

# DSLO: Deep Sequence LiDAR Odometry Based on Inconsistent Spatio-temporal Propagation

Huixin Zhang<sup>†1</sup>, Guangming Wang<sup>†2</sup>, Xinrui Wu<sup>1</sup>, Chenfeng Xu<sup>3</sup>,  
 Mingyu Ding<sup>3</sup>, Masayoshi Tomizuka<sup>3</sup>, Wei Zhan<sup>3</sup>, and Hesheng Wang<sup>1</sup>

**Abstract**—This paper introduces a 3D point cloud sequence learning model based on inconsistent spatio-temporal propagation for LiDAR odometry, termed DSLO. It consists of a pyramid structure with a spatial information reuse strategy, a sequential pose initialization module, a gated hierarchical pose refinement module, and a temporal feature propagation module. First, spatial features are encoded using a point feature pyramid, with features reused in successive pose estimations to reduce computational overhead. Second, a sequential pose initialization method is introduced, leveraging the high-frequency sampling characteristic of LiDAR to initialize the LiDAR pose. Then, a gated hierarchical pose refinement mechanism refines poses from coarse to fine by selectively retaining or discarding motion information from different layers based on gate estimations. Finally, temporal feature propagation is proposed to incorporate the historical motion information from point cloud sequences, and address the spatial inconsistency issue when transmitting motion information embedded in point clouds between frames. Experimental results on the KITTI odometry dataset and Argoverse dataset demonstrate that DSLO outperforms state-of-the-art methods, achieving at least a 15.67% improvement on RTE and a 12.64% improvement on RRE, while also achieving a 34.69% reduction in runtime compared to baseline methods. Our implementation will be available at <https://github.com/IRMVLab/DSLO>.

## I. INTRODUCTION

LiDAR odometry is a pivotal task in the realm of autonomous navigation [1], [2]. Over the past decade, traditional geometry-based methods have been the cornerstone of LiDAR odometry, providing robust interpretability and operational efficiency [3], [4], [5], [6]. However, ideal assumptions in traditional methods can lead to inaccurate system modeling. With advancements in computational hardware, focus has shifted towards leveraging deep learning techniques to tackle LiDAR odometry challenges. Recent works [7], [8], [9], [10], [11], [12] have explored learning deep feature representations or directly estimating vehicle motion through end-to-end training.

<sup>†</sup>The first two authors contributed equally.

This work was supported in part by the Natural Science Foundation of China under Grant 62073222, U21A20480 and U1913204, in part by the Science and Technology Commission of Shanghai Municipality under Grant 21511101900, in part by the Open Research Projects of Zhejiang Lab under Grant 2022NB0AB01. Corresponding Author: Hesheng Wang.

<sup>1</sup>Department of Automation, Key Laboratory of System Control and Information Processing of Ministry of Education, Key Laboratory of Marine Intelligent Equipment and System of Ministry of Education, Shanghai Intelligent Research Center of Intelligent Control and Management, Shanghai Jiao Tong University, Shanghai 200240, China.

<sup>2</sup>Department of Engineering, University of Cambridge, Cambridge CB2 1PZ, U.K.

<sup>3</sup>UC Berkeley, Berkeley, CA 94720 USA.

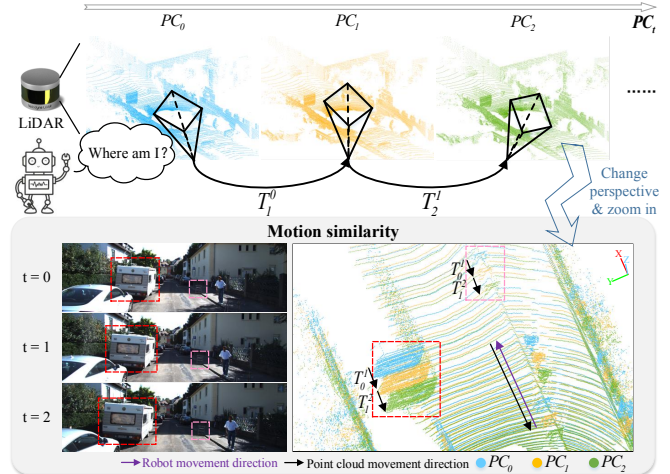


Fig. 1: Inspiration for our work. Boxes of the same color indicate the same rigid object. The robot’s motion can be inferred from these objects, showing high similarity between adjacent frames.

However, learning-based LiDAR odometry faces three main challenges: 1) Projecting unstructured point clouds onto 2D planes [7], [8], [9] results in loss of 3D spatial information and efficiency. 2) Coarse-to-fine optimization [11], [12] fails to account for varying information reliability across scales. 3) Most methods predict poses using only two adjacent frames, ignoring historical motion information.

To address these, we propose DSLO, an end-to-end deep sequence LiDAR odometry method leveraging inconsistent spatio-temporal propagation. Compared to the baseline method PWCLO-Net [11], DSLO accelerates inference with spatial information reuse and sequential pose initialization modules. Gated hierarchical pose refinement is proposed to update features from coarse to fine, using self-learning gate estimations to mitigate incorrect historical matches in the upper layer and sensor noise in the current layer. Furthermore, we explore temporal information fusion in LiDAR odometry for better pose regression. The challenge of modeling temporal motion information in unstructured point clouds and propagating them within inconsistent spatial contexts is addressed. The contributions are as follows:

- Sequential pose initialization is proposed to reduce computation overhead while retaining motion similarity.
- Gated hierarchical pose refinement is proposed, which utilizes multi-scale spatial information for hierarchical updates and employs self-learning gate estimations to filter valid information from different layers.

