LeoVR: Motion-Inspired Visual-LiDAR Fusion for Environment Depth Estimation

Danyang Li, *Graduate Student Member, IEEE*, Jingao Xu, *Member, IEEE*, Zheng Yang, *Fellow, IEEE*, Qiang Ma, *Member, IEEE*, Li Zhang, and Pengpeng Chen

***Abstract*—Environment depth estimation by fusing camera and radar enables a broad spectrum of applications such as autonomous driving, environmental perception, context-aware localization and navigation. Various pioneering approaches have been proposed to achieve accurate and dense depth estimation by integrating vision and LiDAR through deep learning. However, due to the challenges of sparse sampling of in-vehicle LiDARs, high ground-truth annotation overhead, and severe dynamics in real environments, existing solutions have not yet achieved widespread deployment on commercial autonomous vehicles. In this paper, we propose LeoVR , a motion-inspired self-supervised visual-LiDAR fusion approach that enables accurate environment depth estimation. Leveraging the vehicle motion information, LeoVR employs two effective system frameworks to (i) optimize the depth estimation results, and (ii) provide supervision signals for DNN training. We fully implemented LeoVR on both a robotic testbed and a commercial vehicle and conducted extensive experiments over an 8-month period. The results demonstrate that LeoVR achieves remarkable performance with an average depth estimation error of 0.17 m, outperforming existing state-of-the-art solutions by $> 45.9\%$. Besides, even cold-start in real environments by self-supervised training, LeoVR still achieves an average error of 0.2 m, outperforming the related works by $> 47.8\%$ and comparable to supervised training methods.**

***Index Terms*—Depth estimation, factor graph, self-supervised learning, visual -LiDAR fusion.**

I. Introduction

ENVIRONMENT depth estimation aims at obtaining geometric properties of surrounding 3D space from 2D images and associated sensor inputs [1], [2], [3]. Typically, it generates a depth map[1] for each input image [4], [5], [6]. The benefit of depth estimation is to enhance the environmental perception capability of intelligent machines (robots, drones, vehicles, etc.), which lies in the heart of numerous applications such as autonomous driving [7], [8], [9] and robotics [10]. For instance, with accurate depth maps, vehicles and drones can avoid obstacles and better interact with environment [1], [11], [12]. Notably, recent reports suggest that incorrect depth estimation of ambient humans or objects has been identified as the primary cause of a majority of autonomous driving accidents since 2020 [13], [14], [15].

Current environment depth estimation practice on autonomous vehicles typically resorts to fusing camera and LiDAR. Compared to stand-alone visual approaches [6], [16], LiDAR can provide accurate 3D location of essential reflection points in the scene using time-of-flight (TOF), thus significantly compensating for the shortcomings of visual depth misestimation due to the lack of scale information. The state-of-the-art (SOTA) visual-LiDAR fusion approaches design Deep Neural Networks (DNNs) as depth map generators that directly take 2D visual images and associated 3D LiDAR point clouds as input and output the depth maps of surroundings [17], [18], [19].

Albeit inspiring, our 8-month field study reveals that previous solutions face significant challenges in deploying in real-world environments. The crucial drawbacks are twofold:

- *Degraded performance with commercial in-vehicle LiDAR:* The current practice of depth mapping in autonomous vehicles has demonstrated impressive results [1], [2]; however, it heavily relies on high-end and expensive LiDAR sensors such as Velodyne 16-line (VLP-16), 32-line (HDL-32E), and even 64-line (HDL-64E) [21], which come at a considerable cost of approximately $4 k, $20 k, and $80 k respectively, to generate dense point clouds. In contrast, due to the consideration of device cost, most vehicles are merely equipped with lower-cost yet sparsely sampled LiDARs (e.g., Livox Mid-40 costs $500 [22]). As illustrated in Fig. 1(b), Mid-40 generates merely one-third as many 3D points as VLP-16 LiDAR within a 0.1 s laser scan cycle and suffers from a narrower Field-of-View (FoV) coverage. With more sparse and irregular point clouds, the depth estimation performance of existing works degrades. As shown in Fig. 1(c) and (d), the estimation

Manuscript received 20 April 2023; revised 14 September 2023; accepted 15 November 2023. Date of publication 20 November 2023; date of current version 7 May 2024. This work was supported in part by the NSFC under Grants 62302254, 62372265, 61972131, 62272462, and 62202262 and in part by the Natural Science Foundation of Jiangsu Province for Distinguished Young Scholars under Grant BK20230045. An earlier version of this article appeared in International Conference on Mobile Systems, Applications, and Services (ACM MobiSys 2022). Recommended for acceptance by F. Wu. [DOI: 10.1145/3498361.3538918]. *(Danyang Li and Jingao Xu are co-first authors.) (Corresponding author: Zheng Yang.)*

Danyang Li, Jingao Xu, Zheng Yang, and Qiang Ma are with the School of Software and BNRist, Tsinghua University, Beijing 100190, China (e-mail: lidanyang1919@gmail.com; xujingao13@gmail.com; hmilyyz@gmail.com; tsinghuamq@gmail.com).

Li Zhang is with the School of Mathematics, Hefei University of Technology, Hefei, Anhui 230002, China (e-mail: hgdzli@gmail.com).

Pengpeng Chen is with the Department of Computer Science and Technology, China University of Mining and Technology, Xuzhou 221116, China (e-mail: believesea@gmail.com).

Digital Object Identifier 10.1109/TMC.2023.3334271

[1]In this work, a depth map is an additional image channel that contains information relating to the distance of the surfaces or objects from associated image pixels.

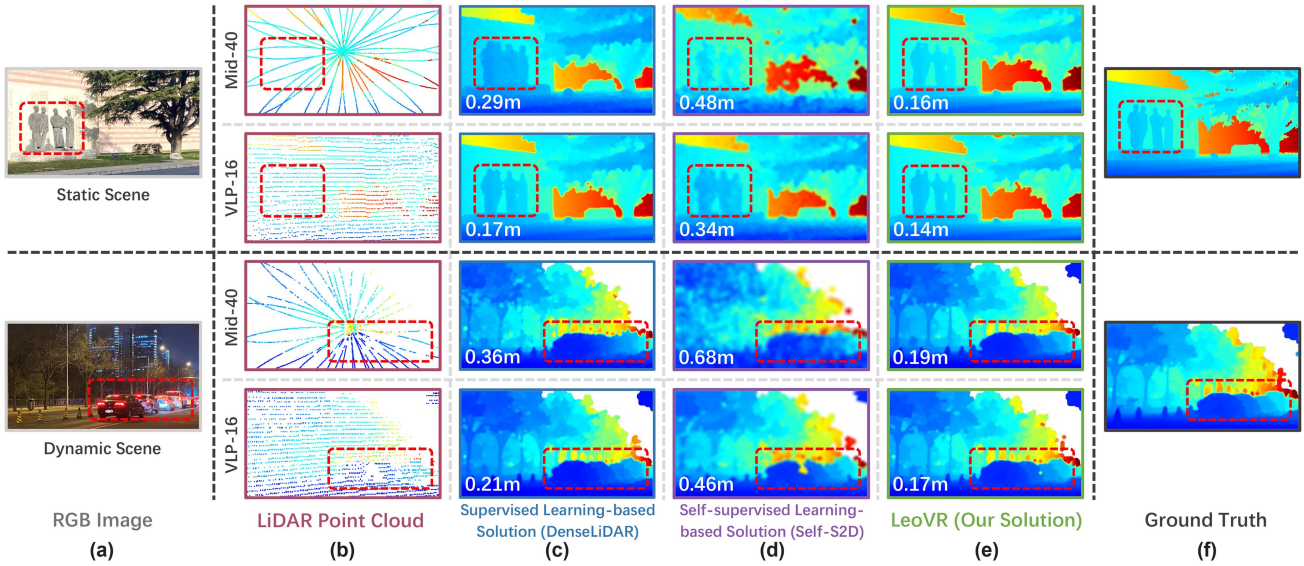


Fig. 1. Glimpse of depth estimation performance of LeoVR and existing solutions. (a) Visual images of the static and dynamic scene. (b) Associated LiDAR point clouds sampled by Mid-40 (cheap yet sparse) and VLP-16 (dense yet expensive). (c)-(e) are the depth map generated from state-of-the-art methods: DenseLiDAR [1] (a supervised learning-based solution), Self-S2D [20] (a self-supervised learning-based solution), and LeoVR (our solution), respectively. (f) is the ground truth. The bluer represents closer while the redder indicates further. The red dotted box in each picture bounds static figure sculptures or dynamic vehicles. And the value in the left-bottom corner is the average estimation error of all pixels in the depth map.

accuracy was reduced by almost 50% when equipped with the sparsely sampled Mid-40. We can also find that an accuracy drop of around 0.15 m would make it difficult to segment important objects in the current scene, as illustrated in Fig. 1(c), where the figure sculptures or vehicles become blurred and indistinguishable with Mid-40.

- *High ground-truth annotation overhead:* These deep learning based solutions typically need pixel-level depth ground truth annotations to train depth map generators in advance. What's worse, the cumbersome training procedure needs to be repeated for different environments, resulting in high labor cost and system overhead [1]. Although some recent works have proposed self-supervised training frameworks to deal with this issue [20], [23], [24], they extract training signals between consecutive frames and highly depend on the *static-rigid world assumption* [25], [26], [27], which is unrealistic in real complicated road conditions with reflections, shadows, and highly dynamic humans or objects (the bottom picture in Fig. 1(a)). As a consequence, the estimation performance of the self-trained model degrades in real environments. As illustrated in Fig. 1(d), compared to the depth maps generated in the static scene, the depth estimation error expands dramatically when the depth map generator cold starts by self-supervised training in a complex environment.

In this work, we aim to solve the above two challenges and propose LeoVR , a self-supervised solution that enables accurate environment depth estimation by fusing camera and LiDAR. Compared to current practice, LeoVR is profitable for generating depth maps with low-cost in-vehicle LiDARs, and the depth generator could be self-trained for cold starts even in real complicated environments. A comparison among LeoVR and related works are recorded in Table I, as well as an illustration in Fig. 1. As seen, LeoVR achieves remarkable performance even with the low-cost LiDAR. Our key insight behind LeoVR is that a vehicle's motion information could provide additional spatio-temporal constraints among successive depth maps generated by the vehicle (e.g., a depth map of the current frame could also be partially inferred by that of the previous frame and the inter-frame motion of the vehicle). These spatio-temporal constraints could be served as prior information to optimize the accuracy of depth maps (Section III-A), and on the other hand, provide guidance for extracting reliable pixel-level training signals to train the depth map generator (Section IV-A). Embedding this *motion-aware* information into the system framework, our design of LeoVR excels in two unique aspects as follows.

TABLE I
OVERALL SYSTEM COMPARISON

System	Average Depth Estimation Accuracy		Self-supervised
	VLP-16 ($4,000)	Mid-40 ($500)	
DeepLiDAR [2]	$0.24m$	$0.41m$	✗
DenseLiDAR [1]	$0.19m$	$0.32m$	✗
Self-S2D [20]	$0.33m$	$0.54m$	✓
LeoVR	$\mathbf{0.14}m$	$\mathbf{0.17}m$	✓

First, to push forward the accuracy of depth maps generated by fusing camera and low-cost LiDAR, at the core of LeoVR is a motion-aware **L***earning*-**e***mbedded* **o***ptimization scheme* for **V**isual-**R**adar fusion. Specifically, we design a factor-graph-based optimization framework which jointly optimizes: (i) the 3D locations of spatial feature points extracted from LiDAR samples and 2D visual images; (ii) the vehicle's motion and

pose estimated by visual-LiDAR odometry; and (iii) the depth maps generated by a DNN. Compared to related works, we do not hastily take the output of a depth map generator from DNN as the final depth map. On the contrary, we extract the intermediate results of the generator as *variable nodes* and refine the depth maps exploiting the spatio-temporal constraints (i.e., associated *factor nodes*) imposed by the vehicle's motion information. Furthermore, to fully leverage the complementary abilities of cross-modal inputs, we introduce an inter-sensor cross-attention mechanism in the depth map generator, enhancing the DNN's capacity to handle environmental uncertainty.

Second, to ease the burden of annotating ground truth for training the depth map generator, we propose a *motion-optical flow instructed self-supervised* framework. In LeoVR , with the awareness of vehicle motion, we exploit the pixel-level dense optical flow between adjacent frames and select pixels whose optical flow is consistent with the camera motion for training the DNN. The motion-optical flow consistency constraint could filter out those pixels whose photometric changes are disturbed by environmental dynamics (e.g., reflections, shadows, or highly dynamic humans or objects) rather than originating from camera's movement. Moreover, to further enhance the reliability of self-supervised training, we introduce a pose quality assurance strategy that utilizes global view information to obtain drift-free motion estimates as supervision signals. On this basis, unlike previous solutions, LeoVR achieves effective self-supervised training performance even in complicated real-world environments.

