Analysis of computational need of 2D-SLAM Algorithms for Unmanned Ground Vehicle

Thrilochan Sharma P*
(Intern)
KPIT Technologies Ltd., Bangalore, India

Sankalprajan P*
(Intern)
KPIT Technologies Ltd., Bangalore, India

Ashish Joel Muppidi
KPIT Technologies Ltd., Bangalore, India

Dr. Prithvi Sekhar Pagala
KPIT Tech Ltd., Bangalore, India

(MTech, Automation and Robotics Engineering, University of Petroleum and Energy Studies, Dehradun, India)*
(MTech, Mechatronics Engineering, Vellore Institute of Technology, Chennai, India)*

***Abstract*—The need for advancement in robotic car or driverless car or autonomous vehicle is vital, as safety concerns and advancements in automotive technologies raise have led to market penetration and acceptance of driverless cars. Simultaneous Localization and Mapping (SLAM) is one of the important features of autonomous transportation. SLAM helps us solve the problem of a vehicle maneuver in unknown locations. It assists the vehicle to understand the surroundings (mapping) and its current position in it (localization). SLAM algorithms can be implemented in autonomous vehicle application using Robot Operating System (ROS). In this paper, four of the commonly used SLAM algorithms, i.e., G-mapping, Hector SLAM, Karto SLAM, and RTAB are implemented on an autonomous vehicle for computational efficiency. Comparison study of computational resource utilization of all these algorithms is done in both simulations and Real-time implementation.**

Keywords—SLAM, Robotic Car, ROS, Gmapping, Hector SLAM, Karto SLAM, RTAB

I. INTRODUCTION

The field of robotics encompasses a wide range of technologies in which computer intelligence is embedded in physical machines, creating systems whose capabilities go far beyond the core components. Such robot systems are then able to perform tasks that cannot be achieved with conventional machines or even with people who work with conventional tools. The ability of a machine to move on its own, i.e. "autonomously", is one that introduced an enormous range of applications. Robotic vehicles are machines that move "autonomously" on the ground, in the air, undersea, or in space. Such vehicles are "unmanned," in the sense that no humans are on board. In general, these vehicles move by themselves, under their own power, with sensors and computational resources onboard to guide their motion. These autonomous vehicles are becoming a new piece of infrastructure. Wide range of applications in logistics and transportation have given autonomous driving technology a great push. Autonomous driving is an area in the automobile sector that has received increased focus in the past two decades. It raised the need for multidomain engineering involving automotive makers, electronic makers, and IT service providers in this technology. Academic research has contributed significantly in producing different prototype designs in different levels of automation.

Levels of autonomy in vehicles is defined by the Society of Automotive Engineers (SAE)[1]. These levels are from Level-0 to Level-5 of autonomy.

- Level 0 – No Driving Automation. The entire Dynamic Driving Tasks (DDT) are under the driver's control. Conventional automobiles are an ideal example of this level of autonomy, e.g. 1976 Chevrolet Chevette.
- Level 1 – Driver Assistance. In this level, either lateral or longitudinal motion control subtask is automated. The remainder of the DDT is performed by the driver, e.g. 2011 Jeep Cherokee, 2014 Chevrolet Impala.
- Level 2 – Partial Driving Automation. It is like Level 1 except that it is characterized by both longitudinal and lateral vehicular motion control subtasks of the DDT. The driver is expected to perform object and event detection along with the response subtasks supervising the Advanced Driver Assistance System (ADAS), e.g. GM's Super Cruise, Tesla's Autopilot.
- Level 3 – Conditional Driving Automation. Environment detection capabilities are embedded in this level. The vehicle can make informed decisions like accelerating past a slow-moving vehicle, etc. but this level still requires a manual override. The driver should be ready to take control to perform a task if the system is unable to execute it e.g. Audi A8 2019.
- Level 4 – High Driving Automation. The vehicle can perform all the required operations in certain circumstances with full control. Geofencing is required. Human override is an available option.
- Level 5 – Full Driving Automation. All driving tasks are performed by the vehicle in all scenarios and circumstances without any human involvement and interaction. The vehicle will be free from geofencing and will be able to go anywhere performing all the tasks that an experienced human driver could do. Fully automated cars are under testing in various R&Ds around the world.

