A Unified Visual Graph-Based Approach to Navigation for Wheeled Mobile Robots

Jan Hartmann, Jan Helge Klüssendorff, and Erik Maehle

Abstract— **The emergence of affordable 3D cameras in recent years has led to an increased interest in camera-based navigation solutions. Yet, while there have been significant efforts in the field of visual simultaneous localization and mapping (VSLAM), a complete navigation package that could rival popular laser-based solutions is not available. In this paper, we will therefore introduce visual solutions to SLAM, localization, and path planning in a unified graph-based framework with the main target of wheeled robots in industrial applications. Novel solutions will be introduced in the fields of place recognition and loop closing, localization, and path planning. Our algorithms will be built for the Robotic Operating System (ROS) and fully replace the popular *gmapping* and *AMCL* algorithms.**

I. INTRODUCTION

Mobile robot navigation generally requires solutions to three different problems: mapping, localization, and path planning. First, a map of the environment has to be built. Assuming that the robot initially has no knowledge of the environment and its own position, the robot position has to be estimated while building the map, which is therefore called simultaneous localization and mapping (SLAM). Second, the robot has to localize itself in the environment. Last, a short and feasible path has to be estimated when driving to a specific place while avoiding obstacles.

Today's state-of-the-art navigation approaches are mainly designed for laser range finders (LRF). They solve mapping and localization in a probabilistic fashion, e.g. with particle or extended Kalman filters. The path planing is addressed on a 2D occupancy grid map created by the mapping system. A camera-based system using the currently popular RGBD (color + depth) cameras has several advantages over the traditional solutions. Most importantly, the camera image enables the SLAM system to close large loops by incorporating image features into the map and therefore increases the scalability of the algorithm. On the downside, the limited field of view and high image processing requirements pose challenges to developing similarly accurate and real-time capable algorithms.

Especially in industrial applications, e.g. automated cleaning and logistics, camera-based navigation may conquer new fields for autonomous mobile robots, due to the low sensor cost and the aforementioned scalability. To exploit the advantages of camera-based navigation while providing a robust, accurate, and real-time performance, in this paper, we will introduce a unified graph-based solution to mapping, localization, and path planning. We assume the target systems to be wheeled robots and therefore use wheel odometry to stabilize the visual position estimation. We will further utilize the recently introduced BRISK [1] feature descriptor to most efficiently solve the problem of place recognition. New approaches will be introduced in the fields of loop closure, graph-based localization, and path planning. The source code of the algorithms presented in this paper as well as the experimental datasets will be made publicly available with the publication of this paper[1].

The authors are with the Institute for Computer Engineering, University of Lübeck, Germany

Contact: hartmann@iti.uni-luebeck.de

A. Related Work

VSLAM has been a field of active research in recent years. The SLAM problem is, in contrast to this work, typically solved using cameras only. Different approaches have e.g. been published by Endres et al. [2], Strasdat et al. [3], and Dryanovski et al. [4], all of which use a graph-based SLAM framework. Graph-based SLAM [5] builds a map by linking particular places (nodes) based on sensor information obtained at the nodes. The graph-based approach is therefore very scalable, as large accumulated errors may be corrected by least squares optimization. It benefits from vision-based approaches, which enable the position-independant linking of nodes based on the camera images. Efficient realtime capable graph optimizers are available [6], [7].

Camera-only approaches have some important disadvantages as compared to our approach when considering industrial applications. The robot movement needs to be estimated from the camera images. This is, on the one hand, time consuming, real-time operation is therefore problematic. On the other hand, the robot movement can only be estimated if there are sufficient descriptive image features in the range of the camera. Solutions to localization and path planning in a visual graph-based map have, to the best of our knowledge, not yet been published.

Considering LRF-based navigation, robust and efficient solutions to mapping, localization, and path planning have been available for many years. Several systems are, for example, integrated in the Robot Operating System (ROS)[2]. A particle filter-based mapping approach (*gmapping*, [8]) is used to build an occupancy grid map. The localization part is then solved using an adaptive monte carlo localization (*amcl*, [9]), which is also based on particle filters. Path planning is performed in the *move_base* framework, where, based on the occupancy grid map, a global planner searches for a shortest

[1] http://www.iti.uni-luebeck.de/navigation.html

[2] ROS is currently the most widely used mobile robot framework and the algorithms in the ROS navigation stack (http://www.ros.org/wiki/navigation) will therefore be considered as benchmarks in this paper.

path to the goal and a local planner, incorporating current sensor readings, issues the actual commands to follow the global path while avoiding obstacles.

