AMOS: Comparison of Scan Matching Approaches for Self-Localization in Indoor Environments

Jens-Steffen Gutmann Christian Schlegel
Research Institute for Applied Knowledge Processing (FAW)
PO-Box 2060, D - 89010 Ulm, Germany
{gutmann,schlegel}@faw.uni-ulm.de

Abstract

This paper describes results from evaluating different self-localization approaches in indoor environments for mobile robots. The examined algorithms are based on 2d laser scans and an odometry position estimate and do not need any modifications in the environment. Due to the goals of our project an important requirement for self-localization is the ability to cope with office-like environments as well as with environments without orthogonal and rectilinear walls. Furthermore, the approaches have to be robust enough to cope with slight modifications in the daily environment and should be fast enough to be used on-line on board of the robot system. To fulfil these requirements we made some extensions to existing approaches and combined them in a suitable manner. Real world experiments with our robot within the everyday environment of our institute show that the position error can be kept small enough to perform navigation tasks.

Keywords: *Mobile Robot, Self-Localization*

1. Introduction

Using a mobile robot as an experimental platform, the AMOS [1] (Autonomous MObile Systems) project is investigating the problem of integrating symbolic and subsymbolic forms of information processing [4, 5].

One of the main features of the architecture is the integration of both symbolic planning as well as subsymbolic reactive mechanisms. This is of great importance in particular for systems which have to act robustly and goaldirectedly even in a dynamic environment and in unforeseen situations.

The experimental platform used in the AMOS project is a three-wheeled industrial vehicle which is driven and steered by a single front wheel. The system is equipped with different sensors such as odometry, ultrasonic, laser range finders and two video cameras.

To perform different tasks this system has to move around in our institute requiring the robot to know its position accurately enough to be able to reach the goal region where if necessary local maneuvers are activated to provide higher accuracy. As position estimation solely based on odometry accumulates position errors without upper bound, self-localization approaches are necessary. Due to the goals of our project self-localization has to work without artificial landmarks and has to be fast enough to be used on-line on board of the robot. Furthermore, it has to work in office-like environments as well as in environments without orthogonal and rectilinear structures and has to cope with slight modifications in a daily environment.

This paper is organized as follows: section 2 gives a small overview of existing self-localization approaches and classifies them, section 3 presents our solution for the self-localization problem and presents a *combined scan matcher*. In section 4 we test the *combined scan matcher* in both simulated and real environments.

2. Related Work

The AMOS platform offers several sensors for the self-localization task. We choose a 2d laser range finder because the other sensors, e.g. ultrasonic are inaccurate and unreliable or require a large amount of processing power (vision). The platform is equipped with two range finders for collecting scans. One range finder with a viewing angle of 220° is mounted in front of the robot at a height of approximately 70cm which allows to detect tables and chairs. The other range finder is mounted on top of the platform with a 360° viewing angle. The height of the first laser range finder is suited for path planning and obstacle avoidance whereas the height of the second one together with its full sight is suited for the self-localization task. We therefore use only the sec-

[1] The AMOS Project is granted by the BMBF (German Federal Ministry Of Research And Technology), grant no. 01 IW 302A | 1

ond range finder for self-localization.

Most approaches using laser scans for self-localization can be classified into two categories: approaches that use a geometric map of the environment and approaches that match a pair of scans.

2.1. Approaches with geometric map

These approaches match a scan or the features of a scan with a global map of features, e.g. [1, 2]. Therefore the environment has to contain enough features for the geometric map. The map is either determined a-priori or built while exploring the environment.

2.2. Scan matching approaches

Here, the robot's position is estimated by matching a pair of scans, e.g. [6, 8]. The first scan serves as reference scan and has previously been taken or is stored in an a-priori map of scans. The second scan (called current scan) is matched against the reference scan and its scan position is determined relatively to the position of the reference scan. The result of this match is a position correction which can be applied to the current robot position.

