Interactive SLAM using Laser and Advanced Sonar

Albert Diosi, Geoffrey Taylor and Lindsay Kleeman
ARC Centre for Perceptive and Intelligent Machines in Complex Environments
Department of Electrical and Computer Systems Engineering
Monash University, Victoria 3800, Australia
{*Albert.Diosi;Geoffrey.Taylor;Lindsay.Kleeman*}*@eng.monash.edu.au*

***Abstract*— This paper presents a novel approach to mapping for mobile robots that exploits user interaction to semi-autonomously create a labelled map of the environment. The robot autonomously follows the user and is provided with a verbal commentary on the current location with phrases such as "Robot, we are in the office". At the same time, a metric feature map is generated using fusion of laser and advanced sonar measurements in a Kalman filter based SLAM framework, which is later used for localization. When mapping is complete, the robot generates an occupancy grid for use in global task planning. The occupancy grid is created using a novel laser scan registration scheme that relies on storing the path of the robot along with associated local SLAM features during mapping, and later recovering the path by matching the associated local features to the final SLAM map. The occupancy grid is segmented into labelled rooms using an algorithm based on watershed segmentation and integration of the verbal commentary. Experimental results demonstrate our mobile robot creating SLAM and segmented occupancy grid maps of rooms along a 70 metre corridor, and then using these maps to navigate between rooms.**

***Index Terms*— SLAM, advanced sonar, laser, occupancy map, room segmentation**

I. Introduction

Most practical applications of mobile robotics require the robot to travel autonomously between multiple locations, typically requiring the robot to localize itself within a map of the environment. Map building is therefore a fundamental problem for which a variety of solutions have been used in previous work, including measurement by hand [4], interactive guidance with manual control [10] or people following [1], and autonomous exploration [2]. Manual map building is time-consuming and usually not considered a practical solution for robots required to operate in different environments. While autonomous exploration overcomes this problem, dangers such as stairwells and automatic doors can pose a real danger to robots with limited sensing. Interactive guidance, particularly using automatic means such as person following, offers a practical compromise that allows the user to quickly highlight both areas of interest and danger zones in a map.

An interactive mapping approach of this type was introduced by Althaus and Christensen [1]. In their system, the robot creates a topological map by autonomously following a tour guide, who indicates each new location (node) by sending signal to the robot through a wirelessly connected laptop. The user also specifies the connections between nodes, such as *door*, *corridor* and *room*. During navigation, the robot corrects its odometry drift whenever passing through a door, travelling down a straight corridor, or revisiting a charging station. While this scheme is simple and efficient, a significant drawback is that the robot can only be directed to locations identified by a node, since free space is not explicitly represented in the map. Many applications, such as cleaning the floor or finding a person in a particular room, require the robot to navigate between arbitrary points. Furthermore, the robot would benefit from knowledge of the extent of free space occupied by each room or corridor. Rather than relying on local sensing, such information can be encoded directly into the map.

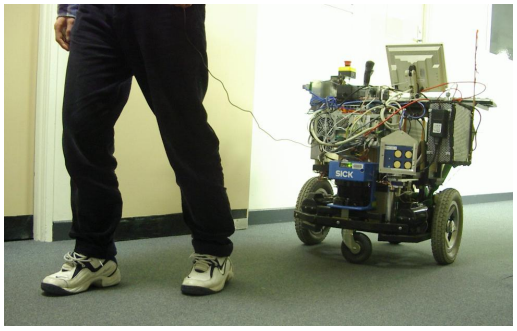

Fig. 1. SLAMbot following a tour guide during map generation.

In this paper, we present an alternative solution to interactive mapping which addresses these problems. The map used for navigation consists of point and line features, and is generated using a simultaneous localization and mapping (SLAM) framework based on fused laser and advanced sonar measurements that we have described in [6]. Using this map, the robot can localize itself to within a few centimetres anywhere in the environment. In addition to the SLAM map, our system generates an occupancy grid map with segmentation information describing the extent of each region of interest. This allows the robot to plan paths between rooms and perform tasks on entire regions. Map generation is simple and intuitive: a tour guide, followed autonomously by the robot, describes each location in the tour using simple voice commands such as *"Robot, we are in the office"* (see Figure 1). At the completion of the tour, the occupancy map is generated and the robot switches from SLAM mode to localization mode. The robot can then be sent back to any location using a voice command such as *"Robot, go to the office"*.

