HOCHSCHULE BONN-RHEIN-SIEG

Scientific Experimentation and Evaluation Manual for Practical Experiments

SUMMER SEMESTER 2018

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.

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

1 Part A - Introduction

1.1 Setting and Background

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

This relation is theoretically influenced by various translations:

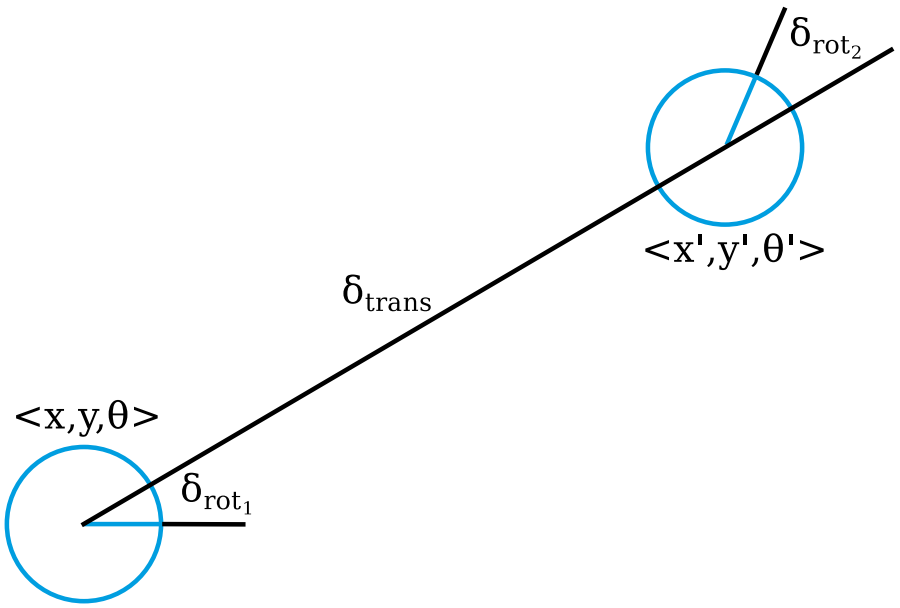
- from desired velocities to theoretically needed motor currents
- from commanded motor currents to actual motor currents
- from actual motor currents to motor torques
- from motor torques to motor spindle rotation rates
- from motor spindle rotation rates to wheel axis rotation rates
- from wheel axis 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. Because of that, 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 ”Probabilistic Robotics” book by Thrun et al.

1.2 Generic Motion Model

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

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}



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

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 calculation of the three parameters δ_{rot1}, δ_{rot2}, and δ_{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.

1.3 Describing Motion Behaviour

The phrase "... such that the modelled motion shows the same behaviour as the real drive unit ..." mentioned above raises the question about 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, this means that 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.

1.4 Task: Model the Motion

Your task will thus 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. These three function will represent distributions, such that if we sample them sufficiently often for the same given u, the final computed poses will show the same spatial distribution as observed on the real robot. In other words, we are ultimately interested in additional parameters $\alpha_{1,\ldots,n}$, which control the distribution of values of our $\delta_x(u)$ functions for a given u.

1.5 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 cannot be reliably done with a robot's onboard sensors, at least not without sensors that reference distinguishable features of the environment (c.f. AMR lecture, chapter *Localisation*). Your first task will thus be to manually observe the robot's motion under a given motion command, while your second 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 task aims at better reproducibility and at eliminating human error.

Assuming that we are able to observe a robot's motion, we need to determine how reliable our observation is, i.e. we need to determine the expected measurement error and trace it back to determinable parameters of the observation system; therefore, part of the second task will be to establish the error model of 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 part of task three.

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

The primary challenge in estimating the motion model parameters - once we have solved the observation problem - comes from the fact that we deal with *indirect* parameters $\alpha_{1,..,4}$ that modify the *variance* in the system model's response to a 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 three will include deriving a computational model of the drive unit based on the generic motion model, which can describe gross motion. Since we haven't yet established the actual parameter values, this model will still have the four free parameters $\alpha_{1,\ldots,4}$.

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 final part of task three.

Part B will describe the individual tasks in greater detail and will split these into weekly assignments. Note that, for practical reasons, the above three steps are mapped to five tasks and eight assignments.

2 Part B - Assignments

2.1 Task 1: Manual Motion Observation

This task consists of two assignments. 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.

2.1.1 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.

1. Construct a simple robot - you're free to choose the design.
2. Equip your robot with a pen, such that you can mark the position of the robot at its stop position.
3. Verify that your robot is running the Lejos OS 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 three 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 the coordinates of each of the robot's end poses relative to its starting pose and tabulate the (x, y) coordinates. Note: Steps 5 and 4 are best done in parallel.

2.1.2 Assignment 1.1: Design of Experiment

1. Consider how to best record the end poses of the three times 20 runs of the robot.
2. Design and build a differential drive robot including some means to obtain the robots end poses.
3. Describe in writing the robots design, the measurement process and the expected problems.

