IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 8 January 2024, accepted 22 January 2024, date of publication 5 February 2024, date of current version 20 February 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3362675

RESEARCH ARTICLE

TDO-SLAM: Traffic Sign and Dynamic Object Based Visual SLAM

SOON-YONG PARK[1] AND JUNESUK LEE[2]
[1]School of Electronic and Electrical Engineering, Kyungpook National University, Daegu 41566, South Korea
[2]42dot, Gangnam-gu, Seoul 06267, South Korea

Corresponding author: Soon-Yong Park (sypark@knu.ac.kr)

This work was supported in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education under Grant 2021R1A6A1A03043144, and in part by the Institute of Information & Communications Technology Planning & Evaluation (IITP) funded by the Korean Government [Ministry of Science and ICT (MSIT)] through the Manipulation and Augmentation for XR in the Real-World Environment under Grant 2021-0-00320.

ABSTRACT This paper introduces a real-time visual SLAM system, TDO-SLAM, using only a stereo vision camera. TDO-SLAM works not only in static but also in dynamic road environment by incorporating the object motion and the planar property of standing traffic signs. Traditional visual SLAM systems assume that the road environment is static. However, a variety of dynamic objects exist in the real-world urban environment. Thus, the traditional SLAM systems are subject to fail due to the various motion of the dynamic objects. To solve this inherent problem in the dynamic environment, TDO-SLAM detects, tracks, and manages the global object identification of dynamic objects and standing traffic signs through a novel Object-Level-Tracking method. We improve the accuracy of camera pose estimation through several steps of bundle adjustments, including the residual terms for the planar constraint of traffic signs and the dynamic object motion. Experimental results show that pose estimation accuracy is improved in complex environment with several dynamic objects and traffic signs. Performance of TDO-SLAM is analyzed and compared with ORB-SLAM2, ORB-SLAM3, and DynaSLAM using three benchmark datasets, KITTI Odometry dataset, KITTI Raw dataset, and Complex Urban dataset.

INDEX TERMS Dynamic SLAM, visual localization, pose estimation, autonomous vehicle.

I. INTRODUCTION

Dynamic visual SLAM (Simultaneous Localization and Mapping) is a task of simultaneous localization and map generation from the image sequence of dynamic indoor or outdoor environment. Conventional visual SLAM systems assume that an outdoor road environment is static without or few moving objects. The visual information such as texture and color from the image sequence is extracted to estimate the 3D pose of vehicle or robot [1], [2], [3], [4]. Conventional SLAM systems perform well in the static environment as presented in a variety of investigations [5], [6], [7], [8]. However, the static environment assumption is unsuitable for complex urban or highway driving scenarios. In this scenario, various dynamic objects, such as moving cars, trucks, bicycles, and pedestrians, hinder the continuous observation of static elements from a vision camera. Dynamic objects moving continuously over time can cause inevitable pose estimation error in the static visual SLAM. This is because the relative motion of a static object from a driving or neighboring vehicle is fixed. On the contrary, the relative motion of a dynamic object changes depending on both object and driving vehicle's motion. For example, as shown in Figure 1(a), moving objects with different dynamic motions in a highway environment cause a significant error in the visual odometry. In Figure 1(a), all objects move in the same direction with the driving vehicle, however if the relative speed of dynamic objects faster than the driving vehicle, motion estimation can be done as if the vehicle is moving relatively backward.

With recent advances in computer vision techniques, the performance of object detection, semantic or instance

The associate editor coordinating the review of this manuscript and approving it for publication was Zheng H. Zhu.

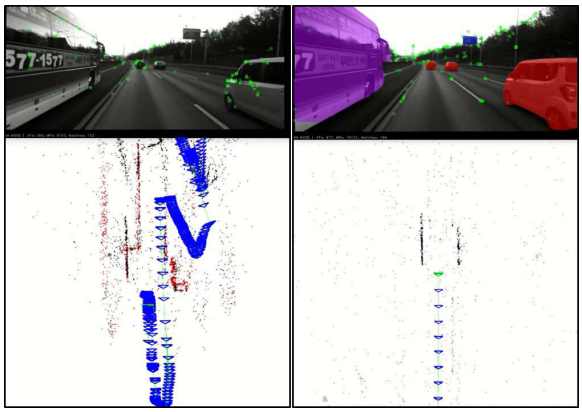

FIGURE 1. Comparison of the proposed dynamic SLAM with a traditional static scene visual SLAM (a) An ORB-SLAM2 result shows that the vehicle odometry is distorted due to the dynamic objects. (b) The vehicle motion is correctly estimated in the same dynamic scene using the proposed TDO-SLAM.

segmentation of object region has been dramatically improved. In this reason, various pose estimation methods in dynamic environment have been introduced by employing the recent learned-based techniques in the visual SLAM [9], [10], [11], [12].

