Deep Learning for Autonomous Driving
Report Project 1

Understanding Multimodal Driving Data

Submitted by

Louis DEZONS & Ghazi CHERIF
ETHZ

2022

Contents

Chapter 1

Exercise 1

For this first task, we had to generate the bird eye view of the lidar's point cloud. To do so, we first rotated lidar's data to have it in the frame of the image. Then we scaled the points coordinate so that they corresponds to image pixel with the right resolution at 0.2. We applied the following transformation:

$$x_{pixel} = (-y_{lidar} - min(y))/0.2 \tag{1.1}$$

$$y_{pixel} = (x_{lidar} - min(x))/0.2 \tag{1.2}$$

The result is rounded (int(...)). then we scale the reflectance from 0 to 255 to associate each pixel to it's right intensity. From that, we obtain the following bird eye view for the demo file:

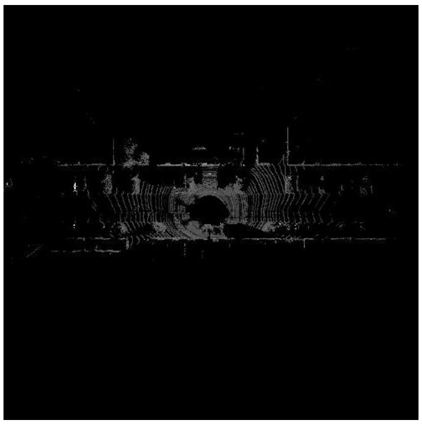

Figure 1.1: Bird eye view of velodyne's point cloud

Chapter 2

Exercise 2

2.1 Question 1

For this second part, we extracted the data from the dictionary and applied respectively the transformation *T_cam0_velo* and *P_rect_20* to velodyne.
However the cloud points was distorted , therefore we had to rescale it. To do so, we took the transformed data and divided it by the z coordinate, we hence obtained a normalized z column and data corresponding to the given example.
After extracting the x and y vectors, we filtered the data by selecting only the data where $x > 0$ as we only want to get a 180 [deg] view.
Eventually we store the color of each point of the points cloud in a vector and plot everything on the image.

2.2 Question 2

Although we were given a large data set *objects*, we only needed some the length, the width, the height and the rotation of the car as well as the car's coordinates in the camera frame.
As stated in the problem, the rotation is around y and therefore the matrix is of the form:

$$\begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix}$$

After declaring an array containing the location of the eight different points characterising the box ,we rotated the array around y and translated every

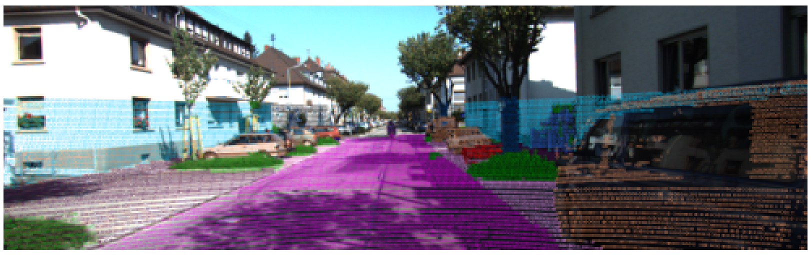

Figure 2.1: Obtained image

coordinate by the car's coordinates in the camera frame.
Eventually before applying *P_rect_20* transformation, we added a line of 1's to allow the rigid transformation.
Finally we normalize the data by the z coordinate and plot a line between every two adjacent points.

2.3 Question 3

In this question, we were asked to associated points from the velodyne's point cloud to its corresponding class. We did the same operation as in question 2.1 without rotating and projecting the point cloud onto the image. We have just associated each point to its class and colored it with the class-determined color. We did exacly the same for the cars boxes. We defined the coordinates of each summit in the lidar's frame and did not project it in any image frame. We obatin the following plot with vispy:

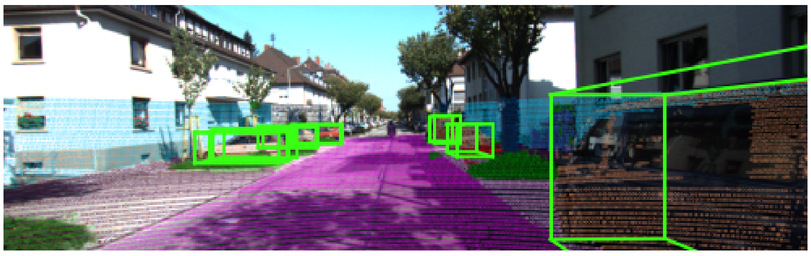

Figure 2.2: Final image showing the boxes around the cars

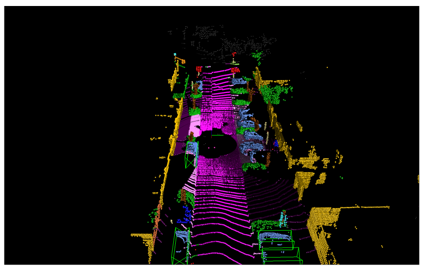

Figure 2.3: Final plot showing the colored point cloud and the cars bounding boxes in vispy

