DEEP LEARNING FOR AUTONOMOUS DRIVING

Understanding Multimodal Driving Data

TEAM 19: Argenziano Italo Nicholas, Pagani Leonardo

March 27, 2022

1 Problem 1. Bird's eye view

To retrieve the BEV view of the proposed image with the suggested resolution we implemented the following steps:

- The maximum (x_{max}, y_{max}) and the minimum(x_{min}, y_{min}) of x,y coordinates in the point cloud were computed (points were given in the velodyne reference frame).
- The dimensions $\dim_x$ and $\dim_y$ of a new grayscale image were computed according to the proposed resolution of 0.2m. In formulas:

  $\dim_x = \lceil (x_{max} - x_{min})/0.2 \rceil$

  $\dim_y = \lceil (y_{max} - y_{min})/0.2 \rceil$
- The position x_{new} and y_{new} of each point in the new matrix was computed cycling through the point cloud. In formulas, if (x,y,z) is a general point belonging to the point cloud, its position on the BEV map is:

  $x_{new} = \lfloor (x - x_{min})/0.2 \rfloor$

  $y_{new} = \lfloor (y - y_{min})/0.2 \rfloor$
- This procedure has allowed us to map each point in the point cloud to an element in the new $\dim_x$ x $\dim_y$ matrix. We have then filled each element x_{ij} of the matrix by computing the maximum intensity of all points mapped to x_{ij}.

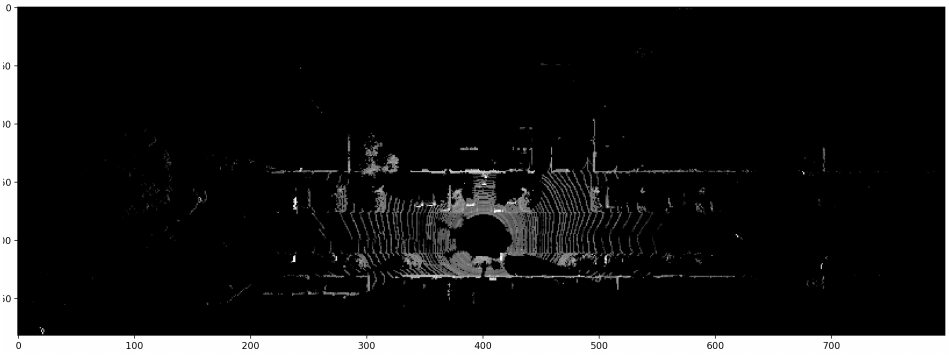

Figure 1: BEV view for the provided dataset

2 Problem 2. Visualization

To project the points on the image we have first expressed them in homogeneous coordinates. A generic point in the cloud now has the form (x_{velo},y_{velo},z_{velo},1) allowing us to perform the projection operation in a more convenient way. At first we have transformed the points in the CAM2 reference using the provided T^2_{velo} matrix.

$$\begin{bmatrix} x_{cam2} \\ y_{cam2} \\ z_{cam2} \\ 1 \end{bmatrix} = \begin{bmatrix} R^2_{velo} & t^2_{velo} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_{velo} \\ y_{velo} \\ z_{velo} \\ 1 \end{bmatrix} \tag{1}$$

Then we have filtered out the points having a negative z-coordinate. Those points are behind camera 2, hence they should not be projected on the image plane in order to avoid artifacts.
Finally, we have projected the points onto the image by using the intrinsic matrix K_{cam2}. This has allowed us to obtain the pixel coordinates of the projected 3D points. In formulas:

$$\begin{bmatrix} x_{image2} \\ y_{image2} \\ w_{image2} \end{bmatrix} = \begin{bmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_{cam2} \\ y_{cam2} \\ z_{cam2} \\ 1 \end{bmatrix} \tag{2}$$

It is important to normalize the pixel coordinates such that the third coordinate w_{image2} equals 1.
In the rest of the code we have recovered the the points' colour to achieve a correct printout.

The second part of the task consists in the retrieval and printout of the bounding boxes.
For each car in the *objects* dictionary we recovered the three dimensions h,w,l, the coordinates (expressed in the CAM-0 reference frame) of the center of the bottom face of the bounding box x_{cen}, y_{cen}, z_{cen} as well as the rotation angle r_y between the bounding boxes and y-axis of CAM-0 reference frame.

