Comparison of SLAM algorithms on omnidirectional four wheel mobile robot

Slaven Petković, Lazar Milić, Milutin Nikolić, Dragiša Mišković, and Mirko Raković

***Abstract*—In this article we present a comparative analysis of various SLAM algorithms. We compared robot trajectories computed by three ROS-based SLAM algorithms to a reference trajectory obtained from Vicon motion capture system. For data acquisition purposes we used mobile robot with four omnidirectional (*mecanum*) wheels. Our mobile platform was equiped with following sensors: 3D lidar, a RGB-D camera and motor encoders. Experiments were conducted indoor in an office environment. Acquired dataset was used as an input data for all algorithms that we tested. Following algorithms have been taken into account: Livox Mapping, RTAB-Map and Cartographer.**

***Index Terms*—Simultaneous Localization and Mapping, SLAM, ROS, Mobile robot**

I. Introduction

ONE of the most important tasks for autonomous mobile robots is to create a consistent map of unknown environment and to determine its location inside that map. This problem is known as Simultaneous Localization and Mapping (SLAM) and it is considered a difficult problem, because robot needs good estimate of its location in order to create a valid map, but at the same time robot needs consistent map to determine its location.

Problem was first defined back in 1986 [1]. There are two main approaches for solving SLAM problem: probabilistic approach and non-probabilistic approach. The probability methods are based on Bayesian estimation method. Many methods were developed, such as SLAM based on Kalman Filter, Extended Kalman Filter [2], Particle Filter [3], Rao-Blackwellized Particle filter [4], etc.

Nowadays there are many different methods for solving SLAM problem. In this article we focus on SLAM algorithms that are available as an *open-source* ROS (Robot Operating System) [5] package and that have support for 3D lidar and/or RGB-D camera. For purpose of acquiring data, mobile robot with four omnidirectional wheels was used and all experiments were done indoor.

II. Related Work

In this section, we provide an insight into papers that deal with the comparison of SLAM algorithms. Paper [6] provides benchmark for two popular SLAM algorithms: RTAB-Map and RGBD SLAM. Paper [7] shows comparison results of three different mapping approaches. Comparison between three modern VSLAM approaches : RTAB-Map, ORB-SLAM3 and OpenVSLAM is presented in [8]. In article [9] authors show comparison of trajectories computed by various ROS-based SLAM systems in office environment. This article presents a different approach to comparing SLAM algorithms.

S. Petković, L. Milić, M. Nikolić, M. Raković are with the Faculty of Technical Science, University of Novi Sad, (e-mail: rakovicm@uns.ac.rs). D. Mišković is with the Institute for Artificial Intelligence Research and Development of Serbia, Novi Sad, Serbia

III. Compared SLAM algorithms

A. RTAB-Map

RTAB-Map (Real-Time Appearance-Based Mapping) [10] is an open-source library released in 2013 and is still in the development. It is a graph-based SLAM approach with memory management available as *rtabmap_ros* ROS package. RTAB-Map is very flexible SLAM approach because it supports wide range of input sensors such as odometry from any source, RGB-D or stereo cameras and optionally 2D/3D lidar data. Package consists of main *rtabmap_ros/rtabmap* node, two nodes for visual odometry (*stereo_odometry* and *rgbd_odometry*) and lidar odometry node (*icp_odometry*). As RTAB-Map supports inputs from multiple different sensors, data obtained from these sensors needs to be synchronized in order to create a valid map. There are two types of synchronization in RTAB-Map: exact synchronization and approximate synchronization. RTAB-Map can be configured with large number of parameters whose description can be found in [11].

B. Cartographer

Google's Cartographer is lidar graph-based SLAM approach. Cartographer is used for indoor mapping and it supports 2D and 3D lidar data as well as odometry data. In the process of creating map Cartographer uses lidar scans or point cloud to create sub-maps and when loop-closure is detected it runs pose optimization in order to minimize error. First part of this process is managed by Cartographer's local SLAM subsystem (also called frontend), while optimization is done using Cartographer's global SLAM subsystem (also called backend). Detailed description of Cartographer can be found in [12]. On official web page [13] one can find algorithm explanation, description of all parameters and tips for parameter adjustment.

