Direct Visual SLAM using Sparse Depth for Camera-LiDAR System

Young-Sik Shin, Yeong Sang Park and Ayoung Kim

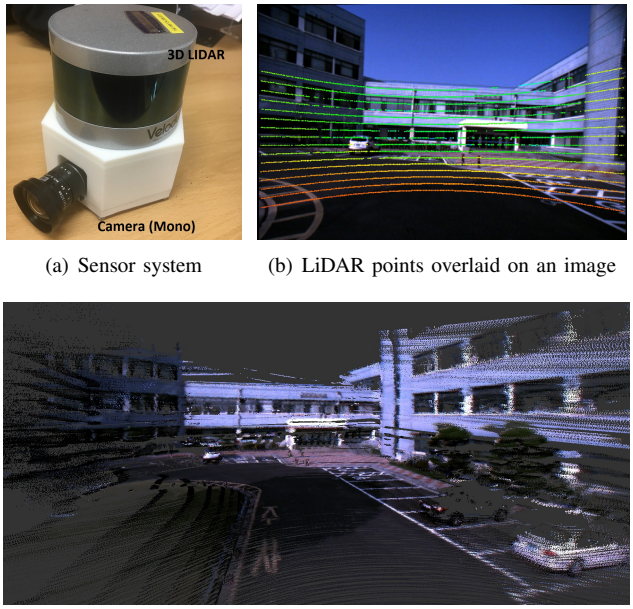
***Abstract*— This paper describes a framework for direct visual simultaneous localization and mapping (SLAM) combining a monocular camera with sparse depth information from Light Detection and Ranging (LiDAR). To ensure real-time performance while maintaining high accuracy in motion estimation, we present (*i*) a sliding window-based tracking method, (*ii*) strict pose marginalization for accurate pose-graph SLAM and (*iii*) depth-integrated frame matching for large-scale mapping. Unlike conventional feature-based visual and LiDAR mapping, the proposed approach is *direct*, eliminating the visual feature in the objective function. We evaluated results using our portable camera-LiDAR system as well as KITTI odometry benchmark datasets. The experimental results prove that the characteristics of two complementary sensors are very effective in improving real-time performance and accuracy. Via validation, we achieved low drift error of 0.98% in the KITTI benchmark including various environments such as a highway and residential areas.**

I. Introduction

Simultaneous localization and mapping (SLAM) has been the focus of many robotics researches for the last two decades. Many visual SLAM studies have been published from monocular cameras to dense RGB-D sensors. However, there has been little interest in visual SLAM systems that exploit relatively sparse depth information from LiDAR. For 3D perception of outdoor robots and autonomous vehicles, the combination of 3D LiDAR and camera has attracted attention as the most practical type of sensor. This sensor system maintains the 3D spatial structure through SLAM, and it enables high level tasks such as path planning, obstacle avoidance, and robot manipulation. While conventional LiDAR sensors provide dense and rich point clouds from high-cost sensors, upcoming LiDARs, including the solid-state sensor type, are expected to provide sparse point data targeting a broader range of consumers.

In robotic 3D mapping, LiDAR is a popular sensor for motion estimation and mapping. Evolving from conventional approaches using the Iterative Closest Point (ICP), recently researchers have utilized dense point data [1], [2]. Conventional ICP-based methods easily failed as reported in [3] due to vertical measurement sparsity. This limited point cloud density leads to inaccurate motion estimation, which causes the 3D environment reconstruction to fail. Achieving a consistent map under data sparsity is critical, especially for loop-closure in a large-scale environment.

In recent years, visual SLAM with a monocular camera has made considerable progress from both visual feature-based indirect methods and direct methods. Direct methods especially have been investigated and show remarkable motion estimation performance without using visual features [4]–[9]. However, when a camera moves along the optical axis, pixel positional change of salient visual features is subtle making camera motion estimation ill-posed. To overcome this scale-ambiguity of the single camera, the monocular system is inevitably required to bootstrap to initialize the geometry for the 3D environment [4], [10], [11]. The quality of this initialization determines the scale error of the real world and the estimation.



(a) Sensor system (b) LiDAR points overlaid on an image

(c) The reconstructed 3D pointcloud map

Fig. 1. A combined system of camera and sparse 3D LiDAR for motion estimation and mapping. The proposed method aims to estimate sensor pose and to reconstruct 3D environments. (a) A prototype of our portable camera-LiDAR sensor system that acquires a single image and sparse depth measurements as in (b). (c) From these sensor configuration, the proposed methods perform the precise motion estimation and the 3D mapping.

In this work, we propose a visual SLAM system that combines images acquired from a camera and sparse depth information obtainable from 3D LiDAR. Note that as shown in Fig. 1, only limited depth information is available from the sensors. Because the association of these sparse depths in a visual feature would result in a shortage of meaningful features, we concluded that the direct method is more effective for handling the sparse depth data. The direct method has advantages for the lower resolution camera and sparse LiDAR combination because the reduction of image resolution less deteriorates visual odometry performance when

Y. Shin, Y. Park and A. Kim are with the Department of Civil and Environmental Engineering, KAIST, Daejeon, S. Korea [youngsik.shin, pys0728k, ayoungk]@kaist.ac.kr

