ROS based Autonomous Mobile Robot Navigation using 2D LiDAR and RGB-D Camera

Sukkpranhachai Gatesichapakorn
Faculty of Engineering
Kasetsart University
Bangkok, Thailand

Jun Takamatsu
Graduate School of Information Science
Nara Institute of Science and Technology
Nara, Japan

Miti Ruchanurucks
Faculty of Engineering
Kasetsart University
Bangkok, Thailand

Abstract—**This paper presents an implementation of autonomous mobile robot with the robot operating system (ROS). The system utilizes 2D LiDAR and RGB-D camera with ROS 2D navigation stack, with low power consumption and inexpensive onboard computer. Safe to property and human is of priority. Regarding software, we use official ROS packages with minimal default parameter changes. For hardware, the limitation of equipment and system setting are among challenges. Our proposed systems can perform navigation with dynamic obstacle avoidance capability. The Contribution of this paper is two system setups of ROS navigation stack are proposed. The first system is implemented on Raspberry Pi 3 using 2D LiDAR only. The second system is implemented on Intel NUC using 2D LiDAR and RGB-D camera. To evaluate the performance, usability testing was performed in multiple experiments. Our experiment results show that the robot can avoid objects in their path, or stop in case of unavoidable. Discussion of problems and solutions are presented after the experiment results.**

Keywords— Autonomous mobile robot; LiDAR; navigation; RGB-D; ROS

I. Introduction

Nowadays, an autonomous mobile robot is widely used for several purposes. Collaboration on various kinds of robots from different makers or even self-developed robots in the same ecosystem are trending. The standard and open-source platform are good choices e.g. Robot Operating System or ROS [1]. ROS is a widely used platform for robots implementation [2]-[3]. ROS official packages are adequate in common robotics task. Furthermore, ROS provides API to build custom packages or communicate with external systems or equipment e.g. interfaces and planner [3].

Regarding laser-based mobile robot navigation and localization, 2D LiDAR is one sensor that is widely used. However, one disadvantage of 2D LiDAR is it senses using only a single horizontal scanning line. On the other hand, 3D LiDAR is available, however more expensive. In this sense, sensor fusion is one of efficient solutions. Currently, cameras with depth are in moderate price. RGB-D camera is a camera that gives us both color image and depth data. OpenNI and Point Cloud Library (PCL) [4] are among tools for handling data from the RGB-D camera. To overcome the 2D LiDAR limitation, our system also utilizes 3D depth point cloud from PCL as a second source for obstacle detection purpose.

Regarding other hardware configurations, our mobile robot was already mounted with 2D LiDAR in position. The robot was designed to use in some special applications that we cannot mention here. We further put just a few more equipment in the mobile robot, with minimal space and power consumption. According to the design, our mobile robot will be performing as a moving base for other works with an individual control system.

This paper presents the complementary ROS navigation stack that utilizes 2D LiDAR and RGB-D camera with consideration on limitation regarding the current position of LiDAR, which is not allowed to alter. The work focuses on implementing the mobile robot navigation and obstacle avoidance in a dynamic environment with ROS. Our key constraint is the robot must be safe to property and human. In case of unavoidable, the robot will stop and wait. In order to do so, there is a concern about objects that are not detected by 2D LiDAR effectively. Hence, we have tried multiple settings and experiments to observe usability in navigation and avoidance. We selected two satisfied settings based on an experimental outcome to contribute in this paper.

Information of our systems is separated into sections. Please be noted that robot and sensor mounting details, ROS installation and settings, data preparation are not mentioned. System and equipment details including camera calibration reference are present in section II System overview. An implementation and experiment preparation is in section III Implementation and Experiment setup. The satisfied systems for the environment with and without an object that is related to 2D LiDAR limitation are in section IV Experiment and result. Section V Conclusion is the conclusion of both systems including some discussion on common problems.

II. System overview

A. Robot Operating System in Container

For software, we use ROS as the main platform to implement a whole system. We present a container software called Docker (see www.docker.com) for deploying our ROS system on the robot side. The same Docker image is used for all robots in our multiple setups. Our Docker images contain ROS and required packages. Dockerfile is a text file that contains all commands to automate creation for Docker image.

We create a simple Dockerfile based on an official image called "ros:kinetic-robot" with additional sets of packages that are list below.