Level 2 features like ADAS assist a driver in driving tasks which increase comfort and safety. Level 3 autonomous features in cars like Adaptive Cruise Control (ACC) give confidence for complete automation of vehicles and highways. These help in reduction of fuel consumption,

harmful emissions and improves traffic flow profile [2]. Both ACC and ADAS are the immediate implementations possible in the current automobile industry. Level 5 autonomous driving technology will lead to a safe and an eco-friendly future. Different automation companies have roadmap towards level-5 autonomy in automobiles.

Automation of vehicles will lead to a transportation system with safety as a paramount feature. Autonomous vehicles' potential to reduce injuries and save lives is one tragic and critical fact as Statistics indicate that 94% of serious crashes are due to human errors while autonomous vehicles possess the potential to reduce the loss [3]. Automation of vehicles will lead to a decrease in human intervention, therefore removing human error factor out of the equation protecting lives. Various Economic and Societal cost factors such as vehicle crash losses, workplace productivity, loss of life and decreased life quality due to injuries will be reduced increasing the efficiency and convenience. Vehicular congestion will be reduced providing a smooth flow of traffic which leads to a reduction of fuel usage and vehicular emissions. Automated vehicles save as much as 50 minutes each day which is previously dedicated to driving [3]. As automation of vehicles introduces new technology in everyday life, it provides many employment opportunities. Semi-autonomous cars which come under Level 2 to Level 3 are available in the market and Level 4 autonomy vehicles are most likely to be introduced in the coming future.

In addition of hardware issues like actuators and sensor fusion, unidentical sensor combinations, etc. autonomous vehicles must address software issues. Data-driven software development can help autonomous vehicles achieve the goals of safety and efficiency. Digital twin concept is emerging which targets an efficient allocation of resources and a better experience for a vehicle owner [4]. In autonomous vehicles, such data-driven approach can be implemented utilizing the sensor data from the vehicle to run simulations simultaneously monitoring the vehicle's performance to achieve better efficiency. Many automobile industries have employed this method to monitor the performance of specific components of their vehicles to ensure the safety of the customers.

But for a vehicle to drive autonomously, it must perform three things simultaneously. They are, understanding the environment or mapping, localization and navigation. Mapping is the vehicle getting to know its surroundings and saving it with the help of inputs obtained from the sensors. Localization means the vehicle knowing its position in the map that is built from the sensor data. Navigation refers to the movement of the vehicle to reach a given target point or destination. Sensors and actuators play a vital role in all these aspects.

The concept of Simultaneous Localization and Mapping (SLAM) is a foundational feature that can help a vehicle to drive autonomously. SLAM lets the vehicle record its environment with the help of sensors mounted on different sections of the vehicle while simultaneously localizing itself relative to the map it generates of the environment around it. Implementation of SLAM enables a vehicle to avoid collisions with obstacles which include other vehicles, people and other objects. SLAM algorithms are of many types and some of the most used and open-source implementations are available for Visual-SLAM, Red Green Blue Depth (RGBD)-SLAM, Extended Kalman filter (EKF)-SLAM, Graph-SLAM, etc. RGBD-SLAM is a SLAM for RGB-D cameras. It can be used to generate high accuracy 3D point clouds or OctoMap [5]. EKF SLAM is a class of algorithms that utilizes the EKF for simultaneous localization and mapping. Typically, EKF SLAM algorithms are feature-based and use the maximum likelihood algorithm for data association. Graph-SLAM is a Simultaneous localization and mapping algorithm that uses sparse information matrices produced by generating a factor graph of observation interdependencies [6].

Although there are various SLAM algorithms available, in this study G-mapping, Hector SLAM, Karto SLAM and the 2D projection of RTAB (Real-Time Appearance Based mapping), a 3D SLAM algorithm, are implemented and the results are analyzed.

