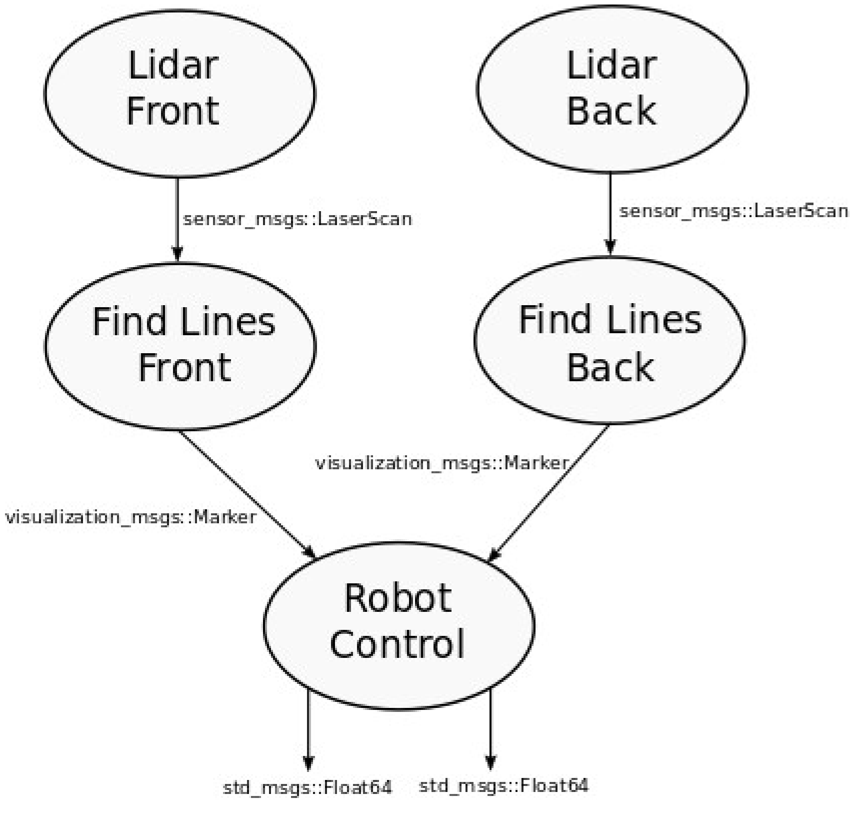
READ ME

This document will explain in a fast manner how the ENGRAIS package works. First, this document will present the libraries created but without details, if you want to know more about it, inside the “src” directory all the libraries are enumerated in order of complexity to facilitate your comprehension, as well as documentations for each one of them. Then, this document will explain the packages created, how they work, what they receive as input and give as an output. As mentioned, the libraries are:

- **Point:** This class implements an object to store (X,Y) coordinates
- **WeightedPoint:** An object to store (X,Y) and a weight to be able to calculate a weighted point average
- **Utility:** Class to implement general-use functions
- **Model:** Class to store a model (in our case, the parameters of a line in the form y = ax+b and points attached)
- **Pearl:** Class to implement the PEARL algorithm to find the lines in a cloud of points
- **Ruby:** Class to improve PEARL to our problem, so this class finds lines in a cloud of points as well. We have 5 different versions of this algorithm
- **FuzzyController:** Implements a fuzzy controller to receive the position and angle of the robot and outputs the left and right wheel command
- **StateMachine:** Class to implement a finite state machine to select which part of the motion the robot is in and outputs the left and right wheel command using the fuzzy controller or the turning motion
- **WeightedModel:** Class to store a model and a counter to be able to calculate an average slope and intercept
- **RobotControl:** Uses weighted models to select the true left and right models and the state machine to calculate the wheels’ output

The general operation of all nodes used to control the robot can be depicted by the following image



The Lidar node is simple, it gets the information from the Lidar sensor and publishes it using the LaserScan message type in ROS. This message is then received by the Find Lines node that uses this information with Ruby algorithm to find the possible models in the cloud of points. These models are then sent as Marker from rviz, using line list. The line list works by receiving a ‘N’ sized vector of points, then it uses 2 consecutive points to print N/2 lines in the rviz screen. To send the model y = 2x+ 2 for example, we have to push 2 points from this model into the line list for example (0,2) and (10, 22). To select the first and the last point, we find the smallest and largest x coordinate point in module and we calculate the y coordinate using the slope and intercept of the model, this way we have he model’s information and where the line begins and ends. The node calculate the average message period allowing the lines calculation to run multiple times, until 80% of the message period has elapsed, sending the lines found each time to have the maximum amount of data possible.

This message is sent using visualization_msgs::Marker making it possible to be shown in rviz. The robot control node begins by subscribing to receive the Makers from both front and back nodes. It then launches a second thread to calculate and send the command to the wheels. This second thread runs every 500 ms (configurable frequency), when it wakes up it selects and sends the left and right models from the set of models it received by the two line finder nodes then it calculates and sends the command to both wheels. Meanwhile, the main thread receives the messages from the nodes, populating, incrementing and adding received models into the models vector. The commands are sent using Float64 ROS messages.

To run the nodes, you can use the command **rosrun engrais_control robot_findlines _subscribe_topic:="<sub>" _publish_topic:="<pub>"** where **<sub>** is the name of the topic to subscribe and **<pub>** is the name of the topic to publish (keep the double quotation marks), or use the **roslaunch engrais_control findlines.launch** to launch both back and front nodes. To see the name of the nodes, use the command **rosnode list** and to see the name of what they publish or subscribe **rostopic list**.

Similarly, to run the robot control node you can run:

**rosrun engrais_control robot_control _subscribe_topic_front:="<sub_front>"
_subscribe_topic_back:="<sub_back>" _publish_topic_left:="<pub_left>"
_publish_topic_right:="<pub_right>"**

Or simply **roslaunch engrais_control central.launch**. To launch the whole gazebo simulation, containing the robot’s model, the Lidar sensors, rviz and all nodes mentioned, run **roslaunch engrais_control simulation.launch.**