Many occupancy grid based map building and localization schemes using laser range measurements have been

proposed in previous work. A common approach is to first generate a laser scan map by matching individual scans collected during map building [9], [14]. The laser scan map is easily converted into a grid map for path planning, and localization is again provided by laser scan matching. An important issue in the scan matching approach to map building is sensitivity to people and other transient moving objects. This can been addressed by identifying moving objects and removing them from the scan [10].

Bourgault *et al* [2] use a similar mapping scheme to ours, involving two maps: a feature map for accurate localization generated using SLAM, and an occupancy grid for planning and exploration. The occupancy grid is generated by aligning each new laser scan using the current pose of the robot estimated by SLAM. However, the main drawback of this approach is that the SLAM feature map is continuously corrected as features are re-observed, causing the previously estimated path of the robot to become invalid. The result is a smearing of the occupancy grid with reduced certainty about the occupancy of each cell. We propose to overcome this problem by storing the robot path and laser scans with respect to neighbouring SLAM features and delaying the generation of the occupancy grid until it is needed. With this approach, the generated occupancy grid will always be consistent with the current SLAM map.

Our segmentation algorithm for dividing the occupancy grid into regions such as rooms and corridors is based on the watershed algorithm used in similar work on room segmentation [3], [7]. However, our scheme is the first to integrate verbal interactivity by guiding the segmentation based on a tour-like description of the environment.

The following section provides an overview of our experimental mobile robot, SLAMbot, and the architecture of the mapping framework. Section III then briefly describes our SLAM approach using fusion of laser and sonar measurements, introduced in [6], and describes the generation of the occupancy grid, including a novel approach to recovering the path of the robot. Section IV follows with details of interactive room segmentation, including people following, verbal marker generation and grid segmentation. Finally, Section V presents experimental results of our system mapping several locations around our labs and using the map to navigate between rooms.

II. System Overview

The testbed for our experiments is an in-house built differential drive robot called SLAMbot (see Figure 1). Sensing capabilities include odometry, two advanced sonar arrays and a Sick LMS laser range finder. The advanced sonar sensors are capable of measuring range and bearing with a standard error of 0.1° and 0.2 mm respectively, and classifying sensed targets into *planes*, *right angle corners* and *edges* [12]. The sensors are mounted on panning mechanisms which are continuously swept back and forth in a 270° arc. The Sick LMS200 generates time-of-flight laser scans in a horizontal plane with 0.5° angular resolution and 1 cm range resolution at 36 Hz. Odometry is based on 2000 count wheel encoders on both driven wheels, which are counted on a custom-programmed Xilinx field programmable gate array (FPGA). Time-stamped encoder counts are reported at precise 10ms intervals. The FPGA also appends time-stamps to the Sick serial packets, which resolves time registration issues and eliminates the need for a hard real-time operating system.

All processing is performed on-board and distributed between a 2.8 GHz hyper-threaded Pentium 4 PC and a 3.0 GHz Pentium 4 laptop, both running Linux. The software is arranged in a three layered architecture from low-level sensing and control to high level planning, with programs at each level communicating through the Sockets API. The lowest level contains a server program that reads and distributes information from sensors, provides closed-loop motor control, and implements low-level path planning and obstacle avoidance using the distance transform [11] in a local 10×10 metre area around the robot. The middle level implements SLAM based on odometry, laser and sonar measurements, stores laser measurements and the robot path during map learning, and generates the occupancy grid when learning is complete. During task execution, the middle level also provides localization information with respect to the generated grid map. Both the lower and middle levels are processed on the desktop PC. Finally the top level, processed on the laptop, implements voice recognition and person following to guide map learning. The IBM ViaVoice API provides voice recognition, with a simple fixed grammar of commands and locations (*lab*, *office*, etc). During task execution, the top level plans a safe global path using the path transform [19] with set-points to guide the robot towards the required goal.