The remainder of this paper will be structured as follows. First, the general graph-based navigation framework will be introduced in Sec. II, including the different parts of the SLAM, localization, and path planning algorithms. Experimental results were conducted on a custom large-scale dataset. A comparison of the performance of this paper's approach and the algorithms provided by ROS will be given in Sec. III. The results will finally be concluded in Sec. IV.

II. VISUAL GRAPH-BASED NAVIGATION

This section describes the general design of our graph-based navigation approach. We assume a similar workflow as with LRF-based solution, where initially a map of the environment is built using a SLAM algorithm. Here the focus is on building a globally consistent and metrically accurate map. We simplify this task by reducing the robot motion to two dimensions even though the camera provides 3D information.

In operation, the robot position is then estimated using a localization algorithm given the SLAM map. Here the focus is on accurate positioning, robustness to perturbations, and the ability to relocalize. The robot position is used to drive to specific points of the map. Here the focus is on finding an optimal path and avoiding obstacles. The approaches presented in this paper are largely based on our previous experiences with probabilistic as well as graph-based VSLAM systems [10], [11].

In the remainder of this section, first, the SLAM, localization, and path planning algorithms will be briefly outlined. Then, the different common modules of the algorithms will be introduced in more detail.

A. SLAM

The task of the SLAM algorithm is to build a graph representation of the environment. It uses both wheel odometry and 3D camera information. New nodes will be added to the SLAM graph whenever a pre-defined distance to the last node is exceeded (in our experiments this is set to $0.3m$ or 10° of rotation). An edge with low uncertainty is first established based on the wheel odometry. Links to previously visited parts of the environment are searched using feature descriptors gained from the current camera image. Based on the descriptors, similar nodes in the graph are found and transformations to the most similar nodes are estimated. These transformations form edges of higher uncertainty. The odometry-based edges thus form the backbone of the SLAM graph and stabilize it against the (possibly erroneous) visual edges. Using the newly established edges, the SLAM graph is optimized to obtain a globally consistent map with minimal edge errors.

The outline of the SLAM algorithm is given in Alg. 1.

Algorithm 1: SLAM

```
if the robot has travelled a given distance then
    extract features from the current image;
    add new node;
    find potential neighbors by place recognition;
    foreach potential neighbor do
        Estimate transformation between new node and
        potential neighbors;
        if tranformation is feasible then
            add link
    optimize graph;
```

Algorithm 2: Localization

```
if the robot has travelled a given distance then
    extract features from the current image;
    find potential neighbors in radius of 1m;
    S = ∅;
    foreach potential neighbor do
        Estimate transformation T to current position;
        if T is feasible then
            S ← S ∪ T
    if |S| == 0 then
        find potential neighbors by place recognition;
        foreach potential neighbor do
            Estimate transformation T;
            if T is feasible then
                S ← S ∪ T
    if |S| > 0 then
        interpolate between all feasible transforms in S;
        set current position to interpolated transform;
    else
        estimate current position using wheel odometry;
```

B. Localization

The localization performs a mere position estimation in a map previously built by the SLAM algorithm. Localization in SLAM graphs has, to the best of our knowledge, not been investigated yet. However, graph-based maps provide a simple means to seamlessly integrate a relocalization scheme to solve such problems as the "kidnapped robot" or initial localization, where the true robot position may vary greatly from the last known position.

In our approach, we largely follow the SLAM algorithm outline (see Alg. 2). Short-term position estimates are obtained from the wheel odometry. To couteract the accumulating odometry error, the robot is relocalized after it has moved a pre-defined distance (as in the SLAM algorithm $0.3m$ or 10° of rotation). To solve the relocalization problem while ensuring that no wrong relocalizations are performed, this is done in two steps. First, the transformation to nodes close to the current position is estimated. Only if no such

Algorithm 3: ConvertToOccupancyGrid

```
if a 2d map is requested then
    setup occupancy grid using graph dimensions
    foreach node in the graph do
        retrieve depth image from node;
        transform depth image into laser scan;
        project laser scan onto 2d occupancy grid
    publish occupancy grid
```

transformation could be found the place recognition is used to find additional neighbors in the whole map and thus perform relocalization. In either case, all feasible transformations that could be obtained are interpolated to find the most stable solution.

C. Path Planning

Path planning is typically performed on 2D occupancy grid maps. In this work, rather than developing a completely new path planning algorithm, existing solutions will be utilized by extracting 2D occupancy grids from the graph map. Our solution provides an interface to the *move_base* framework[3] of ROS. A 2D occupancy grid map of the environment is generated as shown in Alg. 3. Information on the robots movement and its position are provided by wheel odometry and the localization algorithm. The sensor information for obstacle avoidance is gained by converting the depth images of the Kinect camera to 2D laser scans.

