Hochschule Ravensburg-Weingarten

Lidar and Radar Systems: Project Report 1

LiDAR Data and Bounding Boxes

Created by:

Sahil Arora
(Matriculation Number: 32953)

December 10, 2020

Contents

1 Introduction

When we think of an autonomous application such as a self-driving car, a plethora of sensors come to mind that can be employed to do the job. Many of us may have seen prototypes of autonomous Taxis by Waymo with a large black colored protruding LiDAR sensor on top of it [Way]. LiDAR sensor emits laser beams which when strikes an obstacle, reflect back to sensor. The time elapsed can be used to measure the obstacles' distance [Wika]. A lot of companies are investing resources in training and testing, thereby publishing a large amount of datasets openly available for the researchers [Cae+20]. One of these is Astyx HiRes 2019 dataset which is what we use in this project[MK19]. The scope of this project is to provide an estimate of LiDAR measurements (using a known sensor, Velodyne Puck [Lid]) on a hypothetical observed vehicle and then try to validate our approach by finding a scene with similar configuration in Astyx Dataset and comparing the measurements. For simplicity we divide the project into 2 milestones. Section 2 deals with estimation of number of LiDAR measurements based on car dimensions, distance from observation and angular orientation. Section 3 deals with extraction of measurement results with a similar scenario found in real data. Finally, in Section 4 we try to understand the difference in expected and actual results.

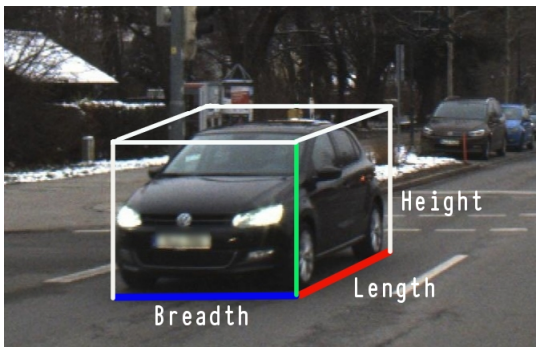
The project requires a few tools and libraries which need to be set up. The second milestone is performed on Ubuntu 18.4 platform running on a Intel i7 personal computer. Source code is written in Jupyter Notebook with support libraries like NumPy and MatPlotLib [Jup] [Num] [M D].

2 Estimate the observable LiDAR points

This section starts with the definition of bounding box of observed vehicle (from here on termed as ‚vehicle‘) dimensions. Further, the specifications of Velodyne Puck sensor are defined for understanding [Lid]. These specifications are then used to determine which specific laser beam(s) from a vertical array would strike the vehicle. Next, we consider the vehicle's breadth and estimate the number of horizontal beams striking the vehicle. Combining both these, we estimate total possible LiDAR measurements on vehicle.

2.1 From Car to Box.

Since cars may have an arbitrary shape designed by manufacturer, it makes sense to general the object with a bounding box [Wikb]. This way we can more efficiently define the area through which we have to search for measurements. To define a bounding box we need the dimensions of the car: namely the length (l), breadth (b) and height (h). Image 2.1A is partly taken from Astyx dataset (which can be downloaded with the given link [Gmb]) but axes and bounding are drawn separately in Matplotlib for better visualization.



(a) Fig 2.1A Car Bounding Box [MK19]

Velodyne Puck Specifications	
Total Channels (n)	16
Vertical FoV/ Resolution (Vr)	30° / 2°
Horizontal FoV/ Resolution (Hr)	360° / 0.1° - 0.4°
Range (r)	100 m

(b) Table 2.2 Velodyne Puck Specs [Lid]

2.2 Specs: Velodyne Puck

Puck is a surround LiDAR sensor that has a horizontal scope of 360 degrees. It emits a 16 channel laser beam that covers a vertical span of 30 degrees. This sensor has a maximum range of 100 mtr, but in our scope we limit our estimations to a vehicle at a distance of less than or equal to 20 mtr. The specification are compiled in Table 2.2 [Lid]. Fig 2.3 and Fig 2.4 respectively show the side and top view of sensor to explain the distribution of laser beams graphically.

2.3 Counting strikes in Vertical Scope ?

We consider that the LiDAR sensor is directly mounted at the top of ego-vehicle (from here on termed as ‚ego‘). The mounting height of sensor is h_m as seen in Fig 2.3. Since in a vertical field of view the beams would disperse more as we go farther from the source. Therefore, the number of beams that would actually strike the vehicle are depending on the distance between ego and vehicle. Let us denote this distance as d.