III. Localization and Mapping

Our implementation of SLAM is based on a Kalman filter framework [5] using odometry, laser and advanced sonar and is detailed in [6]. The following is a brief overview of the scheme. The main novelty of our approach is the fusion of laser and advanced sonar measurements to improve the selection of map features. Firstly, laser measurements help to remove phantom sonar features. Then, segmentation of the laser scan into line segments is guided by the detection of sonar corner and edge reflectors, and a right-angle corner model is fitted to the intersection of lines in the laser scan when the presence of a corner is confirmed by sonar. Corner and plane features that are measured by both advanced sonar and laser are fused by weighting the measurements with their information matrices. The laser line segments, fused laser/sonar corners (carrying both position and orientation information) and sonar point features extracted from clusters of edge or corner reflectors occurring near laser line segments (typically corresponding to doorjambs and wall mouldings) are added as features to the SLAM map. In [6], all processing was performed off-line from log files using Matlab. The system has been rewritten in C (except right-angle corners, which are not used in the current implementation) and is capable of real-time on-board SLAM with at least 300 features.

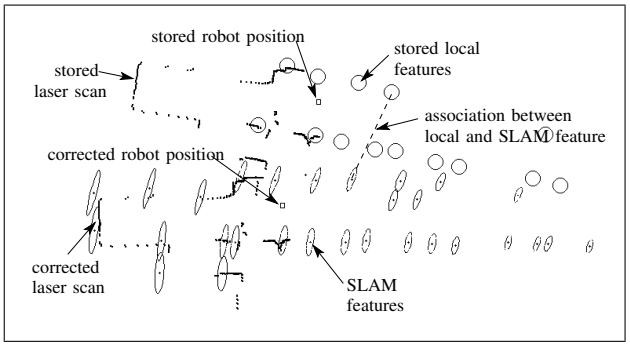

Fig. 2. Path correction using scan matching with local features.

A. Path Tracking

The sparse feature map created by SLAM provides accurate localization, but global path planning and obstacle avoidance (using distance transform based algorithms) require an accurate occupancy grid representation of the environment. If the pose of the robot in a static environment is accurately known at all times, generating an occupancy grid using laser scans without scan matching is a trivial problem. We now present a method to accurately recover the path of the robot to generate such a grid.

Registration of laser scans into an occupancy grid using SLAM based on-the-fly estimation of the robot position was shown in [2]. As noted earlier, problems with this approach arise when features are re-observed and the entire map is updated. For example, consider a robot travelling to the end of a long corridor and returning while suffering from odometry drift. As the robot re-observers a map feature near the initial position, the robot pose, feature location, and location of all correlated features in the map will be updated. For a corridor of 40 metres in length, a small angular correction by the robot at one end may cause correlated features at the other end to be corrected by a shift of several metres. This can cause serious divergence between the feature map and occupancy grid.

To address this problem, our approach involves periodically recording the robot pose and associated laser scan during mapping and delaying grid generation until the SLAM map has sufficiently converged so that the entire path can be corrected. One way to achieve this is to periodically store the robot pose in the Kalman filter state vector, so that the stored robot path is automatically corrected each time the map is updated [8]. However, increasing the size of the state vector has the undesirable effect of increasing the filter update time.

In this paper, we propose an alternative solution inspired by scan matching and illustrated in Figure 2. Based on the reasonable assumption that the SLAM map is always locally correct, our approach is to associate the periodically stored robot pose with local point features. As the map is updated, the path is also corrected provided the local configuration of point features does not change significantly. The scheme is implemented as follows: for every 1 meter of travel or change in orientation of the robot by 30°, the most recent laser scan is stored together with the position of point features in the local neighbourhood, expressed in the local robot frame. Information about feature correspondence is also stored to ease association in the reconstruction stage. To recover the stored position of the robot at a later time, a process identical to scan matching is adopted. Assuming the correspondence between two or more local features (shown as circles in Figure 2) and map points (shown as error ellipses) is known, the equations described in [13] (Appendix C) are applied to recover the pose of the robot in the SLAM map. The accuracy of the result increases as more features are used, since the local configuration of points is likely to change between initially storing the path and the final SLAM map.

B. Occupancy grid generation

After correcting the complete path of the robot in the SLAM map by applying the process described above, the registered laser scans can be assembled into an occupancy grid. The first step in grid generation is to find a suitable orientation of the SLAM map to minimize the required number of grid cells. This reduces both the memory requirements of the occupancy grid and the computational expense of segmentation and path planning. The approximate orientation is found numerically, by rotating the path from 0 to 90° in 5° steps and choosing the orientation that minimizes the area of a rectangular bounding box. This orientation defines the transformation from the SLAM map to the occupancy grid, which is then applied to the stored robot path and registered laser scans.