Chapter 3

Exercise 3

For this third part, we had to identify to what channel belonged every point. After extracting the data, we computed the phase in [deg] of each point using the *arctan* function. For a point to belong to a channel i, it has to satisfy the following condition : $\theta_{channel_i} \geq \theta > \theta_{channel_{i-1}}$
Hence we wrote the function *Classify* to verify the following condition for each of our θ, it takes an array in entry and initializes a 64 values linspace between its max and its min and computes to which channel belongs the point.
We used 4 colors as done in the example, therefore we initialize a variable corresponding to the output of *Classify* modulo 4, each number corresponding to a certain color.
Then we apply successive transformations, keep the data where $x > 0$ and plot the points cloud on the image. We end up with the image below.

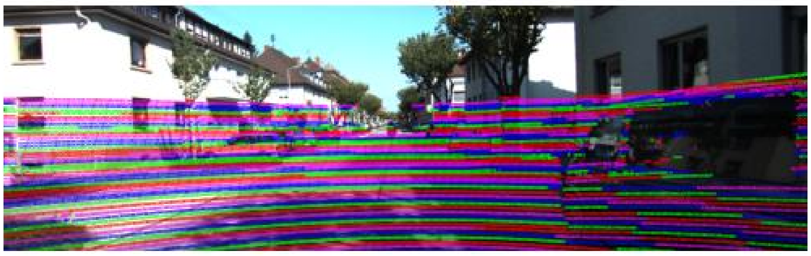

Figure 3.1: Final image showing the different lidar channels

Chapter 4

Exercise 4

For this fourth task, we had to take the motion of the car into account while plotting the Lidar's point cloud onto the images.
First we have plotted the point cloud onto image 37 without making any modifications related to car's motion.

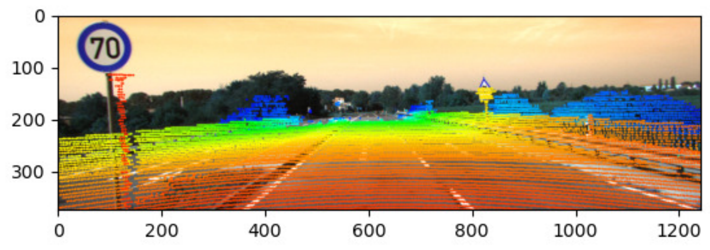

Figure 4.1: Lidar's point cloud projected onto image 37 without any motion correction

To do so we simply used the same transformations as in question two. Namely, the intrinsic and extrinsic corresponding parameters.
Then, we have untwisted the lidar's point cloud using the IMU positional

data. We have first transformed lidar's data to be in the IMU frame. We were provided the rotational and translation matrices to go from IMU's frame to Velodyne's frame. Therefore, we applied the following transformation to lidar's data:

$$IMU_(frame) = R(imu - to - lidar)^{-1} * [Lidar(data) - T(imu - to - lidar)] \tag{4.1}$$

After that, we have applied an untwisting transformation to the points in the IMU's frame. We used the following matrix to rotate and translate the data:

$$M = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 & dx \\ sin(\theta) & cos(\theta) & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

With θ the rotation along the z axis of the car during a rotation of the lidar and the snap of the camera. and dx, dy, dz the infinitesimal translation of the car in the IMU's frame during a lidar's rotation. Then we bring the points back into the lidar's frame doing the inverse of equation (4.1). The points are not placed back in a frame which takes into account the motion of the car. We can therefore plot the corrected points onto the image. For image 37 we obtain the following plot:

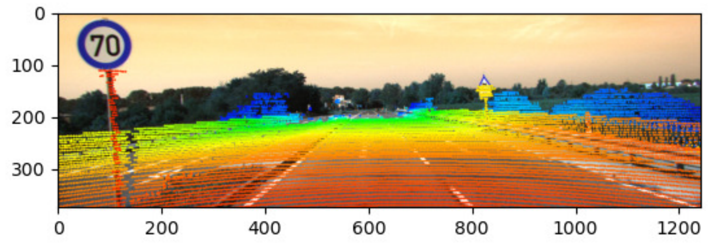

Figure 4.2: Lidar's point cloud projected onto image 37 with motion correction