We have fully implemented LeoVR on a robotic testbed and intelligent vehicles with different types of cameras and LiDARs. Comprehensive experiments are carried out in four different scenarios (two indoor and two outdoor) across 8 months, collecting 3,720 trajectories with 1,126,550 frames. We compare the performance of LeoVR with two learning-based depth estimation methods (DenseLiDAR and DeepLiDAR). The experiment results show that LeoVR achieves an average depth estimation error of 0.141 m and 0.173 m when equipped with a Velodyne VLP-16 and Livox Mid-40 LiDAR respectively, outperforming comparative approaches by $> 25.7\%$ and $> 45.9\%$. We further evaluate the effectiveness of the proposed self-supervised framework with another two state-of-the-art self-supervised visual-LiDAR fusion solutions, Self-S2D and Self-VLO [23]. Without any pre-training and entirely based on self-supervised training, LeoVR still achieves an average depth estimation error of 0.201 m when equipped with Livox Mid-40, which outperforms related works by 47.8% and is comparable to those supervised training methods. Furthermore, as a universal approach to fusing sensing modalities for environment perception, we conducted a case study to showcase the versatility of LeoVR in adapting to indoor environments for RGB-D depth completion.

The key contributions are summarized as follows:

- We propose LeoVR , as far as we are aware of, the first self-supervised solution that enables commercial autonomous vehicles to generate accurate depth maps by fusing vision and low-cost LiDAR. LeoVR pushes forward depth estimation techniques for on-vehicle, low-cost, and large-scale deployments in real environments.

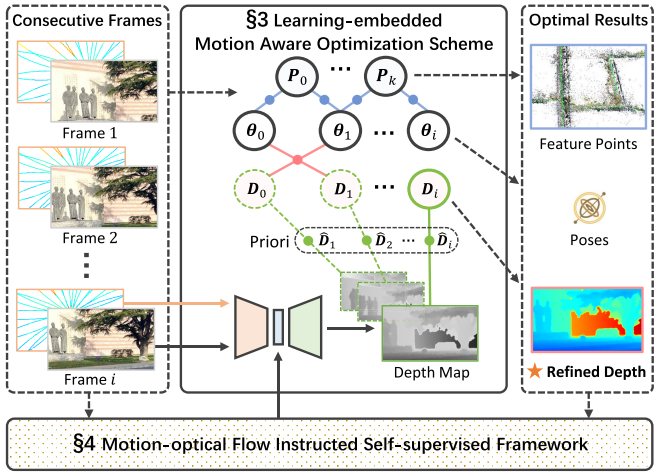


Fig. 2. System architecture of LeoVR .

II. System Overview

Fig. 2 sketches the system architecture of LeoVR . From the top perspective, LeoVR consists of two components: a *learning-embedded motion-aware optimization scheme* (Section III) and a *motion-optical flow instructed self-supervised framework* (Section IV). The former part aims for visual-LiDAR fusion and resulting accurate depth maps estimation. The latter supports the self-supervised training of the depth map generator in a DNN and reduces the ground truth annotation costs.

Specifically, LeoVR takes consecutive time-synchronized monocular RGB images and low-cost LiDAR measurements as inputs. Then, a DNN, named depth map generator, generates initial dense maps which will be further extracted as depth *variable nodes* with prior depth information in a *factor graph*. Thereafter, the factor graph leverages visual and radar inputs as well as the depth *variable nodes* to joint optimize (i) the vehicle's motion; (ii) 3D point clouds; and (iii) acquire refined dense depth maps. Furthermore, these optimized results will be exploited as an instructor to select which pixels could be used for training

the model. Based on this two-way promotion process, LeoVR achieves a promising depth estimation performance through a self-supervised training manner.

III. Learning-Embedded Motion Aware Optimization Scheme

In this section, we present the design of the motion-aware learning-embedded optimization scheme for visual-LiDAR fusion that enables LeoVR to achieve an accurate depth estimation performance. In LeoVR , we use the depth map generator proposed in [28] as the backbone of our model, which ta kes a 2D RGB image and 3D LiDAR point clouds as input and outputs a depth map. To facilitate the complementary interaction between sensors, we incorporate a cross-attention module into the depth map generator, which enables the efficient fusion of features extracted from both modalities. Instead of directly treating the outputs of the depth generator as the final depth maps, LeoVR leverage a *factor-graph* based optimization framework exploiting these intermediate results to jointly optimize the point clouds, vehicle (i.e., camera and attached LiDAR's) motion, and depth maps within a time window. Briefly, an optimization takes consecutive visual images, corresponding LiDAR samples, and intermediate depth information extracted from a depth map generator as inputs and then refines the depth of each pixel in the images. In what follows, we first describe and illustrate some essential variables and definitions of the optimization problem, as well as analyze the rationale behind our insight that joint optimization could improve the depth estimation accuracy. The upcoming section entails an elucidation of crucial variables and definitions of the optimization problem. Additionally, we provide a detailed explanation of the rationale behind our hypothesis that joint optimization has the potential to enhance the accuracy of depth estimation (Section III-A). Then, we provide a brief overview of the network architecture and analyze the underlying mechanism of the cross-attention mechanism in effectively integrating multi-modal features (Section III-B). Further, we present how to formulate the *factor graph* with associated *factor nodes* and *variable nodes* to solve the optimization problem (Section III-C).

A. Optimization Problem Statement

LeoVR utilizes three distinct reference systems: the camera reference system C, the LiDAR reference system L, and the world reference system W. To simplify the notation, we omit the fixed transforms of the camera and LiDAR reference systems, which are rigidly attached to the vehicle. The objective of optimization is to refine the depth map provided by the depth map generator. To exploit the spatial correlation between consecutive depth maps, the pose of the vehicle, represented as a 6-DoF transformation from W to C, is continuously estimated. Specifically, we define the pose at frame i as follows:

$$\boldsymbol{\theta}_i \triangleq \{\boldsymbol{R}_i, \boldsymbol{t}_i\} \in \mathrm{SE}(3), \tag{1}$$

where $\boldsymbol{R}_i$ and $\boldsymbol{t}_i$ are rotation and translation, respectively. Additionally, we also optimize the 3D feature point extracted from multi-view visual images to maintain global consistency of optimization.

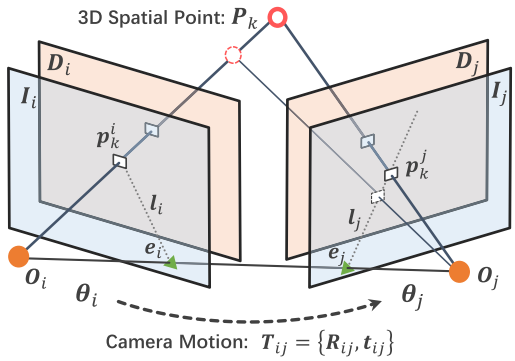


Fig. 3. Illustration of essential variables and their spatial relationships.

Rationale behind the joint optimization: As illustrated in Fig. 3, the spatial location of a 3D feature point $\boldsymbol{P}_k$ and its associated 2D pixel location $\boldsymbol{p}_k^i$, $\boldsymbol{p}_k^j$ on visual images $\boldsymbol{I}_i$, $\boldsymbol{I}_j$ at timestamp t_i and t_j are relevant to the LiDAR samples and camera's motion (i.e., pose transformation $\boldsymbol{T}_{ij}$). Further, the depth value of pixels $\boldsymbol{p}_k^i$ and $\boldsymbol{p}_k^j$ on depth maps, $\boldsymbol{D}_i(\boldsymbol{p}_k^i)$ and $\boldsymbol{D}_j(\boldsymbol{p}_k^j)$, in addition to acquiring through the depth generator, could also be determined by the spatial location of $\boldsymbol{P}_k$ and camera's motion $\boldsymbol{T}_{ij}$. In a nutshell, for each pixel on the depth map, we have two independent yet complementary ways to estimate its depth. Intuitively, we could integrate these two different approaches to achieve higher depth estimation accuracy. To this end, LeoVR proposes a joint optimization framework based on the probabilistic characteristics of these two depth estimation methods.

Optimization goals: The overall optimization objective is to calculate the scene depths, poses, and feature points visible up to the current time t_c:

$$\mathcal{X}_c \triangleq \bigcup_{i \in \mathcal{F}_c} \{\boldsymbol{D}_i, \boldsymbol{\theta}_i\} \bigcup_{k \in \mathcal{P}_c} \{\boldsymbol{P}_k\}, \tag{2}$$

where $\mathcal{F}_c$ is a list of frames within a fixed lag smoothing window, and $\mathcal{P}_c$ is a set of feature points observed by those frames.

We aim to maximize the likelihood of measurements from the monocular camera, LiDAR, and depth map generator, given the history of states:

$$\mathcal{X}_c^* = \arg\max_{\mathcal{X}_c} p\left(\mathcal{X}_c \mid \mathcal{Z}_c\right) \propto p\left(\mathcal{X}_0\right) p\left(\mathcal{Z}_c \mid \mathcal{X}_c\right), \tag{3}$$

where $\mathcal{Z}_c$ is the set of measurements received within the smoothing window, $\mathcal{X}_c$ represents the state variables, and $\mathcal{X}_0$ is the prior state. We express the above equation as a nonlinear least squares problem:

$$\mathcal{X}_c^* = \arg\min_{\mathcal{X}_c} \sum_{i \in \mathcal{F}_c} \Bigg(\sum_{k \in \mathcal{P}_i} \|\boldsymbol{E}_{vision}(\boldsymbol{\theta}_i, \boldsymbol{P}_k)\|_{\Sigma_v}^2 + \sum_{k \in \mathcal{P}_i} \|\boldsymbol{E}_{lidar}(\boldsymbol{\theta}_i, \boldsymbol{P}_k)\|_{\Sigma_l}^2 + \sum_{j \in \mathcal{N}_i} \|\boldsymbol{E}_{depth}(\boldsymbol{D}_i, \boldsymbol{D}_j)\|_{\Sigma_d}^2 \Bigg), \tag{4}$$

where $\mathcal{N}_i$ represents the set of nearby frames of frame i. Each term in the above equation represents the residual associated

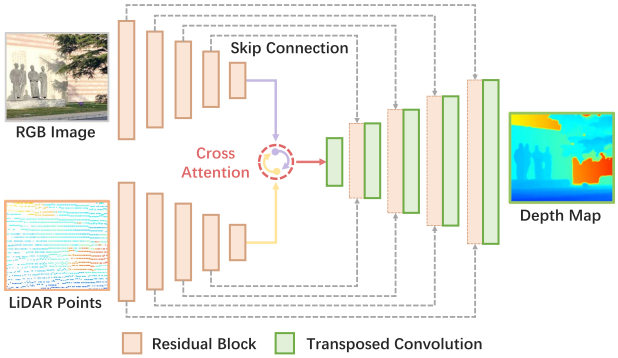


Fig. 4. Network architecture of the proposed *depth map generator*.

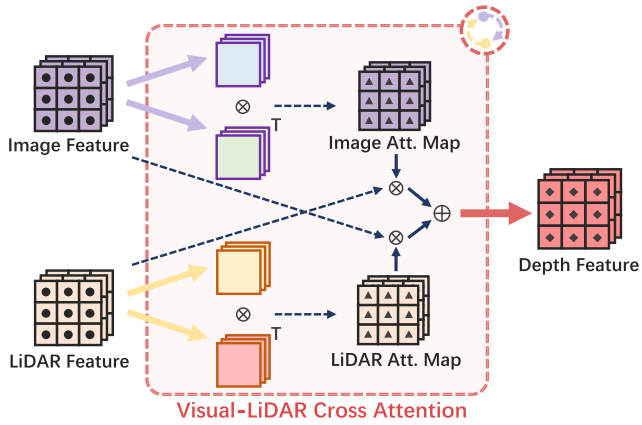


Fig. 5. Workflow of the *visual-LiDAR cross attention* mechanism.

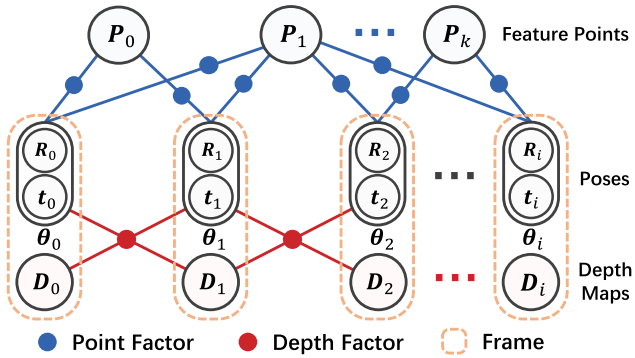


Fig. 6. Overview of the *factor graph* in LeoVR .

with a particular factor type, and is weighted by the corresponding information matrix Σ (i.e., inverse of the covariance matrix, where $\|\boldsymbol{E}\|_{\Sigma}^{2} = \boldsymbol{E}^{\mathrm{T}}\Sigma\boldsymbol{E}$). Specifically, $\boldsymbol{E}_{vision}$ and $\boldsymbol{E}_{lidar}$ represent two types of feature point residuals, whereas $\boldsymbol{E}_{depth}$ represents the residual associated with the depth map.

B. Depth Map Generator

Network Architecture: Our depth map generator utilizes an encoder-decoder structure and includes three main components: an RGB image encoder, a LiDAR points encoder, and a depth estimation decoder, as shown in Fig. 4. Features from each modality are fused using cross-attention to promote complementary interaction. Furthermore, skip connections are employed at each scale to fuse the features from the encoder branches and the decoder to recover fine-grained details in the depth map.

Inter-sensor Cross-attention: We proposed a cross-attention mechanism to equip the multi-modal depth map generator with the ability to adapt to environmental uncertainties. The fundamental idea behind attention is to selectively emphasize features that are more informative for the task at hand, while downplaying less relevant ones, which is particularly useful when dealing with non-local dependencies between input modalities [19], [29]. By selectively attending to important features in both the image and LiDAR inputs, our network can effectively fuse information from the two modalities, leading to more accurate and robust depth estimation results in diverse environments.

