

Supplementary Materials: DeepPointMap2: Accurate and Robust LiDAR-Visual SLAM with Neural Descriptors

Anonymous Authors

1 ADDITIONAL EXPERIMENTS

1.1 Registration Accuracy

The key module of *DeepPointMap2* is its odometry model *i.e.*, DPM Encoder and DPM Decoder. To directly investigate the performance of these two learning-based modules, additional experiments are conducted. We evaluate the point cloud registration of our methods on the KITTI Odometry benchmark and the KITTI-360 benchmark. We randomly sample *approx.* 8000 and 19000 frame-pairs within the distance of 20 m from KITTI Odometry benchmark and KITTI-360 datasets, respectively.

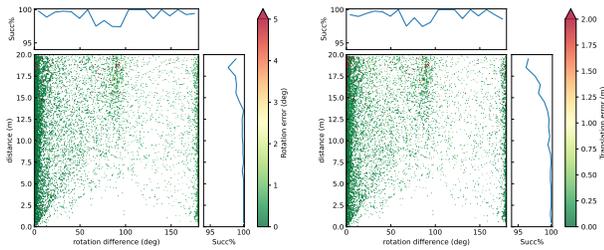


Figure 1: Registration Accuracy and Success Rate. Each dot represents a sampled pair, and the color indicates the registration error (*green* is better). The success rate (%) maintains above 97% even at the ground-truth distances of up to 20 m or rotation of 180°.

Fig. 1 shows the registration translation (left) and rotation (right) accuracy w.r.t. distance and rotation between frame-pair on both evaluation datasets. For both figures, the x-axis represents the ground-truth distance, the y-axis represents the ground-truth rotation difference between frames, and each point indicates a sampled pair. The metrics, *i.e.*, Translation Error (in m) and Rotation error (in °), are color-coded (*green* is better). Adopting the *Registration Recall* metric from Qin et al. [4], we define a success criterion as a Translation Error within 2 m and a Rotation Error within 5°. Failed samples are marked in red. Despite performance decline with increased distance or rotation, our method achieves high success rates within 20 m.

For better visualization, we also calculate the average success-rate under each bin, as indicated beside both axes (*blue* lines). We can observe that the success rate stays almost 100% for the distance from 0 m to 20 m. When the initial distance increased, the success rate slightly decreased as the overlap between the point cloud pairs became smaller, but still remained above 97%.

It is easy to notice that the scatters are not distributed uniformly and most of the samples are likely to be placed at rotation difference of 0°, 90° and 180°. This is due to the fact that *straights* (0°),

turns (90°) and *turnbacks* (180°) are very common in the roadway environment. Of these scenarios, the *turns* scenario had the lowest success rate, which could be due to two reasons (1) at the intersection, the geometric information of the road in all four directions is relatively similar, resulting in the model incorrectly recognize the point cloud with a 90-degree deviation, (2) at the intersection, the front camera results in the non-overlapping of the frontal image information captured by the vehicle, which makes the model rely on the geometric information only for the registration, and (3) the overlap-rate of the point cloud at the intersection is limited. Meanwhile, as the ground-truth distance rises, the overlap between the two frames gradually decreases, making it difficult for the model to be accurately aligned based on a small amount of overlap.

1.2 Loop Detection Accuracy

To ensure global consistency in map reconstruction, loop detection is essential. This experiment evaluates the loop detection capabilities of our proposed model.

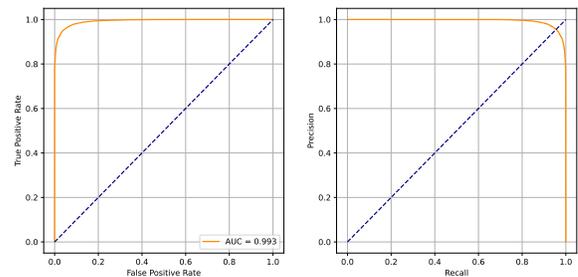


Figure 2: Loop Detection Accuracy. ROC curve (left) and Precision-Recall curve (right).

