An Embedded Particle Filter SLAM implementation using an affordable platform

Martin Llofriu[1,2], Federico Andrade[1], Facundo Benavides[1], Alfredo Weitzenfeld[2] and Gonzalo Tejera[1]

[1]Instituto de Computación, Facultad de Ingeneiría, Universidad de la República, Montevideo, Uruguay
[2]College of Engineering, University of South Florida, Tampa, FL, USA

***Abstract*—The recent growth in robotics applications has put to evidence the need for autonomous robots. In order for a robot to be truly autonomous, it must be able to solve the navigation problem. This paper highlights the main features of a fully embedded particle filter SLAM system and introduces some novel ways of calculating a measurement likelihood. A genetic algorithm calibration approach is used to prevent parameter over-fitting and obtain more generalizable results. Finally, it is depicted how the developed SLAM system was used to autonomously perform a field covering task showing robustness and better performance than a reference system. Several lines of possible improvements to the present system are presented.**

I. Introduction

The recent growth in robotics applications has put to evidence the need for autonomous robots. In order for a robot to be truly autonomous, it must be able to solve the navigation problem, which may be seen as the robot being able to answer four questions: Where am I? Where have I been? Where should I go? How will I get there?[15]

The first two questions may be solved through mapping and localization, whereas the last two questions concern to goal selection and path planing respectively. In this work the problem of simultaneous localization and mapping (SLAM) is addressed. In addition, a working solution has been implemented using a low cost robot. Furthermore, the implemented solution has been tested on the autonomous task of field covering, which may be applied to automated cleaning robots, ground profiling in agriculture and exhaustive search for locally visible items such as demining.

A particle filter SLAM has been implemented, using independent Gaussian estimates for landmark positions, as done in [14]. The system executed in a controlled environment with identifiable artificial landmarks, which were visually sensed. Some minor improvements on the particle filter observation model were implemented, which were not found in the reviewed literature.

The robotic system consisted in a low cost differential robot built using the robotic kit Lego NXT v2.0 and a single board computer Fox Board as the main computing unit. The main sensor used was the Lego NXTCam V3. The use of this sensor presented a huge challenge due to its low color and pixel resolution.

The paper is organized as follows: Section II includes an overview of the studied previous work and a short comparison with the implemented solution. Section III includes an overview of the SLAM problem and its formal framework and the particle filter solution. Section IV describes the software implementation of the SLAM solution. Section V presents the experimental environment configuration and section VI shows the obtained results. Finally, section VII includes a brief discussion and conclusions.

II. Previous Work

Most solutions to the SLAM problem can be divided in three broad categories, Kalman filter SLAM, Graph SLAM and particle filters.

Kalman filters (KF)[11] use high-dimension Gaussian distributions to model the position of both the robot and discrete landmarks. Kalman filters may be used to update the belief about the world's state in an online fashion (i.e. one measurement at a time). Certain extensions of this approach, known as extended Kalman filters (EKF) include the use of non-linear observation or motion models, allowing a more precise modeling of the world. Welch and Bishop have provided a good introductory text to Kalman filters[19].

2D I-SLSJF system[9] uses an Information Filter, which is a variation of the Kalman filter approach. The main contribution of their work consists on the dimensionality reduction of the KF solution by applying a hierarchical approach to the problem. High dimensionality presents one of the biggest challenges in the SLAM problem, specially with KF solutions, as their computational complexity of every update is $O(M^3)$ where M is the number of landmarks in the system. Similarly, CEKF-SLAM system[7] extends Kalman filters including the concept of local and global landmarks, delaying update steps until the robot exits the current local region. Frese's et al. Treemap system tackles the dimensionality problem by using a hierarchical structure too[4], achieving a reported performance of being able to maintain a million landmarks. The present work addresses the issue of dimensionality by using the Rao-Blackwellized particle filter[5], in which landmark observations are treated as independent events, avoiding the growth in computational costs as the number of landmarks increase.

