MTRN4010 – Project 2.

Deterministic processing of speed, gyro and LiDAR measurements

Implementing Dead-reckoning

Implementing feature extraction

Performing Data Association

In this project we commence using (and processing) data from multiple sensors, simultaneously. We do it in a simple way: deterministically. These are preliminary steps, which are useful for solving, later, the localization problem through stochastic Sensor Data Fusion (which will be applied in a subsequent project).
This project involves processing multiple sensing capabilities: LIDAR, wheel encoder and gyroscope (yaw rate, provided by an IMU).
All the modules which are implemented in this task will be used for solving parts of a subsequent project. Properly solving this project does not only give you a good mark in this assignment, but it also facilitates the solution of that subsequent project.
One of the purposes of this project is for understanding and implementing localization by applying "dead-reckoning". The second purpose is getting used to using different coordinate frames, and for implementing a simple feature extraction for detecting objects of interest (OOIs) in the area of operation.

Part A

You are required to implement a "dead-reckoning" process, based on the kinematic model of the platform, and on the measurements provided by certain onboard sensors (speed encoder and gyroscope). The kinematic model is explained in the lecture document "**[AAS_2021]_KinematicModels.pdf**". The way for implementing it, in a discrete time fashion, is explained in that document as well.
The inputs of the process model are the angular rate ω_z and the speed encoder measurements. The necessary data is contained in the Matlab data file "**Measurements_AAS01.mat**" Example code, showing how to use the data, is provided in the M-files "**ShowData2021_Example2.m**" and "**ShowData2021.m**". Read those small programs to understand how to use the data, and how to use some basic Matlab programming. The example program **ShowData2021_Example2.m** is recommended for being modified by the students, for implementing their solutions.

Validation: You will compare (by simply inspection) your estimated path with a solution provided by the lecturer. Additional validation of these results will be done in subsequent parts of this project, when LiDAR data is processed as well.

Part B

This part of project 2 involves processing LIDAR measurements. This processing is necessary for modules which will be implemented in a subsequent project. Those modules need certain features extracted from the LiDAR

scans. In this part of the project you will implement a function which will perform the required processing. The LIDAR used for producing the data is a SICK LMS291 (LMS200 family). The provided example programs do also show how to read and use the LiDAR data.
You are required to implement a module, for processing individual scans, providing the following capabilities:
a) As the native raw scans are in "polar" in the LiDAR's coordinate frame, you are required to show those in a cartesian representation.
b) Show the "brilliant" pixels (those pixels that correspond to highly reflective surfaces), in a different color to that of the rest of the pixels (the "opaque" ones). One of the example programs shows the way to use the intensity information of the LiDAR scans.

The following picture shows some of the poles (which we used as landmarks), which had been deployed in the lab for the purposes of localization. These poles are covered by highly reflective tape which the LiDAR can discriminate from other surfaces that are less reflective ("opaque")

Figure 1: Some poles ("landmarks"), and the UGV, operating in the MTRN lab.

Part C

Based on the set of brilliant pixels, which you extract using your implementation in part B, you are required to detect the poles. As a pole can produce more than one pixel (depending on the distance between that pole and the LiDAR), you are required to implement a segmentation process, whose output will be the centers of geometry of clusters of brilliant pixels. Each of those clusters is the image of a pole (we also call those objects, "Objects of Interest" or OOIs). You can exploit the fact that the OOIs are relatively isolated; each OOI is separated from other OOIs and from opaque objects, by no less than 1 meter (you can appreciate that fact when you inspect the raw scans).
You are free the choose and implement the segmentation approach that you prefer.
To verify consistency of the results, you will show those detected OOIs (their estimated centers of geometry), in a figure, jointly with all the original pixels (you may reuse the visualization tools you implemented in part B).

Part D

In this part you are required to show the LiDAR scans and the detected OOIs, in the global coordinate frame. For achieving that, you will use the estimates of the platform's pose (position and heading) which you obtain applying your solution for part A. Based on those estimates you will apply a coordinate transformation, for showing, in the global coordinate frame, the LiDAR scans and the detected OOIs. Those have been obtained in parts B and C (but were represented locally, in the LiDAR's coordinate frame). The position of the LiDAR on the platform is shown in Figure 2 (this is relevant for a proper transformation from local to global coordinate frame).

Additional specifications

1) It is required that the processing time, per LiDAR scan, must be lower than 50ms, in a normal computer (such those in the MTRN lab). This required processing time is well feasible, even in plain Matlab programing language.

If you realize that your implementation is not fast enough, you are not required to process all the LiDAR scans, i.e. you can process one scan every 2 or 3 available scans.

