HOCHSCHULE
RAVENSBURG-WEINGARTEN
UNIVERSITY
OF APPLIED SCIENCES

Lidar And Radar Systems

Assignment-2

Master Mechatronics

submitted by:

Nishanth Nandakumar

Mtr.No:

February 11, 2022

Prof. Dr.rer.nat. Stefan Elser

1 Assignment 2

1.1 Introduction

The main objective of this project is a comparison of resolution between the Blickfeld and Velodyne point cloud. This is achieved by finding the total number of points detected from the object(car) for both the sensors and comparing them. The dataset consists of the point clouds for the sensors with the ground truth bounding boxes which can be used to achieve this task as shown in Figures 1 and 2.

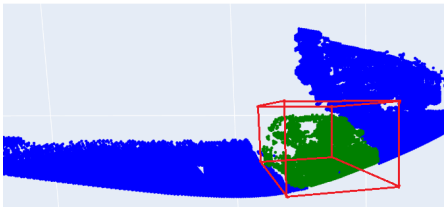

Figure 1: Blickfeld point cloud with bounding box

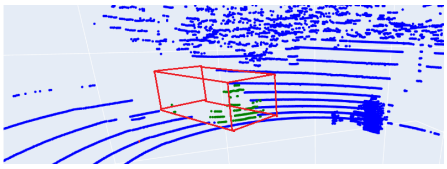

Figure 2: Velodyne point cloud with bounding box

The dataset provided consists of two recorded data, here we will be using the Record 1 data which includes the car moving forward and backward. This is used to find the number of points inside the bounding box in the point cloud data based on the distance from the sensor.

1.2 Algorithm and Calculations

Here, the code provided for assignment 1 is used that provides the visualization of the data points along with the ground truth bounding box. Once we have the eight corners coordinates for the bounding box from the code of assignment 1, we find the maximum and minimum values in x, y, and z directions for these eight corners using the minmaxbb function provided. The bounding_box is an array of shape 8x3, every points x,y,z are compared to obtain the maximum and minimum values.

```
def minmaxbb(bounding_box):
  #Here we get max and min for bounding box
  #Initialize the variables with some maximum and
  #minimum values

  x_max = -10000
  x_min = 10000
  y_max = -10000
```

```
    y_min = 10000
    z_max = -10000
    z_min = 10000
    for corner in bounding_box:
      if corner[0] >= x_max:
        x_max = corner[0]
      if corner[0] <= x_min:
        x_min = corner[0]

      if corner[1] >= y_max:
        y_max = corner[1]
      if corner[1] <= y_min:
        y_min = corner[1]

      if corner[2] >= z_max:
        z_max = corner[2]
      if corner[2] <= z_min:
        z_min = corner[2]


    return x_max, x_min, y_max, y_min, z_max, z_min
```

Now that we have the minimum and maximum values in the x, y, and z directions we define a boundary condition to check if the given point lies within the bounding box using the separatepoints function. Here, every point in the point cloud is checked if they lie between the minimum and the maximum x, y, and z values obtained and the points which fulfill this condition are added to a list. The output of this function is a list consisting of all the points lying within the bounding box which is used to get the number of points inside the bounding box.

```
def separatepoints(data, bounding_box):
  #Here we separate the points from the data
  car = []

  x_max, x_min, y_max, y_min, z_max, z_min =
  minmaxbb(bounding_box)
  for point in data:
    if (point[0] >= x_min) and (point[0] <= x_max)
    and (point[1] >= y_min) and (point[1] <= y_max)
    and (point[2] >= z_min) and (point[2] <= z_max):
      car.append(point)
```

```
return car
```

The number of points obtained per frame is to be compared with the distance of the car from the sensor in that frame. This can be achieved by finding the distance to the front portion of the car. To have uniformity in the readings the minimum value in the y-direction from the bounding box output from minmaxbb function is used. Please note while finding the distance for the Velodyne data we use the absolute of y values as the readings are on the negative axis. The frame-by-frame readings for the Record 1 dataset for both the Blickfeld and Velodyne sensors with the distance of the car are provided in table 1. Figures 6 to 20 in the appendix are provided for better visualization of the work done. As can be seen, the red points represent the eight corners of the bounding box. The green points represent the points within the bounding box that determine the car. The blue points represent all the remaining points detected outside the bounding box.

