Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Lecture: Introduction to Mobile Robotics
Summer term 2007
Prof. Dr. W. Burgard
MSc AI J. Sturm
Dipl.-Inf. B. Lau

Sheet 10

Topic: Monte Carlo Localization I

Submission deadline: Fri 06.07.2007, 11:00 a.m. (before class)

Introduction

A robot moves in an empty 2D environment. Its state is represented as $\vec{x} = \langle x, y, \theta \rangle^T$. The motion commands are given by $u_t = \langle rot_1, trans, rot_2 \rangle^T$, where $trans$ denotes the travelled distance on a straight line in meters, rot_1 an initial rotation (before translation) in rad, and rot_2 a final rotation (after translation) in rad. Indistinguishable landmarks are placed in the environment at known positions. The robot's sensor transmits measurement data $z = \langle d, \varphi \rangle$ with range d in meters and bearing φ in rad.

The task of localization is to compute the positions of the robot, given some measurements and the known motion commands. While local pose estimation (e.g. using an EKF) means tracking of a known pose, global localization estimates the position without initial position knowledge. Particle filters are used to represent the belief about the robot's pose in this case.

This exercise deals with the initialization and motion update of a particle filter. Please look into the source code provided with this sheet. All important functions and script commands are located in the main script `pf_stub1.m`. The other files contain helper functions for data generation and visualization that you do not have to modify or use directly.

In order to make the exercises less dependent on each other, fallback solutions are suggested.

Exercise 1:

Complete the following functions for initial generation of particles in `pf_stub1.m`:

(a) `generateUniformParticles`: This is used for global localization.

(b) `generateGaussianParticles`: This is used for local pose estimation.

For more information, see the comments in the source code. The following octave functions will be helpful:

- `rand`: generates random numbers in (0,1) from a uniform distribution

- `randn`: generates random numbers from a normal distribution N(0,1)
- `sqrtm`: matrix square-root
- `repmat`: repeats a vector/matrix a given number of times. This is useful if you want to avoid for-loops and write fast vectorized code (optional).

Hint for (b): 1-dimensional N(μ,σ) distributed random numbers can be obtained using `randn*sigma + mu`. For this exercise this has to be extended to 3-dimensional gaussian noise with a given covariance matrix.

Fallback for (b): Assume the components to be independent (3 times 1D) and extract their sigmas from the covariance matrix `Cov_x`. This is less general, but correct iff `Cov_x` is a diagonal matrix.

Exercise 2:

Complete the function `evolve`. This function realizes the position update for each particle, given a motion command and additive gaussian noise. The hint and fallback above applies here as well. Test your function using particles initialized with `generateGaussianParticles`.

Exercise 3:

To integrate measurements, particles are weighted using the likelihood of the measurements. What is the likelihood function, given a single measurement $z = \langle d, \varphi \rangle$ with known correspondence to a landmark $l = \langle x_l, y_l \rangle$, a particle $\langle x, y, \theta \rangle$ and the gaussian sensor model with covariance matrix C_z? No implementation required.