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 playback session, 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, to dynamically show 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", during the playback session).
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”.
DA is necessary for other processing parts of the project. DA is demonstrated visually. Implementing some dynamic plots.

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. In this project, in this part you will need to infer two parameters. Details about this part are given in section

Part F requires 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.

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 the full project.

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)

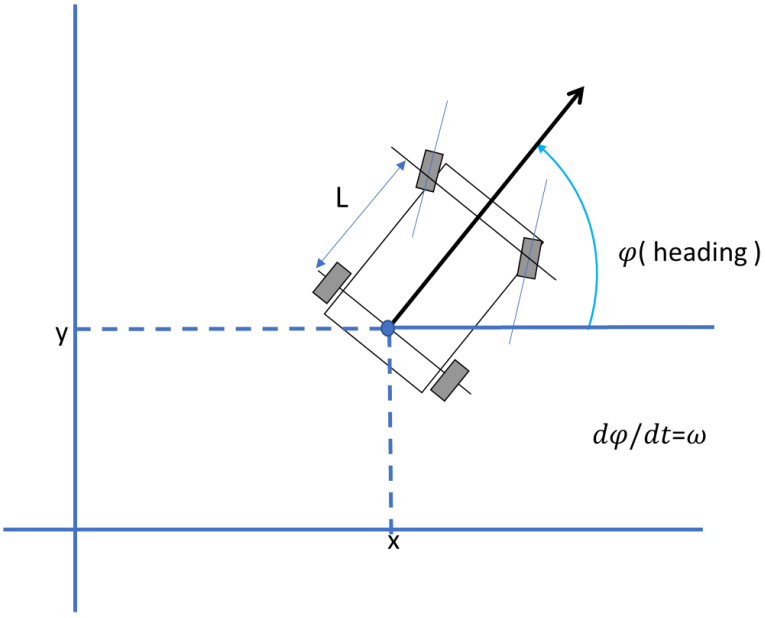
For instance, if you fail in answering all the questions, your Q factor will be 0, which means you would get 60% of the achieved marks of your submitted/demonstrated programs.

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 the measurements of the speed (longitudinal velocity) and the heading rate (from gyroscope).



The needed measurements 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.
For verifying your results, your program will record the estimated poses, in a buffer created for that purpose. For easier comparison with the ground truth, we suggest recording your estimates at the times of the LiDAR events, only.

Marking criteria for part A.

We will verify the accuracy of your solution for part A, when used with a "noise-free" dataset (such as "DataUsr_002.mat"). We will consider how your program performs during the first 30 meters of the trip.

The contributions to the marks of Part A (expressed as % of Part A), are the following:

Item 1 (40%). Accuracy, in terms of position, of your estimates, when compared to the ground truth. The discrepancy must be lower than 3 cm, always.

Item 2 (40%). Accuracy, in terms of heading, of your estimates, when compared to those of the ground truth. The discrepancy must be lower than 1 degree, always (differences must be normalized to the range of angles [-180,+180 degrees]).

Item 3 (20%). Visualization in Global CF. It is required to show the static infrastructure (walls and poles), and dynamically showing the platform's current position, being it refreshed at a rate similar to that of the LiDAR events.

For verifying items 1 and 2, you will plot the discrepancies between your estimates and the ground truth at the end of the playback session.

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 smaller than 20cm), and that has a "brilliant" surface (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/landmarks (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 are 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 for 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 it 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 with 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 unrelated to UNSW). As our problem is specific, your own solution 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.
Solutions that frequently take more than 10ms for processing this task will be considered inadequate, consequently losing 50% of the marks of this part.

Marking criteria for this part.

We will verify the accuracy of your solution for part B, in the same conditions as those of Part A.

Contributions to the marks of Part B (expressed as % of Part B)

Item 1 (15%). Visualization. Necessary for showing results. You will show raw scan points, in LiDAR's CF, using blue dots. In addition, you will show the positions of the detected OOIs, using a clear symbol (e.g., a red asterisk), easy to differentiate from the was scan points. In your plot, you need to show just the center point of each OOI (you are not required to indicate other properties of the detected OOIs).

Item 2 (85%). OOI detection efficacy. By visual inspection, during short parts of the playback session, a demonstrator will inspect your OOI detection, to verify if those OOIs you obtain are mostly consistent with the raw scans. The demonstrator will consider two rules. Rule 1: The positions of the generated OOIs must be close to those clusters of raw points that your common-sense dictates. Rule 2: Evident segments/clusters of points, in the raw scans, that seem to be OOI (from common sense), must be detected to be OOIs. Few and sporadic misses will be ignored by the demonstrator.