2.3. Problems

Building an exact map of the environment is a complicated task. We do not want to measure all distances by hand and then use a CAD tool to build the map. The robot should build the map itself by using its sensors.

Approaches that allow map building while exploring the environment, e.g. [1], normally start with an empty map. Each time a new scan is taken, the robot estimates its position by comparing the current scan with the map built so far and updates the map by integrating the new scan into the map.

This works quite well until the robot's path contains a loop. In this case, a problem arises when the robot closes the loop: for position estimation two parts of the map can be used. The first part consists of scans that have been taken most recently whereas the second part consists of scans that have been taken some time ago. Normally these two parts are inconsistent to each other because of the incremental map creation. Comparison of the current scan with the map will either align the scan to the first or second part of the map. So there may be a position jump in this area. To avoid such position jumps a correction of all scan positions along the robot's path is needed, but this will also imply an update for the geometric map, which can be quite difficult and therefore is not addressed in most implementations.

For these reasons, the approach for map building presented in [7, 6] simply takes a set of scans whose positions were estimated by dead-reckoning only and estimates the position of all scan positions in such a way that a consistent map of scans is built.

3. Our Work

We did some further investigations in scan matching algorithms for the following reasons: scan matching doesn't need a geometric map, there are matching algorithms like the *idc* algorithm [6] that work in arbitrary environments not necessarily being polygonal, and scan matching builds the basis for the global position estimation from [7, 6].

We examined three different scan matching approaches: matching by assigning points to line segments, matching by using the cross correlation function and matching by point-to-point assignments.

3.1. Assigning points to line segments

The idea for matching by assigning points to line segments is orginated by Cox [2] where points of the current scan are matched against an a-priori model consisting of line segments. Of course this works only in environments that are mainly polygonal.

A small and intuitive modification of this algorithm is to extract line segments from the reference scan and use them as a-priori model. This makes the Cox algorithm suitable for scan matching (see figure 1).

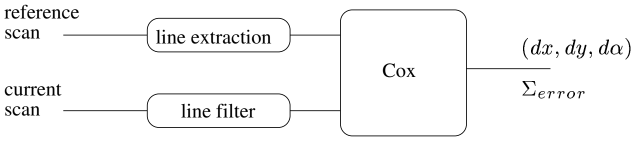

Figure 1. Extended Cox algorithm.

We did some further modifications: first, the current scan is filtered through a line filter, that is, scan points that are not on a line segment are removed. This reduces the amount of false point-to-line segment assignments in situations where the environment is not totally polygonal. Second, a hard coded threshold of $d_{max} = 2000mm$ is used to remove assignments that have a larger distance than d_{max}. And third, the assignments are ordered by their distances (smallest first). Only the first 80 per cent of assignments are used for matching, the rest is treated as outliers and is removed.

3.2. Using the cross correlation function

Another scan matching approach is based on the use of a cross correlation function (*ccf*) [8]. Here both scans are replaced by stochastic representations (histograms) and the matching is solved by finding the maximum of a cross correlation function.

This correlation only works in polygonal environments. Furthermore, the basic algorithm as presented in [8] is especially suited for a perpendicular environment but it is not difficult to modify the algorithm to work in non-perpendicular environments.

