
PROJECT 1: UNDERSTANDING MULTIMODAL DRIVING DATA

Abhinav Aggarwal

Department of Computer Science
University of Zurich
aabhinav@student.ethz.ch
18-748-079

Ankush Panwar

Department of Computer Science
University of Zurich
apanwar@student.ethz.ch
19-763-051

March 24, 2020

1 Bird's Eye View

The Key task of this problem is to display the Bird Eye View (BEV) image of the given scene in **segmentation data.p** with pixel intensities corresponding to the point's respective reflectance values in Velodyne point cloud.

1.0.1 Driving Data Used

Key data points used for this task are:

- P_{velo} : 3D point cloud generated by Velodyne sensor

1.0.2 Methodology

Below is the stepwise methodology we used to create Bird Eye View (BEV) image:

- **Discretization of P_{velo} :** Firstly we discretize points P_{velo} by binning the complete range ($\max_val - \min_val$) with resolution of 0.2m, 0.2m and 0.3m in x, y and z coordinates respectively. Then all the points are allotted bins which is closer to them. So now instead of (x, y, z, r) (where r =reflectance intensity) point values in meters, we have discretized 3D points starting from $(1, 1, 1, r)$.
- **Grouping points:** All the discretized 3D points in one bin are grouped as single point with reflectance value r as maximum value of all the points in that bin.(one point in one bin)
- **Projection on 2D plane:** We took these discretized 3D points and projected them on xy plane. All points with same (x, y) values but different z value are projected as single point with reflectance value as maximum value of all these points. Then we converted these discretized 2D points to a 2D array with (x, y) discretized point location as index of the array and reflectance intensity as array values. Finally this 2D array of reflectance values has been plotted as an image to generate the BEV view.

1.0.3 Results

Output image generated using above mentioned approach can be found in Figure 1

2 Projection onto Image Plane

2.1 Semantic Segmentation: Displaying Semantic Labels

The key task of this problem is to project point cloud of **segmentation data.p** onto the image of Cam 2 and Color each point projected according to their respective label.

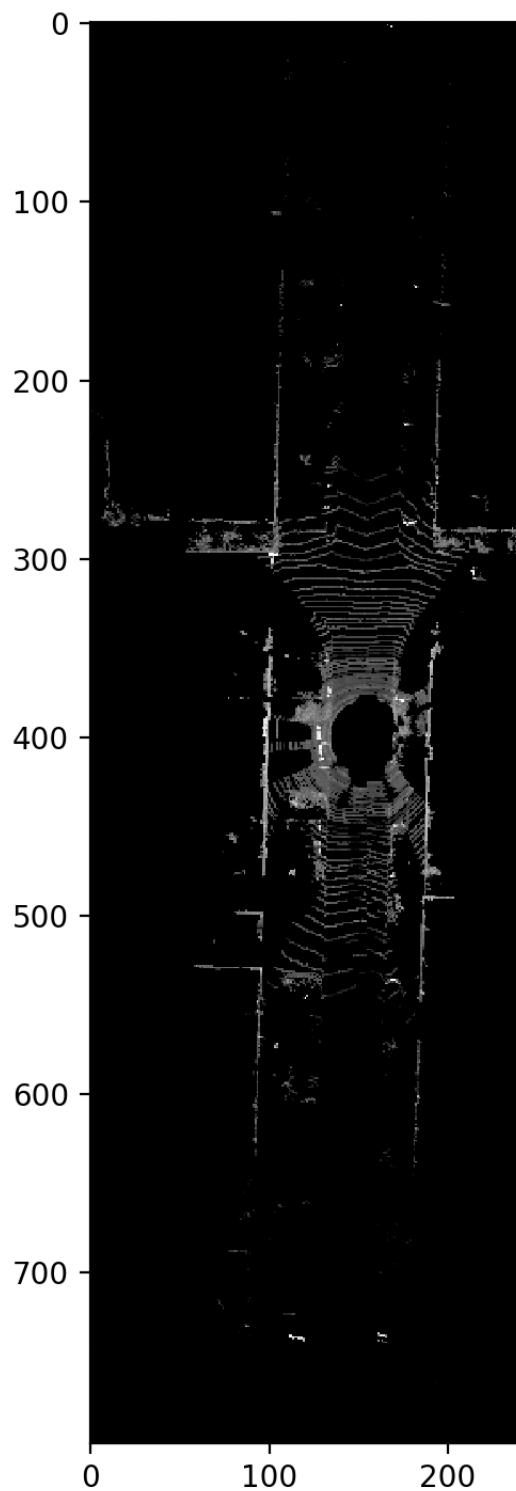


Figure 1: BEV Image

2.1.1 Driving Data Used

Key data points used for this task are:

- Lidar Points (P_{velo}): 3D point cloud generated by Velodyne scanner
- T_{cam2_velo} : Homogeneous velodyne to rectified camera 2 coordinate transformation matrix
- K_{cam2} : Camera intrinsics for cam 2

2.1.2 Methodology

Below is the stepwise methodology we used to do the semantic segmentation:

- First we convert given lidar points to homogeneous coordinates by concatenating 1 as fourth axis apart from x, y, z coordinate values
- Then these homogeneous coordinates are converted from velodyne coordinates to camera 2 coordinate frame by multiplying them with transition matrix T_{cam2_velo} i.e.

$$P_{cam2} = T_{cam2_velo} \times P_{velo} \quad (1)$$

- P_{cam2} with negative z values are filtered out as these points will not be visible in image plane
- P_{cam2} are then multiplied with camera 2 intrinsics i.e K_{cam2} and the result is renormalized to convert it into homogeneous coordinates i.e P_{img}

$$\lambda \times P_{img} = K_{cam2} \times P_{cam2} \quad (2)$$

- Finally all the points which are outside the field of view of image plane are filtered out and color coded to get segmentation mask.

2.1.3 Results

Output image generated using above mentioned approach can be seen in Figure 2.

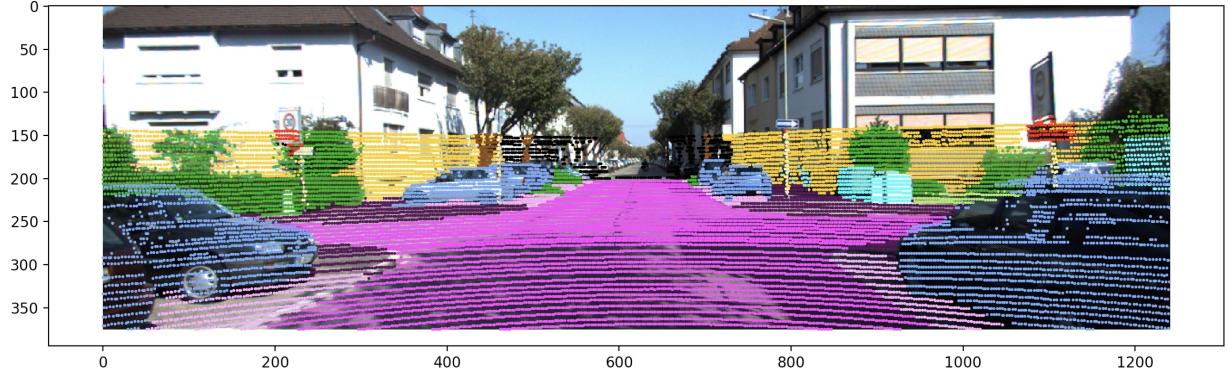


Figure 2: Point cloud projected onto image and colored according to semantic labels

2.2 Object Detection: Drawing 3D Bounding Boxes

The key task of this problem is to project the 3d bounding boxes of all given vehicles, cyclists and pedestrians within the scene of **detection data.p** to the Camera 2 image

2.2.1 Driving Data Used

Key data points used for this task are:

- P_{center} : 3D object location x, y, z in Camera 0 coordinates (in meters) describing the center of the bottom face of the bounding box

- Box_dims : Width, Height and Length (in meters) of bounding box
- ry : Rotation angle ry around Y-axis in Camera 0 coordinates $[-\pi, \pi]$
- P_{cam2_cam0} : Projection matrix of Camera 2 considering Camera 0 as world frame

2.2.2 Methodology

Below is the stepwise methodology we used to create bounding box around vehicles, cyclists and pedestrians:

- **Bounding box coordinates:** We compute the bounding box coordinates of vehicles, cyclists and pedestrians using P_{center} and Box_dims . We calculate bounding box coordinates using following equations:

$$P_{lower} = P_{center} + \text{Box_dims} \odot [i \times 0.5, 0, i \times 0.5] \quad (3)$$

$$P_{upper} = P_{center} + \text{Box_dims} \odot [i \times 0.5, -1, i \times 0.5] \quad (4)$$

where P_{lower} and P_{upper} are coordinates for bottom face and upper face respectively, $i = -1$ and $i = +1$ for backward and forward points respectively and \odot is elementwise product. Here we used -1 for upper face as y-axis is towards downward direction (towards road).