To facilitate communication between the camera and LiDAR sensors, we first propose generating attention maps across their respective features, as shown in Fig. 5. Specifically, we obtain the image feature $\mathbf{z}_I$ and LiDAR feature $\mathbf{z}_L$ from their feature extractors, and then derive two corresponding attention maps:

$$\begin{aligned} \boldsymbol{a}_I &= \text{Softmax}\left((\mathbf{W}_\theta \mathbf{z}_I)^T (\mathbf{W}_\phi \mathbf{z}_I)\right), \\ \boldsymbol{a}_L &= \text{Softmax}\left((\mathbf{W}_\varphi \mathbf{z}_L)^T (\mathbf{W}_\delta \mathbf{z}_L)\right), \end{aligned} \tag{5}$$

where Softmax denotes the non-linear activation function, and the family of $\mathbf{W}$ are learnable weight matrices to project the original feature into an latent space.

Using above attention map and original features, we can obtain the cross-attention depth feature map, denoted as $\mathbf{a}_D$. Specifically, we first perform element-wise multiplication between the attention maps and the complementary feature, i.e., $\boldsymbol{a}_I \odot \mathbf{z}_L$ and $\boldsymbol{a}_L \odot \mathbf{z}_I$. Then, we concatenate these two feature maps using the $\oplus$ operator. Hence, the depth feature output by the cross-attention module can be formulated as:

$$\boldsymbol{a}_D = (\boldsymbol{a}_I \odot \mathbf{z}_L) \oplus (\boldsymbol{a}_L \odot \mathbf{z}_I). \tag{6}$$

The efficacy of cross-attention can be ascribed to its potential to extract global correlations from diverse modalities, thus surmounting the limitations of relying solely on local features of a single sensor. This is particularly relevant in the context of depth estimation tasks that necessitate inputs from both RGB and LiDAR sensors, since the correct context for predicting pixel depth may not necessarily be confined to adjacent locations of current sensor features.

The cross-attention module offers efficient computation, particularly for high-level, downsampled features. The computational complexity of its pairwise computation, as described in (5), is comparable to that of a typical convolutional layer [19]. The efficiency of the cross-attention module is further exemplified in our model architecture, where it handles compact input features with dimensions of width = 14, height = 8, and channel = 128, as detailed in Section V-A.

C. Factor Graph Formulation

The structure of the factor graph is shown in Fig. 6. The factor graph consists of two types of nodes: *variable nodes* indicate the values to be optimized (i.e., depth maps $\boldsymbol{D}_i$, poses

$\boldsymbol{\theta}_i$, and feature points $\boldsymbol{P}_k$), while *factor nodes* represent the probability relationship between two variable nodes (i.e., *point factor* associating pose with feature point, and *depth factor* associating adjacent poses and depth maps). We optimize all variable nodes simultaneously to obtain globally optimal results. We describe the measurements and residuals of two types of point factors, followed by depth factor.

1) Point Factor With Visual Reprojection Error: To optimize the vehicle's poses and feature points, we first analyze the observation of environment by the camera. We consider a conventional pinhole camera model with a projection function $\boldsymbol{\pi} : \mathrm{SE}(3) \times \mathbb{R}^3 \rightarrow \mathbb{R}^2$, which transforms a 3D point $\boldsymbol{P}$ in world reference $\mathtt{W}$ to the image plane given a vehicle's pose $\boldsymbol{\theta}$:

$$\boldsymbol{\pi}\left(\boldsymbol{\theta}, \boldsymbol{P}\right) = \frac{1}{Z}\boldsymbol{K}\boldsymbol{\theta}\boldsymbol{P},\ \boldsymbol{P} = [X, Y, Z]^T, \tag{7}$$

where $\boldsymbol{K}$ is the camera intrinsic matrix. Let $\boldsymbol{P}_k \in \mathbb{R}^3$ denote a 3D feature point and $\boldsymbol{p}_k^i \in \mathbb{R}^2$ denote the corresponding detection on the image plane $\boldsymbol{I}_i$. The residual at pose $\boldsymbol{\theta}_i$ for feature point $\boldsymbol{P}_k$ can be formulated as:

$$\boldsymbol{E}_{vision}(\boldsymbol{\theta}_i, \boldsymbol{P}_k) = \boldsymbol{\pi}\left(\boldsymbol{\theta}_i, \boldsymbol{P}_k\right) - \boldsymbol{p}_k^i. \tag{8}$$

This point factor measures the difference between the pixel location of the observed feature point and the re-projection location of the estimated 3D feature point due to the unknown pose and noises of measurements. We use ORB [30] to detect and describe visual feature points in images.

2) Point Factor With LiDAR Constraint: In order to fully leverage the benefits of combining vision and LiDAR sensing modalities, we further refine our feature point optimization by utilizing LiDAR's overlapping field-of-view to obtain depth estimates. Specifically, we begin by projecting all 3D LiDAR samples $\boldsymbol{L}$ in the LiDAR reference $\mathtt{L}$ between time t_c and t_{c+1} onto the image plane, as follows:

$$\boldsymbol{\pi}(\boldsymbol{L}) = \frac{1}{Z}\boldsymbol{K}\boldsymbol{L},\ \boldsymbol{L} = [X, Y, Z]^T. \tag{9}$$

As the LiDAR points are sampled at different times with different poses due to the continuous motion of the vehicle, we adopt a widely used motion compensation technique based on linear interpolation [31]. By projecting all in-frame LiDAR points onto the current view, we can effectively eliminate the impact of motion blur and enhance the alignment between LiDAR points and visual features.

Next, given a 3D feature point $\boldsymbol{P}_k$ in the world reference $\mathtt{W}$ with the corresponding pixel $\boldsymbol{p}_k^i$ on the current image plane, we search for the projected LiDAR point $\boldsymbol{\pi}(\boldsymbol{L}_k)$ that is closest to $\boldsymbol{p}_k^i$ within a neighborhood of 5 pixels, and apply classical outliers rejection [32] to remove erroneous matches. Finally, the residual is computed as follows:

$$\boldsymbol{E}_{lidar}(\boldsymbol{\theta}_i, \boldsymbol{P}_k) = \boldsymbol{\theta}_i\boldsymbol{P}_k - \boldsymbol{L}_k. \tag{10}$$

Our point factor incorporating LiDAR constraint penalizes deviations between the depth of a feature point and its corresponding LiDAR observation, optimizing both the feature points and the vehicle's pose while simultaneously applying scale to visual measurements. In situations where there is insufficient LiDAR data to associate a depth measurement with a feature point due to sparse input LiDAR samples, we rely solely on our $\boldsymbol{E}_{vision}$ as the point factor.

3) Depth Factors With Geometric Consistency: To fully exploit the capabilities of the depth map from the generator and further optimize the scene depth in a fine-grained manner, we adopt the geometric consistency of the scene depth from different views as a constraint. Given two consecutive depth maps $\boldsymbol{D}_i$ and $\boldsymbol{D}_j$, we can project $\boldsymbol{D}_j$ to the view of $\boldsymbol{D}_i$ based on the relative pose transformation $\boldsymbol{T}_{ij} = \boldsymbol{\theta}_j\boldsymbol{\theta}_i^{-1}$. First, let $\boldsymbol{p}_k^i$ denote the coordinates of a pixel in $\boldsymbol{D}_i$, and the corresponding 3D location is:

$$\boldsymbol{P}_k = \boldsymbol{D}_i(\boldsymbol{p}_k^i)\boldsymbol{K}^{-1}\boldsymbol{p}_k^i. \tag{11}$$

Then, we project $\boldsymbol{P}_k$ to the image plane of $\boldsymbol{D}_j$ with $\boldsymbol{\pi}(\boldsymbol{T}_{ij}, \boldsymbol{P}_k)$. Therefore, we can obtain a depth map which is projected from $\boldsymbol{D}_j$ to the view of $\boldsymbol{D}_i$:

$$\boldsymbol{D}_{j\rightarrow i}(\boldsymbol{p}_k^i) = \boldsymbol{D}_j\left(\boldsymbol{\pi}\left(\boldsymbol{T}_{ij}, \boldsymbol{P}_k\right)\right). \tag{12}$$

When the time interval between two successive frames is small, it is expected that the geometric properties observed from different viewpoints are consistent. Consequently, the residual between depth maps can be formulated as:

$$\boldsymbol{E}_{depth}(\boldsymbol{D}_i, \boldsymbol{D}_j) = \sum_{k\in\Omega}\left\|\boldsymbol{D}_i(\boldsymbol{p}_k^i) - \boldsymbol{D}_{j\rightarrow i}(\boldsymbol{p}_k^i)\right\|^2. \tag{13}$$

The depth factor plays a crucial role in ensuring the continuity and consistency of the estimated depth maps from different views of the scene geometry. To ensure faster convergence, we adopt a stochastic optimization approach by sampling a different set of pixels at each iteration to optimize the residual over the entire depth map. Additionally, a potential limitation associated with this factor pertains to the asynchronous sampling of depth maps from varied viewpoints. Dynamic objects in the scene present a particular challenge. Their states at distinct moments aren't directly tied to the vehicle's motion. Consequently, when this factor is applied for their optimization, the depth map of current frame might be influenced by inconsistent past states. This misalignment can compromise both depth and pose optimization, especially for rapidly moving objects. Thus, to ensure a reliable optimization, we propose masking the incorrect correspondences stemming from these dynamics. Specifically, only pixels of objects within a trusted region [20] are considered for optimization. This trusted region is defined as the convex hull in the pixel space, formed by the consistent 3D feature points extracted and triangulated from visual odometry on static objects.

IV. Motion-Optical Flow Instructed Self-Supervised Framework

To ease the ground truth annotation costs for training the depth map generator, we design a self-supervised framework. Our solution could automatically extract supervision signals to train the DNN by leveraging the camera's motion information. The basic idea behind the framework is illustrated in the left part of Fig. 7. As seen, given the current image $\boldsymbol{I}_i$, the nearby

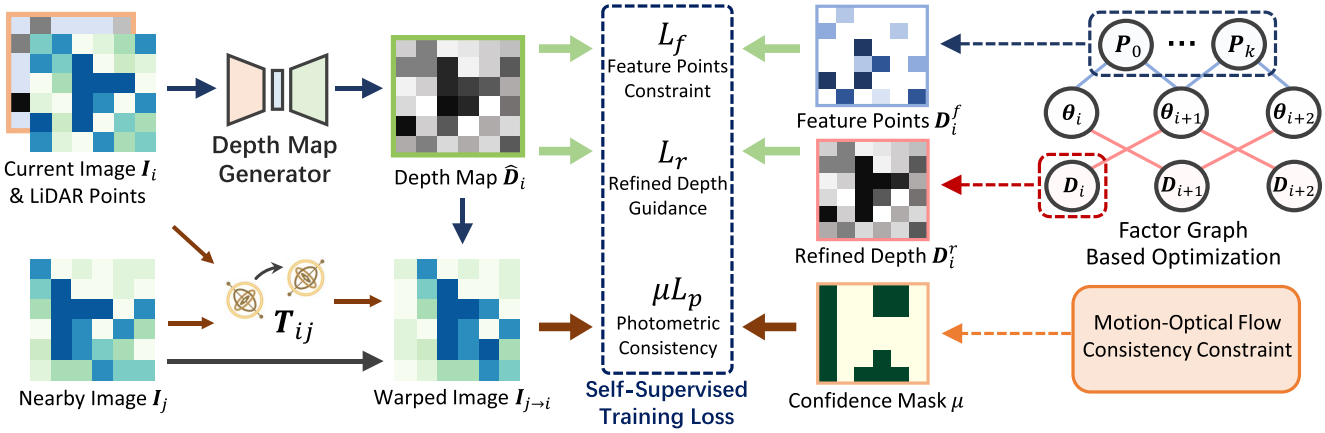


Fig. 7. Workflow of the *motion-optical flow instructed self-supervised framework* in LeoVR .

image $\boldsymbol{I}_j$, the depth map $\hat{\boldsymbol{D}}_i$ output from the DNN generator, as well as the inter-frame camera's motion $\boldsymbol{T}_{ij}$, we can inversely wrap $\boldsymbol{I}_j$ to the view of current image and generate a warped image $\boldsymbol{I}_{j\to i}$. The pixel-level photometric differences between the current image $\boldsymbol{I}_i$ and the warped image $\boldsymbol{I}_{j\to i}$ could be served as supervision signals [20]. This section details the implementation of our basic idea (Section IV-A) in combination with a pose quality assurance strategy for stable model training (Section IV-B). Furthermore, we introduce a *motion-optical flow consistency* constraint in Section IV-C to enhance the robustness of our framework in complex real-world environments. We also present two additional constraints in Section IV-D that can be incorporated to further improve the training performance.

