Motion Inspires Notion: Self-supervised Visual-LiDAR Fusion for Environment Depth Estimation

Danyang Li[1], Jingao Xu[1], Zheng Yang[1]✉, Qian Zhang[1], Qiang Ma[1], Li Zhang[2], Pengpeng Chen[3]
[1]School of Software and BNRist, Tsinghua University
[2]HeFei University of Technology [3]China University of Mining and Technology
{lidanyang1919,xujingao13,hmilyyz,qzhangqz123,tsinghuamq,hgdzli,believesea}@gmail.com

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 visual-LiDAR fusion based self-supervised approach that enables accurate environment depth estimation. LeoVR digs into the vehicle's motion information and designs two effective system frameworks based on it to (i) optimize the depth estimation results, and (ii) provide supervision signals to train a DNN. We fully implement LeoVR on a robotic testbed and commercial vehicle to conduct extensive experiments across 6 months. The results demonstrate that LeoVR achieves remarkable performance with an average depth estimation error of $0.17m$, outperforming existing state-of-the-art solutions by $> 43\%$. Besides, even cold-start in real environments by self-supervised training, LeoVR still achieves an average error of $0.21m$, outperforming the related works by $> 45\%$ and comparable to those supervised training methods.

CCS CONCEPTS

• **Human-centered computing → Ubiquitous and mobile computing**.

KEYWORDS

Visual-LiDAR Fusion; Depth Estimation; Self-supervised Learning

ACM Reference Format:
Danyang Li, Jingao Xu, Zheng Yang, Qian Zhang, Qiang Ma, Li Zhang, Pengpeng Chen. 2022. Motion Inspires Notion: Self-supervised Visual-LiDAR Fusion for Environment Depth Estimation. In *The 20th Annual International Conference on Mobile Systems, Applications and Services (MobiSys '22), June 25–July 1, 2022, Portland, OR, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3498361.3538918

✉ Zheng Yang is the corresponding author.

MobiSys '22, June 25–July 1, 2022, Portland, OR, USA

ACM ISBN 978-1-4503-9185-6/22/06…$15.00
https://doi.org/10.1145/3498361.3538918

Table 1: Overall System Comparison

System	Average Depth Estimation Accuracy		Self-supervised
	VLP-16 ($4,000)	Mid-40 ($500)	
DeepLiDAR[39]	$0.24m$	$0.41m$	✗
DenseLiDAR[16]	$0.19m$	$0.32m$	✗
Self-S2D[33]	$0.33m$	$0.54m$	✓
LeoVR	$\mathbf{0.15}m$	$\mathbf{0.18}m$	✓

1 INTRODUCTION

Environment depth estimation aims at obtaining geometric properties of surrounding 3D space from 2D images and associated sensor inputs [7, 16, 39]. Typically, it generates a depth map[1] for each input image [9, 13, 14]. 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 [19, 48, 54] and robotics [42]. For instance, with accurate depth maps, vehicles and drones can realize context-aware localization, navigation, intelligent interaction, and obstacle avoidance [16, 17, 23]; and according to recent reports, wrongly estimating the depth of ambient humans or objects causes a majority of autonomous driving accidents in 2020 [10, 47].

Current environment depth estimation practice on autonomous vehicles typically resorts to fusing camera and Laser radar (LiDAR). Compared to visual stand-alone approaches [9, 58], LiDAR could provide the accurate 3D location of essential reflection points in the current 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 [21, 36, 63].

Albeit inspiring, according to our 6 months field study, we find previous solutions face grand challenges deploying in real-world environments. The crucial drawbacks are twofold:

• **Degraded performance with commercial in-vehicle LiDAR**. Although current practice achieves remarkable depth mapping performance, they primarily rely on sophisticated yet expensive Velodyne 16-line (VLP-16) or 32-line (HDL-32E) LiDAR [44], which

[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.

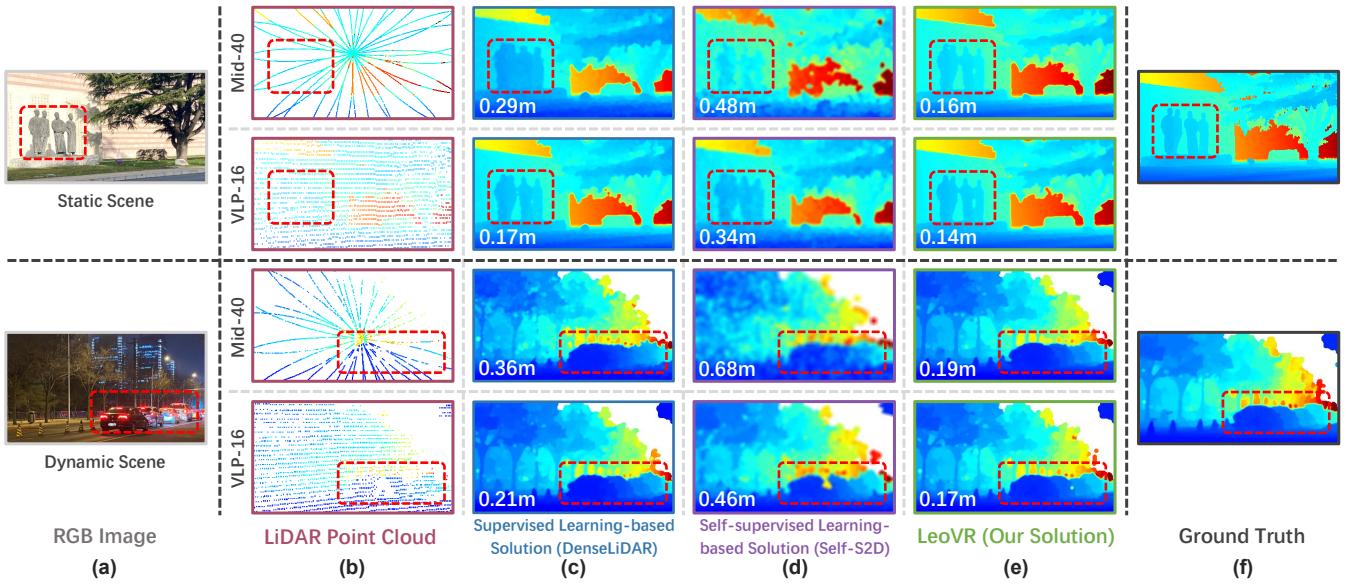


Figure 1: A glimpse of depth estimation performance of LeoVR and existing solutions. (a) Visual images of the static and dynamic scene. (b) Associated LiDAR point clouds sampled by Mid-40 (cheap yet sparse) and VLP-16 (dense yet expensive). (c)-(e) are the depth map generated from state-of-the-art methods: DenseLiDAR [16] (a supervised learning-based solution), Self-S2D [33] (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.

costs around $4k and $20k 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 [29]). As illustrated in Fig. 1b, Mid-40 generates merely one-third as many 3D points as VLP-16 LiDAR within a 0.1s 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. 1c-d, the estimation accuracy was reduced by almost 50% when equipped with the sparsely sampled Mid-40. We can also find that merely an accuracy drop of around $0.15m$ will make it difficult to segment important objects in the current scene. For instance, in Fig. 1c, 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 [16]. Although some recent works have proposed self-supervised training frameworks to deal with this issue [8, 27, 33], they extract training signals between consecutive frames and highly depend on the *static-rigid world assumption* [20, 51, 64], which is unrealistic in real complicated road conditions with reflections, shadows, and highly dynamic humans or objects (the bottom picture in Fig. 1a). As a consequence, the estimation performance of the self-trained model degrades in real environments. As illustrated in Fig. 1d, 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 1, as well as an illustration in Fig. 1. As seen, LeoVR achieves delightful 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 (§3.1), and on the other hand, provide guidance for extracting reliable pixel-level training signals to train the depth map generator (§4.1). Embedding this *motion-aware* information into the system framework, our design of LeoVR excels in two unique aspects as follows.