D. Modules

We have chosen a fine-grained decomposition of the algorithms in several different modules, which the graph backend, i.e. the SLAM and localization algorithms, utilize as shown in Fig. 1. Our hope is to be able to easily replace parts of the algorithm to adjust to new developments or different applications. Each module is implemented as a ROS node, i.e. a distinct process, and modules communicate over defined message interfaces. In the remainder of this section, we will describe each module in detail.

1) Feature Extraction: Feature extraction is a vital part of visual navigation. Feature descriptors are used to efficiently estimate the transformation between two images or to find similar images. Popular feature detectors and descriptors that have been used in VSLAM include gradient histogram-based approaches as the Scale-Invariant Feature Transform (SIFT, [12]) or Speeded-Up Robust Features (SURF, [13]) and binary descriptor approaches as Binary Robust Independent Elementary Features (BRIEF, [14]) and Oriented FAST and Rotated BRIEF (ORB, [15]).

In this work, Binary Robust Invariant Scalable Keypoints (BRISK, [1]) are used to compute feature descriptors for new nodes. BRISK is aimed to combine the accuracy of gradient histogram-based approaches, including robustness to image transformations, at a significantly faster computation

[3] http://www.ros.org/wiki/move_base

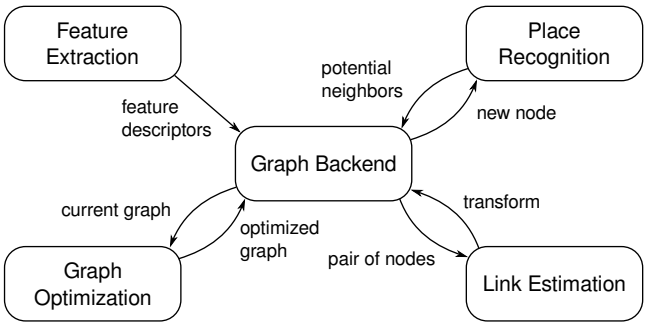

Fig. 1: General dataflow between the different modules used by the SLAM and localization algorithms.

and matching performance. While the computation speed is a necessary requirement for realtime performance, an invariance to image transformations will enable us to match nodes under larger viewpoint changes.

2) Place Recognition: The place recognition module finds similar nodes in the graph map based on the feature descriptors. It is therefore an important part of loop closure, i.e. finding links to previously visited parts of the map, as it enables a position-independant solution to the problem. The SLAM and localization algorithms are therefore able to close arbitrarily large loops and similarly solve the relocalization problem.

Visual vocabulary or bag-of-words approaches, e.g. [16], [17], are commonly used for place recognition in VSLAM. A very efficient vocabulary tree-based approach has been presented by Nistér and Stewénius [18], which has already been used in SLAM [19]. As such solutions to place recognition are designed to be used with floating point descriptors as SURF or SIFT, for this work, we have developed a novel simple and efficient method for place recognition using binary features, which is inspired by Locality Sensitive Hashing (LSH, [20]).

Locality Sensitive Hashing limits the search space for matching binary descriptors by comparing only those descriptors that have the same hash value for a given hash function. The hash function in the case of binary descriptor simply samples a random set of descriptor bits to form the hash value. LSH uses a set of random hash functions to increase the chance of finding the best match. This can be efficiently implemented using several hash tables, where for each hash value a reference to all descriptors that have the corresponding hash value is stored.

We follow a similar idea, but rather than storing a reference to a specific descriptor, we store a reference to a node. If we want to find similar nodes to a specific query node, we then count how often node numbers occur at the hash values of the query node's descriptors. The nodes with the highest number of occurences are the most similar. Fig. 2 further illustrates the place recognition process.

3) Link Estimation: The link estimation module calculates a feasible transformation between two nodes. Features correspondences are found by matching the feature descriptors for

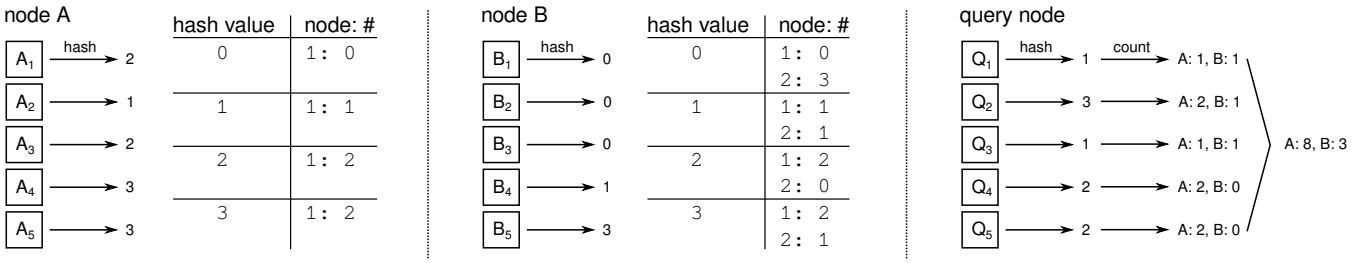

Fig. 2: Illustration of the place recognition algorithm. Two nodes with 5 descriptors each are added by storing the hash value occurences. The place recognition is then queried with a node. At the hash values of the query nodes's descriptors, the occurences are summed up for each known node, yielding node A to be the most similar with 8 occurences.