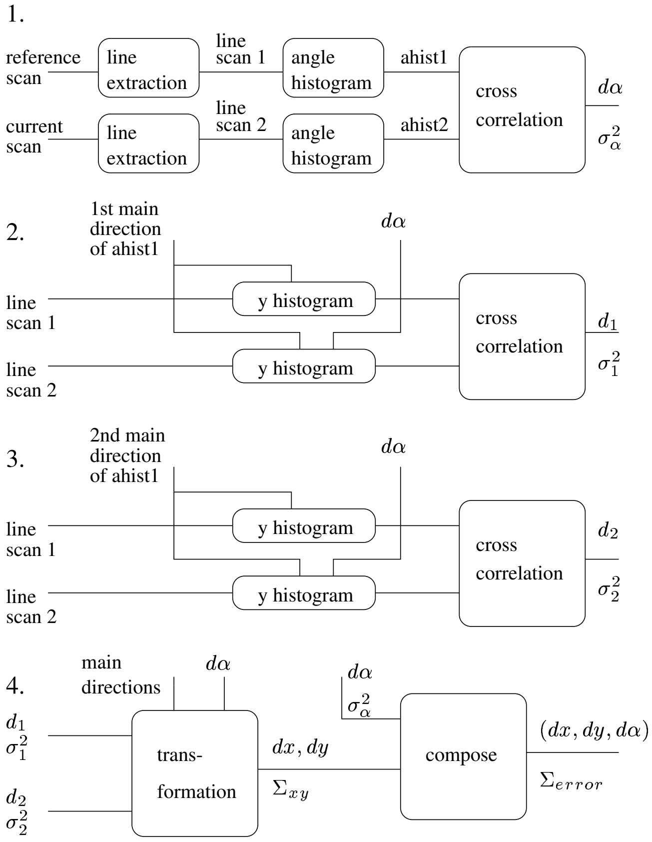

Figure 2. Extended ccf algorithm.

Figure 2 shows our modifications to the *ccf* algorithm. In the first step we extract lines from the scans and calculate the angle difference $d\alpha$ of the two scans along with its error σ^2_α. Using the line segments of the scans ensures that calculated histograms will have high peaks since only points on line segments generate peaks. It also avoids histograms that have high peaks generated by small walls very close to the range finder because these walls are covered with lots of scan points (due to the small range finder distance and the constant angle resolution of the scans).

For solving the problem of non-perpendicular environments we not only use one main direction in the angle histogram but also a second main direction (which is simply another high peak in the angle histogram).

In step two and three we rotate the scans in such a way that the main directions lie on the x-axis. The y histograms therefore always have high peaks.

Beneath the results from the cross correlation function we also calculate an error value σ^2 for each component which is derived from the value of the normalized cross correlation function.

In step four we transform the results from step two and three and compose all components to the final result.

3.3. Point-to-point assignments

In [6] an algorithm called *idc* is presented that allows matching of two scans which do not have to consist of geometric features. This allows the use of the *idc* algorithm even in environments which are not polygonal.

The *idc* algorithm does a point-to-point correspondence for calculating the scan alignment. The correspondence problem is solved by two heuristics: the closest point rule and the matching range rule. Furthermore, a formula is provided for calculating an error covariance matrix of the scan matching (see [6] for details).

Our first implementation of the *idc* algorithm seems to be a little time consuming so we added a filter to reduce the number of scan points (see figure 3).

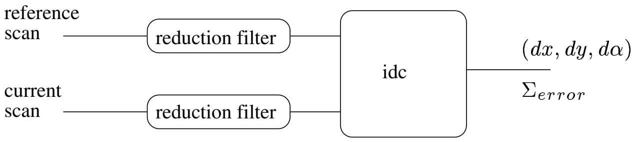

Figure 3. Extended idc algorithm.

This filter replaces clouds of scan points by their center of gravity. This has the effect that the distribution of points over the scan is more smooth. It also greatly reduces the number of scan points without loosing too much information. The number of scan points however is responsible for the runtime of the *idc* algorithm (our implementation of the *idc* algorithm has complexity $O(n^2)$ where n is the number of scan points).

3.4. Combined scan matching algorithm

We tested the three algorithms presented above in simulated and real environments. As expected there are situations where the algorithms exploiting polygonal structures are superior to that one designed for non-polygonal environments and vice versa. These results are shown later.