G-mapping is one of the widely used and well-known SLAM algorithms. Initially, Rao-Blackwellized Particle Filter (RBPF) is introduced to solve SLAM problems. RBPF is complex and based on number of particles. Accuracy of map generation is affected. Grisetti [7] has introduced G-mapping which utilizes a filter with which each particle's pose and position are recorded. G-mapping considers both movement of the vehicle as well as the most recent observations to compute the accurate distribution of the particles.

Hector SLAM is based on the Gauss-Newton approach. This SLAM depends on the scan-match technique to estimate the pose of the vehicle. Kohlbrecher [8] have proposed this SLAM. Unlike other 2D SLAMs, Hector SLAM doesn't need wheel odometry data. The modern Light Detection and Ranging (LiDAR)'s high accuracy and update rates can be leveraged for accurate and fast pose estimations with scan matching.

Karto SLAM is introduced by SRI International. It is a graph-based SLAM algorithm. A graph-based SLAM represents a map employing nodes that store features and pose. The constraints between successive poses are the edges in the graph. A sparse (pose) graph is generated by linearizing the above constraints. The solver used in this is the Cholesky Matrix Decomposition [9]. Sparse Pose Adjustment (SPA) [10] is used for both loop closure and scan matching (pose estimation).

RTAB was introduced as an appearance-based loop closure detection approach. It was then developed into SLAM on various platforms. Labbe and Michaud [11] have extended these capabilities of RTAB to compare various LiDAR and Visual SLAM algorithms for autonomous navigation. It can be used for the visual SLAM approach or LiDAR-based SLAM approach or both, as RTAB is independent of the odometry method employed. This makes it useful in either visual-based SLAM or LiDAR-based SLAM or the mix of both.

Robot Operating System is used to perform the comparison study of the resource's utilization by these SLAM algorithms. ROS is not an operating system in the traditional sense of process management and scheduling. ROS is a middleware that can be used to implement SLAM in autonomous vehicles both in simulation and hardware [12]. ROS is an open-source robotics framework with a

distributed structure and tools which enable the SLAM algorithms to be implemented and tested in real-time and simulations of the vehicle. ROS provides services designed for a heterogeneous computer cluster such as hardware abstraction, low-level device control, implementation of commonly used functionality, message-passing between processes, and package management.

SLAM algorithms enable us to include additional features as demonstrated by [13], in which an EKF based SLAM is proposed. Aided by position and velocity data, in-fusion with Inertial Navigation System (INS) and Global Navigation Satellite System (GNSS), using grid-based SLAM, acts like virtual odometry and virtual compass. Scaling factor is a predominant element as many of the studies are performed using miniature vehicles in scaled-down scenarios. A co-visibility graph-based representation of the map is also introduced [14]. This method enables the SLAM to be independent of the complexity of the size of the map. It helps in the efficient selection of a locally relevant portion and helps in optimizing it in such a way that it seems like performing optimization of the full trajectory. Along with mapping, localization, and navigation, vehicle control can be managed using SLAM algorithms [15]. Graph SLAM can be applied in mapping unknown regions [16], where a part of a forest is mapped using Velodyne LiDAR, stereo camera, Inertial Measurement Unit (IMU) and GPS.

Comparing different SLAM algorithms helps in understanding the nature of each SLAM and its response to given conditions and resources. Various researchers have performed such comparison studies and put forth their conclusions which helped future practices to choose appropriate SLAM for their application. Accuracy of the maps generated by these SLAMs is one of the leading factors of comparison [17] as it directly affects the movement of the vehicle. The environment used for testing these algorithms also plays a major role in choosing the SLAM for an application as the factor affecting the performance varies from indoor and outdoor. That is the main reason for many studies [18],[19], [20] to be conducted in static indoor environments. Sensors play an important role in the execution of SLAM algorithms. A comparison of sensor performances for the same approach can help us understand the requirements of the application. The advantages of Kinect sensor in place of a laser scanner in 2D SLAM implementation is done [21] along with a comparison study of different parameter settings.