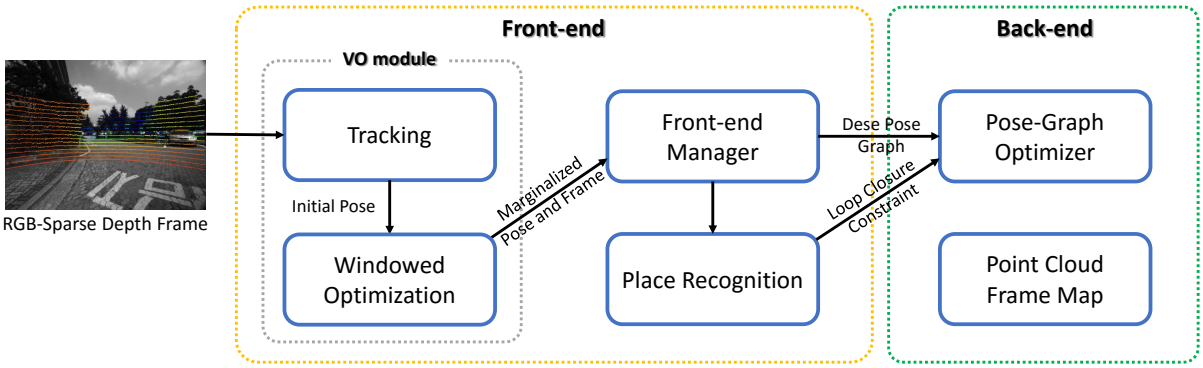


Fig. 2. Block diagram of the direct sparse odometry and mapping system. The proposed method consists of the front-end visual odometry and back-end solver for the graph optimization considering loop-closures.

using the direct method if the corresponding depth is properly associated as described in [5].

Still, the narrow vertical field of view (FOV) of sparse 3D LiDAR guarantees only limited depth association within a partial image region. To overcome this shortcoming and ensure robustness, we perform optimization with a fixed number of frames for odometry and utilize sparse depths of neighboring frames by integrating into keyframes for 6-DOF loop-closure constraints.

This paper proposes a depth-enhanced visual SLAM method for a sensor system that combines a camera and 3D LiDAR. The main characteristics of the proposed method can be explained as follows:

- We propose direct visual SLAM only when sparse depth measurements are available. Visual features are excluded in motion estimation to cope with LiDAR sparsity.
- A sliding window-based optimization method is utilized to overcome the limitations of heterogeneous sensors with different FOVs. When performing the pose-graph optimization, we strictly marginalize the pose of the sliding window.
- To deal with map consistency in a large-scale environment, we use a keyframe-based loop constraint.
- The proposed method has been thoroughly evaluated in a large-scale public dataset that contains various environments as well as directly acquired data.

II. Related Work

A number of studies have been aimed at motion estimation using cameras [12]–[15]. For a long time, the most dominant approach was feature-based approach (indirect) for both monocular and stereo cameras. This feature-based approach, however, was reported to have a critical drawback when the environment has simple and repeated patterns or is featureless, resulting in the failure of the motion estimation.

The recent advent of direct methods has alleviated the existence of feature issues without requiring feature correspondences. Instead of using a feature-based reprojection error model, they mainly use a photometric error model that considers the intensity difference of a pixel [4]–[6], [9], [16]. However, in the case of a single camera, performance is determined by bootstrapping, which causes good initialization. Furthermore, a Graphics Processing Unit (GPU) is often required when estimating both a full-depth map and motion [6], [16].

As compensation between the two abovementioned methods, a semi-dense approach was proposed for central processing unit (CPU)-based, real-time performance [4], [5], [9]. Direct Sparse Odometry (DSO) is a state-of-the-art direct visual odometry method that uses a single camera [5]. It maintains a fixed number of points and performs window-based optimization to optimize all relevant parameters (e.g., camera intrinsics, motion, inverse depth value). The window-based optimization ensures fairly high odometry accuracy. However, it has scale-ambiguity like any single camera method. Inspired by them, we use similar window-based optimization to improve odometry accuracy of the camera-LiDAR system. Differing from their method, the use of LiDAR data solves the scale-ambiguity problem of a single camera. Furthermore, unlike previously mentioned approaches, the proposed method proposes a full SLAM framework.

The application of the direct method is compelling when using RGB-D sensors. Provided with a dense depth of information, adding geometric errors becomes straightforward [7], [8], [17], [18]. Among them, Depth Enhanced Monocular Odometry (DEMO) [18] is the most comparable to ours. DEMO targets a similar sensor configuration (i.e., a monocular camera with LiDAR). DEMO presents a tightly coupled camera and LiDAR for visual odometry and tracks visual features and associated feature and depth measurements when depth is available. Ours is similar to them in that the limitation of a monocular camera was overcome by associating LiDAR data accordingly presenting excellent performance. However, their method is indirect, exploiting dense depth information in the odometry computation.

III. Algorithm Overview

The proposed method consists of a tracking-based visual odometry (VO) module, a loop-closure module and a pose-graph optimizer as shown in Fig. 2. The input frame consists

of synchronized camera images and points from LiDAR. The proposed method consists of a front-end to construct a pose-graph and a back-end to optimize it. The front-end includes a visual odometry module and an appearance-based place recognition module.

Tracking: The visual odometry module performs robust and low-drift motion estimation using a sliding window optimization. Instead of relying on visual features, the process examines raw images in the intensity domain. The incoming new image frame and sparse depth measurements are piped into a tracking module. Similar to the general direct methods, a multiscale image pyramid and constant motion model-based initialization are applied to the tracking process [4], [5], [7].