- "ros-kinetic-openni2-launch" This is a set of packages that contains tools for connecting to the RGB-D camera and handling depth data with PCL in ROS.
- "ros-kinetic-navigation" This is a set of packages that contains navigation stack in ROS.
- "ros-kinetic-lms1xx" This is a set of packages for connecting SICK LMS100 series 2D LiDAR with ROS.
- "libaria-dev" This is a required library packages that contain tools for using our mobile robots with ROS.

Ultimately, we have shared ROS package configurations directory and launch directory from the host system to the container dynamically. With this solution, it is possible to use one image for multiple setups and all different configurations.

B. Equipment and Sensors

For mobile robot body, it is mounted with 2D LiDAR and RGB-D camera. The LiDAR sensor was calibrated and placed at 40 cm above the floor. It is ready to run with ROS by "rosaria", a supported library. An onboard computer connects to the robot with an RS232C-USB Converter. The robot and sensors are listed below.

- "Adept Mobile robot Pioneer 3-DX" This is the base mobile robot. It comes with a two-wheel differential driver, wheel encoders and a flat tray on top. The speed we use is 0.33 m/s with acceleration 0.3 m/s^2.
- "SICK LMS100-10000" This is our 2D LiDAR sensor. Its aperture angle is 270°, sense up to the range of 18 m. The data rate we use is 50 Hz.
- "ASUS Xtion PRO LIVE" This is our RGB-D camera. The distance of use is in between 0.8 m and 3.5 m. The stream resolution we use is 640×480 pixels, 30 frame/s. The depth point cloud output is 640×480 Z16 at 30 Hz.

Before integrating the RGB-D camera into a navigation system, we had calibrated it with a checkerboard by a method in ROS website (see http://wiki.ros.org/camera_calibration).

Our onboard computers are low power consumption and inexpensive machines. We adopt and compare "Raspberry Pi 3 Model B+" and "Intel® NUC Kit D54250WYK". They are supplied by 85 Wh power bank and communicate to the control center via a wireless network. Our complete equipment setup with the mobile robot is shown in Fig. 1 and for the wiring diagram, it is in Fig. 2.

After each device are connected with ROS, the next preparation is a static map. Before mapping process could be done in ROS, firstly we have to correct the coordinate transform for related sensors with the robot. In our system, we have used "tf2_ros" package with static transform type (see http://wiki.ros.org/tf2). Doing so transforms LiDAR and camera coordinate to our robot base coordinate. ROS are help us in the rest of the kinematic and coordinate transformation.

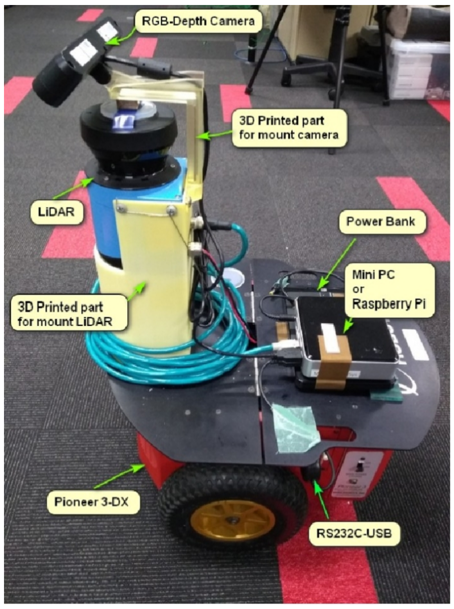

Fig. 1. The mobile robot with equipment

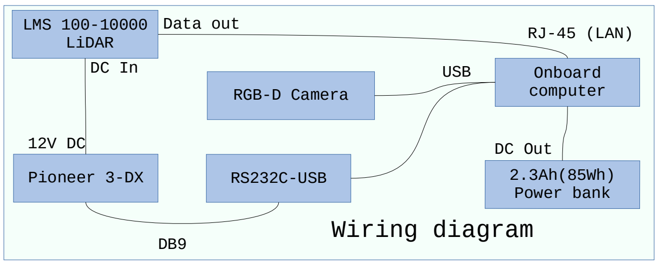

Fig. 2. Equipment wiring diagram

C. Mapping and localization

In our system, we use a static map for navigation reference. A target position or way-points we send to the robot are on static map coordinate. In ROS, they provide a 2D laser-based SLAM (Simultaneous Localization and Mapping) packages in various implementation, [5]. We select "slam_gmapping" packages, an implementation of [6]; not RGB-D SLAM due to calculation time, [7]. With "slam_gmapping", 2D map is created from LiDAR and odometer of a robot. To collect data with SLAM, we use "teleop_twist_keyboard" package for control our robot movement via a wireless network.

