Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Lecture: Introduction to Mobile Robotics
Summer term 2012

Prof. Dr. Wolfram Burgard
PD Dr. Cyrill Stachniss
Juniorprof. Dr. Maren Bennewitz
Juniorprof. Dr. Kai Arras

Sheet 7

Topic: Particle Filter

Submission deadline: June 20, 2012
Submit to: `mobilerobotics@informatik.uni-freiburg.de`

General Notice

In the following exercises you will implement a complete particle filter. A framework containing the motion model (cf. exercise sheet 04) and the measurement model (cf. exercise sheet 06) is provided to you, so you can concentrate on the implementation of the filter itself. The following folders are contained in the tarball:

data This folder contains files representing the world definition and sensor readings used by the filter.

octave This folder contains the particle filter framework with stubs for you to complete.

plots The framework uses this folder to store images generated by the visualization.

To run the particle filter, change into the directory `octave` and launch the *Octave* program. Inside *Octave*, type `particle_filter` to start the particle filter. Running the particle filter may take some time. While the particle filter is running, plots visualizing the state of the filter are generated and stored in the `plots` directory.

Note: You first have to complete resample.m and mean_position.m in order to get the filter working correctly.

We use the librobotics library for some of the visualization. All functions defined in the library are available in the framework.

Some implementation tips:

- Turn off the visualization to speed up the computation by commenting out the line `plot_state(...` in the file `particle_filter.m`.
- While debugging run the filter only for a few steps by replacing the for-loop in `particle_filter.m` by something along the lines of `for t = 1:50`.
- The command `repmat` allows you to replicate a given matrix in many different ways and is magnitudes faster then using for-loops.

- When converting implementations containing for-loops into a vectorized form it often helps to draw the dimensions of the data involved on a sheet of paper.
- Many of the functions in *Octave* can handle matrices and compute values along the rows or columns of a matrix. Some useful functions that support this are `sum`, `sqrt`, and many others.

Exercise 1: Theoretical Considerations

(a) 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 $\bar{x}_t$ given a set of N weighted samples $S_t = \{\langle x_t^{[i]}, w_t^{[i]} \rangle \mid i = 1, \ldots, N\}$.

(b) 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?

Exercise 2: Resampling

A particle filter consists of three steps listed in the following:

(a) Sample new particle poses using the motion model.

(b) Compute weights for the new particles using the sensor model.

(c) Compute the new belief by sampling particles proportional to their weight with replacement.

The motion model (a) is already implemented in the provided framework. In exercise sheet 06, you implemented the measurement model (b). For (c), you have to complete the file `resample.m` by implementing stochastic universal sampling.

Exercise 3: Visualization

In Exercise 1 (a) you described three ways to obtain a single state estimate from the particle cloud. Implement one of these methods in the file `mean_position.m`, which is used to draw the pose of the robot in the visualization.

Note: You can use the function `average_angle`.

With *ffmpeg* you can use the following command to generate the animation from inside the `plots` folder:

```
ffmpeg -r 10 -b 500000 -i pf_%03d.png pf.mp4
```