A. The Basic Idea: Photometric Loss

As mentioned above, the basic supervision signal for training the depth map generator comes from the *photometric consistency*. Let $\boldsymbol{p}_k^i$ denote the pixel coordinates in image $\boldsymbol{I}_i$, and any $\boldsymbol{p}_k^i$ with a corresponding projection in $\boldsymbol{I}_j$ can be obtained as:

$$\boldsymbol{p}_k^j = \boldsymbol{\pi}\left(\boldsymbol{T}_{ij}, \hat{\boldsymbol{D}}_i\left(\boldsymbol{p}_k^i\right)\boldsymbol{K}^{-1}\boldsymbol{p}_k^i\right), \tag{14}$$

where the motion information $\boldsymbol{T}_{ij}$ can be obtained from our factor graph based optimization framework, as previously discussed. Thus, the warped image $\boldsymbol{I}_{j\to i}$ can be created as follows:

$$\boldsymbol{I}_{j\to i}\left(\boldsymbol{p}_k^i\right) = \boldsymbol{I}_j\langle\boldsymbol{p}_k^j\rangle, \tag{15}$$

where $\langle\cdot\rangle$ is the differentiable bilinear sampler [33] that interpolates around the four immediate neighbours of $\boldsymbol{p}_k^j$. Next, we train the model to generate the depth map $\hat{\boldsymbol{D}}_i$ by minimizing the difference between the source image $\boldsymbol{I}_i$ and warped image $\boldsymbol{I}_{j\to i}$. To this end, we define the photometric loss as follows:

$$L_p = \sum_{j\in\mathcal{N}_i} pe\left(\boldsymbol{I}_i, \boldsymbol{I}_{j\to i}\right), \tag{16}$$

where $\mathcal{N}_i$ is the set of image that nearby the current view. To prevent the noise from large differences in geometric views, only the previous and next one of the current frame are used as reference in practical. We using a combination of the average pixel reprojection residual with an L_1 penalty and SSIM [34] to obtain the *photometric error* (i.e., pe in (16)), a perceptual metric that is invariant to local illumination changes:

$$pe\left(\boldsymbol{I}_a, \boldsymbol{I}_b\right) = \frac{\alpha}{2}\left(1 - \text{SSIM}\left(\boldsymbol{I}_a, \boldsymbol{I}_b\right)\right) + \left(1-\alpha\right)\left\|\boldsymbol{I}_a - \boldsymbol{I}_b\right\|_1, \tag{17}$$

where α is the parameter that regulates the sensitivity to local illumination changes, and we set $\alpha = 0.8$ to balance the training.

B. Pose Quality Assurance

The basic idea underlying the self-supervised training of the depth map generator relies on utilizing motion information between adjacent frames to obtain necessary supervision signals. However, inaccurate motion cues can lead to erroneous projections as per formula (14), resulting in inaccurate signals and impaired model training.

To minimize the relative motion estimation error among adjacent frames within the trajectory designated for model training, we introduce a pose quality assurance strategy. As showcased in Fig. 8, by employing the loop-closure [35] and incorporating pre-planned loopback data collection, we mitigate the accumulated loop errors, ensuring a more accurate pose transformation between neighboring frames. The strategy begins with a **Loop Detection** module that identifies re-visited locations. Subsequently, feature-level connections are established between loop candidates for **Loop Validation**. In the final step, **Loop Correction** integrates these feature correspondences into the factor graph as a *loop-closure factor node* [36], delivering drift-free state estimates. The detailed steps of the pose quality assurance strategy using loop-closure are as follows.

Loop Detection: For each frame within a trajectory, we evaluate potential loops. Using DBoW2 [37], we determine the similarity between frame i and its co-visible neighbors based on their bag of words vectors. A higher score signifies a greater degree of similarity. This analysis yields a minimum similarity score, represented as s_i. Subsequently, we examine the other frames in the trajectory, discarding those with scores falling

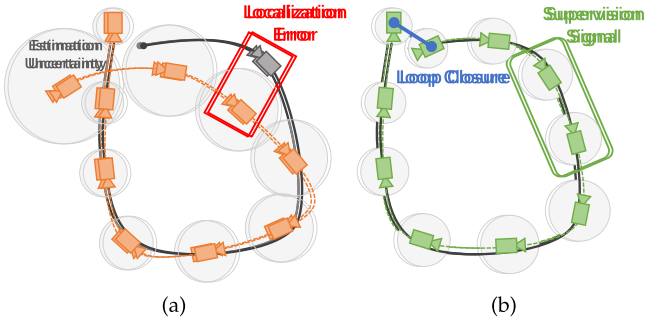


Fig. 8. Pose quality assurance strategy using loop-closure. (a) Trajectory prior to loop-closure. (b) Trajectory following loop-closure. The black line depicts the ground truth trajectory, while the gray bubbles represent the uncertainty in the estimated poses, with larger circle indicating greater cumulative errors. Upon loop-closure optimization, the trajectory aligns tightly with the ground truth. This alignment augments the quality of supervision signals, drawing from accurate motion information.

beneath s_i. Frames surpassing this threshold are categorized as loop candidates. To fortify the accuracy of loop detection, we prioritize loop candidates that appear consecutively for more than three frames.

Loop Validation: To validate potential loops, we enhance the assessment of similarity between frame i and its loop candidates. We first use the ORB features to match their observations and find the 3D-3D map points correspondence so as to compute the Sim(3) transformation, denoted as $\mathbf{S}_i$. If $\mathbf{S}_i$ supports a sufficient number of inliers (i.e., the reprojection error is within an acceptable limit), we accept this loop candidate. Subsequently, the loop is closed by applying the similarity transformation $\mathbf{S}_i$ to align the frames at both ends of the loop.

Loop Correction: To correct the detected loop, we first adjust the pose of the frame at the end of the loop using the similarity transformation $\mathbf{S}_i$. This correction is then progressively applied to all neighbors of the frame i, ensuring alignment of co-visible observations at both ends. Subsequently, the poses of frames along the trajectory, together with the 3D map points, are optimized using the factor graph as delineated in Section III-C. Through the loop correction process, we compensate for and mitigate the loop-closing error, ensuring an accurate alignment of both poses and map points throughout the graph.

While this strategy integrates an additional computational step, no discernible increase in computational costs arises during the training phase. On one hand, loop-closure, applied before model training, processes efficiently with an average latency of less than 20 s per trajectory [38], [39], a stark contrast to the demands of model training. On the other hand, the improved pose quality notably quickens the model convergence, leading to fewer training iterations and a consequent time-saving.

To shed light on the rationale behind the pose quality assurance strategy using loop-closure, we present a concrete example in Fig. 8. Before the loop-closure (Fig. 8(a)), the pose estimation error gradually accumulates with vehicle motion and eventually deviates significantly from the ground truth. However, as shown in Fig. 8(b), after loop-closure, the estimation uncertainty of endpoint and its neighbors is reduced. The elimination of accumulated errors enhances the transformation accuracy between adjacent frames, thereby ensuring the provision of dependable motion information for model training. Furthermore, we provide a quantitative evaluation in Section V-B3.

C. Motion-Optical Flow Constraint

Although one can leverage (16) to train the depth map generator from continuous video frames, the above photometric reprojection formulation implies the *static-rigid world assumption*, where the current scene is: (i) static without moving objects; (ii) no occlusion/disocclusion between current and nearby view; and (iii) a Lambertian surface[2] to ensure the photometric invariance assumption holds [25]. However, the real environments with high dynamics (e.g., reflections, shadows, moving objects) hardly satisfy the above assumption, resulting in a higher penalty and corrupted gradients even if the DNN predicts a correct depth for each pixel, which eventually degrades the self-supervised training performance.

To improve the robustness of self-supervised training with photometric consistency, we propose a *motion-optical flow consistency constraint* to select pixels that satisfy the static-rigid world assumption for training. Our key insight is that the photometric variation of ideal pixels between adjacent images should be entirely attributed to the vehicle's motion. This insight stems from two observations: (i) the projection of static spatial points across two frames can be determined by the camera's motion information, while for dynamic objects, their own movement also plays a role; (ii) even for static objects, points on occluded regions or non-Lambertian surfaces exhibit varying brightness in their corresponding projections between two frames.

We incorporate optical flow[3] to translate this insight into practice. Given its ability to capture the motion of objects on the image plane through photometric consistency, only those pixels whose optical flow vector aligns with the camera's motion vector meet the static-rigid world assumption and are thus used to extract supervision signals.

Specifically, to select the motion-optical flow consistent pixels, we first compute the optical flow vector for each pixel between the current frame $\boldsymbol{I}_i$ and its nearby frame $\boldsymbol{I}_j$ based on the photometric invariance assumption [42]. Thus, given a 3D spatial point with pixel coordinates $\boldsymbol{p}_k^i$ in $\boldsymbol{I}_i$, we can find the corresponding pixel $\boldsymbol{p}_k^j$ in $\boldsymbol{I}_j$ according to the optical flow vector. To ensure that the selected pixels are consistent with camera motion, we analyze the geometric relationship between camera motion and spatial point, as illustrated in Fig. 3. The intersection of the optical centers $\boldsymbol{O}_i$ and $\boldsymbol{O}_j$ with the 3D point $\boldsymbol{P}_k$ forms an epipolar plane. This plane intersects with the image plane to form two epipolar lines $\boldsymbol{l}_i$ and $\boldsymbol{l}_j$. Once the camera motion is determined, the possible location of $\boldsymbol{P}_k$ lies on the ray from $\boldsymbol{O}_i$ to $\boldsymbol{p}_k^i$, and the corresponding potential projection location on

[2]The apparent brightness of a Lambertian surface to an observer is the same regardless of the observer's angle of view.

[3]Optical flow captures the motion of objects between consecutive frames, representing it as a 2D vector field where each vector indicates the pixel's displacement from one frame to the next [41].

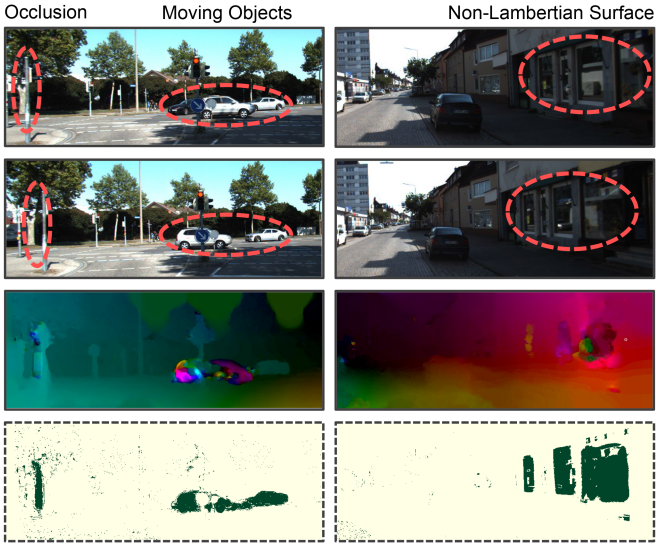


Fig. 9. Qualitative results of motion-optical flow consistency constraint. We show two complicated real-world scenarios from the KITTI dataset [40] that severely impact self-supervised training. The first and second rows show consecutive RGB frames collected during the vehicle's movement. The third row shows the visualized dense optical flow between two frames. The last row depicts the generated confidence masks. Pixels that lie in the darker region will be filtered out and not be leveraged to train the DNN.

the image $\boldsymbol{I}_j$ is on $\boldsymbol{l}_j$. Thus, if $\boldsymbol{p}_k^i$ is consistent with the camera motion, $\boldsymbol{p}_k^j$ calculated by optical flow should lie on $\boldsymbol{l}_j$. We can verify the consistency using the epipolar constraint [43] that evaluates the distance from the projection point to the epipolar line, given by:

$$\left(\boldsymbol{p}_k^j\right)^T \boldsymbol{F} \boldsymbol{p}_k^i = 0, \tag{18}$$

where the fundamental matrix $\boldsymbol{F}$ can be calculated from the camera motion [44]. The above equation holds strictly when the optical flow is completely consistent with the motion. Based on this constraint, we compute confidence mask μ for loss L_p, which filters out pixels that violate motion-optical flow consistency:

$$\mu = \left[\left| \left(\boldsymbol{p}_k^j\right)^T \boldsymbol{F} \boldsymbol{p}_k^i \right| < \lambda \right], \tag{19}$$

where $[\cdot]$ is the Iverson bracket, and λ is the constraint threshold. As shown in Fig. 9, we qualitatively demonstrate the effectiveness of motion-optical flow consistency constraint. In both complicated scenes, the generated per-pixel confidence mask can accurately filter the invalid pixels from occlusion, dynamics, and non-Lambertian surfaces. We also experimentally show that the proposed constraint can solve these factors and bring significant improvements for the self-supervised training (Section V-D2).

D. Additional Constraints

In addition to the vehicle (i.e., camera's) motion information, we also take full advantage of the output of the optimization framework (Section III-C), including the optimized 3D feature points and refined depth maps. Further, we introduce two additional constraints to facilitate the training of the depth map generator. The training performance gain will be demonstrated in Section V-D3.

Feature Points Constraint: As described in Section III, in addition to optimizing the depth maps and poses, we also track all 3D feature points in the global trajectory. We project the feature points which are visible in the current frame $\boldsymbol{I}_i$ onto the image plane and form a sparse depth map $\boldsymbol{D}_i^f$. The feature points supervised depth loss is defined as:

$$L_f = \left\| \left(\hat{\boldsymbol{D}}_i - \boldsymbol{D}_i^f \right) \odot \left[\boldsymbol{D}_i^f > 0 \right] \right\|^2, \tag{20}$$

where $\odot$ is an element-wise multiplication. Since $\boldsymbol{D}_i^f$ is sparse and contains invalid pixels, we only consider those are having valid depth values. This supervision signal delivers higher accuracy, better stability, and faster convergence during model initialization for self-supervised training.

Refined Depth Guidance: In the refinement stage of model training, the factor graph based optimization refines each depth map predicted by the generator. With the guidance from the refined depth $\boldsymbol{D}_i^r$, the loss is defined as follows:

$$L_r = \left\| \hat{\boldsymbol{D}}_i - \boldsymbol{D}_i^r \right\|^2. \tag{21}$$

In practice, introducing L_r in the fine-tuning stage can effectively improve the accuracy, but still a small number of incorrect refined depth maps can cause model training instability. For stable training, we only select those refined depth maps that are geometrically consistent with the optimized feature point cloud for further guidance.

