Privacy Preserving Visual SLAM

Mikiya Shibuya* [1,2], Shinya Sumikura* [1], and Ken Sakurada* [1]

[1] National Institute of Advanced Industrial Science and Technology (AIST)
[2] Tokyo Institute of Technology
{mikiya-shibuya,sumikura.shinya,k.sakurada}@aist.go.jp
shibuya@m.titech.ac.jp

Abstract. This study proposes a privacy-preserving Visual SLAM framework for estimating camera poses and performing bundle adjustment with mixed line and point clouds in real time. Previous studies have proposed localization methods to estimate a camera pose using a line-cloud map for a single image or a reconstructed point cloud. These methods offer a scene privacy protection against the inversion attacks by converting a point cloud to a line cloud, which reconstruct the scene images from the point cloud. However, they are not directly applicable to a video sequence because they do not address computational efficiency. This is a critical issue to solve for estimating camera poses and performing bundle adjustment with mixed line and point clouds in real time. Moreover, there has been no study on a method to optimize a line-cloud map of a server with a point cloud reconstructed from a client video because any observation points on the image coordinates are not available to prevent the inversion attacks, namely the reversibility of the 3D lines. The experimental results with synthetic and real data show that our Visual SLAM framework achieves the intended privacy-preserving formation and real-time performance using a line-cloud map.

Keywords: Visual SLAM, privacy, line cloud, point cloud

1 Introduction

Localization and mapping from images are fundamental problems in the field of computer vision. They have been exhaustively studied for robotics and augmented/mixed reality (AR/MR) [5, 20, 31]. These applications are divided into three main types, where the 6 degree-of-freedom (DOF) camera pose is: (i) in unmeasured regions to be estimated simultaneously with the 3D map through either Structure from Motion (SfM) [1, 4] or Visual SLAM [10, 11, 24]; (ii) in measured regions to be estimated by solving 2D–3D matching between the image and the 3D map and (iii) in both of measured and unmeasured regions, a camera passes through the entire regions. Because of the complexity of this field in computer vision, this study focuses on the literature regarding the applications of (ii) and (iii).

* The authors assert equal contribution and joint first authorship.

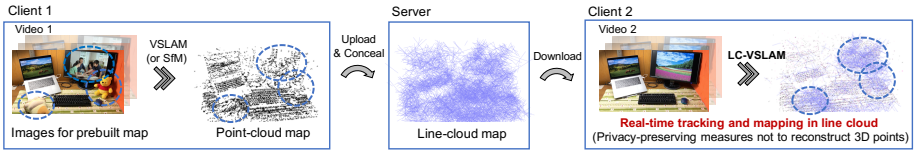


Fig. 1. Example of LC-VSLAM application.

Recent studies have revealed a risk of privacy preservation that 3D points and their descriptors can be inverted to synthesize the original scene images [27]. To prevent this privacy risk, Speciale *et al.* proposed a privacy-preserving method which converts a 3D point cloud to a 3D line cloud to make the inversion attack difficult [33, 34]. However, in the case of camera pose estimation of a single image, the problem after the conversion changes from three 2D point–3D point correspondences (p3P) [12, 17, 29] to six 2D point–3D line correspondences (p6L) [33], which causes the amount of computation to increase and the accuracy to deteriorate. Moreover, for the corresponding search with 2D points, the computational cost and the matching error ratio for a 3D line are higher than those for a 3D point. Hence, it is difficult to directly apply the localization method with p6L to a real-time application with a video sequence, such as Visual SLAM.

For SfM and Visual SLAM, bundle adjustment (BA) is utilized to optimize camera pose and 3D points [40, 42]. In a standard BA, the parameters are optimized by minimizing the error function with distances between the reprojected points and the corresponding 2D points. However, there are two new problems for BA with regard to a line-cloud map from a server. First, BA for the line-cloud map demands an additional definition of every new error function between a 2D point on a client image and the corresponding 3D line from a server. Second, to ensure the irreversibility of a line cloud to the original point cloud, it is inevitable to integrate the line cloud with the point cloud and to globally optimize them without the 2D point coordinates on the keyframes of the line cloud.

To overcome these difficulties, we propose a Visual SLAM framework for real-time relocalization, tracking, and BA with a map mixed with lines and points (Fig. 1 and 2), which we call *Line-Cloud Visual SLAM* (LC-VSLAM). The main contributions of this study are three-fold.

- Efficient relocalization and tracking with 3D points reconstructed by Visual SLAM of a client.
- Motion-only, rigid-stereo, local, and global bundle adjustments for mixed line and point clouds.
- Creation of unified framework for various types of projection models, such as perspective, fisheye, and equirectangular.

First, matching between local 3D points reconstructed with Visual SLAM by a client and a line cloud enables fast and accurate relocalization (Sec. 3.2). Moreover, discretizing the 3D line to 3D points speeds up the 2D–3D matching to achieve real-time tracking (Sec. 3.3).

Second, we propose four types of bundle adjustments for mixed line and point clouds, motion-only, rigid-stereo, local, and global BAs, depending on the

optimization parameters. The 3D lines are simultaneously optimized with the camera poses and 3D points by defining the covariance of the 3D line with that of the original 3D point, whose value in the direction of the line is infinite. The covariance is used to calculate the reprojection error between the 3D line and the corresponding 2D point (or line). In the global BA, a whole map which has already included a line cloud from a server is optimized by adding the virtual observations of 3D lines on the line-cloud keyframes (Sec. 3.4).

Finally, we propose a unified framework that can be applied to various types of projection models by reason of the matching efficiency, where 3D lines are discretized to 3D points (Sec. 3.3). The reprojection error between the 3D line and the virtual observation is defined as the difference between the normal vectors of the planes consisting of the lines and the origin of the local camera coordinates (Sec. 3.4).

In Section 2, we summarize the related work. In Section 3, we explain the details of the proposed framework. In Section 4, we present the experimental results. Finally, in Section 5, we present our conclusions.

2 Related Works

2.1 Visual SLAM

Visual SLAM is broadly utilized for environment mapping, localization in robotics, and camera tracking frameworks in AR/MR applications. The Visual SLAM algorithms are generally divided into three kinds of methods: feature-based [5, 20, 24, 25], direct [10, 11, 26], and learning-based [6, 38, 39, 43–45]. The feature-based methods pertain to camera tracking and scene mapping with feature points extracted from images [2, 3, 23, 30]. The direct methods, in contrast, focus on minimization of photometric errors indicating the difference of the intensity between two frames.

