OptFlow: Fast Optimization-based Scene Flow Estimation without Supervision Supplementary Material

Rahul Ahuja Chris Baker Wilko Schwarting
ISEE AI
{rahulahuja, chrisbaker, wilko}@isee.ai

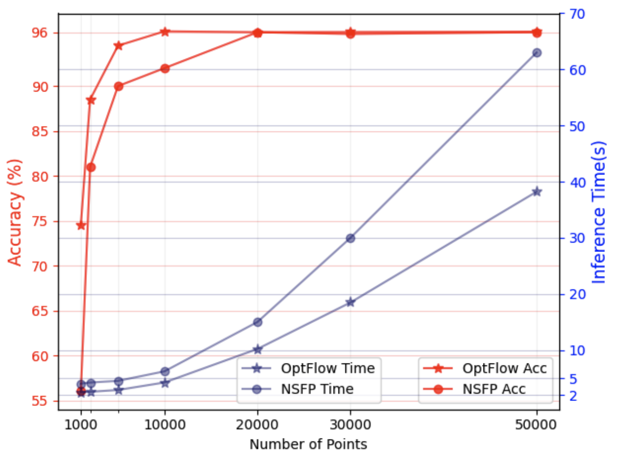

Figure 1. **Performance comparison of our algorithm and NSFP on the KITTI dataset for varying point cloud densities computed as a whole point cloud together.**

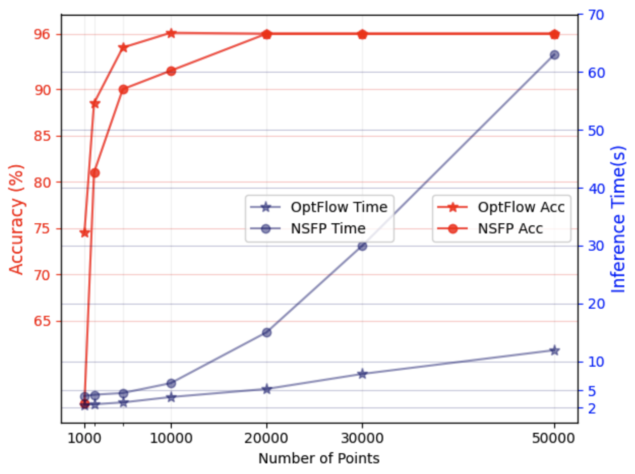

Figure 2. **Performance comparison of our algorithm and NSFP on the KITTI dataset for varying point cloud densities computed in parallel batches of 8192 points. This shows we get around 7x speedup over NSFP [8] as the point cloud density increases.**

1. Faster OptFlow in Batches

The KNN search between the source and target point clouds within the fit function loss constitutes a significant bottleneck, increasing the inference time of our model. As the number of points grows, so does the search space, thereby leading to an extended time for the KNN operation to find the best correspondences. Importantly, as depicted in Figure 1 from above, our model's performance regarding Acc_5 tends to level off around 8k-10k points. As a result of this observation, we chose to process our point clouds in parallel batches of 8192 points. This approach involves randomly sampling sets of 8192 points, batching them together, and running our scene flow estimation algorithm on them in parallel. Upon completion, we collate the results for final evaluation. By limiting the search space for each KNN operation, this method substantially cuts down on processing time while preserving the accuracy of our scene flow estimates. The revised correlation between accuracy and time, adjusted for different point cloud densities, is presented in Figure 2. As indicated by the figure, this strategy allows our model to operate seven times faster than NSFP [8], significantly enhancing its efficiency.

2. Real-Time OptFlow

The original implementation of OptFlow typically takes around 1.5 to 2 seconds to converge. However, its suitability for real-time applications depends on the specific use case, accuracy requirements, and End Point Error (EPE) constraints. To explore its real-time potential, we conducted an experiment using sequential data from the NuScenes dataset [3], with a time interval of 0.2 seconds between frames.

