5th task: Estimating the model parameter

This task has 1 assignment, but for 4 distinct steps, all to be done in the same week.

After recording the trajectories described by the robot and establishing the motion model of the real robot, i.e. determining the distribution of end poses for a given commanded motion form a common origin, the second step is to build the generic motion model and to iteratively refine the model parameters $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ such that the generic model behaves as the real robot. This involves four sub-steps:

1. Building a sufficient approximation of the error-free model
2. Determining and eliminating any systematic error in u_t using the error-free model
3. Guessing sensible start values for the iterative optimisation of the $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ parameters.
4. Iteratively:
   a. computing the resulting distribution for the current parameter set
   b. determining a new parameter set based on current mismatch.

Setup

(No specific hardware setup)

Task

Step 1
The generic motion model approximates any motion by a turn, a straight-line motion and another turn. Obviously, this can not accurately represent large-scale motions, especially not those where the arc and secant between start and endpoint differ significantly. Therefore, you need to time-discretise the commanded v, ω motion and repeatedly apply the model.
Task:

- Build a function that takes v, ω and duration and the discretisation interval and computes the robot's displacement. This function shall be known as your *prediction function*.

Step 2
The commanded v, ω motion may not be what the motors are actually doing; for example due to constant (i.e. *systematic*) inaccuracies in the gears, weak motor amplifiers, friction in the drive chain or other effects. Therefor you need to establish a transformation from the v, ω you commanded in your program, and the effective v, ω that you have observed. For simplicity, we assume a linear relation $v_{\text{eff}} = k_v v + b_v$ and $\omega_{\text{eff}} = k_\omega \omega + b_\omega$.
Task:

- Use the *prediction function* from *Step 1* to compute the expected end position for each v, ω combination and compare with the average of the 10 measurements for selected v, ω combinations. If you are lucky, no significant difference appears and you can advance to *Step 3*
- Adjust the function from *Step 1* to take the additional parameters $k_v, k_\omega, b_v, b_\omega$.
- Use a numerical optimiser of your choice to find the $k_v, k_\omega, b_v, b_\omega$ which minimise the square sum error for the expected vs. average measured position over all 15 v, ω combinations.

Step 3
We aim at estimating the four parameters $\alpha_1, \alpha_2, \alpha_3, \alpha_4$. Unfortunately, these are not really arguments of the model itself, instead they are parameters of distributions added in for which we

only have numerical sampling methods. Even worse, the sampling result influences the robot's final pose in a non-linear way. There is no hope to determine the $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ analytical. So we need to find them using iterative, numerical optimisation.
Most optimisation methods require a not-to-bad start value. In this step you should guess (by carefully looking on the model equations) sensible start values for $\alpha_1, \alpha_2, \alpha_3, \alpha_4$. It might be a good idea to try out a few values and see how the model reacts.
Task:

- Determine a start value for $\alpha_1, \alpha_2, \alpha_3, \alpha_4$. Hint: make the two parameters influencing rotational uncertainty the same.

Step 4
The final step is to iteratively compute the distribution resulting for a given set of $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ and a given v, ω combination and then compare that with the distribution observed. Based on the observation outcome, one alters $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ and repeats until the change in $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ falls under a given threshold and / or the difference between observed and computed distribution falls under a given threshold. There are a number of methods to compared distributions as well as a number of methods to solve the non-linear optimisation problem at hand. One method would be Expectation Maximisation, which is designed to work with probabilistic models. Another, possibly simpler method, is the "Downhill Simplex Method", which is a generic method for multi-dimensional optimisation, designed to work in cases were no derivatives of the target function is known – as it is the case here. You are free to choose what ever numerical optimisation method you find fit, but we would suggest to give the Downhill Simplex Method a try (hint: checkout the "Further Reading" / "External Links" section).
Task:

- Build a function that takes the commanded v, ω and the runtime duration as arguments and produces the resulting expected uncertainty of the end position by computing the multi-variant Gaussian representing the average 3D pose (2D position and 1D orientation) of the robot after executing the commanded v, ω motion for the commanded duration. Use the function from *Step 2* (or *Step 1*, if *Step 2* did not reveal systematic errors).
- Decided on a measure for the difference between two distributions. The "Kullback–Leibler divergence" or the "Mahalanobis distance" are likely candidates.
- Decide on the numerical optimiser you want to try.
- Write the necessary MatLab code (or what ever system you want to use) to run the selected optimiser against your selected measure of difference between two distributions and using your function constructed in the first bullet of this Step 4. Pitfall: You have measured 15 situations (v, ω combinations) but we want to have one singe set of $\alpha_1, \alpha_2, \alpha_3, \alpha_4$. This means you need to combine the mismatch within the individual v, ω combinations to a single overall match indicator, in each iteration.
- Run the code to establish values for the parameters $\alpha_1, \alpha_2, \alpha_3, \alpha_4$.

Deliverables

Written report covering above mentioned topics, including appropriate figures, diagrams and images backing up any claim you make. The report must be self-contained and provide enough details to support any statement you make. Include:

1. Your solution to the gross-motion vs. infinitesimal motion problem, i.e. how does your prediction function look like?
2. A statement including justification if a systematical error is present, how large it is and if

you decided to amend your prediction function. Provide the amended function and parameters, if applicable.

3. Provide your start values for the iterative optimisation, give reason why the chosen values are plausible.
4. Describe the selected optimisation method and distance metric used.
5. Describe your solution to the problem that you optimise over multiple sets of v, ω combinations.
6. Provide the overall convergence error at end of the optimisation and the final parameters found.

End