Windowed optimization: Once successfully tracked, the frame is added to the sliding window, marginalizing the oldest frame. Next, windowed optimization is used to refine the frames within the window. Such sliding window-based optimization allows the output of the tracking process to have low drift.

Loop-closure: The final component of the front-end is the place recognition module. Instead of using all of the frames in the place recognition module, we use only selected keyframes. If it is confirmed as a valid revisit, the depth information is integrated from adjacent frames to perform two-view matching.

Back-end optimizer: The back-end module optimizes the pose-graph, including the loop constraints provided by the place recognition module. We used $g2o$ [19] with an incremental sparse linear algebra for the online operation of the optimization.

IV. Depth-Enhanced Direct Visual Odometry

We first explain the VO module using the direct method. This first phase includes frame management, frame marginalization, tracking and windowed optimization.

A. Notation

A frame $\mathcal{F}_i$ provides an RGB Image $\mathcal{I}_i$ and a sparse point cloud $\mathcal{P}_i$. Each point $\mathbf{p}_k = (x, y, z)^T \in \mathbb{R}^3$ is projected as an image coordinate and can be represented as a pixel $\mathbf{u_k} = (u, v) \in \mathbb{R}^2$ through camera projection function $\pi(\mathbf{p}_k)$. The points are anchored relative to the camera coordinate system of the frame. The pose of each frame is represented by a transformation matrix $\mathbf{T}_n \in SE(3)$ in the world coordinate system. Taking the tracking process as an example, the relative pose transformation between $\mathcal{F}_n$ and $\mathcal{F}_m$ can be denoted by $\mathbf{T}_m^n = \mathbf{T}_n^{-1}\mathbf{T}_m$ and can be used for coordinate transformation of points between the coordinates as follows:

$$\mathbf{p}_n = \mathbf{T}_m^n \mathbf{p}_m = \begin{bmatrix} \mathbf{R}_m^n & \mathbf{t}_m^n \\ \mathbf{0} & 1 \end{bmatrix} \mathbf{p}_m \tag{1}$$

where $\mathbf{p}_m$ and $\mathbf{p}_n$ are points on each frame, $\mathbf{R}_m^n \in SO(3)$ and $\mathbf{t}_m^n \in \mathbb{R}^3$ are a rotation matrix and translation vector. Lie algebra elements $\xi \in \mathbb{R}^6$ are used as linearized pose increments throughout this paper. $\xi = [\mathbf{w}, \mathbf{v}]$ consist of an angular and linear velocity vector. The updated transformation matrix using the increment can be calculated by exponential mapping from Lie algebra to Lie group : $se(3) \times SE(3) \rightarrow SE(3)$.

$$\bar{\mathbf{T}}_m^n = \exp(\hat{\xi})\mathbf{T}_m^n \tag{2}$$

B. Frame Management

1) Frame generation: To use depth measurement of LiDAR directly in the image frame, time synchronization between sensors is required. Two sensors are physically separated, and the 3D LiDAR often operates at a slower rate (10 Hz) than the camera. In order to minimize this difference, we check the closest time stamp and define the synchronized state of LiDAR and camera as an RGB frame with sparse depth. Once the synchronized data are found, the 3D points obtained from LiDAR are transformed into the camera coordinate system.

2) Keyframe management: The proposed method always maintains a fixed number of keyframes, and the most recent frame is tracked by current frames. The strategy of the keyframe is to mitigate the drift of the odometry results and reduce the computation load of the sliding window optimizer. To create these keyframes, we use the following two criteria:

- The new keyframe is created using the ratio of projectable points when tracking is successful. Because the FOV of LiDAR and the camera is fixed and known, point projection between two viewpoints easily determines the keyframe creation event. We only register frames with ratio less than γ_{kf} as a keyframe.
- When using the overlap ratio, a problem occurs when motion is subtle and illumination variation occurs over time. To alleviate this issue, we create a keyframe after a certain period of time (t_{kf}) to secure robustness to the illumination change.

The parameters used in this paper are summarized in Table. I.

3) Keyframe marginalization: To maintain a fixed number of keyframes and construct a pose-graph, we marginalize the oldest frame in the sliding window. This process can be performed each time a new keyframe is created and requires the following additional strategies:

- We always keep at least two keyframes. If a new keyframe is created, such as when the ratio of visible points in the most recent frame is less than γ_{kf}, then the keyframe will be inserted into the keyframe window.
- When the number of keyframes exceeds the size of the window (N_w), marginalization is performed in the oldest order.

When a new keyframe is added the oldest keyframe in the window should be marginalized. Unlike a single camera that estimates landmarks as a state [5], our frame has points acquired from LiDAR with no landmark defined in the state. Using the Schur complement, we explicitly marginalize the factor of the oldest keyframe for the pose-graph. We then perform iterative optimization using the remaining Hessian blocks when a new keyframe is added.

4) Point management: Using entire LiDAR points would burden frame-wise computation. To reduce computational complexity while sustaining motion estimation accuracy, the proposed method utilizes a well-distributed point sampling strategy in the frame. To extract the sample points, we uniformly sample LiDAR data from 2° by 2° spatial bin. Then we select the point with the largest gradient in each point set as the active point. Because points are simply separated by intervals, points in regions with low gradients can also be activated.