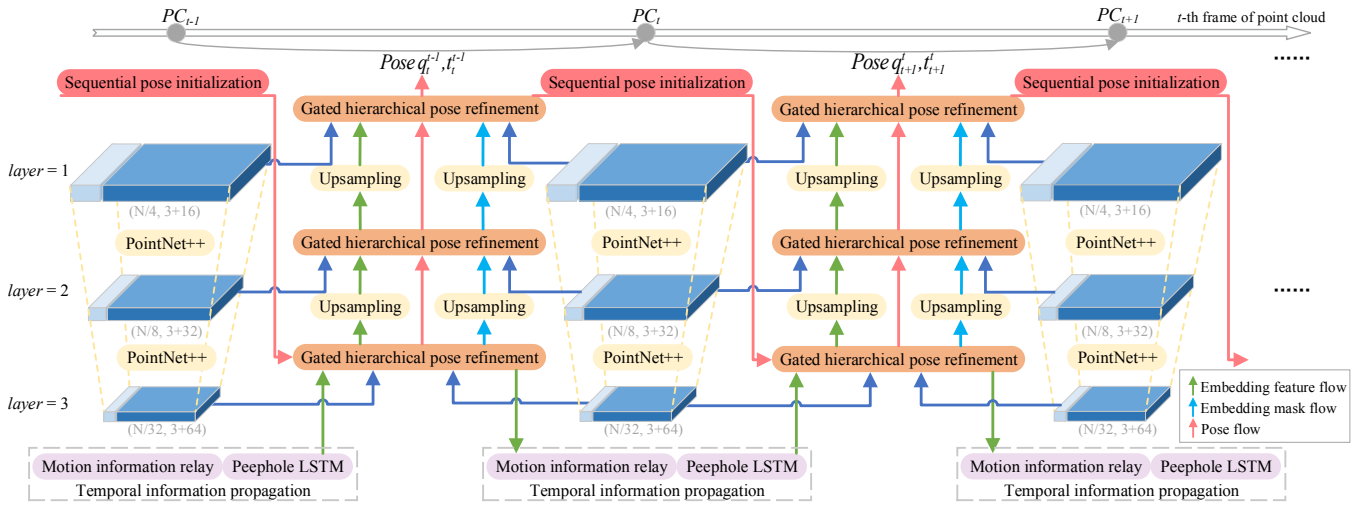


Fig. 2: Overview of our DSLO. For pose estimation between two adjacent frames, we encode the point feature pyramid and use a gated hierarchical pose refinement module to achieve coarse-to-fine update. For multiple frames, we reuse the feature pyramid and utilize the last refined pose as the current initial guess. Temporal feature propagation fuses motion features along time series and addresses the spatial inconsistency of point-wise features between frames.

- Temporal feature propagation is proposed to fuse information across time series. Motion information relay is designed to address the spatial inconsistency of the point-wise motion information between frames.
- Our method is validated on KITTI [13], [14] and Argoverse [15] datasets. It outperforms state-of-the-art learning-based LiDAR odometry approaches and even some geometry-based methods, while achieving real-time performance on consumer-grade GPUs.

## II. RELATED WORK

### A. Deep LiDAR Odometry

Deep LiDAR odometry often preprocesses point cloud data due to its sparse and disordered nature. LodoNet [7] uses spherical projection to create depth images and applies SIFT for keypoint matching. LO-Net [8] projects the point cloud onto a 2D plane and uses normal vector similarity for geometric consistency. DMLO [16] employs cylindrical projection, CNN-predicted point pairs, and singular value decomposition to obtain the rigid transformation.

End-to-end deep 3D LiDAR odometry uses flow embedding to describe global motion between point clouds, circumventing point-to-point matching errors. DeepCLR [17] estimates flow embedding with MLP and uses FC layers for pose prediction. PWCLo-Net [11] predicts flow embedding via attentive cost volume and updates poses in a coarse-to-fine manner.

### B. Spatio-temporal Fusion on 3D Point Cloud Learning

Several 3D point cloud learning models focus on spatio-temporal information fusion. ASTA3DConv [18] introduces a novel spatio-temporal convolution for dynamic 3D point cloud sequences, using spatio-temporal attention in neighborhood aggregation around virtual anchors. A self-supervised 4D convolution neural network [19] predicts the temporal order of point cloud clips to learn 4D spatio-temporal

features, evaluated on nearest neighbor retrieval and action recognition tasks. Similarly, P4Transformer [20] uses point 4D convolution to encode and aggregate spatio-temporal features from point cloud videos. BE-STI [21] predicts class-agnostic motion with bidirectional enhancement of spatio-temporal features, leveraging similarities and differences between consecutive and nonadjacent frames.

