MIE443H1S: Contest 3

Follow Me Robot Companion

1.1 Objective:

A key performance indicator of a companion robot is its ability to engage with users. As a companion robot designer, one main task is for you to design the TurtleBot to be engaging with users. The goal of this contest is to develop an interactive TurtleBot. The robot should be able to find a user (a member of your team) and follow the user while he/she moves in the environment. In addition, the robot should be able to interact with the user via the use of emotions. Based on our discussions in class, your job is to design the TurtleBot to have two primary/reactive emotions and two secondary/deliberative emotions. Each emotion must be distinct and it cannot be duplicated amongst the two categories of emotions (i.e., there must be 4 distinct emotions in total).

1.2 Requirements:

The contest requirements are:

- The TurtleBot must be able to identify and track a user as shown in Figure 1.

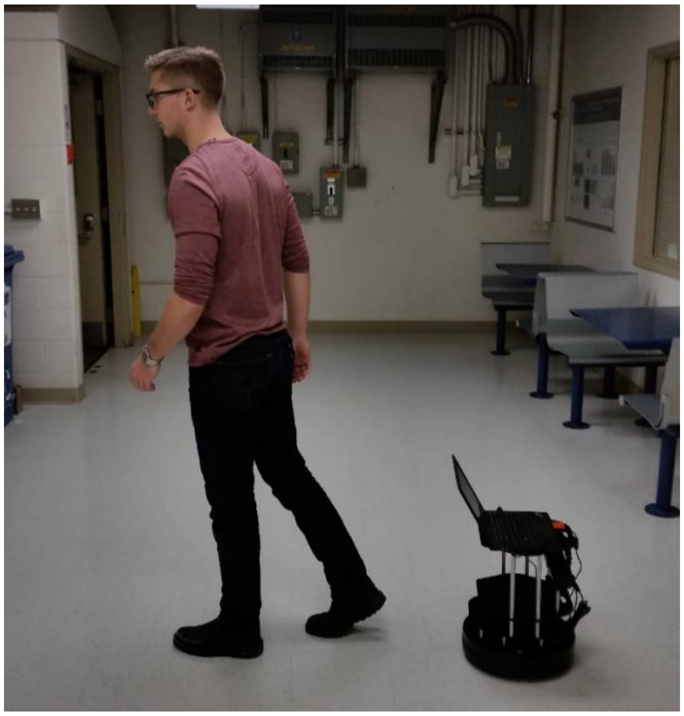

Figure 1: Robot Follower

- The competition environment for this contest will be held in an open area. Within this environment there will be 4 labeled markers that the person leading the robot will be instructed to walk to.

- For the interaction portion of this contest, the TurtleBot will be designed to have unique emotional responses to 4 environmental stimuli:

  1. When it loses track of the person it is following.
  2. When it cannot continue to track a person due to a static obstacle in its path.
  3. and 4. Two other stimuli of your own choosing.

- For the 2nd stimulus, an obstacle will be placed to obstruct the robot's path, i.e. Figure 2. The object will be short enough so that it does not interfere with the depth information provided by the onboard Kinect sensor. The object will be removed from the environment after the TurtleBot has displayed its corresponding emotion.

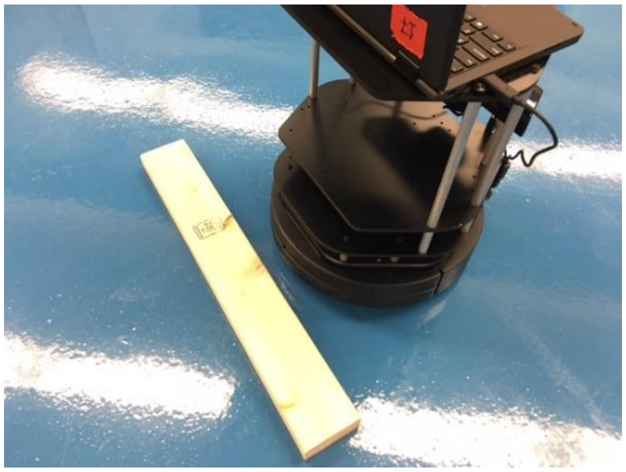

Figure 2: Example Static Obstacle in the TurtleBot's Path

- For this contest all of the TurtleBot sensors are available to use. The callbacks required for the Kinect RGB and depth information, the bumper sensor and velocity commands are already provided; however, if you wish to use any other sensory data, your team will need to add the corresponding callbacks and subscribers.
- You are encouraged to use many input and output modes for the robot. Your team must implement original motion and sound for the emotion interaction part of the contest. You have access to the *play_sound* library which can be used to play stored .wav files. **Please note**: the emotions of the robot cannot be stated by the robot or team members at any time during the contest (for example "*I am feeling sad*.").
- Each team will present their robot in front of a panel of three judges. A time limit of 8 minutes will be allocated to the contest to show your robot's functionality to the judges.