II. Methodology and Setup

In this experiment, a real-time navigation scenario is identified as shown in Figure-1 where a vehicle can navigate itself through an underground parking, mapping the area as it goes along, avoiding obstacles and localizing itself in the map generated.

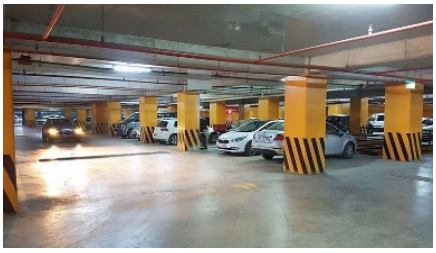


Fig. 1. *The Real-time scenario that has been taken as reference*

This feature can be applied in self-parking applications under Level 3 of vehicle automation and is considered as a mandatory feature in the immediate future of automobile technology [22]. The real-time location is measured, scaled-down and a miniature scenario is set up according to the dimensions obtained after scaling down the real-time scenario. The scaling factor is calculated by the manual comparison of an XUV with real-time environment dimensions. The miniature vehicle is fabricated by considering the overall scaling factor of 7.5. The vehicle is scaled down with a similar scaling factor. The miniature real-time environment measurements are drafted by taking relative dimensions of the actual vehicle and actual obstacles like pillars, walls etc. Tests are performed in these environments built according to the same scale. The basic layout of the scenario after scaling down from the real-time scenario looks like figure 2.

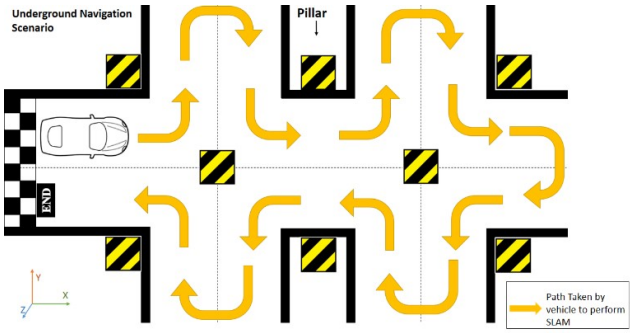
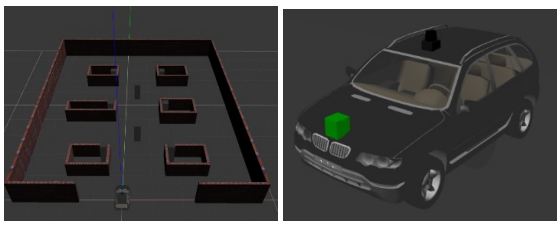


Fig. 2. *Sketch of the scenario to be set up for the analysis*

The complexity of the miniature scenario is increased by placing a pillar in the middle of the pathways. But, since simulation gives liberty of unlimited resources, the simulation environment is put up as an exact version of the sketch prepared. The scaled-down real-time scenario is put up to match the simulation environment. In this study, some features of a real vehicle are not considered such as the powertrain, steering mechanism, vehicular dynamics, etc.

A. Simulation setup

Simulation is one of the best modes of implementation of any system instead of performing it real-time. Simulation of a system gives us the flexibility of analyzing many scenarios that are impossible or hard to implement in the real world. It also saves us resources like time, money, manpower, etc. Simulation of these algorithms is also carried out to test out the results and compare the characteristics.



(a) *(b)*

Fig. 3. *(a) Simulation environment and (b) Vehicle model used for simulation*

An environment is set up using available options in the simulator. These environments are called 'worlds'. Creating a world is simple, as one only needs to drag and drop components from the menu onto the ground plane and arrange them accordingly. The world is then saved in ".world" format which is required to perform the simulations. The vehicle is defined with all the links and joints and is written in the form of a Unified Robot Description Format (URDF). A URDF represents a model in an XML format, a text file consisting of all the required description of a vehicle and sensors used for simulation.

