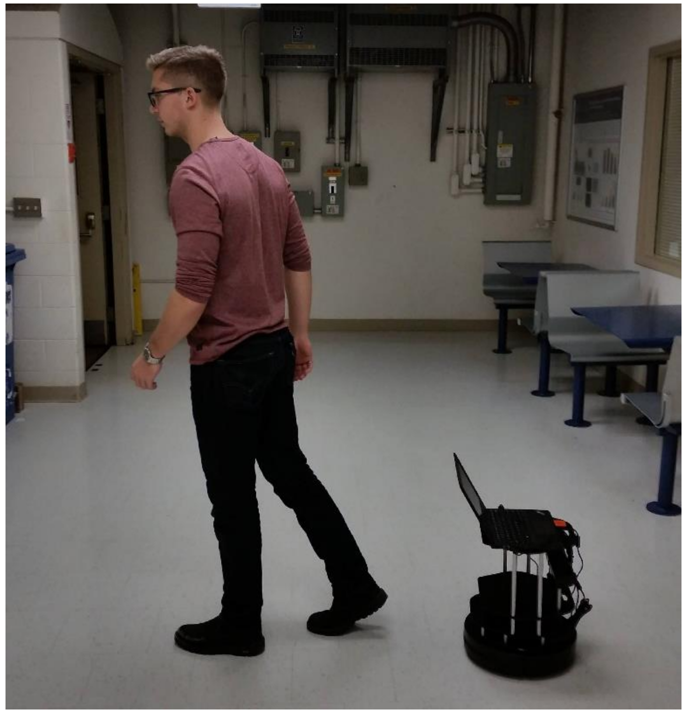
Tutorial 4 - Follower Introduction and Finite State Machine

MIE 443 – Angus Fung

Developed by Christopher Thompson and Goldie Nejat

Turtlebot Follower

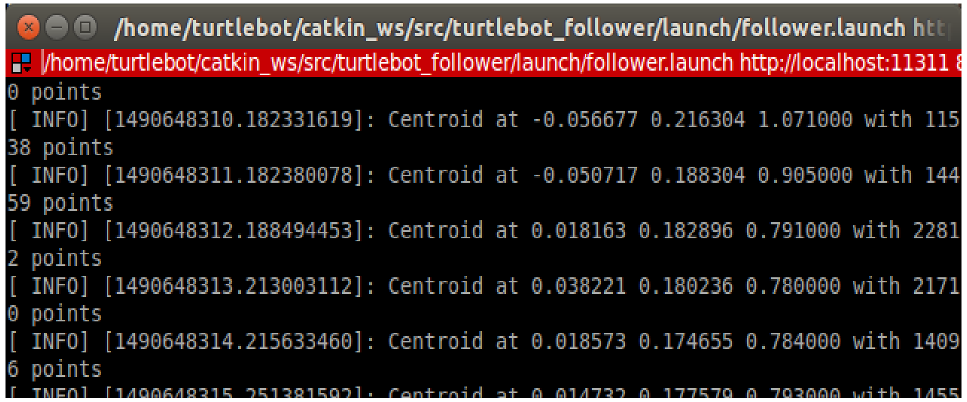
- For Contest 3 your group will be using the provided Turtlebot follower code in your contest package to navigate the Turtlebot to different positions in the contest environment.
- When an object (person) is presented in front of the Turtlebot, the follower code will act as a controller and publish the command velocities that will maintain a distance of 1 meter between the robot and the object (person).



