Robot Operating System (ROS) Research Project

Rotman Daniel Leiva

Artificial Intelligence Lab, University of Rhode Island

Introduction

Working at the Artificial Intelligence Lab at the University of Rhode Island, I was given the opportunity to do research on ROS and robots. The main objective of my research project was to learn about ROS, it's applications in robotics, and also experiment with it to apply the knowledge gained to do cool robot stuff. ROS can be a little complicated to learn at the beginning but eventually everything comes together and there are a lot of cool things that can be done with it. For this project I managed to use a pre-existing open-source package to control a robot via voice commands.

What is ROS?

ROS started in Silicon Valley in 2007 at a company named Willow Garage. Since then, it has expanded into an international community of dedicated developers of all levels. The Robot Operating System (ROS) as it is known today, is a set of software libraries and tools that help developers build robot applications. From drivers to state-of-the-art algorithms, and with powerful developer tools, ROS has what you need for your next robotics project. And it's all open source. [2]

ROS is available for Linux and it has released many versioned packages since 2010. The most up-to-date package is Melodic which was released in 2018, but many developers are still using the Kinetic package which was released in 2016.

Basic Components

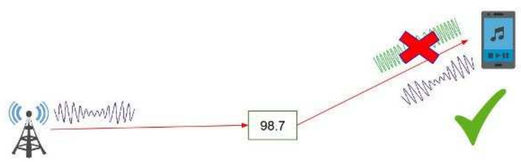
- **Nodes:** ROS is composed of packages that contain nodes which are programs that will give functionality to a robot. Nodes can be written in C++ or Python.
- **Communications Tools:** Tools There are three main communication tools in order to communicate between nodes
  - **Topics:** Used to send data between nodes.
  - **Services:** Used to create a simple synchronous client/server communication between nodes.
  - **Actions:** They are based on topics and provide asynchronous client/server architecture. [1]
- **Messages:** Messages are the data being flowing between nodes using the communication tools mentioned above.

More on Topics

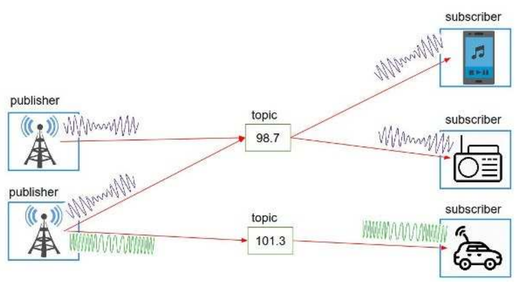
Topics have specific message types. And these message types can be received by or sent to a topic. The way a node receives a message is by subscribing to a topic with a **subscriber** of the same message type. And the way a node sends a message is by publishing a topic with a **publisher** of the same message type as well.

The way roboticsbackend[1] describes this is like a radio transmitter is the publisher transmitting data to a specific radio frequency which you can think of as the topic.

Now, imagine you want to listen to your favorite song on your favorite radio station, in this case 98.7, then you would have to specify the radio frequency in order to receive the radio waves to your device of choice. This analogy equates to a node subscribing to a topic published by another node.



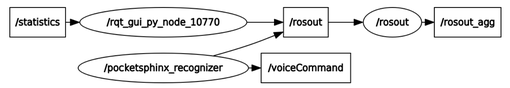
It is also possible to have multiple nodes publishing to the same topic as well as multiple nodes subscribed to the same topic. Nodes can also publish and subscribe to more than one topic, so the possibilities are endless. This is one of the features that makes ROS so powerful.



Research Project

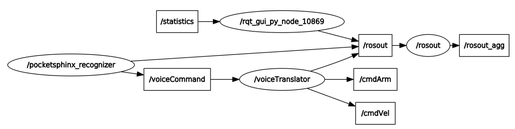
ROS has many libraries and packages available for anyone to use at https://index.ros.org/packages/. One interesting one that was suggested to me by Dr. Mandal was a modified ROS wrapper for pocketsphinx used for voice recognition by University of Texas Nuclear Robotics Group. This package has a list of voice commands that can be recognized using this package. Since we had a keyboard controllable working robot, the goal was to use this voice recognition package to give our robot ears.

Here we can see the pocketsphinx_recognizer node publishing to the voiceCommand topic

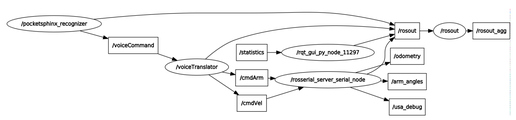


After adding the voice recognition package to the robot we needed a way to connect the voice recognition node to the robot controls, so it was time to create a new node to handle the messages or voice commands from our voice recognition node. Our new node would take in our voice command messages as strings and convert them into motion command messages that our robot would understand.

Here we can see the voice recognition node and voice translation node coming together via the voiceCommand topic.



Finally, to put it all together from our voiceTranslator node we publish the translated voice command to the cmdVel topic which then sends the desired velocity to the ros serial server which controls the motors of our robot.



Results

Given the time constraints and my lack of expertise with ROS I was able to achieve decent results with this project. Due to the limited number of voice commands recognized by the voice recognition package, I was only able to get three voice commands working. The commands are; forward, backward, and stop. There are ways to add more commands and which can be single words or combinations of words. Another not so good byproduct was that when I run the voice translator node, the robot glitches and shakes. I believe this may be to an overloading of messages being sent from the voice translator node. Overall, we managed to achieve our goal and that is great progress.

Future Work

Future improvements could be the addition of more voice commands to the voice recognition package as well as a more efficient way of passing messages. At the moment the voice translator node goes from string to char to float. Perhaps there is a better way to go from string to float directly. Another improvement might be better motion control through voice commands as well as arm control voice commands. The glitching issue would definitely be a much needed improvement. Addition of other packages such as computer vision or path finding AI would be great improvements also.

References

[1] The Robotics Back-End. The Robotics Back-End program robots like a boss, 2020.

[2] Open Source Robotics Foundation. ROS robot operating system, 2020.

Contact information: Rotman Daniel Leiva, 9 Greenhouse Road, Suite 2, Department Computer Science and Statistics, University of Rhode Island, Kingston, RI 02881-2018 – Phone: 401–874–2672 – Email: *rdleiva@my.uri.edu*; Web: *https://web.uri.edu/ai/*