# Learning Polyline Maps from Range Scan Data Acquired with Mobile Robots

Michael Veeck Wolfram Burgard
University of Freiburg
Department of Computer Science
D-79110 Freiburg, Germany

***Abstract*— Geometric representations of the environment play an important role in mobile robotics as they support various tasks such as motion control and accurate localization. Popular approaches to represent the geometric features of an environment are occupancy grids or line models. Whereas occupancy grids require a huge amount of memory and therefore do not scale well with the size of the environment, line models are unable to correctly represent corners or connections between objects. In this paper we present an algorithm that learns sets of polylines from laser range scans. Starting with an initial set of polylines generated from the range scans it iteratively optimizes these polylines using the Bayesian Information Criterion. During the optimization process our algorithm utilizes information about the angles between line segments extracted from the original range scans. We present experiments illustrating that our algorithm is able to learn accurate and highly compact polyline maps from laser range data obtained with mobile robots.**

## I. Introduction

Geometric maps play an important role in mobile robotics since they support various tasks such as path planning and accurate localization. One of the most popular ways to represent the environment of a robot are occupancy grids. Their advantage is that they can easily be updated upon sensory input. Their disadvantages, however, lie in the huge memory requirements, the assumption that neighboring cells are independent of each other, and the discretization problems. To overcome these limitations, many authors studied the generation of more compact representations such as line models from range scan data. Line maps are based on the assumption that most indoor environments consist of planar structures corresponding to walls, doors, cupboards etc. They require much less space since planar objects often can be represented by a small number of lines whereas thousands of grid cells might be required by an occupancy grid. They also overcome the independence assumption since a change of the parameters of a line influences the whole area covered by that line. Finally, they are also more accurate since they provide floating point resolution and do not suffer from discretization problems.

Line maps, however, have the disadvantage that the lines, in contrast to the objects in the environment, are infinite. One solution to this problem is to consider line segments instead of infinite lines. However, even this approach lacks the ability to represent corners or connections between objects and to incorporate these features during the approximation process.

In this paper we consider the problem of learning polyline maps from data gathered with a mobile robot equipped with a laser range scanner. A polyline map consists of a set of polylines which are composed of a sequence of line segments that are connected at their endpoints. This representation clearly is more general than line segments, since every line segment can be represented by a polyline of length one. Additionally, polylines allow the representation of corners that connect individual line segments. The major questions in the task of finding a polyline that best fits a given contour or set of points are where the individual endpoints of the polylines should be placed and how many line segments should be used to approximate the given data.

We present an algorithm that generates polyline maps from raw laser range scans. This algorithm first extracts a set of contours out of the range measurements. It then generates an initial polyline set from these contours. After this initialization it optimizes the polylines using various operators for splitting, joining and adjusting line segment sequences in order to determine the best fit to the given data. Simultaneously it utilizes information about the relative angles between line segments extracted from the individual range scans. To evaluate the models it applies the Bayesian Information Criterion which trades off the approximation error, the number of vertices in the map, and the size of the data set.

This paper is organized as follows. After discussing related work in the next section, we describe how to extract an initial polyline map from raw laser range data acquired with a mobile robot. We then present the different operations used to optimize this map. Finally, we present experiments illustrating that our approach generates highly accurate polyline maps from data gathered in various environments.

## II. Related Work

The most popular approach to fit geometric primitives to range data acquired with mobile robots is the extraction of lines. One of the first attempts has been presented by Crowley [4] who computes line segments from range measurements and combines these segments using a Kalman filter. Pfister et al. [18] extend this approach and also consider the accuracy of the measurements when updating the line model. Arras and Siegwart [1] use a clustering

