Mobile Robotics Winter term 2024/2025

Lecturer: Prof. Kai Arras
Lab instructors: Till Hielscher, Dennis Rotondi, Fabio Scaparro

Socially Intelligent Robotics Lab
Institute for Artificial Intelligence
Faculty of Computer Science
University of Stuttgart

Submission and voting deadline: see ILIAS

Submission Instructions: For the solution of theoretical tasks, use a header with your name and Matrikelnummer on each sheet and combine all files (pictures, scans, LaTeX-ed solutions) into a single `.pdf` document. For code solutions, if not stated otherwise, your code should execute correctly when called from a single `.m` or `.mlx` script (external functions are ok as long as they are called from the script). Each file you submit must include a header with your name and Matrikelnummer. Please add comments to make your code readable and to indicate to which task and subtask it refers to. For submission, all files should be included in a single `.zip` archive named as: `Ex04_YourLastname_Matrikelnummer.zip`. Remember to vote on the tasks that you solved and be ready to present them.

Exercise 04: Odometric Localization

In this exercise you will program odometry propagation and three methods to deal with the uncertainties that come with induced errors.
You are given the main script `OdometricLocalization.m` which is the script to run (no need to change anything here). From this script the four functions that implement the four tasks are called. Fill your solutions of the tasks in the respective function file!

Exercise 04.1: Forward Model First and Second Moments

This task is implemented in the function `propagatefo.m`.

a) The output pose is just the application of the motion model without including errors. Implement the model for the differential drive to find the final pose given the start pose and the movements of the left and right wheel `sL` and `sR`.

b) For the resulting pose generate points to plot an arc that traces the traveled path from the start pose to the final pose (plotting is handled in the main script, therefore just the points have to be returned).

c) Calculate the output covariance matrix using the covariance matrices for the uncertainties in the initial pose and for `sL` and `sR`.

Exercise 04.2: Monte Carlo

This task is implemented in the function `propagatemc.m`.

a) Generate a set of samples which represent possible poses of the robot based on the differential drive model. To generate a sample first inject noise on the input state according to the uncertainties in the covariance matrices for the initial pose and for `sL` and `sR`. Then compute the final pose for all samples for the noisy input and start position by passing the data through the differential drive model.

b) Calculate the mean and covariance of the sample set.

Exercise 04.3: Unscented Transform

This task is implemented in the function `propagateut.m`.

a) Compute the sigma points and the weights for the differential drive given the start pose and the movements of the left and right wheel `sl` and `sr` (use $\omega_0 = 1/3$).

b) Transform sigma points and compute and return the final moments.

Exercise 04.4: Calibrating the growth coefficient The growth coefficient k is the parameter that models the error growth in wheel odometry. Regardless of the error propagation method used, tuning this value is important. If it is too low then the odometry is over-confident, vice-versa if it is too high the covariance matrix grows too large. A practical approach to tuning this parameter is to take the robot for a walk and at the end, time index N, compare the predicted pose and covariance with the ground truth pose. The Mahalanobis distance can be used as an outlier test to determine if the ground truth lies outside the expected confidence level. If it does, this indicates that the growth coefficient k is likely too low.

- in the function `calibK.m`, implement a routine to calibrate the growth coefficient. Start from an under-approximation of k and collect trajectories calling the script `OdometrySim.m`. For each trajectory, see if the ground-truth pose passes the Mahalanobis distance test: with $\mathbf{x}_k = (x_k, y_k, \theta_k)$ we define $\mathbf{p}_k = (x_k, y_k)$ and accordingly Σ_k to be the 2×2 covariance.

$$(\mathbf{p}_{gt} - \mathbf{p}_N)^T \Sigma_N^{-1} (\mathbf{p}_{gt} - \mathbf{p}_N) < \texttt{chi2inv(0.95, 2)} \tag{1}$$

  If it doesn't pass it, then increase k so that the ground truth is inside the significance level.

- Do systematic errors have a role in this estimation process? In what way?