Simultaneous localization and mapping for a mobile robot with laser scanner

Çilli, C.

Award date:
2009

Can Çilli

Simultaneous Localization and Mapping for a Mobile Robot with Laser Scanner

DCT-2009.090 08.09.2009

Master Thesis

Title

Simultaneous Localization and Mapping for a Mobile Robot with Laser Scanner

Author

Can Çilli
Systems & Control Master Student
Technical University of Eindhoven, Mechanical Engineering Department

Date

September 8, 2009

Contents

Acknowledgement

This project is conducted in Philips Applied Technologies (Apptech). The author thanks to Harry Broers for his mentorship and friendship throughout the project period.

Graduation Committee

- Prof.dr.ir. Maarten Steinbuch
- Dr.ir. Marinus J.G. van de Molengraft
- Dr.ir. Dragan Kostić
- ir. Harry Broers (Philips Applied Technologies)

A Line Feature Based, Hybrid Scan Matching Algorithm for the Indoor SLAM Problem

Abstract

Simultaneous localization and mapping is the problem of building a map of an unknown environment with a robot, while localizing the robot on the map constructed so far. There are two dominant approaches for the solution of SLAM problem: probabilistic and scan matching. Probabilistic methods treat the problem as maximum likelihood estimation. Scan matching is computing the amount of rotation and translation between a reference and a current scan. We present a novel, two-stage, feature-to-feature scan matching method as a solution to the indoor SLAM problem using a laser range finder. Each of the stages uses an Extended Kalman Filter to estimate the robot pose. Data association is done on a grid-map, the cells of which contain links to the features, in order to speed up the process. We tested the method in a simulated environment, and obtained promising results.

I. Introduction and Related Work

Simultaneous localization and mapping (SLAM) is the problem of building a map with a mobile robot, based on pose estimates and environment perception sensors, in an unknown environment while simultaneously localizing the robot on the map constructed so far. The robot has to precisely know its location on the map, to correctly integrate new measurements to the map. Otherwise, any error in localization will lead to an incorrect map, and in the next step an incorrect map will cause incorrect localization and so on. By its nature the SLAM problem is a cyclic problem like the chicken-egg problem [1]. According to Thrun [2], the SLAM problem is composed of four challenging components:

- Sensors. Odometry errors accumulate over time, and they affect the way future measurements are interpreted. Environmental perception sensor measurements vary due to lighting or surface properties, which may not return consistent values.
- High dimensionality. A detailed 2D floor plan often requires thousands of entities. Time requirements often demand that internal models must be simple and easily accessible.
- Data association. It is determining if the sensor measurements, taken at different points in time, correspond to the same physical object in the world. When mapping a large cyclic environment, at the moment

of closing the cycle, the robot has to find out where it is relative to previously built map. This is not trivial because of the accumulated pose error.

- Dynamic environment. The dynamism of the robot's environment is another reason of inconsistent sensor measurements.

There are many types of environment perception sensors: camera, sonar, IR, laser range finder, etc. The choice of the environmental sensor depends highly on purpose of mapping and accuracy requirements. Cameras provide a huge amount of information, which is not cost efficient to process and to interpret in real-time. Sonar and IR range finders were a breakthrough in the SLAM field. They are low-cost and fast, but they are not very precise. Laser range finders (laser scanners) are the most popular sensors in the field. They are accurate, reliable and time-efficient, but they are expensive.

There are two dominant approaches to solve SLAM problems [3]. One is the probabilistic approach, in which localization is treated as a maximum likelihood estimation problem. Extended Kalman Filter (EKF) SLAM and FastSLAM [4] are the most popular probabilistic approaches. In EKF-SLAM, uncertainties and correlations are represented with a Gaussian function, and are propagated using an extended Kalman filter. FastSLAM is a particle filter based solution, which represents pose estimate as a discontinuous approximation of Gaussian distribution. The other approach to the SLAM problem is called scan matching, which means finding a maximum overlap of scans through a process of translation and rotation. Probabilistic approaches are able to cope with large errors, but they are computationally inefficient. Scan matching methods are confined to small perturbations of sensor scans, and they cannot recover from large errors, while they are computationally efficient. A mix of both approaches is called hybrid method. Hybrid methods are efficient, accurate and robust to noise [5].

Scan matching is utilized to compute the rigid transformation parameters between a reference scan and a current scan. A generic scan matching method is composed of two steps which are data association and error minimization. Scan matching approaches can be categorized based on their association method: point-to-feature (P2F), point-to-point (P2P) and feature-to-feature (F2F). In P2F matching raw measurement points are matched to some line features in the reference. This method was originated by Cox [6]. Gonzalez [7] extended the idea to polylines, by using an auxiliary grid-map to speed up association. Iterative Closest Point (ICP) [8], Iterative Matching Range Point (IMRP) [9], and Iterative Dual Correspondence (IDC) [9] are the most popular methods in P2P matching category. In P2P matching approaches, the number of scan points is approximately two orders of magnitude larger than in feature based methods. In these methods, the association is performed as a nearest-neighbor search. Thus, computational complexity of P2P methods is not suitable for real-time applications [10]. F2F matching methods have the

shortest run time, since hundreds of range points are reduced to a few dozens of features. For indoor applications, lines, circles, corners and other simple geometrical features are rich and easy to detect. Bailey [11] uses distance and angle relations of features to generate a graph of relations, and then transforms the association problem into a problem of finding the maximum common subgraph. Shaffer [12] uses distance and angle relations of lines and corners, but association is done between minimum-error features. Wolter [13] matches polylines to polylines. Matching is done by manipulating the features to minimize a penalty function. In the Anchor Point Relationship (APR) method [14], two graphs which are generated from distances between point features are matched. In this method, association and error minimization are done in one step.

The fundamental problem of scan matching methods is that there is no good way to estimate the quality of the position estimate, which prohibits an effective fusion with other position estimates from other sensors, and thus, the migration to a hybrid method. Lu [9] assumes white Gaussian noise in x, y coordinates of the scan points, which results in over-optimistic error estimates in situations where there is lack of information in some directions, e.g. corridors. Bengtsson [15] estimated the pose covariance matrix from the Hessian of the scan matching error function, which does not take the uncertainties of the scan points into account but results in better error estimation. Once a realistic error estimate is achieved, it is possible to combine scan matching with any probabilistic method, such as EKF-SLAM [10], [16], [17], FastSLAM [18], etc. In our method, we estimate the pose covariance matrix from the scan matching error and integrate it to represent the accumulated error.

This paper is organized as follows: Our method is summarized in Section II. The first stage of the method which includes feature extraction, data association and error minimization, is described in Section III. Mapping process is discussed in Section IV. The results of the simulations are reported in Section V. Finally, in Section VI, conclusions and future work are presented.

II. Our Method

We focus on the indoor SLAM problem. An indoor environment can easily be described by geometric features, e.g. lines, circles, etc. We were aiming to end up with an efficient SLAM algorithm, such that it must be real-time manageable with low-cost hardware, and sufficiently accurate to compensate variations in a domestic environment. Considering time-complexity of probabilistic mapping methods, feature-based mapping is a better solution. Furthermore, our environment is limited to geometric features,

mostly lines, thus feature-to-feature matching would be more efficient than other scan matching approaches.

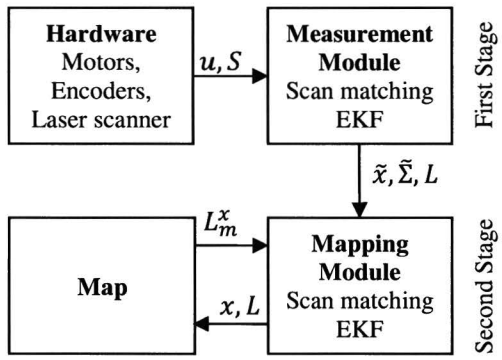


Fig.1. Layout of our approach. u stands for control signal and/or odometry measurement, S is laser scan, $\tilde{x}$ is the robot pose estimate, x is the final robot pose, $\tilde{\Sigma}$ is covariance matrix, L is list of features, L_m^x is list of visible map features at pose x.

We developed a two-stage hybrid method, both stages of which stages are using an identical scan matching approach with Kalman filtering. A layout of the approach is shown in Fig.1. The first stage matches the extracted features in the kth scan (S_k) to the ones in the $(k-1)$th scan (S_{k-1}) to determine the amount of displacement between them. An EKF is used to predict the displacement from control signals and/or odometry measurements, and to update it with the scan matching result. The first stage passes an improved estimate of the robot pose and a covariance matrix, to the second stage. Therefore the first stage behaves like an improved odometry sensor. The second stage matches the features in S_k to the map. Again, an EKF is used to predict the robot pose from the output of first stage, and to update it from the scan matching result. Then, the map is updated with S_k at the best pose estimate.

In order to do scan matching, the features in the scans have to be associated in between. We limit the nearest-neighbor search space by using a grid-map. The cells of the grid-map contain links to the features. A grid-map is generated for each scan to be matched. We translate the uncertainty ellipse of the robot pose (extracted from the covariance matrix) to a point which belongs to a line feature. The translation generates an area around the known position of the point. That area corresponds to some cells on the grid-map. To associate, we look at those cells for some features on the grid-map of the other scan. In other words, we detach the uncertainty of the features from the uncertainty of the robot pose, and account for all uncertainties in the robot pose.

III. Measurement Module

The measurement module takes in control input/odometry measurements and raw laser scanner data; it extracts features from the raw scan data, then it calculates the amount of translation and rotation to fit the new scan onto the previous scan. The robot pose state of the internal EKF is updated with the resulting correction movement, and an improved pose estimate is passed to the mapping module together with the list of features and with the correspondences among them. Fig.2. shows the layout of the measurement module.

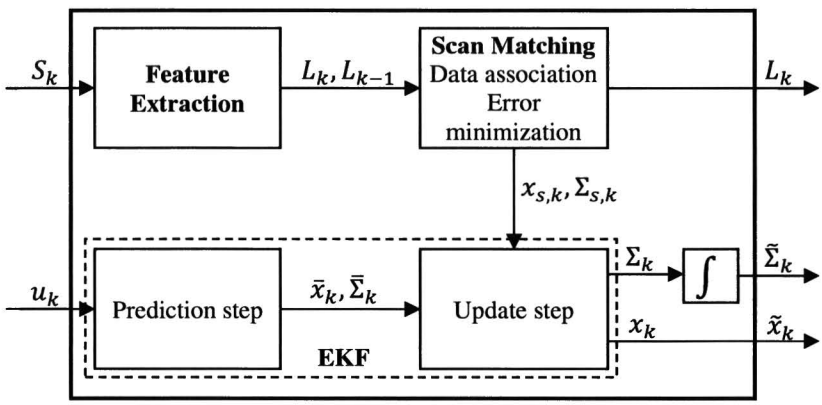


Fig.2. Layout of the measurement module. Inputs are the raw scan (S_k), the control input/odometry data (u_k). Outputs are the latest scan (L_k), the integrated pose covariance ($\bar{\Sigma}_k$) and the pose estimate ($\tilde{x}_k$). Scan matching block outputs the displacement ($x_{s,k}$) between two scans (L_k, L_{k-1}) together with its uncertainty ($\Sigma_{s,k}$).

