

Sheet 5

Topic: Sampling, Motion Models

Due date: 24.05.2017

Exercise 1: Sampling

Implement three functions in Python which generate samples of a normal distribution $\mathcal{N}(\mu, \sigma^2)$. The input parameters of these functions should be the mean μ and the standard deviation σ of the normal distribution. As only source of randomness, use samples of a uniform distribution.

- In the first function, generate the normal distributed samples by summing up 12 uniform distributed samples, as explained in the lecture.
- In the second function, use rejection sampling.
- In the third function, use the Box-Muller transformation method. The Box-Muller method allows to generate samples from a standard normal distribution using two uniformly distributed samples $u_1, u_2 \in [0, 1]$ via the following equation:

$$x = \cos(2\pi u_1) \sqrt{-2 \log u_2}.$$

Compare the execution times of the three functions using Python's built-in function `timeit`. Also, compare the execution times of your own functions to the built-in function `numpy.random.normal`.

Exercise 2: Odometry-based Motion Model

A working motion model is a requirement for all Bayes Filter implementations. In the following, you will implement the simple odometry-based motion model.

- (a) Implement the odometry-based motion model in Python. Your function should take the following three arguments

$$\mathbf{x}_t = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} \quad \mathbf{u}_t = \begin{pmatrix} \delta_{r1} \\ \delta_{r2} \\ \delta_t \end{pmatrix} \quad \boldsymbol{\alpha} = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{pmatrix},$$

where \mathbf{x}_t is the pose of the robot before moving, \mathbf{u}_t is the odometry reading obtained from the robot, and $\boldsymbol{\alpha}$ are the noise parameters of the motion model. The return value of the function should be the new pose \mathbf{x}_{t+1} of the robot predicted by the model.

As we do not expect the odometry measurements to be perfect, you will have to take the measurement error into account when implementing your function. Use the sampling methods you implemented in Exercise 1 to draw normally distributed random numbers for the noise in the motion model.

- (b) If you evaluate your motion model over and over again with the same starting position, odometry reading, and noise values what is the result you would expect?
- (c) Evaluate your motion model 5000 times for the following values

$$\mathbf{x}_t = \begin{pmatrix} 2.0 \\ 4.0 \\ 0.0 \end{pmatrix} \quad \mathbf{u}_t = \begin{pmatrix} \frac{\pi}{2} \\ 0.0 \\ 1.0 \end{pmatrix} \quad \boldsymbol{\alpha} = \begin{pmatrix} 0.1 \\ 0.1 \\ 0.01 \\ 0.01 \end{pmatrix}.$$

Plot the resulting (x, y) positions for each of the 5000 evaluations in a single plot.

Exercise 3: Velocity-Based Motion Model

Remark: This exercise is to be solved without Python.

Consider a robot which moves on a circular trajectory with noise-free constant translational and rotational velocities, v and w . This situation is shown on page 30 of the *Probabilistic Motion Models* slides. The current pose of the robot is (x, y, θ) .

- (a) Derive the following expression for the center of the circle, (x_c, y_c) :

$$\begin{pmatrix} x_c \\ y_c \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} -\frac{v}{w} \sin \theta \\ \frac{v}{w} \cos \theta \end{pmatrix} \quad (1)$$

- (b) Now consider the situation where we are given a start pose (x, y, θ) and an end pose (x', y', θ') , connected by a circular movement. Prove that the center of the circle can be expressed as

$$\begin{pmatrix} x_c \\ y_c \end{pmatrix} = \frac{1}{2} \begin{pmatrix} x + x' \\ y + y' \end{pmatrix} + \mu \begin{pmatrix} y - y' \\ x' - x \end{pmatrix} \quad (2)$$

with some $\mu \in \mathbb{R}$

Hint: Consider the line l that connects the center of the circle and the half-way point P between (x, y) and (x', y') . How are l and P related to the two terms in (2)?

(c) Show that the value of μ is given by

$$\mu = \frac{1}{2} \frac{(x - x') \cos \theta + (y - y') \sin \theta}{(y - y') \cos \theta - (x - x') \sin \theta}.$$

Hint: μ can be calculated by using the fact that the line described by equation (2) and the line from (x_c, y_c) to (x, y) intersect at (x_c, y_c) .