Strict Deadline: Friday, June 5, 2020

Yaşar University
Spring, 2019-2020
SE4406 – Mobile Robotics Programming
Asst. Prof. Dr. Deniz Özsoyeller

Final Project

Notes:

- **Only Sakai Submissions will be accepted. Do not send your project via email!**
- **Late submissions (submissions after deadline) will not be accepted.**
- **This is an individual project (not a group project).**
- **The remaining part of the final project consists of 3 short tasks (Task-1: %10, Task-2: %5, Task-3: %15)**

1. **Call Gazebo simulator's "/gazebo/set_model_state" Service in a C++/Python code (named gazeboservices.cpp/.py) to change the default initial location of turtlebot3. Test your code in the world started with "turtlebot3_stage_1.launch" file.**

   I recommend you to check the following helper websites:

   - http://docs.ros.org/jade/api/gazebo_msgs/html/srv/SetModelState.html
   - http://docs.ros.org/jade/api/gazebo_msgs/html/msg/ModelStates.html
   - http://gazebosim.org/tutorials/?tut=ros_comm#SetModelStateExample
   - Do a web search with the keyword "Calling gazebo service set_model_state in C++ Code"

2. **Modify the "wallfollowing.cpp" code in Sakai (SE4406_ DELecture6), so that it will work in a ROS Stage Simulator for an Erratic Robot equipped with a Hokuyo laser sensor. Name your modified file as "wallfollowingstage.cpp". Use the "world2.world file" in SE4406_DELecture3. Use the closedenv.png image file in Sakai in this world.**

   Some hints:

   - The core wall following algorithm will be the same.
   - But, since now you have a different sensor: Hokuyo 2D laser scanner, the size of the ranges data should be different. In other words, it is not 360 (See SE4406_ DELecture6). So, you have to do necessary changes in MakeSmoothScan() function.
   - Also, the values set for speed, turn_rate, follow_distance, safe_distance should be different in Stage.
   - You should subscribe to the topic **/base_scan** instead of /scan.

3. **Combine the "movetogoal.py" and "wallfollowing.cpp" codes into a single "wallgoal.cpp" file. This node will make Turtlebot3 to first move to a goal point x=0.5 and y=-0.2, stop when it reaches this goal, and then continue with wall following starting from this reached point.**

   Subtasks:

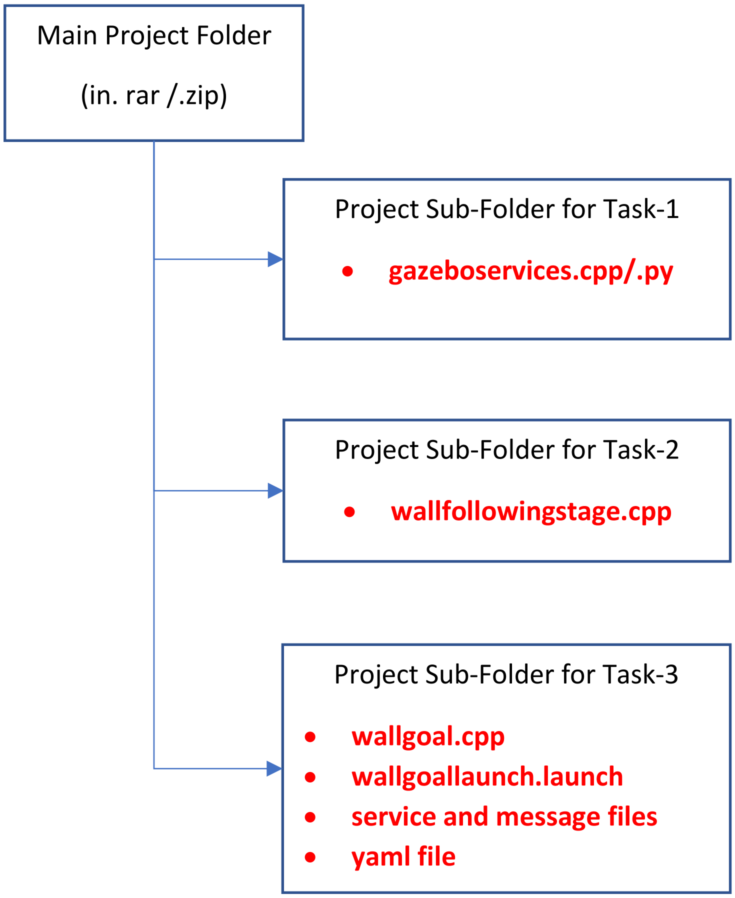
   3.1. You should start executing this node from a launch file named **"wallgoallaunch.launch"**.

   3.2. **Create a Service** yourself which will receive 4 doubles representing the coordinates of two points and returns the euclidean distance between these points. Call this service in the "double euclidean_distance(geometry_msgs::Pose goal_pose)" function of your "wallgoal.cpp" node.

   3.3. You should put the goal point's **parameters** in a **yaml** file. Include this yaml file in your "wallgoallaunch.launch" file. Then get these parameters calling **getParam** function in "wallgoal.cpp" code and assign them to goal_pose.position.x and goal_pose.position.y, respectively.

   Note: Study SE4406_DELecture4 for setting parameters, yaml files and services.

Sakai Submission Guidelines:



Main Project Folder

(in. rar /.zip)

Project Sub-Folder for Task-1

- **gazeboservices.cpp/.py**

Project Sub-Folder for Task-2

- **wallfollowingstage.cpp**

Project Sub-Folder for Task-3

- **wallgoal.cpp**
- **wallgoallaunch.launch**
- **service and message files**
- **yaml file**