## III. METHODOLOGY

Fig. 2 illustrates the overall structure of the proposed network DSLO. Firstly, spatial information reuse is introduced to reduce the computational overhead in Sec. III-A. Secondly, we propose the sequential pose initialization based on motion similarity during two LiDAR sampling intervals in Sec. III-B. Next, motion features and poses are updated from coarse to fine through a gated hierarchical pose refinement module in Sec. III-C. Finally, temporal information propagation is proposed in Sec. III-D, with motion information relay solving the spatial inconsistency of point-wise motion information between frames. The training loss formulation is derived in Sec. III-E.

### A. Spatial Information Reuse

We construct a point feature pyramid based on PointNet++ [22] to extract multi-scale spatial structure information from point clouds. However, one point cloud is used twice for pose estimations of point cloud sequences. The redundant feature extraction of the same point cloud is unnecessary. To address this inefficiency, we introduce the spatial information reuse strategy. As depicted in Fig. 2, pyramid features of  $PC_t$  are stored and reused in two consecutive pose estimations, eliminating redundant operations of downsampling and neighborhood feature aggregation. This strategy contributes to reducing time consumption and computational overhead.

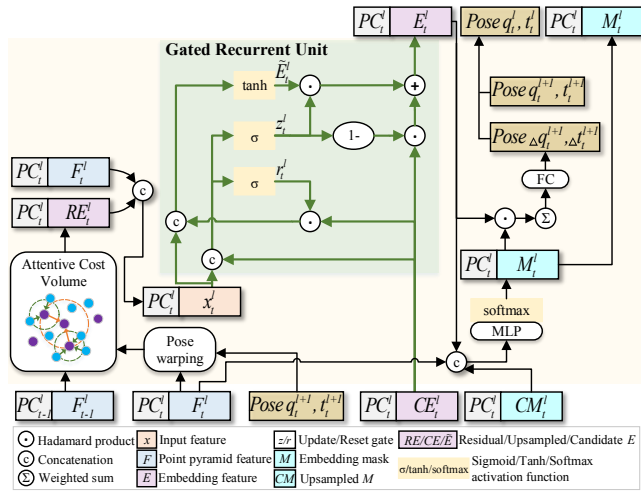


Fig. 3: Gated hierarchical pose refinement module. The residual embedding feature  $RE_t^l$ , point pyramid feature  $F_t^l$  and upsampled embedding feature  $CE^l$  are encoded and fed into a GRU. Subsequently, the output embedding feature  $E_t^l$  assists in refining embedding mask  $M_t^l$  and pose  $q_t^l, t_t^l$ .

### B. Sequential Pose Initialization

As LiDAR is capable of collecting point clouds at a high frequency of 10-100Hz, the pose of LiDAR exhibits minimal variation in two adjacent sampling moments. Inspired by this, we propose a sequential pose initialization strategy to utilize the motion similarity and improve the real-time performance.

When performing the multi-frame LiDAR odometry task on a point cloud sequence, we employ the attentive cost volume [11] method to estimate the pose between the first two frames. Subsequently, we take the last refined pose as the initial value for the current estimation based on the motion similarity. Then the initialized pose acts as a prior and is further refined. As illustrated in Fig. 2, the refined pose  $q_t^{t-1}$  and  $t_t^{t-1}$  between the  $(t-1)$ -th and the  $t$ -th frame serve as the initial guess for the subsequent estimation between the  $t$ -th and  $(t+1)$ -th frames of the point cloud. This sequential pose initialization strategy is iteratively applied throughout the entire sequence.

### C. Gated Hierarchical Pose Refinement

We propose a gated hierarchical pose refinement module that leverages multi-scale spatial information and self-learning gate estimations to selectively discard erroneous motion information from different layers.

Fig. 3 illustrates the module, which uses upsampling layers [23] to connect different layers. The pose in the upper layer is used to warp  $PC_t^l$  and the residual embedding feature  $RE_t^l$  is calculated by attentive cost volume [11] to estimate residual motion. Then, a Gated Recurrent Unit (GRU) [24], [25] structure hierarchically updates the embedding feature. The residual embedding feature  $RE_t^l$ , point pyramid feature  $F_t^l$  and upsampled embedding feature  $CE_t^l$  are fed into GRU to obtain the refined embedding feature  $E_t^l$  as follows:

$$x_t^l = RE_t^l \oplus F_t^l,$$

$$\begin{aligned} z_t^l &= \sigma \left( MLP_z \left( CE_t^l \oplus x_t^l \right) \right), \\ r_t^l &= \sigma \left( MLP_r \left( CE_t^l \oplus x_t^l \right) \right), \\ \tilde{E}_t^l &= \tanh \left( MLP_E \left( r_t^l \odot CE_t^l \oplus x_t^l \right) \right), \\ E_t^l &= (1 - z_t^l) \odot CE_t^l + z_t^l \odot \tilde{E}_t^l, \end{aligned} \quad (1)$$

where  $\oplus$  denotes vector concatenation in the feature dimension and  $\odot$  denotes the Hadamard product.  $\sigma(\cdot)$  denotes the sigmoid activation function.

The two learnable gates in the gated hierarchical pose refinement module, reset gate  $r_t^l$  and update gate  $z_t^l$  estimate the confidence weights of motion information from the upper layer and the current layer. A small activation weight of reset gate  $r_t^l$  prevents unreliable point correspondences in the coarse upper layer, while a small activation weight of update gate  $z_t^l$  eliminates outliers in the dense point cloud caused by occlusion or sensor noise in the current layer.