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

You are required to combine parts A and B. For the LiDAR scan#1 that is available at each LiDAR event, you will estimate the position, in the global coordinate frame (GCF), of the detected OOIs. Figure 1 shows the configuration in which the LiDAR#1 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 car's CF) are included in the provided data (read the example program, for details about reading these parameters.)

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

1) Certain static objects, such as landmarks and walls.
2) 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).

Marking criteria for this part.

We will verify the accuracy of your solution for part C, under the same conditions as those of Parts A and B.

Any OOI that does correspond to a landmark (you infer that fact visually) must appear close to that landmark (closer than 20cm), when shown in the GCF. (However, you are not required to perform Data Association in this part of the project.)

Contributions to the marks of Part C (expressed as % of Part C)

Item 1. Raw scan from LiDAR #1 properly shown in GCF (points that seems to be part of a wall mostly appear at about 10 cm of that wall's center) (30% of relevance)
Item 2. Raw scan from LiDAR #2 properly shown in GCF (points that seems to be part of a wall mostly appear at about 10 cm of that wall's center) (30% of relevance)
Item 3. OOIs appear close to landmarks. (40% of relevance)
Note: The scans from LiDAR 1 and 2 should be clearly visualized using different colors

Note: minor discrepancies and the existence of few outliers will not be considered to be an issue. You may ask the lecturer or the leader demonstrators if your program seems not to be satisfying these mentioned requirements.

Part D – Data Association

When those OOIs, detected in the local CF, are expressed in the GCF, a "data Association" process can be performed. Some of the detected OOIs will usually 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 GCF). The provided example program reads that map and plots the map's landmarks in a figure, to show how to read the map.
A basic approach for solving this DA problem, will be described by the lecturer, during lecture 4.
You are required to implement a basic DA module, and apply it to the OOIs detected in Part B. The tolerance of the association between OOIs and related landmarks will be a parameter, which will usually be set between 0.5 and 1.5 meters.
In addition to the DA implementation, certain visualization resources will be implemented, as part of the requirement in this section. In the figure associated to the global CF (the one also used in parts A and C), you will include some graphic items for showing the set of matched pairs, {OOI / Landmark}. The lecturer will show his approach in Lecture 4, you may like to use that way, or you may try other ways to clearly showing, dynamically, the DA results (e.g., using text labels).

The performance of you implementation will be evaluated in terms of apparent % of success. For that purpose, you (and us) will apply a visual verification, based on your visualization module. If your DA visualization module were not properly working, you would ask the lecturer for an API which provides that capability, for allowing you to demonstrate the performance of your DA solution.

Marking criteria for this part.

We will verify the accuracy of your solution for part D, under the same ideal conditions as those of previous parts.

Item 1 (35%). Visualization of results from DA. In the figure related to the GCF, in which you show OOIs and Landmarks, you will now show the set of associated couples {OOI/Landmark} (that result of your DA process), indicating each associated couple, so that users can appreciate those associations.

Item 2 (65%). Efficacy of the DA process. Your program will need to offer the capability to pause and play the simulation, at any time, so that the demonstrator can stop your program a number of times, for inspecting the results. Your DA must achieve 80% of success at each points.

1. The demonstrator can pause the simulation at any instance, especially at the time where it appears to fail the most DA.
2. The demonstrator will pause 3 times throughout the whole simulation
3. Each failure to associate evident couples of OOI/Landmark will result in a penalty of 5%

Item 1 will be valid if item 2 is able to achieve at least 50% of success.
If item 1 is not working, you can ask the lecturer for an API which offers that capability, so that you can still show the results of your DA. The API will be offered only to students who request it.

Part E – Basic Calibration

It is usual, in our systems, to have certain model parameters whose actual values may not be accurately known to us.
In our project, one of those unknow parameters is the bias that affects the gyroscope. The second parameter is the angular displacement of LiDAR#2 (i.e., that LiDAR is not well aligned with its nominal direction, in the car's CF.)
Although there are methods for implementing off-line and on-line calibration of parameters, those approaches are still to be discussed in subsequent lectures this term; consequently, in this project we will apply a manual approach.
You will estimate both parameters (which are unknown to you), by exploiting your visualization tools, or by other approach you may propose.
For the visual approach you will exploit the fortunate fact that the predicted trajectory is affected by the gyroscope's bias, and not by the LiDAR parameters. Similarly, you may find a way to find the effect of the LiDAR angular shift, independently of the gyroscope's bias.
For solving this tuning problem you may exploit any data offered by the dataset (measurements, ground truth, infrastructure description, initial pose, etc.)