Let us denote each beam of Puck as i with $i \in [0,15]$. Where 0^{th} beam is the one having maximum positive angle w.r.t. horizontal axis. We know that this angle is 15° based on sensor specifications. We can use this information of generalise the angle of any given beam number. Note the positive angles are depicted clockwise and negative angles as anti-clockwise.

$$\alpha = 15 - 2i$$

The height of any i^{th} beam at defined distance will be dependent on angle. By trigonometric identity in a right angle triangle, we can compute the height as follows:

$$\tan\alpha = \frac{h}{d} \Longrightarrow h = d\tan\alpha \tag{1}$$

In fig 2.3A, we can see two triangles that can be used to compute the beam count. We take the upper triangle ABC. The height (segment BC) of $\Delta ABC = h - h_m$. As seen from eq(1), we can say that if beam number **f** is the first beam hitting the vehicle in front, then:

$$\tan(15 - 2f) \leq \frac{h - h_m}{d} \tag{2}$$

Placement of a less than equal to sign is due to the fact that beam may strike the top most point of vehicle or someplace below. Solving the above equation, we can estimate that the

first beam that would strike is:

$$(15 - 2f) \leq \arctan \frac{h - h_m}{d} \tag{3}$$

$$f \geq \frac{1}{2}(15 - \arctan \frac{h - h_m}{d}) \tag{4}$$

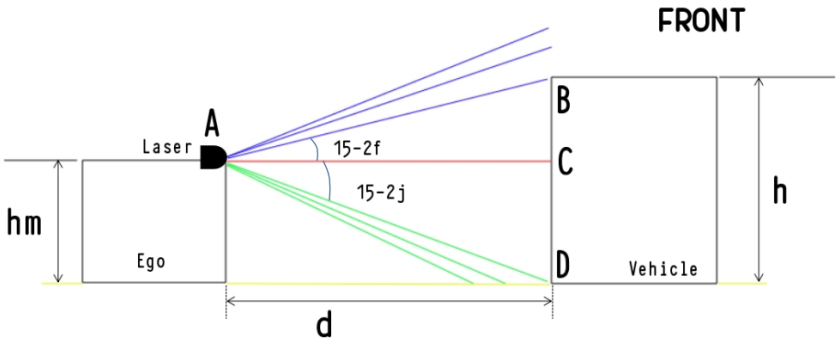


Fig 2.3A Side view. Visualize Vertical Scope.

Following the same approach, we can write for the triangle ΔACD. Take care that segment CD $= h_m$ is in negative co-ordinate system. The last beam striking **j** could hit the lowermost part of vehicle or somewhere above. Thus, height computed by triangle may be greater (because of -ve values) in value or equal to actual height. Hence a $\geq$ sign is put in eq (5). :

$$(15 - 2j) \geq \arctan \frac{-h_m}{d} \tag{5}$$

$$j \leq \frac{1}{2}(15 - \arctan \frac{-h_m}{d}) \tag{6}$$

So, with equation (4) (6) we have the first and last beam striking the car. We can now compute the total number of beams hitting the vehicle.

$$count = j - f$$

2.4 Counting strikes in Horizontal Scope ?

The type of sensor used in this project emits 16 channel rotating laser beam (refer Table 2.2). This means that after a certain time interval the beam rotates by a small angle as seen from top view. This small angle is known as Horizontal Angular Resolution. Puck come with a configurable range of resolution and for this project, alike in Astyx 2019 dataset we choose it to be 0.2 degrees [MK19].

It is very common in driving scenarios that vehicle is not exactly aligned with the ego and thus the area of car visible is much more than in normal condition. Therefore, a generalized approach must be taken. Refer to figure 2.4 that denotes a vehicle inclined at an angle θ w.r.t. ego. Using trigonometry and available dimensions of the vehicle, the frontal area can be estimated as a function of :

$$f(b, l, \theta) = b \cos\theta + l \sin\theta$$

Just as we saw in vertical scope, the cluster of horizontal beams will tend to disperse more with increasing distance between vehicle and ego. A general formula can be written to compute the unit exposure length (exposure per beam, eb_u) of horizontal beams for a given distance d (refer fig 2.4):