In this experiment, we initially processed the first two consecutive point clouds, running our optimization algorithm for a full 500 iterations. We then initialized the flow values for the next pair of point clouds using nearest neigh-

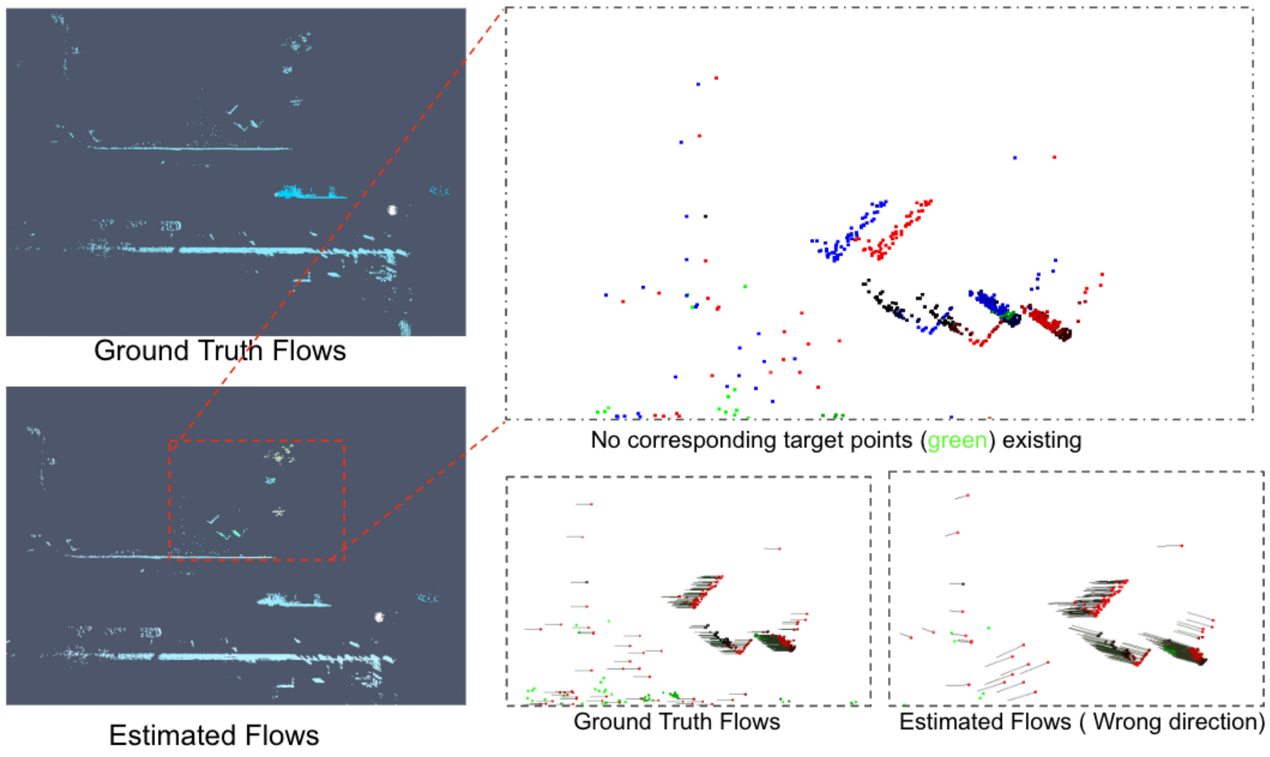

Figure 3. **Failure case due to missing points:** The source point cloud is represented in red, the target point cloud in green, and the warped point cloud in blue. We can observe that the flow estimates for the static points are incorrect. This is due to the lack of corresponding points, which prevents our model from forming accurate scene flow estimations for these points. Despite this, the results are not significantly compromised thanks to our **Adaptive Distance Threshold** feature. This component reduces the limit for finding correspondences, thereby diminishing the likelihood of numerous false positives.

bor matches from the previously estimated flows. In dynamic scenes, scene flow vectors for objects in consecutive frames tend to be similar or exhibit slight changes if objects are in motion. This allows us to reduce the number of iterations required for subsequent runs while maintaining good accuracy.