Given the dimensions of each bounding box, one can easily retrieve the position of the eight corners of the box relative to the center of the bottom face. Assuming that the center of the bottom face coincides with the origin of the camera reference frame and that $r_y = 0$, the coordinates of the corners expressed in the camera frame are:

bottom-left-back = [-$l/2$, 0, $w/2$]
bottom-right-back = [-$l/2$, 0, -$w/2$]
bottom-right-front = [$l/2$, 0, -$w/2$]
bottom-left-front = [$l/2$, 0, $w/2$]
top-left-back = [-$l/2$, -h, $w/2$]
top-right-back = [-$l/2$, -h, -$w/2$]
top-right-front = [$l/2$, -h, -$w/2$]
top-left-front = [$l/2$, -h, $w/2$]

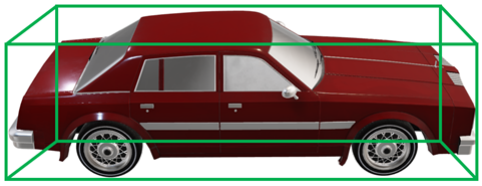

Figure 2: Bounding box around a target car at r_y=0

Let a corner be identified by the coordinates (*x'*,*y'*,*z'*), it is possible to take the rotation r_y into account by applying some trigonometric reasoning:

$x_{cam0} = x' \cos(r_y) + z' \sin(r_y) + x_{cen}$
$y_{cam0} = y' + y_{cen}$
$z_{cam0} = z' \cos(r_y)$–$x' \sin(r_y) + z_{cen}$

Finally, to project the corners onto the image, we can apply a similar reasoning as before. The idea is to perform multiple homogeneous transformations to bring the coordinates from the CAM-0 frame to the CAM-2 frame, via the velodyne frame. Then the usual projection using the intrinsic matrix is performed. In formulas:

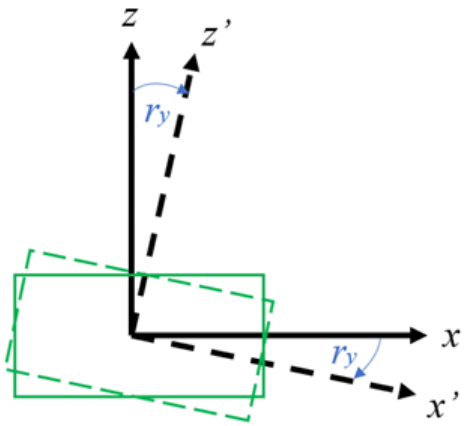

Figure 3: Bounding box rotation

$$\begin{bmatrix} x_{image2} \\ y_{image2} \\ w_{image2} \end{bmatrix} = \begin{bmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R^2_{velo} & t^2_{velo} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R^{velo}_0 & t^{velo}_0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_{cam0} \\ y_{cam0} \\ z_{cam0} \\ 1 \end{bmatrix} \tag{3}$$

Again, it is crucial to normalize the pixel coordinates such that the third homogeneous coordinate w_{image2} equals 1.
The lines joining the eight corners are then drawn by using the *Polygon* function from the *matplotlib* library.

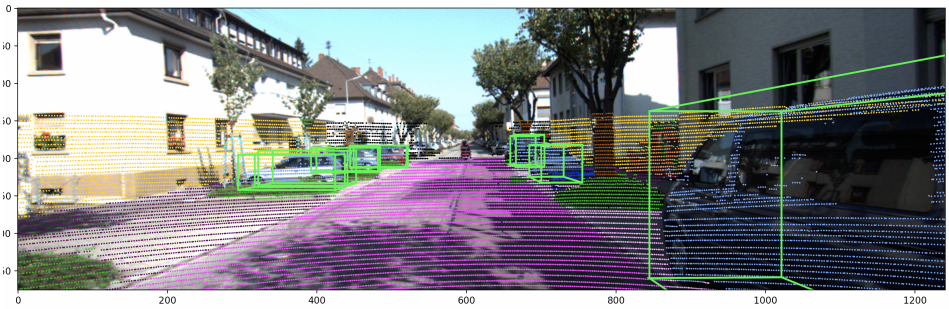

Figure 4: Visualization of the point cloud projected onto image2. Point colors are assigned according to their semantic label. Bounding boxes are in green.

In the third request of the task, we seek to achieve a 3D-visualization of the point cloud. To color each point and draw the bounding boxes around the cars,

we can basically use the same code. We notice that **the built-in network fails to identify two of the cars that lie within the camera FOV**.

