SEE – Scientific Evaluation and Experimentation
Manual for practical experiments

In the SEE course, we cover practical experimentation and (statistical) data evaluation and data visualisation. This manual deals with practical experimentation, measurement and data analysis, using the experimental analysis of the characteristics of a differential drive as a running example.

This document is organised into two parts: Part A describes the overall task and provides additional theoretical background. Part B contains the individual, weekly assignments.

Part A – Introduction

Setting and Background

Your overall task is to analyse the motion model of a simple differential drive robot. We are interested in the probabilistic motion model, which relates the drive commands issued to the motors to the observable motion behaviour.

This relation is theoretically influenced by various translations:

- from desired velocities to theoretically needed motor currents
- from commanded motor currents to actual motor currents
- from actual motor currents to motor torques
- from motor torques to motor spindle rotation rates
- from motor spindle rotation rates to wheel axis rotation rates
- from wheel axis rotation rates to arc length traveled by a wheel perimeter point per second
- from arc length traveled by a wheel perimeter point per second to wheel axis displacement over ground per second

It is plausible that any attempt to actually *model* this translation chain in a closed-form equation and to accurately determine all needed parameters is extremely difficult, if not impossible.

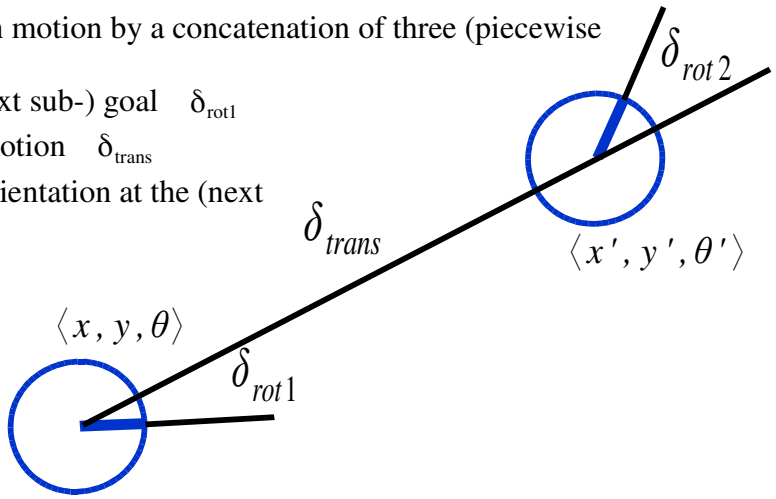
Instead, we will replace the complex, unknown theoretical true model with a simplified empirical model. A number of such models exist, but for this course we will use the well-known generic motion model discussed in chapter five of the "Probabilistic Robotics" book by Thrun et.al.

Generic motion model

The generic motion model describes each motion by a concatenation of three (piecewise constant) motion:

1. a turn into the direction to the (next sub-) goal δ_{rot1}
2. a constant velocity straight line motion δ_{trans}
3. a final turn towards the desired orientation at the (next sub-) goal δ_{rot2}

$$\text{new pose} = \begin{cases} x' = x + \delta_{trans}\cos(\theta + \delta_{rot1}) \\ y' = y + \delta_{trans}\sin(\theta + \delta_{rot1}) \\ \theta' = \theta + \hat{\delta}_{rot1} + \delta_{rot2} \end{cases}$$



Each of the three parameters must be determined such that the modelled motion shows the same behaviour as the real drive unit with its unknown motion model. Exact calculating the three parameters $\delta_{\text{rot1}}, \delta_{\text{rot1}}, \delta_{\text{trans}}$ from (measured or a priori known) true parameters of the drive unit would amount to the same intractable problem as constructing the true model in the first place.

Describing motion behaviour

The before mentioned phrase "*… such that the modelled motion shows the same behaviour as the real drive unit …*" raises the question what characterises the behaviour of a drive unit. For our purpose, a drive unit is fully described by its average response to a finite, time-constant motion command. The term "*average response*" indicates the expectation, that the robot will follow slightly different paths, when commanded the seemingly same motion in the seemingly same situation. In essence, the robot will show a *random* behaviour, resulting in a random distribution of the robot's final pose around the theoretical goal pose expected from a perfect robot. The exact nature of this distribution (and its possible dependence on the commanded motion) is taken as the description of the behaviour of the drive unit.