1.2.1 Resolution Comparison

The data obtained from table 1 is used to plot graphs with a number of points on the y axis and distance to the car in meters on the x-axis as shown in figures 3 and 4.
The graph obtained from Blickfeld sensor is smoother compared to the Velodyne sensor. We can observe that for both the sensors the number of data points decreases as the distance between the sensor and the car increases.
The car is closest to both the sensors in frame 20 which is at a distance of 3.87m for Blickfeld and 3.94m for Velodyne. The number of points detected at these distance are 3942 and 252 for Blickfeld and Velodyne respectively. As can be observed the detected points for Velodyne at about 3.9m are just 6.4% of the points detected by Blickfeld sensor. The number of points detected exponentially decreases from 3942 to 98 points at a distance of 23.21m for Blickfeld. The number of points detected by Velodyne decreases to 0 as distance increases but not uniformly as we observe in the Blickfeld measurements.
The graph in figure 5 displays both the plots for Blickfeld and Velodyne. We can clearly observe that the measurements of Blickfeld is much higher than the Velodyne sensor at any particular distance. For better comparison, the area under the curves is found. As we have the points on the graph we use the trapezoidal rule in python to find the area under each graph. It is found that the areas are 1850100.0 m^2 and 100842.0 m^2 for Blickfeld and Velodyne respectively. The overall resolution of Velodyne is only 5.4% of the Blickfeld sensor based on the area under the curve.

1.3 Conclusion

The forward/backward scene of the dataset is used to get the data points of the car detected from the sensors at different distances. The algorithm to find the data

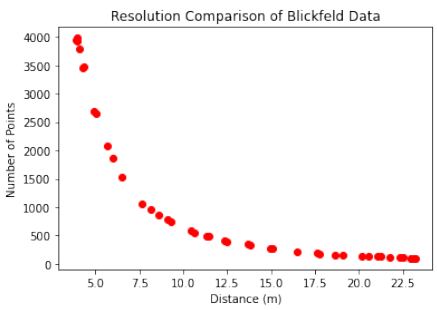

Figure 3: Resolution Comparison graph for Blickfeld data

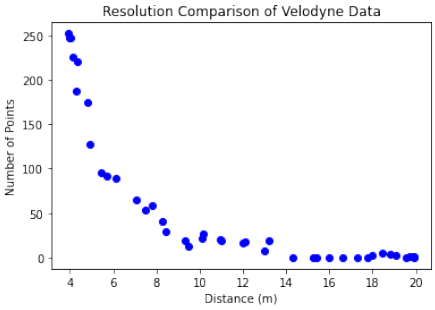

Figure 4: Resolution Comparison graph for Velodyne data

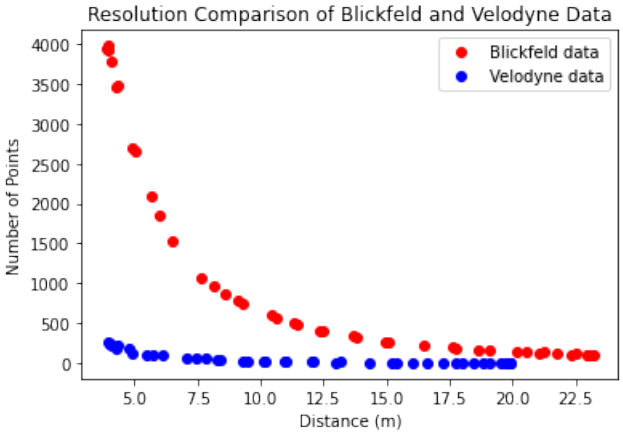

Figure 5: Resolution Comparison graph for both the sensors

points is provided and explained in the earlier section. We also plot the graphs for both the sensors and compare them with respect to each other. After thorough observation, we can conclude the resolution or the number of measurements of Blickfeld is higher compared to Velodyne.