The occupancy grid is generated using a ray-casting method similar to the algorithm presented in [18]. All grid cells are initially set to zero (occupied), and free space is "carved" out by incrementing the count in each cell for every intersecting laser beam. To minimize the effect of orientation error, laser beams are truncated at a range of 7 metres. After processing all scans, cells with sufficiently high count (using a fixed threshold) are labelled as unoccupied. This approach to grid generation is computationally efficient: path correction and occupancy map generation for about 700 path points/laser scans requires only 1.4 seconds of on-board processing time. This suggests the possibility of generating on-the-fly grid maps during exploration.

As noted earlier, the main advantage of our approach is to store the laser scans with respect to local SLAM features, and delay grid generation until it is needed. Thus, when the occupancy grid is eventually generated, it will always be consistent with the current SLAM map.

IV. Interactive Room Segmentation

Interactive segmentation of the map into labelled rooms is a two stage process. First, the user places virtual markers in the map by verbally describing each location during map building. The markers are then used to guide a watershed segmentation of the grid map to identify complete rooms. The details of each stage are described below.

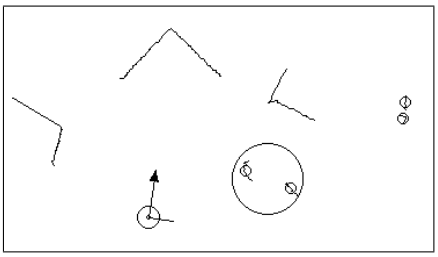

Fig. 3. People tracking in a segmented laser scan. The tracked person indicated by large circle and robot position/bearing shown by arrow.

A. People Tracking and Marker Placement

Our people tracking scheme is illustrated in Figure 3. First, the laser scan is segmented at jumps in adjacent range samples greater than a fixed threshold (depth discontinuities). Segments are labelled as leg candidates if the distance between end-points falls within the range of expected leg diameters, and four such candidates are shown in Figure 3 (small circles). Finally, pairs of leg candidates are labelled as possible people when the distance between candidates conforms to the expected leg separation, and the position of the person is calculated as the centroid of range samples. Tracking is initialized by selecting the person closest to the robot, and the candidate closest to the previous position is tracked for each new scan. Tracking is reinitialized if the motion between scans exceeds a threshold, no people candidates are found, or the user provides a verbal reset command.

During interactive map building, the robot is driven towards the tracked person with a new set-point generated every second, and the user describes each location using verbal phrases such as *"Robot, we are in the office"*. With each new phrase, the robot creates a virtual marker storing the identity of the room, a timestamp, and the range and bearing to the tracked person in the robot frame. For effective room segmentation, markers need only be placed near both sides of a boundary between rooms (such as a door). When all rooms have been visited, the user issues the verbal command *"Robot, mapping complete"* to indicate that a grid map and registered path should now be generated. Markers are transformed to the grid map by searching for the nearest (in time) robot location in the registered path, and adding the stored relative range and bearing of the marker.

B. Marker-Guided Room Segmentation

Segmentation of the grid map into labelled rooms is based on the watershed algorithm [15] modified to guide the merging stage with room markers, as illustrated in Figure 4. Taking the occupied cells of the grid as goal points, a distance transform is applied to free cells and the result is smoothed to reduce artifacts of the rectilinear grid. To calculate the watershed segmentation, the distance transform is traced uphill from each free cell to the first local maxima, typically near the centre of the room. All free cells leading to the same local maxima are identified

