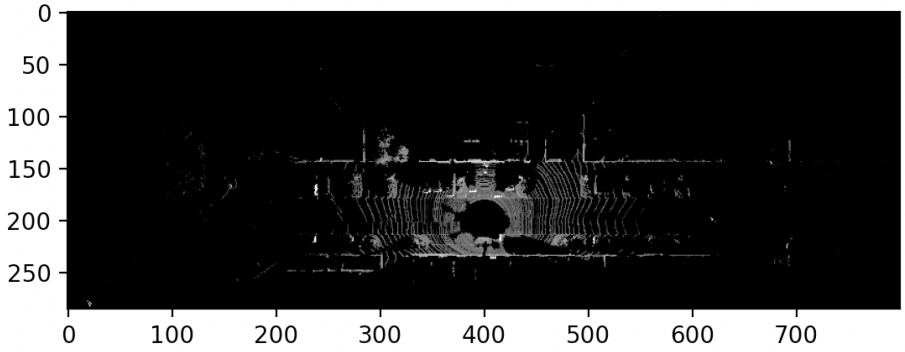
DLAD Project 1

Silvio Mazzucco, Andrea Bionda

March 2022

1 Problem 1

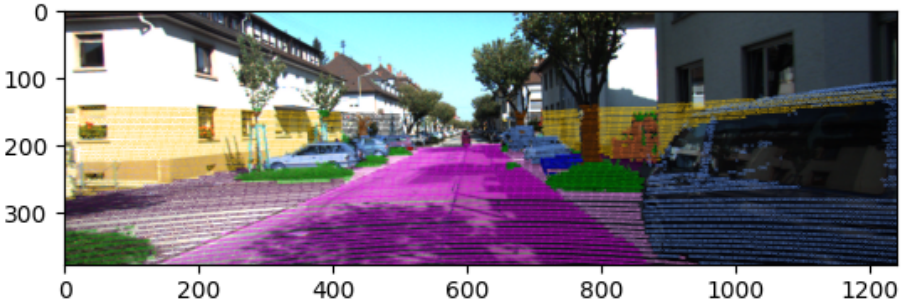
In this problem we had to get a BEV image from the 3D points captured by the velodyne. Firstly, we loaded the 3D point coordinates and reflectance values from the data file. Then, we initialized with zeros a 2D-array. In order to do so, we computed the min and max X Y coordinates of the points, we multiplied them by 5 (same as dividing by 0.2, which is the resolution) and padded it adding 5 for both X and Y for plotting purposes. Now we shifted all the points coordinates adding the min coordinate values (which is negative for both X and Y), to avoid problems in the projection on BEV image. To get the BEV coordinates, for every point we verified in which bin they were placed, taking the floor value of the XY coordinates multiplied by 5. Then, if the reflectance value of the point was greater than the current one set in the image, and the Z value was lower that a certain threshold (here set at 3, which can be a good approximation of the max height of a truck) to avoid flying objects, we updated the bin/pixel intensity. Finally, we rotated by 90° the picture as in the problem description, and we plotted it.



2 Problem 2

In the first part, we had to project the 3D points onto the image taken by Camera 2 paying attention to associating them to the correct colour (directly

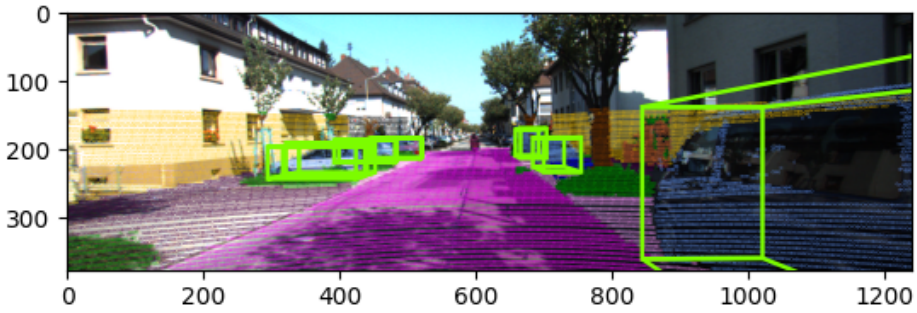
linked to the category belonging). For every 3D point taken by the velodyne which has positive X coordinate (it is not behind the camera thus it can be seen by it), we multiplied it with the transformation matrix to obtain the points in the Cam 2 reference frame. Then, we multiplied by the intrinsic matrix K to project the points to the camera plane, and normalized the pixel coordinates dividing by the third element of the array. Afterwards we took the label corresponding to the point (they have the same index), and retrieved the color. After converting it from BGR to RGB, we appended every point color to the previously created 1D array. We finally divided the color values by 255 to have them in the range 0 to 1, as asked by the function plt.scatter.



