Dynamic 3D Scene Analysis by Point Cloud Accumulation (Supplementary material)

Shengyu Huang[1] Zan Gojcic[2] Jiahui Huang[3]
Andreas Wieser[1] Konrad Schindler[1]

[1]ETH Zürich [2]NVIDIA [3]BRCist

In this supplementary document, we first present additional information about our network architecture and implementation in § A. We then elaborate on technical details of ego-motion estimation, iterative pose refinement, and scene reconstruction in § B. Precise definitions of loss functions and evaluation metrics are provided in § C. Further analysis of the two datasets, *nuScenes* and *Waymo*, is presented in § D, followed by additional quantitative results including scene flow estimation, instance association, and the TubeNet motion model in § E. Finally, we show more qualitative results in § F.

A Network and implementation

Network architecture. The detailed network architecture is depicted in Fig. S6. Our network is a sequential model consisting of (i) per-frame feature extraction used to estimate ego-motion, (ii) multi-frame feature extraction to segment dynamic objects and regress offset vectors towards the associated instance center, and (iii) TubeNet to regress the rigid motions of dynamic objects. The Pillar encoder and the two UNets operate without Batch Normalisation [5], this speeds up training and inference without any loss in performance [9]. Our network achieves flexibility w.r.t. the number of input frames via global max-pooling along the temporal dimension in the InitConv3D block (Fig. S6).

Implementation details. We use `torch_scatter`[1] to efficiently convert point-wise features to pillar/instance-level global features. To spatially align the backbone features we use `grid_sample`, implemented in PyTorch [8]. Before clustering, we apply voxel down-sampling implemented in TorchSparse [11] to reduce the point density and improve clustering efficiency. The voxel size is set to 15 cm. Instance labels at full resolution are recovered by indexing points to their associated voxel cell.

[1] https://github.com/rusty1s/pytorch_scatter

B Methodology

Ego-motion estimation. Given two sets $(\mathbf{P}^1, \mathbf{P}^t)$ of pillar centroid coordinates and associated L_2-normalised features $(\mathbf{F}^1_{\text{ego}}, \mathbf{F}^t_{\text{ego}})$, we first compute the cost matrix $\mathbf{M}^t = 2 - 2\langle \mathbf{F}^t_{\text{ego}}, \mathbf{F}^1_{\text{ego}} \rangle$ and an Euclidean distance matrix $\mathbf{D}^t_{l,m} = \|\mathbf{p}^t_l - \mathbf{p}^1_m\|_2$ from pillar coordinates. We then pad $\mathbf{M}^t$ with a learnable slack row and column to accommodate outliers, before iteratively alternating between row normalisation and column normalisation[2] for five times to approximate a doubly stochastic permutation matrix $\mathbf{S}^t$ that satisfies

$$\sum_{l=1}^{N_{\text{ego}}+1} \mathbf{S}^t_{l,m} = 1, \forall m = 1, ..., N_{\text{ego}}, \quad \sum_{m=1}^{N_{\text{ego}}+1} \mathbf{S}^t_{l,m} = 1, \forall l = 1, ..., N_{\text{ego}}, \quad \mathbf{S}^t_{l,m} \geq 0. \tag{S.1}$$

Here $\mathbf{S}^t_{l,m}$ represents the probability of $(\mathbf{p}^t_l, \mathbf{p}^1_m)$ being in correspondence. $\mathbf{p}^t_l$ is considered as an outlier and should be ignored during pose estimation if its slack column value $\mathbf{S}^t_{l,-1} \to 1$. We further mask $\mathbf{S}^t$ using a support matrix $\mathbf{I}^t$ computed from $\mathbf{D}^t$ as:

$$\mathbf{I}^t = \left(\mathbf{D}^t < s\right), \quad s = v \cdot \Delta t \,, \tag{S.2}$$

where v is the maximum speed and Δt is the interval between two frames. The final corresponding point $\phi(\mathbf{p}^t_l, \mathbf{P}^1)$ of $\mathbf{p}^t_l$ and its weight w^t_l are computed as

$$\phi(\mathbf{p}^t_l, \mathbf{P}^1) = (\mathbf{I}^t \odot \mathbf{D}^t)_{[l,:-1]} \mathbf{X}^1, \quad w^t_l = \sum_{m=1}^{N_{\text{ego}}} (\mathbf{I}^t \odot \mathbf{D}^t)_{l,m} \,, \tag{S.3}$$

with $\odot$ the Hadamard product. Eq (3) is solved with the Kabsch algorithm. For a detailed derivation, please refer to [3]. The value of v is dataset-specific, we set it to 30 m/s for the *Waymo*, respectively 10 m/s for *nuScenes*.

