SDV-LOAM: Semi-Direct Visual–LiDAR Odometry and Mapping

Zikang Yuan, Qingjie Wang, Ken Cheng, Tianyu Hao, and Xin Yang, *Member, IEEE*

***Abstract*—Visual-LiDAR odometry and mapping (V-LOAM), which fuses complementary information of a camera and a LiDAR, is an attractive solution for accurate and robust pose estimation and mapping. However, existing systems could suffer nontrivial tracking errors arising from 1) association between 3D LiDAR points and sparse 2D features (i.e., 3D-2D depth association) and 2) obvious drifts in the vertical direction in the 6-degree of freedom (DOF) sweep-to-map optimization. In this paper, we present SDV-LOAM which incorporates a semi-direct visual odometry and an adaptive sweep-to-map LiDAR odometry to effectively avoid the above-mentioned errors and in turn achieve high tracking accuracy. The visual module of our SDV-LOAM directly extracts high-gradient pixels where 3D LiDAR points project on for tracking. To avoid the problem of large scale difference between matching frames in the VO, we design a novel point matching with propagation method to propagate points of a host frame to an intermediate keyframe which is closer to the current frame to reduce scale differences. To reduce the pose estimation drifts in the vertical direction, our LiDAR module employs an adaptive sweep-to-map optimization method which automatically choose to optimize 3 horizontal DOF or 6 full DOF pose according to the richness of geometric constraints in the vertical direction. In addition, we propose a novel sweep reconstruction method which can increase the input frequency of LiDAR point clouds to the same frequency as the camera images, and in turn yield a high frequency output of the LiDAR odometry in theory. Experimental results demonstrate that our SDV-LOAM ranks 8th on the KITTI odometry benchmark which outperforms most LiDAR/visual-LiDAR odometry systems. In addition, our visual module outperforms state-of-the-art visual odometry and our adaptive sweep-to-map optimization can improve the performance of several existing open-sourced LiDAR odometry systems. Moreover, we demonstrate our SDV-LOAM on a custom-built hardware platform in large-scale environments which achieves both a high accuracy and output frequency. We have released the source code of our SDV-LOAM for the development of the community.**

***Index Terms*—Mapping, pose estimation, visual -LiDAR sensor system.**

Manuscript received 17 February 2022; revised 14 March 2023; accepted 26 March 2023. Date of publication 29 March 2023; date of current version 4 August 2023. This work was supported in part by the National Natural Science Foundation of China under Grants 62122029, 62061160490, and U20B2064, and in part by Wuhan Science and Technology Bureau under Grants 2020010601012167. Recommended for acceptance by H. Ling. *(Corresponding author: Xin Yang.)*

Zikang Yuan is with the Institute of Artificial Intelligence, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: yzk2020@hust.edu.cn).

Qingjie Wang, Ken Cheng, Tianyu Hao, and Xin Yang are with the Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: wqj@hust.edu.cn; kencheng@hust.edu.cn; hty@hust.edu.cn; xinyang2014@hust.edu.cn).

Digital Object Identifier 10.1109/TPAMI.2023.3262817

I. Introduction

VISUAL odometry and Light Detection and Ranging (LiDAR) odometry, which are two widely-used solutions for 6-DOF pose estimation and mapping, are fundamental techniques for many robotics and computer vision applications, e.g., driverless cars and automatic navigation. Using a RGB camera or a LiDAR alone as the only input sensor for an odometry has complementary advantages and drawbacks. Visual odometry [1], [2], [3], [4], [5], [6], [7] can output pose and map points at a high frequency (i.e., 30-60 Hz), but the accuracy is not as good as LiDAR odometry due to its poor robustness to blurred images and low textures. In contrast, LiDAR odometry [8], [9], [10], [11] can generally provide more accurate pose estimation and mapping than visual odometry, while its output frequency is limited by the low frequency of input point clouds (e.g., 10 Hz).

Many works have been conducted to combine complementary advantages of visual and LiDAR sensors in the literature. State-of-the-art visual-LiDAR odometry methods can be classified into two categories, i.e., loosely integrated and tightly integrated. Loosely integrated methods (e.g., DEMO [12], LIMO [13], DVL-SLAM [14], [15]) use only 3D points from LiDAR to provide depth measurements for the visual odometry, while the advantage of accurate pose estimation in a LiDAR odometry is completely ignored. Therefore, these systems are only considered as LiDAR-assisted depth-enhanced visual odometry. In comparison, tightly integrated systems [16], [17] provide a more in-depth combination of the two techniques to take better use of their complementary advantages. The most notable work of the tightly integrated method is V-LOAM [16], in which a LiDAR-assisted depth-enhanced visual odometry runs at a high frequency, and a LiDAR odometry refines the pose from the visual module at a low frequency. However, many camera poses are not refined by the LiDAR module in V-LOAM. Furthermore, V-LOAM employs a feature-based method for its visual module which suffers inevitably from 3D-2D depth association errors as few 2D sparse feature points could have associated 3D depth points and in turn yields interpolation errors. Utilizing the direct methods for visual odometry has the potential to avoid the 3D-2D depth association error, yet methods rely on the photometric error minimization for pose estimation which are easy to fall into a local minimum and sensitive to photometric changes and errors in camera intrinsic parameters compared to feature-based methods [18].

In this paper, we present SDV-LOAM which combines a semi-direct LiDAR-assisted depth-enhanced visual odometry

and a LiDAR odometry for accurate and robust pose estimation and mapping. Different from V-LOAM, our visual module employs a semi-direct method to combine the success-factors of the feature-based method (re-projection error minimization for accurate pose estimation) with the direct method (eliminating 3D-2D depth association errors). Our semi-direct visual odometry is significantly enhanced based on the core idea of SVO [4] to address unique challenges in visual-LiDAR integration:

1) To ensure that a sufficient amount of 2D pixels have associated 3D depth points from LiDAR, we propose a new point extraction method which extracts high-gradient points that are uniformly distributed on an image and are from the projections of 3D LiDAR points on an image.

2) Due to the low frequency of 3D point cloud, the interval between frames with projected 3D LiDAR points is large which could yield large scale differences between matching frames. To detect reliable pixel correspondences between frames, we design a novel point matching method which propagates points of a host frame to an intermediate keyframe which is closer to the current frame, and then obtain the matching points between current frame and host frame by finding the corresponding pixels of the intermediate keyframe on current frame, to reduce scale differences.

3) To improve robustness of our visual module in various scenes, we adaptively add extra high-gradient points without LiDAR points as a complement when the texture of current tracking points is limited.

Another major problem of most existing LiDAR and visual-LiDAR odometry systems (e.g., A-LOAM, LeGO-LOAM, Fast-LOAM, ISC-LOAM and MULLS) is that LiDAR points tend to provide weak constraints in the vertical direction as majority of surfaces whose normal vectors pointing vertically in the outdoor scenes are the ground. These surfaces typically distribute on the same plane (i.e., ground plane) and enforce weak constraints in 3 vertical DOF (i.e., roll, pitch, upward) pose optimization via traditional Iterative Closest Point (ICP). Although the latest LiDAR odometry CT-ICP [19] ameliorates this problem by introducing a weighting strategy to favor planar neighborhoods, that was originally proposed in IMLS-SLAM [20], into the traditional ICP algorithm. However, the performance is still not well when geometric constraint in vertical direction is too limited. For these cases, optimizing the 3 vertical DOF could adversely degrade the overall accuracy of pose estimation. To address this problem, we design an adaptive sweep-to-map optimization method which automatically chooses to optimize 6 full DOF pose or only the 3 horizontal DOF according to the richness of geometric constraints in the vertical direction of the current input sweep.

In addition, we propose a sweep reconstruction module, which overlaps and reconstructs the sweep according to the time stamps of camera images to increase the input frequency of LiDAR point clouds, and in turn increase the output frequency of LiDAR odometry. Experimental results on public datasets [21], [22], [23] demonstrate that: 1) our visual module outperforms state-of-the-art LiDAR-assisted depth-enhanced VO including DEMO [12], LIMO [13], Huang et al. [15] and DVL-SLAM [14]; 2) our adaptive sweep-to-map optimization can improve the performance of several existing open-sourced

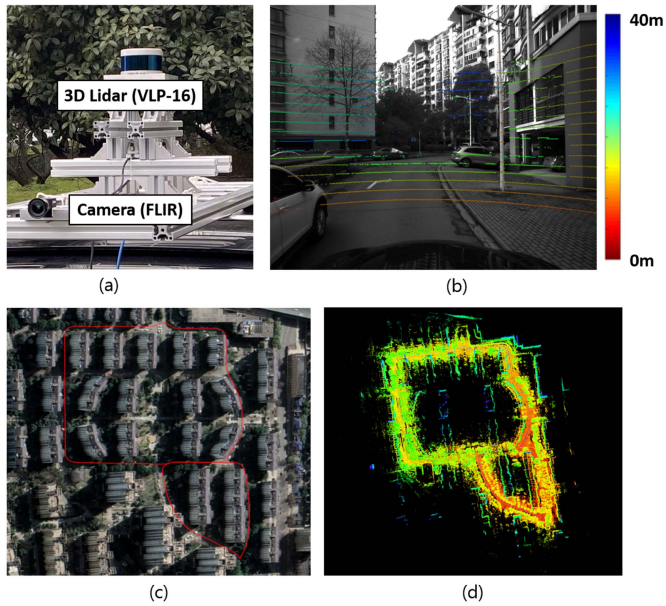


Fig. 1. (a) Our camera-LiDAR sensor system for collecting gray images and 3D point cloud measurements. (b) Exemplar input data collected by our camera-LiDAR sensor system. (c) The estimated trajectory overlaid on Google Map for visual illustration. (d) The 3D reconstruction result.

LiDAR odometry systems including A-LOAM, LeGO-LOAM [9], Fast-LOAM, ISC-LOAM [11], MULLS [24] and CT-ICP [19]; and 3) our SDV-LOAM achieves 0.47%, 0.46%, 0.038% relative translational error on the KITTI [21] training set, KITTI-360 [22], KITTI-CARLA [23] which outperforms A-LOAM, LeGO-LOAM, Fast-LOAM, ISC-LOAM, MULLS and CT-ICP, and 0.60% on the KITTI test set which ranks the 8th place on the KITTI odometry benchmark (before 2023.01), ranks 6th compared with existing LiDAR-only/monocular visual-LiDAR systems and ranks 2th compared with all open-sourced systems. We further build our visual-LiDAR odometry system with a wide-angle camera FL3-U3-13E4M-C (60 Hz) and a Velodyne VLP-16E LiDAR (10 Hz) as illustrated in Fig. 1. Live experiments in large-scale outdoor environments demonstrate that our system with sweep reconstruction can increase the frequency of input LiDAR point clouds and output poses from LiDAR module with high accuracy.

To summarize, the main contributions of this work are five folds:

1) We propose a novel visual-LiDAR odometry and mapping system, named SDV-LOAM, which achieves competitive accuracy and high output frequency in theory compared with the state-of-the-art works;

2) We propose a semi-direct method for the visual module of our SDV-LOAM, which can address unique challenges in visual-LiDAR integration (i.e., 3D-2D depth association error, long interval induced pixel matching error and poor robustness in low-textured scenes) and achieves superior performance to state-of-the-art LiDAR-assisted depth-enhanced VO [12], [13], [14], [15];

3) We propose an adaptive sweep-to-map optimization method, which automatically optimize 3 horizontal DOF pose

or 6 full DOF pose according to the geometric constraints in the scene. Our adaptive method can improve the accuracy of several existing open-sourced LiDAR odometry systems;

4) We propose a sweep reconstruction module, which can increase the input frequency of LiDAR point clouds and in turn accelerate the output pose frequency of LiDAR odometry in theory;

5) We have released the source code of this work for the development of the community.[1]

The rest of this paper is organized as follows. In Section II, we briefly discuss the relevant literature. Section III provides preliminaries. Section IV introduces the framework of our system. Sections V and VI details our visual and LiDAR module respectively followed by experimental evaluation in Section VII. Section VIII concludes the paper.

II. Related Work

RGB camera and LiDAR are two widely-used sensors for 6-DOF pose estimation. In general, vision-based odometry/SLAM systems can be categorized into three classes: 1) feature-based methods [1], [25], [26], [27], [28] which rely on local image features for pose tracking, 2) direct methods [2], [3], [6], [7], [29] which directly use raw sensor measurements (e.g., intensity values) for tracking, and 3) semi-direct method [4], which first estimates an initial pose by minimizing the photometric error as direct method, then utilizes the initial pose to find correspondences between frames via point matching, and finally refines the pose by minimizing the re-projection error of matched features as feature-based method. An obvious advantage of visual odometry(VO) is its high output frequency for pose estimation. However, its accuracy could degrade significantly in presence of large noise in image data and the obtained map is usually sparse.

LiDAR-based odometry systems [8], [9], [10], [30] rely on geometric information contained in LiDAR points for tracking, and constantly update the point cloud map with rich geometric structures. For instance, LOAM [8], [30] extracts edge and surface points from raw input point cloud, and then builds point-to-line and point-to-plane residuals for 6-DOF ICP pose estimation. LOAM performs tracking and mapping in parallel to ensure a high efficiency of pose estimation. However, due to huge number of points to be processed, the output frequency remains low. Based on LOAM, LeGO-LOAM [9] excludes a large number of points with weak geometric information from processing to significantly improve the efficiency. However, how to effectively exclude irrelevant points, e.g., outliers caused by moving objects or erroneous measurements, is a nontrivial task and incorrect removal of useful points would in turn reduce the robustness of the system. SuMa [10] proposes to represent the map via a surfel-based representation that aggregates information from LiDAR points. However, GPU acceleration is necessary for SuMa to achieve real-time performance and the pose estimation accuracy is not better than systems based on the framework of LOAM. Wang et al. [11] propose Fast-LOAM which excludes the sweep-to-map optimization module, and only retains sweep-to-sweep pose estimation. Then an intensity scan context (ISC) is proposed to improve the performance of loop detection based on Fast-LOAM. Unlike traditional LiDAR odometry systems which only extract edge and surface features, MULLS [24] proposes to utilize more types of features (e.g., facade, beam, pillar and ground) in LiDAR odometry. However, the weights of residual terms constructed by different features are very difficult to set, and using all of these features could adversely reduce the accuracy on some sequences of KITTI [21]. A major drawback commonly exists in most LIDAR systems [8], [9], [11], [24], [30] is that these methods perform 6-DOF ICP pose estimation and typically suffer large cumulative errors in the vertical direction because the vertical constraints in outdoor scenarios are usually poor. To alleviate this problem, IMLS-LOAM [20] proposes an IMLS pose solution algorithm to replace conventional transport ICP. However, large computational cost of IMLS makes IMLS-LOAM impossible to run in real time. Based on IMLS-SLAM, CT-ICP [19] employs an elastic formulation of the trajectory, with a continuity of poses intra-scan and discontinuity between scans, to improve the robustness for fast movement. Despite of higher accuracy and efficiency of CT-ICP compared with other LiDAR systems, its performance is still not well when running in scenes where geometric information in the vertical direction is limited (e.g., KITTI 01 and KITTI 03). LOL [31] applies a place recognition method to detect geometrically similar locations between the online 3D point cloud and a priori offline map via the Segment Matching algorithm. However, the prior offline map may not be available in practice and Segment Matching requires powerful GPU for computing. LIO-SAM [32] formulates LiDAR-inertial odometry as a factor graph for multi-sensor fusion and global optimization. However, the computational burden of LIO-SAM [32] has reached its limitation, therefore other sensors (e.g., camera, wheel odometer) are difficult to be fused in the existing system. In addition, they did not test their work on any published dataset, which makes it difficult to compare with the literature.

