Professor: Wing-Yue Geoffrey Louie

Date: April 20, 2020

**ECE5532 Final project: IGVC CoursE**

Tkac, Sam

samtkac@oakland.edu

Grudzien, Benjamin

bgrudzien@oakland.edu

**Abstract**

The ROS Navigation Stack was used to autonomously navigate a differential drive robot through the IGVC course. To achieve this the “Move Base” parameters of the ROS Navigation stack were adjusted. The ROS Navigation stack utilizes simulated Lidar sensor data which allow the robot to autonomously drive. Also, a concise algorithm was implemented which fed the differential drive robot the goal waypoint locations for it to drive to. The differential drive robot was able to successfully complete the basic IGVC courses and the advanced level IGVC courses.

**Table of Contents**

[**Introduction** 1](#_Toc38121071)

[**Differential Drive Kinematics** 2](#_Toc38121072)

[**ROS Navigation Stack Setup** 3](#_Toc38121073)

[**Algorithm** 4](#_Toc38121074)

[**Implementation** 5](#_Toc38121075)

[**Results** 6](#_Toc38121076)

[**Honorable Mention** 6](#_Toc38121077)

[**Conclusions** 6](#_Toc38121078)

[**References** 6](#_Toc38121079)

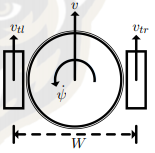
**Instructions:** CTRL + Click heading to jump to that section of the report.

# **Introduction**

At a high level, this report documents the steps taken and the required components necessary for completion the IGVC course. This report covers the kinematics for a differential drive robot, how the ROS Navigation Stack was utilized to autonomously avoid obstacles, it covers how an algorithm was implemented to give the robot goal waypoints to drive to, and it provides supplemental figures and descriptions of how the overall system was implemented in ROS.

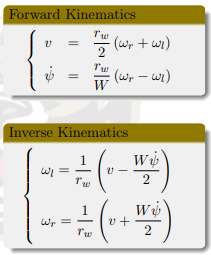
# **Differential Drive Kinematics**

The robot kinematics are based on the differential drive kinematics of the Roundbot. It is important to note that the ROS Navigation Stack move base parameters had a default “Roundbot Local Planner.” This parameter implemented the kinematics. Therefore, the kinematics did not need to be manually implemented. For reference, the Roundbot is shown and labeled in Figure 1.



**Figure 1:** The Roundbot

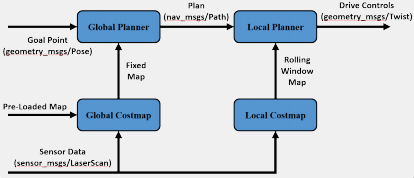
The Roundbot kinematics are controlled using forward kinematics set the velocity of this robot. In ROS, the Roundbot movement commands are determined by speed command in the “linear x” direction and in the “angular z” direction. The Roundbot drives each wheel independently. The forward and inverse kinematics of the Roundbot are shown in Figure 2.



**Figure 2:** Roundbot kinematics

# **ROS Navigation Stack Setup**

The ROS Navigation Stack enables the robot to autonomously avoid obstacles on the way to its goal using move base parameters. Move base parameters are implemented in the launch file. These parameters are instantiated using YAML files. The YAML files develop the global and local costmaps for the robot as well as a global and local planner. The global planner receives the desired waypoint and the global costmap. The local planner receives the desired path from the global planner and the local costmap. The global costmap refers to the boundaries of the entire course before the robot traverses it. The local costmap updates in real time based on the Lidar sensors obstacle detection. A flow chart of this interaction is shown in Figure 3. Figure 4 shows our launch file.



**Figure 3:** The ROS Navigation Stack



**Figure 4:** The launch file for basic course north

This launch file first generates the desired course in RViz and instantiates the starting position of the robot. Then the move base parameters are initialized. The move base parameters refer to YAML files in the project package. The YAML file parameters determine the size of the costmaps and designated cost relative to certain situations. Specifically, the costmaps size is dependent on the robot radius and inflation parameters. They are shown in Figure 5 and Figure 6.