This study also proposes a dynamic visual SLAM system, TDO-SLAM, to solve the pose estimation problem of a stereo vision camera in the dynamic road environment. The motion of dynamic objects and the planar constraints of the traffic sign are combined in Bundle Adjustment (BA) for the pose estimation of the camera. In this study, the camera pose is considered as the same as the vehicle pose. The proposed system employs a lightweight instance segmentation module, YOLACT-Edge [13] and the famous framework of ORB-SLAM2 [5] in the consideration of real-time application.

The proposed system works in the presence of both dynamic and static objects through the instance segmentation as shown in Figure 1(b). Local Static BA (LSBA) and Local Dynamic BA (LDBA) are introduced for robust and accurate pose estimation in dynamic and static environments. LSBA improves the modeling of the traffic sign and camera pose estimation accuracy by including the planar constraint of the traffic sign. LDBA simultaneously estimates the motion of dynamic objects and the pose of the stereo camera. Since the motion estimation of a moving object provides the relative pose with respect to the camera, it can improve both the robustness and the accuracy of camera pose estimation.

The contributions of the proposed TDO-SLAM are summarized as follows:

i) A real-time dynamic visual SLAM system is proposed, which can run in for both static and dynamic environments.
ii) Local Static BA is proposed to incorporate the 3D plane constraint of standing traffic signs. LSBA improves the robustness of camera localization in scenarios where traffic signs are detected in either static or dynamic road environments.
iii) Local Dynamic BA is proposed to estimate dynamic object's motion state and motion transformation. When a moving object is found, LDBA determines the motion state of the moving object and simultaneously estimates the motion transformation between the camera and the moving object. In addition, in a scenario where traffic signs and dynamic objects are found simultaneously, pose estimation performance can be enhanced by simultaneously performing LDBA on the motion of dynamic objects and LSBA on the planar traffic signs.
iv) This study is the first attempt using the Complex Urban dataset [14] for evaluation of dynamic visual SLAM. The Complex Urban dataset contains many moving objects in several cluttered urban environments, which make difficult for running dynamic visual SLAM methods.

To evaluate the performance of the proposed system, we provide experimental results using the KITTI dataset [15] and the Complex Urban dataset [14]. In addition, comparative evaluation with ORB-SLAM2, ORB-SLAM3, and DynaSLAM [12] is provided using the same dataset. TDO-SLAM achieves improved performance compared with other studies.

II. RELATED WORK

A. TRADITIONAL VISUAL SLAM

A traditional visual SLAM based on well-known image feature is ORB-SLAM2 [5]. As a representative example, ORB-SLAM2 shows real-time performance because the system is configured based on multithreading and BA for reprojection error minimization of feature points. It can also reconstruct the mapping scale through stereo matching and shows high performance in various benchmarks. However, because the static environment is assumed, its performance is degraded in such scenarios involving moving and dynamic objects. In addition, pose estimation fails when image features are distributed mostly in dynamic objects. To overcome this inherent problem, the proposed TDO-SLAM system extends the framework of ORB-SLAM2 and presents a new solution that works robustly and accurately in both static and dynamic environments.

B. DYNAMIC VISUAL SLAM WITHOUT DYNAMIC OBJECT MOTION

DynaSLAM [12], StaticFusion [16], and ElasticFusion [17] are such approaches that extract only the static object and background areas by masking the dynamic object area to estimate the camera pose in the dynamic environment. DynaSLAM extends ORB-SLAM2, and the dynamic object segmentation is based on Mask-RCNN [18], one of the learned semantic segmentation techniques. The feature points in the dynamic areas are removed to reduce the pose error due to the dynamic features. However, it shows lower accuracy than ORB-SLAM2 in some scenarios (e.g., a car waiting for

a traffic light) because even non-moving dynamic objects are removed. In addition, real-time performance cannot be guaranteed due to the heavy resources of the learned segmentation module.