Therefore, we combined one of the algorithms developed for polygonal environments (Cox, *ccf*) with the *idc* algorithm which works fine even in non-polygonal environments and constructed a new algorithm, we call *combined scan matcher* (see figure 4). We preferred the extended Cox algorithm rather than the extended *ccf* algorithm because in

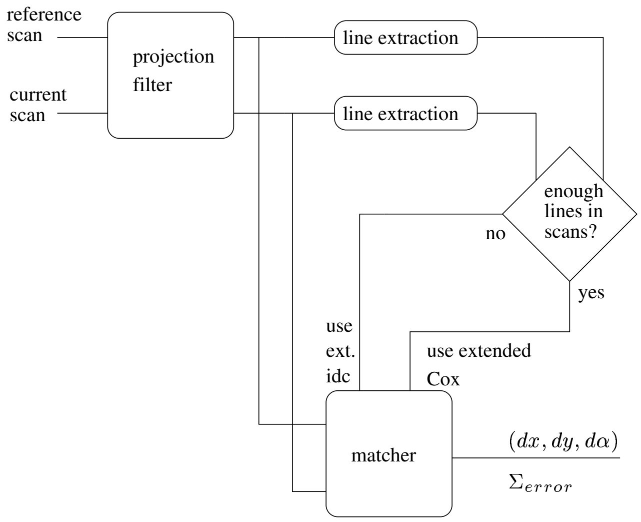

Figure 4. Combined scan matcher.

our implementation the extended Cox algorithm seems to be a bit faster than the extended *ccf* one. However, the same results can be achieved by using the extended *ccf* algorithm.

To decide when to use which algorithm within the *combined scan matcher* we examine the line segments in the scans. If there are enough scan points lying on line segments then the extended Cox algorithm is used, otherwise the extended *idc* algorithm is used.

Before matching two scans, we apply a filter called projection filter removing scan points in both scans that are not visible from each other's scan position. This way only points remain that are visible from both scan positions and therefore false point-to-line or point-to-point correspondences are minimized. Since we don't know the exact scan positions when applying the projection filter, the position information from dead-reckoning is used here.

Details on the projection filter can be found in [6].

3.5. Selflocator module

The self-localization procedure we use on the AMOS platform requires an exploration phase for preparation.

Exploration. The environment is explored and a set of scans is taken. The scan positions are determined by dead-reckoning only. For exploration the robot is driven by hand or algorithms are used that allow the robot to do this on its own. After exploration the set of scans is transferred to a workstation where all scan positions are calculated in such a way that a consistent map of scans is generated. Therefore, the algorithm from [7, 6] builds a matrix of relative position estimations for each pair of scan that have enough overlap (determined by using the projection filter). The algorithm then estimates all scan positions by minimizing an error function.

The scans with the corrected positions are transferred back to the robot. This set of consistent reference scans now enables the robot to estimate its position in the explored environment.

Self-localization. For self-localization the robot simply takes a new scan, uses the position estimated by dead-reckoning to search for a reference scan and matches the two scans with the *combined scan matcher* (see figure 5). For determining the match partner in the map of reference scans we simply use the one whose position is nearest to the current robot position.

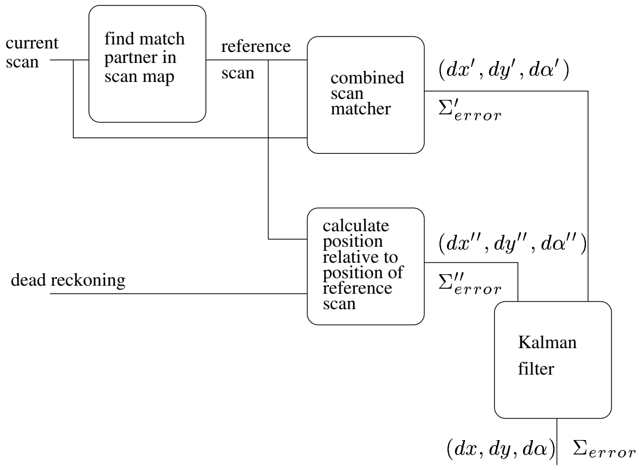

Figure 5. Selflocator module.