During navigation, we use "amcl" package for localizing nodes. The node is implemented by AMCL (Adaptive Monte Carlo Localization) method (see http://wiki.ros.org/amcl).

III. Implementation and Experiment setup

This section is realized by two main components. First is an onboard computer with ROS navigation stack in Docker image prepared. Second is the control center with ROS that runs way-point control script. Furthermore, virtualization tools are utilized for monitoring robot status in experimentation. The experiment area is indoor environment with a public space and a narrow way. An obstacle objects including people are randomly appeared in the robot path. The robot was perform an obstacle avoidance task in experiment at least 2 times per hour.

A. Way-point control script

This part controls and sends target poses to the mobile robot. We selected targets from a derived map and transform to pose by adding orientation. Fig. 3 shows the way-point targets used in experiment, marked on the map in the order of A to J.

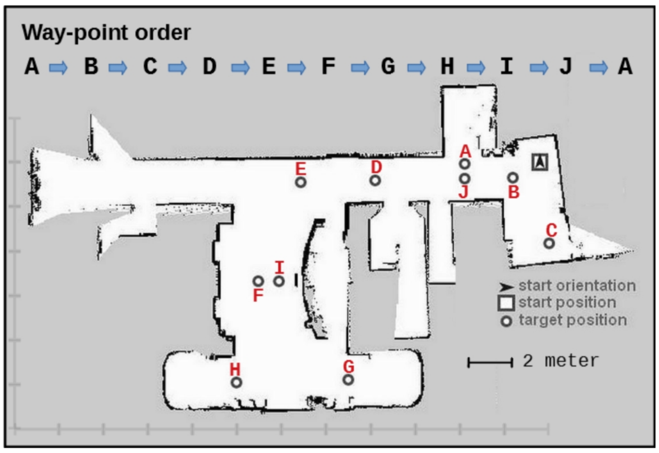

Fig. 3. Way-point position and target order in a map

The flow chart of our control script is shown in Fig. 4. It is written in Python and interfacing with ROS by "actionlib" package (see http://wiki.ros.org/actionlib).

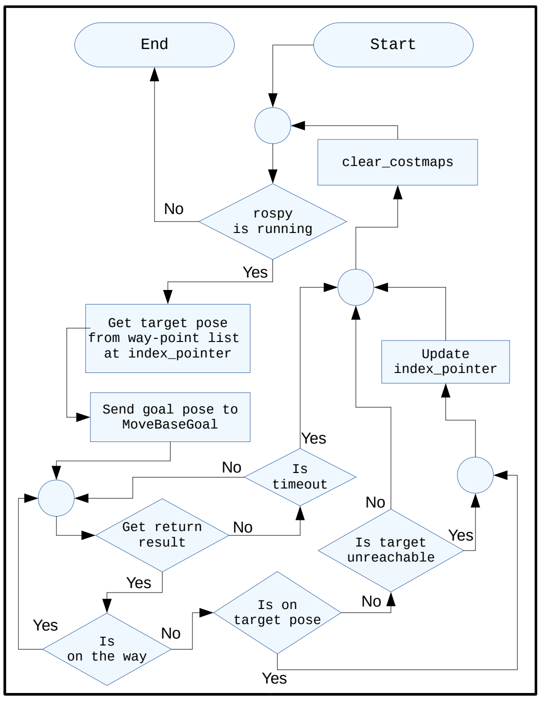

Fig. 4. Way-point control script flow chart

B. Visualize and simulation tools

In our experiment, "rviz" is adopted as the 3D visualization tool (see http://wiki.ros.org/rviz). With rviz, one can see what is going on (e.g. robot position) or set goal pose for the robot from a remote place. We used "Gazebo" for simulation purpose, this software can communicate with ROS. Before some experiment on the actual mobile robot, we have tested our system in simulation to observe work-ability and safety.