You will need to describe your approach in a brief section of the report to be submitted with this project. Brief: no more than one page). A dataset which includes data affected by gyroscope bias, and with orientation misalignment of LiDAR1, will be released in week 4,

Marking criteria for this part.

Item 1 (50%) You will show to the demonstrator you approach/procedure for estimating the parameters. procedure. You will use a data file provided by the demonstrator, specifically for you.
Item 2 (50%). Report. In a section of the report, you will explain your approach for estimating the required parameters. We expect a brief section, of no more than 3 pages.
In the report, we are looking for your thoughts on the inaccuracy of the model parameters, how it will affect the result (if it those correct parameters are ignored), and your description of the procedure you applied to solve the issue.
We will apply gptzero to analyze reports presented in this project. If a report is found generated by AI tools, the report will be considered invalid, and the case may likely be reported to the School Ethics Officer
(https://www.student.unsw.edu.au/notices/2023/02/academic-integrity-reminder-chatgpt)

Part F – Deterministic localization

By exploiting the fact that certain OOIs (whose positions in the car's CF are known), do correspond to known landmarks (i.e., landmarks whose positions, in the GCF are known), estimates of the car's pose (in the GCF) can be obtained. Solving this problem implicitly means solving a set of equations, in practical terms (e.g., minimizing some "error function").
In this part of the project, you are required to implement that estimation approach. You will apply it at each LiDAR event, by exploiting the successfully detected and associated OOIs. Your solution will provide the following capabilities.
1] At each LiDAR event, when more than 2 OOIs/Landmarks are available, your module will provide estimates of the car's pose.
2] Your program will record, in a dedicated buffer, the estimated (by this approach) car's pose at that LiDAR event. If no solution was achieved (e.g., due to insufficient OOIs), your estimation would report the flag value, [0;0;0];
3] At the end of the playback session, you will plot the positions of the estimated poses. You will also plot, at the end of the playback session, in separate plots, the discrepancies between your estimates and the ground truth.
In this part of the project, you are required to simply produce the estimated pose, as result of knowing how a set of points do simultaneously appear in two coordinate frames.

Note about accuracy matters: Do not expect a great accuracy of the results, even when using "noise-free" data. This is because the LiDAR measurements are still affected by quantization errors in the ranges measurements; and are also affected by the limited angular resolution. Consequently, you may appreciate discrepancies (between your estimates and the ground truth) of up to 10cm in the positions, and up to 1 degree in the headings. However, these estimates are well valuable, because the errors are usually bounded.
In respect to your solution, discrepancies in position of up to 20cm will be considered acceptable.

In addition to the accuracy of the results, we will pay attention to the processing cost of your implementation (visualization is excluded of this cost, as we are exclusively talking about the cost of evaluating the estimates). Your solution must be achieved in less than 8 milliseconds (in the lab computers). Please, read about the definition of "processing time", in the appendix.

Marking criteria for this part.

Item 1 (05%) Plot, in separate figure, the discrepancies between estimates and ground truth, for all LiDAR events, in terms of position and in term of heading.
Item 2 (25%). Based on the plots presented in item 1, discrepancies in position must be lower than 20cm.
Item 3 (25%). Based on the plots presented in item 1, discrepancies in heading must be lower than 3 degrees.
Item 4 (20%). The processing time of you estimator must be lower than 8ms.
Item 5 (20%). Brief report (up to 3 pages), describing your approach (The approach must be consistent with your submitted solution).

