Toward open-source robotics – ROS use case in industrial and mobile robotics

Nikola Knežević[1], David Seničić[2], Kosta Jovanović[1]
[1] *University of Belgrade – School of Electrical Engineering, Belgrade, Serbia*
[2] *Htec, Belgrade, Serbia*
knezevic@etf.rs, senicic.david@gmail.com, kostaj@etf.rs

Abstract: Many industrial facilities use robots daily, for a variety of tasks. Besides this, service robots are playing an increasing role in our lives, starting from vacuum cleaning robots to robots that serve as hosts in many hotels nowadays or robots that assist people in recovery of injuries. Increasing demand in the market for the number of robots is also occurring in a large number of robot manufacturers. Each of them has its closed architecture and software. In this way, manufacturers provide specified characteristics, but also limiting end-users and force them to use specific software solutions. In the past decade, great effort has been made by the open-source community to make robotics available to a wider range of users. To this end, a Robot Operating System (ROS) was developed. ROS is a meta operating system that provides developers a large number of libraries and tools for easy and fast development of robotic applications. In this paper, the main concepts of ROS and how it can be used in industrial and mobile robots will be presented.

Keywords: open-source; Robot Operating System; robotics; industrial robots; mobile robots.

I. Introduction

In 2017, robot sales increased by 30% to 381,335 units. Robot sales in the automotive industry increased by 22% and remain still the major customer of industrial robots with a share of 33%. The electrical/electronics industry has been catching up with 32% of the total share [1]. Many of these robots use commercial softwares. The reason for this is safety features that robot manufacturers can guarantee by remaining close to end-users. Open-source software today doesn't offer real safety features, and developers/end-users need to develop their safety systems. Without these safety systems, robots can not be part of the factory.

When it comes to service robots, there are a lot of these robots, but still, we aren't seeing many of them in our homes and in our everyday use. This type of robot must deal with the unstructured human environment. If they are capable of dealing with these conditions, they can perform tasks like vacuum cleaning, cooking, or hosting people in hotels [2][3]. A technological breakthrough, especially in the field of artificial intelligence (AI), gives robots a possibility to work in the human's environment [4]. Besides technology readiness, robots need to be cheaper and more open to users in order to be widely adopted.

Nowadays, modern factories work on the popular concept of Smart Industry or Industry 4.0 that focuses on flexible and smart automatization [5].

ROS (Robot Operating System) community is trying to overcome all these obstacles to introduce open-source concepts to robotics [6]. The main concept of ROS, besides being open-source, is modular and reusable software in order to make robot programming accessible to everyone [7]. In this way, ROS promotes flexibility and accelerates deploy time. Because of its good features ROS is a part of academic research for a while. For it to become a part of an industrial environment standardization is very important. ROS community recently released ROS 2.0 that has increased safety and synchronization being one step closer to real-time framework [8]. ROS community is trying to become open-source middleware for robotics systems that will be running in factories, homes, hotels, etc.

In section II, basic ROS tools and libraries for fast prototyping are presented. Section III and Section IV point out an industrial robot and a mobile robot ROS use cases respectively while concluding remarks are given in Section V.

II. ROS tools and packages

ROS provides a set of software libraries and tools to help developers to build robot applications. Since it is an open-source platform, the main goal of the community is to introduce ROS as a standard in robotics. ROS allows researchers and developers to use different programming languages for creating their applications. ROS works with C++, Python, and Lisp. There is a beta version of client libraries that supports Java, C#, R, and other languages.

ROS program is called a node. Nodes can communicate with each other. Communication between ROS nodes is peer-to-peer. There are three methods of communication: through topics, services, and action services.

A. ROS tools

The main tools ROS provides are *rviz*, *rosbag*, *rqt_bag*, *rqt_plot*, *rqt_graph*, *command-line* tool, and other. For 3D visualization *rviz* is used as presented in Figure 1. This tool can let you combine sensor data, robot models, and for example work cell 3D model for a better understanding of the ongoing scenario. For data logging and visualization of sensor data, ROS use *rosabg* and *rqt_bag*. The tool for making plots is *rqt_plot*. By selecting the desired topic, this tool automatically generates a plot from its data. And to see what nodes, topics, and services are running on the system *rqt_graph* is needed. Gazebo software can be used to run physics simulation on the ROS platform.

