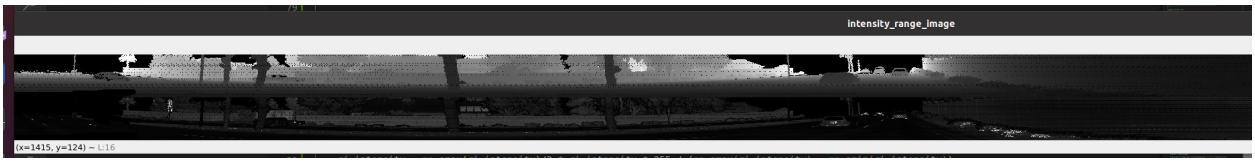
Section 1 : Compute Lidar Point-Cloud from Range Image

Visualize range image channels (ID_S1_EX1)

In this section we visualize a first image showing:

- Range image
- Intensity image
- Intensity and Range one in top of the other (vertically stacked)

This image represents the cropped area in front of the vehicle (+- 45º). The full area can be seen in the image below:



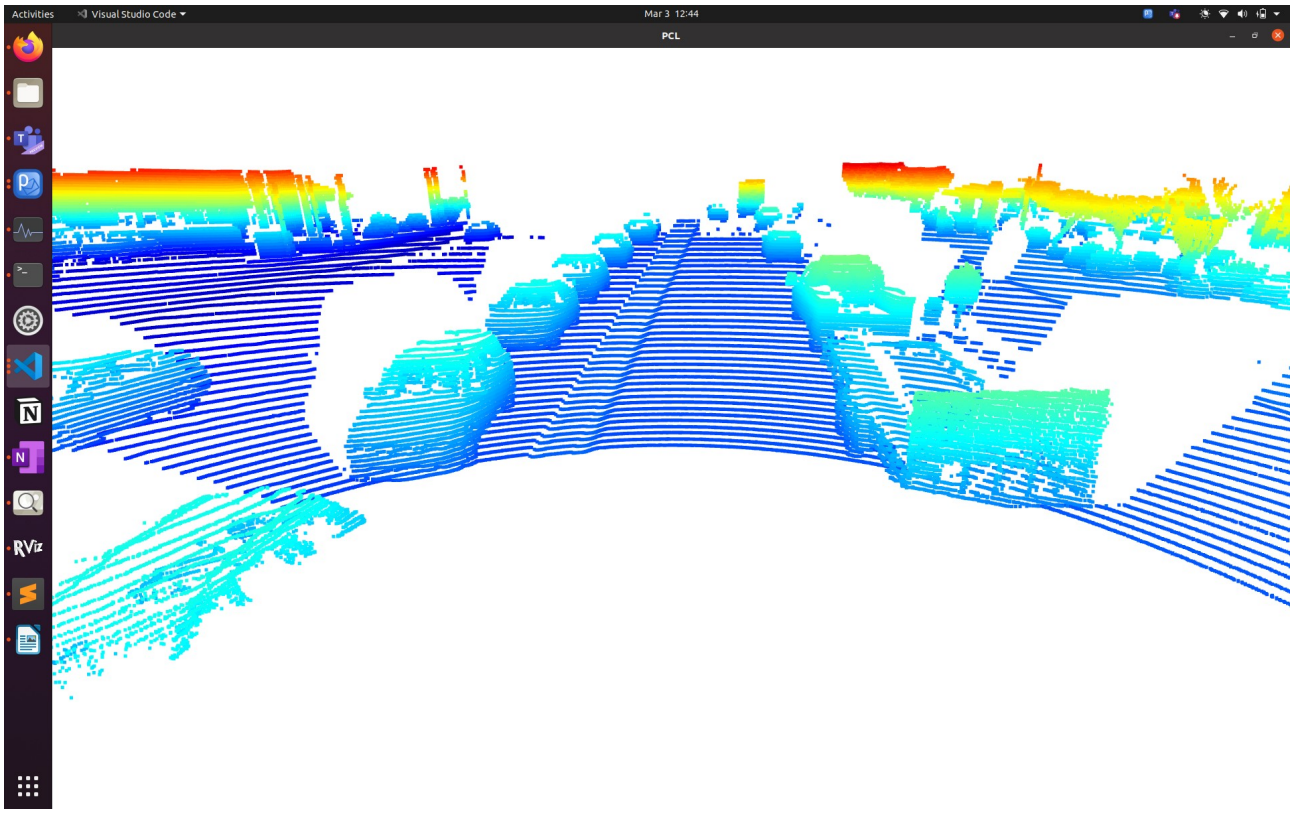
The code for adapting the range and intensity inputs into the desired output is:

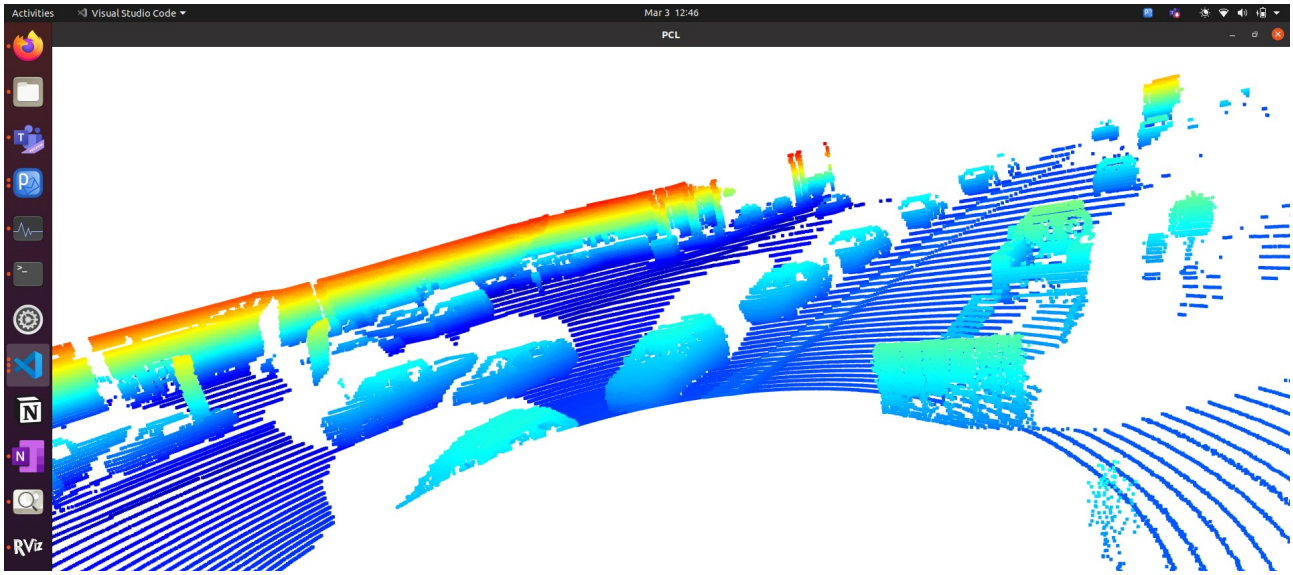
ri_range = ri_range * 255 / (np.amax(ri_range) – np.amin(ri_range))

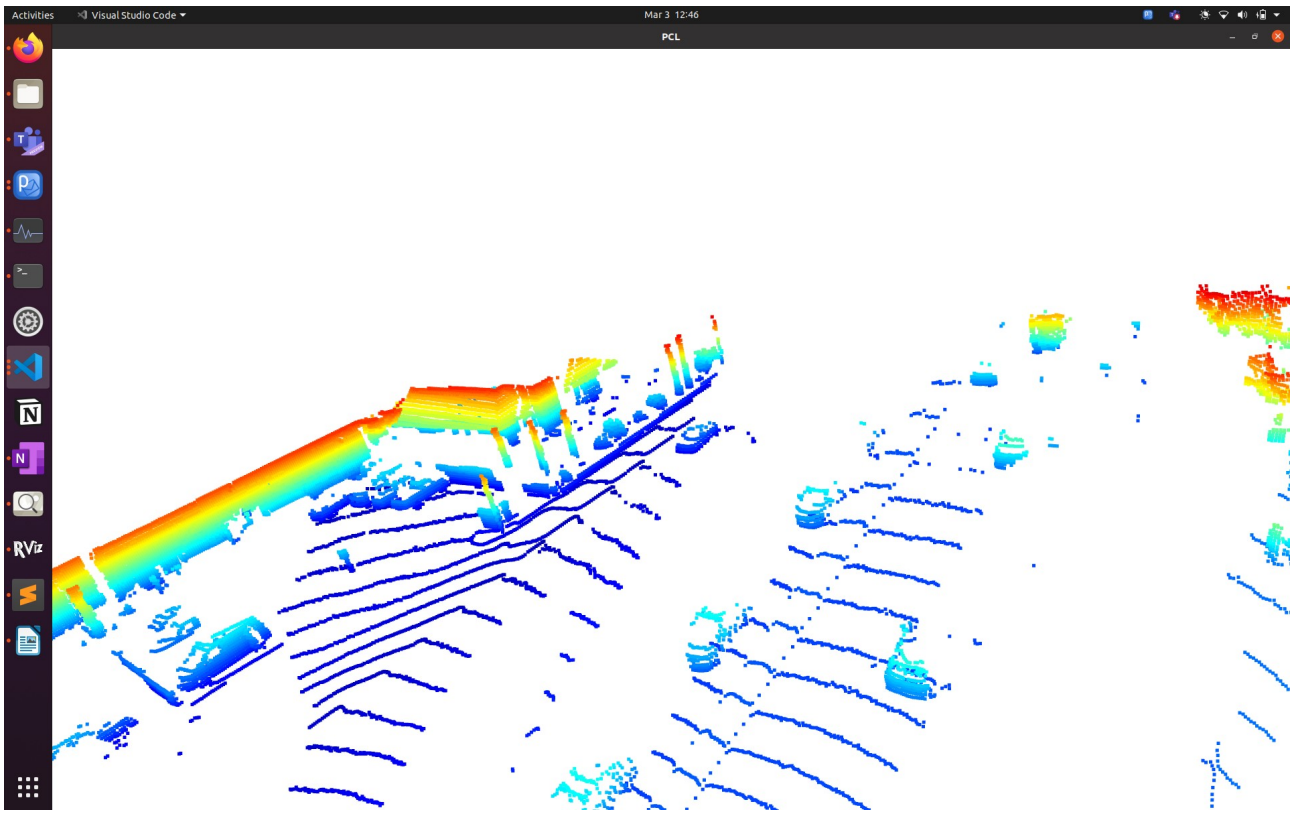
ri_intensity = np.amax(ri_intensity)/2 * ri_intensity * 255 / (np.amax(ri_intensity) – np.amin(ri_intensity))