To illustrate above equations, suppose $P_{center} = [-0.65, 1.71, 46.70]$ (in order x, y, z) and $\text{Box_dims} = [1.67, 1.65, 3.64]$ (in order Width, Height and Length).

$$P_{lower}^f = [-0.65, 1.71, 46.70] + [1.67, 1.65, 3.64] \odot [0.5, 0, 0.5] \quad (5)$$

$$P_{upper}^b = [-0.65, 1.71, 46.70] + [1.67, 1.65, 3.64] \odot [-0.5, -1, 0.5] \quad (6)$$

- **Incorporating Rotation:** Rotation angle ry is used to incorporate rotation of bounding box around y-axis. First we transform coordinate frame from Camera 0 to P_{center} using simple translation matrix $T_{p0} = [I, -P_{center}]$. Then box coordinates are transformed in the translated coordinate frame using rotation matrix R_y around y-axis.

$$R_y = \begin{bmatrix} \cos ry' & 0 & \sin ry' \\ 0 & 1 & 0 \\ -\sin ry' & 0 & \cos ry' \end{bmatrix} \quad (7)$$

where $ry' = \frac{\pi}{2} + ry$.

We add $\frac{\pi}{2}$ to ry because ry is in $[-\pi, \pi]$ and we want it in $[-\frac{\pi}{2}, \frac{3\pi}{2}]$. Finally rotation transformed points are again translated back to Camera 0 coordinate space using $T_{0p} = [I, P_{center}]$

- **Projection to image plane:** Finally transformed box corner coordinates are projected in Camera 2 image plane using projection matrix P_{cam2_cam0} i.e.

$$\lambda \times P_{img} = P_{cam2_cam0} \times P_{cam0}^t \quad (8)$$

where P_{cam0}^t is the transformed points using above mentioned steps.

2.2.3 Results

Output image generated using above mentioned approach can be seen in Figure 3.

3 ID Laser ID

The Key task of this problem is to identify the laser ID of each 3D velodyne point in **segmentation data.p** and project them to Cam2.

3.0.1 Driving Data Used

Key data points used for this task are:

- P_{velo} : 3D point cloud generated by Velodyne sensor
- T_{cam2_velo} : Homogeneous velodyne to rectified camera 2 coordinate transformation matrix
- K_{cam2} : Camera intrinsic matrix for Cam 2



Figure 3: 3D bounding box projected onto the image.

3.0.2 Methodology

Below is the stepwise methodology we used to identify laser id from the 3D points and project them on image 2:

- Firstly we measured the angle (α_i) formed by the vector (3D point say i) onto velodyne's x-y plane using below mentioned formula for each 3D point.

$$\alpha_i = \arctan \frac{z_i}{\sqrt{(x_i^2 + y_i^2)}} \quad (9)$$

- Considering uniform distribution of laser range among (α), we labeled all 3D velodyne points with laser ID $laser_id_i$ with numbers 1 – 64 using below equation:

$$laser_id_i = \lfloor \frac{\alpha_i - \alpha_{min}}{\alpha_{max} - \alpha_{min}} \times 63 \rfloor + 1 \quad (10)$$

- We convert given 3D lidar points (with known $laser_id_i$) to homogeneous coordinates by concatenating 1 as fourth axis apart from x, y, z coordinate values
- Then these homogeneous coordinates are converted from velodyne coordinates to camera 2 coordinate frame by multiplying them with transition matrix T_{cam2_velo} i.e.

$$P_{cam2} = T_{cam2_velo} \times P_{velo} \quad (11)$$

- P_{cam2} with negative z values are filtered out as these points will not be visible in image plane
- P_{cam2} are then multiplied with camera 2 intrinsics i.e K_{cam2} and the result is renormalized to convert it into homogeneous coordinates i.e P_{img} (with known $laser_id_i$)

$$\lambda \times P_{img} = K_{cam2} \times P_{cam2} \quad (12)$$

- Finally all the points which are outside the field of view of image plane are filtered out and valid points are plotted on $image_2$ with 4 alternate colors to indicate the identified IDs.

