An evaluation of Lidar-based 2D SLAM techniques with an exploration mode

X Fan, Y Wang and Z Zhang
Shandong Provincial Key Laboratory of Automotive Electronics Technology, Institute of Automation, Qilu University of Technology (Shandong Academy of Sciences), Jinan, P.R.China
{fanxj, wangy, zhangzh}@sdas.org

Abstract. SLAM is a fundamental problem in robotic field and there have been many techniques on it. It is necessary to give an insight on weakness and strength of these techniques specific to the intended final application. This paper presents a study of three most common laser-based 2D SLAM techniques: Gmapping, KartoSLAM and Cartographer. Each technique was applied to construct maps combined with autonomous exploration. All the approaches have been evaluated and compared in terms of inaccuracy of constructed maps against the ground truth. In order to draw conclusions on the performance of the tested techniques, a metrics of average distance to the nearest neighbor (ADNN) was applied. Moreover, the computational load of each technique is examined.

1. Introduction
SLAM is a fundamental step towards fully autonomous robotic operation, where a robot acquires an environment map while simultaneously localizing itself relatively to this map. It is also a well investigated problem so far and there have been a large number of methods on it. These methods can be generally divided into two groups. One is filter-based methods and the other is graph-based methods (GraphSLAM).

The filter-based methods are inspired by Bayesian filtering theory. They consist of two main steps [1]: (1) prediction, where robot localization and map state are updated using the previous information about the system state and input control commands; (2) measurement update, where current sensor data are matched to the system state in order to make a new state prediction (called posterior). This kind of approach has various implementations. The earlier ones were based on Extended Kalman filters (EKF) [2], or on Particle Filters (PF), e.g., FastSLAM [3]. FastSLAM makes use of a modified PF to estimate the posterior. Each particle possesses N Kalman filters that estimate the N landmark locations. It was shown that its computational cost is lower than EKF-based approaches. Also, the approach deals with a large number of landmarks even with a small set of particles and the result still remains appropriate. An approach based on PF is proposed in [4]. More details of this work are discussed in Section 3-1.

GraphSLAMs cover some weaknesses of PFs and EKFs techniques [5]. These algorithms represent the map by means of a graph, which is composed by nodes and arcs. A node in the graph represents a pose of the robot along its trajectory and a set of sensor measurements. An arc represents a constraint between successive poses, which can be a motion event or a measurement event. The map is computed by finding the spatial configuration of the nodes which are consistent with constraints from the arcs.
This type of SLAM algorithm was first presented in [6]. However, its applicability in large scenarios is impracticable due to the optimization process used. The GraphSLAM presented by Thrun et al. [5] extended its behavior to large-scale urban environments by a reduction process, which removes map variables from the optimization process. In addition, Hess et al. [7] also developed a GraphSLAM, which is discussed in Section 3-3.

The SLAM problem can be solved with the use of different sensors. 2D lidar SLAM systems are currently presented in different packages like Gmapping [4], Hector SLAM [8], and Cartographer [7]. On the other hand, monocular and stereo cameras are good low-cost passive sensors to acquire information about an environment, which can effectively solve SLAM problem that referring to Visual SLAM [9]. The research presented here is the first step for our ultimate goal, which is to propose a SLAM technique for an unmanned vehicle (UV) within a specific area. The choice of suitable sensors plays a special role in the effective operation of UV due to the limited autonomous resource. Since the performance of visual SLAM systems strongly depends on computational resources, which are limited in the UV, this study focuses on the comparison on only 2D lidar SLAM systems.

2. Related work
Naturally, it is necessary to study weakness and strength of various SLAMs to serve as guidance to our later application. The comparison of existing SLAMs is not a new topic.

A comparison of five laser-based 2D SLAMs was conducted in [10]: Gmapping, KartoSLAM, Hector SLAM, CoreSLAM and LagoSLAM. Maps were constructed from the test data acquired in real world experiments and compared with the metrics based on the k-nearest neighbors. The results show more favorable outcomes from the last three.

