

ECE5532 Final Project

IGVC Flatland Course

Ben Grudzien

Sam Tkac



Table of contents

<i>Introduction</i>	3
<i>Differential Drive Kinematics</i>	4
<i>ROS Navigation Stack</i>	5
<i>Algorithm</i>	6
<i>Implementation</i>	7
<i>Results</i>	8
<i>Honorable Mention</i>	9
<i>Conclusions</i>	10
<i>References</i>	11

Introduction

- ROS Navigation Stack is an effective tool for autonomous robot navigation
- Nav Stack utilizes a camera and lidar data to guide a differential drive robot
- Nav Stack implementation can be evaluated by navigating a course with obstacles
- Provided IGVC _Flatland courses of increasing difficulty were used to build and test Navigation Stack
- Nav Stack controls robot kinematics via “Move Base” parameters

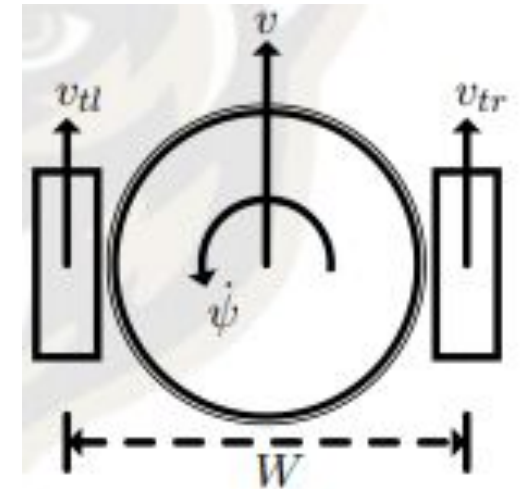
Differential Drive Kinematics

- Differential drive Roundbot was used for this project
- Moves using inputs to linear.x and angular.z parameters
- Nav Stack incorporates a navigation planner that provides its own kinematic inputs so no separate node to control this was required

Example

Move forward -> linear.x = 2 angular.z = 0 -> Left / right wheels spin at same rate

To turn, angular.z value changes and then left or right wheel will spin faster than the other to turn



Forward Kinematics

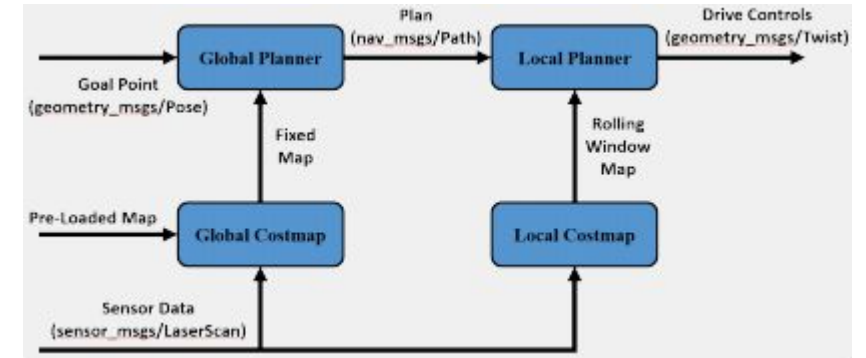
$$\begin{cases} v = \frac{r_w}{2} (\omega_r + \omega_l) \\ \dot{\psi} = \frac{r_w}{W} (\omega_r - \omega_l) \end{cases}$$

Inverse Kinematics

$$\begin{cases} \omega_l = \frac{1}{r_w} \left(v - \frac{W\dot{\psi}}{2} \right) \\ \omega_r = \frac{1}{r_w} \left(v + \frac{W\dot{\psi}}{2} \right) \end{cases}$$

ROS Navigation Stack Setup

- Differential drive Roundbot was used for this project
- Moves using inputs to linear.x and angular.z parameters
- Nav Stack incorporates a navigation planner that provides its own kinematic inputs so no separate node to control this was required