3.0.3 Results

Output image generated using above mentioned approach can be found in Figure 4

4 Remove Motion Distortion

The key task of this problem is to remove distortions in acquired data from a fast moving car while the LiDAR rotates.

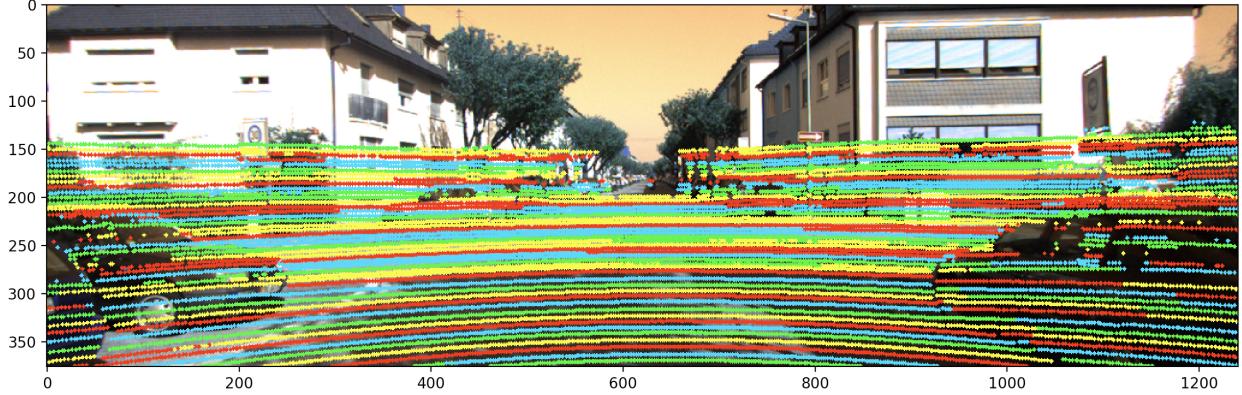


Figure 4: Identified Laser IDs.

4.0.1 Driving Data Used

Key data points used to correct distortions for each frame are:

- P_{velo} : Point cloud generated by velodyne LiDAR for each frame
- v : Instantaneous vehicle velocity as recorded by GPS/IMU for each frame
- ω_z : Instantaneous angular velocity around z-axis as recorded by GPS/IMU for each frame. Since for GPS and LiDAR z-axis points towards same direction, hence we can say that its angular velocity around z-axis of LiDAR
- T_{cam0_velo} : Transformation matrix to convert velodyne coordinates to Camera 0 coordinates
- P_{cam2_cam0} : Projection matrix to convert velodyne coordinates to Camera 2 coordinates

4.0.2 Methodology

Below is the stepwise methodology we used to remove distortions from LiDAR point cloud

- **Determining LiDAR start angle relative to front view:** We compute LiDAR start angle relative to front view using lidar_start, lidar_end and camera_triggered timestamps mentioned in velodyne_points folder for problem4 data. Angle θ can be computed using following equation:

$$\theta = \frac{t_{start} - t_{cam}}{t_{end} - t_{start}} \times 360^\circ \quad (13)$$

where t_{start} , t_{cam} , t_{end} are LiDAR start time, camera trigger time and LiDAR end time respectively. We computed LiDAR start angle θ using Eq. 13 for initial 6 frames and all results are mentioned in following table

Frame	Start_time	End_time	Camera_time	Angle
0000000001	48291.691564602	48291.795132685	48291.743348643	-180
0000000002	48291.795132685	48291.898679074	48291.846905164	-180
0000000003	48291.898679074	48292.002266726	48291.9504729	-180
0000000004	48292.002266726	48292.105837213	48292.054051969	-180
0000000005	48292.105837213	48292.209421015	48292.157628397	-180
0000000006	48292.209421015	48292.313032132	48292.261226573	-180

Table 1: Results for angle computation study for first 6 frames

As seen from this table LiDAR start point is exactly opposite to from view (camera view) which mean that LiDAR as to complete half a circle when camera is triggered to click an image.