**Figure 5:** YAML screenshot of robot radius

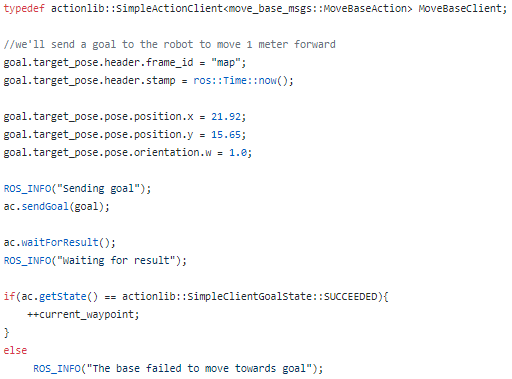
**Figure 6:** YAML screenshot of inflation

The robot wants to take the lowest cost path to its navigation goal. The robot radius determines the highest cost areas of the costmap which is designated with a blue color. The inflation parameter adds additional cost to the costmap, so the robot doesn’t get to close to obstacles. This is designated in red. The cost scaling parameter is an exponential decay factor that adjusts the cost of the inflation radius.

Finally, the launch file localizes the frames by transforming the world frame to the map frame. It runs the map server node to generate the global costmaps, runs the waypoint algorithm, and launches a specific RViz configuration.

# **Algorithm**

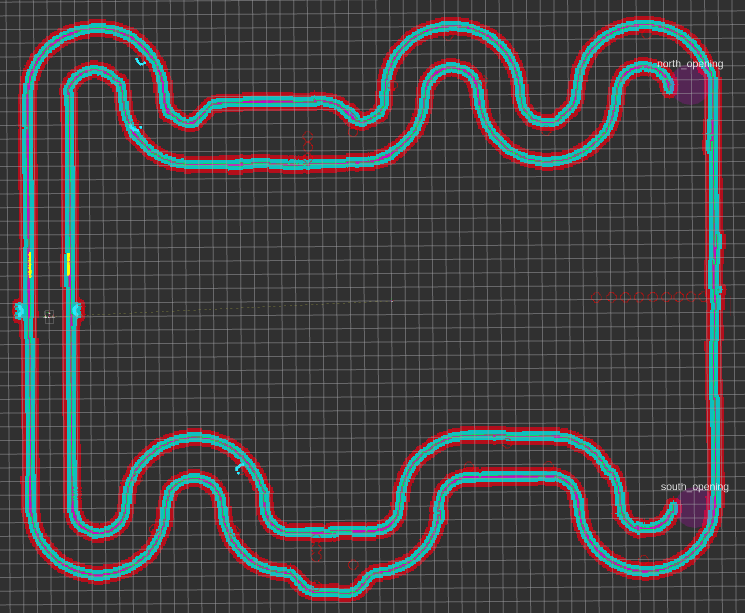
The algorithm utilizes the actionlib package to publish a move base goal to the robot. This works by instantiating a move base goal using the action client and then monitoring the action client’s status relative to that move base goal. The move base goals are then updated using a case statement to set the next goal once the previous goal has been hit. A screenshot of this is shown in Figure 7.



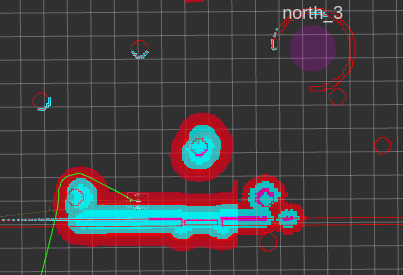
**Figure 7:** Screenshot of the algorithm for the “Basic Course North”

# **Implementation**

This section analyzes how the ROS Navigation Stack and algorithm were implemented to navigate the Roundbot through the IGVC course. A screenshot of the global costmaps are shown in Figure 8. The local costmaps are shown in Figure 9. The green line in the figures shows the robots plan to the goal based on the costmaps.