B. Real-time implementation setup

The next step after the simulation is prototyping. A scaled-down real-time environment is set up with all the obstacles as in simulation to perform SLAM. To test the algorithms, a mini vehicle is fabricated. The differential drive is employed for the movement of this vehicle. The sensors required for experimenting i.e., LiDAR, Stereo camera are mounted on the chassis.

A pi3 B+ is setup as Remote PC (RPC) and is placed on the vehicle. The RPC is commanded by the Master PC (MPC). The full specifications of Master and Remote PCs, scaled-down vehicle prototype and sensors used in this study are given in Table 1 & Table 2.

(a) *(b)*

Fig. 4. *Scaled-down (a) real-time environment and (b) vehicle prototype used for the study*

Communication between MPC and RPC is carried out by a modem. This vehicle is operated by the MPC using 'Teleop' while mapping. The vehicle is typically a two-storied structure with provisions for LiDAR and Stereoscopic Sensors. The base is carried on by 4 wheels out of which two are castor wheel one each at the front and rear centers.

TABLE I. MASTER PC SPECIFICATIONS

Parameter	Configuration
Processor	Intel Core™ i5-8250 CPU @ 2.70GHz x 4
GPU	Nvidia MX150 4GB
RAM	8 GB DDR4
OS	Ubuntu 16.04
ROS	Kinetic Kame
Storage	Samsung EVO SSD

TABLE II. VEHICLE PROTOTYPE SPECIFICATIONS

Parameter	Configuration
Remote PC	Low Cost SBC
LiDAR	Low cost 2D LiDAR
Camera	Stereo Camera
Motors	SPG30E-200K DC Geared Motor 17RPM 80 N.cm 12V
ROS	Kinetic Kame
Storage	Samsung EVO SSD

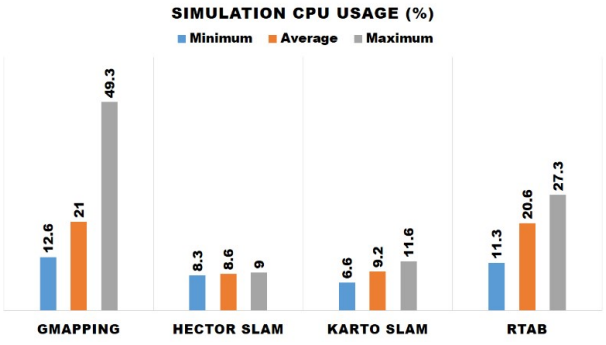
The vehicle is moved around in the environment in which boxes are put up as obstructions. A 2D LiDAR is mounted on the vehicle. The vehicle will map the environment as it maneuvers in the environment. This map data is loaded onto the vehicle while localization. When the target location is given, the vehicle moves on its own to reach the target location avoiding the obstacles dynamically.

III. RESULTS AND ANALYSIS

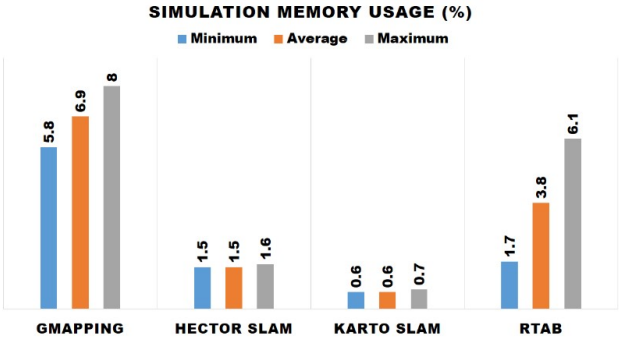
The model is maneuvered in the simulation and real-time environments in the same pattern every time while creating maps using various SLAM algorithms. This helps in creating a similar ground to compare resources utilized by the SLAM algorithms. CPU and Memory usage are observed, and comparison analysis is performed between all the SLAM algorithms.

A. Simulation Results

Computational resources like CPU and memory are observed and the data is analyzed to obtain the results for comparison. Coming to the resource usages during simulation, we evaluated each SLAM algorithm's resource usage. Peak usage of CPU is obtained when the sensors capture new data, i.e., when sensors collect data from an unexplored region of the environment. And then it drops as there is no new data left to capture. Hence those spikes are recorded and are analyzed to obtain the results.