- **Determining order of points in point cloud:** Point order is directly related to LiDAR start angle θ computed in previous part. Since there is no significant movement along z axis of lidar frame, hence we can directly relate order of points in point cloud to by angle α_i between x-axis and y-axis. Now, since x-axis is pointing

towards front view and LiDAR rotation starts from -180° , therefore $\alpha_i \in [-\pi, \pi]$ and points will lower α value would have been taken before ones with higher value. Angle α_i is computed using following equation:

$$\alpha_i = \arctan\left(\frac{y_i}{x_i}\right) \quad (14)$$

where x_i and y_i are x and y coordinate values of points i in LiDAR coordinate frame. Please note that α will be different for each point.

- **Undistorting point cloud:** Finally, to undistort the points in point cloud, we need to understand that lidar coordinate frame is not stationary and is having both rotation and translation motion due to movement of car. So, points can be undistorted if we can somehow find their coordinate values in the camera frame (in which image has been taken).

To achieve this, we have to first find the coordinate values in LiDAR frame when the laser is in forward direction (camera view/ front view). Hence we have to find transition matrix between frame when point is taken and when laser is point forward. Now since there is no movement assumed towards z-axis, hence we can find rotation matrix R'_i and translation vector t'_i for point i without considering any movement along z-axis

$$R'_i = \begin{bmatrix} \cos \beta_i & -\sin \beta_i \\ \sin \beta_i & \cos \beta_i \end{bmatrix} \quad (15)$$

$$t'_i = \begin{bmatrix} t_x^i \\ t_y^i \end{bmatrix} \quad (16)$$

where β_i , t_x^i and t_y^i is the angle rotated, translation along x-axis and translation along y-axis respectively by car between the time of capture of point i and LiDAR laser pointing towards forward direction. Here β_i , t_x^i and t_y^i are computed in following manner:

$$\beta_i = -0.1\omega_z \times \frac{\alpha_i}{2\pi} \quad (17)$$

$$t_x^i = v_x \times \frac{\alpha_i}{2\pi} \quad (18)$$

$$t_y^i = v_y \times \frac{\alpha_i}{2\pi} \quad (19)$$

where $\frac{\alpha_i}{2\pi}$ gives the time taken by LiDAR from point i to front view and multiplying them with respective velocities give the required rotation and translation motion of car. Please note that 0.1 is multiplied in Eq.17 before velodyne sensor is rotating at frequency of 10 Hz and negative sign are there to incorporate the fact that LiDAR is running in clockwise direction. Finally all the points P_{init}^i in homogeneous coordinates will be multiplied with their respective transition matrix to get undistorted points P_{undist}^i in velodyne frame.

$$P_{undist}^i = [R'_i \quad t'_i] P_{init}^i \quad (20)$$

Atlast P_{undist}^i is multiplied with transformation matrix T_{cam2_velo} to get undistorted points P_{final}^i in Camera 2 frame

$$P_{cam0}^i = T_{cam0_velo} \times P_{undist}^i \quad (21)$$

. Finally these points P_{cam0}^i are projected to Camera 2 frame and plotted to image plane using same methodology as mentioned in problem 2

$$P_{final}^i = P_{cam2_cam0} \times P_{cam0}^i \quad (22)$$

4.0.3 Results

Below are the output image (Fig.6, 7, 8) with undistorted point cloud for frames ‘0000000037’, ‘0000000310’ and ‘0000000320’. Here (a) Top image is the frame image, (b) Middle image shows how motion distortion looks like when projected to an image (c) Bottom image shows how it looks like when corrected

