PROJECT #1

Part A - Dead reckoning localization.
Part B - Feature extraction from LIDAR data.
Part C - Basic data association.
Part D - Processing real data.
Part E - 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, as it has been seen in problem 4 of tutorial 2.

Part B focusses on processing LiDAR scans, individually. 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.

For solving part B, you do not need part A, consequently, you can solve them in any order.

Part C is focuses on combining parts A and B, and on implementing a process called "Data Association" (which will be described in Lecture 3.). 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 D focuses in working with data from a real platform, dealing with noise and other issues for which our deterministic approach is not well suited.

Part E focuses in implementing a classic approach for localization based on processing measurements from sensors such as the LiDAR.

Deadline for submission, of the full project, is Tuesday Week 7 (29/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 (Q).

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

Part A:	up to 15% of the project maximum mark (23 marks)
Part B:	up to 20%
Part C:	up to 11%
Part D:	up to 27%
Part E:	up to 27%

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 - Dead reckoning localization.

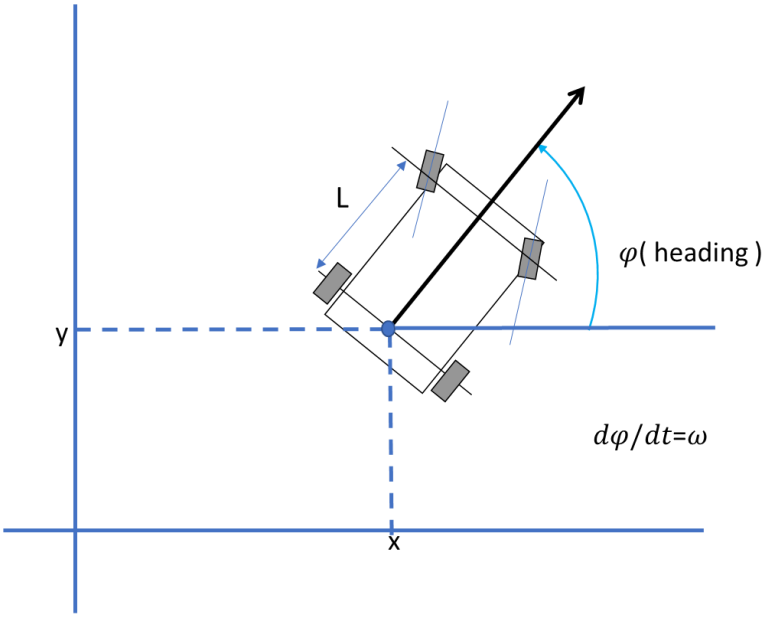
Estimation of the vehicle position and heading

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.

Kinematic Model

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ \varphi \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} v \\ \omega \end{bmatrix}$$

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}, \mathbf{u})$$



This part is similar to problem 4 of the tutorial 2.

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

The module must be implemented in a way to allow being integrated with the LiDAR processing. Following the approach shown in the example code, allows such capability.

Part B - Feature extraction from LIDAR data

Detection of the objects of interest

Implement a function for processing individual scans (generated by LiDAR); 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.

A LiDAR scan usually detects multiple surfaces, being some of them associated to our OOIs (which are small, i.e. are poles having diameters of 10 cm.)

The input data is assumed to be a vector, of size 321 x 1, class uint16 (16 bits unsigned int.)

The vector's content is the raw data, associated to one LiDAR scan, whose FoV is [-80:80] degrees, having angular resolution 0.5 degrees (as those used in Tutorials 1 and 2)

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).

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 simple 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.