C. Tracking

Inspired by sparse direct methods [4], [5], our tracking method defines a photometric error of a point $\mathbf{p}_m$ in a patch Ω of a specific sparse pattern at the image $\mathcal{I}_m$ of the latest keyframe $\mathcal{F}_m$. This patch-based approach requires more computation on its size but ensures robustness with image degradation such as motion blur and noise. In the tracking stage, the point $\mathbf{p}_m$ of the keyframe is projected to the image $\mathcal{I}_n$ of the current frame $\mathcal{F}_n$ by an initial transformation, and the residuals representing the difference of the brightness in grayscale are defined as

$$r(\mathbf{p}_m) = \mathcal{I}_n(\pi(\mathbf{T}_m^n \mathbf{p}_m)) - \mathcal{I}_m(\pi(\mathbf{p}_m)). \tag{3}$$

Finally, our objective function for tracking is defined below:

$$E_{track} = \sum_{\mathbf{p}\in\Omega} w(r(\mathbf{p}))(r(\mathbf{p}))^2 \tag{4}$$

,where Ω is the patch of the sparse pattern and the function $w(\cdot)$ is a weight function based on the t-distribution, as shown in [7].

$$w(r) = \frac{\nu + 1}{\nu + (\frac{r}{\sigma_r})^2} \tag{5}$$

Here, ν is the degree of freedom that was set to five and the variance of residual, σ_r, is iteratively estimated while performing the Gauss-Newton optimization algorithm. The effect of weighting is reported in [7]. Now, we use a coarse-to-fine scheme to prevent optimization from local minima and to handle large displacements.

D. Window Based Optimization

In the previous section, the tracking process estimates the relative motion between the new frame and the reference frame. If the new frame satisfies the new keyframe creation criteria, it is added to the window and the oldest frame in the window is marginalized. When a new keyframe is added, we perform window-based batch optimization using the Gauss-Newton algorithm. The local optimization overcomes the lack of information that depth measurement available only in some regions of the image. However, this window based approach has a trade-off between computation speed and accuracy depending on its size. For this reason, we kept a fixed number of 5 to 10 frames.

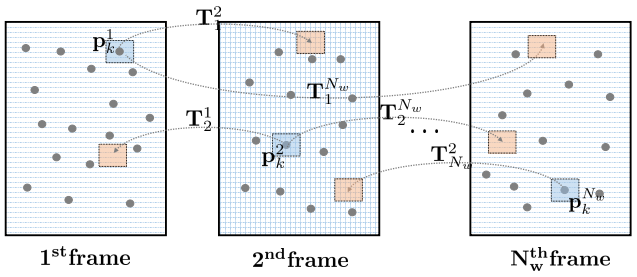


Fig. 3. Example of window-based optimization. The window consists of N_w^{th} keyframes, and each keyframe has its image and pointcloud. Gray dots represent the pointcloud and the rectangle represent images. A point $\mathbf{p}_k$ in keyframe is projected into all keyframes with covisibility. A photometric residuals are calculated between the patch at the point and the other patch at projected point.

As depicted in Fig. 3, each keyframe ($\mathcal{F}_i$) in the window ($\mathbf{W}_\mathcal{F}$) is associated with points ($\mathbf{P_i}$) and its own pose ($\mathbf{T_i}$). A point ($\mathbf{p}_k$) is projected into all keyframes with covisibility within the window. This procedure generates a pose-graph, where we only perform optimization on the pose of keyframes. The final photometric error between all keyframes in the sliding window is defined over a patch Ω.

$$E_{win} = \sum_{\mathcal{F}_i\in\mathbf{W}_\mathcal{F}} \sum_{\mathbf{p}_k\in\mathbf{P_i}} \sum_{\mathbf{p}_k\in\Omega} w(r(\mathbf{p}_k))(r(\mathbf{p}_k))^2, \tag{6}$$

The main difference to the cost function in the tracking process is at reflecting the error between all keyframes. We use the t-distribution weight function $w(\cdot)$ in this process as well.

E. Loop Closure

The visual odometry module provides a pose for the strictly marginalized keyframe. We add the pose of all keyframes to a node of the pose-graph to generate a consistency point map from LiDAR. Once the keyframe is marginalized, we extract the Oriented FAST and Rotated BRIEF (ORB) features and descriptor. Then we apply DBoW with trained vocabulary from ORB-SLAM [15], [20]. To prevent false positives, we use the similarity score between the frame and its neighboring frame as criteria for allowing loop candidates.

We further perform a cross-check by comparing two relative pose estimations. Given a loop candidate frame $\mathcal{F}_c$ and the current keyframe $\mathcal{F}_n$, we separately compute relative pose when projecting candidate frame to the current frame ($\mathbf{T}_c^n$) and when projecting current frame to the candidate frame ($\mathbf{T}_n^c$). Since the candidate frame and the current keyframe have depth information from LiDAR, we can independently calculate T_c^n and T_n^c. Only when this cross-comparison presents strong agreement (i.e., small discrepancy), we add it to the pose-graph as a loop constraint. Finally, we apply Dynamic Covariance Scaling (DCS) to the loop edge to cope with unreliable constraints [21].

V. Result

Our proposed method was validated using various camera-LiDAR systems. The proposed method was applied to both a publicly available KITTI odometry dataset [22] and our portable camera-LiDAR sensor system. All experiments were performed on a laptop with an Intel Core i7-7700HQ processor (4 cores, max 3.8 GHz) and 8 GB RAM. In particular, the KITTI dataset was used without loop-closure to verify odometry performance. Although a stereo image is provided from the dataset, we verified the proposed method using only the single camera and a LiDAR sensor.