Task: Model the motion

Therefore, your task will be to first establish the distribution of final poses under a given commanded motion shown by your robot, and then in a second step to establish the three functions $\delta_{\text{trans}}(u)$, $\delta_{\text{rot1}}(u)$ and $\delta_{\text{rot2}}(u)$ where u is the commanded motion. The three function will be distributions such that if we sample them sufficiently often for the same given u, then the final computed poses will show the same spatial distribution as observed from the real robot.

In other words, we are ultimately interested in additional parameters $\alpha_{1,\cdots,n}$ controlling the distribution of values of our $\delta_x(u)$ functions for a given u.

Challenges in estimating the motion model parameters

To estimate the motion model parameters, we need to observe the actual motion. Observing a mobile robots motion is a non-trivial task, and can not reliably done with the robots own onboard sensors, at least not without sensors that reference distinguishable features of the environment (c.f. AMR lecture, chapter "Localisation"). Therefore, your first task will be to manually observe the robots motion under a given motion command, and your second and third task will be to setup an automated motion observation system. The first task establishes a baseline for all other motion estimation tasks in the course. The second and third aim at better reproducibility and to eliminate human error.

Assuming we are able to observe a robots motion, we need to determine how reliable our observation is, that is we need to determine the expected measurement error and trace it back to determinable parameters of the observation system. Therefore, the forth task will be to establish the error model for your automated observation system, relating the primary data (what your sensor actually reads) to the wanted, derived quantities (what your system finally delivers).

Since we have a long chain of processing steps from sensor primary (or raw) data to final position estimate, we have to make an attempt to identify and eliminate systematic errors. This will be task five.

Once we have an observation system we trust, we can use the observations made to finally estimate the motion model parameters. We do this in steps and twice, the first time ignoring the established error model of our observation system, and the second time considering the observation error.

The primary challenge in estimating the motion model parameters – once we solved the observation

problem – comes from the fact, that we deal with *indirect* parameters $\alpha_{1,\cdots,4}$ that modify the *variance* in the system model's response to given input u and need to drive this variance, which we can only *sample*, but not analytically compute, towards the observed variance of the real robot. A secondary challenge lies in the fact, that the presented generic motion model is not suitable to describe gross motions. Due to its inherent linearisation, it is only suitable for describing infinitesimal motions or – for practical matters – motions in fine time discretisation. Therefore, task six will be to derive a computational model of the drive unit, based on the generic motion model, which can describe gross motion. Since we haven't yet established actual parameter values, this model will still have the four free parameters $\alpha_{1,\cdots,4}$.

Once we have a parametrized model, we can finally estimate the parameters using a numerical optimisation method. Implementing and running such a numerical optimisation will be the seventh and last task.

The following part B will describe in greater detail the weekly assignments. Note that for practical reasons, above seven steps or “tasks” are mapped to five assignments covering six weeks. Week numbers here are weeks of experimentation, not weeks of the overall course.

Part B – Assignments

1st week: Manual motion observation

Your task is, to construct a LEGO NxT differential drive robot and manually measure the observable pose variation for three different constant velocity motions: an arc to the left, a straight line ahead and an arc to the right.

Setup

Get a LEGO NxT construction kit and a large cardboard sheet from your supervisor. You will run the NxT across the cardboard and record the end positions for multiple runs.

Task

1. Construct the "simple robot" shown on the instruction sheet included in the LEGO construction kit.
2. Equip your robot with a pen, such that you can mark the position of the robot at its stop position.
3. Verify your robot is running LeJOS (http://www.lejos.org/) and program it to run for a certain distance (approximately 2/3 of the cardboard's length for a straight line forward motion at the selected speed used in your experiment).
4. Run 3 experiments, each repeated at least 20 times (i.e. obtain at least 60 data points, better more):
   a. program the robot to run with constant angular and translational velocities for a fixed time period, such that it describes an arc to the left.
   b. program the robot to run with constant angular velocity of zero degree per second and same translational velocity as in the previous experiment for the same fixed time period, such that it describes a straight line motion ahead.
   c. program the robot to run with constant angular and translational velocities for a fixed time period, such that it describes an arc to the right, with the same velocities and times as before.