E. Put Together

For the smoothness of generated depth maps, we further use edge-aware smoothness loss L_s to minimize the L_1 norm of the second-order gradient for the depth prediction, and the form of L_s is similar to [5], [20]. Putting these constraints together, our final objective for the entire self-supervised training is:

$$L = \mu L_p + \lambda_f L_f + \lambda_r L_r + \lambda_s L_s, \tag{22}$$

where λ_f, λ_s, and λ_r are the relative weightings to balance the terms. We will describe the training details in Section V-A.

V. Implementation and Evaluation

A. Experimental Methodology

Device Setup: We prototype LeoVR on a robotic testbed (indoor experimental scenarios) and commercial vehicle (outdoor scenarios) with different cameras and LiDARs. The robot and vehicle are equipped with a Logitech C920E (1080p, 30 Hz) and Ladybug5+ (2 k, 30 Hz) RGB camera for capturing images respectively. We also equip both VLP-16 ($4,000) and Mid-40 ($500) LiDARs on each device to compare LeoVR with related works under different LiDAR settings. As shown in Fig. 10, Mid-40 is a solid-state LiDAR with Risley prism that adopts non-repetitive scanning (rosette-like scanning pattern), and the scanning trajectory never repeats. Mid-40 has a front facing, conical shaped, 38.4 ° FoV with a sampling rate of 100,000 points/s [31]. VLP-16 is a conventional mechanical

TABLE II
DETAILS OF DATA COLLECTION IN DIFFERENT SCENARIOS

#	Scene type	Sensors for data collection (Camera, LiDAR)	No. of frames	Trajectory length (km)	No. of trajectories for training	No. of trajectories for evaluation
1	City Road	Ladybug5+, Mid-40 & VLP-16	959,104	332.1	524	2524
2	Campus	Ladybug5+, Mid-40 & VLP-16	58,851	10.49	105	118
3	Classroom Building	Logitech C920E, Mid-40	56,024	2.91	102	95
4	Office Building	Logitech C920E, Mid-40	52,571	2.56	119	133

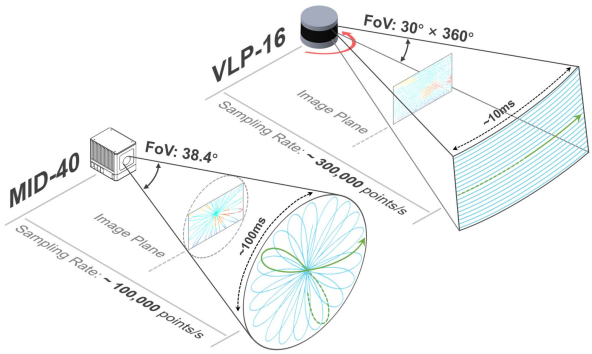


Fig. 10. The scanning principles and basic specifications of Mid-40 and VLP-16 LiDARs.

spinning LiDAR, which rotates 16 uniformly distributed lasers simultaneously for scanning with a rotation frequency of 5-20 Hz. VLP-16 has a 360 ° horizontal and 30 ° vertical FoV with a sampling rate of 300,000 points/s [45]. In brief, for the same detection area, VLP-16 can acquire denser point clouds with higher FoV coverage at a faster rate (the FoV coverage of Mid-40 is 20% in 0.1s [46]). For the optimization module of LeoVR , the server is equipped with Intel(R) Xeon(R) CPU E5-2620 v4 of 2.10 GHz main frequency and 256 G RAM, running the Ubuntu 18.0.4 operating system. For the model training and inference, the GPUs we use are two GeForce RTX 2080ti with CUDA version 10.1 and cuDNN v7.6.2.

Metrics and Ground Truth: For each generated depth map, we use the Mean Absolute depth Error (MAE) of all pixels in the image to measure the depth estimation performance, which is a golden indicator adopted by related works [1], [2]. To obtain the ground truth, we use a high-precision LiDAR (an 80-lines RoboSense Ruby Lite with $15,800) to run a LiDAR-based environmental mapping algorithm (LOAM [47]) to generate dense depth maps.

Dataset: We generated a dataset using the collected RGB images and LiDAR samples for further analysis, as summarized in Table II. The dataset includes 3,720 trajectories over a period of 8 months, comprising 1,126,550 frames. Roughly 30% of the trajectories contain loops, enabling us to evaluate the effectiveness of our pose quality assurance strategy. We conduct experiments in four representative scenarios: city road and campus as outdoor scenarios, while classroom building and office building as indoor scenarios. These scenes enjoy diverse spatial geometric layouts and environmental dynamics and thus provide different challenges for environment depth estimation. In the classroom and office buildings, there is only a small amount of pedestrian dynamic interference, the scene geometry is regular, and the distance between the target object and the sensing device is close. However, in the city road and campus, there are many moving vehicles and a variety of target objects (e.g., buildings, pedestrians, vegetation, road signs), and the perceived distance is relatively long compared to the indoor environment. The dataset of city road contains an extra four times the frames for model training and further analysis of the impact of training data number on the self-supervised training (Section V-E2). In our experiments, different models are trained for different scenarios. The training and evaluation data are taken from different regions in the same scenario, rather than uniformly extracted from the dataset, which helps to verify the generalizability of the model.

Comparative Methods: To extensively evaluate the performance of LeoVR , we additionally implement 4 state-of-the-art approaches based on visual-LiDAR fusion for comparison. We evaluate the depth estimation performance of LeoVR with: *DeepLiDAR* and *DenseLiDAR*, two SOTA learning-based depth estimation model. Specifically, when comparing LeoVR with these two works, we train the depth map generators of them and our LeoVR on the annotated dataset in advance. Furthermore, to evaluate the effectiveness of the proposed self-supervised framework, we compare LeoVR with another two existing self-supervised training based systems, *Self-S2D* and *Self-VLO*. In this part of experiments, without complex pre-training, the depth map generators in these systems and LeoVR cold start by self-supervised training.

Model Architectures & Training Details: The architecture of the depth map generator is depicted in Fig. 4. Input images are cropped to a resolution of 448x256 pixels. The encoder is constructed using ResNet-34 [48], with each layer having a stride of 2 for downsampling. Following each encoding layer, the number of channels in the feature map is [16, 32, 64, 128, 128] for both the RGB image branch and the LiDAR points branch. Batch normalization [49] and ReLU activation are applied to all layers except for the last one. The Adam optimizer [50] is employed with $\beta_1 = 0.9$ and $\beta_2 = 0.999$, and a batch size of 8 is used to train the model for 24 epochs. The initial learning rate is set to 10^{-4} and is halved every 6 epochs. Empirical hyper-parameters are set as follows: $\lambda_f = 0.4$, $\lambda_r = 0.1$, and $\lambda_s = 0.2$.

Our self-supervised training process involves two distinct stages, each with specific training protocols that rely on the motion information and geometric structure constraints provided by the factor graph. The first stage, called the **initialization stage** (first 18 epochs), relies solely on the point factor in the factor graph to gather motion information. During this stage, we do not use refined depth guidance (as explained in Section IV-D).

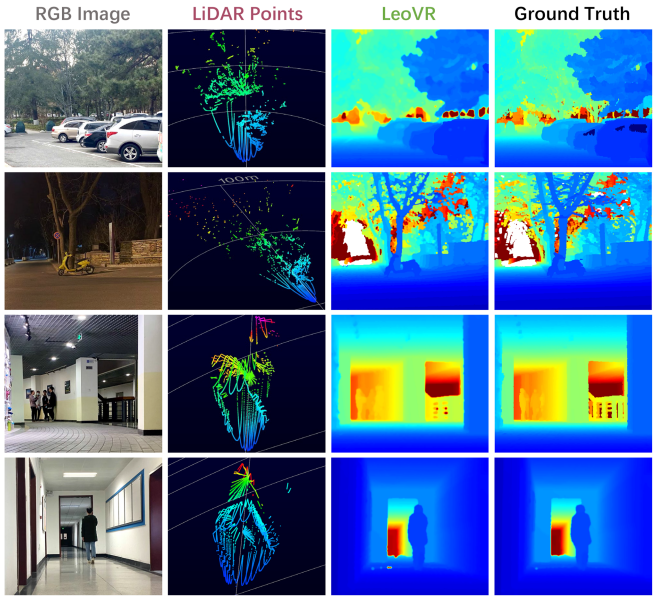


Fig. 11. Generated depth maps from LeoVR . From top to bottom are examples from city road, campus, classroom, and office building scenarios, respectively.

In contrast, the second stage, called the **refinement stage** (last 6 epochs), employs the full factor nodes and complete training constraints to progressively enhance the depth map refinement ability. To achieve this, we update the confidence mask and supervision signals provided by the factor graph every two epochs. Additionally, we implement the pose quality assurance strategy for each loop trajectory (Section IV-B) throughout the entire training process to ensure the reliability of self-supervised training.

B. Overall Performance

1) Depth Estimation Performance: We first evaluate the depth estimation performance of LeoVR as well as the above two comparative systems in different scenarios with a Livox Mid-40 LiDAR. As depicted in Fig. 12(a), LeoVR achieves the optimal performance in all scenarios with different environmental complexity. Specifically, LeoVR achieves estimation accuracies of 0.173 m, 0.134 m, 0.052 m, and 0.047 m in the city road, campus, classroom building, and office building scenarios, respectively, outperforming DenseLiDAR and DeepLiDAR by more than 45.9% and 57.8%. Qualitative results of LeoVR in different scenarios are shown in Fig. 11. It can be observed that the generated depth maps are capable of profiling the details of objects and are comparable to the ground truth. Compared to the raw LiDAR points, the depth maps provide a more complete and intuitive representation of the environment.

We further evaluate the performance of LeoVR and comparative systems on the city road dataset with two different types of LiDAR. As shown in Fig. 12(b), the MAE of LeoVR with Velodyne VLP-16 and Livox Mid-40 is 0.141 m and 0.173 m respectively, representing a $>$ 25.7% and 45.9% reduction compared to existing works. The results demonstrate that LeoVR consistently brings significant performance gains when using different types of LiDAR, especially for the commercial yet sparsely sampled Mid-40. In the case of sparser 3D LiDAR point cloud, LeoVR leverages the vehicle's motion information to provide additional spatio-temporal constraints among those continuously generated depth maps, which would further optimize the depth prediction accuracy. We will further demonstrate the effectiveness of the proposed motion-aware optimization framework in Section V-D1.

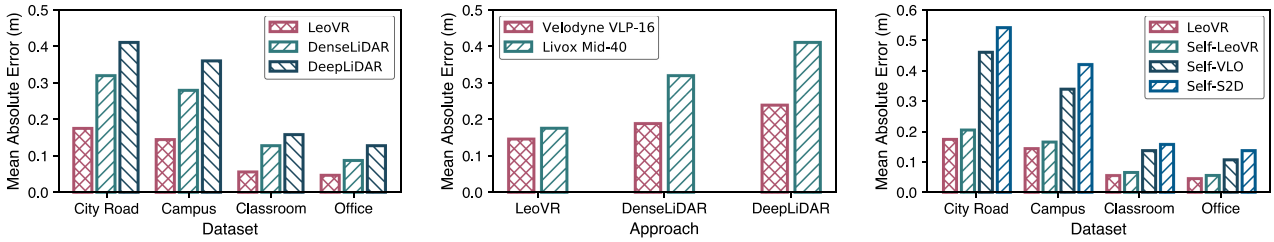
TABLE III
REAL-TIME TRACKING PERFORMANCE COMPARISON

System	Absolute Trajectory Error (cm)			
	City Road	Campus	Classroom	Office
V-LOAM	5.1	3.9	3.4	3.2
LeoVR	**4.1**	**3.3**	**3.1**	**2.8**

2) Self-Supervised Performance: We also conduct experiments to evaluate the performance of the proposed self-supervised framework. In this evaluation, LeoVR and two comparative self-supervised systems, Self-VLO and Self-S2D, are equipped with the Mid-40 LiDAR and cold start in different scenarios without pre-training. For each dataset, as described in Table II, we leverage partial trajectories for self-supervised training and use the remaining trajectories to evaluate the MAE of each system. The results are shown in Fig. 12(c). As seen, the depth estimation performance of the self-supervised LeoVR is 0.201 m, 0.158 m, 0.063 m, 0.055 m in four different scenarios respectively, which decrease by 13.9%, 15.1%, 17.4%, and 14.5% compared to that of the supervised version. Meanwhile, the self-supervised LeoVR outperforms comparative approaches by $>$ 47.8%. Especially in the city road scenario where the environment suffers from the most dynamics, LeoVR outperforms existing works by 56.5%. The impressive results demonstrate that the motion-optical flow-instructed self-supervised framework could boost the model training performance, especially in complex real-world environments. A detailed understanding of the contribution from each constraint (i.e., supervision signals) will be presented in Section V-D3.