For the consecutive runs, we executed our algorithm for just 30-40 iterations, taking approximately 0.17-0.2 seconds to complete. Importantly, we retained the flow vectors and the ICP transformation function from the previous run. Although we lacked ground truth flow annotations for these samples, we devised an evaluation metric called End Point Error Difference (EPED), calculated as follows: $EPED = EPE(flow_{500}, flow_{30})$. Here, $flow_{500}$ represents a run with 500 iterations, while $flow_{30}$ represents flow values obtained from 30 iterations.

Remarkably, our experiments yielded an $EPED$ value of only 0.053 meters. This implies that by reducing the number of iterations and achieving an inference speed of 200 milliseconds for OptFlow, we only compromised 0.05 meters of EPE accuracy, making it a viable option for real-time applications.

This real-time OptFlow optimization will be further explored in our future work.

3. Failure Cases

Our method, which relies on the nearest neighbor search in the fit function, can potentially encounter failure cases. Specifically, this can occur when no correspondence exists between a point in the source point cloud and another point in the target point cloud. Under such circumstances, the loss function may falsely identify a match, leading to false positives. This scenario is frequently encountered with static background points such as trees, poles, and occluded objects as these often present missing points in subsequent point clouds. The problem is amplified when we evaluate using point clouds of 2048 or 8192 points, given that points are randomly sampled. When the point cloud is sparse, this may lead to the generation of false positives. Nevertheless, as point density increases, the risk of this failure decreases, enhancing our evaluation results.

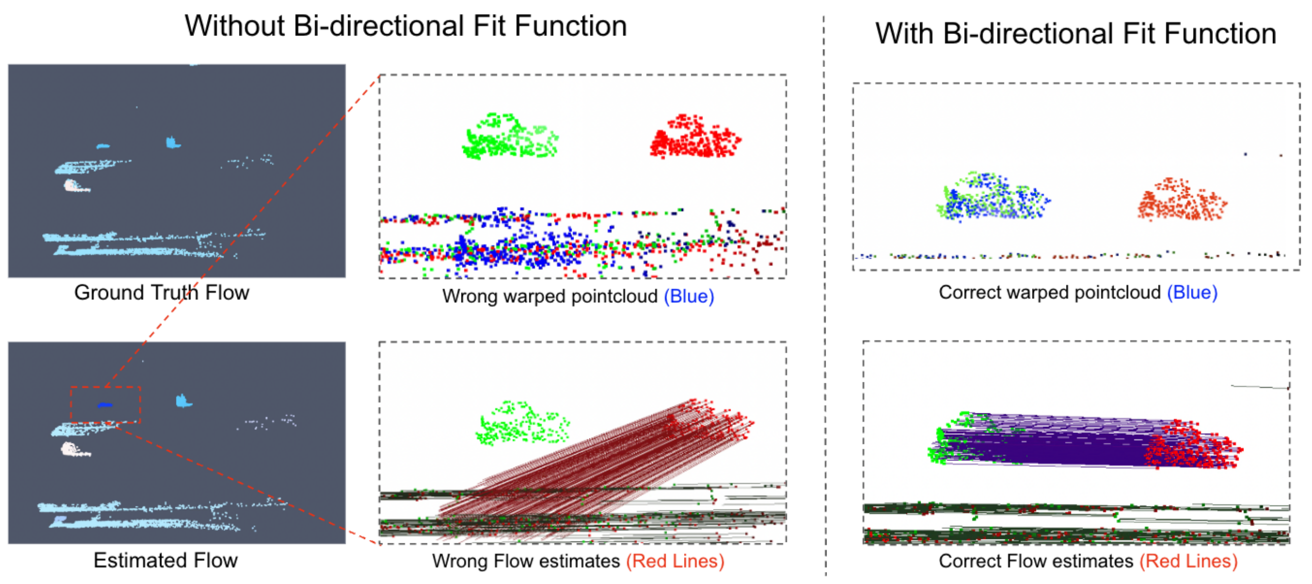
Without Bi-directional Fit Function
With Bi-directional Fit Function
Ground Truth Flow
Wrong warped pointcloud (Blue)
Correct warped pointcloud (Blue)
Estimated Flow
Wrong Flow estimates (Red Lines)
Correct Flow estimates (Red Lines)