C. ROS navigation using 2D LiDAR

Regarding our preliminary navigation schemes, we utilize "costmap_2d" package with static map layer, obstacle layer, and inflation layer (see http://wiki.ros.org/costmap_2d); 2D LiDAR for obstacle layer. We serve a map for static map layer by "map_server" node. For the inflation layer, we use optimized parameters for our robots e.g. robot footprint. Furthermore, we optimize parameters in other nodes (planner) to reduce calculation time and increase performance for the target system. The system diagram is shown in Fig. 5. This system focuses on a capability to avoid human while moving.

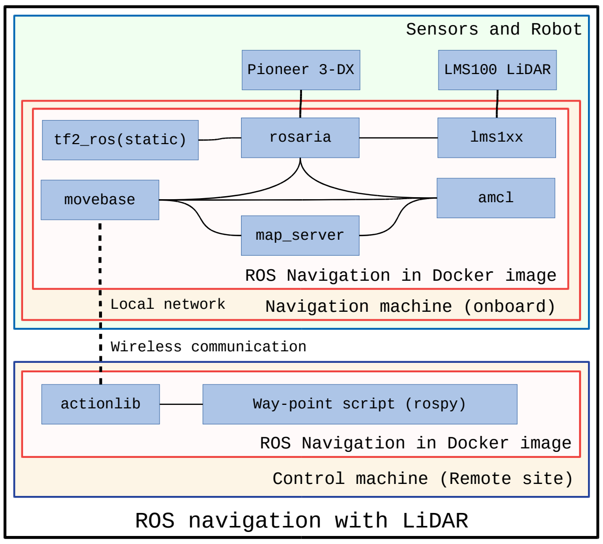

Fig. 5. Our ROS navigation stack with LiDAR

In addition, we present an implementation and deploy the system in Raspberry Pi 3 with "Raspbian Stretch" operating system; low power consumption and inexpensive. The ROS image and configuration files are deployed with Docker. Furthermore, the navigation parameters are tuned for Raspberry Pi. Howsoever, this image and parameters are able to be deployed in other computers as well. We have tried on Laptop and Intel NUC with the same setting as Raspberry Pi. Not surprisingly, both of them are passed all cases in the experiment outcome on navigation, avoidance and recovery.

D. ROS navigation using 2D LiDAR and RGB-D camera

Regarding our navigation schemes, The advantage of this system is an ability to avoid objects that are not detected by 2D LiDAR effectively; able to avoid objects that not below 5 cm. We utilize "costmap_2d" package in similar ways as the previous setup, with one additional layer.

Regarding improvement, with two obstacle layer type. First, utilize 2D LiDAR as an "ObstacleCostmapPlugin". Second, utilize RGB-D camera as a "VoxelCostmapPlugin". Furthermore, we apply filters on depth data (PCL) before passing to obstacle layer; downsampling with "voxel_grid" and noise reduction with "StatisticalOutlierRemoval". We utilize "openni2_launch" package for connecting to RGB-D camera and handling PCL (see http://wiki.ros.org/openni2_launch). The system diagram is shown in Fig. 6. This system is focused on a capability to avoid objects and human while moving.

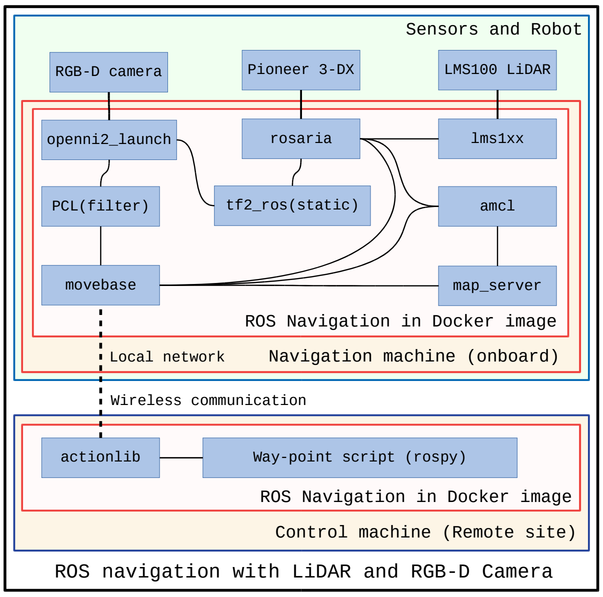

Fig. 6. Our ROS navigation stack with LiDAR and RGB-D camera

In addition, we deploy the system in Intel NUC with “Ubuntu 16.04” operating system; low power consumption and inexpensive. The navigation parameters are tuned for Intel NUC. Howsoever, we use the same ROS image for all system setup. We have tried this setting on Laptop and Raspberry Pi. The Laptop is passed all cases in experiment. Nevertheless, Raspberry Pi was unutilized due to the limitation of processing. An openni2 take over 90% of processing power and produce depth point cloud only at 0.3 Hz, this outcome is not stratify us.