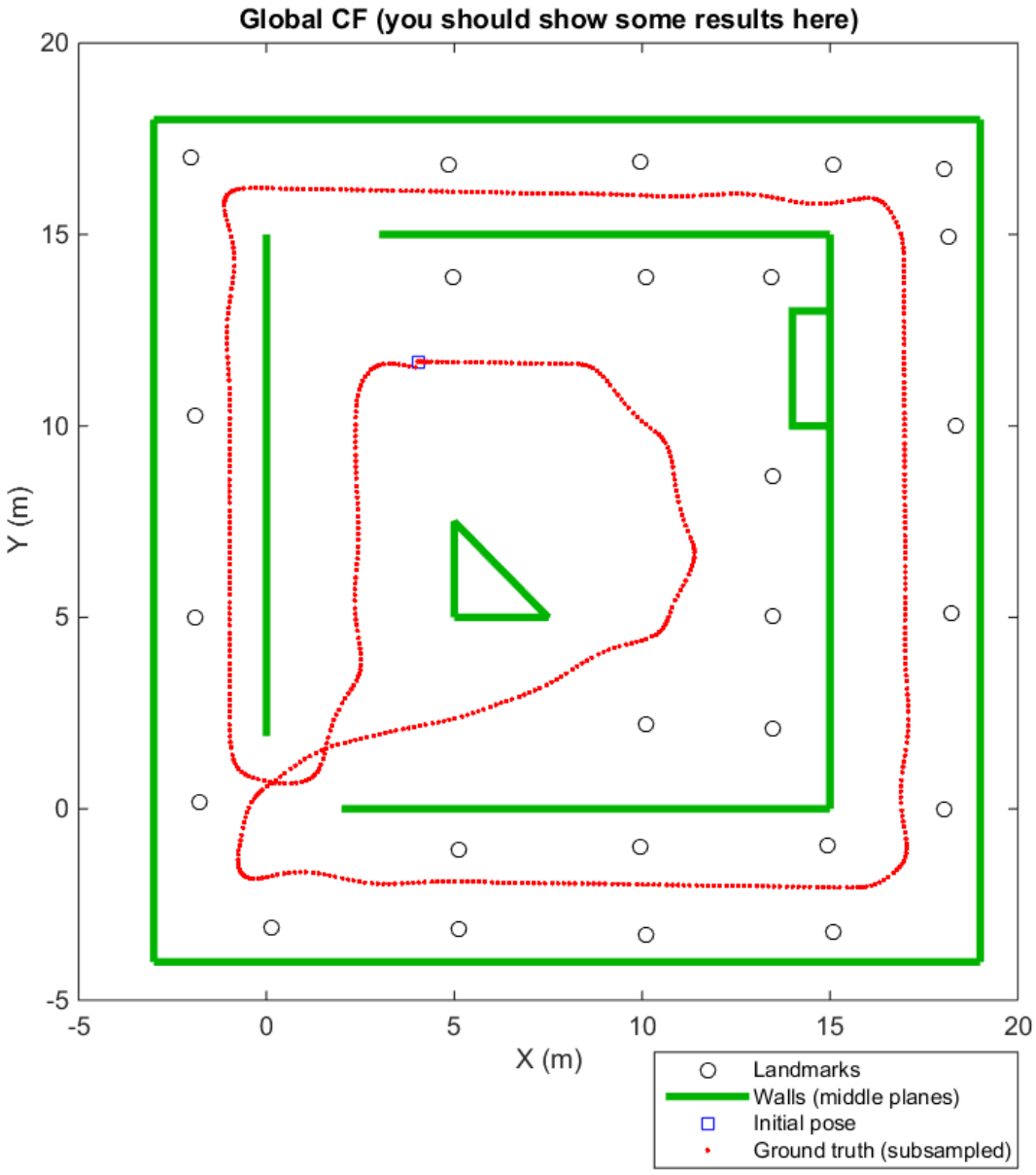
Validating the accuracy of your solutions part A and part B

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 we provide with the data (the provided example program shows how to read it from the data). Discrepancies of less than 2 centimeters are required.
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.
“Properly shown” means that are free from drifts and other artifacts as result of errors and inaccuracies in processing 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 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 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 a landmark (less than 5cm away). Of course, you can, alternatively, express the landmarks in the LiDAR CF, for performing the same comparison, but in that local CF.
The evaluator of your demonstration will consider those validations, to verify the accuracy of your solution.

Example program “ExampleUsingDataProject1_ABC.m”, 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, “ExampleUsingDataProject1_ABC.m”, from which you can see how to read that validation data.

Part C - Basic data association

Position estimation of the objects of interest in GCF

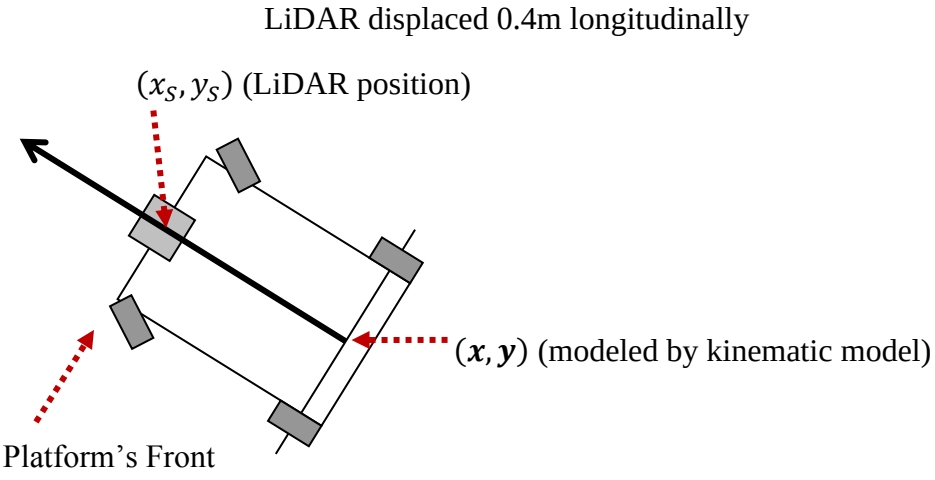
You are required to combine parts A and B. For each LiDAR scan that is available, you will estimate the position, in the global coordinate frame (GCF), of the detected OOIs. Figure 1 shows the configuration in which the LiDAR is installed on the platform. The sensor is perfectly aligned with the platform's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

Figure 1: LiDAR position



Part D and Part E- To be released on week 4.

==========

Note about datasets being used to test your program.

Some of the processing performed in Project 1, will be applied on simulated datasets, and other parts on real datasets.
Simulated data: we can generate a high number of different datasets(different infrastructure, deployment of landmarks, paths followed by the platform controller intruder dynamic objects, level of noise, frame rates of sensors, etc), so that you should expect that in your demonstration you will be asked to try something new, but sill similar in complexity. So, that tuning your solution for solving a "local minimum" may be a risk. We will upload some other datasets, so you can try your programs.

==========