Iterative refinement of TubeNet estimates. To improve the estimation of the transformation parameters for dynamic objects, we unroll TubeNet for two iterations, as often done in point cloud registration [13,4]. Specifically, for a dynamic object $\mathbf{X}_k$, we first estimate the initial rigid transformation $\mathbf{T}^{0,t}_k$ of the t^{th} frame $\mathbf{X}^t_k$ following Eq (7). We then obtain the transformed points $\mathbf{X}^{t'}_k = \mathbf{T}^{0,t}_k \circ \mathbf{X}^t_k$. Next, we update the positional feature $\tilde{\mathbf{f}}^{t'}_{\text{pos}} = \text{PN}(\mathbf{X}^{t'}_k)$ and regress the residual transformation matrix $\mathbf{T}^{t,1}_k$ again, according to Eq (7). The final transformation is $\mathbf{T}^{t,1}_k \cdot \mathbf{T}^{t,0}_k$. For better stability during training, the gradients between the two iterations are detached. We assign higher weight to the latter iteration to improve accuracy. The overall loss $\mathcal{L}^{\bullet}_{\text{obj}}$ is:

$$\mathcal{L}^{\bullet}_{\text{obj}} = 0.7 \cdot \mathcal{L}^{\bullet,0}_{\text{obj}} + \mathcal{L}^{\bullet,1}_{\text{obj}} \tag{S.4}$$

[2] To improve training stability, row and column normalisations operate in log-space.

Scene reconstruction. To show the benefits of our method for downstream tasks, we use the points accumulated with different methods as a basis for 3D surface reconstruction, see Fig. S4 and S5. Specifically, we use the accumulated points as input to the Poisson reconstruction [6], implemented in Open3D [14]. To estimate point cloud normals, the neighborhood radius is set to 0.5 m and the maximum number of neighbors is set to 128. The depth for the Poisson method is set to 10, and the reconstructed meshes are filtered by removing vertices with densities below the 15^{th} percentile.

C Loss functions and evaluation metrics

C.1 Loss functions

Weighted BCE loss. To compensate for class imbalance, we use a weighted BCE and compute the weights of each class on the fly. Specifically, for a mini-batch with N_{pos} positive and N_{neg} negative samples, the associated weights w_{pos} and w_{neg} are computed as:

$$w_{\text{pos}} = \min(\sqrt{\frac{N_{\text{pos}} + N_{\text{neg}}}{N_{\text{pos}}}}, w_{\text{max}}) \qquad w_{\text{neg}} = \min(\sqrt{\frac{N_{\text{pos}} + N_{\text{neg}}}{N_{\text{neg}}}}, w_{\text{max}}) \ , \tag{S.5}$$

where w_{max} is the maximum weight of a class.[3] The final weighted BCE loss $\mathcal{L}_{\text{bce}}(\mathbf{x}, \overline{\mathbf{x}})$ is

$$\mathcal{L}_{\text{bce}}(\mathbf{x}, \overline{\mathbf{x}}) = \frac{1}{|\mathbf{x}|} \sum_{i=1}^{|\mathbf{x}|} w_i (\overline{\mathbf{x}}_i \log(\mathbf{x}_i) + (1 - \overline{\mathbf{x}}_i) \log(1 - \mathbf{x}_i)) \ , \tag{S.6}$$

with $\mathbf{x}$ and $\overline{\mathbf{x}}$ are predicted and ground truth labels, and w_i the weight of the i^{th} sample, computed as

$$w_i = \begin{cases} w_{\text{pos.}}, & \text{if } \overline{\mathbf{x}}_i = 1 \\ w_{\text{neg.}}, & \text{otherwise} \end{cases} \ . \tag{S.7}$$