a minimal Hamming distance. Outlier correspondences are then eliminated in a Random Sample Consensus (RANSAC, [21]) scheme, where transformations are estimated using 3D positions of three feature correspondeces. Here, a simple Singular Value Decomposition (SVD)-based transformation as in [2] is performed.

Transformations are checked for feasibility by two heuristics. First, the amount of RANSAC correspondences needs to exceed a certain number (in our experiments this is fixed to 75, which has proven to eliminate most inaccurate transformations). Then, the translational and rotational components of the transformation must not be too high, assuming an accurate transformation can only be estimated for limited viewpoint changes. Here, in our experiments, the maximum translation length is fixed to $1m$, the maximum degree of rotation to $45°$.

4) Graph Optimization: The odometry error accumulates with time. Edges that were estimated based on the visual information of nodes provide a way to correct this error by using graph optimization. In this work, we use the General Graph Optimization framework (g^{2}o, [7]). g^{2}o utilizes an efficient sparse least squares optimization, graph maps can therefore easily be optimized everytime a new node is added to the graph.

An important parameter to the graph optimization is the edge uncertainty, i.e. covariance matrix. As previously stated, local edges established by the wheel odometry are assigned a low uncertainty, while visual edges are assigned a higher uncertainty. To achieve some tolerance to low accuracy in visual links and possible wrong loop closures, we go one step further and dynamically adjust the uncertainty of visual edges based on the edge error. The edge uncertainty is increased by an amount relative to the edge error, if the edge error exceeds a pre-defined value (in our experiments $0.3m$), and decreased up to a minimum uncertainty for visual edges otherwise. If the uncertainty then exceeds a maximum uncertainty, the edge is deleted. This, generally, has led to an improved accuracy and robustness of the SLAM algorithm.

III. EXPERIMENTAL RESULTS

In this section, experimental results for the SLAM as well as the localization and path planning algorithms will be shown. Experiments were performed on the PeopleBot from MobileRobots[4], which is equipped with wheel-odometry and a SICK LMS-200 LRF. A Microsoft Kinect camera facing to the front and two Asus Xtion Pro cameras facing to the side and back were mounted to the robot platform, of which in this work only the front facing Kinect will be used.

The experimental focus was on determining the accuracy of the SLAM solution, while experiments on localization and path planning were performed on the real robot. The accuracy is measured using the LRF-based mapping and localization solutions that come with the Robot Operating System (ROS) framework, *gmapping*[5] and *amcl*[6]. We assume that the LRF, due to its accuracy and range, is able to generate maps and provide localization accurate enough to be used as ground truth.

We have further tried to compare our approach to the most similar VSLAM solution, the recently published mapping system of Dryanovski et al. [4], which uses a fast visual odometry for real-time performance. Yet, due to failures of the visual odometry in regions of few texture and failure to find loop closures, a sensible comparison was not possible. Similarly, it was not possible to generate a map using gmapping and 2D scans extracted from the Kinect depth data.

A. Test Sequences

The test sequences used for this evaluation were recorded in the computer science building of the University of Lübeck. Here, the environment captures many of the properties that we expect in our target applications. It exhibits different lighting conditions, from direct sunlight to artificial light. The hallway of the building is repetitive and provides few visual features. It further requires to close large loops.

Three test sequences were evaluated, as shown in Fig. 3. The first sequence encompasses a large loop through the hallway. In the second sequence, laboratories and conference rooms were repeatedly traversed in varying directions. The third sequence provides a mixture of the other two. For a comparison with the results presented below, Fig. 4 shows the raw wheel odometry for each test sequence. With the publication of this paper, the datasets will be made publicly

