MTRN4010/2023 PROJECT #1

Part A - Dead reckoning localization.
Part B - Feature extraction from LIDAR data.
Part C - Mapping LiDAR scans in the global coordinate frame.
Part D - Data association.
Part E – Basic calibration.
Part F - Applying deterministic localization (triangulation/trilateration).

Part A requires implementing a dead-reckoning process. This process generates predictions of the platform's pose, based on a kinematic model, which has speed and angular rate measurements as inputs.
Your program should perform the prediction of the platform's poses during the full duration of the test, by properly applying the kinematic model. In addition, the estimated platform's pose, at the times of each "sensor event" (see note 1, about sensor events) must be recorded in a buffer, for being plotted at the end of the run for thus being able to validate your results.
In Project 1, some of the datasets we use are almost free of uncertainty (marginal bias in gyroscope's measurements, minor quantization errors due to discretization of the analog signals), consequently your solution should be very accurate. Achieving good accuracy in this part of the project, means that you are properly applying the kinematic model (you are debugging it before exposing it to more complicated cases in which sensors' measurements are polluted).

Part B focusses on processing LiDAR scans, individually, performing "Feature Extraction". Outputs generated by this module will be crucial for the whole project; however, you are developing and testing it separately, to be sure it performs properly, before being integrated with other modules. Each time at which there is a LiDAR event, you will process its associated LiDAR scan. The result of the processing will be the detection of OOIs (potential navigation landmarks) that may be present in the scanned area. All the results (positions of OOIs) are to be expressed in the LiDAR's coordinate frame, in Cartesian representation. Those results are required to be shown in a figure, dynamically, each time a new LiDAR scan is available.
For solving part B, you do not need part A, consequently, you can solve them (A and B) in any order.

Part C requires you to combine parts A and B, for dynamically showing the LiDAR scans in the global coordinate frame. For that to be feasible, you will exploit the estimates, which you generate, of the platform's pose at the time of the LiDAR scan (as you keep estimating the platform pose at the time of each event (the "present time").

In addition to the raw scans, you are required to dynamically show, in global coordinate frame (in the same figure), the OOIs you detect by exploiting your solution in part B.

Your visualization in the global frame is also required to show additional information.

1) Certain static objects, such as landmarks and walls.
2) To dynamically indicate the current position of the platform (by simply using a colored symbol).
3) The scan provided by LiDAR #2, using a color different to that used for showing LiDAR 1 scans (just the scan's points, as you are not, for the moment, required to detect OOIs in scans from LiDAR 2).

Part D focuses on implementing a process called "Data Association" (which will be described in Lecture 4). Then part A and part B need to be successfully solved before attempting part C. To verify the accuracy of your solutions of part A and part B follow the procedure described in the section "validating the accuracy of your solution".

Part E requires implementing off-line calibration approaches, for estimating certain system parameters (e.g., LiDAR's position or its orientation on the vehicle), gyroscope bias, etc. By improving our knowledge about those parameters, we can improve the performance of the classic estimation processes we apply in Project.

Part F required implementing a classic approach for localization, based on processing measurements from sensors such as the LiDAR, and on exploiting a navigation map (map of known landmarks). The concept about localization based on a map is to be discussed during Lecture 4. In this part of the project, you will be free to decide how to solve the task, which requires solving a set of equations, for which your knowledge in Mathematics would help.

Details about parts D, E and F will be released Monday week 4, and explained in lecture time, during which the lecturer will show his solution working. You work during week 3, until the lecture on week 4, should be focused on properly implementing the required processing modules, for parts A,B,C.

Deadline for submission, of the full project, is Tuesday Week 7 (30/March), 23:55
Submission will be via Moodle. Details about how your program files must be organized (names, author details) will be specified in the release document of parts D and E.

Marking criteria

Project 1 if 100% successfully completed provides 23 points of the course final mark.

In addition to the submission of your implementation, you need to demonstrate, to a member of the teaching staff, that your submitted program is working.
Both, submission and demonstration are necessary conditions. A project which is not submitted or not demonstrated will get no marks.

The **demonstration will take place one week after the nominal submission deadline**, and it will be based on the submitted material (which is to be kept, securely, in the Moodle submission repository).

Your achieved project mark depends on the implementation and demonstration of the project parts, and on a knowledge factor about the project (variable **Q** in the marking equation).

The relevance of the implementation and demonstration of the project parts is as follows:

Part A:	up to 15% of the project mark (= 0.15*23 marks)
Part B:	up to 20%
Part C:	up to 15%
Part D:	up to 20%
Part E:	up to 10%
Part E:	up to 20%

The addition of the values obtained in each part is the “Submitted and Demonstrated Project Mark”.

