Why should we use a Robot Software Platform?

Why should we learn ROS, which is the new concept of robot software platform? This is a frequently asked question at offline ROS seminars. The short answer is because it can reduce development time. Often people say they do not want to spare their time learning a new concept and would rather stick to their current methods to avoid changing the already built system or existing programs. However, ROS does not require one to develop the existing system and programs all over again, but can rather easily turn a non-ROS system to a ROS-system by simply inserting a few standardized codes. In addition, ROS provides various tools and software that are commonly used, and it allows users to focus on the features that they are interested in or would like to contribute in, which ultimately reduces the development and maintenance time. Let us look at the five main characteristics of ROS.

First is the reusability of the program. A user can focus on the feature that the user would like to develop, and download the corresponding package for the remaining functions. At the same time, they can share the program that they developed so that others can reuse it. As an example, it is said that for NASA to control their robot Robonaut29 used in the International Space Station, they not only used programs developed in-house but also used ROS, which provides various drivers for multi-platforms, and OROCOS, which supports real-time control, message communication restoration and reliability, in order to accomplish their mission in outer space. The Robotbase above is another example of thoroughly implemented reusable programs.

Second is that ROS is a communication-based program. Often, in order to provide a service, programs such as hardware drivers for sensors and actuators and features such as sensing, recognition and operating are developed in a single frame. However, in order to achieve the reusability of robot software, each program and feature is divided into smaller pieces based on its function. This is called componentization or modularization according to the platform. Data should be exchanged among nodes(a process that performs computation in ROS) that are divided into units of minimal functions, and platforms have all necessary information for exchanging of data among nodes. The network programming, which is greatly useful in remote control, becomes possible when communication among node is based on network so that nodes are not restricted by hardware. The concept of network connected minimal functions is also applied to Internet of Things (IOT), so ROS can replace the IoT platforms. It is remarkably useful for finding errors because

programs that are divided into minimal functions can be debugged separately.

Third is the support of development tools. ROS provides debugging tools, 2D visualization tool (rqt, rqt is a software framework of ROS that implements the various GUI tools in the form of plugins) and 3D visualization tool (RViz) that can be used without developing necessary tools for robot development. For example, there are many occasions where a robot model needs to be visualized while developing a robot. Simply matching the predefined message format allows users to not only check the robot's model directly, but also perform a simulation using the provided 3D simulator(Gazebo). The tool can also receive 3D distance information from recently spotlighted Intel RealSense or Microsoft Kinect and easily convert them into the form of point cloud, and display them on the visualization tool. Apart from this, it can also record data acquired during experiments and replay them whenever it is necessary to recreate the exact experiment environment. As shown above, one of the most important characteristics of ROS is that it provides software tools necessary for robot development, which maximizes the convenience of development.

Fourth is the active community. The robot academic world and industry that have been relatively closed until now are changing in the direction of emphasizing collaboration as a result of these previously mentioned functions. Regardless of the difference in individual objectives, collaboration through these software platforms is actually occurring. At the center of this change, there is a community for open source software platform. In case of ROS, there are over 5,000 packages that have been voluntarily developed and shared as of 2017, and the Wiki pages that explain their packages are exceeding 18,000 pages by the contribution of individual users. Moreover, another critical part of the community which is the Q&A has exceeded 36,000 posts, creating a collaboratively growing community. The community goes beyond discussing the instructions, and into finding necessary components of robotics software and creating regulations thereof. Furthermore, this is progressing to a state where users come together and think of what robot software should entail for the advancement of robotics and collaborate in order to fill the missing pieces in the puzzle.

Fifth is the formation of the ecosystem. The previously mentioned smartphone platform revolution is said to have occurred because there was an ecosystem that was created by software platforms such as Android or iOS. This type of progression is likewise underway for the robotic field. In the beginning, every kind of hardware technology was overflowing, but there was no operating system to integrate them. Various software platforms have developed and the most esteemed

platform among them, ROS, is now shaping its ecosystem. It is creating an ecosystem that everyone — hardware developers from the robotic field such as robot and sensor companies, ROS development operational team, application software developers, and users — can be happy with it. Although the beginning may yet be marginal, when looking at the increasing number of users and robot-related companies and the surge of related tools and libraries, we can anticipate a lively ecosystem in the near future.

ROBOTIS Senior Research Engineer
YoonSeok Pyo