   For each experiment a – c execute at least 20 runs and mark the robots end pose after each run with a mark on the cardboard using the pen mechanism build into the robot.
5. Determine coordinates of each of the robot's end poses relative to its starting pose and tabulate the (x, y) coordinates. Note: Step 5 and 4 are best done in parallel.

Deliverables

Write a report detailing your experimental setup, observations made while executing the experiments and summarising your findings. Your report should cover:

1. The relevant aspects of the design of the robot, especially how you mark the stop position and how you ensure identical start positions.
2. The program and parameters used to drive the robot.

3. Any observation made during the execution, that may help to understand the outcome of the experiments (see also next week).
4. The observed data as Excel or LibreOffice Calc file or in similar machine-readable form.
5. An estimate of the accuracy and precision of the observed data (i.e. the measurement process), including how you arrived at these estimates and why they are plausible.

2nd week: Statistical evaluation of previous experiment

Last week, you run three experiments to estimate your robot's motion behaviour when executing simple "*v, ω*" motions, i.e. motions characterised by an overlay of an angular velocity ω and a translational velocity *v*. This week you will characterise the observed behaviour in term of statistical parameters.

Setup

No specific hardware setup, will use data obtained in previous assignment.

Task

Think of suitable methods to visualise the observed motion. Provided visualisations of the overall motions as well as of the spread-out of the stop positions of the robot. If applicable, show differences between the left and right arc experiments.

Estimate the distribution governing the spread-out of the stop positions. Try to fit a multi-dimensional Gaussian for each of the three experiments and judge the fit: Does your data fit to a Gaussian? If not, which other distribution you know that might be a good fit? Why? If you get a poor fit for a Gaussian, which effects may have cause this? Refer to your observations from last week.

Compare the actual observed motions with the commanded motions. Which similarities and which differences can you see? Can the differences fully be attributed to random noise, or is the noise (and hence the Gaussians) distorted by systematic effects originating either from your experimental setup and/or from the true characteristics of your robots drive chain?

Compare the estimated uncertainty of the measurement process itself with the statistical uncertainty of the measured stop positions and the difference from the theoretically expected stop positions. Which conclusions can be drawn from this comparison?

Note: If needed, re-run the experiment from last week to obtain usable data.

Deliverables

Written report dealing above mentioned questions, 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 a section on problems encountered, if applicable.

List of used software, including source of any function you wrote for performing your analysis.

When analysis the data with respect to the executed motions, which characteristic of the data do you establish here: Its accuracy or its precision or both?

3rd and 4th week: Calibrating an optical tracking system

So far you observed your robot's motion manually, which introduces a hard to estimate human error in your data. Next you will setup a system for automatic position tracking. By contrast to the previous experiment, this tracking system will not only obtain the start and stop position of your robot, it will also record the whole trajectory. Since you buildup the tracking system from independent parts, you will have to calibrate it before you can use it to measure your robot's motion. This two weeks assignment will cover the setup and calibration of the tracking system, including empirically determining the accuracy of the tracking system and comparing this with the theoretically expected accuracy.

Setup

You will use provided a high-quality web cam mounted on a tripod such that it can record your robot's path over the cardboard. You will equip your robot with appropriate markers such that you can track its position and orientation when performing similar runs as in week 1. The camera must be able to capture the whole path, from the robots start position up to all expected end positions.

One half of the class will use one method (call “method A” later) for doing the actual tracking, the other half will use what is called “method B” below. In class, we will compare the advantages and disadvantages of both methods.

Task

Get a camera form your supervisors and connect to your computer such that you can obtain still images. Get the camera calibration toolbox for Matlab, read and understand its manual. Create a checker board as required by the Matlab toolbox and perform the necessary image captures to allow the toolbox to compute your cameras optical parameters. In more detail:

1. Determining intrinsic camera model:
    a. Read the documentation about camera calibration coming with the calibration toolbox. Look at http://www.vision.caltech.edu/bouguetj/calib_doc/htmls/parameters.html for a description of calibration parameters. For a worked-out calibration example, pay special attention to http://www.vision.caltech.edu/bouguetj/calib_doc/htmls/example.html .
    b. Mount the camera on a tripod, build a checker board that you can present to the camera from different perspectives and run a calibration, using an appropriate number of images to ensure reliable results.
    c. Based on your understanding and experimental results so far, decide which camera model fits best for your task of estimating the robot's position with the camera you have. Justify your decision in writing, taking reference to your calibration experiment. Hint carefully look at http://www.vision.caltech.edu/bouguetj/calib_doc/htmls/example.html and search for a discussion of distortion models.
2. Determine the extrinsic parameters:
    Note: This has to be re-done every time you mechanically change for setup
    a. Place the camera and tripod such that the camera can observe the whole cardboard area where your robot will run in week 5, essentially redoing the experiments from week 1. S the traces from week one are a good guess.
    b. Place your checker board on top of the cardboard and select a suitable origin. Let the

toolbox establish the extrinsic camera parameters. Judge if the compute parameters are plausible.

3. Setup a robot tracking system using method A or B

   Method A: Using colour blob detection

   a. Construct a function in Matlab, that takes at least an image and the camera calibration parameters as arguments and identifies the centre coordinates of two bright dots in an otherwise dark image (the dots should clearly stick out, but the rest of the image does not necessarily need to be black), transform the pixel coordinates to 3D world coordinates and returns the 3D coordinates corresponding to the dots expressed in the reference frame established in step 2.b. Hint: reuse existing image processing functions. If you are familiar with ImageJ or OpenCV, you may also use these tools for the blob centre extraction, as long as you can get sub-pixel resolution.
   b. Determine the accuracy of the blob centre detection, by creating a set of artificial test images assuming a Gaussian brightness distribution over a blob's area and recomputing the blob's centre point. Test with sub-pixel resolution and perspective distortion as well as white noise spread over the test image. You also use a set of real images of your robot on the cardboard and superimpose the test blobs on those images for more realistic tests.
   c. Compare the accuracy of the blob detection with the pixel error from the camera calibration. Which one may have the bigger impact on calculating the 3D pose of a robot observed by the system?

   Method B: Using QR marker detection

   a. Construct a function in Matlab, that takes at least an image and the camera calibration parameters as arguments and identifies the centre coordinates of a 2D squared marker and the marker orientation in an image showing your robot on the cardboard and transforms the pixel coordinates to 3D world coordinates and returns the 3D coordinates corresponding to the marker's centre expressed in the reference frame established in step 2.b. Hint: Use the ALVAR 3D marker tracking library or a similar library for detection and localisation of squared markers in an image.
   b. Determine the accuracy of the marker centre detection, by creating a set of artificial test images assuming a Gaussian point spread function (1 to 2 pixel wide) and uniform white noise. Generate a set of markers (by rotating and translating a template and sampling it with sub-pixel resolution according to the point spread function) and superimpose these and the white noise on to a real image of the cardboard. Test the marker centre and orientation detection with sub-pixel resolution. Hint: ImageJ is a good tool for such image processing.
   c. Compare the accuracy of the marker detection with the pixel error from the camera calibration. Which one may have the bigger impact on calculating the 3D pose of a robot observed by the system?

4. Error estimation (same for both methods A and B above)

   a. Check the accuracy of your system by performing an exhaustive round-trip check: Accurately place your robot (either with an attached marker or with tow lit LEDs mounted) on at least 50 selected positions (approximately uniformly distributed over the camera-visible portion of the cardboard) and

      i. record the selected position where you placed it,

      ii. the pixel coordinates where the robot (or its marker) is found in the image,

      iii. the calculated position of the robot in the real world based on the pixel coordinates

      iv. and the expected pixel coordinates where a robot placed on the selected real-world position should have been found by the image processing system.

b. Based on these 50+ data points, determine the actual accuracy of your tracking system by comparing expected and estimated positions. Judge which part of the tracking system is likely the primarily contributor to the observed uncertainty in calculated 3D positions.

Deliverables

Write a report describing the calibration process, including a theoretical part that describes the camera and lens errors measured and corrected by the calibration.

In week 3:

1. Describe the images poses used for calibration and report the found camera parameters, including error estimates where available.
2. Provide arguments for which camera model best fits your situation and hence should be used.