approach to learn line models from laser data. Their approach also considers the uncertainty in the measurements when clustering points into linear segments. The approach developed by Gonzales et al. [8] computes point clusters from each range scan based on the distance between consecutive points. They apply linear regression to fit lines to these clusters and iteratively combine lines to a global map. Leonard et al. [13] use a Hough transform to extract linear features from sequences consecutive sonar measurements. These features are then maintained using a Kalman filter. Several approaches apply the well-known iterative end-point fit or split and merge algorithm [5] for fitting lines to scans. The approach of Schröter et al. [21] is to cluster scans using the split and merge algorithm, which is also used by Gutmann et al. [10], and combine nearby segments using a weighted variant of linear regression. Also Newmann et al. [17] use this approach in combination with the Ransac algorithm [6] to extract linear models from laser data. Baltzakis and Trahanias [3] propose an approach for simultaneous localization and mapping in which the features are extracted using the split-and-merge algorithm. MacKenzie and Dudek [15] use a clustering strategy to associate measurements that arise from the same object and then recursively subdivide these clusters to obtain subsets with good linear approximations. Liu et al. [14] apply the EM-algorithm to extract planar structures from 3d data. An online variant of this approach has been presented by Thrun et al. [16]. Austin and McCarragher [2] describe a technique to match geometric primitives like line segments and arcs to range data. The algorithm described in this paper differs from these techniques in that it computes sets of polylines and this way considers the connections between the line segments during the optimization process.

Unfortunately, the transition from infinite lines or fixed types of geometric primitives to polylines is not an easy endeavor. This is due to the high dimensionality of the search problem and since one needs to check various other parameters like connections between line segments, whether or not a polyline can be converted into a polygon, whether a polyline is simple etc. Some approaches for approximating range data with polylines therefore rely on simplifying assumptions. For example, González-Baños and Latombe [9] extract polylines from range scans by exploiting the order of the individual beams given by the range scanner and applying a variant of iterative end-point fit algorithm. Our approach presented in this paper in contrast operates on arbitrary point sets and does not assume any order at all on these points.

The problem of learning polylines has been studied extensively in the field of computer graphics and related areas where the goal is to find good approximations of digitized curves or to optimize meshes. Many of these approaches assume that the vertices of the resulting polygons or meshes are a subset of the given data points [19], [7], [20]. Recently Kreylos and Hamann [12] proposed an approach that uses a simulated annealing scheme to compute positions on a given contour that minimize the distance between the contour and the resulting polylines. They do, however, constrain the resulting polylines to lie on the contour. The approach presented here, however, allows arbitrary positions for the vertices of the polylines and does not require the vertices to lie on a given contour or to be a subset of the data points. Additionally, it uses information about the relative alignment of consecutive line segments in the individual scans.

## III. Polyline Extraction from Range Data

Our approach to learn polylines from a set of data acquired with a mobile robot assumes that an accurate pose estimate is given for all laser scans acquired with the robot. Our current system applies the scan-matching technique developed by Hähnel et al. [11] to determine the pose of the robot during mapping. The input to our algorithm is a set of aligned laser range scans each consisting of 180–361 beams depending on the type of the range scanner used.

One of the key problems when computing polyline approximations is to find a good initial estimate for the polylines. Due to the high dimensionality of the search problem it is of utmost importance to limit the number of necessary operations during the optimization phase described in the next section. To determine an initial set of polylines we proceed in three steps. In the first step we generate a grid map given the laser range scans. The next phase is to compute the contours of that grid map, i.e., the list of cells that correspond to surfaces of the objects. In a third step we compute the initial polylines from the contours.

The grid map generated in the first step stores in every cell the probability that a laser beam is reflected by this cell. To compute this map we perform a ray-casting operation for all laser beams given the locations of the robot and count for each cell how often a laser beam was reflected by that cell (hits) and how often it intercepted a cell without being reflected (misses). The value of a cell $m_{ij}$ that has been intercepted at least once is then given by

$$m_{ij} = \frac{\mathit{hits}_{ij}}{\mathit{hits}_{ij} + \mathit{misses}_{ij}}. \tag{1}$$

Note that the ray-casting operation carried out when using this approach allows to cope with a limited amount of spurious measurements reflected by dynamic objects such as people.

The second step is to compute a contour that corresponds to the surfaces of the objects in the environment of the robot. To determine this contour we again perform a ray-casting operation using all beams and label only those cells as contour where the beam for the first time intercepts a cell whose value $m_{ij}$ exceeds a threshold of $0.5$.

The final step is the computation of a set of polylines from the contours in $m$. To achieve this, we repeatedly extract an endpoint of a contour which becomes a starting point of a new polyline and traverse the contour starting with its neighbor. Thereby we add the traversed cells as new polygon points to the current polyline. If no starting points can be found any more, we process the remaining

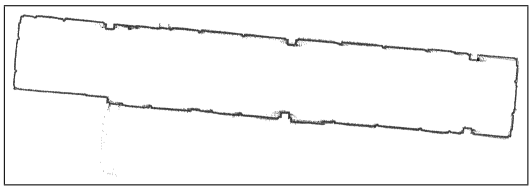

Fig. 1. Endpoints of the aligned range scans acquired with a SICK LMS laser range scanner installed on an Active Media Pioneer II robot.