Similar to the experiment above, we evaluate the loop detection performance on both the KITTI Odometry benchmark and the KITTI-360 benchmark. We randomly sample *approx.* 8000+19000 frame-pairs, half of them are with a distance less than 20 m while others are greater than 20 m. The Precision-Recall (left) and ROC (right) curves in Figure Fig. 2 demonstrate our method’s effectiveness, achieving an AUC score of 0.993.

1.3 Reference Point Number vs. Performance

We extend our analysis to examine the performance of *DeepPointMap2* across varying numbers of reference-points, building on the superiority of the Farthest-Point Sampling (FPS) strategy established in Sec. 5.4 in the manuscript. This experiment focuses on the runtime consumption in relation to the reference-point numbers.

As detailed in Tab. 1, the runtime remains consistent for a low count of reference-points (<256). However, with an increase in

Table 1: Run-time Performance vs. Reference-Point Number.

#Points	Speed	GPU	RAM
128	0.1564 s	1.1504 GiB	4.5964 GiB
256	0.1569 s	1.1590 GiB	4.6892 GiB
512	0.1559 s	1.5011 GiB	4.9930 GiB
1024	0.1627 s	2.6852 GiB	4.6790 GiB
2048	0.1850 s	6.6047 GiB	4.7693 GiB
4096	0.3257 s	11.4332 GiB	4.8450 GiB
8192	-	>24 GiB	-

reference-points, the DPM Decoder’s cross-attention module, which introduces a quadratic space complexity of $O(n^2)$, leads to a substantial rise in GPU memory usage. At 8192 reference-points, the model’s requirements exceed the available GPU memory on a single NVIDIA RTX 3090.

2 MODEL DETAILS

2.1 DPM Encoder

DPM Encoder consists of three main parts: (1) LiDAR backbone, (2) Image backbone and (3) *Visual-Point Fusion Module*.

LiDAR Backbone. We utilize a modified PointNeXt [3] network as the LiDAR backbone. The model detailed structure is shown in Tab. 2.

Table 2: LiDAR Backbone Configuration.

Stage	#Point	Query	Radius	Channel
MLP	16k	-	-	16
SA	16k	ball	r=0.02	32
			r=0.04	32
SA	4096	ball	r=0.04	64
			r=0.08	64
SA	1024	ball	r=0.08	128
			r=0.16 (×4)	128
SA	256	ball	r=0.16	256
			r=0.32	256
			r=0.16	512
SA	64	ball	r=0.16	512
			r=0.32	512
FP	16	knn	k=3	256
FP	64	knn	k=3	128
FP	256	knn	k=3	128

Before feeding into the point cloud backbone, we apply the pre-processing include: (1) Voxel Downsample: reducing the original scan using a grid size of 0.3 m, (2) Padding/Randomly Sampling: limiting (or padding) the point cloud size to 16,384, (3) Distance Clip: removing points beyond 60 m or closer than 1 m, (4) Normalization: scaling all point coordinates by a factor of 60 m to a range of 0 to 1.0.

Image Backbone. As for the image backbone, we directly use ConvNeXt [2] with the ‘tiny’ configuration, pre-trained on *IMAGENET1K*. An FPN [1] is subsequently applied to aggregate multi-scale image feature tokens.

Before feeding into the image backbone, we apply the pre-processing as follows:(1) Color Normalization: normalizing the image with mean (0.485, 0.456, 0.406) and standard deviation (0.229, 0.224, 0.225), (2) Resize: resizing the input image to the required size (224×768 for KITTI and KITTI-360, 384×512 for KITTI-Carla).

Visual-Point Fusion Module. We introduce a novel *Visual-Point Fusion Module* designed to integrate multi-scale feature tokens. As depicted in Fig. 3, the module employs a Multi-Modal Transformer (MMT) block for each scale to synthesize the tokens from different modalities. Each MMT block is composed of a Self Attention module, a Biased Attention module, and a Feed-Forward Network (FFN). The Ref-Points \mathbf{R} , serving as Querys, are initially processed by the Self Attention module to refine their features. Subsequently, the Query, Key, and Value are directed into the Biased Attention module, where multi-modal tokens (Key and Value) are exchanged and consolidated with the Ref-Points (Query). The resultant aggregated features are then further enhanced by the FFN, which is essentially an MLP with residual connections.