The factor Q is obtained based on your performance answering questions, during the demonstration, and/or via a quiz if needed. Factor Q is represented in scale [0:100]

The influence of Q in the overall project mark can be seen in the following formula.

Overall Project Mark = [Submitted and Demonstrated Project Mark] * (0.6+0.4*Q/100)

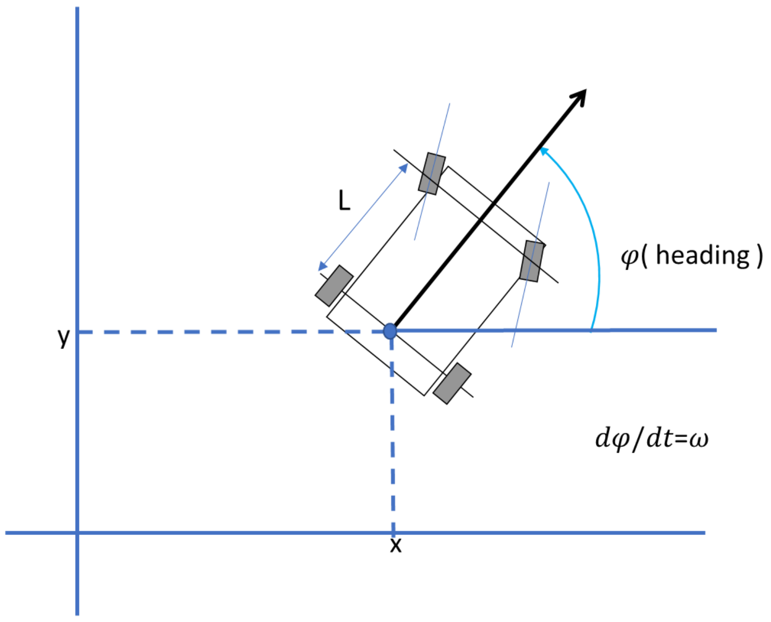
For instance, if you fail in answering all the questions, your Q factor will be 0, which means you would get 60% of the achieved marks of your submitted/demonstrated programs.

Questions: Via Moodle Forum or email to lecturer (j.guivant@unsw.edu.au)

Part A – Localization based on Dead reckoning.

Estimation of the vehicle position and heading

Implement a module for estimating the platform position and heading, based on a kinematic model, and on measurements of the speed (longitudinal velocity) and heading rate.



The measured values and their timestamps are provided in the data (whose format is explained in document "**data_structure_Project_1_MTRN4010.pdf**", or, alternatively, it may be easily inferred from the example program.

The platform's initial pose is included with the provided data for this project, as well.

Your module must be implemented in a way to allow being integrated with the LiDAR processing.

Following the approach shown in the example code (file "**ExampleUsingDataProject1_2023.m**"), would allow such capability.

You will be able to validate your results (accuracy of your estimates/predictions) by comparing your achieved trajectory with that of a provided ground truth (which is included in the data file itself).

The section "Validating results" gives information about how to infer if your implementation is working well.

Part B - Feature extraction from LIDAR data

Detection of the objects of interest

Implement a function for processing individual scans (generated by one of the onboard LiDARs); to detect objects of interest (OOI), from the raw measurements provided by the sensor.

We consider as OOI any object that seems to be a pole, i.e., that has a defined size (apparent diameter between 5 and 20cm), and that has certain color (At least one of the pixels that constitute the segment, must have intensity bits >0).

A LiDAR scan usually detects multiple clusters of points, being some of them associated to our OOIs (which are small, i.e., are poles having diameters of 10 cm) . Most of the scan's points are usually associated to walls and other massive objects that are "background", which is usually useful, but which we do not directly use in our task (except for some visual validations).

In addition to the apparent size of a cluster of points, to be certain, you will need to verify that the "color" of the OOI is "brilliant". If that condition is satisfied, you will consider that cluster to be an OOI (because its size and brightness gave sufficient evidence to the object to be considered an OOI)

Details about how to extract range and "color", from the LiDAR raw scans, can be found in the example code (which is mentioned in part A).

The input data (LiDAR scan), to your processing function/module, is assumed to be a vector, of size 301 x 1, class uint16 (16 bits unsigned int). The vector's content is the raw data, associated to one LiDAR scan, whose FoV is [-75:+75] degrees, having angular resolution 0.5 degrees (so that in contains 301 ranges, covering the FoV).
For each segment (cluster of points) that seems to compose an OOI, you need to estimate its center of geometry (CoG).
The output of your function will be a list the positions of the detected OOIs. The CoGs are expressed in Cartesian representation, in the LiDAR's coordinate frame.
The way you organize the function's output data is your decision, as it will be used in subsequent parts of the project, by you.
You can solve this part of the project by implementing your approach, or by using a third-party implementation (in that last case, you will need to clearly indicate the source of that tool, and that source should be external to UNSW). As our problem is specific, it can be well optimized, resulting in that your implementation may be better than other options you may get from the web.
Keep in mind that some solutions which are publicly offered may have been designed for other purposes, consequently requiring more computational cost than that which is needed for our specific case. Your solution is expected (and required) to process a LiDAR scan, **in less than 10ms**, in a normal PC (such as those in the Lab). That processing time does not include the extra time for refreshing plots when you test your solution. A well implemented solution would take less than one millisecond to process the LiDAR scan; however, we do not require such performance, but we specify a top limit on the *average* processing time (10ms).
We will tell you how to test that average processing time, during the lab/tutorial sessions.

