Mapping and Localization with Sparse Range Data

Jochen Schmidt, Chee K. Wong, and Wai K. Yeap
Institute for Information Technology Research, Auckland University of Technology
PO Box 12397, Auckland, New Zealand
{jochen.schmidt, chee.wong, wai.yeap}@aut.ac.nz

Abstract

We present an approach for indoor mapping and localization with a mobile robot using sparse range data, without the need for solving the SLAM problem. The paper consists of two main parts. First, a split and merge based method for dividing a given metric map into distinct regions is presented, thus creating a topological map in a metric framework. Spatial information extracted from this map is then used for self-localization. The robot computes local confidence maps for two simple localization strategies based on distance and relative orientation of regions. The local confidence maps are then fused using an approach adapted from computer vision to produce overall confidence maps. Experiments on data acquired by mobile robots equipped with sonar sensors are presented.

Keywords: Mobile Robots, Mapping, Localization

1 Introduction

Mapping and self-localization play an important role when using mobile robots for the exploration of an unknown environment. Particularly for indoor applications, where a 2-D map is usually sufficient, geometric maps obtained from time-of-flight devices, such as laser or sonar, are widely used. In this paper, we present an algorithm for mapping and localization which was developed originally for testing ideas about cognitive mapping [1]. In the latter, the concern is less on accuracy and speed but more on developing an appropriate process for testing ideas about cognitive mapping (as opposed to robot mapping). In particular, in cognitive mapping, the initial map computed is fragmented and highly inaccurate. Nonetheless, we believe our algorithm developed could be useful for roboticists who are interested only in the mapping and localization problem. Consequently, in this paper we present our approach in that light and show further results. From a robot mapping perspective, what is most interesting about this work is that the robot can localize itself even with a map that is highly inaccurate in metric terms. In the first part of the paper a method for dividing a given metric map into distinct regions, e. g., corridors or rooms is presented, thus creating a metric-topological map. The region splitting is based on a metric map, which we obtain from sonar sensor readings that are processed into a map consisting of geometric boundaries. This is a more compact representation of the environment compared to grid maps, while details such as size and shape of objects are maintained. As we use only two sonar sensors, the available range data is very sparse, therefore making the map highly inaccurate in certain areas. We will show that this sparse and inaccurate data can nevertheless be used for self-localization. Our work is inspired by [2], where a cognitive map is regarded as a network of local spaces, each space described by its shape and its exits to other local spaces. Related approaches can be found, e. g., in [3], which is a hybrid approach that combines topological and metric maps and identifies gateways and path fragments. In [4], topological maps are constructed from grid maps using Voronoi diagrams; the grid maps are split into regions and critical lines (gateways) are detected. In contrast to these methods, our approach is based on a region split and merge algorithm [5, 6, 7]. Many algorithms have been developed to solve the simultaneous mapping and localization problem (SLAM) for mobile robots. For some examples of recent work in this area see [8] and the references therein. The approach followed in this paper (based on [2]), is different, as we do not solve the SLAM problem, but simulate a cognitive mapping process instead. The latter refers to the process in which humans and animals learn about their environment [9]. Our mobile robot explores the environment, moving from A to B, and computes the topological map mentioned earlier. The question we ask is: how does the robot use the map to find its way home from B to A? Recall that the map computed is inaccurate due to sensor errors and sparse sampling. Therefore, the map computed in the reverse journey will be different from the original one. We implemented two localization strategies using both distance and orientation information extracted from the metric-topological representation. We show how local confidence maps can be computed using both strategies, and show how they can be fused into a global map based on the *Democratic Integration* approach [10, 11], which originated in image processing.

The paper is structured as follows: Section 2 describes the process used for generating a metric-topological map based on sparse range data, while Sect. 3 shows how to use the information extracted from this map can be used for localization. Experimental results are presented in Sect. 4.

2 Mapping

The mapping process described in this section is used in two ways: First, the robot explores its environment and collects data. When this is finished, all acquired data are processed; the result of this initial mapping stage will be called the *original map* further on. This is the map the robot will use for returning home. On its way home, the robot basically performs the same data gathering and processing steps as described here, except that the data is processed each time the robot stops, in contrast to building the original map, where the data is processed only at the end of the data gathering stage. An overview over the map-processing algorithm will be given in the following. A more detailed evaluation of the algorithm including the influence of the parameters involved is given in [1].

