ICP-Flow: LiDAR Scene Flow Estimation with ICP

Supplementary Material

6. ICP-Flow: cluster pairing

This section details the optimized cluster pairing procedure introduced in Section 3.5, where the goal is to coarsely pair clusters that are likely to be correspondences. Further, we improve over Section 3.5 by leveraging the cluster indices from HDBSCAN, and explain its reasoning in detail. We start with clusters that share the same cluster index, *i.e*, $\mathbf{C}_m^t$ and $\mathbf{C}_n^{t+\Delta t}$ where $m = n$, as they are highly likely to be static or slow-moving. This is because HDBSCAN tends to group close-by points as one. We pair these clusters and send them to ICP matching (Section 3.6), a procedure that measures to what extent a cluster aligns with the paired one. Afterward, we reject unreliable pairs if the inlier ratio r or distance d exceeds the predefined threshold, *i.e.* $r < \tau_r$ or $d > \tau_d$ where τ_r or τ_d are manually defined in Section 3.7. We remove successfully matched pairs from the original set of clusters obtained from HDBSCAN. This way we substantially reduce the search space.

We then process the remaining unmatched clusters after the aforementioned procedure. We search for possible matches in a local neighborhood around $\mathbf{C}_m^t$, *i.e.* a square region of size $\tau_x \times \tau_y$ where τ_x and τ_y (in *meters*) are the maximal translation possible within Δt along the x and y dimensions. We pair each cluster $\mathbf{C}_m^t$ with remaining clusters at time $t + \Delta t$ that lie in the predefined region. Subsequently, we feed these pairs to ICP matching (Section 3.6) and cluster association (Section 3.7) for further validation.

7. ICP-Flow: tracking over multiple scans

We detail the design of the proposed Ours+Tracker in Section 4.5, which estimates scene flow from a sequence of scans. Simply speaking, we first estimate scene flow from every pair of nearby scans, thus obtaining a set of matched clusters, together with their cluster indices and transformations. Then, given a random cluster as a query, we iteratively search for its correspondence over each pair of nearby scans, starting from the current scan and stopping at the initial scan. Finally, we transform the query cluster sequentially by estimated transformation at each time step and recover the scene flow for a longer time duration. By this means we avoid missing matches over time. It is worth mentioning that Ours+Tracker does use intermediate frames while other models do *not* use intermediate scans in Tab. 4. Additionally, we show a comparison, in Tab. 5, with PCA+Tracker[17], where the learned spatio-temporal associator in the original design is replaced by a constant-velocity Kalman tracker [54, 55]. Simply speaking, the Kalman tracker solves association *over time* by greedily matching the centroids of clusters based on $\mathbf{L}^2$ distance. We directly use the result from [17]. The comparison between PCA and PCA+Tracker shows that the simple Kalman tracker underperforms considerably as it suffers from incorrect centroid estimation. In comparison, Our+Tracker is able to outperform PCA on dynamic foreground thanks to the ICP-based tracking.

8. Comparison with RigidFlow [25]

We additionally compare with RigidFlow [25] on the KITTI$_o$ dataset [28], as both models follow the "clustering + ICP" pipeline for flow estimation. A key difference is that RigidFlow uses a deep network for initial pose estimation, while ours uses histogram-based initialization without relying on learning from data. We report the result using the official checkpoint from authors on KITTI$_r$ [24] and using trained checkpoint by ourselves on Waymo [41]. Since RigidFlow does not support full point cloud inference on our device due to the high demand for GPU memory, we randomly sample a maximum of 40,000 points from each scan for inference. As shown in Tab. 6, our model outperforms RigidFlow [25] substantially, despite its simplicity. We did not include results on longer sequences as RigidFlow fails to produce a visually reasonable prediction.

9. Ablation study

We test the added value of the histogram-based initialization for ICP matching (Section 3.6) in Tab. 7. We compare against the commonly used centroid alignment. As shown in the result, a good initialization is essential for ICP matching as Ours *(centroids)* underperforms significantly. Fig. 4 shows a failure case of centroid subtraction, which happens frequently over a longer temporal horizon. Additionally, we also test the performance of our design (Ours+KISS-ICP) in the case where ego-motion information is unavailable. We use KISS-ICP [46] to estimate a relative transformation between scans. Results show a considerable performance drop on static background. Our observation aligns with [9] on the importance of ego motion compensation. However, it is a valid and common assumption for autonomous driving to have ego motion available. Additionally, instead of using $\arg\min$ for cluster association, we also test Hungarian matching [11] which yields marginally better results than the default setup.

Metrics	Label	Dynamic Foreground			Static Foreground			Static Background		
		EPE (m) ↓	Acc-S (%) ↑	Acc-R (%) ↑	EPE (m) ↓	Acc-S (%) ↑	Acc-R (%) ↑	EPE (m)↓	Acc-S (%) ↑	Acc-R (%) ↑
PCA [17]	✓	**0.1970**	53.31	77.49	**0.0216**	**97.16**	**99.44**	**0.0289**	**97.16**	**99.44**
Ours	-	0.2209	**67.59**	**84.66**	0.0272	96.08	99.16	0.0711	96.49	97.96
PCA+Kalman Tracker [17]	✓	0.5860	36.30	61.60	0.0270	-	-	0.0300	-	-
Ours+Tracker	-	0.1799	58.98	80.98	0.0341	88.53	97.73	0.0722	93.74	97.46

Table 5. **Scene flow on Waymo dataset [41], over a longer temporal horizon (5 consecutive frames, 0.4 seconds).** Given a clip of 5 consecutive scans, we compute the flow between the first frame and the other frames. The result is averaged over all points. We split models that use intermediate scans (with "Tracker" in their names) from others. We highlight that Ours+Tracker is able to further improve the quality of scene flow by leveraging intermediate frames.

Datasets	KITTI_o			Waymo		
Metrics	EPE	Acc-S	Acc-R	EPE (Dynamic)	EPE (Static Foreground)	EPE (Static Background)
RigidFlow (KITTI_r)	0.1192	40.99	69.77	0.2575	0.1299	0.2517
RigidFlow (Waymo)	0.2748	7.47	26.25	0.2904	0.1673	0.3100
Ours	**0.0423**	**94.30**	**94.42**	**0.0799**	**0.0165**	**0.0270**

Table 6. **Comparison with RigidFlow on KITTI_o and Waymo (0.1 seconds).** We indicate the training dataset in the bracket. Despite being simple, our model outperforms RigidFlow by a large margin, without relying on large quantities of data for training and powerful compute.

	Dynamic Foreground	Static Foreground	Static Background
Ours	0.2209	0.0272	0.0711
Ours *(centroid alignment)*	0.3511	0.0789	0.1861
Ours *(Hungarian Matching)*	**0.2163**	**0.0260**	**0.0681**
Ours+KISS-ICP [46]	0.2617	0.0572	0.3386

Table 7. **Ablation study.** We report EPE errors on Waymo over 5 consecutive frames [17, 41]. Without the histogram-based initialization, the performance decreases substantially. Precise ego-motion is also critical for scene flow, particularly for static background. When replacing $\arg\min$ by Hungarian matching [11] during cluster assignment, our model yields marginally better results.

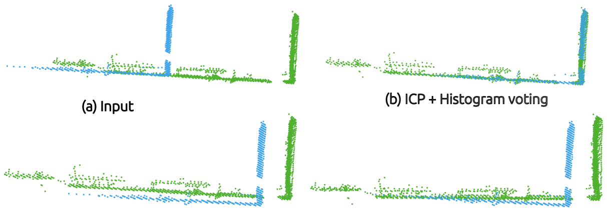
(a) Input
(b) ICP + Histogram voting
(c) Centroid subtraction/alignment
(d) ICP + Centroid alignment

Figure 4. **ICP with centroid alignment.** We show a pair of associated clusters in (a), colored in green and blue respectively. They are the bird-eye view of a moving truck. ICP fails (d) when simply subtracting the centroids (c).

10. Visualization

We visualize the predicted scene flow from our model and highlight several failure cases in Fig. 5, Fig. 8, and Fig. 7. These qualitative results show the capability of ICP-Flow to extract scene flow in various scenarios reliably.

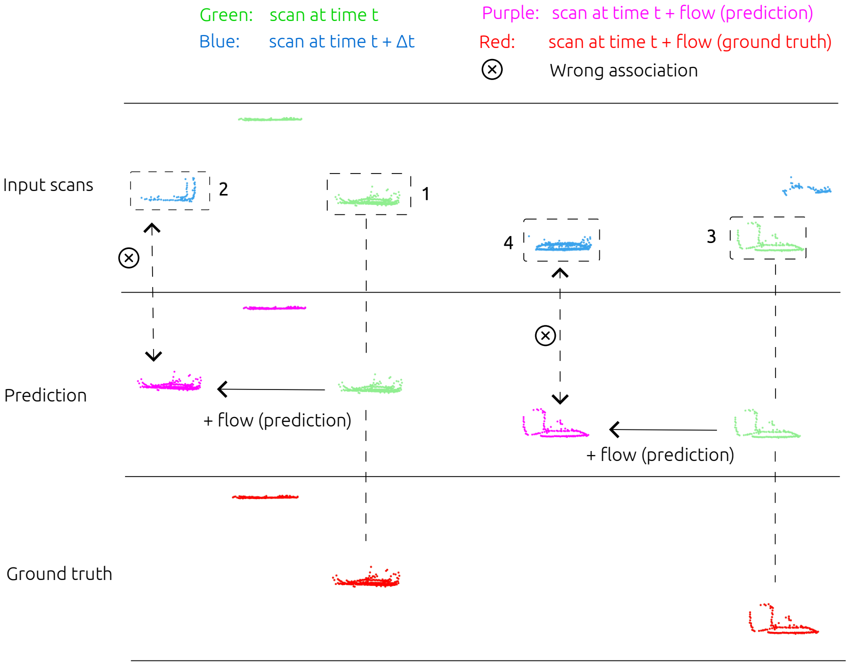

Figure 5. **Visualization of predicted scene flow.** We qualitatively compare our prediction to the ground truth. For better visualization, we crop the region of interest from the entire scan. We plot the input scans at time t and $t + \Delta t$, namely $\mathbf{X}_t$ and $\mathbf{X}_{t+\Delta t}$, in green and blue, respectively. We color the flow-compensated scan at time t, namely $\mathbf{X}'_t$, in purple by adding the predicted scene flow $\mathbf{F}_t$ to $\mathbf{X}_t$. In comparison, we use red to indicate the flow-compensated scan at time t, namely $\mathbf{X}^*_t$, by adding the *ground truth* flow. The left figure is composed of $\mathbf{X}_t$, $\mathbf{X}_{t+\Delta t}$ and $\mathbf{X}'_t$. ICP-Flow is able to output reasonable predictions once the blue and purple points align (*i.e.* overlap) with each other. However, ICP-Flow fails in certain scenarios by associating the wrong clusters, as indicated by the box on the top. We highlight this failure in the right figure, where ✗ denotes a wrong association. As indicated by the dashed lines on the left, ICP-Flow associates clusters 1 and 2 (or $\mathbf{C}_1^t$ and $\mathbf{C}_2^{t+\Delta t}$), and estimates a transformation that best aligns them. Unfortunately, $\mathbf{C}_1^t$ remains static within Δt according to the ground truth (in red). Similarly, we observe that $\mathbf{C}_3^t$ and $\mathbf{C}_4^{t+\Delta t}$ are also falsely associated. Interestingly, after careful examination, we find this an annotation error in the preprocessed Waymo dataset [17], as explained in Fig. 6.

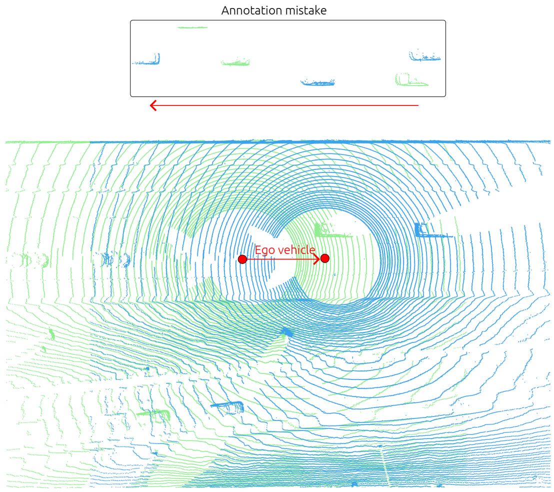

Figure 6. **Visualization of the original scans from Fig. 5, after ego-motion compensation.** After careful examination, we find that Fig. 5 is not perfectly annotated and ICP-Flow is actually making a reasonable prediction. We highlight the clusters that a visual examiner intends to associate in boxes, based on the observation that they are heading from right to left (indicated by the red arrow below the box). However, in the preprocessed Waymo dataset [17], these points (in green and inside the box) are labeled as static (*i.e.,* without having correspondences), which we assume to be a mistake during preprocessing. We manually examined numerous examples and did not find other annotation errors.

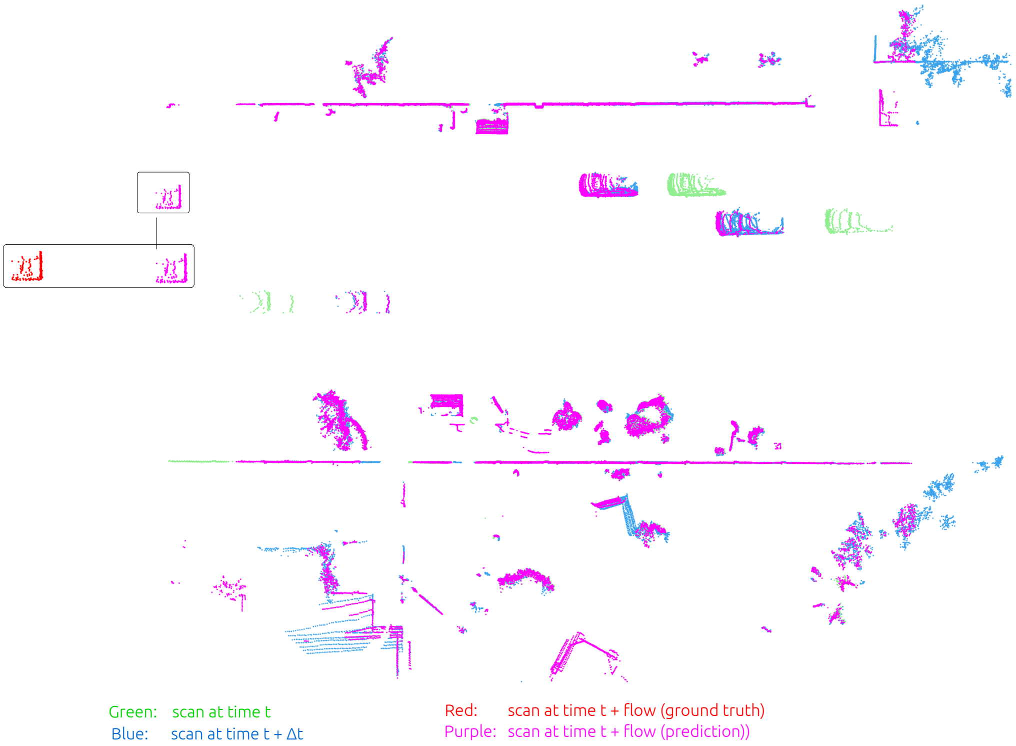
Green: scan at time t
Blue: scan at time t + Δt
Red: scan at time t + flow (ground truth)
Purple: scan at time t + flow (prediction))

Figure 7. **Failure case.** We show another failure case where a cluster moves out of the perception range, as indicated in the box. Thus ICP-Flow fails to associate and outputs zero scene flow, *i.e.* the cluster moves identically to the ego autonomous vehicle. This often leads to substantially large errors for dynamic foreground.

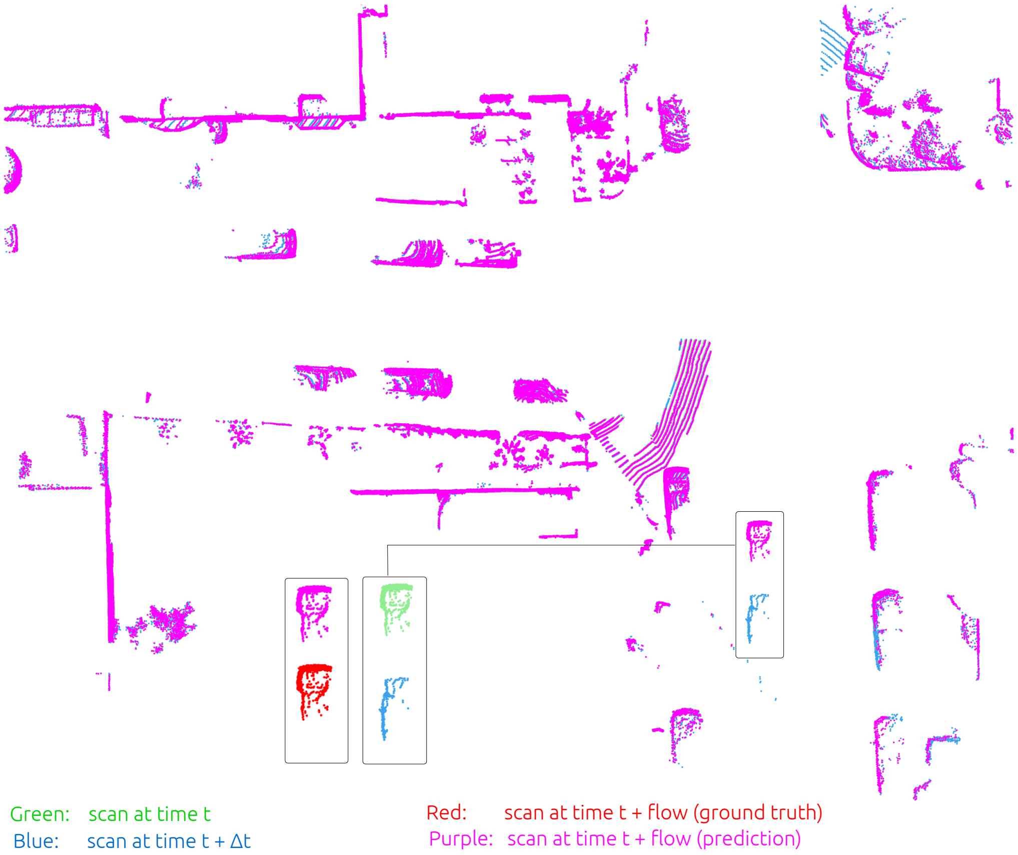
Green: scan at time t
Blue: scan at time t + Δt
Red: scan at time t + flow (ground truth)
Purple: scan at time t + flow (prediction)

Figure 8. **Failure case.** We show a failure case where occlusion happens. We highlight this failure in boxes, where our model predicts zeros for the given cluster (in green), as the purple and green points overlap seamlessly. This results from (1) low inlier ratio, as the blue cluster at time $t + \Delta t$ consists of much fewer points than the green cluster at time t; (2) partial occlusion, as we are unable to observe the blue cluster from a similar view, thus making it hard to match with the green cluster.