C. Livox Mapping

Livox mapping is a SLAM algorithm developed for creating a map using only Livox lidars. It is available as *livox_mapping* ROS package which supports multiple Livox 3D lidars. Algorithm also uses odometry data, for example wheel odometry. In the development of this package, authors reference to [14]-[16].

 ISBN 978-86-7466-930-3

IV. System setup and Data set acquisition

In this section, we will present the hardware and software used to conduct the experiment. Experiment consists of three main modules: (1) Vicon system for motion capture, (2) a mobile robot for acquiring sensor data and (3) a notebook for running and comparing SLAM algorithms.

A. Vicon system

Vicon system is a highly accurate motion capture system. This system can be used in multiple applications, like robot tracking, human motion capture, etc. In our experiment, the main task of this system was to capture robot's motion and compute trajectory of the robot. This trajectory was used as a reference trajectory to compare and evaluate different SLAM algorithms. Our indoor testing environment with Vicon system and mobile robot is shown in Figure 1. In this experiment Vicon system consisted of (1) seven Vicon MX T-20S cameras, (2) Vicon MX Giganet and (3) desktop computer with Vicon Nexus software used for data processing. Calibration of cameras is done using Vicon Active Wand calibration device and Vicon Nexus software. In this experiment, the operating frequency of Vicon cameras was set to 200 Hz.

Fig. 1: Experimental setup with Vicon cameras and mobile robot

In order to record the movement of the mobile robot, it was necessary to place reflective markers on the mobile robot, which can be tracked by motion capture system. Position and orientation of these markers defines coordinate system of tracking object created in Vicon Nexus software. The orientation of the coordinate system depends on the order in which the markers are selected in Vicon Nexus software. Figure 2 shows three markers placed on the mobile robot. To make it easier to compare data later, the position and orientation of the markers on the mobile robot is chosen to match *base_link* coordinate system defined in robots URDF model (*Unified Robot Description Format*).

B. Mobile robot

Mobile robot used in this experiment was a four wheel omnidirectional mobile robot (Figure 3). This construction with

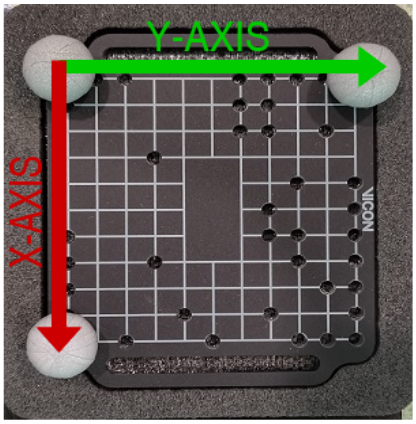


Fig. 2: Reflective markers used to track robot motion by Vicon system. Position of markers determines position and orientation of coordinate system of tracked object

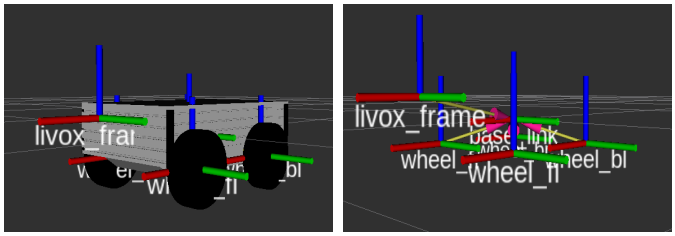
omnidirectional wheels (*mecanum wheels*) enables it to move in multiple directions and change direction rapidly. Mecanum wheel consists of k rollers made of rubber positioned at 45 degrees angle offset from the wheel rotation around its circumference. Paper [17] describes geometry and kinematics of mecanum wheels. Main problem with this type of mobile robots is that robot's wheels slip, and odometry based on encoders from wheels can not be considered as reliable.