Finally, the refinement of the embedding mask  $M_t^l$  is facilitated by the combined contributions of  $E_t^l, F_t^l$ , and the upsampled embedding mask  $CM_t^l$ . It is noted that in the 3-rd layer,  $M_t^l$  is predicted solely by  $E_t^l$  and  $F_t^l$  since there is no higher layer. The pose is then updated by inputting the weighted sum of  $E_t^l$  and  $M_t^l$  into an FC layer following [11].

### D. Temporal Feature Propagation with Inconsistent Spatial Context

We propose a novel temporal feature propagation module to fuse motion features along time series, so that pose estimations can benefit from historical motion information and local constraints in the time domain.

The structure of our temporal feature propagation module is depicted in Fig. 4. A peephole LSTM [26], [27] is introduced for the temporal information propagation. However, spatial inconsistency between unstructured point clouds presents challenges in propagating temporal information between frames.

To solve this problem, motion information relay is proposed to initialize the temporal features  $c_{t-1}^l$  and  $E_{t-1}^l$  embedded in  $PC_t^l$  from  $c_{t-1}$  and  $E_{t-1}$  embedded in  $PC_{t-1}^l$ . Based on the initial pose between two point clouds, we first warp the  $(t-1)$ -th frame of point cloud to roughly align two point clouds. Then, we utilize an MLP and max pooling to estimate  $c_{t-1}^l$  and  $E_{t-1}^l$ :

$$\begin{aligned} c_{t-1}^l &= \text{MAX}_{k=1,2,\dots,K} (MLP(c_{t-1})), \\ E_{t-1}^l &= \text{MAX}_{k=1,2,\dots,K} (MLP(E_{t-1})). \end{aligned} \quad (2)$$

Then, a peephole LSTM [26], [27] is adopted to propagate embedding features along time series. A peephole LSTM comprises a cell state  $c_t$  indicating long-term memory and a hidden state  $h_t$  indicating short-term memory. We characterize the hidden state  $h_t$  with the embedding feature  $E_t$  to capture motion information in a short local sequence.

Finally, the embedding feature  $E_t$  and the cell state  $c_t$  are sequentially propagated as follows:

$$x^t = RE^L \oplus F^L,$$

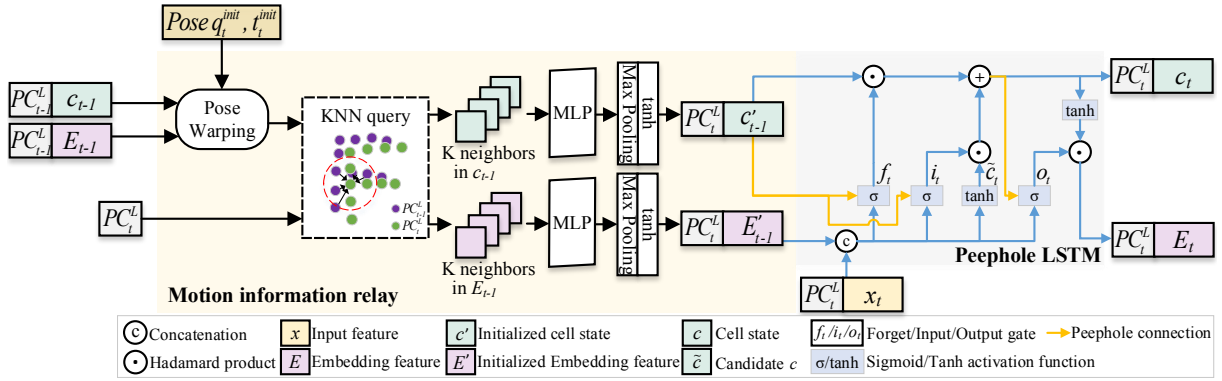


Fig. 4: Temporal feature propagation with inconsistent spatial context. The historical motion information  $E_{t-1}$  and LSTM cell state  $c_{t-1}$  embedded in  $PC_{t-1}^L$  are passed to  $PC_t^L$  with a learning-based motion information relay method. Then peephole LSTM is employed to fuse the temporal motion information.

$$\begin{aligned}
 f_t &= \sigma(MLP_f(c'_{t-1} \oplus E'_{t-1} \oplus x_t)), \\
 i_t &= \sigma(MLP_i(c'_{t-1} \oplus E'_{t-1} \oplus x_t)), \\
 \tilde{c}_t &= \tanh(MLP_c(E'_{t-1} \oplus x_t)), \\
 c_t &= f_t \odot c'_{t-1} + i_t \odot \tilde{c}_t, \\
 o_t &= \sigma(MLP_o(c_t \oplus E'_{t-1} \oplus x_t)), \\
 E_t &= o_t \odot \tanh(c_t),
 \end{aligned} \quad (3)$$

where  $f_t$ ,  $i_t$ , and  $o_t$  denote the activation weights of the forgetting gate, input gate, and output gate respectively.

By propagating temporal information in LiDAR odometry, context and continuity in the time domain provide guidance for the pose estimation, resulting in enhanced robustness and accuracy of LiDAR odometry.

### E. Training Loss

During the training process, three frames of point clouds are processed simultaneously. Then the estimated poses between each two frames are integrated into the loss  $\ell$ :

$$\ell = \sum_{l=1}^L \alpha^l \ell_{t,t+1}^l + \sum_{l=1}^L \alpha^l \ell_{t+1,t+2}^l + \sum_{l=1}^L \alpha^l \ell_{t,t+2}^l, \quad (4)$$

where each loss term is determined by a weighted sum of losses from different layers of point clouds. Here,  $\alpha^l$  denotes the weights at the  $l$ -th layer.

For the loss at the  $l$ -th layer, we adopt the design proposed in [8], [11], [31]:

$$\begin{aligned}
 \ell_{t,t+1}^l &= \|t_{gt} - t^l\| \exp(-s_t) + s_t \\
 &+ \left\| q_{gt} - \frac{q^l}{\|q^l\|_2} \right\|_2 \exp(-s_q) + s_q,
 \end{aligned} \quad (5)$$

where  $\|\cdot\|$  and  $\|\cdot\|_2$  denote the  $\ell_1$  and  $\ell_2$  norm respectively.  $s_t$  and  $s_q$  are two learnable parameters designed to balance the scale differences in translation and rotation errors of the estimated poses.  $t^l$  and  $q^l$  represent the pose predicted at the  $l$ -th layer, while  $t_{gt}$  and  $q_{gt}$  denote the ground truth.

## IV. EXPERIMENTS

### A. Implementation Details

The proposed network is implemented with PyTorch and trained/tested on an NVIDIA RTX2080Ti GPU and an Intel Xeon W-2265 3.50GHz CPU. The initial learning rate is

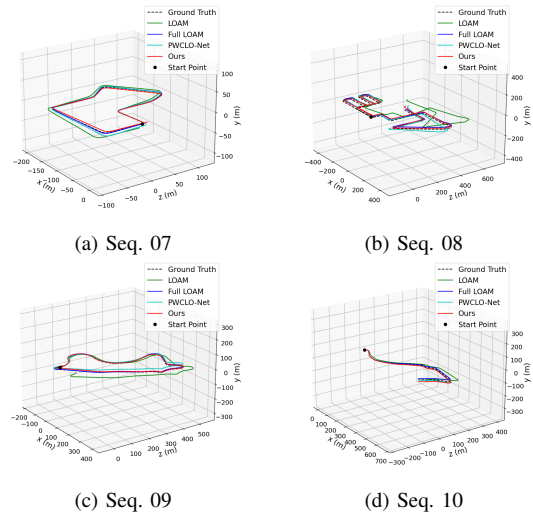


Fig. 5: 3D trajectory results on KITTI Seq. 07-10.

0.001, decaying by 0.7 every 26 epochs until it reaches 0.00001. We use the Adam optimizer with  $\beta_1 = 0.9$  and  $\beta_2 = 0.999$ , a batch size of 8, and an input point cloud size of  $N = 8192$ . A dropout layer is included during pose regression to mitigate overfitting.

In Equation (4),  $L = 4$ ,  $\alpha^1 = 1.6$ ,  $\alpha^2 = 0.8$ ,  $\alpha^3 = 0.4$ ,  $\alpha^4 = 0.2$ . The initial values for learnable parameters in the loss function are  $s_t = 0$  and  $s_q = -2.5$  in Equation (5).

During training, three frames of point cloud are processed simultaneously. During testing, poses are continuously estimated to maintain motion information across the sequence.

### B. Accuracy Evaluation

1) *Experiments on KITTI odometry dataset:* We use KITTI Seq. 00-06 for training and 07-10 for validation. The quantitative results are compared with recent LiDAR odometry methods [3], [4], [8], [11], [28], [29], [30] in Table I. The original ICP-po2po [28] performs poorly due to improper initial values, while its variants ICP-po2pl [28] and GICP [4] perform better but lack robustness, with large localization errors on certain sequences. Feature-based methods like LOAM [3] and CLS [29] use hand-crafted features,

TABLE I: The LiDAR odometry experiment results on KITTI odometry dataset [13], [14]. RTE and RRE mean the relative translation error (%) and relative rotation error ( $^{\circ}/100m$ ) respectively, which are calculated by Root Mean Squared Error (RMSE) of the relative transformation on all possible subsequences in the length of 100, 200, ..., 800  $m$ . “\*” means the training sequence. The best results are bold. The percentage increase in accuracy and reduction in runtime are calculated by comparing our DSLO with the baseline method PWCLO-Net [11].

