DVLO: Deep Visual-LiDAR Odometry with Local-to-Global Feature Fusion and Bi-Directional Structure Alignment (Supplementary Materials)

Jiuming Liu[1], Dong Zhuo[1], Zhiheng Feng[1], Siting Zhu[1]
Chensheng Peng[2], Zhe Liu[3], and Hesheng Wang[1]*

[1] Department of Automation, Shanghai Jiao Tong University
[2] University of California, Berkeley
[3] MoE Key Lab of Artificial Intelligence, Shanghai Jiao Tong University
{liujiuming, paryi555, wanghesheng}@sjtu.edu.cn

In this supplementary material, we provide additional details about our network and show more experimental results of the proposed methods. In Sec. 1, we illustrate more implementation details of our network, including the feature extraction pyramid and our Local-to-Global Fusion (LoGo Fusion) module. In Sec. 2, we give the details about data augmentation, coordinate transformation, and fusion mask. In Sec. 3, we provide a graphical demonstration of our LoGo Fusion. In Sec. 4, we compare our method with other deep visual odometry methods on the KITTI odometry dataset [1]. In Sec. 5, we show more trajectory visualization results on KITTI 00-10 sequences.

1 Network Settings

As illustrated in Tab. 1 and Tab. 2, more implementation details of our network are shown. Tab. 1 includes settings of our point and image feature extraction pyramid in each layer. For our Local-to-Global fusion (LoGo Fusion), we retain the feature channels of extracted point features $F_P \in \mathbb{R}^D$ and image features $F_I \in \mathbb{R}^C$ to be the same ($C = D$). The extracted features are then fed into our LoGo Fusion module. Therefore, the "input dim" of our Local and Global Fuser should be the same as the channels of extracted features as shown in Tab. 2.

2 Data Preprocessing

Data Augmentation. Far points at the edge of point cloud frames are commonly considered outliers, which undermine the final accuracy. Thus, we filter these outliers and maintain points within a 30m $\times$ 30m square space before the point feature extraction. For the robustness of the network, we also augment the training dataset with a matrix T_{aug}, which is generated by random rotation

* Corresponding author. The first two authors contribute equally.

Table 1: Implementation details of our feature extraction pyramid. In the point feature extraction pyramid, 'N' indicates the number of selected neighbor points in down-sampling. 'MLP' represents the output dimension of each MLP layer. 'Sample rate' is the stride in point feature down-sampling. In the image feature extraction pyramid, 'Stride' includes the stride sizes in each 2D convolutional layer. We set the feature channels of extracted point features and image features to be the same.

	Layer	N	MLP	Sample rate
Point Feature Extraction Pyramid	Layer 0	32	[8,8,16]	1/32
	Layer 1	32	[16,16,32]	1/4
	Layer 2	16	[32,32,64]	1/4
	Layer 3	16	[64,64,128]	1/2
	Layer	Input Dim	Output Dim	Stride
Image Feature Extraction Pyramid	Layer 0	3	16	[2,3]
	Layer 1	16	32	[1,2]
	Layer 2	32	64	[1,2]
	Layer 3	64	128	[1,2]

Table 2: Implementation details of our LoGo Fusion. 'Input dim' and 'Output dim' respectively represent the input and output channels.

	Layer	Input Dim	Output Dim
Local Fuser	Layer 0	16	16
	Layer 1	32	32
	Layer 2	64	64
	Layer 3	128	128
Global Fuser	Layer 0	16	16
	Layer 1	32	32
	Layer 2	64	64
	Layer 3	128	128

angles following a Gaussian distribution. Finally, the input point clouds PC are as follows:

$$PC_f = T_{aug} \cdot PC \ (where\, x < 30m,\, y < 30m)\,, \tag{1}$$

where PC is the raw point cloud from the KITTI training dataset. PC_f is the final point cloud. The final point cloud is then fed into the network for point feature extraction.