Next, the proposed method was validated using our custom-built sensor platform. Our sensor system consists of a camera and 3D LiDAR. The camera provides an RGB image with a resolution of 800×600, a frame rate of 30 Hz and a horizontal FOV of $90°$. In our camera-LiDAR system, a relatively cost efficient 3D LiDAR is used (VLP-16). The VLP-16 provides a sparse point cloud from 16-ray range measurements, only guaranteeing depth for a small image region. The proposed method requires precise calibration between heterogeneous sensors for optimal performance. For our sensor system, intrinsic calibration of the camera and extrinsic calibration between the camera and LiDAR were precisely performed in a conventional manner [23]. The proposed method was evaluated in indoor and outdoor environments. The indoor experiment was carried out in the hall with the device in hand as shown in Fig. 7. The outdoor experiment was tested by attaching the device to our mobile mapping system [24] as shown in Fig. 8. Several parameters for keyframe management are introduced in Table. I. The overall process and projected point cloud map are also displayed in `camlidar.mp4`.

TABLE I

Parameters for keyframe management

	Description	KITTI	Indoor	Outdoor
γ_{kf}	Ratio for keyframe creation	0.8	0.8	0.8
t_{kf}	Time for keyframe creation (sec)	1.0	1.0	1.0
N_w	Window size (number of frames)	5	7	5

A. Visual Odometry Evaluation using KITTI

From the KITTI dataset, we use the left monochrome image and LiDAR for validation. We performed the motion estimation at an original image resolution of 1230×370. The main validation focus was the robustness of our method in terms of LiDAR sparsity. To simulate and validate the proposed approach, we sampled 64 laser beams into 16, and 8 beams in order to simulate the sparse data (Fig. 4). Via this test, we evaluated only the visual odometry performance without using loop-closures.

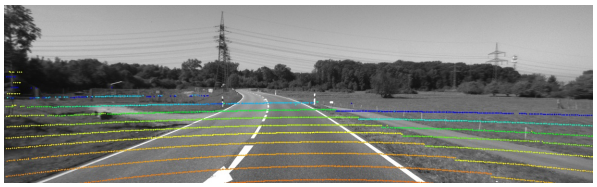
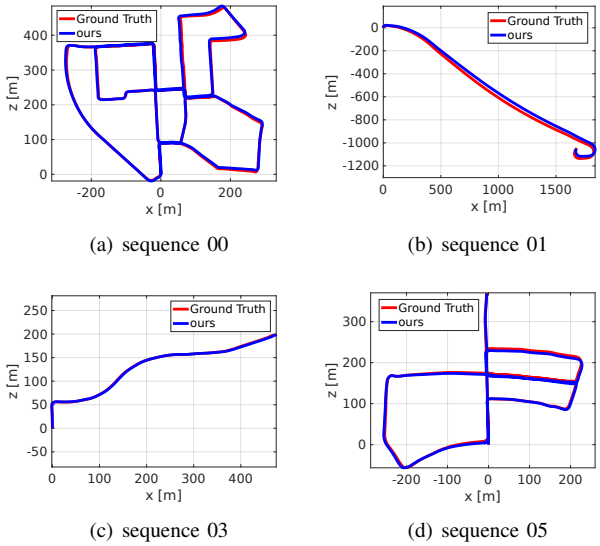


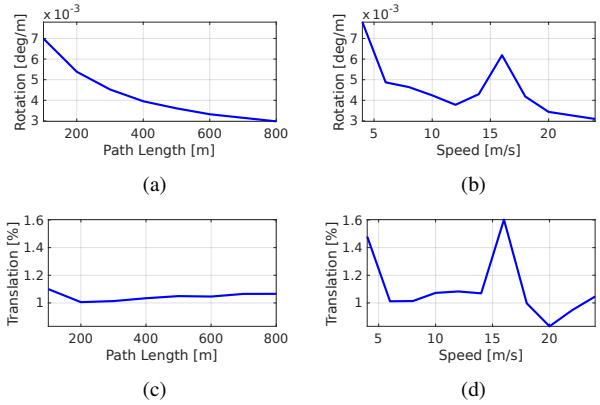
Fig. 4. Given a rich LiDAR data (64 laser beam from HDL64-E), we simulate a sparse LiDAR by sampling laser. The color is coded by the depth value.



(a) sequence 00 (b) sequence 01

(c) sequence 03 (d) sequence 05

Fig. 5. Sample comparison results with ground truth. The results are chosen from each environmental type. The red line represents the ground truth and the blue line represents the result of the proposed method.



(a) (b)

(c) (d)

Fig. 6. The detailed results in the overall training sequences. The top and the bottom rows show rotational errors and translational The errors left and the right columns show the results according to path length and speed.

The evaluation was performed using the provided ground truth over 11 data sequences. Fig. 5 shows the comparison result of the estimated path between the proposed method and the ground truth in sample sequences. Note that sequence `01` is a highway dataset moving at high speed with moving objects.

The comparison results with DEMO on overall training sequences are shown in Table. II. Since the optimal parameters of DEMO are unknown for the reduced LiDAR data of 16 ray, we directly refer to results reported in [18]. The measured translational error is the mean of the error obtained by dividing the segment of the trajectory by 100 m, 200 m, $\cdots$, 800 m. The proposed method shows an average translational error of 0.98% in 64 ray, which is less than the

TABLE II

COMPARISON RESULT OF KITTI DATASET. THE RESULT REPRESENT AN AVERAGING TRANSLATIONAL ERROR AT EACH SEGMENT LENGTH. THE BOLD TYPE INDICATES THE BEST ACCURACY.

Seq No.	Path len. [m]	Environment	DEMO (64 ray) [18]	ours (64 ray)	ours (16 ray)	ours (8 ray)
0	3714	Urban	1.05	0.93	**0.93**	0.95
1	4268	Highway	1.87	1.47	**1.09**	7.25
2	5075	Urban+Country	**0.93**	1.11	1.26	0.95
3	563	Country	0.99	**0.92**	0.98	0.94
4	397	Country	1.23	**0.67**	1.08	1.08
5	2223	Urban	1.04	0.82	**0.77**	0.77
6	1239	Urban	0.96	0.92	**0.75**	0.75
7	695	Urban	**1.16**	1.26	1.30	1.42
8	3225	Urban+Country	**1.24**	1.32	1.30	1.30
9	1717	Urban+Country	1.17	**0.66**	0.72	0.72
10	919	Urban+Country	1.14	0.70	**0.57**	0.57
avg			1.16	0.98	**0.98**	1.52

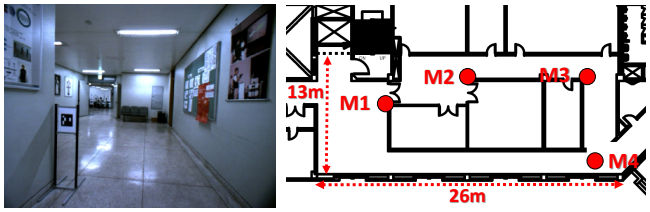
error obtained when using DEMO. We further provide our results as the LiDAR becomes sparser to 16 and 8 robustness to sparsity. The average translational error on the 16 ray is 0.98%, which is almost similar to the 64 ray. However, in case of 8 ray, it shows considerable error in sequence `01`. The dataset has similar texture repeats and fast moving vehicles appear. Particularly, due to the influence of the moving vehicle in a specific area, a large error of 7.25% has occurred.

We show the detailed results according to the length of the trajectory segments in Fig. 6. The top and bottom rows show rotational and translational errors. We represent the rotational error with respect to the path length in Fig. 6(a) and, the speed in Fig. 6(b). The translational errors are shown in Fig. 6(c) and Fig. 6(d) according to the same criteria. As path length increases, the rotational error decreases in Fig. 6(a). On the other hand, translational errors show a stable trend regardless of path length in Fig. 6(c). This means that the proposed method may be slightly biased for translational motion without acceleration, though it is considerably consistent. The error according to the speed of the right column contains outliers. The rotation error in Fig. 6(b) tends to decrease as the velocity increases. This is caused by the nature of the vehicle with non-holonomic constraints and is only seen as a feature of the dataset itself.

B. Indoor Camera-LiDAR Dataset

To provide the ground truth in an indoor environment, we performed experiments with ArUCo markers [25] set at four locations as in Fig. 7(b). Fig. 7(a) shows the markers that were installed for the experiment. Since the absolute positions of the markers are not known accurately in indoor environment, we used the Root Mean Sqaured Error (RMSE) between relative pose computed using the markers and relative pose in the estimated trajectory when returning to the same area.

Evaluation was performed on the four points (M1 to M4). Table. III summarizes the error between the relative pose measured by the marker and the relative pose estimated by the proposed method. Due to the drift reduction effect of the pose graph optimization, the proposed method demonstrates



(a) Indoor environment (b) Floor plan with markers

Fig. 7. Indoor experimental environment. (a) shows an image of the corridor environment in which the experiment was performed. (b) shows the floor plan of the indoor environment, and red dot indicates the point where the ArUCo marker is installed.

excellent performance with a relative error of about 0.105 m even though the movement is about 60 m during each loop.

The qualitative results of the overall trajectory and the resulting point cloud maps are shown in Fig. 9. Fig. 9(a) shows consistent trajectory results for three laps when using appropriate loop constraints. The colored point cloud map obtained from the results is shown in Fig. 9(b). It provides a sparse map compared to the dense RGB-D camera. As the points are accumulated by the movement, the map data are acquired for the whole area. Even without performing any point matching, loop closure yielded an excellent-quality point cloud map.

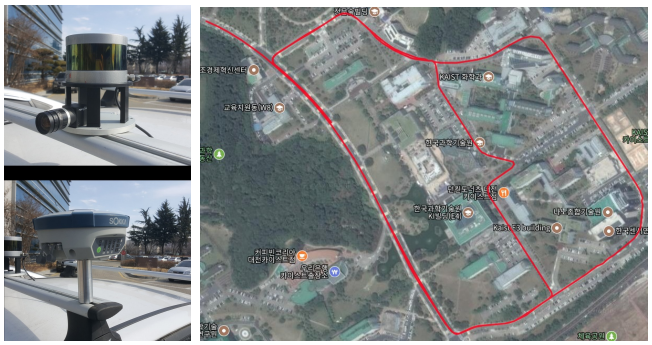
C. Outdoor Camera-LiDAR Dataset

We carried out experiments to verify the proposed method in a large-scale environment by attaching a portable camera-LiDAR device on a car-like vehicle. As shown in Fig. 8(a), the vehicle was equipped with Virtual Reference Station

TABLE III

RELATIVE POSITION ERROR BETWEEN THE MEASURED RELATIVE POSE BY THE MARKER AND THE RELATIVE POSE IN THE ESTIMATED TRAJECTORY

	M1	M2	M3	M4
loop 1-2	0.1263m	0.0712m	0.1074m	0.0535m
loop 2-3	0.0655m	0.0633m	0.2540m	0.0668m
loop 3-1	0.1323m	0.0770m	0.1837m	0.0636m



(a) Sensors (b) Experimental route

Fig. 8. Outdoor experimental environment. (a) shows an sensor configuration in the experiment. The top-left is a camera-LiDAR sensor on vehicle and the top-right is a VRS-GPS that is only used for validation. (b) shows a satellite map with the experimental route.