Figure 4. **Impact of the bi-directional Fit Function:** The source point cloud is depicted in red, the target point cloud in green, and the warped point cloud in blue. On the **left**, we observe incorrect scene flow estimates that do not align with the ground truth flow values. In the **center** image, the assigned correspondences are inaccurate, resulting in a misplaced warped point cloud. On the **right**, we see the dramatic improvement achieved by applying the bi-directional fit function. The correspondences are significantly corrected and closely match the target point cloud.

Figure 3 exemplifies the impact of these missing points using a scene from the Argoverse dataset. In the color-coded map, our scene flow estimations for the area marked in red are observed to deviate from the ground truth flows. Upon closer examination of these points, we find that corresponding points are absent in the target point cloud, as shown by the absence of green points. The estimated flows for these points are visually represented.

Despite the flaws in our flow angles, the flow estimates are not excessively erroneous. This is attributable to the adaptive distance threshold. In the limitations highlighted by works such as [13] and [7], these warped point clouds ($P_{T-1}+F$) would have shrunk towards the nearest available point. However, in our case, the adaptive distance threshold curtails the impact of false positives, hence yielding better estimates.

3.1. FlyingThings3D

Continuing with the discussion of challenging scenarios, the FlyingThings3D dataset presents an interesting case due to its frequent occlusions and partial visibility. It's worth noting that our algorithm exhibits lower performance in terms of End Point Error compared to learning-based methods that were specifically trained on the FlyingThings3D dataset. As previously explained, our reliance on nearest neighbor loss means that in cases of occlusions and limited visibility, the assigned nearest correspondence may not be correct. An example provided below illustrates a situation where the source point cloud, when combined with the estimated flow, misses the target point cloud.

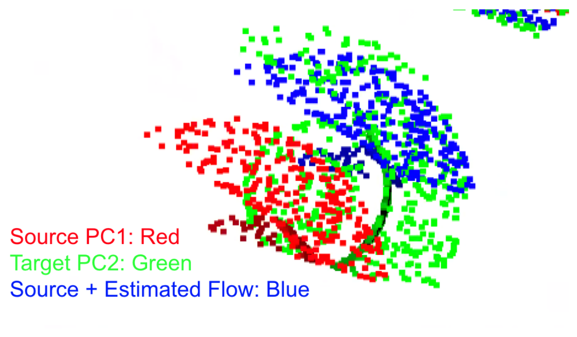

Figure 5. **FlyingThings3D failure** This is a part of an object from a scene in the FlyingThings3D dataset. The red points denote the source point cloud, while the green points represent the target point cloud. Additionally, the blue points illustrate the source point cloud after integrating estimated flows. A notable observation is that the source point cloud contains fewer points, indicating partial visibility compared to the target point cloud. Consequently, our algorithm estimates scene flow for the reduced set of points in the source cloud, leading to missing flow estimations for numerous points in the target point cloud. This disparity in flow estimations results in higher End Point Error (EPE) values and lower accuracy scores in such scenarios.

Change in visibility is a common challenge faced by optimization-based methods. However, despite these difficulties, our approach still manages to achieve a remarkable 12% improvement in Acc_5 compared to the current state-of-the-art optimization-based method [8]. This demonstrates the significant contributions our methodology brings to the field, even in demanding scenarios.