Fig. 3: Omnidirectional four-wheeled mobile robot

Running SLAM algorithms on ROS requires a file that describes robot's physical state to ROS. Creating robot model was done using URDF. URDF is an XML file format used to describe all components of a robot. Figure 4 presents model of our mobile robot and positions of coordinate systems presented in RViz.

Robot's software runs on Ubuntu 18.04 operating system and ROS Melodic. Following sensors were mounted on mobile robot in this experiment: 3D lidar, RGB-D camera and wheel

 ISBN 978-86-7466-930-3



(a) Robot visualization in RViz (b) Robot coordinate systems

Fig. 4: Mobile robot model in RViz

encoders. Detailed information about robot configuration is presented in Table I.

Hardware	
CPU	AMD Ryzen 7 2700u
GPU	AMD Radeon Vega 10 Graphics
RAM	16 GB DDR4
Sensors	
Lidar	Livox MID-70
Camera	Intel RealSense D435i
Software	
Operating System	Ubuntu 18.04 Bionic Beaver
ROS	ROS Melodic Morenia

TABLE I: Hardware and software specifications of mobile robot

C. Data processing

In order to get the most valid comparison results, the goal was to record data from robot sensors to one file, so every SLAM algorithm tested in this experiment could use the same input data. For this reason we run offline all the SLAM algorithms that are evaluated. Detailed information about computer configuration that was executing SLAM algorithms is shown in II.

Hardware	
CPU	Intel Core i7-10750H
GPU	NVIDIA RTX 2060 6GB
RAM	16 GB DDR4 3200 MHz
Software	
Operating System	Ubuntu 18.04 Bionic Beaver
ROS	ROS Melodic Morenia

TABLE II: Hardware and software specifications of computer used for data processing

V. Metrics and data comparison

In this section, we give an insight into how the Vicon system and ROS represents robot pose. We also present metrics for analyzing performance of SLAM algorithms by comparing trajectories and robot poses generated by each SLAM algorithm to a ground truth trajectory and robot poses computed by Vicon system. We did not compare generated maps. The idea was to compare predicted robot pose to an actual robot pose in every moment in time. As robot pose in plane consists of three components, x and y coordinate and rotation around z-axis, we compared each component of robot motion separately. Then the error for all three components of robot motion was calculated.

A. Data representation

1) Vicon system: Vicon Nexus software is able to export data in number of different formats. We have chosen to work with *.csv* file. Table III shows an example of *.csv* file generated by Vicon Nexus software. Values *RX*, *RY* and *RZ* represent rotation of robot's base coordinate frame around x, y and z axis respectively, while *TX*, *TY* and *TZ* represent position of robot's coordinate frame in x, y and z direction respectively with respect to the reference frame. All values are expressed in relation to the origin of the coordinate system defined by the position of the Vicon Active Wand during calibration. Frame column was used to calculate timestamp.

Frame	Sub Frame	RX	RY	RZ	TX	TY	TZ
number	number	[deg]	[deg]	[deg]	[mm]	[mm]	[mm]

TABLE III

2) SLAM algorithms: The ability of Robot Operating System to record data from any available topic was used for generating data set. Recorded data was saved to *.bag* file. ROS represents robot pose as *geometry_msgs/Pose* ROS message and it also enables us to export in *.csv* file. Unlike Vicon system, ROS uses Quaternion to display robot rotation.

B. Data comparison

Comparing two trajectories requires them to consist of same number of points (robot poses) and to be synchronized in time. In our case, neither of these two conditions was met.

As previously said, we collected data from sensors mounted on mobile robot and run algorithms offline using collected data to generate test data set. Reference trajectory was computed by Vicon system. These systems are not synchronized in time.

First problem was how to equalize the beginnings of two compared trajectories in time. In order to do that, we decided to find moment in time in which robot made a movement for 1 mm in x or y direction.

Second problem was that every ROS algorithm publishes messages about robot pose at different rate and Vicon system frequency was 10 to 20 times greater than frequency at which ROS algorithms publish messages. In order to deal with this problem, there were two possibilities: to reduce the size of reference data set or to expand test data set. We opted for the latter and decided to do an interpolation over data generated by SLAM algorithms.

