# Lab 4: Action Server for Turtlebot

Sasha Belotserkovskaya, avb27@case.edu; Kristina Collins, kvc2@case.edu; Shawn Qian, xxq82@case.edu; Connor Wolfe, cbw36@case.edu

## I. Introduction

This assignment presents an example of action server/action client pair functionality. The action client is designed to send a path goal to the action server. The action server is designed to attempt to complete the goal upon receiving it. We include sensor input from the lidar alarm, which when activated will suspend the goal execution and halt the robot. When the alarm returns false again, the action server will resume the prior goal execution.

## II. Testing

### A. Demonstration Steps

First, connect to Turtlebot using the instructions in the lab assignment. The Turtlebot acts as the ROS master. Recommended to use Google's DNS for stability. Also, note that connecting to the Turtlebot in one terminal does NOT connect you to it in other terminals. To wit, for deeplearning02:

- **host deeplearning02w 8.8.8.8** Take note of the robot's IP, using Google's DNS.
- **ifconfig** Take note of the computer's IP.
- **ping deeplearning02w 8.8.8.8** Check to ensure that data can be exchanged.
- **source /opt/ros/indigo/setup.bash**
- **export ROS_MASTER_URI = http://deeplearning02w.eecs.cwru.edu:11311** Set the Turtlebot as ROS master.
- **export ROS_IP=XXX.XX.XXX.XX** Use the computer's IP here.
- **source devel/setup.bash** Switch the source to match the location of the written nodes.s

It's wise to verify connection by checking rostopics with the command **rostopic list**. Next, run the program:

- **roscd**
- **rosrun my_action_server my_action_server_client &**
- **rosrun my_action_server my_action_server**

The Turtlebot will start to move towards its first goal. When it reaches a goal it will stop, turn towards the next goal and move towards it until it has moved to every goal the client sent. If an object is in the way, the Turtlebot will detect the object and stop until the object is no longer in the way. Note that the Turtlebot will not set a new goal to move away from the object as in the homework, but instead will wait until there is nothing in its path.

### B. Tips and Notes

It's wise to keep an eye on the Turtlebot and make sure it doesn't bump into anything.
Had error 'failed to contact master'. To solve this we ran all of our code in one tab of the terminal window, because having multiple tabs open confused the computer.

## III. Code Changes

### A. Action Client

In order to have the robot halt when it detects an object, rather than set a new goal to move away from the object, we changed the main method of the my_action_client file. When the client receives a lidar alarm, we set the z field of the orientation to 1 and sent this orientation to the server. We will explain how the server will not move when this z field is equal to one.

### B. Action Server

We added an if statement to check if the z field of the orientation is 1 before moving. This field is only set to 1 if there is a lidar alarm. If z is equal to 1, then we halt. When the object is no longer in the way, we reset the goal to that last unaccomplished goal and continue to work through the goal list.

## IV. Testing Results

As shown in our video, we ran a qualitative test of the code, which consisted of the robot chasing us around in a pleasing, calisthenic and undignified manner. The robot was able to successfully identify obstacles (us, mostly) and halt in order to avoid them. You can see the robot motionless as long as an object obscures its path. As a note to the viewer, the lidar appeared to have a minimum range as well, so it would only respond to obstacles between approximately .25m and 1m away. Standing immediately in front of the robot, therefore, would not usually set off the lidar alarm and stop it from moving forward. The video can be accessed with the following link:

https://www.youtube.com/watch?v=Anpmkdmlct8feature=youtu.be

Our code may be accessed on Github from the following link:

https://github.com/shacks2017/teambacon

## V. Conclusion

This exercise provided an example of feedback control, adapting simulation-oriented code to a physical robot, and action server-action client interaction. The obstacle avoidance methods developed here may be of use in future projects concerning locomotion and path planning in dynamic environments.