Algorithm

- Actionlib package publishes a move base goal to the robot
- Instantiated through action client and monitoring status to determine when robot has reached goal position
- Case statement is used to move to next goal

```
typedef actionlib::SimpleActionClient<move_base_msgs::MoveBaseAction> MoveBaseClient;

//we'll send a goal to the robot to move 1 meter forward
goal.target_pose.header.frame_id = "map";
goal.target_pose.header.stamp = ros::Time::now();

goal.target_pose.pose.position.x = 21.92;
goal.target_pose.pose.position.y = 15.65;
goal.target_pose.pose.orientation.w = 1.0;

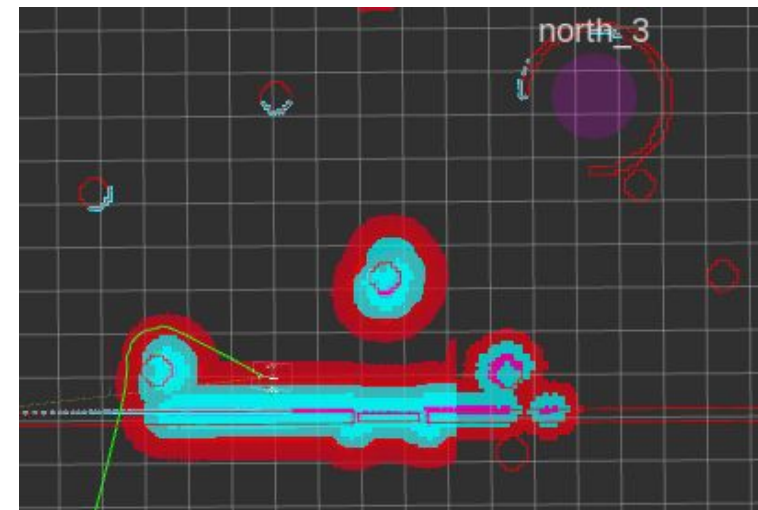
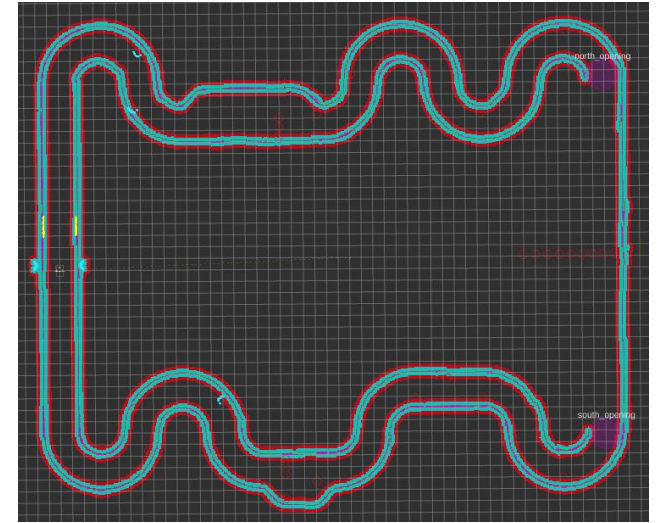
ROS_INFO("Sending goal");
ac.sendGoal(goal);

ac.waitForResult();
ROS_INFO("Waiting for result");

if(ac.getState() == actionlib::SimpleClientGoalState::SUCCEEDED){
    ++current_waypoint;
}
else
    ROS_INFO("The base failed to move towards goal");
```

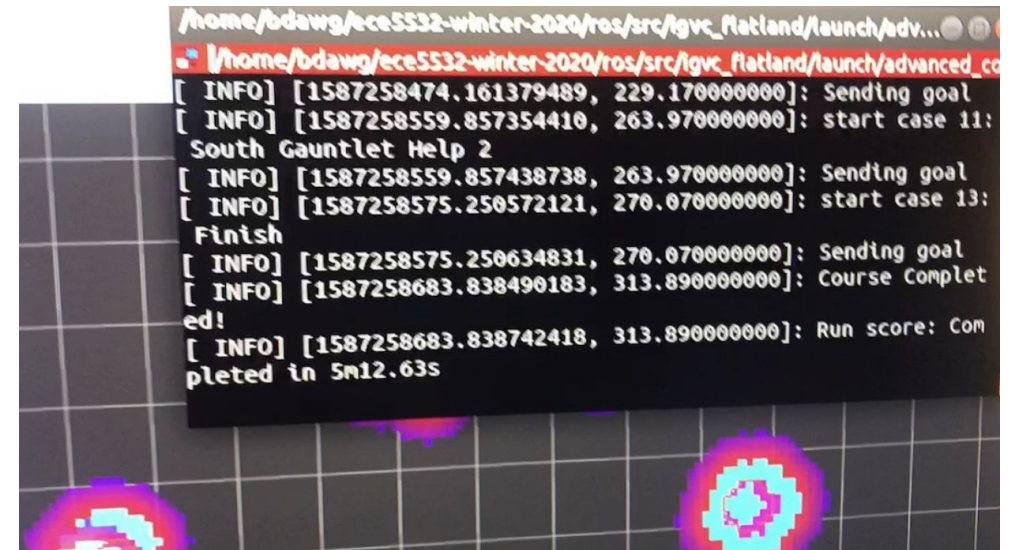

Implementation

- Local costmap shows detected objects and the associated cost to maneuver the robot through that zone
- Green line shows planned path that updates as new objects are detected
- Global and local costmap values were adjusted to during test runs to determined planned path and path that robot actually followed