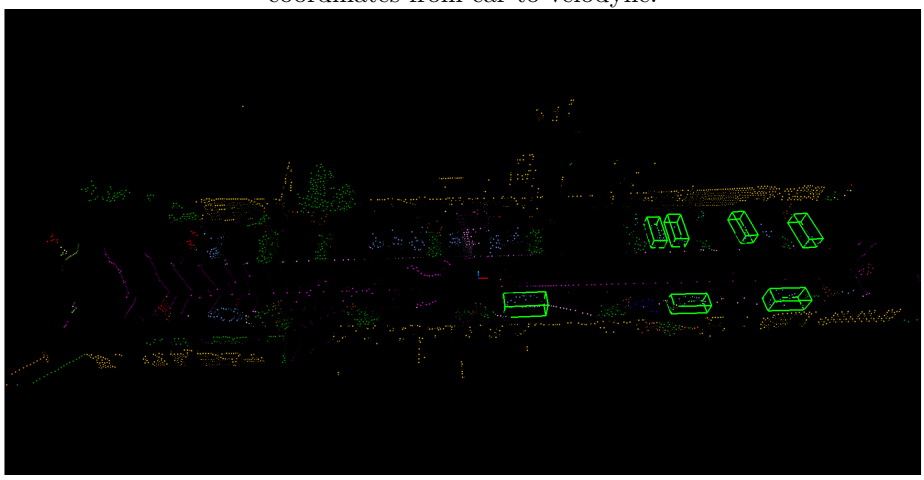
Now, we had to project the 3D bounding boxes of the visible cars onto the camera 2 image. In order to do so for each car we created an array containing the 8 vertices of the bounding box in the car's coordinate system, using the height, width and length of the car acquired from the data. In particular, by considering the center of the car as the center of the reference frame and the width direction as the x axis, the height direction as the y axis and the length direction as the z axis the 8 vertices were obtained as follow:

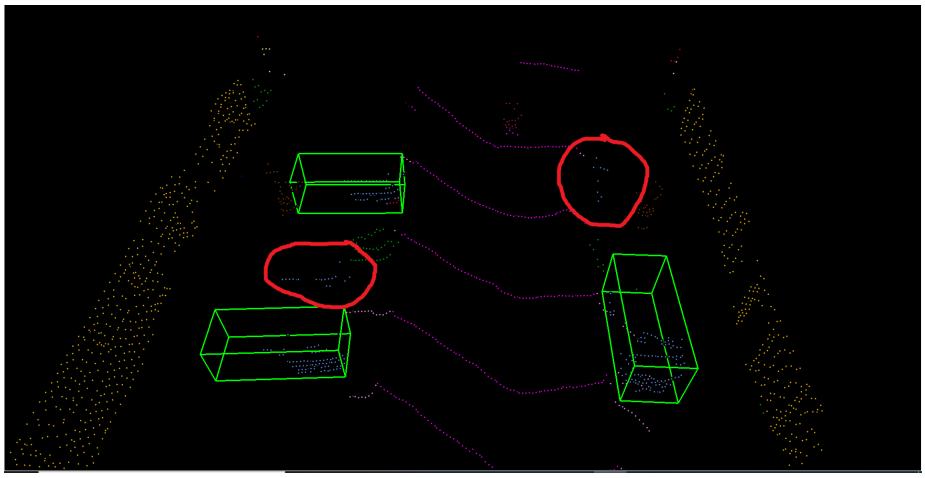
vertices = [-w_car/2, 0, l_car/2 w_car/2, 0, l_car/2
w_car/2, 0, -l_car/2 -w_car/2, 0, -l_car/2
-w_car/2, -h_car, l_car/2 w_car/2, -h_car, l_car/2
w_car/2, -h_car, -l_car/2 -w_car/2, -h_car, -l_car/2]