Part C – Position estimation of the objects of interest in GCF Data Association

You are required to combine parts A and B. For each LiDAR scan that is available, you will estimate the position, in the global coordinate frame (GCF), of the detected OOIs. Figure 1 shows the configuration in which the LiDAR is installed on the platform. The sensor is perfectly aligned with the

platform's chassis; however, it is displaced from the point being modeled by the kinematic model. The parameters Lx and Ly (which define the position of the LiDAR in the cars' CF) are included in the provided data (read the example program, for details about reading the parameters and other components of the provided dataset.)
When those OOIs are expressed in the GCF, a "data Association" process must be performed. Some of the detected OOI may correspond to certain poles, called "Landmarks", whose positions in the GCF are well known by us (e.g., it may be the case we installed them, and surveyed their positions for navigational purposes!).
For that purpose, a navigation map is provided (a table listing the known landmarks and their positions expressed in the GCG). The provided example program reads that map, and plots the map's landmarks' in a figure.
A basic approach for solving this problem, will be described by the lecturer, in the lecture on week 3.

Validating the accuracy of your solutions part A, B and C

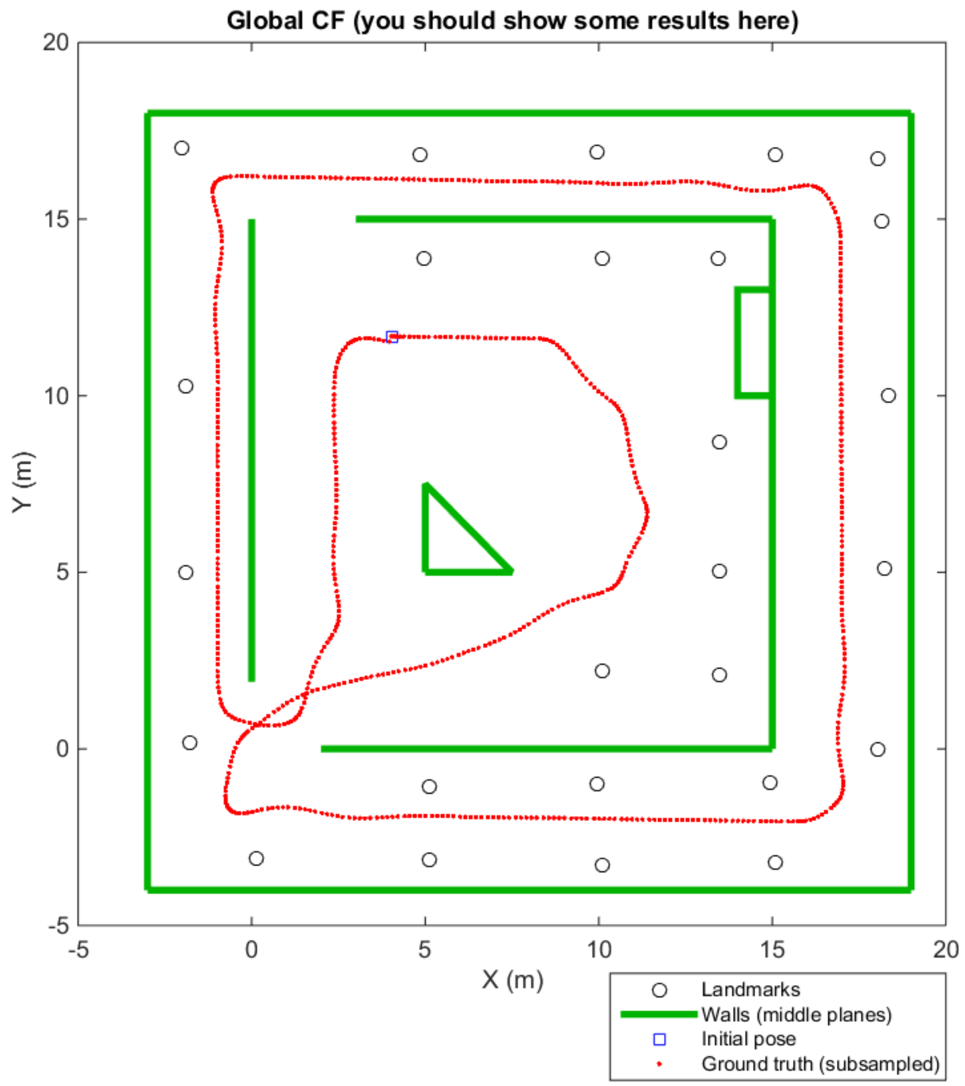
The performance of each of the project parts needs to be evaluated.