2.1 Data Acquisition and Pre-Processing

For data acquisition we use a mobile robot equipped with eight sonar sensors and an odometer; as the emphasis in this paper is on processing sparse range data, only the two side sensors are actually used. However, it is important to note that the algorithms presented here are not restricted to sparse data or that type of sensors. In fact, the performance will be even better when more and densely sampled range data (e. g., from laser) are available. The robot acquires sonar readings while moving on a straight line (as far as the drift allows) until it runs into an obstacle. At this point an obstacle avoidance algorithm is used, after which the robot can wander straight on again. A single one of these straight movements will be called *robot path* throughout this paper. We store the sonar readings separately for each robot path; this is not mandatory for the presented methods, but it simplifies processing later on. Based on the raw sonar sensor readings we build a simplified geometric map containing the robot movement path as well as linear surfaces approximated from the sonar data. In a first step, the recorded sonar data is low-pass filtered and converted to surfaces, being a piecewise linear approximation of the sonar distances. The surfaces are simplified further by grouping them, thus removing small gaps. The pre-requisite for the algorithm presented in the following is a geometric map that contains the robot movement path as well as surfaces in terms of line approximations of the original range sensor data. The goal is to split the map into distinct regions, e. g., corridors and rooms. Splitting is done along the robot movement path, using an objective function that computes the quality of a region, based on criteria such as the average room width (corridors are long and narrow compared to rooms) and overall direction (e. g., a corridor is separated from another one by a sharp bend in the wall). Additionally, a regularization term is used in order to avoid the formation of very small regions, which may originate from missing (gaps) or unreliable sensor data.

2.2 Split and Merge

The basis of the map-processing algorithm is the well-known split and merge method [7, 6, 5]. In pattern recognition this algorithm is traditionally used for finding piecewise linear approximations of a set of contour points. Other applications include segmentation of image regions given a homogeneity criterion, e. g., with respect to color or texture [5]. The pre-requisite for applying split and merge is an ordered set of (contour) points, which is to be approximated. For this purpose a parametric family of functions $\mathscr{F}$ (e. g., lines) has to be chosen, as well as a metric for computing the residual error ε of the approximation (e. g., mean square error), or, when used for regions, a homogeneity or quality criterion. The result of the algorithm is a piecewise approximation of the original points, where every single residual error is below a given threshold θ_l. The single steps of the algorithm are as follows [5]:

1. **Start** with an initial set of points $\mathscr{S}^0$, which consists of n_0 parts, $\mathscr{S}^0 = \{\mathscr{S}^0_0, \ldots, \mathscr{S}^0_{n_0-1}\}$. Each part $\mathscr{S}^0_i$ is approximated by a function from $\mathscr{F}$. Compute the initial residual error ε^0_i for each part of $\mathscr{S}^0$.
2. **Split** each part $\mathscr{S}^k_i$ where $\varepsilon^k_i > \theta_\mathrm{l}$ into two parts $\mathscr{S}^{k+1}_j$ and $\mathscr{S}^{k+1}_{j+1}$, compute the approximation and residuals ε^{k+1}_j, ε^{k+1}_{j+1}. Repeat until $\varepsilon^k_i \leq \theta_\mathrm{l}\ \forall i = 0, \ldots, n_k - 1$.
3. **Merge** two adjacent parts $\mathscr{S}^k_i$, $\mathscr{S}^k_{i+1}$ into one new part $\mathscr{S}^{k+1}_j$ if $\varepsilon^{k+1}_j \leq \theta_\mathrm{l}$. Repeat until merging is not possible any more.
4. **Shift** the split point shared by two adjacent parts $\mathscr{S}^k_i$, $\mathscr{S}^k_{i+1}$ to left and right while leaving the overall number of parts fixed. Keep the split that reduces the overall error, repeat until no further changes occur.