(VRS)-global positioning system (GPS) as well as camera-LiDAR device. The VRS-GPS produces an average error of 20 mm in the fixed state and an error up to 1 m in the floating state, which is used to determine ground truth vehicle positions. Fig. 8(b) shows the satellite image and route where the experiment was performed. The total traveled distance for the experiment is 3.97 km with two loops.

We compared our estimated trajectory against the baseline provided by VRS-GPS. The GPS provides fairly accurate information in both fixed and floating states, so only the measured values in those states are compared. However, the VRS-GPS is regarded as non-continuous because the fixed and floating states do not occur due to the multi-path phenomenon on a narrow road passing between buildings.

Table. IV shows the relative position error between the proposed method and VRS-GPS, only at a fixed state. In order to compare the performance with more sparse depth information, we compared the actual sensor data of 16 rays to 8 ray data. Because the measured GPS data are discontinuous, we calculated and averaged the error over the 11 intervals in which the data were continuously acquired. The proposed method shows a relative position error of 1.11%. When the proposed method is applied to 8-ray data, it shows an error of 1.4%. We conclude that the SLAM performs fairly well even with sparse LiDAR data of narrow vertical FOV (30°).

TABLE IV

RELATIVE POSITION ERROR BETWEEN THE VRS-GPS AND THE ESTIMATED POSE

	16-ray	8-ray
Outdoor experiment	1.11%	1.42%

Our method includes a loop-closure and presents a consistent trajectory even in a large-scale environment. Additionally, the sensor-system uses the depth information from LiDAR projected on the camera, so we can acquire a colored point cloud map directly with the trajectory. Fig. 9(e) shows the point cloud map obtained from the experiment. Even though point matching is not performed at all, it is possible to obtain a fairly accurate point map. Note that the result is only obtained with visual odometry and loop-closure. Further refinement from mapping as in [26] would substantially improve the accuracy.

VI. CONCLUSION

In this work, we introduced a direct visual SLAM method for camera-LiDAR sensor systems where sparse depth is available. The proposed method showed robust SLAM results even if extremely sparse depth measurements (8 ray) were available in a large-scale environment. Sliding window-based pose estimation prevents local level drift, while the Appearance-based place recognition module and the pose graph optimizer maintain the consistency of the global level. The method was evaluated using our own experiment and KITTI benchmark dataset. Our method provides accurate trajectory and point cloud maps of real-world metrics without performing point matching at all.

In future work, we plan to extend our approach to robust camera-LiDAR SLAM systems for illumination changes using geometric information from LiDAR. In addition, we consider ensuring accuracy and robustness by handling moving objects.

ACKNOWLEDGMENT

This material is based upon work supported by MSIP (No. 2015R1C1A2A01052138) and by Korea Ministry of Land, Infrastructure and Transport(MOLIT) as 'U-City Master and Doctor Course Grant Program'.

REFERENCES

[1] A. Segal, D. Haehnel, and S. Thrun, "Generalized-ICP," in *Proceedings of the Robotics: Science & Systems Conference*, Seattle, USA, 2009.

[2] J. Serafin and G. Grisetti, "NICP: Dense normal based point cloud registration," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015, pp. 742–749.

[3] M. Velas, M. Spanel, and A. Herout, "Collar line segments for fast odometry estimation from velodyne point clouds," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2016, pp. 4486–4495.

[4] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2014, pp. 15–22.

[5] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PP, no. 99, pp. 1–1, 2017.

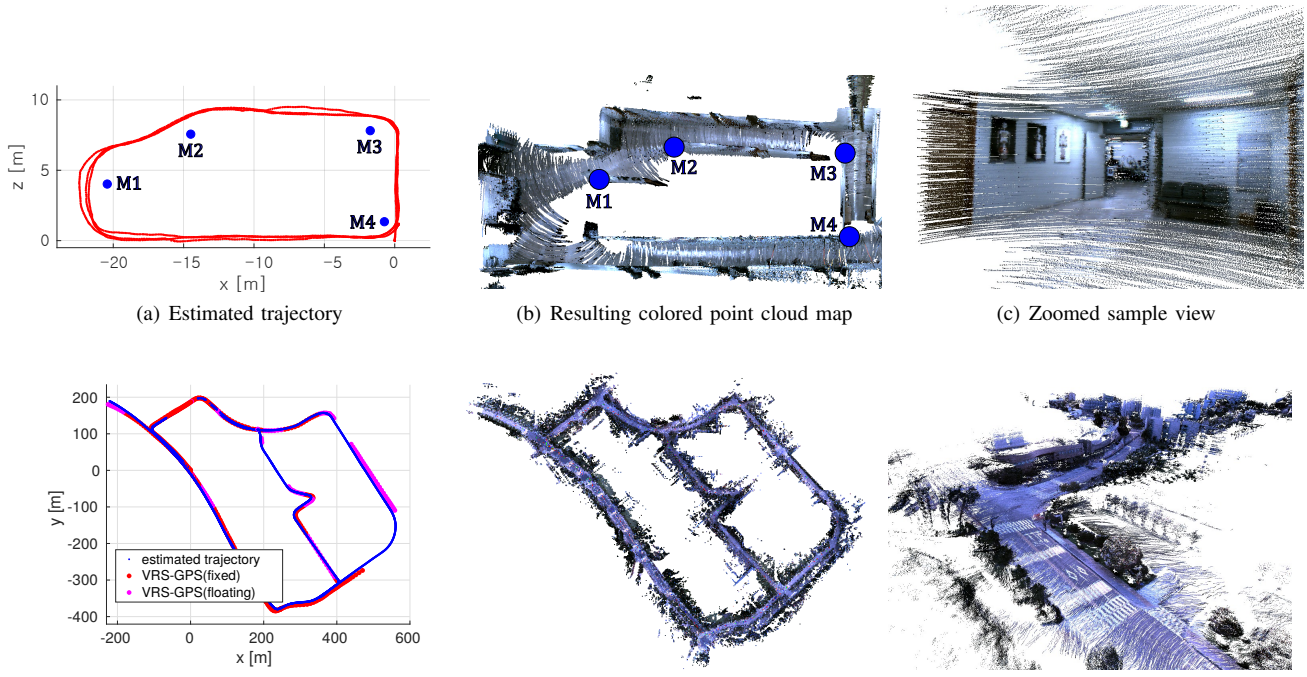
[6] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "DTAM: Dense tracking and mapping in real-time," in *Proceedings of the IEEE International Conference on Computer Vision*, 2011, pp. 2320–2327.

