The Journal of Open Source Software

SLAM Toolbox: SLAM for the dynamic world

Steve Macenski[1] **and Ivona Jambrecic**[2]

1 Open-Source Robotics Engineering Lead, Samsung Research **2** Software Engineering Intern, Simbe Robotics

DOI: 10.21105/joss.02783

Software

- Review
- Repository
- Archive

Editor: Arfon Smith

Reviewers:

- @mosteo
- @carlosjoserg

Submitted: 13 August 2020
Published: 11 May 2021

Summary

Developments in the field of mobile robotics and autonomous driving have resulted in the use of robots and vehicles in retail stores, hospitals, warehouses, on the roads, and on sidewalks. These deployed areas are both dynamic and frequently massive in scale. The average size of a Walmart store is over 16,000 m^2 (Walmart, 2020) and a single square block in Chicago is over 21,000 m^2 (Heramb, 2007). Retail and warehouse spaces can change drastically throughout the year and the state of roadways can be changing by the hour. Much work has been made to address changing environments in robot perception (Macenski, Tsai, et al., 2020), but less has been built in open-source to represent maps of dynamic spaces.

For fully autonomous deployed systems to operate in these large and changing environments, they require tools that can be used to accurately map an area specified for their operation, update it over time, and scale to handle mapping of some of the largest indoor and outdoor spaces imaginable. The field of Simultaneous Localization and Mapping (SLAM) aims to solve this problem using a variety of sensor modalities, including: laser scanners, radars, cameras, encoders, gps and IMUs. The most commonly used perception sensor used for localization and mapping in industrial environments is the laser scanner (Chong et al., 2015). SLAM methods using laser scanners are generally considered the most robust in the SLAM field and can provide accurate positioning in the presence of dynamic obstacles and changing environments (Cole & Newman, 2006).

Previously existing open-source laser scanner SLAM algorithms available to users in the popular Robot Operating System (ROS) include GMapping, Karto, Cartographer, and Hector. However, few of these can build accurate maps of large spaces on the scale of the average Walmart store. Even fewer can do so in real-time using the mobile processor typically found in mobile robot systems today. The only package that could accomplish the above was Cartographer. However, it was abandoned by Google and it is no longer maintained.

We propose a new fully open-source ROS package, SLAM Toolbox, to solve this problem. SLAM Toolbox builds on the legacy of Open Karto (Konolige et al., 2010), the open-source library from SRI International, providing not only accurate mapping algorithms, but a variety of other tools and improvements. SLAM Toolbox provides multiple modes of mapping depending on need, synchronous and asynchronous, utilities such as kinematic map merging, a localization mode, multi-session mapping, improved graph optimization, substantially reduced compute time, and prototype lifelong and distributed mapping applications.

This package, `slam_toolbox` is open-source under an LGPLv2.1 at https://github.com/SteveMacenski/slam_toolbox.git and is available in every current ROS distribution. It was also selected as the new default SLAM vendor in ROS 2, the second generation of robot operating systems, replacing GMapping. SLAM Toolbox was integrated into the new ROS 2 Navigation2 project, providing real-time positioning in dynamic environments for autonomous navigation (Macenski, Martín, et al., 2020). It has been shown to map spaces as large as 24,000 m^2, or 250,000 ft^2, in real-time by non-expert technicians. An example map can be seen in Figure 1.

