Vector Maps in Mobile Robotics

Ales Jelinek
Brno University of Technology
Brno, Technicka 12
Email: xjelin08@stud.feec.vutbr.cz

***Abstract*—The aim of this paper is to provide a brief overview of vector map techniques used in mobile robotics and to present current state of the research in this field at the Brno University of Technology. Vector maps are described as a part of the simultaneous localization and mapping (SLAM) problem in the environment without artificial landmarks or global navigation system. The paper describes algorithms from data acquisition to map building but particular emphasis is put on segmentation, line extraction and scan matching algorithms. All significant algorithms are illustrated with experimental results.**

I. Introduction

Mapping in robotics is a subject of research for a number of years and large progress was already achieved. Paper [1] describes theoretical way, how to build and update a consistent map, which converges to a precise image of the real environment (at certain level of detail). Through the years, more improvements and new algorithms were invented for this task, good tutorial papers are for example [2] [3], or the book [4]. Formulation and mathematical description of the SLAM problem is unified and generally accepted today, but a complete solution for 3D dynamic environment is still not known and SLAM as a whole is still under active research.

The most of today's work in the field is concentrated on two main bottlenecks of known algorithms:

1) Computational optimization: Every map contains some data describing the real environment. Computational efficiency is usually evaluated in big O notation with respect to n, reflecting the number of cells in an evidence-grid map, the number of landmarks in a geometric map, etc. Non-optimized solutions have $O(n^2)$ or even exponential complexity, but several algorithms are more efficient, for example FastSLAM [5] with $O(m\ log(n))$ complexity (where n is the number of landmarks and m the number of particles in the Rao-Blackwellized filter). This algorithm can handle orders of magnitude more landmarks than $O(n^2)$ solutions. Computational efficiency is a key to large maps with a lot of details.

2) Feature recognition and matching: Robust and reliable feature (landmark) extracting and matching is crucial for most of SLAM algorithms. New data has to be associated with older measurements precisely, or at least with negligible number of mismatches. Feature extraction is needed when camera or laser scanner is used to provide information about robot's surroundings. For example corners and T-junctions are usually searched in input data and then the whole pattern is correlated with already known map. Several complications may arise, because the same feature may look different from different points of view and data registration in its straightforward implementation is exponentially complex task.

Data structure of a map is closely related to both of these problems. Operations executed on the map have to be as fast as possible - especially updates with the new information and feature search/comparison are crucial. Representation of obstacles, free space and unexplored areas effects feature recognition robustness. Another attribute of a map is its ability to approximate the real environment. Importance of this attribute depends on further usage of the map. If the robot is meant to explore unknown area and provide information about it, requirements will be probably higher than in case of a robot, which just needs to avoid obstacles and memorize a short history of its path.

There are more possible ways to represent a robotic map. One of the most common is robot evidence grid, which parcels continuous world into a set of squares (cubes in 3D). Size of a square is optional and may be even adaptive to better fit the reality, but this style of approximation of smooth world is still rather crude, because straight edges which are not colinear with grid lines are jagged regardless of grid resolution. Evidence grids are widely used and well established in many applications, where this downside is negligible. Visually better results are achieved with vector maps, where edges are represented with line segments. Good example of classical vector based mapping system is VecSLAM described in [6]. The aim of authors work is to create a system for high quality mapping, therefore vector maps were chosen as a good approximation of the real world.

Map representation is only one part of the problem. Robot exploring unknown environment needs to update its map with new information. Data are usually obtained from rangefinder sensors such as laser scanners and sonars in the form of a point cloud and need to be incorporated in the current map. Probably the most used technique for point cloud registration is Iterative Closest Point (ICP). Good description and historical survey of this method is available in [7]. Efficient variants of this method were published in [8]. Furthermore, it is possible to use ICP for registration of more complex formations, such as surfaces of scanned objects. This approach is described in [9] and [10].

