A Point Cloud Sequence Preprocessing

Our approach is similar to [13], except that we use the entire point cloud sequence instead of range images to reconstruct the ground plane.

To remove points on the ground, we observe that the ground plane is a smooth surface that approximately aligns with the x-y plane. We solve an optimization problem to estimate the ground height $h_{i,j}$ at a set of 2D grid cells with centers $(x_{i,j}, y_{i,j})$ on the x-y plane; our implementation uses a 1000×1000 grid. We denote the height of the k-th input point as z_k, the set of points in grid cell (i, j) as $G_{i,j}$, and the neighboring cells of cell (i, j) as $\mathcal{N}_{i,j}$. The optimization problem is

$$\min_{\boldsymbol{h}} \sum_i \left[\frac{1}{|G_i|} \sum_{j \in G_i} \|h_i - z_j\|_1 + \frac{\lambda}{2} \sum_{(i',j') \in \mathcal{N}_{i,j}} \|2h_{i,j} - h_{i',j'} - h_{2i-i',2j-j'}\|_1 \right]. \tag{4}$$

We use a piecewise-linear first term to tolerate noisy inputs. We solve (4) using gradient descent, with $\lambda = 0.1$ and stopping condition $\|\Delta \boldsymbol{h}\|_\infty < 0.01$ m. Then, point $j \in G_i$ is removed if $z_j < h_i + \delta$; our experiments use $\delta = 0.5$ m. An example of the resulting point cloud after ground plane removal can be seen in Fig. 3b.

B Object Trace Tracking Algorithm

For tracking object trace, we begin with the geometric center of the object cluster and zero velocity and iteratively register the object cluster to the adjacent frames; see Algorithm 1. We use two stopping conditions for trace tracking to reject large average registration errors and large acceleration magnitudes, with thresholds $\sigma_0 = 1.0$ m and $\sigma_1 = 3.0$m/s^2.

For each object cluster, the output of tracking is a sequence of rigid transformations that transforms the cluster across frames. We collect the points that are geometrically close to any of the transformed object cluster points within distance $r = 0.3$ m. If the points of two object traces overlap by at least 10% intersection over union (IoU), we reject the one with the smaller number of points. Figure 3(d) shows example extracted object traces.

Algorithm 1 Object Trace Tracking from s-th to t-th frame.

Input: Point cloud sequence $\{P_i\}$, object cluster proposal $C_s \subset P_s$, kalman filter KF, registration error threshold σ_0, acceleration threshold σ_1.
1. Initialize KF with state (avg(C_s), **0**).
2. $\mathcal{T} \leftarrow \{C_s\}$, $\boldsymbol{v}^- \leftarrow \mathbf{0}$.
for $i = s + 1$ **to** t **do**
  3. $C \leftarrow C_{i-1}$ + KF.CurrentVelocity().
  # get deformed points, registration error
  4. $C_i, \sigma \leftarrow$ Rigid-ICP(C, P_i)
  5. $\boldsymbol{v} \leftarrow$ avg(C_i) − avg(C_{i-1})
  if $\|\boldsymbol{v} - \boldsymbol{v}^-\| > \sigma_1$ **or** $\sigma > \sigma_0$ **then**
    BREAK
  end if
  7. KF.UpdateState(avg(C_i))
  8. $\mathcal{T} \leftarrow \mathcal{T} \cup \{C_i\}$, $\boldsymbol{v}^- \leftarrow \boldsymbol{v}$
end for
Output: $\mathcal{T}$

C Details on 3D bounding box estimation

Here we elaborate the loss functions d_1 and d_2 from (3). We first unpack the box attribute vector $\boldsymbol{b}_i \in \mathbb{R}^7$ as $\boldsymbol{c}_i \in \mathbb{R}^3$, $\boldsymbol{s}_i \in \mathbb{R}^3$ and θ_i, representing box center, box size and orientation, respectively. Since the box size is fixed for each trace, d_1 is defined as

$$d_1(b_i, b_j) = \gamma_1 \|c_i - c_j\|^2 + \|\sin(\theta_i) - \sin(\theta_j)\|^2 + \|\cos(\theta_i) - \cos(\theta_j)\|^2 \tag{5}$$

where γ_1 is set to 0.1.

For the definition of d_2, we define outward the normal vector $\boldsymbol{n}_{i,1} \dots \boldsymbol{n}_{i,6}$ and face center $\boldsymbol{c}_{i,1}, \dots \boldsymbol{c}_{i,6}$ for the faces of the 3D bounding box represented by b_i:

$$d_2(\boldsymbol{b}_i, C_i) = \frac{\gamma_2}{|C_i|} \sum_{p \in C_i} \sum_{k=1}^{6} \max(\langle \boldsymbol{p} - \boldsymbol{c}_{i,k}, \boldsymbol{n}_{i,k} \rangle, 0), \tag{6}$$

where γ_2 is set to 0.1 to make sure that this loss function can tolerate a small fraction of outlier points in the point clusters.

D Details on Label Propagation from moving to static objects

Since the goal of label propagation is to generate 3D bounding boxes for static objects using the distilled knowledge about the appearance of moving objects, we apply the trained single-frame object detector on all training point clouds. Specifically, the input to the label propagation algorithm is a set of predicted bounding boxes in a point cloud sequence. And we aim to find more bounding boxes purely based on geometric appearance to improve generalization to static objects. To further enforce consistency, we track the bounding boxes using the exact algorithm in Weng and Kitani [49] while looking for bounding boxes that do not move across temporally adjacent frames. Specifically, we build a spatial-temporal graph of bounding boxes and connect a pair of bounding boxes only if 1) they are overlapping with IoU greater than 0.5, 2) the translation between box centers is not more than 0.3m and 3) they are classified as the same class. We then collect all chain of bounding boxes that are at least 10 frames long, and add all collected 3D bounding boxes to the training set. We apply an iterative algorithm that takes chains of bounding boxes without replacement, and stops when no more chains satisfying all three conditions can be found.

We visualize the intermediate results of our label propagation stated in §4.2 in figure 6.

E Kalman Filter Algorithmic Components

We adapt standard definition[2] to derive our version of Kalman filter, which tracks a moving object point cluster across a sequence of temporally consecutive frames of point cloud.

Kalman filter contains three algorithmic components, i.e. initialization, prediction and update. In algorithm 1, we demonstrated how to use each component of Kalman filter. In below, we illustrate each component separately.

Initialization. Each Kalman filter is initialized with a 6D state vector $\boldsymbol{x} = (\boldsymbol{c}, \boldsymbol{v})$, indicating a 3D location and a velocity. It is also parameterized with three hyper-parameters: Δ_t, a, σ, indicating elapsed time between frames, maximum acceleration, and standard deviation of measurement noise, respectively. During initialization, we set

$$H \leftarrow (I_3; \quad 0) \in \mathbb{R}^{3 \times 6}, \quad A \leftarrow \begin{pmatrix} I_3 & I_3 \\ & I_3 \end{pmatrix}, \quad Q \leftarrow a \begin{pmatrix} \frac{\Delta_t^4}{4} I_3 & \frac{\Delta_t^3}{2} I_3 \\ \frac{\Delta_t^3}{2} I_3 & \Delta_t^2 I_3 \end{pmatrix}$$
$$R \leftarrow \sigma I_3, \quad P \leftarrow I_6$$

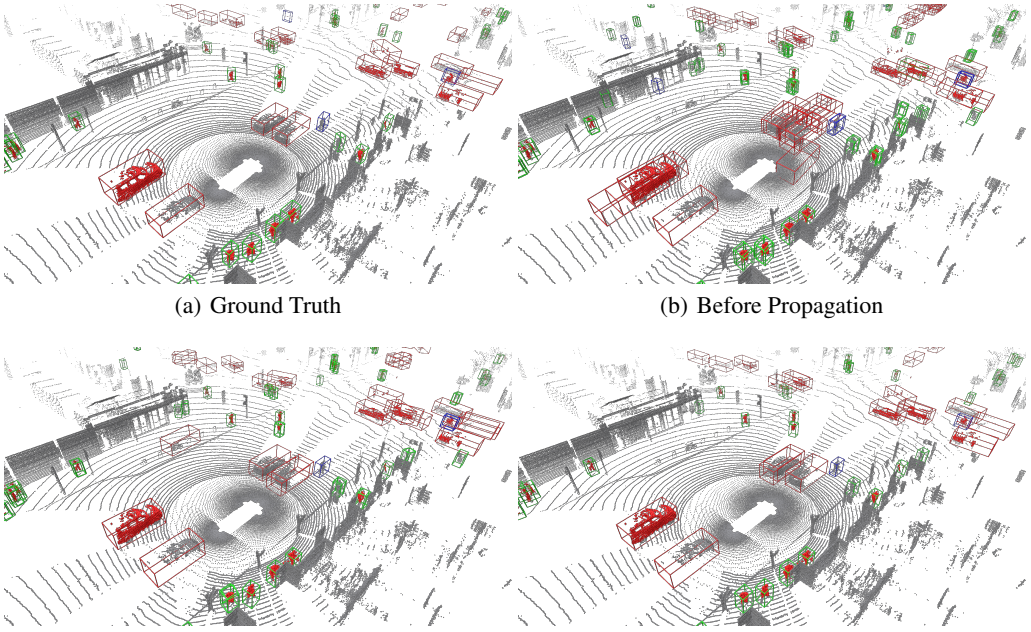
Prediction. This is where we predict the next 3D location. Specifically, we set

$$x \leftarrow A\boldsymbol{x}, \quad P \leftarrow APA^T + Q \tag{7}$$

Update. When we observe the next observed location $\boldsymbol{z}$, we update state vectors. Specifically, we set

$$S \leftarrow HPH^T + R \tag{8}$$
$$K \leftarrow PH^T S^{-1} \tag{9}$$
$$x \leftarrow x + K(z - Hx) \tag{10}$$
$$P \leftarrow (I_6 - KH)P \tag{11}$$
$$\tag{12}$$

[2] https://en.wikipedia.org/wiki/Kalman_filter



(a) Ground Truth (b) Before Propagation

(c) First Iteration (d) Second Iteration

Figure 6: Qualitative results of our label propagation algorithm. The top-left figure shows the ground truth bounding boxes, with moving object points in highlighted in red and boxes colored based on their object class (red: Vehicle, green: pedestrian, blue: cyclist). The next three figures shows the effect of label propagation, essentially we are making more and more correct predictions of bounding boxes on static objects. The second iteration corresponds to the prediction of our final object detector.