Coordinate Transformation. In order to fuse the underlying multi-modal information and merge corresponding image features in a local region, we need to transfer the raw point cloud coordinates to the image coordinate system to obtain 2D coordinates Y for the feature gathering. In the transformation process, we calculate the transformation matrix T_{l2c} between the LiDAR and camera

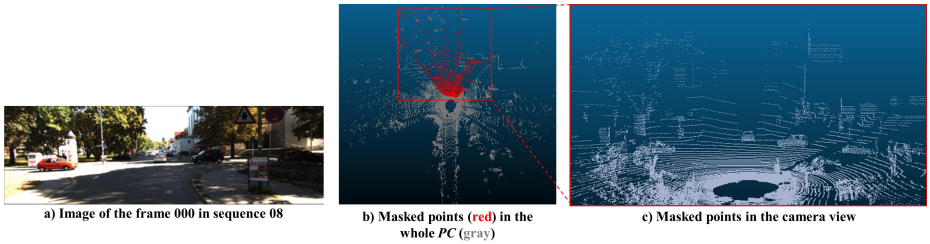

Fig. 1: The visualization of the masked points in the whole point cloud PC. a) is the image captured from cameras. b) shows the masked points (colored **red**) in the whole point cloud PC (colored **gray**). c) is the masked points in the camera view.

coordinate systems by utilizing the camera intrinsic and extrinsic matrices:

$$T_{l2c} = R_{r2c} \cdot T_{l2r}, \tag{2}$$

where T_{l2r} represents the transformation matrix from LiDAR to reference camera coordinates. R_{r2c} is the rotation matrix from the reference camera coordinate system to the left camera coordinate system. We then apply T_{l2c} to the raw point cloud coordinates to obtain the 3D coordinates X in the camera coordinate system. After the transformation, we project X onto the image plane to obtain the 2D coordinates Y by utilizing the projection matrix P_{c2i}:

$$Y = P_{c2i} \cdot X. \tag{3}$$

Fusion Mask. However, there is a large difference in the range that can be captured by the camera and LiDAR sensors. Cameras can only acquire images within a limited range, while LiDAR can capture information from 360° in the horizontal direction around it, resulting in a large number of points not being able to be projected onto the image plane. In order to fully utilize the visual-LiDAR information and preserve the 3D structure of the raw point cloud as much as possible, we adopt divide and conquer strategy. Specially, we design a binary fusion mask M, which can indicate which point can be projected onto the image plane as:

$$M = \begin{cases} 1, & 0 < x' < W_I,\ 0 < y' < H_I \\ 0, & \text{otherwise} \end{cases}, \tag{4}$$

where x', y' are 2D coordinates projected from LiDAR in the image coordinate system. W_I, H_I are the width and height of the corresponding image. We utilize this mask to indicate which point can be projected onto the image plane. We only perform feature fusion on these points. For those which can not be projected onto the image plane, we retain their original extracted features. The visualization of the masked points in the whole point cloud PC is shown in Fig. 1.

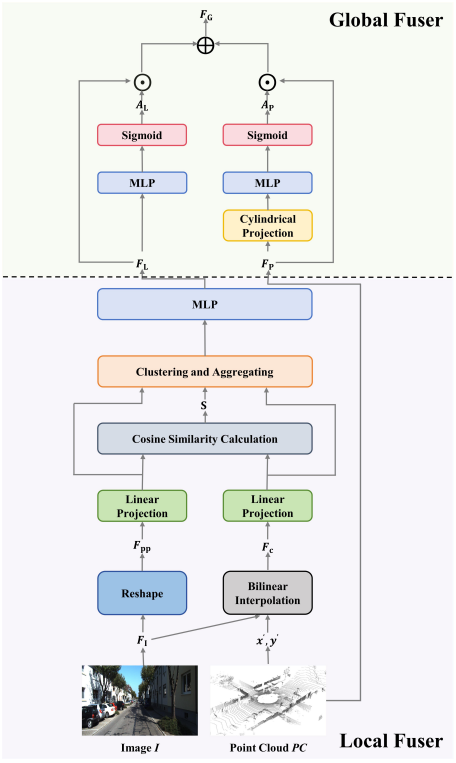

Fig. 2: Architecture of our LoGo Fusion. Our LoGo Fusion consists of two main components: Local Fuser and Global Fuser. In Local Fuser, the image (pseudo points) feature will be clustered and aggregated based on the cosine similarities. An MLP layer is applied to the aggregated features to generate the local fused features F_L. In Global Fuser, adaptive weights A_P and A_L are obtained from F_P and F_L through MLP and sigmoid activation layers. Finally, we aggregate F_P and F_L with A_P and A_L by element-wise product $\odot$ and weighted sum $\oplus$ to generate the global fused features F_G.