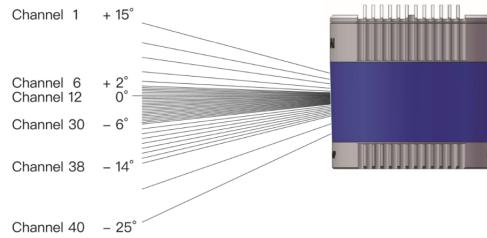


Figure 5: Multi-Channel LiDAR

5 Bonus Questions

1. Eye safety: it is not safe for the human eye to look at a spinning LiDAR when it is too close. Why the risk is higher when we are closer to the sensor?
 - As shown in Figure: 5, there are multiple laser channels in one LiDAR. Thus if human eye is very close to LiDAR, then multiple lasers can enter into the eye simultaneously and can do greater damage as compared to situation when it is farther distance from LiDAR. When eye is at farther distance from LiDAR then, fewer lasers can enter the eye simultaneously thus reducing the damage.
2. Wet road poses challenges for both cameras and LiDAR, what are the challenges and why?
 - Water turns the road into a specular, mirror-like surface that reflects light in a very different pattern than in dry conditions.
 - For a **camera**, the challenge comes from reflections that appear on the road that may create confusing mirror images of objects.
 - 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.
3. In this exercise, you have projected LiDAR points onto images. In the setup in Fig.1, the LiDAR sensor and the Cameras are non-cocentered – it can never be exactly non-cocentered. What problem this may cause for the data projection between the two sensors (LiDAR and Cam2 for instance)? Do you think this problem will be more severe or less severe when the two sensors are more distant from each other?
 - Since LiDAR sensor and Cameras are non-cocentered, hence the transformation matrix between these two sensors will keep on changing due to relative motion between Lidar frame and Camera Frame when car moves left or right. For example, if we take Camera0 and Velodyne sensor in the setup in Fig.1 and assume that they are exactly aligned along the forward direction of car, then also the transformation matrix between these two coordinate systems will keep on changing as the car turns right or left. This will yield inaccurate results when LiDAR points are projected to camera frame using a fixed matrix.
 - This problem will be more severe when the two sensors are more distant as this can lead to significant relative motion of Lidar and camera frames and ultimately changing transformation matrix during the left-right movement of car, thus more inaccurate projected points will come from one frame to other.