Control input or odometry is used in the prediction step of the EKF. In case both are provided, the most uncertain one can be used for prediction and the other can be used as update to the EKF together with $x_{s,k}$. Both inputs require a probabilistic motion model and a probabilistic observation model to use with the EKF.

The module updates the pose estimate at arrival of each control input, but it outputs a new pose estimate only when a new laser scan arrives. The measurement module does not have feedback on the long term (map-wise feedback). Thus, the pose error is still subject to increase unboundedly as time proceeds, whereas the covariance of EKF becomes confident after each scan matching operation. It cannot represent the accumulating behavior of the error. This is a contradicting situation. To represent the accumulating error of the pose estimate, we output the sum of all previous covariance matrices.

A. Feature Extraction

The feature extraction step takes a 360° range scan $S = (r_i, \phi_i)_{i=1,\dots,N}$ as input, where r is the range value, ϕ is the bearing of the measurement wrt the robot coordinate frame, and N is the number of data points; and outputs a list of line features (L) together with a grid-map (G), the cells of which include the ID of the line feature passing through them. A line feature $\ell = (a, b, \phi_{e,1}, \phi_{e,2}, \sigma_\perp, \sigma_\angle, t_l, t_{e,1}, t_{e,2})$ is defined by line formula: $y = ax +$

b. The endpoints 1 and 2 are determined by the bearings $\phi_{e,1}, \phi_{e,2}$, i.e. the jth endpoint is the intersection point of the fitted line with the $y = \tan(\phi_{e,j})x$ line. Uncertainty of the fitted line is represented by $\sigma_\perp, \sigma_\angle$ which are the standard deviations of the perpendicular distance to the line from the origin and from the orientation of the line, respectively. The parameters $t_l, t_{e,1}, t_{e,2}$ are some type definitions for the feature. The reason of defining lines in parametric form, instead of with two end points, is that the matching operation is done more efficiently in parametric form.

We have used two types of features (t_l): lines and polylines. The lines represent structural objects like walls, couches etc. of which two sides are intersecting with an angle close to 90°; whereas the polylines represent non-flat shaped objects. According to the positions of the features in the scan, we assign three types to the edges of the features (t_e): definite, indefinite and incomplete. A definite edge corresponds to a corner of a structured object, i.e. a 90° corner; whereas an indefinite edge corresponds to a corner of a polyline which does not have an angle close to 90°. Indefinite edges might occur due to approximation of a non-flat surface with some lines. Definite and indefinite edges are regarded as complete edge. Incomplete edges are assigned when some part of a line is occluded by another one or is out of range of the laser scanner.

Feature extraction begins with segmentation of scan S based on jump edges which are detected basically by a jump of consecutive range readings. These jumps are indicators of different features. Then, we fit lines onto the data points. A common way to fit lines is the least squares method which is tended to return erroneous results in the presence of outliers. In order to improve the quality of the fits, we employed the Random Sample Consensus (RANSAC) [19] method, which is an iterative method used for model fitting, hypothesis testing and outlier removal, on the former segmentation results. For each segment, the maximum error limit of RANSAC is updated based on the sensor model. RANSAC returns the line parameters (a, b), the inliers and their errors (ϵ). $\phi_{e,1}, \phi_{e,2}$ are the bearings of the first and the last inliers. $\sigma_\perp$ is equal to the standard deviation of the errors of the inliers. For computation of $\sigma_\angle$, the analytical method presented in [20] is overcomplicated and expensive. For that reason we chose to approximate $\sigma_\angle$ as in (1) where $\overrightarrow{c_j} \in \mathbb{R}^2$ is the vector from origin to the jth inlier point among the N_i inliers of the segment.

$$\sigma_\angle = \tan^{-1}\left(\frac{max|\epsilon|}{0.5 \cdot \left\|\overrightarrow{c_1} - \overrightarrow{c_{N_i}}\right\|}\right) \tag{1}$$

After fitting lines to the measurements, the lines are projected on a local grid-map. The grid-map is of the size of the maximum range of the sensor and only used for data association between the current and previous scans. Fig.3. shows the input (a) and outputs (b,c) of the feature extraction.

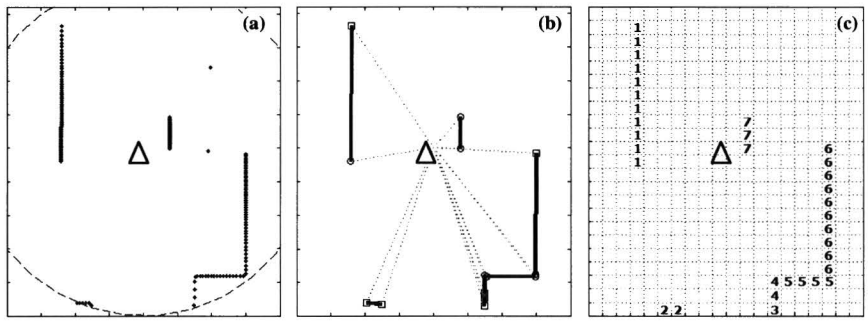


Fig.3. (a) Raw measurement points. Maximum range circle is shown by dash-line. (b) Extracted features. Circle and square edges are for complete and incomplete edges, respectively. (c) Grid-map, composed of links to features, shown by link numbers. The triangle is the robot.

B. Scan Matching

Scan matching is composed of two steps: Data association and error minimization.

In the data association step, we try to associate features in the current scan to the ones in the reference. Most of the studies in literature [6], [8], [9] use the nearest-neighbor algorithm to find the correct. The nearest-neighbor search algorithm is an expensive method and becomes intractable as the number of features increases. To increase the efficiency of the search method, we could constrain the search region in $\mathbb{R}^2$, if we had known where the features were, without checking all of them. Thus, we propose a grid (discrete) map, the cells of which contain links to the features. By doing so, we search for a feature only at the maximum likely region. The maximum likely region is extracted from the uncertainty of the robot pose. The uncertainty of the robot pose is transfigured like an extruded ellipse as it is shown Fig.4.

We use three points to determine where a feature is. We take two endpoints and the midpoint of the lines and check the neighborhood of the corresponding points in the other map to see which feature is there. The neighborhood of a point is equal to the shaded region shown in Fig.4. Once a feature is found in the neighborhood, we increase the number of correspondences between those two features, which are stored in a $n \times m$ correspondence matrix where n and m are the amount of features in reference and current scans, respectively. At the end of the search, we conclude that the feature couple (i, j) is correspondent, only if the (i, j) element of the correspondence matrix is the maximum in ith row and in jth column at the same time. This heuristic approach does not eliminate one-to-multiple feature correspondences.

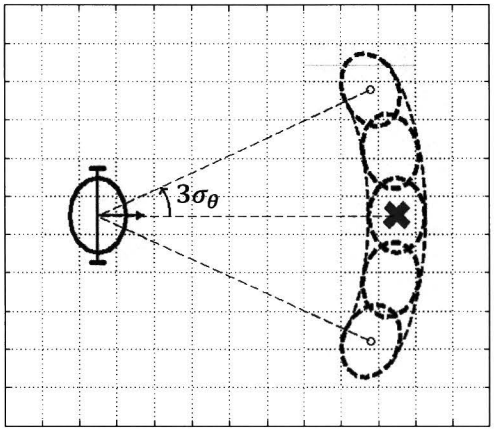


Fig.4. Robot uncertainty ellipse, shown on left, is translated onto a point feature, shown by cross. The ellipse is extruded for an angle of $3\sigma_\theta$ where σ_θ^2 is the variance of the robot orientation. Shaded cells show the discretization of the maximum likely region.

Next, we extract relations between the corresponding features. A relation is defined as a geometric property of a feature. The relation of a feature in the current scan is compared to the relation of the corresponding feature in the reference scan. The result of the comparison is called relation error and is used in the error minimization. We use four types of relations: orientation (θ), perpendicular distance from the origin (d), coordinates of feature end points (x_e, y_e). The first two relations are defined for all features, whereas the end point relations are only defined for features with definite edges. The relations have different weights on the error minimization, based on the variances of the features. The weight of a relation, $w_{i,j}$, between feature couple (i,j) is defined in (2).

$$w_{i,j} = 1/\max\left(\sigma_i^2; \sigma_j^2\right) \tag{2}$$

Weights of the perpendicular distance and orientation relations are directly computed from the variances, $\sigma_\perp^2, \sigma_\angle^2$. Weights of the end point coordinates are computed from σ_x^2, σ_y^2, which are obtained from $\sigma_\perp^2, \sigma_\angle^2$ via the following transformation.

$$\begin{bmatrix}\sigma_x^2 \\ \sigma_y^2\end{bmatrix} = J_{xy} \cdot \begin{bmatrix}\sigma_\perp^2 & 0 \\ 0 & \sigma_\angle^2\end{bmatrix} \cdot J_{xy}^T \tag{3}$$

$$J_{xy} = \begin{bmatrix}\partial f_{xe}/\partial d & \partial f_{xe}/\partial \phi_\perp \\ \partial f_{ye}/\partial d & \partial f_{ye}/\partial \phi_\perp\end{bmatrix} \tag{4}$$

where $f_{xe}(d, \phi_\perp, \phi_e)$, $f_{ye}(d, \phi_\perp, \phi_e)$ are functions returning the x and y coordinates of the edge of the line feature, at angle ϕ_e, transformed from polar coordinate representation of the line feature.

Lastly, the errors of the relations are attempted to be minimized in a weighted least-squares sense by using the Gauss-Newton method, given in (5), (6). For this method to converge to a correct solution, it must be guaranteed that the initial orientation error is less than 90°, which is well within the accuracy of dead reckoning [12].

$$\Delta\xi = -(J^T W J)^{-1} \cdot J^T W \cdot e \tag{5}$$

$$J = \begin{bmatrix} \partial f_{r1}/\partial x & \partial f_{r1}/\partial y & \partial f_{r1}/\partial \theta \\ \cdots & \cdots & \cdots \\ \partial f_{rn}/\partial x & \partial f_{rn}/\partial y & \partial f_{rn}/\partial \theta \end{bmatrix} \tag{6}$$

where e is the vector of relation errors, $\Delta\xi = (\Delta x, \Delta y, \Delta\theta)^T$ is the difference vector between the transformation parameters on successive iterations, W is the diagonal weight matrix and J is the Jacobian of size $n \times 3$. $f_{ri}(\Delta\xi)$ is a function of transformation parameters, returning the error of the particular type of relation i.