There are several alternatives to ICP, which are trying to generalize the notion of "closeness" or "similarity" from points to lines, triangles etc. Similarity of points is usually measured as their Euclidean distance. An edge is often described as a segment of a line. Having two such segments, we can define an area they demarcate and use it as a measure of similarity. This definition is not unified and it is possible to find several different approaches, for example sum of squared distances from point in a scan to the nearest edge in the map [11], penalizing function of "non collinearity" and spatial distance [12], or a criterion based on newly introduced matrix

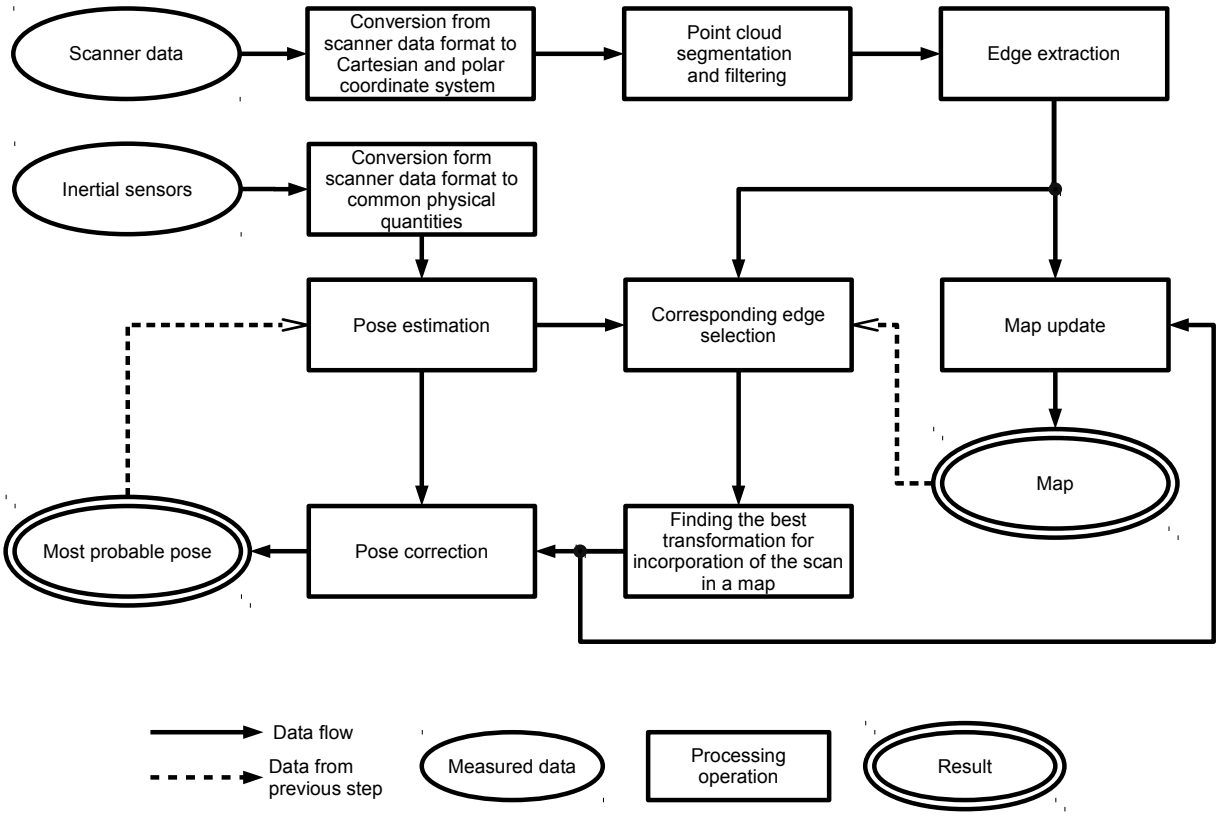


Fig. 1. Overall scheme of point cloud processing system from data acquisition to map and pose update.

scalar product [13]. Another interesting approach is critical ray method proposed in [14], which does not use line similarity at all. Diversity of these approaches is high and none of them is considered to be remarkably better than the others, therefore the author have decided to develop his own method and test new ideas, because this field of research is definitely not fully explored.