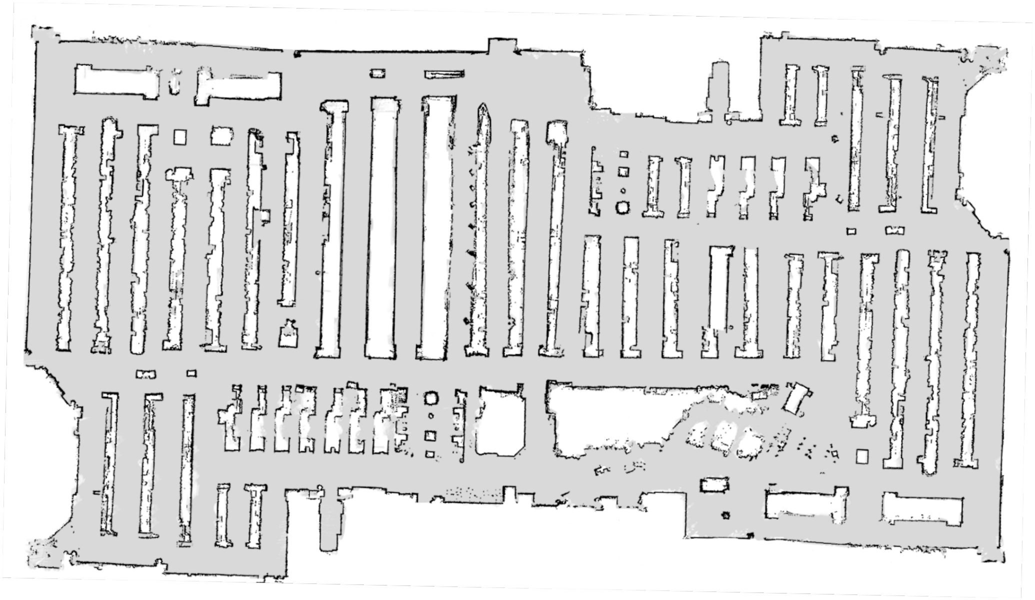

Figure 1: Retail store map created using SLAM Toolbox (Macenski, 2019).

Statement of Need

This software package provides an approach for multi-session mapping and localization. It also utilizes modern graph-optimization techniques and provides first of its kind pose-graph manipulation tools and demo applications.

Related Work

SLAM algorithms can be classified into two groups: the earlier algorithms that use the Bayes-based filter approaches (Thrun et al., 2005), and newer graph-based methods (Thrun & Montemerlo, 2006). Significant filter-based implementations available as ROS packages are GMapping (Grisetti et al., 2007) and HectorSLAM (Kohlbrecher et al., 2011). Cartographer (Hess et al., 2016) and KartoSLAM (Konolige et al., 2010) are the major graph-based implementations available.

GMapping is one of the most commonly used SLAM libraries, presented in 2007. It uses a particle filter approach to SLAM for the purpose of building grid maps from 2D lidar data. However, GMapping is not well suited for large spaces and fails to accurately close loops at an industrial scale. Additionally, filter-based approaches cannot be easily reinitialized across multiple sessions.

HectorSLAM relies on lidar scan matching and 3D navigation filter based on EKF state estimation. This method focuses on real-time robot pose estimation and generates 2D map with high update rate. Unlike other mentioned methods, Hector does not use odometry data, which can cause inaccurate pose and map updates when lidar scans arrive at a lower rate, or when mapping large or featureless spaces. HectorSLAM however does not provide loop closure capabilities, making it unsuitable for reliable mapping of large spaces or when using laser scanners with low update rates.

KartoSLAM and Cartographer are both graph-based algorithms that store a graph of robot poses and features. Graph-based algorithms have to maintain only the pose-graph, which usually makes it efficient in handling resources, especially while building maps of a large scale. KartoSLAM uses Sparse Bundle Adjustment for loop-closure graph optimization. Cartographer consists of front-end, which is in charge of scan matching and building trajectory and submaps, and back-end that does the loop closure procedure. The graph solver used in Cartographer is Google Ceres (Agarwal et al., n.d.). Cartographer provides pure localization mode, when user has a satisfactory map for usage. It also provides data serialization for storing processed submaps only. However, Cartographer has stopped maintenance and support from Google and has been largely abandoned. Further, it fails to build suitable maps for annotation and localization with other localization software packages on robotic platforms without exceptional odometry. The software's unusual complexity makes it challenging to modify or resolve seemingly simple issues, making it not suitable for many applications.

Features

SLAM Toolbox is able to map spaces effectively using mobile Intel CPUs commonly found on robots well in excess of 100,000 ft^2. It can be done easily using untrained technicans typically hired to deploy robot solutions or remotely using monitoring systems. Some applications have been created to automatically map a space using SLAM Toolbox as well as paired with exploration planners.

It can also serialize a current mapping session and deserialize it at a later time to continue refining or expanding an existing map. This serialization saves the complete raw data and pose-graph rather than submaps, as in Cartographer, allowing a variety of novel tools to be developed and more accurate multi-session mapping. These utilities include manual pose-graph manipulation, whereas a user can manually manipulate the pose-graph nodes and data to rotate a map or assist in a challenging loop closure, shown in Figure 2. It also includes kinematic map merging, the process of merging multiple serialized maps into a composite map. A 3D visualizer plugin was also created to assist in utilization of these tools and the core SLAM library capabilities. Many additional tools and utilities could be developed using this representation as well.