Lovász-Softmax loss. The Jaccard index (ratio of Intersection over Union) is commonly used to measure segmentation quality. In the binary classification setting, we can set the ground truth labels as $\overline{\mathbf{x}}_i \in \{-1, 1\}$, then the Jaccard index of the foreground class J_1 is computed as

$$J_1(\overline{\mathbf{x}}, \mathbf{x}) = \frac{|\{\overline{\mathbf{x}} = 1\} \cap \{\text{sign}(\mathbf{x}) = 1\}|}{|\{\overline{\mathbf{x}} = 1\} \cup \{\text{sign}(\mathbf{x}) = 1\}|} \ , \quad J_1(\overline{\mathbf{x}}, \mathbf{x}) \in [0, 1] \ , \tag{S.8}$$

[3] We find that in some extreme cases, there are very few (< 10) positive samples and way more negative samples. We thus bound w_{max} at 50 to ensure stability.

with $\mathbf{x}$ the prediction and sign() the sign function. The corresponding loss $\Delta_{J_1}(\overline{\mathbf{x}}, \mathbf{x})$ to minimise the empirical risk is

$$\Delta_{J_1}(\overline{\mathbf{x}}, \mathbf{x}) = 1 - J_1(\overline{\mathbf{x}}, \mathbf{x}) \ . \tag{S.9}$$

However, this is not differentiable and cannot be directly employed as a loss function. The authors of [1] have proposed to optimise it using a Lovász extension. The Lovász extension $\dot{\Delta}$ of a set function Δ is defined as:

$$\dot{\Delta}(\mathbf{m}) = \sum_{i=1}^{p} \mathbf{m}_i \, g_i(\mathbf{m}) \ , \tag{S.10}$$

with

$$g_i(\mathbf{m}) = \Delta(\{\pi_1, \ldots, \pi_i\}) - \Delta(\{\pi_1, \ldots, \pi_{i-1}\}), \tag{S.11}$$

where π denotes a permutation that places the components of $\mathbf{m}$ in decreasing order. Considering $\mathbf{m}_i = \max(1 - \mathbf{x}_i \overline{\mathbf{x}}_i, 0)$, the Lovász-Softmax loss $\mathcal{L}_{ls}(\overline{\mathbf{x}}, \mathbf{x})$ is defined as

$$\mathcal{L}_{ls}(\overline{\mathbf{x}}, \mathbf{x}) = \dot{\Delta}_{J_1}(\mathbf{m}). \tag{S.12}$$

Inlier loss. Previous works [13,3] have observed that the entropy-regularized optimal transport [2] has a tendency to label most points as outliers. To alleviate this issue, we follow [13,3] and use an inlier loss $\mathcal{L}_{\text{inlier}}$ on the matching matrix $\mathbf{D}^t$, designed to encourage inliers. The inlier loss is defined as

$$\mathcal{L}^t_{\text{inlier}} = \frac{1}{2N_{\text{ego}}}(2N_{\text{ego}} - \sum_{l=1}^{N_{\text{ego}}} \sum_{m=1}^{N_{\text{ego}}} \mathbf{D}^t_{l,m}) \ . \tag{S.13}$$

C.2 Evaluation metrics

Instance association metrics. To quantitatively measure the spatio-temporal instance association quality, we report weighted coverage ($WCov$) as well as recall and precision at a certain threshold. Given the ground truth clusters $\mathcal{G}$ and the estimated clusters $\mathcal{O}$, recall measures the ratio of clusters in $\mathcal{G}$ that have an overlap above some threshold with a cluster in $\mathcal{O}$, while precision does the same in the opposite direction. Weighted coverage $WCov(\mathcal{G}, \mathcal{O})$ is computed as

$$WCov(\mathcal{G}, \mathcal{O}) = \sum_{i=1}^{|\mathcal{G}|} \frac{1}{|\mathcal{G}|} w_i \max_j \text{IoU}(r_i^G, r_j^0), \qquad w_i = \frac{|r_i^G|}{\sum_k |r_k^G|} \ , \tag{S.14}$$

where r_i^G and r_j^O are clusters from $\mathcal{G}$ and $\mathcal{O}$, and $\text{IoU}(r_i^G, r_j^O)$ denotes the overlap between two clusters.

ECDF. The Empirical Cumulative Distribution Function (ECDF) measures the distribution of a set of values:

$$\text{ECDF}(x) = \frac{\big|\{o_i < x\}\big|}{\big|O\big|} \ , \tag{S.15}$$

where $O = \{o_i\}$ is a set of samples and $x \in [\min\{O\}, \max\{O\}]$.

