Robot Localization

Exercise 1: Dead Reckoning

The goal of this exercise is to estimate the position of the robot by using the inertial dead reckoning method.

1. Get acceleration and orientation data from the */imu* topic.
2. Transform acceleration from body frame to world frame by using the *quatrotate* function.
3. Integrate acceleration to get robot's velocity and position.
4. Move TurtleBot in the space and check the localization performance.
5. Compare the implemented dead reckoning method with the TurtleBot's odometry from the */tf* topic.

Exercise 2: Lidar SLAM

The goal of this exercise is to build a map of the environment from 2-D lidar scans using simultaneous localization and mapping (SLAM) algorithm.

1. Create a *lidarSLAM* (https://au.mathworks.com/help/nav/ref/lidarslam.html) object for SLAM using the 2-D lidar scans.
2. Get the lidar scan form the */scan* topic.
3. From the scan, create the *lidarScan* (https://au.mathworks.com/help/nav/ref/lidarscan.html) object by using either ranges and angles or Cartesian coordinates.
4. Add the scan to the map by using the *addScan* (https://au.mathworks.com/help/nav/ref/lidarslam.addscan.html) function. This function rejects scans if they are too close to consecutive scans.
5. Find the pose of the robot by using the *PoseGraph* property of *lidarSLAM* or by using the *scansAndPoses* (https://au.mathworks.com/help/nav/ref/lidarslam.scansandposes.html) function.
6. Move TurtleBot and start over from Point 2 by continuing adding scans to the map.
7. Compare the implemented lidar SLAM with the dead reckoning method implemented in Exercise 1 and the TurtleBot's odometry from the */tf* topic.
8. [Optional] Generate an occupancy map by using the *occupancyMap* (https://au.mathworks.com/help/nav/ref/occupancymap.html) which represents the environment as a probabilistic occupancy grid.
9. [Optional] Optimize the pose graph by using the *optimizePoseGraph* (https://au.mathworks.com/help/nav/ref/optimizeposegraph.html) function.
10. [Optional] Extract the optimized absolute poses from the pose graph by using the *nodeEstimates* (https://au.mathworks.com/help/nav/ref/posegraph.nodeestimates.html) function and update the trajectory to build an accurate map of the environment.