**ELEE 4200/5200: Autonomous Mobility Robotics**
**Term I, 2018**
**Homework 5: Drive Robot to Avoid Obstacles**

Guidelines:

- Due date: Thursday, October 25, 2018 by 12 Noon.
- Each group of no more than two students must work on its own in completing this assignment! Feel free to consult with others (and with me and the TAs) in developing solution ideas, but the final implemented code must be your work product alone. Refer to the Syllabus, where the policy on academic integrity is clearly outlined, our classroom discussion on this topic, and consult with me if you have any questions!
- State the full names and T# of the students in the group on the cover page of every document that you submit.
- Submit the report by responding to this assignment posting in Blackboard.
- The submission should at least include the following documents, bundled together into a single zip file with the name *YourNameHW5* (use one of the group member names).
  - The main report (following the template provided).
  - The main report in 'pdf' form.
  - The MATLAB program code.
- A hard copy (printout) of the 'pdf' report with MATLAB code; staple all pages together and follow the TA's instructions on how to submit.

Goals:

- To drive a robot around continuously, while avoiding obstacles, with the help of LIDAR data.
- To understand the nature and structure of LIDAR data.
- To accomplish the task while using both LIDAR units (Kinect & Hokuyo with field of views of +/- $35^0$ and +/-$135^0$ respectively) and observe the difference in robot behavior between the two.
- To learn how to create your own Gazebo world and use it for simulations. However, a specific Gazebo world file is being uploaded to Blackboard for this assignment and you are required to use it.

Specific Tasks:

You are required to compare the behavior of the Turtlebot with two different Lidar sensor views. This requires careful program construction and observation, followed by appropriate comparison and discussion.

a) Construct a program in MATLAB that drives a simulated Turtlebot (in Gazebo), that has a Hokuyo LIDAR, continuously around while avoiding hitting any obstacles. For this part use the “Random Walk” obstacle course and start at (1,5) facing East (using the “set model state” command).

b) Repeat, this time using the Kinect LIDAR and the same starting pose.

c) Compare robot behavior corresponding to the two cases above. Your comments should be precise and to the point!

d) Move over to the U-trap part of the course, using the starting position of (-8,-6) and the direction the robot faces when you open the Gazebo world file. Use the Hokuyo data and run the robot so that it is able to handle the U-trap.

e) Repeat (d), this time using the Kinect Lidar data. Again, the task is to deal with the U-trap.

f) Compare robot behavior for parts (d) & (e).

Other information (read carefully!):

- Instructions are provided later in this document in the “Appendix” on how to incorporate the specified world file into your plans.
- You can start the Turtlebot at any location within the environment. However, it is a good idea to test your code by starting it at different locations and observing corresponding behavior.
- Produce some good videos of robot runs and include them in your submission.
- Some sources are provided below from which you can get ideas on constructing your program. However, you are still responsible for explaining how you arrived at your program and the attendant robot behavior. An important point to note is that *the process of arriving at the results is more important than obtaining results that appear correct, but you don't have any idea of how you got them!* This is embodied in the following sequence of actions – an algorithmic idea translated correctly to code, examination of the attendant behavior of the robot in multiple experiments, modification of the algorithm to change or correct behavior desirably, then repeat this sequence as necessary.
- In the assigned task, there is sometimes the risk of the robot finding itself in a trap. This situation is not necessarily one in which the robot is stationary; it is possible for it to be in a repetitive or oscillatory condition, where it essentially makes no progress. These situations, if encountered, can be handled by an “escape” behavior (even probabilistically driven!), essentially designed to get the robot “unstuck”. A simple escape behavior is incorporated in [1] below.
- You can build your program using the following sources of information:
    - The “Jackal_New.m” file found within [2] is a source of information to get ideas on building your own program. Specifically, it works by figuring out the location and angle of the closest obstacle in front of it and taking appropriate evasive action.

- Reference [1] is another source from which you can draw inspiration, in particular the "Receive Scan Data" and "Simple Obstacle Avoidance" sections. Feel free to incorporate your own ideas on top of the basic concept embedded in these two program sources!