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.

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. 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 (two indoor and two outdoor) different scenarios across 6 months, collecting 3,203 trajectories with 961,175 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 $0.15m$ and $0.18m$ when equipped with a Velodyne VLP-16 and Livox Mid-40 LiDAR respectively, outperforming comparative approaches by $> 21\%$ and $> 43\%$. 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 [27]. Without any pre-training and entirely based on self-supervised training, LeoVR still achieves an average depth estimation error of $0.21m$ when equipped with Livox Mid-40, which outperforms related works by 45.4% and is comparable to those supervised training methods.

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.
- We provide a fresh perspective to embed a vehicle's motion information into the system framework design. Based on an in-depth exploration of the spatio-temporal constraints behind motion information, we design a *motion-aware* fusion framework to boost the depth estimation accuracy and a *motion-instructed* self-supervised paradigm to ease the pixel-level ground truth annotation burden.
- We extensively evaluate the performance of LeoVR with 4 comparison works in 4 different real scenarios across 6 months. The results demonstrate that LeoVR could greatly broaden the capabilities of LiDAR-based mapping, especially realizing remarkable depth estimation and self-supervised training performance even with low-cost LiDARs.

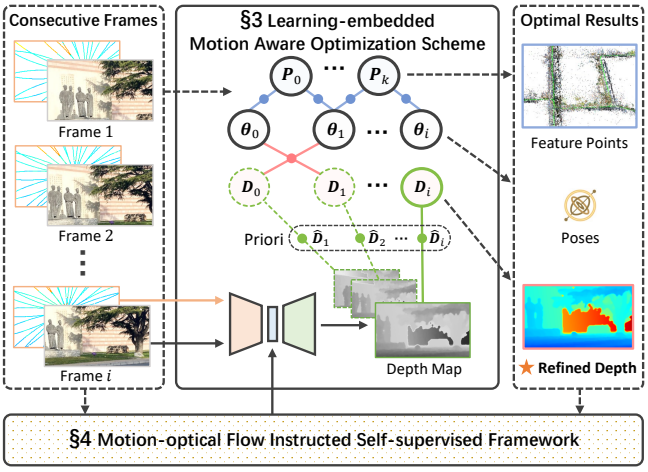


Figure 2: System architecture of LeoVR.

2 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* (§3) and a *motion-optical flow instructed self-supervised framework* (§4). 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 delightful depth estimation performance through a self-supervised training manner.

3 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 a similar DNN proposed in [56] as the initial depth map generator (implementation detailed in §5.1), which takes a 2D RGB image and 3D LiDAR point clouds as input and outputs a depth map of the current scene. However, 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

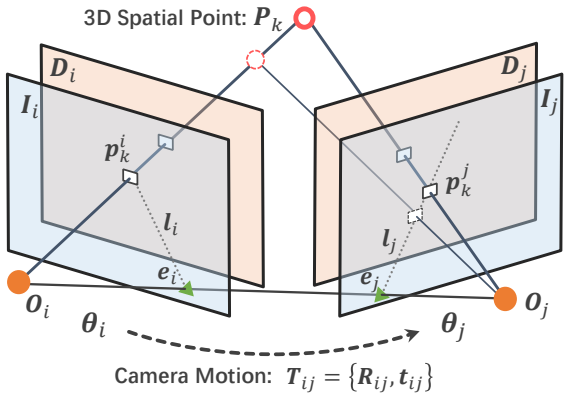


Figure 3: Illustration of some essential variables and their spatial relationships.

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 (§3.1). Further, we present how to formulate the *factor graph* with associated *factor nodes* and *variable nodes* to solve the optimization problem (§3.2).

3.1 Optimization Problem Statement

We first briefly describe and illustrate some essential definitions in our factor graph based optimization. There are three relevant reference systems in LeoVR: camera reference system C, LiDAR reference system L, and world reference system W. Therein, C and L are rigidly attached on a vehicle, and thus we omit the fixed transforms between them for simplicity. The ultimate goal of optimization is to obtain the refined depth map $\boldsymbol{D}_i$ with the initial value provided by the depth map generator. To fully exploit the spatial correlation between consecutive depth maps, we also continuously estimate the pose of the vehicle. Here, we denote the vehicle's 6 degrees of freedom (6-DoF) pose as a transformation from W to C. Specifically, the pose at the frame i is defined 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. In addition to the depth map and vehicle's pose, we also optimize the 3D feature point $\boldsymbol{P}_k$ extracted from multi-view visual images.

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.

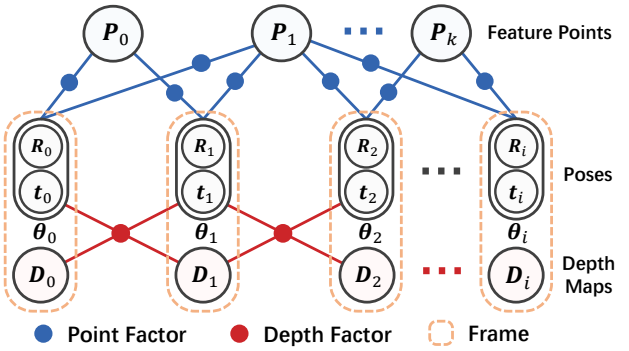


Figure 4: Overview of the factor graph in LeoVR.

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.

Denote $\mathcal{Z}_c$ as the full set of measurements from monocular camera C, LiDAR $\mathcal{L}$, and depth map generator $\mathcal{G}$ received within the smoothing window. We maximize the likelihood of the measurements $\mathcal{Z}_c$, given the history of states $\mathcal{X}_c$:

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

which can be formulated as the following nonlinear least squares problem:

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

where $\mathcal{N}_i$ are nearby frames of frame i. Each term is the residual associated to a factor type, weighted by the information matrix Σ (i.e., inverse of the covariance matrix, where $\|\boldsymbol{E}\|_\Sigma^2 = \boldsymbol{E}^\mathrm{T}\Sigma\boldsymbol{E}$). $\boldsymbol{E}_{vision}$ and $\boldsymbol{E}_{lidar}$ are two types of feature point residuals, and $\boldsymbol{E}_{depth}$ is depth map residual.

3.2 Factor Graph Formulation

The structure of the factor graph is shown in Fig. 4. 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.