4. Give a rough (but justified by some arguments) estimation of the expected accuracy and precision of your measurement process.

Note: you are not yet requested to actually run the robot. This assignment deals with the experimental design.

2.1.3 Deliverables 1.1

Write a report detailing your envisaged experimental setup, expected problems and expected performance. In your report, use terminology from the lecture (i.e. measurement system, measurand, measured quantity, and so forth) to describe your experiment. 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. An estimate of the expected accuracy and precision of the to-be-observed data (i.e. the measurement process), including how you arrived at these estimates and why they are plausible.

2.1.4 Assignment 1.2

1. Refine your experiment design according to feedback in class, if necessary
2. Run the requested experiment, by driving the robot 20 time on an arc to the right, 20 times straight ahead, and 20 times on an arc to the left with the same magnitude, but opposite direction of the commanded curvature to the right
3. Record the end poses and any other observation that may prove useful for later analysis of the experiment
4. Describe in writing the final robots design, the final measurement process and any observation made during execution of the experiment
5. Think of suitable methods to visualise the observed motion. Provide 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
6. Perform some statistical data analysis and give an estimation of the achieved accuracy and precision of your measurements

2.1.5 Deliverables 1.2

Write a report detailing your execution of the experiment, including all observations made while running the robot. Especially detail and visualize the observed robot end poses and report statistical precision of the observed end poses. Summarise your findings, does the observed behaviour of the robot matches your expectations? 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 one understand the outcome of the experiments (see also next week)
4. The observed data as Excel or LibreOffice Calc file or in a similar machine-readable form
5. Visualisation of the end poses as well as visual documentation of all aspects of setup and execution you deem important
6. An estimate of the accuracy and precision of the observed data, including how you arrived at these estimates and why they are plausible

2.2 Task 2: Statistical Evaluation of the Previous Experiment

Last week, you ran 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 terms of statistical parameters.

2.2.1 Setup

No specific hardware setup; you will use the data obtained in the previous assignment.

2.2.2 Task

Estimate the distribution governing the spread of the stop positions. Try to fit a multidimensional Gaussian for each of the three experiments and judge the fit: Does your data fit to a Gaussian? If not, which other distribution do you know that might be a good fit? Why? If you get a poor fit for a Gaussian, which effects may have caused 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 be fully attributed to random noise or is the noise (and hence the Gaussian) distorted by systematic effects originating either from your experimental setup and/or from the true characteristics of your robot's 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 necessary, rerun the experiment from last week to obtain usable data.

2.2.3 Assignment 2

1. Fix and, if necessary, rerun your experiment according to the feedback in class
2. Compare the observed and commanded motions: do they fit? if not, describe possible causes for this
3. Fit a Gaussian to your data and judge if the data is truly Gaussian or if it follows another distribution
4. Compare the estimated uncertainty of the measurement process itself with the statistical uncertainty of the observed end poses

2.2.4 Deliverables 2

Update your previous week's report and add a description of the test for a match with a Gaussian distribution as well as a description of the observed differences between the observed and the computed motion and the actual and expected accuracy and precision. Include appropriate figures, diagrams, and images backing up any claims you make. The report must be self-contained and provide enough details to support any statement you make. If applicable, include a section on problems encountered. Your update should cover:

1. Any possible pre-processing of your data, like outlier detection and removal
2. Fit of a Gaussian, either two individual ones in the x and y directions for each of the three cases or a proper two-dimensional distribution per case
3. Check whether the data are actually distributed according to a Gaussian distribution
4. List of used software, including source of any function you wrote for performing your analysis
5. An answer to the following question: When analysing the data with respect to the executed motions, which characteristic of the data do you establish here: the accuracy, the precision, or both?

2.3 Task 3: Calibrating an Optical Tracking System

This task consists of two assignments.

Up until now, you were observing your robot's motion manually, which introduces a hard-to-estimate human error in your data. In the following weeks, you will use the optical tracking system in the AICISS Lab for automatic position tracking. In contrast to the previous experiment, this tracking system will not only obtain the start and stop positions of your robot, but will also record its complete trajectory.

The system in the AICISS lab is calibrated and ready to use; however, your first task is to perform one of the calibration steps by yourself.

2.3.1 Setup

You will use a provided high-quality web cam mounted on a tripod or other fixture and a calibration tool to determine the intrinsic camera parameters of the provided cameras. The actual calculation will be done using the camera calibration toolbox from Matlab.

2.3.2 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 and read and understand its manual. Create a calibration tool (checkerboard) as required by the Matlab toolbox and perform the necessary image captures to allow the toolbox to compute your camera's optical parameters. In particular:

1. 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, take a look at `http://www.vision.caltech.edu/bouguetj/calib_doc/htmls/example.html`

2. Mount the camera on a tripod, build a checkerboard that you can present to the camera from different perspectives, and run a calibration using an appropriate number of images to ensure reliable results

3. 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

2.3.3 Assignment 3.1

Read and understand the papers from Zhang and Heikkil (provided in LEA). Based on your understanding of camera calibration, design a calibration setup and detail the actual calibration process.

