup@Home wiki site: `http://robocup.rwth-aachen.de/athomewiki`

6. Grisetti, G., Stachniss, C., Burgard, W.: Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters. IEEE Transactions on Robotics 23(1), 34–46 (2007)
7. Haegele, M., Neugebauer, J., Schraft, R.: From Robots to Robot Assistants. In: Proceedings of the International Symposium on Robotics (2001)
8. Hart, P.E., Nilsson, N.J., Raphael, B.: A Formal Basis for the Heuristic Determination of Minimum Cost Paths. IEEE Transactions on Systems Science and Cybernetics 4(2), 100–107 (1968)
9. Holz, D., Lörken, C., Surmann, H.: Continuous 3D Sensing for Navigation and SLAM in Cluttered and Dynamic Environments. In: Proceedings of the International Conference on Information Fusion, FUSION (2008)
10. Indiveri, G.: Kinematic Time-invariant Control of a $2d$ Nonholonomic Vehicle. In: Proceedings of the 38th Conference on Decision and Control (1999)
11. Indiveri, G., Corradini, M.L.: Switching linear path following for bounded curvature car-like vehicles. In: Proceedings of the IFAC Symposium on Intelligent Autonomous Vehicles (2004)
12. Lörken, C.: Introducing Affordances into Robot Task Execution. In: Kühnberger, K.-U., König, P., Ludewig, P. (eds.) Publications of the Institute of Cognitive Science (PICS), vol. 2-2007. University of Osnabrück, Osnabrück, Germany (2007)
13. Lorusso, A., Eggert, D.W., Fisher, R.B.: A Comparison of Four Algorithms for estimating 3-D Rigid Transformations. In: Proceedings of the British conference on Machine vision, BMVC, pp. 237–246 (1995)
14. Moravec, H.P.: Sensor fusion in certainty grids for mobile robots. Sensor Devices and Systems for Robotics, 253–276 (1989)
15. Mount, D., Arya, S.: ANN: A library for approximate nearest neighbor searching. In: Proceedings of the 2nd Annual Fall Workshop on Computational Geometry (1997)
16. Nüchter, A., Lingemann, K., Hertzberg, J., Surmann, H.: 6D SLAM – 3D Mapping Outdoor Environments. Journal of Field Robotics 24(8-9), 699–722 (2007)
17. Pollack, M.E., Engberg, S., Matthews, J.T., Thrun, S., Brown, L., Colbry, D., Orosz, C., Peintner, B., Ramakrishnan, S., Dunbar-Jacob, J., McCarthy, C., Pineau, J., Montemerlo, M., Roy, N.: Pearl: A Mobile Robotic Assistant for the Elderly. In: Proceedings of the AAAI Workshop on Automation as Eldercare (August 2002)
18. Rusinkiewicz, S., Levoy, M.: Efficient Variants of the ICP Algorithm. In: Proceedings of the International Conference on 3D Digital Imaging and Modeling (2001)
19. Siegwart, R., Arras, K.O., Bouabdallah, S., Burnier, D., Froidevaux, G., Greppin, X., Jensen, B., Lorotte, A., Mayor, L., Meisser, M., Philippsen, R., Piguet, R., Ramel, G., Terrien, G., Tomatis, N.: Robox at Expo.02: A large-scale installation of personal robots. Robotics and Autonomous Systems 42(3-4), 203–222 (2003)
20. Wang, C.-C.: Simultaneous localization, mapping and moving object tracking. PhD Thesis CMU-RI-TR-04-23, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA (April 2004)
21. Wisspeintner, T., Bose, A.: The VolksBot Concept – Rapid Prototyping for real-life Applications in Mobile Robotics. it – Information Technology 47(5), 274–281 (2005)
22. Wulf, O., Nüchter, A., Hertzberg, J., Wagner, B.: Benchmarking Urban Six-Degree-of-Freedom Simultaneous Localization and Mapping. Journal of Field Robotics 25(3), 148–163 (2008)
23. Zivkovic, Z., Booij, O., Kröse, B.: From images to rooms. Robotics and Autonomous System 55(5), 411–418 (2007)