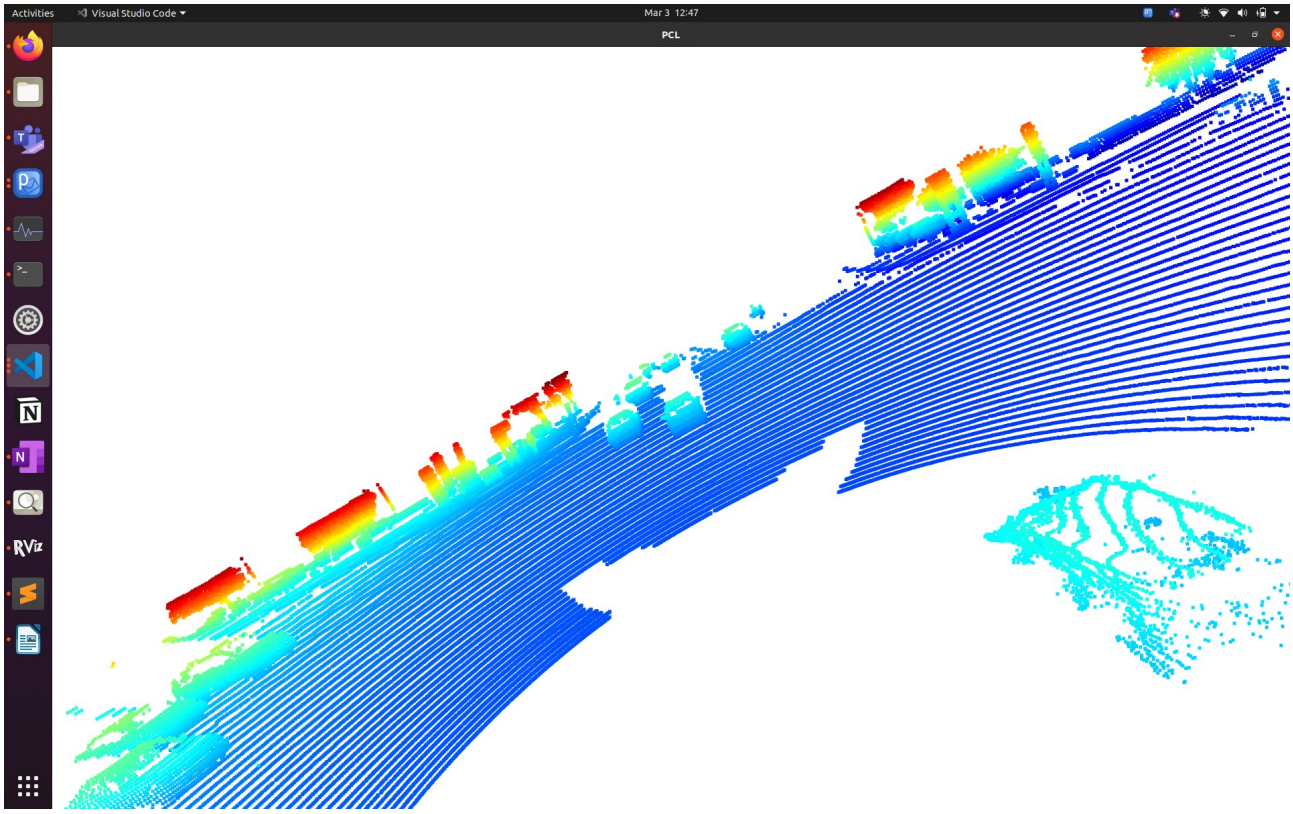
Visualize lidar point-cloud (ID_S1_EX2)