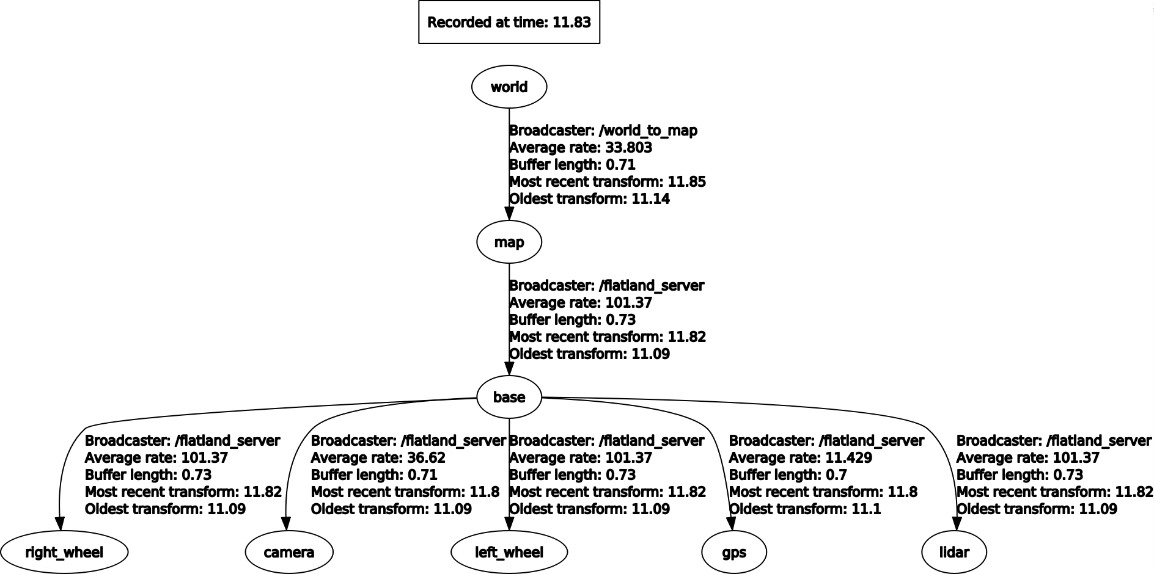


**Figure 8:** Global costmaps for the Advanced Course North

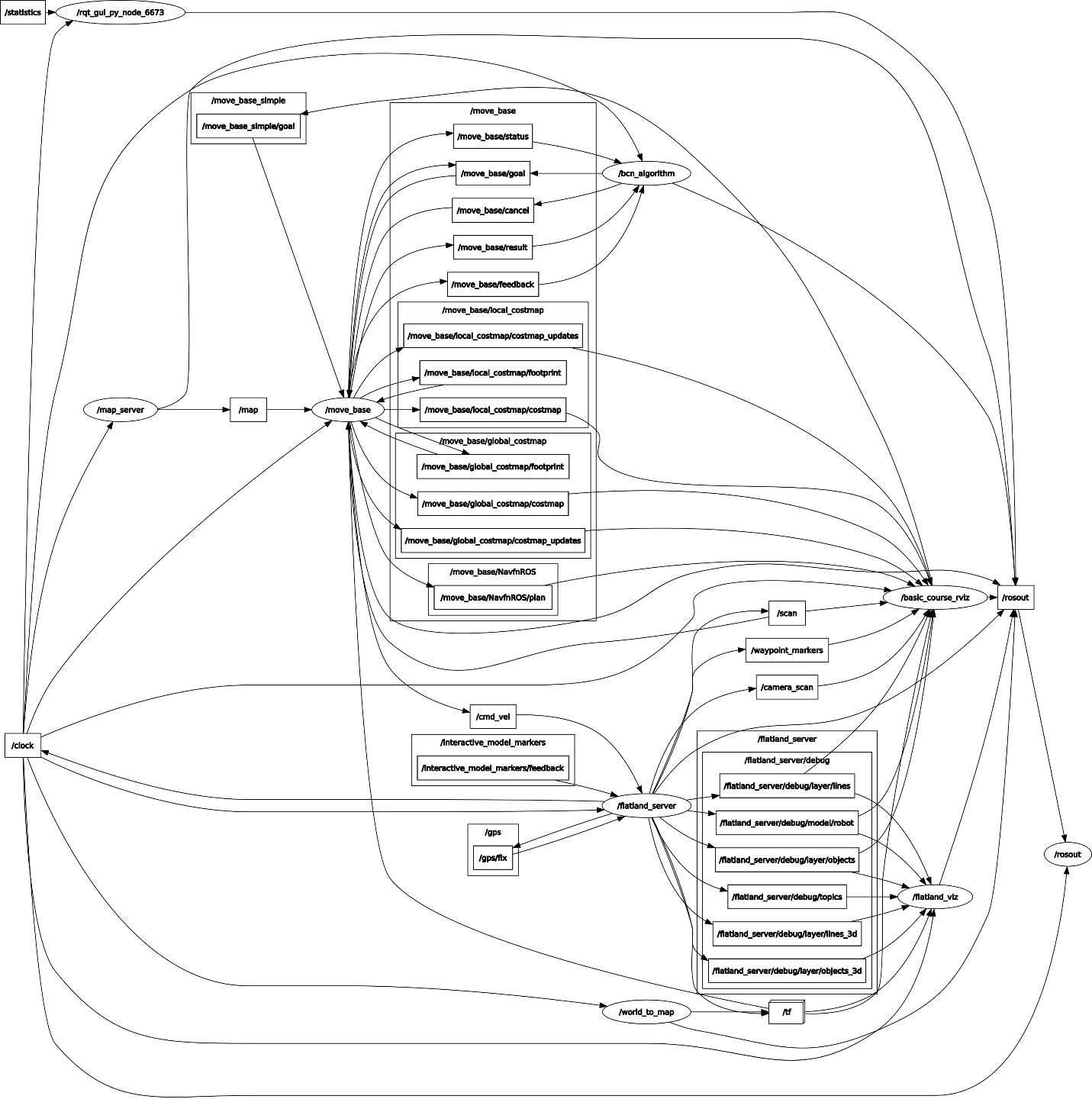


**Figure 9:** Local costmaps

The transformation tree is shown in Figure 10. This shows how the world frame is localized to the map frame of the simulation. It also shows how the base frame of the robot relates to its wheels and the sensors. There are two lidar sensors on this robot, one to detect obstacles and one that detects the lanes of the course. Figure 11 shows the rqt graph which demonstrates how the topics are communication with each other and with the algorithm.



**Figure 10:** The transformation tree



**Figure 11:** The rqt graph for advance course north

# **Results**

This system’s utilization of the ROS Navigation Stack and the algorithm were able to complete the advanced and basic IGVC courses. Figure 12 is a video demonstrating this.

**Figure 11:** Embed the video here?

# **Honorable Mention**

R3D3? We could make this part short or even omit it if we go over 10 pages after adding the rest of the figures and revising the report

# **Conclusions**

In conclusion, we were successfully able to complete the IGVC basic and advanced course using ROS. It was not perfect however, as the ROS Navigation Stack parameters do successfully complete the course 100% of the time. This is the primary area of improvement relative to this project. Overall, since the robot was able to complete successful runs as shown in the demonstration video, this project was a success.

# **References**

Link to algorithm code: <http://wiki.ros.org/navigation/Tutorials/SendingSimpleGoals>

Link to ROS wiki on Move Base: <http://wiki.ros.org/move_base>

Consider actual bibliography