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: `Ex06_YourLastname_Matrikelnummer.zip`. Remember to vote on the tasks that you solved and be ready to present them.

Exercise 06: Markov and EKF Localization

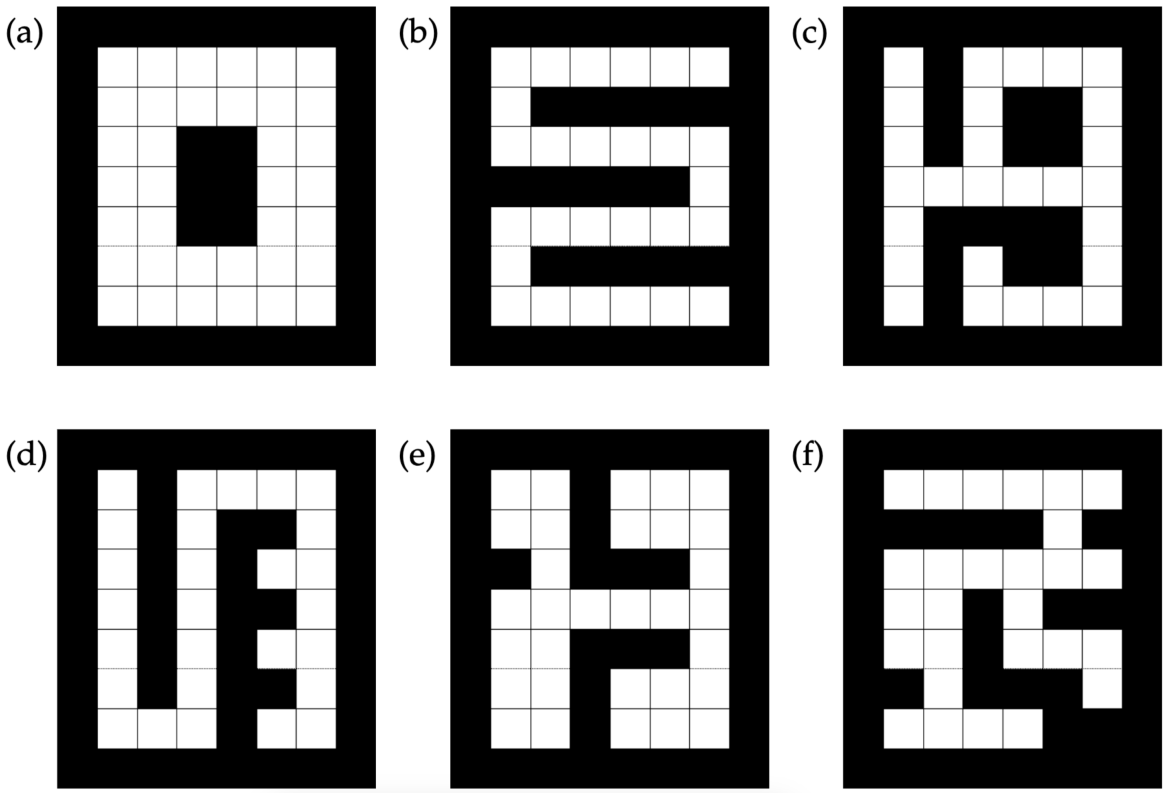
Exercise 06.1: Simplified Global Localization in Grid World

You are provided a robot with the action capabilities described below and the capability to detect when an obstacle is directly next to one of its sides. The robot needs to localize in the six grid environments given below.
The **initial position** of the robot is random but always on a free (white) cell.
The **actions** the robot can perform are:

- Action **U**: Move up until an obstacle is hit.
- Action **D**: Move down until an obstacle is hit.
- Action **L**: Move left until an obstacle is hit.
- Action **R**: Move right until an obstacle is hit.

Find shortest action sequences to localize a robot in the six grid environments depicted below. At the end of the action sequence, the robot must be at a predictable location.
Note the sequence and the predictable location for (a) - (f).

If you cannot find a solution for any of the environments, describe the problem and propose a solution that can exploit extensions of the capabilities of the robot.