[4] http://www.mobilerobots.com/ResearchRobots/PeopleBot.aspx

[5] http://www.ros.org/wiki/gmapping

[6] http://www.ros.org/wiki/amcl

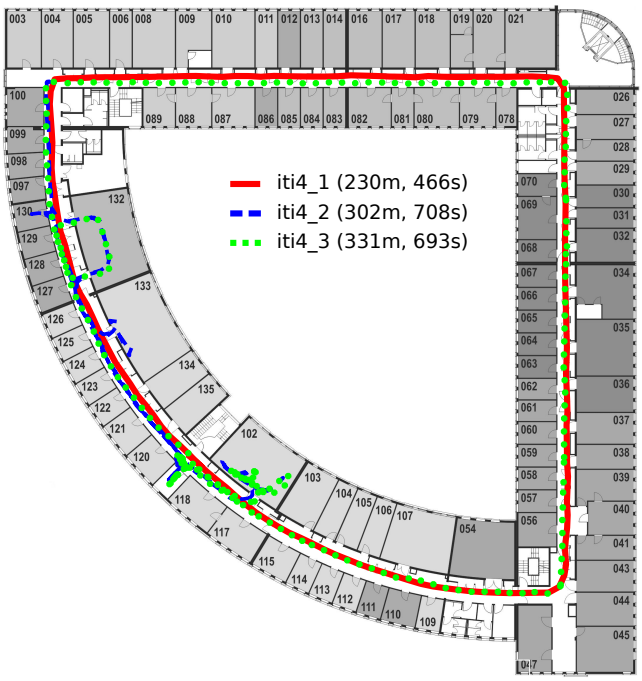

Fig. 3: Plot of the three test sequence paths hand-fitted to a floorplan of the test building. Path lengths and duration are shown in brackets.

available at our website[7], where we will further provide sample images and videos from the datasets.

B. SLAM

Fig. 5 shows a boxplot of the mean root mean squared error (RMSE) of 80 test runs for each of the three test sequences. In all test sequences, the RMSE stays well below $1m$ or 0.3% of the path length. Here, the localization accuracy is primarily dependant on the accuracy of the wheel odometry in the larger loops (note that the accuracy in the second sequence is higher, as only part of the hallway is traversed). Considering the limited accuracy and range of the Kinect camera, we feel that the RMSE cannot be significantly improved without extensive post-processing, e.g. 3D registration of the depth images.

As for the industrial application, repeatability and robustness of the SLAM algorithm are important. Here, the RMSE results are in a range of close to $20cm$ for all test sequences with no outliers, showing that loop closures are robustly detected.

C. Localization and Path Planning

The localization and path planning algorithms have been examined in combination. First, occupancy grid maps were extracted from the graph map. Results for the three test sequences can be seen in Fig. 6. While the extracted occupancy grids are largely consistent, the second test sequence (center image) shows one major problem of our approach (or VSLAM in general for that matter). In the second test

[7] http://www.iti.uni-luebeck.de/navigation.html

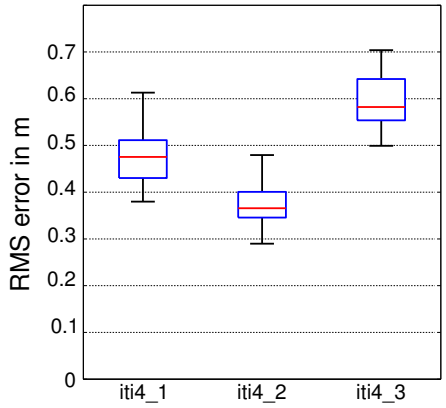

Fig. 5: Boxplot of the RMS SLAM error for all three datasets.

sequence, the hallway was travelled in both directions. This results in blurred walls in the occupancy grid, as the small field of view as well as the dependance of image features on the viewpoint prevent the creation of links between the two directions.

The experiments where therefore performed on a map of the first test sequence. Fig. 7 shows the planned path in dashed blue and the graph localization position in solid red. In the first part, the lower corner of the map was to be passed. In the second part, the robot had to follow part of the hallway with two obstacles (green circles) that were not present at the creation of the map. In both cases, the initial position was found by the localization algorithm and the goal position could be successfully reached. The localization was accurate enough to closely follow the planned path and obstacles could be avoided by incorporating the Kinect depth data. Note that the experiments were conducted several weeks after the test sequences were recorded, indicating that the localization algorithm is robust to slight illumination changes and changes in the environment.