Equation (5) is supposed to be solved iteratively for $\Delta\xi$, until the magnitude of relation errors is less than some tolerance. However, we observed that iterating more than once does not provide additional minimization of the error. Usually, a feature does not exactly match to the associated feature due to outlying scan points, even if two scans are obtained at the same position. Moreover, errors of incorrectly-associated features cannot be minimized. These constant errors badly influence the result of error minimization. Thus, after the first iteration, we eliminate relations with relatively large error, and repeat the iteration once more. Eventually, there will always be a non-zero error in the estimated transformation. We need to predict how large the error is. Therefore, we compute the uncertainty of the resulting estimate in the form of a covariance matrix, $\Sigma_{\Delta\xi}$.

$$\Sigma_{\Delta\xi} = J_e \bar{W} E J_e^T \tag{7}$$

$$J_e = \begin{bmatrix} \frac{\partial f_{e1}}{\partial e_1} & \cdots & \frac{\partial f_{en}}{\partial e_n} \end{bmatrix} \tag{8}$$

where $\bar{W}$ is the normalized weight matrix, E is the diagonal matrix of squared errors and $f_{ei}(e_i)$ is the transformation function of the coordinate frame based on the resulting error regarding to the ith relation.

IV. Mapping Module

The mapping module matches the latest scan to the current map and updates the position of the robot accordingly. The mapping module includes the same scan matching process, explained in Section III.2, but a separate EKF than the measurement module. The output of the measurement module is used as a prediction to the EKF, and the result of the scan matching is used as

the update to EKF. Thus, the actual pose estimate of the robot is based on the pose in this module.

A dual map representation, which is composed of a list of features (Λ) and a grid-map (Γ), is used for the global map, as well. A line feature of the map, $\lambda = (e_1(x,y), e_2(x,y), \sigma_\perp, \sigma_\angle, \eta, LSQ)$, is defined by two end points (e_1, e_2) in global coordinate frame, uncertainties of the line $(\sigma_\perp, \sigma_\angle)$, a list of its neighbor features (η), and some parameters $(LSQ = \{\Sigma x, \Sigma y, \Sigma xy, \Sigma x^2\})$ required for incremental least squares method. The grid-map is used for efficient feature association, but it also enables the path-planner algorithms to optimize the travel distance by avoiding obstacles.

To represent a line feature, it is sufficient to describe two edges. However, due to limited maximum range of laser scanners, or due to occlusion, it is not always possible to fully-observe a feature. Most of the time, we cannot see even any edges for long features. Feature extraction algorithm can distinguish between incomplete and complete (definite/indefinite) edges. When two complete edges are observed, the feature is directly added to the map, by averaging previously observed edge coordinates. If one or both of the edges are not observed yet, we use least-squares line fitting on the previously seen incomplete edges. If one complete edge is already observed, then we try to fit a line while assuring the line passes through the edge. The uncertainties $\sigma_\perp, \sigma_\angle$ are computed as in (9) and (10), respectively.

$$\sigma_\perp = \max\left(\|\vec{v}_{e1}\|; \|\vec{v}_{e2}\|; \sigma_{LSQ}\right) \tag{9}$$

$$\sigma_\angle = \tan^{-1}\left(\frac{\sigma_\perp}{0.5 \cdot L}\right) \tag{10}$$

where $\vec{v}_{ej}$ is a vector which is projection of the principal components of the covariance of the jth edge, onto the line perpendicular to the feature, σ_{LSQ} is the standard deviation of the residuals of the least-squares which is used unless both edges are observed, and L is the length of the feature.

Having the features know their neighbors reduces the search time for reconstruction of a scan for matching. When a new feature is observed, it is added to the neighbor lists of all features which are present in the same scan. At the time of reconstructing a scan for matching, we start from the closest feature to the robot pose and search the neighbor features which are in the visible range. At the moment of loop-closing, since the robot will not know about the neighborhood relations among new and old features, in scan reconstruction the old features will not show up. Thus the new features, corresponding to the old features, will remain unassociated. Nevertheless, in the map update step, we use the grid-map to associate old features to the unassociated new features, and update the map accordingly. By doing so, the

neighborhood relations among the old and new features are updated for the next iteration.

V. Simulations

The aim of the simulations was to see how good the robot could generate the map of the environment while not any particular exploration strategy is employed, and how the robot would localize itself when it travels to previously visited areas. We did not test the complete approach on a mobile robot in a real environment but in a Player/Stage simulation environment [21]. We worked offline in MATLAB® on the data set generated by Player/Stage. A laser scanner module with $360°$ view angle and maximum range of 2.5 meters is used as the perception sensor. The odometry error is modeled as 10% of the travelled distance. We constructed two rooms of sizes $12m \times 8m$ and $7m \times 11m$. Some rectangular obstacles (like cupboards, couches etc.) are used to include visual occlusion.

Fig.5. shows the map generated from the first data set. Fig.6. shows the maps generated from poor odometry measurements. Fig.7. shows the comparison of the position errors of different stages. In this simulation, we used a grid-cell size of $0.1m$. An error less than the grid size would result in the exact map association. In other words, the map would be globally consistent. Otherwise, partial consistency would occur.

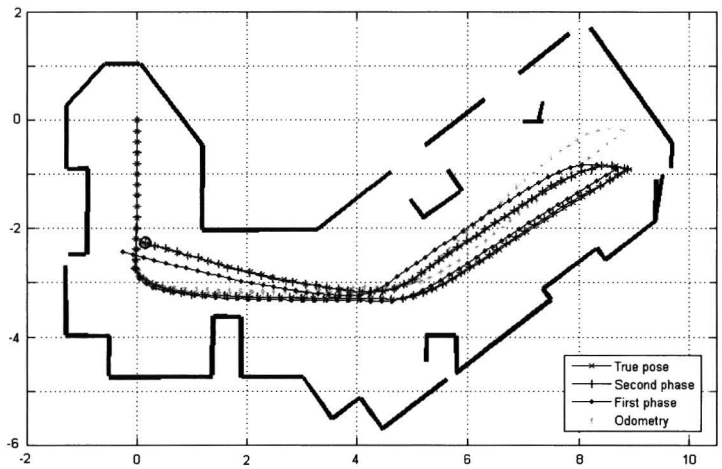


Fig.5. Generated map and the path of the robot. Robot starts from (0,0). The output of the second phase is almost always coinciding with the true pose, whereas pure odometry and the output of the first phase deviate.

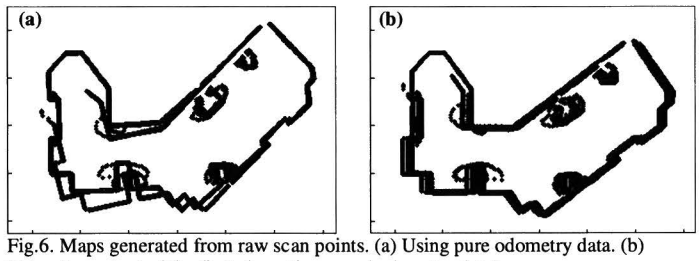


Fig.6. Maps generated from raw scan points. (a) Using pure odometry data. (b) Using the output of the first phase (improved odometry data).

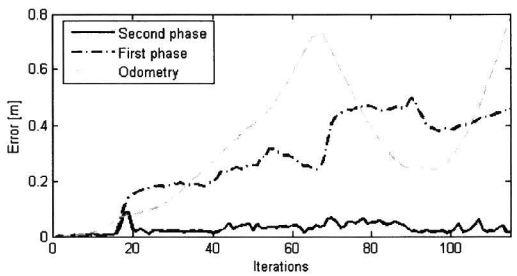


Fig.7. Position error throughout the trip. Error of the second phase is always less than the grid size, 0.1m.

The second room has a loop inside, thus we could test the loop-closing performance of our method. The robot travelled in counter-clockwise direction, starting from the lower left corner. After making one complete cycle, it stopped at lower right corner. We used a grid-cell size of $0.1m$. Fig.8. shows the map generated. Fig.9. shows the maps generated from poor odometry measurements. Fig.10. shows the localization error throughout the trip.

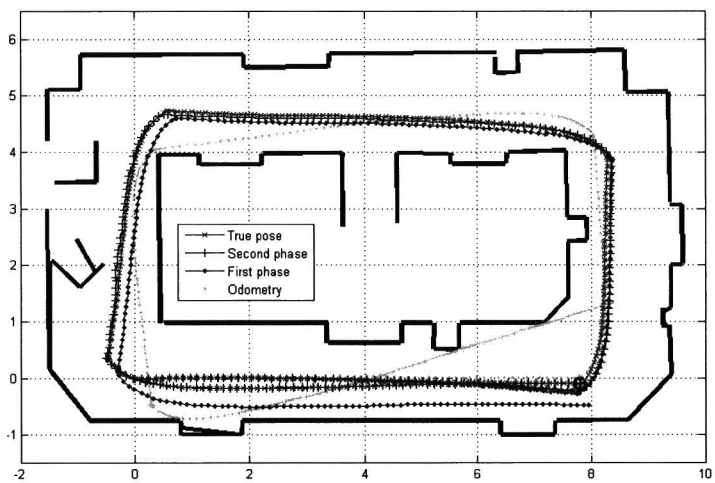


Fig.8. Generated map and the path of the robot. Robot starts from (0,0) and makes one turn around the room.

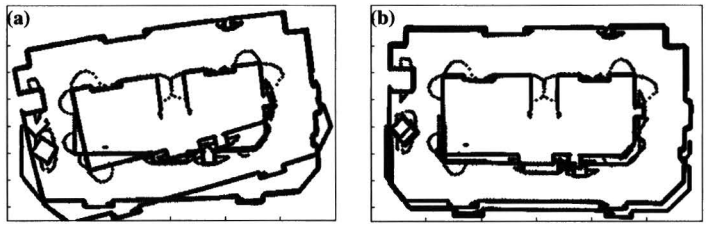


Fig.9. Maps generated from raw scan points. (a) Using pure odometry data. (b) Using the output of the first phase (improved odometry data).

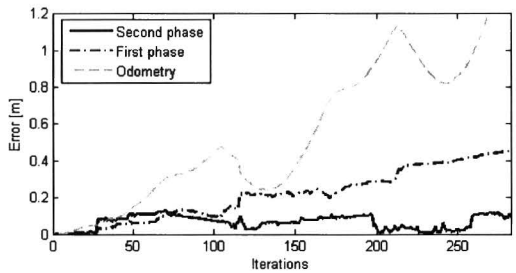


Fig.10. Position error throughout the trip. Grid size is 0.1m.

As shown in Fig.8., the predicted robot pose (output of the second stage) is always less than the grid-cell size, which implies the map is globally consistent. Output of the first stage and pure odometry errors grows unboundedly. In both of the simulations, the magnitude of orientation error for the second stage was always less than 0.5 degree, whereas the magnitude of orientation error for the first stage could grow up to 5 degrees while the robot rotates. This is the reason why the pose estimates of the two stages differ. The second stage compensates the orientation error.

Sometimes, the localization error exceeded the grid-cell size, causing the generated map to be inconsistent with the true map. But still the robot was able to close the loop. When the robot travelled to inconsistent regions again, it could localize itself without losing track due to inconsistent laser scans.