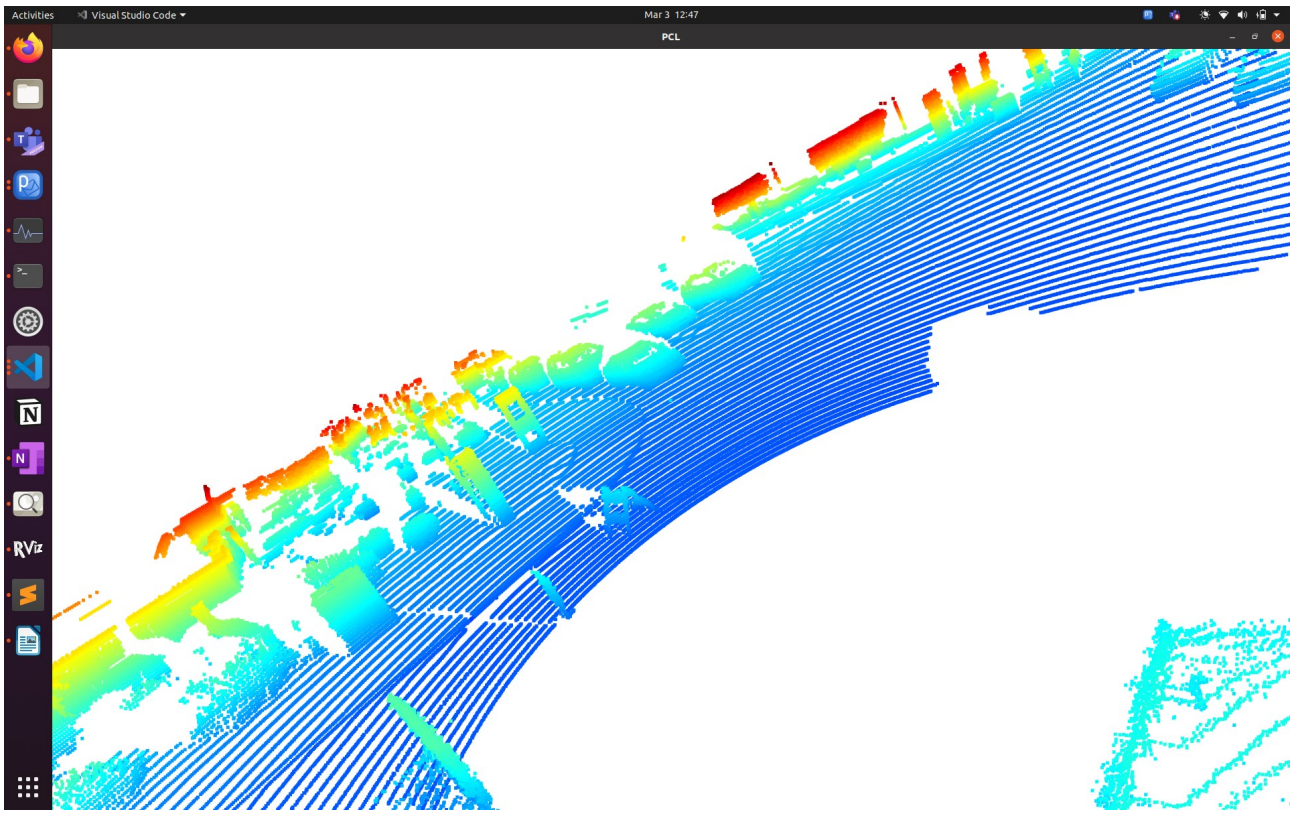
In this section we are visualizing the point cloud in a 3D environment as shown in the images below with the purpose of identifying the main and most stable vehicle features no matter the car's degree of visibility. Here are 5 images showing different cars in different visibility conditions:






Activities
Visual Studio Code
Mar 3 12:46
PCL
RViz


Activities
Visual Studio Code
Mar 3 12:47
PCL
RViz



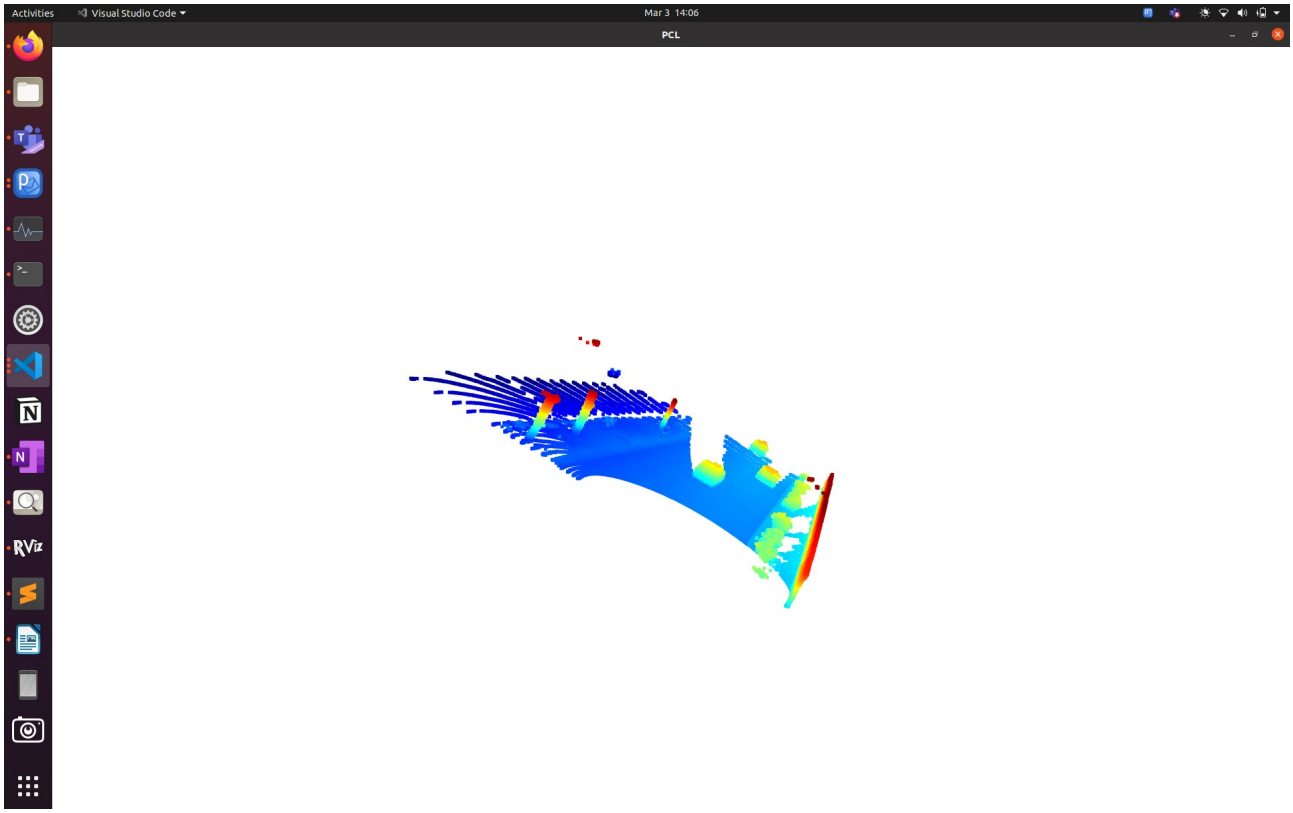
The main and most stable vehicle features are:

- Horizontal surface corresponding to the vehicle’s roof.
- Void in the pcl corresponding to the windows.
- A big cluster of points corresponding to the hood of the car.
- If you visualize the color of the points depending on the intensity channel, the license of the cars (and even the front and back lights) are highlighted.

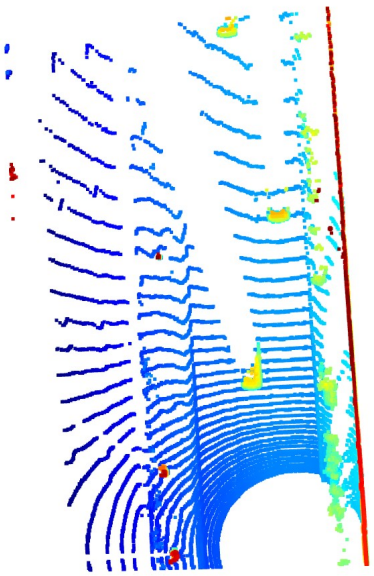
Section 2 : Create Birds-Eye View from Lidar PCL

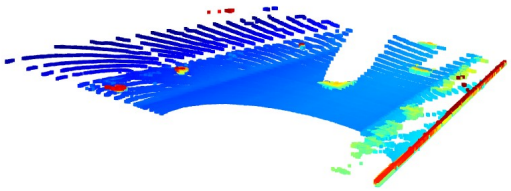
Convert sensor coordinates to BEV-map coordinates (ID_S2_EX1)

In this first image we can see the pointcloud in the same way as in the previous section:

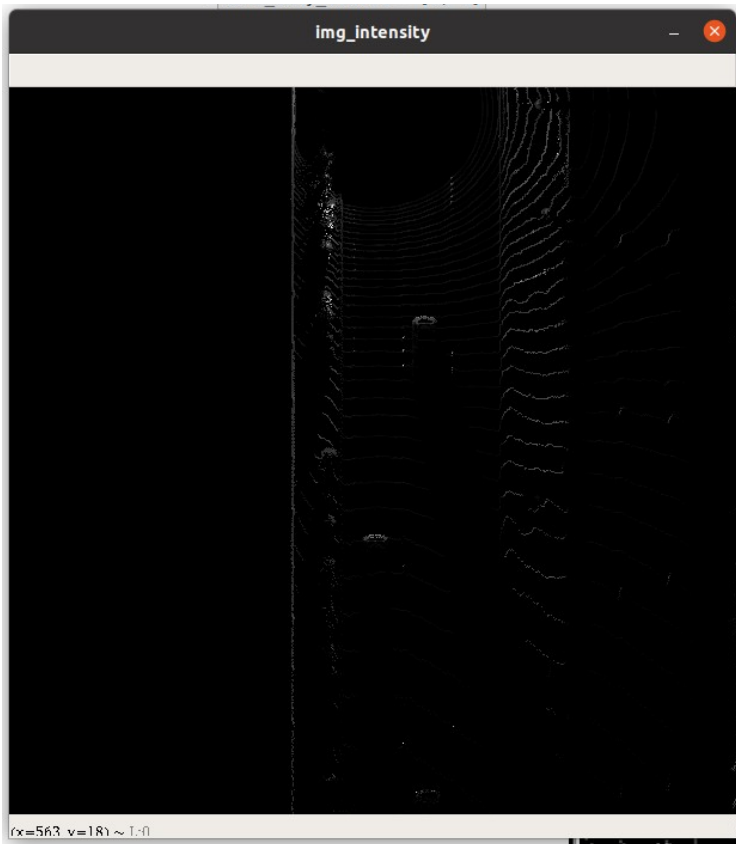


Now, we transform the coordinates of that pcl and convert the height into a different dimension, with the result shown in the 2 images below:

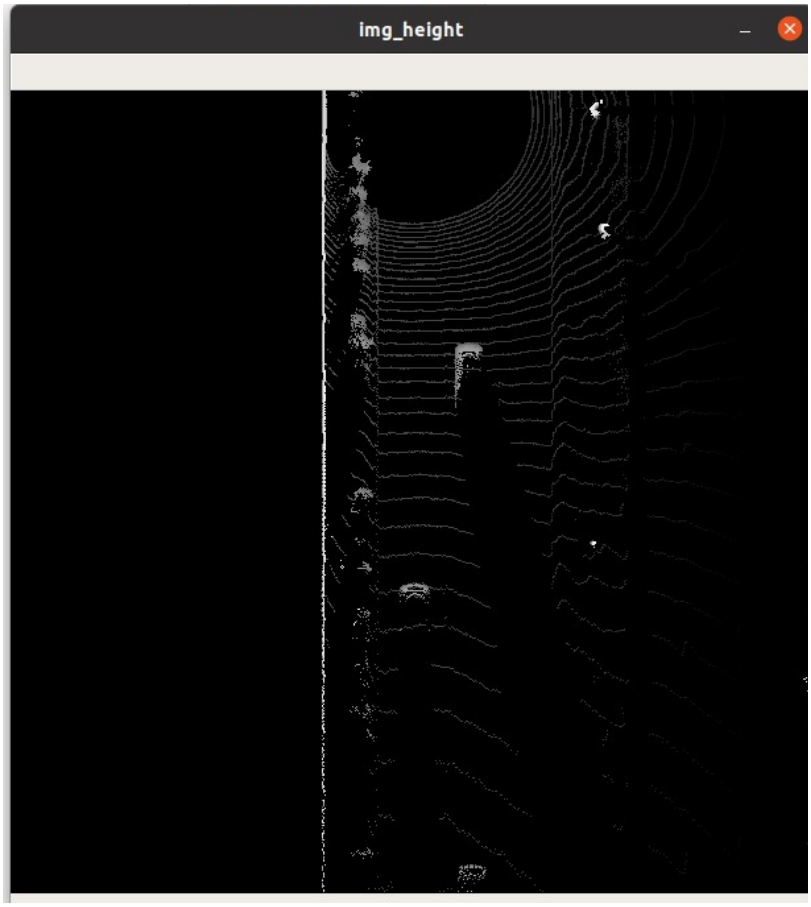




Compute intensity layer of the BEV map (ID_S2_EX2)