$$eb_u = d \sin(Hr) = d \sin 0.2^\circ$$

To compute the total number of beam clusters striking the vehicle $(beams)_t$, frontal area $f(b, l, \theta)$ can be divided by unit exposure length (eb_u):

$$(beams)_t = \frac{b \cos\theta + l \sin\theta}{d \sin 0.2^\circ}$$

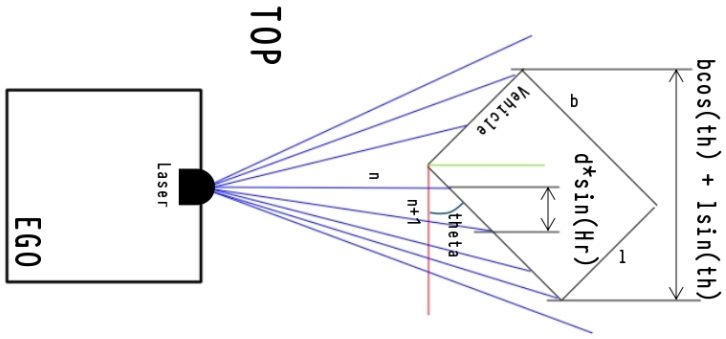


Fig 2.4 Top view. Visualize Horizontal Scope.

2.5 Let's work some real numbers ?

Based on the findings in above two sections, we can estimate the number of lidar estimations as:

$$end = (beams)_t * count$$

To be able to validate our hypothesis, let's take some value of distance(d) and rotation angle (θ). Below are the values chosen, distance in meters and angle in degrees :

$$d = [5, 10, 15, 20]$$

$$\theta = [0, 45, 90]$$

The calculations are programmed and the results are tabulated in Table 2.5.

In the above shown calculation, few points are to be noted:

1. h_m is taken for Hyundai i40 vehicle. A similar ego was used in Astyx dataset [Ast].
2. We assume horizontal resolution same as used in Astyx dataset for better comparison [MK19].
3. We have assumed the vehicle dimensions to be similar as Volkswagen Golf, an easily spotable car in Germany.

hm, mounting height of LiDAR	1.47	Dimensions	L	4.2
Hr	0.2		b	1.8
			h	1.4

			Distance (mtr)			
			5	**10**	**15**	**20**
Angle (radians)	**0**	f	8	8	8	8
		j	15	11	10	9
		count	7	3	2	1
		f(b,L,theta)	1.800			
		eb_u	0.01745	0.03491	0.05236	0.06981
		$(beams)_t$	103	52	34	26
		end	722	155	69	26
	0.785	f	8	8	8	8
		j	15	11	10	9
		count	7	3	2	1
		f(b,L,theta)	4.243			
		eb_u	0.01745	0.03491	0.05236	0.06981
		$(beams)_t$	243	122	81	61
		end	1702	365	162	61
	1.571	f	8	8	8	8
		j	15	11	10	9
		count	7	3	2	1
		f(b,L,theta)	4.200			
		eb_u	0.01745	0.03491	0.05236	0.06981
		$(beams)_t$	241	120	80	60
		end	1684	361	160	60

Table 2.5 Measurement Examples.

3 Extract data from Astyx Dataset

3.1 How is the data stored ?

The data is majorly divided into 5 different directories that can be accessed with help of python *os* library. The dataset contains Ground Truth, Camera Views, Lidar and Radar recordings and their calibration data [MK19]. There are a total of 546 scenes available. Python library *matplotlib* is be used to visualize the camera images.

For our comparison, we store the identities of scenes and objects having a distance of objects in range $\in$ [5,10,15,20]. A deviation within 1 mtr for first three and 2 mtr for last element of range list is allowed. We identify the scenes by running a loop, checking one scene at a time and storing the scene number and Object ID that corresponds to specified categories.

3.2 Transformation of LiDAR points.

The LiDAR point set is in a different co-ordinate system than RADAR point set. To visualize everything in a base frame, we required transformation matrices to convert the LiDAR point set. These are given in the calibration directory. We define a python class to open the corresponding file and read the LiDAR-to-reference matrix. Then using the homogeneous coordinate transformation $(x, y, z) \rightarrow (x, y, z, 1)$, we multiply the new co-ordinates with the LiDAR-to-reference matrix and get the transformed points [BR20].

3.3 3D to 2D conversion and Bounding Box.