Then, we created the rotation matrix from car's coordinate system to Cam 0's one using the rotation angle around Y axis, and the translation vector using the bounding box's center coordinates. From them, we obtained the transformation matrix from car to Cam 0. Now we got the transformation matrix from cam 0 to cam 2, using the ones we had from velodyne to both cameras. Multiplying it with the one found previously, we got the transformation matrix from car to cam 2. We then projected the bounding box's corners to the cam 2 image plane as we did in part 1, and finally drew the lines between them. We could now display the bounding boxes on the image created in part 1.



In the last part of the problem we had to display the 3D point cloud, with semantic colors and bounding boxes, modifying the function 3dvis. Firstly we modified the corner order in self.connect, then we changed the function update. We initialized a 1D array and filled it with the colors related to the semantic labels, as in part 1. Then we computed the bounding box's corners as in part 2, and the transformation matrix from car to velodyne using the ones from car to cam 0 and to velodyne. Finally we changed the corners' coordinates from car to velodyne.





From the last image we can see that our network failed to identify two vehicles, even if some clusters of point were captured from the velodyne.

3 Problem 3

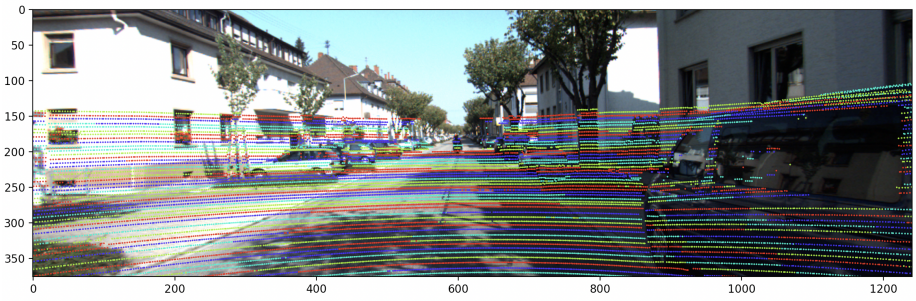
In this third task, as our first step, we decided to recalculate the angular vertical resolution of the Lidar sensor, instead of directly taking the one provided in the datasheet, to achieve a better accuracy. In order to do this, we computed the angle along the y axis of the velodyne reference frame for each point captured in the point cloud:

$$angle = arctan\left(\frac{z}{x^2 + y^2}\right) \tag{1}$$

Where x, y, and z are the coordinates of the point in the velodyne reference frame. The resolution could then be simply computed by subtracting the maximum and minimum angles and dividing by 64 (the number of lasers' IDs).

$$resolution = \frac{max\_angle - min\_angle}{64} \tag{2}$$

At this point, we divided the points by starting from the maximum angle and considering the computed resolution iterating for 64 times and associating the same colour to the points belonging to the same range (as coming from the same laser Id) after having projected them to the image plane.
Here the result:



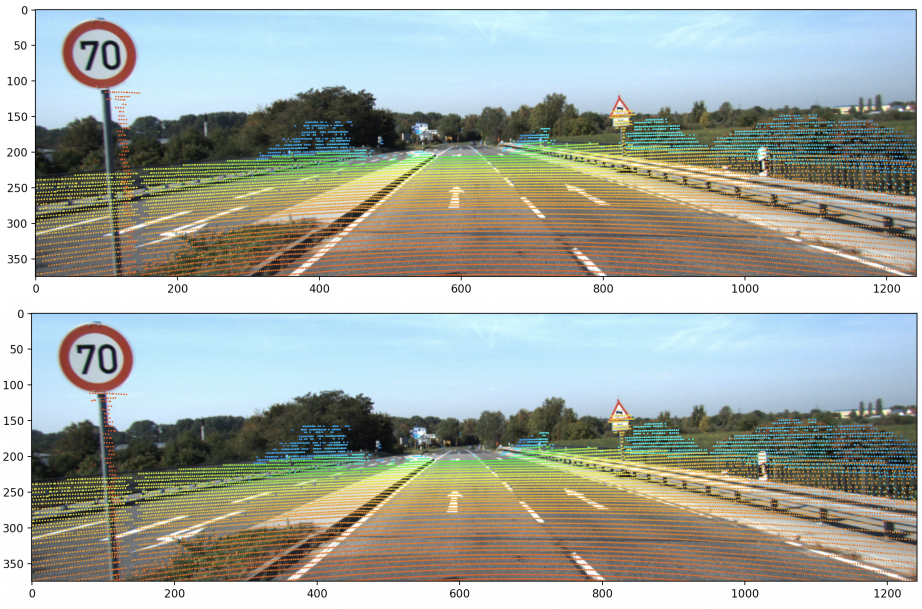
4 Problem 4

For the fourth and last task we started by assembling the homogeneous transformation matrices with the provided rotation matrices and displacements; we then computed the angle at which the image was taken by camera 2, considering the rotation around the z axis of the velodyne reference frame and the x axis as $angle = 0$ and we converted it to the range [-π, π]. We then computed the angle of each point with respect to the same reference frame as before, in order to understand if it was taken by the Lidar before or after the snapshot of the camera, by checking whether the point has an angle smaller or bigger than the image angle; in the first case the point needs to be translated forward while in the second case the point needs to be translated backwards. Thus, by converting the angle in a time measure (a delta in time) and multiplying it by the translational velocity we can compute the translational displacement which needs to be applied to the point coordinates. For the rotational part we applied the same reasoning but starting from the new points obtained by removing the translational motion distortion, thus by correcting the angles we had computed before for each point (always multiplying the delta this time with the rotational velocity) and applying this correction to the translated points. Then we exploited cylindrical coordinates to apply the correction, thus the new coordinates of the points are:

$$points = [\ \rho cos(corrected\_angle), \quad \rho sin(corrected\_angle), \quad z\ ] \tag{3}$$

Where ρ is computed from the translated points coordinates and $corrected\_angle$ is as previously specified.

Here the original image and the corrected one:



5 Problem 5

1) The Lidar consists of several lasers which emit beams with an high power density, consequently the closer the human eye gets to the source the smaller the eye section hit by the laser beam is and the whole power focuses on a tiny area increasing the riskiness. On the other hand if the distance from the source is relevant the laser beam diverges, hitting a bigger surface but with a lower power density thus reducing the dangerousness. In particular the "beam spread" is an intrinsic feature of each laser, the higher it is the safer that particular laser is for the human eye.

2) For what concerns the camera, the main issue is due to the fact that some objects may be reflected on the road, fooling the object recognition software. As a result there can be an effect of doubling or of wrong perceived distance. For the Lidar, instead, we occur into a reduction of the sensor range on the road surface because of the reflection of the laser beams; we don't have a diffuse, incident light which goes back to the sensor as planned but instead a specular reflection and glaring are experienced and the beams going back to the source is not guaranteed anymore.

3)The problem caused by the distance between the 2 sensors is mainly a loss of information. In fact as the distance rises the Lidar will detect more points which are not seen by the camera and won't detect some points relevant for it, thus in the case of the Lidar being more on the left, the content from it projected on the right of the camera image will be less accurate because some

meaningful points may not be reached properly by the sensor resulting in an asimmetry of the points projected on the image. We can conclude that as the distance between the two sensor increases also the severeness of the issue is more relevant. Taking also into account the motion correction implemented in exercise 4, having the Lidar translated sideways with respect to the camera would result in a loss of the synchronization between the 2 sensors and thus recovering the timestamp of the 3D points with respect to the projected ones on the camera would become more difficult.