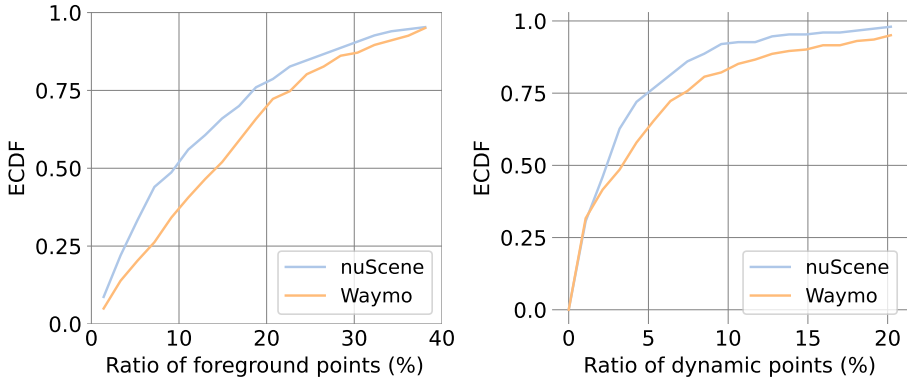

Fig. S1. *ECDF* curve of points lying on foreground objects and on dynamic objects, for both the *Waymo* and *nuScenes* datasets.

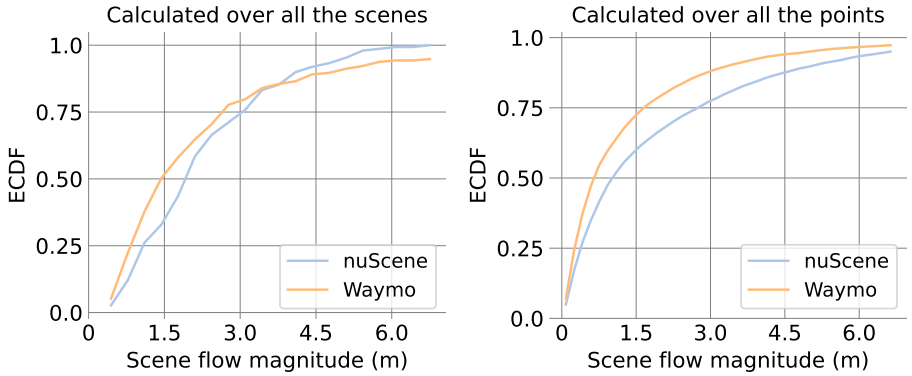

Fig. S2. Scene flow magnitudes of dynamic objects in the *Waymo* and *nuScenes* datasets.

D Dataset analysis

In total, we have 150, respectiverly 202 scenes as held-out test sets in *nuScenes* and *Waymo*. The *ECDF* curve of points belonging to foreground and dynamic objects are shown in Fig. S1. As can be seen, the ratios of foreground and dynamic points span a large range (40% and 20%). Recalling that the scene flow estimation performance of the dynamic parts falls far behind that of the static parts (Tab. 1), this large range of ratios of dynamic objects hints at different difficulties across the scenes. The median fractions of foreground points are 16.2%/9.4% in *Waymo*/*nuScenes*, the median fractions of points on moving objects are 3.5%/2.4%. In other words, roughly 75% of all foreground objects are static. This motivates our strategy to start with motion segmentation, so as to make explicit the large static component (including many objects that could move) whose scene flow is identical to the ego-motion.

Dataset	Method	Static part				Dynamic foreground			
		EPE avg.↓	AccR↑	AccS↑	ROutlier↓	EPE avg. ↓	AccR↑	AccS↑	ROutliers ↓
Waymo	PPWC-Net [12]	0.475 ± 0.543	35.0	14.2	13.5	0.658 ± 0.696	27.1	7.9	22.9
	FLOT [10]	0.381 ± 0.516	68.8	51.8	13.0	0.772 ± 0.711	30.1	11.2	31.9
	WsRSF [3]	1.415 ± 1.352	34.6	23.0	56.9	1.764 ± 1.744	21.0	8.6	61.6
	NSFPrior [7]	0.159 ± 0.231	87.1	73.5	4.3	0.355 ± 0.456	63.7	41.3	14.3
	Ours	**0.088 ± 0.237**	**91.6**	**81.9**	**2.3**	**0.169 ± 0.259**	**76.8**	**52.9**	**5.3**
nuScenes	PPWC-Net [12]	0.488 ± 0.402	34.2	12.7	17.5	0.784 ± 0.547	22.8	6.9	35.0
	FLOT [10]	0.597 ± 0.582	53.3	35.1	26.6	1.156 ± 0.714	13.2	3.7	56.5
	WsRSF [3]	0.658 ± 0.483	47.5	31.1	31.5	0.925 ± 0.627	29.8	15.0	42.2
	NSFPrior [7]	0.501 ± 0.344	57.8	37.7	21.3	0.743 ± 0.537	39.1	19.9	31.1
	Ours	**0.226 ± 0.206**	**72.3**	**46.7**	**7.4**	**0.394 ± 0.26**	**47.8**	**22.7**	**17.3**