3D is relatively easier for a human being to understand and interpret. For computers, it may not not be an easy job. So it is a good idea to convert 3D point cloud in 2D data which can be plotted on an image. We convert it to the so called FoV (Field of View). To do this, we need to first extract the Camera-to-reference matrix from calibration file and invert it to get Ref-to-Camera matrix. We can then compute the full camera matrix (combining the intrinsics and extrinsics, Sec 2.1.5 of [Sze10]) and compute dot product of LiDAR points with this matrix. Convert the obtained points into homogeneous system. Further, we store a list of point sets which lie inside the corresponding camera image dimensions. The output from source code is depicted in Fig 3.3 with laser points plotted on image (darker the laser point, more is its reflectivity).

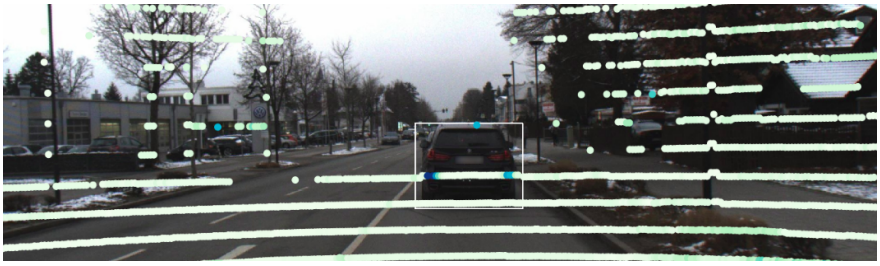


Fig 3.3 Projection of Laser onto Image (Scene 198) [MK19] [Git]

The bounding box information is provided in the Ground Truth file of corresponding Scene ID. We employed a python class that reads this file and stores variable like dimensions of Vehicle ($ie : l, b, h$) and distance from Ego to Vehicle ($ie : d$). These values are used to compute the corner points of object in global frame and then convert to image frame using transformation matrix.

3.4 Output from Astyx - Result.

To count the number of LiDAR measurement recorded inside the bounding box, we programmed a looping mechanism for all points and check if they lie inside the bounding box dimensions. They are added to a new list which plot only the points specifically inside the box and displays the count. Fig 3.4 is an output of the source code. The source code can be referred at personal Github repository [Git].

```
Total LiDAR measurements for given object = 305
```

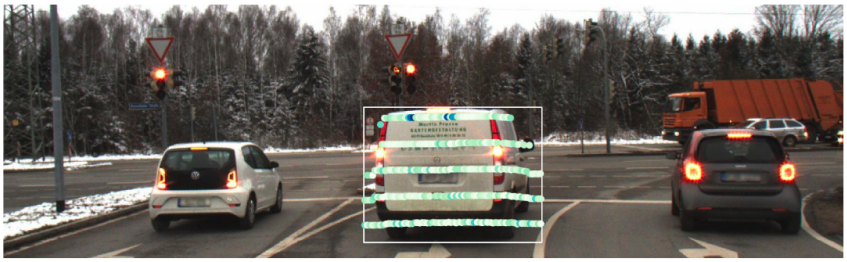


Fig 3.4 Demo: counting relevant points (Scene 77) [MK19] [Git]

In the next section, we take a few scenes and count the LiDAR measurement and try to describe the difference with estimated value, if any. The scenes are listed as below and output of these scenes are depicted in Fig 4.1-4.3.

4 Evaluation - Prediction vs Astyx.

Why is there differences in expected and actual values ?
In **Scene 85 (Object id: 0)** with a vehicle **10.7 meters** away, we expect around 155 laser beams to strike the car as per Table 2.5. On the other hand we actually only **130 recordings**. From the Fig 4.1 we can see that the vehicle is neither occluded or rotated. But, we notice that a bunch of laser beams are not reflected from the rear windshield area. These types of surprises are common in the 3D driving world. Here it may be the case that reflectivity of the glass is not so good under given light conditions and beams actually pass through it rather than reflecting. Or it may happen that they may be reflected in a scattered direction with low intensity, thus making it impossible for receiver to register the returns.
Rotation (in Quaternions): [0.999, -0.031, -0.018, -0.023]

Fig 4.1 Counting relevant points (Scene 85 [MK19] [Git])