DOI: 10.5281/zenodo.3466748

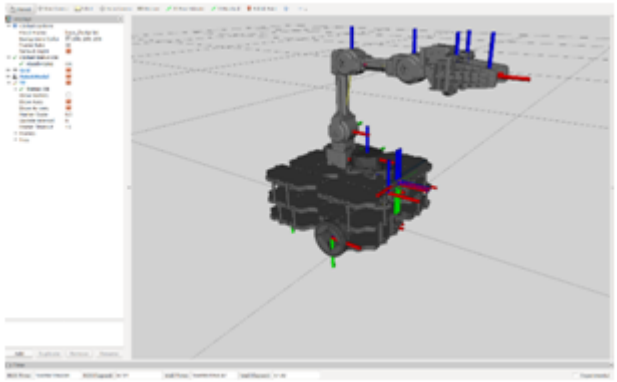

Figure 1: Rviz UI.

The Gazebo provides a necessary interface between simulations and robots. Weall know that in every robotic application good and precise simulations are precious for fast and accurate prototyping and algorithm testing. Also, without simulations, practically most of the AI robotic tasks can not be trained. Therefore, in reinforcement learning, you need to let robot to explore the environment. This action for real robot can lead to robot damage.

A. ROS packages

ROS software is organized in packages. A package might contain nodes, independent libraries, configuration files, third-party software, or anything else that constitutes a useful module. The main goal of organizing software in packages is to provide easy-to-consume and reusable software. ROS packages follow a "Goldilocks" principle: enough functionality to be useful, but not too much that the package is heavy-weight and difficult to use from other software.

In order to control a robot from ROS, appropriate drivers need to be installed. Also, all drivers for any device are organized in packages.

One of the most valuable ROS packages for industrial robots is *ros_control*. The *ros_control* is a set of packages that includes controller interface, controller managers, transmissions, and hardware interface. The package takes as input joint state data form the robot's encoders and an input set point. It uses a generic control loop feedback mechanism to control the output sent to actuators. Also, tracking of coordinate frames is very important in robotics. The package that lets users keep tracking of multiple coordinate frames over time is *tf*. In a robotic system, there are many 3D coordinate frames that change over time, such as a world frame, base frame, gripper frame, etc. This is very important if there is a need, in an application, for calculating pose of the object in robot gripper relative to its base. Or if information about the current pose of the base frame in the map frame is needed.

For mobile robotics, there are a couple of important packages, such as *move_base*, *navigation*, *robot_pose_ekf*, *gmapping*, etc. The *move_base* provides an implementation of the algorithm that, for a given goal in the world, will try to reach it with a mobile robot base. The *navigation* package combines information from odometry, sensor streams, and tries to compute safe velocity commands. These commands use as input to *the move_base* package. Because of noise, slippage, and incorrect modeling of robot geometry odometry might not give a correct robot position. The *robot_pose_ekf* package applies Extended Kalman Filtrar (EKF) to sensor data to improve "the belief" of robot position. In the end, the robot needs to know the environment and use *gmapping* package for map building. This package uses OpenSlam Gmapping algorithm for mapping. As the output from this package user gets 2-D occupation grid map.

Most robotic applications have demands for image processing. In order to provide a real-time computer vision, *vision_opencv* package is implemented for ROS. This package provides a popular OpenCV library for ROS. Also, nowadays AI algorithms are common part of robotics applications. ROS provides *openai_ros* package with complete infrastructure for Reinforcement Learning. This package can execute learning algorithms using Gazebo simulator in order to collect data necessary for learning algorithms.

III. Industrial use case for ROS

Industrial robotics has always been tied to manufacturer of industrial robots. Robotics solutions were primarily implemented via the framework provided by manufacturers. System integrators, main users of these frameworks, made programs per clients specifications. ROS - Industrial, ROS - I for short, offered open-source based framework with the idea to replace the manufacturer's one. Main idea is to create a framework that should work on all industrial robots regardless of size and type of the controller. Additional support for this idea came from research that showed the main focus for industrial robotics in the last 24 years was welding and material handling [9]. Figure 2 shows distribution from research.

Industry lacked flexibility, an adaptation of new technologies, lacking vision systems, and connectivity to Industry 4.0. Additional problem for ROS came in a form of standardization and real-time ROS for control. All of these problems will be solved from ground up in ROS2 which will strongly emphasize on real-time part of robot operations.