2.2 DPM Decoder

Following DeepPointMap [6], the DPM Decoder contains four parts: (a) Descriptor-wise Transformer, (b) Similarity Head, (c) Offset Head and (d) Overlap Head, as demonstrate in Fig. 4.

Descriptor-wise Transformer. The network architecture is detailed in Fig. 4 (a). Adopting the approach from Yew and Lee [5], we implement three symmetric Transformer Decoder Layers to facilitate information exchange between descriptors \mathcal{R}_{t_1} and \mathcal{R}_{t_2} . Each layer sequentially integrates self-attention, cross-attention, and an FFN. We apply sinusoidal positional encoding to all attention modules.

Similarity Head. The similarity head aims to find the correlation between two descriptors. As shown in Fig. 4 (b), after an MLP, the correspondence matrix S is calculated (denoted as \odot) using pair-wise cosine similarity by:

$$S_{ij} = \text{MLP}\left(r_i^{\text{feat}}\right) \odot \text{MLP}\left(r_j^{\text{feat}}\right), r_i \in \mathcal{R}_{t_1}, r_j \in \mathcal{R}_{t_2} \quad (1)$$

After that, the correspondence σ is subsequently selected based on matrix S .

Offset Head. As discussed in the sec. 3.3 in the manuscript, even if the correspondence σ is perfect, the sparsity of descriptors may still lead to inaccuracy registration. To tackle this problem, the offset head (Fig. 4 (c)) is introduced to predict the offset δ_{ij} between descriptor pairs such that:

$$R_1\left(r_i^{\text{xyz}} + \delta_{ij}\right) + t_1 = R_2\left(r_j^{\text{xyz}}\right) + t_2, (r_i, r_j) \in \sigma \quad (2)$$

Here, R_1, t_1 and R_2, t_2 represent the ground truth poses of frames t_1 and t_2 , respectively. To achieve this, the offset first concatenates the feature of the descriptor pair, and then utilizes an MLP to predict such offset by:

$$\delta_{ij} = \text{MLP}\left(r_i^{\text{feat}} | r_j^{\text{xyz}}\right), (r_i, r_j) \in \sigma \quad (3)$$

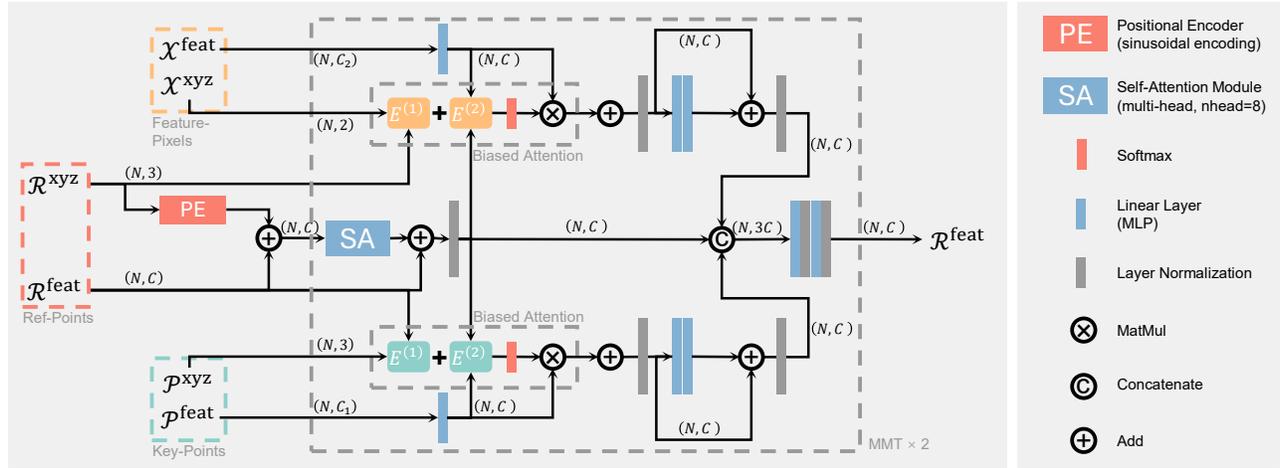


Figure 3: The Details of Visual-Point Fusion Module.