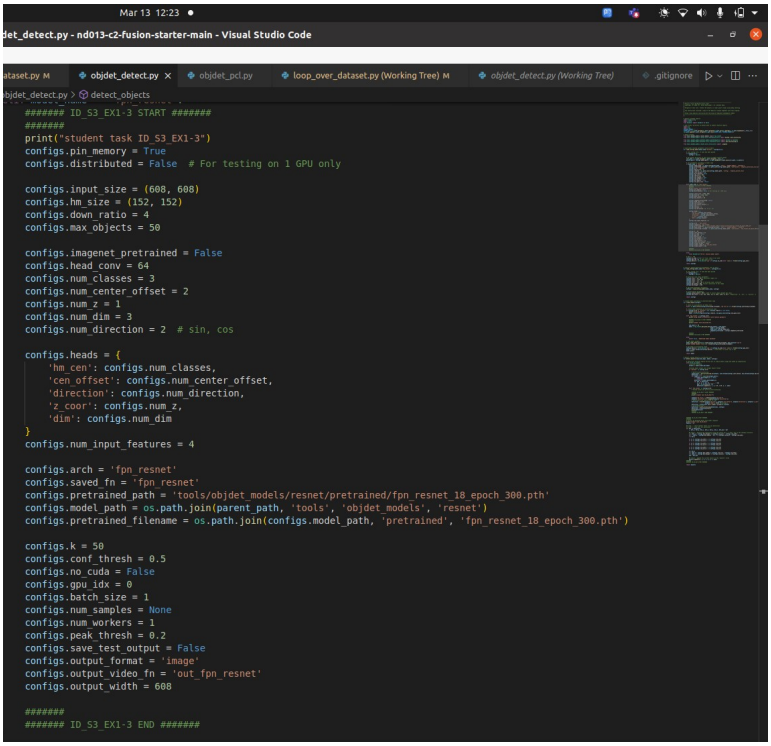


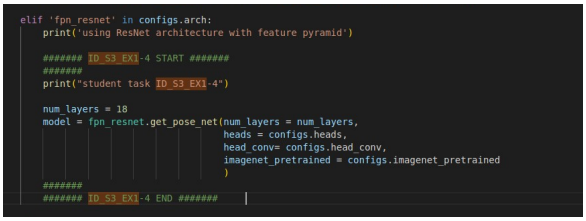
Compute height layer of the BEV map (ID_S2_EX3)

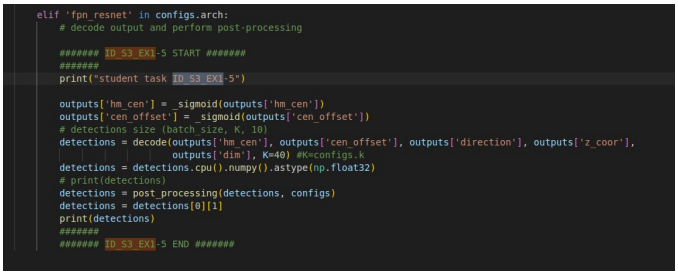


Section 3 : Model-based Object Detection in BEV Image

Add a second model from a GitHub repo (ID_S3_EX1)