We use a Kalman filter to combine the measurement from the scan matcher with the information from dead-reckoning for maximum accuracy.

Changes in the environment. The scan matching algorithms can handle small modifications in the environment, e.g. closed/opened doors, new cabinets in a hallway, etc. However if the changes are too significant the scan matcher will produce poor results. In this case the component values of the calculated error covariance matrix will be large and therefore the calculated position update of the scan matcher has little effect on the Kalman filtered update.

By examining the calculated error covariance matrix of the scan matcher it is possible to detect situations where there are too many changes in the environment and a part of the prestored scans should be retaken and corrected by the global position estimation procedure.

4. Results

We tested the scan matching algorithm and the global position estimation procedure in simulated and real environments and compared the results of the three extended matching algorithms.

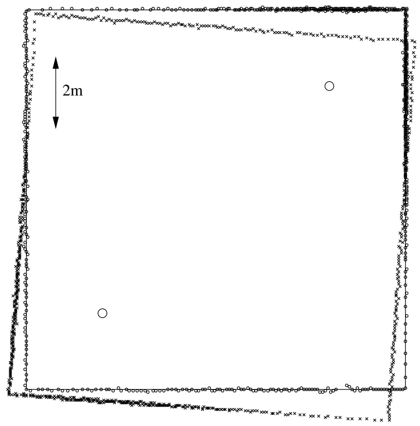

Figure 6. Test environment for comparing the three scan matching algorithms. Scans have been taken at the upper right and the lower left corner.

4.1. Simulated data

Figure 6 shows a simulated test environment used to compare the three extended scan matching algorithms. The environment consists of four long walls with a length of $10m$ each. We use this ideal environment for determining the maximum accuracy of the three algorithms. The picture contains two scans: the reference scan taken at the upper right position and the current scan taken at the lower left position. The current scan has been displaced by a position offset of $(282mm, 282mm, 4.0°)$.

We ran this experiment with 1000 scans at both positions. Each scan has been taken by a simulated range finder with $\sigma = 25mm$ accuracy and an angle resolution of $0.5°$.

Algorithm	runtime
extended Cox	4s
extended *ccf*	9s
extended *idc*	10s

Figure 7. Runtimes on a transputer T805 30MHz.

Figure 7 shows the runtimes of the three scan matching algorithms. With our implementation the extended Cox algorithm is faster than the extended *ccf* and the extended *idc* algorithm. However the extended *ccf* algorithm has not been optimized and therefore faster implementations should be possible. For the reason of speed we use the faster extended Cox algorithm in the *combined scan matcher* whenever it is applicable.

Alg.	Δx	Δy	$\Delta \alpha$	σ_x	σ_y	σ_α
real	282mm	282mm	4.0°			
ext.Cox	283mm	281mm	4.0°	2.8mm	3.0mm	0.03°
ext.*ccf*	280mm	280mm	4.0°	0.0mm	0.0mm	0.00°
ext.*idc*	274mm	283mm	4.0°	3.6mm	3.1mm	0.04°

Figure 8. Comparison of calculated matches.

Figure 8 shows the calculated position updates by the three algorithms in form of mean value and error variance. Here the extended Cox algorithm gives the best mean value. The extended *ccf* algorithm returns an error variance of 0 in all components. This seems to show high accuracy, however, this is also the result of the discretization that is used in the algorithm.

We note that for the task of self-localization all three algorithms provide high accuracy.

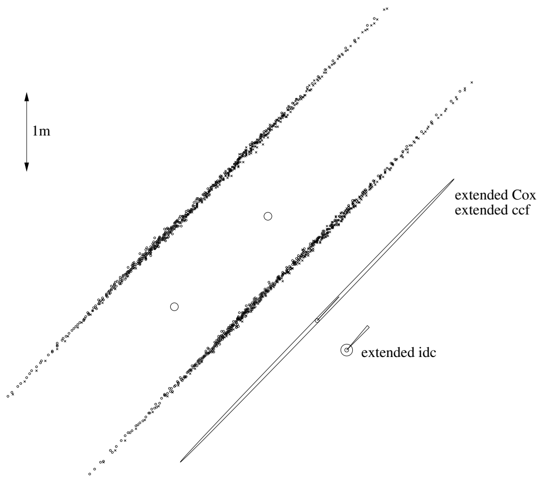

Figure 9. Comparison of calculated error covariance matrices.