C. Metrics

Interpolation was done by time for every component of robot motion: $x(t)$, $y(t)$ and $\theta(t)$. Figure 5 shows example of interpolation in case of function $x(t)$.

As previously sad, SLAM algorithms publish messages about robot pose less frequently than Vicon system. In this paper interpolation was used to calculate new value in test data set for every moment in time of reference data set. Let's observe point (t_i, x_i) from the reference data set. In order to calculate new value we decided to find two points in the test data set which were closest in time to the observed point. On figure 5 points A(t_j, x_j) and B(t_{j+1}, x_{j+1}) represent two closest points. Equation of line connecting points A and B was calculated. Inserting value of t_i into new equation, interpolated value x_{interp} is calculated.

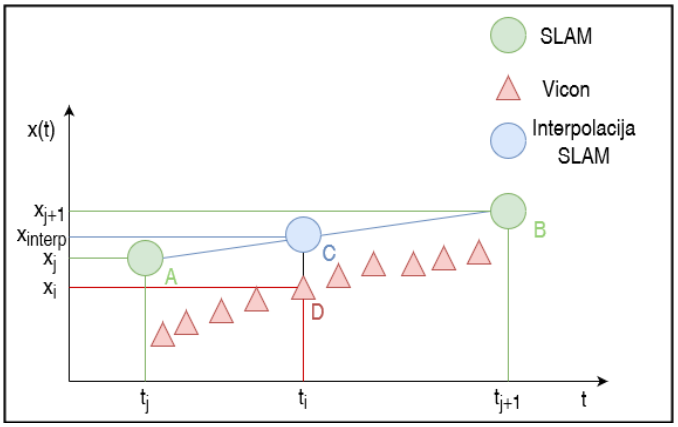


Fig. 5: The figure illustrates an example of interpolation used to generate new data in test data set

Data processing and comparison was done using the Python 3 programming language and the *NumPy* library, as one of the best open-source tools for working with data and numerical calculations.

VI. Results and Analysis

In this section we present results of our experiment. Results are presented graphically and numerically. For visualizing results, we used Python programming language and *Matplotlib* library. Results are presented in four graphs for each tested SLAM algorithm. First figure contains estimated trajectory (shown in red) computed by SLAM algorithm and reference trajectory computed by Vicon system (shown in green). Other three figures show comparison of *x(t)*, *y(t)* and $\theta(t)$ functions respectively. Table IV presents numerical results of our experiments. Error is represented as RMSE (*Root Mean Square Error*), standard deviation, mean absolute error and maximum absolute error.

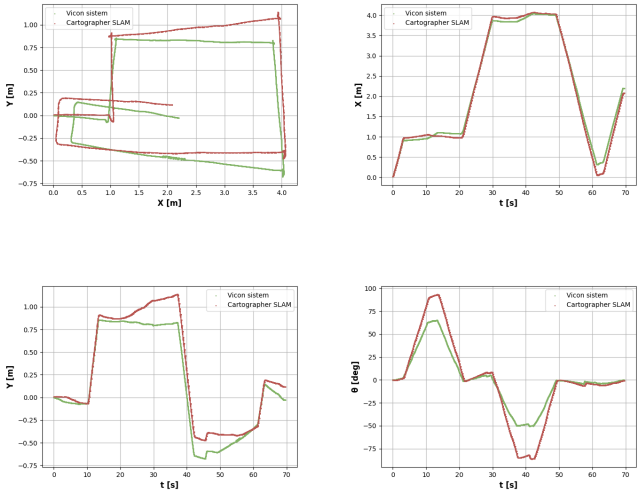


Fig. 6: Trajectory recovered from Cartographer SLAM (red) vs Vicon system reference trajectory (green)