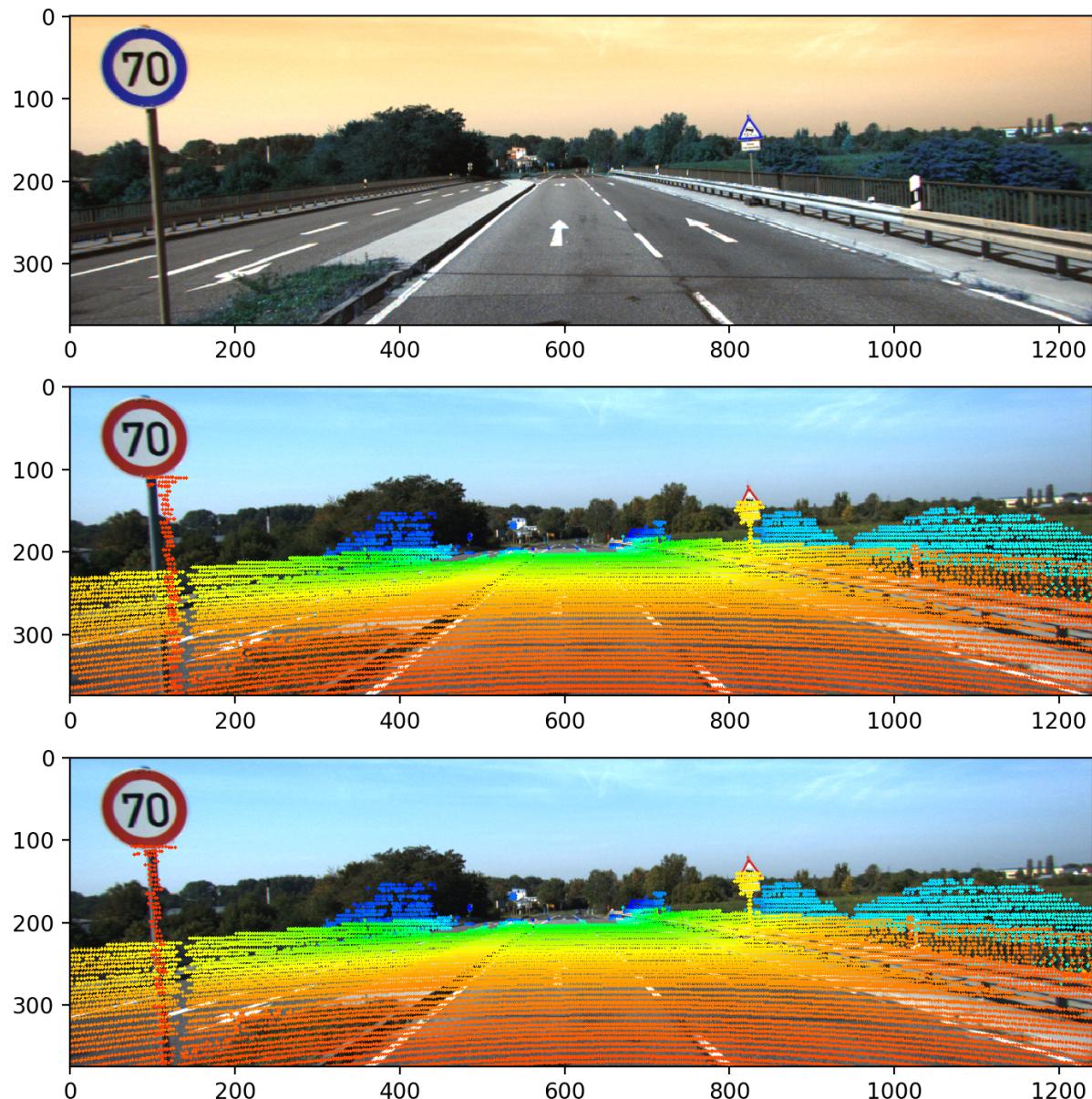


Figure 6: Frame 0000000037

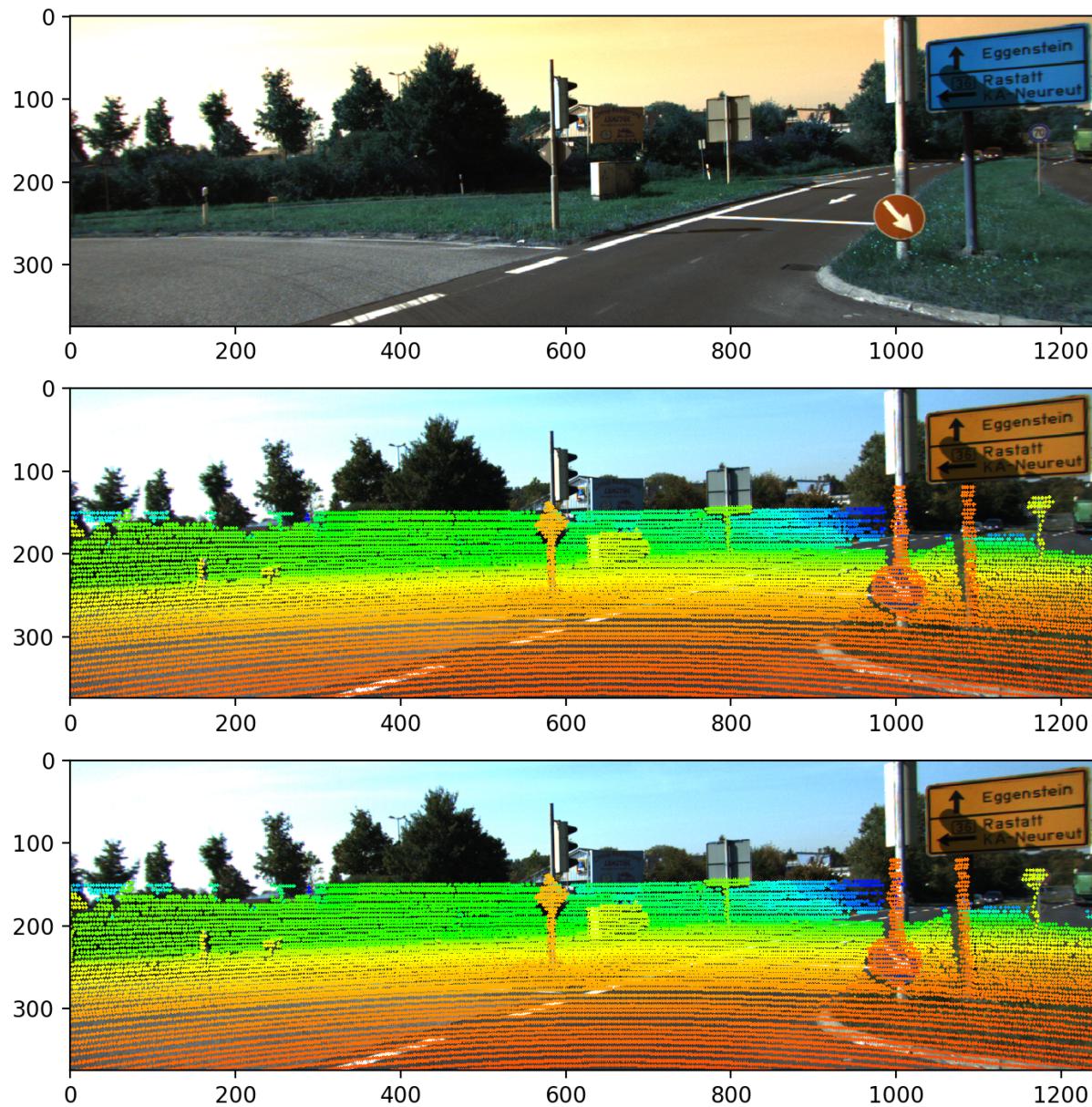