- The robot's emotions must be chosen from the list provided below:

a. Fear	h. Hate

b. Positively excited
c. Infatuated
d. Pride
e. Anger
f. Sad
g. Discontent

i. Resentment
j. Surprise
k. Embarrassment
l. Disgust
m. Rage

1.3 Scoring:

- Your team will be evaluated based on two contributions: 1) tracking of the user based on his/her movements (2 marks), and 2) emotional response to environmental stimuli (16 marks).
- The scores for user tracking are determined as follows: The user will be asked to go to 4 locations in the room (identified by markers on the ground). Two marks will be provided for tracking the user at the 1m distance while he/she moves to the locations (successful implementation of person tracking: 0.5 for each location).
- The scores for the emotional response are as follows: If a judge correctly recognizes an emotion for the robot due to a stimulus, one point is given. The total number of points available for recognition is: 4(Judge #1) + 4(Judge #2) + 4(Judge #3) = 12 points. An additional point for each emotion will be given based on the creativity and complexity of the emotional behavior displayed by the robot. Hence, you are strongly encouraged to display the robot's emotions using multi-modal outputs. The total number of points available for creativity and complexity: 4 points.

1.4 Procedure:

Code Development - The following steps should be followed to complete the work needed for this contest:

- Download the mie443_contest3.zip from Quercus. Then extract the contest 3 package (right click > Extract Here) and then move the mie443_contest3 folder to the catkin_ws/src folder located in your home directory.
- Download the turtlebot_follower.zip from Quercus. Then extract the turtlebot follower package (right click > Extract Here) and then move the turtlebot_follower folder to the catkin_ws/src folder located in your home directory.
- Open the terminal (ctrl + t) and enter the following commands in order:

1.	cd ~/catkin_ws/src/turtlebot_follower/scripts/
2.	sudo chmod +x switch.py
3.	cd ~/catkin_ws/src/turtlebot_follower/cfg/
4.	sudo chmod +x Follower.cfg

Important: these command lines enable both files (switch.py and Follower.cfg) to be executed. Otherwise, you will not be able to run follower.launch with roslaunch.

- If the above does not work ("[user] is not in the sudoers file"), please locate the files (switch.py, and Follower.cfg), and click right click, select Properties, select

Permissions, and check the box “Allow executing file as program”. You should be able to run “catkin_make” after this.

- In terminal, change your current directory to ~/catkin_ws using the cd command. Then run catkin_make to compile the code to make sure everything works properly. ****Please note if you do not run this command in the correct directory a new executable of your program will not be created.****
- For this contest, you will be utilizing the TurtleBot follower package for person tracking that we have edited and provided to you. The TurtleBot follower package uses the depth information from the Kinect sensor to locate and follow an object/person closest to the robot. A PD controller is used to control the movements of the robot. Namely, velocity commands are published to the Kobuki base to drive the robot to align to the centroid of this object.
- To portray emotions, you can add sound files in the form of .wav files to be played by your TurtleBot. To add emotions, first add your desired sound files in the sounds directory within the mie443_contest3 package. Then, to play them in your code refer to the main loop in the code breakdown of contest3.cpp.
- The contest3.cpp file located in the code package at the address ../mie443_contest3/src is where the majority of development for this contest will take place.
- Code Breakdown – contest3.cpp
    - The following callback functions are triggered whenever there is a new follower command published or the bumper is hit.

```
void followerCB(const geometry_msgs::Twist msg){
      follow_cmd = msg;
}

void bumperCB(const kobuki_msgs::BumperEvent msg){
      //Fill with code
}
```

-
    - Main block and ROS initialization functions are shown below:

```
int main(int argc, char **argv)
{
      ros::init(argc, argv, "image_listener");
      ros::NodeHandle nh;

      ros::Subscriber bumper;
      ros::Subscriber follower;
      sound_play::SoundClient sc;
      path_to_sounds = "/home/catkin_ws/src/mie443_contest3/sounds/";

      ros::Publisher vel_pub = 
nh.advertise<geometry_msgs::Twist>("/cmd_vel_mux/input/navi",1);
```

-
    - Declaration of the subscribers for the follower velocity and bumper event topics.

```
      follower =
nh.subscribe("follower_velocity_smoother/smooth_cmd_vel", 10,
&followerCB);
      bumper = nh.subscribe("mobile_base/events/bumper", 10, &bumperCB);
```

