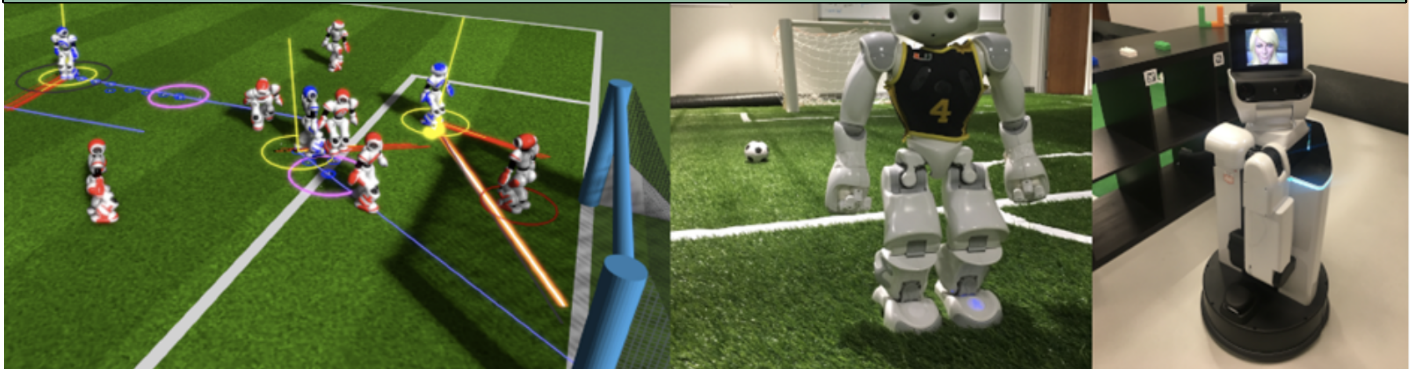
Fall 2022 - CSC752 Autonomous Robotic Systems - Assignment 4



Due date: 10/27/2022, 11:00am, before class starts. This assignment is worth 20 points.
The goal is to implement a particle filter for a landmark-based self-localization. A Matlab framework is provided to simulate a robot that perceives landmarks (with range and bearing) in a restricted field of view in front of the robot.

General comments:

- The script `agentgui.m` starts a GUI, that makes the testing much easier. The button *Run* executes the `agent.m`.
- The main script of this simulation is the `agent.m`. It loads landmark positions and logged sensor information using `read_data.m` from files in the `data` subdirectory. A loop gives the current sensor values in each time step to the function `particle_filter`. The other files in the root directory are used e.g., for drawings or a GUI.
- The directory `particlefilter` contains several files for the particle filter. All implementation for this assignment will be done in these files. The general structure and the motion model are already given.
- The directory `librobotics` contains some functions for drawings, but also angle functions that can be useful for your implementation.
- Try to avoid loops in Matlab. Code in a vectorized form is usually much faster. Take a look at the documentation of the function `repmat`. It can replicate a given matrix in different ways, which can be very useful for avoiding for-loops.

What do you need to do?

1. Download the file `a4_matlab.tar.gz` from the class home page and extract it into your user directory of the SVN in a directory "assignment4".
2. (4 points) Implement the calculation of the weights in `measurement_model.m` using only the measured distances (the range). The weights can directly be the likelihood assuming Gaussian noise with the standard deviation $\sigma = 0.4$. Use matrix operations where possible.

   Test your model using `test_measurement_model.m`.
3. (6 points) Implement the universal stochastic sampling in `resample.m`.

   This will complete the three parts needed for a particle filter: sampling from the motion model, calculating weights using the sensor model and resampling. Therefore, the filter is already able to estimate the robot pose. Run `agentgui`. It is not very accurate but the particles should follow the robot.
4. (2 points) Particle filters use a set of weighted state hypotheses, which are called 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 $S_t = \{\langle x_t^{[i]}, w_t^{[i]}\rangle | i = 1, ..., N\}$.
5. (2 points) Implement one of the methods you have described in 4. in the `mean_position.m`. This function is used to draw the estimated pose of the robot (the blue pose).
6. (1 point) So far the filter only uses 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 perceptions is used.
7. (4 points) Add the calculation of weights using the bearing in `measurement_model.m`. For the noise in the measured angles you can use a Gaussian distribution with the deviation $\sigma = 0.2$.
8. (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:

- Add and commit the complete directory "assignment4" with all Matlab files to the SVN.
- The answers for the theoretical questions have to be submitted as a LaTeX document. You can find a template for this on the class' home page.