D. Runtime and Memory

The realtime applicability of our approach was one of the requirements with the target application in mind. Here, SLAM is the most critical part of the navigation system. The SLAM runtime was measured whenever a new node was added to the graph map and for each module separately, including communication overhead. Measurements were performed on a desktop computer with an Intel Core i7 processor clocked at $3.4GHz$. In Fig. 8, results are shown with the example of the second and longest test sequence. Feature extraction and link estimation depend on the number of features that are extracted and the number of links to be established, with mean and maximum runtimes of $22ms$ and $42ms$ for feature extraction and $38ms$ and $174ms$ for link estimation respectively. Here, the highest runtime of the link estimation corresponds to the large loop closure towards the end of the test sequence. The runtime of the place recognition algorithms, too, is rather dependant on the

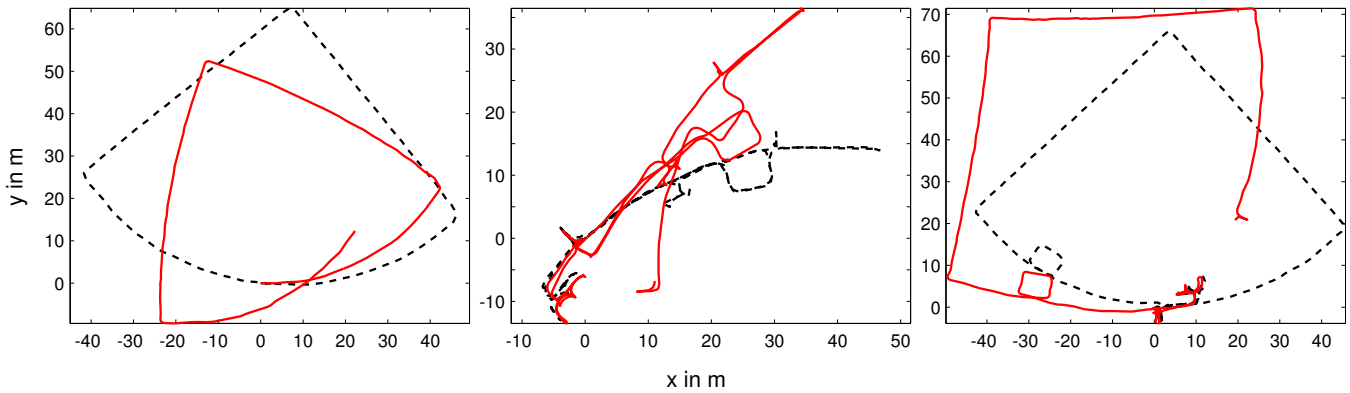

Fig. 4: Raw wheel odometry (red) for the three test sequences. Ground truth data is shown in dashed black.

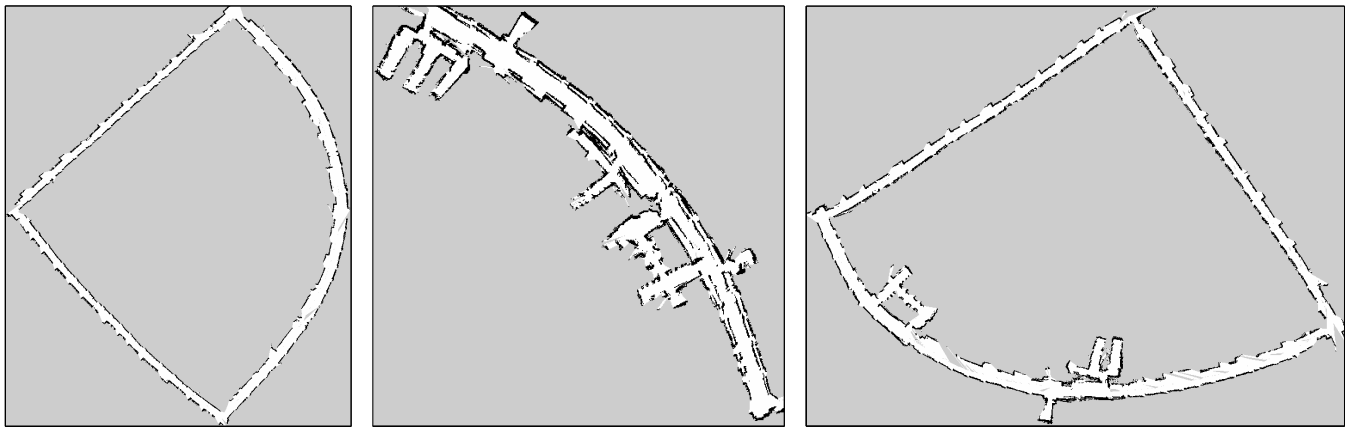

Fig. 6: Extracted occupancy grid maps for the three test sequences.

number of features than on the number of nodes, with a mean and max runtime of $10ms$ and $40ms$. The remaining graph optimization runtime grows as the number of nodes (of which a total of 1012 are constructed) in the graph map grows, to a maximum of $120ms$.

Considering that, in this test sequence, in the mean a new node was created only every $679ms$, the maximum total SLAM update time of $310ms$ more than satisfies the requirements for realtime performance. The only part of the algorithm that significantly grows with the size of the map is the graph optimization. This may for example be solved by optimizing the graph asynchronously. Similar results hold for the localization algorithm, which matches the SLAM performance with the exclusion of the graph optimization time. The time taken for the laser scan extraction for the path planning application, on the other hand, is negligible.