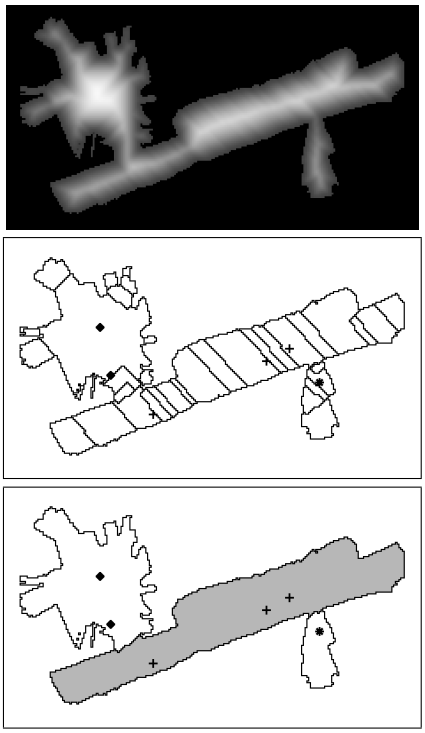

Fig. 4. Watershed-based room segmentation: distance transform (top), watershed (middle) and marker-guided merging (bottom, room markers indicated with unique symbols).

as a unique region. A typical segmentation result is shown in Figure 4 (middle). This result is comparable (in terms of over-segmentation) to other map segmentation approaches, such as the *critical line* algorithm [17].

The over-segmented watershed result is compensated by merging segments based on the interactively placed room markers. Each segment containing a marker is first labelled as belonging to the associated room, and the remaining unlabelled segments are iteratively merged with neighbouring labelled segments. At each iteration, the labelled and unlabelled pair of segments that minimize the distance from the centroid of the unlabelled segment to the closest marker in the labelled segment are merged. The typical result after merging is shown in Figure 4 (bottom). For simplicity, we assume each room is convex and simply connected such that the centroid is a valid goal for path planning.

V. Experimental results

To demonstrate the validity of the interactive SLAM framework proposed in this paper, SLAMbot was given the task of mapping and navigating between rooms along a 70 metre corridor in the vicinity of our lab. The mapping phase commenced with the tour guide issuing the vocal command *"Robot, follow me"* via a wired headset. The robot then followed the guide around the building, and vocal descriptions such as *"Robot, we are in the office"* were provided as each new area was visited. Six distinct labels were used to describe the regions in the map (see Figure 8): *lab*, *corridor*, *office*, *stairwell*, *kitchen* and *foyer*. It should be noted that the robot did not actually enter

the office or stairwell due to space constraints, but these locations could nevertheless be mapped from the door. After visiting all locations, the robot was issued with the vocal command *"Robot, mapping complete"*. This triggered the path correction, occupancy map generation and room segmentation algorithms to be applied to the recorded SLAM path and laser scans.

The features in the final SLAM map are shown in Figure 5; the slight curvature in the corridor is most likely due to systematic odometry errors. Figure 6 plots the stored laser scans as recorded at the robot positions provided by SLAM during the experiment, overlaid on the final SLAM map. Clearly, the laser scans no longer align with the SLAM map, as the SLAM map changes when features are re-observed. If the robot had visited each location more than only once, the walls would therefore appear in the laser data multiple times. Figure 6 clearly demonstrates the need for path correction.

Figure 7 shows the laser scans overlaid on the corrected robot path (using the matching scheme described in Section III-A) which demonstrates a significant improvement over Figure 6. Importantly, the laser scans are now consistent with the final SLAM map. Finally, Figure 8 shows the grid map generated using the process in Section III-B. Based on the room markers shown, the map was segmented into six areas using the process described in Section IV.

Following map generation, the robot entered navigation mode and was issued with a voice command *"Robot, go to the office"*. As shown in Figure 8, a global path was generated between the initial position in the lab and the centre of the office, and the task was successfully executed (with the robot stopping at the door of the office).

VI. Summary and Future Work

We have presented an interactive framework that enables a robot to generate a segmented metric map of an environment by following a tour guide and storing virtual markers created through verbal commands. Throughout the mapping process, the robot performs Kalman filter based SLAM using a fusion of advanced sonar and laser measurements. Two maps are generated at the completion of the mapping process: a SLAM map consisting of point and line features used for localization, and an occupancy grid for task planning. Generation of an accurate occupancy grid is central to our framework, and has been addressed with the development of a novel technique for laser scan registration. During mapping, the location of the robot and an associated laser scan are periodically recorded, along with several local features in the current SLAM map. The path of the robot can be recovered later by matching the stored local features to points in the final SLAM map using a modification of the laser scan matching algorithm. An occupancy grid consistent with the SLAM map is recovered by overlaying the laser scans on the corrected robot path. Experimental results have demonstrated the necessity of path correction, and verify that our approach generates accurate occupancy grids.