Recently, a combination of Convolutional Neural Networks (CNNs) and either of the aforementioned kinds of algorithms (feature-based or direct) has been under extensive investigation. The feature-based methods use CNN-based architectures in conventional algorithms to detect and describe their feature points [6, 38, 44]. For the direct methods, CNN-based depth prediction techniques are utilized for the initialization of depth estimation [39, 43]. As opposed to the fusion of conventional Visual SLAM and learning-based methods, end-to-end tracking and mapping methods based on Deep Neural Networks (DNNs) have been recently studied [45].

The feature-based methods can localize frames in a prebuilt map quickly and accurately [14, 32]. These characteristics are required for our LC-VSLAM to localize camera pose against a prebuilt 3D line cloud, to track camera trajectory, and to simultaneously expand the map. Therefore, we constructed the LC-VSLAM algorithm based on the feature-based Visual SLAM.

2.2 Map Representation with Line Cloud

In conventional AR/MR applications, each client downloads a prebuilt map created by other clients from a server and performs localization/tracking based on the map. In this case, the clients share only the 3D point cloud and its optional attributes (e.g., color, descriptor, and visibility of each point and camera poses). However, Pittaluga *et al.* proved that fine images at arbitrary viewpoints can be restored only with the sparse point cloud and its optional attributes [27]. They referred to this restoration as an inversion attack.

To address this problem, Speciale *et al.* proposed a map representation based on a 3D line cloud [33]. They also formulated a method to localize an image in the prebuilt line cloud. The line cloud is built by converting each 3D point to a 3D line that has a random orientation and passes through the original point. It is quite difficult to directly restore the original point cloud from the line cloud because the point coordinates can be reparameterized arbitrarily on the corresponding line.

To the best of our knowledge, there has been no study on how to track camera poses continuously in real time with a 3D line cloud. As a straightforward method, the camera pose of every frame can be estimated successively using the p6L solver proposed in [33]. However, the p6L solver has a much larger computational cost than the typical p3P solvers [12, 17, 29]. In contrast, some methods achieve Visual SLAM using edges in a scene as landmarks like feature points [7, 19, 28]. These methods build 3D lines based on the structure and color distribution in a scene. That is, the 3D lines explicitly represent the scene structures. Our method, however, to be resistant against inversion attacks, avoids such explicitness by utilizing the randomly oriented 3D line cloud.

2.3 Bundle Adjustment for Map Optimization

Conventional Visual SLAM and SfM methods largely utilize pose graph optimization (PGO) [16, 21, 35] and bundle adjustment (BA) [22, 40, 42] for accurate pose estimation and map construction. PGO can be almost directly applied to loop closure for a line cloud, but the conventional BA cannot. This is because a reprojection error that constrains a 3D line and a 2D feature point has not been defined. For the Visual SLAM methods based on structural edges, point-to-line distances between a 2D line and two endpoints of a reprojected 3D line segment are used as a reprojection error between 2D and 3D lines [7, 28]. However, this formulation targets the structure-based edges, i.e., the 3D lines which explicitly represent the scene structures; thus, they cannot be applied to tracking and mapping with a prebuilt map of randomly oriented 3D lines.

We therefore propose a reprojection constraint between randomly oriented 3D lines and 2D feature points in order to conduct BA for the map representation with mixed line and point clouds. In its formulation, an error ellipse of each 3D line is decided according to the covariance of the corresponding 3D point before being converted to the 3D line. The error ellipse of the 3D line has not been considered in the previous study [33]. Furthermore, the proposed algorithms do

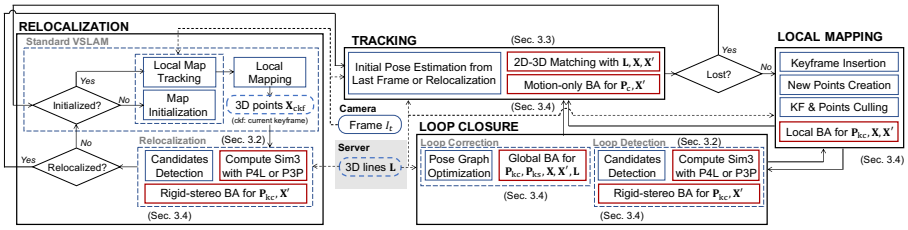


Fig. 2. Overview of LC-VSLAM system. It should be noted here that the three threads run in parallel: tracking, local mapping, and loop closure.

not depend on the difference of projection models. Hence, the real-time LC-VSLAM framework can be realized in various types of projection models.

3 Proposed Method

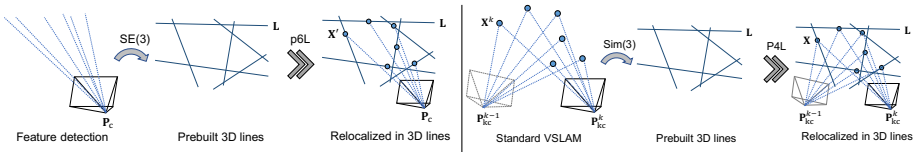
3.1 System Overview

The proposed LC-VSLAM system consists of four modules: relocalization, tracking, local mapping, and loop closure (Fig. 2). The system works to estimate the parameters set for the following camera poses and 3D mapping:

(1) Camera pose of the current client frame $\mathbf{P}_{\mathrm{c}}$,
(2) Camera poses of client keyframes $\mathbf{P}_{\mathrm{kc}}$,
(3) Camera poses of server keyframes for 3D lines $\mathbf{P}_{\mathrm{ks}}$,
(4) 3D lines $\mathbf{L}$,
(5) 3D points reconstructed only from 2D points $\mathbf{X}$,
(6) 3D points reconstructed from 3D lines and 2D points $\mathbf{X}'$.

First, for relocalization in a line cloud, the system performs a standard Visual SLAM [24, 45] for video sequence input $I_{1:t}$ to reconstruct the local 3D points of the current keyframe $\mathbf{X}_{\mathrm{ckf}}$ (Sec. 3.2). The camera poses of the keyframes in the line cloud $\mathbf{P}_{\mathrm{kc}}$ are calculated with $\mathbf{X}_{\mathrm{ckf}}$ by computing Sim(3) with four 3D point3D line (P4L) [37] or three 3D point3D point (P3P) [9, 18, 41] correspondences after the candidate detection based on DBOW [14]. Then, the camera poses $\mathbf{P}_{\mathrm{kc}}$ and the reconstructed 3D points $\mathbf{X}'$ are optimized in the rigid-stereo bundle adjustment (Sec. 3.4). The loop detection performs a similar processing operation (Sec. 3.2). After the relocalization in the line cloud, the other three LC-VSLAM modules (tracking, local mapping, and loop closure) start.