The paper [1] describes a qualitative comparison of various SLAMs using different visual sensors (monocular and stereo cameras) and laser. 2D laser-based methods (Gmapping, Hector SLAM, Cartographer) and 7 visual based SLAMs were performed on a mobile platform in an indoor environment. The experimental results demonstrate encouraging results for lidar based Cartographer SLAM, Monocular ORB SLAM and Stereo RTAB Map methods.

The research [11] extends the comparative analysis of Gmapping, Hector SLAM and Cartographer, using a FARO laser tracker as the precise ground truth. The experiment used data from 2D lidar that was placed on an autonomous mobile robot. ADNN-based errors were calculated for maps provided by the three SLAMs relative to the ground truth.

However, the above works evaluated SLAMs only on quite simple scenarios and the robot was remotely teleoperated using a joystick or keyboard, which implies that the robot's trajectory is always fixed and correct during map construction. In our work, we extended the performance evaluation of various SLAMs into more complex environments, where a robot performs autonomous exploration.

The paper is organized as follows. Section 3 introduces SLAM algorithms to be compared. Section 4 describes the methodology of experiments, including metrics for evaluation, SLAM algorithm implementation, achieved results, and comparative analysis of maps built by different SLAMs. Finally, conclusions are drawn and further work is briefly suggested in Section 5.

3. 2D SLAM algorithms
In this section, a brief description of three SLAM techniques is conducted, namely, Gmapping, KartoSLAM and Cartographer.

3.1. Gmapping
Gmapping is a Rao-Blackwellized PF SLAM approach as described by [4]. Furthermore, it is the most widely used SLAM package for robot navigation. To obtain good results, the PF family of algorithms usually requires a high number of particles, which results in high computational complexity. Also, the depletion problem associated with the PF resampling process decreases the algorithm accuracy. An adaptive resampling technique has been developed in [4], which minimizes the particle depletion problem, since this process is only performed when needed. The authors also proposed a way to decrease the uncertainty of the robot's pose in the prediction step, which computes an accurate
distribution by taking into account not only the movement of the robotic platform, but also the most recent observations. As a consequence, the required number of particles is greatly decreased.

3.2. KartoSLAM
KartoSLAM is a GraphSLAM developed by SRI International’s Karto Robotics. It succeeds to obtain good performance in the real world, and is not easily affected by noise. KartoSLAM uses a highly-optimized and non-iterative Cholesky matrix decomposition for sparse linear systems as its solver [12]. In its ROS version, the Sparse Pose Adjustment (SPA) is responsible for both scan matching and loop-closure procedures [13]. For SLAMs, the larger the number of landmarks, the more amount of memory is required, especially when maintaining the map of a large-scale environment. However, it is extremely efficient in the particular case of KartoSLAM, since it only maintains a pose graph.

3.3. Cartographer
Cartographer provides a real-time mapping solution in 2D and 3D across multiple platforms and sensor configurations [7]. Since grid-based mapping becomes resource intensive as maps become large, Cartographer does not use particle filter algorithm. Cartographer solves the problem of error accumulation by pose estimation during long iterations. Laser scans are iteratively inserted into a submap at the best estimated position referred to as frames. Scan matching occurs at a recent submap, therefore it only depends on recent scans. After each submap is finished, all submaps and scans are automatically checked for loop closure. If they are close enough based on current pose estimates, a scan matcher tries to find the scan in the submap. The paper [7] describes the conversion process for scan points from a scan frame into a submap frame. Submaps are represented in form of probability grids. For each grid point, the corresponding pixel that contains all points closest to that grid point is defined. All grid points are updated with the appropriate probabilities. Scan matching in Cartographer is based on a Ceres-based scan matcher [14]. It is responsible for finding a scan pose that maximizes the probabilities at the scan points in the submap.

4. Experiments
4.1. System setup
Tests were conducted to study the behavior of these SLAMs in Stage\(^1\), which is a 2D robot simulator integrated with ROS. In the simulation experiments, the robot model was defined to simulate Turtlebot3 and the model of the range sensor was defined just like the one used in real world experiments, Rplidar A2\(^2\), which has a maximum range of about 10 meters. For all of these three SLAM methods, we use their ROS versions\(^3,4,5\). Note that the abstraction layer provided by ROS wrapper allows to use the same source code for both simulation and real-world experiments.