3.2.1 ***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

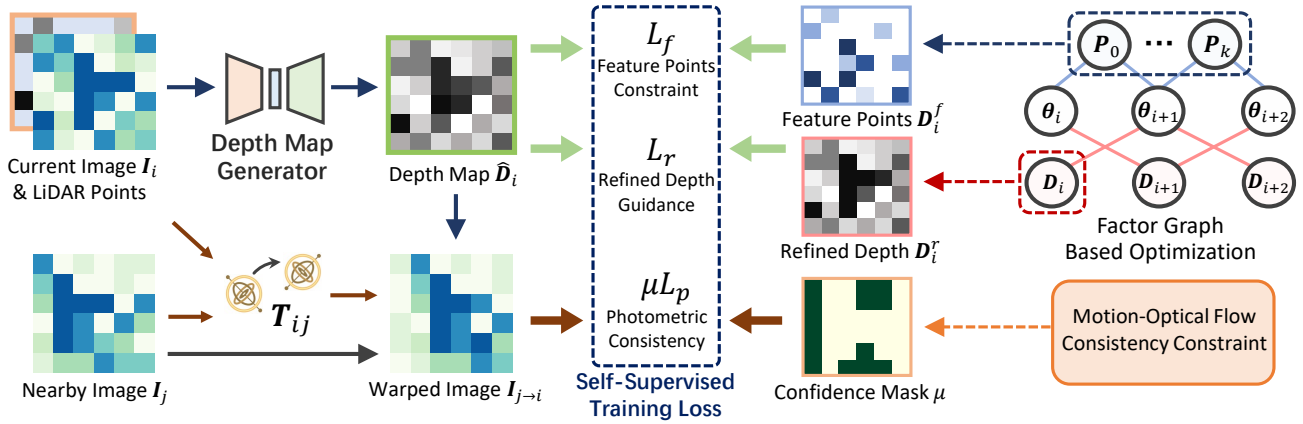


Figure 5: Workflow of the *motion-optical flow instructed self-supervised framework* in LeoVR.

$\pi : \mathrm{SE}(3) \times \mathbb{R}^3 \to \mathbb{R}^2$, which transforms a 3D point P in world reference W to the image plane given a vehicle's pose θ:

$$\pi(\theta, P) = \frac{1}{Z} K\theta P, \ P = [X, Y, Z]^T, \tag{5}$$

where K is the camera intrinsic matrix. Let $P_k \in \mathbb{R}^3$ denote a 3D feature point and $p_k^i \in \mathbb{R}^2$ denote the corresponding detection on the image plane I_i. The residual at pose θ_i for feature point P_k can be formulated as:

$$E_{vision}(\theta_i, P_k) = \pi(\theta_i, P_k) - p_k^i. \tag{6}$$

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 [40] to detect and describe visual feature points in images.

3.2.2 ***Point Factor with LiDAR Constraint.*** To take full advantage of the fusion of vision and LiDAR sensing modalities, we also optimize the feature points using LiDAR's overlapping field-of-view to provide depth estimates. We first project all the 3D LiDAR points $L \in \mathcal{L}_c$ in reference L between time t_c and t_{c+1} onto the image plane:

$$\pi(L) = \frac{1}{Z} KL, \ L = [X, Y, Z]^T. \tag{7}$$

For a 3D feature point P_k in reference W with the corresponding pixel p_k^i on the current image plane I_i, we find the projected LiDAR point $\pi(L_k)$ that is closest to p_k^i within a neighborhood of 5 pixels. Finally, the residual is computed as:

$$E_{lidar}(\theta_i, P_k) = \theta_i P_k - L_k. \tag{8}$$

This point factor punishes deviation of the feature point's depth from the LiDAR observation, optimizing the feature points and vehicle's pose while applying scale to visual measurements. As a reminder, we use only E_{vision} as the point factor when we cannot associate any LiDAR depth to a feature point due to the sparse resolution of the input LiDAR samples.

3.2.3 ***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 D_i and D_j, we can project D_j to the view of D_i based on the relative pose transformation $T_{ij} = \theta_j \theta_i^{-1}$. First, let p_k^i denote the coordinates of a pixel in D_i, and the corresponding 3D location is:

$$P_k = D_i(p_k^i) K^{-1} p_k^i. \tag{9}$$

Then, we project P_k to the image plane of D_j with $\pi(T_{ij}, P_k)$. Therefore, we can obtain a depth map which is projected from D_j to the view of D_i:

$$D_{j\to i}(p_k^i) = D_j\Big(\pi\big(T_{ij}, P_k\big)\Big). \tag{10}$$

When the sampling interval between two consecutive frames is short, the observations of the scene from different views should be consistent. Consequently, the residual between depth maps can be formulated as:

$$E_{depth}(D_i, D_j) = \sum_{k\in\Omega} \left\| D_i(p_k^i) - D_{j\to i}(p_k^i) \right\|^2. \tag{11}$$

This depth factor constraints the depth map of adjacent frames based on the consistency of the scene geometry in different views, ensuring the continuity and consistency of the depth maps estimated from moving vehicle. For faster convergence, we sample a different set of pixels at each iteration to stochastically optimize the residual over the whole depth map. Furthermore, while this depth factor can offer effective optimization in most cases, we should notice that the constraint breaks down when the target object is moving fast and close to the vehicle. As a result, we consider masking the invalid correspondences caused by the above dynamics for stable optimization, i.e., only pixels of objects within a trusted region are available for optimization, which is defined as the convex hull on the pixel space formed by the stable 3D feature points extracted and triangulated from visual odometry.

4 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. 5. As seen, given the current image I_i, the nearby image I_j, the depth map $\hat{D}_i$ output from the DNN generator, as well as the inter-frame camera's

motion $\boldsymbol{T}_{ij}$ acquired from the previous optimization scheme, 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 [33]. In this section, we first present how to implement this basic idea (§4.1), followed by additionally introducing a *motion-optical flow consistency* constraint to improve the robustness of this framework in real complicated environments (§4.2). We further provide another two constraints that could be put together for a better training performance (§4.3).

4.1 The Basic Idea: Photometric Loss

As mentioned above, the basic and essential supervision signal for training the depth map generator comes from the *photometric consistency*. Similar to the formulation in Section 3.2.3, let $\boldsymbol{p}_k^i$ denote the coordinates of a pixel in image $\boldsymbol{I}_i$, and any $\boldsymbol{p}_k^i$ having the corresponding projection in $\boldsymbol{I}_j$ as:

$$\boldsymbol{p}_k^j = \boldsymbol{\pi}\Big(\boldsymbol{T}_{ij}, \hat{\boldsymbol{D}}_i(\boldsymbol{p}_k^i)\boldsymbol{K}^{-1}\boldsymbol{p}_k^i\Big). \tag{12}$$

Consequently, we can create a warped image $\boldsymbol{I}_{j\to i}$ by:

$$\boldsymbol{I}_{j\to i}(\boldsymbol{p}_k^i) = \boldsymbol{I}_j\langle \boldsymbol{p}_k^j\rangle, \tag{13}$$

where $\langle\cdot\rangle$ is the differentiable bilinear sampler [24] that interpolates around the four immediate neighbours of $\boldsymbol{p}_k^j$. Then, we train the model to predict a depth map $\hat{\boldsymbol{D}}_i$ by minimizing the difference between $\boldsymbol{I}_i$ and $\boldsymbol{I}_{j\to i}$ and define the photometric loss as follows:

$$L_p = \sum_{j\in\mathcal{N}_i} pe\left(\boldsymbol{I}_i, \boldsymbol{I}_{j\to i}\right), \tag{14}$$

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 $SSIM$[49] to obtain the *photometric error* (i.e., pe in Eq. 14), a perceptual metric that is invariant to local illumination changes:

$$\begin{aligned} pe\left(\boldsymbol{I}_a, \boldsymbol{I}_b\right) =& \frac{\alpha}{2}\Big(1 - \mathrm{SSIM}\left(\boldsymbol{I}_a, \boldsymbol{I}_b\right)\Big) \\ &+ (1-\alpha)\left\|\boldsymbol{I}_a - \boldsymbol{I}_b\right\|_1. \end{aligned} \tag{15}$$

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

4.2 Motion-Optical Flow Constraint

Although one can leverage Eq. 14 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 [64]. Briefly, the basic idea assumes the photometric variation of pixels between adjacent images primarily depends on the motion of the vehicle itself. 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.