Test the provided camera on your laptop and ensure you can capture still images.

2.3.4 Deliverables 3.1

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

1. A description of the setup for calibration, including possible pitfalls
2. An estimation of the number of images and image positions required
3. A description of the parameters (what do they mean?) calculated by the Matlab calibration toolbox
4. Discuss possible problems or error sources that can disturb the calibration process. Include any observation you may have made while testing the proper functioning of the camera with your laptop

2.3.5 Assignment 3.2

Perform the camera calibration experiment designed in assignment 3.1. Document all relevant aspects needed to asses the quality of your obtained camera parameters.

2.3.6 Deliverables 3.2

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

1. Describe the images poses used for calibration and report the found camera parameters including any error estimates (where applicable)
2. Provide arguments for which camera model best fits your situation and hence should be used
3. Discuss possible problems or error sources that can disturb the calibration process

2.4 Task 4: Probability Distribution of the Robot's Drive System

This task consists of two assignments. Our final goal is to estimate the four empirical parameters of the generic motion model $x_t = f(x_{t-1}, u_t, \alpha_{1:4})$ described in part A of this handbook, where u_t is a commanded v, ω motion and x_t is the robot's pose, such that the probability $p(\hat{x}_t|x_{t-1}, u_t)$ of the robot ending up in pose $\hat{x}_t$ given a previous pose x_{t-1} and an attempted motion u_t is the same as the observed one 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 x_t predicted by our model is the same distribution as the observed distribution. In order to fit the model parameters $\alpha_{1:4}$ in $f(x_{t-1}, u_t, \alpha_{1:4})$ 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. This week, you will determine the distribution for the real robot.

2.4.1 Setup

You need to have the robot programmed to run on arcs for a fixed time and the optical tracking system.

2.4.2 Task

1. Verify the accuracy of the optical tracking system by measuring at least 4 representative robot positions and comparing with ground truth

2. Program your robot to follow trajectories as indicated in Figure 1. Use three different translational velocities v (excluding zero) and five different rotational velocities ω (including zero and symmetrical ones regarding clockwise/counter-clockwise, example: $20^o/s$, $10^o/s$, $0^o/s$, $-10^o/s$, and $-20^o/s$). In total, these are $15v, \omega$ combinations, each of which will later be executed 10 times.

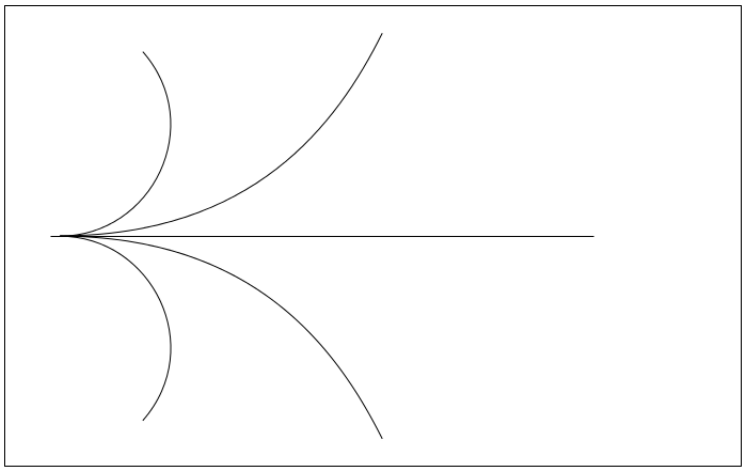


Figure 1: Robot trajectories for estimating the drive system characteristics

3. Record the robot's trajectory and determine the start and endpoint for each motion. Make sure that the robot runs with constant velocities while tracking it, i.e. use the robot's position

a fixed short time after the start of motion as a start point and the position shortly before it stops as an 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 a multivariate Gaussian representing the average 3D pose (2D position and 1D orientation)

5. Similarly, determine the uncertainty of the start position for each v, ω combination. by building a multivariate 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

7. Estimate the uncertainty of the tracking system and compare this estimated uncertainty with the uncertainties determined in step 6

2.4.3 Assignment 4.1

Familiarize yourself with the AICISS optical tracking system. How does it work from a users perspective? How to prepare the robot and how to retrieve the pose of the centre of motion of the robot? Obtain ground truth for a number of selected poses and verify the correct working of the tracking system.

2.4.4 Deliverable 4.1

Write a short report summarizing the operation of the AICISS optical tracking system. Your report should cover:

1. The relevant aspects of the design of the robot, especially how you attached the marker
2. The method to retrieve the pose of the centre of motion of the robot
3. Documentation of the tests conducted to ensure correct operation of the system
4. An estimation of the pose measurement error based on your tests

2.4.5 Assignment 4.2

Perform the robot runs requested in the task 4 description. Record the runs including timing information using the AICISS optical tracking system.