Results

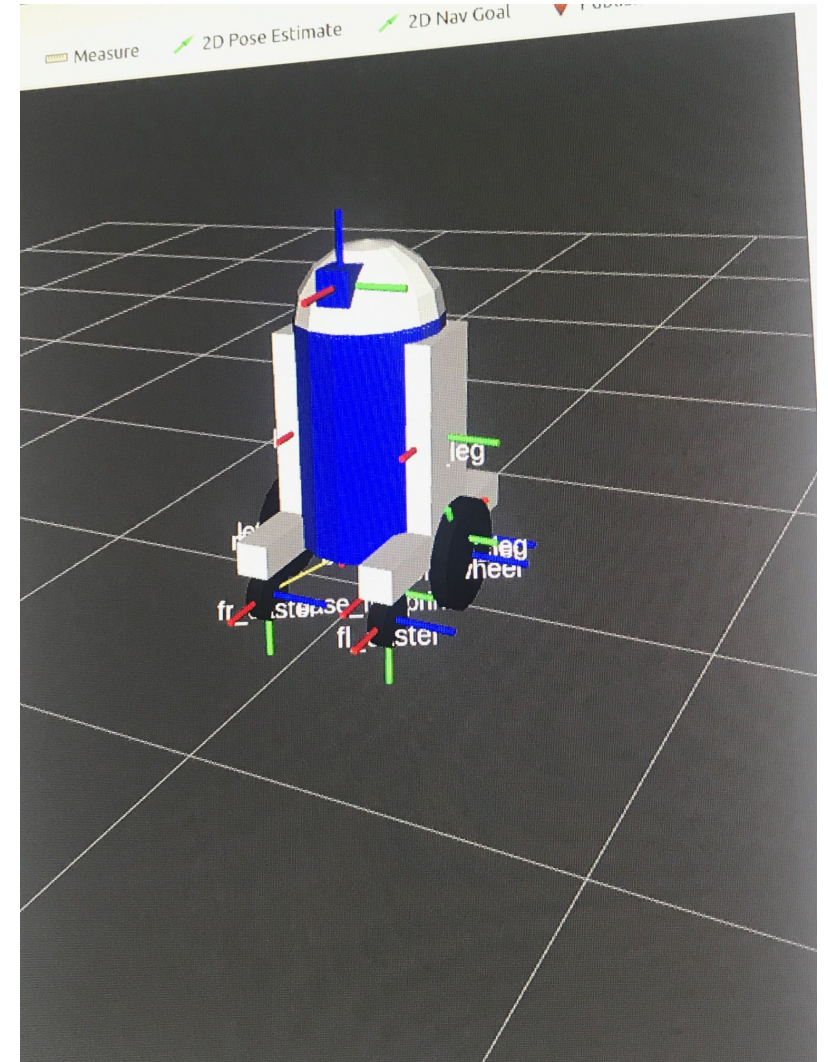
- Basic and advanced north courses were completed
- This required several hours of parameter tuning
- Additional case statements beyond just waypoint targets were added to provide more guided paths in spots where robot was struggling to maneuver



```
/home/bdawg/ece5532-winter-2020/ros/src/igvc_flatland/launch/adv...
[ INFO] [1587258474.161379489, 229.170000000]: Sending goal
[ INFO] [1587258559.857354410, 263.970000000]: start case 11:
South Gauntlet Help 2
[ INFO] [1587258559.857438738, 263.970000000]: Sending goal
[ INFO] [1587258575.250572121, 270.070000000]: start case 13:
Finish
[ INFO] [1587258575.250634831, 270.070000000]: Sending goal
[ INFO] [1587258683.838490183, 313.890000000]: Course Complet
ed!
[ INFO] [1587258683.838742418, 313.890000000]: Run score: Com
pleted in 5m12.63s
```


Honorable Mention

- Originally, we thought we could simulate this in 3D in Gazebo.
- This is the “R3D3” that was developed based of the Star Wars R2D2
- Made with URDF files



Conclusions

- Both courses were completed
- Building the code took more time than tuning parameters
- Robot struggled to complete advanced course consistently
 - improve tuning in the future
- ROS Navigation Stack is a powerful gateway into world of autonomy
- Open source access to these programs allows for future self-learning

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