A Split & Merge Approach to Metric-Topological Map-Building

Jochen Schmidt, Chee K. Wong, and Wai K. Yeap
Robotics Research Group, 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 a novel split and merge based method for dividing a given metric map into distinct regions, thus effectively creating a topological map on top of a metric one. The initial metric map is obtained from range data that are converted to a geometric map consisting of linear approximations of the indoor environment. The splitting is done using an objective function that computes the quality of a region, based on criteria such as the average region width (to distinguish big rooms from corridors) and overall direction (which accounts for sharp bends). A regularization term is used in order to avoid the formation of very small regions, which may originate from missing or unreliable sensor data. Experiments based on data acquired by a mobile robot equipped with sonar sensors are presented, which demonstrate the capabilities of the proposed method.

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 discuss a novel method for dividing a given metric map into distinct regions, e. g., corridors or rooms. The information extracted this way is a topological map, each region representing a node of the map. All nodes are implicitly linked by paths, which are defined by the transitions from one region to another. Also, after splitting the map into regions, each region can be processed separately, which facilitates, e. g., further feature extraction. The main focus of this paper is on the region splitting given a metric map, which we obtain from sonar sensor readings that are processed into a map consisting of geometric boundaries, particularly piecewise linear approximations. This is a more compact representation of the environment compared to grid maps, while details such as size and shape of objects are maintained.

Our own work is inspired by [9], 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 and identifies gateways and path fragments. In [8], 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 [6, 7, 3]. Split and merge has been very popular in robotics for generating geometric maps from range data by extracting lines [2, 1, 5], but to our knowledge has never been applied before in order to generate topological information from a metric map as presented in the following.

2. Split and Merge

The basis of the proposed algorithm is the well-known split and merge method [3, 7, 6]. 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 [6]. As this algorithm is very important for the remaining part of the paper, the classic version used for contour approximation will be described shortly in the following. The prerequisite 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 $\mathcal{F}$ (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_1. The single steps of the algorithm are as follows [6]:

1. Start with an initial set of points $\mathcal{P}^0$, which consists of n_0 parts, $\mathcal{P}^0 = \{\mathcal{P}^0_0, \ldots, \mathcal{P}^0_{n_0-1}\}$. Each part $\mathcal{P}^0_i$ is approximated by a function from $\mathcal{F}$. Compute the initial residual error ϵ^0_i for each part of $\mathcal{P}^0$.
2. **Split** each part $\mathcal{P}^k_i$ where $\epsilon^k_i > \theta_\text{l}$ into two parts $\mathcal{P}^{k+1}_j$ and $\mathcal{P}^{k+1}_{j+1}$, compute the approximation and residuals ϵ^{k+1}_j, ϵ^{k+1}_{j+1}. Repeat until $\epsilon^k_i \leq \theta_\text{l}\ \forall i = 0, \ldots, n_k - 1$.
3. **Merge** two adjacent parts $\mathcal{P}^k_i$, $\mathcal{P}^k_{i+1}$ into one new part $\mathcal{P}^{k+1}_j$ if $\epsilon^{k+1}_j \leq \theta_\text{l}$. Repeat until merging not possible.
4. **Shift** the split point shared by two adjacent parts $\mathcal{P}^k_i$, $\mathcal{P}^k_{i+1}$ 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.

3. Map Processing

Currently we use a mobile robot equipped with sonar sensors and an odometer. We would like to emphasize, however, that the region splitting as proposed in this paper is in no way restricted to that type of sensors, but is independent of the actual type used. In fact, the performance will be even better when more range data (e. g., from laser) is available.

The mapping process used in our system, which is basically a wandering robot that records sonar data, is as follows: 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. Note that we store the sonar readings separately for each robot path; even though this is not mandatory for the proposed region splitting algorithm, it simplifies the processing later on. After the whole map consisting of raw sonar sensor readings is acquired, we build a simplified geometric map containing the robot movement path as well as linear surfaces approximated from the sonar data. If desired, this processing and splitting of the map into regions can be done at any intermediate stage as well. 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. These 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. Gaps may be visible, too. The objective 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.

3.1. Initialization

The initialization step described here generates an initial split of the map into regions using only the robot movement, on the basis that at locations where the direction of the robot movement changes substantially, the environment changes as well, as the robot would have gone straight on otherwise. This is basically a simplification of the robot paths that evens out zig-zag movements of the robot due to obstacle avoidance. Whether or not this kind of initialization can be done depends on the algorithm used for generating the map, and may also be omitted.