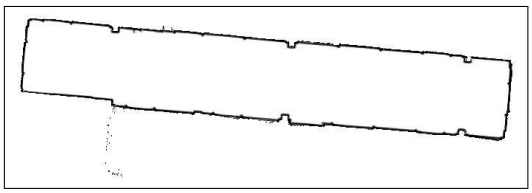

Fig. 2. High-resolution grid map computed from the range data depicted in Figure 1.

cyclic contours by selecting arbitrary points on these contours as starting points. To reduce the size of the initial polyline set we also remove points if their extraction does not change the length of the corresponding polyline.

Figures 1 to 3 show a typical example of this preprocessing phase. The original data shown in Figure 1 depicts the 90,176 endpoints of the laser range scans obtained with a Active Media Pioneer II robot equipped with a laser range scanner after the application of the scan alignment procedure. The size of this environment is 25m times 2.6m. Figure 2 shows the grid map computed from this data. The spatial resolution of this grid map is $2.5$cm and the overall number of contour cells is 1735. The polyline map computed from these contours is depicted in Figure 3. It contains 11 polylines with a total of 622 points.

To enhance readability we display the polylines dashed and dotted so that the individual polylines can be distinguished. During the contour generation process we only create contours for the first cell whose value exceeds a threshold of $0.2$.

## IV. Polyline Optimization

After generating an initial set of polylines the optimization process starts. The goal of this process is to compute a polyline map that minimizes a Bayesian score function based on the Bayesian Information Criterion. We start with a description of the error of a given model.

Suppose $d = d_1, \ldots, d_n$ are the range data and $\mathcal{P}$ is a polyline map. Then the error $E(d \mid \mathcal{P})$ of the data $d$ given $\mathcal{P}$ is defined as

$$E(d \mid \mathcal{P}) = \sum_{i=1}^{n} dist(l^*(i), d_i)^2 + \gamma \sum_{i=1}^{k} 1 - p(\alpha_i \mid x_i, y_i), \quad (2)$$

where

$$l^*(i) = \operatorname*{argmin}_{l \in \mathcal{P}} dist(l, d_i). \quad (3)$$

Here $l^*(i)$ is the line segment $l$ in $\mathcal{P}$ that has the minimum distance $dist(l, d_i)$ to the scan point $d_i$.

Whereas $k$ is the number of vertices in the current polyline, the quantity $p(\alpha_i \mid x_i, y_i)$ specifies the probability of the angle $\alpha_i$ at the position $x_i$ and $y_i$ of vertex $i$ in the scan data. This probability is obtained according to a statistics about the angles between lines fitted to point clusters in the individual input range scans. The term $\gamma$ is a weighing factor that computes the tradeoff between the distance of the data points from the polyline and the angle error.

Unfortunately, there is no closed-form solution to compute the polyline map $\mathcal{P}^*$ with a fixed number of line segments that minimizes $E(d \mid \mathcal{P})$:

$$\mathcal{P}^* = \operatorname*{argmin}_{\mathcal{P}} E(d \mid \mathcal{P}) \quad (4)$$

Therefore, we start with the initial polyline map and use local search to minimize Eq. (2). During this search we apply several operators on the endpoints of the line segments and on the polylines themselves. These operations, which are described in more detail in the remainder of this section, are also depicted in Figure 4.

### A. Overlaps

The first operation deals with polylines that overlap at their ends (see Figure 4). Such overlaps mostly appear in the initial map where they are caused by slight alignment errors of the scan matching procedure. In the computer graphics literature this operation is also know as mesh-zippering.

### B. Merging Polylines

The second operation connects two polylines when their endpoints are closer than $15$cm to each other. In our current system this operation is also able to convert a polyline into a closed polygon.

### C. Splitting Polylines

An important operation is to split a polyline into two separate sequences of line segments. This operation is necessary in situations in which there is a sufficiently long interval on a line segment in which there are no data points. In our current implementation the threshold for splitting line segments is 20cm.