Reducing point cloud to a set of line segments (in 2D) or triangles (in 3D) has very useful properties. At first, information from hundreds or thousands of points is expressed with (at most) tens of objects. Data reduced in count are processed faster and therefore more complex algorithms may be used. On the other hand, mathematics becomes more complicated, because lines and faces have more degrees of freedom and constraints than points and as written above, similarity is not as easy to enumerate.

The process of line extraction has generalizing ability. After interpolation by line segments (or triangles), corners become sharper and noise is suppressed, which results in better looking (more corresponding to the real world) map. In addition (depending on line extraction algorithm) edges might be estimated with higher accuracy than single point due to averaging effect.

There is no restriction on using $N-1$ dimensional objects in N dimensional space. A generalization of ICP for lines in 3D space is described in [15]. Paper [16] describes whole set of algorithms for matching sets of lines in 3D and many others are possible to find.

For even better approximation of the real world, curves and curved planes could be used. These are more complicated for computation and not too much papers are related to this topic. An example of such approach is a method [17] based on graph theory.

II. Point cloud processing system

The overall diagram of the measurement and mapping process is depicted in Fig. 1. Every time a measurement is acquired, the whole process is run through. Each rectangular box in the diagram means one algorithm performing operation on its input data. The whole process leads to update of the map the robot creates and the pose of the robot itself. SLAM itself is implemented in a very elementary way and no attempts to build converging map were made yet. The main objective in current state of development is to test data processing, edge extraction and scan registration algorithms.

Laser scanner used to obtain all data in this chapter is Velodyne 32 HDL, which provides approximately two thousand points (per turn) in polar coordinates for each of its thirty two lasers. Only horizontal set of measurements was taken into account for 2D map.

At the beginning, all data from sensors need to be converted into the data format useful for following algorithms. This step

also ensures usability of the system for any instrumentation with suitable properties. A disadvantage of this process is a certain enlargement of data size, because laser scanners and other sensors usually use some sort of data compression to minimize data flow. In case of Velodyne 32 HDL this means conversion from 16-bit integer to 32-bit floating point number.

To speed up corresponding edge selection, crude pose estimation is useful. It enables to specify part of the map, where the robot most probably is, and therefore, the search space of correspondences is reduced a lot. As is well known in robotics community, techniques such as inertial sensors or odometry cause integration of error of pose estimation. This makes them useless after certain amount of time. In the presented algorithm, these are only used for measurement of relative pose change between two following scans. This ensures that the estimated pose error is always in a known limit. To get an absolute pose, this estimation is added to the last known pose, determined in the previous iteration of the data processing (dashed arrow leading to "Pose estimation" block in Fig. 1).

A. Filtering and clustering

Treatment of raw point cloud is a more complex task. At first it is necessary to filter out noise and outliers and to find dense clusters of points defining edges of solid objects in the environment. Raw data from one set of measurements are shown in Fig. 2a. It is clear that scans are full of irrelevant points and odometry was not very accurate.

Some noise is always present. Laser scanner measures distance with certain precision, which is stated in the datasheet. The other sake of noise is too dissected environment, where individual faces of objects are so small that rangefinder cannot scan them properly. A good examples are treetops and bushes. Noisy parts of point cloud are best to treat as a solid body with defined borders and density. Other possibility is to filter them out, which causes loss of information, but on the other hand, with a careful filtering, it is possible to "see" through low density objects (e.g., see wall behind a bush).