[7] C. Kerl, J. Sturm, and D. Cremers, "Robust odometry estimation for RGB-D cameras," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2013, pp. 3748–3754.

[8] F. Steinbrcker, J. Sturm, and D. Cremers, "Real-time visual odometry from dense RGB-D images," in *Proceedings of the IEEE International Conference on Computer Vision*, 2011, pp. 719–722.

[9] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *Proceedings of the European Conference on Computer Vision*, 2014.

[10] S. Gauglitz, C. Sweeney, J. Ventura, M. Turk, and T. Höllerer, "Live tracking and mapping from both general and rotation-only camera motion," in *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality*, Atlanta, Georgia, 2012, pp. 13–22.



(a) Estimated trajectory (b) Resulting colored point cloud map (c) Zoomed sample view

(d) Estimated trajectory (e) Resulting colored point cloud map (f) Zoomed sample view

Fig. 9. Results of the indoor and the outdoor experiment. The experiment was performed by turning around the same hall three times. (a) indicate the estimated trajectory result and marker position. (b) shows resulting acquired point cloud map without rendering. (d) Both ground truth trajectory and the estimated trajectory are overlaid for comparison. Blue dot represents the estimated trajectory, and red and magenta represent the position measured by the VRS-GPS in the fixed and floating states. (e) Resulting point cloud map from the outdoor test.

[11] J. Civera, A. J. Davison, and J. M. M. Montiel, "Inverse depth parametrization for monocular slam," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 932–945, 2008.

[12] B. Kitt, A. Geiger, and H. Lategahn, "Visual odometry based on stereo image sequences with ransac-based outlier rejection scheme," in *Proceedings of the IEEE Intelligent Vehicle Symposium*, 2010.

[13] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality*, 2007, pp. 225–234.

[14] H. Strasdat, A. J. Davison, J. M. M. Montiel, and K. Konolige, "Double window optimisation for constant time visual slam," in *Proceedings of the IEEE International Conference on Computer Vision*, 2011, pp. 2352–2359.

[15] R. Mur-Artal, J. M. M. Montiel, and J. D. Tards, "ORB-SLAM: A versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[16] M. Pizzoli, C. Forster, and D. Scaramuzza, "REMODE: Probabilistic, monocular dense reconstruction in real time," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2014, pp. 2609–2616.

[17] T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J. J. Leonard, and J. McDonald, "Real-time large-scale dense RGB-D slam with volumetric fusion," *International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 598–626, 2015.

[18] J. Zhang, M. Kaess, and S. Singh, "A real-time method for depth enhanced visual odometry," *Autonomous Robots*, vol. 41, no. 1, pp. 31–43, 2017.

[19] R. Kuemmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A general framework for graph optimization," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Shanghai, China, 2011, pp. 3607–3613.

[20] D. Gálvez-López and J. D. Tardós, "Bags of binary words for fast place recognition in image sequences," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, 2012.

[21] P. Agarwal, G. D. Tipaldi, L. Spinello, C. Stachniss, and W. Burgard, "Robust map optimization using dynamic covariance scaling," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2013, pp. 62–69.

[22] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the KITTI vision benchmark suite," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[23] A. Kassir and T. Peynot, "Reliable automatic camera-laser calibration," in *Australasian Conference on Robotics and Automation (ACRA)*, G. Wyeth and B. Upcroft, Eds. Brisbane, Queensland: ARAA, 2010.

[24] J. Kim, J. Jeong, Y.-S. Shin, Y. Cho, H. Roh, and A. Kim, "LiDAR configuration comparison for urban mapping system," in *Proceedings of the International Conference on Ubiquitous Robots and Ambient Intelligence*, Jeju, S. Korea, 2017, pp. 854–857.

[25] S. Garrido-Jurado, R. M. noz Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280 – 2292, 2014.

[26] J. Zhang and S. Singh, "Visual-lidar odometry and mapping: low-drift, robust, and fast," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2015, pp. 2174–2181.