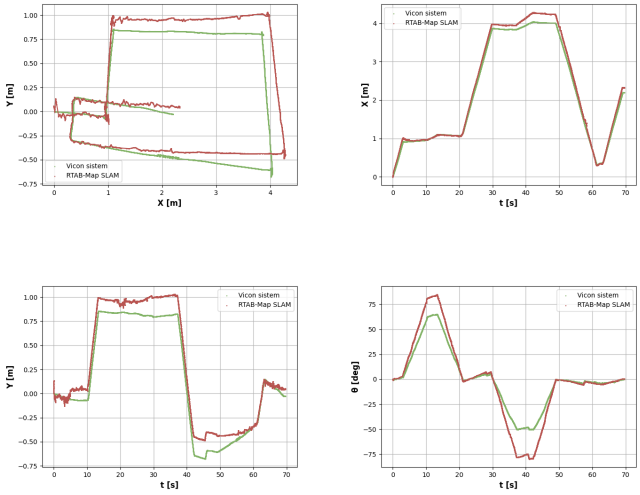


Fig. 7: Trajectory recovered from RTAB-Map (red) vs Vicon system reference trajectory (green)

VII. Conclusion

In this paper, we have presented a method for analyzing performance of ROS-based SLAM algorithms that compares trajectories computed by SLAM algorithm to a reference trajectory obtained from motion capture system. We proposed a metric for calculating error in estimated trajectory. This approach allows us to compare algorithms that use different sensor information.

Since omnidirectional wheels are slipping during the motion of robot, the localisation of the robot has to be based on sensory system that is not measuring motion of wheels directly. Therefore, the robot is build with 3D Lidar and RGB-D camera as sensors for localisation of the robot and mapping of the environment. To better determine what algorithm is the best for sensory system that our robot has, we evaluated three

 ISBN 978-86-7466-930-3

Benchmarking results

SLAM Algorithm	RMSE [cm]	Standard deviation [cm]	Mean absolute error [cm]	Maximum Absolute Error [cm]
Comparison of error in x direction				
RTAB-Map	1.2518514	7.6750762	9.8897118	27.0362502
RTAB-MAP Visual odometry	1.5254185	7.9799980	13.0003765	29.3477365
Google Cartographer	1.1652417	7.1982585	9.1631812	33.8634761
Livox Mapping	1.6213981	9.1069006	13.4148259	38.366215
Comparison of error in y direction				
RTAB-Map	1.3296018	6.5810257	11.5531029	23.8609514
RTAB-MAP Visual odometry	2.3063390	8.1394842	13.0003765	35.0320307
Google Cartographer	1.6002339	10.2693004	21.5793597	37.0653832
Livox Mapping	1.1607506	6.6658583	9.5026594	24.9565594
Comparison of error in rotation				
	RMSE [°]	Standard deviation [°]	Mean absolute error [°]	Maximum Absolute Error [°]
RTAB-Map	12.1708190	9.2212436	7.9433937	29.8499295
RTAB-MAP Visual odometry	12.4845912	9.2257357	8.4113505	29.9491947
Google Cartographer	16.1672031	12.0188858	10.8131790	37.3971806
Livox Mapping	19.1045268	13.6542612	13.3620393	46.4193679

TABLE IV: Benchmarking results

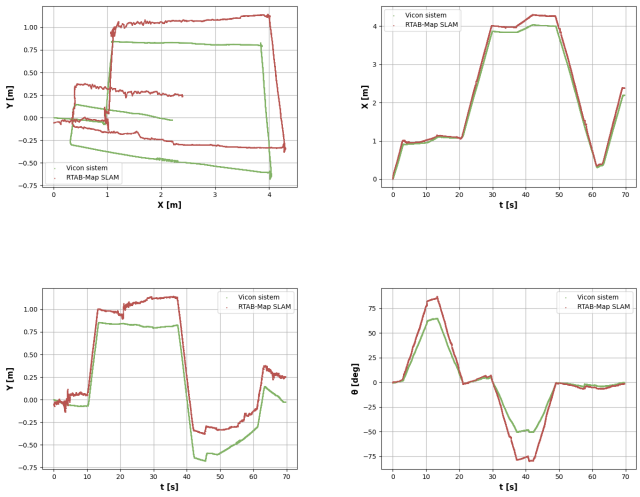


Fig. 8: Trajectory recovered from RTAB-Map visual odometry (red) vs Vicon system reference trajectory (green)