- For the Turtlebot the "kinect scan" sensor represents the conversion of camera views into a pseudo laser scan view, through a software program interface. There are inherent limitations with the Kinect-based laser scan arrangement due to the limitations of the Kinect camera. These are explained in [1] in which only the "kinect scan" is available. Most important, the horizontal field-of-view is limited to +/- $35^0$, which makes algorithm design more challenging! An additional Hokuyo Lidar sensor is added to our Turtlebot.

References

1. https://www.mathworks.com/help/robotics/examples/explore-basic-behavior-of-the-turtlebot.html
2. https://www.clearpathrobotics.com/2015/03/matlab-robotics-system-toolbox-and-ros/

## APPENDIX

Process for Incorporating the new world file

The launch command that you may have been using so far is:

*$ roslaunch turtlebot_gazebo turtlebot_world.launch*

That is:

$ (launch command) (package folder) (launch file referencing world file)

Do the following first:

- Locate the "turtlebot_gazebo" folder.
- Within this folder there should be two directories – "launch" & "worlds". They should contain the "turtlebot_world.launch" file and the "empty.world" respectively.
- Copy the new world file for this assignment ("obstacle_avoid.world") to the "worlds" directory (using the "cp" command and appropriate path prefixes). You will need "sudo" permission to do this!

The new launch command to use is:

*$ roslaunch turtlebot_gazebo turtlebot_world.launch world_file:="/opt/ros/indigo/share/turtlebot_gazebo/worlds/obstacle_avoid.world"*

Essentially the command extension (highlighted above) substitutes the new world file for the old world file!

Included below is the listing of “turtlebot_world.launch” in case you are curious about the background to this. However, your curiosity will only be satisfied if you read through the relevant sections from Jason O’Kane’s “A Gentle Introduction to ROS”!

Essentially the use of “default” in the second line of the launch file enables the above substitution; if “value” had been used instead (see third line), we would not have been able to do it like this!

File: turtlebot_world.launch

```
<launch>
  <arg name="world_file"  default="$(env TURTLEBOT_GAZEBO_WORLD_FILE)"/>

  <arg name="base"      value="$(optenv TURTLEBOT_BASE kobuki)"/> <!-- create, roomba -->
  <arg name="battery"   value="$(optenv TURTLEBOT_BATTERY /proc/acpi/battery/BAT0)"/>  <!--
/proc/acpi/battery/BAT0 -->
  <arg name="gui" default="true"/>
  <arg name="stacks"    value="$(optenv TURTLEBOT_STACKS hexagons)"/>  <!-- circles, hexagons -->
  <arg name="3d_sensor" value="$(optenv TURTLEBOT_3D_SENSOR kinect)"/>  <!-- kinect, asus_xtion_pro -->

  <include file="$(find gazebo_ros)/launch/empty_world.launch">
    <arg name="use_sim_time" value="true"/>
    <arg name="debug" value="false"/>
    <arg name="gui" value="$(arg gui)" />
    <arg name="world_name" value="$(arg world_file)"/>
  </include>

  <include file="$(find turtlebot_gazebo)/launch/includes/$(arg base).launch.xml">
    <arg name="base" value="$(arg base)"/>
    <arg name="stacks" value="$(arg stacks)"/>
    <arg name="3d_sensor" value="$(arg 3d_sensor)"/>
  </include>

  <node pkg="robot_state_publisher" type="robot_state_publisher" name="robot_state_publisher">
    <param name="publish_frequency" type="double" value="30.0" />
  </node>

  <!-- Fake laser -->
  <node pkg="nodelet" type="nodelet" name="laserscan_nodelet_manager" args="manager"/>
  <node pkg="nodelet" type="nodelet" name="depthimage_to_laserscan"
        args="load depthimage_to_laserscan/DepthImageToLaserScanNodelet laserscan_nodelet_manager">
    <param name="scan_height" value="10"/>
    <param name="output_frame_id" value="/camera_depth_frame"/>
    <param name="range_min" value="0.45"/>
    <remap from="image" to="/camera/depth/image_raw"/>
    <remap from="scan" to="/scan"/>
  </node>
</launch>
```