EKFMonocularSLAM[2] implements SLAM using a Kalman filter. They use a monocular camera as their only sensor, as does the presented work. Their main contribution consists of the inclusion of a RANSAC outlier filter, which they tightly integrate to the KF's predict and update steps.

Although this work includes some mechanisms to avoid landmark mismatching, the inclusion of such systems would be of great value, as discussed in Sec. VII.

The second class of SLAM systems is known as Graph SLAM, due to the way it model the problem as a soft-constraint graph. To do this, it transforms observations made by the robot into soft constraints represented as edges in a graph, while the nodes in this graph represent robot positions. After the whole graph has been built, well-known optimization techniques are applied to find the set of node positions that minimizes the measurement error under the soft constraints.

G2O is a library that solves the Graph SLAM optimization problem, given a constraint graph is provided[12]. This library is used to compare results against other Graph SLAM approaches, for it makes no optimization hypothesis. Thus, it returns the best, but computationally expensive, result. iSAM provides a least squares solution as well and works with a Bayesian network provided as input[10]. HOG-Man uses a hierarchical approach to optimize the graph, performing online local updates and seldom global updates[6]. Graph SLAM systems are not easily comparable to the present work, because the former usually performs off-line processing of the information, which is not applicable to our context of low computational power and on-line processing needs.

The last category are particle filter SLAM systems. Fast-SLAM uses a particle filter to estimate the robot's position over time (the robot's path) and one Kalman filter for every pair of landmarks and particles[14]. This division of work tackles the problem of dimensionality by solving localization and mapping separately. In addition, Fast-SLAM maximizes the likelihood of the movement measures, in a neighbor of the raw estimate, using the same observation information that is later on used in the update process. This technique allows Fast-SLAM to decrease the number of particles, which is important as the computational cost grows linearly with it.

DP-SLAM reduces the cost of copying the map of particle-filter based SLAMs that use occupancy grids as their map[3]. Only one map is maintained and a tree-like data structure is used to keep record of all updates done by all different particles. GridSLAM also uses an occupancy grid and performs scan-matching as a way to maximize the likelihood of odometry measurements[8], in the same way Fast-SLAM does. GMapping, addresses the problem of over-resampling the particle set and introduces a measurement of particle variance into the decision process of whether to resample or not[5].

This work combines some features from Fast-SLAM with GMapping selective resampling techniques. In addition, we introduce minor novelties in the measurement models and the computation of resampling weights, as will be discussed in Sec. IV.

III. Simultaneous Localization and Mapping

The SLAM problem can be described as the task of building a map of the environment and localizing the robot within that map, simultaneously. This problem can be formulated as a bayesian network linking three different variables over time t, as shown in Figure 1:

- Map description and robot position (x_t): this variable involves the robot's model of the world including itself. This variable is also referred as the hidden state, when casting the problem as a hidden Markov model. Sometimes, this variable is split into two components, the map m_t and the robot position x_t.
- Observations (z_t): this variable represents the observations performed by the robot. In the case of this work, it corresponds to the perceived location of an observed landmark.
- Self-movement (u_t): this variable includes the self-movement information provided by systems such as wheel odometry.

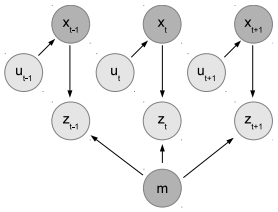

Fig. 1: Bayesian network for the SLAM problem. Dark grey nodes represent hidden variables, while light gray nodes represent observable ones.