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

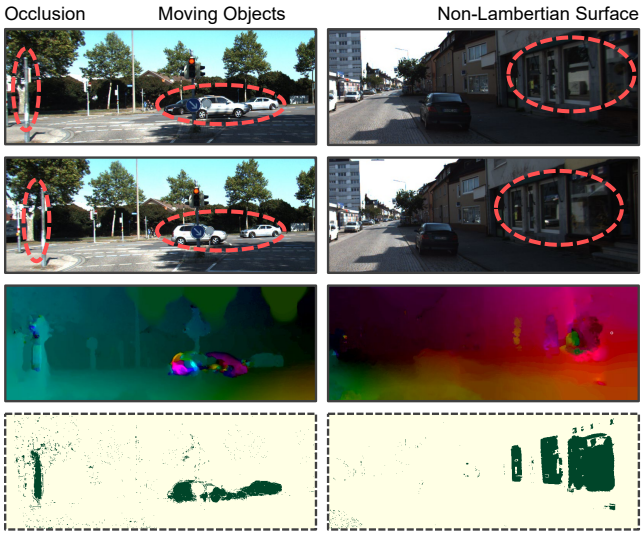


Figure 6: Qualitative results of motion-optical flow consistency constraint. We show two complicated real-world scenarios from the KITTI dataset [26] that severely impact self-supervised training. The first and second rows show consecutive RGB frames collected during the vehicle's movement. The third 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.

To improve the robustness of self-supervised training with photometric consistency, we propose an *motion-optical flow consistency constraint*. The key idea behind this constraint is to select pixels whose associated spatial points meet the static-rigid world assumption for training. Our insight is based on two observations: *(i)* the motion of ideal points (i.e., static and non-occluded spatial points) relative to the camera is exactly opposite to the motion of the camera itself; and *(ii)* the optical flow of spatial points on a non-Lambertian surface is not consistent with its actual motion. In a nutshell, merely those pixels whose optical flow is consistent with the camera's motion could be exploited to extract supervision signals. Here, optical flow is the motion pattern of image objects between two consecutive frames caused by the movement of object or camera. It is a 2D vector field in which each vector represents the displacement of points from one frame to the next [12].

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 [11]. 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. Then, we analyze the geometric relationship between camera motion and spatial point, as shown in Fig. 3. The optical centers $\boldsymbol{O}_i$ and $\boldsymbol{O}_j$ can determine an epipolar plane with the 3D point $\boldsymbol{P}_k$. The intersection line between the epipolar plane and the image plane is the epipolar line (i.e., $\boldsymbol{l}_i$

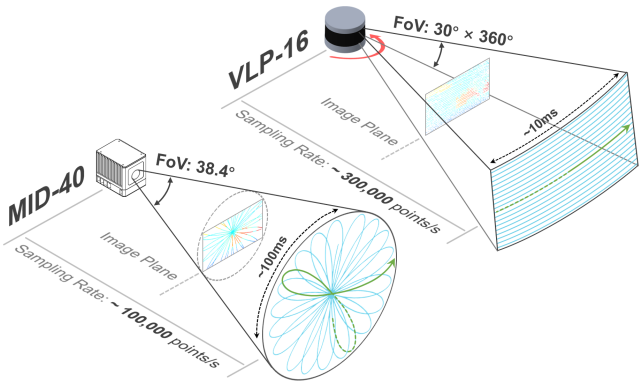


Figure 7: The scanning principles and basic specifications of Mid-40 and VLP-16

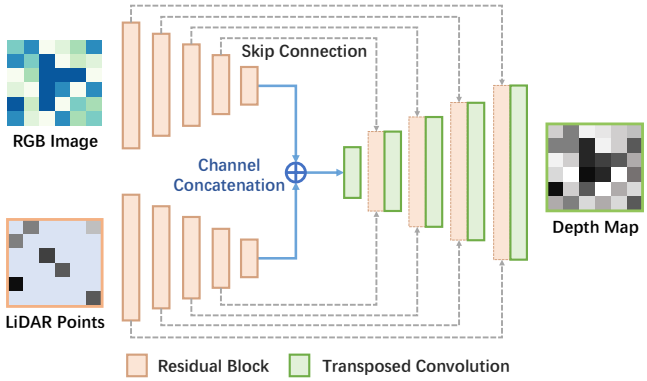


Figure 8: Network architecture of the proposed *depth map generator*

and $\boldsymbol{l}_j$). When the camera motion is determined, the ray $\boldsymbol{O}_i\boldsymbol{p}_k^i$ is the possible location of $\boldsymbol{P}_k$, and $\boldsymbol{l}_j$ is the corresponding potential projection location on the image $\boldsymbol{I}_j$. Therefore, if $\boldsymbol{p}_k^i$ is consistent with the camera motion, $\boldsymbol{p}_k^j$ calculated by optical flow should fall on $\boldsymbol{l}_j$. The epipolar constraint [62] can be applied to verify whether the point satisfy the above property:

$$(\boldsymbol{p}_k^j)^T \boldsymbol{F} \boldsymbol{p}_k^i = 0, \tag{16}$$

which evaluates the distance from the projection point to the epipolar line. And the fundamental matrix $\boldsymbol{F}$ can be calculated from the camera motion. 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:

$$\mu = \left[\left| (\boldsymbol{p}_k^j)^T \boldsymbol{F} \boldsymbol{p}_k^i \right| < \lambda \right], \tag{17}$$

where $[\cdot]$ is the Iverson bracket, and λ is the constraint threshold. As shown in Fig. 6, 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 (§5.4.2).

4.3 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 (§3.2), 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 §5.4.3.

Feature Points Constraint. As described in Section 3, 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:

$$L_f = \left\| (\hat{\boldsymbol{D}}_i - \boldsymbol{D}_i^f) \odot \left[\boldsymbol{D}_i^f > 0 \right] \right\|^2, \tag{18}$$

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:

$$L_r = \left\| \hat{\boldsymbol{D}}_i - \boldsymbol{D}_i^r \right\|^2. \tag{19}$$

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.

4.4 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 [14, 33]. Putting these constraints together, our final objective for the entire self-supervised training is:

$$L = \mu L_p + \lambda_f L_f + \lambda_r L_r + \lambda_s L_s, \tag{20}$$

where λ_f, λ_s, and λ_r are the relative weightings to balance the terms. We will describe the training details in Section 5.1.

5 IMPLEMENTATION AND EVALUATION

We first present the experimental methodology (§5.1), followed by the overall performance of LeoVR compared against SOTA systems (§5.2). We then conduct experiments under different conditions to evaluate the robustness of LeoVR (§5.3) Finally, we conduct an ablation study to understand each framework module or self-supervised constraint in LeoVR (§5.4).

Table 2: 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	822,856	295.6	488	2164
2	Campus	Ladybug5+, Mid-40 & VLP-16	49,879	8.76	88	100
3	Classroom Building	Logitech C920E, Mid-40	47,064	2.37	87	79
4	Office Building	Logitech C920E, Mid-40	41,376	1.97	92	105

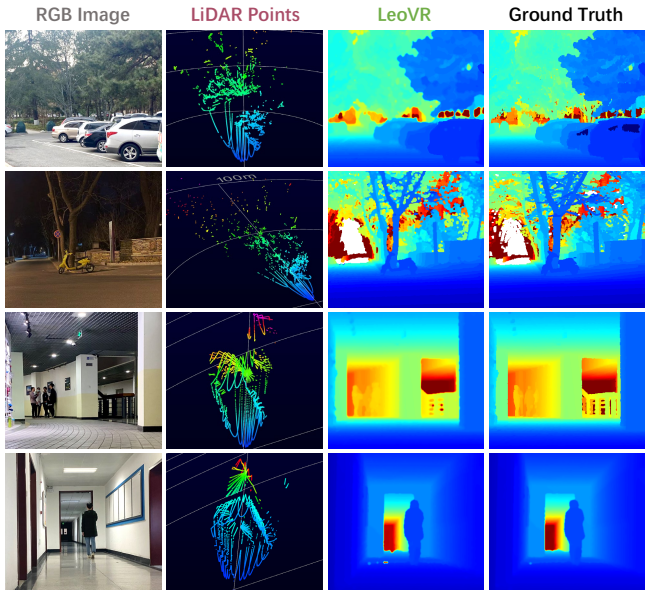


Figure 9: Generated depth maps from LeoVR. From top to bottom are examples from city road, campus, classroom, and office building scenarios, respectively.

