DLAD Homework1

Yunke Ao, Kaiyue Shen

July 4, 2021

1 Bird's eye view

Given the 3D coordinates of the measured points $\{[x_i, y_i, z_i]^T\}_{i=1}^N$, firstly we compute the 2D position of each point in the BEV image $\{[u_i, v_i]\}_{i=1}^N$:

$$u_i = int(\frac{x_i - x_{min}}{0.2})$$
$$v_i = int(\frac{y_i - y_{min}}{0.2})$$

As there may exist may points with duplicated image position, we choose the points with the highest intensity. First we sort the points with u and v coordinate. Then the two pointer technique is used to remove the duplicated positions while recording the highest intensity of each position. The rotated final BEV image is shown in Fig.1.

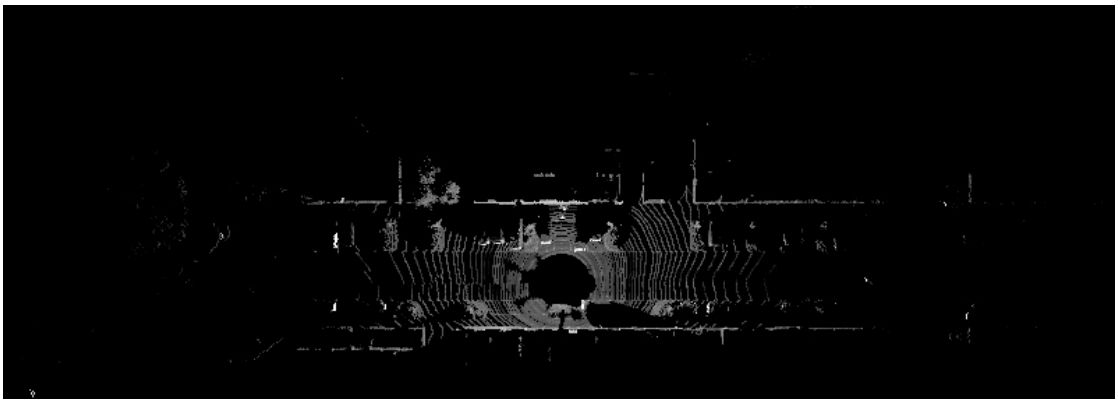

Figure 1: BEV image from the velodine point cloud. The highest intensity points are sampled when multiple points lie within the same bin.

2 Visualization

2.1 Visualization of 3D Semantic Segmentation on 2D image

Given 3D point cloud in the velodyne coordinate system, we first filter out point cloud that are behind the camera 0, i.e., only select points whose x coordinate are larger than 0.27. Next, we multiply T_cam0_velo to transform homogeneous velodyne to rectified camera 0 coordinate and further multiply P_rect_20 to project the rectified point to image plane of camera 2. Since we find some projected points are beyond the scope of the image, we filter out these points. Finally, we use color_map to map semantic label to RGB color for remaining points.

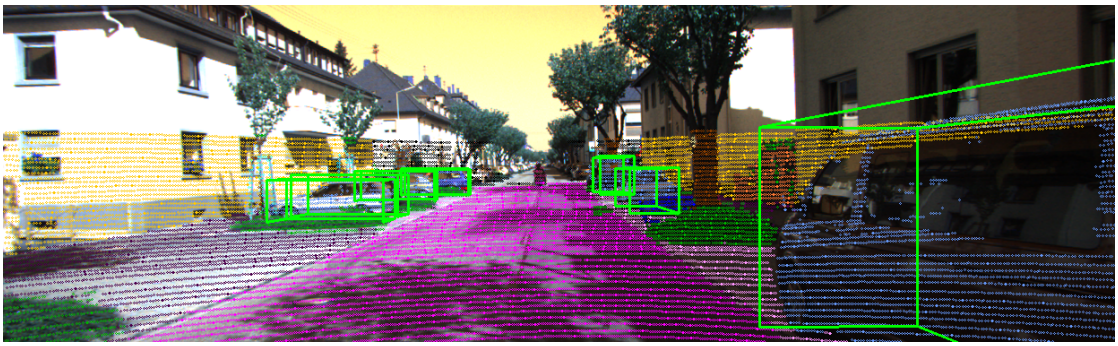

Figure 2: LiDAR point cloud projected on the Cam 2 image along with the predicted bounding boxes. The point cloud is colored using color_map and the predicted pointwise semantic labels.

From Figure 2, we can see the 3D semantic segmentation result is quite ideal. Objects belong to different categories are labeled with different colors, for example, the road is colored magenta and cars are colored blue.

2.2 Visualization of 3D Detection on 2D image

First, for each detected object, we compute the 3D location of 8 bounding box corners in the camera 0 coordinate: With length, width and height of the bounding box, we get the position of 8 corners relative to the bottom center before the rotation. Then we apply rotation R around Y-axis and translation t in Eq 1, where θ is the rotation angle, $[x_c, y_c, z_c]$ is the 3D location of the bottom center c. Now with the 3D location in camera 0 coordinate, we only need to multiply P_rect_20 to project 8 points to image plane of camera 2 and draw lines between them.

$$R = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad t = \begin{bmatrix} x_c \\ y_c \\ z_c \\ 0 \end{bmatrix} \tag{1}$$

The result is shown in Figure 2, we can see that green bounding boxes approximately crop the target object, in our case, cars. Therefore, the 3D object detection is also successful.

2.3 Visualization of 3D Semantic Segmentation and Detection in 3D space

To visualize the semantic segmentation, we use the color_map to map from semantic label to corresponding color for each point. To visualize the detection, similar to task 2.2, we first apply relative rotation and translation to get the location of 8 corners in the camera 0 coordinate. Since in this task, we want to visualize the bounding boxes in 3D velodyne space, we apply the transformation from camera 0 coordinate to velodyne coordinate, i.e., the inverse of T_cam0_velo. Finally, we use to given functions *update* and *update_boxes* to plot the final result as shown in Figure 3.

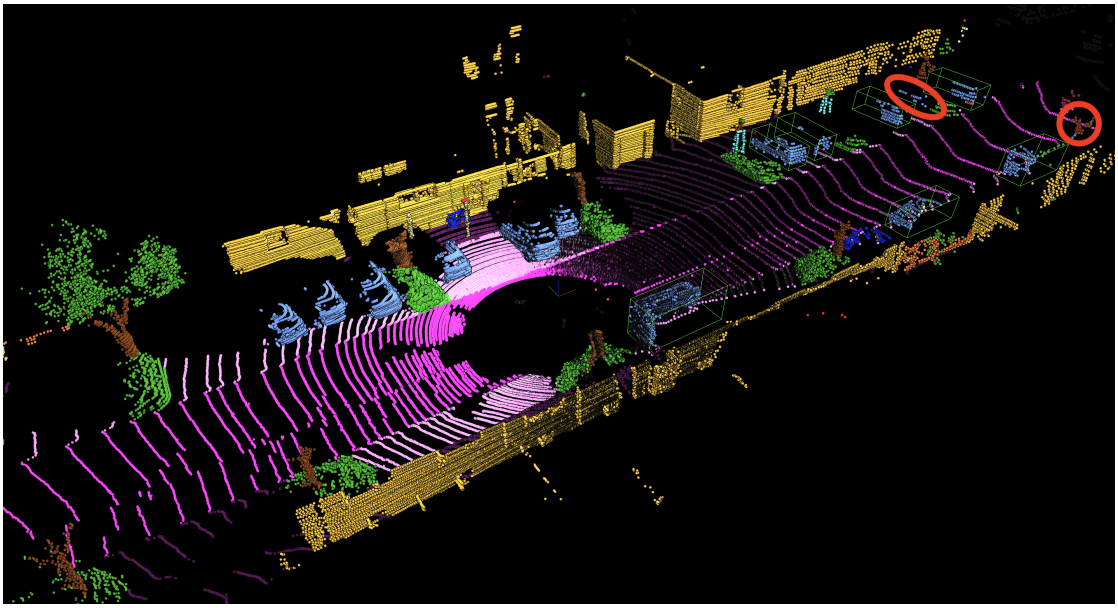

Figure 3: LiDAR point cloud visualized in 3D

We can see that within the camera field-of-view, 2 cars fail to be identified, which are in the red circles.

3 Laser ID

First, same as task 2.1, we project the 3D point cloud onto camera 2 and filter out those beyond the image range. For the remaining points, we compute the vertical angle using the ratio of vertical axis and horizontal axis: $\frac{z}{\sqrt{x^2+y^2}}$, and select the minimum and maximum to get the true Field of View (FOV). We divide the FOV into 64 equal ranges

and assign each point into one range according to its angle. After identifying the Laser IDs, we color them using four alternating colors. The result is displayed in Figure 4.

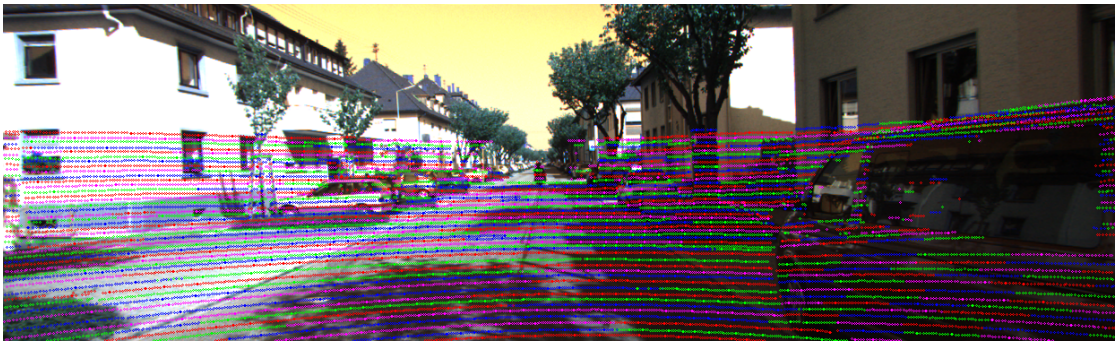

Figure 4: Visualization of Identified Laser ID, use four alternating colors to indicate the IDs

We find that most points visually at the same horizontal level are colored with the same color, though there also exist some small discontinuities between line sections.

4 Remove Motion Distortion

Firsly, the transformations and projection matrixes $T_{vi}, T_{iv}, T_{cv}, P_{20}$ needed are read and computed from the calibration results, where "v", "i" and "c" denotes "velo", "imu" and "camera". Here the transformations and projections $T(P)\_\{destination\}\_\{origin\}$ transform the point from the origin frame to the destination frame by multiplying the homogeneous coordinates in the origin frame. For example, T_{vi} means the transformation matrix that transform the coordinates from the IMU frame to the Lidar frame.

The second step is to compute the relative timestamp of each point measured by Lidar w.r.t. the timestamp of camera. Given the starting time t_s and ending time t_e of Lidar scanning, the angular velocity could be compute:

$$\omega_l = \frac{2\pi}{t_e - t_s}$$

Besides, the relative horizontal angle from front to each point (from 0 to 2π) could be computed with:

$$\alpha_i = \text{argtan2}(y_i, -x_i) + \pi$$

The relationship between the timestamp and angle of each point are shown in Fig 5. Given the timestamp of the camera t_c, the absolute angle from the first point scanned at the starting time to front (the point scanned when the camera is triggered) is:

$$\alpha_s = \frac{t_c - t_s}{t_e - t_s} \times 2\pi$$

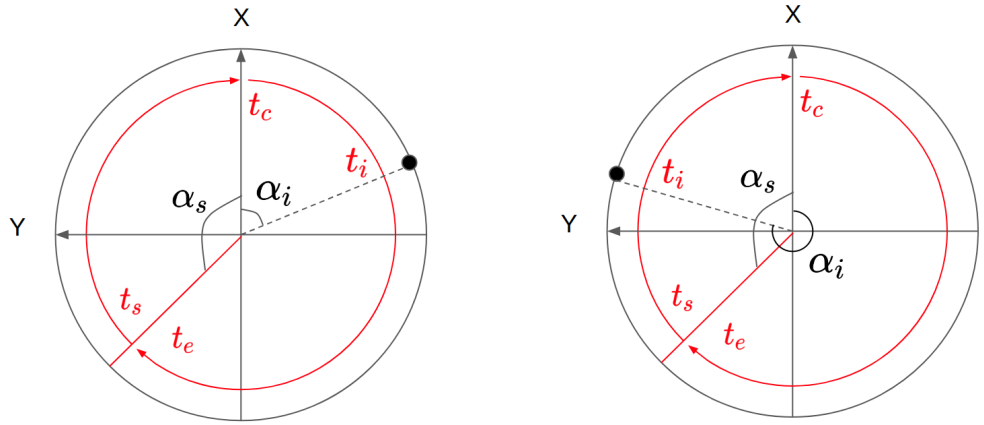

Figure 5: The relationship between the computed horizontal angles and timestamps. The left and right images show the time and angle of points that are scanned later and earlier than camera being triggered respectively.

when $\alpha_i > 2\pi - \alpha_s$, the timestamp of the ith point is earlier than the triggered time of camera, therefore the relative timestamp t_i for each point could be computed with:

$$t_i = \begin{cases} \frac{\alpha_i}{\omega_l}, & 0 < \alpha_i \leq 2\pi - \alpha_s \\ \frac{\alpha_i - 2\pi}{\omega_l}, & 2\pi - \alpha_s < \alpha_i \leq 2\pi \end{cases}$$

The next step is to compute the relative transformation from each point to the camera frame, given the relative time t_i and the constant velocity v_c and angular velocity ω_c. If treating the translation and rotation separately, the transformation is computed with:

$$\theta_i = \omega_c t_i$$

$$R_i = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 \\ \sin(\theta_i) & \cos(\theta_i) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$p_i = v_c t_i$$

$$T_i = \begin{bmatrix} R_i & p_i \\ \mathbf{0} & 1 \end{bmatrix}$$

Then the measured points with Lidar $\{P_v^i\}_{i=1}^N$ could be undistorted and projected to the Cam 2 frame $\{P_c^i\}_{i=1}^N$ with:

$$\hat{P}_c^i = \lambda P_{20} \cdot T_{cv} \cdot T_{vi} \cdot T_i \cdot T_{iv} \cdot P_v^i$$

$$P_c^i = \frac{\hat{P}_c^i[0:2]}{\hat{P}_c^i[2]}$$

Finally, these points are visualized using the *depth_color* and *print_projection_plt* functions. The original and undistorted images are shown in Fig 6 and Fig 7. Our

resulting projected point cloud matched well with the objects in the image. For example, the projected point cloud of the sign coincide well with the corresponding pixels in the image.

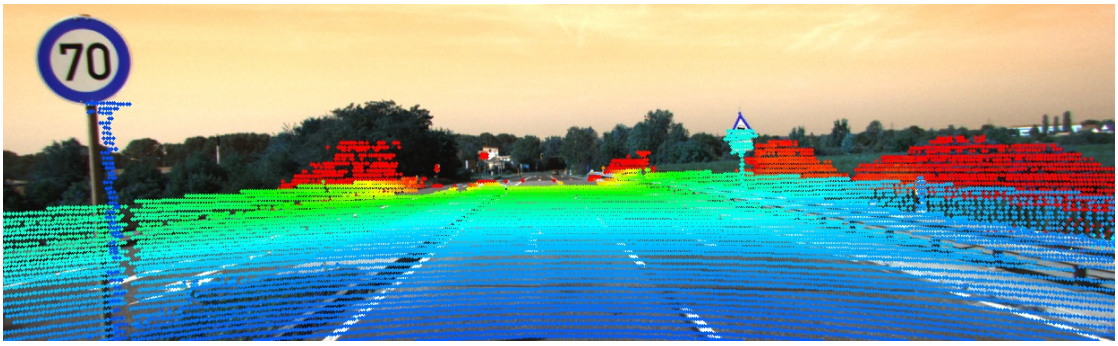

Figure 6: Distorted 3D points without removing distortion. Here the max_d parameter is set to 70.

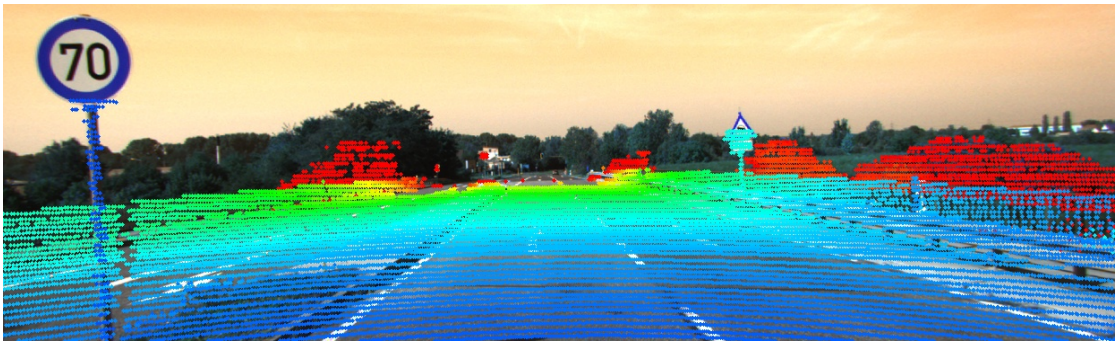

Figure 7: Undistorted 3D points after removing distortion. Here the max_d parameter is set to 70.

5 Bonus Questions

1. When the eyes are closer to the sensor, not only the intensity of light is higher, but also wider range of light are absorbed by the eyes, which results in higher risk.

2. For cameras, the wet road looks like a mirror and its reflections will create confusing mirror images of objects. For LiDAR, the mirror-like surface will reflect the laser light off the water and away from the sensor, making it difficult to detect the road surface.

3. If the camera has more distance from the Lidar in the forward direction, then less Lidar points will be projected to the image plane and more information is

lost. On the other hand, if the camera are further from the Lidar towards left or right, then the projected points are more imbalanced in the image plane, which is also annoying if we need information from both the left and right side. When the distance is larger, these problems are more severe. Additionally, because when the Lidar and camera are non-centered, we always need to calibrate the non-zero extrinsic between the camera and the Lidar to project measured points. The accuracy of extrinsic will influence the quality of projection. When the camera is further from the Lidar, the absolute err have less influence on the projection.