4. Bi-directional fit function

As outlined in our primary submission, we apply our fit function 1 in a bi-directional manner.

$$E_{fit} = \sum_{i=1}^{n_1} ||Tp_i + f_i - q_{avg_i}||_2^2. \tag{1}$$

This approach mimics the effect of the Chamfer Distance [5]. This original metric is a nearest neighbor loss function, which identifies one nearest neighbor for each point in the source point cloud P_{T-1} in relation to the target point cloud P_T, and subsequently calculates the mean squared difference between the two. This process is then reciprocated, with the source serving as the target and vice versa. The mean of these two error terms constitutes the Chamfer loss.

In line with this, our approach initially designates P_{T-1} as the source and P_T as the target. In the second round, these roles are reversed with P_T serving as the source and P_{T-1} as the target point cloud. The corresponding q_{avg} is determined through the locally correlated weight matrix applied to each target point cloud.

Table 2 presents the performance of our model when using the bi-directional fit function compared to a unidirectional application. As the results indicate, the bidirectional application of our fit function yields notably improved performance.

In figure 4, we also depict how the flow estimates fail without the bi-directional fit function.

Experiment	$EPE\downarrow$	$\%_5\uparrow$	$\%_{10}\uparrow$	θ_ϵ
(a) Bi-directional fit loss	0.049	88.25	95.1	0.13
(b) Non Bi-directional fit loss	0.073	85.26	92.7	0.16

Table 2. **Effect of Bi-directional fit function on KITTI using 2048 points.**

5. Implementation Details

In this section, we share the hyperparameters that resulted in the outcomes presented in our paper. We employed RayTune [9] and Hyperopt [2] for an extensive hyperparameter search. Throughout our experiment, we maintained K_{rigid} at a constant value of 50, as this yielded the most optimal results and was recommended in [13].

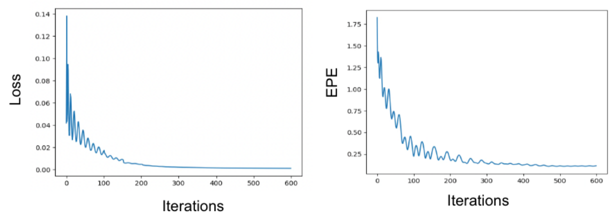

Figure 6. **Left:** Loss curve vs iterations, **Right:** EPE vs iterations

The parameters α_{rigid} and K_{local}, which are used in the nearest neighbor search for the local target correlation matrix, played a vital role in achieving the desired outcomes. Table 3 outlines the hyperparameters applied in our experiment.

We advise conducting your own hyperparameter search to identify the most suitable solution for your specific scenario.

Dataset	K_{local}	α_{rigid}
(a) FlyingThings3D [11]	12	14.2
(b) NuScenes [3]	20	19.6
(c) KITTI [6]	15	9.57
(d) Argoverse [4]	15	19.2

Table 3. **Hyperparameters used for each dataset.**

6. Visualizations

6.1. Comparison of OptFlow with Baseline

In Figure 7, we contrast our proposed OptFlow method with the baseline method, Graph Prior [13]. It becomes evident that our flow estimates significantly outperform those produced by Graph Prior, particularly in correctly predicting flow values for static background points. This comparison clearly demonstrates the advantages of OptFlow in terms of accuracy in scene flow prediction.

6.2. Effect of integrated ego-motion compensation

In many state-of-the-art scene flow estimation algorithms, the primary focus is typically on dynamic objects, with static objects or points often receiving less attention. This approach may work satisfactorily for dense datasets such as KITTI, but it can result in a high likelihood of false positives when dealing with sparse datasets.

The motivation for our research to incorporate an integrated ego-motion compensation was to mitigate these false positives. By aligning the static points effectively, our algorithm is better positioned to accurately identify the scene flow estimates of dynamic objects.