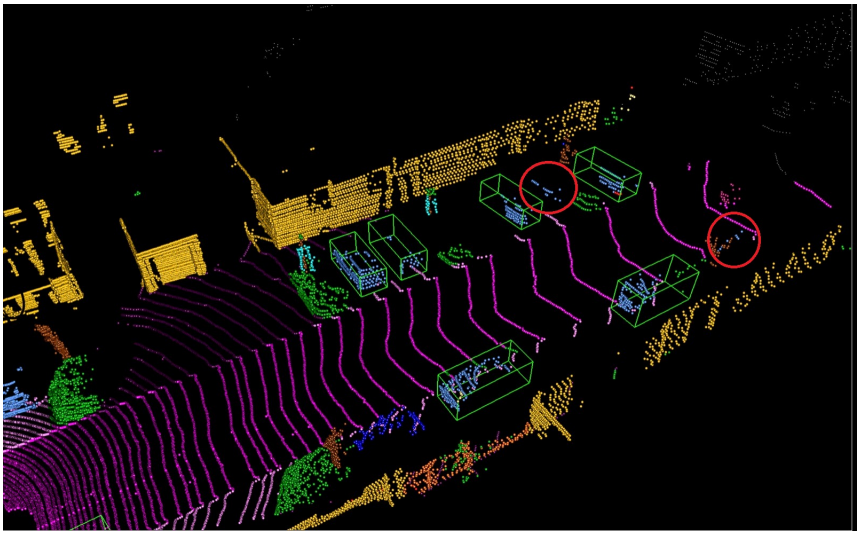

Figure 5: 3D visualization of the point cloud, restricted to the camera FOV. The two clusters of cyan points circled in red represent cars missed by the network.

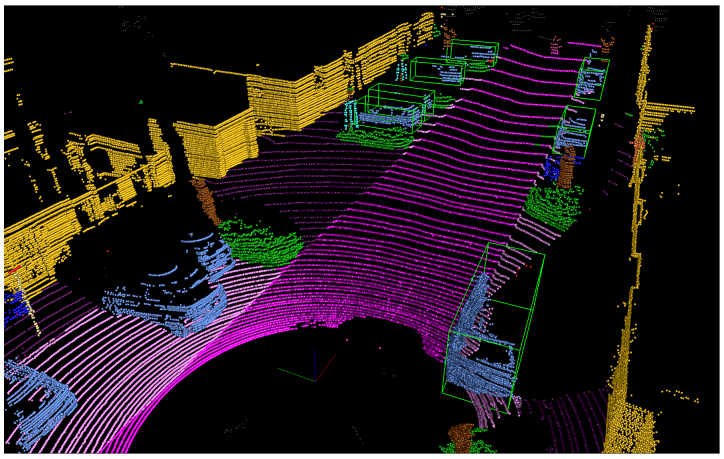

Figure 6: Full visualization of the point cloud.

3 Problem 3. Laser ID

In order to identify all different laser beams, we adopted a reasoning based on the lidar's vertical field of view. We computed the elevation angle of each point of the point cloud (excluding those behind the camera) based on their z-coordinate and their distance on the xy-plane from the velodyne (with all coordinates expressed in the velodyne reference frame).
Given a generic point of coordinates (x_p, y_p, z_p), the elevation was computed as:

$$\epsilon_p = \arctan\left(\frac{z_p}{\sqrt{x_p^2+y_p^2}}\right)$$

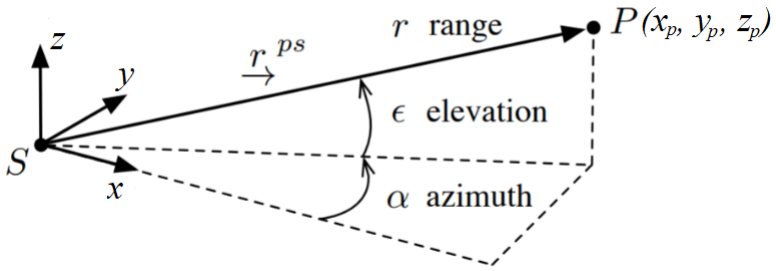

Figure 7: Angular coordinates.