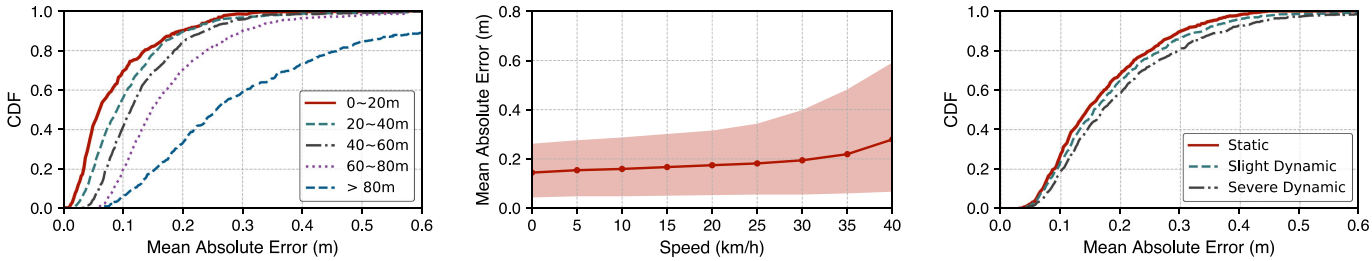
3) Motion Tracking Performance: The effective utilization of vehicle motion information is a key factor in LeoVR achieving outstanding depth estimation and self-supervised training performance. Therefore, the ability to accurately track vehicle motion (i.e., 6-DoF pose) is the basis of the entire system. In this experiment, we evaluate the system's tracking performance during both the online and the offline phases. While the online phase involves real-time data acquisition for pose and depth estimations, the offline phase utilizes pre-acquired trajectory data to estimate pose, further enhancing the model training.

We conduct this experiment with Livox Mid-40, and the ground truth is obtained through LOAM leveraging the 80-line RoboSense Ruby Lite LiDAR. We use the *absolute trajectory error* (ATE [51], in cm) to evaluate the motion tracking accuracy. As illustrated in Table III, during the online phase, LeoVR is benchmarked against V-LOAM [52] across various scenarios. LeoVR consistently achieves an ATE of less than 4.1 cm in all scenarios, surpassing V-LOAM by 19.6%. This



(a) Different Environments (b) Different LiDARs (c) Different Self-supervised Methods

Fig. 12. Overall depth estimation performance comparison.



(a) Impact of Scene Distance (b) Impact of Moving Speed (c) Impact of Scene Dynamics

Fig. 13. System robustness evaluation.

TABLE IV
COMPARISON OF POSE QUALITY FOR MODEL TRAINING

Approach	Absolute Trajectory Error (cm)			
	City Road	Campus	Classroom	Office
w/o loop-closure	4.6	3.5	3.3	3.0
w/ loop-closure	**1.7**	**1.1**	**0.9**	**0.9**

boost is attributed to the factor graph optimization, particularly with the introduction of *depth factor*. During the offline phase, we evaluate the improvements in pose estimation attributed to loop-closure, as detailed in Table IV. For training trajectories with loop closures, our pose assurance strategy adeptly leverages global information, resulting in a notable enhancement in pose accuracy. Compared with the approach without loop-closure, the error drops by at least 60%, ensuring reliable self-supervised training signals. While this strategy necessitates collecting trajectories with loop closures, our empirical observations indicate no significant additional overhead in data collection.

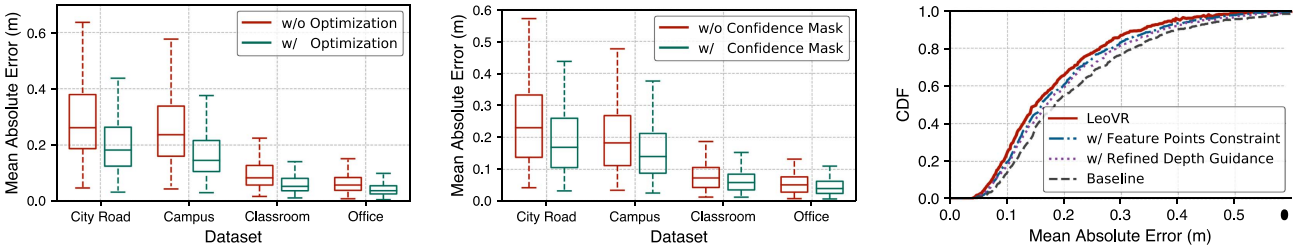
C. System Robustness Evaluation

In this part, we analyze the system's robustness in the most challenging city road using commercial autonomous vehicles equipped with Livox Mid-40.

1) Impact of Scene Distance: We examine the impact of scene distances on the depth estimation task. We divide the frame into five parts according to the distance from objects to the camera and separately calculate the MAE for each part. As shown in Fig. 13(a), the accuracy of estimation gradually decreases as the scene distance increases. Specifically, the MAEs are 0.09 m, 0.11 m, 0.14 m, and 0.18 m for the distances of 0-20 m, 20-40 m, 40-60 m, and 60-80 m, respectively. When the scene distance exceeds 80 m, the average error increases to 0.31 m, which can be attributed to the decrease in accuracy and density of the point cloud with increasing sensing distance. Nonetheless, our proposed method is capable of achieving a $95th$ percentile error below 0.36 m within the 80 m range.

2) Impact of Vehicle Speed: We further verify the robustness of LeoVR on different speeds of the vehicle. As depicted in Fig. 13(b), for speeds less than 20 km/h, LeoVR 's performance remains stable, with an average depth error of 0.18 m. When the moving speed = 0 km/h, the final optimization result can be considered as an average of the continuous depth maps within the sliding window, which are supposed to be consistent since the camera is stationary. As a reminder, the sampling area of Mid-40 varies from cycle to cycle, even when stationary, and the depth maps generated by the model are not identical. At speeds above 40 km/h, LeoVR achieves an average depth error of 0.27 m and a $95th$ percentile error of less than 0.59 m. The main reasons for the degradation are as follows: (i) The sampling circle of LiDAR is long and the vehicle is fast, resulting in an inherent bias when accumulating sampling points; and (ii) the view of adjacent frames at high speed differs greatly, posing a challenge for matching visual feature points.

3) Impact of Environmental Dynamics: As discussed in Section III-C3, dynamic objects within the scene can potentially degrade the depth map optimization. To understand the system's robustness against such challenges, we conduct evaluations across varying levels of scene dynamics, including static (scenes without moving entities), slight dynamic (scenes with occasional pedestrians and vehicles), and severe dynamic (scenes



(a) Motion-aware Optimization Framework (b) Motion-Optical Flow Constraint (c) Additional Constraints

Fig. 14. Ablation study.

TABLE V
IMPACT OF LIGHTING CONDITIONS

Approach	Average Depth Estimation Accuracy (cm)		
	Normal	Refractive	Dark
w/o cross-attention	17.5	21.9	23.1
w/ cross-attention	**16.8**	**18.3**	**18.6**

characterized by fast moving objects). As shown in Fig. 13(c), the average depth estimation errors are 0.162 m, 0.176 m, and 0.204 m in static, slight dynamic, and severe dynamic scenes, respectively. This represents a 7.9% improvement in error from static to slightly dynamic scenes and a 20.5% improvement in error from static to severely dynamic scenes. Further, when examining the $95th$ percentile of errors, the values are 0.37 m, 0.39 m, and 0.45 m for the static, slight dynamic, and severe dynamic scenes, respectively. These results highlight the system's ability to consistently limit substantial depth estimation errors, regardless of scene dynamics, attesting to the robustness of the employed optimization strategies.

4) Impact of Lighting Conditions: Both visual camera and LiDAR are sensitive to lighting conditions, making it a pivotal factor in the robustness of LeoVR . We evaluate LeoVR 's performance under three distinct lighting scenarios: normal illumination, refraction (induced by materials like glass and metal), and dark (nighttime scenes with sparse lighting). As shown in Table V, without the cross-attention module, the depth estimation error in refractive scenes increases by 25.1% due to LiDAR's inaccurate ranging. Similarly, in dark scenes, visual insensitivity leads to a 31.8% rise in error compared to normal conditions. However, when integrating the cross-attention module, the errors in refraction and dark scenes decrease by 16.4% and 19.5%, respectively, compared to the errors observed without the cross-attention module. Compared to traditional convolutional structures, cross-attention excels in fusing the strengths of both modalities and leveraging non-local correlations to overcome challenging scenarios, thereby enhancing the robustness of the system.

D. Ablation Study

We then conduct an ablation study to understand the effectiveness of some modules in LeoVR .

1) Effectiveness of the Motion-Aware Optimization: In this experiment, we compare the depth estimation performance without (w/o) and with (w/) the proposed *motion-aware learning-embedded optimization framework* to show the performance gains it brings into the overall system. When we close the optimization framework, the depth map output by the system is entirely generated by the self-supervised DNN. The experimental results are shown in Fig. 14(a). As seen, with our optimization framework, the system achieves an enhanced performance, where both MAE and variance of the depth estimation significantly decrease. Specifically, the MAE of LeoVR with the optimization framework is 0.201 m, 0.158 m, 0.063 m, and 0.055 m in city road, campus, classroom building, and office building, respectively, which represents a reduction of over 31.2% compared to LeoVR without optimization. These findings demonstrate the effectiveness of the motion-aware optimization framework in improving the accuracy of depth estimation in various settings.

2) Effectiveness of the Motion-Optical Flow Constraint: To demonstrate the effectiveness of motion-optical flow consistency constraint in the proposed self-supervised learning framework, we evaluate the model training performance on four different datasets with and without the confidence masks generated upon the motion-optical flow consistency through (19). The results are shown in Fig. 14(b). The proposed training method with confidence masks improves the depth estimation accuracy by 17.6% and 13.2% for the classroom building and office building scenarios, respectively. For the city road and campus scenarios, the performance is increased by 22.7% and 21.1%, respectively. The two outdoor scenarios are more complicated and dynamic, where the *rigid-static world assumption* would be violated frequently. The motion-optical flow constraint could be leveraged to select stable and consistent pixels among adjacent frames to generate supervision signals, resulting in a training performance lift, especially in dynamic environments.

3) Effectiveness of Additional Constraints: We evaluate the effectiveness of our additionally proposed *feature points constraint* and *refined depth guidance* for self-supervised training. In this experiment, we take the self-supervised model trained by the basic *photometric loss* and *motion-optical flow constraint* as a baseline and introduce these two additional constraints into the framework individually. We focus on the performance gains each of them brings to LeoVR . As shown in Fig. 14(c), the

TABLE VI
DEPTH MAP GENERATOR PERFORMANCE IMPROVEMENT

Approach	City Road		Classroom	
	Mean	95th	Mean	95th
Model w/o both	31.5	97.2	8.8	18.4
Model w/ cross-attetnion	30.6	76.5	8.7	14.6
Model w/ loop-closure	29.8	86.1	8.5	17.2
Model w/ both	**29.1**	**71.8**	**8.4**	**13.5**
LeoVR w/o both	18.4	46.8	5.5	9.9
LeoVR w/ both	**17.3**	**38.6**	**5.2**	**7.8**

The bold indicates the best case achieved in the Model and LeoVR systems, respectively.

MAE decreases by 7.8% when the *feature points constraint* is added on top of the baseline. Similarly, the introduction of *refined depth guidance* leads to a 6.5% improvement in depth estimation performance. These results demonstrate that the additional constraints can provide useful supervision signals for training the DNN, resulting in improved performance.

4) Depth Map Generator Performance Improvement: In this work, we build upon the [28] model and propose two strategies to enhance the robustness of the depth map generator: (i) a cross-attention mechanism to improve the complementary ability between the visual and LiDAR modalities, and (ii) a loop-closure based pose quality assurance strategy utilizing global motion information to ensure the reliability of self-supervised training. In this experiment, we first assess the individual and combined enhancements these techniques bring to the model. Subsequently, we analyze their systemic impact, especially in terms of gains post factor-graph optimization.

As detailed in Table VI, the cross-attention and loop-closure techniques independently reduce the average error by 2.8% and 5.4% in the city road scenario, respectively. When combined, they yield a cumulative improvement of 7.6%. Furthermore, in contrast to the baseline's 95th percentile error of 97.2 cm, the cross-attention and loop-closure strategies enhance performance by 21.3% and 11.4%, respectively. Their joint application results in a 26.1% uplift, underscoring their efficacy in constraining larger errors. From a holistic system perspective, their incorporation elevates LeoVR 's average accuracy by 6.2%, while diminishing the 95th percentile error by 17.5%. Considering that the model's depth map output is subject to joint optimization, the enhancements of LeoVR from these techniques are slightly diminished. Nonetheless, their effectiveness in enhancing model robustness remains evident.

It is noteworthy that the integration of these strategies for enhanced robustness doesn't compromise the end-to-end latency. On one hand, the loop-closure strategy is exclusively employed during the offline phase for model training. On the other hand, by producing high-precision depth maps, LeoVR necessitates fewer iterations in subsequent optimizations, effectively shortening the overall latency (Section V-F).

E. Parameter Study

We conduct a parameter study to understand the impact of the selection of some critical parameters on the system performance.

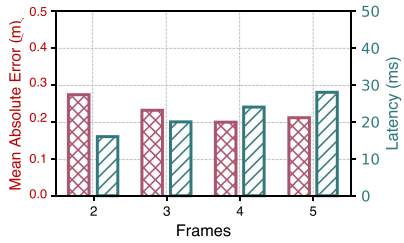


Fig. 15. Impact of sliding window.

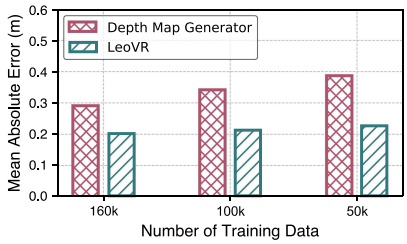


Fig. 16. Impact of training data.