To combine the complementary advantages of visual and LiDAR systems, some studies explore visual-LiDAR hybrid odometry/SLAM by integrating the two sensors. Existing approaches can be generally categorized into two classes, i.e., loosely integrated and tightly integrated. Loosely integrated methods [12], [13], [14], [15] utilize LiDAR points to assist depth estimation and mapping, and then rely on visual tracking based on the improved map for pose estimation. For instance, DEMO [12] associates extracted 2D features with 3D points to assign depth value to each feature point (i.e., depth association). To improve the accuracy, LIMO [13] adds a loop closure module to DEMO and utilizes semantic information to identify moving objects. In addition to point features, Huang et al. [15] proposes to use line features which are less sensitive to noise [33], large viewpoint changes [34], and motion blurs [35] in tracking and mapping. However, the 3D points from LiDAR do not correspond to 2D feature points and lines one to one. Thus, interpolation is inevitably demanded in depth association and in turn introduces errors. To avoid the depth association errors, DVL-SLAM [14] proposes to use the direct method for visual tracking. In DVL-SLAM, the LiDAR points are projected on

[1]https://github.com/ZikangYuan/SDV-LOAM

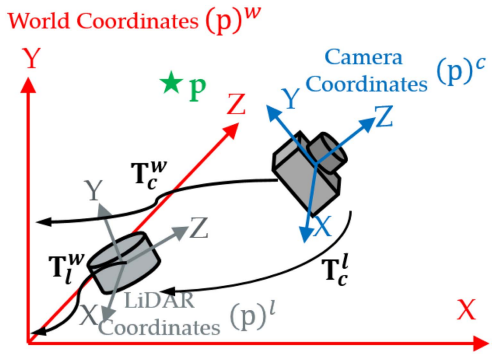


Fig. 2. Illustration of the coordinate transformation of three coordinate systems. $(\cdot)^w$, $(\cdot)^c$ and $(\cdot)^l$ are defined as a 3D point in the world coordinates, in the camera coordinates and in the LiDAR coordinates respectively. The world coordinate coincides with $(\cdot)^l$ at the starting position. In all coordinates the x-axis points to the right, the y-axis points upward, and the z-axis points forward. We assume that the transformation $\mathbf{T}_c^l$ between camera and LiDAR is known and remains constant during operation.

an image to extract identity values of located pixels. However, the pose estimation of DVL-SLAM relies on photometric error minimization which usually performs worse than re-projection error minimization.

Tightly-integrated methods [16], [17] aim at an in-depth combination of the strength of visual and LiDAR odometry. One of the most notable examples is V-LOAM [16] which first uses a LiDAR-assisted depth-enhanced visual odometry running at a high frequency (i.e., image frame rate) to perform coarse frame-to-frame pose estimation, and then refines the pose by a LiDAR odometry running at a lower frequency (i.e., LiDAR point cloud rate). Then the high-frequency camera poses are integrated into the low-frequency LiDAR poses to obtain the final pose at an image frame rate. Since many poses from the visual module are not refined by the LiDAR module, the final pose results consist of output from two systems (i.e., LiDAR odometry and LiDAR-assisted depth-enhanced odometry). It is noteworthy that this problem is not caused by the limited calculation of LiDAR odometry, but because the difference between the acquisition frequency of LiDAR point clouds (10 Hz) and an RGB camera images (30∼60 Hz).

III. Preliminaries

A. Coordinate Systems

As shown in Fig. 2, we denote $(\cdot)^w$, $(\cdot)^c$ and $(\cdot)^l$ as a 3D point in the world coordinates, in the camera coordinates and in the LiDAR coordinates respectively. We define the world coordinate $(\cdot)^w$ coincides with $(\cdot)^l$ initially. In all coordinates, the x-axis points to the right, the y-axis points upward, and the z-axis points forward. We denote the camera coordinates for taking the i_{th} image frame at time t_i as c_i and the corresponding LiDAR coordinates at t_i as l_i, then the transformation matrix (i.e., extrinsic parameters) from the camera coordinates c_i to the LiDAR coordinates l_i is denoted as $\mathbf{T}_{c_i}^{l_i} \in SE(3)$ of the special euclidean group, where $\mathbf{T}_{c_i}^{l_i}$ consists of a rotation matrix $\mathbf{R}_{c_i}^{l_i} \in SO(3)$ and a translation vector $\mathbf{t}_{c_i}^{l_i} \in \mathbb{R}^3$ as

$$\mathbf{T}_{c_i}^{l_i} = \begin{bmatrix} \mathbf{R}_{c_i}^{l_i} & \mathbf{t}_{c_i}^{l_i} \\ \mathbf{0} & 1 \end{bmatrix} \tag{1}$$

The external parameters are usually calibrated once offline and remain constant during online pose estimation; therefore, we can represent $\mathbf{T}_{c_i}^{l_i}$ using $\mathbf{T}_c^l$ for simplicity.

B. Camera Projection Model

The coordinate of a 3D point in the world coordinate is denoted as $\mathbf{p}^w = (x, y, z)^T \in \mathbb{R}^3$ and its projection to a 2D image is denoted as $\mathbf{u} = (u, v)^T \in \mathbb{R}^2$, which can be computed based on the camera projection model π: $\mathbb{R}^3 \rightarrow \mathbb{R}^2$

$$\mathbf{u} = \pi\left(\mathbf{T}_l^c \mathbf{T}_w^l \mathbf{p}^w\right) \tag{2}$$

where π is determined by the 3×3 intrinsic camera parameters $\mathbf{K}$. Similarly, the 3D points $\mathbf{p}^c$ in the camera coordinates can be recovered from their 2D projections $\mathbf{u}$ by the inverse projection model π^{-1}: π: $\mathbb{R}^2 \rightarrow \mathbb{R}^3$

$$\mathbf{p}^c = \pi^{-1}\left(\mathbf{u}, d_{\mathbf{u}}\right) \tag{3}$$

where $d_{\mathbf{u}} \in \mathbb{R}$ represents the depth of 2D point $\mathbf{u}$ in the camera coordinates.

C. Data Management

We define a sweep as a full scan completed by a LiDAR at one time. To build our sensor system, we utilize Velodyne VLP-16 as our LiDAR sensor which produces a full 360° point cloud by rotation of LiDAR diodes (10 Hz), and FL3-FW-14S3M-C as our camera sensor (60 Hz). As the two sensors run at different frequency, we align their timestamps. Assume the camera collects an image at time t_i, the LiDAR begins a scan at t_i-0.1 s and finishes the scan at time t_i. The point cloud collected from a full scan is defined as S in the following text. For evaluation on the public datasets [21], [22], [23], the acquisition frequency of the camera and LiDAR are both 10 Hz, and the providers have aligned the timestamps and transformations between the two sensors, and [21] has performed motion distortion compensation for all sequences.

In the visual module of our system, we define a frame as a set consisting of an image and the associated LiDAR points. As the acquisition frequency of the camera (e.g., 60 Hz) could be higher than that of the LiDAR (e.g., 10 Hz), only a subset of images has corresponding LiDAR points. For images without corresponding LiDAR points, the frame contains only an image and an empty set of LiDAR points. Only frames with nonempty LiDAR points could be selected as keyframes (KFs).

For 2D pixels in an image, on which the corresponding LiDAR points can be directly projected, we define them as *depth point* D in the following text. For other 2D pixels without corresponding 3D points, we define them as *non-depth point*. For a depth point whose pixel coordinate is $\mathbf{u} = (u, v)^T \in \mathbb{R}^2$ associated with a LiDAR point $\mathbf{p}^l = (x, y, z)^T \in \mathbb{R}^3$, the depth $d_{\mathbf{u}}$ of $\mathbf{u}$ is calculated as

$$d_{\mathbf{u}} = \left[\mathbf{T}_l^c \mathbf{p}^l\right]_z \tag{4}$$

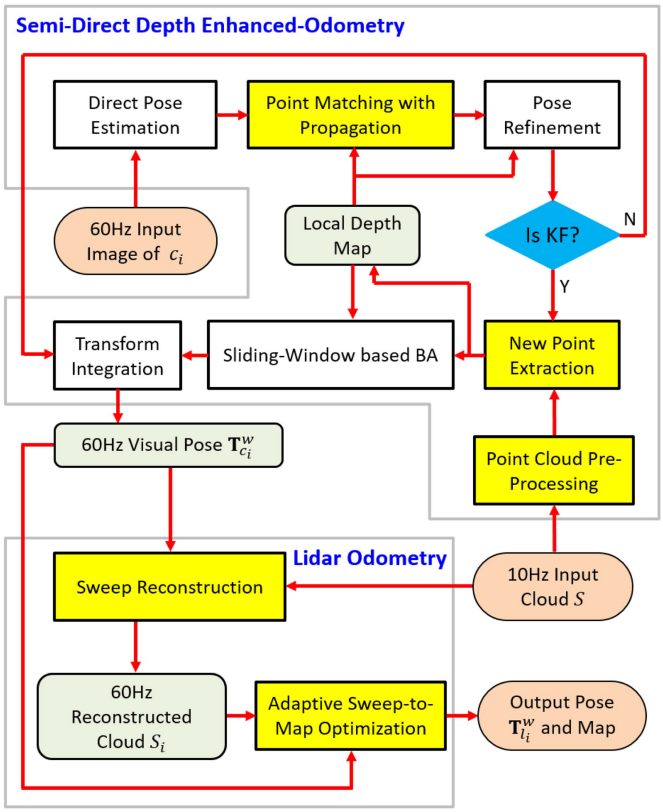


Fig. 3. Overview of our SDV-LOAM which consists of two main modules: a semi-direct depth-enhanced visual odometry and a LiDAR odometry. The visual module and LiDAR module are combined by a sweep reconstruction block, which increases the input frequency of LiDAR point clouds to the same frequency as the camera images. The yellow blocks highlight our main contributions compared with existing methods. It is worth explaining that the "60 Hz visual pose" and the "60 Hz reconstructed cloud" are the ideal frequency assuming infinite computing resources. We use 60 Hz visual pose and reconstructed cloud here for the convenience of description. In fact, both our vision and LiDAR module can run at around 20 Hz.

where $[\cdot]_z$ is the z coordinate of the element. For simplicity, in (4) we do not consider motion distortion of LiDAR points, which can be compensated by the constant velocity model [8].

IV. System Overview

Fig. 3 illustrates the framework of our system which consists of two main modules: a semi-direct depth-enhanced visual odometry and a LiDAR odometry. The yellow blocks highlight our main contributions compared with existing methods.

Our visual module is implemented based on DSO [6], a popular monocular direct-based VO, with significant modifications to extend it to a semi-direct depth-enhanced VO with LiDAR assistance. *Local depth map* is the local map that stores the depth observation of each extracted point whose host frame is in the sliding window, and the *host frame* is the frame from which the high-gradient point is extracted. The semi-direct depth-enhanced visual odometry runs at 60 Hz to estimate pose of each camera frame c_i. First, an initial pose of c_i is estimated by minimizing the photometric error between c_i and the latest keyframe k_n in the direct pose estimation block. Then the initial pose and the local depth map is utilized to find pixel correspondences between frames via point matching with propagation. Finally, we utilize the pixel correspondences to further refine the pose by minimizing the re-projection error between c_i and the local depth map in the pose refinement block. If c_i is not selected as a new keyframe, it is sent to the transform integration block. Otherwise, c_i is sent to the new point extraction block. For every new keyframe k_n, the high gradient depth and non-depth points are extracted. Then, in the sliding window-based bundle adjustment (BA) block, we find pixel correspondences between newly extracted points and pixels on previous keyframes in the sliding window. The re-projection residuals are computed according to all existing correspondences in the sliding window and jointly optimized. After each optimization, the oldest keyframe is marginalized and then dropped from the sliding window and points extracted from this keyframe are removed from the local depth map. Finally, we combine the frame-to-frame pose from pose refinement block and the frame-to-world pose from sliding window-based BA block in the transform integration block to derive the final output pose at the same frequency with input camera images. The point cloud pre-processing block is designed to adaptively add extra high-gradient points without LiDAR depth measurements as a complement when the texture of current tracking points is limited.

Our LiDAR module is implemented based on CT-ICP [19], with two main enhancements to improve the accuracy and output frequency. First, we propose a sweep reconstruction method which overlaps and reconstructs a sweep to match the acquisition frequency of camera images (e.g., 60 Hz). Second, we present an adaptive sweep-to-map pose optimization method which adaptively determine to optimize 3 horizontal DOF or 6 full DOF pose according to the richness of geometric information in the vertical direction.

V. Semi-Direct Depth-Enhanced VO

Existing methods [12], [13], [15] mainly utilize feature-based method for LiDAR-assisted visual tracking. However, the projection of a 3D LiDAR point may not coincide exactly with a 2D feature point. As a result, interpolation of 3D points is usually required in the process of associating 2D features with 3D LiDAR points, yielding nontrivial approximation errors. On the other hand, direct method [14] can effectively avoid this problem as it can directly utilize 2D pixels where the 3D LiDAR points projected for visual tracking. However, the pose solution method of photometric error minimization (utilized in direct methods) is prone to fall into local minimum and sensitive to photometric changes and errors in camera intrinsic parameters [18].

In this work, we propose to use semi-direct method for tracking, which combines the success-factors of feature-based methods (re-projection error minimization for accurate pose estimation) with direct methods (eliminating the error in the process of 2D-3D data association caused by sparse feature extraction module when integrating LiDAR information into a VO). We also design a series of improvement solutions (i.e., new point extraction, point matching with propagation and adaptively

adding extra non-depth points) to solve technological difficulties retained in semi-direct LiDAR-assisted depth-enhanced VO.

Our vision module only stores the local depth map, which consists of all extracted depth and non-depth points whose host frames are in the sliding window. Any point $\mathbf{p}$ in the local map is stored as $(\mathbf{u}, d_{\mathbf{u}})$, where $\mathbf{u}$ is the 2D pixel coordinate of $\mathbf{p}$ on its host frame and $d_{\mathbf{u}}$ is the depth value of pixel $\mathbf{u}$. If a frame is removed from the sliding window, the points extracted from it are also removed from the local map. As the visual module targets to output high-frequency pose estimations, it does not store a global map.