Outliers in point cloud are usually caused by scanning too distant or too close objects, which are out of the range of the particular laser scanner. These are easy to identify and can be removed without serious impact on the quality of a scan. More interesting to deal with are outliers originating from a reflection. In a common environment, there can always be objects, which are able to reflect laser beam and corrupt the measurement. If the reflecting object is small and curved, few beams, which hit it, are scattered wide apart and appear as randomly distributed single points in a point cloud. These are generally easy to remove. Another case are large, flat objects such as mirrors and windows, which uniformly reflect large amount of rangefinder beams. This effect causes phantom objects to appear in the scan and there is probably no way to detect this without additional sensors or a set of scans from different positions.

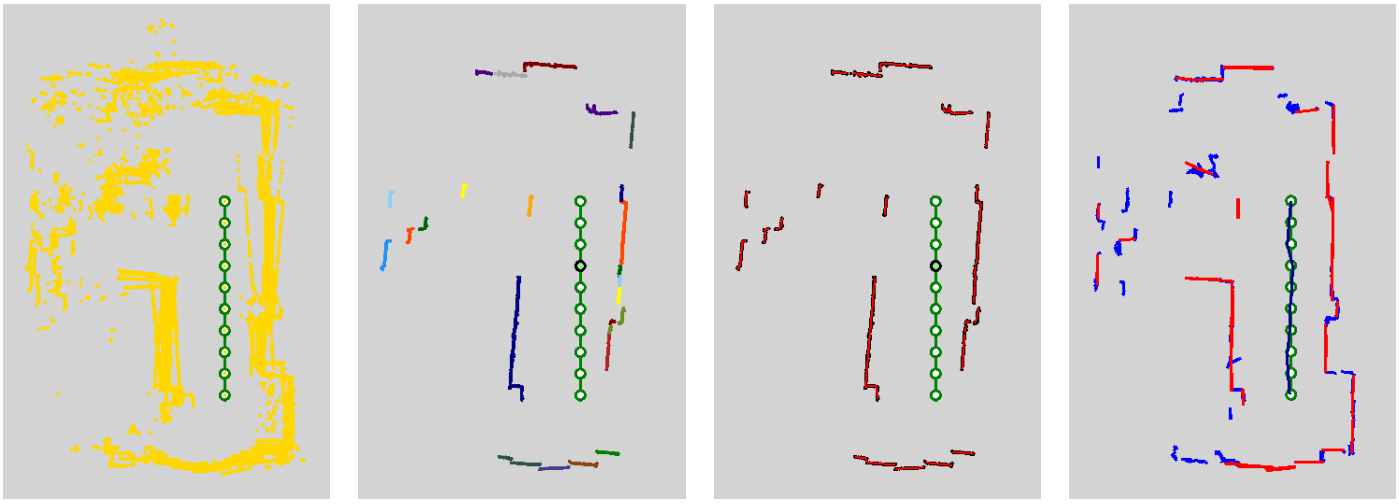
Clustering is used to remove all unnecessary points from a scan and to determine dense clusters of points, which could potentially form an edge of a real object. An algorithm used to get result depicted in Fig. 2b is as follows: From the laser scanner we get data sorted by an angle. Algorithm proceeds from one point to another and checks if k previous points are closer or farther, than a certain distance. Three alternatives may occur. A point might be too far away from all of the previous ones. This leads to establishment of a new cluster. If the working point is close enough to some previous points and all these points belong to the same cluster, the working point is added to it. Third case arises, when the working point can belong to more than one cluster. In such a situation, all those clusters and the working point are joined together. In the end, clusters with the size under a certain threshold are called outliers and are removed from the scan. From a practical point of view, it is wise to make the distance comparison threshold adaptive, because the farther the detected obstacle is, the sparser the measured points are. Scaling the threshold linearly with the distance between the working point and the robots position is a good strategy.

The proposed algorithm safely removes noise from too dissected objects as well as random outliers, because these create large number of very small clusters (typically less than ten points), which are removed in the final stage of the process. Phantom objects are not recognized. Noise caused by laser scanner itself removes the following algorithm for line extraction.

