Sonar-based FastSLAM in an Underwater Environment Using Walls as Features

Dariush Forouher, Jan Hartmann, Marek Litza, Erik Maehle

***Abstract*— A lot of research has been done in the area of Simultaneous Localization and Mapping (SLAM). It is particularly challenging in underwater environments where many of the self-localization methods used at land no longer work. This paper presents a method to use a rotating scanning sonar as the sole sensor to perform SLAM in harbors or natural waterways. We present a feature extraction process which is capable of extracting walls of arbitrary shape. These extracted wall features are then used to perform SLAM using the well-known FastSLAM algorithm. We show that SLAM is possible given this type of sensor and using our feature extraction process. The algorithm was validated on an open water test site and will be shown to provide localization accuracy generally within the error of the GPS ground truth.**

I. Introduction

Underwater vehicles are used to move in marine environments for a while now. As many marine environments are hostile to humans, Remotely Operated Vehicles (ROVs) have been used successfully for many years to perform underwater tasks. They are, however, limited by the need of a human operator and usually a tether for communication. In recent years, therefore, interest in developing Autonomous Underwater Vehicles (AUVs) has increased [1], [2]. AUVs are promising because of the potential cost savings in omitting a human operator and in performing tasks too difficult or dangerous for people to do. Developing an AUV, however, is a complex task. One of the more important problems to be solved is localization. A robust method to achieve localization underwater is to deploy artificial landmarks with known positions, e.g. active sonar beacons or highly reflective sonar markers. The AUV can estimate its position relative to those beacons. There are situations, however, where setting up artificial landmarks is undesirable. An example is deploying an AUV in an previously unknown environment. Localizing a robot in such a scenario, without any artificial landmarks, is called self-localization. In such a scenario one has to rely on natural landmarks to achieve localization. Doing this in an unknown environment, having no map to rely on, is called Simultaneous Localization and Mapping (SLAM). SLAM as a theoretical problem is considered to be solved for a while now [3], and on land it has been shown to work in many scenarios. Implementing it in an underwater environment proves to be more difficult, due to limiting choices in sensors. The main motivation for us to develop SLAM on an AUV was to successfully perform in the Student Autonomous Underwater Vehicle Challenge - Europe (SAUC-E) underwater robotics competition [4]. In the competition an AUV has to perform tasks autonomously in a confined water basin. Localization is essential to move between the different tasks.

The authors are with the Institute of Computer Engineering at the University of Lübeck, Germany {forouher|hartmann|litza|maehle}@iti.uni-luebeck.de

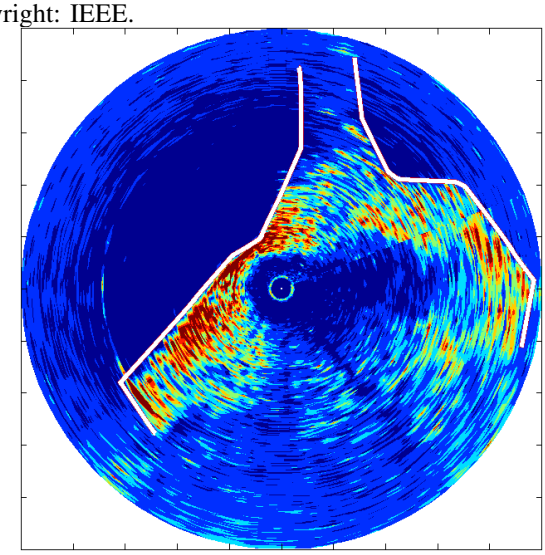


Fig. 1. A 360° sonar measurement with a range of 50 meters, taken at the test site. The robot is in the center. The white lines indicate the true location of the walls.

In this paper we describe how we implemented SLAM with an underwater vehicle with the goal to perform in environments similar to natural water ways, basins or harbor areas. Those areas often contain walls with sufficiently complex structures to enable localization. Our goal was to localize an AUV using a mechanically rotating scanning sonar, which is cheap compared to more sophisticated sonar devices. Furthermore we did not use any additional sensors, like Inertial Navigation Systems (INS) or Doppler Velocity Logs (DVL) for odometry, to avoid the cost of those devices. Our setup, with one of the least expensive sonar heads available, is currently likely one of the cheapest possible setups to perform SLAM in an underwater environment.

