thws Technische Hochschule Würzburg-Schweinfurt
Faculty of Computer Science and Business Information Systems
5172080: Fundamentals of Mobile Robotics
Prof. Dr. Pascal Meißner
pascal.meissner@thws.de
+49 (0)931 35118353
Franz-Horn-Str. 2 (room K.3.12)

Computer Practical / Laboratory 2

Topic: Probabilistic Motion Models

On Monday 12:10-13:10, Week 45 of the calendar year,
you will complete the following tasks:

T 1 – Sample From a Normal Distribution

Implement two functions in Python that draw samples from 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. The only source of randomness should be samples from the `numpy.random.uniform` function. A framework containing stubs of both functions is provided to you with `lab_2_t_1_framework.py`. Hint: Use the Python module `math` to optimize the execution times of your implementations.

- In the first function, draw samples from a zero-centered normal distribution by summing up 12 uniform distributed samples, as defined in the lecture. To convert from a zero-centered distribution to another normal distribution, add the mean μ.
- Use the Box-Muller transformation method in the second function. The Box-Muller method allows to draw samples from a standard normal distribution using the following equation:

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

  $u_1, u_2 \in [0, 1)$ are two uniformly distributed samples. The samples x from the standard normal distribution can be converted to those of another normal distribution by multiplying them by the standard deviation σ and adding the mean μ.

Using the `compute_execution_times` function in `lab_2_t_1_framework.py` to draw 10.000 samples with the parameters $\mu = 10$ and $\sigma = 4$, answer the following questions:

1.a

What are the execution times of the two functions in μs? Comment on how the times compare to each other.

1.b

What is the execution time of the built-in function `numpy.random.normal` in μs? Comment on how it compares to the times of your own functions.

Using the `compute_sample_mean_and_std_dev` function in `lab_2_t_1_framework.py` to draw 10.000 samples with the parameters $\mu = 10$ and $\sigma = 4$, answer the following questions:

1.c

What are the means and standard deviations of the samples drawn using each of your two functions? Comment on how they compare to the parameters of the true normal distribution.

1.d

What is the mean and standard deviation of the samples drawn using `numpy.random.normal`? Comment on how their accuracy compares to the sample means and standard deviations of your own implementations.

T 2 – Sample Odometry Motion Model

A working motion model is a prerequisite for all Bayes Filter implementations. In the following, you will implement the sample odometry motion model as presented in the lecture "'Probabilistic Motion Models – 2"', slide 13. A framework containing a stub of the model is provided to you with `lab_2_t_2_framework.py`.

2.a

Implement the odometry-based motion model in Python and comment on the meaning of each line of code. Your function should take the following three arguments

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

where $\mathbf{x}_t$ is the pose of the robot before the movement, $\mathbf{u}_t$ are the odometry values obtained from the robot, and α are the noise parameters of the motion model. The return value of the function should be a `numpy.array` containing the new pose $\mathbf{x}_{t+1}$ of the robot predicted by the model.

Since we do not assume that the odometry readings are perfect, you need to take noise into account when implementing your function. Use the `numpy.random.normal` function to draw zero-centered normally distributed random numbers for the noise in the motion model.

2.b

Use the `plot_and_save` function in `lab_2_t_2_framework.py` to evaluate your motion model 5000 times for the following values

$$\mathbf{x}_t = \begin{bmatrix} 2.0 \\ 4.0 \\ 0.0 \end{bmatrix} \qquad \mathbf{u}_t = \begin{bmatrix} \frac{\pi}{2} \\ 1.0 \\ 0.0 \end{bmatrix} \qquad \alpha = \begin{bmatrix} 0.05 \\ 0.05 \\ 0.01 \\ 0.01 \end{bmatrix}.$$

Plot y vs x for the 5000 predicted (x, y) positions together with the initial robot position in a single plot.

2.c

Evaluate your motion model again and again with the same initial position, odometry reading, and noise parameters. Does the return value of your motion model change or not? Why?

2.d

Comment on the effect of changing the noise parameters to $\alpha = [0.1, 0.1, 0.01, 0.01]^T$ on the spread of the predicted (x, y) positions.

2.e

Comment on the effect of changing the noise parameters to $\alpha = [0.01, 0.01, 0.05, 0.05]^T$ on the spread of the predicted (x, y) positions.