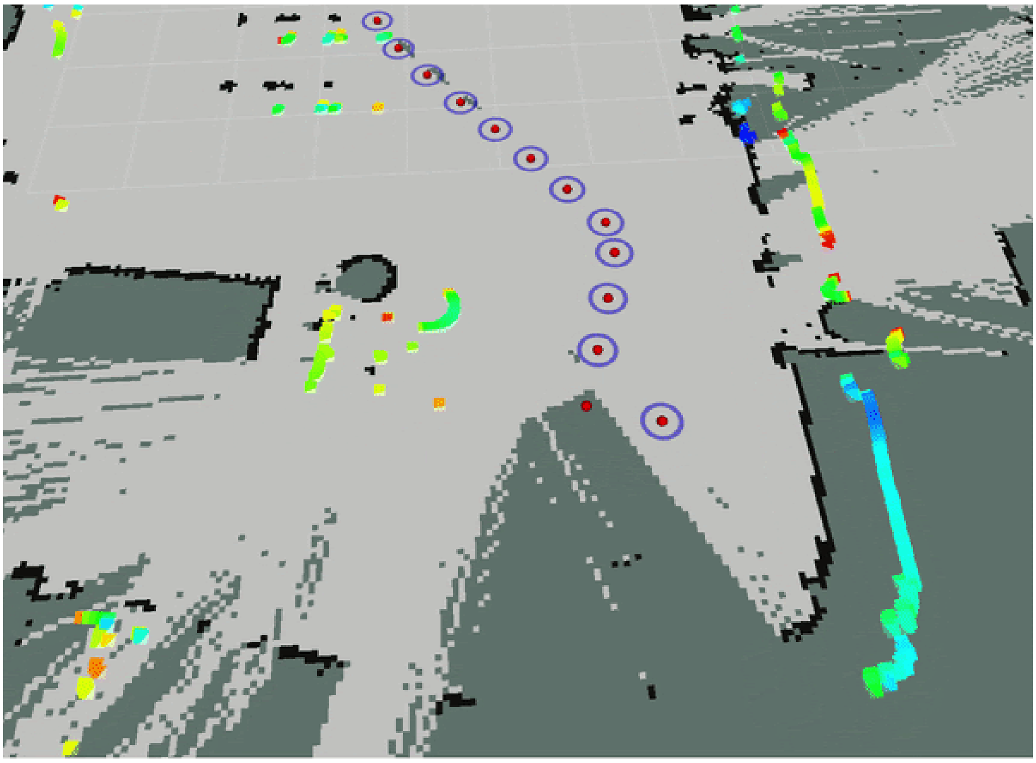

Figure 2: Pose-graph manipulation in progress to manually match a node's laser scan to the map (Macenski, 2019).

It provides 3 major operation modes and executables: synchronous mapping, asynchronous mapping, and pure localization. Synchronous mapping provides the ability to map and localize in a space keeping a buffer of measurements to add to the SLAM problem. This can be advantageous when the quality of the map is of particular importance or when doing offline processing. By contrast, the asynchronous mode will only process new measurements when the last measurement is completed and the new update criteria are met. As a result, this will never lag behind real-time when running complex loop closures. However, the map may not include all valid measurements if processing the last one takes too long. This mode is advantageous when the quality of real-time localization is of particular importance. Both of these modes can be used for multi-session SLAM, the process of reloading a prior session and continuing to refine the pose-graph. Figure 3 shows a map of a large office building created by partially mapping in one session and completing the map in another session. This map has multiple loop closures between the two datasets and was later used with the pure-localization mode to navigate autonomously.

The pure localization mode cannot be used to persist changes in the environment. Instead, it uses a rolling buffer of measurements in the current session and matches them against the original session(s) measurements and pose-graph. The current session's measurements will be added to the pose-graph with new constraints and nodes in the graph. This allows changes in the environment to be embraced to increase localization quality based on new features or moved objects. Over time, the measurements in the rolling buffer will "expire" and be removed from the pose-graph and localization problem, reverting the pose-graph to its original state for that region. The authors refer to this process as elastic pose-graph deformation. An interesting side effect is that the pure-localization mode can be used for effective lidar odometry when paired with no prior mapping session data. It will simply match against its local buffer and keep only a recent view of the environment, allowing lidar odometry to scale to infinite sized spaces.

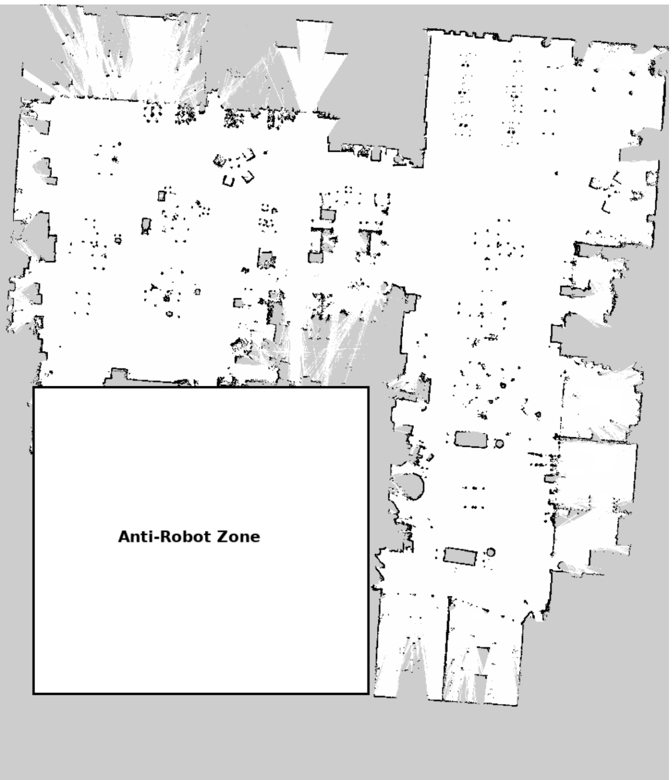

Figure 3: Large office building map created using multiple SLAM sessions (Macenski, 2019).