Table S1. Scene flow estimation results on *Waymo* and *nuScenes* datasets. Numbers are averaged over all test scenes.

In Fig. S2, we show the *ECDF* curve of scene flow magnitudes (L_2-norm of scene flow vectors) for the dynamic portions of the two datasets. The motions span a large range, but 75% of the flow vectors are of moderate magnitude <3 m. *nuScenes* has slightly larger overall flow magnitudes than *Waymo*, but *Waymo* contains more instances of large motions (Fig. S2 (left)).

E Additional results

Results averaged over scenes. In Tab. 1 we report evaluation metrics calculated over all the points in the test set. However, this does not fully reveal the difficulties encountered in different scenes. Here, we first calculate evaluation metrics per scene, then report the average over scenes in Tab. S1. For *EPE avg.*, we additionally report the standard deviations. We can see that for both static and dynamic parts, all methods have large standard deviations, which indicates varying difficulty of the scenes, as well as gross errors from challenging samples. Our model still achieves the smallest flow errors and standard deviations under this evaluation setting, for both datasets .

Spatio-temporal instance association. We plot instance association metrics at different thresholds in Fig. S3. As can be seen, offset prediction improves association recall and precision by >5%, across a range of thresholds. Such improvement becomes more significant as one increases the IoU threshold, reaching ≈10% at $IoU = 0.9$. We conclude that offset prediction is important to retain high-quality spatio-temporal instances, which can subsequently improve the accuracy of motion modelling for the dynamic parts (*AccS* increases by 9.2% in Tab. 4).

Dynamic object motion modelling. We additionally compare the proposed TubeNet to two baseline methods. We naively align each frame $\mathbf{X}_k^t$ ($t > 1$) of an instance $\mathbf{X}_k$ to frame $\mathbf{X}_k^0$ by translating the centroids, and term this method

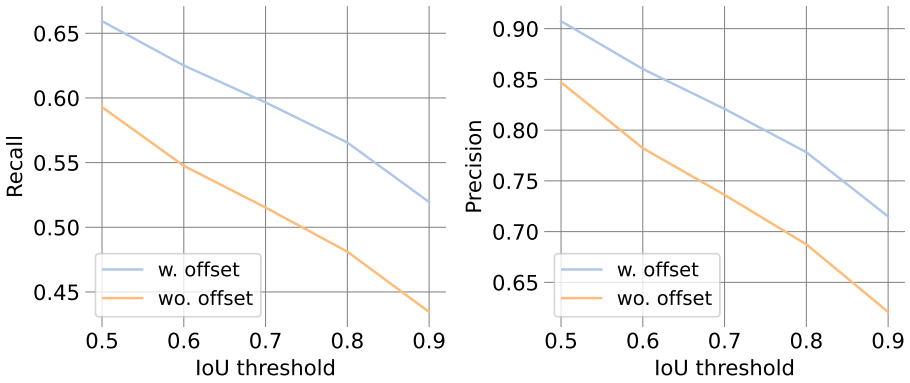

Fig. S3. Spatio-temporal instance association performance on *Waymo* with and without offset prediction.