The tracking module continuously estimates the camera pose for the current frame. The tentative camera pose is estimated by assuming a linear motion of the camera. In the 2D point3D line matching, 3D lines are discretized to 3D points to improve the computational efficiency (Sec. 3.3). Using all the correspondences of the 2D point3D point and the 2D point3D line, the motion-only bundle adjustment optimizes the camera pose of the current frame $\mathbf{P}_{\mathrm{c}}$ and the reconstructed 3D points $\mathbf{X}'$ (Sec. 3.4).



(a) Single shot (b) Video sequence (ours)

Fig. 3. Overview of relocalization and loop detection with a line cloud.

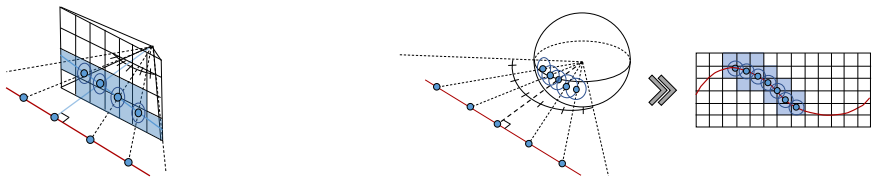
In the local mapping module, 3D points $\mathbf{X}, \mathbf{X}'$ are newly created or restored using the keyframes with the camera pose estimated in the tracking module, according to the 2D point2D point and the 2D point3D line correspondences. Subsequently, the local bundle adjustment optimizes the camera poses of the client keyframes $\mathbf{P}_{\mathrm{kc}}$ and the reconstructed 3D points $\mathbf{X}, \mathbf{X}'$ simultaneously (Sec. 3.4).

For correcting errors of the 3D lines and points, the loop-closure module detects the loops in the same manner as relocalization. After the pose graph optimization [16, 35], the global bundle adjustment optimizes all of the parameters for the map $\mathbf{P}_{\mathrm{kc}}, \mathbf{P}_{\mathrm{ks}}, \mathbf{X}, \mathbf{X}', \mathbf{L}$ by introducing the virtual observations of the 3D lines on the line-cloud keyframes (Sec. 3.4).

3.2 Relocalization and Loop Detection with a Line Cloud

In this study, we assume that the visibility of 3D lines $\mathbf{L}$ from a server for the keyframes $\mathbf{P}_{\mathrm{ks}}$ is known. Hence, for the global localization problem using a line-cloud, we utilize a bag-of-words strategy such as DBOW [14], in the same manner as in a standard Visual SLAM [24, 45] to efficiently detect loop candidates. After the loop candidate detection, the geometric verification with a RANSAC-based solver rejects the outliers of their descriptor matches [13]. As shown in Figure 3(a), the increase in the computational cost of the p6L solver [33], compared to that of the typical p3P solvers, prevents real-time processing due to requiring more points to solve a minimal problem [12, 17, 29]. Therefore, we utilize local 3D points of the current keyframe $\mathbf{X}_{\mathrm{ckf}}$, which are reconstructed by a standard Visual SLAM of a client, to match with the 3D lines $\mathbf{L}$ [Fig. 3(b)]. More concretely, we utilize four 3D point-3D line correspondences (P4L) to calculate the relative Sim(3) pose $\Delta\mathbf{P}_{\mathrm{kc}}^{\mathrm{Sim3}}$. The P4L solver [37] is more efficient than the p6L one. (The p6L solver cannot be directly applied to the Sim(3) estimation for the scale drift-aware loop closure [35].)

In cases where 3D points have already been reconstructed in the line-cloud map (e.g., relocalization after tracking loss and loop detection after exploring the line cloud), both 3D lines and points are utilized for the Sim(3) estimation. To be more precise, we use P4L if $N_{\mathrm{PL}}/N_{\mathrm{PP}} > 4/3$ and P3P otherwise, where N_{PL} and N_{PP} represent the numbers of 3D point-3D line and the 3D point-3D point correspondences, respectively. After the initial estimation, the pose graph optimization is conducted with the relative camera pose [16, 35], and the rigid-



(a) Perspective projection model (b) Equirectangular projection model

Fig. 4. Overview of matching between 2D points and discretized 3D lines.

stereo BA optimizes the camera pose $\mathbf{P}_{\mathrm{kc}}^{\mathrm{ckf}}$ and the reconstructed 3D points X' (Sec. 3.4).

3.3 2D–3D Matching with 3D Lines and Points

Real-time tracking with a line cloud requires a fast search between corresponding 2D points and 3D lines as well as between 2D and 3D points for standard feature-based methods [24]. However, especially for the equirectangular projection model, efficient search is difficult because the reprojected 3D lines are not straight to correspond the 3D points in the image coordinates. Hence, in our system, a 3D line is discretized to 3D points, and they are reprojected onto the image. This discretization strategy brings about an advantage of efficient search for corresponding 2D points that narrows search ranges of distances and image coordinates. Moreover, this method can be directly applied to various types of projection models, such as the perspective and equirectangular models. Figure 4 shows the overview of this 2D point-3D line matching.

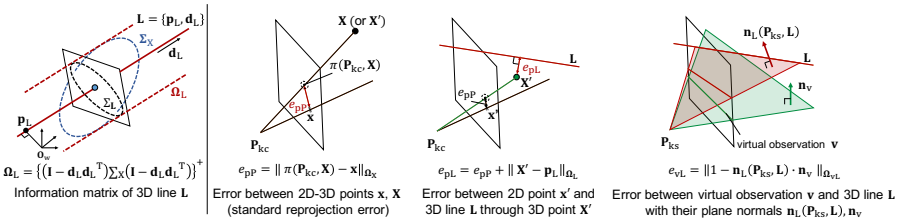
3.4 Bundle Adjustments with a Line Cloud

To achieve bundle adjustments with a line cloud, first we define the information matrix of a 3D line with the covariance matrix of the original 3D point. Next, we also define error metrics between a 2D point (or line) and a 3D line. Finally, utilizing the error metrics, we introduce new error functions for each bundle adjustment.