1) Impact of Sliding Window Length: As aforementioned, LeoVR uses a set of frames within a sliding window to optimize the depth map using the factor-graph. We evaluate the impact of the sliding window length on the accuracy and latency, and the results. As depicted in Fig. 15, when using only two adjacent frames, the depth estimation error is 0.275 m with a 18 ms optimization delay. With the window length increasing to 4 frames, the error decreases by 27.3%, and the optimization latency increases to 24 ms. However, continuously increasing the window length beyond 4 frames will not improve the accuracy of LeoVR because there will be a large gap in the FoV among these frames. To balance the accuracy and latency, we use a sliding window length of 4 frames in LeoVR .

2) Impact of Training Data Amount: The self-supervised training framework in LeoVR allows the DNN model (i.e., the depth map generator) to be trained by unlabeled data, however, the model training effectiveness is inevitably sensitive to the amount of the training data. We further evaluate the impact of training data amount on the performance of LeoVR with and without (i.e., merely using the DNN model) the optimization framework in the city road dataset. As shown in Fig. 16, the average estimation error of LeoVR with and without optimization are 0.291 m and 0.201 m trained with 160 k frames using self-supervised learning. When the training data number decreased to 50 k, the error scale up to 0.387 m and 0.226 m, respectively. The performance degradation of LeoVR without optimization is 24.8%, while the entire LeoVR is only 11.1%. The above results demonstrate that although the self-supervised training performance degrades with limited training data, the subsequent optimization scheme could still maintain the estimation accuracy.

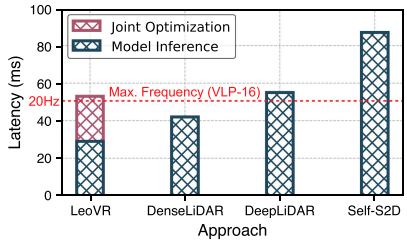


Fig. 17. System efficiency.

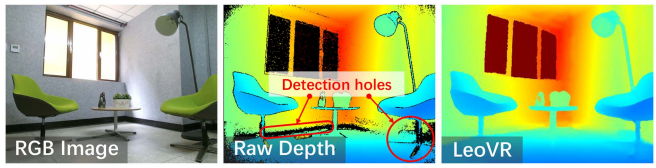


Fig. 18. Depth map generated by LeoVR with RGB-D camera. From left to right, RGB image and raw depth captured by Kinect, and the complete depth map generated by LeoVR . In depth maps, black represents detection holes, warm colors represent distant objects, and cool colors represent close objects.

F. Efficiency Study

As a depth estimation system towards auto-driving scenarios, we further evaluate the efficiency of LeoVR . As shown in Fig. 17, we measure the end-to-end latency of four depth estimation systems. Note that unlike existing DNN standalone approaches, LeoVR involves both a DNN and a joint optimization scheme. Therefore, we separately report the latency of these two modules in LeoVR . The end-to-end latency of LeoVR , DenseLiDAR, DeepLiDAR, and Self-S2D are 53 ms, 42 ms, 55 ms, and 87 ms, respectively. The latency of the DNN and the optimization scheme in LeoVR are 29 ms and 24 ms, respectively. Overall, LeoVR can operate with a frequency of 18 Hz, which is in line with the LiDAR sampling frequency (e.g., VLP-16, 5-20 Hz).

Compared to previous version [28], LeoVR achieves improvements in both model inference and optimization speeds. First, the integration of cross-attention, coupled with a streamlined model architecture, as detailed in Section V-A, accelerates the model inference. Second, the strategies aimed at fortifying model robustness result in more accurate initial depth maps, leading to faster convergence in subsequent optimization with reduced iterations.

G. Case Study: Depth Completion With RGB-D Camera

RGB-D camera is commonly used for capturing depth information in indoor environments. However, one of the major challenges with this sensor is the presence of detection holes in depth maps, arising from reflections, refractions, and black absorbing surfaces [53]. Filling these holes is a complex task for models, and obtaining ground truth data for training can be a challenging endeavor.

This case study investigates the application of our motion-inspired multi-modal fusion method for depth completion in indoor environments using RGB-D camera. While LeoVR is primarily designed for autonomous driving scenarios, we demonstrate its seamless adaptation to indoor environments. Specifically, LeoVR take RGB image and raw depth data, both cropped to a resolution of 360*480 pixels, as the inputs, while incorporating motion information from the RGB-D camera to enhance self-supervised training and depth optimization. As a result, LeoVR generates accurate and complete indoor depth maps that capture the structural details of the environment.

Experimental Setup: For data acquisition in our experiments, we used a Microsoft Kinect to capture RGB images and depth maps simultaneously. We conducted the data collection process in diverse indoor environments, including classrooms and offices, and gathered a continuous stream of 209,180 frames to train our models. Additionally, a separate dataset consisting of 1,800 frames was obtained from a student activity center, with frames captured continuously at a rate of 30 Hz over a period of one minute, specifically for qualitative analysis. It should be noted that quantitative evaluation and comparison with other methods were not conducted in this case study due to the lack of specialized equipment or established strategies for obtaining ground truth data. Additionally, our training procedure relied solely on self-supervised learning without the use of manually annotated datasets.

Performance Evaluation: As shown in Fig. 18, we present a qualitative result of LeoVR in the student activity center. The raw depth map collected by Kinect contains numerous detection holes, with 20.1% of the pixels affected by factors of refraction and color absorption. However, by utilizing the RGB image and mobility of the device, LeoVR is able to generate a complete and visually plausible depth map. The improved performance can be attributed to LeoVR can exploit geometric cues in the RGB image to complete the missing depth information, as well as the ability to optimize observations from multiple perspectives using motion information. In this case study, to achieve better multi-view optimization results, a sliding window of length 12 was used given the relatively slow movement speed of the hand-held Kinect device.

Limitations: Despite the promising results, we note that there are still limitations to LeoVR . In particular, we found that our model struggles to accurately estimate depth in challenging scenarios such as scenes with glass windows (e.g., the estimated distance is greater than the actual distance, as shown in the dark-red region of Fig. 18 - LeoVR). In these situations, the performance of both the RGB and depth sensors is compromised due to glass refraction and glare from sunlight. Furthermore, our motion-optical flow constraint strategy filters out these types of interference during model training, making it difficult to handle such out-of-distribution scenarios. Addressing these challenges will be a focus of our future work, by further leveraging semantic information from vision.

VI. Discussion and Future Work

LeoVR is an early step towards ubiquitous environmental depth estimation by fusing camera and low-cost LiDAR. We briefly discuss limitations and future works in this section.

Degraded performance in adverse weather: Both of conventional LiDAR and camera are sensitive to weather conditions and are not expected to work fully in adverse weather like fog, making the perception results unreliable [29]. Aside from lidar and camera, radar has been widely deployed on autonomous vehicles. Specifically, radar uses millimeter-wave signals whose wavelength is much larger than the tiny particles forming fog, rain, and snow, and hence easily penetrates or diffracts around them [54]. We think integrating mmWave radar into LeoVR would greatly enhance the robustness of system in harsh weather.

Additional latency for optimization: The learning-embedded optimization scheme of LeoVR pushes the limits of the depth estimation accuracy, however, the fine-grained optimization also introduces additional computational delay. Optimizing the computational overhead required by the algorithm [55], or offloading some computation-intensive yet delay-tolerant tasks to an edge server [56] is also a promising research direction.

Adaptation in high-speed scenarios: As the vehicle speed increases, the LiDAR point clouds generated by Mid-40 as well as visual features also suffer from obvious motion blur [31], [57] due to their hardware nature, resulting in LeoVR not capable of handling high-speed scenarios (e.g., vehicle speed $>$ 40 km/h). We think introducing IMU [58] or wheel odometry [59] into the optimization framework to provide a high frequency ego-motion estimation could help to tackle the above challenge, which is left as future works.

Visual Semantic Guidance: Incorporating the visual modality has undoubtedly assisted in the restoration of missing pixels in the depth map. However, the presence of shadows, reflections, and refractions in the visual image may cause irregularities in pixel values, which can compromise the model training and introduce biases in the depth optimization process. Incorporating a semantic understanding of the environment can be instrumental in overcoming these challenging scenarios. Therefore, utilizing semantic segmentation networks [60], [61], [62] to guide the generation of depth maps is a promising direction for future research.

VII. Related Work

Optimization-based Visual-Radar fusion: In recent years, the fusion of multi-modal sensors, especially leveraging vision or radar, has attracted a wide range of attention from both industry and academia [32], [52], [63], [64], [65], [66]. Among them, RF-Fusion [65] fuses vision and RFID to enable robots to recognize objects. ITrackU [67] leverages IMU and UWB radar for tracking a pen-like instrument. V-LOAM [52] integrates vision and LiDAR in a loosely coupled manner to track a camera's motion and generate environmental 3D point clouds. VILENS [58] and FollowUpAR [68] utilize a factor graph to tightly integrate vision and laser or mmWave radar features for real-time object point cloud registration. However, all of the above optimization-based approaches could merely generate object- or surface-level sparse point clouds instead of pixel-level depth maps of surrounding environments. Some recent works [32] rely on an expensive LiDAR to sample dense point clouds, limiting their deployment on commercial devices. In contrast, LeoVR could achieve dense depth map generation with low-cost LiDAR.

Learning-based visual-LiDAR fusion for depth estimation: In recent years, deep learning has enabled environmental perception for IoT devices [69], [70], [71], [72]. Some notable works [17], [18], [23] have designed learning-based frameworks to fuse visual and LiDAR data for environment depth estimation. To instruct the higher-level network to generate a more accurate depth map, recent works such as DeepLiDAR [2] and DenseLiDAR [1] have utilized pseudo-depth maps obtained from morphological operations. However, the performance of these solutions highly depends on the density of the LiDAR point clouds and suffers from severe degradation when equipped with commercial in-vehicle LiDARs. To improve prediction capability, previous works such as [73] and [53] have incorporated self-attention modules [19] into their networks to capture the correlation between the scene and object elements. In contrast, our proposed LeoVR model employs a cross-modal attention mechanism that further focuses on non-local dependent features between modalities, enhancing sensor fusion efficiency.

Self-supervised depth estimation: Most existing works on depth map estimation rely on densely annotated ground truth for model training, which burdens these systems for widespread deployment. Recently, some self-supervised solutions have been proposed [5], [20], [25], [74]. For instance, Self-S2D [20] takes a sequence of images with depth maps as inputs and uses Perspective-n-Point to align them for photometric consistency. [25] proposes an unsupervised learning framework for monocular depth and camera motion estimation from unstructured video sequences. However, these methods suffer from degraded performance in practical scenarios, with noisy pixels (dynamic objects, occlusion, and non-Lambertian surfaces) and unguaranteed pose quality being the main culprits. To address these limitations, LeoVR , leverages the motion information of the vehicle to extract robust supervision signals in complex environments. Additionally, we introduce the loop-closure strategy for guaranteeing the pose quality of self-supervised training data. Although loop-closure [75], [76] is a well-established global trajectory optimization technique in the SLAM field [35], [39], our work represents the first application of this approach to the quality assurance of self-supervised signals in depth map generation. In comparison to existing works, our approach demonstrates superior performance in challenging scenarios, highlighting the efficacy of incorporating motion information and pose quality assurance in self-supervised depth estimation.

VIII. Conclusion

We have presented the design and implementation of LeoVR, a self-supervised environment depth estimation system with visual-LiDAR fusion. LeoVR takes fully advantages of the vehicle's motion information and designs (i) a motion-aware learning-embedded optimization scheme for generating accurate environmental depth maps even with low-cost LiDARs; and (ii) a motion-optical flow instructed self-supervised framework that enables self-supervised training of the DNN. We implement

LeoVR on a robotic testbed and commercial vehicles, conducting extensive experiments in real environments across 8 months. The results demonstrate its superior performance over previous schemes in all scenarios with different types of LiDAR, promising adaptability for future LiDAR with different specifications. Being fully self-supervised and achieving an accurate depth estimation performance, LeoVR makes a great process towards fortifying environmental perception to an essential capability for large-scale on-vehicle deployment.

REFERENCES

[1] J. Gu, Z. Xiang, Y. Ye, and L. Wang, "DenseLiDAR: A real-time pseudo dense depth guided depth completion network," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 1808–1815, Apr. 2021.

[2] J. Qiu et al., "DeepLiDAR: Deep surface normal guided depth prediction for outdoor scene from sparse LiDAR data and single color image," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 3308–3317.

[3] Y. Chen, B. Yang, M. Liang, and R. Urtasun, "Learning joint 2d-3D representations for depth completion," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 10023–10032.

[4] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao, "Deep ordinal regression network for monocular depth estimation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2002–2011.

[5] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow, "Digging into self-supervised monocular depth estimation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 3827–3837.

[6] J. Czarnowski, T. Laidlow, R. Clark, and A. J. Davison, "DeepFactors: Real-time probabilistic dense monocular SLAM," *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 721–728, Apr. 2020.

[7] Y. Wang, W.-L. Chao, D. Garg, B. Hariharan, M. Campbell, and K. Q. Weinberger, "Pseudo-LiDAR from visual depth estimation: Bridging the gap in 3D object detection for autonomous driving," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 8437–8445.

[8] F. Xue, G. Zhuo, Z. Huang, W. Fu, Z. Wu, and M. H. Ang, "Toward hierarchical self-supervised monocular absolute depth estimation for autonomous driving applications," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 2330–2337.