Then, the problem can be casted as the optimization of the posterior probability of $x_{1:T}$, i.e. all maps and positions from the start (time 1) to the end (time T), given the observations $z_{1:T}$ and self-movement cues $u_{1:T}$, as described in Eq. 1.

$$argmax_{x_{1:T}} p(x_{1:t}|z_{1:t}, u_{1:t}) \tag{1}$$

On-line SLAM systems like Kalman or particle filters use the fact that given the position of the robot and the map at a given time, past and future observations z_t and movements u_t are independent from each other. Then, every hidden state x_t can be calculated iteratively by incorporating the last observation and movement information to the system. This follows the factorization shown in Eq. 2.

$$p(x_{1:t}|z_{1:t}, u_{1:t}) = p(x_1) \prod_{t=2}^{t=T} p(x_t|x_{t-1}, u_t) p(z_t|x_t) \tag{2}$$

The factorization in Eq. 2 elucidates two fundamental components of probabilistic SLAM systems. The first one being the motion model, which corresponds to the probability shown in Eq. 3. It models the probability distribution of the robot being in a position x_t, given it was in x_{t-1} and the perceived the self-motion information u_t.

$$p(x_t|x_{t-1}, u_t) \tag{3}$$

The observation model shown in Eq. 4 relates the position of the robot and map configuration x_t with the observation z_t. It describes the probability of getting a specific measurement at

a given state of the world. It is usually called the measurement likelihood.

$$p(z_t|x_t) \tag{4}$$

For a good introduction to bayesian networks see [1]. The reader may also review Thrun's book[18] for a detailed description of the SLAM problem in the bayesian networks framework.

IV. The Implemented SLAM

The SLAM system description has been divided into its three main components: its core module, its sensor model and its motion model. An additional section describes the calibration process.

A. Core module

This system implements the four main functionalities for the particle filter: particle bookkeep, map update after observations are made (i.e. update step), weight assignment and particle selection (i.e. resampling step) and the update of every particle position after a movement is performed (i.e. predict step).

1) Particles: Every particle keeps its own estimate of the robot position. Besides, each particle keeps its own estimate of every landmark in the map. Each landmark position estimate is kept using a 2-dimension Kalman filter. This filter maintains an estimate of the landmark's (x, y) coordinates and a covariance matrix, which may be interpreted as the accuracy of that estimate.

2) Predict : Upon every robot move, each particle's position estimate is updated to reflect that movement.

Each landmark position is represented in an absolute coordinate framework. Thus, there is no need to update the landmarks' position in this step.

3) Update: After every observation, the landmark's Kalman filter of every particle is updated to incorporate this new information. This update tends to move the estimate towards the measured position for that landmark. In addition, the covariance's main components are reduced during the update. As a consequence, the position estimate converges with each observation.

Particle weights are updated as well after every observation. The likelihood of the observation was used as the new weight.

The fact that mapping and weights assignment are done as separate steps corresponds to the Rao-Blackwellized aspect of the particle filter. Taking into account that all observations are independent of each other given the path of the robot is known, one can estimate just the position using the particle filter and leave the mapping part aside.

It is also important to notice that the sensor model parameter (see Sec. IV-B) is used twice in this step. First, it is used to represent the estimate of the observed landmark position. The mean and covariance dictated by the sensor model are passed as the measurement to the Kalman filter. Secondly, it is used to evaluate the likelihood of a an estimate, given the observed data.

4) Resampling: The particles' variance determines the need for resampling, as done in GMapping system[5]. Resampling is performed if it is greater than a system parameter.

If resampling is to be made, a new set of particles is selected with replacement from the old set. The probability of choosing each particle is proportional to its weight. A weighted roulette algorithm is used for this purpose.

B. Sensor Model

The information obtained from the vision module was used to detect artificial landmarks. A bayesian approach was used to identify landmarks and filter out the noise of each BLOB's reported position.

Once the image position of the landmark and its size was obtained, a trained polynomial interpolation system was used to derive the distance and angle, relative to the robot.

These data is used to compute an estimate of the landmark position, using trigonometry. The resulting point is considered the mean of a 2-dimensional estimated gaussian distribution of the landmark position. The covariance matrix is built as αI, where α is a fixed system parameter called sensor model covariance and I is the identity matrix.