VI. Conclusions and Future Work

In this paper, we presented a novel SLAM algorithm, based on feature-to-feature scan matching. The first stage is regarded as an improved odometry sensor, which is subject to unbounded error. The second stage is the mapping module, which matches the current scan to the map. Extended Kalman filter is used at each stage to fuse sensor measurements. We observed that the algorithm is able to maintain the localization error sufficiently small, to close loops in the environment.

Uncertainties of map features are not stored in the Kalman filter state. Thus, at the moment of loop-closing, error correction cannot be back propagated to the features. A drawback of this result is that the map may not be globally consistent. On the other hand, the time-complexity of the

algorithm is kept an order of magnitude low and constant. One possible solution for the map inconsistency problem is combining the second stage with a particle filter. The robot pose uncertainty will be represented with particles. Each particle will hold a map, which is assumed to be correct. At the moment of loop-closing, the particle with minimum error survives. Moreover, for particles, which are in the same cell, data association will be the same, which will speed up the process of scan matching for each particle. In this way, we foresee that a globally consistent map can be obtained.

A grid-map is used to speed up the data association. However, since we only have a non-optimized MATLAB implementation of the method, we cannot provide any performance measures. In the future, software will be optimized and re-implemented in C/C++ for benchmarking.

References

[1] S. Thrun, W. Burgard and D. Fox, *Probabilistic Robotics*. Cambridge, MA: MIT Press, 2005.

[2] S. Thrun, "Robotic mapping: A survey," in G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002.

[3] C. Wang and C. Thorpe, "Simultaneous localization and mapping with detection and tracking of moving objects," *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2002.

[4] M. Montemerlo, S. Thurn, D. Koller and B. Wegbreit, "FastSLAM: A factored solution to the simultaneous localization and mapping problem," *Proc. AAAI National Conf. on Artificial Intelligence*, Edmonton, Canada, 2002.

[5] J. Gutmann, W. Burgard, D. Fox and K. Konolige, "An experimental comparison of localization methods," *Proc. Int. Conf. Intelligent Robots and Systems (IROS)*, Victoria, BC, Canada, 1998.

[6] I. J. Cox, "Blanche-an experiment in guidance and navigation of an autonomous robot vehicle," *IEEE Trans. Robotics and Automation*, vol. 7, is. 2, Apr. 1991.

[7] J. Gonzalez, A. Stentz, A. Ollero, "A mobile robot iconic estimator using a radial laser scanner," *SpringerLink J. Intelligent and Robotic Systems*, vol. 13, num. 2, Jun. 1995.

[8] P. J. Besl and N. D. McKay, "A method for registration of 3D shapes," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, Feb. 1992.

[9] F. Lu and E. Milios, "Robot pose estimation in unknown environments by matching 2d range scans," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Jun. 1994.

[10] A. A. Aghamohammadi, A. H. Tamjidi and H. D. Taghirad, "SLAM using single laser range finder," *Proc. 17th World Cong. Int. Federation of Automatic Control (IFAC)*, Seoul, Korea, Jul. 2008.

[11] T. Bailey, E. M. Nebot, J. K. Rosenblatt and H.F. Durrant-Whyte, "Data association for mobile robot navigation: a graph theoretic approach," *IEEE Int. Conf. Robotics and Automation (ICRA)*, 2000.

[12] G. K. Shaffer, "Two-dimensional mapping of expansive unknown areas," Ph.D. dissertation, Robotics Inst. Carnegie Mellon Univ., Pittsburgh, PA, Oct. 1995.

[13] D. Wolter, L. J. Latecki, R. Lakamper and X. Sun, "Shape-based robot mapping," *Proc. 27th German Conf. Artificial Intelligence*, 2004.

[14] J. Weber, K. W. Jorg and E. v. Puttkamer, "APR – global scan matching using anchor point relationships," *Proc. Int. Conf. Intelligent Autonomous Systems (IAS)*, Venice, Italy, Jul. 2000.

[15] O. Bengtsson and A. J. Baerveldt, "Localization in changing environments – estimation of a covariance matrix for the IDC algorithm," *Proc. Int. Conf. Intelligent Robots and Systems (IROS)*, Hawaii, USA, Oct. 2001

[16] A. Diosi and L. Kleeman, "Laser scan matching in polar coordinates with application to SLAM," *Int. Conf. Intelligent Robots and Systems (IROS)*, 2005.

[17] A. Garulli, A. Giannitrapani, A. Rossi and A. Vicino, "Mobile robot SLAM for line-based environment representation," *Proc. 44th IEEE Conf. Decision and Control*, 2005.

[18] D. Hahnel, W. Burgard, D. Fox and S. Thrun, "An efficient fastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements," *Proc. IEEE Intelligent Robots and Systems (IROS)*, vol. 1, Oct. 2003.

[19] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to analysis and automated cartography," *Comm. of the ACM*, vol. 24, 1981.

[20] R. Siegwart and I. R. Nourbakhsh, *Introduction to Autonomous Mobile Robots*. Cambridge, MA: MIT Press, 2004.

[21] B. Gerkey, R. T. Vaughan and A. Howard, "The player/stage project: tools for multi-robot and distributed sensor systems," *Proc. Int. Conf. Advanced Robotics (ICAR)*, Coimbra, Portugal, Jun. 2003.

Appendix A

Contents

Introduction

This report is an appendix for the presented paper. The theories and equations, which are overlooked in the paper, are elaborated in this appendix.

1. Probabilistic Motion Model

Probabilistic motion model is the name of the kinematic model which comprises the state transition probability,

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

where $p(x_t)$ denotes the probability of state x at time t, and u_t is the input to the system at time t. Equation (A.1) represents the probability of the new state, knowing the new input and the previous state.

State transition probability (A.1) plays an essential role in the prediction step of probabilistic state estimation techniques like Extended Kalman Filter (EKF) or particle filter. The goal of a proper probabilistic model is to accurately model the specific types of uncertainty that exist in robot actuation.

In Player/Stage simulations, we used a differential drive robot. There are two types of motion models defined for differential drive robots: velocity motion model, odometry motion model.

According to Thrun[1], "In practice, odometry models tend to be more accurate than velocity models, for the simple reason that most commercial robots do not execute velocity commands with the level of accuracy that can be obtained by measuring the revolution of the robot's wheels. However, odometry is only available after executing a motion command. Hence it cannot be used for motion planning. Planning algorithms such as collision avoidance have to predict the effects of motion. Thus, odometry models are usually applied for estimation, whereas velocity models are used for probabilistic motion planning."

We made use of the odometry model, since we had direct access to odometry measurements. A probabilistic odometry model, $\xi_k = f(\xi_{k-1}, u_k)$, can be formulated as follows:

$$\xi_k = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} = \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \theta_{k-1} \end{bmatrix} + \begin{bmatrix} \cos(\theta_{k-1} + 0.5\Delta\theta) & 0 \\ \sin(\theta_{k-1} + 0.5\Delta\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \Delta d \\ \Delta\theta \end{bmatrix} \tag{A.2}$$

where robot pose $\xi_k = (x_k, y_k, \theta_k)^T$ is a vector of Cartesian coordinates of the robot and its orientation wrt the global reference frame. Δd is the amount of travel between measurements, $\Delta\theta$ is the amount of orientation change between odometry measurements taken at time $k-1$ and k. For this model the system input is $u_k = (\Delta d, \Delta\theta)^T$.

The uncertainty of the pose estimate (A.2) is computed based on EKF as follows:

$$\Sigma_k = J_{\xi,k}\Sigma_{k-1}J_{\xi,k}^T + J_{u,k}QJ_{u,k}^T \tag{A.3}$$

where $\Sigma_k^{3\times3}$ is the covariance matrix of pose estimate ξ_k, $J_{\xi,k}^{3\times3}$ is the Jacobian of (A.2) wrt the robot pose, $J_{u,k}^{3\times2}$ is the Jacobian of (A.2) wrt the system input, $Q^{2\times2}$ is the diagonal noise covariance matrix of the system input.

[1] S. Thrun, W. Burgard and D. Fox, Probabilistic Robotics, Cambridge, MA. MIT Press, 2005.

2. Extended Kalman Filter

The Kalman filter, is invented by R.E. Kalman in 1960, is a recursive filter that estimates the state of a linear dynamic system from a series of noisy measurements. In other words, the Kalman filter is a technique for producing an optimal estimate of the system state based on the system and measurement models, the description of the system noise and measurement errors. Kalman filter can only be applied to linear systems, which is not the case for robotic systems. The EKF is developed to be applied on nonlinear systems.

The EKF, so the Kalman filter, estimates the system state with its uncertainty. EKF is composed of two phases: prediction and update phases. In prediction phase, the effect of the noisy control input on the system is estimated. In update phase, the system state is updated with the noisy measurements. The update phase decreases, and the prediction phase increases, the uncertainty of the state. Definitions of the prediction phase and the update phase are given in (A.4) and (A.5), respectively.

$$\begin{aligned} \bar{x}_t &= f(x_{t-1}, u_t) \\ \bar{\Sigma}_t &= F\Sigma_{t-1}F^T + F_u Q F_u^T \end{aligned} \tag{A.4}$$

$$\begin{aligned} K &= \bar{\Sigma}_t H(H\bar{\Sigma}_t H^T + R)^{-1} \\ x_t &= \bar{x}_t + K(z_t - h(\bar{x}_t)) \\ \Sigma_t &= (I - KH)\bar{\Sigma}_t \end{aligned} \tag{A.5}$$

where barred ($\bar{\cdot}$) parameters represent the predictions; x_t is the state and Σ_t is the uncertainty of the state at time t. Similar to (A.3), F and F_u are the Jacobians. $h(\bar{x}_t)$ is the measurement function which returns the estimated measurements according to the state estimate, and H is the Jacobian of $h(\bar{x}_t)$. z_t is the measurement. R is the measurement noise covariance matrix. K is called as the Kalman gain. Note that (A.4) is identical to the probabilistic motion model.

In our method, we use separate EKFs in each stage. First phase takes in odometry data coming from the wheels of the robot, whereas the second phase takes in virtual odometry data, which is the output of first phase. Nevertheless, they are in the same form, thus we use (A.2) and (A.3) in the prediction phase of both EKFs. We only use the robot pose as the state of the EKF. The measurement (z_t) of both stages is the result of scan matching, which is in the same form with the robot pose. Thus, the measurement function $h(\bar{x}_t)$ is equal to $\bar{\xi}_k$ and H is identity matrix. Consequently, the EKF algorithm which is used in our method becomes:

$$\begin{aligned} \bar{\xi}_k &= f(\xi_{k-1}, u_k) \\ \bar{\Sigma}_k &= J_{\xi,k}\Sigma_{k-1}J_{\xi,k}^T + J_{u,k}QJ_{u,k}^T \end{aligned} \tag{A.6}$$

$$\begin{aligned} K &= \bar{\Sigma}_k\left(\bar{\Sigma}_k + \Sigma_{s,k}\right)^{-1} \\ \xi_k &= \bar{\xi}_k + K(\xi_{s,k} - \bar{\xi}_k) \\ \Sigma_k &= (I - K)\bar{\Sigma}_k \end{aligned} \tag{A.7}$$

where scan matching results, $\xi_{s,k}$ and $\Sigma_{s,k}$, are already shown in Fig.2. of the reference paper.

3. Relation Functions

As described in Section III.2 of the reference paper, a relation is a geometric property of a feature. The relation of a feature in the current scan is compared to the relation of the corresponding feature in the reference scan. The result of the comparison, called relation error, is used in error minimization step. The relations of orientation (θ) and the perpendicular distance from the origin (d) are defined for all features. The relations of end point coordinates (x_e, y_e) are only defined for features with definite edges. For each relation a function $f_r(\Delta\xi)$ is defined, which returns the change of the geometric relation when the robot pose changes with an amount of $\Delta\xi = (\Delta x, \Delta y, \Delta\theta)^T$. Mathematically speaking, $f_r(\Delta\xi)$ is equivalent to $\partial r/\partial\xi$ where r is the geometric relation function. For clarity of the following explanations, note that:

- $\vec{e} = (\Delta x, \Delta y)^T$ defined in coordinate frame 1, (x_1, y_1).
- The coordinate frame 2, (x_2, y_2), is obtained by translating the coordinate frame 1 to $\vec{e}$ and rotating it $\Delta\theta$.
- The geometric relations wrt the coordinate frame 2 are distinguished with a prime ($\cdot'$).
- Geometric descriptors of the features, e.g. $\phi_\perp, d, x_e, y_e$, are known from the line representation and are in coordinate frame 1.

a. Orientation relation

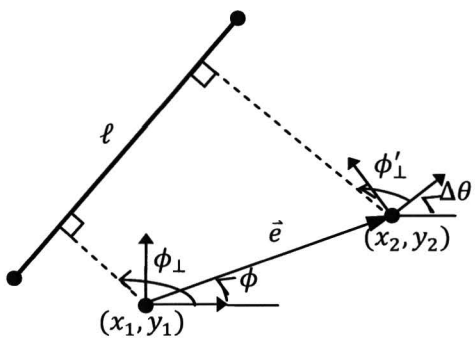
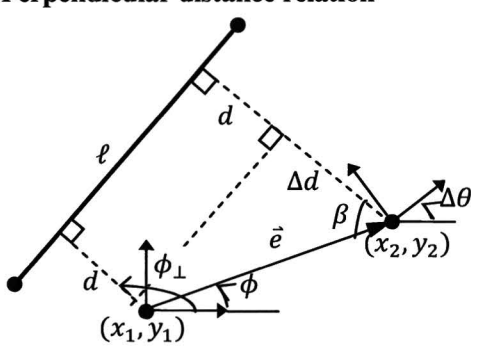


Fig.1. Orientation relation.

$$f_{r\theta}(\Delta\xi) = \phi_\perp - \phi'_\perp = \Delta\theta \tag{A.8}$$

b. Perpendicular distance relation

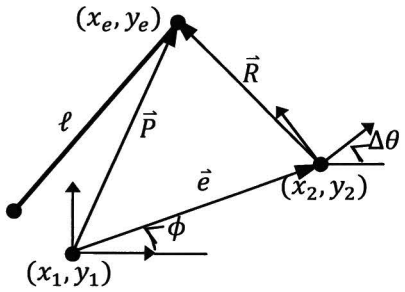


$$f_{rd}(\Delta\xi) = \Delta d = \|\vec{e}\|\cos\beta \tag{A.9}$$

where,

$$\|\vec{e}\| = \sqrt{(\Delta x)^2 + (\Delta y)^2}$$
$$\beta = \pi - (\phi_\perp - \phi)$$

c. End point relations



$$\vec{R} = \vec{P} - \vec{e}$$
$$\vec{R}' = \begin{bmatrix} \cos\Delta\theta & -\sin\Delta\theta \\ \sin\Delta\theta & \cos\Delta\theta \end{bmatrix} \vec{R}$$

$$f_{rx}(\Delta\xi) = x_e - R_x' \quad \text{(A.10)}$$
$$f_{ry}(\Delta\xi) = y_e - R_y' \quad \text{(A.11)}$$

where $\vec{P} = (x_e, y_e)^T$

4. Covariance Matrix Computation from Relation Errors

In the error-minimization step of scan matching, it is not always possible to obtain zero error. Especially with scans taken at two different times, there will always be a non-zero error in the estimated transformation. We need to predict how large the error is. Therefore, we compute the uncertainty of the resulting estimate in the form of a covariance matrix, $\Sigma_{\Delta\xi}$.

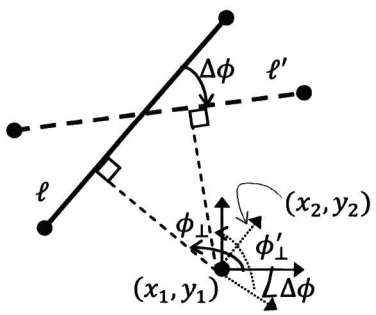
$$\Sigma_{\Delta\xi} = J_e \bar{W} E J_e^T \quad \text{(A.12)}$$

$$J_e = \begin{bmatrix} \frac{\partial f_{e1}}{\partial e_1} & \cdots & \frac{\partial f_{en}}{\partial e_n} \end{bmatrix} \quad \text{(A.13)}$$

where $\bar{W}$ is the normalized weight matrix, E is the diagonal matrix of squared errors and $f_{ei}(e_i)$ is the transformation function of the coordinate frame based on the resulting error regarding to the ith relation. e_i is taken as the relation error based on the final transformation parameters, i.e. $e_i = f_{ri}(\Delta\xi)$.

For each type of relation, a function f_e is defined. $f_e(e_i)$ returns the required displacement of the coordinate frame (x_1, y_1), to satisfy the amount of error in relation while maintaining other geometric properties constant. In other words, it is like inverse of $f_{ri}(\Delta\xi)$.

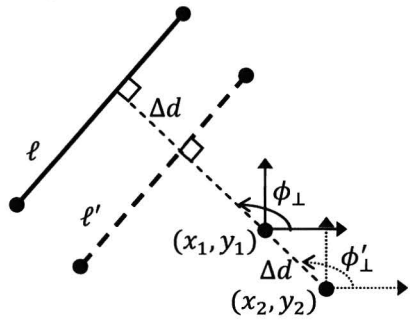
a. Orientation relation



Input: $e = \Delta\phi$,
Constant geometric properties:
$\phi_\perp = \phi_\perp'$ and $x_1 = x_2$, $y_1 = y_2$

$$f_{e\theta}(\Delta\phi) = (0, 0, \Delta\phi)^T \quad \text{(A.14)}$$

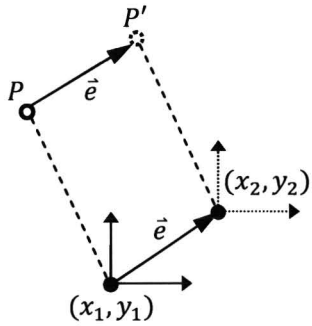
b. Perpendicular distance relation



Input: $e = \Delta d$,
Constant geometric properties:
$\phi_\perp = \phi'_\perp$ and $\theta_1 = \theta_2$ where θ is orientation of coordinate frame.

$$f_{ed}(\Delta d) = \begin{bmatrix} \Delta d \cdot \cos\phi_\perp \\ \Delta d \cdot \sin\phi_\perp \\ 0 \end{bmatrix} \tag{A.15}$$

c. End point relations



Input: $\vec{e} = (\Delta x, \Delta y)^T$
Constant geometric properties:
$\theta_1 = \theta_2$ where θ is orientation of coordinate frame.

$$f_{ex}(\Delta x) = (\Delta x, 0, 0)^T \tag{A.16}$$
$$f_{ey}(\Delta y) = (0, \Delta y, 0)^T \tag{A.17}$$

5. Determining End Point Variances from Variances of Polar Parameters

For calculation of orientation and perpendicular distance relation weights, the variances, which are calculated in feature extraction, are directly used. The weights of the end point coordinates are computed from σ_x^2, σ_y^2, which are obtained from $\sigma_\perp^2, \sigma_\angle^2$ via the following transformation (A.18), (A.19):

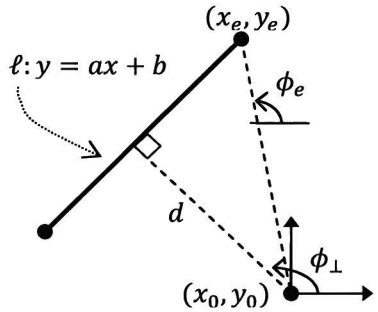
$$\begin{bmatrix} \sigma_x^2 \\ \sigma_y^2 \end{bmatrix} = J_{xy} \cdot \begin{bmatrix} \sigma_\perp^2 & 0 \\ 0 & \sigma_\angle^2 \end{bmatrix} \cdot J_{xy}^T \tag{A.18}$$

$$J_{xy} = \begin{bmatrix} \partial f_{xe}/\partial d & \partial f_{xe}/\partial \phi_\perp \\ \partial f_{ye}/\partial d & \partial f_{ye}/\partial \phi_\perp \end{bmatrix} \tag{A.19}$$

$$x_e = b(\tan\phi_e - a)^{-1}, \quad y_e = a x_e + b \tag{A.20}$$

$$a = \tan\left(\phi_\perp + \frac{\pi}{2}\right), \quad b = d\sin\phi_\perp - ad\cos\phi_\perp \tag{A.21}$$

where (x_e, y_e) are end point coordinates, calculated in (A.20) as the intersection point of line ℓ with $y = \tan(\phi_e)x$ line. In (A.21), conversion from known polar parameters to Cartesian parameters is done. Eventually, (A.22) is obtained by substituting (A.21) in (A.20).



$$f_{xe}(d, \phi_\perp, \phi_e) = \frac{d\sin\phi_\perp - d\tan\left(\phi_\perp + \frac{\pi}{2}\right)\cos\phi_\perp}{\tan\phi_e - \tan\left(\phi_\perp + \frac{\pi}{2}\right)} \tag{A.22}$$

6. Solution of Rank Deficiency Problem in Gauss-Newton Error Minimization

The errors of the relations are attempted to be minimized in weighted least-squares sense by using Gauss-Newton method, given in (A.23), (A.24).

$$\Delta\xi = -(J^T W J)^{-1} \cdot J^T W \cdot e \tag{A.23}$$

$$J = \begin{bmatrix} \partial f_{r1}/\partial x & \partial f_{r1}/\partial y & \partial f_{r1}/\partial \theta \\ \cdots & \cdots & \cdots \\ \partial f_{rn}/\partial x & \partial f_{rn}/\partial y & \partial f_{rn}/\partial \theta \end{bmatrix} \tag{A.24}$$

where e is the vector of relation errors, $\Delta\xi = (\Delta x, \Delta y, \Delta\theta)^T$ is difference vector between the transformation parameters on successive iterations, W is the diagonal weight matrix and J is the Jacobian of size $n \times 3$.

