Localisation and Mapping with a Mobile Robot Using Sparse Range Data

Jochen Schmidt, Chee K. Wong, and Wai K. Yeap

Centre for Artificial Intelligence Research
Auckland University of Technology, Auckland, New Zealand
jochen.schmidt@aut.ac.nz, chee.wong@aut.ac.nz, wai.yeap@aut.ac.nz

Summary. We present an approach for indoor mapping and localisation using sparse range data, acquired by a mobile robot equipped with sonar sensors. 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-localisation on the return home journey. The robot computes local confidence maps for two simple localisation strategies based on distance and relative orientation of regions. These local maps are then fused to produce overall confidence maps.

1 Introduction

Mapping and self-localisation 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 localisation using sparse range data acquired by only two sonars, and show that the robot can localise 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. As we use only two sonar sensors, the available range data are very sparse, therefore making the map highly inaccurate. We will show that these data can nevertheless be used for self-localisation. Our work is inspired by [11], 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 [4], which is a hybrid approach that combines topological and metric maps. In [8], topological maps are constructed from grid maps using Voronoi diagrams; the grid maps are split into regions and gateways are detected. In contrast to these methods, our approach is based on a region split and merge

algorithm [5]. Many algorithms have been developed to solve the simultaneous mapping and localisation problem (SLAM) for mobile robots. For some examples of recent work in this area see [2] and the references therein. The approach followed in this paper 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. We implemented two localisation strategies using both distance and orientation information extracted from the metric-topological map. We show how local confidence maps (location estimate of the robot) can be computed, and how to fuse them. For more details and experimental results refer to our earlier work [7, 10].

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 (origin of original map). 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 [6].

For data acquisition we use a mobile robot equipped with an odometer and two sonar sensors located on the left and right side. Note that the algorithms presented here are not restricted to sparse data or that type of sensors; the performance will be even better when more range data are available. The robot acquires sonar readings while moving on a "straight" line (we are not concerned about drift compensation or correction) 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. Based on the raw sonar sensor readings we build a geometric map containing the robot movement path as well as linear surfaces approximated from the sonar 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). The basis of the map-processing algorithm is the well-known split and merge method [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 colour or texture.

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. After this step, the actual division of the map into distinct regions is performed based on a split and merge that uses a residual error function $g(\mathcal{S}_i, \mathcal{S}_j)$ which compares two regions $\mathcal{S}_i$ and $\mathcal{S}_j$ and computes the homogeneity of the two regions (low values of $g(\mathcal{S}_i, \mathcal{S}_j)$ means homogeneous, high values very inhomogeneous). This function is used during the split phase for deciding whether a region $\mathcal{S}_i^k$ will be split again at a given position into two new regions $\mathcal{S}_j^{k+1}$ and $\mathcal{S}_{j+1}^{k+1}$. If the homogeneity is above a given threshold θ_r, the region will be split again. When no further splitting is possible, the algorithm tries to merge adjacent regions (which were not necessarily generated by a single split) by checking whether the created region is still homogeneous. 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:

$$g(\mathcal{S}_i, \mathcal{S}_j) = \frac{\max\{f_\mathrm{w}(\mathcal{S}_i), f_\mathrm{w}(\mathcal{S}_j)\}}{\min\{f_\mathrm{w}(\mathcal{S}_i), f_\mathrm{w}(\mathcal{S}_j)\}} + s_\mathrm{r} r(\mathcal{S}_i, \mathcal{S}_j) \tag{1}$$

where $f_\mathrm{w}(\mathcal{S}_i)$ is the average width of region $\mathcal{S}_i$, and $r(\mathcal{S}_i, \mathcal{S}_j)$ is a regularisation term that takes care of additional constraints during splitting. The average width is given by $f_\mathrm{w}(\mathcal{S}_i) = \frac{A_{\mathcal{S}_i}}{l_{\mathcal{S}_i}}$, where $A_{\mathcal{S}_i}$ is the area of region $\mathcal{S}_i$, and $l_{\mathcal{S}_i}$ 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 $l_{\mathcal{S}_i}$ 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 regularisation term $r(\mathcal{S}_i, \mathcal{S}_j)$ ensures that the regions do not get too small. In contrast to a threshold, which is a clear decision, a regularisation term penalises 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, centred at n, which is the desired minimum size of a region:

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

The exponent is basically the area of the smaller region in relation to the maximum area $A_{\max}$ of the smallest allowed region. 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.