Seq.	LOAM[3]		ICP-po2po[28]		ICP-po2pl[28]		GICP[4]		CLS[29]		Velas et al.[30]		LO-Net[8]		PWCLO-Net[11]		Ours	
	RTE	RRE	RTE	RRE	RTE	RRE	RTE	RRE	RTE	RRE	RTE	RRE	RTE	RRE	RTE	RRE	RTE	RRE
00*	1.10	0.53	6.88	2.99	3.80	1.73	1.29	0.64	2.11	0.95	3.02	NA	1.47	0.72	<b>0.78</b>	0.42	<b>0.78</b>	<b>0.40</b>
01*	2.79	0.55	11.21	2.58	13.53	2.58	4.39	0.91	4.22	1.05	4.44	NA	1.36	0.47	0.67	<b>0.23</b>	<b>0.66</b>	<b>0.23</b>
02*	1.54	0.55	8.21	3.39	9.00	2.74	2.53	0.77	2.29	0.86	3.42	NA	1.52	0.71	0.86	0.41	<b>0.77</b>	<b>0.34</b>
03*	1.13	0.65	11.07	5.05	2.72	1.63	1.68	1.08	1.63	1.09	4.94	NA	1.03	0.66	0.76	0.44	<b>0.67</b>	<b>0.37</b>
04*	1.45	0.50	6.64	4.02	2.96	2.58	3.76	1.07	1.59	0.71	1.77	NA	0.51	0.65	0.37	<b>0.40</b>	<b>0.31</b>	0.47
05*	0.75	0.38	3.97	1.93	2.29	1.08	1.02	0.54	1.98	0.92	2.35	NA	1.04	0.69	<b>0.45</b>	<b>0.27</b>	0.50	0.30
06*	0.72	0.39	1.95	1.59	1.77	1.00	0.92	0.46	0.92	0.46	1.88	NA	0.71	0.50	<b>0.27</b>	<b>0.22</b>	0.57	0.38
07	0.69	0.50	5.17	3.35	1.55	1.42	0.64	0.45	1.04	0.73	1.77	NA	1.70	0.89	0.60	0.44	<b>0.58</b>	<b>0.41</b>
08	1.18	0.44	10.04	4.93	4.42	2.14	1.58	0.75	2.14	1.05	2.89	NA	2.12	0.77	1.26	0.55	<b>1.16</b>	<b>0.51</b>
09	1.20	0.48	6.93	2.89	3.95	1.71	1.97	0.77	1.95	0.92	4.94	NA	1.37	0.58	0.79	0.35	<b>0.72</b>	<b>0.33</b>
10	1.51	0.57	8.91	4.74	6.13	2.60	1.31	0.62	3.46	1.28	3.27	NA	1.80	0.93	1.69	0.62	<b>1.29</b>	<b>0.49</b>
Mean on 07-10	1.145	0.498	7.763	3.978	4.013	1.968	1.375	0.648	2.148	0.995	3.218	NA	1.748	0.793	1.085	0.490	<b>0.938</b>	<b>0.435</b>
																	( $\uparrow$ 15.67%)	( $\uparrow$ 12.64%)
Time(s)	0.069		0.61		0.98		3.21		2.36		0.067		0.08		0.066		<b>0.049</b> ( $\downarrow$ 34.69%)	

TABLE II: Odometry experiments on Argoverse dataset [15]. The best results are bold.

Method	Mean on 00-23	
	ATE	RPE
LeGO-LOAM [5] w/o mapping	4.537	0.110
SUMA [6] w/o mapping	3.663	0.039
PyLiDAR [32] w/o mapping	6.900	0.109
A-LOAM [33] w/o mapping	4.138	0.066
Ours	<b>0.111</b>	<b>0.027</b>

TABLE III: Runtime of DSLO and other LiDAR odometry.

Method	Runtime/s
ICP-po2po [28]	0.61
ICP-po2pl [28]	0.98
GICP [4]	3.21
LOAM [3]	0.069
CLS [29]	2.36
Velas et al. [30]	0.067
LO-Net [8]	0.080
PWCLO-Net [11]	0.066
Ours w/o pose initialization & pyramid feature sharing	0.072
Ours w/o pose initialization	0.056
Ours(full, with pose initialization & pyramid feature sharing)	<b>0.049</b>

whereas our DSLO uses a learning-based feature extraction strategy, achieving superior accuracy. Unlike other learning-based LiDAR odometry methods, such as LO-Net [8] and Velas et al. [30], which project LiDAR data onto 2D planes, our DSLO processes raw point clouds directly. This enables it to learn more comprehensive 3D spatial structure information. DSLO also exhibits accuracy improvements over the baseline method PWCLO-Net [11]. PWCLO-Net estimates pose between two point clouds independently. In contrast, our network ensures motion consistency through sequential pose initialization and temporal feature propagation. Overall, our DSLO outperforms all recent LiDAR odometry methods with at least a 15.67% improvement on RTE and a 12.64% improvement on RRE on average evaluation.

Qualitative results in Fig. 5 show the trajectory estimated by our DSLO coincides with the ground truth best.

2) *Experiments on Argoverse dataset*: To assess the generalization of our method, we conduct experiments on the Argoverse dataset [15]. We employ Absolute Trajectory Error (ATE) and Relative Pose Error (RPE) for evaluation due to the short sequence length. Our DSLO model is trained and tested on the official Argoverse training/testing split. For comparison, we assess four geometry-based odometry methods [5], [6], [32], [33] with the mapping thread disabled. As shown in Table II, our DSLO significantly outperforms these methods, particularly in the ATE metric.