Definition: A prebuilt map contains the 3D lines $\mathbf{L} = \{\mathbf{p}_{\mathrm{L}}, \mathbf{d}_{\mathrm{L}}\}$. They are converted from the 3D points $\mathbf{X}_{\mathrm{L}}$, whose covariance matrix is defined as $\mathbf{\Sigma}_{\mathrm{X_L}}$. The vectors $\mathbf{p}_{\mathrm{L}}, \mathbf{d}_{\mathrm{L}}$ represent the base point and the directional vector, respectively. To conceal the information regarding the coordinates in the direction $\mathbf{d}$ of the original 3D points $\mathbf{X}_{\mathrm{L}}$, we introduce an information matrix of the 3D line $\mathbf{L}$:

$$\mathbf{\Omega}_{\mathrm{L}} = \{(\mathbf{I} - \mathbf{d}\mathbf{d}^{\mathsf{T}})\mathbf{\Sigma}_{\mathrm{X_L}}(\mathbf{I} - \mathbf{d}\mathbf{d}^{\mathsf{T}})\}^{+}, \tag{1}$$

where $\mathbf{A}^{+}$ is the pseudo-inverse matrix of $\mathbf{A}$ [Fig. 5(a)]. The information value of $\mathbf{\Omega}_{\mathrm{L}}$ in the direction $\mathbf{d}$ is zero.



(a) (b) (c) (d)

Fig. 5. Information matrix of a 3D line and error metrics for bundle adjustments. (a) information matrix of 3D line $\mathbf{L}$, (b) error between 2D-3D points $\mathbf{x}$, $\mathbf{X}$, (c) error between 3D point $\mathbf{x}'$ and 3D line $\mathbf{L}$ through 3D point $\mathbf{X}'$ and (d) error between virtual observation $\mathbf{v}$ and 3D line $\mathbf{L}$ with their plane normals $\mathbf{n}_{\mathrm{L}}(\mathbf{P}_{\mathrm{ks}}, \mathbf{L})$, $\mathbf{n}_{\mathrm{v}}$.

In a standard BA [Fig. 5(b)], a reprojection error for optimizing camera poses and 3D points is defined as

$$e_{\mathrm{pP}}(\mathbf{P}_{\mathrm{kc}}, \mathbf{X}, \mathbf{x}) \coloneqq \|\pi(\mathbf{P}_{\mathrm{kc}}, \mathbf{X}) - \mathbf{x}\|^2_{\mathbf{\Omega}_{\mathrm{X}}}, \tag{2}$$

where $\pi(\cdot)$ is the projection function, $\mathbf{x}$ is the observation points from which the 3D points $\mathbf{X}$ are reconstructed, $\mathbf{\Omega}_{\mathrm{X}} = \mathbf{\Sigma}_{\mathrm{X}}^{-1}$, and $\|\mathbf{e}\|^2_{\mathbf{\Omega}_{\mathrm{X}}} = \mathbf{e}^{\mathsf{T}}\mathbf{\Omega}_{\mathrm{X}}\mathbf{e}$.

BA with a line cloud, however, requires an error metric between a 2D point $\mathbf{x}'$ and a 3D line $\mathbf{L}$, where $\mathbf{x}'$ is the observation points from which the 3D points $\mathbf{X}'$ are reconstructed. Hence, as shown in Figure 5(c), we define the error metric using the 3D point $\mathbf{X}'$, which is initially reconstructed as the intermediate point between the viewing direction of $\mathbf{x}'$ and $\mathbf{L}$, as

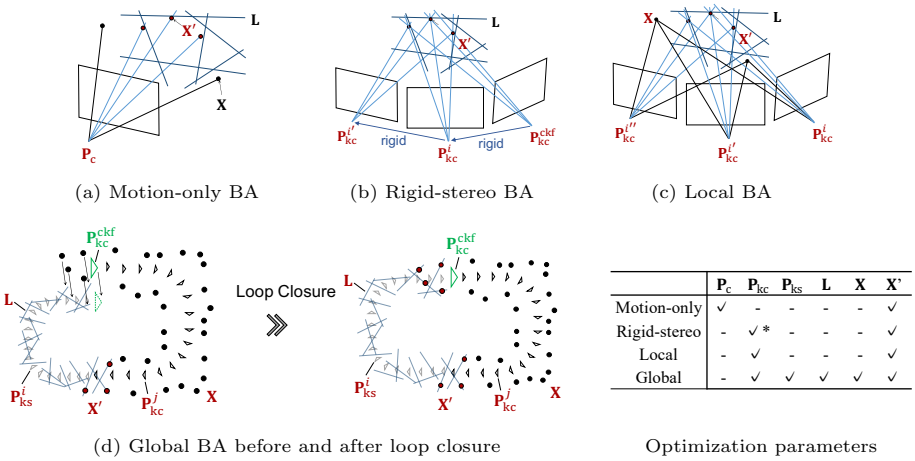
$$e_{\mathrm{pL}}(\mathbf{P}_{\mathrm{kc}}, \mathbf{X}', \mathbf{x}', \mathbf{p}_{\mathrm{L}}) \coloneqq e_{\mathrm{pP}}(\mathbf{P}_{\mathrm{kc}}, \mathbf{X}', \mathbf{x}') + \|\mathbf{X}' - \mathbf{p}_{\mathrm{L}}\|^2_{\mathbf{\Omega}_{\mathrm{L}}}. \tag{3}$$

The first term of Eq. (3) is the standard reprojection error, which is the constraint between the 2D point $\mathbf{x}'$ and the reconstructed 3D point $\mathbf{X}'$, while the second term is the constraint between the 3D point $\mathbf{X}'$ and the 3D line $\mathbf{L}$. Through this error metric, the 3D line $\mathbf{L}$ and the camera pose $\mathbf{P}_{\mathrm{kc}}$ can constrain each other.

Furthermore, the prebuilt line-cloud map may contain errors such as scale drift, which additional observations by other clients can correct. However, observation points on the image coordinates should be dropped when a user uploads a line cloud to a server because they can recover the corresponding 3D points. As a result, there is no constraint between the 3D lines $\mathbf{L}$ and their keyframe camera poses $\mathbf{P}_{\mathrm{ks}}$ for a BA.

Here, we introduce the virtual observation $\mathbf{v}$, which is the projection of the initial 3D line $\mathbf{L}_0$ onto the keyframe. Strictly speaking, $\mathbf{v}$ is represented by the normal vector $\mathbf{n}_{\mathrm{v}}$ of the plane defined by the camera center and $\mathbf{L}_0$. Similarly, for the current state, the normal vector of the plane defined by the camera center and the 3D line $\mathbf{L}$ is represented as $\mathbf{n}_{\mathrm{L}}(\mathbf{P}_{\mathrm{ks}}, \mathbf{L})$. We define the error metric between the 3D lines $\mathbf{L}$ and the virtual observation $\mathbf{v}$ with their normals as

$$e_{\mathrm{vL}}(\mathbf{P}_{\mathrm{ks}}, \mathbf{L}, \mathbf{v}) \coloneqq \|1 - \mathbf{n}_{\mathrm{L}}(\mathbf{P}_{\mathrm{ks}}, \mathbf{L}) \cdot \mathbf{n}_{\mathrm{v}}\|^2_{\Omega_{\mathrm{vL}}}. \tag{4}$$



(a) Motion-only BA (b) Rigid-stereo BA (c) Local BA

(d) Global BA before and after loop closure

	$\mathbf{P}_c$	$\mathbf{P}_{kc}$	$\mathbf{P}_{ks}$	$\mathbf{L}$	$\mathbf{X}$	$\mathbf{X}'$
Motion-only	✓	-	-	-	-	✓
Rigid-stereo	-	✓*	-	-	-	✓
Local	-	✓	-	-	-	✓
Global	-	✓	✓	✓	✓	✓

Optimization parameters

Fig. 6. Optimization parameters of BAs for: (a) motion-only, (b) rigid-stereo, (c) local, and (d) global (in red). * Only the rigid-stereo BA is set under the rigid constraint on the relative camera poses between keyframes.