We also compared the calculated error covariance matrices of the three algorithms. Therefore we used the scenario shown in figure 9. Two scans have been taken in a long hallway. In this scenario a self-localization can only estimate orientation and position towards the walls but not

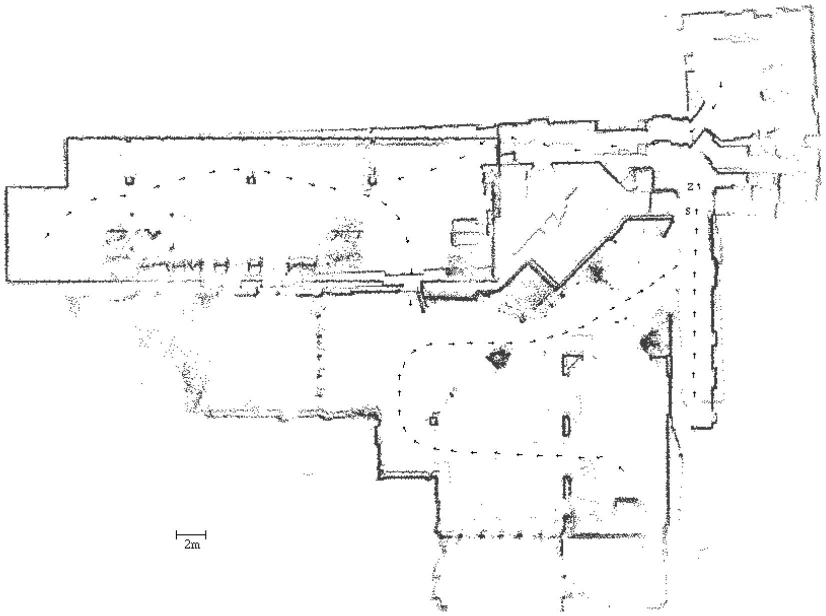

Figure 10. Set of scans. Positions have been estimated by dead-reckoning only.

along the walls. This has to be reflected by the calculated error covariance matrix.

Whereas the extended Cox and extended *ccf* algorithm produce good results, the extended *idc* algorithm falls into problems (error ellipse). These problems come from wrong point-to-point correspondences due to the used heuristic for solving the correspondence problem in the *idc* algorithm [3].

If we examine the *combined scan matcher* for the situation in figure 9 we notice that this situation is handled by the extended Cox algorithm since there are enough line segments in the scan. It is really difficult to find situations where there are only few line segments and the *idc* algorithm produces wrong error covariance matrices. Therefore the *combined scan matching* algorithm covers nearly every indoor situation.

4.2. Real data

Of course we tested the *combined scan matcher* with real data, too. Figure 10 shows a map of 74 scans whose positions have been estimated by dead-reckoning only. The scans have been taken by the range finder mounted on top of the AMOS platform. The accuracy of the range finder is about $\sigma = 25mm$ and the used angle resolution is $0.5°$.

The robot started at the position denoted with an S and moved down the hallway. At each position marked with a small arrow a scan has been taken. The robot explored the environment and moved back to the start position. But now due to errors in the dead-reckoning process the robot assumes it is at position Z. The distance from S to Z is about $1300mm$ and the orientation error of the robot is about $2.5°$.

After collecting scans we applied the global position estimation algorithm described in [7, 6]. For the pairwise scan matching algorithm we used our *combined scan matcher*.