		EPE avg.↓	EPE med.↓	AccS↑	AccR↑	ROutliers↓
Waymo	center	0.265	0.095	36.9	62.9	9.9
	center + ICP	0.212	0.047	61.2	80.0	7.7
	Ours	0.197	0.062	53.3	77.5	5.9
	Ours + ICP	**0.173**	**0.043**	**69.1**	**86.9**	**5.1**
nuScenes	center	0.553	0.258	13.5	32.7	28.2
	center + ICP	0.525	0.179	23.8	43.7	25.5
	Ours	0.301	0.146	26.6	53.4	**12.1**
	Ours + ICP	**0.301**	**0.135**	**32.7**	**56.7**	13.7

Table S2. Comparison to centroid-based motion estimation baseline.

center. For *center+ICP* we refine the simple translational alignment by a subsequent ICP. The detailed comparison is shown in Tab. S2. Our learned TubeNet achieves the best performance on both datasets. The improvement is larger on the challenging *nuScenes* data, where the point clouds are sparser and less complete, so centroids computed from partial observations are not an accurate proxy for the object location. Our learned TubeNet can implicitly exploit prior knowledge about object shape and surface-level correspondence, leading to more robust and accurate motion modelling.

F Qualitative results

We show additional qualitative results in Fig. S4 and Fig. S5. Benefiting from the explicit *multi-body* assumption, our model achieves accurate scene flow estimation of both static parts (Fig. S4 (1) and Fig. S5 (2)) and dynamic parts (Fig. S4 (3) and Fig. S5(1)). Errors in the automatically generated pseudo-ground truth

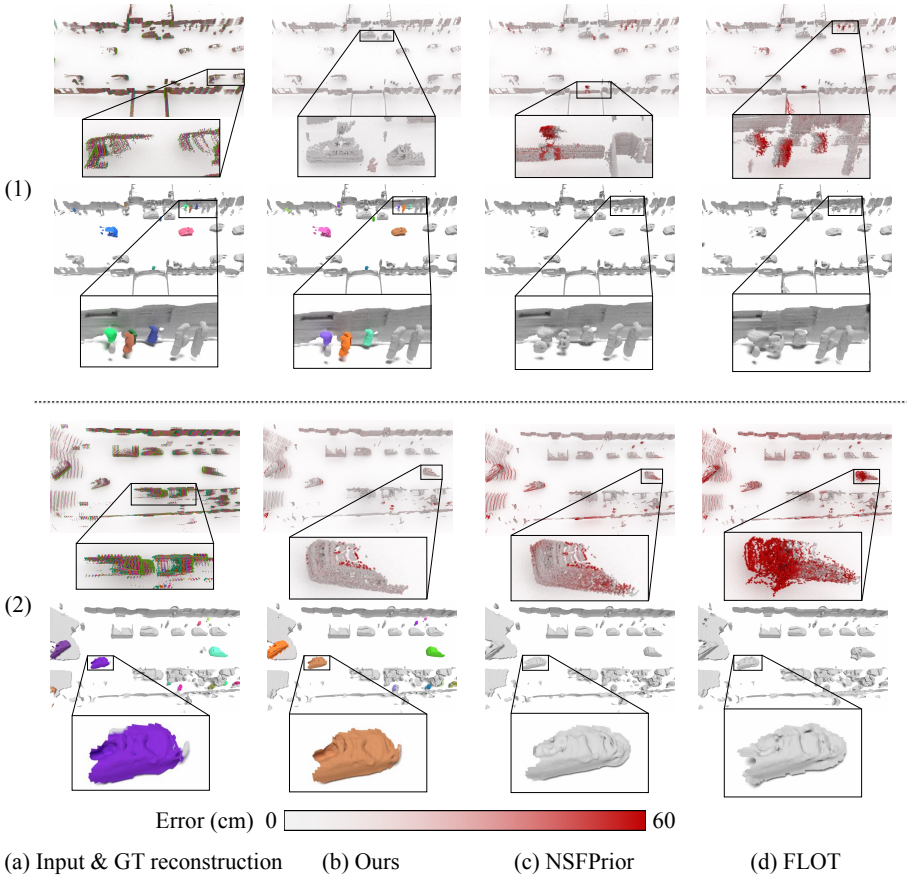

Fig. S4. Qualitative results showing scene flow estimation (top) and surface reconstruction (bottom) for three example scenes from the *Waymo* dataset.

are shown in Fig. S5(3), in this case our model achieves more accurate flow estimation and reconstruction.