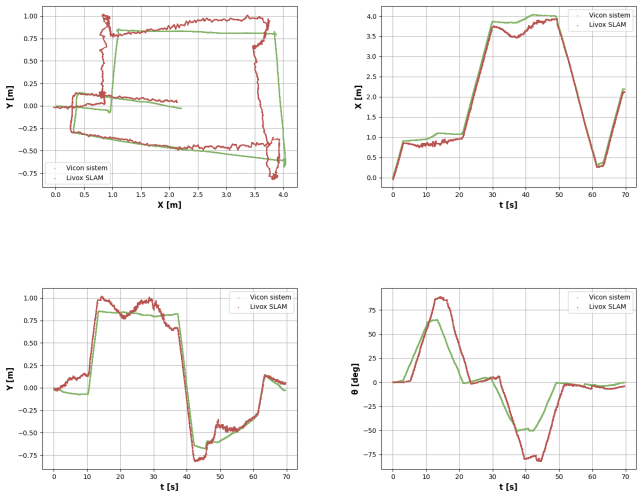


Fig. 9: Trajectory recovered from Livox Mapping (red) vs Vicon system reference trajectory (green)

up-to-date algorithms. Based on benchmark results we can see different algorithms provide different best performance. In most cases, RTAB Map has shown the best results compared to other algorithms.

REFERENCES

[1] Durrant-Whyte, Hugh, and Tim Bailey. "Simultaneous localization and mapping: part I." IEEE robotics & automation magazine 13.2 (2006): 99-110.

[2] Bailey, Tim, et al. "Consistency of the EKF-SLAM algorithm." 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2006.

[3] Montemerlo, Michael, et al. "FastSLAM: A factored solution to the simultaneous localization and mapping problem." Aaai/iaai 593598 (2002).

[4] Grisetti, Giorgio, Cyrill Stachniss, and Wolfram Burgard. "Improved techniques for grid mapping with rao-blackwellized particle filters." IEEE transactions on Robotics 23.1 (2007): 34-46.

[5] Quigley, Morgan, et al. "ROS: an open-source Robot Operating System." ICRA workshop on open source software. Vol. 3. No. 3.2. 2009.

[6] Kasar, Amey. "Benchmarking and comparing popular visual SLAM algorithms." arXiv preprint arXiv:1811.09895 (2018).

[7] Burgard, Wolfram, et al. "A comparison of SLAM algorithms based

on a graph of relations." 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2009.
[8] Merzlyakov A, Macenski S. A Comparison of Modern General-Purpose Visual SLAM Approaches. arXiv preprint arXiv:2107.07589. 2021 Jul 15.
[9] Filipenko, Maksim, and Ilya Afanasyev. "Comparison of various slam systems for mobile robot in an indoor environment." 2018 International Conference on Intelligent Systems (IS). IEEE, 2018.
[10] Labbe, Mathieu, and François Michaud. "RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation." Journal of Field Robotics 36.2 (2019): 416-446.
[11] https://github.com/introlab/rtabmap/blob/master/corelib/include/rtabmap/core/Parameters.h
[12] Hess, Wolfgang, et al. "Real-time loop closure in 2D LIDAR SLAM." 2016 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2016.
[13] https://google-cartographer-ros.readthedocs.io/en/latest/index.html
[14] Zhang, Ji, and Sanjiv Singh. "LOAM: Lidar Odometry and Mapping in Real-time." Robotics: Science and Systems. Vol. 2. No. 9. 2014.
[15] Zhang, Ji, and Sanjiv Singh. "Enabling aggressive motion estimation at low-drift and accurate mapping in real-time." 2017 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2017.
[16] Zhang, Ji, Michael Kaess, and Sanjiv Singh. "On degeneracy of optimization-based state estimation problems." 2016 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2016.
[17] Gfrerrer, Anton. "Geometry and kinematics of the Mecanum wheel." Computer Aided Geometric Design 25.9 (2008): 784-791.

 ISBN 978-86-7466-930-3