A. Point Cloud Pre-Processing

Depth values calculated from LiDAR points according to (4) are of high precision, and remain unchanged throughout the operation of our system. If majority of LiDAR points can be projected to regions with rich visual textures, the extracted high gradient depth points from the new point extraction block are sufficient for pose tracking. However, it is common that some regions of a scene are untextured or highly similar (e.g., ground and leaves). If most depth points located in these untextured regions, estimating pose only based on the depth points could become inaccurate. In this case, we demand high gradient non-depth points in textured regions as a supplement to assist tracking. Although depth values of non-depth points are obtained by depth filtering [6], which are not as precise as that of depth points, they still contain sample effective visual information and thus are useful for improving tracking accuracy. We define utilizing only depth points as *mode 1*, and utilizing both depth and non-depth points as *mode 2*.

The function of point cloud pre-processing is to analyze the input point cloud and adaptively determine the mode. The point cloud pre-processing module is executed for every frame with nonempty LiDAR point clouds. Once the mode is determined, it stays the same until the point cloud pre-processing module is performed again.

The point cloud pre-processing module performs as follows. We first use a ground segmentation method [36] to roughly detect candidate ground points based on raw input points, where the ground is usually an untextured region. Then, we utilize the candidate ground points to fit an accurate ground by RANSAC based on which we can obtain refined ground points of the current frame. We calculate the percentage of ground points in the current sweep. If the percentage is higher than a threshold (e.g., 0.8 in our system), we utilize *mode 2* for visual module. Otherwise, we use a fast segmentation method [37] to further distinguish points with irregular structural information. The projections of irregular structures in outdoor scenes usually locate on leaves or grasses, whose intensities are very similar to surrounding pixels and are indistinctive for visual pose estimation. If the percentage of irregular points of non-ground points is higher than a threshold (e.g., 0.5 in our system), we utilize *mode 2* for visual module. Otherwise, we utilize *mode 1*.

B. Direct Pose Estimation

We track the current frame c_i with respect to the newest keyframe k_n by projecting extracted high gradient points of k_n (i.e., P_{k_n}) to c_i, and then calculate the photometric error between k_n and c_i as

$$E_{k_n c_i} = \sum_{\mathbf{u}\in P_{k_n}} \sum_{\widetilde{\mathbf{u}}\in N_{\mathbf{u}}} \omega_{\widetilde{\mathbf{u}}} \left\| I_{c_i}[\widetilde{\mathbf{u}}'] - b_{c_i} - \frac{e^{a_{c_i}}}{e^{a_{k_n}}} \left(I_{k_n}[\widetilde{\mathbf{u}}] - b_{k_n}\right) \right\|_{\gamma}^2 \tag{5}$$

where $\|\cdot\|_{\gamma}$ is the Huber norm, $I(\cdot)$ represents pixel intensity and $\omega_{\mathbf{u}}$ is a weight that down-weights high image gradient pixels as

$$\omega_{\mathbf{u}} = \frac{con^2}{con^2 + \|\nabla I_{c_i}(\mathbf{u})\|_2^2} \tag{6}$$

con is a constant (e.g., 50 in our system), $\mathbf{u}$ is the 2D pixel of an extracted high gradient depth point in P_{k_n}. Even if *mode 2* is used according to the result of point cloud pre-processing block, the depths of those non-depth points of k_n do not converge. Therefore, only high gradient depth points in P_{k_n} are used for tracking. $\mathbf{u}'$ is the projection of $\mathbf{u}$ in c_i calculated as

$$\mathbf{u}' = \pi\left(\mathbf{T}_{k_n}^{c_i}\, \pi^{-1}\left(\mathbf{u}, d_{\mathbf{u}}\right)\right) \tag{7}$$

where $d_{\mathbf{u}}$ is the depth of $\mathbf{u}$, which is calculated according to (4). In (5), $a_{c_\cdot}$, $b_{c_\cdot}$ are the hardware related parameters affecting the illumination transfer function in the optimization process. $N_{\mathbf{u}}$ denotes the 8-point pattern of $\mathbf{u}$, $\widetilde{\mathbf{u}}'$ is the projection of the pattern point $\widetilde{\mathbf{u}}$ into I_{c_i}. Finally, $\mathbf{T}_{k_n}^{c_i}$, a_{c_i} and b_{c_i} are estimated by minimizing $E_{k_n c_i}$.

C. Point-Matching With Propagation

Once the initial pose $\mathbf{T}_{k_n}^{c_i}$ of current frame c_i is estimated, we transform it to the pose relative to $(\cdot)^{c_0}$ by $\mathbf{T}_{c_i}^{c_0} = \mathbf{T}_{k_n}^{c_0}\, \mathbf{T}_{k_n}^{c_i\ -1}$ and then utilize $\mathbf{T}_{c_i}^{c_0}$ to find correspondences between the local depth map and c_i, where c_0 is the first frame in the camera coordinate system.

The authors of SVO [4] minimize the photometric error between patches (denoted by black squares in Fig. 4) to find 2D pixel correspondences between current frame c_i and the closest keyframe k_n. Specially, for an arbitrary point $\mathbf{p}_1 = (\mathbf{u}_1, d_{\mathbf{u}_1}) \in P_{k_j}$ whose host frame is $k_j (j \in [1, n-1])$ in the sliding window, SVO first takes the correspondence of $\mathbf{u}_1$ on k_n (i.e., $\mathbf{u}_1^{k_n}$), which is the closest keyframe to c_i. Then $(\mathbf{u}_1^{k_n}, d_{\mathbf{u}_1^{k_n}})$ is projected to c_i to derive $\mathbf{u}_1^{c_i}$ via $\mathbf{T}_{c_0}^{c_i}$ to find the initial position of 2D pixel correspondence of $\mathbf{p}_1$ on c_i, then optimizes the position of $\mathbf{u}_1^{c_i}$ by minimizing (9) within the neighboring regions of $\mathbf{u}_1^{c_i}$ and $\mathbf{u}_1^{k_n}$ (i.e., point matching)

$$\mathbf{u}_1^{c_i} = \pi\left(\mathbf{T}_{c_0}^{c_i}\left(\mathbf{T}_{k_n}^{c_0}\, \pi^{-1}\left(\mathbf{u}_1^{k_n}, d_{\mathbf{u}_1^{k_n}}\right)\right)\right) \tag{8}$$

$$\mathbf{u}_1^{c_i} = \underset{\mathbf{u}_1^{c_i}}{\operatorname{argmin}} \sum_{\overline{\mathbf{u}}\in N_{\mathbf{u}_1^{c_i}}, \widetilde{\mathbf{u}}\in N_{\mathbf{u}_1^{k_n}}} \left\| I_{c_i}[\overline{\mathbf{u}}] - \mathbf{A}_1 I_{k_n}[\widetilde{\mathbf{u}}] \right\|^2 \tag{9}$$

where $\mathbf{A}_1$ is the affine warping matrix which is detailed in [4].

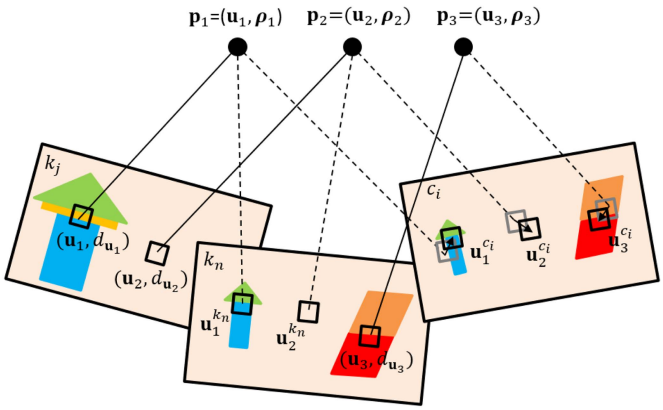


Fig. 4. Illustration of point matching with propagation. We first finds correspondences between a host keyframe k_j and the current frame c_i by minimizing photometric error between 2D patches (black squares) in c_i and k_j. For an arbitrary 3D point $\mathbf{p}_1$, we project it from its host frame k_j to c_i to derive $\mathbf{u}_1^{c_i}$, then we take $\mathbf{p}_1$'s previously matched correspondence $\mathbf{u}_1^{k_n}$ from k_n that is closest to c_i, and align $\mathbf{u}_1^{k_n}$ and $\mathbf{u}_1^{c_i}$ which have similar scale to refine the position of $\mathbf{u}_1^{c_i}$.

Different from semi-direct methods using stereo and RGB-D cameras, not all extracted points have depth observations from LiDAR points. This phenomenon causes that a correspondence found in current frame cannot be used to find new correspondence in the next frame. As shown in Fig. 4, $\mathbf{u}_1$ is an extracted point from k_j and its depth value $d_{\mathbf{u}_1}$ is calculated from the 3D LiDAR point projected on $\mathbf{u}_1$ according to (4). $\mathbf{u}_1^{k_n}$ is the correspondence found by point matching on k_n. However, in LiDAR-assisted VO, it is hard to ensure there is exactly one LiDAR point projected onto $\mathbf{u}_1^{k_n}$. As a result, we may not be able to provide depth observation $d_{\mathbf{u}_1^{k_n}}$ for $\mathbf{u}_1^{k_n}$ in k_n and in turn cannot find the correspondence of $\mathbf{u}_1$ on c_i using the method in SVO even if k_n is close to c_i.

A logical solution to the above problem is to perform point matching between frames with intervals. For instance, we can directly find the correspondence of $\mathbf{u}_1$ on current frame c_i. However, this point matching method has a key limitation: the accuracy of resulted pixel correspondences greatly depends on the scale difference of two image frames I_{c_i} and I_{k_j}. As the interval between two frames increases, the scale difference could increase greatly, and in turn yield large matching errors. As shown in Fig. 4, for an exemplar point $\mathbf{p}_1 = (\mathbf{u}_1, d_{\mathbf{u}_1})$ whose host frame k_j is far from the current frame c_i, the neighborhood of $\mathbf{u}_1$ could be quite different from the neighborhood of $\mathbf{u}_1^{c_i}$ as some regions exist in the neighborhood of $\mathbf{u}_1$ could disappear (i.e., orange rectangle in k_j) due to reduced resolution of I_{c_i}. Consequently, the matching error could be large for $\mathbf{u}_1^{c_i}$. We also quantitatively test the influence of intervals between frames on the matching error in Section VII-A1. Results show that as the interval increases from 1 keyframes to 7 keyframes, the error of pixel correspondences increases from 2.86 pixels to 10.40 pixels.

To address the above problem, we propose our point matching with propagation which propagates points with depth observations from LiDAR to an intermediate keyframe for reducing intervals. Specifically, for the exemplar point, e.g., $\mathbf{p}_1 = (\mathbf{u}_1, d_{\mathbf{u}_1})$ whose host frame is k_j, we first utilize (10) to project it to obtain an initial position $\mathbf{u}_1^{c_i}$. Then, we take $\mathbf{p}_1$'s previously matched correspondence $\mathbf{u}_1^{k_n}$ from k_n that is closest to c_i, and optimize the coordinate of $\mathbf{u}_1^{c_i}$ by aligning $\mathbf{u}_1^{k_n}$ and $\mathbf{u}_1^{c_i}$ with consideration of the influence of photometric parameters

$$\mathbf{u}_1^{c_i} = \pi\left(\mathbf{T}_{c_0}^{c_i}\left(\mathbf{T}_{k_j}^{c_0}\pi^{-1}\left(\mathbf{u}_1, d_{\mathbf{u}_1}\right)\right)\right) \tag{10}$$

$$\mathbf{u}_1^{c_i} = \underset{\mathbf{u}_1^{c_i}}{\operatorname{argmin}} \sum_{\overline{\mathbf{u}}\in N_{\mathbf{u}_1^{c_i}}, \widetilde{\mathbf{u}}\in N_{\mathbf{u}_1^{k_n}}} \left\| I_{c_i}[\overline{\mathbf{u}}] - b_{c_i} - \frac{e^{a_{c_i}}}{e^{a_{k_n}}}\left(\mathbf{A}_1 I_{k_n}[\widetilde{\mathbf{u}}] - b_{k_n}\right)\right\|^2 \tag{11}$$

Instead of aligning $\mathbf{u}_1$ and $\mathbf{u}_1^{c_i}$, we utilize $\mathbf{u}_1^{k_n}$ as an intermediate when performing point matching because the host frame of $\mathbf{u}_1^{k_n}$ (i.e., k_n) and current frame c_i are sufficiently close so their local resolutions are similar. As shown in Fig. 4, the neighborhood of $\mathbf{u}_1^{k_n}$ is more similar as that of $\mathbf{u}_1^{c_i}$ than that of $\mathbf{u}_1$. Therefore, we can obtain more accurate pixel correspondence of $\mathbf{u}_1$ on $\mathbf{c}_i$ by aligning $\mathbf{u}_1^{c_i}$ with $\mathbf{u}_1^{k_n}$. Naturally, after c_i has been selected as a new keyframe k_{n+1}, when the next camera frame c_{i+1} arrives, we can find the correspondence of $\mathbf{p}_1$ on c_{i+1} by aligning $\mathbf{u}_1^{k_{n+1}}$ (i.e., $\mathbf{u}_1^{c_i}$) and $\mathbf{u}_1^{c_{i+1}}$.

D. Pose Refinement

After obtaining correspondences between the local depth map and c_i, we build the re-projection error to further refine $\mathbf{T}_{c_i}^{c_0}$

$$\mathbf{T}_{c_i}^{c_0} = \underset{\mathbf{T}_{c_i}^{c_0}}{\operatorname{argmin}} \sum_{k_m\in K}\sum_{\mathbf{p}_s\in P_{k_m}} \left\|\pi\left(\mathbf{T}_{c_0}^{c_i}\mathbf{T}_{k_m}^{c_0}\mathbf{p}_s\right) - \mathbf{u}_s^{c_i}\right\|_\gamma^2 \tag{12}$$

where $\mathbf{u}_s^{c_i}$ is the correspondence of $\mathbf{p}_s$ on c_i, K is the set of all keyframes in the sliding window and P_{k_m} is the set of extracted points from k_m. If c_i is selected as the newest keyframe, it is sent to the new point extraction block, otherwise the pose of c_i is transformed to frame-to-frame pose by $\mathbf{T}_{c_i}^{k_n} = \mathbf{T}_{c_0}^{k_n}\mathbf{T}_{c_i}^{c_0}$ and then sent to the transform integration block.

E. New Point Extraction

Our new point extraction module is composed of two parts: new depth point extraction and new non-depth point extraction.