This theme is directly related to line simplification algorithms (because points in a scan are sorted) and both of these are discussed in a lot of fields of research. In addition to robotics, similar approaches can be found in cartography, computer vision and computer graphics. Plenty of algorithms were developed during last decades, having different properties, such as speed, memory requirements and quality of approximation. Good survey comparing line extraction/simplification algorithms in robotics is [18]. Results of this paper clearly show that algorithms such as RANSAC or Hough transform are not fast enough for online processing of point clouds.

Good performance was observed when using split-merge, incremental and line regression algorithms. Split-merge (also known as Douglas-Peucker algorithm [19]) was the fastest one and belongs to wide family of $O(n)$ and $O(n\ log(n))$ complex algorithms used in cartography [20]. Main drawback of these algorithms is that they are based on point elimination, and therefore, a lot of information is discarded.

The edge extraction algorithm from the diagram in Fig. 1 is based on line regression and works as follows: It passes through the cluster and computes the least square approximation and the variance of selected set of points. If the variance gets over predefined limit, the line is saved, and algorithm proceeds computing new line. The last line terminates at the end of the cluster. This approach is advantageous, because all points contributes to the result and contained information is better utilized. Although loss of information was overcome by the least squares approximation, the algorithm is still not optimal in terms of "best fit poly-line". All but the last line have maximal permitted variance, which means, that for too high threshold lines go over corners and an approximation is worse, than it could be. More technically, the total sum of squared distances from each point to its regression line is higher, than is achievable with given number of lines. Fine tuning of the threshold is therefore necessary. Complexity of the algorithm is $O(n)$. Result of line extraction is depicted in Fig. 2c.



(a) Raw data (b) Segmentation (c) Line extraction (d) Complete map

Fig. 2. a) Depiction of all measured point clouds (yellow) with estimated robot path (green). No scan fitting or other postprocessing is used, image corresponds to raw data from sensors. b) Segmentation of one scan is shown on this figure. Each color means a cluster, which is believed to be continuous edge in reality and which is separate from the others. Noise and outliers are filtered out and not not displayed. c) Line extraction in segmented point cloud (black). Each cluster is approximated by a line or a poly-line (red). d) A map composed using the line similarity criterion. Red edges are long enough to participate in scan fitting process, blue edges are shorter and just add details to the map. Green trajectory is acquired from odometry and dark blue one is an estimation of true trajectory.

B. Scan fitting

When updating a map with new scan, corresponding edges must be selected at first. It is responsibility of an operator or a path planing algorithm to obtain new scans frequently enough to ensure overlap of a new scan and the map. This means the robot always has to see part of the already known environment (except the first scan of course). Acquiring more scans from one location leads to more precise map (averaging effect).

Scan fitting is the point, where crude pose estimation comes in useful, because searching through the whole map would be very computationally expensive. Knowing approximate pose and its maximal error allows us to considerably reduce search space and keep computational complexity constant, independent on the size of the whole map (assuming the number of visible details is comparable across the map). If the robot is suddenly moved far away from its previous location, it becomes lost and the algorithm is not capable of working correctly anymore.

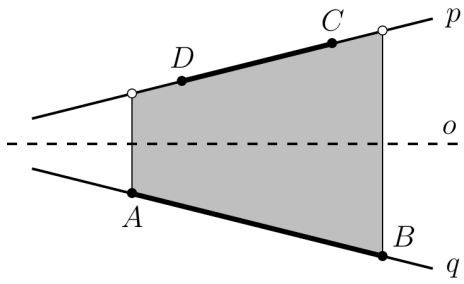
Only long enough edges are used for the process, because the shorter ones are not determined with a sufficient precision. The maximal angular error of the pose estimation is directly used as the maximal angle between possibly similar edges. If the angle is inside a tolerance, shortest distances of end points of first edge to the second edge are computed. If at least one is shorter than maximal permitted shift between two scans, edges are overlapping each other and can belong to one real edge.