5.1 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, 30Hz) and Ladybug5+ (2k, 30Hz) 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. 7, 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 degree FoV with a sampling rate of 100,000 points/s [28]. VLP-16 is a conventional mechanical spinning LiDAR, which rotates 16 uniformly distributed lasers simultaneously for scanning with a rotation frequency of 5-20Hz. VLP-16 has a 360 degree horizontal and 30 degree vertical FoV with a sampling rate of 300,000 points/s [46]. 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 [34]). For the optimization module of LeoVR, the server is equipped with Intel(R) Xeon(R) CPU E5-2620 v4 of 2.10GHz main frequency and 256G 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. We continuously evaluate the system performance across 6 months, collecting 3203 trajectories with 961,175 frames.

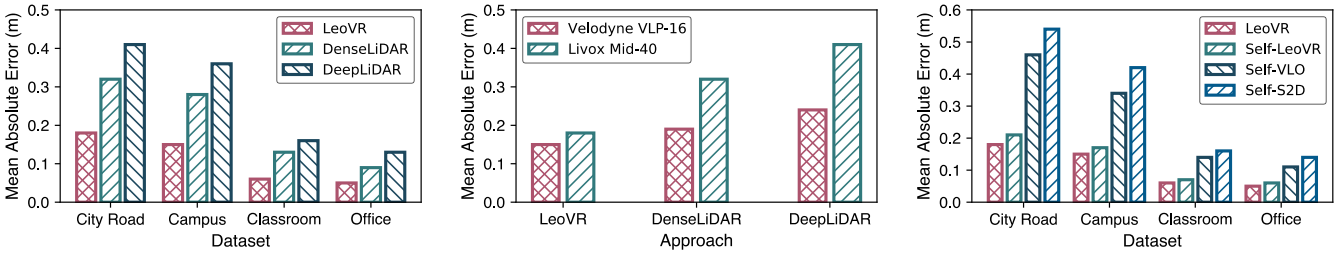
Table 3: Motion Tracking Performance Comparison

System	Absolute Trajectory Error (cm)			
	City Road	Campus	Classroom	Office Building
V-LOAM	5.1	3.9	3.4	3.2
LeoVR	**4.2**	**3.3**	**3.1**	**2.9**

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 [16, 39]. 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 [60]) to generate dense depth maps.

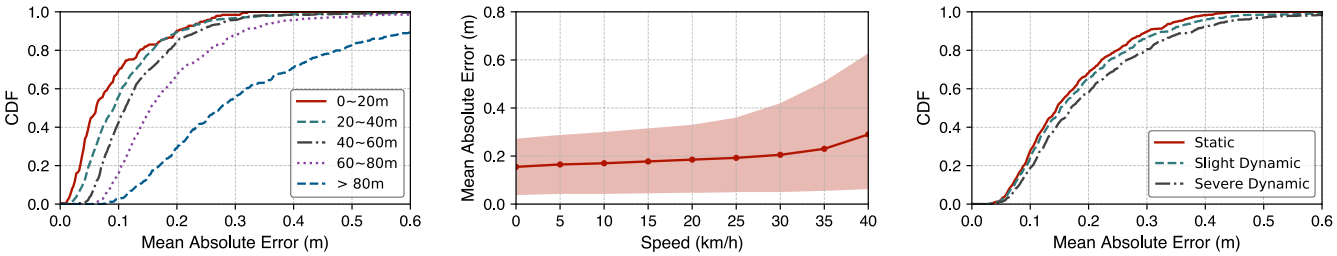
Dataset. As summarized in Table 2. We build datasets using the collected RGB images and LiDAR samples for further reuse. We conduct experiments in four representative scenarios: city road and campus as outdoor scenarios, while classroom building and office building as indoor scenarios. These scenes enjoy 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 5.5.2). 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



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

Figure 10: Overall Depth Estimation Performance Comparison



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

Figure 11: System Robustness Evaluation

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 network architecture of our depth map generator is illustrated in Fig. 8. It follows an encoder-decoder structure and consists of three major components: an RGB image encoder, a LiDAR points encoder, and a depth estimation decoder. The features extracted in each branch are fused by channel concatenation and further fused together with decoder features via skip connection at each scale. We use the residual blocks of ResNet-34 [18] and a stride of 2 for downsampling in the encoders. After every encoding layer, the number of channels in the feature map is [32, 64, 128, 256, 256]$^*k$, where $k = 3/4$ for three channels RGB image branch, and $k = 1/4$ for one channel depth branch,. We use batch normalization [22] and ReLU activation for all layers except the last one.

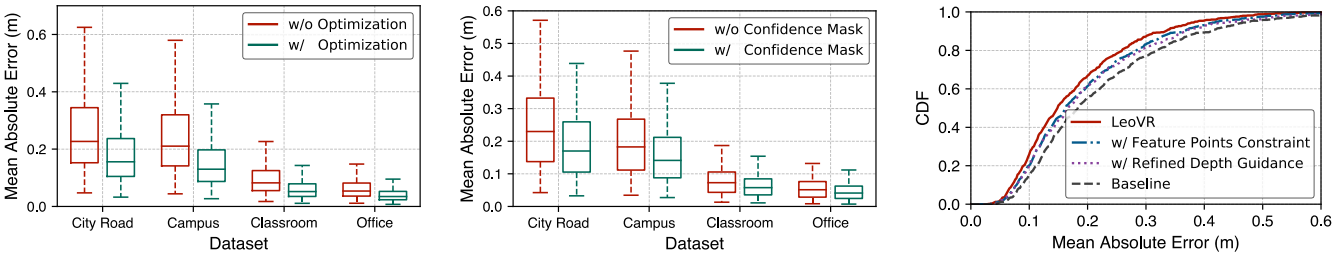
We implement the proposed model based on PyTorch [37]. We use Adam optimizer [25] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ and batch size of 8 to train the model for 32 epochs. The initial learning rate starts from 10^{-4} and decreases by half every 8 epochs. We update the confidence mask and supervision signals provided by the factor graph every two epochs in the last 8 fine-tune epochs during self-supervised training. We empirically set the hyper-parameters as follows: $\lambda_v = 0.2$, $\lambda_r = 0.1$, and $\lambda_s = 0.1$.

5.2 Overall Performance Comparison

5.2.1 Depth Estimation Accuracy Comparison. 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. 10a, LeoVR achieves the optimal performance in all scenarios with different environmental complexity. The estimation accuracy of LeoVR is 0.18m, 0.15m, 0.06m, 0.05m in city road, campus, classroom building, and office building, outperforming DenseLiDAR and DeepLiDAR by more than 43.8% and 56.1%. Fig. 9 shows qualitative results of LeoVR in different scenarios. Intuitively, the generated depth maps could profile the details of objects and be comparable to the ground truth.