3 Architecture of LoGo Fusion

In this section, we provide a graphical demonstration of our Local-to-Global Fusion (LoGo Fusion) module. As illustrated in Fig. 2, our LoGo Fusion consists of two main components: Local Fuser and Global Fuser. Given point features F_P, their corresponding 2D coordinates x', y' in the image coordinate system and image features F_I, we firstly reshape F_I as a collection of pseudo points F_{pp}. Then, we view points as the cluster centers and compute the center features F_c by the bilinear interpolation on F_I based on x', y'. We divide all pseudo points into several clusters according to the pair-wise cosine similarity matrix S between center features F_c and pseudo point features F_{pp}. Finally, we aggregate all pseudo point features F_{pp} within the same cluster based on the similarities to the cluster center F_c, dynamically obtaining the local fused features F_L. F_L is then fed into

Table 3: Comparison with different deep visual odometry methods on the KITTI odometry dataset [1]. t_{rel} and r_{rel} mean the average sequence translational RMSE (%) and the average sequence rotational RMSE (°/100m) respectively on 03, 04, 05, 06, 07 and 10 sequences in the length of 100, 200, ..., 800m. The best results for each sequence are **bold**, and the second best results are underlined.

Method	03		04		05		06		07		10		Mean	
	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}
DeepVO [5]	8.49	6.90	7.19	7.00	2.62	3.60	5.42	5.80	3.91	4.60	8.11	8.80	5.96	6.00
ESP-VO [6]	6.72	6.50	6.33	6.10	3.35	4.90	7.24	7.30	3.52	5.00	9.77	10.20	6.16	6.60
GFS-VO [7]	5.44	3.30	2.91	1.30	3.27	1.60	8.50	2.70	3.37	2.30	6.32	2.30	4.97	2.30
GFS-VO-RNN [7]	6.36	3.60	5.95	2.40	5.85	2.60	14.58	5.00	5.88	2.60	7.44	3.20	7.68	3.20
BRNN [9]	4.74	3.10	3.90	2.20	6.02	2.70	9.59	2.40	5.28	2.30	7.04	3.30	6.10	2.60
BeyondTracking [8]	3.32	2.10	2.96	1.80	2.59	1.30	4.92	1.90	3.07	1.80	3.94	1.70	3.47	1.80
DAVO [2]	5.50	2.70	6.03	2.40	2.28	1.10	4.19	2.70	4.11	2.60	4.26	1.70	4.40	2.20
DeepAVO [10]	3.64	1.90	3.88	0.60	2.57	1.20	4.96	1.30	3.36	2.20	5.49	2.50	3.98	1.20
Miao *et al.* [3]	1.85	0.85	1.40	1.10	1.79	1.10	1.67	0.92	2.55	1.50	1.87	0.92	1.86	1.10
Ours	**0.62**	**0.49**	**0.69**	**0.34**	**0.75**	**0.37**	**1.16**	**0.47**	**0.56**	**0.33**	**1.26**	**0.59**	**0.84**	**0.43**

Table 4: Robustness comparison for sensor noise and asymmetric sensor degradation.

Method	Settings	07		08		09		10		Mean(07-10)	
		t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}
EfficientLO [4]	None	0.37	0.26	1.22	0.48	0.87	0.38	0.91	0.50	0.86	0.41
Ours	None	0.46	0.33	1.09	0.44	0.85	0.36	0.88	0.46	0.82	0.41
EfficientLO [4]	Add noise	0.41	0.29	1.35	0.54	0.97	0.43	1.12	0.52	0.96(+12%)	0.44(+7%)
Ours	Add noise	0.46	0.32	1.14	0.47	0.91	0.38	0.93	0.51	0.86(+5%)	0.42(+2%)
EfficientLO [4]	Drop 50 Frames	0.45	0.37	1.40	0.56	0.98	0.43	2.21	0.76	1.26(+47%)	0.53(+29%)
Ours	Drop 50 Frames	0.53	0.36	1.23	0.51	0.96	0.39	1.01	0.51	0.93(+13%)	0.44(+7%)

the Global Fuser with F_P. In Global Fuser, we convert the sparse LiDAR points to structured pseudo images by the cylindrical projection. Then we apply MLP layers and sigmoid activation function on both point (pseudo image) features F_P and local fused features F_L to obtain their corresponding adaptive weights A_P, A_L, respectively. Finally, we aggregate F_P and F_L by element-wise product and weighted sum to generate the global fused features F_G.