2.4.6 Deliverable 4.2

A written report covering your observations, 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. The program and parameters used to drive the robot
2. Any observation made during the execution, that may help to understand the outcome of the experiments
3. Visualisation of the obtained traces (make sure that the aspect ratio of the coordinate axes is the same)
4. The calculated uncertainties for the start and stop positions, as well as any notable characteristics of the data and a comparison of the uncertainties
5. An analysis of the data quality for the start and end poses as well as for the obtained traces
6. The observed data as an Excel or LibreOffice Calc file or in a similar machine-readable form

2.5 Task 5: Estimating the Model Parameters

This task has one assignment, but four distinct steps, all of which will 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 the same as the real robot. This involves four substeps:

1. Building a sufficient approximation of the error-free model
2. Determining and eliminating any systematic errors in u_t using the error-free model
3. Guessing sensible start values for the iterative optimisation of the $\alpha_{1:4}$ parameters
4. Iteratively:
   (a) Computing the resulting distribution for the current parameter set
   (b) Determining a new parameter set based on current mismatch

2.5.1 Setup

No specific hardware setup necessary.

2.5.2 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, the 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; therefore, 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 $v_{eff} = k_v v + b_v$ and $\omega_{eff} = k_\omega \omega + b_\omega$.

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 the 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 $k_v, k_\omega, b_v, b_\omega$ that minimise the sum square error for the expected vs. average measured position over all 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, but 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, so there is no hope to determine $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ analytically. We thus need to find them using iterative, numerical optimisation.

Most optimisation methods require a not-too-bad start value. In this step, you should guess (by carefully looking at 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 the 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 the observed and the computed distribution falls under a given threshold.

There are a number of methods to compare distributions and 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 multidimensional optimisation, designed to work in cases were no derivatives of the target function is known - as 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[1] 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 multivariate 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 any systematic errors).
- Decide on a measure for the difference between two distributions. The *Kullback-Leibler divergence*[2] or the *Mahalanobis distance*[3] 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 single set of $\alpha_1, \alpha_2, \alpha_3, \alpha_4$. This means that 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 $\alpha_1, \alpha_2, \alpha_3, \alpha_4$

[1] `https://en.wikipedia.org/wiki/NelderMead_method`
[2] `https://en.wikipedia.org/wiki/KullbackLeibler_divergence`
[3] `https://en.wikipedia.org/wiki/Mahalanobis_distance`

2.5.3 Deliverable 5.1

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 systematic 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 reasons 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 combinations.

6. Provide the overall convergence error at end of the optimisation and the final parameters found.

2.6 Task 6: Measuring the Accuracy and Precision of a KUKA youBot Arm

This experiment is designed to estimate the *accuracy* and *precision* of a robotic arm. The experiment involves a robot performing different *pick and place* tasks; in particular, our robot (a KUKA youBot arm) will be placing objects with three different masses and in three different poses (straight, left, right). The object **poses** $(x,\ y, \theta)$ are determined by tracking *ArUco* markers that are placed on top of the objects; in particular, an external vision system (a camera) is placed above the arm's workspace in order to determine the marker poses.

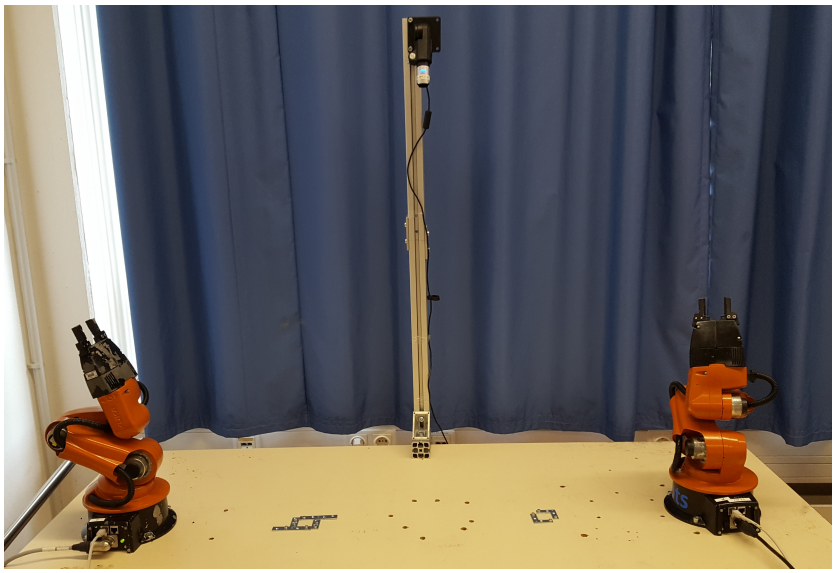


Figure 2: KUKA youBot arms used for performing pick and place tasks together with an external vision system (a Microsoft LifeCam)

Your task is to perform the experiment 20 times for each object-place combination, which results in 180 experimental trials in total. At the end, you need to perform a statistical analysis of the data in order to estimate the achieved accuracy and precision of the youBot arm motions.