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. 10b, the MAE of LeoVR with Velodyne VLP-16 and Livox Mid-40 is 0.15m and 0.18m respectively,a $>$ 21.1% and 43.8% 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 §5.4.1.

5.2.2 Self-Supervised Performance Comparison. 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 2, 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. 10c. As seen, the depth estimation performance of the self-supervised LeoVR is 0.21m, 0.17m, 0.07m, 0.06m in four different scenarios respectively, which decrease by 14.3%, 11.7%, 14.2%, and 16.7% compared to that of the supervised version. Meanwhile, the self-supervised LeoVR outperforms comparative approaches by



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

Figure 12: Ablation Study

> 45%. Especially in the city road scenario where the environment suffers from the most dynamics, LeoVR outperforms existing works by 54.3%. The delightful results demonstrate that the motion-optical flow-instructed self-supervised framework could boost the model training performance, especially in complicated real-world environments. A detailed understanding of the contribution from each constraint (i.e., supervision signals) will be presented in §5.4.3.

5.2.3 Motion Tracking Performance Comparison. Taking full advantage of the vehicle's motion information is the essential reason why LeoVR could achieve delightful depth estimation and self-supervision training performance. Therefore, the ability to preciously track the vehicle's motion (i.e., 6-DoF pose) is the basis of the entire system. In this experiment, we evaluate the motion tracking performance of LeoVR and compare it with V-LOAM [61], a visual-LiDAR odometry work for camera motion tracking. Although there are SOTA robotic localization systems by leverage RF signals [1, 32], the vision and LiDAR based solutions are more suitable for vehicles in outdoors. We conduct this experiment with Livox Mid-40, while the ground truth is obtained through V-LOAM leveraging the 80-line RoboSense Ruby Lite LiDAR. And we use the *absolute trajectory error* (ATE [43], in *cm*) to evaluate the motion tracking accuracy. The results are shown in Table 3. As seen, LeoVR achieves an ATE within 4.2cm in all scenarios, and the performance outperforms V-LOAM by around 17%. The results demonstrate the effectiveness of the optimization framework for a vehicle's motion tracking, and such an accurate tracking accuracy ensures the reliable performance of the subsequent modules in LeoVR.

5.3 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.

5.3.1 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. The results are shown in Fig. 11a. As seen, the accuracy of estimation gradually decreases with increasing distance, and the MAE are 0.09m, 0.11m, 0.14m, and 0.19m for the distances of 0-20m, 20-40m, 40-60m, and 60-80m, respectively. When the scene distance rises to over 80m, the average error is up to 0.34m because the accuracy and density of the points cloud decrease significantly with increasing sensing distance. Nevertheless, LeoVR can still achieve 95th percentile error less than 0.38m within the 80m range.

5.3.2 Impact of Vehicle Speed. We further verify the robustness of LeoVR on different speeds of the vehicle. As shown in Fig. 11b, at speed less than 20km/h, the performance of LeoVR is stable, and the average depth error is 0.19m. When the moving speed = 0km/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. When the speed exceeds 40km/h, the average depth error of LeoVR is 0.29m, and 95th percentile error is within 0.63m. 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.

5.3.3 Impact of Environmental Dynamics. We evaluate the robustness of LeoVR on different levels of scene dynamics. As shown in Fig. 11c, we evaluate LeoVR in static (without moving objects), slight dynamic (few pedestrians and cars), and severe dynamic (crowded traffic) scenes. The average depth estimation error are 0.171m, 0.184m, and 0.207m in static, slight, and severe dynamic scenes, respectively, which are decreased by 7.5% and 17.6% from static to slight and severe scenes, In addition, LeoVR can still achieve 95th percentile error less than 0.44m in severe dynamic environments.

The rationale behind the results of the above robustness experiments lies in twofold: *(i)* The introduction of the motion-optical flow consistency constraint during self-supervised training enhances resistance of the DNN to dynamics; and *(ii)* The motion-aware optimization framework improves the system robustness. Although the dynamics of the environment, vehicle's speed, etc. , could harm the depth estimation of a single frame, by leveraging the full information (i.e., depth maps, 3D feature points, and vehicle's motion) within a time window, we can optimize a more accurate result that meets the spatio-temporal constraint.

5.4 Ablation Study

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

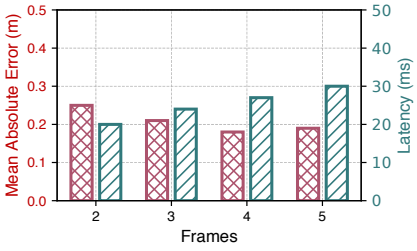


Figure 13: Impact of Sliding Window

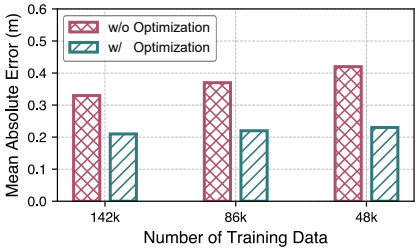


Figure 14: Impact of Training Data

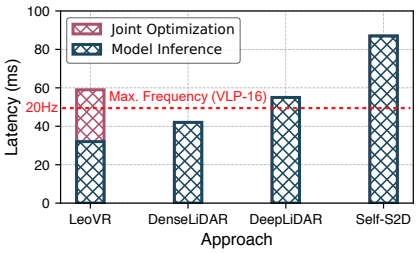


Figure 15: System Efficiency

5.4.1 Effectiveness of the Motion-aware Optimization Framework. 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 DNN. The experimental results are shown in Fig. 12a. 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 $0.18m$,$0.15m$,$0.06m$,$0.04m$ in city road, campus, classroom building, and office building respectively, reducing that of LeoVR without optimization by $> 31.3\%$. Especially in the complicated campus environment, the MAE reduction rate is $> 38.2\%$. Additionally, with the optimization framework, the maximum value of the estimation error could be bounded by $0.47m$. The above delightful performance demonstrates the effectiveness of the proposed motion-aware optimization framework.

5.4.2 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 Eq. 17. The results are shown in Fig. 12b. For the classroom building and office building, the proposed training method with confidence masks can improve 17.8% and 13.1%. As for the city road and campus, the performance is increased by 23.4% and 21.5%, 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.

5.4.3 Effectiveness of Additional Constraints. We finally 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 depicted in Fig. 12c, the MAE reduced by 9.8% when the *feature points constraint* is introduced on the basis of the baseline. Similarly, when the *refined depth guidance* is introduced, there is a 7.1% lift on the depth estimation performance (i.e., reducing the MAE from $0.212m$ to $0.197m$). These delightful results demonstrate that the additional two constraints could also bring useful supervision signals to train the DNN.

5.5 Parameter Study

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

5.5.1 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. 13, when using only two adjacent frames, the depth estimation error is $0.25m$ with a 20ms delay (the computational delay considers all factor constraints presented in Section 3.2). With the window length increasing to 4 frames, the error decreases by 27.3%, and the optimization latency increases to 27ms. 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.

5.5.2 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. 14, the average estimation error of LeoVR with and without optimization are $0.33m$ and $0.21m$ when using 142k frames for self-supervised training. When the training data number decreased to 48k, the error scale to $0.42m$ and $0.23m$, respectively. The performance degradation of LeoVR without optimization is 27.3%, while the entire LeoVR is only 9.2%. The above results demonstrate that although the self-supervised training performance degrades with limited training data amount, the subsequent optimization scheme could still maintain the estimation accuracy.