In **Scene 197 (Object id: 0)** we can see a SUV at a distance of approx **14.2 meters**. As per our estimation in Table 2.5, we expect ($n_e = 69$) laser points, but in reality we get ($n_e = 85$) **LiDAR measurements**. The actual LiDAR measurements recorded would have been much greater but some beams are not reflected back due to the reflectivity of glass hypothesized in paragraph above (see Fig 4.2). But the greater reason for the difference in estimated and recorded value is the size of the car. The vehicle in the scene is far more broader and higher than the selected vehicle for estimations.
Rotation (in Quaternions): [0.999, -0.031, -0.018, -0.012]

Fig 4.2 Counting relevant points (Scene 197) [MK19] [Git]

Scene 331 (Object id: 0) has an vehicle at distance of approx **18.9 meters**. The

actual laser recordings ($n_a = 71$) is way higher than expected value ($n_e = 26$). There could be two possible reasons of this mismatch. First, the vehicle appears to be a SUV which is much more in dimensions as compared to a VW Golf, the considered vehicle for estimation. Greater height of the vehicle means more beams could strike it. Second reason is that the vehicle is not directly in front of the ego (see Fig 4.3). Rotation of the vehicle allows for more exposure area and thus more beam measurements.
Rotation (in Quaternions): [0.9985, -0.0328, -0.0164, 0.0385]

Fig 4.3 Counting relevant points (Scene 331) [MK19] [Git]

Unfortunately, we couldn't setup any scene from the Astyx dataset to compare to the 5 meter distance estimations. Most of the scenes contain incomplete information which can't be compared (see Fig 4.4).
Rotation (in Quaternions): [-0.108, 0.021, -0.030, -0.9933]

Fig 4.4 Counting relevant points (Scene 31) [MK19] [Git]

5 Future Scope

We have taken a few assumptions in comparing the count of beams. We have not considered the rotation of vehicle during data extraction from Astyx. This approach could be further improved to get more accurate data for comparison. Assumptions were also taken during computing the distance of vehicle from ground truth files of Astyx dataset. The distance is given in x, y, z co-ordinates but for our estimation we considered x dimension only. This can be improved when considering angle of rotation. With further refinement of the estimation technique, the validation could produce much more promising results.

References : Books

[Sze10] R. Szeliski. *Computer Vision: Algorithms and Applications, 1st ed.* Springers, 2010.

Articles

[MK19] M. Meyer and G. Kuschk. "Automotive Radar Dataset for Deep Learning Based 3D Object Detection". In: (2019), pp. 129–132.

[BR20] F. Berens and Hochschule Ravensburg Weingarten. "Lecture notes in Lidar and Radar Systems, How to Astyx Complex Yolo". In: (Nov. 2020).

[Cae+20] Holger Caesar et al. "nuScenes: A Multimodal Dataset for Autonomous Driving". In: (June 2020).

Webpages

[Ast] Astyx. *Astyx Dataset Specification*. URL: `https://www.astyx.com/fileadmin/redakteur/dokumente/Astyx_Dataset_HiRes2019_specification.pdf`. (accessed: 10.12.2020).

[Git] Github. *sa32953/lidar*. URL: `https://github.com/sa32953/lidar`. (accessed: 10.12.2020).

[Gmb] Astyx GmbH. *Astyx HiRes2019 Dataset*. URL: `https://www.astyx.com/development/astyx-hires2019-dataset.html`. (accessed: 10.12.2020).

[Jup] Project Jupyter. *Project Jupyter*. URL: `https://jupyter.org/index.html`. (accessed: 10.12.2020).

[Lid] Velodyne Lidar. *Puck Lidar Sensor*. URL: `https://velodynelidar.com/products/puck/`. (accessed: 10.12.2020).

[M D] et al M. Droettboom. *MatplotLib*. URL: `https://matplotlib.org/`. (accessed: 10.12.2020).

[Num] Numpy. *NumPy*. URL: `https://numpy.org/`. (accessed: 10.12.2020).

[Way] Waymo. *Waymo*. URL: `https://waymo.com/`. (accessed: 10.12.2020).

[Wika] Wikipedia. *Lidar*. URL: `https://en.wikipedia.org/wiki/Lidar`. (accessed: 10.12.2020).

[Wikb] Wikipedia. *Minimum Bounding Box*. URL: `hhttps://en.wikipedia.org/wiki/Minimum_bounding_box`. (accessed: 10.12.2020).