Then, we computed the maximum and minimum elevation angle and discretized the whole interval in 64 bins. We computed the angular resolution $\Delta\epsilon$ as the ratio between the total span of the interval and the number of bins. Finally, we associated a laser id to each 3D point based on how many times its elevation angle contained the angular resolution. In formulas:

$$\Delta\epsilon = \frac{\epsilon_{\max}-\epsilon_{\min}}{64} \qquad ID_p = \left\lceil \frac{\epsilon_p - \epsilon_{min}}{\Delta\epsilon} \right\rceil$$

Below is the result, with the points projected onto the image plane of Camera 2 and the colors retreived using the provided *line_color* function.

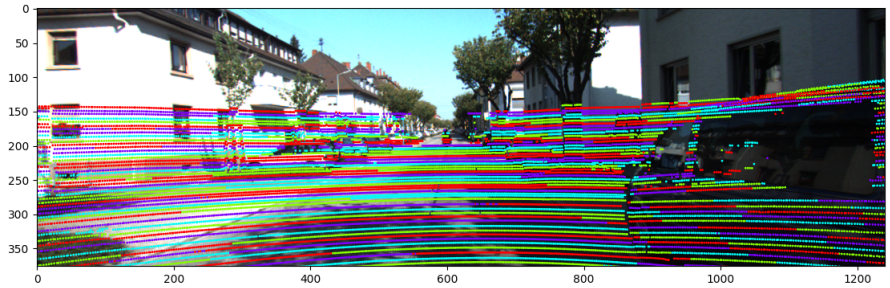

Figure 8: Points projected on image 2, colored according to their laser ID.

4 Problem 4. Remove Motion Distortion

In order to handle the translational motion correction, we developed the following physical model.
Let us define three reference frames $\mathcal{S}_0, \mathcal{S}_1, \mathcal{S}_2$, which represent the Velodyne coordinate system at three different instants: the time t_0 when the LiDAR starts scanning the current point cloud, the time t_1 when a certain point P is scanned, and the time t_2 when the LiDAR is facing forward and triggers the camera acquisition. The position of the selected point in each of the three frames is denoted by $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2$ respectively.
At the beginning of the scan, the Velodyne's scanning direction forms an angle ϕ_0 with the axis x_0, where $\phi_0 = \omega(t_0 - t_2)$. Here ω is the LiDAR's angular velocity. This angle ϕ_0 is negative since we are assuming that positive angles and angular velocities are defined clockwise (according to the Velodyne's scanning pattern) and the zero-angle condition occurs when the LiDAR is facing forward (i.e., aligned with the x axis). The angle ϕ_1 that the LiDAR forms at t_1 can be defined as $\phi_1 = -\arctan(y_1/x_1)$, where x_1 and y_1 are the point's coordinates as seen in the point cloud. The minus sign appears because this angle grows counterclockwise in the xy plane, as opposed to the aforementioned convention. The time interval $\Delta t = t_1 - t_0$ that the LiDAR takes to get to the selected point can be computed as $\Delta t = \frac{\phi_1 - \phi_0}{\omega}$. By performing some basic vector operations, one can note that:

$$\mathbf{P}_0 = \mathbf{P}_1 + \mathbf{v}\Delta t, \quad \mathbf{P}_0 = \mathbf{P}_2 + \mathbf{v}(t_2 - t_0)$$

Hence, the desired corrected position $\mathbf{P}_2$, referred to the instant when the image was taken, can be computed as:

$$\mathbf{P}_2 = \mathbf{P}_0 - \mathbf{v}(t_2 - t_0) = \mathbf{P}_1 + \mathbf{v}\Delta t - \mathbf{v}(t_2 - t_0) = \mathbf{P}_1 + \mathbf{v}\frac{(t_1 - t_2)}{\omega} = \mathbf{P}_1 + \mathbf{v}\frac{\phi_1}{\omega}$$

Similarly to the translational case, in order to implement the rotational motion correction, we defined three reference frames $\mathcal{S}_0, \mathcal{S}_1, \mathcal{S}_2$, representing the Velodyne coordinate systems at the instants t_0, t_1, and t_2 respectively. The angles ϕ_0 and ϕ_1, as well as the time difference Δt, are defined exactly as above. The quantity ϕ_L represents the angle travelled by the LiDAR scanning direction from the start of the scan to the time when it scans the selected point, and can be computed as $\phi_L = \Delta t(\omega_L - \omega_C)$. Here ω_L is the Velodyne's angular velocity and ω_C is the angular velocity deriving from the rotational motion of the vehicle (positive if counterclockwise, as opposed to ω_L). We also introduce the angle $\phi_C = -\omega_C \Delta t$, i.e. the angle that the car travels in the mentioned time interval Δt. By inspecting the geometry and physical definition of the aforementioned angles and the respective signs, one can note that:

$$\phi_0 = \phi_C + \phi_1 - \phi_L$$

which leads, by plugging in the definitions, to:

$$\Delta t = \frac{\phi_1 - \phi_0}{\omega_L}$$

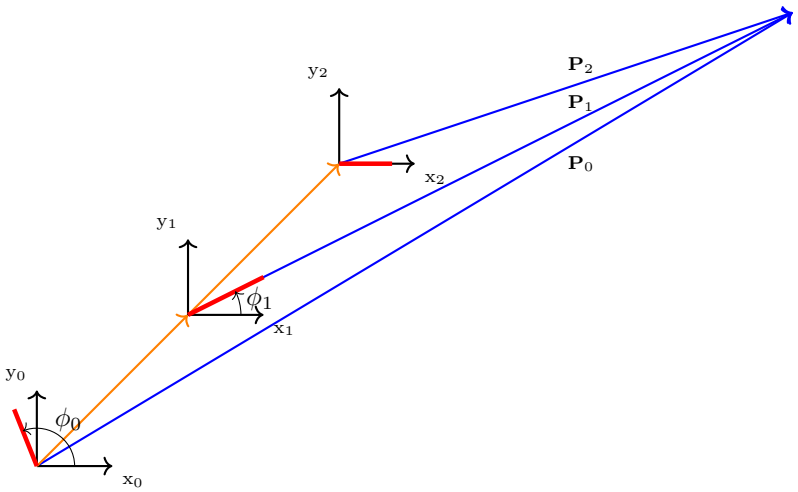

Figure 9: Geometric setup for translation correction. The red line indicates the LiDAR scanning direction, while the orange vector indicates the motion of the vehicle.

Consequently, ϕ_C can be calculated and used to build a first rotation matrix $\mathbf{R}_1^0$, which transforms the selected point from the frame to the frame $\mathcal{S}_1$ to $\mathcal{S}_0$. A second rotation matrix $\mathbf{R}_0^2$ can be used to obtained the desired point's coordinates referred to the instant when the photo was taken. This matrix is based on the angle $\phi_2 = \omega_C(t_0 - t_2)$ that the car travels from the start of the scan to the activation of the camera. In formulas:

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = \mathbf{R}_0^2(-\phi_2)\mathbf{R}_1^0(\phi_C) \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix}$$

Since both the linear and the angular velocities are measured by GPS/IMU sensors, we transformed all points in the GPS/IMU reference frame (using the provided rotation matrix and translation vector *"imu_to_velo"*), applied the correction there, and finally converted corrected points back to the velodyne frame, in order to project them onto the image plane as previously discussed.

Below are the results.

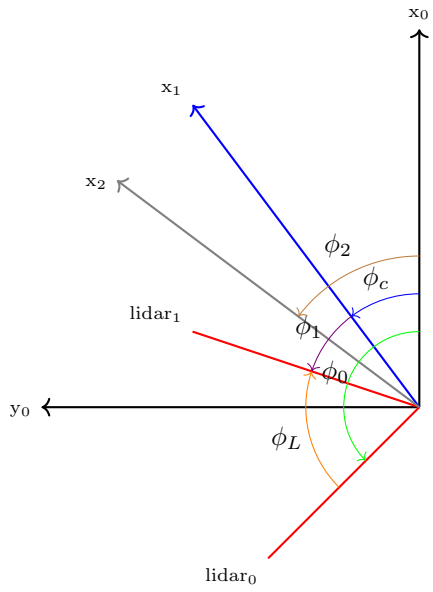

Figure 10: Geometric setup for rotation correction.