If *mode 1* is selected by the point cloud pre-processing module, only new depth points are extracted in this module. In SVO, 2D features are extracted for each newly selected keyframe. However, approximate interpolation is inevitable when associating 2D features with sparse 3D LiDAR points to obtain depth observations. To avoid the depth association error, we choose to extract high gradient points with depth observations instead of 2D features. First, we project the point cloud of k_n onto the image according to (4) to obtain the corresponding depth point set, which is defined as D_{k_n}. Then, we extract a subset P_{k_n} from D_{k_n}, which consists of uniformly distributed high-gradient image points. *Pseudo Code* V-E summarizes the process of our new depth point extraction which is mainly divided into two steps. In the first step, we divide an image equally into blocks of size 32×32, $\{T_1, T_2 \cdots T_q\}$, then we calculate the corresponding

Pseudo Code 1: New Depth Point Extraction.

Input: the image of the newest keyframe k_n and its depth point set D_{k_n}
Output: the set of extracted points P_{k_n}
Procedure1: Calculate the Intensity Threshold
  Divide k_n into 32×32 blocks $\{T_1, T_2 \cdots T_q\}$
  for $T_i = T_1 \cdots T_q$
    Calculate a gradient threshold $th(T_i)$ based on the intensity histogram of T_i
Procedure2: Adaptive Point Extraction
  Divide k_n into $n \times n$ blocks $\{m_1, m_2 \cdots m_l\}$ corresponding the sub-set of D_{k_n} : $\{S_1, S_2 \cdots S_l\}$
  for $m_i = m_1 \cdots m_l$
    Extract the highest gradient point $\mathbf{p}_h = (\mathbf{u}_h, d_{\mathbf{u}_h})$ from S_i and find T_p where $\mathbf{u}_h$ locates
    If $(grad(\mathbf{p}_h) > th(T_p))$
      Add $\mathbf{p}_h$ into P_{k_n}, $num++$
  If (num does not meet the requirement)
    Clear P_{k_n}, $num = 0$
    Adjust n and do **Procedure2** again
  Else
    Return P_{k_n}

gradient threshold $th(T_i)$ for each block T_i according to the intensity histogram of T_i. In the second step, we uniformly divide the image into $n \times n$ blocks $\{m_1, m_2 \cdots m_l\}$, where n is inversely proportional to the expected number of points extracted on each image. The initial value of n is calculated by $n = \sqrt{\frac{w*h}{num\_expect}}$, where w and h are the width and height of input image respectively, and num_expect is the number of points we want to extract from each frame. For each block m_i, we extract all points locating in it and belonging to D_{k_n} to form a subset S_i. Next, we calculate gradients for all points in S_i, and we define the point that corresponds to the maximum gradient as $\mathbf{p}_h = (\mathbf{u}_h, d_{\mathbf{u}_h})$. If the gradient of $\mathbf{p}_h$ is higher than $th(T_p)$ (T_p is the 32×32 block where $\mathbf{u}_h$ locates), we add $\mathbf{p}_h$ into P_{k_n} as a newly extracted high gradient point. If the number of points in P_{k_n} does not meet our requirement, we adjust the value of n appropriately, and repeat the second step.

If *mode 2* is selected by the point cloud pre-processing module, which means most 3D LiDAR points locate on untextured area. In this case, we extract additional high gradient but non-depth points and add them into P_{k_n}. The non-depth points dominate the convergence of sliding window optimization. Meanwhile, the depth points in P_{k_n} are mainly used to maintain an accurate metric scale.

F. Sliding Window-Based BA

The sliding window-based bundle adjustment (BA) performs multi-frame joint optimization by minimizing the re-projection error to improve accuracy and consistency. For depth points whose depth values are obtained from LiDAR measurements, we do not optimize their depth values in BA. For non-depth points whose depth values are estimated by the depth filter, we optimized their depth values with the pose of keyframe.

For an arbitrary point $\mathbf{p}_1 = (\mathbf{u}_1, d_{\mathbf{u}_1}) \in P_{k_h}$ whose host frame is k_h ($h \in [1, n-1]$), we can find the matching point of $\mathbf{p}_1$ (e.g., $\mathbf{u}_1^{k_j}$) on other keyframe (e.g., k_j) in the sliding window by our point matching with propagation. However, having matching pairs does not mean that the bidirectional re-projection errors can be built as the depth observation of $\mathbf{u}_1^{k_j}$ could be unavailable if no LiDAR point is projected onto $\mathbf{u}_1^{k_j}$. Specially, the re-projection residual of matching pair $(\mathbf{u}_1, \mathbf{u}_1^{k_j})$ from k_j to k_h is written as

$$\mathbf{r}_{k_h k_j}^{\mathbf{p}_1} = \pi \left(\mathbf{T}_{c_0}^{k_h} \mathbf{T}_{k_j}^{c_0} \pi^{-1} \left(\mathbf{u}_1^{k_j}, d_{\mathbf{u}_1}^{k_j} \right) \right) - \mathbf{u}_1 \tag{13}$$

where the depth value $d_{\mathbf{u}_1}^{k_j}$ is necessary for building a residual yet it is very likely that no LiDAR point is projected onto $\mathbf{u}_1^{k_j}$ and in turn make $d_{\mathbf{u}_1}^{k_j}$ unavailable. Generating $d_{\mathbf{u}_1}^{k_j}$ via interpolation could yield nontrivial errors. Therefore, we can always build the re-projection error for the matching pair $(\mathbf{u}_i, \mathbf{u}_i^{k_j})$ from k_h to k_j, but may not have the re-projection error for the pair k_j to k_h. This limitation makes the re-projection error can only be built from old keyframes to new keyframes, and in turn leads to non-equal status of keyframes in the sliding window and degrades the effectiveness of multi-frame joint optimization.

Our proposed back-projection approach can effectively address the above problem without incurring any depth interpolation errors. Specifically, when the newest keyframe k_n arrives, we find 2D correspondences of P_{k_n} on $k_1 \sim k_{n-1}$ by the method mentioned in Section V-C. To avoid the long interval between matched points, we first find 2D correspondence of $\mathbf{p}_i \in P_{k_n}$ on k_{n-1}. Then we match $\mathbf{u}_i^{k_{n-2}}$ with $\mathbf{u}_i^{k_{n-1}}$ when finding the 2D correspondence of $\mathbf{p}_i$ on k_{n-2}. The entire process is executed recursively until the correspondence of $\mathbf{p}_i$ is found on k_1. After point back-matching for each keyframe k_n we find the correspondence of its extracted points (i.e., P_{k_n}) on other keyframes (i.e., $k_j \in K$) in the sliding window. Then we perform the sliding window-based BA to optimize $\boldsymbol{\chi} = \{\mathbf{T}_{k_1}^{c_0}, \mathbf{T}_{k_2}^{c_0} \ldots \mathbf{T}_{k_n}^{c_0}\}$ and $\boldsymbol{\phi} = \{d_{\mathbf{u}_1}, d_{\mathbf{u}_2} \cdots d_{\mathbf{u}_m}\}$, where $\boldsymbol{\phi}$ is the set of depth value of all non-depth points in sliding window

$$\boldsymbol{\chi}, \boldsymbol{\phi} = \underset{\boldsymbol{\chi}, \boldsymbol{\phi}}{\operatorname{argmin}} \sum_{k_h \in K} \sum_{\mathbf{p}_i \in P_{k_h}} \sum_{k_j \in K} \left\| \mathbf{r}_{k_j k_h}^{\mathbf{p}_i} \right\|_{\gamma}^2 \quad (j \neq h) \tag{14}$$

$\mathbf{r}_{k_j k_h}^{\mathbf{p}_i}$ is a single re-projection residual determined by three elements: the host frame k_h, the point $\mathbf{p}_i = (\mathbf{u}_i, d_{\mathbf{u}_i})$ and the 2D correspondence of $\mathbf{p}_i$ on the target frame k_j (i.e., $\mathbf{u}_i^{k_j}$). Eq 14 aims to jointly minimize all re-projection residuals in the sliding window.

Similar as [6], we utilize marginalization to alleviate the computational burden of windowed optimization while retaining previous information.

G. Transform Integration

For a keyframe k_j, we directly use the $\mathbf{T}_{k_j}^{c_0}$ output by the sliding window-based BA module as its final pose. However, for a non-keyframe c_i, we integrate the frame-to-keyframe pose

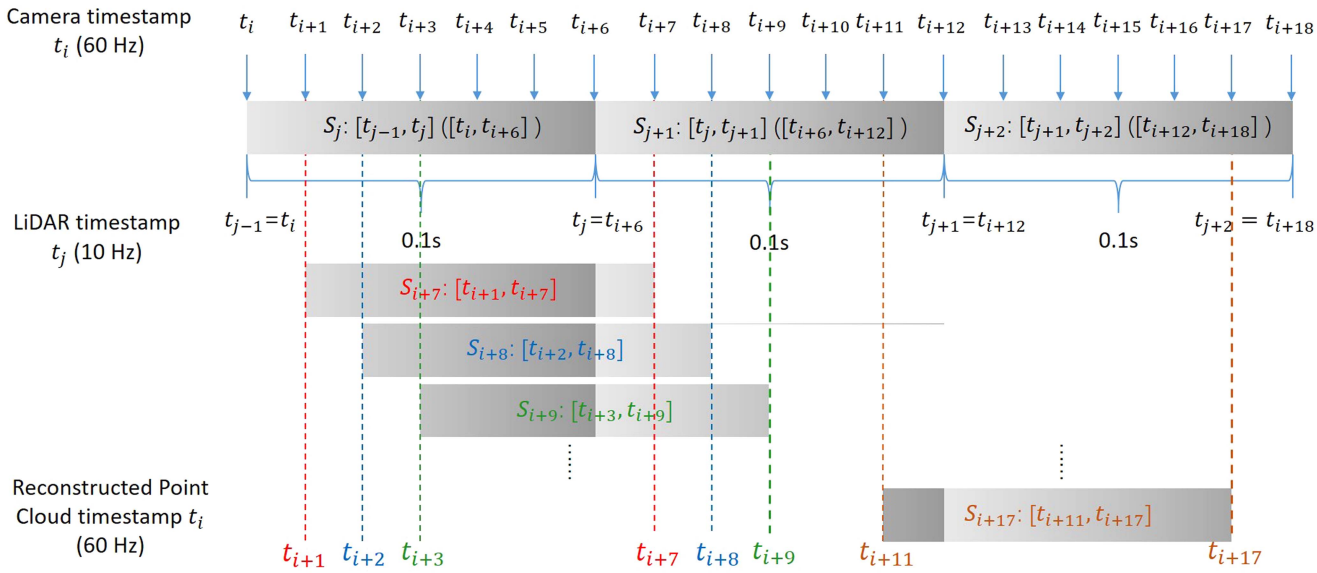


Fig. 5. Illustration of our sweep reconstruction algorithm. 10 Hz input raw point cloud S_j, S_{j+1} and S_{j+2} are overlapped and reconstructed to obtain 60 Hz reconstructed point clouds S_{i+7}, S_{i+8}, . . ., S_{i+17}.

between two frames from the pose refinement block and the keyframe-to-map pose from the sliding window-based BA block to obtain its final output pose $\mathbf{T}_{c_i}^{c_0}$ by $\mathbf{T}_{c_i}^{c_0} = \mathbf{T}_{k_n}^{c_0}\mathbf{T}_{c_i}^{k_n}$, where $\mathbf{T}_{k_n}^{c_0}$ is the pose of the closest keyframe k_n before c_i. Then $\mathbf{T}_{c_i}^{c_0}$ is converted to $\mathbf{T}_{l_i}^{w}$ by $\mathbf{T}_{l_i}^{w} = \mathbf{T}_{c}^{l}\mathbf{T}_{c_i}^{c_0}\mathbf{T}_{l}^{c}$ before being sent to LiDAR odometry module.

VI. LiDAR Odometry

One problem of combining output poses from the visual and the LiDAR modules is that the input frequency of image and LiDAR points are different. Existing methods [16], [17] integrates the high-frequency camera poses with the low-frequency LiDAR poses to obtain the final output poses. In this paper we choose to overlap and reconstruct sweeps according to the time stamps of camera images to ensure every output pose from visual module can be refined by the LiDAR odometry.

In addition, directly utilizing LiDAR points for tracking could suffer large drifts in the vertical direction in scenarios with limited vertical geometric constraints. To alleviate this problem, we propose an adaptive sweep-to-map optimization, which can automatically judge whether to optimize the 3 vertical DOF (i.e., roll, pitch, upward) or to optimize 6-DOF poses according to the amount of geometric constraints in the vertical direction of the current input sweep.

The LiDAR module contains both the local map and the global map. The global map is stored as a voxel map, and the local map is part of the whole voxel map. Every time we register the point cloud into the local map, the global map is updated accordingly. We adopt the same sampling strategy as CT-ICP to sample the point cloud of the current sweep before registering point cloud to the local map. When registering the sampled point cloud of the current sweep into the local map, we utilize the refined pose of the current sweep to transform each point to the world coordinates, and find the voxel it belongs to. To update the global/local voxel grid, we adopt the procedure of CT-ICP. That is, if there are already 20 points in a voxel, the point would not be added to the voxel. Otherwise, the point is added to the voxel.

A. Sweep Reconstruction

Sweep reconstruction aims to derive 60 Hz reconstructed point cloud S_i from the 10 Hz original input point cloud S. Fig. 5 illustrates the process of our sweep reconstruction. Assuming for a LiDAR sweep $S_j : [t_{j-1}, t_j]$, which begins at t_{j-1} and finishes at t_j, there are 6 camera images captured during $[t_{j-1}, t_j]$. The 6 camera images are captured at t_{i+1}, t_{i+2}, t_{i+3}, t_{i+4}, t_{i+5} and t_{i+6} respectively ($t_i = t_{j-1}, t_{i+6} = t_j$). Only the camera image captured at t_{i+6} has the corresponding LiDAR points.

Based on the characteristics of continuous acquisition over a period of time of LiDAR, we denote the points captured from t_{i+1} to t_{i+7} as a reconstructed sweep $S_{i+7} : [t_{i+1}, t_{i+7}]$, which corresponds to the image captured at t_{i+7}. Similarly, we can also obtain reconstructed sweep $S_{i+8} : [t_{i+2}, t_{i+8}]$, $S_{i+9} : [t_{i+3}, t_{i+9}]$, . . ., $S_{i+17} : [t_{i+11}, t_{i+17}]$, which correspond to camera image captured at t_{i+8}, t_{i+9}, . . ., t_{i+17} respectively. By this way, we obtain reconstructed sweeps which have the same frequency as the input camera images.

It is worth mentioning that we would not register each reconstructed sweep S_i in the mapping process, but only added points to the final output map at 10 Hz to avoid adding points repeatedly.