4 Additional Comparison Experiments

Comparison with deep visual odometry methods. Since many deep visual odometry methods [2, 3, 5–10] are trained on 00, 02, 08 and 09 sequences and tested on 03, 04, 05, 06, 07 and 10 sequences, we also train our DVLO accordingly and compare our performance with them on the same testing sequences. The results are shown in Tab. 3, demonstrating that our DVLO which utilizes visual-LiDAR information outperforms all these pure visual odometry works on all testing sequences. Compared with Miao *et al.*'s work [3], our method's mean errors t_{rel} and r_{rel} on sequences 03, 04, 05, 06, 07, and 10 sequences have a 53.8% and a 60.0% decline, respectively. The gratifying results show the effectiveness of fusing multi-modal information to enhance the accuracy of pose estimation.

Robustness to noise and asymmetric sensor degradation. Compared with the single-modality method [4], our multi-modal odometry has better robustness for noise and sensor drop as in Tab. 4. We add the same Gaussian noise and sensor drop (remove consecutive 50 frames of each sequence) to LiDAR here.

Also, another motivation is sensor complementarity since VO commonly fails in texture-less regions and LO is highly influenced by fog and snow.

5 The visualization of Trajectories on KITTI

Although we have shown some of our trajectory visualization results, we give a more comprehensive visualization of all sequences. Firstly, the 2D trajectory of our network has been visualized in Fig. 3. We compare our 2D trajectories with the ground-truth, which demonstrates that our trajectory can track the 2D ground-truth path well on all sequences.

Additionally, we also compare the 3D trajectories of our network and the ground truth in Fig. 4. It's shown that our odometry has high accuracy and can track the 3D ground-truth path well.

References

1. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: The kitti dataset. The International Journal of Robotics Research **32**(11), 1231–1237 (2013)
2. Kuo, X.Y., Liu, C., Lin, K.C., Lee, C.Y.: Dynamic attention-based visual odometry. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. pp. 36–37 (2020)
3. Miao, Y., Deng, H., Jiang, C., Feng, Z., Wu, X., Wang, G., Wang, H.: Pseudo-lidar for visual odometry. IEEE Transactions on Instrumentation and Measurement (2023)
4. Wang, G., Wu, X., Jiang, S., Liu, Z., Wang, H.: Efficient 3d deep lidar odometry. IEEE Transactions on Pattern Analysis and Machine Intelligence **45**(5), 5749–5765 (2022)
5. Wang, S., Clark, R., Wen, H., Trigoni, N.: Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. In: 2017 IEEE international conference on robotics and automation (ICRA). pp. 2043–2050. IEEE (2017)
6. Wang, S., Clark, R., Wen, H., Trigoni, N.: End-to-end, sequence-to-sequence probabilistic visual odometry through deep neural networks. The International Journal of Robotics Research **37**(4-5), 513–542 (2018)
7. Xue, F., Wang, Q., Wang, X., Dong, W., Wang, J., Zha, H.: Guided feature selection for deep visual odometry. In: Asian Conference on Computer Vision. pp. 293–308. Springer (2018)
8. Xue, F., Wang, X., Li, S., Wang, Q., Wang, J., Zha, H.: Beyond tracking: Selecting memory and refining poses for deep visual odometry. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 8575–8583 (2019)
9. Xue, F., Wang, X., Wang, Q., Wang, J., Zha, H.: Visual odometry with deep bidirectional recurrent neural networks. In: Pattern Recognition and Computer Vision: Second Chinese Conference, PRCV 2019, Xi'an, China, November 8–11, 2019, Proceedings, Part III 2. pp. 235–246. Springer (2019)
10. Zhu, R., Yang, M., Liu, W., Song, R., Yan, B., Xiao, Z.: Deepavo: Efficient pose refining with feature distilling for deep visual odometry. Neurocomputing **467**, 22–35 (2022)

