**Lane Separator detection based on LIDAR data**

Name: Lengyel Péter Mihály

ID: -

Date: 2020.08.07

#### Table of contents

[1. Introduction 2](#_Toc45221542)

[2. Development Stratagy Creation 3](#_Toc45221543)

[3. Performed Tasks 3](#_Toc45221544)

[3.1 Visual Studio Project Setup 3](#_Toc45221545)

[3.2 Creation of Visualization Tool 3](#_Toc45221546)

[3.3 Integration of Auxiliary Libraries 3](#_Toc45221547)

[4.Sources 3](#_Toc45221548)

# Introduction

Initially very little was knows about the data meant to be used for Lane Separator Detection (this feature will be a hit). The assumption was that the data is from a moving vehicle, and the LIDAR used was a roof mounted 360 degrees rotational LIDAR e.g. Velodyne’s HDL-64E.

The concern with this assumption was that the data had no current vehicle speed associated with the timestamps (based on data provided by the CAN). This would mean the solution would require the implementation of a SLAM algorithm to estimate EGO velocity.

Further observations:

* It was observed that the **adjustedtime** and **timestamp** columns were exactly the same.
* It was observed that there are 64 laser IDs
* It was observed that each timestamp is associated with 128 data points, which means at each timestamp, each of the 64 lasers recorded 2 measurements (This is assumed to be due to double return).
* It was observed that the first 64 and second 64 points of the 128 data points belonging to the same timestamp are extremely similar, with only a minimal difference (the azimuth is the same).
* It was observed that the difference between timestamps (frame rate) is 173ms (?).
* It was observed that the azimuth increases seemingly randomly, alternating between a step of 18 and 36 each 128 timestamps.
* It was observed that the 64 laser IDs are not 64 in value, since the data for some if the IDs are missing (the exact number varies). The reason for this is unknown (Assumed that no return is detected).

To better understand the data, two python tools were created to visualize the data points. The first (cloudalization\_dynamic.py) was created visualize in an animated form, so that it would be possible to compensate for the motion of the EGO.

TODO: Insert picture of visualized data

During usage of this tool, it was realized that this was unnecessary, since each .csv contained the data of a single revolution of the LIDAR, and this was recorded or when the vehicle was stationary, or when the speed was so low that no noticeable point skew was created.

Observations Based on the Visualization:

* The implementation of a SLAM algorithm is not required, in fact, impossible with this amount of data. (No motion can be derived from the data, and the data has its origin at the position of the sensor)
* Only one out of the five data sets has a lane separator (4.csv). This means the algorithm should return with no lane separator for data sets 1, 2, 3 and 5.
* The land separator seen in dataset 4 seems to be a solid concrete lane separator typically seen on highways. This gives better chance for recognition, as opposed to a temporary lane separator composed of posts.
* The direction of travel is orientated along y axis, and opposed to the x axis which is the typically used typically used convention. This must be taken into consideration by the final algorithm.

Following this initial analysis of the available data, the development is moved on the development strategy creation, where the necessary tasks are collected.

# Development Stratagy Creation

|  |  |
| --- | --- |
| Action | Reason |
| Reading the .csv | The .csv file has to be read and stored in memory. This may require and external library. |
| Preprocessing the data | Some data points are missing. It should be checked if these can be compensated with data in the 2nd measurement (part 2 of the 128 timestamps) |
| Preprocessing the data | Data points which are out of the scope of a lane separator could be filtered. Must take into note that on a 3 lane highway, the separator might still need to be recognized. Also in left hand driving countries, the separator will be on the right side of the EGO. |
| Vehicle parameters. | No vehicle parameters are given. The location of LIDAR sensor compared to the vehicle is unknown. Knowing this may assist in the algorithm, so this may have to be guessed or calculated by a script based on the given data. The can be done by calculating some average of the z coordinates of points closest to the vehicle. |
| Config file IO | Might be needed to configure the algorithm from an external file which has data such as: Expected road width, Sensor position, Sensor model and its parameters. |
| Algorithm implementation | Current best concept is the usage of some method of least squares to fit a plane to the general region a lane separator is expected. |
| Addition of UT library | A unit testing framework needs to be integrated into the project. |
| Addition of visualization library | A cpp library capable of visualizing the results. |

# Performed Tasks

## Visual Studio Project Setup

|  |  |
| --- | --- |
| GIT project setup |  |
| VS file struct setup |  |
|  |  |
|  |  |

## Creation of Visualization Tool

|  |  |
| --- | --- |
| Visualization tool |  |

## Adding Premake

Premake was added as a project management utility to assist the building of subdirectories such as GoogleTest and possible GLFW.

## Software Development

|  |  |
| --- | --- |
| Impl. .csv IO | Reading in of the .csv |
| Impl config file | Enumerations.h contains the user set variables for the filtering of the data. |
| Impl. filtering 1. | Filtering based on User set parameter for e.g. point distance, laser\_id etc. |
| Impl. filtering 2. | Filtering of the 2nd, 3rd, etc laster returns. Original data only has 2 returns, current config filters 2nd returns to reduce data. |

The software shall have the following structure.

IOHandler

PreProcessor

Scan

Algo

Visualizer

Enums.h

# Sources

**There are no sources in the current document.**