The obtained gaussian estimation of the landmark is what is actually passed on to the SLAM system as a measurement. That way, the Kalman filters corresponding to each particle's current estimate for the landmark position can be updated using a linear observation matrix H, namely the identity matrix.

As was mentioned above, the sensor model parameter is used to determine the likelihood of new observations as well. When doing so, the likelihood distribution of sensing given the current map estimate, $p(z_t|m_t)$, is built as a convolution of two Gaussian distributions. The first one being the current estimate of the landmark position, as explained above. The other distribution used is a Gaussian with $(0, 0)$ as its mean and the sensor model covariance as its covariance matrix. The advantage of using this approach instead of just taking a symmetric distribution around the estimated position relies on the possibility to include the 'shape' of the uncertainty into account. Namely, suppose a landmark is known to be estimated with high uncertainty in the y axis, but with high accuracy in the x axis. Then, a measurement of its position that is far from the mean in the y direction is not as unlikely as a measurement that is equally far in the x direction.

C. Motion Model

The robot was always commanded to perform pure rotations or pure forward-backward motions. That is, no arc movements were ever performed. As a consequence, the implemented motion model was a simple one.

The motion commands issued to the control system were inputed to the SLAM system too. These commands were used to estimate the actual movement for the robot.

For every particle, during the predict process, the actual movement was calculated using these issued motion commands plus two additive sources of Gaussian noise, a rotation noise and a translational noise. These were white noise

sources, i.e. the noise distributions were centered around zero, and their variance was controlled by two corresponding system parameters.

D. Calibration using Genetic Algorithms

As has been shown this far, the SLAM system depends on four parameter: the resampling parameter, the sensor model parameter, and the translation and rotation motion model parameters.

The calibration for this four parameters was done using a genetic algorithm. First, a small dataset was recorded. The dataset included both landmark measurements and self-motion cues observed by the robot while it navigated randomly in the environment described in Sec. V. In addition, the actual robot position was recorded using a global camera.

Once the dataset had been recorded, the SLAM system could be executed off-line and the SLAM estimate of the robot position could be compared to the actual position recorded by the global camera. To get an actual estimate of the goodness of a set of parameters, the dataset was inputed to the SLAM algorithm using those parameters, over ten repetitions with different random seeds. The median of the average estimation error of the executions was taken as the 'fitness' value of that particular set of parameters.

V. Experimental Environment

The percepts, actions, goals and environment framework (PAGE)[17] is used here to describe the experimental environment.

A. Percepts

An NXTCam-v3 was included in the robot. This vision module preprocesses the image and outputs the location of a set of detected color BLOBs. The BLOBs are recognized by matching a set of preconfigured color ranges. The camera can capture 30 frames per second with a resolution of 144 by 88 pixels.

The detected BLOBs where passed on to the landmark detection system described in Section IV.

B. Actions

The robot task was to cover as much of the field as possible. The navigation algorithm used for this purpose is shown in Alg. 1. The routine getRobotPosition returns the estimated location of the robot's current position. The function getNextWaypoint return the next waypoint to be navigated. The waypoints are created so as to follow a route as shown in Figure 3. Once the robot reaches the end of that route, it turns back and travels the other way using the same algorithm.

C. Environment

The differential robot shown in Figure 2 was used for the experiments. It had two motored wheels and a third pivot wheel. It featured a Lego NXT brick and a Fox Board G20 as its processing units.

A small field of 2.4 by 2.0 meters was laid out. Six different landmarks were place in different places of this environment.

Algorithm 1 Navigation algorithm used to solve the covering task.

```
do forever:
        pos = getRobotPosition()
        wp = getNextWaypoint(pos)
        travelToWP(wp)
```

The landmarks consisted of a stack of three 5 cm edge colored cubes. The environment was surrounded with walls to prevent the background from interfering with the landmark recognition process. Figure 2 shows the configuration of the environment and the robot.