In contrast to runtime, memory consumption may be a problem for larger environments. In all test sequences, the SLAM and localization algorithms need 2 to $3GB$ of RAM to hold the graph map. When stored, graph maps required up to $1.3GB$. This is mainly due to the depth image, which we currently store for each node to build the occupancy grid map and which we want to use in the future to perform an offline post-processing to improve the accuracy. To enable mapping of larger environments, therefore, a submapping approach will need to be implemented.

IV. CONCLUSION

In this paper, we have presented a complete visual graph-based navigation approach, including graph-based visual simultaneous localization and mapping (VSLAM), graph-based localization, and graph-based path planning. The solutions presented in this paper were specifically developed for the application of industrial mobile robots, e.g. automated logistics and cleaning, where the scalability and low cost of camera-based approaches are important factors. Novel solutions were introduced for place recognition and loop closing, localization, and path planning.

Both VSLAM and localization benefit from the incorporation of wheel odometry measurements for local positioning, while visual cues are used to correct large-scale errors. In case of VSLAM, this leads to a robust and accurate mapping system. In case of localization, challenges as initial localization and the “kidnapped robot” problem are seamlessly solved. The path planning algorithm integrates into 2D occupancy grid-based approaches, enabling us to use the tried and tested global and local planning algorithms of the Robotic Operating System (ROS) framework.

Results were shown for three large scale test sequences with path length of 230 to 330 m. The test sequences were recorded in an office environment and exhibit challenges

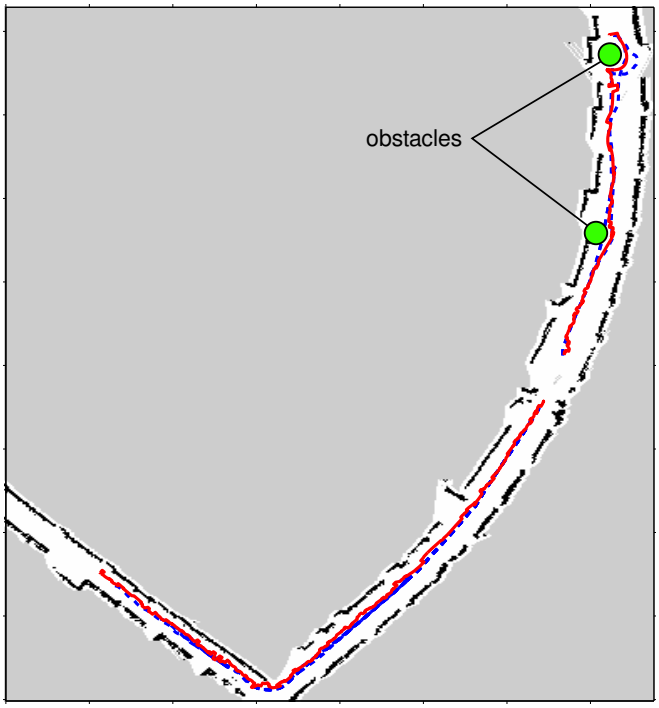

Fig. 7: Path planning and localization experiments on a map built from the first test sequence.

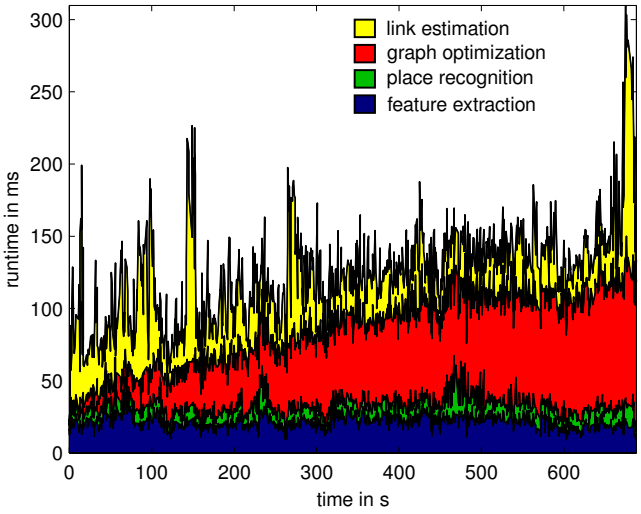

Fig. 8: SLAM update time by module for the second test sequence (accumulated values). From bottom to top: feature extraction, place recognition, graph optimization, and link estimation.

that we expect to face in the target applications: repetitve environment, different lighting conditions, regions with few texture, and large loops. Our algorithms were evaluated against a LRF-based localization solution, which we assume to be more accurate due to the higher accuracy and range of the LRF. Here, the graph-based VSLAM solution exhibits maximum errors of less than $1.0m$ or less than 0.3% of the path length. At the same time, the proposed algorithms require only a fraction of the available time for localization and mapping, enabling the real-time application. Further, the localization and path planning algorithms were tested on our robot, showing that the localization is accurate enough to closely follow the planned path and avoid obstacles.