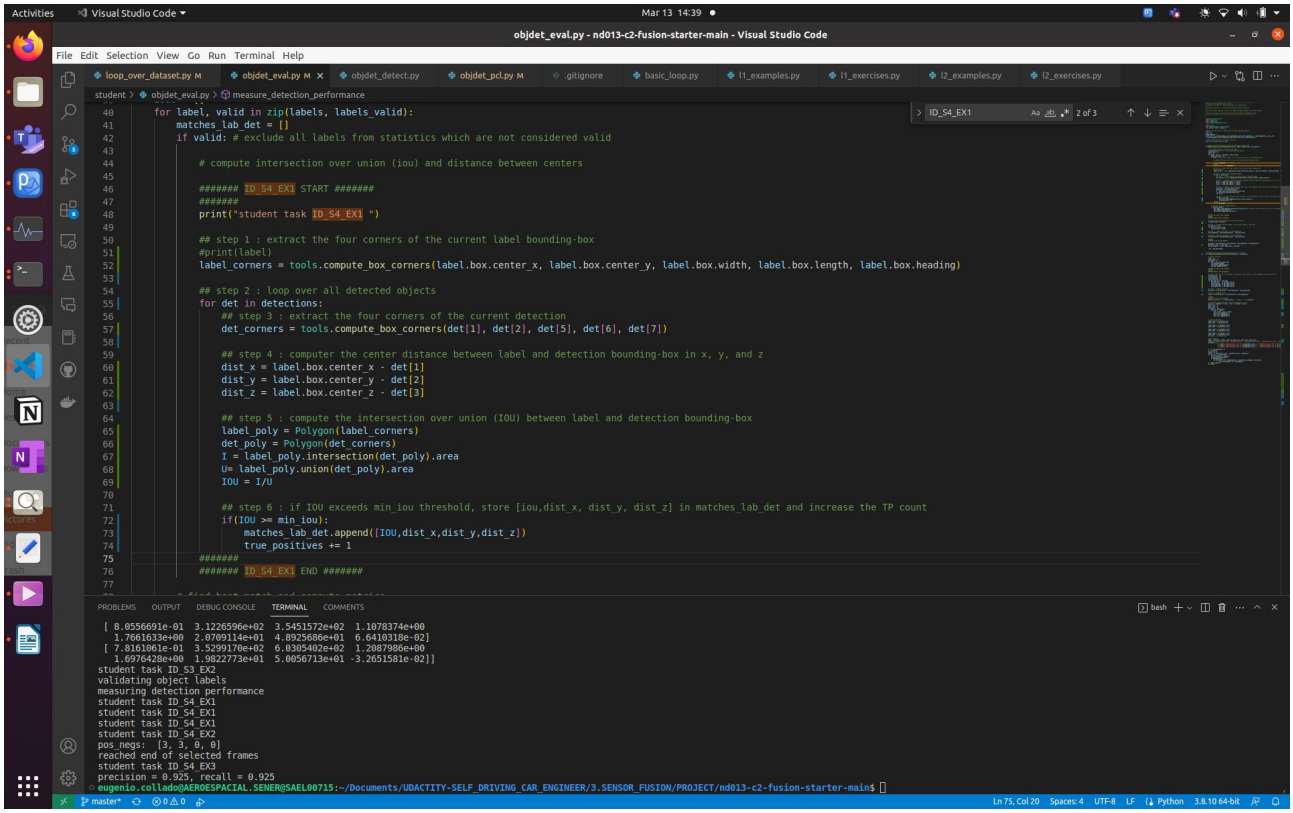


Extract 3D bounding boxes from model response (ID_S3_EX2)

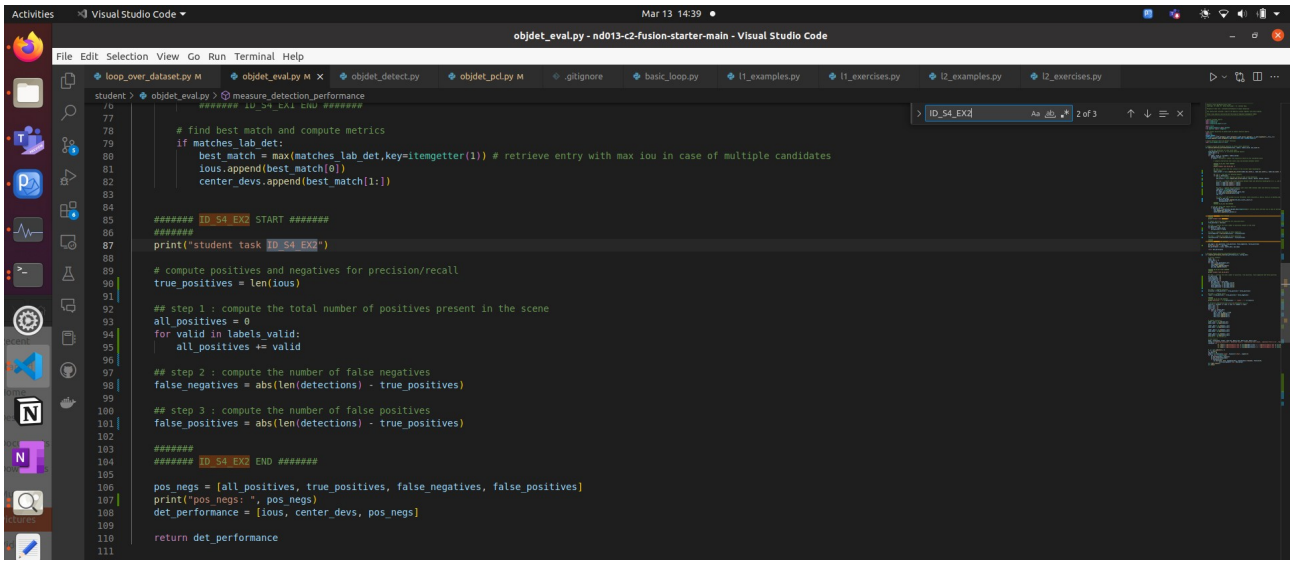
In this section we are working with the transformed pcl into RGB image wich is fed to the model generated in the previous task. The detections predicted by that model can be seen in the images below:

Section 4 : Performance Evaluation for Object Detection

Compute intersection-over-union between labels and detections (ID_S4_EX1)



Compute false-negatives and false-positives (ID_S4_EX2)



Compute precision and recall (ID_S4_EX3)

In this last task, we are computing the precision and recall of the model. In the image below is shown a frame where the model is predicting the presence of a car due to the confusion with the plants in the side of the road (false positive).

H

The precision and recall for the 50 first frames are: