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 19, 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 $(max\_val - min\_val)$ in intervals of 0.2m, 0.2m and 0.3m in x, y and z coordinates respectively. Then all the points all allotted bin point which is closer to them.
- **Grouping points:** Here all the points in one bin are grouped as single point with reflectance value as maximum value of all the points in that bin.
- **Projection on 2D plane:** Now since we want to get BEV of point cloud, so we have to project 3D points on 2D plane i.e. project all points on x-y plane. All the points with same x-y values but different z value are projected as single point with relectance value as maximum value for all these points. **[[write about coordinate system used]]**

1.0.3 Results

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

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.

2.1.1 Driving Data Used

Key data points used for this task are:

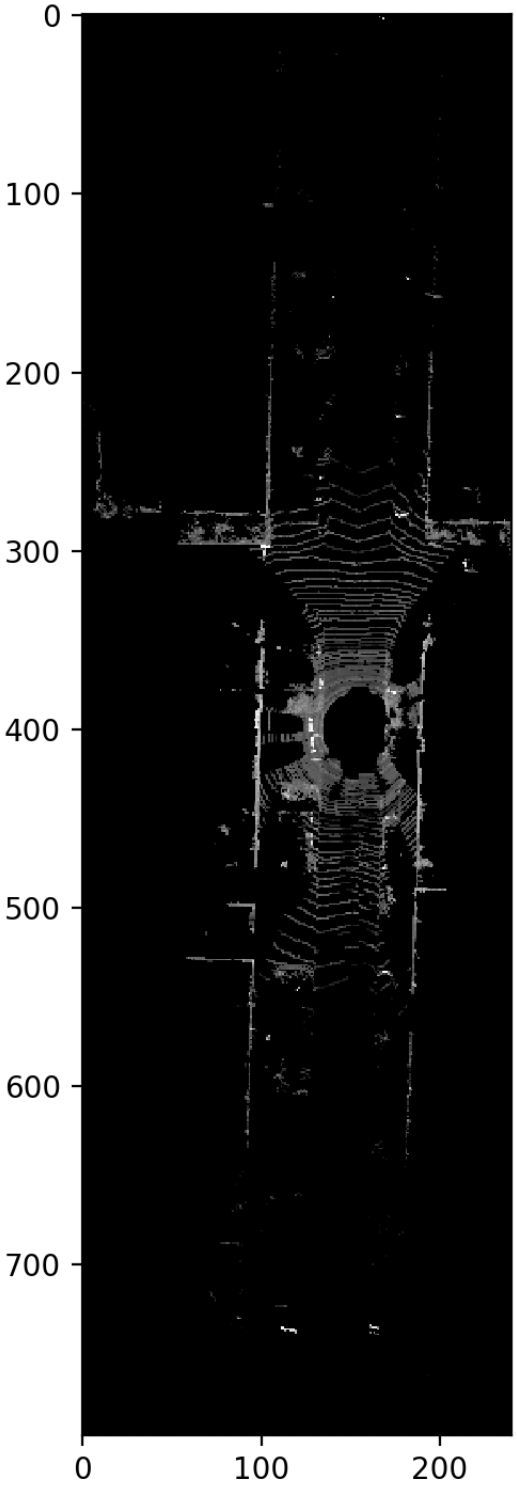

Figure 1: BEV Image

- 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.

$$P_{cam2} = T_{cam2\_velo} \times P_{velo} \tag{1}$$

- 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}

$$\lambda \times P_{img} = K_{cam2} \times P_{cam2} \tag{2}$$

- 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.

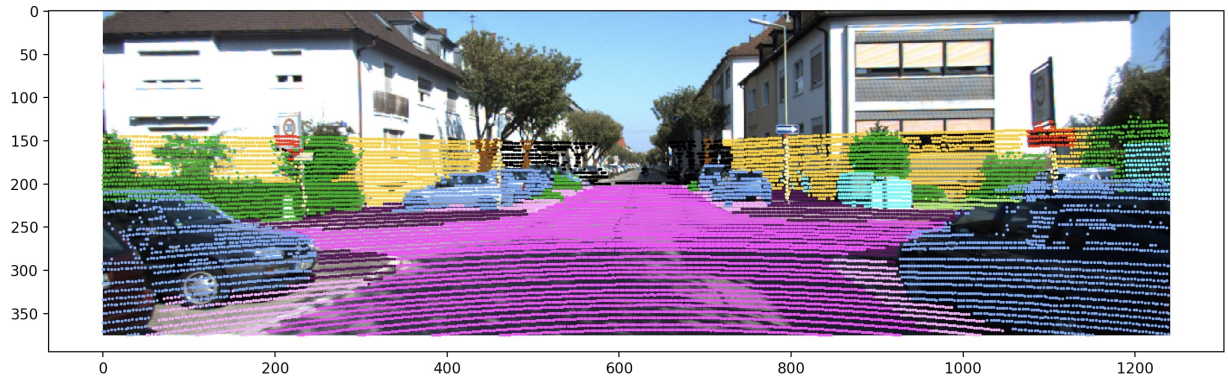

Figure 2: Point cloud projected onto image and colored according to semantic labels

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:

$$P_{lower} = P_{center} + Box\_dims \odot [i \times 0.5, 0, i \times 0.5] \tag{3}$$

$$P_{upper} = P_{center} + Box\_dims \odot [i \times 0.5, -1, i \times 0.5] \tag{4}$$

  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 Box_dims = [1.67, 1.65, 3.64] (in order Width, Height and Length).

$$P^f_{lower} = [-0.65, 1.71, 46.70] + [1.67, 1.65, 3.64] \odot [0.5, 0, 0.5] \tag{5}$$

$$P^b_{upper} = [-0.65, 1.71, 46.70] + [1.67, 1.65, 3.64] \odot [-0.5, -1, 0.5] \tag{6}$$

- **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.

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

  where $ry' = \frac{\pi}{2} + ry$.
  We add $\frac{\pi}{2}$ to ry because ry is in $[-\pi, \pi]$ and we want it in $[-\frac{\pi}{2}, \frac{3\pi}{2}]$ **[[add justification for this]]** 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.

$$\lambda \times P_{img} = P_{cam2\_cam0} \times P^t_{cam0} \tag{8}$$

  where P^t_{cam0} 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.

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

3 ID Laser ID

The Key task of this problem is to 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

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.

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

- 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:

$$laser\_id_i = \lfloor \frac{\alpha_i - \alpha_{min}}{\alpha_{max} - \alpha_{min}} \times 63 \rfloor + 1 \tag{10}$$

- 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.

$$P_{cam2} = T_{cam2\_velo} \times P_{velo} \tag{11}$$

- 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 $laser\_id_i$)

$$\lambda \times P_{img} = K_{cam2} \times P_{cam2} \tag{12}$$

- 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.

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_{cam2_velo}: Transformation matrix to convert velodyne coordinates to Camera 2 coordinates

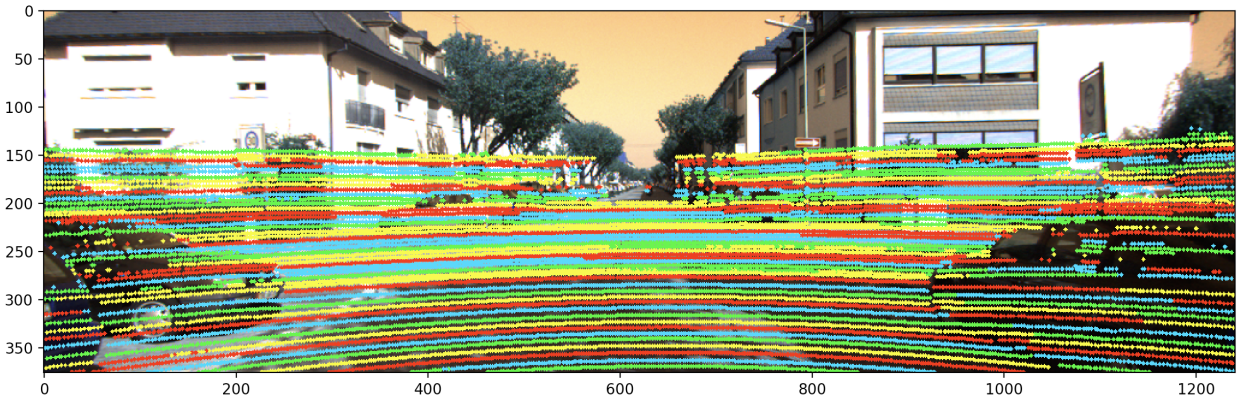

Figure 4: Identified Laser IDs.