	Blickfeld Record1		Velodyne Record 1	
Frame	ymin/Distance (m)	No. of Points	ymin/Distance (m)	No. of Points
0	22.95437643	106	19.90259253	1
1	22.95703498	103	19.8965012	0
2	22.50125517	113	19.52123492	0
3	21.74555365	125	18.82169614	4
4	21.27045351	131	18.43598271	5
5	20.54237322	136	17.77503699	0
6	19.06806435	159	16.63106814	0
7	17.72109108	184	15.38571562	0
8	15.06021983	264	13.17913053	19
9	13.66206908	342	12.01718784	17
10	12.45097303	396	10.99892831	19
11	11.45011534	488	10.17153696	27
12	10.63803858	554	9.500728939	13
13	9.289910405	748	8.425628855	29
14	8.582054127	864	7.815913921	58
15	7.624037007	1068	7.067722848	65
16	6.012849298	1855	5.720524473	92
17	4.938858399	2695	4.813979418	174
18	4.354321774	3475	4.353029162	221
19	3.951416833	3980	3.99565968	247
20	3.873332499	3942	3.943696057	252
21	3.971940488	3917	4.031291706	247
22	4.062099801	3793	4.110348735	226
23	4.304687854	3452	4.29386939	187
24	5.048569684	2659	4.936966319	127
25	5.676127592	2087	5.458691067	96
26	6.488637808	1535	6.109125134	89
27	8.143620215	964	7.464304665	54
28	9.111589426	783	8.273792232	41
29	10.42229682	593	9.321032923	19
30	11.34738787	497	10.09689282	22
31	12.35914549	408	10.94361376	20
32	13.82702873	323	12.11537831	18
33	14.92107208	265	13.00618443	7
34	16.50141548	221	14.28525975	0
35	17.64140787	193	15.22354994	0
36	18.65266078	162	15.99896978	0
37	20.17348569	137	17.26888029	0
38	21.06269629	126	17.97583655	2
39	22.34914889	109	19.05437058	2
40	23.13686151	102	19.69401348	1
41	23.21248488	98	19.83666306	1

Table 1: Resolution Comparison between Blickfeld and Velodyne for Record 1 dataset