2.3 Splitting the Map

Before a region split and merge algorithm on the geometric map can be applied, it is necessary to create an initial split of the map. The easiest way to do so is to treat the whole map as a single large region defined by the start and end points of the journey. More sophisticated initializations can be used as well, e. g., based solely on the robot movement without taking into account range data [1]. After the initialization step, the actual division of the map into distinct regions is performed based on a split and merge that uses a residual error function $g(\mathscr{S}_i, \mathscr{S}_j)$ which compares two regions $\mathscr{S}_i$ and $\mathscr{S}_j$ and computes the homogeneity of the two regions (low values of $g(\mathscr{S}_i, \mathscr{S}_j)$ means homogeneous, high values very inhomogeneous). This function is used during the split phase for deciding whether a region $\mathscr{S}^k_i$ will be split again at a given position into two new regions $\mathscr{S}^{k+1}_j$ and $\mathscr{S}^{k+1}_{j+1}$, and in the merge (or shift) phase to determine whether two adjacent regions can be merged (or the splitting point be shifted).

When the homogeneity is above a given threshold θ_r, the region will be split again (or not merged/shifted). The quality of a region now incorporates both, robot path as well as the range data. The basic idea is to use the average width of a region in the map as a criterion for splitting, as a width change resembles a changing environment, e. g., a transition from a corridor to a big room. The homogeneity (residual) function used is:

$$g(\mathscr{S}_i, \mathscr{S}_j) = \frac{\max\{f_w(\mathscr{S}_i), f_w(\mathscr{S}_j)\}}{\min\{f_w(\mathscr{S}_i), f_w(\mathscr{S}_j)\}} + s_r r(\mathscr{S}_i, \mathscr{S}_j) \quad (1)$$

where $f_w(\mathscr{S}_i)$ is the average width of region $\mathscr{S}_i$, and $r(\mathscr{S}_i, \mathscr{S}_j)$ is a regularization term that takes care of additional constraints during splitting. The average width is given by $f_w(\mathscr{S}_i) = \frac{A_{\mathscr{S}_i}}{l_{\mathscr{S}_i}}$, where $A_{\mathscr{S}_i}$ is the area of region $\mathscr{S}_i$, and $l_{\mathscr{S}_i}$ is its length. In practice, the computation of both needs a bit of attention. Particularly the definition of the length of a region is not always obvious, but can be handled using the robot movement paths, which are part of each region. The length $l_{\mathscr{S}_i}$ is then defined by the length of the line connecting the start point of the first robot path of a region and the end point of the last path of the region. This is a simple way to approximate a region's length without much disturbance caused by zig-zag movement of the robot during mapping.

Regarding the area computation, the gaps contained in the map have to be taken into account, either by closing all gaps, or by using a fixed maximum distance for gaps. Both approaches have their advantages as well as drawbacks, e. g., closing a gap is good when it originated from missing sensor data, but may distort the splitting result when the gap is an actual part of the environment, thus enlarging a room. We decided to use a combined approach, i. e., small gaps are closed in a pre-processing step already, while large ones are treated as distant surfaces.

The regularization term $r(\mathscr{S}_i, \mathscr{S}_j)$ ensures that the regions do not get too small. In contrast to a threshold, which is a clear decision, a regularization term penalizes small regions but still allows to create them if the overall quality is very good. We use a sigmoid function that can have values between -1 and 0, centered at n, which is the desired minimum size of a region:

$$r(\mathscr{S}_i, \mathscr{S}_j) = \frac{1}{1 + \exp\left(-\frac{\min\{A_{\mathscr{S}_i}, A_{\mathscr{S}_j}\}}{A_{\max}} + n\right)} - 1\,. \quad (2)$$

The exponent is basically the area of the smaller region in relation to the maximum area $A_{\max}$ of the smallest allowed region. Thus, the smallest ratio is 1, and it increases when the region gets larger. This term only has an influence on small regions, making them less likely to be split again, while it has virtually no influence when the region is large, as the sigmoid reaches 0 asymptotically.

The influence of the regularization can be controlled using the factor s_r (see (1)), which is given by $s_r = s\theta_r$, where $0 \leq s \leq 1$ is set manually and defines the percentage of the threshold θ_r that is to be used as a weight. θ_r is the threshold mentioned earlier, which determines that a region should be split into two when the first region is θ_r times as large as the second one.