The concatenation operator “|” is used to combine features.

Overlap Head. Finally, to achieve robust and fast loop detection, the overlapping head (Fig. 4 (d)) firstly utilizes global average pooling on the descriptors to get frame-wise descriptor $\mathcal{R}'_{t_1}, \mathcal{R}'_{t_2}$, and then use an MLP with residual connect to predict the loop probability between these two frames by:

$$p_o = \text{MLP}(\mathcal{R}'_{t_1} | \mathcal{R}'_{t_2}) \quad (4)$$

2.3 Mapping

Pose-Graph. Our reconstructed map is maintained using a pose-graph structure, defined as $G = \langle V, E \rangle$, where V represents keyframes and E encodes the inter-frame positional relations. Each vertex $v_i \in V$ contains a descriptor and its 6-DOF pose, while each edge $e_{ij} \in E$ holds the relative transformation matrix and confidence value.

Keyframe selection is based on frame-wise distance measurements. For a new frame v_i , the nearest keyframe v_j is retrieved, and frame-to-frame odometry estimates the pose. If the distance between v_i and v_j surpasses a dynamic threshold, v_i is assigned as a keyframe. To refine the pose estimation, a frame-to-map registration is conducted, and the updated frame v_i and edge e_{ij} are integrated into the pose-graph.

The loop detection, as a crucial task for map consistency, is performed solely on keyframes. Upon inserting a new keyframe v_i , the system identifies all candidate keyframes $V_{\text{candidate}}$ within a distance threshold ϵ_l . The Overlap Head then predicts the pairwise loop probability p_{ij} among them. Exceeding a probability threshold ϵ_p triggers the insertion of edge v_{ij} and initiates a map-to-map registration for refined pose estimation. Subsequently, a global optimization procedure is executed.

Global Optimization. Global optimization is conducted using the standard PoseGraph Optimization with the Levenberg-Marquardt algorithm from the Open3D library [7] following the confirmation of loop detection.

2.4 Training

As described in Sec. 2.3, our method requires frame-to-frame, frame-to-map, and map-to-map registration abilities. To achieve this, we use the curriculum learning method to train our model. In Phase One, the training procedure can be described as follows:

- (1) Randomly generate a number $S \in [2, K]$;
- (2) Select $S - 1$ additional scans within $20m$ with respect to the current training scan;
- (3) Process these scans using DPM Encoder and obtain S descriptor clouds individually;
- (4) These descriptor clouds are randomly divided into two groups with sizes S_1 and $S - S_1$, and merge into 2 map-level descriptor clouds by concatenating the descriptor clouds;
- (5) The 2 merged descriptor clouds are processed by DPM Decoder;
- (6) Apply Losses. Update weights.

We start with a smaller K to learn scan-level odometer, and gradually increase K over epochs to learn map-level localization. By employing this curriculum learning strategy, *DeepPointMap2* learns progressively and adapts to multi-scale registrations, leading to improved performance and robustness in SLAM tasks.

In the second stage, the training procedure is described as follows:

- (1) Randomly select a scan;
- (2) Select another scan inside / outside the distance of $20m$ from the first scan, with equal probability;
- (3) Extract the corresponding descriptors using DPM Encoder;
- (4) Predict the Overlap Probability of these two scans;
- (5) Apply Losses. Update weights.

In both training phases, we apply the following data augmentations:

- (1) (LiDAR) Random Drop: randomly discarding points with a probability of 0 to 0.5.
- (2) (LiDAR) Random Occlusion: Randomly generate some virtual boxes with random size and position. Each box introduces

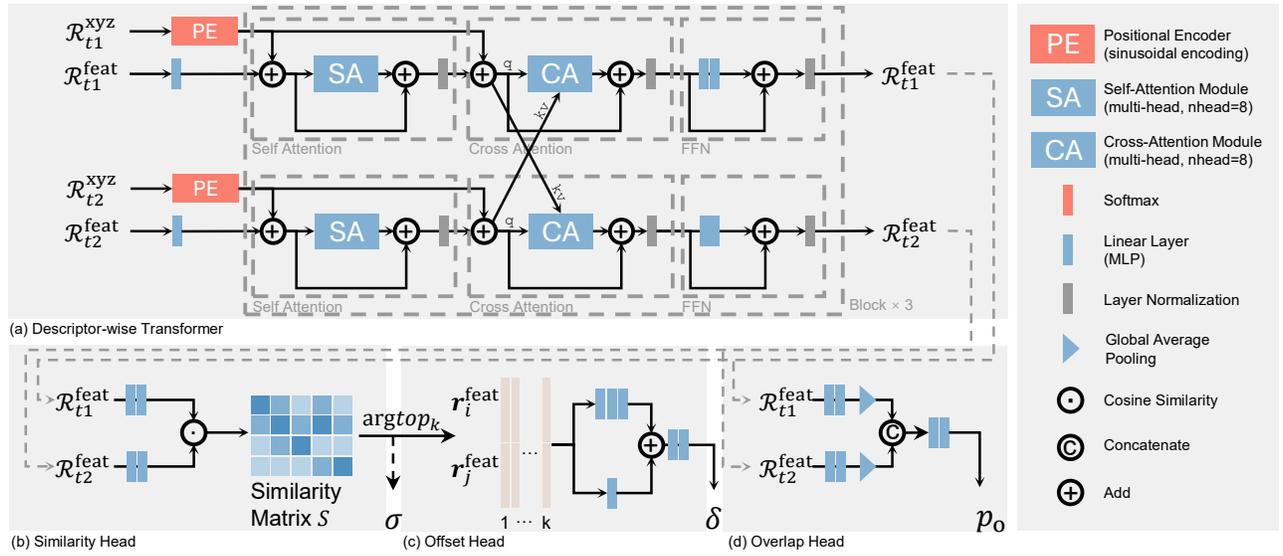


Figure 4: The Details of DPM Decoder.

an occlusion effect by removing all points that pass through it.

- (3) (LiDAR) Random Rotation and Translation: applying a random transformation to the point cloud.
- (4) (LiDAR) Random Jitter: introducing random noise to each point's coordinates.
- (5) (Image) Random Flip, Color Jitter.

REFERENCES

- [1] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. 2017. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2117–2125.
- [2] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. 2022. A convnet for the 2020s. In *Proceedings of the IEEE/CVF*

conference on computer vision and pattern recognition. 11976–11986.

- [3] Guocheng Qian, Yuchen Li, Houwen Peng, Jinjie Mai, Hasan Hammoud, Mohamed Elhoseiny, and Bernard Ghanem. 2022. Pointnext: Revisiting pointnet++ with improved training and scaling strategies. *Advances in Neural Information Processing Systems* 35 (2022), 23192–23204.
- [4] Zheng Qin, Hao Yu, Changjian Wang, Yulan Guo, Yuxing Peng, Slobodan Ilic, Dewen Hu, and Kai Xu. 2023. GeoTransformer: Fast and Robust Point Cloud Registration With Geometric Transformer. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023).
- [5] Zi Jian Yew and Gim Hee Lee. 2022. Regtr: End-to-end point cloud correspondences with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6677–6686.
- [6] Xiaze Zhang, Ziheng Ding, Qi Jing, Yuejie Zhang, Wenchao Ding, and Rui Feng. 2024. DeepPointMap: Advancing LiDAR SLAM with Unified Neural Descriptors. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 10413–10421.
- [7] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. 2018. Open3D: A modern library for 3D data processing. *arXiv preprint arXiv:1801.09847* (2018).