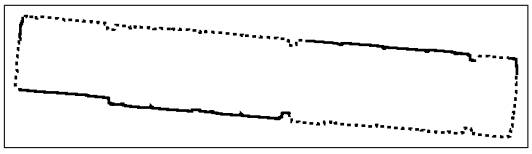

Fig. 3. Resulting initial set of polylines.

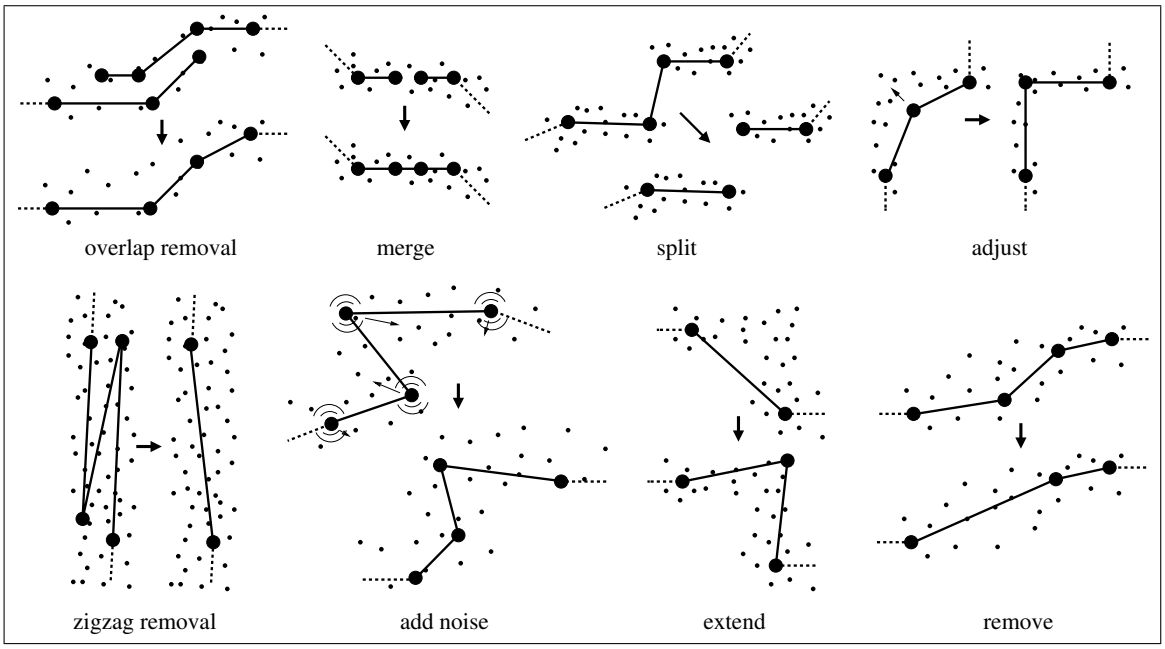


Fig. 4. Operators applied to the polylines during the optimization phase.

### D. Adjusting the Position of Vertices

One of the most important operations for the optimization of the polylines is to determine better positions for the endpoints of the line segments. To achieve this we consider each polyline and perform a single hill-climbing step for each endpoint of its line segments. We repeat this process until no further improvement is obtained. In extensive experiments we found that eight directions and a step-width of 1cm yields an optimal trade-off between computational requirements and accuracy.

### E. Removing Zigzag Lines

Especially during the previously described adjustment procedure it frequently happens that so-called zigzags lines are introduced. The operation to detect and remove such zigzag lines is similar to the removal of overlaps.

### F. Adding Noise

During the process described so far we sometimes observed that the overall system gets stuck in local minima. Reasons for this are the crisp associations in the error function and the limited step-width in the adjustment process. To overcome such situations we added an operation that introduces random noise to the position of the vertices. The noise is normally distributed with a variance of $5cm$. During the overall optimization process this variance is continuously decreased.

### G. Adjusting the Number of Vertices

Another key issue is to choose the appropriate number of model components. In the context of our problem this corresponds to estimating the number of points in $\mathcal{P}$. In previous approaches the number of endpoints of the polylines is typically controlled by fixed thresholds [19] or by more complex functions weighing different aspects [7].

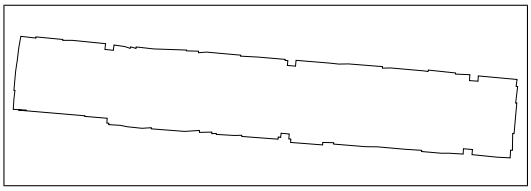

Fig. 5. Resulting single closed polyline with 109 points of laser range scan data depicted in Figure 1.