5.6 System Efficiency Study

As a depth estimation system towards auto-driving scenarios, we further evaluate the efficiency of LeoVR. As shown in Fig. 15, 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 59ms, 42ms, 55ms, and 87ms. The latency of the DNN and

the optimization scheme in LeoVR are 32ms and 27ms, respectively. In general, LeoVR could work with a frequency of 16Hz, which matches the LiDAR sampling frequency (e.g., VLP-16, 5-20Hz). Additionally, benefiting from our optimization scheme, LeoVR could select a more light-weight model to guarantee the real-time performance.

6 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 [31]. 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 [30]. 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 [2], or offloading some computation-intensive yet delay-tolerant tasks to an edge server [52] 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[28, 45] due to their hardware nature, resulting in LeoVR not capable of handling high-speed scenarios (e.g., vehicle speed > $40km/h$). We think introducing IMU [50] or wheel odometry [5] 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.

7 RELATED WORK

We briefly review the most related works in this section.

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 [4, 15, 35, 41, 61, 65]. Among them, RF-Fusion [4] fuses vision and RFID to enable robots to recognize objects. ITrackU [6] leverages IMU and UWB radar for tracking a pen-like instrument. V-LOAM [61] integrates vision and LiDAR in a loosely coupled manner to track a camera's motion and generate environmental 3D point clouds. VILENS [50] and FollowUpAR [53] 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 [15] 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 Depth Estimation with visual-LiDAR fusion. Recent years have witnessed the emergence of environmental perception for IoT devices through deep learning [3, 36, 38, 57, 59]. Some most related works [21, 27, 63] design learning-based framework to fuse vision and LiDAR for environment depth estimation. Specifically, each of them designs a DNN, which takes monocular images and LiDAR point clouds as inputs and predicts the depth for each pixel in an image. Recently, DeepLiDAR [39] and DenseLiDAR [16] exploit pseudo-depth maps obtained from morphological operations to instruct the higher-level network to generate a more accurate depth map. PENet [21] proposes a two-branch backbone that consists of a color-dominant branch and a depth-dominant branch to extract distinctive features for further fusion and optimization. However, the performance of these solutions also highly relies on the density of the point clouds scanned by the LiDAR and suffers from severe performance degradation when equipped with commercial in-vehicle LiDARs.

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 [14, 33, 55, 64]. For instance, Self-S2D [33] takes a sequence of images with depth maps as inputs and uses Perspective-n-Point to align them for photometric consistency. [64] proposes an unsupervised learning framework for monocular depth and camera motion estimation from unstructured video sequences. However, the training performance of the above self-supervised solutions degrades in practical scenarios with dynamic objects, occlusion, and non-Lambertian surfaces, leaving room for improvement. Compared with these works, LeoVR further digs into the vehicle's motion information and designs several motion-aware constraints to extract supervision signals in complicated environments, improving the self-supervised training performance.

8 CONCLUSION

We have presented the design and implementation of LeoVR, a self-supervised environment depth estimation system with visual-radar 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 6 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.

9 ACKNOWLEDGMENTS

We sincerely thank our shepherd, Prof. Eric Rozner, and the anonymous reviewers for their valuable feedback in improving this work. This work is supported in part by the NSFC under grant No. 61832010, No. 61972131, and No. 62002193.

REFERENCES

[1] Roshan Ayyalasomayajula, Aditya Arun, Chenfeng Wu, Sanatan Sharma, Abhishek Rajkumar Sethi, Deepak Vasisht, and Dinesh Bharadia. 2020. Deep learning based wireless localization for indoor navigation. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*. 1–14.

[2] Michael Bloesch, Jan Czarnowski, Ronald Clark, Stefan Leutenegger, and Andrew J Davison. 2018. CodeSLAM—learning a compact, optimisable representation for dense visual SLAM. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2560–2568.

[3] Sudershan Boovaraghavan, Anurag Maravi, Prahaladha Mallela, and Yuvraj Agarwal. 2021. MLIoT: An end-to-end machine learning system for the Internet-of-Things. In *Proceedings of the International Conference on Internet-of-Things Design and Implementation*. 169–181.

[4] Tara Boroushaki, Isaac Perper, Mergen Nachin, Alberto Rodriguez, and Fadel Adib. 2021. RFusion: Robotic Grasping via RF-Visual Sensing and Learning. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*. 192–205.

[5] Martin Brossard and Silvere Bonnabel. 2019. Learning wheel odometry and IMU errors for localization. In *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 291–297.

[6] Yifeng Cao, Ashutosh Dhekne, and Mostafa Ammar. 2021. ITrackU: tracking a pen-like instrument via UWB-IMU fusion. In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*. 453–466.

[7] Yun Chen, Bin Yang, Ming Liang, and Raquel Urtasun. 2019. Learning joint 2d-3d representations for depth completion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 10023–10032.

[8] Jaehoon Choi, Dongki Jung, Yonghan Lee, Deokhwa Kim, Dinesh Manocha, and Donghwan Lee. 2020. SelfDeco: Self-Supervised Monocular Depth Completion in Challenging Indoor Environments. *arXiv preprint arXiv:2011.04977* (2020).

[9] Jan Czarnowski, Tristan Laidlow, Ronald Clark, and Andrew J Davison. 2020. Deepfactors: Real-time probabilistic dense monocular slam. *IEEE Robotics and Automation Letters* 5, 2 (2020), 721–728.

[10] Autonomous driving accident. 2020. https://www.motortrend.com/news/tesla-model-3-crash-taiwan-autopilot-accident/

[11] Gunnar Farnebäck. 2003. Two-frame motion estimation based on polynomial expansion.

[12] OpenCV: Optical Flow. 2022. https://docs.opencv.org/3.4/d4/dee/tutorial_optical_flow.html

[13] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. 2018. Deep ordinal regression network for monocular depth estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

[14] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J Brostow. 2019. Digging into self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*.

[15] Johannes Graeter, Alexander Wilczynski, and Martin Lauer. 2018. Limo: Lidar-monocular visual odometry. In *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*.

[16] Jiaqi Gu, Zhiyu Xiang, Yuwen Ye, and Lingxuan Wang. 2021. DenseLiDAR: A Real-Time Pseudo Dense Depth Guided Depth Completion Network. *IEEE Robotics and Automation Letters* (2021).

[17] Christian Häne, Christopher Zach, Jongwoo Lim, Ananth Ranganathan, and Marc Pollefeys. 2011. Stereo depth map fusion for robot navigation. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 1618–1625.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

[19] Yuze He, Li Ma, Zhehao Jiang, Yi Tang, and Guoliang Xing. 2021. VI-eye: semantic-based 3D point cloud registration for infrastructure-assisted autonomous driving. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*. 573–586.

[20] Mina Henein, Jun Zhang, Robert Mahony, and Viorela Ila. 2020. Dynamic SLAM: The need for speed. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*.

[21] Mu Hu, Shuling Wang, Bin Li, Shiyu Ning, Li Fan, and Xiaojin Gong. 2021. PENet: Towards Precise and Efficient Image Guided Depth Completion. *arXiv preprint arXiv:2103.00783* (2021).

[22] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*.

[23] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et al. 2011. KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*. 559–568.

[24] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. 2015. Spatial transformer networks. *Advances in neural information processing systems* (2015).

[25] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[26] KITTI-Dataset. 2017. http://www.cvlibs.net/datasets/kitti/eval_depth.php?benchmark=depth_completion

[27] Bin Li, Mu Hu, Shuling Wang, Lianghao Wang, and Xiaojin Gong. 2021. Self-supervised Visual-LiDAR Odometry with Flip Consistency. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*.

