CS 458 Assignment 3
Integrating Motion Model & Observation Model for Localization

This is the second half of the 2D SLAM project. In this assignment, you will implement both feature-based and featureless localization using the observation models to correct the robot trajectory from the robot motion model. All implementations must be in done Python3 language. Make sure you have NumPy, matplotlib, and tkinter installed in your environment. We provide a supplementary file called helper_functions.py containing the correct implementations of all functions in assignment 2 along with other useful helper functions.

Environment Setup/Dataset

A turtlebot starting at $(1, 0)$ moves freely inside an arena of size 6×6 with six landmarks. Initially, its heading direction is parallel to the positive x-axis, i.e., the heading angle is 0. The left and right boundary of the arena locates at $x = 0$ and $x = 6$. While the top and bottom boundary sit at $y = 3$ and $y = -3$. For every 0.5 seconds, we gather robot ground truth location, motion reading, lidar scan reading and store them in the following files:

- location.txt - Each line contains two floating numbers, which are the ground truth (x, y) coordinate of the robot. Use this file as a reference to evaluate your result.
- robot_motion.txt - Two floating numbers in each line represent the velocity v in the robot's heading direction and its angular velocity v_θ in radius, respectively.
- lidar_scan.txt - Each line represent a lidar scan composed of multiple lasers at different angles. The first three numbers are the minimal laser angle α_{min}, maximum laser angle α_{max}, and angle difference between adjacent lasers $\Delta\alpha$ in radius. The remaining numbers are the detected distances $d(l_i)$ from the robot to the nearest landmarks/boundaries of all rays starting from α_{min} to α_{max}. Note that the laser angle corresponds to the robot's heading angle instead of the global coordinate.

Question 1: Feature-based localization

In assignment 2, you stop after pairing the detected and group truth landmarks. In feature-based localization, you will implement the following functions:

- estimate_transform(list 1, list 2, fix scale flag): This function estimates the least square transformation from points in list 1 to that of list 2. The resulting transform estimation can be used to refine the trajectory obtained sorely from the robot motion model. The returned transformation T should be in the form (scale, $\cos\phi$, $\sin\phi$, tx, ty) where (tx, ty) is the translation while ϕ is the rotation. Note that if the fix scale flag is set, the returning scale should be 1.

- transform_estimate_and_correction(robot node list, robot motion): This is the main function that refines the robot trajectory. At each time step, first, obtain the landmark pairs, and use the function you implemented above to estimate the least square transform T. Then, you will need to apply T to the result from the motion model and modify the robot status in place by changing the attributes of robot node instances. Check the provided file to fully use the helper functions. Note that when the motion model deviates from the ground truth path, there may not be matching landmark pairs, think about what to do in this case.

The report of this section should include the following components:

- math derivation of your least square transform estimation

- visualization of the robot trajectory after feature-based localization and the ground truth

- your findings and interpretations of the result, how does it differ from only using the motion model and why is that?

Question 2: Featureless localization

In this part, you will explore featureless localization using the same data. Instead of matching landmarks, you need to use the iterative closest point(ICP) algorithm to estimate transform T by matching points of the arena boundaries at each time step and further refine the robot trajectory. In featureless localization, you will implement the following functions:

- get_subsample_rays(robot node): In order to speed up the calculation, we sub-sample all the lasers and perform boundary point matching. Use a subsampling factor of 10 in this function.

- get_correponding_points_on_boundary(points): For each sub-sampled laser, check if it reaches the boundaries of the arena with a threshold of 0.1. If so, store the laser ending point and the corresponding closest point on the arena boundaries in different lists but with the same index. This function should return two lists of points of the same length.

- get_icp_transform(points, iteration): After you have two lists of points, you need to implement the iterative closest point algorithm to estimate the transform. During each iteration, you want to first estimate the transform between all matched laser ending points and arena boundary points. Then, apply the transform to the laser ending points and try to find matches again. You will need to use the estimate_transform function

implemented in question 1. The final concatenated transform across all iterations will be the final estimated transform. Use iteration number of 100 in this function.

- featureless_transform_estimate_and_correction(robot node list, robot motion): This is the main function of featureless localization. Following the same idea of the feature-based localization, you need to apply the estimated transform from the ICP algorithm to the motion model. Note that when the motion model deviates from the ground truth path, the estimated transform from the ICP algorithm will be inaccurate since it matches the wrong boundaries, think about what to do in this case.

The report of this section should include the following components:

- visualization of the robot trajectory after featureless localization and the ground truth
- your findings and interpretations of the result, how does it differ from only using the motion model as well as the feature-based localization?

Submission instructions

Your submission should consist of two items:

- CS458_assignment3.py with all functions mentioned above implemented.
- A PDF report. This report should contain all results discussed above.

  Note: if you have any special explanations of how your code works, or instructions for running it, feel free to place them either in the report or in a separate README file.

Bundle all files into a zip or tarfile, and submit it via Brightspace. Name your submission "<your_username>.zip" or "<your_username>.tar.gz". For example, I would name my submission "zixing.zip".