[9] Y. He, L. Ma, Z. Jiang, Y. Tang, and G. Xing, "VI-eye: Semantic-based 3D point cloud registration for infrastructure-assisted autonomous driving," in *Proc. 27th Annu. Int. Conf. Mobile Comput. Netw.*, 2021, pp. 573–586.

[10] L. Teixeira, M. R. Oswald, M. Pollefeys, and M. Chli, "Aerial single-view depth completion with image-guided uncertainty estimation," *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 1055–1062, Apr. 2020.

[11] C. Häne, C. Zach, J. Lim, A. Ranganathan, and M. Pollefeys, "Stereo depth map fusion for robot navigation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2011, pp. 1618–1625.

[12] S. Izadi et al., "Kinectfusion: Real-time 3D reconstruction and interaction using a moving depth camera," in *Proc. 24th Annu. ACM Symp. User Interface Softw. Technol.*, 2011, pp. 559–568.

[13] "A reminder that tesla's autopilot still requires a pilot," 2020. [Online]. Available: https://www.motortrend.com/news/tesla-model-3-crash-taiwan-autopilot-accident/

[14] "Self-driving car systems were involved in 400 crashes since 2021," 2022. [Online]. Available: https://news.yahoo.com/self-driving-car-systems-were-142925251.html

[15] J. Wang, L. Zhang, Y. Huang, and J. Zhao, "Safety of autonomous vehicles," *J. Adv. Transp.*, pp. 1–13, 2020.

[16] Z. Yin and J. Shi, "GeoNet: Unsupervised learning of dense depth, optical flow and camera pose," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 1983–1992.

[17] M. Hu, S. Wang, B. Li, S. Ning, L. Fan, and X. Gong, "PENet: Towards precise and efficient image guided depth completion," *IEEE Int. Conf. Robot. Automat.*, 2021, pp. 13656–13662.

[18] S. Zhao, M. Gong, H. Fu, and D. Tao, "Adaptive context-aware multi-modal network for depth completion," *IEEE Trans. Image Process.*, vol. 30, pp. 5264–5276, 2021.

[19] J. Park, K. Joo, Z. Hu, C.-K. Liu, and I. S. Kweon, "Non-local spatial propagation network for depth completion," in *Proc. Comput. Vis. 16th Eur. Conf.*, 2020, pp. 120–136.

[20] F. Ma, G. V. Cavalheiro, and S. Karaman, "Self-supervised sparse-to-dense: Self-supervised depth completion from LiDAR and monocular camera," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2019, pp. 3288–3295.

[21] "Velodyne," 2021. [Online]. Available: https://velodynelidar.com/products/

[22] "Livox," 2021. [Online]. Available: https://www.livoxtech.com/application/autonomous-driving

[23] B. Li, M. Hu, S. Wang, L. Wang, and X. Gong, "Self-supervised visual-LiDAR odometry with flip consistency," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis.*, 2021, pp. 3844–3852.

[24] J. Choi et al., "SelfDeco: Self-supervised monocular depth completion in challenging indoor environments," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 467–474.

[25] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 6612–6619.

[26] M. Henein, J. Zhang, R. Mahony, and V. Ila, "Dynamic SLAM: The need for speed," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 2123–2129.

[27] J. Wulff, L. Sevilla-Lara, and M. J. Black, "Optical flow in mostly rigid scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 6911–6920.

[28] D. Li et al., "Motion inspires notion: Self-supervised visual-LiDAR fusion for environment depth estimation," in *Proc. 20th Annu. Int. Conf. Mobile Syst. Appl. Serv.*, 2022, pp. 114–127.

[29] C. X. Lu et al., "milliEgo: Single-chip mmWave radar aided egomotion estimation via deep sensor fusion," in *Proc. 18th Conf. Embedded Networked Sensor Syst.*, 2020, pp. 109–122.

[30] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Proc. Int. Conf. Comput. Vis.*, 2011, pp. 2564–2571.

[31] J. Lin and F. Zhang, "Loam livox: A fast, robust, high-precision LiDAR odometry and mapping package for LiDARS of small FoV," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 3126–3131.

[32] J. Graeter, A. Wilczynski, and M. Lauer, "LIMO: Lidar-monocular visual odometry," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 7872–7879.

[33] M. Jaderberg et al., "Spatial transformer networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015.

[34] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[35] T. Qin, P. Li, and S. Shen, "VINS-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 1004–1020, Aug. 2018.

[36] F. Dellaert, "Factor graphs and GTSAM: A hands-on introduction," Georgia Inst. Technol., Atlanta, GA, USA, Tech. Rep., 2012.

[37] D. Gálvez-López and J. D. Tardos, "Bags of binary words for fast place recognition in image sequences," *IEEE Trans. Robot.*, vol. 28, no. 5, pp. 1188–1197, Oct. 2012.

[38] A. J. Ben Ali, M. Kouroshli, S. Semenova, Z. S. Hashemifar, S. Y. Ko, and K. Dantu, "Edge-SLAM: Edge-assisted visual simultaneous localization and mapping," *ACM Trans. Embedded Comput. Syst.*, vol. 22, pp. 1–31, 2022.

[39] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, Oct. 2015.

[40] "Kitti-dataset," 2017. [Online]. Available: http://www.cvlibs.net/datasets/kitti/eval_depth.php?benchmark=depth_completion

[41] "Opencv: Optical flow," 2022. [Online]. Available: https://docs.opencv.org/3.4/d4/dee/tutorial_optical_flow.html

[42] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Proc. 13th Scand. Conf.*. Springer Berlin Heidelberg, 2003, pp. 363–370.

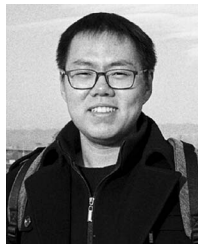
[43] Z. Zhang, "Determining the epipolar geometry and its uncertainty: A review," *Int. J. Comput. Vis.*, vol. 27, pp. 161–195, 1998.

[44] Q.-T. Luong and O. D. Faugeras, "The fundamental matrix: Theory, algorithms, and stability analysis," *Int. J. Comput. Vis.*, vol. 17, no. 1, pp. 43–75, 1996.

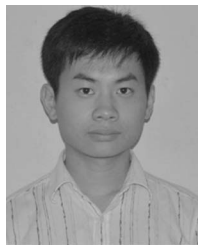
[45] "VLP-16," 2022. [Online]. Available: https://velodynelidar.com/products/puck/

[46] "Mid-40-specs," 2022. [Online]. Available: https://www.livoxtech.com/cn/mid-40-and-mid-100/specs
[47] J. Zhang and S. Singh, "LOAM: LiDAR odometry and mapping in real-time," in *Proc. Robot.: Sci. Syst.*, 2014.
[48] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[49] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.
[50] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[51] "Tum tools," 2021. [Online]. Available: https://vision.in.tum.de/data/datasets/rgbd-dataset/tools
[52] J. Zhang and S. Singh, "Visual-LiDAR odometry and mapping: Low-drift, robust, and fast," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2015, pp. 2174–2181.
[53] Y.-K. Huang, T.-H. Wu, Y.-C. Liu, and W. H. Hsu, "Indoor depth completion with boundary consistency and self-attention," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshops*, 2019, pp. 1070–1078.
[54] C. X. Lu et al., "See through smoke: Robust indoor mapping with low-cost mmWave radar," in *Proc. 18th Int. Conf. Mobile Syst. Appl. Serv.*, 2020, pp. 14–27.
[55] M. Bloesch, J. Czarnowski, R. Clark, S. Leutenegger, and A. J. Davison, "CodeSLAM—learning a compact, optimisable representation for dense visual SLAM," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2560–2568.
[56] J. Xu et al., "Edge assisted mobile semantic visual SLAM," in *Proc IEEE Conf. Comput. Commun.*, 2020, pp. 1828–1837.
[57] A. R. Vidal, H. Rebecq, T. Horstschaefer, and D. Scaramuzza, "Ultimate SLAM? combining events, images, and IMU for robust visual SLAM in HDR and high-speed scenarios," *IEEE Robot. Automat. Lett.*, vol. 3, no. 2, pp. 994–1001, Apr. 2018.
[58] D. Wisth, M. Camurri, S. Das, and M. Fallon, "Unified multi-modal landmark tracking for tightly coupled LiDAR-visual-inertial odometry," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 1004–1011, Apr. 2021.
[59] M. Brossard and S. Bonnabel, "Learning wheel odometry and IMU errors for localization," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2019, pp. 291–297.
[60] A. Kirillov et al., "Segment anything," 2023, *arXiv:2304.02643*.
[61] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2961–2969.
[62] P.-Y. Chen, A. H. Liu, Y.-C. Liu, and Y.-C. F. Wang, "Towards scene understanding: Unsupervised monocular depth estimation with semantic-aware representation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 2624–2632.
[63] C. Park, P. Moghadam, S. Kim, A. Elfes, C. Fookes, and S. Sridharan, "Elastic LiDAR fusion: Dense map-centric continuous-time SLAM," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 1206–1213.
[64] X. Zuo et al., "LIC-fusion 2.0: LiDAR-inertial-camera odometry with sliding-window plane-feature tracking," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 5112–5119.
[65] T. Boroushaki, I. Perper, M. Nachin, A. Rodriguez, and F. Adib, "RFusion: Robotic grasping via RF-visual sensing and learning," in *Proc. 19th ACM Conf. Embedded Networked Sensor Syst.*, 2021, pp. 192–205.
[66] S. Sami, Y. Dai, S. R. X. Tan, N. Roy, and J. Han, "Spying with your robot vacuum cleaner: Eavesdropping via LiDAR sensors," in *Proc. 18th Conf. Embedded Networked Sensor Syst.*, 2020, pp. 354–367.
[67] Y. Cao, A. Dhekne, and M. Ammar, "ITrackU: Tracking a pen-like instrument via UWB-IMU fusion," in *Proc. 19th Annu. Int. Conf. Mobile Syst. Appl. Serv.*, 2021, pp. 453–466.
[68] J. Xu et al., "FollowUpAR: Enabling follow-up effects in mobile AR applications," in *Proc. ACM 19th Annu. Int. Conf. Mobile Syst. Appl. Serv.*, 2021, pp. 1–13.
[69] C. Yoo, S. Sarmin, I. Hwang, E. Rozner, and M. Cho, "Poster: Deepfind: Sensor-driven inference acceleration for continuous deep mobile vision applications," in *Proc. 21st Int. Workshop Mobile Comput. Syst. Appl.*, 2020.
[70] S. Yao et al., "Eugene: Towards deep intelligence as a service," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst.*, 2019, pp. 1630–1640.
[71] H. Qiu et al., "Towards robust vehicular context sensing," *IEEE Trans. Veh. Technol.*, vol. 67, no. 3, pp. 1909–1922, Mar. 2018.
[72] S. Boovaraghavan, A. Maravi, P. Mallela, and Y. Agarwal, "MLIoT: An end-to-end machine learning system for the internet-of-things," in *Proc. Int. Conf. Internet-of-Things Des. Implementation*, 2021, pp. 169–181.
[73] S. Srivastava and G. Sharma, "Self attention guided depth completion using RGB and sparse LiDAR point clouds," in 2021*IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 2643–2650.
[74] S. Yan, Y. Zheng, W. Ao, X. Zeng, and M. Zhang, "Does unsupervised architecture representation learning help neural architecture search?," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020.
[75] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2D LiDAR SLAM," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 1271–1278.
[76] H. Osman, N. Darwish, and A. Bayoumi, "LoopNet: Where to focus? detecting loop closures in dynamic scenes," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 2031–2038, Apr. 2022.

Danyang Li (Graduate Student Member, IEEE) received the BE degree from the School of Software, Yanshan University in 2019, and the ME degree from the School of Software, Tsinghua University in 2022. He is currently working toward the PhD degree with the School of Software, Tsinghua University. His research interests include Internet of Things and mobile computing.

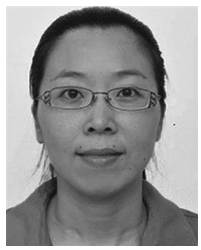


Jingao Xu (Member, IEEE) received the BE and PhD degrees from the School of Software, Tsinghua University in 2017 and 2022, respectively. He is currently a postdoc research fellow with the School of Software, Tsinghua University. His research interests include Internet of Things and mobile computing.



Zheng Yang (Fellow, IEEE) received the BE degree in computer science from Tsinghua University in 2006, and the PhD degree in computer science from the Hong Kong University of Science and Technology in 2010. He is currently an associate professor with Tsinghua University. His research interests include Internet of Things and mobile computing. He is the PI of the National Natural Science Fund for Excellent Young Scientist. He was the recipient of the State Natural Science Award (second class).

Qiang Ma (Member, IEEE) received the BS degree from the Department of Computer Science and Technology, Tsinghua University, China, in 2009, and the PhD degree from the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, in 2013. He is currently an assistant researcher with the Software School, Tsinghua University. His research interests include wireless sensor networks, mobile computing, and privacy.



Li Zhang received the BS degree in applied mathematics from Anhui Normal University in 1999, and the MS and PhD degrees in computational mathematics and computer science and technology from the Hefei University of Technology, in 2004 and 2009, respectively. Her research interests include computer-aided geometric design, computer graphics, and image processing.

Pengpeng Chen received the PhD degree from the Department of Computer Science and Technology, Ocean University of China in 2011. He is currently a professor with the Department of Computer Science and Technology, China University of Mining and Technology. His research interests include sensor networks, distributed measurement systems, ocean observation networks, and data modeling.