Compared with transform intergration (utilized in V-LOAM) which also aims to obtain the high-frequency final output pose, our sweep reconstruction can better maintain the consistency of estimated trajectory. In the following, we provide detailed explanation of the inconsistency problem in transform integration.

For a camera frame captured at t_i, we denote the nearest sweep before t_i is finished at t_n and the sweep-to-map pose at t_n is $\mathbf{T}_{l_n}^{w}$. Transform integration estimates the final pose at t_i (e.g., $\mathbf{T}_{l_i}^{w}$) as

$$\mathbf{T}_{l_i}^{w} = \mathbf{T}_{l_n}^{w}\mathbf{T}_{l}^{c-1}\mathbf{T}_{c_i}^{c_n}\mathbf{T}_{l}^{c} \tag{15}$$

where $\mathbf{T}_{c_i}^{c_n}$ is the pose from VO, $\mathbf{T}_{l}^{c}$ is the external parameters between the LiDAR and the camera. In practice, $\mathbf{T}_{l}^{c}$ is usually calibrated once offline and remain constant during online pose estimation. However, accurate calibration of external parameters is nontrivial, and the widely-used calibration method [38] still contain obvious errors. In addition, the external parameters usually change online due to vibration during the movement of the vehicle platform. Once the $\mathbf{T}_{l_i}^{w}$ is calculated by (15), transformation integration does not perform any optimization for $\mathbf{T}_{l_i}^{w}$, but directly output it as the final pose at time t_i. As a result, the above-mentioned errors in $\mathbf{T}_{l}^{c}$ are not reduced and in turn result in the inconsistency of estimated trajectory, which is illustrated in Section VII-B3.

Compared with transform integration, our sweep reconstruction can increase the frequency of input LiDAR sweeps and thus every output pose from VO can be optimized by the LiDAR module. Specially, for an exemplar camera pose at t_i (e.g., $\mathbf{T}_{c_i}^{c_0}$), we first transform it from the camera coordinates to the LiDAR coordinates as

$$\mathbf{T}_{l_i}^{w} = \mathbf{T}_{l}^{c^{-1}}\mathbf{T}_{c_i}^{c_0}\mathbf{T}_{l}^{c} \tag{16}$$

Then $\mathbf{T}_{l_i}^{w}$ is sent to the adaptive sweep-to-map optimization module and the error from $\mathbf{T}_{l}^{c}$ can be eliminated during optimization. Therefore, our sweep reconstruction can achieve a high accuracy and efficiency for pose estimation and meanwhile maintain a good local consistency of the estimated trajectory.

B. Adaptive Sweep-to-Map Optimization

We design our pose solution in our LiDAR module based on the method in CT-ICP [19] with an enhancement of adaptive sweep-to-map optimization. That is, we automatically determine whether to optimize the 3 horizontal DOF (i.e., yaw, forward, right) or 6 full DOF poses according to the amount of geometric constraints in the vertical direction of a reconstructed sweep S_i. Specifically, we examine whether or not the surfaces pointing to the vertical direction are evenly distributed on the same plane. If almost all point-to-vertical surfaces are on the same plane (i.e., usually is ground plane), the residuals provided by them are hard to constrain the vertical direction well and in this case we only optimize 3 horizontal DOF poses. The evaluation of pose error in vertical direction is detailed in Section VII-A1.

When each reconstructed sweep S_i arrives, we utilized the method introduced in Section V-A to detect the ground and calculate the normal vector of ground n_G. When building the point-to-plane residuals for pose estimation, we find the surface ε around each target point $\mathbf{p}$ and calculate the normal n of ε. If the direction of n and n_G are very similar, we consider $\mathbf{p}$ as a ground candidate point. We count the number of candidate points N in S_i, and the number of points M which locate on the ground. We calculate the percentage of ground surfaces to vertical surfaces by M/N. If M/N is higher than a preset threshold (0.8 in our system), the vertical geometric constraint is regarded as poor. In this case, we only optimize 3 horizontal DOF (i.e., yaw, forward, right) for the current reconstructed sweep S_i. Otherwise, we optimize 6 full DOF as processed in CT-ICP.

We define S_i as the point cloud of the currently reconstructed sweep in the coordinate system $(\cdot)^{l_i}$, t_b and t_e as the begin time and end time of S_i respectively. For each S_i, we first extract a set of points P_i from S_i via the down-sampling strategy in CT-ICP. For an arbitrary point $\mathbf{p}^l \in P_i$ which is collected at time t_p, we project $\mathbf{p}^l$ from $(\cdot)^{l_i}$ to $(\cdot)^{w}$ via the initial pose of S_i obtained from the vision module to obtain $\mathbf{p}^w$. Then, we find the 20 nearest points around $\mathbf{p}^w$ from the local map to fit a plane and calculate the normal vector $\mathbf{n}$ of the plane. Accordingly, we can build the point-to-plane residual $r^{\mathbf{p}^l}$ for $\mathbf{p}^l$ as follow:

$$r^{\mathbf{p}^l} = \omega_p \mathbf{n}^T\left(\mathbf{p}^w - \mathbf{q}\right) \tag{17}$$

$$\mathbf{p}^w = \mathbf{R}_{t_p}^{w}\mathbf{p}^l + \mathbf{t}_{t_p}^{w} \tag{18}$$

$$\mathbf{R}_{t_p}^{w} = slerp\left(\mathbf{R}_b, \mathbf{R}_e, \alpha_p\right) \tag{19}$$

$$\mathbf{t}_{t_p}^{w} = \left(1-\alpha_p\right)\mathbf{t}_b + \alpha_p \mathbf{t}_e \tag{20}$$

$$\alpha_p = \frac{t_p - t_b}{t_e - t_b} \tag{21}$$

where $\mathbf{q}$ is the closest point to $\mathbf{p}^w$, ω_p is defined by [20], $\mathbf{R}_b$, $\mathbf{R}_e$ are the rotation matrices with respect to $(\cdot)^w$ at t_b and t_e respectively, $\mathbf{t}_b$ and $\mathbf{t}_e$ are the translation vectors with respect to $(\cdot)^w$ at t_b and t_e respectively. Both $\mathbf{R}_b, \mathbf{R}_e, \mathbf{t}_b, \mathbf{t}_e$ are variables to be refined in our adaptive sweep-to-map module, and the initial value of $\mathbf{R}_e$, $\mathbf{t}_e$ are obtained from the vision module.

In order to better maintain the consistency of the trajectory, we additionally optimize the poses within the time interval (t_b, t_e). We assume there are k reconstructed sweeps acquired at time $t_j (0 \le j \le k-1)$ and $t_j \in (t_b, t_e)$. For point $\mathbf{p}^l$ which is being processed and obtained in the sub time interval (t_j, t_{j+1}), we optimize the corresponding poses (i.e., $\mathbf{R}_j$, $\mathbf{R}_{j+1}$, $\mathbf{t}_j$, $\mathbf{t}_{j+1}$) in the current optimization round. Specially, we build an additional point-to-plane residual $r_a^{\mathbf{p}^l}$ which is similar as $r^{\mathbf{p}^l}$ in (17)

$$r_a^{\mathbf{p}^l} = \omega_p \mathbf{n}^T\left(\mathbf{p}^w - \mathbf{q}\right) \tag{22}$$

$$\mathbf{p}^w = \mathbf{R}_{t_p}^{w}\mathbf{p}^l + \mathbf{t}_{t_p}^{w} \tag{23}$$

$$\mathbf{R}_{t_p}^{w} = slerp\left(\mathbf{R}_j, \mathbf{R}_{j+1}, \alpha_p\right) \tag{24}$$

$$\mathbf{t}_{t_p}^{w} = \left(1-\alpha_p\right)\mathbf{t}_j + \alpha_p \mathbf{t}_{j+1} \tag{25}$$

$$\alpha_p = \frac{t_p - t_j}{t_{j+1} - t_j} \tag{26}$$

where $r_a^{\mathbf{p}^l}$ is an additional residual which is related to variables $\mathbf{R}_j$, $\mathbf{R}_{j+1}$, $\mathbf{t}_j$, $\mathbf{t}_{j+1}$. By this way, for an arbitrary point $\mathbf{p}^l$, we can build two residuals (e.g., $r^{\mathbf{p}^l}$ and $r_a^{\mathbf{p}^l}$) to optimize eight variables, i.e., $\mathbf{R}_b$, $\mathbf{R}_e$, $\mathbf{R}_j$, $\mathbf{R}_{j+1}$, $\mathbf{t}_b$, $\mathbf{t}_e$, $\mathbf{t}_j$, $\mathbf{t}_{j+1}$.

To facilitate 3 horizontal DOF optimization, we represent rotation as

$$\mathbf{R} = \mathbf{R}_y\mathbf{R}_x\mathbf{R}_z \tag{27}$$

TABLE I
ABLATION STUDY OF OUR VISUAL ODOMETRY

Dataset	Seq no.	Ours w/o Pro	Ours w/o Bi-Dir	Ours w/o Ada	Our VO module
KITTI	00	0.69	0.70	0.81	**0.67**
	01	0.98	1.09	2.92	**0.96**
	02	0.80	0.76	0.90	**0.75**
	03	0.86	0.84	**0.80**	0.86
	04	**0.75**	**0.75**	1.30	0.77
	05	0.68	0.67	0.86	**0.66**
	06	0.46	0.58	0.48	**0.44**
	07	0.82	0.81	0.88	**0.74**
	08	1.11	1.10	1.19	**1.07**
	09	0.55	0.60	0.55	**0.53**
	10	**0.50**	0.60	0.57	0.51
	00-10 avg	0.75	0.77	1.02	**0.72**
	11-21 mean	-	-	-	**0.88**
KITTI-360	00	1.03	1.11	1.32	**0.81**
	02	0.93	0.81	1.11	**0.79**
	03	**0.83**	1.41	1.22	0.90
	04	1.29	1.69	1.60	**1.14**
	05	4.95	2.40	2.94	**1.86**
	06	1.22	1.05	1.47	**0.94**
	07	1.33	10.26	1.72	**1.31**
	09	1.21	1.15	1.43	**1.03**
	10	2.26	4.00	3.07	**1.89**
	avg	1.67	2.65	1.76	**1.18**
KITTI-CARLA	01	2.59	0.45	0.64	**0.32**
	02	2.63	1.27	1.01	**0.61**
	07	1.88	0.69	0.97	**0.38**
	avg	2.37	0.80	0.87	**0.44**

Denotations: All errors are represented as **RTE**[%] (the smaller the better). **Bold** fonts denote the first place. w/o: without, Pro: propagation, Bi-Dir: bi-directional point matching, Ada: adaptive tracking module selection, -: not available.

TABLE II
PIXEL ERROR OF CORRESPONDENCES AT DIFFERENT FRAME INTERVALS ON KITTI 00-10

Interval (keyframe)	1	2	3	4	5	6	7
w/p Pro (pixel)	2.86	4.87	6.55	7.93	8.86	9.04	10.40
w Pro (pixel)	1.85	3.15	4.28	5.26	6.04	6.39	6.77

TABLE III
RELATIVE TRANSLATIONAL ERROR OF FAST-LOAM AND FAST-LOAM++ IN X-Y-Z DIRECTION [%]

Seq no.	Fast-LOAM			Fast-LOAM++		
	X	Y	Z	X	Y	Z
00	0.77	29.41	0.50	0.83	9.74	0.69
01	0.88	138.10	0.56	0.77	19.20	1.28
02	0.95	46.96	0.31	1.29	11.66	0.58
03	0.51	10.77	1.16	0.71	2.56	2.64
04	28.59	70.87	0.11	39.05	10.51	0.52
05	0.60	25.79	0.78	1.23	7.04	1.40
06	4.64	31.81	0.18	8.38	7.99	0.32
07	0.41	23.43	0.36	0.74	10.37	1.15
08	1.27	21.24	1.41	1.88	13.58	1.73
09	0.52	32.05	0.36	0.81	10.26	0.39
10	0.76	28.77	0.66	0.38	4.43	0.49

Fig. 6. Sensor configuration for experiments on our own platform. The sensor system is mounted on a car-like vehicle for outdoor experiments.

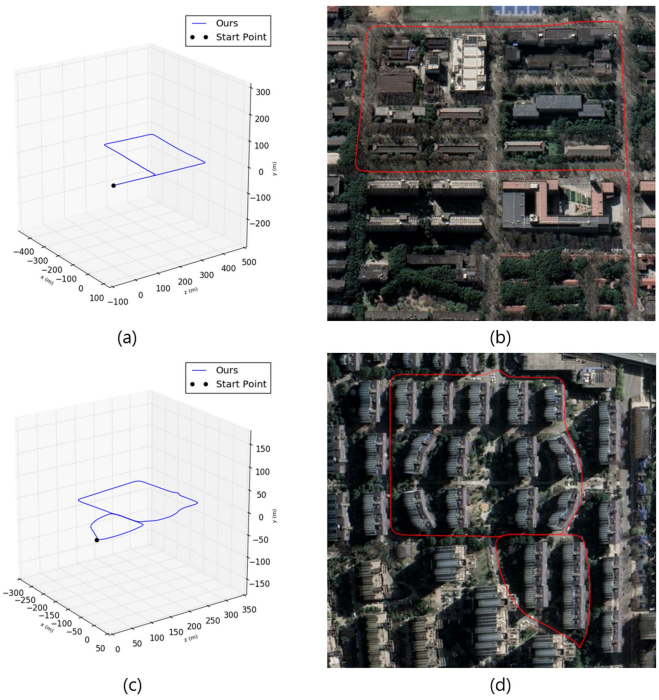


Fig. 7. The results of our outdoor experiments. (a) and (c) are the trajectory estimated by our system on two different vehicle routes. We overlaid (a) and (c) with Google Map to obtain (b) and (d) for better evaluation.

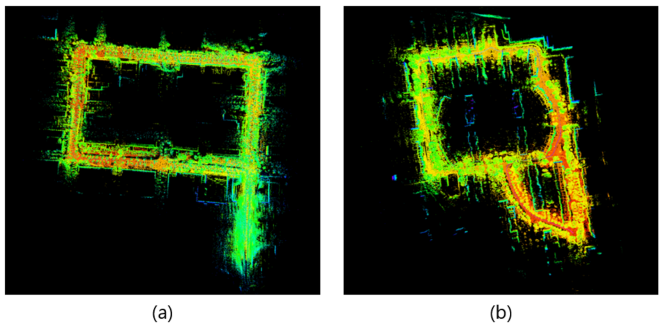


Fig. 8. The resulting global point cloud map of trajectory Fig. 7(a) and (c).

TABLE IV
ABLATION STUDY OF OUR LIDAR ODOMETRY

Dataset	Seq no.	A-LOAM	A-LOAM+	A-LOAM++	LeGO-LOAM	LeGO-LOAM+	LeGO-LOAM++	Fast-LOAM	Fast-LOAM+	Fast-LOAM++
KITTI	00	0.96	0.88	**0.77**	5.03	1.25	**0.92**	0.96	0.76	**0.62**
	01	2.68	2.65	**1.91**	21.02	3.35	**1.27**	2.80	1.11	**1.00**
	02	5.04	1.52	**1.47**	2.80	1.74	**0.83**	1.57	1.09	**0.84**
	03	1.16	1.15	**1.03**	1.32	1.12	**0.85**	1.09	1.67	**0.94**
	04	1.39	1.36	**0.77**	1.63	1.73	**0.61**	1.43	0.77	**0.61**
	05	0.71	0.68	**0.65**	0.93	0.87	**0.51**	0.80	0.70	**0.57**
	06	0.72	0.72	**0.55**	0.84	0.85	**0.44**	0.72	0.48	**0.47**
	07	0.54	0.53	**0.49**	0.77	0.66	**0.48**	**0.55**	1.04	0.76
	08	1.18	1.18	**0.98**	1.58	1.35	**0.94**	1.16	1.17	**1.08**
	09	1.21	1.19	**0.88**	1.48	1.75	**0.61**	1.29	0.74	**0.69**
	10	1.61	1.57	**1.43**	1.86	1.96	**1.09**	1.77	0.86	**0.49**
	00-10 avg	1.56	1.22	**0.99**	3.57	1.51	**0.78**	1.34	0.94	**0.76**
	11-21 mean	-	-	-	-	-	-	-	-	-
KITTI-360	00	1.03	1.00	**0.95**	1.60	1.64	**1.25**	1.17	0.74	**0.60**
	02	1.66	0.83	**0.76**	1.89	1.42	**1.19**	0.93	0.89	**0.79**
	03	1.86	1.81	**1.78**	2.44	2.50	**1.29**	1.88	0.79	**0.62**
	04	1.60	1.33	**1.27**	1.86	1.68	**1.61**	1.38	1.16	**1.03**
	05	1.01	**1.00**	1.01	1.60	1.36	**1.33**	**1.00**	1.17	1.08
	06	1.11	1.13	**1.02**	1.76	1.45	**1.36**	1.23	1.12	**0.94**
	07	4.09	1.72	**1.68**	3.24	3.08	**2.47**	1.69	1.38	**1.20**
	09	1.13	1.05	**0.97**	1.63	1.68	**1.44**	1.21	0.81	**0.68**
	10	**1.74**	2.03	1.88	2.94	2.40	**1.28**	1.84	1.37	**1.31**
	avg	1.69	1.32	**1.25**	2.10	1.91	**1.47**	1.37	1.05	**0.91**
KITTI-CARLA	01	4.15	0.17	**0.13**	0.45	0.78	**0.41**	0.12	0.11	**0.10**
	02	0.12	**0.10**	**0.10**	2.07	0.88	**0.63**	0.08	0.09	**0.07**
	07	3.23	0.62	**0.20**	2.49	1.25	**0.90**	1.64	0.40	**0.38**
	avg	2.50	0.30	**0.14**	1.67	0.97	**0.65**	0.61	0.20	**0.18**
Dataset	Seq no.	ISC-LOAM	ISC-LOAM+	ISC-LOAM++	MULLS	MULLS+	MULLS++	CT-ICP	Ct-ICP+	CT-ICP++
KITTI	00	1.21	1.03	**0.84**	0.53	**0.52**	**0.52**	**0.49**	**0.49**	0.50
	01	2.80	1.11	**1.00**	0.72	0.67	**0.58**	0.73	0.69	**0.62**
	02	1.72	1.23	**0.84**	0.59	0.59	**0.57**	0.52	0.52	**0.51**
	03	1.09	1.67	**0.94**	**0.60**	0.62	**0.60**	0.71	0.70	**0.56**
	04	1.43	0.77	**0.61**	0.48	0.46	**0.42**	0.37	**0.36**	0.37
	05	0.84	0.98	**0.62**	**0.31**	**0.31**	**0.31**	0.26	**0.25**	0.27
	06	0.69	0.48	**0.47**	0.29	0.29	**0.26**	**0.28**	**0.28**	**0.28**
	07	**0.55**	1.04	0.76	**0.33**	0.34	**0.33**	**0.32**	**0.32**	**0.32**
	08	1.16	1.17	**1.08**	0.82	**0.81**	**0.81**	0.81	0.81	**0.79**
	09	1.29	0.74	**0.69**	0.55	0.51	**0.47**	0.49	0.48	**0.47**
	10	1.77	0.86	**0.49**	0.64	0.64	**0.52**	0.49	0.49	**0.47**
	00-10 avg	1.32	1.01	**0.76**	0.53	0.52	**0.49**	0.50	0.49	**0.47**
	11-21 mean	-	-	-	0.92	-	-	0.62	-	**0.60**
KITTI-360	00	-	-	-	1.62	1.58	**1.14**	0.42	**0.40**	**0.40**
	02	-	-	-	1.31	1.33	**1.26**	0.32	0.32	**0.31**
	03	-	-	-	1.04	1.02	**0.74**	0.37	0.37	**0.35**
	04	-	-	-	1.69	1.65	**1.52**	0.68	0.67	**0.66**
	05	-	-	-	1.31	1.29	**1.21**	0.39	0.38	**0.37**
	06	-	-	-	1.81	1.80	**1.41**	0.43	0.42	**0.40**
	07	-	-	-	5.72	5.24	**1.31**	0.54	0.53	**0.52**
	09	-	-	-	1.28	1.28	**1.16**	0.43	0.43	**0.42**
	10	-	-	-	0.93	0.92	**0.83**	0.73	0.71	**0.70**
	avg	-	-	-	1.85	1.79	**1.17**	0.48	0.47	**0.46**
KITTI-CARLA	01	-	-	-	0.56	0.54	**0.53**	0.038	0.038	**0.035**
	02	-	-	-	0.20	0.23	**0.15**	0.040	0.039	**0.038**
	07	-	-	-	1.04	1.01	**0.39**	0.044	0.045	**0.040**
	avg	-	-	-	0.60	0.59	**0.36**	0.041	0.041	**0.038**

Denotations: All errors are represented as **RTE**[%] (the smaller the better). **Bold** fonts denote the first place. +: using the estimated pose of our visual module as initial value of LiDAR module, ++: using the adaptive sweep-to-map optimization strategy on the basis of "+", -: not available, the data on the right of "/" are results recorded in the original paper, and the data on the left of "/" are the results we measured using the open source code provided by the authors.

TABLE V
RELATIVE TRANSLATIONAL ERROR COMPARISON OF LIDAR-ASSISTED DEPTH-ENHANCED VO ON KITTI ODOMETRY

Seq no.	DEMO	LIMO*	Huang et al.	DVL-SLAM	Our VO module
00	1.05	1.12	0.99	0.93	**0.67**
01	1.87	**0.91**	1.87	1.47	0.96
02	0.93	-	1.38	1.11	**0.75**
03	0.99	-	**0.65**	0.92	0.86
04	1.23	0.53	**0.42**	0.67	0.77
05	1.04	-	0.72	0.82	**0.66**
06	0.96	-	0.61	0.92	**0.44**
07	1.16	-	**0.56**	1.26	0.74
08	1.24	-	1.27	1.32	**1.07**
09	1.17	-	1.06	0.66	**0.53**
10	1.14	-	0.83	0.70	**0.51**
00-10 avg	1.16	0.85	0.94	1.98	**0.72**
11-21 mean	1.14	0.93	-	-	**0.88**

Denotations: All errors are represented as **RTE**[%] (the smaller the better). **Bold** fonts denote the first place. U: urban road, H: highway, C: country road, *: with semantic information, -: not available.

where $\mathbf{R}_x$, $\mathbf{R}_y$, $\mathbf{R}_z$ are reconstructed from the Euler angles r_x, r_y, r_z in the Y-X-Z sequence rule. Accordingly, the final variables that need to be solved are

$$\boldsymbol{x} = [r_x, r_y, r_z, t_x, t_y, t_z]^T \tag{28}$$

$$\boldsymbol{\chi} = \left[\boldsymbol{x}^b, \boldsymbol{x}^e, \boldsymbol{x}^0 \ldots, \boldsymbol{x}^{k-1}\right] \tag{29}$$

where $\mathbf{t} = [t_x, t_y, t_z]^T$. Integrating all the point-to-plane residuals into one optimization function, we can obtain

$$\boldsymbol{\chi} = \underset{\boldsymbol{\chi}}{\operatorname{argmin}} \sum_{\mathbf{p}^l \in P_i} \rho\left(r^{\mathbf{p}^l} + r_a^{\mathbf{p}^l}\right) \tag{30}$$

where $\rho(\cdot)$ is a robust loss function to minimize the influence of outlier residuals. If the vertical geometric constraint is regarded as poor, we only optimize 3 horizontal DOF (i.e., yaw, forward, right) for the currently reconstructed sweep, while the variables that need to be solved are

$$\boldsymbol{x}_3 = [r_y, t_x, t_z]^T \tag{31}$$

$$\boldsymbol{\chi}_{3-DOF} = \left[\boldsymbol{x}_3^b, \boldsymbol{x}_3^e, \boldsymbol{x}_3^0, \ldots, \boldsymbol{x}_3^{k-1}\right] \tag{32}$$

and the optimization function is

$$\boldsymbol{\chi}_{3-DOF} = \underset{\boldsymbol{\chi}_{3-DOF}}{\operatorname{argmin}} \sum_{\mathbf{p}^l \in P_i} \rho\left(r^{\mathbf{p}^l} + r_a^{\mathbf{p}^l}\right) \tag{33}$$

If the vertical geometric constraint is sufficient, we set (29) as the variables and perform 6 full DOF according to (30), which is same as CT-ICP [19]. For both 3-DOF and 6-DOF optimization, we utilize the local consistency constraint and the constant velocity constraint as CT-ICP in our system.

After the pose of the currently reconstructed sweep has been estimated, we register the points in P_i into the local map based on the methods described in Section VI.

VII. EXPERIMENTS

We evaluated our SDV-LOAM on both the public datasets [21], [22], [23] and our own camera-LiDAR sensor hardware system. A consumer-level laptop equipped with an Intel Core i7-11700 and 16 GB RAM is used for all experiments.

A. Evaluation on Public Datasets

We evaluate our SDV-LOAM on three public datasets: KITTI [21], KITTI-360 [22], and KITTI-CARLA [23]. Both KITTI and KITTI-360 are logged with sensors mounted on top of a passenger vehicle in road driving scenarios. The vehicle of KITTI is equipped with a color stereo camera, a monochrome stereo camera, a Velodyne HDL-64E laser scanner, and a high accuracy GPS/INS for collecting the ground truth. The vehicle of KITTI-360 is equipped with a perspective stereo camera, two fisheye cameras, a Velodyne HDL-64E laser scanner and a high accuracy GPS/INS for collecting the ground truth. KITTI-CARLA is a virtual dataset generated by the CARLA simulator [39], where the data is generated by a simulated 64-channel spinning LiDAR sensor and a simulated stereo camera. For those three datasets, we only take the LiDAR point cloud and the left image of the stereo camera as input. Both LiDAR point clouds and camera images of KITTI, KITTI-360 and KITTI-CARLA are collected at 10 Hz. Motion distortions of the point cloud data provided by KITTI have been eliminated, and the motion distortions of the point cloud in KITTI-CARLA are compensated by us using the high-frequency ground truth. Therefore, we did not handle the distortion problem when evaluating on KITTI and KITTI-CARLA.

The KITTI Odometry benchmark contains 11 sequences with the GPS/INS ground truth provided (i.e., training set) and 11 sequences without the ground truth (i.e., testing set). The maximum driving speed in the training set reaches 85 km/h (23.6 m/s). The data covers mainly three types of environments: "urban" with buildings around, "country" on small roads with vegetations in the scene, and "highway" where roads are wide and the vehicle speed is fast. We evaluate each module of our system on the KITTI training set. For the testing set, we publish the results on the official website.[2] KITTI-360 [22] contains 8 sequences that have similar scenes as KITTI yet are much longer than KITTI in length. KITTI-CARLA [23] consists of 7 sequences and each sequence contains 5000 stereo images and sweeps. We only test sequence 01, 02 and 07 of KITTI-CARLA in this work because all other sequences are untextured scenes on which our VO fails. For all three datasets, we utilize the universal evaluation metrics - KITTI Relative Translational Error (RTE) as the evaluation metrics. As our SDV-LOAM consists of a visual odometry and a LiDAR odometry, we perform the ablation study for each odometry in Section VII-A1, followed by comparison of our VO and SDV-LOAM with the state-of-the-art methods in Secs. VII-A2 and VII-A3.

1) Ablation Study: Ablation Study of Our Visual Odometry. We examine the effectiveness of the three proposed components in our VO by excluding one of them from our visual module. In Table I, Ours w/o Pro, Our w/o Bi-Dir and Ours w/o Ada denotes our VO without point matching with propagation, bi-directional point matching and adaptive tracking respectively.

[2]http://www.cvlibs.net/datasets/kitti/eval_odometry.php

TABLE VI
RELATIVE TRANSLATIONAL ERROR COMPARISON OF VISUAL-LIDAR ODOMETRY

Dataset	Seq no.	A-LOAM+	LeGO-LOAM+	Fast-LOAM+	ISC-LOAM+	MULLS+	CT-ICP+	V-LOAM	Ours
KITTI	00	0.88	1.25	0.76	1.03	0.52	**0.49**	-	0.50
	01	2.65	3.35	1.11	1.11	0.67	0.69	-	**0.62**
	02	1.52	1.74	1.09	1.23	0.59	0.52	-	**0.51**
	03	1.15	1.12	1.67	1.67	0.62	0.70	-	**0.56**
	04	1.36	1.73	0.77	0.77	0.46	**0.36**	-	0.37
	05	0.68	0.87	0.70	0.98	0.31	**0.25**	-	0.27
	06	0.72	0.85	0.48	0.48	0.29	**0.28**	-	**0.28**
	07	0.53	0.66	1.04	1.04	0.34	**0.32**	-	**0.32**
	08	1.18	1.35	1.17	1.17	0.81	0.81	-	**0.79**
	09	1.19	1.75	0.74	0.74	0.51	0.48	-	**0.47**
	10	1.57	1.96	0.86	0.86	0.64	0.49	-	**0.47**
	00-10 avg	1.22	1.51	0.94	1.01	0.52	0.49	-	**0.47**
	11-21 mean	-	-	-	-	-	-	**0.54**	0.60
KITTI-360	00	1.00	1.64	0.74	-	1.58	**0.40**	-	**0.40**
	02	0.83	1.42	0.89	-	1.33	0.32	-	**0.31**
	03	1.81	2.50	0.79	-	1.02	0.37	-	**0.35**
	04	1.33	1.68	1.16	-	1.65	0.67	-	**0.66**
	05	1.00	1.36	1.17	-	1.29	0.38	-	**0.37**
	06	1.13	1.45	1.12	-	1.80	0.42	-	**0.40**
	07	1.72	3.08	1.38	-	5.24	0.53	-	**0.52**
	09	1.05	1.68	0.81	-	1.28	0.43	-	**0.42**
	10	2.03	2.40	1.37	-	0.92	0.71	-	**0.70**
	avg	1.32	1.91	1.05	-	1.79	0.47	-	**0.46**
KITTI-CARLA	01	0.17	0.78	0.11	-	0.54	0.038	-	**0.035**
	02	0.10	0.88	0.09	-	0.23	0.039	-	**0.038**
	07	0.62	1.25	0.40	-	1.01	0.045	-	**0.040**
	avg	0.30	0.97	0.20	-	0.59	0.041	-	**0.038**

Denotations: All errors are represented as **RTE**[%] (the smaller the better). **Bold** fonts denote the first place. -: not available.