### C. Real-time Performance Evaluation

To validate the real-time performance of the proposed method, we evaluate the runtime of DSLO, DSLO with efficiency modules removed, and the baseline method PWCLO-Net [11] on KITTI Seq. 04 with a batch size of 1. The experimental results in Table III demonstrate that our sequential pose initialization and spatial information reuse modules reduce runtime by 34.69% compared to PWCLO-Net. Furthermore, DSLO achieves a comparable inference speed with other registration-based [4], [28], feature extraction-based [3], [29], and learning-based methods [8], [30].

### D. Ablation Study

1) *Spatial information reuse and sequential pose initialization*: We first eliminate sequential pose initialization, using attentive cost volume for initial pose prediction across the sequence. Next, we remove spatial information reuse, resulting in redundant calculations of the same point feature pyramid. As shown in Table IV (a), both techniques improve accuracy and enhance real-time performance (Table III).

2) *Gated hierarchical pose refinement*: We replace the gated hierarchical pose refinement module with an MLP structure. Results in Table IV (b) indicate that the gated module outperforms the MLP in refining poses. The gated hierarchical pose refinement module is adept at preserving hierarchical information through self-learning gate states. It effectively filters valid information from different layers.

TABLE IV: The ablation study results of DSLO for the network structure on KITTI odometry dataset [13], [14].

Method	07		08		09		10		Mean on 07-10	
	RTE	RRE	RTE	RRE	RTE	RRE	RTE	RRE	RTE	RRE
(a) Ours, w/o sequential pose initialization	0.60	0.49	1.33	0.49	0.99	0.39	<b>1.20</b>	0.62	1.029	0.496
Ours, w/o sequential pose initialization, w/o spatial information reuse	<b>0.58</b>	0.50	1.28	<b>0.42</b>	0.85	0.38	1.44	0.58	1.039	0.470
Ours (full, with sequential pose initialization & spatial information reuse)	<b>0.58</b>	<b>0.41</b>	<b>1.16</b>	0.51	<b>0.72</b>	<b>0.33</b>	1.29	<b>0.49</b>	<b>0.938</b>	<b>0.435</b>
(b) Ours, replace gated hierarchical pose refinement with MLP	1.05	0.85	1.40	0.60	0.90	0.43	<b>1.20</b>	0.53	1.136	0.603
Ours (full, with gated hierarchical pose refinement)	<b>0.58</b>	<b>0.41</b>	<b>1.16</b>	<b>0.51</b>	<b>0.72</b>	<b>0.33</b>	1.29	<b>0.49</b>	<b>0.938</b>	<b>0.435</b>
(c) Ours, w/o motion information relay, w/o peephole LSTM	0.73	0.53	1.29	<b>0.46</b>	0.93	0.43	<b>1.22</b>	0.57	1.041	0.498
Ours, with nearest neighbor query & peephole LSTM	1.50	0.76	1.53	0.54	0.95	0.45	0.88	0.58	1.220	0.580
Ours (full, with motion information relay & peephole LSTM)	<b>0.58</b>	<b>0.41</b>	<b>1.16</b>	0.51	<b>0.72</b>	<b>0.33</b>	1.29	<b>0.49</b>	<b>0.938</b>	<b>0.435</b>

3) *Temporal information propagation with inconsistent spatial context*: We first remove the entire temporal feature propagation module, including motion information relay and peephole LSTM. Subsequently, the learning-based motion information relay method is replaced with a direct nearest neighbor query. Specifically, the LSTM state value of the nearest point in the  $(t-1)$ -th frame is assigned to the corresponding point in the  $t$ -th frame. Results in Table IV (c) demonstrate the significance of the proposed temporal feature propagation method, particularly the learning-based motion information relay, in improving pose regression accuracy.

## V. CONCLUSION

We propose a novel deep sequence LiDAR odometry based on inconsistent spatio-temporal propagation, achieving high accuracy and real-time performance. Our spatial information reuse and sequential pose initialization reduce the computational overhead without degrading the accuracy. The gated hierarchical pose refinement enables efficient coarse-to-fine pose updates, with self-learning gate estimations discarding incorrect historical matches and mitigating sensor noise. Additionally, our temporal feature propagation fuses motion information over time, addressing inconsistent spatial context and improving LiDAR odometry accuracy.

## REFERENCES

- [1] H. Li, J. Zhao, J.-C. Bazin, P. Kim, K. Joo, Z. Zhao, and Y.-H. Liu, "Hong kong world: Leveraging structural regularity for line-based slam," *TPAMI*, vol. 45, no. 11, pp. 13035–13053, 2023.
- [2] H. Li, J. Zhao, J.-C. Bazin, and Y.-H. Liu, "Robust estimation of absolute camera pose via intersection constraint and flow consensus," *TIP*, vol. 29, pp. 6615–6629, 2020.
- [3] J. Zhang and S. Singh, "LOAM: lidar odometry and mapping in real-time," in *RSS*, vol. 2, no. 9, 2014, pp. 1–9.
- [4] A. Segal, D. Hähnel, and S. Thrun, "Generalized-icp," in *RSS*, vol. 2, no. 4, 2009, p. 435.
- [5] T. Shan and B. Englot, "Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," in *IROS*, 2018, pp. 4758–4765.
- [6] J. Behley and C. Stachniss, "Efficient surfel-based slam using 3d laser range data in urban environments," in *RSS*, vol. 2018, 2018, p. 59.
- [7] C. Zheng, Y. Lyu, M. Li, and Z. Zhang, "Lodonet: A deep neural network with 2d keypoint matching for 3d lidar odometry estimation," in *ASE*, 2020, pp. 2391–2399.
- [8] Q. Li, S. Chen, C. Wang, X. Li, C. Wen, M. Cheng, and J. Li, "Lo-net: Deep real-time lidar odometry," in *CVPR*, 2019, pp. 8465–8474.
- [9] J. Liu, G. Wang, C. Jiang, Z. Liu, and H. Wang, "Translo: A window-based masked point transformer framework for large-scale lidar odometry," in *AAAI*, vol. 37, no. 2, 2023, pp. 1683–1691.
- [10] J. Liu, G. Wang, Z. Liu, C. Jiang, M. Pollefeys, and H. Wang, "Regformer: An efficient projection-aware transformer network for large-scale point cloud registration," in *ICCV*, 2023, pp. 8451–8460.