2 Appendix

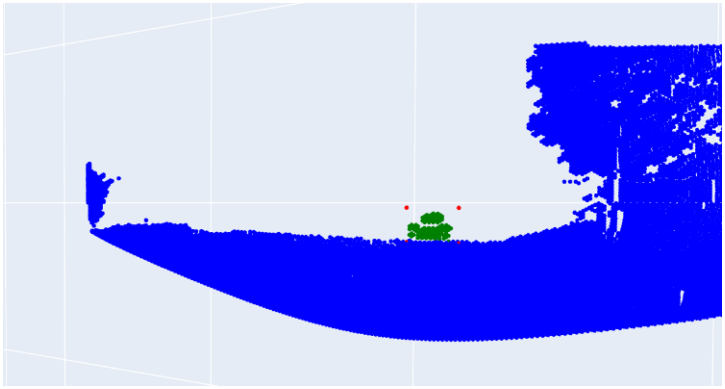

Figure 6: Blickfeld datapoints with object in green for frame 0

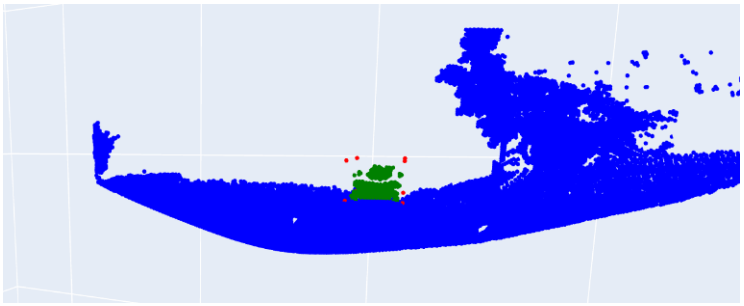

Figure 7: Blickfeld datapoints with object in green for frame 8

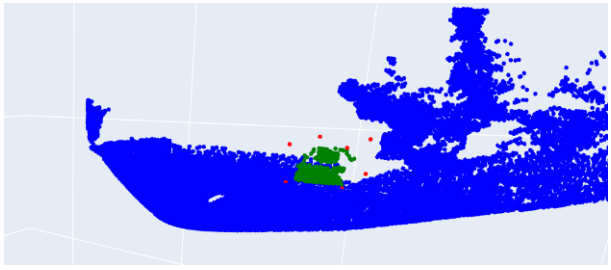

Figure 8: Blickfeld datapoints with object in green for frame 9

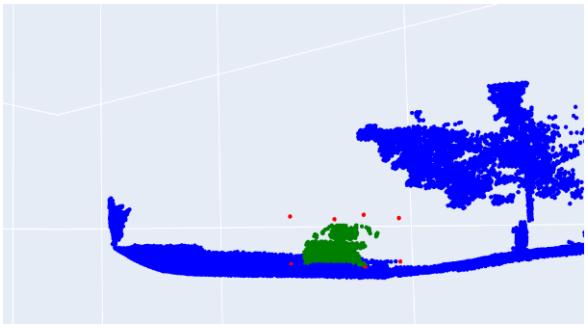

Figure 9: Blickfeld datapoints with object in green for frame 12

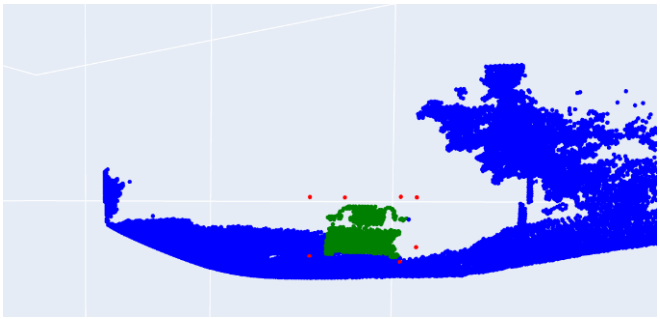

Figure 10: Blickfeld datapoints with object in green for frame 14

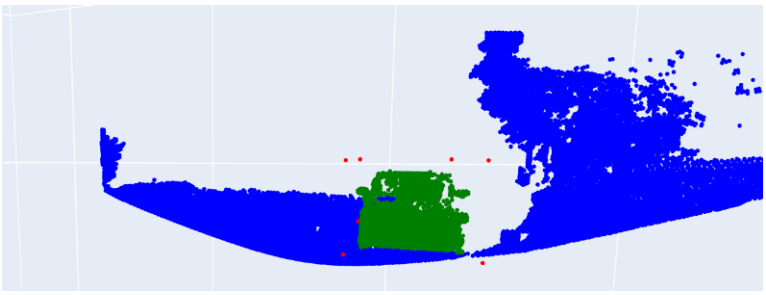

Figure 11: Blickfeld datapoints with object in green for frame 17

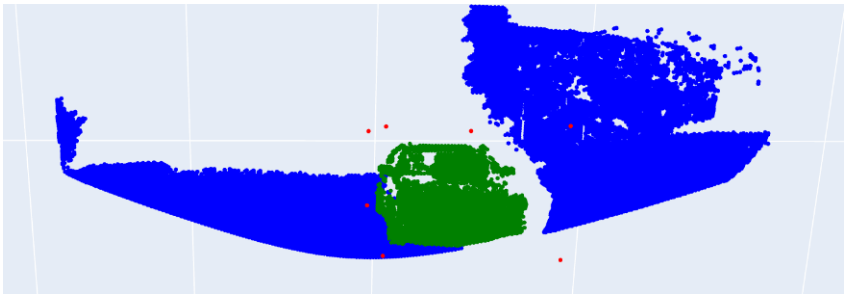

Figure 12: Blickfeld datapoints with object in green for frame 18

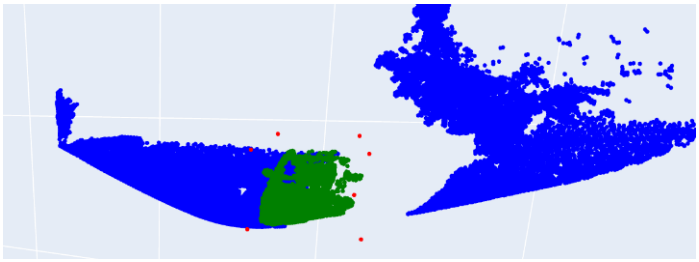