The sensor used is a wide beam imaging sonar[1] with a range of 50 meters. In our setup it worked with a resolution of 20 cm and rotated in 6 degree steps around its axis. Acquiring an 360 degree image took about 8 seconds using these parameters. Fig. 1 shows a typical sonar image. The most prominent features visible are usually the walls and

[1]Imagenex 852 Ultra-Miniature Scanning Sonar

reflections of the ground (compare Fig. 2 on p. 3).

The main challenge in solving the SLAM problem underwater lies with the feature extraction, as sonar images are noisy and significant post processing is required to extract wall features from them. Additionally, the limited update rate of the sonar constrains the maximum possible angular and linear speed of the robot. Any movement during sonar recordings will induce distortions. While stopping the vehicle during recordings might mitigate this, doing so is generally undesirable. We developed an algorithm that robustly extracts walls from raw sonar images (Fig. 1) and uses them for performing SLAM. The walls can be of arbitrary shape and are not restricted to straight lines or sharp corners. We perform localization solely using these wall features.

The remainder of this paper is structured as follows: First we describe related work (Section II), followed by an introduction into SLAM (Section III). Then we outline our feature extraction process (Section IV) and finish with experimental results (Section V) and the conclusion (Section VI).

II. Related work

Ribas et al. describe an approach to SLAM in [2] using a scanning sonar and an AUV. They extracted line features from sonar images using the *Hough transform* and used them as landmarks for localization. Distortions due to movement of the vehicle were corrected by estimating the movement of the robot with help of a DVL and correcting the sonar images accordingly. For cost reasons we neither used a DVL nor an INS, thus prohibiting a similar correction step. Further, our approach is more generic as it is not constrained to straight line features.

Williams [5] performed SLAM using artificial landmarks, which provide easily identifiable sonar returns as the primary source of localization information. We instead aimed for achieving localization without any artificial landmarks.

In [6] the authors describe an implementation of FastSLAM using occupancy grids. They mapped underwater archaeological sites with their SLAM implementation and used a sonar comparable to our one. The main difference to our work is that they used a grid-based approach, while we model the environment as a feature-based map.

III. FastSLAM

SLAM can formally be defined as estimating the robots path $s_{1:t}$ and the map θ using all previous observations $z_{1:t}$ and controls $u_{1:t}$:

$$p(s_{1:t}, \theta | z_{1:t}, u_{1:t}) \tag{1}$$

This is called the SLAM posterior. The robot's path and the map (a list of landmarks) combined form a high-dimensional state space. Estimating this high-dimensional state using only very limited information is a hard problem [7]. Many algorithms have been developed to estimate the SLAM posterior, including EKF-SLAM [8] and FastSLAM [9].

EKF-SLAM simplifies the problem by assuming the posterior to be approximately a Gaussian distribution. It uses one single Extended Kalman Filter (EKF) to estimate the posterior. While EKF-SLAM has achieved good results in many situations [8] it scales poorly with the number of landmarks and is brittle against data association errors [7]. Therefore we chose FastSLAM over EKF-SLAM for its better scalability and for its ability to track multiple hypotheses by using a particle filter.

In FastSLAM the SLAM posterior is factorized into [9]:

$$p(s_{1:t}, \theta | z_{1:t}, u_{1:t}) = \underbrace{p(s_{1:t} | z_{1:t}, u_{1:t})}_{\text{path posterior}} \underbrace{\prod_{n=1}^{N} p(\theta_n | s_{1:t}, z_{1:t}, u_{1:t})}_{\text{landmark estimators}} \tag{2}$$

This separation decorrelates the position of the landmarks from each other, and allows each landmark's position to be estimated independently. The path posterior is estimated using a particle filter, with each particle representing one possible path and one possible map. The position of the landmarks are estimated using an EKF [8]. Each particle contains one EKF for each landmark in its map. If one uses M particles and N landmarks (on average, the number may vary between particles), the overall computational complexity is $\mathcal{O}(MN)$. FastSLAM therefore scales linearly with the number of landmarks [9].