As depicted in Figure 8, the inclusion of an integrated

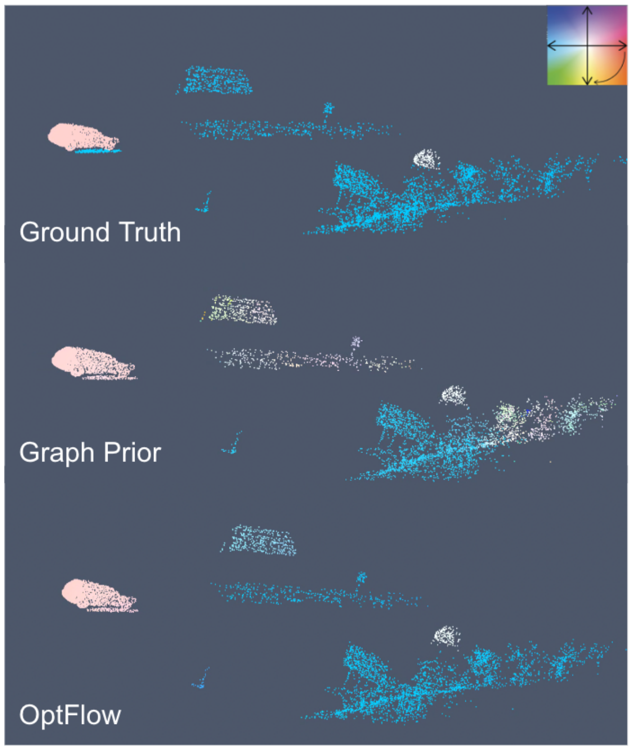

Figure 7. **Scene Flow Visualization of Graph Prior [13] Method vs. OptFlow (Ours)**: Our method's estimated scene flow, particularly in the background, presents a more accurate representation compared to the Graph Prior method. The ground truth scene flow aligns more closely with our estimation.

ego-motion compensation transformation function significantly reduces false positives and yields estimates that are more closely aligned with the ground truth flow values.

6.3. Tough Example: Ego vehicle surrounded by dynamic objects

In figure 9, we provide an enlarged view of Figure 3 from the main paper. This visualization illustrates a challenging scenario where the ego vehicle is navigating a turn and is surrounded by numerous dynamic objects. Remarkably, our algorithm continues to deliver excellent performance even in these complex conditions.

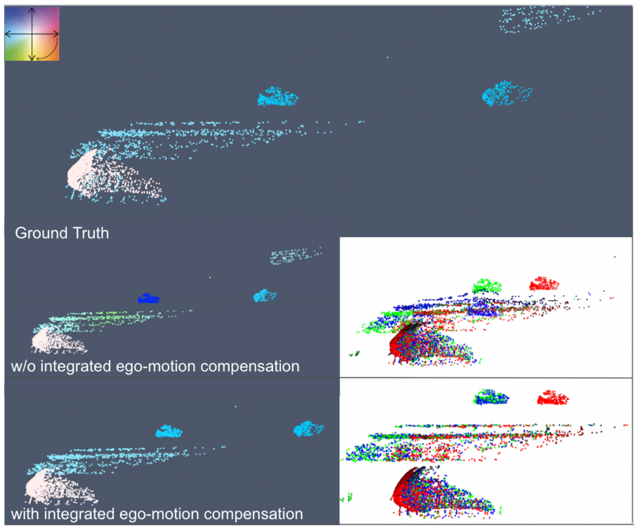
Ground Truth
w/o integrated ego-motion compensation
with integrated ego-motion compensation

Figure 8. **Effect of integrated ego-motion compensation within our objective function**: The transformation function aids in more accurate point cloud alignment, significantly reducing the likelihood of false positives, as demonstrated above.

References

[1] Brian Amberg, Sami Romdhani, and Thomas Vetter. Optimal step nonrigid icp algorithms for surface registration. 06 2007.

[2] James Bergstra, Brent Komer, Chris Eliasmith, Dan Yamins, and David D Cox. Hyperopt: a python library for model selection and hyperparameter optimization. *Computational Science Discovery*, 8(1):014008, 2015. 4

[3] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020. 1, 4

[4] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8748–8757, 2019. 4

[5] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017. 4

[6] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013. 4

[7] Itai Lang, Dror Aiger, Forrester Cole, Shai Avidan, and Michael Rubinstein. SCOOP: Self-Supervised Correspondence and Optimization-Based Scene Flow. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 3

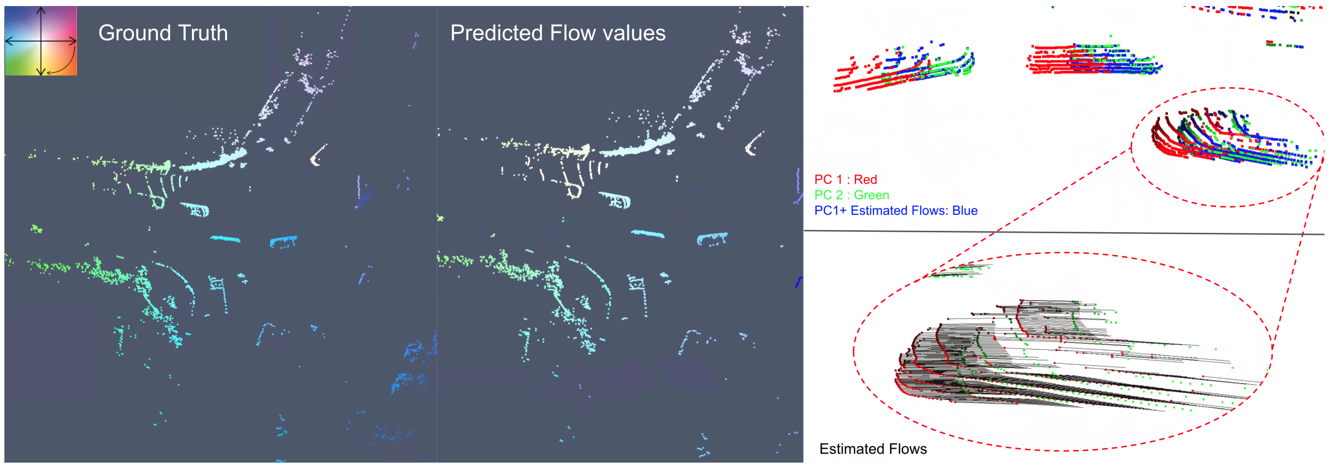

Figure 9. **Predicting Scene Flow in a Dynamic Scene**: This visualization depicts the estimated scene flow surrounding an ego vehicle. The left image uses color gradients to indicate flow velocity and direction. Similar color values depict similarity in flow values. The image on the right depicts estimated flow vectors (blue) onto the actual target point cloud data (green). The proximity of the predicted and target points demonstrates the accuracy of the scene flow model in this busy urban setting.

[8] Xueqian Li, Jhony Kaesemodel Pontes, and Simon Lucey. Neural scene flow prior. *Advances in Neural Information Processing Systems*, 34:7838–7851, 2021. 1, 4

[9] Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E Gonzalez, and Ion Stoica. Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*, 2018. 4

[10] Xingyu Liu, Charles R. Qi, and Leonidas J. Guibas. Flownet3d: Learning scene flow in 3d point clouds, 2019.

[11] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4040–4048, 2016. 4

[12] Himangi Mittal, Brian Okorn, and David Held. Just go with the flow: Self-supervised scene flow estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[13] Jhony Kaesemodel Pontes, James Hays, and Simon Lucey. Scene flow from point clouds with or without learning. In *2020 international conference on 3D vision (3DV)*, pages 261–270. IEEE, 2020. 3, 4, 5

[14] Wenxuan Wu, Zhi Yuan Wang, Zhuwen Li, Wei Liu, and Li Fuxin. Pointpwc-net: Cost volume on point clouds for (self-) supervised scene flow estimation. In *European Conference on Computer Vision*, pages 88–107. Springer, 2020.