There are three possible situations after this step. For some edges from the new scan, there is no similar one in the map. These probably belong to an area which was not explored yet, or where only short edges were determined. New edges do not contribute to scan fitting process. The second group of new edges has one similar match in the map and these are directly used for scan matching. In the last group, there are edges with more than one possible match. At this situation, it is hard to say, which possible pair is the best one, or if more edges should be connected together. Current practice is to remove edges already used in the second group and from the rest to select the closest one (in the sense of overlapping). In rare cases, the edges corresponding in real world are not the closest ones. This leads to mismatch and can even corrupt the map, but in most cases, correctly matched edges prevail and the mismatch is unnoticeable.

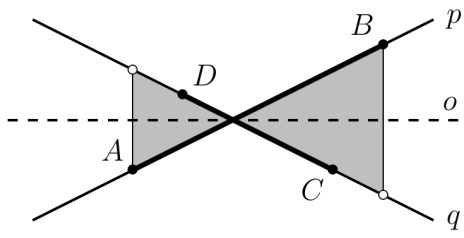
At this point, pairs of corresponding edges from the new scan and the map are known and the actual fitting may take place. As well as in the previous step, the line similarity criterion will be used, but now with slightly different properties. An idea, that lines are more similar as the angle they form is smaller is still true, but overlapping criterion used before is not advantageous. Instead, as already mentioned in the introduction, an area defined by both edges is used. The main idea of the criterion is depicted in Fig. 3.

Let two edges be defined by lines p and q with the end points AB and CD. At first, an axis of angle formed by these edges is found. Then, for all end points, their perpendicular projections (with respect to the axis o) on an opposite line are found. Two end points and their projections form a quadrangle as shown in Fig. 3 for points AB (both cases). There are six possible quadrangles and the largest one has to be used for the criterion enumeration. The square of that area is used as a similarity metric.

The function for the similarity enumeration is not strictly convex, therefore for one pair of edges, there is not a single minimum defining one pose in which both edges are the most similar. Instead, there is an infinite number of possible poses, in which the criterion is zero. This property is clear from the geometrical visualization of collinear edges sliding over each other and corresponds to the problem of localization in a long corridor. If a robot “sees” only collinear walls, it is



(a) No intersection.



(b) With intersection.

Fig. 3. Line segments similarity criterion. The smaller is the gray area, the more similar are both line segments.

not able to determine the displacement along an axis of the corridor - one degree of freedom cannot be estimated from available data. For an exact solution, at least two skew pairs of edges must be used.

Fitting itself is an iterative process, because no closed form solution was found so far. A new scan is transformed using the standard homogeneous transformation. The goal is to find such translation and rotation, so that the sum of the squared areas of all edge pairs is the smallest possible. Nelder-Mead simplex method was used to find the global minimum. For longer edges, the area they demarcate is large even if an angle they form is relatively small. This means, that longer edges have greater impact on scan fitting, which is correct, because long edges are determined by more points, and therefore, be more precisely known.

C. Map building

After the fitting process is finished, a new scan is transformed using the found parameters and edge pairs are joined to form a new set of edges in updated map. The same transformation, which was used to update the map, is applied on the path estimate to correct the odometry error. Assuming static environment, the information about pose gotten from scan fitting is absolute, and therefore, an error should not have integrative characteristics, however, this is ideal state which requires precise map corresponding to the environment.

The main problem of map building process in current state of development is that the map does not converge. Scans are incrementally fitted to the map, but new measurement influences only the part of the map, which is currently visible and new information is not propagated through the whole map to assure convergence. In other words, change in length of one edge should affect all connected edges (including those which are not actually seen), but current implementation is not able to handle this. Integrative characteristics of error of pose and map is therefore not suppressed. On the other hand, using edges and line similarity criterion for scan fitting is very accurate and divergence process is much slower.