In week 4

1. Describe the outcome of the blob or marker detection accuracy test.
2. Describe the execution and outcome of the accuracy test of the fully assembled and working tracking system.
3. Provide the final established error bounds of the full system, including a justification why they are plausible. Which part of the calibration and accuracy test dominates the final estimate?
4. Is there a better way to establish the accuracy of the system? Furthermore, does the test really establish the system's accuracy, or does it establish precision or both?
5. Do you expect accuracy and precision to be of similar scale, or different? I there a way to determine both independently?

5th week: estimating the probability distribution describing the real robot's drive system

Our finally goal is, to estimate the four empirical parameters of the generic motion model $x_t = f(x_{t-1}, u_t, \alpha_1, \alpha_2, \alpha_3, \alpha_4)$ described in part A of this handbook, where u_t is a commanded v, ω motion and x_t the robot's pose, such that the probability $p(\hat{x}_t | x_{t-1}, u_t)$ to end up in a pose $\hat{x}_t$, given a previous pose x_{t-1} and an attempted motion u_t, is the same as observed in the experiment described below. In other words, we want to estimate the four parameters $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ such that the probability distribution over the x_t predicted by our model is the same distribution as the distribution observed. To fit the model parameters $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ in $x_t = f(x_{t-1}, u_t, \alpha_1, \alpha_2, \alpha_3, \alpha_4)$ such that x_t follows the same distribution as a real robot's drive system, we need to compare different probability distributions and determine their similarity.
In this week you will first determine the distribution for the real robot.

Setup

You need the robot programmed to run on arcs for a fixed time and your tracking system as created in the previous two weeks.

Task

1. Verify your camera model and translation between pixel coordinates and 3D world coordinates by measuring at least 4 representative points and comparing with ground truth.
2. Program your robot to follow trajectories as indicated in the figure 1. Use three different translational velocities v (excluding zero) and 5 different rotational velocities ω (including zero and symmetrical regarding clockwise/counter-clockwise, example: 20°/s, 10°/s, 0°/s, -10°/s -20°/s). In total these are 15 v, ω combinations, each will later be executed 10 times.

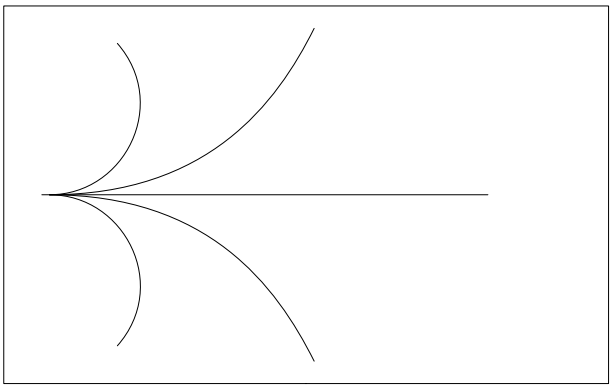


Figure 1: Robot trajectories for estimating the drive system characteristics.

3. Record the robot's trajectory and determine the start and endpoint for each motion. Make sure the robot runs with constant velocities while tracking it, i.e. use the robot's position a fixed short time **after** start of motion as start point and the position shortly before it stops as endpoint. Run and record each combination of translational and rotational velocities 10 times, keeping the effective start position and expected effective end position constant, i.e. use the same time offset for all runs of a given v, ω combination. In total this produces 150 traces.
4. For each combination of translational and rotational velocities v, ω, determine the uncertainty of the end position by building the multi-variant Gaussian representing the average

3D pose (2D position and 1D orientation).

5. Similar, determine the uncertainty of the start position for each v, ω combination. by building the multi-variant Gaussian representing the average 3D pose (2D position and 1D orientation)
6. For each v, ω combination, compare the amount of uncertainty of the start point with the uncertainty of the endpoint and the precision of the position estimate of your setup (i.e. the error bound when going from pixel coordinates to world coordinates, as established in last weeks assignment)

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. Result from recalibration of experimental setup.
2. The relevant aspects of the design of the robot, especially how you attached the LEDs or marker
3. The program and parameters used to drive the robot.
4. Any observation made during the execution, that may help to understand the outcome of the experiments.
5. Visualisation of obtained traces.
6. The calculated uncertainties for start and stop positions. Any notable characteristics of the data.
7. An analysis of data quality.
8. The observed data as Excel or LibreOffice Calc file or in similar machine-readable form.

6th week: Estimating the model parameter

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 off 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 ned 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 react.

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 distribution 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