Figure 11 shows the result of the global position estimation. The two positions S and Z have been merged and all scan positions have been determined in such a way that a consistent map is formed.

This map consists of single scans and can now be used as an a-priori map for the self-localization task.

The generated map also shows the quality of the *combined scan matching* algorithm in the final self-localization task. Since the *combined scan matcher* is used for both the calculation of a consistent map of high accuracy and the self-localization task, figure 11 shows that the same accuracy is obtained when this scan matcher is used in the self-localization module.

5. Conclusion

We presented a self-localization module based on 2d laser scans and a *combined scan matcher*. The combination of different scan matching algorithms allows robust self-localization in typical everyday indoor environments. These

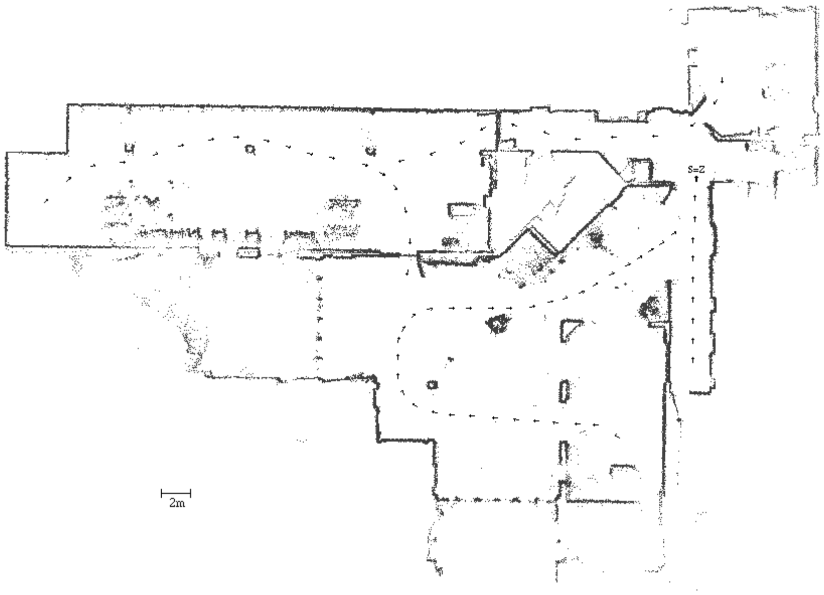

Figure 11. Set of scans after global position estimation.

include office-like structures as well as non-rectilinear regions. Real world experiments show that the *combined scan matcher* avoids problems where one of the underlying scan matchers solely may fail.

References

[1] S. Borthwick and H. Durrant-Whyte. Simultaneous localisation and map building for autonomous guided vehicles. In *IROS*, pages 761–768, 1994.

[2] I. Cox. Blanche – an experiment in guidance and navigation of an autonomous robot vehicle. *IEEE Trans. on Robotics and Automation*, 7(2):193–204, April 1991.

[3] J. Gutmann and F. Lu, 1995. Personal communication.

[4] M. Knick and F. J. Radermacher. Integration of subsymbolic and symbolic information processing in robot control. In *Third annual Conference on AL, Simulation and Planning in High Autonomy Systems*, pages 238–243. IEEE Computer Society Press, July 1992.

[5] M. Knick and C. Schlegel. AMOS: Active Perception of an Autonomous System. In *Proc. IEEE/RSJ Conf. on Intelligent Robots and Systems*, pages 281–289, September 1994.

[6] F. Lu. *Shape Registration using Optimization for Mobile Robot Navigation*. PhD thesis, University of Toronto, 1995.

[7] F. Lu and E. Milios. Optimal global pose estimation for consistent sensor data registration. In *IEEE Int. Conf. on Robotics and Automation*, 1995.

[8] G. Weiss and E. Puttkamer. A map based on laserscans without geometric interpretation. In U. R. et al., editor, *Intelligent Autonomous Systems*, pages 403–407. IOS Press, 1995.