(a)



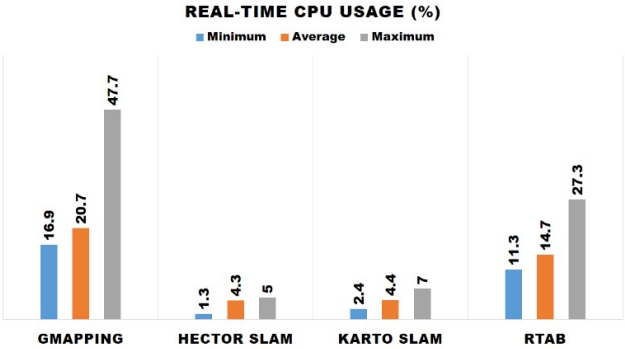
(b)

Fig. 5. *Simulation computational resources utilized by various Algorithms (a) CPU Usage (b) Memory Usage*

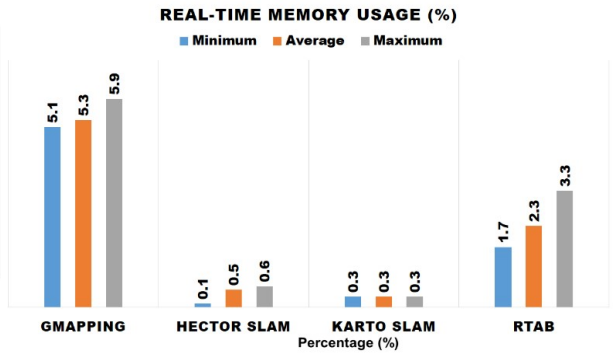
G-mapping utilized the highest computational resources since it is based on the particle filter approach. Due to that reason, G-mapping has large data to capture and store in terms of individual particles. As Hector SLAM only depends on 2D laser data and has zero dependencies on odometry information, it uses lesser computational resources than G-mapping. Karto SLAM being a graph-based approach is expected to use minimum computational resources than all other SLAM techniques. But in times, due to the node formation to optimize the edges, Karto SLAM uses more CPU and memory than Hector. RTAB being the only algorithm which is using a stereo camera uses a decent amount of CPU and memory for computation. But since it is embedded with a memory management approach, it optimizes the usage of CPU and memory.

B. Real-Time Implementation Results

Implementation of SLAM in real-time is to physically verify the performance of the algorithms with provided resources. Real-time implementation of the above SLAM algorithms has produced a similar pattern of results as simulation. Many external factors like odometry error due to surface friction, interference of external light or reflections caused due to polished surfaces in the environment affect the real-time implementation results. All the algorithms are run on the vehicle prototype and the results are captured.



(a)



(b)

Fig. 6. *Real-time computational resources utilized by various Algorithms (a) CPU Usage (b) Memory Usage*

Computational resource utilization results are obtained by implementing SLAM algorithms with specific system resources as mentioned in Table 1 and Table 2. The results may vary when the system configuration is altered. However, the pattern of results may be similar when we use a different configuration of computational resources. Observations are recorded every 1 second. Maximum, minimum and average values of CPU and memory consumption are calculated.

G-mapping dominates in terms of CPU usage. This is because G-mapping is a particle filter-based approach. It calculates pose and velocity information at particle level which will be in a huge number in the map obtained. The same applies in case of memory usage concerning G-mapping. Hector SLAM utilizes lesser resources than G-mapping as it does not have much computational requirement. Since it only utilized LiDAR data, Hector uses less CPU as the processing requirement is low and lesser memory as the data needs less computation. Karto SLAM is graph-based and the data it works on is much lesser. But it optimizes the edge data utilizing higher CPU than Hector, but memory usage is as expected, least in all the algorithms. RTAB utilizes a pretty good amount of CPU and memory as it collects and processes data from the depth sensor as well as LiDAR. Since the depth sensor maps point cloud data as well, it stands second in terms of the utilization of computational resources.

