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: `Ex07_YourLastname_Matrikelnummer.zip`. Remember to vote on the tasks that you solved and be ready to present them.

Exercise 07: Particle Localization

Exercise 07.1: Particle Weight Updates – Complete function `updateparticleweights.m`

In this task implement the likelihood field observation model to update particle weights.

a) Implement the likelihood field observation model as seen in the lecture using `scan` and `globalMap` information. As getting the distance of a point to the closest obstacle on the `globalMap` does not contribute to understanding the concepts of the model and is dependent on the map representation you are given the function `closestobstacledistance`(x, y, M) where x and y are the coordinates of the point and M is the `globalMap` which contains the obstacle information.

   Update the particle weights using the computed observation likelihoods

b) **Bonus:** You will (very likely) encounter computational performance difficulties using the model algorithm as shown in the lecture. To speed things up precompute a the likelihood field into a lookup-table such that you only need to do the computationally intensive distance calculation during runtime.

Exercise 07.2: Particle Filter Localization Implementation

a) Particle Propagation – Complete function `propagateparticles.m`

   Propagate the set of particles using 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.

b) Multinomial Resampling – Complete function `resamplemultinomial.m`

   Implement multinomial resampling as introduced in the lecture.

c) Localization – Complete function `particlefilterstep.m`

   Use the functions implemented in the previous task and subtasks to complete the particle filter localization algorithm. Be aware that you need to compute the `localizationEstimate` return value.

Exercise 07.3: Advanced Resampling

a) Systematic Resampling – Complete function `resamplesystematic.m`

   Implement systematic resampling as introduced in the lecture.

b) Adaptive Monte Carlo Localization – Complete function `resamplekld.m`

   In this task implement an modified version of Adaptive Monte Carlo Localization (AMCL), which was introduced in the lecture.

   The modification is the following: Instead of implementing the KLD bounded sampling over the whole particle filter step, where only sampled particles are propagated and updated, this version will implement a KLD bounded resampling.
   The theory for adaptively determining the particle set size is the same as introduced in the lecture – just that instead of changing the code for the whole step you only deal with another resampling function which you can call like the others and which resamples the previously propagated and updated particle set in an adaptive manner.

Exercise 07.4: Evaluation

There are many parameters in particle filter localization. In order to understand the effects of these parameters different values are evaluated in this task.

There is a section in the code of the main script prepared for this task. The place for the calculation of metrics is free to choose. You can make use of the `logRunData` struct which is passed through the function for the particle filter step and finally returned from the function `runpflocalization`. The robots ground truth state is already logged for posiible exemplary usage.

Assume the default configuration to be:

```
config.NINITIALPARTICLES=500
config.NEFFTHRESHOLD=0.66
config.RESAMPLINGALGORITHM="systematic"
```

For each subtask create a plot that compares the different runs. In MATLAB comments, write a (very) short summary of the interpretation of the results.

a) From the default configuration change the resampling threshold config.NEFFTHRESHOLD to the values $[0.0, 0.1, 0.5, 1.0]$.
The evaluation of the different runs is performed for the following metrics for each run:

- the mean value over the run taking from each step (if resampling is performed in this step): the number of maximum occurrences of a single particle after resampling
- the mean value over the run taking from each step: the localization error (Euclidean)

b) From the default configuration change the number of particles config.NINITIALPARTICLES to the values $[50, 500, 2000]$.
The evaluation of the different runs is performed for the following metrics for each run:

- the mean value over the run taking from each step: the localization error (Euclidean)
- the time taken for the run (tip: use MATLAB functions `tic` and `toc`)

c) From the default configuration change the used resampling algorithm config.RESAMPLINGALGORITHM to the values
$[''\textit{multinomial}'', ''\textit{systematic}'', ''\textit{kld}'', ''\textit{none}'']$.
The evaluation of the different runs is performed for the following metrics for each run:

- the mean value over the run taking from each step (if resampling is performed in this step): the number of maximum occurrences of a single particle after resampling
- the mean value over the run taking from each step: the localization error (Euclidean)
- the time taken for the run (tip: use matlab functions `tic` and `toc`)
- for the `"kld"` run and **plotted into a separate plot**: the number of particles over the steps of the run

d) For global localization set `config.SAMPLEGT = 0` to disable ground truth sampling. Try to find a configuration that works.

e) Reflect on the results and also relate them to sample impoverishment and degeneracy. Write a short summary (MATLAB comments are sufficient).