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:

$$\theta = \frac{t_{start} - t_{cam}}{t_{end} - t_{start}} \times 360^o \tag{13}$$

  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^o, 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:

$$\alpha_i = \arctan(\frac{y_i}{x_i}) \tag{14}$$

  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

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

$$t'_i = \begin{bmatrix} t^i_x \\ t^i_y \end{bmatrix} \tag{16}$$

where β_i, t^i_x and t^i_y 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^i_x and t^i_y are computed in following manner:

$$\beta_i = -0.1\omega_z \times \frac{\alpha_i}{2\pi} \tag{17}$$

$$t^i_x = v_x \times \frac{\alpha_i}{2\pi} \tag{18}$$

$$t^i_y = v_y \times \frac{\alpha_i}{2\pi} \tag{19}$$

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^i_{init} in homogeneous coordinates will be multiplied with their respective transition matrix to get undistorted points P^i_{undist} in velodyne frame.

$$P^i_{undist} = [R'_i \quad t'_i]\, P^i_{init} \tag{20}$$

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

$$P^i_{final} = T_{cam2\_velo} \times P^i_{undist} \tag{21}$$

. Finally these points P^i_{final} are projected and plotted to image plane using same methodology as mentioned in problem 2

4.0.3 Results

Below is the output image 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

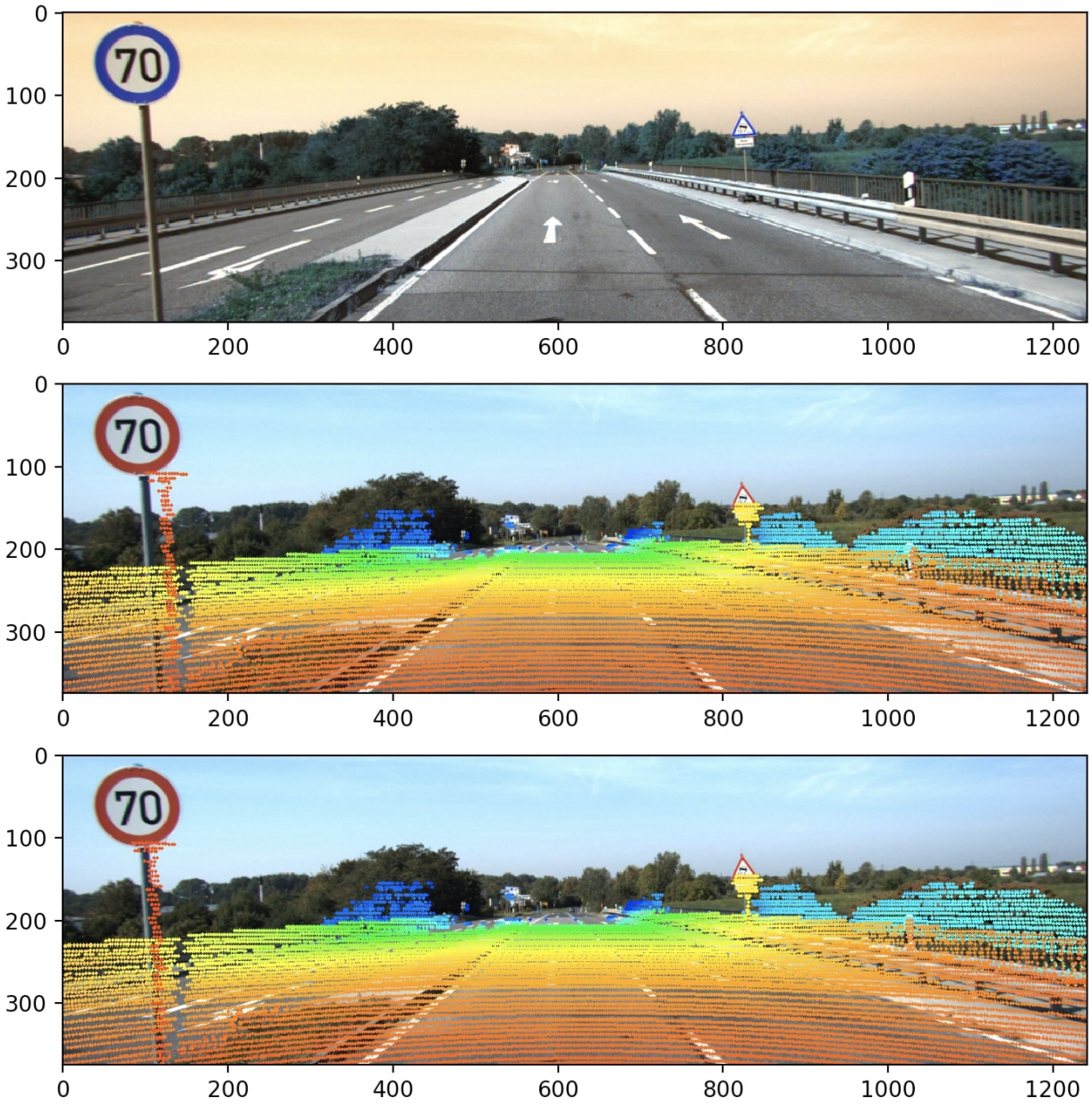

Figure 5: Frame 0000000037

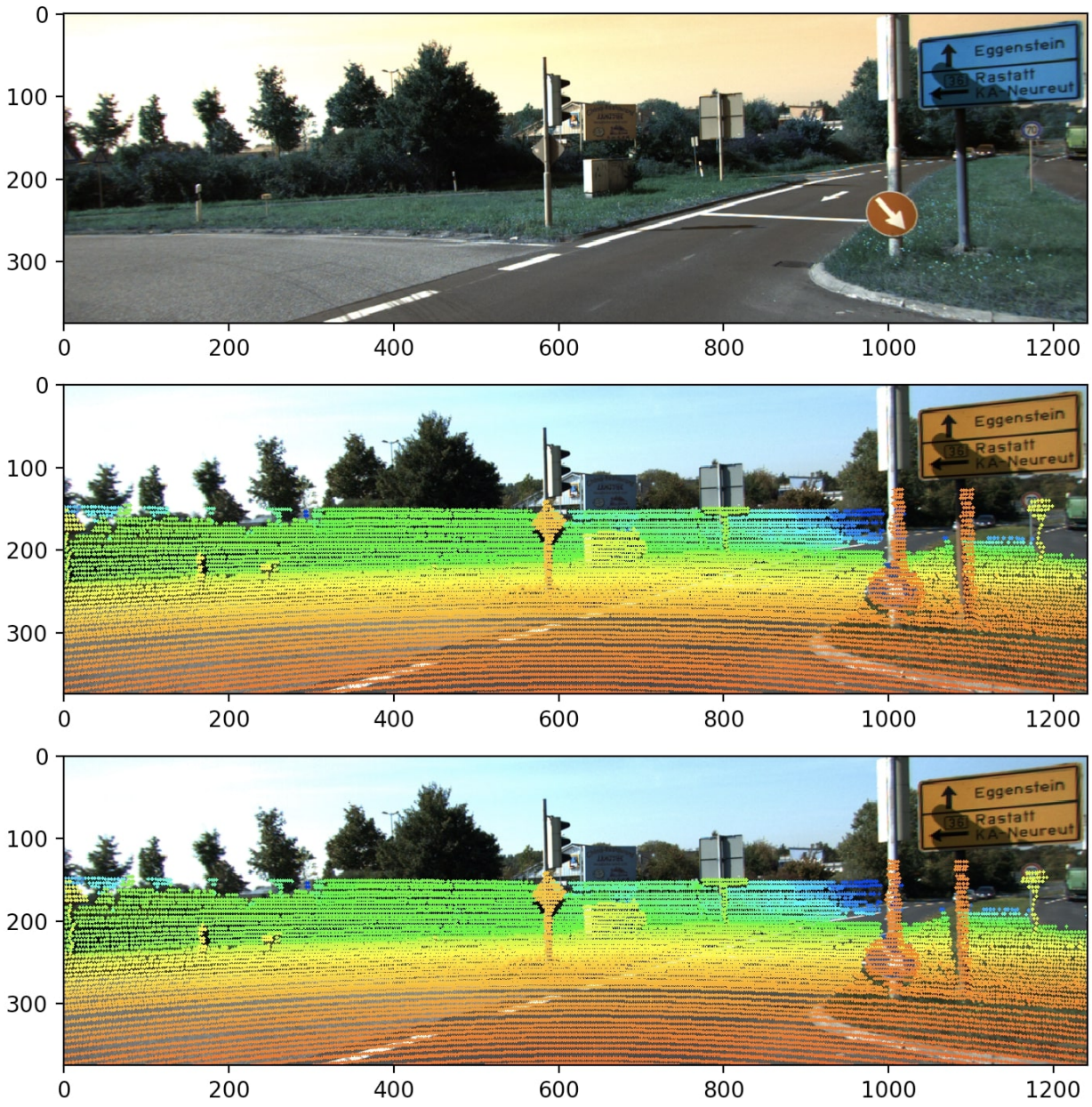

Figure 6: Frame 0000000310

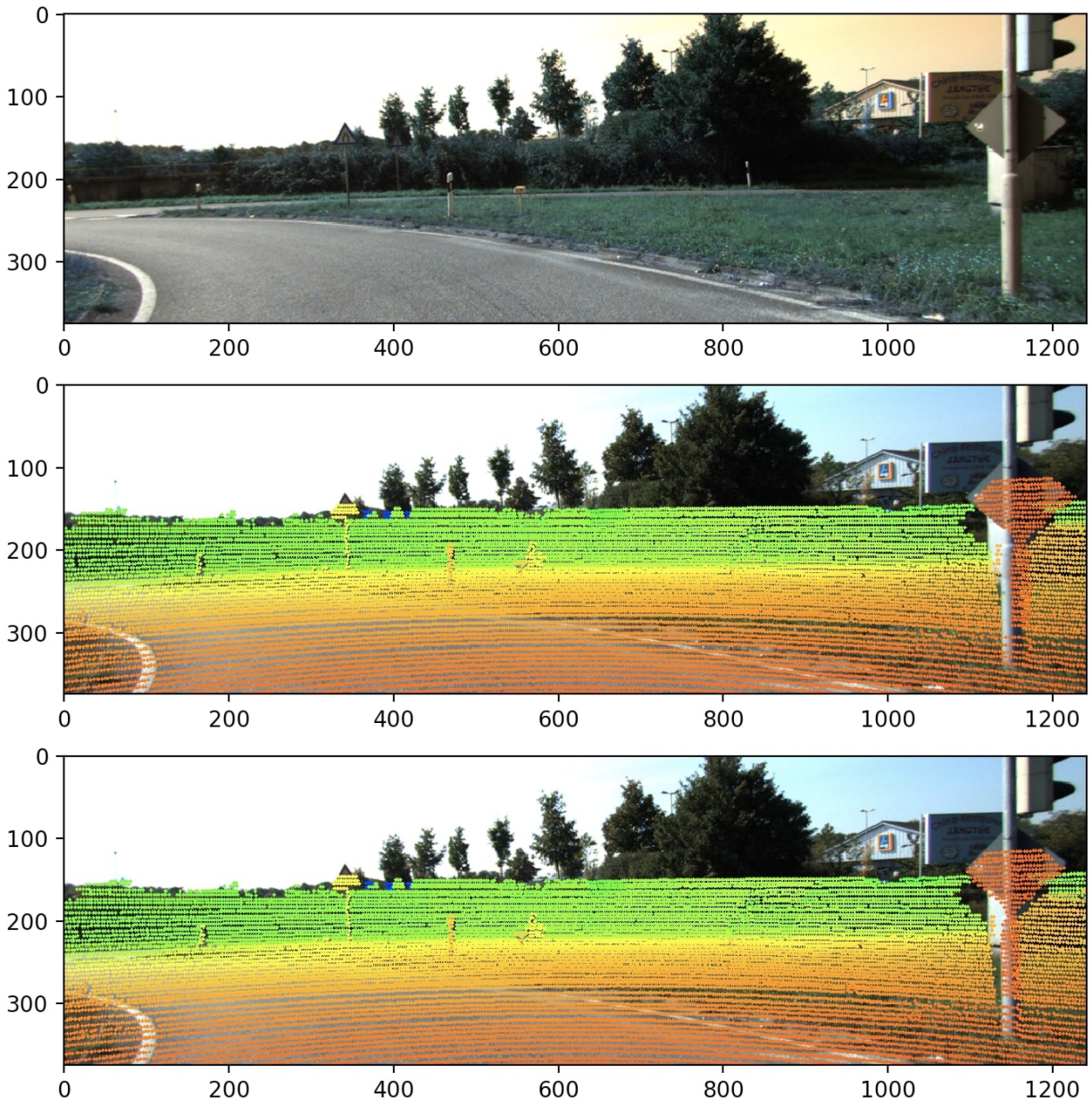

Figure 7: Frame 0000000320