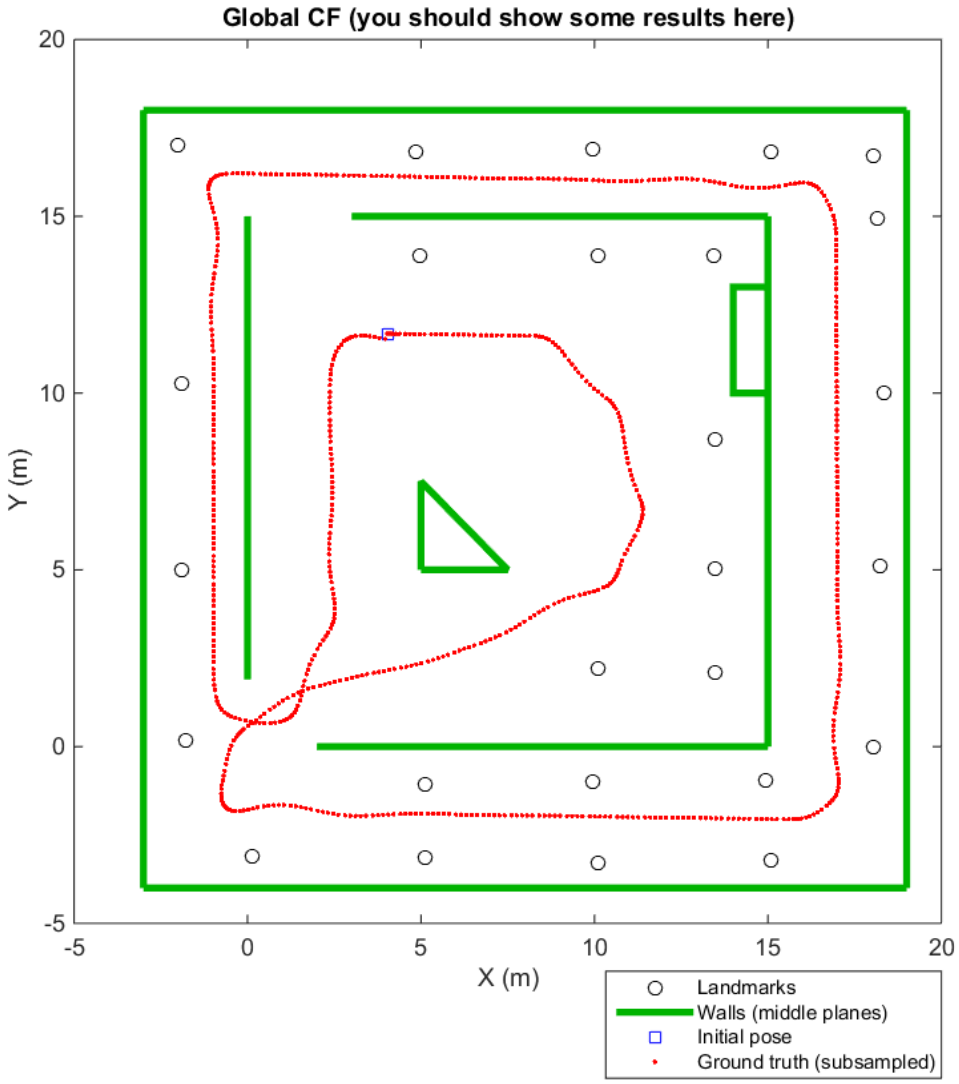
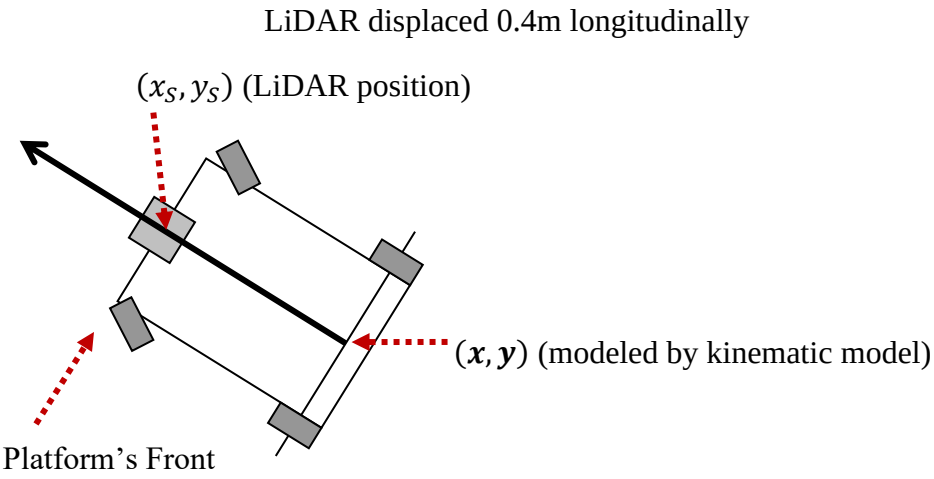


Figure 1, in which static infrastructure (walls and poles) are shown. The map's landmarks are shown by black circles, and the middle planes of the walls, are shown by green lines. All those data components are useful for validating your solutions.

The dataset also contains a subsampled ground truth (of the platform's pose), which is shown in the figure, plotted in red dots.

This figure was generated by the provided example program (but using a different dataset). From the same example program you can understand how to read that validation data.

Figure 2: LiDAR position



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

Please, watch 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.

Document “**MeasuringProcessingTimeInMatlabMTRN410.pdf**” describes an approach for measuring the processing time spent by your modules, to verify if those satisfy the requirements related to processing times. Alternatively, the profiler tool (provided by Matlab) may also be used to verify those processing times.

(End of Document)