Figure 13: Blickfeld datapoints with object in green for frame 21

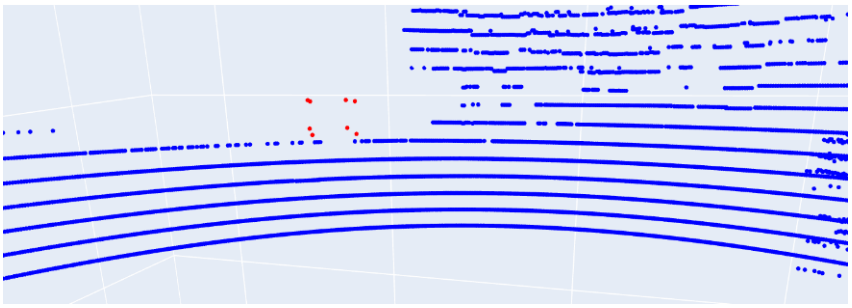

Figure 14: Velodyne datapoints with object in green for frame 1

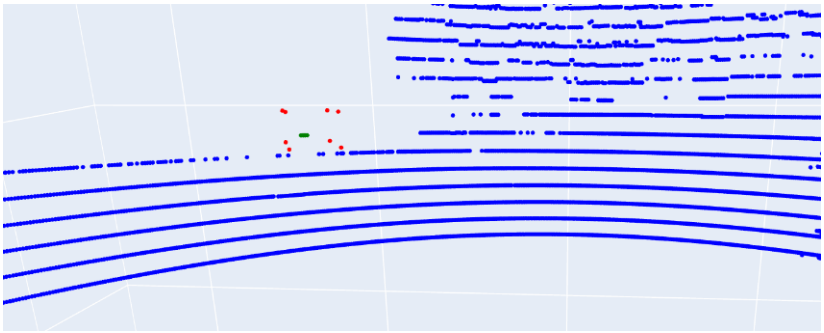

Figure 15: Velodyne datapoints with object in green for frame 3

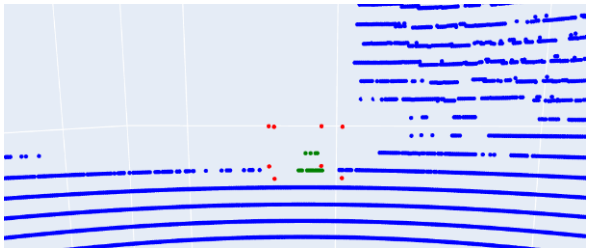

Figure 16: Velodyne datapoints with object in green for frame 9

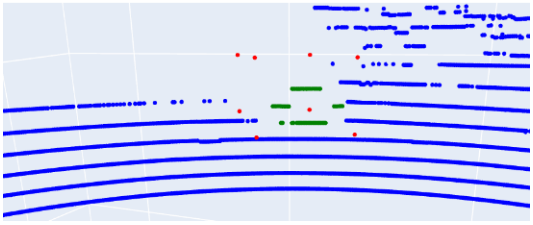

Figure 17: Velodyne datapoints with object in green for frame 14

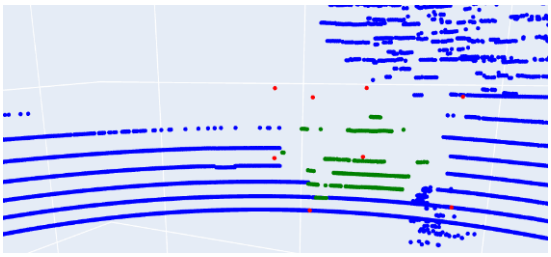

Figure 18: Velodyne datapoints with object in green for frame 18

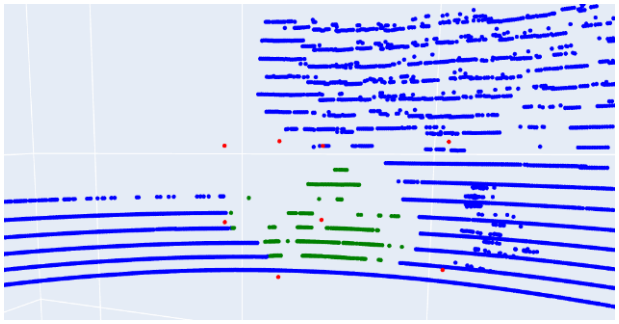

Figure 19: Velodyne datapoints with object in green for frame 19

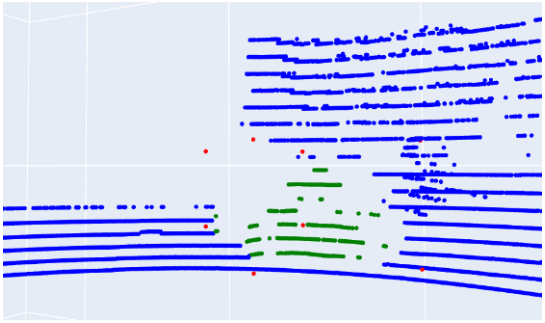

Figure 20: Velodyne datapoints with object in green for frame 20