The use case for ROS in the industry is presented through H2020 project RAMPup [10]. The main idea of

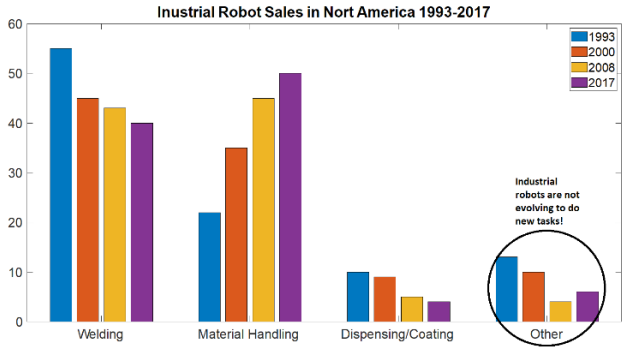

Figure 2: Industrie use of robots.

this 4-year project is to develop complete solution for common industrial tasks, screwing, riveting, gripping, and gluing for example. Each of these tasks will be constructed as a module. This included hardware as well as software and it needs to be ROS centered. An additional task is to enable easy to use and safe programming for the robot with this RAMPup module. Main users and testers will be SME which have small batches, 50 - 1000 units per batch, of a similar product and need to reprogram industrial robot with RAMPup module on a weekly basis. This will empower SMEs to use industrial robots as easy to use and safe tools to accelerate production and improve quality. The subtask of this project is a force-based insertion. Robot task is to pick up an object on a known location and carry it to the area of insertion. The area is roughly known. Robot task is to probe the area with object and measure force and torques to get an idea about the hole location. The second stage is to guide object into a hole. The object can have multiple pegs for multiple holes. A task can be opposite, the object can have holes and needs to be fitted onto multiple pegs. This was a task for SME which tested this module.

Future goals of the force based were to:

1) Create plug-and-play module that provides force based insertion which will be tested in real factory production line.

2) Modul should be agnostic toward type of robot as well as type of force-torque sensors attached to robot flange. Configurations should be available and easy to use. This is the main power of ROS.

3) Simulation of work should be available before testing on real robot workcell. This is valuable for testing real-life simulations of workcell, debugging possible solutions, and testing how force sensor should interact.

4) End users can program and modify robot's task enabling modularity.

5) Modul should be independent of its hardware as much as possible. If user needs different piece of equipment, for example larger gripper. Software side is constructed for modularity in mind.

ROS proved to be a useful tool in handling different sensors. Module main sensing of force and torque came from force and torque sensor attached to the robot end flange. Three industrial grade sensors were used, Robotiq FT300, Optoforce HEX – E model, and ATI sensor Delta SI 330 – 30. Robotic sensor had ROS enabled driver by manufacturer, Optoforce and Delta had their driver written by community for community. Because ROS is built upon nodes, changes were needed to be done only in one node that published on topics /sensor/force and /sensor/torque. The second thing that needed change is correction in orientation of coordinate system of force sensor as well new dimensions of end tool point. ROS provided drivers to communicate to KUKA controller which controlled KUKA Agilus KR10. This communication provided critical information about robot current status, for example internal coordinates of robot motors. Sending points of trajectory was also done via this open-source experimental ROS KUKA packet.

ROS2 will be the next step towards real-time constraint systems and industrial standard for the future.

- Links 1 and 2 are video demonstration of RAMPup current progress, and Link 3 provides demonstration of enforcement learning.

- Link 1 - basic functionality
- Link 2 - RAMPup demonstrations
- Link 3 - Demonstration of enforcement Learning on High-Precision Assembly Tasks with ROS - Industrial

ROS-I needs to become a stable base for research as well as industry so that use of industrial robots can be offloaded to ROS and only solving one problem. Open-source community had many issues because lack of availability of industrial robots. ROS-I is collaborating with robot manufacturers to enable simulating robot as much as possible.

IV. Mobile robots use case for ROS

The mobile robotics use case is closely related to an ongoing course at the University of Belgrade, School of Electrical Engineering on Autonomous Mobile Robots. In this course, students learn how to develop their nodes and how to exchange data through topics. Also, they learn how to control differential drive robot, how to use lidar data to detect walls, and how to implement EKF for correction of localization estimation. For this course TurtleBot3 is used, presented in Figure 3. Designed by ROBOTIS, this robot uses as an educational platform for learning ROS and Autonomous Mobile Robot basic principles.