- [11] G. Wang, X. Wu, Z. Liu, and H. Wang, "Pwclonet: Deep lidar odometry in 3d point clouds using hierarchical embedding mask optimization," in *CVPR*, 2021, pp. 15905–15914.
- [12] B. Zhou, Y. Tu, Z. Jin, C. Xu, and H. Kong, "Hpplo-net: Unsupervised lidar odometry using a hierarchical point-to-plane solver," *IEEE TIV*, vol. 9, no. 1, pp. 2727–2739, 2023.
- [13] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *CVPR*, 2012, pp. 3354–3361.
- [14] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *IJRR*, vol. 32, no. 11, pp. 1231–1237, 2013.
- [15] M.-F. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, *et al.*, "Argoverse: 3d tracking and forecasting with rich maps," in *CVPR*, 2019.
- [16] Z. Li and N. Wang, "Dmlo: Deep matching lidar odometry," in *IROS*, 2020, pp. 6010–6017.
- [17] M. Horn, N. Engel, V. Belagiannis, M. Buchholz, and K. Dietmayer, "Deepclr: Correspondence-less architecture for deep end-to-end point cloud registration," in *ITSC*, 2020, pp. 1–7.
- [18] G. Wang, H. Liu, M. Chen, Y. Yang, Z. Liu, and H. Wang, "Anchor-based spatio-temporal attention 3-d convolutional networks for dynamic 3-d point cloud sequences," *IEEE TIM*, vol. 70, pp. 1–11, 2021.
- [19] H. Wang, L. Yang, X. Rong, J. Feng, and Y. Tian, "Self-supervised 4d spatio-temporal feature learning via order prediction of sequential point cloud clips," in *WACV*, 2021, pp. 3762–3771.
- [20] H. Fan, Y. Yang, and M. Kankanhalli, "Point 4d transformer networks for spatio-temporal modeling in point cloud videos," in *CVPR*, 2021, pp. 14204–14213.
- [21] Y. Wang, H. Pan, J. Zhu, Y.-H. Wu, X. Zhan, K. Jiang, and D. Yang, "Be-sti: Spatial-temporal integrated network for class-agnostic motion prediction with bidirectional enhancement," in *CVPR*, 2022, pp. 17093–17102.
- [22] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *NIPS*, 2017, p. 5105–5114.
- [23] X. Liu, C. R. Qi, and L. J. Guibas, "Flownet3d: Learning scene flow in 3d point clouds," in *CVPR*, 2019, pp. 529–537.
- [24] Y. Kittenplon, Y. C. Eldar, and D. Raviv, "Flowstep3d: Model unrolling for self-supervised scene flow estimation," in *CVPR*, 2021.
- [25] Z. Teed and J. Deng, "Raft: Recurrent all-pairs field transforms for optical flow," in *ECCV*, 2020, pp. 402–419.
- [26] F. Gers and E. Schmidhuber, "Lstm recurrent networks learn simple context-free and context-sensitive languages," *IEEE TNN*, vol. 12, no. 6, pp. 1333–1340, 2001.
- [27] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, "Learning precise timing with lstm recurrent networks," *JMLR*, vol. 3, no. 1, pp. 115–143, 2003.
- [28] P. J. Besl and N. D. McKay, "A method for registration of 3-d shapes," *TPAMI*, vol. 14, no. 2, pp. 239–256, 1992.
- [29] M. Velas, M. Spanel, and A. Herout, "Collar line segments for fast odometry estimation from velodyne point clouds," in *ICRA*, 2016, pp. 4486–4495.
- [30] M. Velas, M. Spanel, M. Hradis, and A. Herout, "Cnn for imu assisted odometry estimation using velodyne lidar," in *ICARSC*, 2018.
- [31] G. Wang, X. Wu, S. Jiang, Z. Liu, and H. Wang, "Efficient 3d deep lidar odometry," *TPAMI*, vol. 45, no. 5, pp. 5749–5765, 2022.
- [32] P. Dellenbach, J.-E. Deschaud, B. Jacquet, and F. Goulette, "What's in my lidar odometry toolbox?" in *IROS*, 2021, pp. 4429–4436.
- [33] T. Qin and S. Cao. A-LOAM: advanced implementation of loam. [Online]. Available: <https://github.com/HKUST-Aerial-Robotics/A-LOAM>