[28] Jiarong Lin and Fu Zhang. 2020. Loam livox: A fast, robust, high-precision LiDAR odometry and mapping package for LiDARs of small FoV. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*.

[29] Livox. 2021. https://www.livoxtech.com/application/autonomous-driving

[30] Chris Xiaoxuan Lu, Stefano Rosa, Peijun Zhao, Bing Wang, Changhao Chen, John A Stankovic, Niki Trigoni, and Andrew Markham. 2020. See through smoke: robust indoor mapping with low-cost mmWave radar. In *MobiSys*.

[31] Chris Xiaoxuan Lu, Muhamad Risqi U Saputra, Peijun Zhao, Yasin Almalioglu, Pedro PB de Gusmao, Changhao Chen, Ke Sun, Niki Trigoni, and Andrew Markham. 2020. milliEgo: single-chip mmWave radar aided egomotion estimation via deep sensor fusion. In *SenSys*.

[32] Zhihong Luo, Qiping Zhang, Yunfei Ma, Manish Singh, and Fadel Adib. 2019. 3D backscatter localization for fine-grained robotics. In *16th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 19)*. 765–782.

[33] Fangchang Ma, Guilherme Venturelli Cavalheiro, and Sertac Karaman. 2019. Self-supervised sparse-to-dense: Self-supervised depth completion from lidar and monocular camera. In *2019 International Conference on Robotics and Automation (ICRA)*.

[34] Mid-40-specs. 2022. https://www.livoxtech.com/cn/mid-40-and-mid-100/specs

[35] Chanoh Park, Peyman Moghadam, Soohwan Kim, Alberto Elfes, Clinton Fookes, and Sridha Sridharan. 2018. Elastic lidar fusion: Dense map-centric continuous-time slam. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*.

[36] Jinsun Park, Kyungdon Joo, Zhe Hu, Chi-Kuei Liu, and In So Kweon. 2020. Non-local spatial propagation network for depth completion. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIII 16*.

[37] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* (2019).

[38] Hang Qiu, Jinzhu Chen, Shubham Jain, Yurong Jiang, Matt McCartney, Gorkem Kar, Fan Bai, Donald K Grimm, Marco Gruteser, and Ramesh Govindan. 2017. Towards robust vehicular context sensing. *IEEE Transactions on Vehicular Technology* 67, 3 (2017), 1909–1922.

[39] Jiaxiong Qiu, Zhaopeng Cui, Yinda Zhang, Xingdi Zhang, Shuaicheng Liu, Bing Zeng, and Marc Pollefeys. 2019. Deeplidar: Deep surface normal guided depth prediction for outdoor scene from sparse lidar data and single color image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[40] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. 2011. ORB: An efficient alternative to SIFT or SURF. In *2011 International conference on computer vision*.

[41] Sriram Sami, Yimin Dai, Sean Rui Xiang Tan, Nirupam Roy, and Jun Han. 2020. Spying with your robot vacuum cleaner: eavesdropping via lidar sensors. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*. 354–367.

[42] Lucas Teixeira, Martin R Oswald, Marc Pollefeys, and Margarita Chli. 2020. Aerial single-view depth completion with image-guided uncertainty estimation. *IEEE Robotics and Automation Letters* (2020).

[43] TUM tools. 2021. https://vision.in.tum.de/data/datasets/rgbd-dataset/tools

[44] Velodyne. 2021. https://velodynelidar.com/products/

[45] Antoni Rosinol Vidal, Henri Rebecq, Timo Horstschaefer, and Davide Scaramuzza. 2018. Ultimate SLAM? Combining events, images, and IMU for robust visual SLAM in HDR and high-speed scenarios. *IEEE Robotics and Automation Letters* 3, 2 (2018), 994–1001.

[46] VLP-16. 2022. https://velodynelidar.com/products/puck/

[47] Jun Wang, Li Zhang, Yanjun Huang, and Jian Zhao. 2020. Safety of autonomous vehicles. *Journal of advanced transportation* (2020).

[48] Yan Wang, Wei-Lun Chao, Divyansh Garg, Bharath Hariharan, Mark Campbell, and Kilian Q Weinberger. 2019. Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[49] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing* (2004).

[50] David Wisth, Marco Camurri, Sandipan Das, and Maurice Fallon. 2021. Unified Multi-Modal Landmark Tracking for Tightly Coupled Lidar-Visual-Inertial Odometry. *IEEE Robotics and Automation Letters* (2021).

[51] Jonas Wulff, Laura Sevilla-Lara, and Michael J Black. 2017. Optical flow in mostly rigid scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[52] Jingao Xu, Hao Cao, Danyang Li, Kehong Huang, Chen Qian, Longfei Shangguan, and Zheng Yang. 2020. Edge assisted mobile semantic visual slam. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 1828–1837.

[53] Jingao Xu, Guoxuan Chi, Zheng Yang, Danyang Li, Qian Zhang, Qiang Ma, and Xin Miao. 2021. FollowUpAR: Enabling Follow-up Effects in Mobile AR Applications. In *Proceedings of the ACM MobiSys*.

[54] Feng Xue, Guirong Zhuo, Ziyuan Huang, Wufei Fu, Zhuoyue Wu, and Marcelo H Ang. 2020. Toward hierarchical self-supervised monocular absolute depth estimation for autonomous driving applications. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

[55] Shen Yan, Yu Zheng, Wei Ao, Xiao Zeng, and Mi Zhang. 2020. Does unsupervised architecture representation learning help neural architecture search? *Advances in Neural Information Processing Systems* 33 (2020).

[56] Yanchao Yang, Alex Wong, and Stefano Soatto. 2019. Dense depth posterior (ddp) from single image and sparse range. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3353–3362.

[57] Shuochao Yao, Yifan Hao, Yiran Zhao, Ailing Piao, Huajie Shao, Dongxin Liu, Shengzhong Liu, Shaohan Hu, Dulanga Weerakoon, Kasthuri Jayarajah, et al. 2019. Eugene: Towards deep intelligence as a service. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 1630–1640.

[58] Zhichao Yin and Jianping Shi. 2018. Geonet: Unsupervised learning of dense depth, optical flow and camera pose. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

[59] Chungkuk Yoo, Saiyma Sarmin, Inseok Hwang, Eric Rozner, and Minsik Cho. 2020. Poster: DeepFind: Sensor-driven Inference Acceleration for Continuous Deep Mobile Vision Applications. In *HotMobile'20: Proceedings of the 21st International Workshop on Mobile Computing Systems and Applications*.

[60] Ji Zhang and Sanjiv Singh. 2014. LOAM: Lidar Odometry and Mapping in Real-time.. In *Robotics: Science and Systems*.

[61] Ji Zhang and Sanjiv Singh. 2015. Visual-lidar odometry and mapping: Low-drift, robust, and fast. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*.

[62] Zhengyou Zhang. 1998. Determining the epipolar geometry and its uncertainty: A review. *International journal of computer vision* (1998).

[63] Shanshan Zhao, Mingming Gong, Huan Fu, and Dacheng Tao. 2021. Adaptive context-aware multi-modal network for depth completion. *IEEE Transactions on Image Processing* (2021).

[64] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. 2017. Unsupervised learning of depth and ego-motion from video. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

[65] Xingxing Zuo, Yulin Yang, Patrick Geneva, Jiajun Lv, Yong Liu, Guoquan Huang, and Marc Pollefeys. 2020. Lic-fusion 2.0: Lidar-inertial-camera odometry with sliding-window plane-feature tracking. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.