Stage simulations were performed using three different terrains: maze, plan and fr052b, which are shown in figure 1. maze enables the behavior analysis of the SLAMs in a scenario with few features. This is important to evaluate the robustness on lack of landmarks for each approach. fr052b enables the performance evaluation in a large and very complex scenario.

The simulations were performed in a PC equipped with an Intel Core i7-5500U and 8Gb of RAM, Ubuntu 16.04 installed with ROS Kinetic.

4.2. Metrics for evaluation
To evaluate the quality of the maps obtained, an analysis of the error between the generated map and the ground truth was conducted. A performance metric based on the k-nearest neighbor concept was used \((k=1)\), as introduced in [10].

\(^1\) http://www.ros.org/wiki/stage
\(^2\) http://www.slamtec.com/cn/Lidar/A2
\(^3\) http://wiki.ros.org/gmapping
\(^4\) http://www.ros.org/wiki/karto
\(^5\) http://wiki.ros.org/cartographer
where $d_i$ equals the distance between cell $i$ of the ground-truth map to its nearest occupied cell in the generated map, and $N$ is the total number of occupied cells.

4.3. Implementation and results

In the experiments, the simulated robot was guided by frontier exploration\(^6\) to go through the dataset. The figure 2 shows part of fr052b test data with the robot model in Stage environment, and the visualization of the lidar data in RViz, demonstrating 2D laser point cloud and the built map.

In the tests that were conducted, the output of each technique, as described previously, was a respectively generated 2D occupancy grid map. Occupancy grid is a map representation in 2D space, which is configured with a grid, each cell indicating a state in a given space location. A cell may take one of three states: occupied, free and unknown. For each algorithm, the resolution of the final map was set to 0.05 meters/pixel. In all experiments, the default parameter setting was used. For

\(^6\)http://www.ros.org/wiki/stage http://wiki.ros.org/frontier_exploration
Table 1. Error Estimation for Each Algorithm

|       | Cartographer | KartoSLAM | Gmapping |
|-------|--------------|-----------|----------|
| maze  | 2.3056       | 3.0242    | 2.6513   |
| plan  | 1.6110       | 1.7005    | 1.8172   |
| fr052b| 1.8536       | 1.9334    | 1.9397   |

Figure 3. Comparison of CPU/Memory load of three SLAM methods. Notice that the CPU/Memory loads are normalized according to Cartographer.

example, the number of particles for Gmapping was 30.

The constructed maps were compared to the ground truth to confirm their accuracy, which could be seen in figure 4 - figure 6. The comparison of ADNN-based errors for maps provided by Gmapping, Cartographer and KartoSLAM relative to the ground truth is presented in Table 1. Beyond the error analysis conducted, an evaluation of the computational cost using each technique was carried out. A comparison of the CPU/Memory load running each algorithm is presented in figure 3. As shown in figure 3, Cartographer presented the highest percentages of CPU/Memory usage. Moreover, this load analysis reveals that KartoSLAM is the most computation-efficient method in terms of resources required.

4.4. Discussion

The experimental results show that all three SLAMs construct fairly reliable maps. These algorithms are robust enough to the different type of terrains. In all scenarios Cartographer constructs maps with the smallest error relative to the ground truth. Since Cartographer uses global map optimization cycle and local probabilistic map updates, it makes this system more robust to environmental changes. Gmapping maps are not very far from Cartographer maps' quality. It demonstrates reasonably good outcomes in 2D map construction even without loop closure. The reason is that Gmapping apply odometry for localization rectification and map correction.

Both KartoSLAM and Cartographer are graph-based SLAM approaches, and their results were a little different. Cartographer provided accurate maps with high computational cost, while KartoSLAM generated less accurate maps with low computational cost. This can be explained by the fact that Cartographer is a complex system and its flexibility in term of configuration with a wide range of parameters. However, compared to KartoSLAM, parameter tuning to a particular environment is difficult for Cartographer since a good understanding of its inner working is required.