2.6.1 Setup

Two KUKA youBot arms (IDs 1 and 3) are placed in our C022 lab and both of them are available for performing the experiment (that is, two groups can run the experiment at the same time). In particular, each group of students will use **one** youBot arm to conduct the experimental trials. Each arm is preprogramed to perform three different placing motions; the ground-truth initial and final poses for each of the three motions with respect to the camera frame are given below:

Table 1: youBot arm 1

Pose	$x(cm)$	$y(cm)$	$\theta(rad)$
Pick	-85.41	-55.60	1.38
Straight	-86.89	-71.31	1.45
Left	-45.47	-45.00	0.88
Right	-112.13	-70.58	1.78

Table 2: youBot arm 3

Pose	$x(cm)$	$y(cm)$	$\theta(rad)$
Pick	-79.27	-99.87	-1.62
Straight	-79.40	-87.48	-1.72
Left	-109.65	-102.75	-2.10
Right	-41.73	-88.63	-0.92

A complete list of the hardware and software components that are provided for performing the experiment is given below:

Hardware

- A KUKA youBot arm placed on a table in the C022 lab
- A computer used for controlling the arm
- Objects with three different masses; these will be used by the robot for picking and placing
- Visual markers attached on the top of the objects
- An external camera placed above the robot's workspace and connected to an additional PC; you are not going to use this PC explicitly
- A fixed container on the table which ensures that the initial object position is kept constant throughout the experimental trials

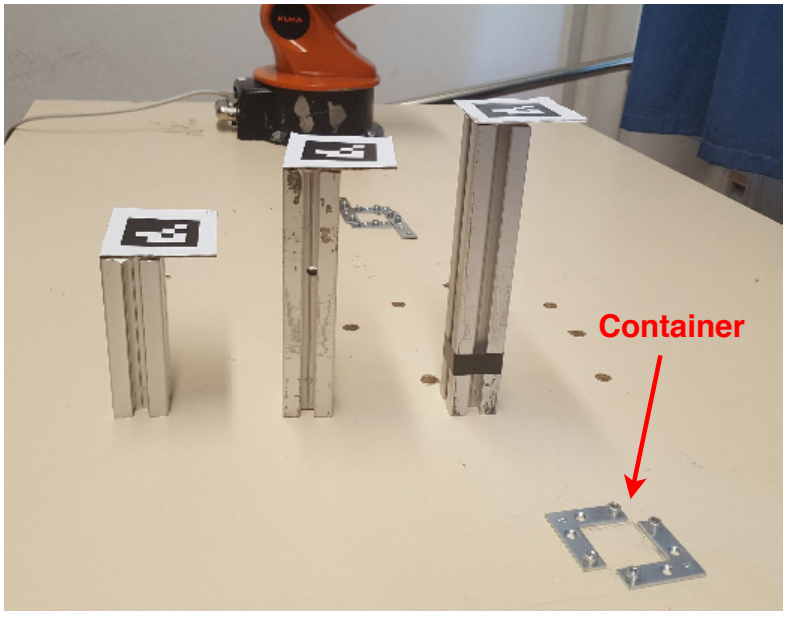


Figure 3: Objects with different masses and containers used for initial object placement

Software

- KUKA youBot drivers required for controlling the arm; these are set up on the provided computer
- Additional control script with pre-saved pick and place poses for the arm (also available on the provided computer); you will be using this script to command the robot to move in one of the three predefined placing poses
- A marker pose subscriber script (available for download on LEA); this script subscribes to the poses of the ArUco markers that are attached to the objects, which are published over ZMQ by a camera server

Guidelines For making sure that the experiment is performed correctly, several conditions must be satisfied throughout the experimentation process:

- An object should be placed in the container only **after** the arm has reached a pregrasp pose (see figures 4 and 5).