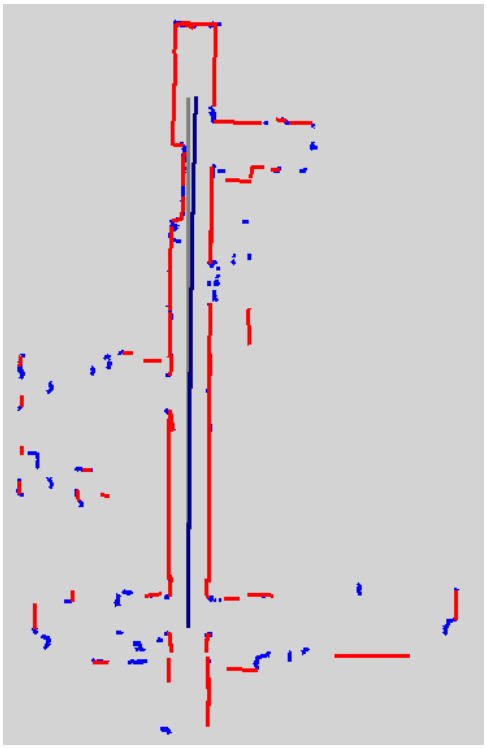


Fig. 4. A larger map composed using the presented processing system. Edges are colored in the same way as in Fig. 2d. Gray trajectory is acquired from odometry and dark blue one is an estimation of the true trajectory.

Results of this step are maps in Fig. 2d and Fig. 4. In comparison with a naive scan merging in Fig. 2a, new maps are much more usable for navigation as well as environment documentation. Further processing for even better results can follow at this point, one example of possible operations is additional merging of shorter edges (blue edges in Fig. 2d and Fig. 4) to form a cleaner map.

D. Experimental results

Practical testing was made in office and laboratory environment. Algorithms based on edge detection are well suited for such application, because artificial objects are usually simply shaped with lot of flat surfaces. On the contrary, a natural environment is much more complicated with lot of details and lack of flat surfaces, therefore effective line extraction would be very complicated or even impossible.

The robot used for testing was Orpheus X3, which is four-wheeled vehicle with differential steering. Laser scanner used was Velodyne 32 HDL. During the experiments, the robot was manually operated and all measurements were taken, when the robot was not moving.

The first experiment used for demonstration of the processing pipeline is depicted in Fig. 2a to Fig. 2d. Ten measurements 60 centimeters apart were taken. The odometry error was intentionally increased to demonstrate correction ability of

scan fitting algorithm. Outliers and noise filtration was also proved to be working, for example small cloud of points on top of Fig. 2a was correctly removed.

The second experiment was made in a larger scale on a corridor and resulting map is shown in Fig. 4. Robot moved from starting position to the end of the corridor and than returned back. Total length of the traveled path was 46.710 meters and 33 measurements were taken along the way. The difference between the real and the estimated position of the last point of trajectory was only 6 mm, which is even less than the precision of used laser scanner (20 mm). The experiment was held only once, therefore repeatability is not yet determined.

Fairly good results are caused by two main factors. At first, the line extraction algorithm with the least squares approximation is capable of reducing noise through averaging and provide more accurate edge positions, than single points in a point cloud. The second reason is that the robot moves in one room and always sees edges from the first scan, therefore an error of map and pose is not integrating. If the robot would move along a path with lot of turns and after some time would not see its starting location, this problem would start to manifest. However, these experiments were made to proof the concept of scan fitting via edge similarity criterion and SLAM itself is not yet at the center of the research.

III. Conclusion and future work

The paper presents the set of algorithms used to process data from laser scanner and build a vector based map. The whole scheme of the process is depicted in Fig. 1. Most algorithms used are new and still under testing, but findings stated in literature (e.g. [10], [12], [16]) were proved to be true, because scan matching based on edge detection yields quality results and definitely can be used for solving SLAM problem. At least, considering results in Fig. 2d and Fig. 4, current state of the research is very promising.