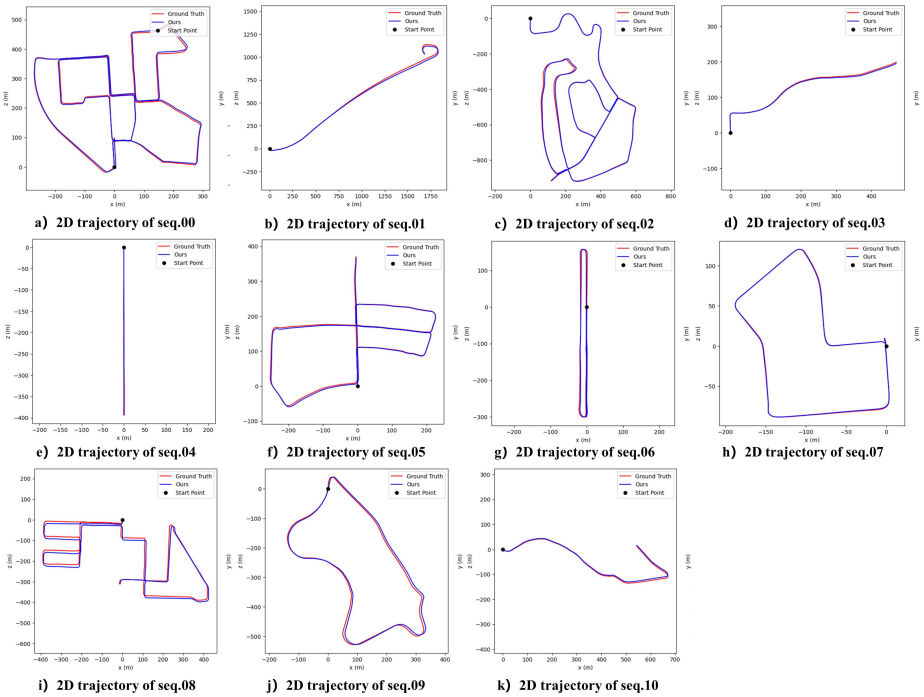

Fig. 3: The 2D trajectory of ground-truth and ours. Comprehensive 2D trajectory results are shown here on 00-10 sequences.

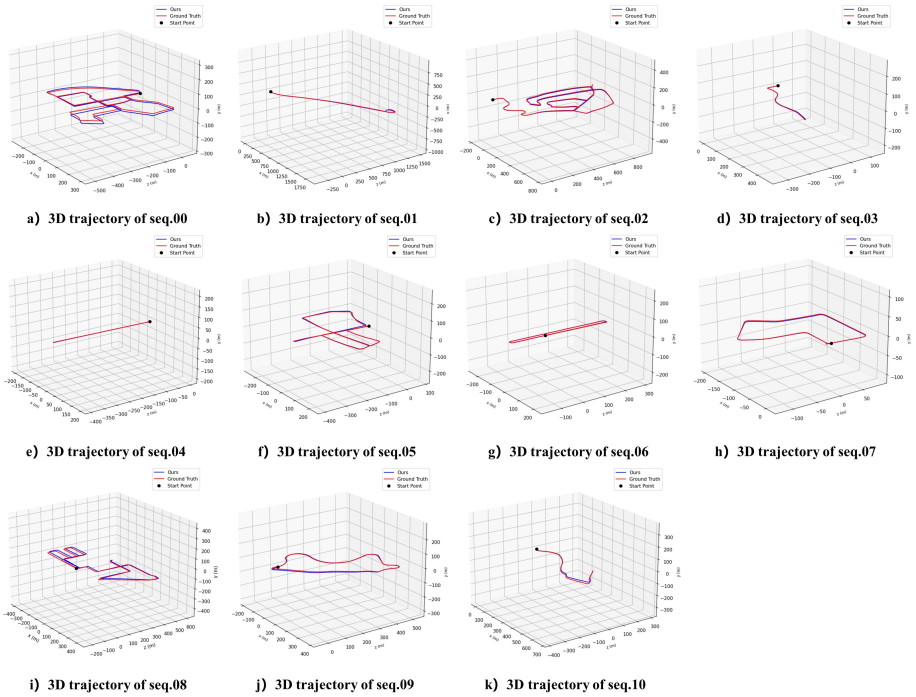

Fig. 4: The 3D trajectory of ground-truth and ours. Comprehensive 3D trajectory results are shown here on 00-10 sequences.