3 Localization

Once the original map has been generated, we instruct the robot to return home based on the acquired map. In the following, we will describe the strategies that we use for localization based on this information, and a data fusion algorithm that allows for an overall position estimate computed from the single localization methods. On its way home, the robot basically performs the same data gathering and processing steps as described in Sect. 2. Each time the robot stops on its return journey, which is normally because of an obstacle in its way, it performs a map processing. Therefore, at each of these intermediate stops a new high-level representation (in terms of regions) is available and can be used in combination with the original map for localization. The result of this step is the index of the region the robot believes it is currently in, which is a rough estimate of its global position. As it is argued in [2], this estimate is sufficient for navigation, and an accurate map will not be necessary as long the robot can find the exits to adjacent regions.

Two different strategies for localizing the robot based on the original map generated on its way to the current position are presented in Sect. 3.2. Each method computes a local confidence map that contains a confidence value for each region of the original map. All local confidence maps are then fused into a single global one using the method described in Sect. 3.1.

3.1 Fusion of Strategies

The fusion of all local confidence maps, which may have been generated by different robot localization methods with varying reliability, is based on the idea of *Democratic Integration* introduced in [10, 11]. It was developed for the purpose of sensor data fusion in computer vision and computes confidence maps directly on images. The original method has been extended and embedded into a probabilistic framework in [12, 13], still within the area of machine vision. We extend the original approach in a way that we do not use images as an input, but rather generate local confidence maps using various — more or less reliable — techniques for robot localization. A main advantage of this approach is that the extension to more than two strategies is straightforward, as is the replacement of a method by a more sophisticated one. Each local confidence map contains a confidence value between 0 and 1 for each region of the original map. As in [10, 11] these confidence values are not probabilities, and they do not sum up to one; the interval has been chosen for convenience, and different intervals can be used as desired.

The actual fusion is straightforward, as it is done by computing a weighted sum of all local confidence maps. The main advantage of using democratic integration becomes visible only after that stage, when the weights get adjusted dynamically over time, dependent on the reliabilities of the local map. Given M local confidence maps $\boldsymbol{c}_{\text{loc}i}(t) \in \mathbb{R}^N$ (N being the total number of regions in the original map) at time t generated using different strategies, the global map $\boldsymbol{c}_{\text{glob}}(t)$ is computed as:

$$\boldsymbol{c}_{\text{glob}}(t) = \sum_{i=0}^{M-1} w_i(t)\boldsymbol{c}_{\text{loc}i}(t) \quad , \tag{3}$$

where $w_i(t)$ are weighting factors that add up to one. An estimate of the current position of the robot with respect to the original map can now be computed by determining the largest confidence value in $\boldsymbol{c}_{\text{glob}}(t)$. Its position b in $\boldsymbol{c}_{\text{glob}}(t)$ is the index of the region that the robot believes it is in. The confidence value c_{glob_b} at that index gives an impression about how reliable the position estimate is in absolute terms, while comparing it to the second best one (and maybe even third best one) shows the reliability relative to other regions.

In order to update the weighting factors, the local confidence maps have to be normalized first. The normalized map $\boldsymbol{c}'_{\text{loc}i}(t)$ is given by $\boldsymbol{c}'_{\text{loc}i}(t) = \frac{1}{N}\boldsymbol{c}_{\text{loc}i}(t)$. The idea when updating the weights is that local confidence maps that provide very reliable data get higher weights than those which are unreliable. Different ways for determining the quality of each local confidence map are presented in [11]. We use the normalized local confidence values at index b, which has been determined from the global confidence map as shown above, i. e., the quality $q_i(t)$ of each local map $\boldsymbol{c}_{\text{loc}i}(t)$ is given by $c'_{\text{loc}b}(t)$. Normalized qualities $q'_i(t)$ are computed by:

$$q'_i(t) = \frac{q_i(t)}{\sum_{j=0}^{M-1} q_j(t)} \quad . \tag{4}$$

The new weighting factors $w_i(t+1)$ can now be computed from the old ones:

$$w_i(t+1) = w_i(t) + \frac{1}{t+1}\left(q'_i(t) - w_i(t)\right) \quad . \tag{5}$$

This is a recursive formulation of the average over all qualities from time zero to t. Using this update equation and the normalization of the qualities in (4) ensures that the sum of the weights equals one at all times [11].

3.2 Localization Strategies

Two strategies for computing local confidence maps will be described in the following, one based on distance traveled, the other based on orientation information obtained from the maps. Note that these strategies are mainly used to illustrate how local confidence maps can be computed from information that is readily available. Depending on the sensors used, more sophisticated ones can be added to enhance the localization accuracy. A main feature of using a fusion approach is that each strategy taken on its own may be quite simple and not very useful for localization; it is the *combination* of different strategies which makes localization possible.

3.2.1 Distance

The first strategy is based on using the distance the robot traveled from its return point to the current position. Note that neither do we care about an exact measurement, nor do we use the actual distance traveled as provided by odometry. Using the odometry data directly would result in very different distances for each journey, as the robot normally moves in a zigzag fashion. Instead we use distance information computed from the region splitting of the maps, i. e., region length, which is defined by the distance between the "entrance" and the "exit" (split points) the robot used when passing through a particular region. The basic strategy is to compare the distance d traveled when returning home, measured in region lengths taken from the intermediate map computed on the return journey, to the lengths taken from the original map computed during the mapping process.

The confidence for each region in the local confidence map $\boldsymbol{c}_{\text{Dist}}$ depends on the overall distance d traveled on the return journey; the closer a region is to this distance from the origin, the more likely it is the one the robot is in currently. As the distance traveled is an unreliable estimate, adjacent regions should be considered as well, the more the closer they are to the most likely one. We decided to use a Gaussian to model the confidences for each region, the horizontal axis being the distance traveled in mm. The Gaussian is centered at the current overall distance traveled d. Its standard deviation σ is dependent on the distance traveled, and was chosen as $\sigma = 0.05d$. Note that although a Gaussian is used here, we do not try to model a probability density. A Gaussian was rather chosen for a number of reasons making it most suitable for our purpose: It allows for a smooth transition between regions, and the width can be easily adjusted by altering the standard deviation. This is necessary as the overall distance traveled gets more and more unreliable (due to slippage and drift) the farther the robot travels. The confidence value for a region is determined by sampling the Gaussian at the position given by the accumulated distances from the origin (i. e., where the robot started the homeward journey) to the end of this region. After a value for each region is computed, the local confidence map $\boldsymbol{c}_{\text{Dist}}$ is normalized to the interval $[0;1]$.

3.2.2 Relative Orientation

The second method for computing estimates of the robot's position is based on using relative orientation information generated while dividing the map into regions. During its journey, the robot enters a region at one location and exits at a different one, usually including zig-zag movements in between.

We define the direction of a region as the direction of the line connecting the entrance and exit points. Certainly this direction information varies every time the robot travels through the environment, but the overall shape between adjacent regions is relatively stable. Therefore, we propose to use angles between region directions as a simple measure of the current position of the robot. It has the advantage that angles between adjacent region directions are a local measure of direction changes, thus keeping the influence of odometry errors due to drift and slippage to a minimum.

Firstly, all angles $\alpha_1, \ldots, \alpha_{N-1}$ between adjacent regions in the original map are computed. In the remapping process while going home, new regions are computed in the new map based on data gathered while the robot travels. Using the direction information contained in this map, the angle β between the current region and the previous one can be computed. "Comparing" this angle to all angles of the original map gives a clue (or many) for the current location of the robot, resulting in a local confidence map c_{Dir}:

$$c_{\mathrm{Dir}\,i} = \frac{1}{2}(\cos|\alpha_i - \beta| + 1), \quad i = 1, \ldots, N-1 \quad . \tag{6}$$

This results in high values for similar angles and low values for dissimilar ones.

4 Experimental Results

The main features of the office environment where we conducted the experiments are corridors, which open into bigger areas at certain locations, doors that are located on the left and right of the corridors, and obstacles like waste paper baskets that can be found on the floor in various positions. The acquisition of the original maps and the experiments for using the map in order to find the way back to the starting position was done on different days, so the environment was different for each experiment (e. g., doors open/closed). Furthermore, we used two different robots, one for generating the original map, the other one for using the map for localization, to demonstrate the robustness of the methods. Both robots are from Activmedia (a Pioneer 2 and a Pioneer 3), equipped with eight sonar sensors and an odometer. Only the two side sensors were used in order to obtain sparse range data. Results for two experiments are shown in the following.