The main task for the nearest future is development of good representation of the topological-metric map, to enable real SLAM to be implemented. Topological information is necessary for visible (or just close) edges finding algorithm and for path planning. Metric information is crucial for mapping itself, scan matching, and path planning. Term "good representation" means such data structure, that would be efficient for search and computing algorithms. Nearest neighbor of polygon search or finding an edge in area with certain coordinates are only a few examples of problems which are necessary to be efficient in a good mapping algorithm.

The second requirement expected from a good map is a convergence. As the robot explores surrounding environment, the map should adapt to be closer to reality. This is not achieved now and lot of work is focused on this task. With authentic map and well working fitting algorithm, localization itself should not be too complicated and a solution for the SLAM problem using vector maps should be completed (at least in its main aspects).

Acknowledgment

This work was supported by the project CEITEC - Central European Institute of Technology (CZ.1.05/1.1.00/02.0068) from the European Regional Development Fund.

References

[1] M. Dissanayake, P. Newman, S. Clark, H. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (SLAM) problem," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 3, pp. 229–241, Jun. 2001.

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

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

[4] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. The MIT Press, 2005.

[5] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A factored solution to the simultaneous localization and mapping problem," *AAAI/IAAI*, 2002.

[6] H. J. Sohn and B. K. Kim, "VecSLAM: An Efficient Vector-Based SLAM Algorithm for Indoor Environments," *Journal of Intelligent and Robotic Systems*, vol. 56, no. 3, pp. 301–318, Feb. 2009.

[7] L. Oswald, "Recent development of the Iterative Closest Point algorithm," p. 39, 2010.

[8] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm," *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*, pp. 145–152.

[9] P. Besl and H. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[10] Z. Zhang, "Iterative Point Matching for Registration of Free-Form Curves and Surfaces," *International Journal of Computer Vision*, vol. 13, no. 2, pp. 119–152, 1994.

[11] H. Pottmann, S. Leopoldseder, and M. Hofer, "Registration without ICP," *Computer Vision and Image Understanding*, vol. 95, no. 1, pp. 54–71, Jul. 2004.

[12] J. Elseberg, R. T. Creed, and R. Lakaemper, "A line segment based system for 2D global mapping," *2010 IEEE International Conference on Robotics and Automation*, pp. 3924–3931, May 2010.

[13] Q. Li and J. Griffiths, "Iterative closest geometric objects registration," *Computers & Mathematics with Applications*, vol. 40, no. 10-11, pp. 1171–1188, Nov. 2000.

[14] E. Tsardoulias and L. Petrou, "Critical Rays Scan Match SLAM," *Journal of Intelligent & Robotic Systems*, vol. 72, no. 3-4, pp. 441–462, Feb. 2013.

[15] M. Alshawa, "ICL : Iterative closest line - A novel point cloud registration algorithm based on linear features," *Ekscentar*, no. 10, pp. 53–59, 2007.

[16] B. Kamgar-Parsi and B. Kamgar-Parsi, "Algorithms for matching 3D line sets." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 5, pp. 582–93, May 2004.

[17] K. Demarsin, D. Vanderstraeten, T. Volodine, and D. Roose, "Detection of closed sharp edges in point clouds using normal estimation and graph theory," *Computer-Aided Design*, vol. 39, no. 4, pp. 276–283, Apr. 2007.

[18] V. Nguyen, S. Gächter, A. Martinelli, N. Tomatis, and R. Siegwart, "A comparison of line extraction algorithms using 2D range data for indoor mobile robotics," *Autonomous Robots*, vol. 23, no. 2, pp. 97–111, Jun. 2007.

[19] D. H. Douglas and T. K. Peucker, "Algorithms for the Reduction of the Number of Points Required to Represent a Diditized Line or its Caricature," *Cartographica: The International Journal for Geographic Information and Geovisualization*, vol. 10, no. 2, pp. 112–122, Oct. 1973.

[20] W. Shi and C. Cheung, "Performance Evaluation of Line Simplification Algorithms for Vector Generalization," *The Cartographic Journal*, vol. 43, no. 1, pp. 27–44, Mar. 2006.