TurtleBot3 is a differential drive robot equipped with cost-effective and small-size Single Board Computer (SBC) that is suitable for the robust embedded system, lidar sensor, and 3D printed technology. The core technologies are SLAM, Navigation, and Manipulation, making this robot suitable for home service tasks. With appropriate ROS packages, this robot can be controlled using PC, joypad, gamepad, and any wifi or Bluetooth controller. Because of the modularity of ROS packages and the topic principle of communication, this is possible.

Firstly, students need to establish communication between the robot and their PC. Applying the correct configuration of the network parameters, communication can be automatically established. That means if any of the nodes is publishing some data, collecting that data can be performed from any device that is part of ROS network. Using one command, ROS allows running every service that will bring up the robot. At that moment, users can collect data from the actuator encoders, a point cloud of lidar data, IMU data, and robot status information.

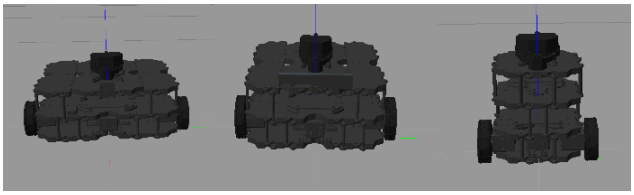

Figure 3: TurtleBot 3, left: Burger, center: Waffle, right: Waffle PI.

Also, the user can send commands to the robot to manipulate them. The second thing that students need to learn is how to develop an algorithm for moving robot around. Using proper drivers, there is no need to worry about motor control or to think about how communication works. Students only need to focus on the algorithm that they develop.

Following link demonstrates TurtleBot3 performing SLAM: Link 4 – TurtleBot3 SLAM.

If you need to focus on research of a new algorithm for robot control or testing algorithm for processing of point cloud data, the ROS packages provide an interface for your robotics project.

V. Conclusion

This paper presented how ROS can lead to open-source robotics. Although, ROS is not on a level like a commercial manufacturer, because of safety standards and real-time features, the ROS community is making efforts to overcome these problems. This paper also presented basic tools and packages that can accelerate the process of developing and deploying. Use case for industrial robots shown that ROS poses capabilities for running applications in which a force/torque sensor integration. In the case of mobile robots, it has been shown that students can easily learn basic postulates with ROS. Using ROS packages developing specific algorithms becomes an independent task.

ROS community is showing that in the future ROS might become a robotic standard for industrial and mobile robots. Also, they are making efforts to introduce robotic programming as easy to learn and modular.

References

[1] https://ifr.org/downloads/press2018/Executive_Summary_WR_2018_Industrial_Robots.pdf [Assessed on Sep. 15, 2019]

[2] J. Forlizzi and C. DiSalvo, "Service robots in the domestic environment: a study of the roomba vacuum in the home," in *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction (HRI '06)*, New York, NY, USA, 2006, pp. 258-265, doi: https://doi.org/10.1145/1121241.1121286.

[3] I. P. Tussyadiah, and S. Park, "Consumer Evaluation of Hotel Service Robots," Information and Communication Technologies in Tourism pp. 308–320, 2018, doi: https://doi.org/10.1007/978-3-319-72923-7_24.

[4] W. Burgard, M. Moors, D. Fox, R. Simmons and S. Thrun, "Collaborative multi-robot exploration," in *Proceedings of the IEEE International Conference on Robotics and Automation ICRA*, San Francisco, CA, USA, 2000, pp. 476-781, doi: https://doi.org/10.1109/ROBOT.2000.844100.

[5] M. A. K. Bahrin, M. F. Othman, N. N. Azli and M. F. Talib, "Industry 4.0: A review on industrial automation and robotic," Jurnal Teknologi, vol. 78, no. 6-13, pp. 137-143, 2016.

[6] http://wiki.ros.org/ [Assessed on Sep. 15, 2019]

[7] J. Kerr and K. Nickels, "Robot operating systems: Bridging the gap between human and robot," in *Proceedings of the 44th Southeastern Symposium on System Theory (SSST)*, Jacksonville, FL, USA, 2012, pp. 99-104, doi: https://doi.org/10.1109/SSST.2012.6195127.

[8] https://index.ros.org/doc/ros2/ [Assessed on Sep. 15, 2019]

[9] https://rosindustrial.org/the-challenge [Assessed on Sep. 15.2019]

[10] https://rampupproject.eu/ [Assessed on Sep. 15, 2019]