UDACITY

Return to "Computer Vision Nanodegree" in the classroom

Landmark Detection & Tracking (SLAM)

REVIEW

CODE REVIEW 2

HISTORY

Meets Specifications

Hey, Good job. It looks that you have given the project respect and time during its implementation which is really required for
SLAM projects.

Good luck for the future and keep learning!

`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 sense function is very well written!

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 constraints are correctly initialized

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.

The values of constraint matrices are affected by the sensor measurements and it shows the robot is updating its pose as per the measurements. It's the right implementation.

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

The values of constraint matrices are affected by the motion and it shows the robot is moving as per the motion measurements. It's the right implementation.

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`.

mu is calculated correctly. The values of mu and estimated locations are making sense which is satisfactory! Good job!

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

The slam estimate and true final post of the robot are similar, which makes sense. And the answer is analytically correct.:D

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

The test results are close and make sense. Getting your test results right means your code is working well and its always nice to validate your implementation, its a very good coding practice!

DOWNLOAD PROJECT

2 CODE REVIEW COMMENTS ❯

RETURN TO PATH

Rate this review