It is possible that all of the features, in the scan, lie in the same direction and there is not sufficient information in orthogonal direction to that. In such a case, the matrix multiplication $Z = J^T W J$ cannot be inverted due to rank deficiency. Principal component analysis of Z returns the direction (ψ) which is not observable. We omit the column, which causes the rank deficiency, in J and obtain J^*. As a result, $Z^* = (J^*)^T W J^*$ will have an inverse and will minimize the error. However, the resulting translation vector $\vec{P}$ is expressed in robot coordinates. Unless the principal component directions coincide with the robot coordinates, $\vec{P}$ will include some arbitrary value in ψ direction. The arbitrary component has to be eliminated for a reliable pose estimate. Fig.1. illustrates the situation.

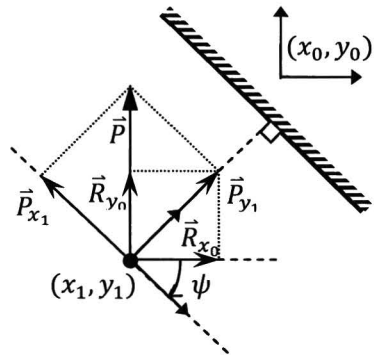


Fig.A.1. Rank deficiency solution.

In Fig.A.1. (x_0, y_0) is the robot reference frame. (x_1, y_1) is the coordinate frame coincident with the principal component directions of Z. The arbitrary component of $\vec{P}$, which is $\vec{P}_{x_1}$, is taken to be zero, since there is no information in that direction. Then, $\vec{P}_{y_1}$ is decomposed to components $(\vec{R}_{x_0}, \vec{R}_{y_0})$ in reference frame (x_0, y_0). Eventually, the updated result of error minimization becomes $\Delta\xi = \left(R_{x_0}, R_{y_0}, \Delta\theta\right)^T$.

Rank deficiency in Z also affects the covariance matrix $\Sigma_{\Delta\xi}$. Since, there is no information in ψ direction, $\Sigma_{\Delta\xi}$ will have a zero eigenvalue pointing to ψ direction. In order to maintain the consistency of covariance matrix with the pose estimate, we change that eigenvalue with a constant non-zero value, which represents a constant uncertainty in that direction.

Appendix B

Contents

1. Laser scanner hardware
2. Distance measuring
3. A laser scan example

Introduction

Apart from the presented paper, some work is done on a laser scanner module. This report explains the developed laser scanner module. The structural properties of the module, theoretical and practical methods used for distance measuring and an example of a 360 degree scan are explained in detail. Due to timing restrictions, no SLAM experiments are done with the constructed laser scanner module. Nevertheless, feature extraction tests are done and presented in the following chapters.

1. Laser Scanner Hardware

The laser scanner module is formed of two sub-modules. These are a pan-tilt unit, which is used for 360 degree rotation, and a laser range finder setup, which is duplicated from the work of Konolige[1].

a. Pan-Tilt Unit

A pan-tilt unit of model PTU-46-17.5, which is produced by Directed Perception (USA), is used for actuation. Fig.B.1. shows the pan-tilt unit. It has two degrees of freedom, i.e. pan and tilt motions. The lower motor is used for the pan motion and the upper motor is used for the tilt motion. In Fig.B.1. there is a camera, which we do not have, mounted on the unit. The unit has a height of $15cm$. The base plate area of the unit is $12cm \times 12cm$.

Fig.B.1. Pan-Tilt Unit.

The pan-tilt unit has its own position controller. Velocity and acceleration of the motor is adjustable. We are using the unit at the highest allowable velocity and acceleration values. The embedded controller generates a trajectory as soon as the new position order is received.

Communication with the unit is achieved via serial port of the computer (RS232). Simple commands are sent to the unit to move the motor to a desired position. The unit sends signals to inform that a command is processed. A simple driver is built for achieving the two-way communication.

b. Laser Range Finder

The laser range finder setup has been built in Philips Apptech. An ordinary laser beamer, which emits laser with maximal $5mW$ optical power, at $635nm$ wavelength, and a uEye UI1226LE-M camera behind a red-passing filter are used. A feasibility study was made about usage of the principle for mobile robot navigation. Fig.B.2. shows the laser range finder setup. The setup has a height of $8cm$. Distance measuring is achieved at the height of $5cm$. The base plate area of the setup is $8cm \times 15cm$.

[1] K. Konolige, J. Augenbraun, N. Donaldson, C. Fiebig and P. Shah, "A low-cost laser distance sensor," *Proc. IEEE Int. Conf. Robotics and Automation*, Pasadena, CA, USA, May 2008.

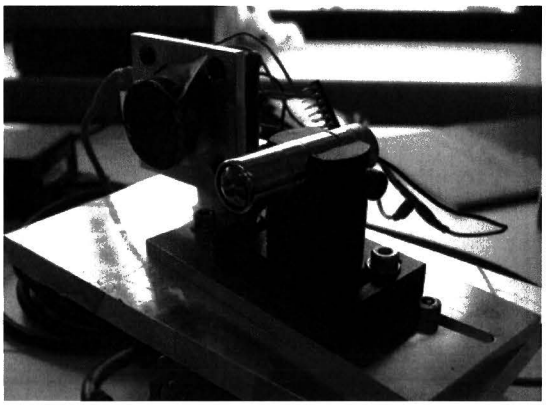


Fig.B.2. Laser range finder setup.

The laser range finder uses triangulation principle to measure the distance. Fig.B.3. illustrates the triangulation principle. In Fig.B.3. the distance s and the angle β are structural properties. The focal distance of the imager (camera) is denoted by f. The reflection angle of the laser ray on the imager changes according to the distance, between the object and scanner, denoted by q. The distance x on the imager provides a measure for the real distance q.

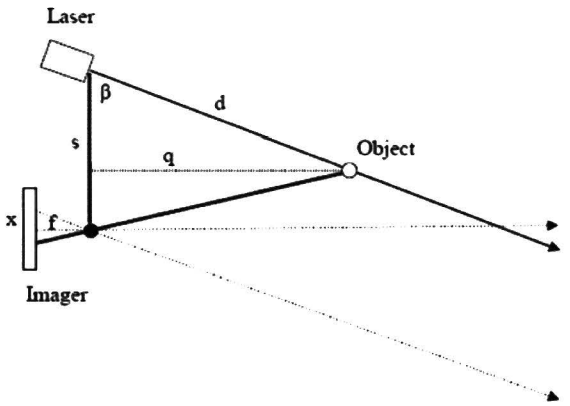


Fig.B.3. Triangulation principle. The distance to the object is given by the angle of the laser spot in the image. The distance x in the image is measured between the ray parallel to the laser beam and the ray from the object. [Konolige]

For our setup, s is $60mm$ and β is 82 degrees. The focal distance of the camera is $16mm$ and pixel size is $6\mu m$. According to Konolige, with these parameters, the minimum distance we can measure is $23cm$. The theoretical maximum distance is determined according to a sensitivity limit. By expecting to be able to resolve the laser dot on the image within 0.1 pixels or better [Konolige], the maximum range becomes $6m$. At that range, 0.1 pixel change on image corresponds to $27mm$ in distance. With the same expectation, our setup is supposed to measure distances up to $7m$. However, due to robustness concerns the maximum measurable range drops to a distance between $200cm$ and $250cm$. At this range, 0.1 pixel change on image corresponds to $3mm$ in distance.

2. Distance Measuring

a. Laser Scanning

Laser scanning is executing the distance measuring operation at several angles. Multiple angle measurement is realized by rotating the laser range finder device with the pan-tilt unit. At each angle, an image is captured and the image is processed to find the distance.

The camera is located behind a red-passing filter. This filter suppresses non-red light, but having a filter is not sufficient to suppress other objects in the image. The laser is a highly-dense and relatively high power light, so even with a low exposure time; it is visible in the image. We set the exposure time to the lowest possible value, namely $0.30ms$, to capture the laser reflection clearly while not having any other objects in the image.

The laser beam reflects from the closest surface and a "white dot" appears in the camera image. As the distance to the reflecting surface increases, the "white dot" travels to left on a horizontal line. Therefore, by measuring the position of the "white dot" on the image, one can determine the distance to the surface. Processing the whole image is unnecessary, so we crop the image to process. Around the horizontal line which the "white dot" moves along, 5 pixels are cropped out as the region of interest. Fig.B.4. shows how the camera images and the regions of interest look like.

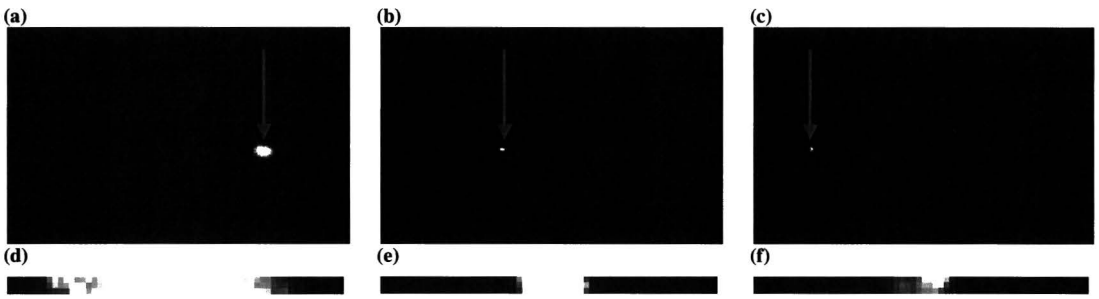


Fig.B.4. Camera images at various distances. The red arrows indicate the white dots. (a) Full image at 30cm. (b) Full image at 60cm. (c) Full image at 120cm. (d) Region of interest at 30cm. (e) Region of interest at 60cm. (f) Region of interest at 120cm.

The region of interest (ROI) is a bitmap which can be treated as a matrix of size 5×752. In order to measure the distance, we compute the center of mass of the "white dot" in the ROI. It is expected that a "white" region will always exist in the ROI, if the object, from which the laser reflects, is between the minimum and maximum measurable distances. Nevertheless, as the distance increases the size and the brightness of the "white dot" decreases; and eventually the white dot disappears before reaching the maximum distance. Thus, existence of a white dot in the ROI is not guaranteed as well as the brightness of the dot. Consequently, a robust center of mass measuring algorithm has to be implemented. After calculating the position of the "white dot" in pixels, a nonlinear calibration function is used to convert a pixel value to distance.

A software library is built in C# environment for laser scanning operations. Connection to the pan-tilt unit, image capturing, image processing and distance calibration etc. are done within the library.

b. Center of Mass Algorithm

The relation between the position of the "white dot" and the real distance is mathematically well-defined. However, due to the discrete image it is not possible to know the exact position. The only way to estimate the position of the "white dot", is calculating the center of mass of the white dot.