Of course, this is only a first step in the direction of graph-based navigation. While the presented approaches provide a fully functional navigation system, the graph-based framework allows for improvements in various fields. Besides more thorough experimental evaluation of the localization and path planning algorithms, our next effort will be directed towards the most apparent problems that were obtained from the experiments. We will therefore investigate the use of multiple 3D cameras to increase the field of view and include additional sensors, e.g. inertial measurement units (IMU). The automatic extrinsic calibration and sensor fusion of these sensors will be handled inside the graph system. A submapping approach will be further developed to maintain the realtime capability and restrict memory consumption in larger environments. We are further aiming to fuse the SLAM and localization algorithms in a lifelong mapping approach, as suggested in [22]. Finally, we will investigate specific graph-based path planning and obstacle avoidance solutions.

References

[1] S. Leutenegger, M. Chli, and R. Y. Siegwart, "BRISK: Binary robust invariant scalable keypoints," in *Proc. of the IEEE Int. Conf. on Computer Vision (ICCV)*, 2011, pp. 2548–2555.

[2] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard, "An evaluation of the RGB-D SLAM system," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2012.

[3] H. Strasdat, A. J. Davison, J. M. M. Montiel, and K. Konolige, "Double window optimisation for constant time visual SLAM," in *Proc. of the IEEE Int. Conf. on Computer Vision (ICCV)*, 2011, pp. 2352–2359.

[4] I. Dryanovski, R. G. Valenti, and J. Xiao, "Fast visual odometry and mapping from rgb-d data," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2013, to appear.

[5] G. Grisetti, R. Kummerle, C. Stachniss, U. Frese, and C. Hertzberg, "Hierarchical optimization on manifolds for online 2d and 3d mapping," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2010, pp. 2432–2437.

[6] G. Grisetti, C. Stachniss, and W. Burgard, "Nonlinear constraint network optimization for efficient map learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 3, pp. 428–439, 2009.

[7] 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 and Automation (ICRA)*, 2011, pp. 3607–3613.

[8] 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 and Automation (ICRA)*, 2005, pp. 34–46.

[9] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust Monte Carlo localization for mobile robots," *Artificial Intelligence*, vol. 128, no. 1-2, pp. 99–141, 2001.

[10] J. Hartmann, D. Forouher, M. Litza, J. H. Klüssendorff, and E. Maehle, "Real-time visual SLAM using FastSLAM and the Microsoft Kinect camera," in *Proc. of the 7th German Conf. on Robotics (ROBOTIK 2012)*, Munich, 2012, pp. 458–463.

[11] J. Hartmann, W. Stechele, and E. Maehle, "Self-adaptation for mobile robot algorithms using organic computing principles," in *Architecture of Computing Systems (ARCS)*, Prague, 2012, pp. 232–243.

[12] D. Lowe, "Object recognition from local scale-invariant features," in *Proc. of the IEEE Int. Conf. on Computer Vision (ICCV)*, 1999, pp. 1150–1157.

[13] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features," *Computer Vision and Image Understanding (CVIU)*, vol. 110, no. 3, pp. 346–359, 2008.

[14] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: Binary robust independent elementary features," in *Proc. of the European Conf. on Computer Vision (ECCV)*, 2010, pp. 778–792.

[15] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Proc. of the IEEE Int. Conf. on Computer Vision (ICCV)*, 2011, pp. 2564–2571.

[16] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *Workshop on Statistical Learning in Computer Vision (ECCV)*, 2004, pp. 1–22.

[17] J. Sivic and A. Zisserman, "Self-adaptation for mobile robot algorithms using organic computing principles," in *Proc. of the IEEE Int. Conf. on Computer Vision (ICCV)*, 2003, pp. 1470–1477.

[18] D. Nistér and H. Stewenius, "Scalable recognition with a vocabulary tree," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2006, pp. 2161–2168.

[19] H. Strasdat, J. Montiel, and A. J. Davison, "Scale drift-aware large scale monocular SLAM," in *Robotics: Science and Systems (RSS)*, 2010.

[20] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *Proc. of the Int. Conf. on Very Large Data Bases (VLDB)*, 1999, pp. 518–529.

[21] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[22] K. Konolige and J. Bowman, "Towards lifelong visual maps," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (IROS)*, 2009, pp.

1156–1163.