It should be noted that Eq. (4) is directly applicable to other projection models, such as the equirectangular model, because it is defined in the local camera coordinates.

Figure 6 shows the optimization parameters in each BA with a line cloud: (a) Motion-only BA, (b) Rigid-stereo BA, (c) Local BA and (d) Global BA. LC-VSLAM, as the standard Visual SLAM, reconstructs 3D points as $\mathbf{X}$, which are corresponded with 3D lines $\mathbf{L}$ in relocalization and loop detection. The corresponded 3D points $\mathbf{X}$ are identified as $\mathbf{X}'$ for the optimization. Utilizing the above error metrics, the four bundle adjustments with a line cloud are defined as follows.

Motion-only BA: A tracking thread estimates the camera pose for each input frame. For real-time tracking, as shown in Figure 6(a), the motion-only BA optimizes only the camera pose of the current frame $\mathbf{P}_c$ and the reconstructed 3D points $\mathbf{X}'$ with fixed 3D points $\mathbf{X}$ and lines $\mathbf{L}$ as

$$\mathbf{P}_c^*, \mathbf{X}'^* = \underset{\mathbf{P}_c^*, \mathbf{X}'^*}{\arg\min} \sum_j e_{\mathrm{pP}}(\mathbf{P}_c, \mathbf{X}^j, \mathbf{x}^j) + \sum_k e_{\mathrm{pL}}(\mathbf{P}_c, \mathbf{X}'^k, \mathbf{x}'^k, \mathbf{p}_{\mathrm{L}}^k), \tag{5}$$

where j, k indicate the indices of the 3D points $\mathbf{X}, \mathbf{X}'$ which are visible from the current frame, respectively. The first term pertains to the constraints between the camera pose of the current keyframe $\mathbf{P}_c$ and the 3D points $\mathbf{X}'$ that have been already reconstructed with the previous frames. The second one refers to those between the camera pose $\mathbf{P}_c$ and the 3D lines $\mathbf{L}$.

Rigid-stereo BA: For relocalization and loop detection, the local 3D points $\mathbf{X}$, which are merged as $\mathbf{X}'$ after matching with the 3D lines $\mathbf{L}$, have already

been reconstructed and locally optimized with the local keyframes. As shown in Figure 6 (b), the rigid-stereo BA can optimize the camera pose of the current keyframe $\mathbf{P}_{\mathrm{kc}}^{\mathrm{ckf}}$ and the 3D points $\mathbf{X}'$ as

$$\mathbf{P}_{\mathrm{kc}}^{\mathrm{ckf}*}, \mathbf{X}'^{*} = \underset{\mathbf{P}_{\mathrm{kc}}^{\mathrm{ckf}}, \mathbf{X}'}{\arg\min} \sum_i \sum_j e_{\mathrm{pL}}(\Delta\mathbf{P}^i \mathbf{P}_{\mathrm{kc}}^{\mathrm{ckf}}, \mathbf{X}'^j, \mathbf{x}'^{i,j}, \mathbf{p}_{\mathrm{L}}^j) \tag{6}$$

where $\Delta\mathbf{P}^i = \text{const.}$ is the relative camera pose between the current keyframe and the i-th neighboring keyframe which shares the 3D points ($\Delta\mathbf{P}^i = \mathbf{I}$ if $i = \mathrm{ckf}$).

Local BA: The rigid-stereo BA is a special case of the local BA. In the local mapping thread, new keyframes of the client $\mathbf{P}_{\mathrm{kc}}$ are inserted, and the 3D points $\mathbf{X}$ are newly reconstructed from only their 2D points. Hence, as shown in Figure 6 (c), the camera pose of the local keyframes $\mathbf{P}_{\mathrm{kc}}$ and their 3D points $\mathbf{X}$ and $\mathbf{X}'$ are optimized as

$$\begin{aligned} \mathbf{P}_{\mathrm{kc}}^{*}, \mathbf{X}^{*}, \mathbf{X}'^{*} = & \underset{\mathbf{P}_{\mathrm{kc}}, \mathbf{X}, \mathbf{X}'}{\arg\min} \sum_i \sum_j e_{\mathrm{pP}}(\mathbf{P}_{\mathrm{kc}}^i, \mathbf{X}^j, \mathbf{x}^{i,j}) \\ & + \sum_i \sum_k e_{\mathrm{pL}}(\mathbf{P}_{\mathrm{kc}}^i, \mathbf{X}'^k, \mathbf{x}'^{i,k}, \mathbf{p}_{\mathrm{L}}^k). \end{aligned} \tag{7}$$

Global BA: After loop detection and pose graph optimization [16, 35], the camera poses and 3D structures, which include the 3D lines $\mathbf{L}$ and the camera poses of their keyframes $\mathbf{P}_{\mathrm{ks}}$, are globally optimized with the virtual observation $\mathbf{v}$ and its error metric e_{vL} as

$$\begin{aligned} \mathbf{P}_{\mathrm{kc}}^{*}, \mathbf{P}_{\mathrm{ks}}^{*}, \mathbf{X}^{*}, \mathbf{X}'^{*}, \mathbf{L}^{*} = & \underset{\mathbf{P}_{\mathrm{kc}}, \mathbf{P}_{\mathrm{ks}}, \mathbf{X}, \mathbf{X}', \mathbf{L}}{\arg\min} \sum_i \sum_j e_{\mathrm{pP}}(\mathbf{P}_{\mathrm{kc}}^i, \mathbf{X}^j, \mathbf{x}^{i,j}) \\ & + \sum_i \sum_k e_{\mathrm{pL}}(\mathbf{P}_{\mathrm{kc}}^i, \mathbf{X}'^k, \mathbf{x}'^{i,k}, \mathbf{p}_{\mathrm{L}}^k) + \sum_{i'} \sum_l e_{\mathrm{vL}}(\mathbf{P}_{\mathrm{ks}}^{i'}, \mathbf{L}^l, \mathbf{v}^{i',l}), \end{aligned} \tag{8}$$

where i, i' are the indices of all client and server keframes, respectively, and l is the index of the 3D line $\mathbf{L}$.