The influence of the term can be controlled using the factor s_{r} in (1), which is given by $s_{\mathrm{r}} = s\theta_{\mathrm{r}}$, where $0 \leq s \leq 1$ is set manually and defines the percentage of the threshold θ_{r} mentioned earlier that is to be used as a weight.

3 Localisation

Once the original map has been generated, we instruct the robot to return home based on the acquired map. Each time the robot stops on its return journey (because of an obstacle), it performs a map processing as described in Sect. 2, making a new metric-topological map available at each intermediate stop that can be used for localisation.

In the following, we describe two strategies for localisation, and a data fusion algorithm that allows for an overall position estimate computed from the individual localisation methods. 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.

The fusion of local confidence maps, which may have been generated by different robot localisation methods with varying reliability, is based on the idea of *Democratic Integration* introduced in [9]. 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 [1, 3], 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 techniques for robot localisation. A main advantage of this approach is that the extension to more than two strategies is straightforward. Each local confidence map contains a confidence value between 0 and 1 for each region of the original map. As in [9] 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 $\boldsymbol{c}_{\mathrm{loc}i}(t) \in \mathbb{R}^N$ (N being the total number of regions in the original map) at time t generated using different strategies, the global map $\boldsymbol{c}_{\mathrm{glob}}(t)$ is computed as $\boldsymbol{c}_{\mathrm{glob}}(t) = \sum_{i=0}^{M-1} w_i(t)\boldsymbol{c}_{\mathrm{loc}i}(t)$, 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 $\boldsymbol{c}_{\mathrm{glob}}(t)$. Its position b in $\boldsymbol{c}_{\mathrm{glob}}(t)$ 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 other ones shows the reliability relative to other regions.

In order to update the weighting factors, the local confidence maps have to be normalised first; they are given by $\boldsymbol{c}'_{\mathrm{loc}\,i}(t) = \frac{1}{N}\boldsymbol{c}_{\mathrm{loc}\,i}(t)$. 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 [9]. We use the normalised 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 $\boldsymbol{c}_{\mathrm{loc}\,i}(t)$ is given by $c'_{\mathrm{loc}\,b}(t)$. Normalised qualities $q'_i(t)$ are computed by $q'_i(t) = \frac{q_i(t)}{\sum_{j=0}^{M-1} q_j(t)}$. The new weighting factors $w_i(t+1)$ can now be computed from the old ones: $w_i(t+1) = w_i(t) + \frac{1}{t+1}\left(q'_i(t) - w_i(t)\right)$. Using this update equation and the normalisation of the qualities ensures that the sum of the weights equals one at all times [9].

Two strategies for computing local confidence maps are described below, one based on distance travelled, the other based on orientation information. Depending on the sensors used, more sophisticated ones can be added to enhance localisation accuracy. A main feature of data fusion is that each strategy taken on its own may be quite simple and not very useful for localisation; it is the *combination* of different strategies which makes localisation possible.

The first strategy is based on using the distance the robot travelled 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 travelled 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 zig-zag 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 idea is to compare the distance d, 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 $\boldsymbol{c}_{\mathrm{Dist}}$ depends on the overall distance d travelled 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. We decided to use a Gaussian to model the confidences for each region, the horizontal axis being the distance travelled in mm. The Gaussian is centred at the current overall distance travelled d. Its standard deviation σ is dependent on the distance travelled, and was chosen as $\sigma = 0.05d$. A Gaussian was chosen not to model a probability density, but 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 travelled 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 $\boldsymbol{c}_{\mathrm{Dist}}$ is normalised to the interval $[0; 1]$.

The second strategy is based on using relative orientation information. We define the direction of a region as the direction of the line connecting the "entrance" and "exit". Certainly this direction information varies every time the robot travels through the environment, but the overall shape between adjacent regions is relatively stable. Therefore, angles between region directions can be used as a measure of the current position of the robot. It has the advantage that angles between adjacent region directions are a local measure, thus keeping the influence of odometry errors to a minimum.