In the context of computer graphics this approach appears to be justified since the goal is to determine a model that provides a mostly accurate impression to the observer. In our case, however, we are interested in a reasonable compromise between number of model components and the quality of the approximation. Obviously, a model that connects all data points would provide an optimal fit since the error would be zero. On the other hand, a model consisting of just one line segment would minimize the number of model components. In the field of machine learning a popular measure is the *Bayesian Information Criterion* also denoted as *BIC*. In the context of the polyline fitting problem, the Bayesian information criterion is

$$E_{BIC}(d \mid \mathcal{P}) \quad = \quad \alpha E(d \mid \mathcal{P}) + k \log n \qquad (5)$$

where $k$ is the number of model components and $n$ is the number of data points. The constant $\alpha$ is a scaling factor that depends on the accuracy of the underlying sensor. It was set to $500$ in all our experiments. The major advantage of the *BIC* is that it incorporates all parameters of a learning problem, namely the approximation error, the number of model components and the size of the data set.

After each application of the optimization operators described above we use the *BIC*-value to determine whether or not to introduce an additional vertex or to remove an

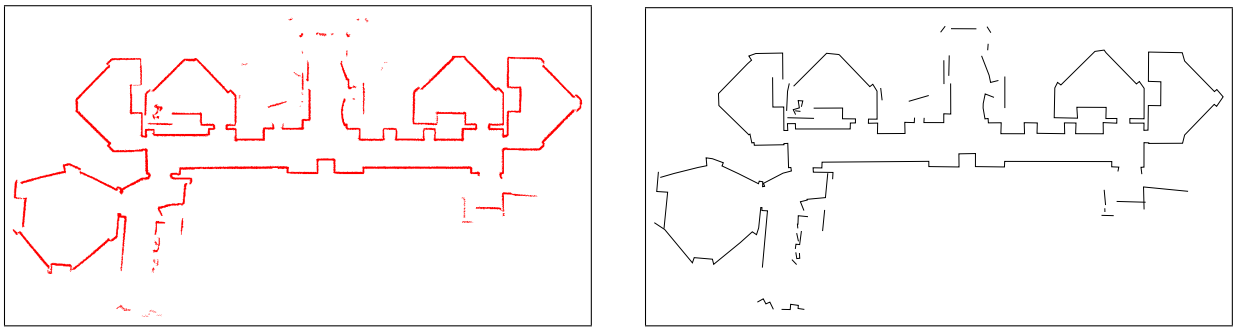

Fig. 6. Scan data and polyline map of the museum in Herakleion. The size of the environment is 38 times 18 m and the input data set contained 268,640 scan points. The resulting polyline map contains 41 polylines with 262 vertices.

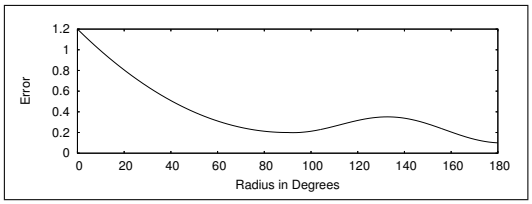


Fig. 7. The density $1 - p(\alpha_i \mid x_i, y_i)$ used for the indoor experiments.

existing one. To add a vertex to a polyline we determine the line segment $l$ which has the largest average squared distance to its associated data points. We then replace $l$ by two line segments which connect the point furthest from $l$ with the two endpoints of $l$. We then introduce slight noise to the complete polyline and apply the adjustment procedure. If the *BIC*-value decreases we accept the new polyline, otherwise we keep the old one. Additionally we check whether there is a vertex that reduces the *BIC*-value when we remove it from the map. If this is the case, we delete the vertex yielding the highest reduction of the *BIC*-value. The extension or removal operators are applied iteratively until no improvement can be achieved any more.

Our algorithm repeatedly runs over the whole data set and tries to apply these operations. It always stores the model with the best *BIC*-value found so far. The overall process is stopped until no improvement can be obtained for five consecutive iterations.