In grayscale bitmaps, white represents the highest value, 255, and black represents the lowest value, 0. In the camera images, the laser reflection is so strong that it usually appears as white. However, it is not guaranteed that the laser reflection will always appear as white in the image. The brightness of the reflection depends on the distance, the orientation of the surface, the color of the surface etc. There is also noise existing in the image. The algorithm is needed to distinguish between noise and reflection, and to compute the center of mass of the brightest region compared to the rest of the image. The algorithm is as follows:

Let, I_θ be the region of interest of the captured image at an angle θ. I_θ is a matrix of m rows and n columns which comprises the bitmap. Let, $\overrightarrow{\imath_\theta}$ be a vector of length n. Fig.B.5. shows an example of $\overrightarrow{\imath_\theta}$.

$$i_{\theta,k} = \sum_{j=1}^{m} I_{\theta,j,k} \quad for\ k = 1,2 \dots n \tag{B.1}$$

where $i_{\theta,k}$ represents the k-th element of $\overrightarrow{\imath_\theta}$ and $I_{\theta,j,k}$ represents the j-th row, k-th column element of I_θ.

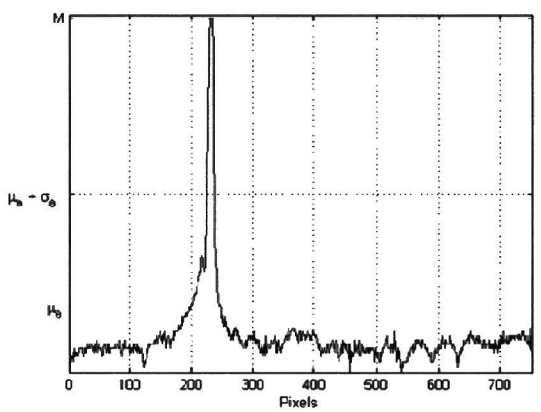


Fig.B.5. An example for vector $\overrightarrow{\imath_\theta}$. Horizontal axis represents the index (pixel) values.

Let, μ_θ be the mean value of the data $\overrightarrow{\imath_\theta}$ and σ_θ be the standard deviation of $\overrightarrow{\imath_\theta}$.

$$\mu_\theta = \frac{1}{n}\sum_{k=1}^{n} i_{\theta,k} \tag{B.2}$$

$$\sigma_\theta = \sqrt{\frac{1}{n}\sum_{k=1}^{n}\left(i_{\theta,k} - \mu_\theta\right)^2} \tag{B.3}$$

Fig.B.6. shows a zoom in view of the peak shown in Fig.B.5.

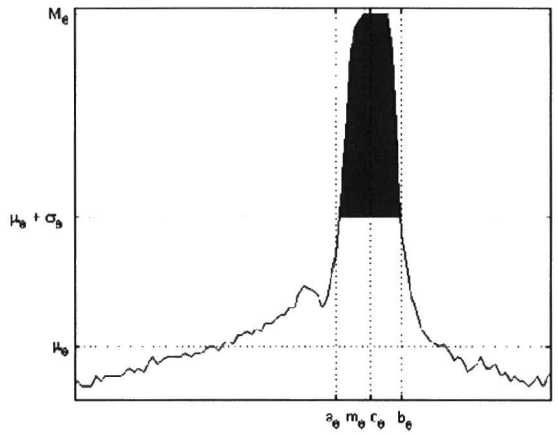


Fig.B.6. A closer view to the peak.

Let, M_θ be the maximum value of $\overrightarrow{\iota_\theta}$ and m_θ is the smallest index position of M_θ in $\overrightarrow{\iota_\theta}$.

The mass, of which center is going to be used, is defined around m_θ. As shown in Fig.B.6., there is one region, shaded with red color. The red region is limited from above by the vector values and from below by $\mu_\theta + \sigma_\theta$. This region is called as the mass. According to Fig.B.6., a_θ is defined as the smallest (leftmost) index of the mass, b_θ is defined as the greatest (rightmost) index of the mass. The center of mass is shown as c_θ which is defined as:

$$c_\theta = m_\theta + \frac{\sum_{k=a_\theta}^{b_\theta}\left(i_{\theta.k} \cdot (k - m_\theta)\right)}{\sum_{k=a_\theta}^{b_\theta} i_{\theta.k}} \tag{B.4}$$

In theory, the c_θ value must be in between 0 and n. However, due to hardware limits and concerns about reliability of the algorithm, the range cannot be entirely exploited.

In our case, the raw image is of size 480×752 $(row \times col)$ and the ROI, I_θ, is of size 5×752 $(m \times n)$ which is cropped around row 270. Consequently, the vector $\overrightarrow{\iota_\theta}$ has length of $n = 752$. According to the experiments, the range of c_θ is between 50 and 715. However, the sensitivity of c_θ with respect to distance change is very small at lower values, so the lower range is shifted to 90 for reliable operation.

c. Calibration

Konolige defined the calibration formula as follows. The parameters are described in Fig.B.3.

$$q = \frac{fs}{x} \tag{B.5}$$

According to the formula x is the position of the reflection on the camera sensor in μm, fs is in mm^2 and q is in cm. The output of the center of mass algorithm is pixel values instead of length, as x. The conversion from pixel value, p, to x is $x = 0.06 \cdot p$ where x is in μm.

Despite the assumption that f and s values known exactly from the structure and they are constant; during the calibration tests, it is figured out that they are not constant as expected. This result might be due to the imperfect structure of the laser scanner, or it indicates that the formula neglects some other parameters. Eventually, a function is fitted for fs values according to results of calibration tests.

For calibration, two range sweeps are done with two different reflecting surfaces, in other words, the measurable range is sampled at each $10cm$ and measurements are taken with one white and one black object. The result is presented in Fig.B.7.

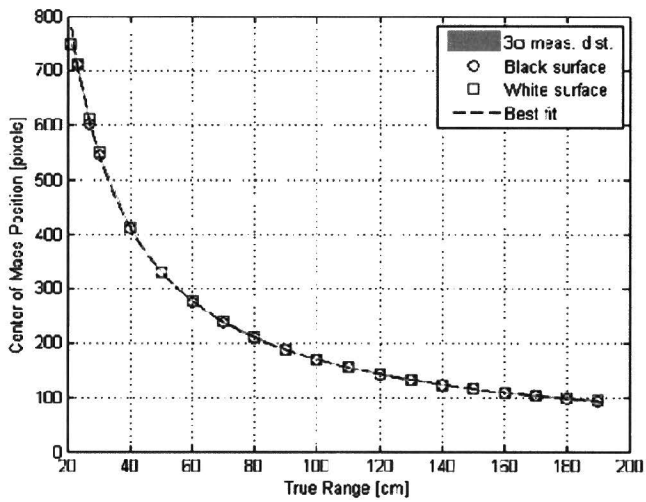


Fig.B.7. Result of calibration tests.

The residuals in measurements are shown in Fig.B.8. In the figure, blue and red bars correspond to black and white surface measurements, respectively.

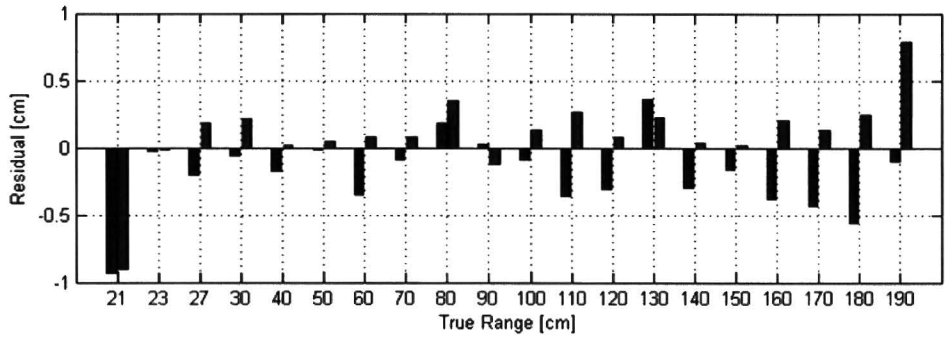


Fig.B.8. Calibration residuals.

In order to stick to the formula which theory showed (B.5), a rational function is fitted for fs values, to eliminate the square sum of errors. Fig.B.9. shows the distribution of fs values.

$$fs(p) = \frac{0.02756p^2 + 949.6p + 19780}{p + 8.676} \tag{B.6}$$

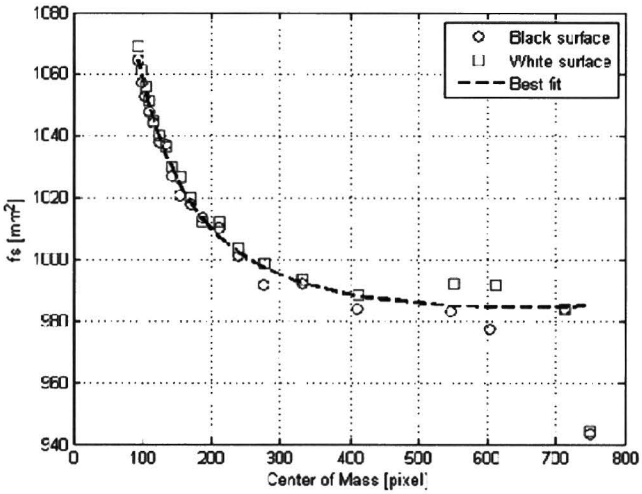


Fig.B.9. Fitted function on fs values.

d. Reliability of Results

Not all of the measurements are as reliable as any other. Based on many factors, such as the brightness of the environment, the illumination of the object, the color of the object, the surface orientation of the object or the distance to the object etc., the properties of the "white dot" may change. Therefore, it is handy to know how certain a measurement is.

During the scan, places out of the measurable range will not reflect any laser light, so no mass will exist in the image to find the center. In such a case, the algorithm may return any value due to noise in the image. Thus, out-of-range and low-reflectance surface measurements are totally unreliable and must not be taken into consideration. In commercial range finders, regardless of the principle they employ, there is not a reliable method to detect false measurements.

In our approach, it is possible to define a reliability measure in terms of contrast of the "white dot" relative to the black region in the image. We call the reliability measure as confidence value. The confidence value is between 0 and 1. A confidence value of 1 means the measurement is very clear and almost certain, while 0 means the measurement is totally unreliable, even random. The confidence value of a measurement taken at angle θ is defined as follows:

$$C_\theta = 1 - \frac{\mu_\theta + \sigma_\theta}{M_\theta} \tag{B.7}$$

Equation (B.7) is guaranteed to be between 0 and 1, since statistically it is guaranteed that the maximum value will remain outside of the 1σ confidence region. Experimentally, it is determined that confidence values greater than 0.55 can be considered as reliable. The limiting confidence value corresponds to a mass which has a maximum value of 2.2 times the 1σ confidence value.

Fig.B.10. shows some measurement examples and their confidence values. As the distinctiveness of the peak becomes poor, the confidence value decreases.