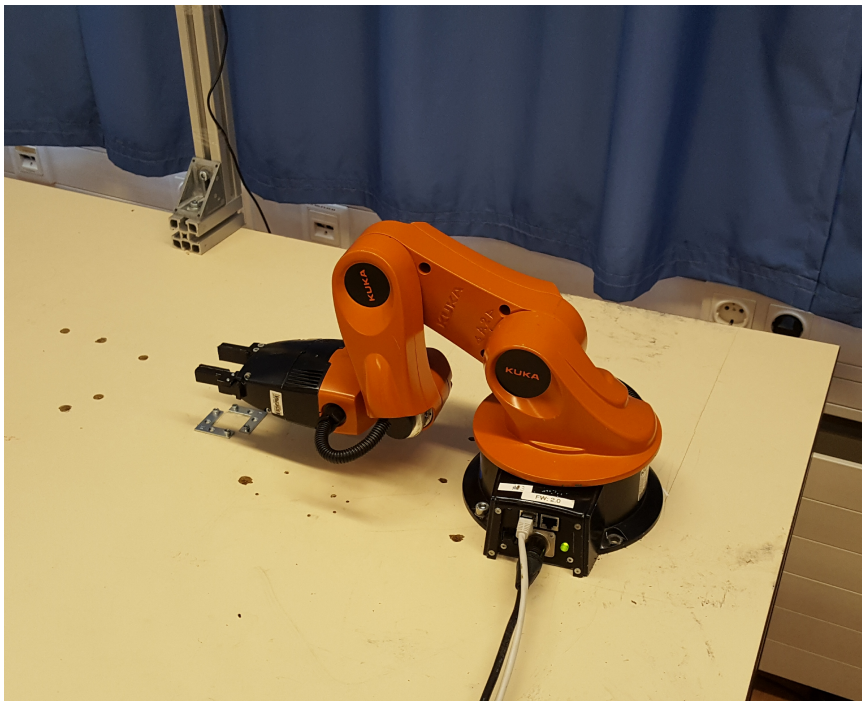


Figure 4: A KUKA youBot arm in its pregrasp pose

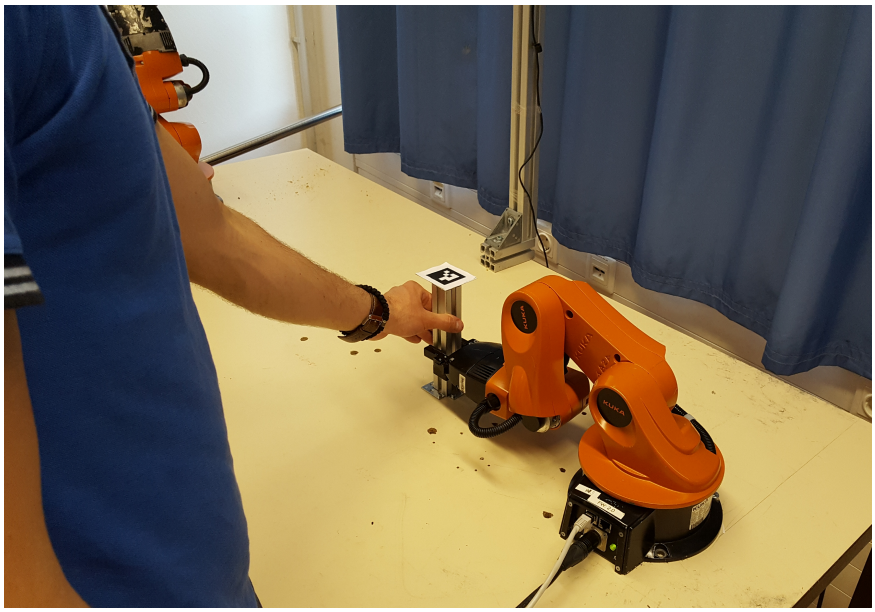


Figure 5: An object placed in the container after the arm has reached the pregrasp pose

- The chosen object must be placed in the container properly in order to ensure that the picking pose is always the same. More precisely, the object must not be touching any part of the upper container's surface, i.e. it must be placed **in between** the four borders of the container (see figure 6).

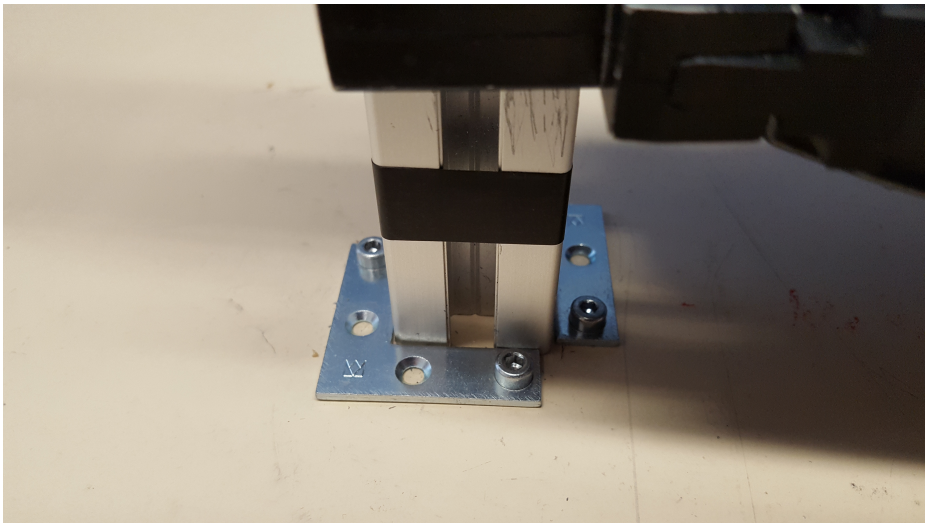


Figure 6: An object placed in the container; this ensures that the initial (grasping) position is constant

- Additionally, you must ensure that the object is placed in the container with a constant orientation. i.e. the marker attached to the object must always have the **same orientation** (in order for the ground-truth orientation measurements to make any sense). The correct orientations for the objects used by arms 1 and 3 are shown in figures 7 and 8 respectively.

