Landmark Detection & Tracking (SLAM)

REVIEW | CODE REVIEW | HISTORY

Meets Specifications

Well done on completing the SLAM project, hope you had fun working on it! Happy learning and stay udacious

`robot_class.py`: Implementation of `sense`

✓ **Implement the `sense` function to complete the robot class found in the `robot_class.py` file. This implementation should account for a given amount of `measurement_noise` and the `measurement_range` of the robot. This function should return a list of values that reflect the measured distance (dx, dy) between the robot's position and any landmarks it sees. One item in the returned list has the format: `[landmark_index, dx, dy]`.**

The implementation of `sense` is perfectly done, measurement_noise and measurement_range of the robot have been taken into account and list is returned in the required format.

Notebook 3: Implementation of `initialize_constraints`

✓ **Initialize the array `omega` and vector `xi` such that any unknown values are `0` the size of these should vary with the given `world_size`, `num_landmarks`, and time step, `N`, parameters.**

The initialization of `omega` and `xi` are correct and take into account the size/dimension ie calculated using N time steps and landmark postions. The dimension is correctly multiplied by 2 to take into account x,y coordinate values.

Notebook 3: Implementation of `slam`

✓ **The values in the constraint matrices should be affected by sensor measurements *and* these updates should account for uncertainty in sensing.**

Updating the constraint matrix accounts for all sensor measurements using a series of additions that take into account the measurement noise and uncertainty in sensing

✓ **The values in the constraint matrices should be affected by motion `(dx, dy)` *and* these updates should account for uncertainty in motion.**

Updating the constraint matrix takes into account for all motion(dx,dy) and motion noise

✓ **The values in `mu` will be the x, y positions of the robot over time and the estimated locations of landmarks in the world. `mu` is calculated with the constraint matrices `omega^(-1)*xi`.**

Good use of inv function from numpy linalg, the value of `mu` is correctly calculated as per the formula `omega^(-1)*xi`

✓ **Compare the `slam`-estimated and *true* final pose of the robot; answer why these values might be different.**

The reasoning and comparison for the difference in true and estimated values of the final pose of robot are correct 👍

✓ **There are two provided test_data cases, test your implementation of slam on them and see if the result matches.**

The implementation passes on the two test cases provided.

DOWNLOAD PROJECT

RETURN TO PATH