This paper has also introduced a novel method for interactive segmentation of the occupancy grid, based on the watershed algorithm with an additional merging stage guided by the verbally generated markers. This approach was successfully demonstrated to segment the map of an office environment into six labelled regions. The segmented map was used to plan a path between rooms, but knowledge of the extent of free space in each room could also be utilized for tasks such as floor cleaning.

A limitation of our implementation is the computational complexity of $O(n^2)$ for the covariance update, which limits the number of landmarks that can be handled by SLAM in real-time (this problem is not encountered in the topological mapping approach used in [1]). However, we have successfully demonstrated real-time SLAM with over 200 landmarks. Furthermore, computational expense can be reduced using a local map variant of SLAM.

In future work, we intend to extend the system by implementing large loop-closing, and a global localization strategy to determine the pose of the robot anywhere in the SLAM map. Furthermore, visual sensing could lead to a number of improvements by providing additional features for both SLAM and person following. Our recent work in multiple hypothesis laser-based people tracking [16] could also lead to improved people following behaviour.

Acknowledgments

This work was supported by the Australian Research Council funded Centre for Perceptive and Intelligent Machines in Complex Environments. We also gratefully acknowledge Steve Armstrong for technical support.

References

[1] P. Althaus and H. I. Christensen. Automatic map acquisition for navigation in domestic environments. In *Proc. ICRA*, 2003.

[2] F. Bourgault, A. A. Makarenko, S. B. Williams, B. Grocholsky, and H. F. Durrant-Whyte. Information based adaptive robotic exploration. In *Proc. IROS*, 2002.

[3] P. Buschka and A. Saffiotti. A virtual sensor for room detection. In *Proc. IROS*, 2002.

[4] I. J. Cox. Blanche–an experiment in guidance and navigation of an autonomous robot vehicle. *IEEE Trans. Robotics and Automation*, 7(2):193–203, April 1991.

[5] A. Davison. *Mobile Robot Navigation Using Active Vision*. PhD thesis, University of Oxford, 1998.

[6] A. Diosi and L. Kleeman. Advanced sonar and laser range finder fusion for simultaneous localization and mapping. In *Proc. IROS*, 2004.

[7] E. Fabrizi and A. Saffiotti. Extrating topology-based maps from gridmaps. In *Proc. ICRA*. IEEE, 2000.

[8] R. Garcia, J. Puig, P. Ridao, and X. Cufi. Augmented state kalman filtering for AUV navigation. In *Proc. ICRA*, 2002.

[9] J.-S. Gutmann and K. Konolige. Incremental mapping of large cyclic environments. In *Int. Symp. on Computational Intelligence in Robotics and Automation (CIRA'99)*, Monterey, November 1999.

[10] D. Hähnel, R. Triebel, W. Burgard, and S. Thrun. Map building with mobile robots in dynamic environments. Proc. ICRA, 2003.

[11] R. A. Jarvis. Collision-free trajectory planning using distance transforms. *Mechanical Engineering Trans.*, 10:187–191, 1985.

[12] L. Kleeman. On-the-fly classifying sonar with accurate range and bearing estimation. In *Proc. IROS*, pages 178–183. IEEE, 2002.

[13] F. Lu and E. Milios. Robot pose estimation in unknown environments by matching 2d range scans. *J. of Intelligent and Robotic Systems*, 20:249–275, 1997.

[14] Feng Lu. *Shape Registration Using Optimization for Mobile Robot Navigation*. PhD thesis, University of Toronto, 1995.

