

TurtleBot2 People Follower

Progress Report

Drake Melancon

Jacob Mitchell

Madelyn Hutchinson

Donovan Groschang (team leader)

Engineering Department

Louisiana State University

5 November 2019

Content

Introduction	2
I. TurtleBot2	2
II. Robot Operating System (ROS)	2
1) Definition and advantages	2
2) Communication architecture	2
III. Timeline	3
1) Methods and progress this far	3
2) Future goals and potential methods	3
3) Gantt Diagram (Annex 1)	3
IV. Workload	4
V. References	4
Annex 1: Gantt Diagram	5

Introduction

Our project consists in the improvement of an existing algorithm allowing the robot TurtleBot2 to detect and follow peoples.

In order to achieve this, we will have to understand what the features of TurtleBot2 are and which one are we going to use. We will also have to deeply understand how the Robot Operating System (ROS) works and how can we use it to interact with the turtleBot2. Finally, we will have to implement the existing template, identify what we could improve and implement those changes in the code.

This report establishes our progress and understanding of the subject since the beginning of the project and shouldn't be considered are representative of the whole project.

I. TurtleBot2

TurtleBot 2 is a popular low cost and open source robot for education and research [1]. It is equipped with a Kobuki robot base, a dual-core netbook, Orbbec Astra Pro Sensor and a gyroscope [2]. It also possesses bumper sensor allow him to detect any collision with its environment.

The TurtleBot2 has been designed to function with ROS (see next section) and profit of a large panel of open-source tutorials to help us better understanding it and build our project easier.

For our specific project, we should have to use the Astra Pro sensor (camera and distance sensor) as well as the bumper sensors to prevent the robot to hurt someone, another robot, or himself according to Asimov's laws of robotics.

II. Robot Operating System (ROS)

1) Definition and advantages

ROS is an open-source flexible framework designed for writing software for robots. ROS aims to simplify the tall task of making intricate and robust robot behavior across a wide array of robotic platforms [3]. ROS encourages collaborative robotics software development for different kinds of robots with different purposes and provides a uniformity across the vast field of robotics.

ROS provides hardware abstraction, device control and common functionalities used in robotics. It also provides software high-level abstraction as well as low-level interaction if needed. Finally, it allows between robots and between robots and computers [4].

2) Communication architecture

ROS provides a flexible communication architecture. ROS processes are called nodes. Each node can communicate with other nodes via topics. Connection between are managed by a master follow the following process:

- A first node (publisher) warn the master that it has a data to share
- A second node (subscriber) warn the master that it wishes to have access to a certain data
- The master creates a connection between the 2 nodes through a topic
- The publisher publishes the data to the topic which is redirected to the subscriber who subscribe to that topic

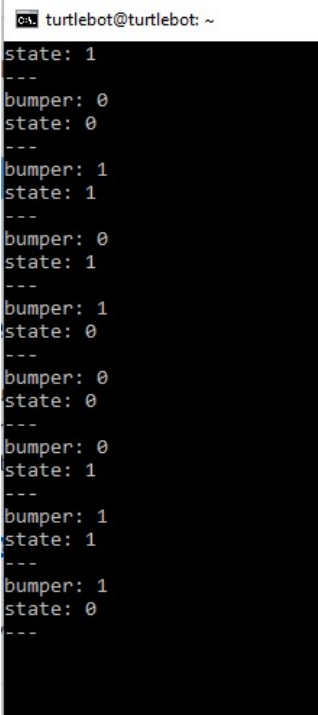
Messages sent to the topics are for the most standardized which makes the system very flexible.

ROS allows communication between machines. Nodes running on different machines but being are of the same master can transparently communicate.

III. Timeline

1) Methods and progress this far

Our progress thus far has been focused on initializing our robot with the Robot Operating System. It was important for our team members to familiarize themselves with the ROS wiki page and how the ROS system will allow us to accomplish our goal. This includes initializing our robot drivers and allowing our robot to communicate with a separate PC (via a SSH connection) to allow us to code more comfortably. We then tested our connection by testing the bumper sensors and at the same time make sure that those ones were working properly (Figure 1). This working then allowed us to install and run the template for people following bot on the ROS wiki page.



```
turtlebot@turtlebot: ~
state: 1
---
bumper: 0
state: 0
---
bumper: 1
state: 1
---
bumper: 0
state: 1
---
bumper: 1
state: 0
---
bumper: 0
state: 0
---
bumper: 0
state: 1
---
bumper: 1
state: 1
---
bumper: 1
state: 0
---
```

Figure 1: SSH connection and bumper testing

2) Future goals and potential methods

We will be making edits into the camera code as well as planning on altering that part of the code so that the robot more effectively will follow humans/objects. This includes all color and distance aspects of the code that are involved with the camera. We will ultimately edit the code again and again until the robot is the perfect people/object follower.

3) Gantt Diagram (Annex 1)

IV. Workload

- Drake: 20% (tutorials, report, Gantt)
- Jacob: 20% (tutorials, report, presentation, Gantt)
- Madelyn: 20% (tutorials, report, presentation)
- Donovan: 40% (code, tutorials, camera issues, report, presentation, Gantt)

V. References

1. “TurtleBot 2 - Open source personal research robot,” *Clearpath Robotics*. [Online]. Available: <https://clearpathrobotics.com/turtlebot-2-open-source-robot/>. [Accessed: 05-Nov-2019].
2. “Astra Series,” *Orbbec*, 26-Feb-2019. [Online]. Available: <https://orbbec3d.com/product-astra-pro/>. [Accessed: 05-Nov-2019].
3. “About ROS,” *ROS.org*. [Online]. Available: <https://www.ros.org/about-ros/>. [Accessed: 05-Nov-2019].
4. “Robot Operating System,” *Wikipedia*, 14-Oct-2019. [Online]. Available: https://en.wikipedia.org/wiki/Robot_Operating_System. [Accessed: 05-Nov-2019].
5. “Wiki,” *ros.org*. [Online]. Available: <http://wiki.ros.org/Robots/TurtleBot>. [Accessed: 05-Nov-2019].
6. V. Mazzari and V. Mazzari, “ROS - Robot Operating System,” *Génération Robots - Blog*, 26-Jul-2016. [Online]. Available: <https://www.generationrobots.com/blog/en/ros-robot-operating-system-2/>. [Accessed: 05-Nov-2019].

Annex 1: Gantt Diagram

