Project 2: Particle Filter SLAM

Collaboration in the sense of discussion is allowed, however, the assignment is individual and the work you turn in should be entirely your own. See the collaboration and academic integrity statement here: https://natanaso.github.io/ece276a. *Books may be consulted but not copied from. Please acknowledge in writing people you discuss the problems with and provide references for any books or papers you used.*

Submission

Please submit the following files on **Gradescope** by the deadline shown at the top right corner.

1. Programming assignment: upload all code you have written for the project (do not include the training and test datasets) and a README file with a clear, concise description of each file.
2. Report: upload your report in pdf format. You are encouraged but not required to use an IEEE conference template[1] for your report.

Problems

In square brackets are the points assigned to each part.

1. [40 pts] Implement simultaneous localization and mapping (SLAM) using odometry, 2-D laser scans, and RGBD measurements from a humanoid robot. Use the odometry and laser measurements to localize the robot and build a 2-D occupancy grid map of the environment. Use the RGBD information to color the floor of your 2-D map.
   - **Data**: available here: https://drive.google.com/drive/folders/1TFdfDbLHE6z8NnHBWQ4BujH0vdGbhUVF?usp=sharing
   - Detailed information about the data can be found in THOR_Configuration.pdf. See Fig. 1 for an illustration.
     - The head frame has a fixed translation (33.0 cm) above the center of mass. The head can rotate in two directions.
     - The Lidar and Kinect sensors are on the head with fixed translation and no rotation. The Lidar is 15.0 cm above the head and the Kinect is 7.0 cm above.
   - The goal of the project is to use a particle filter with the laser-grid correlation observation model for localization and an occupancy grid for mapping. Here is an outline of the necessary operations:
     - **Mapping**: Try mapping using the first scan and display the map to make sure your transforms are correct before you start estimating the robot pose. Remove scan points that are too close, too far, or hit the ground. Transform the laser points from the lidar frame to the world frame. Use bresenham2D or cv2.drawContours to obtain the occupied cells and free cells that correspond to the lidar scan. Update the map log-odds according to these observations.
     - **Prediction**: Implement a prediction-only particle filter at first. In other words, use the lidar odometry to estimate the robot trajectory and build a 2-D map based on this estimate before correcting it with the laser readings. Try dead-reckoning (prediction with no noise and only a single particle) and plot the robot trajectory to see if it makes sence. Use a motion model with input from the relative lidar poses and the head joint angles to predict new positions and orientations for each particle.
     - **Update**: Once the prediction-only filter works, include an update step that uses scan matching to correct the robot pose. Remove scan points that are too close, too far, or hit the ground. Try the update step with only $3-4$ particles at first to see if the weight updates make sense. Transform the lidar scan to the world frame using each particle's pose hypothesis. Compute the correlation between the world-frame scan and the occupancy map using mapCorrelation. Call

[1] https://www.ieee.org/conferences_events/conferences/publishing/templates.html

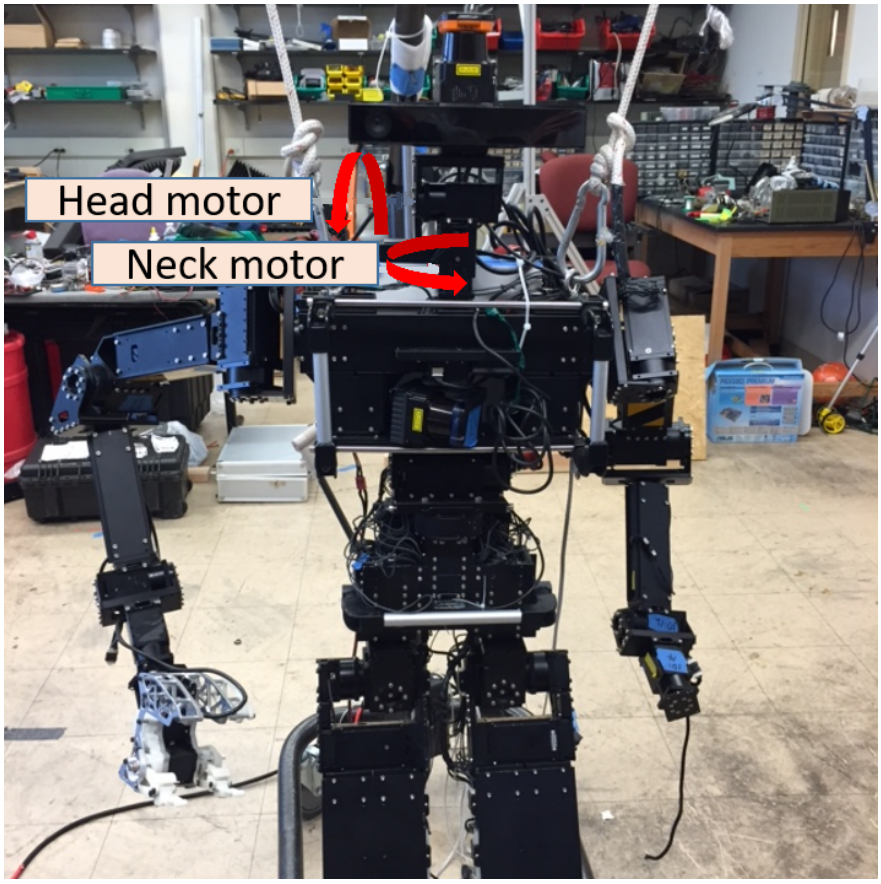

Figure 1: The data was collected by the THOR robot. The head is above the center of mass (body frame) and can rotate in two directions. The head motor rotates the head up and down. The neck motor rotates the head left and right.

mapCorrelation with a grid of values (e.g., 9×9) around the current particle position to get a good correlation (see p2 utils.py). You should consider adding variation in the yaw of each particle to get good results.

– **Texture map**: Project colored points from the RGBD sensor onto your occupancy grid in order to color the floor. Determine the depth of each RGB pixel from the depth image and transform the RGB values to the world frame. Find the ground plane in the transformed data via thresholding on the height. Color the cells in an occupancy grid with RGB values according to the projected points that belong to the ground plane. Note that sequences 1 and 2 do not contain RGBD data so texture mapping should only be done for sequences 0, 3, and 4.

2. Write a project report describing your approach to the SLAM and texture mapping problems. Your report should include the following sections:
   - [5 pts] **Introduction**: discuss why the problem is important and present a brief overview of your approach
   - [10 pts] **Problem Formulation**: state the problem you are trying to solve in mathematical terms. This section should be short and clear and should define the quantities you are interested in precisely.
   - [35 pts] **Technical Approach**: describe your approach to SLAM and texture mapping.
   - [10 pts] **Results**: present your training results, test results, and discuss them – what worked, what did not, and why. Make sure your results include (a) images of the trajectory and occupancy grid map over time and (b) textured maps over time. If you have videos, include them in the zip file and refer to them in your report.