To compute this initial region splitting, we run a standard split and merge as described in Sect. 2 on the robot path, where the splitting points can be located only at the beginning and end of each path. As a single robot path can be a few meters long, depending on the environment and whether there were obstacles, the resolution of the initial splitting would be very coarse in some areas (where there are long movements) and fine in others (short paths). Therefore, large robot paths are divided into smaller ones, the size depending on the required resolution of the final region split. A length of about 50 cm proved to be a good choice in practice. For measuring the residual error during split and merge we use the maximum norm, i. e., the residual ϵ^k_i for a line segment $\mathcal{P}^k_i$ that approximates several robot movement paths is defined by the maximum of all distances of the split points within this segment from the line approximation.

3.2. Region Splitting of the Map

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 $q(\mathcal{P}_i, \mathcal{P}_j)$ which compares two regions $\mathcal{P}_i$ and $\mathcal{P}_j$ and computes the homogeneity of the two regions (low values of $q(\mathcal{P}_i, \mathcal{P}_j)$ means homogeneous, high values very inhomogeneous). This function is used during the split phase for deciding whether a region $\mathcal{P}^k_i$ will be split again at a given position into two new regions $\mathcal{P}^{k+1}_j$ and $\mathcal{P}^{k+1}_{j+1}$, 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).

In contrast to the initialization, which depends solely on the robot path, the quality of a region now incorporates both, robot path as well as the sonar data (in terms of surfaces). 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:

$$q(\mathcal{P}_i, \mathcal{P}_j) = \frac{\max\{f_\text{w}(\mathcal{P}_i), f_\text{w}(\mathcal{P}_j)\}}{\min\{f_\text{w}(\mathcal{P}_i), f_\text{w}(\mathcal{P}_j)\}} + s_\text{r} r(\mathcal{P}_i, \mathcal{P}_j) \ , \quad (1)$$

where $f_\text{w}(\mathcal{P}_i)$ is the average width of region $\mathcal{P}_i$, and $r(\mathcal{P}_i, \mathcal{P}_j)$ is a regularization term that takes care of additional constraints during splitting. The factor s_r controls the influence of $r(\mathcal{P}_i, \mathcal{P}_j)$. This factor as well as both functions will be discussed in more detail in the following. Obviously, the average width is given by $f_\text{w}(\mathcal{P}_i) = \frac{A_{\mathcal{P}_i}}{l_{\mathcal{P}_i}}$, where $A_{\mathcal{P}_i}$ is the area of region $\mathcal{P}_i$, and $l_{\mathcal{P}_i}$ is its length. In practice, the computation of both needs a bit of attention, though. 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 $l_{\mathcal{P}_i}$ 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. While 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 currently 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 main purpose of the regularization term $r(\mathcal{P}_i, \mathcal{P}_j)$ is to ensure that the regions do not get too small. Depending on the treatment of gaps as mentioned above, the algorithm could create a large number of very small regions. 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 in terms of single robot paths, the size of a single path being dependent on the resolution chosen during initialization (cf. Sect. 3.1):

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

The exponent is basically the area of the smaller region in relation to the maximum area A_max of the smallest region possible, i. e., A_max can be determined from the maximum distance of a surface allowed (usually defined by a gap) and the resolution used for the division of the robot paths during pre-processing. Thus, the ratio is 1 for the smallest region possible, and increases when the region gets larger. Note that 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 factor s_r in (1) is used to increase or decrease the influence of the regularization term, and is determined by $s_\text{r} = s\theta_\text{r}$, 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 (cf. (1)) that a region should be split into two when the first region is θ_r times as large as the second one.

4. Experimental Results

For experimental evaluation of the algorithm presented in this paper, we used a Pioneer 2 robot from Activmedia equipped with eight sonar sensors and an odometer in order to map an indoor environment. The main features of this environment 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. Neither the locations of these obstacles nor the state of the doors (open or closed) was controlled. Maps were acquired on different days, so the environment was different every time a mapping was done.

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 $\theta_\text{r} = 2.0$ and $s = 0.1$; the desired minimum size of a region was 1500 mm, which correspondents to $n = 3$ at a path-resolution of 500 mm. 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. Sometimes this leads to splits at positions that are undesired. This problem is mainly due to the use of sonar sensors, and can be circumvented by more reliable sensors such as lasers, where dense surfaces can be generated.