Generally, the error on maze is much larger than the other two terrains, which can be seen in Table 1. That means that mapping the maze terrain is equally challenging for all three SLAMs. We suppose that this is due to the terrain's lack of characteristics which can easily cause false matches for pose estimation and loop closure optimization. As a contrast, although being a complex environment with a
Figure 4. Map comparison constructed from SLAMs (green) and ground truth (magenta) for maze

Figure 5. Map comparison constructed from SLAMs (green) and ground truth (magenta) for plan with local enlargement to see more details

Figure 6. Map comparison constructed from SLAMs (green) and ground truth (magenta) for fr052b with local enlargement to see more details
high density of obstacles, the fr052b terrain contains many distinguishable features, which makes it easier for the SLAMs to achieve a high optimization performance.

5. Conclusions and future work
This work provides an insight of three representative SLAMs: Gmapping, KartoSLAM and Cartographer. The results of execution for SLAMs were investigated and compared with the use of metrics. A discussion on the weaknesses and strengths of each solution has also been done. The main results of our investigations: Cartographer provides accurate solutions for map building, while KartoSLAM is most computationally efficient. However, the comparison of different SLAMs is really difficult since the performance of a SLAM technique is greatly influenced by its parameter settings, and the best parameter setting may depend on a particular environment. Nevertheless, this paper still sheds some light on the choice of an appropriate SLAM approach for the intended application. Based on the results observed in this article, further evaluation will be conducted on different parameter settings and sensor data with noise. Evaluation with a physical robot in various real-world scenarios will also be considered in future work.

References
[1] Filippenko M and Afanasyev I 2018 Comparison of various slam systems for mobile robot in an indoor environment Proc. 9th IEEE Int. Conf. on Intelligent Systems pp 400-407
[2] Bailey T, Nieto J, Guivant J, et al. 2006 Consistency of the ekf-slam algorithm Proc. IEEE Int. Conf. on Intelligent Robots and Systems (IROS) pp 3562-3568
[3] Montemerlo M, Thrun S, Koller D, et al. 2002 Fastslam: A factored solution to the simultaneous localization and mapping problem Proc. AAAI National Conf. on Artificial Intelligence pp 593-598
[4] Grisetti G, Stachniss C and Burgard W 2007 Improved techniques for grid mapping with Rao-Blackwellized particle filters IEEE Trans. on Robotics 23 no 1 34-46
[5] Thrun S and Montemerlo M 2005 The GraphSLAM algorithm with applications to large-scale mapping of urban structures Int. J. on Robotics Research 25 no 5 pp 403-429
[6] Lu F and Milios E 1997 Globally consistent range scan alignment for environment mapping Autonomous Robots 4 no 4 333-349
[7] Hess W, Kohler D, Rapp H and Andor D 2016 Real-time loop closure in 2D lidar slam Proc. 2016 IEEE Int. Conf. on Robotics and Automation (ICRA) pp 1271-1278
[8] Kohlbrecher S, Stryk O V, Meyer J and Klingauf U 2011 A flexible and scalable SLAM system with full 3D motion estimation Proc. 9th IEEE Int. Symp. on Safety, Security, and Rescue Robotics (SSRR) pp 155-160
[9] Taketomi T, Uchiyama H and Ikeda S 2017 Visual SLAM algorithms: a survey from 2010 to 2016 IPSJ Trans. on Computer Vision and Applications 9 no 1 16
[10] Santos J M, Portugal D and Rocha R P 2013 An evaluation of 2D slam techniques available in robot operating system Proc. Int. Symp. on Safety, Security, and Rescue Robotics (SSRR) pp 1-6
[11] Yagfarov R, Ivanou M and I. Afanasyev 2018 Map comparison of Lidar-based 2D SLAM algorithms using precise ground truth Proc. Int. Conf. on Control, Automation, Robotics and Vision (ICARCV)
[12] Vincent R, Limketkai B and Eriksen M 2010 Comparison of indoor robot localization techniques in the absence of GPS Proc. SPIE: Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XV
[13] Konolige K, Grisetti G, Kummerle R, et al. 2010 Efficient sparse pose adjustment for 2D mapping Proc. Int. Conf. on Intelligent Robots and Systems (IROS)
[14] Agarwal S, Mierle K, et al. 2012 Ceres solver

Acknowledgments
This work was supported by Shandong Provincial Key R&D Program under Grant No. 2017GGX50107.