Individual Lab Report

Bikramjot Singh Hanzra
Team B – Auto Pirates
Teammates – Tushar Chugh, Shiyu Dong, Tae-Hyung Kim, William Seto
ILR11

April 13, 2016

Contents

List of Figures

1 Introduction

During the past couple of weeks, the team concentrated on 3 issues –

- Elliptical Inflation of the obstacles.
- Developing a GUI so that a user can select a destination as per the use case of the project.
- Fixing minor issues with the path planning algorithm.

I worked on adding synthesized voice messages to inform the user about the events that happen using the journey to improve the user experience.

2 Implementation

To generate the synthesized voice messages, I used the `sound_play` package [1] that is available on ROS website. The `sound_play` package provides convenient functionality that translate messages published on the `robotsound` topic to sounds that are played on the speakers. The speakers need to be configured to play sounds using the `sound_play` package.

The `sound_play` package does speech synthesis using `festival` speech synthesis system. I used the python bindings of the `sound_play` package to synthesize the voice messages.

Voice commands can be synthesized in 2 ways –

- By specifying a string that needs to be synthesized.
- By specify a full path to an audio file that needs to be synthesized. The audio files that are supported are `.ogg` and `.wav`.

In the python bindings, the `sound_play` module provides access to the `SoundClient()` class that gives access to a number of functions that can be used to publish messages on the `robotsound` topic. As pointed before, `SoundClient()` prefers all sound assets to be `.wav` or `.ogg` files. `SoundClient()` may produce a segmentation fault if too many sound commands are given in a short period of time, so sound commands are published on the `SoundClient()` node after some delay. The `SoundClient().voiceSound` member function is used to publish a string on the `robotsound` topic and the `SoundClient().waveSound` member function is used to publish the content of an audio file on the `robotsound` topic.

The sound is played from the speakers by the `soundplay_node.py` node of the `sound_play` package that subscribes to the `robotsound` topic. If this node is not started before sending the voice messages, the node that publishes the voice messages will give an error message.

Currently the below messages are synthesized based on the location and the other events that happen during the journey –

- Taxi is starting from NREC and heading to South Side.
- Taxi is starting from NREC and heading to PNC Park.
- Taxi is starting from PNC Park and heading to South Side.
- Taxi is starting from PNC Park and heading to NREC.

- Taxi is starting from South Side and heading to PNC Park.
- Taxi is starting from South Side and heading to NREC.
- Taxi is starting from <GPS cooridinates> and heading to South Side.
- Destination Reached. Have a nice day !!