Exercise 06.2: Markov Localization

Within the subfolder "Ex06_2_MarkovLocalization", you are given a grid world simulation environment in `Ex06_2_MarkovLocalization.m`.

In it, a robot can move (controls with W, A, S, D for the respective direction, Q to quit) and sense the environment with bumper sensors on all sides.

The transitions of the robot are uncertain, with a probability of reaching the desired cell, a probability of staying in the cell the robot already was, and a probability that the robot moves to one of the ahead diagonal cells.

a) Observation Model:

Implement the observation model in `observationmodel.m`.
The observations are given as obs = $\{\text{North}, \text{South}, \text{West}, \text{East}\}$ where each direction can either be 1 or 0.

For your convenience, the cell occupancy, given the map and the state, has already been calculated. Anyways, try to understand how the state representation is related to the matrix map representation.

Using the following equations, You can then calculate the `observationPropabaility` p_{obs}.

$$p_{obs} = \prod_{sides} p_{bumper}$$

where

$$p_{bumper} = \begin{cases} p_{measure} & \text{if the observation is correct} \\ 1 - p_{measure} & \text{if the observation is incorrect} \end{cases}$$

b) Filter Implementation:

Implement the filter for Markov localization. The simulation already generates the next state and observation with the functions `getnextstate()` and `getobservation()` by applying inverse transform sampling using the transition model and your observation model (from a)).
From there on, first, predict the robot's position belief based on the transition model. Next, update the robot's position belief based on your observation model. Finally, normalize the belief and save it to the field of the map data structure (map.belief).

Try different values for probabilities in transition and observation uncertainty. Also, try the other map `officemap.txt`.

Observe the results of your implementation and different values and reason them with a few sentences.

If you want, you are free to create your own challenging environments following the form that the two given to you already have.

EKF Localization

In this exercise, you will apply EKF filtering to localize a mobile robot in a warehouse environment. For this, you are given a simulation that can be run by executing `Ex06_3_4_EKFLocalization.m` within the subfolder "Ex06_3_4_EKFLocalization". Make yourself familiar with the simulation and the data it provides. The beacon observation that the simulation generates is especially important. You can access the data in the variable `beacons`.

Exercise 06.3: Predict and Match For the EKF implementation, you need data for innovation, the jacobian of the measurement function, and the measurement uncertainties. In this first subtask, you will implement the function `predictandmatch.m`, which should return all of the above data. Note that `globalMap` is data extracted from "lib/data/MapWarehouse.m" and you should look into the file to understand how to use the data in your code. This also requires implementing `measurementfunction.m`, which is used to transform the coordinates of beacons extracted from the global map into the coordinate frame of the robot. The transformation is given as

$$\texttt{transformedGlobalBeacon} = \begin{bmatrix} cos(\theta) & sin(\theta) \\ -sin(\theta) & cos(\theta) \end{bmatrix} (p_{beacon} - p_{robot})$$

where $p = \begin{bmatrix} xPosition \\ yPosition \end{bmatrix}$ and θ is the orientation of the robot.

Doing this then allows you to compare the observation to known map data.

Match the observed `localBeacons` to the transformed global beacons using the nearest neighbor method where the distance metric is given by the Mahalanobis distance.

Make sure to only proceed with matches where statistical compatibility is given for $\alpha = 0.95$. (Hint: you can use Matlab function `chi2inv`)

Exercise 06.4: EKF Implementation Using the data from the previous subtask, implement the EKF localization in the section with the same name in `Ex06_3_4_EKFLocalization.m`. For propagation of your estimated state, you can use the following call:

```
[state,C,~]= ododdforward(state,C,noisyDeltaWheelAngles(1),
                          noisyDeltaWheelAngles(2),
                          config.B,config.RL,config.RR,
                          config.KL,config.KR);
```

where `noisyDeltaWheelAngles` is a noisy version of `deltaWheelAngles` to which random Gaussian noise scaled by the encoder error `config.ENCERR` is added.

Experiment with different configuration settings like `config.CBEAC` (measurement error), initial `C` (robot state uncertainty), `config.MAXR` (sensor range), `config.ENCERR` (encoder error), `config.MANUALCONTROL` (toggle for manual control), ... to test the implementation limits.