4 Experiments

4.1 Experimental Setting

The performance of LC-VSLAM was tested to quantitatively and qualitatively evaluate from multiple perspectives (see the algorithm in Sec. 3). We have carried out all experiments with a Core i9-9900K (8 cores @ 3.60GHz) with a 64 GB RAM. Considering its practical usability, we evaluated the performance from the following viewpoints.

Tracking time: We evaluated the tracking time of each frame to confirm the real-time performance of the proposed framework. Based on the mean tracking time, we compared LC-VSLAM with p6L, which applies a single-shot localization algorithm in the 3D line cloud to every frame [33].

Accuracy of camera pose estimation: We evaluated the accuracy of camera poses after a local map was registered to a prebuilt map for LC-VSLAM and the previous method. First, a local 3D point-cloud map was created by a client, and geometrically registered to a global 3D line-cloud map downloaded from the server using the estimated 3D transformation between them. After the registration, the camera pose accuracy was evaluated using the latest keyframe of the registered local map in the coordinates system of the global map.

Comparison to a conventional Visual SLAM system: The foregoing perspectives were evaluated run on the synthetic dataset generated by the CARLA Simulator [8] and the real image dataset KITTI [15]. We also compared three camera types (perspective, fisheye, and equirectangular) in the evaluation to confirm that the proposed algorithms work well for various types of projection models.

4.2 Implementation Details

We implemented the LC-VSLAM system based on OpenVSLAM [36] by integrating the four dedicated modules as follows: (i) add the data structure of a line, such as line direction and covariance, (ii) add the P4L solver and the rigid-stereo BA [Eq. (6)] to the modules of the loop detector and the relocalizer, (iii) adapt the one-to-many feature matching to the many-to-many one, and (iv) replace the cost functions in the motion-only, local, and global BAs with those of [Eqs. (5), (7), and (8)].

4.3 Dataset and Prebulit Map Creation

All the evaluation was performed on our new CARLA dataset because there are no publicly available benchmarks for evaluating LC-VSLAM. The dataset should satisfy the following two requirements to evaluate the effectiveness of LC-VSLAM: (i) a sequence pair contains sufficient overlaps and loops to allocate a sequence for a prebuilt map and the other for an input of LC-VSLAM to evaluate the tracking and the loop closure, and (ii) image sequences of various types of camera models are available to evaluate the versatility on projection models. KITTI camera stereo dataset [15] is one of the publicly available datasets for evaluating accuracy of Visual SLAM systems and meets the requirement (i). However, in the dataset, the baseline is very short, and the image pairs are synchronized, which make tracking too easy. The KITTI dataset contains only perspective projection images, and thus does not meet the requirement (ii). Therefore, we performed all quantitative evaluations on our new CARLA datataset, and verified the LC-VSLAM's applicability to real image datatasets on the KITTI stereo dataset. Our *Desk* and *Campus* datasets were used to qualitatively evaluate the effectiveness of the LC-VSLAM on real scenes.

Table 1. Tracking time of each frame, mean absolute pose errors (APE) for translation [m] and rotation [deg] of the single-shot localization by p6L [33] and LC-VSLAM. The image resolution is 640×360.

	Tracking time [ms]	APE for trans. [m] / rot. [deg]
p6L [33]	140.3	0.7815 / 0.5896
LC-VSLAM (ours)	**31.09**	**0.1979** / **0.2841**

Table 2. Mean absolute pose errors (APE) for translation [m] and rotation [deg] of synthetic images by the CARLA simulator and KITTI dataset. Lower is better.

	CARLA			KITTI
Prebuilt map	Perspective	Fisheye	Equirectangular	Perspective
3D points	3.290 / 0.6273	2.883 / 0.4402	3.079 / 0.2375	3.801 / 1.012
3D lines (ours)	3.651 / 0.8416	3.177 / 0.5941	3.075 / 0.2766	4.488 / 1.309

CARLA Dataset: We used the CARLA simulator [8] to create a dataset for evaluating the accuracy of our tracking and bundle adjustment methods, which utilize a line cloud as a prebuilt map (see the supplementary material for details). The simulator allows synthesis of photo-realistic images with the camera poses of outdoor scenes. Hence, to evaluate the tracking time, we generated an image sequence pair (#01) which almost overlaps each other with small displacements. Additionally, we created eight pairs of mid-scale image sequences (#02-09) for each camera type to evaluate the localization accuracy and created three pairs of large-scale image sequences with loop-closure points (#10-12) to evaluate the effectiveness of the global bundle adjustment. The sequence pairs (#02-12) satisfy the predifined requirements and each pair partially overlaps each other, exclusive of the #01 pair because there is no loop-closure point between the sequences. This dataset will be publicly available.

KITTI Dataset To evaluate the effectiveness of LC-VSLAM, we selected two sequences of the KITTI stereo dataset also meeting the requirement (i), #00 and #05. We prebuilt maps with the odd-numbered images of the left camera and input the even-numbered images of the right camera to LC-VSLAM. The prebuilt maps were constructed as follows: (I) perform a standard Visual SLAM to estimate initial camera poses and 3D points, (II) replace the estimated camera poses with the ground truth, (III) perform a bundle adjustment to correct the errors found in the ground truth and to refine the positions of the 3D points.

Campus and Desk Datasets We also created two datasets for qualitative evaluations. The sequences of Campus dataset (Scene A and B) were captured by cameras with wide-angle and fisheye lenses (Panasonic LUMIX GX7MK3) and a panoramic camera (RICOH THETA Z1) independently. Their common camera path included both indoor and outdoor scenes. The sequences of Desk dataset were captured by a camera with a wide angle lens, and a pair of successive sequences was processed to assure the privacy protection by means of removing two personal objects in the original image and changing the displays.

Table 3. Mean APE and RPE for trans. [m] of LC-VSLAM with/without the pose graph optimization (PGO) and the global bundle adjustment (Global BA) for each camera device data.