Figure 1 shows four maps, including the locations of the splitting points marked by dots. These are located on a set of connected lines that resembles the path the robot took while mapping the environment. To the left and right of that path, the (simplified) surfaces representing the environment can be seen. For splitting purposes, gaps were treated as distant surfaces, having a distance of 6000 mm from the position of the robot. Units are given in millimeters, and the robot started the mapping process at the origin. All maps were processed using the same parameter values, namely $\theta_\mathrm{r} = 2.0$ and $s = 0.1$; the desired minimum size of a region was 1500 mm. We found that the overall robustness to changes in the parameters is quite high, i. e., the choice of the actual values is usually noncritical; for an evaluation see [1]. It can be observed that the splits are located at the desired positions, i. e., where the environment changes, either from corridor to big room or at sharp bends in the corridor. Note that gaps imply a rapid change as well, because they are treated like distant surfaces, which sometimes leads to splits at positions that may be undesired, but do not pose a problem when using the map for localization. The leftmost and center-left maps shown in Fig. 1 were generated from the mapping and going home processes respectively for Experiment 1; the center-right and rightmost figures are the maps generated from the mapping and going home processes respectively for Experiment 2. Comparing the maps generated during mapping and going home highlights the difficulty in using these maps directly for localization. Each time, the robot goes through the same environment, it will generate different representations due to sensory inaccuracies.

Figure 2 shows two confidence maps for each experiment computed at different locations during the return home journey. The light dotted lines represent the region estimate using the region length information (distance method) and the dark dashed lines depicts the region estimate using the angles between regions (relative orientation method). The solid line is the overall region estimate for the corresponding region. The confidence maps in Fig. 2 illustrate different situations during localization. A narrow peak for the overall confidence signifies the robot being very confident of being in a particular region. A wider confidence curve shows that the robot is at the transition from one region to another, as more than one region has a high confidence value, and the robot is unsure which of the regions it is in. Comparisons of the estimated position to the actual position have shown that the localization is usually correct, with possible deviations of ± 1 in areas where the regions are extremely small.

5 Conclusion

We have presented methods for mapping and localization using sparse range data without the need for solving the SLAM problem. An initial metric map obtained from sonar sensor readings is divided into distinct regions, thus creating a topological map on top of the metric one. A split and merge approach has been used for this purpose, based on an objective function that computes the quality of a region, including a regularization term that avoids the formation of very small regions. Based on spatial information derived from these maps, we showed how simple localization strategies can be used to compute local confidence maps. These are fused into a single global one, which provides an estimate in the form of confidence values, one for each region, that reflects the confidence of the robot being in a particular region. The main advantages of the fusion approach are that it can easily be extended by more sophisticated methods or additional sensors, and that

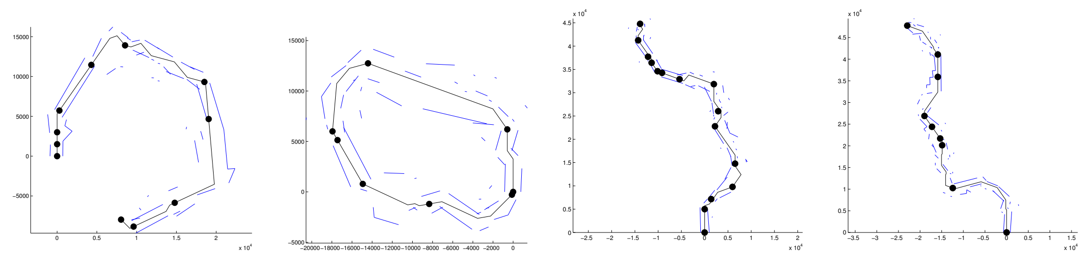

Figure 1: Left: original map for Experiment 1, center-left: map generated during the homeward journey in Experiment 1, center-right: original map for Experiment 2, right: homeward journey in Experiment 2. The black dots indicate the points where the map is split into separate regions. The robot movement always starts at the origin. The origin of the homeward journey is (approximately) the same position as the end point coordinate of the respective original map; in particular this means, that the map of the homeward journey for Experiment 2 (right) is upside-down compared to the original map (center-right).