C. Comparison Analysis

The results from figures (5) and (6) can be compared to observe the performance of the SLAM algorithms in simulation environment and in real-time application. Since resources usage is mostly based on the collection of data from sensors, mean values can be considered as an effective factor for comparison. As mean is calculated from the observations made at regular intervals, there is very less probability of data omission. The mean values of both CPU usage as well as Memory consumption are shown in figure (7).

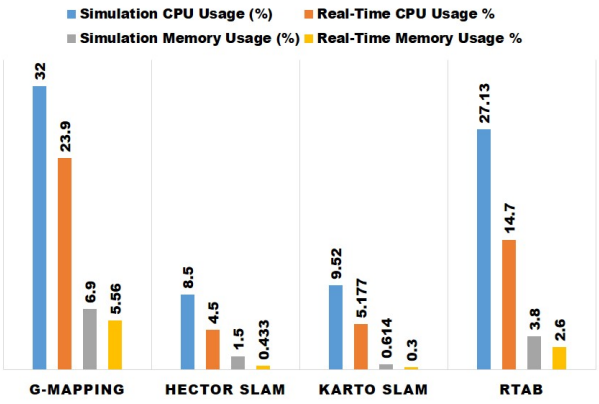


Fig. 7. *Computational Resources' mean of various algorithms*

The Mean of CPU and Memory usage can be observed to analyze the pattern of different SLAM algorithms' computational resource utilization. G-mapping draws the highest portion of the resources than all other SLAM algorithms followed by RTAB, Karto SLAM, and Hector SLAM. Simulation resource utilization is quite higher when compared with real-time usage. In the simulation, the scenario is created in the virtual world and the sensors are also, virtual collecting data from the scenario. This has a higher processing requirement than a real-time application.

IV. Conclusion

This paper has described the performance of G-mapping, Hector SLAM, Karto SLAM, and RTAB algorithms in terms of computational efficiency in simulation and real-time environments. The results obtained and analyzed in terms of usage of computational resources. The overall comparison establishes that for given resources, the Karto SLAM algorithm possesses lower computational requirements providing fair amount of accuracy and quality in map generation.

Further research can be carried out in developing a robotic car prototype with actual powertrain, transmission and other elements which can help in bringing robotic cars in daily commute. A SLAM algorithm can be developed with combined characteristics of particle filter and graph slam which consumes less computational resources and provide higher mapping accuracy and quality.

References

[1] A. Documents et al., "Sae-J3016-2018," pp. 1–2, 2014.

[2] B. Van Arem, B. Van Arem, J. G. Van Driel, and R. Visser, "The Impact of Cooperative Adaptive Cruise Control on Traffic-Flow Characteristics SURF-STAD View project Meaningful Human Control of Automated Driving Systems (MHC-ADS) View project The Impact of Cooperative Adaptive Cruise Control on Traffic-Flow Charact," IEEE Trans. Intell. Transp. Syst., vol. 7, no. 4, p. 429, 2006.

[3] N. Highway Traffic Safety Administration and U. Department of Transportation, "Key Findings Fatalities by Speeding Involvement, 2008-2017 Year Speeding Involvement Total Not Speeding Speeding Number Percent Number Percent Number Percent Annual Report File (ARF)," 2017.

[4] R. Magargle et al., "A Simulation-Based Digital Twin for Model-Driven Health Monitoring and Predictive Maintenance of an Automotive Braking System," in Proceedings of the 12th International Modelica Conference, Prague, Czech Republic, May 15-17, 2017, 2017, vol. 132, pp. 35–46.

[5] K. Melbouci, S. N. Collette, V. Gay-Bellile, O. Ait-Aider, and M. Dhome, "Model based RGBD SLAM," in Proceedings - International Conference on Image Processing, ICIP, 2016, vol. 2016-August, pp. 2618–2622.

[6] S. Thrun, "Probabilistic robotics," Commun. ACM, vol. 45, no. 3, Mar. 2002.