Figure 7: Frame 0000000310

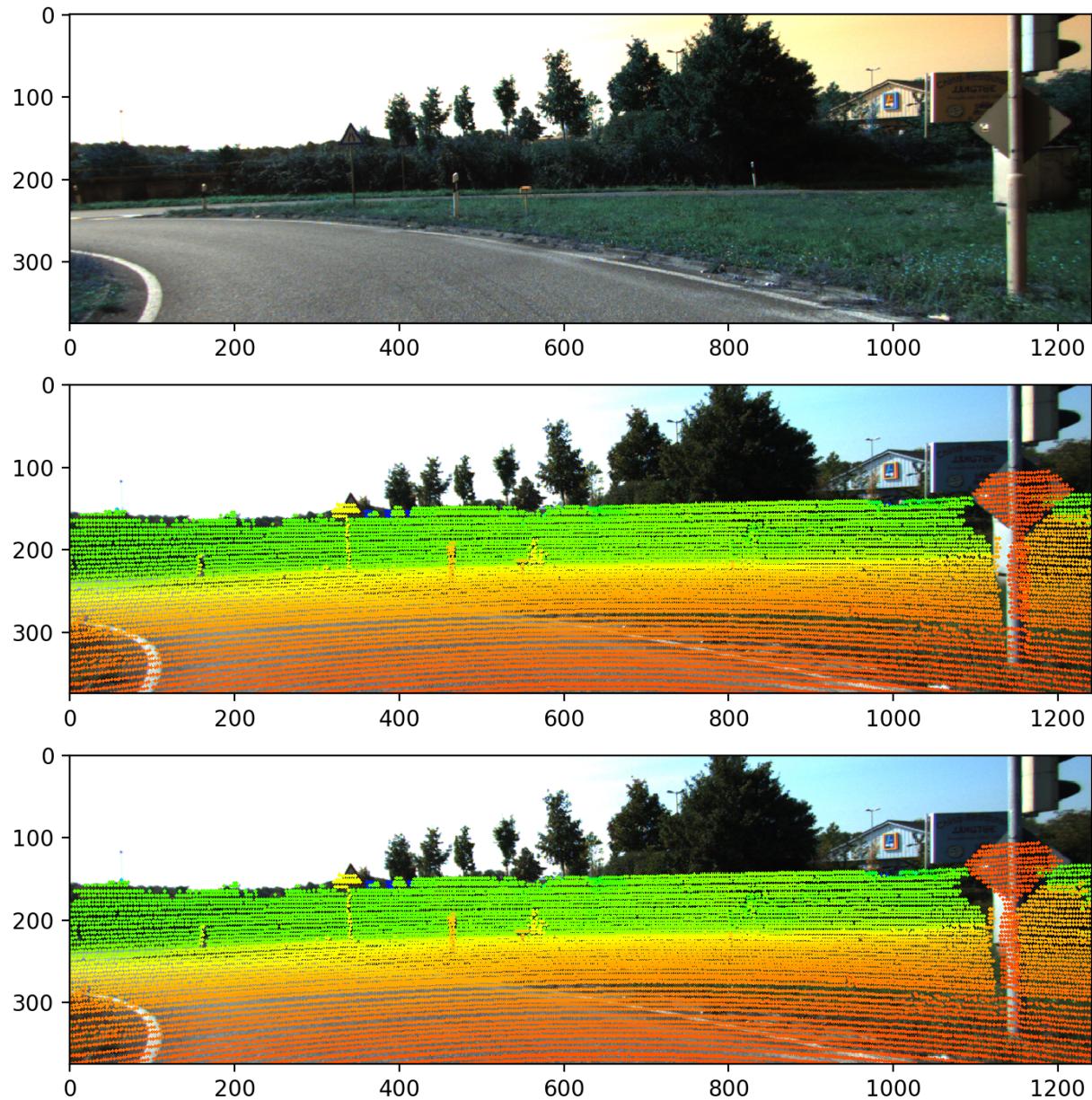


Figure 8: Frame 0000000320