An example polyline map obtained for the initial contour map depicted in Figure 3 is displayed in Figure 5. The map consists of a single polygon containing 109 vertices. It took our system 90 minutes and 9 iterations to compute this map. Figure 7 depicts the function $1 - p(\alpha_i \mid x_i, y_i)$ used to calculate the error of relative angles in the polyline given the initial range data. This function was used in all indoor experiments described in this paper.

## V. Experimental Results

The algorithm described above has been implemented and tested successfully using various data sets gathered with real robots. The experiments presented in this section are designed to illustrate that our system can learn highly accurate maps from the range data. They also demonstrate that our algorithm simultaneously yields a serious compression of the input data of up to several orders of magnitude.

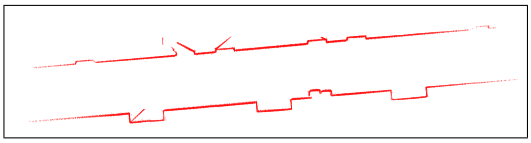

Fig. 8. Laser range data of a hallway at Stanford University. This data set contains 70,269 scan points.

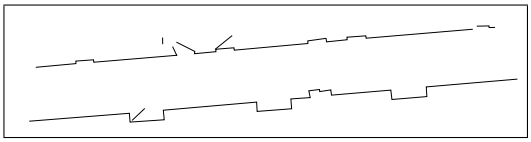

Fig. 9. Polyline map for the scan data depicted in Figure 8. This map contains 7 line sequences and 53 endpoints.

The first experiment described in this section has been carried out using data recorded in the Herakleion Museum in Greece. This data set consists of 268,640 laser beams. The polygon generated for the contour map contained 122 polylines consisting of a total of 1,888 vertices. The overall optimization took 6,037 seconds. In the end we obtained 41 line sequences consisting of 262 vertices which corresponds to a reduction by three orders of magnitude. The original scan data and the resulting polyline map are depicted in Figure 6. As can bee seen from the figure, the approximation is highly accurate.

The second experiment is designed to illustrate the overall learning process. Here the system was applied to the laser range data depicted in Figure 8. The resulting polyline map is depicted in Figure 9. Figure 10 shows the evolution of the *BIC*-value over time. As can be seen from the figure, the accuracy of the model improves seriously

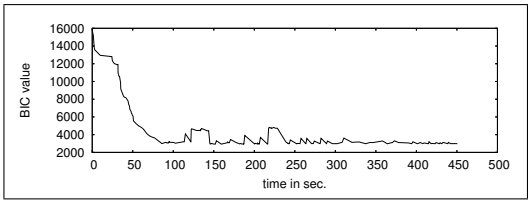


Fig. 10. Evolution of the value of the Bayesian Information Criterion over time during the learning of the hallway map shown in 9. The peaks in the evaluation correspond to situations in which noise was added or geometric simplifications had been applied.

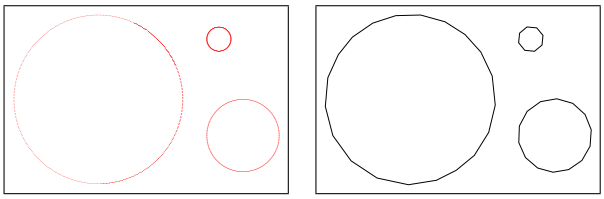

Fig. 11. Circular range scans with radius 1, 3, and 7m and consisting of 360 measurements each (left image) as well as polygons computed by our algorithm (right image).

during the first iterations of our algorithm. This is mainly due to removals of vertices. After 100 seconds the system only achieves slight improvements. The peaks in the curve come from the operation that adds noise. This operation slightly increases the error which in turn is then corrected by the procedure optimizing the positions of the vertices.

The goal of the final experiment is to illustrate that our algorithm can yield accurate approximations of even circular structures. Figure 11 shows the result of approximating three circles with our algorithm. The overall number of data points per circle is 360. During this process the prior about relative angles between line segments extracted from the range data was uniformly distributed. The resulting number of polylines is three with eight, fourteen and twenty segments each. Note that the *BIC* results in different numbers of vertices per circle. In contrast to other criteria, which add more model components to areas with higher curvature, the *BIC* yields more vertices in the larger circles. This is natural, since the *BIC* considers the sum of squared errors. If two circles are approximated by polygons of the same length, we obtain a greater reduction in the error if we add a line segment added to the polygon that approximates the larger circle than if we add it to the polyline fitted to the smaller polygon.