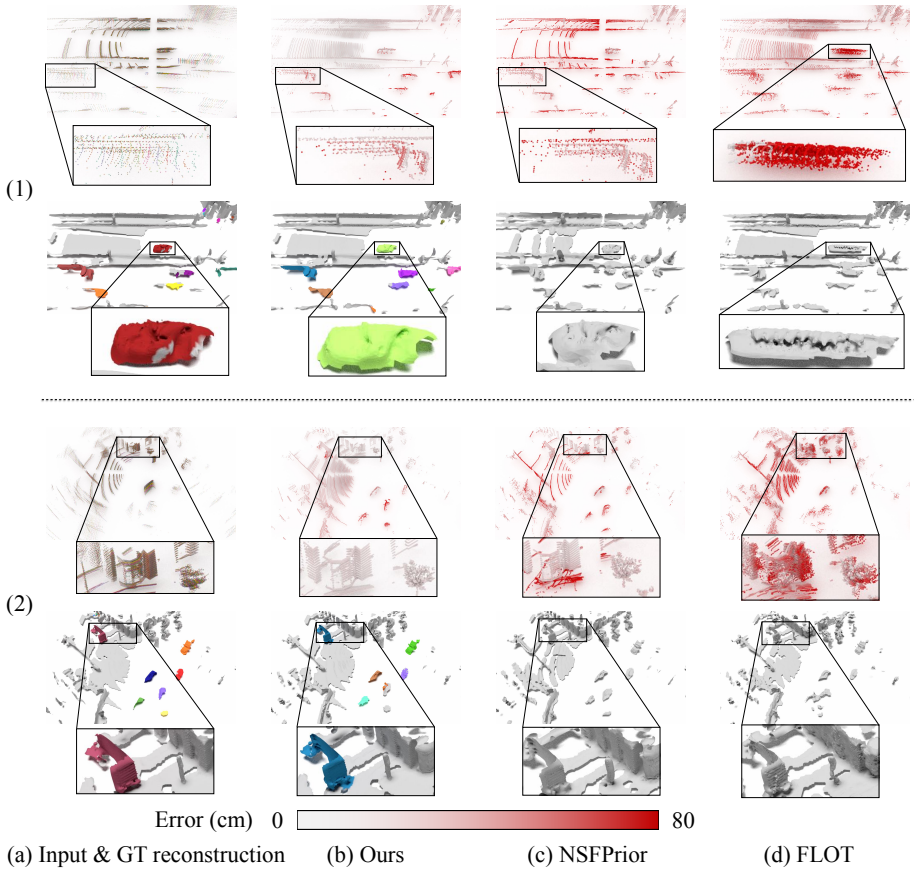

Fig. S5. Qualitative results showing scene flow estimation (top) and surface reconstruction (bottom) for three example scenes from the *nuScenes* dataset.