- ○ Declaration of the image transport classes that will receive the (1) RGB information from either the laptop webcam or Kinect sensor, and (2) depth data from the Kinect sensor.

```
      imageTransporter rgbTransport(”camera/image/”,
sensor_msgs::image_encodings::BGR8); //--for Webcam
      //imageTransporter rgbTransport(”camera/rgb/image_raw”,
sensor_msgs::image_encodings::BGR8); //--for turtlebot Camera

      imageTransporter
depthTransport(”camera/depth_registered/image_raw”,
sensor_msgs::image_encodings::TYPE_32FC1);
```

- ○ Initalization of state variable and local velocity commands.

```
      int state = 0;

      double angular = 0.0;
      double linear = 0.0;

      geometry_msgs::Twist vel;
      vel.angular.z = angular;
      vel.linear.x = linear;
```

- ○ Main while loop that will run until the termination of the code. It is in this loop that you should implement a controller to handle the different emotions the robot should perform depending on environmental stimuli. A sample line of code is also included to demonstrate playing .wav files using the *sound_play* libraries.

```
sc.playWave(path to sounds + "sound.wav");
```

- The following is the main while loop. The program remains within this loop until a termination signal is received (sigterm, sigkill, etc.) or 480 seconds have elapsed. The *ros::spinOnce()* checks the ROS framework to see if new sensor values have been published to the running topics and whether the registered callback functions need to be called.

```
	while(ros::ok() && secondsElapsed <=480){
		ros::spinOnce();

		if(world_state == 0){
			vel_pub.publish(follow_cmd);

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

	return 0;
}
```

- Every time the above code is changed, you must compile it from the terminal in the catkin_ws directory using the following command. If not, a new executable will not be created when you run it.

```
	catkin_make
```

Robot Simulation in Gazebo- To simulate the TurtleBot before implementing your code on the physical TurtleBot, the following steps should be followed:

- Begin by launching the simulated world through the following command:

```
	roslaunch turtlebot_gazebo turtlebot_world.launch
```

- The following starts the sound_play node to handle noises:

```
	rosrun sound_play soundplay_node.py
```

- Then run the following command to start running your code in simulation:

```
	rosrun mie443_contest3 contest3
```

- ****Please note that Gazebo in its current form will only allow your team to test the TurtleBot's emotions/motions. This is because Gazebo does not have a model for a moving person that the simulated TurtleBot can track.****

Robot Execution - When you are ready to run your program on the TurtleBot, the following steps should be followed:

- Connect the laptop to the two USB ports on the TurtleBot. One USB port is for the Kobuki base, and the other USB port is for the Kinect sensor.
- With the Kobuki base and laptop turned on, run the following commands each in their own terminal:
    - The following starts the TurtleBot Kobuki base and starts publishing topics:

      ```
      roslaunch turtlebot_bringup minimal.launch
      ```

    - The following starts the sound_play node to handle noises:

      ```
      rosrun sound_play soundplay_node.py
      ```

    - Next run the following command to start the person tracking program:

      ```
      roslaunch turtlebot_follower follower.launch
      ```

- **Before executing your code ensure that the person that you intend the robot to be following is standing in front of the robot at the required following distance.**
- After all the previous nodes have been launched, execute your code by running the following line in its own terminal:

  ```
  rosrun mie443_contest3 contest3
  ```

- If all is working correctly, your robot will begin following the user and react to the stimulus in the environment.
- When you are done, cancel all processes by pressing Ctrl-C in their respective terminals.

1.5 Report:

- The report for each contest is worth half the marks and should provide detailed development and implementation information to meet contest objectives (15 marks).
- Marking Scheme:
    - The problem definition/objective of the contest. (1 mark)
        - Requirements and constraints
    - Strategy used to motivate the choice of design and winning/completing the contest within the requirements given. (2 marks)
    - Detailed Robot Design and Implementation including:
        - Sensory Design (4 marks)
            - Sensors used, motivation for using them, and how are they used to meet the requirements including functionality.
        - Controller Design, both low-level and high-level control (including architecture and all algorithms developed) (5 marks)

        - Architecture type and design of high-level controller used (adapted from concepts in lectures)
        - Low-level controller
        - All algorithms developed and/or used
        - Changes and additions to the existing code for functionality
    - Future Recommendations (1 mark)
        - What would you do if you had more time?
        - Would you use different methods or approaches based on the insight you now have?
    - Full C++ ROS code (in an appendix). Please do not put full code in the text of your report. (2 marks)
    - Contribution of each team member with respect to the tasks of the particular contest (robot design and report). (Towards Participation Marks)
    - The contest package folder containing your code will also be submitted alongside the softcopy (PDF version) of your report. (Towards the Code Marks Above)
- Your report should be a maximum of 20 pages, single spaced, font size 12 (not including appendix).

1.6 Reference Materials for the Contest:

The following materials in Appendix A will be of use for Contest 3:

- A.1.1
- A.1.2
- A.2.1
- A.2.2
- A.2.3