Proc. 2004 IEEE Int. Conf. on Systems, Man & Cybernetics (SMC2004), The Hague, Netherlands, pp. 5398-5403, IEEE/Omnipress, ISBN 0-7803-8567-5

Robust Map Building for an Autonomous Robot using Low-cost Sensors*

Christof Schroeter, Hans-Joachim Boehme, Horst-Michael Gross
Ilmenau Technical University, Germany
christof.schroeter@tu-ilmenau.de

Abstract – *The paper describes an approach for building a map of an indoor environment with a mobile robot, using a combination of odometry and sonar range sensors. Aiming for real-time large scale mapping on low-cost platforms with limited sensory and computational equipment, we discard high-complexity techniques like probabilistic SLAM. Instead, the algorithms presented here involve odometry correction and automatic pose recalibration, enabling us to build a coherent map that can be used for navigation and self-localization. Experimental results from different environments prove the efficiency of our approach.*

Keywords: Odometry correction, localization, mapping

1 Introduction

Self-Localization and Map-Building are two of the key requirements for an autonomous mobile robot. The robot must know where it is located within a certain reference frame and it must be aware of its environment by means of an environment model (map). The purpose of the map is to enable a robot to act reasonably within its environment. Therefore, it must contain those features that are required for the desired behaviour. Since the behaviour of an autonomous robot is based on its sensorical input, the environment model preferrably should be built from sensor readings too, instead of providing a designer model.

For a mobile robot, localization and mapping bear mutual interdependencies. With the pose known, the robot can integrate sensor readings into a common frame as it moves around, iteratively updating its environment model. Likewise, with a known map, it is possible to determine the robot pose by comparison of sensor input and features stored in the map. In the absence of accurate pose information as well as a model of the environment, both localization and mapping present complex tasks.

In this paper we will focus on the aim of learning a consistent geometric map from sonar range readings, using odometry (egomotion measurements) as main pose information and combining it with internal map alignment and visual input for increased accuracy. The remainder of the paper is structured as follows. In section 2 we give an overview of related research work in the field of pose estimation. Section 3 explains in detail the algorithms used in our approach and demonstrates results of mapping different environments. Section 4 contains our conclusions and plans for further research.

2 Related Work

Usually, a mobile robot is equipped with wheel encoders that enable it to measure its own motion and determine the current position by dead-reackoning, i.e. integrating incremental motion information over time. Odometry is widely used because it provides high sampling rates and good short-term accuracy for a low price. However, it inevitably leads to accumulation of systematic and non-systematic measurement errors, resulting in a growing difference between believed and real position. Therefore, uncorrected odometry alone is not a reliable method for tracking a mobile robot's pose and building consistent large-scale maps.

Borenstein et al. described calibration for a robot's odometry system in [1]. Their approach focussed on identifying possible error sources for a differential drive robot and determining parameters to explicitly correct those errors. Martinelli [6] presented a strategy for experimentally determining the systematic and non-systematic errors in a synchro-drive robot, estimating error parameters from several observables.

External sensors have been used to increase the accuracy of odometry. A very popular approach is scan matching, where overlapping sensor readings from different locations are compared and matched to compensate for the error aquired while moving from one position to the next one [5]. Scan matching depends on high accuracy and resolution of the sensor for good matching results. Hence, usually laser scanners are used for this task.

As aforementioned, another possible approach to localization is using a given map to determine the current

position relative to known features by comparison with sensor readings. Although in this paper, we are explicitely considering a situation where no prior map is available, we give a short overview of such algorithms.

A simple method would be to provide positions of easily recognizable landmarks that can be detected by either range sensors or visual input. Localization depends on observation of one or more landmarks, while between observations the position is maintained by odometry. Newer approaches estimate probability distributions for the robot state that are updated with motion and observations, using probabilistic motion and observation models. Several algorithms like Kalman filters [7], grid-based Markov methods [3] and particle filters (Monte-Carlo-Localization, MCL) [2] have been used for probability estimation.

The chicken-and-egg problem of mapping and localization is addressed by Simultaneous Localization and Mapping (SLAM). In SLAM, mapping and localization are solved parallely observing and exploiting the mutual dependencies. The base of SLAM is an extended state space that contains the position as well as the map. A probability distribution over this common state space is maintained that converges with motion and observations [10], [8]. Most successful applications are using laser scanners, while visual SLAM and adaption to arbitrarily large environments are subjects of ongoing research.