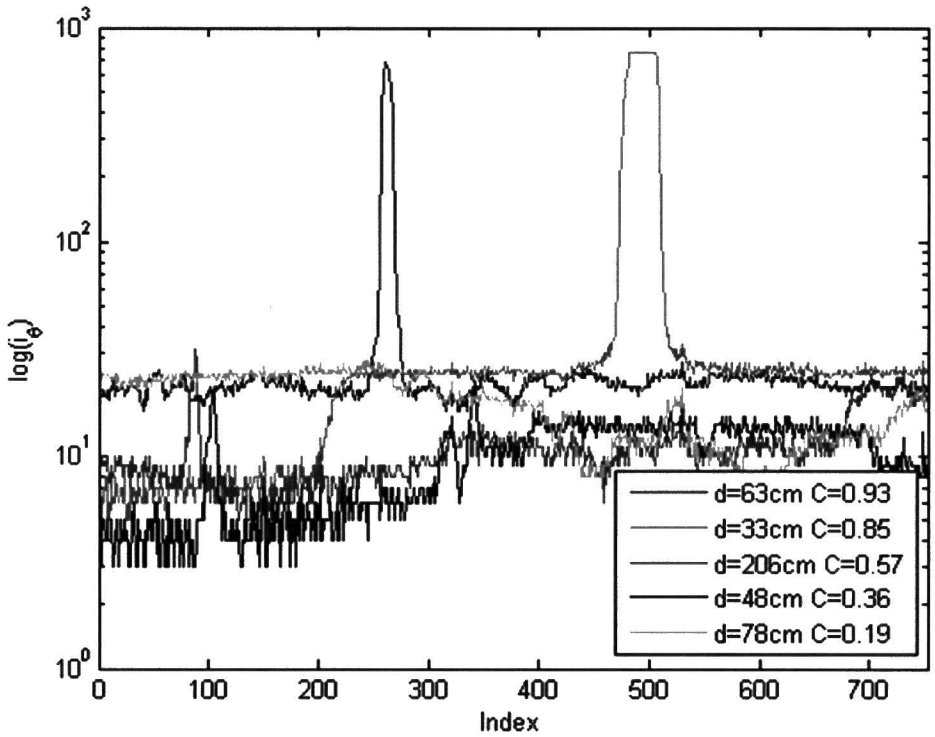


Fig.B.10. Measurements and corresponding confidence values. Logarithm of the vertical axis is taken for visual purposes.

3. A Laser Scan Example

A 360 degree scan with a resolution of 0.5 degrees is performed in an office environment with regular-shaped (geometric) objects. The results are as follows:

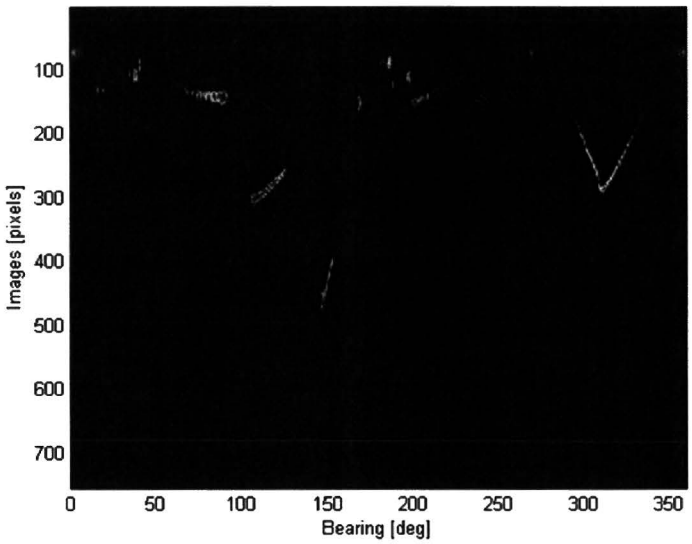


Fig.B.11. Raw measurement in series order. Columns represent $\overrightarrow{l_\theta}$. Dark blue corresponds to black regions while dark red corresponds to white regions.

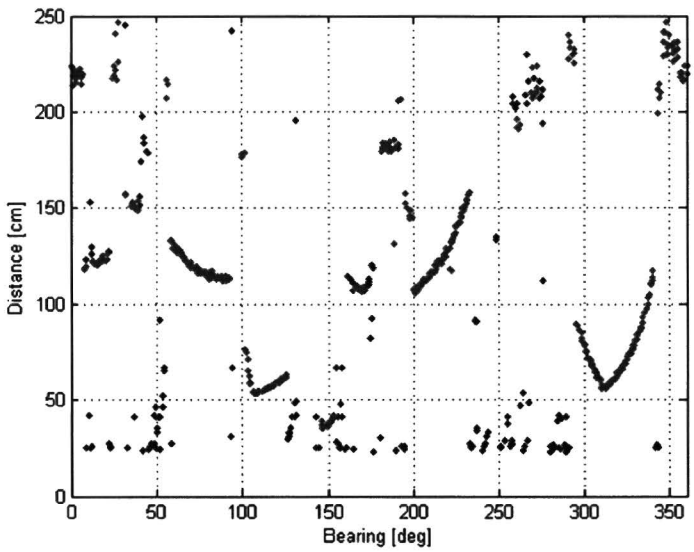


Fig.B.12. Processed and calibrated measurements. Red dots are the reliable measurements while blues are unreliable ones.

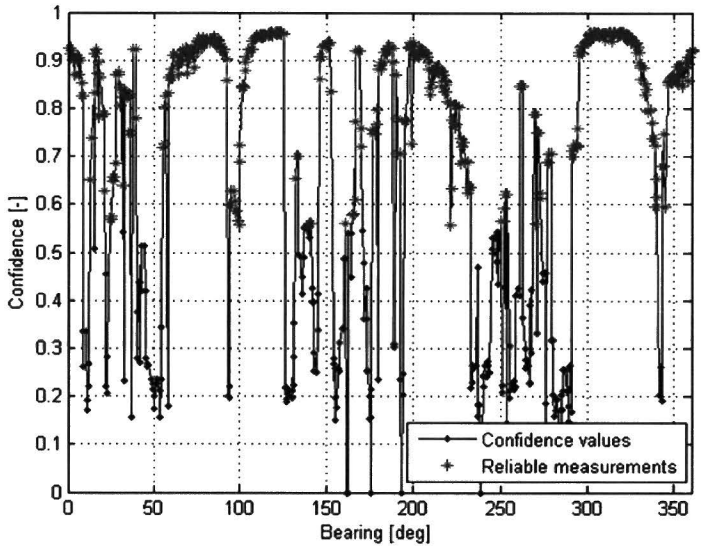


Fig.B.13. Confidence values of the measurements. Reliability threshold is taken as 0.55.

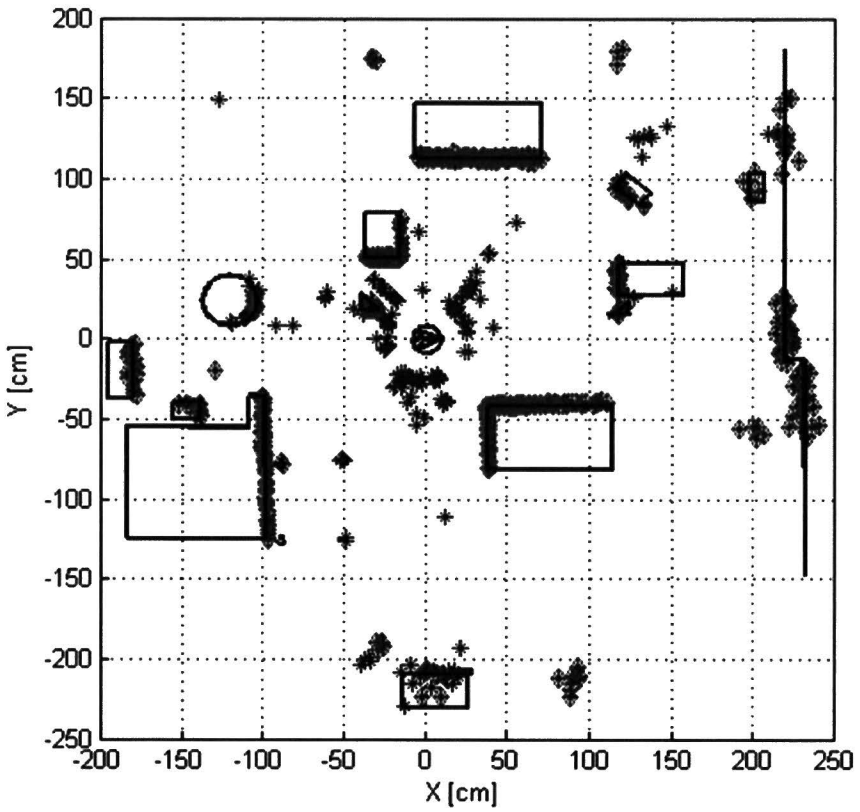


Fig.B.14. The measurements are drawn in Cartesian space. Reliable measurements are shown in red. The robot is shown in (0,0). The black shapes represent the original map of the environment.

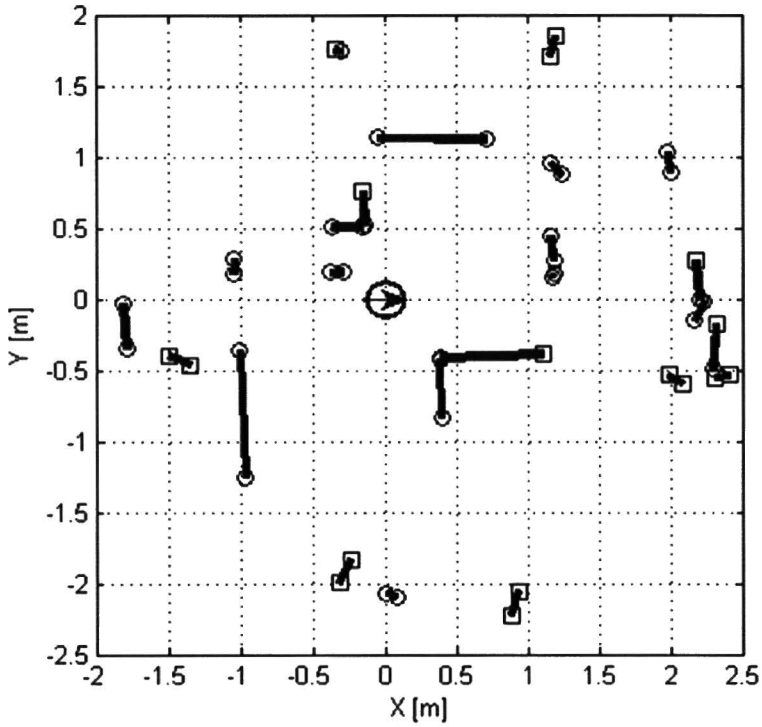


Fig.B.15. Extracted features from the reliable measurements. The robot is shown in (0,0).

From SLAM point of view, the significance of the reliable measurement concept is that one does not need to distinguish between noisy measurements and true measurements. By simply utilizing the confidence values, one can extract or match features easily. Moreover, the confidence concept also eliminates misinterpretation of irregular-shaped features. In other words, if there was no measure of reliability of scans, a feature extraction/matching algorithm might easily consider the irregular-shaped feature as a false measurement of a regular-shaped feature, or vice versa.