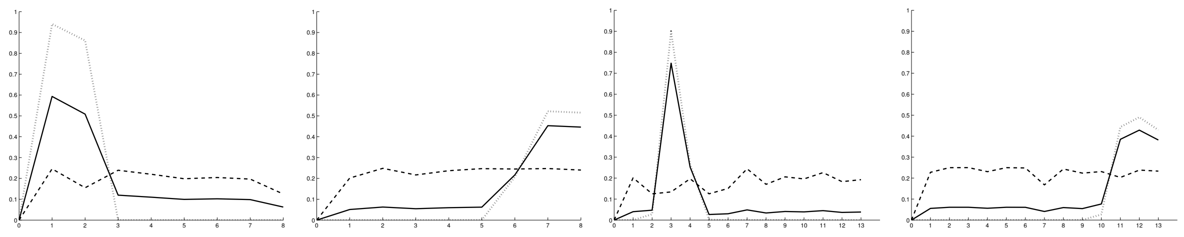

Figure 2: Confidence maps computed at different locations during the return home journey. Left/center-left: Experiment 1. Center-right/right: Experiment 2. The plots show: distance (light dotted), relative orientation (dark dashed), overall confidence (solid). Horizontal axis: region index; vertical axis: confidence (0 to 1).

the influence of unreliable strategies on the global confidence map automatically decreases over time. Experimental results show that the robot was highly successful in using our proposed method in localizing itself in the environment. Even though this approach does not provide the robot's exact pose, we believe the current output is sufficient for navigation purposes.

6 References

[1] J. Schmidt, C. K. Wong, and W. K. Yeap. A Split & Merge Approach to Metric-Topological Map-Building. In *Int. Conf. on Pattern Recognition (ICPR)*, volume 3, pages 1069–1072, Hong Kong, (2006).

[2] W. K. Yeap and M. E. Jefferies. Computing a Representation of the Local Environment. *Artificial Intelligence*, 107(2):265–301, (1999).

[3] B. Kuipers, J. Modayil, P. Beeson, M. MacMahon, and F. Savelli. Local Metrical and Global Topological Maps in the Hybrid Spatial Semantic Hierarchy. In *Int. Conf. on Robotics and Automation*, pages 4845–4851, New Orleans, LA, (2004).

[4] S. Thrun. Learning Metric-Topological Maps for Indoor Mobile Robot Navigation. *Artificial Intelligence*, 99(1):21–71, (1998).

[5] H. Niemann. *Pattern Analysis and Understanding*, volume 4 of *Springer Series in Information Sciences*. Springer, Berlin, 2nd edition, (1990).

[6] T. Pavlidis and S. L. Horowitz. Segmentation of Plane Curves. *IEEE Trans. on Computers*, C-23:860 – 870, (1974).

[7] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, New York, USA, (1973).

[8] C. Estrada, J. Neira, and J. D. Tardos. Hierarchical SLAM: Real-Time Accurate Mapping of Large Environments. *IEEE Trans. on Robotics*, 21(4):588–596, (2005).

[9] R. M. Downs and D. Stea. *Image and Environment: Cognitive Mapping and Spatial Behaviour*. Aldine Publishing Coy, Chicago, (1973).

[10] J. Triesch. *Vision-Based Robotic Gesture Recognition*. Shaker Verlag, Aachen, (1999).

[11] J. Triesch and Ch. von der Malsburg. Democratic Integration: Self-Organized Integration of Adaptive Cues. *Neural Computation*, 13(9):2049–2074, (2001).

[12] J. Denzler, M. Zobel, and J. Triesch. Probabilistic Integration of Cues From Multiple Cameras. In R. Würtz, editor, *Dynamic Perception*, pages 309–314. Aka, Berlin, (2002).

[13] O. Kähler, J. Denzler, and J. Triesch. Hierarchical Sensor Data Fusion by Probabilistic Cue Integration for Robust 3-D Object Tracking. In *Proc. of the 6th IEEE Southwest Symp. on Image Analysis and Interpretation*, pages 216–220, (2004).