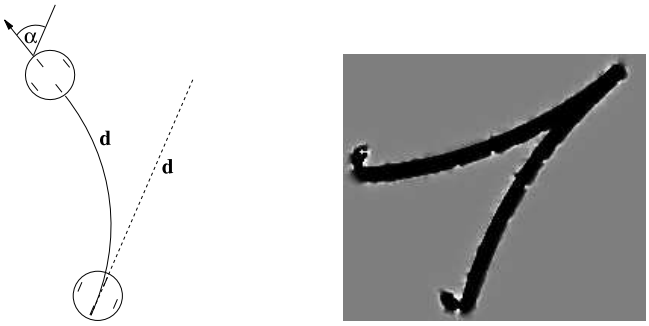

Figure 1: left: The robot drives a circular arc instead of a straight line. right: The map is distorted due to the wrong perception of the robot's own motion. Actually the environment consists of one straight corridor (black: free space, white: obstacles, grey: unexplored)

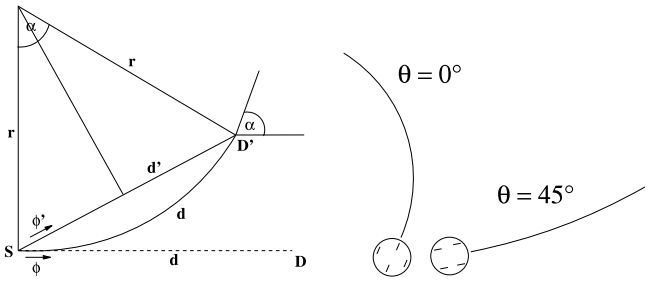

Figure 2: left: S is the point where the robot started a straight line movement. D is where the robot believes to be, when it is actually at D'. While the actual path is an arc, the new position can be calculated by assuming the robot went a direct line from S to D', calculating substitution parameters ϕ' and d'. right: The arc radius is expected to depend on the motion direction w.r.t. the wheel base.

3 Odometry Correction

3.1 Odometry Calibration

Our current experimental platform is a RWI B21r with 4 synchronously steered and driven wheels (synchro-drive). Translation and rotation are measured independently by incremental encoders. Our observations show that while odometry information is highly accurate concerning the driven distance, deviations in the believed orientation (motion direction) can become quite high. This is caused by the robot not driving straight lines but circular arcs when it should go ahead. A possible explanation for this behaviour would be different wear on the wheels. In contrast to differential drive, in the synchro-drive, as the robot turns the position of each wheel w.r.t. the heading direction changes, altering that specific wheels influence on the driving behaviour. Therefore, determining the systematic errors takes more effort than for the differential drive. On the other hand, experimental data suggests that the error can be described with a plain model, specifying an arc radius for each specific wheel direction.

Fig. 1 (left) shows what happens when the robot should go straight ahead, but the wheels cover different distances. The actual driven path is a circular arc, but the wheel configuration w.r.t. the driving direction does not change, leading the robot to believe it has gone straight ahead. The measured distance d is (almost) exactly the length of the arc. During this motion, the robot's heading direction w.r.t. an external reference frame has changed by angle α. Fig. 1 (right) demonstrates the effects of this behaviour on the built map. When the robot really drives a straight line (steered by a human), the odometry systems measures an arc motion, which results in straight corridors appearing bent in the map.

To correct these erroneous measurements, we need to determine the radius of the resulting arc. If the radius of the arc is known, we can compute the destination position from the driven distance d, by assuming a straight motion in direction ϕ' with distance d' (Fig. 2 left). We find:

$$\frac{d}{2\pi r} = \frac{\alpha}{2\pi} \tag{1}$$

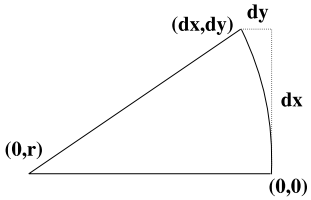

Figure 3: The arc radius can be computed from measured lengths dx and dy.

$$\alpha = \frac{d}{r} \tag{2}$$

$$sin(\frac{\alpha}{2}) = \frac{\frac{d'}{2}}{r} \tag{3}$$

$$d' = 2r * sin(\frac{\alpha}{2}) \tag{4}$$

$$\phi' = \phi + \frac{\alpha}{2} \tag{5}$$

Substituting d by d' and ϕ by ϕ' we can do a regular odometry position update now. However, we need an additional correction for the robot heading, as the drive direction no longer is the same as the heading direction. Note that with a perfect synchro-drive, the orientation of the wheel base w.r.t. an external reference frame never changes during regular operation, therefore the robot heading is always the same as the torsion towards the wheelbase. However, for our approach we are maintaining 2 parameters. ϕ describes the heading of the robot in the external frame and changes with translation (due to described measurement errors) as well as rotation. θ is the torsion of the robot, which changes only with rotation and affects the influence of each single wheel on the motion behaviour. Therefore θ will be needed for odometry correction. Fig. 2 (right) demonstrates how the arc changes with drive direction w.r.t. the wheelbase, as the varying wheel configuration determines the drive behaviour. For example assume one wheel has a smaller diameter than all the others. This will lead to strong arc motion if the wheel is located at the outer edge, while it will have almost no effect if the wheel is near the center line of the robot (in drive direction). Therefore we do not only need to determine the radius once, but must find a function mapping the drive direction θ to the arc radius. Our approach here was to sample the radius at a number of directions and interpolate between those sample points. The ideal method of measuring the radius of the robot path would be to have the robot move in a full circle and measure the diameter. Unfortunately, as the occuring radii are up to 2000m, this is only feasible with a very large planar area. Instead, we can only drive fragments of full circles and try to measure observables that allow computation of the arc radius. We propose 2 methods here.

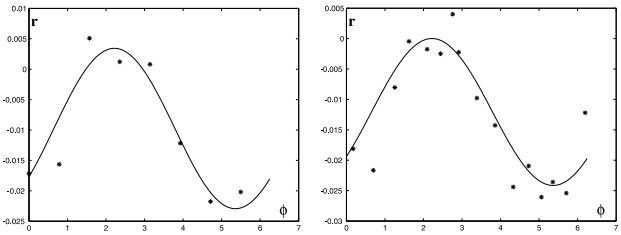

Figure 4: left: Sample points show 1/r as function of the heading direction, measured with method 1 (measurement of dx and dy). right: Same with method 2 (measurement of pathlength d and differential angle α

The first method is illustrated in Fig. 3. The robot is commanded to drive a straight line and the motion in x and y direction is measured. Without any error, it would only move in x direction. Without loss of generality we can assign coordinates (0,0) to the starting point and (dx, dy) to the end point. The center of the circle is (0,r). Inserting the end point into the circle equation we get

$$(dx - 0)^2 + (dy - r)^2 = r^2 \tag{6}$$

$$r = \frac{dx^2 + dy^2}{2dy} \tag{7}$$

This method only requires the measurement of 2 ranges, but also depends on very exact alignment of the robot and measurement, as we must ensure the measurement coordinate system matches the inital heading direction of the robot. Measurement results are shown in Fig. 4 (left).

The second method needs no alignment at all, but requires the measurement of angles. Using equation 2, we can calculate r by measuring the difference between initial and final heading along the arc path (α). d and α are marked in Fig. 2 (left).

$$r = \frac{d}{\alpha} \tag{8}$$

For direct angle measurement, the resolution that can be achieved with simple methods is lower than for measuring the ranges as in the first method. However, with increasing distance d the resulting error in the radius r is decreasing. Therefore, the second method should be preferred if a reasonably large area is available for experiments.

Fig. 4 (right) shows measurement samples taken with method 2. We have plotted the inverse $\frac{1}{r}$ as a function of the orientation here. In both plots it is obvious that the function approximately matches a sine wave. This supports the hypothesis that the errors originate from different conditions of the wheels, as those should become more or less significant with the wheels taking a

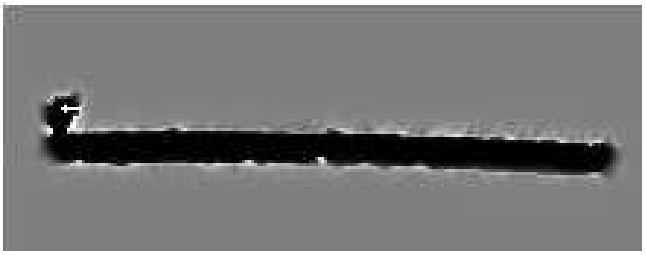

Figure 5: A map of the same area as Fig.1, with odometry correction applied.

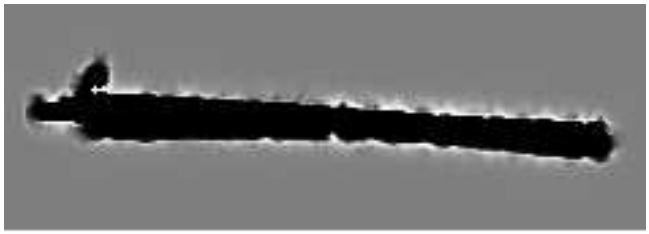

Figure 6: After travelling the corridor several times, position errors have grown enough to distort the map.

place near the center of the robot or closer to the edge (w.r.t. drive direction), following a sine curve as well. An additional best match sine curve is displayed in both results plots. It is also visible that both methods yield comparable results in the mean as well as the variance of measurements. We must be aware that the results contain impreciseness of measurements as well as non-systematic errors in the robot motion itself.

Fig. 5 shows the area from Fig. 1 again, this time mapped using the described odometry correction.

3.2 Map Matching

Fig. 6 shows the map of the corridor after the robot has travelled from one end to the other several times. It is clearly visible that the position error has grown large enough to destroy the consistency of the map. The odometry calibration from subsection 3.1 reduces the pose errors significantly, however the main problem of odometry remains: the position uncertainty will still grow unbounded over arbitrary time. Therefore, an additional recalibration of the robot's position belief is needed. The idea in our approach is that whenever the robot re-enters an area that has already been mapped before, a wrong position estimation leads to a discrepancy between the map and the environment as it is perceived by the robot's sensors. The idea is somewhat similar to scan matching, with the difference that we do not match consecutive measurements. Furthermore, since we are working with unreliable sonar range

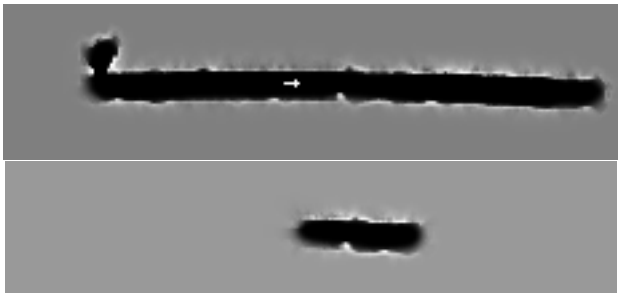

Figure 7: top: The static map contains the whole area that was already mapped by the robot. bottom: The temporary map is only built from the most recent measurements over a path of a few meters.

sensors, single scans do not seem very promising.

Instead, it seems reasonable to match maps against each other. The purpose of the map building algorithm is to compensate for erroneous measurements, making the resulting map much more reliable than single scans. To enable matching, we need to discriminate between a temporary map, containing only recent measurements within a certain travel distance window and a static map, containing all older measurements that are not part of the temporary map. This is implemented by using a queue where all measurements are stored together with the respective odometry position. The scans leave the queue and are integrated into the static map when the robot has moved away from that position by a certain distance. The temporary map is rebuilt each time from all the scans that are still in the queue.

As the robot moves along its path, the temporary map will run ahead of the static map, always covering the most recently crossed area. When the robot enters an area it has seen before, the static map will already contain that area. Only in that case there is an overlap between the static and temporary map. If the robot has aquired an error in position belief since the area was mapped last time, it will now assume a wrong position with reference to the static map, which leads to a displacement between the static and temporary map. By determining this displacement, we can calculate and then correct the position error.

In order to find the best match between the static and temporary map, we use a simple cross-correlation algorithm. Since errors can occur as shift in x- or y-direction or as rotation, the temporary map needs to be shifted and rotated by different values within certain bounds and a similarity measure has to be computed each time. The scalar product is used for that purpose here. The result is a 3-dimensional matrix containing a fit value for each combination of x-shift, y-shift and rotation $(dx, dy, d\phi)$.

$$C(dx, dy, d\phi) = \sum_{x,y} static(x, y) \cdot temporary(x', y') \quad (9)$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos(d\phi) & sin(d\phi) \\ -sin(d\phi) & cos(d\phi) \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} dx \\ dy \end{bmatrix}$$

One problem with the scalar product is that for "flat" matrices it produces wide maxima instead of narrow peaks. Since the maps usually consist of wide areas of free space (nearly same value), the results are very unspecific. If the matrices to match were merely shifted and rotated copies of each other, we would still get a distinctive peak for the true match. The mapping procedure may lead to slight differences between the maps though, especially since the temporary map does not contain very many scans. Overall, this often leads to bad matches especially in the rotation component.

Therefore it is preferrable not to use the map for matching as it is, but first extract certain features, avoiding areas of equal value. We chose the edges between obstacles and free space as features here (see Fig. 7). With the resulting sparse matrices, maxima are defined better in the correlation matrix.

Another problem is introduced by the typical structure of indoor environments. These mostly consist of long corridors with parallel walls. When trying to determine the best match between the maps, the correlation is very badly defined in the direction of the corridor. This may lead to the correlation maximum not reflecting the real position error. In such cases, the correlation matrix usually shows a wide maximum that stretches in one direction but is very narrow in the perpendicular direction. If it is possible to identify such situations we can apply correction only in the direction where the offset is well known (usually across the corridor) while delaying correction in the other direction until we get a more reliable estimation. The distribution in the correlation matrix can be described by calculating its eigenvectors and corresponding eigenvalues. Here we first look for the best rotation match, then calculate eigenvectors for the remaining 2d matrix (Fig. 8). The first eigenvector points at the direction of main distribution of matrix elements, the eigenvalue represents the length of this distribution. If the eigenvalue is very high, the correlation is badly defined. In that case, correction is only applied in the direction of the second eigenvector.

With the map gridsize known, the correlation can directly be converted into a position offset that is used for correction of the position belief. Since the scans in the queue are stored with the erroneous position belief, they need to be corrected too, with a correction factor decreasing over the distance to the current position.

The matching procedure is applied continiously in certain intervals. A correction is only calculated though

Figure 8: left: features in the temporary map, center: features in the static map, right: correlation matrix (2d, for best rotation angle $d\phi$, with eigenvectors highlighted)

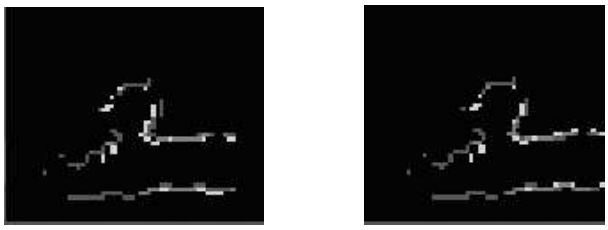

Figure 9: left: features from static (dark grey) and temporary map (light grey) overlayed without correction, right: features from static and temporary map overlayed after correction. A position error of about 1 pixel in x direction (about 20 cm) has been corrected.

if there is an overlap between the static and the temporary map (remember the static map runs behind the temporary map when the robot moves through an area for the first time) and when enough feature points (e.g. walls) are found. That way, whenever the robot enters an area it had mapped before, the position is recalibrated automatically. To build a consistent map, we must ensure we do return to a known area before the error in position becomes too large. In theory, a larger error can still be compensated by using a larger search window, but this significantly increases computational cost as well as the possibility of a false match.

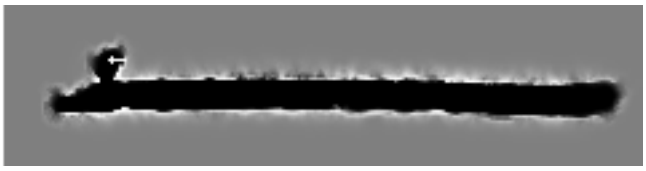

Figure 10: With map matching the resulting map is consistent, however remaining small errors between correction intervals lead to less details in the map.

Fig. 10 shows the same environment as Fig. 6, with the map matching correction. Due to slight shifts in position some minor details (like the door halfway down the corridor) may not be represented very well in the

map, but the position errors were kept low enough to build a consistent map.

3.3 Combination with Visual Odometry Correction

As was stated before, the larger the possible error in position belief, the higher is the computational cost for determining the true position. On the other hand, it is often impractical having to return to a known area too soon. Therefore, any additional correction method that further decreases the deviation of position belief from the true position is helpful. In our intended application, a service robot in a home store, we are applying an additional correction based on observation of the floor structure. The distinct structure is used as reference for the robot heading, ensuring true orientation belief at any time [9]. Since accuracy for translation measurement is significantly higher than for rotation measurement, this greatly improves the reliability of odometry. Furthermore, since orientation is correct at any time now, the map matching only needs to determine correlation in x and y dimension, since errors in ϕ do not occur.

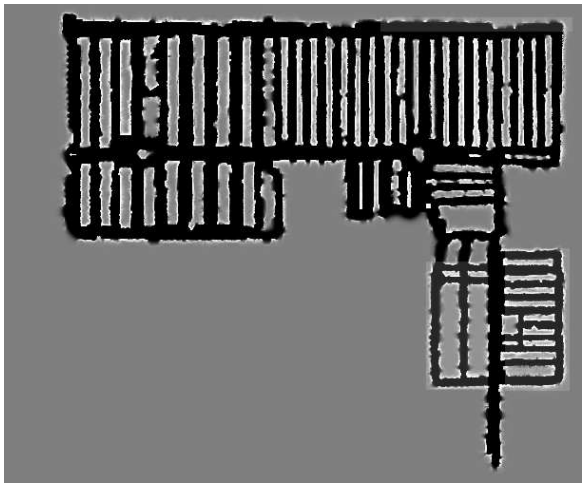

Figure 11: Dimensions of this map are about 100m * 100m.

4 Conclusions

We have presented an approach to odometry correction and map building with sonar sensors that allows us to build consistent maps of virtually arbitrarily large areas. The algorithms are computationally efficient and require no high precision sensors like laser range finders. A limitation of the algorithm is that the position belief is only recalibrated when a known area is re-entered. Position errors that occured between recalibration points may still have an effect on the map. Therefore, the robot should not move too far without closing a loop, returning to a known place.

The same algorithms can be used to localize a robot in a given map when its position is known approximately and to keep track of the correct position during motion within a known environment. We also intend to integrate the map match feature into MCL [4] for increased reliability and accuracy.

References

[1] J. Borenstein and L. Feng. Measurement and correction of systematic odometry errors in mobile robots. *IEEE Trans. on Robotics and Automation*, 12(5), 1996.

[2] D. Fox, W. Burgard, F. Dellert, and S. Thrun. Monte carlo localization: Efficient position estimation for mobile robots. In *Proc. of the AAAI Nat. Conf. on Artifical Intelligence*, 1999.

[3] D. Fox, W. Burgard, and S. Thrun. Markov localization for mobile robots in dynamic environments. *Journal of Art. Intelligence Research*, 11, 1999.

[4] H.-M. Gross, A. Koenig, H.-J. Boehme, and C. Schroeter. Vision-based monte carlo self-localization for a mobile service robot acting as shopping assistant in a home store. In *Proc. of the 2002 IEEE/RSJ Intl. Conf. on Intelligent Robots and System*, pp. 265–262, 2002.

[5] J.-Ste. Gutmann and C. Schlegel. Amos: Comparison of scan matching approaches for self-localization in indoor environments. In *Proc. of the 1st Euromicro Workshop on Advanced Mobile Robots (EUROBOT '96)*, 1996.

[6] A. Martinelli. The odometry error of a mobile robot with a synchronous drive system. *IEEE Trans. on Robotics and Automation*, 18(3):399–405, June 2002.

[7] P.S. Maybeck. The Kalman filter: An Introduction to Concepts. In I. Cox and G. Wilfong, editors, *Autonomous Robot Vehicles*, pp. 194–204. Springer-Verlag, 1990.

[8] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM:A factored solution to the simultaneous localization and mapping problem. In *Proc. of the AAAI Natl. Conf. on Artificial Intelligence*, 2002.

[9] C. Schroeter, H.-J. Boehme, and H.-M. Gross. Extracting of orientation from floor structure for odometry correction in mobile robotics. In *Proc. of the 25th Pattern Recognition Symposium (DAGM 2003*, 2003.

[10] R. Smith, M. Self, and P. Cheeseman. A stochastic map for uncertain spatial relationships. In *4th Intl. Symposium on Robotic Research*. MIT Press, 1987.