UNIVERSITY of TORONTO

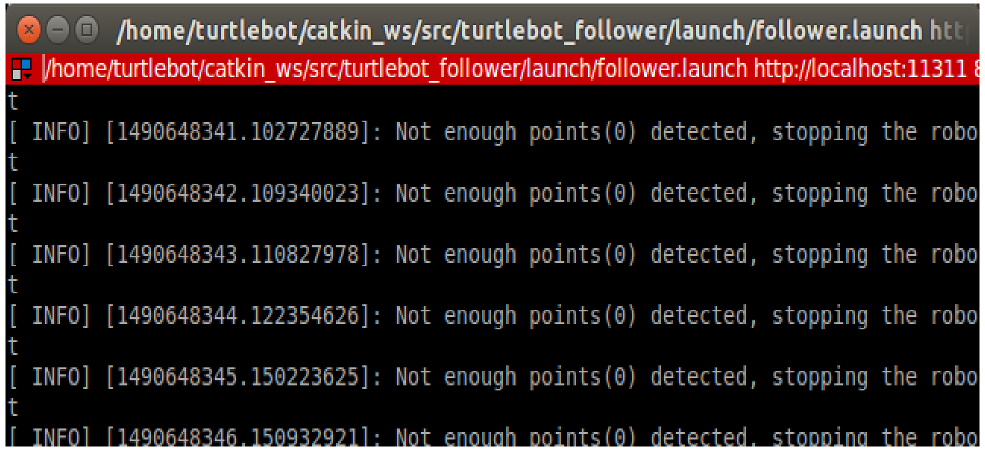
Turtlebot Follower (cont.)

- When the Turtlebot detects an object in front of it, it will begin publishing tracking velocities and will print the following messages to the terminal:



```
/home/turtlebot/catkin_ws/src/turtlebot_follower/launch/follower.launch htt
/home/turtlebot/catkin_ws/src/turtlebot_follower/launch/follower.launch http://localhost:11311 8
0 points
[ INFO] [1490648310.182331619]: Centroid at -0.056677 0.216304 1.071000 with 115
38 points
[ INFO] [1490648311.182380078]: Centroid at -0.050717 0.188304 0.905000 with 144
59 points
[ INFO] [1490648312.188494453]: Centroid at 0.018163 0.182896 0.791000 with 2281
2 points
[ INFO] [1490648313.213003112]: Centroid at 0.038221 0.180236 0.780000 with 2171
0 points
[ INFO] [1490648314.215633460]: Centroid at 0.018573 0.174655 0.784000 with 1409
6 points
[ INFO] [1490648315.251381592]: Centroid at 0.014732 0.177579 0.793000 with 1455
```

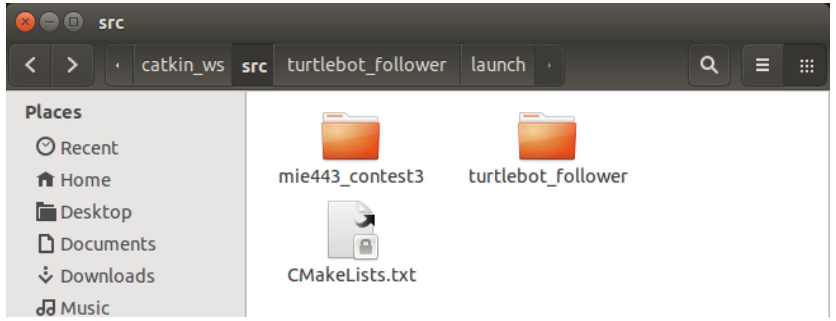
- When the Turtlebot cannot see an object in front of it, it will print the following messages to the terminal:



```
/home/turtlebot/catkin_ws/src/turtlebot_follower/launch/follower.launch htt
/home/turtlebot/catkin_ws/src/turtlebot_follower/launch/follower.launch http://localhost:11311 8
t
[ INFO] [1490648341.102727889]: Not enough points(0) detected, stopping the robo
t
[ INFO] [1490648342.109340023]: Not enough points(0) detected, stopping the robo
t
[ INFO] [1490648343.110827978]: Not enough points(0) detected, stopping the robo
t
[ INFO] [1490648344.122354626]: Not enough points(0) detected, stopping the robo
t
[ INFO] [1490648345.150223625]: Not enough points(0) detected, stopping the robo
t
[ INFO] [1490648346.150932921]: Not enough points(0) detected, stopping the robo
```

Turtlebot Follower Setup

- Begin by moving the turtlebot_follower package into the *catkin_ws/src* folder



- Before compiling or running the turtlebot_follower code, the internal permissions of one of the files within the package must be updated with the following commands in a terminal:
  - `cd ~/catkin_ws/src/turtlebot_follower/cfg/` - change the directory
  - `sudo chmod +x Follower.cfg` – updates file permissions

- After running these lines the package can be compiled from the catkin_ws directory