First, comparing Ours w/o Pro and Our VO module we observe that our point matching with propagation can reduce the RTE of most sequences. We further show the matching errors (i.e., distance between a true matching pixel and the matched one) at different frame intervals. Results in Table II show that without our point matching method, the matching error increases sharply as the interval between frames increases. In comparison, our point matching method can greatly decrease matching error for different frame intervals. Second, comparing Ours w/o Bi-Dir and Our VO module in Table I, we observe that our bi-directional point matching method can reduce the RTE by $2.47\sim 87.23\%$ and $28.89\sim 51.97\%$ for all sequences of KITTI-360 and KITTI-CARLA respectively, and $1.32\sim 24.14\%$ for the KITTI sequences 00-10 except for the sequence 04. Third, from the results of Ours w/o Ada and Our VO module we observe that our adaptive tracking mode selection method can reduce the tracking error by $23.84\sim 38.64\%$ for KITTI-360, $39.60\sim 60.82\%$ for KITTI-CARLA and $3.64\sim 67.12\%$ for the training sequences 00-10 of KITTI except for sequence 03.

Ablation Study of Our LiDAR Odometry. We analyze the effectiveness of our adaptive sweep-to-map optimization in the LiDAR odometry. To this end, we propose the directional pose error which is calculated by averaging the relative errors using segmented trajectories at $100\ \mathrm{m}, 200\ \mathrm{m}, \ldots, 800\ \mathrm{m}$, based on 3D coordinates in each direction, i.e., X (right), Y (upward) and Z (forward). Specially, for each length $\Delta \in \{100\ \mathrm{m}, 200\ \mathrm{m}, \ldots, 800\ \mathrm{m}\}$, we define the estimated pose at the beginning of Δ as $\widehat{\mathbf{T}}_b$ and the estimated pose at the end of Δ as $\widehat{\mathbf{T}}_e$. Similarly, we define the ground truth pose at the beginning of Δ as $\mathbf{T}_b$ and at the end of Δ as $\mathbf{T}_e$. Then, the relative pose from the end of Δ to the beginning of Δ is calculated by: $\Delta\widehat{\mathbf{T}} = \widehat{\mathbf{T}}_b^{-1}\widehat{\mathbf{T}}_e$ and $\Delta\mathbf{T} = {\mathbf{T}_b}^{-1}\mathbf{T}_e$. Accordingly, the pose error is computed as $\Delta\widehat{\mathbf{T}}^{-1}\Delta\mathbf{T}$. We also count the total distance traveled in X-Y-Z during Δ to obtain s_x, s_y and s_z. Finally, the error in X-Y-Z during Δ is calculated by

$$\begin{aligned} e_x &= \left[\Delta\widehat{\mathbf{T}}^{-1}\Delta\mathbf{T}\right]_x / s_x * 100\% \\ e_y &= \left[\Delta\widehat{\mathbf{T}}^{-1}\Delta\mathbf{T}\right]_y / s_y * 100\% \\ e_z &= \left[\Delta\widehat{\mathbf{T}}^{-1}\Delta\mathbf{T}\right]_z / s_z * 100\% \end{aligned} \tag{34}$$

where $[\cdot]_x$, $[\cdot]_y$ and $[\cdot]_z$ denote the translation in direction X, Y and Z of the element. Table III shows the translation errors of Fast-LOAM and Fast-LOAM++ in the X, Y and Z directions respectively on the KITTI sequences 00-10, where "++" means the result with our adaptive sweep-to-map optimization and with the initial pose from our visual module. Obviously, the translational error in the Y direction is obviously higher than that in X and Z.

We further apply our adaptive sweep-to-map optimization to six state-of-the-art open-sourced methods, including A-LOAM, LeGO-LOAM [9], Fast-LOAM, ISC-LOAM [11], MULLS [24] and CT-ICP [19]. A-LOAM is an engineering optimized version based on LOAM. Both Fast-LOAM and LeGO-LOAM are accelerated versions of LOAM. Based on Fast-LOAM, ISC-LOAM proposes a novel global descriptor, intensity scan context (ISC), that explores both geometry and intensity characteristics, to improve the efficiency for loop closure detection. MULLS proposes to utilize more types of features (e.g., facade, beam, pillar and ground) in LiDAR odometry instead of traditional edge and surface features. CT-ICP uses an elastic formulation

of the trajectory, with a continuity of poses intra-scan and discontinuity between scans, to be more robust to high frequency in the movements of the sensor.

For each method, we test three results on each sequence of KITTI, KITTI-360 and KITTI-CARLA, i.e., the estimated poses by the pure LiDAR system, by the LiDAR system without adaptive sweep-to-map optimization and with the assistance of our visual module (denoted using the suffix "+" in Table IV), and by the LiDAR system with adaptive sweep-to-map optimization and with the assistance of our visual module (denoted using the suffix "++" in Table IV). Almost all systems do not release the final version of the source code that could reproduce the accuracy recorded in their papers. For a fair comparison, in the ablation study we obtain the results based on the source code provided by the authors.

Results in Tables IV show that our proposed adaptive sweep-to-map optimization can significantly improve the accuracy of all six LiDAR odometry on almost all sequences of the KITTI, KITTI-360 and KITTI-CARLA, demonstrating the effectiveness of our proposed method. It is noteworthy that we did not provide the results of ISC-LOAM on KITTI-360 and KITTI-CARLA because the sequences of KITTI-360 and KITTI-CARLA are too long for ISC-LOAM to perform global graph optimization properly.

2) Performance of Our Visual Odometry: We compare our visual odometry with four state-of-the-art LiDAR-assisted depth-enhanced visual odometry systems, i.e., DEMO [12], LIMO [13], Huang et al. [15] and DVL-SLAM [14]. The former two are popular feature-based VO systems and LIMO utilizes semantic information to identify moving objects and includes loop closure. Huang et al. utilizes both point and line features and DVL-SLAM employs the direct based method. The average RTE of DEMO, Huang et al. and DVL-SLAM on sequence 00-10 are directly referred from [12], [15] and [14] respectively, and the results of LIMO on sequence 00-10 are referred from the results tested in [15]. Since the semantic label data required by LIMO is available for sequences 00,01,04, only results of these sequences are recorded. The mean RTE on sequence 11–21 are recorded from the original DEMO [12] and LIMO paper [13], while the other methods do not evaluate on sequence 11–21. In addition, we utilize a unified parameter configuration for all sequences in the entire test process.

Results in Table V demonstrate that our visual module outperforms DEMO for all sequences and outperform LIMO, Huang et al. and DVL-SLAM for most sequences on KITTI, where the RTE of our method is 37.9% lower than DEMO, 15.3% lower than LIMO, 23.4% lower than Huang et al. 26.5% lower than DVL-SLAM on the average results of sequence 00-10, and 22.8% lower than DEMO, 5.4% lower than LIMO on the mean results of sequence 11–21. For sequence 03, 04, 07 the accuracy of our system is lower than Huang et al. because there are rich line features in these three sequences, where the accuracy of Huang et al. achieves improvements. The runtime of our VO is 0.06 s/frame on an Intel Core i7-11700 CPU. In comparison, LIMO needs semantic label information to identify moving objects and reject outliers and thus GPU is demanded for speedup. Meanwhile, both LIMO and Huang et al. cannot run in real time. In contrast, our approach, DEMO and DVL-SLAM, are much more lightweight and can efficiently run in real time on a CPU-only platform. The testing results on sequence 11–21 of our visual odometry are also published on the official website with the name abbreviation as "SD-DEVO". It is worth mentioning that DEMO is the visual module utilized in V-LOAM. Therefore, the comparison result with DEMO can also demonstrate superiority of our VO to that of V-LOAM.

We validate the effectiveness of our depth-enhanced VO on KITTI-360 and KITTI-CARLA and show the results in Table I. We did not evaluate the state-of-the-art depth-enhanced visual odometry systems on these two datasets as they either did not release their source code (DEMO [12] and Huang et al. [15]) or the provided source code (DVL-SLAM [14] and LIMO [13]) are unable to compiled and run successfully. In addition, these methods are not evaluated on the two datasets.

3) Performance of SDV-LOAM: To provide fair comparisons with the state-of-the-art methods, we modified several open-sourced LiDAR odometry systems (i.e., A-LOAM, LeGO-LOAM [9], Fast-LOAM, ISC-LOAM [11], MULLS [24] and CT-ICP [19]) in which we concatenate our visual module before the LiDAR odometry to provide motion prior estimations. The corresponding modified visual-LiDAR odometers are denoted as A-LOAM+, LeGO-LOAM+, Fast-LOAM+, ISC-LOAM+, MULLS+ and CT-ICP+. In addition, we also compare our method with the state-of-the-art tightly integrated visual-LiDAR system V-LOAM [16].

Results in Table VI show that our SDV-LOAM outperforms A-LOAM+, LeGO-LOAM+, Fast-LOAM+, ISC-LOAM+ and MULLS+ on almost all sequences of the KITTI benchmark, KITTI-360 and KITTI-CARLA, demonstrating the effectiveness of our proposed system. Even on the most advanced visual-LiDAR system CT-ICP+, our system still gets great improvement on KITTI 01 and KITTI 03. Although our system cannot achieve more accurate pose estimation than V-LOAM on KITTI, our system can improve the frequency of input LiDAR point clouds (details are in Section VII-B2). In addition, we will release the source code of SDV-LOAM while V-LOAM neither release source code nor publish the literature for the latest version of their system.

In the testing set, according to the results evaluated by the KITTI official website, we achieve 0.60% drift in translation, which ranks the 8th place on the testing set of the KITTI odometry benchmark (before 2023.01), ranks 6th compared with all LiDAR-only/monocular visual-LiDAR systems and ranks 2nd compared with all open-sourced systems. Although the CT-ICP achieves 0.59% relative translational error, which ranks higher than us, however, the released code provided by them only achieves 0.62% relative translational error, while our system achieves 0.60%.

B. Evaluation on Custom-Built Sensor System

To evaluate the performance of our system in live environment and verify the effectiveness of our proposed sweep reconstruction block, we built our own camera-LiDAR sensor system (see Fig. 6) to acquire visual and LiDAR data, and run the data on a

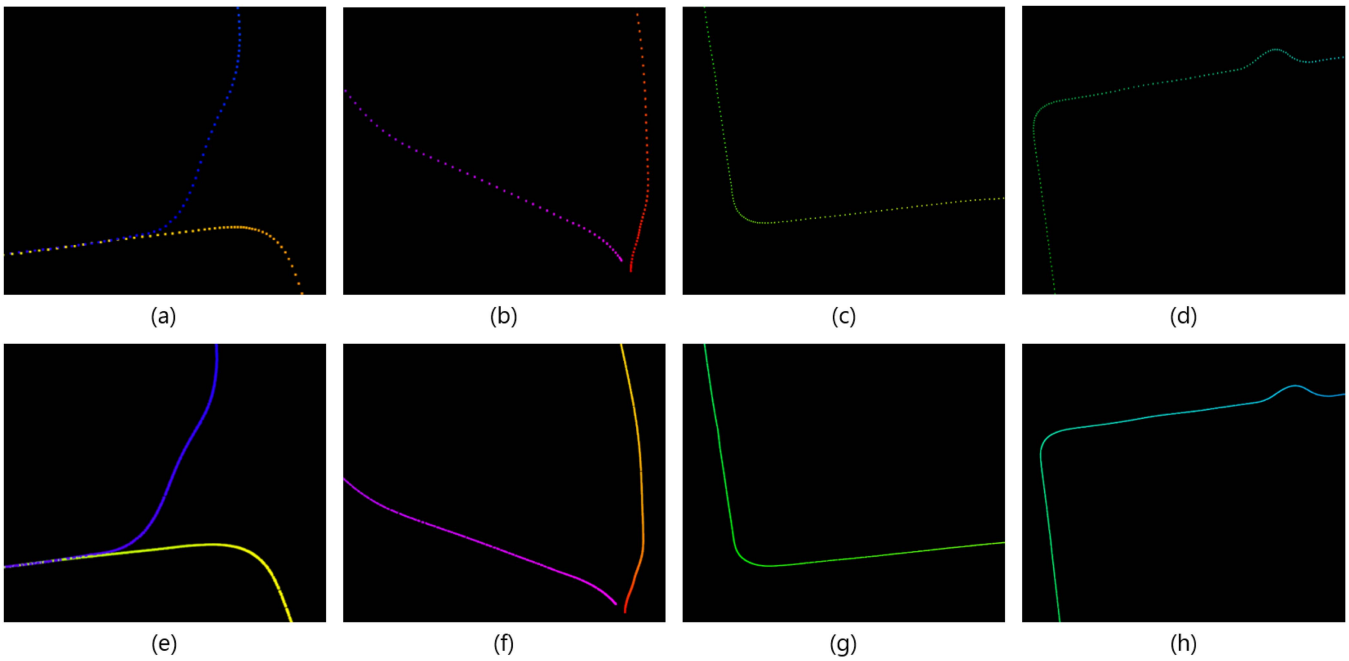


Fig. 9. The local screenshots of trajectories of Fig. 7(c) in the form of points. (a)-(d) are the results without sweep reconstruction while (e)-(h) are the results with sweep reconstruction. The resolution of points on the trajectory reflects the frequency of the output pose from our LiDAR odometry. The comparison between (a)-(d) and (e)-(h) demonstrates that our sweep reconstruction block can effectively improve the frequency of LiDAR pose estimation.

laptop offline. Our camera-LiDAR sensor consists of a FL3-U3-13E4M-C industrial camera and a Velodyne VLP-16 LiDAR. The FLIR camera provides a gray image with a resolution of 1280×1040 running at the frame rate of 60 Hz. The VLP-16 provides a relatively sparse point cloud over 16 laser beams at 10 Hz.

In order to achieve the optimal performance of the sensor system, we perform offline calibration between the heterogeneous sensors. Specifically, we utilized the tool Autoware [38] to first perform a calibration to obtain intrinsic parameters of the camera, and then calibrate the relative transformation between the camera and the LiDAR.