The influence of the two parameters θ_r and s was investigated as well. Splitting results on the map shown in Fig. 1 (left) are depicted in Fig. 2 for different values of θ_r (top) and s (bottom), respectively. The maps in the top row were computed using values for θ_r of 1.3 (left) and 2.7 (right), while the other parameter was fixed to $s = 0.1$. In the bottom row, the parameter s was varied, using values of 0.05 and 0.9, while θ_r was fixed to 2.0. As θ_r controls the ratio of the average width of adjacent regions, increasing this parameter results in less splitting points and therefore larger

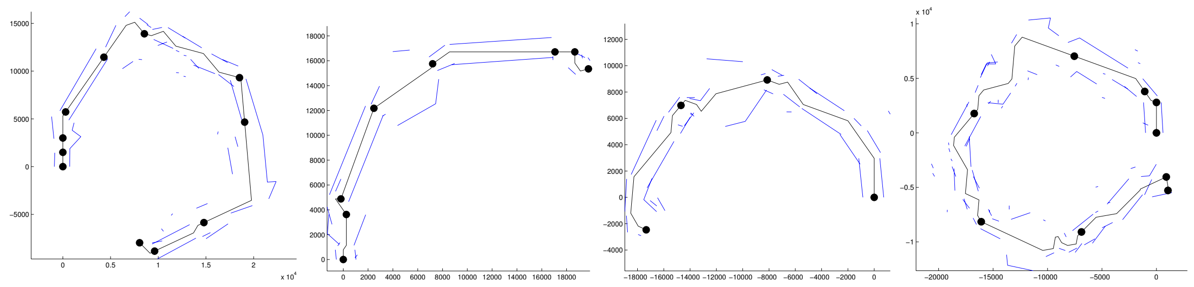

Figure 1. Different Maps, split points indicated by dots located on the robot path, which is represented by connected lines. The separate regions were generated using $\theta_r = 2.0$ and $s = 0.1$. The surfaces computed from sonar measurements are located to the left and right of the path.

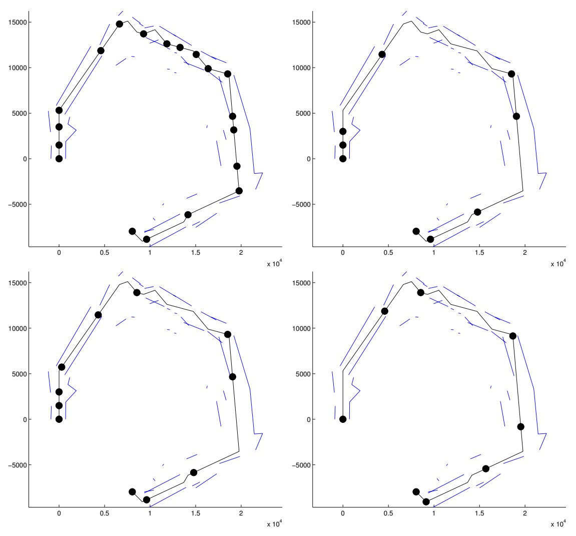

Figure 2. Influence of the parameters. Top row: $\theta_r = \{1.3, 2.7\}$, $s = 0.1$. Bottom row: $s = \{0.05, 0.9\}$, $\theta_r = 2.0$.

regions. The parameter s shows a similar behavior, i. e., the higher its value the higher the influence of the regularization term, which results in larger regions on average.

The experimental results can be summarized as follows: The same parameter values can be used for processing different maps, i. e., once the parameters are adjusted properly, there is no need to change them for different mapping runs. This is very important for practical purposes, because the algorithm can be used without human intervention. 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.

5. Conclusions

We have presented a novel split and merge based method for dividing a given metric map into distinct regions, thus effectively creating a topological map on top of a metric one. The initial metric maps are obtained from sonar sensor readings. 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 region width and overall direction. A regularization term avoids the formation of very small regions, which may originate from missing or unreliable sensor data.

A series of experiments was conducted, which demonstrate the capabilities of the proposed method. For practical purposes it is important to note that the resulting regions are quite robust with respect to changes in the algorithm's parameters as well, which makes choosing their actual values noncritical.

References

[1] H. Baltzakis and P. Trahanias. An Iterative Approach for Building Feature Maps in Cyclic Environments. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robotics and Systems (IROS)*, pages 576–581, Lausanne, Switzerland, 2002.

[2] G. Borges and M.-J. Aldon. A Split-and-Merge Segmentation Algorithm for Line Extraction in 2-D Range Images. In *Proc. 15th Int. Conf. on Pattern Recognition (ICPR'00)*, volume 1, pages 441–444, Barcelona, 2000.

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

[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] P. Newman, J. Leonard, J. Tardds, and J. Neira. Explore and Return: Experimental Validation of Real-Time Concurrent Mapping and Localization. In *Int. Conf. on Robotics and Automation*, pages 1802–1809, Washington, DC, 2002.

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

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

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

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