UNIVERSITY of TORONTO

Contest3.cpp - Velocity Callback

- In the Contest 3 code there is a subscriber to retrieve the velocities published from the Turtlebot follower code and save it to a global twist variable which contains the 6 DOF velocity instructions

```
void followerCB(const geometry_msgs::Twist msg){
          follow_cmd = msg;
}
     .
     .
     .
     .
     ros::Publisher vel_pub =
nh.advertise<geometry_msgs::Twist>("/cmd_vel_mux/input/navi",1);
```

Finite State Machine Overview

- The following block of code in contest3.cpp provides a skeleton example of a state machine architecture that will be an important implementation during this contest
- As you can see in a finite state machine there is a state variable that dictates what actions will be taken in that state
- The state variable is usually updated based on sensory input or other system stimuli

```
while(ros::ok()){
    ros::spinOnce();
    //.....**E-STOP DO NOT TOUCH**.......
    eStop.block();
    //...................................

    if(world_state == 0){
        vel_pub.publish(vel);
        sleep(0.5);

        sc.playWave(path_to_sounds+"sound.wav");

    }else if(world_state == 1){
        /* fill with your code
        ..
        ..
        */
    }
}
```

Default Example State

- In the skeleton state machine provided a default state is created that publishes arbitrary velocity commands and plays a default sound when engaged
- This state can be changed to send any commands or sounds your group needs, for example, it can be changed to publish the velocity commands from the follower callback to make the robot follow what is in front of it

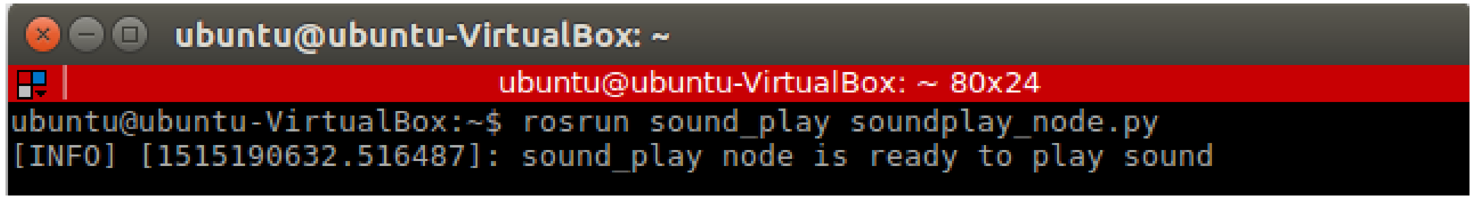
```
if(world_state == 0){
    vel_pub.publish(vel);
    sleep(0.5);

    sc.playWave(path_to_sounds+"sound.wav");

}
```

UNIVERSITY of TORONTO

SoundPlay Library

- Contest 3 requires you to program emotions for the robot which can consist of sounds
- The soundplay library can assist with this by allowing you to play .wav files from the Contest 3 code
- To play sounds, the SoundPlay server must be running in the background before running your Contest 3 code
- The following command will launch the SoundPlay server
  - `rosrun sound_play soundplay_node.py`

- When the server is ready to play sounds it will print this message:


ubuntu@ubuntu-VirtualBox: ~
ubuntu@ubuntu-VirtualBox: ~ 80x24

```
ubuntu@ubuntu-VirtualBox:~$ rosrun sound_play soundplay_node.py
[INFO] [1515190632.516487]: sound_play node is ready to play sound
```

SoundPlay Library (cont.)

- To play sounds, the sound play client is first created as shown below
- To play a sound, the *playWave* member function can be called with the absolute file path to the sound at that address, this will overwrite any sound that is currently playing as well
- To stop a sound from playing you can call the *stopWave* member function with the same file path to end the first clip
- ****Important**** the *playWave* function is non blocking, meaning that it will not stop the code from running while the sound is playing

```
sound_play::SoundClient sc;
..
..
if(world_state == 0){
    vel_pub.publish(vel);
    sleep(0.5);

    sc.playWave(path_to_sounds+"sound.wav");
    sleep(1.0);

    sc.stopWave(path_to_sounds+"sound.wav");
}
```

Questions

UNIVERSITY of TORONTO