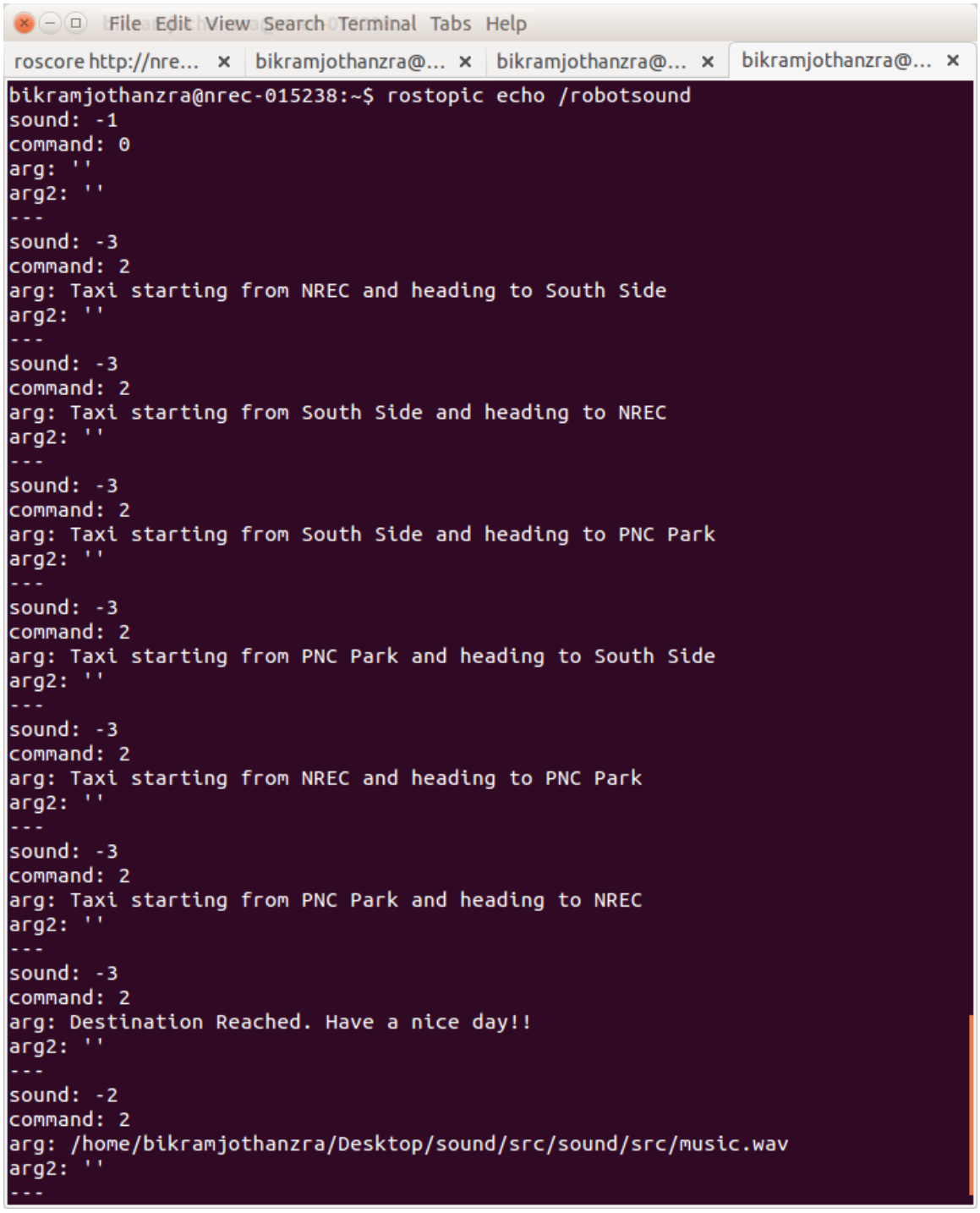

Figure 1: Figure shows messages published on the `robotsound` topic.

Figure 1 shows the messages published on the `robotsound` topic. The first message published is a blank message. After that the actual messages are published. The `soundplay_node.py` node differentiates between an audio file and a string based the value of the `sound` variable in the message. It is set to -3 for a string and to -2 for an audio file.

3 Challenges

The main challenge that I faced while synthesizing the voice messages was that the documentation was pretty terse and I had to spend a fair amount of time in figuring out the right functions to use. Also, there were a few bugs in the python scripts which I had to remove to be able to run the these scripts.

One of the downside of the `sound_play` package is that after publishing messages on the `robotsound` topic, there is no way to tell when a sound stops playing. Also, by default there is no queue where a published message to the `robotsound` topic is appended. This leads to an undesirable situation where a new message interrupts a previous message that is being played. In order to deal with this issue, I had to added a delay between 2 successive messages to be published at the publisher side. This solutions did solve the problem, but the delay has to be set manually for each string or audio file based on the time that it will take to finish playing the message which has to set differently for each message.

4 Teamwork

The work done by the rest of the team members is discuss below –

- **William Seto** – William worked on inflating the cost of the obstacles using elliptical dilation. He used the OpenCV library [2] to inflate the cost of the obstacles.
- **Tushar Chugh** – Tushar worked on fixing the minor bugs in the path planning code. He added the code to stop the boat when it reaches the destination and tweaked the planner code to skip a few waypoints.
- **Shiyu Dong** – Shiyu worked on integrating the GUI with the rest of the code. He added the functionality to display the velocity of the boat and the routes that the user can select to the GUI.
- **Tae-Hyung Kim** – Tae-Hyung worked on analyzing the rosbag files that he recorded during the last field test using a package from Novatel.

5 Work Overview for Coming Week

We will be going for a field test tomorrow (04/14/2016) where we will be testing the GUI, the elliptical cost inflation and the fixes in the planning code. This will be possibly be our last field test before the SVE and it is important that we test everything properly. Also, we may possibly render Google Maps in RViz to make it aesthetically pleasing. Our last field test went really well and we are confident that after this field test, we will have a nice product to show at the SVE.

References

[1] ROS `sound_play` package
`http://wiki.ros.org/sound_play`

[2] OpenCV library
`http://opencv.org`