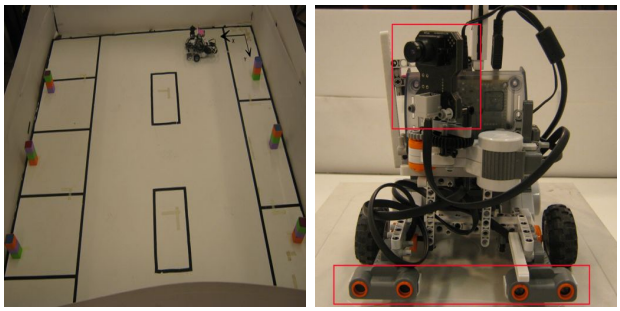

Fig. 2: On the left side, the experimental environment. The six color piles correspond to the artificial landmarks. The robot is located in the usual start position, in the upper right corner of the image. The black lines in the floor are just for visual guidance and debugging. On the right side the robotic platform, the red squares show the robot sensors.

D. Goals

The agent performance depended on how much terrain was covered in each run. The robot was able to perform covering for 100 minutes or until it exited the environment due to navigational errors.

Two main performance metrics were assessed, the amount of covered terrain and the error in the estimation of the position.

1) Covered terrain: In order to measure covering, the environment was divided into a fine grained grid. Then, the robot position was recorded continuously. At each instant, all cells in the grid within a square of 20 by 20 cm centered on the recorded position were marked as covered terrain. The covered terrain at each step was superimposed to get a total covering measurement. Figure 3 shows the covered terrain after the robot started the task.

Given that the robot was supposed to navigate only in an interior area of the entire field, a zone called useful zone was delimited. The useful zone corresponded to the terrain that an ideal robotic system should cover following the employed navigation algorithm. The useful zone is shown in Figure 3.

Three different coverage indicators were taken into consideration:

- Useful coverage: this metric counted the number of grid cells in the useful zone that were covered at least once.
- Wasted coverage: this metric counted the number of covered grid cells outside the useful zone.

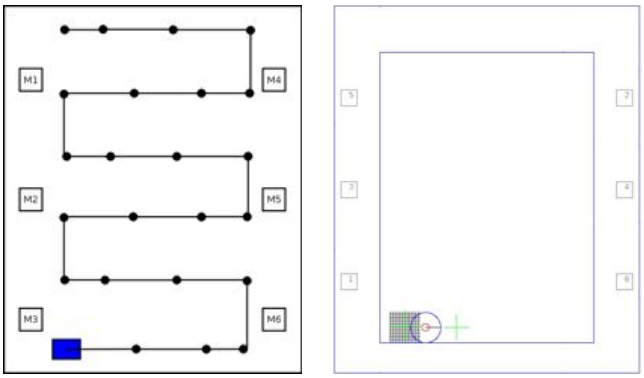

Fig. 3: On the left, the covering navigation pattern. The robot (blue rectangle) is at the starting position. The landmarks are shown as squares labeled M1..M6. The route is composed of a set of waypoints (black dots) laid out to cover the entire space. On the right, the covering done by the robot (dark grey) right after starting the task. The big circle represents the estimated position by the SLAM system. The right green crosses represent the starting waypoint (left) and the next goal waypoint (right). The inner big rectangle shows the zone to be covered.

- Total useful coverage: this metric assess the amount of visited cells considering that a same cell could be visited more than once.

2) Estimation error: The estimated position of each system was compared to the absolute position reported by the global camera at every waypoint. Then, the average of this errors was computed. The error measure used was RMS, which is calculated as shown in Eq. 5.

$$RMS = \sqrt{\frac{1}{n}\sum_{1}^{n}(x_{estimated} - x_{read})^2} \tag{5}$$

VI. Results

In order to be able to compare results, another system was implemented using odometry integration as the sole source of information for localization.

