MTRN4010 – Project #01

Feature Extraction from LIDAR Data

This project involves preprocessing LIDAR measurements, in an OFF-LINE fashion (for simplifying the complexity of this project). This preprocessing is necessary for modules (to be implemented in subsequent projects) which need the detected features in order to operate.
In Part A, you will implement a function, which will actually perform the required processing for a given scan of the LIDAR. In the second part of this project, part B, you will use the function implemented in Part A, for processing a sequence of scans, and for showing the results.
The LIDAR used for producing the data is a SICK LMS291 (LMS200 family). Details, about how the sensor was installed on the platform, will be given during the explanation in class; however, those details are not really needed for solving this project.

Part A.

Implement a function for processing individual scans (generated by a laser scanner, aka LIDAR); to detect and classify objects of interest (OOI), from the raw measurements provided by the sensor. We consider as OOI any object that has a defined size: apparent diameter between 5cm and 20cm (e.g. a pole).

Input argument: The input data is assumed to be a vector, of size 361 x 1, class uint16 (16 bits unsigned int.) The vector's content is the raw data, associated to one laser scan; which is composed by a sequence of 361 "pixels". The function initially needs to extract, from the data, the range and intensity of each of the pixels, and properly scale the ranges. The raw data correspond to a POLAR representation, where the 361 ranges are associated to 361 angles, from 0^o to 180 degrees, having a separation of 0.5^o.
After parsing and scaling, the data is processed for inferring clusters (aka: segments or objects). The segments that seem to have certain size (approximate) are selected and saved in a list of OOIs. The list includes the properties of each detected OOI, those properties are: position, size and "color". The function must return an instance of a structure, designed to contain those fields, as follows:

a) **Center of geometry**.
b) **Size** (approximate diameter).
c) "**Color**". Two possible values: Brilliant (=1) or Opaque (=0). We consider a pixel to be highly reflective (HR) if its associated intensity field is higher than zero. An OOI is said to be *brilliant* if at least one of its constituent pixels is HR; and it is considered to be *opaque* if none of its pixels is HR.

For instance, the following fields are a good implementation of the required result:

N : Scalar which represents the number of detected OOIs (in the currently processed scan).
Colors : Vector (of size 1 x N) having the "colors" of the N detected OOIs (see note 1).
Centers : Matrix (of size 2 x N) for storing the X,Y coordinates of the OOIs' centers.
Diameters: Vector (of size 1 x N) for storing the sizes of the OOIs.

Consequently, object #i (provided that $1 \leq i \leq N$), would have its position stored in OOI.Centers(:, i), its diameter in OOI.Diameters(i), and its intensity in OOI.Colors(i).

Part B

Implement a program for using your previous function (from part A), in a periodic fashion; for processing a sequence of images. The images are contained in a Matlab's data file (MAT file). The file contains real data (acquired from a laser scanner, according to a specified format). See the example programs "ExampleProcessLaserData.m" and "ExampleUseLaserData.m" for details about reading those data files. Your program will be implemented as a Matlab function, whose argument is assumed to be a string, for specifying the name of the data file.
Note: you must read the examples; paying attention to the comments. If you ask for help, do it after having invested proper effort, reading the examples.

Output of the program: The program will process the sequence of laser scans (which are contained in the input data file) by applying the function developed in part A. It will process each individual laser scan separately, one by one. For each individual scan it will produce a list of OOIs and their associated characteristics.

The result, for each processed laser scan, will be shown, dynamically, in a figure. The plots will include the following features:

a) Raw points associated to the laser scan. These points will be shown by blue dots.
b) High intensity points will be shown as red dots or similar symbol.
c) The centers of the brilliant OOIs will be shown as green stars (opaque OOIs do not need to be shown).

(All the plots must be presented in a Cartesian coordinate frame; not in the native polar representation).

The program will contain a main loop (based on 'while' or 'for' loop) for sequentially processing all the laser scans contained in the input data file. After processing an individual laser scan, the program will wait for an amount of time (e.g. for approximately processing the scans at a rate = 10Hz), before processing the next laser scan. The refreshing of the figures will be done efficiently, by using handles of the graphical objects (as shown in one of the provided examples.)

Deadline: Demonstration of this project will be on Week 4. In the demonstration, you will briefly show the evaluators that your program is working. You may need to explain parts of your program, and answer questions, as part of the demonstration.

Quiz: A brief questionnaire may take place during the first minutes of your session, the day of your demonstration. The score in the quiz has a defined relevance in the final mark of the project. Students showing serious lack of knowledge, about the project being demonstrated, will be interviewed by the lecturer.

Report: There is not report associated to this project.

Note: Although you are not required to submit a report, this project involves implementing resources which you will use in subsequent projects; for that reason, it is a good investment dedicating time in order to properly complete this project.
In addition, try to make the program efficient, because it will be used in subsequent projects, whose programs will run in an on-line fashion. The processing time of this task (NOT including plotting) should be less than 50ms per laser scan, in the computers of the laboratory (A well implemented approach, in Matlab, would take less than 1ms to process 1 scan).

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