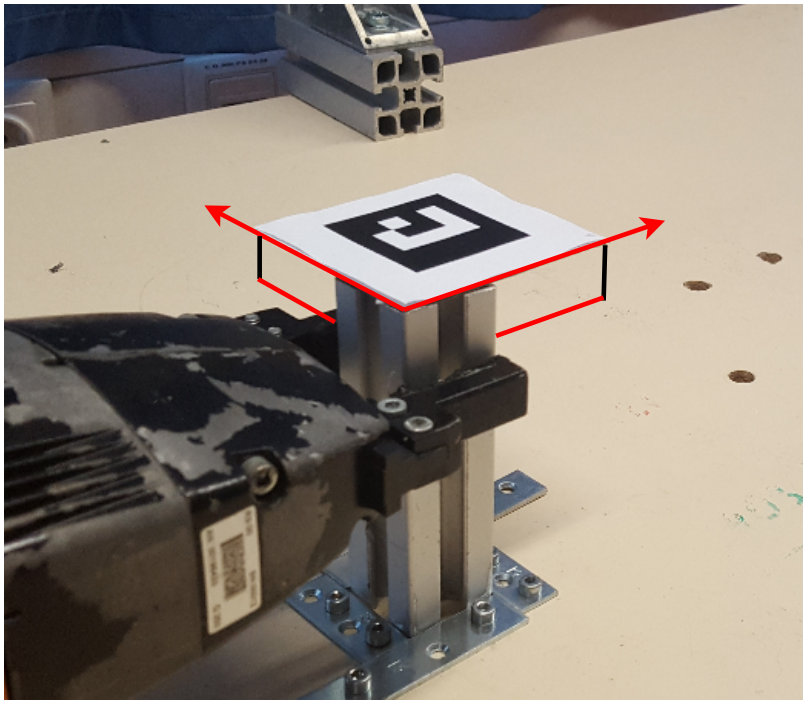


Figure 7: Correct orientation of an object used by youBot arm 1

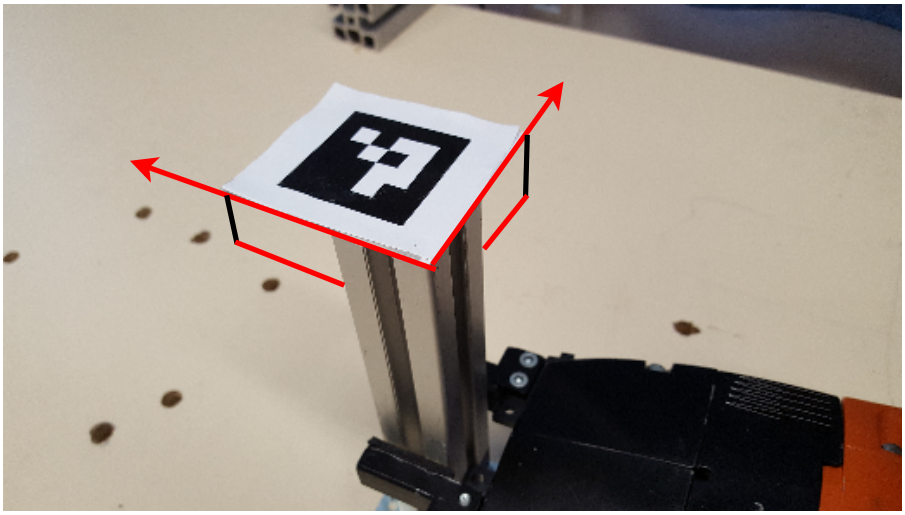


Figure 8: Correct orientation of an object used by youBot arm 3

- You must ensure that there is no object in the workspace of the arm while it is performing a motion since collisions with objects may damage the arm.
- In case of an **emergency situation**, the arm can be shut down by pressing the **small green button** that is located next to the arm's power cord (see figure 9). If such a situation occurs, the arm must be held while the button is being pressed; this is necessary to avoid the arm falling on the table and damaging itself. After the arm's motion is stopped, the arm should be set in its folded configuration (which is illustrated in figure 2).

Figure 9: The emergency (shutdown) button on a KUKA youBot arm. A green color of the button represents the running state of an arm, while a red color of the button would represent the state where all motors and controllers in the arm are shutdown.

- Lastly, while performing the experiment, the light distribution in the lab should be **uniform**. In order to satisfy these lighting conditions, **close the lab curtains completely** and **turn on the lights** in the lab (as shown in figure 2). These conditions are important for producing good object pose estimates by the external vision system.