Firstly, all angles $\alpha_1, \ldots, \alpha_{N-1}$ between adjacent regions in the original map are computed. In the re-mapping 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 $\boldsymbol{c}_{\mathrm{Dir}i} = \frac{1}{2}(\cos|\alpha_i - \beta| + 1)$. 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, and obstacles like waste paper baskets in various positions. The acquisition of the original maps and the experiments for using the maps for localisation was done on different days, so the environment was different for each experiment (e. g., doors open/closed). We used an Activmedia Pioneer robot, equipped with an odometer and eight sonar sensors; only the two side sensors were used in order to obtain sparse range data.

Figure 1 shows four maps, including the locations of split points marked by dots. These are located on a set of connected lines that resemble 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 6 m from the position of the robot. The robot started the mapping process at the origin. All maps were processed using the same parameter values, $\theta_{\mathrm{r}} = 2.0$ and $s = 0.1$; the desired minimum size of a region was 1.5 m. 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 [6]. It

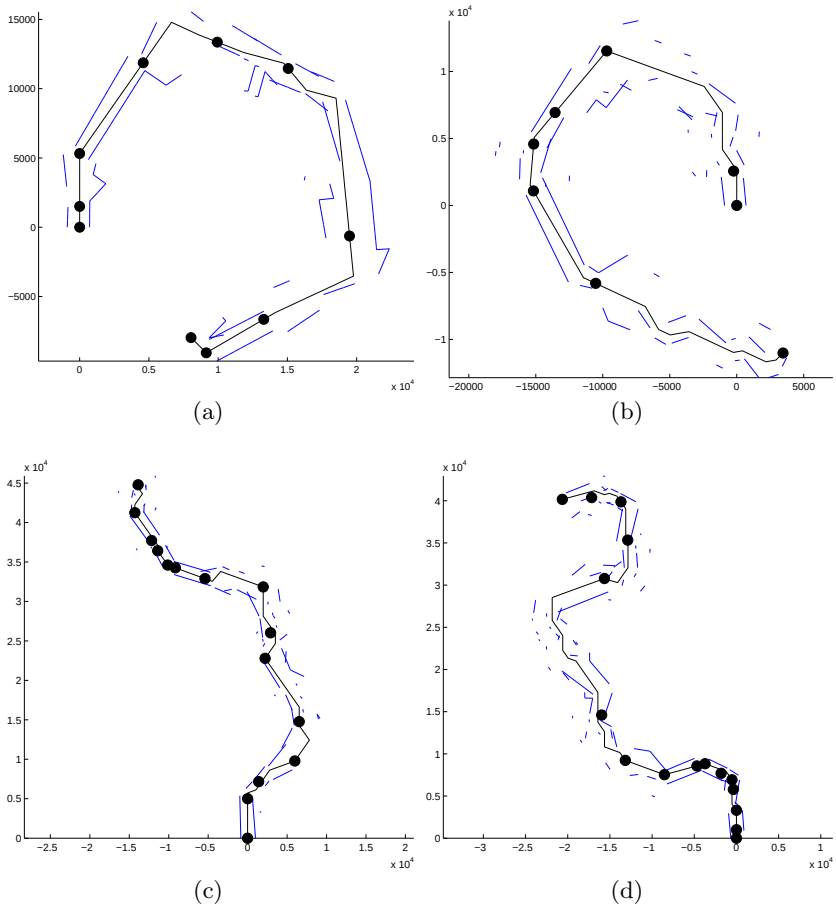

Fig. 1. (a) original map Experiment 1, (b) map generated during homeward journey Experiment 1, (c) original map Experiment 2, (d) homeward journey Experiment 2. Black dots indicate split points, robot movement starts at the origin.