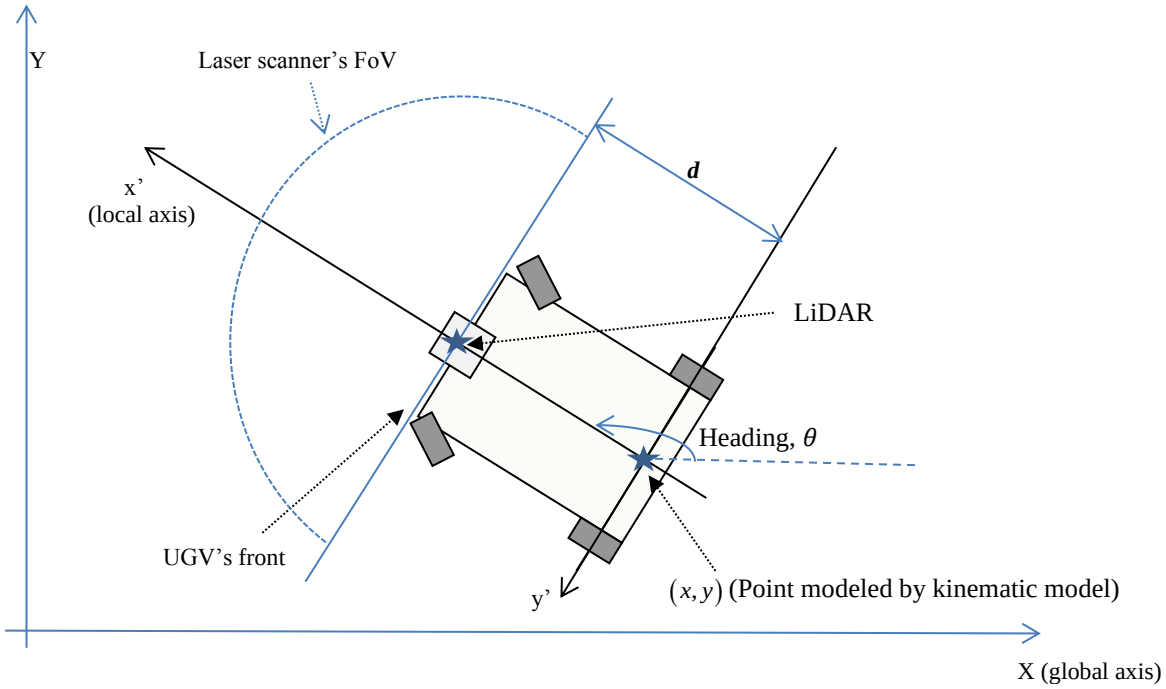

Figure 2: The position of the LiDAR is displaced 46cm, ahead the modelled platform's position. The LiDAR is aligned with the platform, pointing ahead the platform.

Validating parts A, B, C and D.

If your estimates of the UGV's pose are correct, and your transformation from local to global coordinate frame is correct, then you will see that the LiDAR scans describe the area of operation of the UGV (e.g. certain walls), and those seem to be static (except some fluctuations due the LiDAR's noise, and additionally due to a minor shift, related to the error introduced by the dead-reckoning process based on the kinematic model).

You can also compare the detected OOIs with the position of the map's landmarks which are provided in the map file. Each of those OOIs should always appear near a landmark, except some drift which may increase with the time (see the lecturer's solution, in class)

Part E

Based on the results achieved in parts A, B, C, D, we are now in condition to solve to the Data Association part of the project. This component (jointly with those of A, B, C, D) will be instrumental for solving project 3, in which we implement a localizer, based on the EKF estimator.

A map of landmarks is provided by the lecturer. Those landmarks were surveyed, and their positions recorded in a table (file "MapMTRN4010.mat").
In your data association process, you will infer the identity of the detected OOIs, each time you process a LiDAR scan. A way to do it is by expressing in the global coordinate frame the positions of the detected OOIs, and then matching those OOIs and map's landmarks, based on the distance between them.
You need to visually indicate the matching of those OOI successfully associated to map's landmarks. You are free to visually show that information in the way you prefer. It is a good idea to show it in the figure which shows the OOIs in the global coordinate frame (the one which solved part D). The lecturer will show, in class, his way of showing results; however, that is just an example about how to present your results.

Deadline: submission of project 2 is: Friday Week 6, 10PM. Your demonstration of project 2 will be on Week 7.

The relevance of the different project's components is described in the following paragraph. Adding all the marks gives an overall result of 100 marks. That overall project's mark is then scaled to the marks (14 marks) this project contributes to the course's final mark.

Implementation of Part A: 16 marks.
Implementation of Part B: 03 marks
Implementation of Part C: 16 marks
Implementation of Part D: 10 marks
Implementation of Part E: 15 marks (this part needs to be working, jointly with the previous parts)
Each of the mentioned marks corresponds to that implementation working satisfactorily.

Report: **R** marks: (Up to 20 marks)
Quiz: **Q** marks: (Up to 20 marks)

The quiz is internally marked in scale [0:100], and its contribution to the project's mark, **Q**, can be of up to 20 marks. **Q** is based on the following equation: **Q = [marks achieved in parts A,B,C,D,E]/60 *20*q/100**; in which **q** is the mark in the quiz, in scale [0:100].

The report can contribute with up to 20 marks. The report will include a brief description about the project's parts for which you have submitted an implementation. The report will then contribute to each solved part, by a number of marks which is proportional to the achieved mark in that part's implementation. You do not need to include sections about parts whose implementation has not been submitted. The report should be brief, not exceeding six pages of content (e.g. about one page dedicated to each solved part.). The report will be submitted, electronically, in a PDF file. Its submission will be done jointly with the programs (same deadline).

Quiz: A basic and brief online questionnaire (associated to this project) will take place during week 7. The score in the quiz has a defined relevance in the final mark of the project, as indicated by the variable **Q**, in the formula of the overall mark of project 2.
The quiz will focus on basic matters, which you should know if you solved (or seriously tried to solve) the project.

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