	Perspective	Fisheye	Equirectangular
None	24.06 / 1.292	10.16 / 1.064	14.28 / 3.682
Only PGO	3.301 / 1.151	1.670 / **0.8039**	9.640 / 2.790
PGO & Global BA	**3.018** / **1.100**	**1.593** / 0.8525	**8.320** / **2.404**

4.4 Quantitative Evaluation

We quantitatively evaluated the tracking time of each frame and means of the absolute pose errors (APE) for translation and rotation of the single-shot localization by the proposed LC-VSLAM and p6L [33] on the sequence pair #01 (Table 1). In 640×360 image resolution, the tracking time for LC-VSLAM is 31.09 ms ($\approx$32[fps]), much faster than the 140.3 ms for p6L, which can be defined as *real time*.

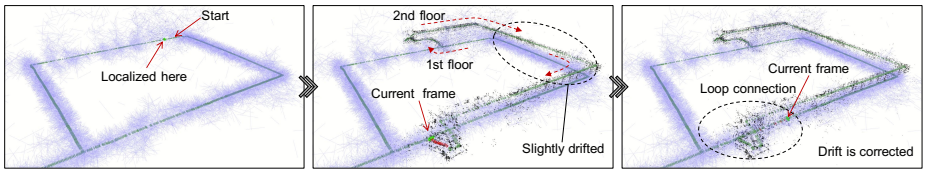
LC-VSLAM also achieves a better result in APE, 0.1979/0.2841 [m]/[deg], than p6L does: 0.7815/0.5896 [m]/[deg]. This means the proposed method outperforms p6L in its tracking speed and localization accuracy because LC-VSLAM can utilize the continuity of input images. Moreover, Table 2 shows the APE for translation and rotation on synthetic images via the CARLA simulator for each camera device (#02-09) and on real images of the KITTI stereo datatset for perspective cameras. LC-VSLAM can estimate camera poses using a 3D line map with accuracy similar to using a 3D point map for all the camera types. In the case of fisheye projection, the estimation error is relatively larger than that of other projection models. These results verify the accuracy and the efficiency of the LC-VSLAM.

To evaluate the effectiveness of the global bundle adjustment with a line cloud, we compared the localization accuracy of LC-VSLAM with and without the pose graph optimization (PGO) and the global bundle adjustment on the sequence pairs of the CARLA dataset (#10-12). Table 3 shows the mean APE and the relative pose errors (RPE) for translation on three sequences. The PGO corrects the estimation errors, especially for the APE, as with a standard Visual SLAM, and our global BA can refine the PGO results.

4.5 Qualitative Evaluation

We applied the proposed framework to various scenes, and confirmed that the algorithms work effectively. The images of Figure 7 show the scenes with a prebuilt 3D line cloud (blue), a reconstructed 3D point cloud (black), and keyframes (green or red), which were all made from the video sequences captured with the panoramic camera (Scene A of Campus dataset). The order from the first (left) to the last (right) columns represents an example of the reconstruction process.

LC-VSLAM localizes and tracks the camera in the prebuilt 3D line-cloud map soon after each sequence starts. Additionally, mapping as well as tracking continuously perform well even in the area outside of the prebuilt map. In other



(a) Relocalization (b) Before loop closure (c) After loop closure

Fig. 7. Example of a reconstructed 3D map in case of a equirectangular model, which includes a prebuilt 3D line cloud (blue), a reconstructed 3D point cloud (black), and keyframes (green or red). (See all the other results in the supplementary material.)

words, the prebuilt map can be extended effectively with the LC-VSLAM processing on the client. Finally, a loop connection point is correctly found when the camera goes back to the prebuilt map area.

Furthermore, in the tracking process of our framework, 3D points may be subsequently restored near 3D lines in a prebuilt map. In a case wherein an object to be protected against inversion attack is present in the prebuilt map but does not in a sequence that a client captures, it is necessary to guarantee that the 3D points of the concealed privacy objects are not restored near the corresponding 3D lines. Figure 1 shows a typical example of the case with the Desk dataset and that privacy related to an object that only exists in the prebuilt map is protected.

The foregoing results lead us to believe that the proposed LC-VSLAM framework works well for various scenes and cameras.

5 Conclusions

In this paper, we proposed a privacy-preserving Visual SLAM framework for real-time tracking and bundle adjustment with a line-cloud map, which we refer to as LC-VSLAM. In the framework, we have presented efficient methods of relocalization and tracking by utilizing 3D points reconstructed by a Visual SLAM client and discretizing 3D lines to 3D points. For optimization in terms of both 3D points and lines, we proposed four types of bundle adjustments by introducing error metrics for 3D lines. These methods are applicable to various types of projection models, such as perspective and equirectangular models. The experiments on videos captured with various types of cameras verified the effectiveness and the real-time performance of LC-VSLAM. Thus, the proposed framework enables real-time tracking/mapping with a line-cloud map in practical applications such as AR and MR. The protective function of scene privacy is in place for map sharing among multiple users.

For future studies, we will refine the formulation for the error metric of the virtual observation $\mathbf{v}$. In this study, Ω_{vL} was set as a constant value because the methodology is not trivial to convert the information matrix of the 3D line $\mathbf{L}$ to that of the cosine distance between their plane normals. The refined formulation will enable a more accurate global optimization with prebuilt line clouds.