1) Visualization for Trajectory and Map: Our SDV-LOAM is validated by the sensor system in outdoor environments including loop-closure. The sensor system is mounted on a vehicle (as illustrated in Fig. 6) moving in the campus and uptown. Fig. 7(a) and (c) shows the trajectories estimated by our system on two different vehicle routes. In order to have a better evaluation, we overlaid the trajectories Fig. 7(a) and (c) with Google Map and obtain the visualization results in Fig. 7(b) and (d). We do not include loop-closing module in our full system. Even so, Fig. 7(a)–(d) show that few deviations are visible at the closed loop area of trajectory, demonstrating the high accuracy of our pose estimation on large-scale scenes even without loop closing.

Our system can also build an accurate map based on the accurate pose estimated by our LiDAR odometry. The final global map of Fig. 7(a) and (c) built by our system is shown in Fig. 8(a) and (b), where the map is represented by LiDAR point clouds.

2) Visualization for Sweep Reconstruction: To visualize the effectiveness of our sweep reconstruction, we display the local

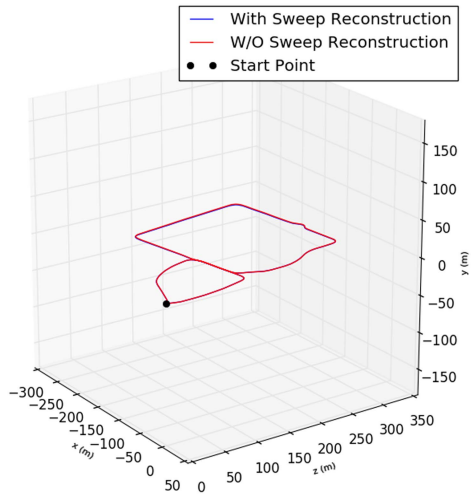


Fig. 10. The comparative result of trajectory Fig. 7(c) with and without sweep reconstruction, which are almost coincide. This result demonstrates that our method can increase the output frequency without affecting the accuracy of pose estimation.

screenshots of trajectory Fig. 7(c) in the form of points for in Fig. 9. Results show that the output frequency of the LiDAR odometry without sweep reconstruction (Fig. 9(a)–(d)) is obviously lower than that with sweep reconstruction (Fig. 9(e)–(h)), i.e., higher frequency can produce denser trajectory.

We further plot the trajectory estimated by our system with/without sweep reconstruction in a same coordinate (as shown in Fig. 10). Results show that our sweep reconstruction can increase the output frequency of LiDAR odometry without degrading the accuracy. It is worth explaining that our

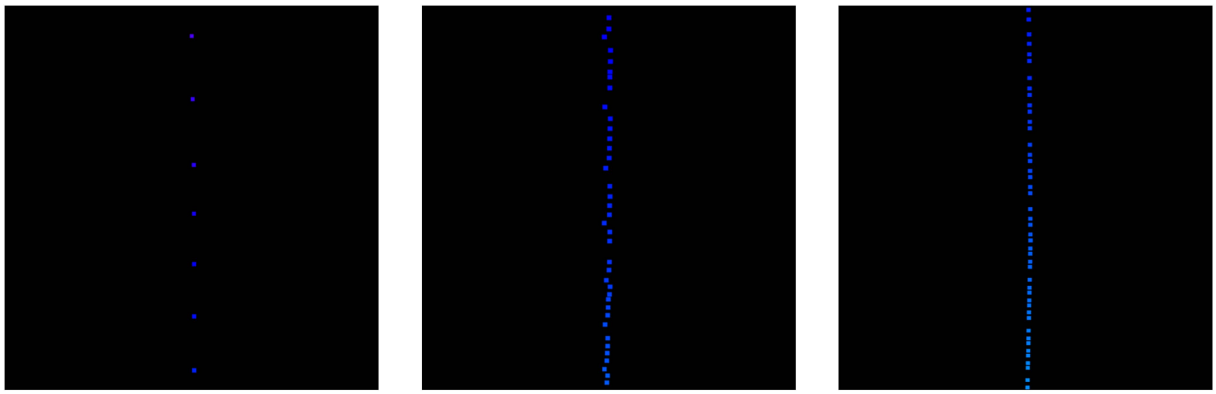


Fig. 11. (a) The 10 Hz final pose output of our LiDAR module. (b) and (c) are the 60 Hz final pose output obtained by transform integration and our sweep reconstruction respectively.

SDV-LOAM can output pose at 60 Hz in theory. Yet both our visual and LiDAR module can only output at a frequency of around 20 Hz in practice due to the limitation of the equipped computational resources. The output frequency can be further improved by using high-end computing devices or via multi-threading acceleration.

3) Evaluation of Sweep Reconstruction: We evaluate the benefit of our sweep reconstruction in our self-collected dataset in which the camera frames are captured at 60 Hz and the LiDAR sweeps are acquired at 10 Hz. We could not conduct this ablation study on KITTI [21], KITTI-360 [22] and KITTI-CARLA [23] as the frequency of camera images and LiDAR sweeps are both 10 Hz. Fig. 11 shows the trajectories of the 10 Hz final pose output based on our LiDAR module (Fig. 11(a)), the 60 Hz final pose output based on transform integration (Fig. 11(b)) and from our sweep reconstruction module (Fig. 11(c)) respectively. Although our self-collected dataset does not have ground truth to enable a quantitative evaluation, the comparison of Fig. 11(b) and (c) still can qualitatively demonstrate that our sweep reconstruction can better maintain local consistency of the estimated trajectory than transform integration.

VIII. Conclusion

This article introduces SDV-LOAM, an accurate and robust monocular visual-LiDAR odometry and mapping method, which fuses a semi-direct sparse depth enhanced visual odometry, a LiDAR odometry with adaptively sweep-to-map optimization in a complete system. Different from existing methods which employ feature-based or direct method as visual module, our SDV-LOAM adopts the semi-direct method which combines the success-factors of feature-based methods (re-projection error minimization for accurate pose estimation) with direct methods (eliminating the errors of depth association caused by sparse feature extraction when integrating LiDAR information into visual module). Aiming at the problem of inaccurate pose estimation of most existing LiDAR odometry in scenes with weak vertical geometric information, we proposed a adaptive sweep-to-map optimization block, which can automatically judge whether to optimize the 3 vertical DOF (i.e., roll, pitch, upward) according to the richness of geometric information in vertical direction of current sweep. In addition, different from existing visual-LiDAR systems which add the high-frequency but coarse camera poses on the low-frequency but accurate LiDAR poses to obtain the final output poses, we propose a sweep reconstruction block, which overlaps and reconstructs the sweep according to the time stamps of camera images to increase the input frequency of point clouds, and in turn increase the output frequency of LiDAR odometry in theory.

The proposed method achieves promising results on the public datasets as well as our own custom-built hardware system. Our whole system achieves 0.47% relative translational error on the training set of KITTI, 0.60% on the testing set of KITTI odometry benchmark, 0.46% on KITTI-360, and 0.038% on KITTI-CARLA. Meanwhile, live evaluations based on our custom-built hardware platform in outdoor environments prove the accuracy and robustness of our system in practical scenarios. Future work includes trying to utilize LiDAR depth as a soft constraint in visual BA.

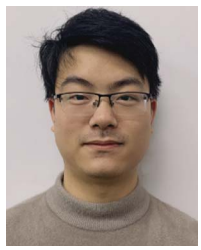
References

[1] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source slam system for monocular, stereo, and RGB-D cameras," *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017.
[2] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 834–849.
[3] J. Engel, J. Stückler, and D. Cremers, "Large-scale direct slam with stereo cameras," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2015, pp. 1935–1942.
[4] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2014, pp. 15–22.
[5] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, "SVO: Semidirect visual odometry for monocular and multicamera systems," *IEEE Trans. Robot.*, vol. 33, no. 2, pp. 249–265, Apr. 2017.
[6] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 3, pp. 611–625, Mar. 2018.
[7] R. Wang, M. Schwörer, and D. Cremers, "Stereo dso: Large-scale direct sparse visual odometry with stereo cameras," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 3923–3931.
[8] J. Zhang and S. Singh, "LOAM: Lidar odometry and mapping in real-time," in *Proc. Robot.: Sci. Syst.*, vol. 2, no. 9, 2014, pp. 1–9.

[9] T. Shan and B. Englot, "LeGO-LOAM: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 4758–4765.
[10] J. Behley and C. Stachniss, "Efficient surfel-based slam using 3D laser range data in urban environments," in *Proc. Robot.: Sci. Syst.*, vol. 2018, 2018, p. 59.
[11] H. Wang, C. Wang, and L. Xie, "Intensity scan context: Coding intensity and geometry relations for loop closure detection," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 2095–2101.
[12] J. Zhang, M. Kaess, and S. Singh, "A real-time method for depth enhanced visual odometry," *Auton. Robots*, vol. 41, no. 1, pp. 31–43, 2017.
[13] 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.
[14] Y.-S. Shin, Y. S. Park, and A. Kim, "DVL-SLAM: Sparse depth enhanced direct visual-lidar slam," *Auton. Robots*, vol. 44, no. 2, pp. 115–130, 2020.
[15] S.-S. Huang, Z.-Y. Ma, T.-J. Mu, H. Fu, and S.-M. Hu, "Lidar-monocular visual odometry using point and line features," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 1091–1097.
[16] 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.
[17] J. Zhang and S. Singh, "Laser–visual–inertial odometry and mapping with high robustness and low drift," *J. Field Robot.*, vol. 35, no. 8, pp. 1242–1264, 2018.
[18] Z. Yuan, K. Cheng, J. Tang, and X. Yang, "RGB-D DSO: Direct sparse odometry with RGB-D cameras for indoor scenes," *IEEE Trans. Multimedia*, vol. 24, pp. 4092–4101, 2022.
[19] P. Dellenbach, J.-E. Deschaud, B. Jacquet, and F. Goulette, "CT-ICP: Real-time elastic LiDAR odometry with loop closure," in *Proc. Int. Conf. Robot. Automat.*, 2022, pp. 5580–5586.
[20] J.-E. Deschaud, "IMLS-SLAM: Scan-to-model matching based on 3D data," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 2480–2485.
[21] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 3354–3361.
[22] Y. Liao, J. Xie, and A. Geiger, "KITTI-360: A novel dataset and benchmarks for urban scene understanding in 2D and 3D," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 3, pp. 3292–3310, Mar. 2023.
[23] J.-E. Deschaud, "KITTI-CARLA: A kitti-like dataset generated by carla simulator," 2021, *arXiv:2109.00892*.
[24] Y. Pan, P. Xiao, Y. He, Z. Shao, and Z. Li, "MULLS: Versatile LiDAR SLAM via multi-metric linear least square," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 11633–11 640.
[25] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, Oct. 2015.
[26] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap SLAM," *IEEE Trans. Robot.*, vol. 37, no. 6, pp. 1874–1890, Dec. 2021.
[27] 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.
[28] C. Yu et al., "DS-SLAM: A semantic visual slam towards dynamic environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 1168–1174.
[29] C. Kerl, J. Sturm, and D. Cremers, "Dense visual SLAM for RGB-D cameras," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2013, pp. 2100–2106.
[30] J. Zhang and S. Singh, "Low-drift and real-time lidar odometry and mapping," *Auton. Robots*, vol. 41, no. 2, pp. 401–416, 2017.
[31] D. Rozenberszki and A. L. Majdik, "Lol: Lidar-only odometry and localization in 3D point cloud maps," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 4379–4385.
[32] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, "LIO-SAM: Tightly-coupled lidar inertial odometry via smoothing and mapping," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 5135–5142.
[33] P. Smith, I. Reid, and A. Davison, "Real-time monocular slam with straight lines," in *Proc. Brit. Mach. Vis. Conf.*, 2006, pp. 17–26.
[34] A. P. Gee and W. Mayol-Cuevas, "Real-time model-based SLAM using line segments," in *Proc. Adv. Vis. Comput.: 2nd Int. Symp.*, Tahoe, NV, USA, 2006, pp. 354–363.
[35] G. Klein and D. Murray, "Improving the agility of keyframe-based SLAM," in *Proc. Eur. Conf. Comput. Vis.*, 2008, pp. 802–815.
[36] M. Himmelsbach, F. V. Hundelshausen, and H.-J. Wuensche, "Fast segmentation of 3D point clouds for ground vehicles," in *Proc. IEEE Intell. Veh. Symp.*, 2010, pp. 560–565.
[37] I. Bogoslavskyi and C. Stachniss, "Fast range image-based segmentation of sparse 3D laser scans for online operation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 163–169.
[38] S. Kato et al., "Autoware on board: Enabling autonomous vehicles with embedded systems," in *Proc. IEEE/ACM 9th Int. Conf. Cyber- Phys. Syst.*, 2018, pp. 287–296.
[39] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proc. 1st Annu. Conf. Robot Learn.*, 2017, pp. 1–16.

Zikang Yuan received the BE degree from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2020. He is currently working toward the PhD degree with HUST, School of Institute of Artificial Intelligence. He has published two papers on ACM MM and one paper on TMM. His research interests include monocular dense mapping, RGB-D simultaneous localization and mapping, visual-inertial state estimation and visual-LiDAR pose estimation.

Qingjie Wang received the BE degree from the HuaZhong University of Science and Technology(HUST), Wuhan, China, in 2020. She is currently working toward the graduate degree with HUST, School of Electronic Information and Communications. Her research interests include visual-LiDAR odometry, object reconstruction and deep-learning based flow, depth estimation.



Ken Cheng received the BE degree from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2019. He is currently working toward the graduate degree with HUST, School of Electronic Information and Communications. His research interests include RGB-D simultaneous localization and mapping and visual-LiDAR pose estimation.

Tianyu Hao received the BE degree from the Guilin University of Electronic Technology (GUET), Guilin, China, in 2018. He is currently working toward the postgraduate degree with the School of Electronic Information and Communications in Huazhong University of Science and Technology (HUST). He has published one paper on CVPR2021 and one paper on *IEEE Sensors*. His research interests include visual simultaneous localization and mapping(V-SLAM), hardware design and embedded device development.

Xin Yang (Member, IEEE) received the PhD degree from the University of California, Santa Barbara, in 2013. She worked as a Post-doc in Learning-based Multimedia Lab, UCSB (2013-2014). She is current professor of Huazhong University of Science and Technology School of Electronic Information and Communications. Her research interests include simultaneous localization and mapping, augmented reality, and medical image analysis. She has published more than 90 technical papers, including *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *International Journal of Computer Vision*, *IEEE Transactions on Medical Imaging*, MedIA, CVPR, ECCV, MM, etc., co-authored two books and holds 3 U.S. Patents. He is a member of ACM.