## VI. Conclusions

In this paper we presented an approach to learn polyline maps from range data. Our approach first processes the input data to compute a contour of the objects in the environment of the robot. Based on this contour it generates an initial set of polylines which are optimized subsequently. During the optimization process it utilizes information about relative angles between line segments in the original scan data. To control the number of vertices in the model it applies the Baysian Information Criterion (*BIC*). The approach has been implemented and tested on real data gathered with mobile robots in different environments. The resulting maps are highly accurate and at the same time require three orders of magnitude less space than the original input data.

Despite these encouraging results there are several warrants for future research. The current system requires a huge amount of computational resources and can only be applied offline after the data has been recorded. In the future we therefore will investigate how to speed-up the learning process in order to obtain an on-line variant and to even further increase the accuracy. Additionally we will investigate how to learn models that contain more features than line segments.

## Acknowledgment

This work has been supported by the German Science Foundation (DFG) under the research grant SFB/TR-8.

## References

[1] K. Arras and R. Siegwart. Feature extraction and scene interpretation for map-based navigation and map building. In *Proc. SPIE, Mobile Robotics XII*, volume 3210, 1997.

[2] D.J. Austin and B.J. McCarragher. Geometric constraint identification and mapping for mobile robots. *Robotics and Autonomous Systems*, 35(2), 2001.

[3] H. Baltzakis and P. Trahanias. An iterative approach for building feature maps in cyclic environments. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2002.

[4] J. Crowley. World modeling and position estimation for a mobile robot using ultrasound ranging. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1989.

[5] R.O. Duda and P.E. Hart. *Pattern classification and scene analysis*. Wiley, New York, 1973.

[6] M.A. Fischler and R.C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image and analysis and automated cartograph. *Comm. of the ACM*, 24(6), 1981.

[7] M. Garland and P. Heckbert. Fast polygonal approximation of terrains and height fields. Technical Report CMU-CS-95-181, Carnegie Mellon University, School of Computer Science, Pittsburgh, PA 15213, 1995.

[8] J. Gonzales, A. Ollero, and A. Reina. Map building for a mobile robot equipped with a 2d laser rangefinder. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1994.

[9] H. González-Baños and J.-C. Latombe. Robot navigation for automatic model construction using safe regions. In *Proc. of Int. Symposium on Experimental Robotics (ISER 01)*, 2000.

[10] J.S. Gutmann, T. Weigel, and B. Nebel. A fast, accurate, and robust method for self-localization in polygonal environments using laser-range-finders. *Advanced Robotics*, (8):651–668, 2001.

[11] D. Haehnel, D. Schulz, and W. Burgard. Map building with mobile robots in populated environments. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2002.

[12] O. Kreylos and B. Hamann. On simulated annealing and the construction of linear spline approximations to scattered data. *IEEE Trans. Visualization and Computer Graphics*, 7(1):17–31, 2001.

[13] J. Leonard, P. Newmann, R.J. Rikoski, J.D. Tardós, and J. Neira. Towards robust data association and feature modeling for concurrent mapping and localization. In *Proc. of the Tenth International Symposium on Robotics Research (ISRR)*, 2001.

[14] Y. Liu, R. Emery, D. Chakrabarti, W. Burgard, and S. Thrun. Using EM to learn 3D models of indoor environments with mobile robots. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, 2001.

[15] P. MacKenzie and G. Dudek. Precise positioning using model-based maps. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1994.

[16] C. Martin and S. Thrun. Online acquisition of compact volumetric maps with mobile robots. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2002.

[17] P. Newmann, J. Leonard, J.D. Tardós, and J. Neira. Explore and return: Experimental validation of real-time concurrent mapping and localization. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2002.

[18] S.T. Pfister, S.I. Roumeliotis, and J.W. Burdick. Weighted line fitting algorithms for mobile robot map buildi8ng and efficient data representation. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2003.

[19] P. Rosin. Techniques for assessing polygonal approximations of curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6), 1997.

[20] M. Salotti. Optimal polygonal approximation of digitized curves using the sum of square deviations criterion. *Pattern Recognition*, 35(435-443), 2002.

[21] B. Schröter, M. Beetz, and J.-S. Gutmann. RG mapping: Learning compact and structured 2d line maps of indoor environments. In *Proc. of the International Workshop on Robot and Human Interactive Communication (ROMAN)*, 2002.