Finally, many updates were made to the OpenKarto SLAM libary. The measurement matching methods were restructured for a 10x speed-up enabling multi-threaded processing. The provided Sparse Bundle Adjustment optimization interface was replaced with Google Ceres, providing faster and more flexible optimization settings. Additionally, the optimizer interface was turned into a run-time dynamically loaded plugin interface to allow future developers to use the latest and greatest in optimization technologies without modifying the original code. Serialization and deserializion support was enabled to allow for saving and reloading mapping sessions. Finally, new processing modes and K-D tree search were developed to process measurements to enable localization and multi-session mapping. Various smaller improvements and optimizations were also made but excluded for brevity.

Robots Using `slam_toolbox`

SLAM Toolbox has been integrated, tested, and deployed on a number of robot platforms across the world by both industry and researchers. It is also the default SLAM vendor for ROS 2.

A few known examples where SLAM Toolbox has been used or is being used are:

- Simbe Robotics' Tally
- ROBOTIS' Turtlebot3
- Samsung Research America and Russia's research teams
- Rover Robotics' Rover
- Pal Robotics ARI
- Intel's Open Source Group
- Queensland University of Technology researchers

- Robosynthesis' EXTRM SC2.0
- Unbounded Robotics UBR-1
- Byte Robotics' Platform
- Unmanned Life's Platform
- MT Robot AG
- Magazino robot datasets
- 6 River Systems

Acknowledgements

We acknowledge this work was largely developed at Simbe Robotics and later continued open-source support and development at Samsung Research.

References

Agarwal, S., Mierle, K., & Others. (n.d.). *Ceres solver*. http://ceres-solver.org

Chong, T. J., Tang, X. J., Leng, C. H., Yogeswaran, M., Ng, O. E., & Yu Zheng, C. (2015). Sensor technologies and simultaneous localization and mapping (SLAM). *Procedia Computer Science*, *76*, 174–179. https://doi.org/10.1016/j.procs.2015.12.336

Cole, D. M., & Newman, P. M. (2006). Using laser range data for 3D SLAM in outdoor environments. *Proceedings - IEEE International Conference on Robotics and Automation*, *2006*, 1556–1563. https://doi.org/10.1109/ROBOT.2006.1641929

Grisetti, G., Stachniss, C., & Burgard, W. (2007). Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Transactions on Robotics*, *23*(1), 34–46. http://www.ros.org/wiki/gmapping

Heramb, C. (2007). *Street and site plan design standards*. City of Chicago.

Hess, W., Kohler, D., Rapp, H., & Andor, D. (2016). Real-time loop closure in 2D LIDAR SLAM. *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 1271–1278.

Kohlbrecher, S., von Stryk, O., Meyer, J., & Klingauf, U. (2011). A flexible and scalable SLAM system with full 3D motion estimation. *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, 155–160. http://www.ros.org/wiki/hector_slam

Konolige, K., Grisetti, G., Kümmerle, R., Burgard, W., Limketkai, B., & Vincent, R. (2010). Efficient sparse pose adjustment for 2D mapping. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 22–29. https://doi.org/10.1109/IROS.2010.5649043

Macenski, S. (2019). On use of SLAM toolbox. *ROSCon*.

Macenski, S., Martín, F., White, R., & Ginés Clavero, J. (2020). The marathon 2: A navigation system. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. https://github.com/ros-planning/navigation2

Macenski, S., Tsai, D., & Feinberg, M. (2020). Spatio-temporal voxel layer: A view on robot perception for the dynamic world. *International Journal of Advanced Robotic Systems*, *17*(2). https://doi.org/10.1177/1729881420910530

Thrun, S., Burgard, W., & Fox, D. (2005). *Probabilistic robotics*. MIT Press. ISBN: 0262201623 9780262201629

Thrun, S., & Montemerlo, M. (2006). The graph SLAM algorithm with applications to large-scale mapping of urban structures. *I. J. Robotic Res.*, *25*, 403–429. https://doi.org/10.1177/0278364906065387

Walmart. (2020). *Walmart u.s. Our business*. Walmart.