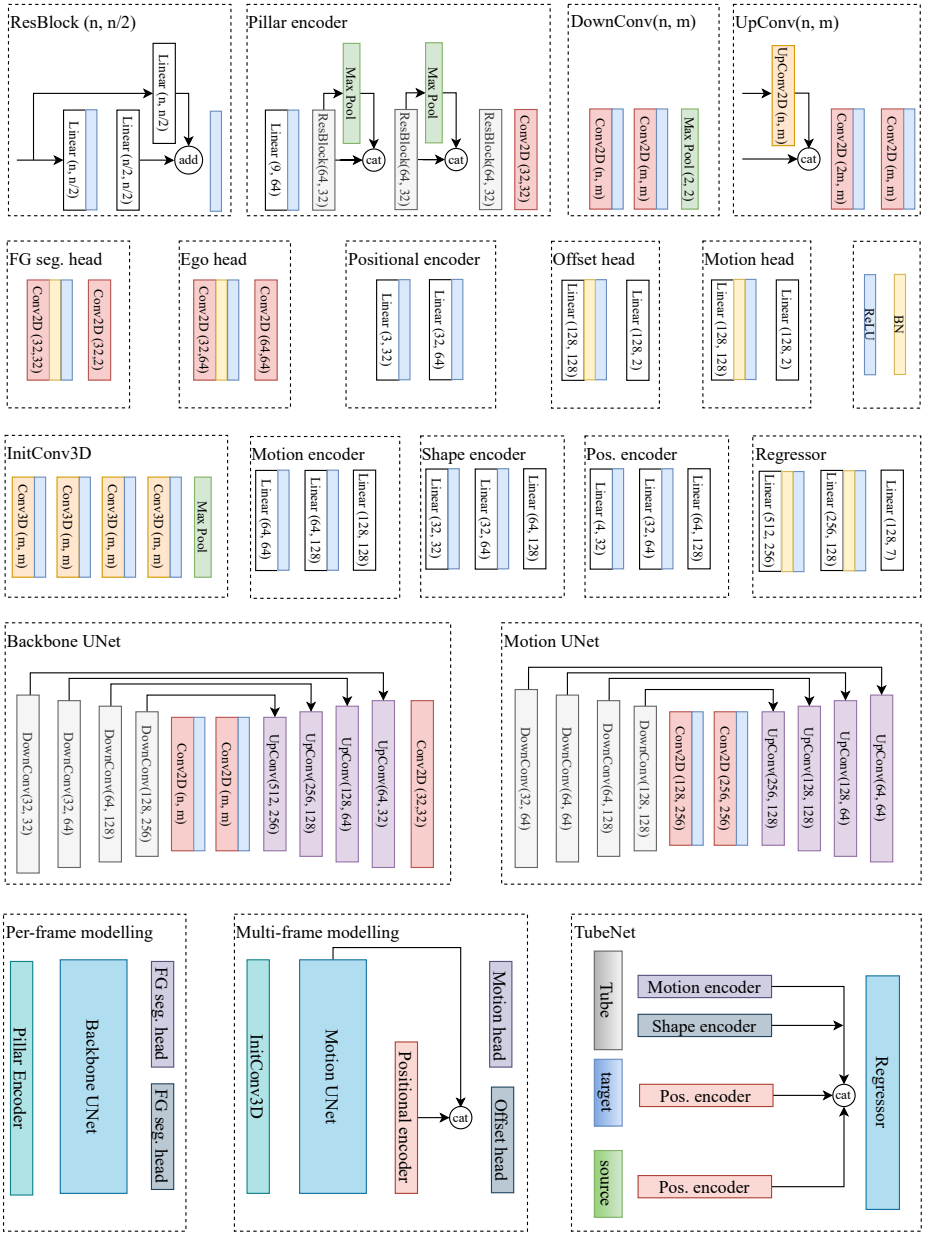

Fig. S6. Detailed network architecture. All convolutional layers have kernel size 3×3. (n, m) in Conv, UpConv, DownConv, and Linear layers denote the input and output feature dimensions.

References

1. Berman, M., Triki, A.R., Blaschko, M.B.: The Lovász-Softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks. In: Proc. CVPR (2018) 4
2. Cuturi, M.: Sinkhorn distances: Lightspeed computation of optimal transport. Proc. NeurIPS (2013) 4
3. Gojcic, Z., Litany, O., Wieser, A., Guibas, L.J., Birdal, T.: Weakly supervised learning of rigid 3d scene flow. In: Proc. CVPR (2021) 2, 4, 6
4. Gojcic, Z., Zhou, C., Wegner, J.D., Guibas, L.J., Birdal, T.: Learning multiview 3d point cloud registration. In: Proc. CVPR (2020) 2
5. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Proc. ICML (2015) 1
6. Kazhdan, M., Bolitho, M., Hoppe, H.: Poisson surface reconstruction. In: Eurographics (2006) 3
7. Li, X., Kaesemodel Pontes, J., Lucey, S.: Neural scene flow prior. Proc. NeurIPS (2021) 6
8. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. Proc. NeurIPS (2021) 1
9. Peng, S., Niemeyer, M., Mescheder, L., Pollefeys, M., Geiger, A.: Convolutional occupancy networks. In: Proc. ECCV (2020) 1
10. Puy, G., Boulch, A., Marlet, R.: FLOT: Scene flow on point clouds guided by optimal transport. In: Proc. ECCV (2020) 6
11. Tang, H., Liu, Z., Li, X., Lin, Y., Han, S.: TorchSparse: Efficient Point Cloud Inference Engine. In: MLSys (2022) 1
12. Wu, W., Wang, Z.Y., Li, Z., Liu, W., Fuxin, L.: Pointpwc-net: Cost volume on point clouds for (self-) supervised scene flow estimation. In: Proc. ECCV (2020) 6
13. Yew, Z.J., Lee, G.H.: RPM-Net: Robust point matching using learned features. In: Proc. CVPR (2020) 2, 4
14. Zhou, Q.Y., Park, J., Koltun, V.: Open3D: A modern library for 3D data processing. arXiv preprint arXiv:1801.09847 (2018) 3