E. Experiment setup

Our experiment for an autonomous mobile robot is running with an automate way-point control script at least one hour. We observe usability by performing a test case during run-time. Basic test case for all experiment is roughly listed below.

- Navigation: a capability to reach all point in way-point.
- Avoidance: a capability to avoid human or objects.
- Recovery: a capability to continue moving after stop.

We monitoring robot status using rviz, as shown in Fig. 7. In case of emergency, the system will be stopped immediately. We check equipment, battery level, network connection, and test area; Then, we move the robot to a starting point and set an initial pose before starting the navigation system.

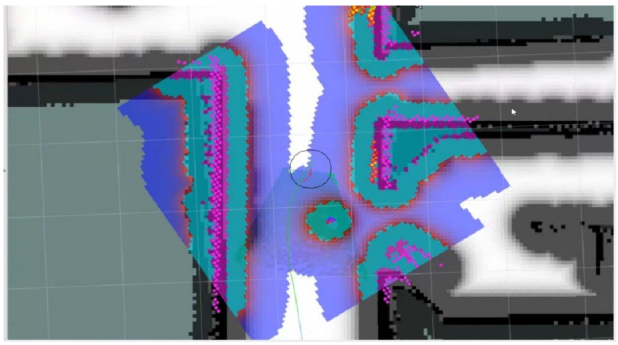

Fig. 7. Visualize robot navigation in rviz

IV. Experiment and Result

The experiment performs in two environments, by concern on objects that are not detected by 2D LiDAR effectively.

A. Experiment for LiDAR only

In an experiment, we test with the only object that detectable by 2D LiDAR; including walking human. The result shows that our system passes all cases. The additional test is long-term running. Raspberry Pi is the longest working time without crashing until we stop at six-hour passed.

B. Experiment for LiDAR and RGB-D camera

In an experiment, we test with an object that high in between 5 cm to 1 m; included walking human. The result shows that our system passes all cases. The additional result, Intel NUC is the longest working time without crashing until an experiment ends; after the battery for our onboard computer was used up at four-hour passed.

V. Conclusion

Both of the paradigms achieve their goal regarding guiding the autonomous mobile robot using only a target area as a destination of navigation without collision. Nevertheless, our system implementation is based on basic ROS navigation stack. In any case, during an investigation, depth information requires high processing time and bandwidth. Furthermore, network communication delay affects ROS node in machines differently. At the present time, the robot cannot move in a backward direction due to some certain safety reasons. Remote calculation for depths data or even navigation stacks node is not satisfied due to heavy throughput and transmission delay.

References

[1] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler and A. Y. Ng, "Ros: an open-source robot operating system," ICRA workshop on open source software, vol. 3, no. 3.2, pp. 5, 2009.

[2] S. Zaman, W. Slany and G. Steinbauer, "ROS-based mapping, localization and autonomous navigation using a Pioneer 3-DX robot and their relevant issues,” 2011 Saudi International Electronics, Communications and Photonics Conference, Riyadh, 2011, pp. 1-5.

[3] Y. Ochiai, K. Takemura, A. Ikeda, J. Takamatsu and T. Ogasawara, "Remote control system for multiple mobile robots using touch panel interface and autonomous mobility," 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, 2014, pp. 3272-3277.

[4] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," 2011 IEEE International Conference on Robotics and Automation, Shanghai, 2011, pp. 1-4.

[5] J. M. Santos, D. Portugal and R. P. Rocha, "An evaluation of 2D SLAM techniques available in Robot Operating System," 2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), Linkoping, 2013, pp. 1-6.

[6] G. Grisetti, C. Stachniss and W. Burgard, "Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters," in IEEE Transactions on Robotics, vol. 23, no. 1, pp. 34-46, Feb. 2007.

[7] J. Sturm, N. Engelhard, F. Endres, W. Burgard and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura, 2012, pp. 573-580.