2.6.2 Experiment Workflow

Step 1 The first part of the experiment involves running the software components that are required for controlling the arm and recording observable poses for all object-place combinations.

Before starting the experiment, you should make sure that the selected youBot arm is turned on. Additionally, you must connect the control computer to the corresponding youBot arm. Each given computer has a preconfigured static Ethernet connection with the arm (called *youBot_static_connection*). For using the control software components, it is necessary to open four terminals on the provided computer and run the following components:

```
roscore
roslaunch youbot_driver_ros_interface youbot_driver.launch
roslaunch youbot_moveit move_group.launch
roslaunch youBot_placing_experiment youBot_placing.launch
```

In order to familiarize yourself with the provided setup and the task in this experiment, you should see the example videos provided on LEA. The experiment should be performed 20 times for each object-pose combination. More specifically, the robot should be placing objects with three different masses and in three different positions (straight, left, right); this results in nine object-pose combinations, so the complete experiment involves 180 experimental trials.

As already described before, an external vision system is observing the poses of the objects that are placed by the arm. The generated observation data is published over the lab network using a ZMQ publisher that runs on a separate machine. For collecting the observation data, namely the values of the observed object poses, you need to download a ZMQ subscriber script (*subscriber.py*) from LEA. The computer on which the ZMQ script is downloaded and used is arbitrary, i.e. you can use your own laptop for this. Before running the subscriber, you should connect your computer to the lab WiFi network (the password is written on the router in the lab). For running the script, the following command should be executed in a terminal:

```
python <path_to_downloaded_script>/<script_name>.py <marker_ID>
```

Here, the *marker_ID* argument can take the values 0 or 1 depending on the youBot arm that is used in the experiment: youBot arm 1 should be using the objects with marker ID 1, while youBot arm 3 should be using the objects with marker ID 0. If these conditions are violated, the data received from the subscriber script will not correspond to the ground-truth values and the results of the experiment will not be fully usable. The marker IDs are written on the white surface of the corresponding markers.

For each experimental trial, you are advised to take multiple object pose readings in order to estimate the final pose of an object after placing. This is because the camera pose estimates may have slight variations due to camera errors - even when an object is static - so by taking multiple measurements and filtering those, we are minimizing the effect of this error. The best way to obtain such data is to run the subscriber script only after the arm has completed the placing task, i.e. after the object has been placed and the arm has moved back (as shown in figure 10 and the LEA instruction video).

Figure 10: A KUKA youBot arm in its post-place position in from of a placed object

The filtering process of the pose estimates involves (i) detecting and removing any outliers from the noisy data and (ii) averaging the readings so that a single experimental trial is represented by a single pose. You should however also keep the raw pose measurements (before filtering) because we want to use them when analysing the data later.

2.6.3 Deliverable 6.1

Run the complete experiment with a KUKA youBot arm, i.e. run all 180 experimental trials and perform any necessary preprocessing of the data. You should submit a written report covering your observations, including appropriate figures. Your report should include:

1. Any observations made during the execution that may help to understand the outcome of the experiments; for instance, are there any particular sources of error that may affect the results of the experiment?

2. A description of the pose filtering procedure you used and any observations that you may have made during the filtering process (e.g. on average, how many outliers are there per single experimental trial)

3. The saved preprocessed data as Excel or LibreOffice Calc file or in a similar machine-readable form

4. A visualization of the obtained final object poses. You are free to use any suitable visual representation for the data (using what you have already learned during the LEGO experiment). *Hint*: Given the different object-place combinations in the experiment, think about what we might want to illustrate with the visualization (e.g. the distribution of poses per object? the distribution of poses per motion direction? the distribution of all poses?)

Step 2 After performing the experiment and gathering all the necessary data, you need to analyze the data and estimate the accuracy and precision of the used KUKA youBot arm. We are particularly interested in the following:

- Knowing the ground-truth values of the expected final object poses, what is the accuracy of the arm? What about its precision?
- Does the distribution of the final object poses follow a Gaussian distribution?
- How much does changing the shape and mass of an object affect the final placing pose? Is this effect statistically significant? If the effects are indeed significant, what might have caused them?
- If we are to use a single camera measurement for determining the final object pose instead of multiple filtered measurements, is the effect on the final pose estimates significant?

For analyzing the data, you should choose a statistical analysis technique that is most suitable for answering the above question (once again using what you have already learned during the LEGO experiment). *Hint*: Statistical significance can only be determined by performing appropriate hypothesis tests.

2.6.4 Deliverable 6.2

Update your previous week's report and include the following:

1. If necessary, rerun your experiment according to the feedback in class and describe how the new run improves on the previous run
2. Find suitable statistical techniques for analyzing the experimental results (using what you've learned during the LEGO experiment) and answer the list of questions mentioned above
3. Mention the list of used software and include the source of any functions you wrote for performing your analysis