[7] G. Grisetti, C. Stachniss, and W. Burgard, "Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters," IEEE Trans. Robot., vol. 23, no. 1, pp. 34–46, Feb. 2007.

[8] S. Kohlbrecher, O. Von Stryk, J. Meyer, and U. Klingauf, "A flexible and scalable SLAM system with full 3D motion estimation," 9th IEEE Int. Symp. Safety, Secur. Rescue Robot. SSRR 2011, pp. 155–160, 2011.

[9] R. Vincent, B. Limketkai, and M. Eriksen, "Comparison of indoor robot localization techniques in the absence of GPS," Detect. Sens. Mines, Explos. Objects, Obs. Targets XV, vol. 7664, p. 76641Z, 2010.

[10] K. Konolige, G. Grisetti, R. Kümmerle, W. Burgard, B. Limketkai, and R. Vincent, "Efficient sparse pose adjustment for 2D mapping," in IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings, 2010, pp. 22–29.

[11] M. Labbé and F. Michaud, "RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation," J. F. Robot., vol. 36, no. 2, pp. 416–446, 2019.

[12] M. Quigley, K. Conley, B. Gerkey, … J. F.-I. workshop on, and undefined 2009, "ROS: an open-source Robot Operating System," willowgarage.com.

[13] K. W. Chiang, G. J. Tsai, H. W. Chang, C. Joly, and N. EI-Sheimy, "Seamless navigation and mapping using an INS/GNSS/grid-based SLAM semi-tightly coupled integration scheme," Inf. Fusion, vol. 50, pp. 181–196, 2019.

[14] G. Castro, M. A. Nitsche, T. Pire, T. Fischer, and P. De Cristóforis, "Efficient on-board Stereo SLAM through constrained-covisibility strategies," Rob. Auton. Syst., vol. 116, pp. 192–205, 2019.

[15] S. Kato, E. Takeuchi, Y. Ishiguro, Y. Ninomiya, K. Takeda, and T. Hamada, "An open approach to autonomous vehicles," IEEE Micro, vol. 35, no. 6, pp. 60–68, 2015.

[16] M. Pierzchała, P. Giguère, and R. Astrup, "Mapping forests using an unmanned ground vehicle with 3D LiDAR and graph-SLAM," Comput. Electron. Agric., vol. 145, pp. 217–225, Feb. 2018.

[17] Y. Abdelrasoul, A. B. S. H. Saman, and P. Sebastian, "A quantitative study of tuning ROS gmapping parameters and their effect on performing indoor 2D SLAM," 2016 2nd IEEE Int. Symp. Robot. Manuf. Autom. ROMA 2016, 2017.

[18] I. Z. Ibragimov and I. M. Afanasyev, "2018 15th Workshop on Positioning, Navigation and Communications, WPNC 2018," 2018 15th Work. Positioning, Navig. Commun. WPNC 2018, 2018.

[19] A. Buyval, I. Afanasyev, and E. Magid, "Comparative analysis of ROS-based monocular SLAM methods for indoor navigation," Ninth Int. Conf. Mach. Vis. (ICMV 2016), vol. 10341, no. Icmv 2016, p. 103411K, 2017.

[20] M. Rojas-Fernandez, D. Mujica-Vargas, M. Matuz-Cruz, and D. Lopez-Borreguero, "Performance comparison of 2D SLAM techniques available in ROS using a differential drive robot," 2018 28th Int. Conf. Electron. Commun. Comput. CONIELECOMP 2018, vol. 2018-Janua, pp. 50–58, 2018.

[21] K. Kamarudin, S. M. Mamduh, A. Y. Md Shakaff, and A. Zakaria, "Performance analysis of the microsoft kinect sensor for 2D simultaneous localization and mapping (SLAM) techniques," Sensors (Switzerland), vol. 14, no. 12, pp. 23365–23387, Dec. 2014.

[22] "US10308244B2 - Systems for automatic driverless movement for self-parking processing - Google Patents." [Online]. Available: https://patents.google.com/patent/US10308244B2/e