References

1. Agarwal, S., Furukawa, Y., Snavely, N., Simon, I., Curless, B., Seitz, S.M., Szeliski, R.: Building Rome in a day. Communications of the ACM **54**(10), 105–112 (2011)
2. Alcantarilla, P.F., Bartoli, A., Davison, A.J.: KAZE features. In: European Conference on Computer Vision (ECCV). pp. 214–227 (2012)
3. Bay, H., Ess, A., Tuytelaars, T., Van Gool, L.: Speeded-up robust features (SURF). Computer Vision and Image Understanding (CVIU) **110**(3), 346–359 (2008)
4. Cui, H., Gao, X., Shen, S., Hu, Z.: HSfM: Hybrid structure-from-motion. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1212–1221 (2017)
5. Davison, A.J., Reid, I.D., Molton, N.D., Stasse, O.: MonoSLAM: Real-time single camera SLAM. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) **29**(6), 1052–1067 (2007)
6. DeTone, D., Malisiewicz, T., Rabinovich, A.: Superpoint: Self-supervised interest point detection and description. In: IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). pp. 337–349 (2017)
7. Dong, R., Fremont, V., Lacroix, S., Fantoni, I., Changan, L.: Line-based monocular graph SLAM. In: IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI). pp. 494–500 (2017)
8. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: CARLA: an open urban driving simulator. In: Conference on Robot Learning (CoRL). pp. 1–16 (2017)
9. Eggert, D.W., Lorusso, A., Fisher, R.B.: Estimating 3-d rigid body transformations: a comparison of four major algorithms. Machine Vision and Applications (MVA) **9**(5-6), 272–290 (1997)
10. Engel, J., Koltun, V., Cremers, D.: Direct sparse odometry. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) **40**(3), 611–625 (2018)
11. Engel, J., Schöps, T., Cremers, D.: LSD–SLAM: Large-scale direct monocular SLAM. In: European Conference on Computer Vision (ECCV). pp. 834–849 (2014)
12. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM **24**(6), 381–395 (1981)
13. Fischler, M.A., Bolles, R.C.: Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. Communications of the ACM **24**(6), 381–395 (1981)
14. Galvez-Lopez, D., Tardos, J.: Bags of binary words for fast place recognition in image sequences. IEEE Transactions on Robotics (TRO) **28**(5), 1188–1197 (2012)
15. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the KITTI vision benchmark suite. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3354–3361 (2012)
16. Grisetti, G., Kümmerle, R., Stachniss, C., Burgard, W.: A tutorial on graph-based SLAM. IEEE Transactions on Intelligent Transportation Systems (ITS) Magazine **2**, 31–43 (2010)
17. Haralick, R.M., Lee, C.n., Ottenburg, K., Nölle, M.: Analysis and solutions of the three point perspective pose estimation problem. In: International Conference on Computer Vision and Pattern Recognition (CVPR). vol. 91, pp. 592–598 (1991)
18. Horn, B.: Closed-form solution of absolute orientation using unit quaternions. Journal of the Optical Society of America A (JOSA A) **4**, 629–642 (04 1987)
19. Huizhong, Z., Danping, Z., Pei, L., Ying, R., Liu, P., Wenxian, Y.: StructSLAM: Visual SLAM with building structure lines. IEEE Transactions on Vehicular Technology (TVT) **64**(4), 1364–1375 (2015)

20. Klein, G., Murray, D.: Parallel tracking and mapping for small AR workspaces. In: Proceedings of IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR). pp. 225–234 (2007)
21. Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K., Burgard, W.: g2o: A general framework for graph optimization. In: IEEE International Conference on Robotics and Automation (ICRA). pp. 3607–3613 (2011)
22. Lourakis, M.I.A., Argyros, A.A.: SBA: A software package for generic sparse bundle adjustment. ACM Transactions on Mathematical Software (TOMS) **36**(1) (2009)
23. Lowe, D.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision (IJCV) **60**, 91–118 (11 2004)
24. Mur-Artal, R., Montiel, J.M.M., Tardós, J.D.: ORB–SLAM: a versatile and accurate monocular SLAM system. IEEE Transactions on Robotics (TRO) **31**(5), 1147–1163 (2015)
25. Mur-Artal, R., Tardós, J.D.: ORB–SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. IEEE Transactions on Robotics (TRO) **33**(5), 1255–1262 (2017)
26. Newcombe, R., Lovegrove, S., Davison, A.: DTAM: Dense tracking and mapping in real-time. In: Proceedings of IEEE International Conference on Computer Vision (ICCV). pp. 2320–2327 (2011)
27. Pittaluga, F., Koppal, S.J., Kang, S.B., Sinha, S.N.: Revealing scenes by inverting structure from motion reconstructions (2019)
28. Pumarola, A., Vakhitov, A., Agudo, A., Sanfeliu, A., Moreno-Noguer, F.: PL-SLAM: Real-time monocular visual SLAM with points and lines. In: IEEE International Conference on Robotics and Automation (ICRA). pp. 4503–4508 (2017)
29. Quan, L., Lan, Z.: Linear N-point camera pose determination. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) **21**(8), 774–780 (1999)
30. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: ORB: an efficient alternative to SIFT or SURF. In: IEEE International Conference on Computer Vision (ICCV). pp. 2564–2571 (2011)
31. Sattler, T., Leibe, B., Kobbelt, L.: Fast image-based localization using direct 2D-to-3D matching. In: IEEE International Conference on Computer Vision (ICCV). pp. 667–674 (2011)
32. Schlegel, D., Grisetti, G.: HBST: A Hamming distance embedding binary search tree for feature-based visual place recognition. IEEE Robotics and Automation Letters (RAL) **3**(4), 3741–3748 (2018)
33. Speciale, P., Schonberger, J.L., Kang, S.B., Sinha, S.N., Pollefeys, M.: Privacy preserving image-based localization. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5493–5503 (2019)
34. Speciale, P., Schonberger, J.L., Sinha, S.N., Pollefeys, M.: Privacy preserving image queries for camera localization. In: IEEE International Conference on Computer Vision (ICCV). pp. 1486–1496 (2019)
35. Strasdat, H., Montiel, J., Davison, A.J.: Scale drift-aware large scale monocular SLAM. Robotics: Science and Systems VI **2**(3) (2010)
36. Sumikura, S., Shibuya, M., Sakurada, K.: OpenVSLAM: A Versatile Visual SLAM Framework. In: ACM International Conference on Multimedia (ACMMM). pp. 2292–2295. ACM (2019)
37. Sweeney, C., Fragoso, V., Höllerer, T., Turk, M.: gDLS: A scalable solution to the generalized pose and scale problem. In: European Conference on Computer Vision (ECCV). pp. 16–31. Springer (2014)

38. Tang, J., Ericson, L., Folkesson, J., Jensfelt, P.: GCNv2: Efficient correspondence prediction for real-time SLAM. IEEE Robotics and Automation Letters (RAL) **4**(4), 3505–3512 (2019)
39. Tateno, K., Tombari, F., Laina, I., Navab, N.: CNN-SLAM: Real-time dense monocular SLAM with learned depth prediction. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 6243–6252 (2017)
40. Triggs, B., McLauchlan, P.F., Hartley, R.I., Fitzgibbon, A.W.: Bundle adjustment — a modern synthesis. In: International Workshop on Vision Algorithms. pp. 298–372 (1999)
41. Umeyama, S.: Least-squares estimation of transformation parameters between two point patterns. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) (4), 376–380 (1991)
42. Wu, C., Agarwal, S., Curless, B., Seitz, S.: Multicore bundle adjustment. In: International Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3057–3064 (2011)
43. Yang, N., Wang, R., Stckler, J., Cremers, D.: Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry. In: European Conference on Computer Vision (ECCV). pp. 835–852 (2018)
44. Yi, K., Trulls, E., Lepetit, V., Fua, P.: LIFT: Learned invariant feature transform. In: European Conference on Computer Vision (ECCV). pp. 467–483 (2016)
45. Zhou, H., Ummenhofer, B., Brox, T.: DeepTAM: Deep tracking and mapping. In: European Conference on Computer Vision (ECCV). pp. 851–868 (2018)