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. The maps shown in Figs. 1(a) and 1(b) were generated from the mapping and going home processes respectively for Experiment 1; the Figs. 1(c) and 1(d) 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 localisation. 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

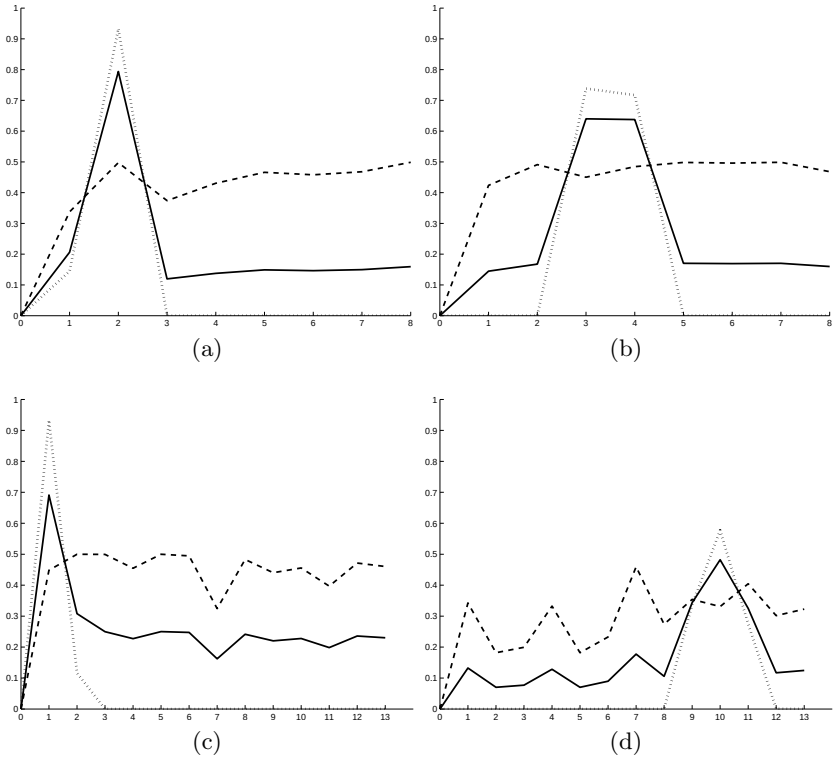

Fig. 2. Confidence maps at different locations. (a),(b) Experiment 1. (c),(d) Experiment 2. The plots show: distance (light dotted), relative orientation (dark dashed), overall confidence (solid). Horizontal axis: region index; vertical axis: confidence.

represent the region estimate using the region length information (distance method) and the dark dashed lines depict the region estimate using the angles between regions (relative orientation method). The solid line is the overall region estimate. The confidence maps in Fig. 2 illustrate different situations during localisation. 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 localisation 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 localisation using sparse range data. An initial metric map obtained from sonar sensor readings is divided into distinct regions, thus creating a metric-topological map. A split and merge approach has been used for this purpose, based on an objective function that computes the quality of a region. Based on spatial information derived from these maps, we showed how simple localisation strategies can be used to compute local confidence maps that are fused into a single global one, which reflects the confidence of the robot being in a particular region. The fusion can easily be extended by more localisation strategies or additional sensors.

References

1. 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.
2. 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.
3. O. Kähler, J. Denzler, and J. Triesch. Hierarchical Sensor Data Fusion by Probabilistic Cue Integration for Robust 3-D Object Tracking. In *IEEE Southwest Symp. on Image Analysis and Interpretation*, pages 216–220, Nevada, 2004.
4. 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.
5. T. Pavlidis and S. L. Horowitz. Segmentation of Plane Curves. *IEEE Trans. on Computers*, C-23:860 – 870, 1974.
6. 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.
7. J. Schmidt, C. K. Wong, and W. K. Yeap. Mapping and Localisation with Sparse Range Data. In S.C. Mukhopadhyay and G. Sen Gupta, editors, *Proceedings of the Third International Conference on Autonomous Robots and Agents (ICARA 2006)*, pages 497–502, Palmerston North, New Zealand, 2006.
8. S. Thrun. Learning Metric-Topological Maps for Indoor Mobile Robot Navigation. *Artificial Intelligence*, 99(1):21–71, 1998.
9. J. Triesch and Ch. von der Malsburg. Democratic Integration: Self-Organized Integration of Adaptive Cues. *Neural Computation*, 13(9):2049–2074, 2001.
10. C. K. Wong, J. Schmidt, and W. K. Yeap. Using a Mobile Robot for Cognitive Mapping. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2243–2248, Hyderabad, India, 2007.
11. W. K. Yeap and M. E. Jefferies. Computing a Representation of the Local Environment. *Artificial Intelligence*, 107(2):265–301, 1999.