For part A, you will compare your estimated path of the platform with the ground truth, which is provided in the dataset itself. (The provided example program shows how to read it from the data.)
Discrepancies of less than 3 centimeters (in the estimates of position) are required for the cases in which the dataset just includes measurements "free of noise" (like the dataset named "DataUsr_002.mat", in this release).
In addition to that, a second validation is considered. This validation is based on visual inspection of the LiDAR scans which are projected to the GCF.
If the walls and poles seem to be properly shown in the GCF, we will assume your pose estimates and lidar processing are correct, in addition to your implementation of coordinate transformations.
"Properly shown" means that are almost free from drifts and other artifacts as result of errors in the implementation and the use of the Kinematic model.
The dataset also provides the positions of the walls and landmarks (expressed in the GCF), which were present in the area of operation, in that particular test.
Your LiDAR scans, when shown in the GCF, should match part of those walls. As the walls have a thickness of 20cm, the LiDAR points will ideally appear at 10 cm respect to the axial central planes of

the walls. The lecturer will show that effect (when presenting his solution, as an example of solution), during the lecture on week 3.

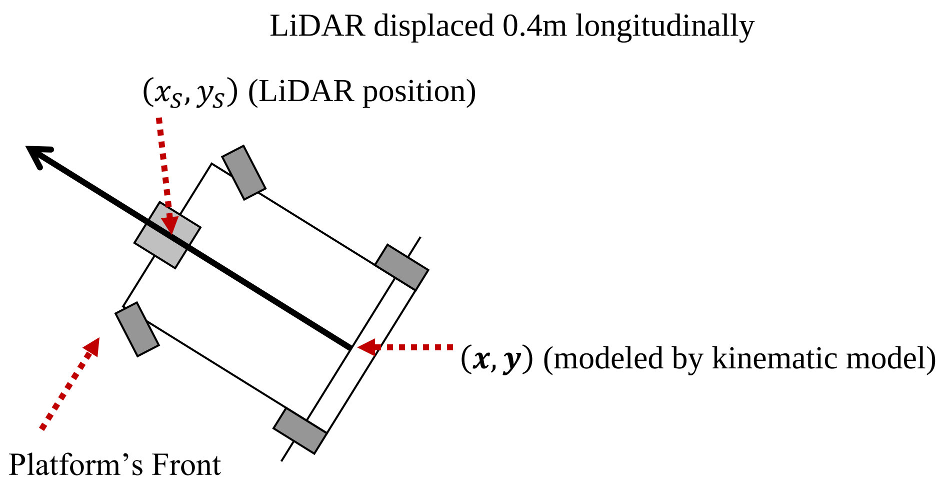
In addition, for part B, the center of geometry of each of the OOIs, when represented in the GCF, must appear close to one of the landmarks (less than 5cm away). Of course, you can, alternatively, express the landmarks but in the LiDAR CF, for performing the same comparison, at that LiDAR event time. The evaluator of your demonstration will consider those validations, to verify the accuracy if your solution.

The example program shows how to read that validation data from the dataset file. An example is that data being presented in a Matlab figure, is shown in the following figure.



The dataset contains a subsampled ground truth (of the platform's pose), plotted in red dots, in this figure. The map landmarks are shown by black circles, and the middle planes of the walls, are shown by green lines. All those data components are useful for validating your solutions. This figure was generated by the provided example program, from which you can see how to read that validation data.

Figure 1: LiDAR position


LiDAR displaced 0.4m longitudinally
(x_S, y_S) (LiDAR position)
$(\boldsymbol{x}, \boldsymbol{y})$ (modeled by kinematic model)
Platform's Front

LiDAR position on the platform needs to be considered (Configuration parameters are provided in that data files)

.

Parts D,E and F will be released during this week. However, you need to solve, first, parts A ,B and C. Detailed marking criteria for each of the parts will be also indicated in the complementary release. A set of specific individual requirements will be listed, being each of those items worth a percentage of the full mark of each part (A,B, etc.)

Please, what the video in which an example of possible solution is shown. In addition, we also suggest reading and playing with the example program. You may use it as an initial program, to gradually add necessary components. You can modify that program in any way you prefer. You may simply read the program to understand how to use the data, and thus proceed implementing your program from scratch.