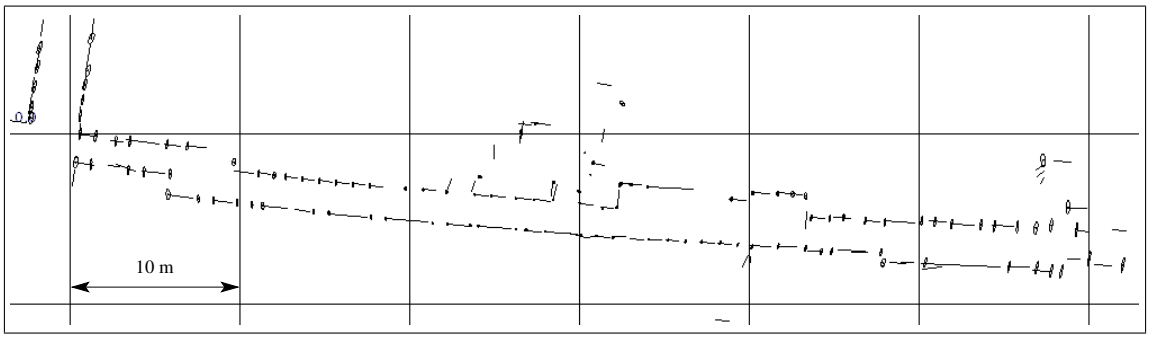

Fig. 5. Features in final SLAM map for corridor experiment (point features represented by error ellipses).

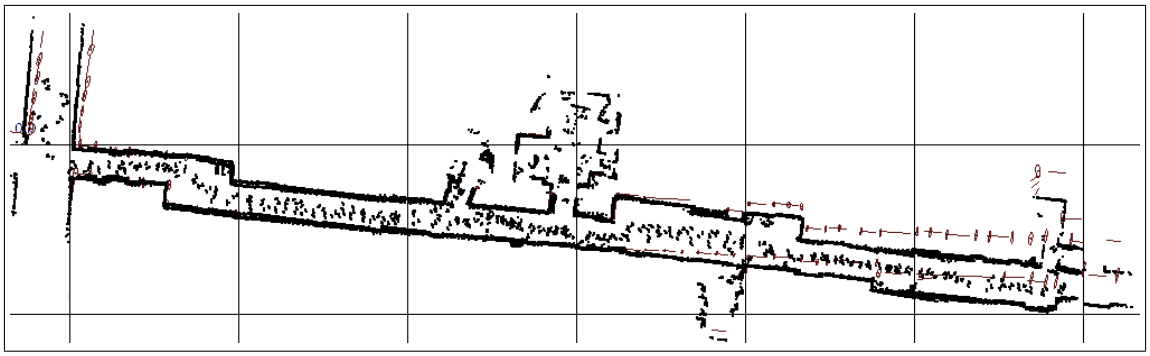

Fig. 6. Laser scans overlaid on SLAM map using recorded robot path from SLAM.

Fig. 7. Laser scans overlaid on SLAM map using corrected robot path.

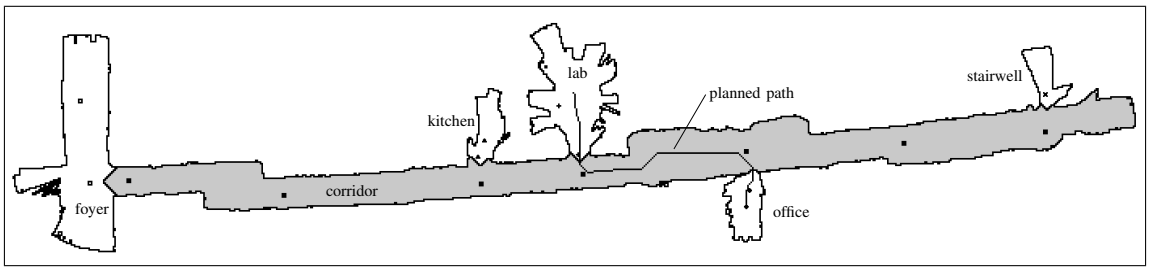

Fig. 8. Room segmentation and path planning (room markers indicated with different symbols).

[15] J. C. Russ. *The Image Processing Handbook*. 2nd ed, CRC Press, 1994.

[16] G. Taylor and L. Kleeman. A multiple hypothesis walking person tracker with switched dynamic model. In *Proc. Australasian Conf. on Robotics and Automation (ACRA)*, 2004.

[17] S. Thrun and A. Bücken. Integrating grid-based and topological maps for mobile robot navigation. In *Proc. Thirteenth Nat. Conf. on AI*, 1996.

[18] M. Veeck and W. Burgard. Learning polyline maps from range scan data acquired with mobile robots. In *Proc. IROS*, 2004.

[19] A. Zelinsky. Using path transforms to guide the search for findpath in 2D. *I. J. Robotic Res. 13(4)*, pages 315–325, 1994.