The experiment results are summarized in Table I. All coverage measurements are reported as a percentage of the total possible coverage. Total useful coverage figures are above 100% due to the fact that the robot covers the field several times.

TABLE I: Experiment results summary.

	Reference System		Implemented SLAM	
Metric	Mean	Variance	Mean	Variance
Useful Coverage	95.13%	3.61%	96.25%	1.17%
Wasted Coverage	24.39%	4.31%	20.27%	1.35%
Total Useful Coverage	4218.16%	2146.22%	9320.36%	1427.16%
RMS error	0.64 m	0.26 m	0.42 m	0.18 m
Executions with errors	5/5	-	0/5	-

The SLAM system performed slightly better for the useful and wasted coverage measures while the it notably outperformed the reference system for the total useful coverage metric. This was due to the fact that the reference system accumulated rotation errors which made it to exit the field early, while the SLAM system was able to perform for the full 100 minutes.

In order to further analyze this aspect the evolution of the useful coverage and total useful coverage over time was studied. Figure 4 shows both plots of the percentage of covered terrain over time. The coverage shown is the average of five runs for each system.

As it can be deduced from the plots, the reference system performed better at first, but the SLAM system outperformed it in the long run. Due to the vision module's noise, the SLAM system robot had to stop for several seconds in order to take the required number of images for the bayesian approach described in Sec. IV-B. As the other system did not use the vision module at all, it navigated the field faster. However, the SLAM system showed a more robust behavior, which allowed it to operate autonomously for the entire 100 minutes with no errors.

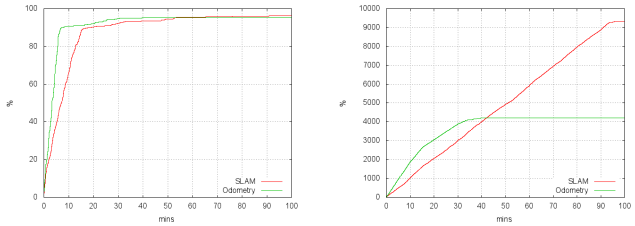

Fig. 4: Coverage measures as a function of time, useful coverage (right) and total useful coverage (left). Coverage is expressed as a percentage of the entire field to cover and time is expressed in minutes.

Although the position estimation error was better for the SLAM system than for the odometry one, it was lower than expected. Thus, a study of the evolution of the error was carried out, comparing the estimated rotation and position to the actual ones. Figure 5 shows the evolution of rotation and position error for one of the runs of the SLAM system. The data for this particular run suggests that system converges to a fixed difference between the estimated position and the real one. This may have been caused by a difference in the coordinate frameworks used by the SLAM system and the global camera.

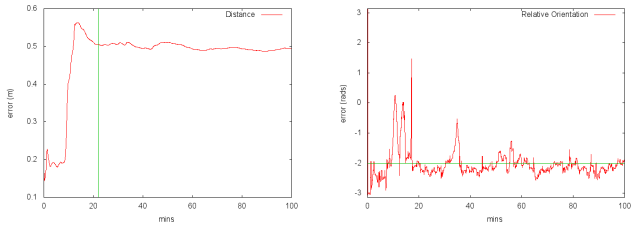

Fig. 5: Error evolution study for the third run of the SLAM system, distance (right) and relative angle (left). The green line shows where the estimation error stabilizes (right) and average angle value (left).

VII. Discussion And Conclusions

A particle filter SLAM has been implemented fully embedded and some novel ways of calculating a measurement likelihood has been introduced. A genetic algorithm calibration approach was used to prevent parameter over-fitting and obtain more generalizable results. Finally, the developed SLAM system was used to implement a general field coverage task and showed better performance than the reference system.

