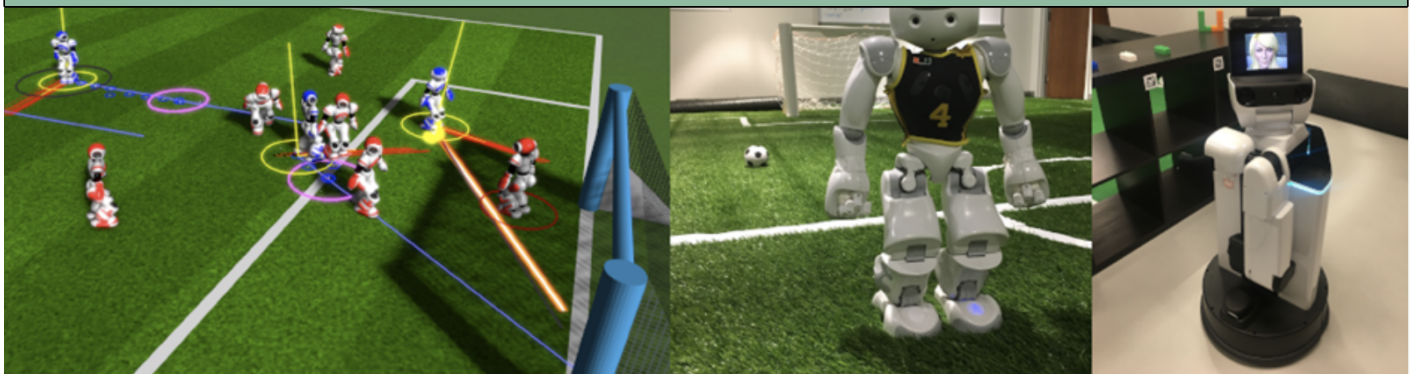
Fall 2024 - CSC398 Autonomous Robots - Assignment 5



Due date: 10/24/2024, 11:59 pm. This assignment is worth 20 points.

The objective of this exercise is to implement a particle filter utilized for landmark-based robot self-localization. Your task is to utilize the captured real-time sensor data input to localize the robot utilizing base laser scan and odometry. Data capture, processing, and significant amounts of the particle filter implementation are provided to you with some portions to be implemented. This will be detailed in the following steps:

1. New modifications have been made to the catkin workspace. `git pull` to update.
2. Install necessary dependencies by running `pip install -r src/hsr_isaac_localization/requirements.txt`.
3. Run `isaac_sim_hsr_localization_start_complete_student.sh` to start the isaac sim world, HSR initialization, particle filter, and robot lab mover behavior. This script will handle all these functions for you, click the `Play` button on the Isaac Sim window, and the PF-localization behavior and the robot motion will begin. If you'd like to run the Isaac Sim world and subsequently run your desired behaviors, run the `isaac_sim_hsr_localization_start.sh` and then (in separate terminals) `rosrun` your desired behaviors, e.g. localization and lab mover script(s).
4. The script you will be modifying is under the `scripts` directory called `hsr_pf_localization_student.py` in which designated portions of the particle filter's measurement model, resampling, and mean_position functions are to be implemented.
5. (4 points) Implement the calculation for the particle weights under the `measurement_model` function, using only measured distances and range. The weights can directly be the likelihood assuming Gaussian noise with a standard deviation of $\sigma = 0.01$.
   You can use the `test_measurement_model.py` script and corresponding function(s) to test for correctness.
6. (6 points) Implement universal stochastic sampling in the `resample` function. This completes the three necessary components for a particle filter; sample from the motion model (prior implementation), calculate weights using the sensor model, and resample, Therefore, the particle filter is now able to estimate the robot pose. Run `isaac_sim_localization_start_complete_student.sh`. The results should not be very accurate but the particles will follow the robot.
7. (2 points) Particle filters utilize a set of weighted state hypotheses, which are referred to as particles, to approximate the true state x_t of the robot at every time step t. Think of three different techniques to obtain a single state estimate x_t given a set of N weighted samples $S_t = \{\langle x_t^{[i]}, w_t^{[i]} \rangle | i = 1, ..., N\}$.
8. (2 points) Implement one of the previous methods you've described in the `mean_position` function. The function is utilized to draw the estimated pose for the robot (the yellow Rviz arrow marker).
9. (1 point) So far, the filter only utilizes measured distances. The mean position also shows the estimated orientation of the robot. Explain why the orientation is roughly correct, although no angle information of the perception is used.
10. (4 points) Add the calculation of weights using bearing in the `measurement_model` function. For the noise in the measured angles, you can use a Gaussian distribution with the deviation of $\sigma = 0.01$.
    You can use the `test_measurement_model.py` script and corresponding function(s) to test for correctness.
11. (1 point) How does the computational cost of the particle filter scale with the number of particles and the number of dimensions in the state vector of the particles? Why can a large dimensionality be a problem for particle filters in practice?

Submission:

1. Add and commit modifications to the provided package to github classroom.
2. Provide a `README` file describing relevant submission details and submit answers to theoretical questions as a `LaTeX` document. An introduction of `LaTeX` is provided by Overleaf by clicking this link.