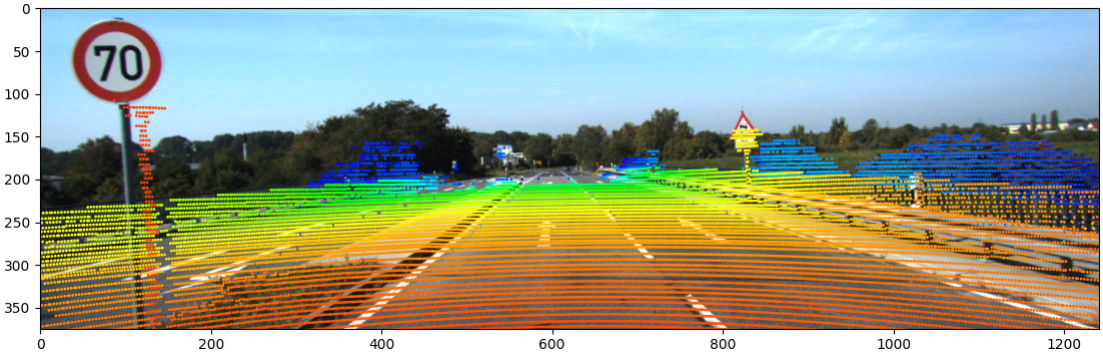

Figure 11: Projected point cloud before removing motion distortion

5 Problem 5. Questions

1. Laser impulses emitted by LiDARs are potentially dangerous for human eyes. Eye safety is a complex combination of factors that aren't just based on the wavelength of a laser. The safety rating of a LiDAR depends on the power, divergence angle, pulse duration, exposure direction, as well as the wavelength. In general, our retinas are more exposed to wavelengths if we happen to be close to the LiDAR. Without going into too much detail, which would require to find the solution of the wave equation for specific LiDAR sensors, it is reasonable to model the power P transmitted by the

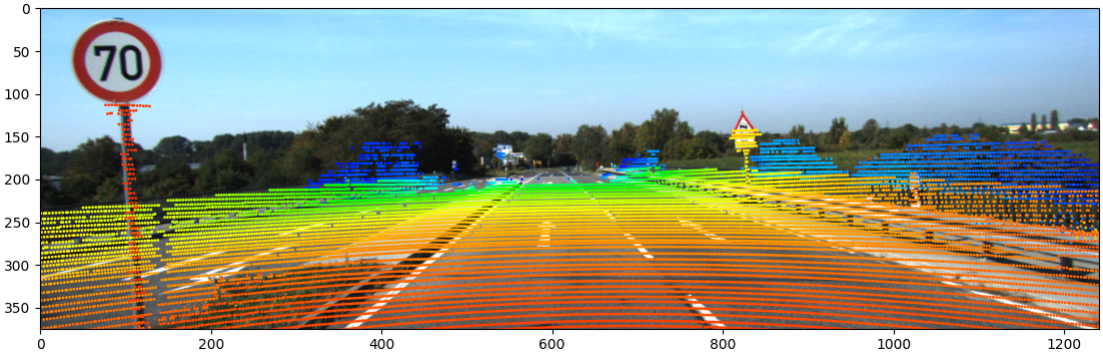

Figure 12: Projected point cloud after removing motion distortion

LiDAR by an equation the form:

$$P(r) \approx \frac{P_0}{r^2}$$

where r denotes the range of the impulse and P_0 the power of the emitted lasers. That means the power of the impulse generally decays quadratically as the waves travel away from the sensor.

2. On a wet road, the water acts as a specular, mirror-like surface that reflects light in a very different pattern than in dry conditions. This reflection creates challenges for both sensors, creating confusing reflections for the camera and reducing the range of the lidar sensor. On a dry road, the rough asphalt of the road will diffuse light, sending laser light bouncing in all directions. But on a wet road, the water turns the road into an imperfect specular surface, acting like a mirror reflecting a portion of the light.For a camera, the challenge comes from reflections that appear on the road that may create confusing mirror images of objects. In the image below, you can see the headlights of a car reflecting off the road surface. For a camera with object recognition software, this can potentially confuse the system into thinking that a second car may be present, or that the car may be closer than it actually is.For the lidar sensor, the adverse impact is that the range of the sensor is reduced on the surface of the road. A portion of the laser light emitted by the lidar sensor reflects off the water on the road and away from the sensor. This means that the sensor is less able to see the road surface at long ranges. That said, the range of the sensor is unaffected on all other objects.

3. Since the LiDAR and camera are not cocentered, the main challenge that could arise is the loss of important points belonging to the cloud as they are transformed to image coordinates. A very common and plausible scenario is that certain points are in the FOV of the LiDAR but not in the FOV of the camera and are therefore not projected onto the image plane. The

intersection of the two fields of views crearly increases as the sensors are closer, therefore the loss of image points is clearly a more serious issue as the distance between the camera and the LiDAR increases.