SLAM is usually represented recursively as a Markov process [7]. To implement a SLAM algorithm one needs to define a motion model g, which describes the movement of the robot between two states and an observation model h, which describes what observations are to be expected based on the current state. We modeled the motion as a Gaussian distribution

$$s_t = g(s_{t-1}, u_t) + \epsilon = s_{t-1} + \epsilon. \tag{3}$$

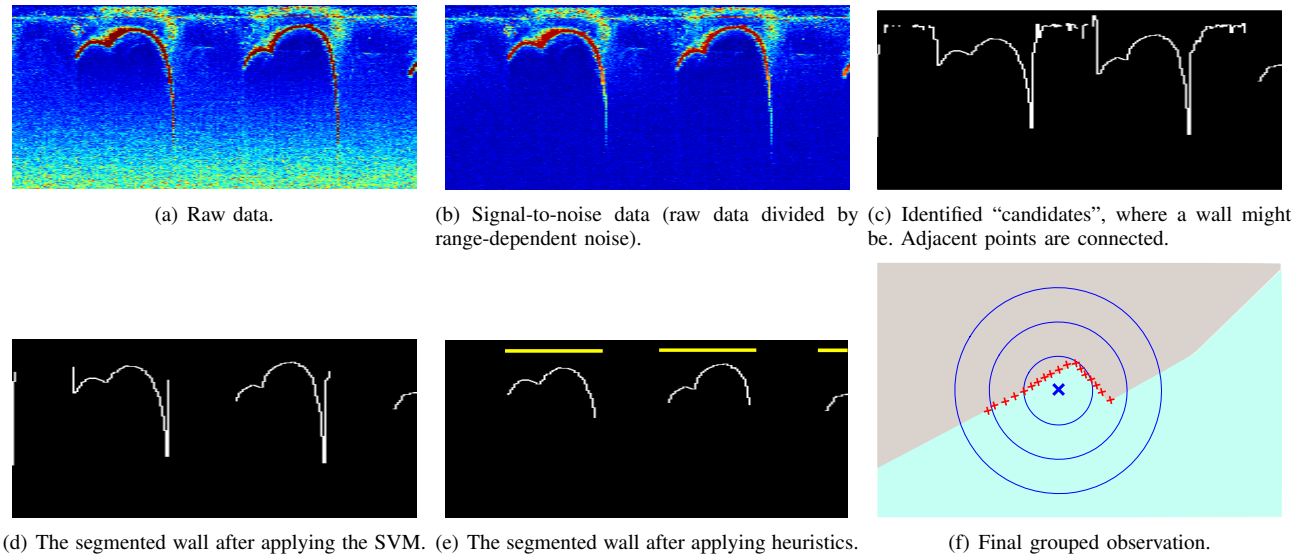
ϵ represents a sample of a Gaussian distribution [8]. Our motion model is based on the maximum speed the robot may experience during its run. We therefore derived the covariance of the Gaussian distribution from that maximum speed. Refining the motion model by including more sensor information, like heading from a magnetic compass, is possible. However, for the purpose of this paper we avoided the additional process of sensor fusion.

The observations were modeled as points with range and bearing. The error was modeled as

$$z_t = h(s_t) + \delta. \tag{4}$$

Again δ represents a sample of a Gaussian distribution. A single observation is thus a Gaussian distribution centered around the measured position.

An important subproblem of SLAM is data association, which is the mapping of observations to landmarks. In this paper we represent landmarks as points in a 2-D map without any signature. This makes data association non-trivial. We use a simple nearest-neighbor approach to data association.



(a) Raw data.

(b) Signal-to-noise data (raw data divided by range-dependent noise).

(c) Identified "candidates", where a wall might be. Adjacent points are connected.

(d) The segmented wall after applying the SVM.

(e) The segmented wall after applying heuristics.

(f) Final grouped observation.

Fig. 2. Sonar recording taken at the Media docks, Lübeck, in polar coordinates. Each column is a sonar echo response, with the nearest responses at the top. The horizontal axis indicates the progress of time, as the sonar head rotates around its axis. About 2.5 full rotations of the sonar are visible in this image. Therefore the same wall is seen multiple times. (a) At the beginning the raw sonar data contains range-depending background noise. (b) After filtering out this noise the signal-to-noise ratio remains. (c) Using a fixed threshold wall candidates are identified. Neighboring points are drawn by a connected line (d) The classifier removes as many false-positives as possible. (e) Additional heuristics remove more points. (f) The final observation, a group of points. The horizontal lines depicted in (e) marks the observations into which points are grouped together.

If there exists a landmark within a certain distance of the observation, the observation is associated with that landmark. If there is no landmark in the vicinity, a new landmark is created at the position of the observation. The data association is not mutually exclusive, i.e. a landmark may be matched to multiple observations. The rationale behind this is that neither observations nor landmarks represent single objects, but are always part of a cloud of points representing a wall. This data association is conceptually simple and results in many cases of miss-associations if the particle is badly placed. This is compensated, however, by the large amount of particles we deploy (> 1000). The particle filter weeds out the particles with bad data association in its resampling step [8].

IV. Feature Extraction

Images recorded with a scanning sonar contain lots of different types of noise. Robustly extracting a particular type of feature is critical. We decided to use walls as landmarks as they are the most prominent and easy to extract features in a sonar image (compare Fig. 2). For localization based on walls to succeed, the environment needs to contain walls of sufficient complexity, i.e. there have to be multiple walls in sight, a corner or at least a wall curved significantly. This constrains the environments in where the robot may be deployed to areas similar to harbor areas or small artificial water basins.

In this context we define a wall as a structure which stands out in a sonar image by its strong sonar echo and the property that "behind" a wall the signal-to-noise ratio (SNR) drops significantly. We have developed a filter chain that exploits both these properties to extract walls robustly. The filter chain analyzes scan results one by one and avoids expensive feature detection algorithms like the Hough Transform [10]. This makes the filter chain easy to implement and very fast.

A. Filter Chain

Fig. 2 gives an overview of the filter chain. The steps are now described in more detail:

1) First the SNR is calculated by using previously known noise-levels (which depend on the specific parameter settings of the sonar head but are environment-independent).
2) Then "wall candidates" are identified, denoted by x_w^t. They are found by going through a scan from far to near and stopping at the first echo that exceeds a certain SNR threshold (Fig. 3). This step exploits the fact that walls usually reflect all sonar energy. Behind a wall the sonar image will thus appear "dark". In identifying a wall, this step has negligible false-negatives, but it produces a large amount of false-positives, especially in the face of noise.
3) Thus the next filter step attempts to remove those false-positives by applying a classifier. We decided to use a support vector machine (SVM) [11] for this, and created a list of 9 features to be used by it. The SVM is trained by a manually classified training set. Currently this training set must be created for each environment for sufficient classification performance.
4) Following the SVM classification several heuristics are applied to remove more false-positives. This is based on the insight that incorporating spurious (i.e. erroneous) landmarks have a greater effect on the SLAM performance than missing a few good ones.
5) Finally adjacent wall points are grouped together as a combined cloud of points and handed over to the

SLAM algorithm. This grouping simplifies data association: As walls are made up of clouds of points, taking just one point and trying to match it to a landmark would have been infeasible. Therefore many points are used to make data association robust.

B. Support Vector Machine

The SVM classifies sonar returns based on nine features $(F_1, F_2, ..., F_9)$, which have been carefully selected. These features are:

1) Find the highest peak in the vicinity of the wall candidate ($k = 20$ indicates the window size):

$$x_p = \underset{x \in [x_w^t - k, w_w^t + k]}{\arg\max} \; SNR_t(x) \tag{5}$$

And use the maximum of the signal at that peak as a feature:

$$F_1 = SNR_t(x_p) \tag{6}$$

2) The ratio of the mean signal strength in front vs. behind of the wall candidate (P being the farthest data point):

$$F_2 = \frac{mean\{SNR_t(\delta) | \delta \in [1, x_w^t]\}}{mean\{SNR_t(\delta) | \delta \in [x_w^t + 1, P]\}} \tag{7}$$

3) The ratio of the signal variance in front vs. behind of the wall candidate:

$$F_3 = \frac{var\{SNR_t(\delta) | \delta \in [1, x_w^t]\}}{var\{SNR_t(\delta) | \delta \in [x_w^t + 1, P]\}} \tag{8}$$

4) The distance itself:

$$F_4 = x_w^t \tag{9}$$

5) The difference in distance between the previous wall candidate and the current one (providing there is a previous wall candidate):

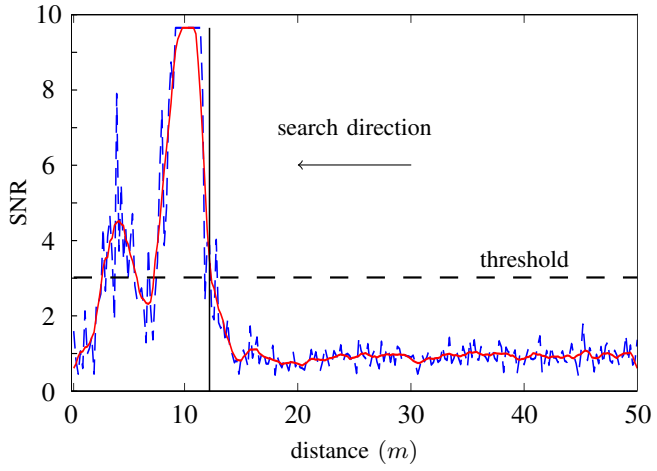


Fig. 3. This plot illustrates how a wall candidate is found. The dashed line denotes the SNR, while the solid line represents the SNR smoothened with a Gaussian kernel. A scan line is swept from the right to left and matches the first point which crosses the SNR threshold (dashed).

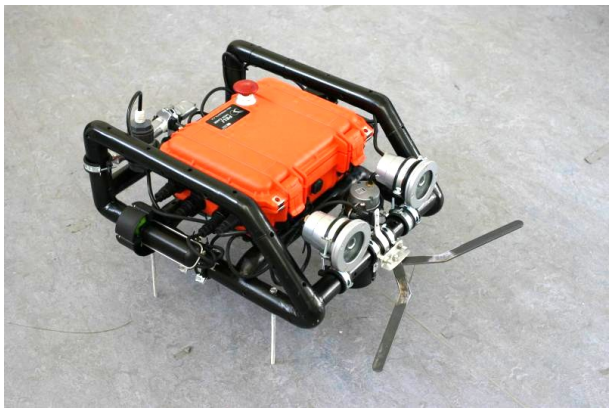


Fig. 4. AUV HANSE, which was used for the tests.

$$F_5 = x_w^t - x_w^{t-1} \tag{10}$$

6) The mean over a window centered around the peak x_p (with $k_p = 5$ again denoting a window size):

$$F_6 = mean\{SNR_t(x_p + \delta) | \delta \in [-k_p, k_p]\} \tag{11}$$

7) Similarly calculate the variance:

$$F_7 = var\{SNR_t(x_p + \delta) | \delta \in [-k_p, k_p]\} \tag{12}$$

8) Calculate the mean over a window centered around the wall candidate (with $k_m = 3$ again denoting a window size):

$$F_8 = mean\{SNR_t(x_w^t + \delta) | \delta \in [-k_m, k_m]\} \tag{13}$$

9) Similarly calculate the variance:

$$F_9 = var\{SNR_t(x_w^t + \delta) | \delta \in [-k_m, k_m]\} \tag{14}$$

C. Heuristics

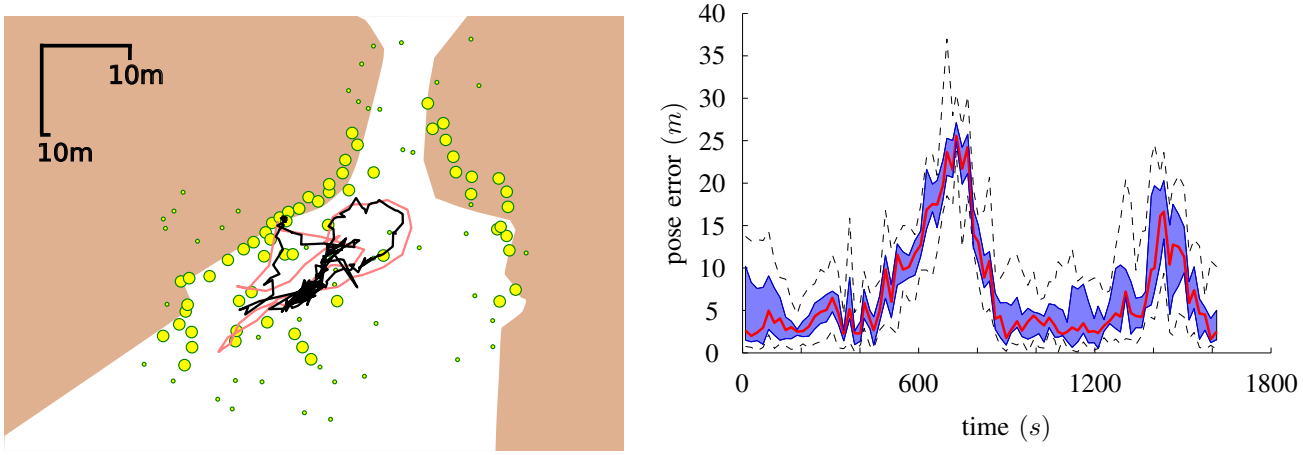
After the classification, two heuristics are applied to further reduce the amount of false-positives. First, very short wall segments are removed, as they are unlikely in the environments we encountered. Secondly, walls are generally continuous, i.e. sudden jumps in a line of wall points are unlikely. Therefore wall points are removed if they feature a high variance, i.e. if they have an unusually large distance to their next wall point neighbors.

V. Evaluation Results

The implementation was done in Matlab. For FastSLAM we relied on an existing Matlab implementation by Tim Bailey[2]. The map was stored in an kd-tree, for which we used the implementation by Steven Michael[3]. While developing

[2]FastSLAM 1.0 implementation, `http://www-personal.acfr.usyd.edu.au/tbailey`

[3]`http://mathworks.com/matlabcentral/fileexchange/7030`



(a) Estimated robot path (black) vs. ground truth (gray).

(b) Boxplot of the pose error, average over 10 iterations.

Fig. 5. Test results, 2000 particles were used. (a) The estimated path is painted black, the GPS path is colored gray. The circles visualize landmarks. The size of the circles indicates the confidence in it. A large landmark was observed often and has a small covariance, whereas a small landmark was observed only a few times and still has a large covariance. The largest deviation from the ground truth occurred after 600-800 seconds, where an entire path segment had been badly placed.

the algorithm, we evaluated it on two test sets. Both provided comparable results, thus we only present the results of one site in this paper. The only parameter that had to be changed between the test sets (besides the SVM, which had to be retrained) was the control model. This was necessary as the recordings were done with different average speeds.

As the test platform we used the AUV HANSE (Fig. 4), which was developed by a team of students at the Institute for Computer Engineering at the University of Lübeck to participate in the SAUC-E 2009 and 2010 underwater robotics competitions [4]. HANSE's frame is made of PVC pipes, onto which all individual components are attached. This design provides freedom for modifications. The main electronics are contained in a water-proof carrying case, attached to the frame. HANSE has four thrusters for motion control, one downward looking camera for floor observation, a stereo camera system, a pressure sensor and the scanning sonar used in this paper. It also has a microelectromechanical systems (MEMS) magnetic compass, accelerometers and gyroscopes, neither of which were used in this paper. The central processing unit is an off-the-shelf notebook, running a custom written software framework to perform autonomous tasks during the competition. HANSE can be remotely monitored and controlled by tether or by using Wifi, when being near the surface.

The test set was obtained by driving HANSE manually in the target area[4]. During the run, sonar recordings and GPS measurements were taken simultaneously (see Fig. 5(a) for a map of the area). The speed averaged at about 0.5 km/h. Special care was taken to avoid sudden movements of the AUV to minimize the distortions on the sonar images. The whole run took about 25 minutes. The SVM used in the feature extraction process was trained on a manually classified subset of 1397 out of 15167 total scans. Fig. 5(a) visualizes the results of performing SLAM. Shown is the recorded GPS path against the estimated path. Fig. 5(b) shows the pose error plotted over time. Most of the time the error stayed within 5 meters and only on two occasions the error did increase significantly. Both were caused by insufficient extraction of wall features (the average error over the full run was 7 meters). As seen in Fig. 5(a), the robot moved south-west (ground truth, gray), while the SLAM estimation (black) kept estimating the position farther north-east. The reason for this mislocalization was that in this area only one straight wall was successfully extracted. That is enough to estimate the distance to the wall, but not enough to estimate the position parallel to the wall.

A limitation of the performance is the increased variance of the pose error. While the median error stayed low for a high percentage of iterations, sometimes the SLAM algorithm diverged and produced a wrong map.

[4]Latitude/longitude: 53.862686, 10.703795

VI. Conclusion and Future Work

In this paper we showed that extracting walls from sonar data is feasible even in natural environments without straight walls. The algorithm is sufficiently robust to work even in the face of strong ground noise.

The SLAM algorithm is able to construct a map and localize the vehicle in it without using any odometry. The only constraint on the environment is that is has to contain walls of sufficiently complex shape. No artificial landmarks are needed. By using feature-based maps the algorithm stays computationally feasible. These results were validated by realistic test runs in open-water basins. The average localization error compared with GPS was about 7 meters.

Despite the results shown in this work, there remain problems to be solved. The feature extraction algorithm puts constraints on the environment, especially the need of characteristic walls. It would be very useful to either extend it to lift the constraints or to combine it with other types

of landmarks. This might allow localization in more generic environments and larger areas. The latter would also allow to explore the limits of the SLAM algorithm in terms of larger (and more gradually explored) maps.

The map model leaves room for improvement. Map pruning and incorporating negative evidence (the information that a certain landmark was *not* observed) might improve the quality of the map and reduce the likelihood that the particle filter might diverge.

Work has been done to combine particle filters with occupancy grids while remaining computationally efficient [12]. It would be interesting to compare our feature-based approach to an occupancy grid representation.

REFERENCES

[1] F. Maurelli, S. Krupinski, Y. Petillot, and J. Salvi, "A particle filter approach for AUV localization," in *OCEANS 2008*, 2009, pp. 1–7.

[2] D. Ribas, P. Ridao, J. D. Tardos, and J. Neira, "Underwater SLAM in a marina environment," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, 2007, pp. 1455–1460.

[3] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (SLAM): part II," *IEEE Robotics & Automation Magazine*, vol. 13, no. 3, pp. 108–117, 2006.

[4] C. Osterloh, D. Forouher, J. Hartmann, M. Litza, B. Meyer, J. H. Kluessendorf, T. Tosik, and E. Maehle, "HANSE - autonomous underwater vehicle for the SAUC-E competition 2010," 2010. [Online]. Available: http://www.iti.uni-luebeck.de/fileadmin/user_upload/Paper/sauce_2010.pdf

[5] S. B. Williams, "Efficient solutions to autonomous mapping and navigation problems," Ph.D. dissertation, University of Sydney, 2001.

[6] C. M. Clark, C. S. Olstad, K. Buhagiar, and T. Gambin, "Archaeology via underwater robots: Mapping and localization within maltese cistern systems," in *Control, Automation, Robotics and Vision, 2008. ICARCV 2008. 10th International Conference on*, 2009, pp. 662–667.

[7] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part i," *IEEE Robotics & Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[8] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT Press, 2005.

[9] M. Montemerlo, S. Thrun, and B. Siciliano, *FastSLAM: A scalable method for the simultaneous localization and mapping problem in robotics*. Springer Verlag, 2007.

[10] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Commun. ACM*, vol. 15, no. 1, pp. 11–15, 1972.

[11] S. Theodoridis and K. Koutroumbas, *Pattern recognition*. Academic Press, 2006.

[12] A. I. Eliazar and R. Parr, "DP-SLAM 2.0," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 2, 2005, pp. 1314–1320.