Results show a competitive performance in both useful coverage and wasted coverage metrics. However, they do not show the improvement level expected by the authors. This is attributed to the fact that landmarks were only seen by the robot within a short range. This prevented continuos corrections to the robot's estimated positions and led to navigation errors. The total useful coverage metric, however, showed the advantage of using the implemented SLAM in the long run. The SLAM corrections allow the robot to operate without leaving the environment for a longer time, thus enabling a better coverage. Moreover, the improvement in performance is greater if one takes into account the fact that the odometry system moves faster than the SLAM system. Namely, if the vision module is replaced with one that allows to sense while moving or with shorter stop times, the difference in total useful coverage would be even greater.

Further studies should be carried out to elucidate the reason behind the errors in the estimated position against the one obtained with the global camera. More data should be collected and analyzed in order to determine if the position estimation error found is due to a difference in the coordinate frameworks. Given the SLAM system's map is totally independent from the global camera system's map, it would be expected for them to end up with similar maps but with shifted coordinate systems.

There are several lines of possible improvements to the present system. Replacing the Lego vision module with a state of the art camera and an image processing software such as OpenCV is considered the most urgent one. The addition of automatic landmark detection systems such as SIFT[13] or FAST[16] would help scale the system to real environments.

Apart from that, additional data structures should be implemented to enable for the maintenance of a large number of particles, as it is done in Thrun's work[14].

Acknowledgments

The present work has been supported in part by the "Agencia Nacional de Investigacion e Innovación (ANII) - Uruguay". This work is funded by NSF IIS Robust Intelligence research collaboration grant #1117303 at USF and U. Arizona entitled "Investigations of the Role of Dorsal versus Ventral Place and Grid Cells during Multi-Scale Spatial Navigation in Rats and Robots".

References

[1] David Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, March 2012.

[2] Javier Civera and Andrew J Davison. 1-point RANSAC for EKF filtering. application to real-time structure from motion and visual odometry. *Journal of Field Robotics*, 27(5):609–631, 2010.

[3] A. Eliazar and R. Parr. DP-SLAM 2.0. In *IEEE International Conference on Robotics and Automation (ICRA)*, April 2004.

[4] Udo Frese. Closing a million-landmarks loop. In *In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing. submitted*, page 5032–5039, 2006.

[5] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *Robotics, IEEE Transactions on*, 23(1):34–46, February 2007.

[6] Giorgio Grisetti, Rainer Kümmerle, Cyrill Stachniss, Udo Frese, and Christoph Hertzberg. Hierarchical optimization on manifolds for online 2D and 3D mapping. *Robotics and Automation (ICRA)*, 2010.

[7] Jose Guivant and Eduardo Nebot. Optimization of the simultaneous localization and map building algorithm for real time implementation. *IEEE Transactions on Robotics and Automation*, 17:242–257, 2001.

[8] D Hahnel, W Burgard, D Fox, and S Thrun. A highly efficient FastSLAM algorithm for generating cyclic maps of large-scale environments from raw laser range measurements. 2003.

[9] Shoudong Huang, Zhan Wang, Gamini Dissanayake, and Udo Frese. Iterated SLSJF: a sparse local submap joining algorithm with improved consistency. In *Proceedings of the Australasian Conference on Robotics and Automation, Canberaa*, 2008.

[10] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: incremental smoothing and mapping. *Robotics, IEEE Transactions on*, 24(6):1365 –1378, 2008.

[11] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[12] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. G2o: A general framework for graph optimization. In *2011 IEEE International Conference on Robotics and Automation (ICRA)*, page 3607–3613, 2011.

[13] David G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, ICCV '99, page 1150–, Washington, DC, USA, 1999. IEEE Computer Society.

[14] Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *In Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, page 1151–1156, 2003.

[15] Robin Murphy. *An Introduction to AI Robotics*. MIT Press, 2000.

[16] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. *Computer Vision – ECCV*, 2006.

[17] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach (3rd Edition)*. Prentice Hall, 3 edition, December 2009.

[18] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. The MIT Press, 2005. Published: Hardcover.

[19] Greg Welch and Gary Bishop. An introduction to the kalman filter. Technical report, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, 1995.