Guan et al. [19] use ORB-SLAM3 [20] as the backbone of a dynamic SLAM method. They use YOLOv5 [21] to detect objects and remove image features which are on the dynamic object areas. Their method is only evaluated using the TUM-RGBD [22] dataset which consists of only indoor image sequences. DE-SLAM [23] proposes a dynamic visual SLAM method which can be applied to a field robot. In this method, MobileNet V2 [24] is used to extract the deep features in input images and those features on moving objects are removed to improve the robot's localization performance. The performance of DE-SLAM is evaluated by using several indoor and outdoor scenes. However, all evaluations are done with only moving human objects, not including road objects.

Chen et al. [25] introduces a semantic SLAM method to address the real time performance in dynamic environment. PSPNet18 [26]. The key idea of the method is to assign dynamic probability to each image feature point. Initially, they assign probability 0.5 to each feature and use the semantic segmentation information to update the probability in keyframes. If the probability of a feature point is high, it is not used in the BA optimization process.

RDS-SLAM [27] is similar with the method proposed by Chen et al. [25] in that the moving probability is used to detect and remove outliers from feature tracking. The initial probability of a map point is assigned as 0.5 and it is updated by the Bayesian filtering [28].

Su et al. [29] use YOLOv5 to detect dynamic feature points and propose to use the homography matrix in the optical flow tracking module for more efficient and real-time tracking. In the optical flow step of the ORB-SLAM2, if the magnitude of a feature tracking flow is larger than a threshold, then the corresponding feature is removed and its object mask value is set to zero to remove from the tracking.

C. DYNAMIC VISUAL SLAM WITH DYNAMIC OBJECT MOTION

CubeSLAM [30], ClusterVO [31], and VDO-SLAM [32] apply the motion of the dynamic objects to camera pose estimation by utilizing the feature information of dynamic objects detected in the segmentation task. The performance of pose estimation by simply removing dynamic objects depends on the number of feature points included in the dynamic objects. In the worst case, if all feature points in an image are included in the moving objects, camera pose estimation may fail because all feature points can be removed. To improve this problem, the individual motion of dynamic objects and camera poses are optimized by BA simultaneously. This approach has been proven through studies such as DyanSLAM II [11] and DOT-SLAM [9], where motion estimation of moving objects improves the accuracy of camera pose estimation.

DynaSLAM II [11] is an extended study of DynaSLAM [12], and handles static and dynamic features separately by masking dynamic objects using instance segmentation information. In the method, dynamic features are not removed and simultaneous estimation of camera pose and dynamic object motion is proposed by considering the linear and angular velocity of dynamic object motion. However, it can't run in real-time because object segmentation must be done offline in advance.

DOT-SLAM [9] solves the real-time problem caused by the processing time of the instance segmentation module. By using the mask propagation method, DOT-SLAM can run in real-time regardless of the computation time of the segmentation module. In addition, the camera motion is decided through dynamic object motion estimation for dynamic feature points. An object without motion is considered as a static feature and included in the existing SLAM system to be used to estimate the camera pose.

DOE-SLAM [33] proposes a method of determining whether a dynamic object is moving. It also proposes a method of estimating the camera pose when the dynamic object area is large. DOE-SLAM even utilizes the motion of dynamic objects when it is impossible to estimate the camera pose using only background features due to the moving dynamic objects. In addition, an object modeling method was proposed to estimate the correspondences between the dynamic objects detected by Mask-RCNN [18] in consecutive frames.

III. SYSTEM OVERVIEW

The system architecture of the proposed TDO-SLAM is shown in Figure 2, which is the extension of the architecture of ORB-SLAM2. The proposed system consists of Instance segmentation, Tracking, Local mapping, and Loop closing thread modules. The input of the system is synchronized stereo images, and outputs are poses of the camera and map points of static and moving objects. The instance segmentation thread module detects dynamic objects and standing traffic signs from the left stereo image. The tracking thread module performs stereo matching from a frame sequence and ORB-feature tracking with the previous frame, and estimates the camera pose after excluding dynamic features.

In the proposed system, an object-based feature classification & tracking module classifies object feature categories and assigns a global object identification (ObjectID) through object tracking between previous frames. The camera tracking module estimates the camera pose after excluding the features included in the dynamic object. The local mapping module includes LSBA and LDBA modules, instead of ORB-SLAM2's local BA module. LSBA performs BA for reprojection error minimization using only map-points from background and traffic signs. Non-linear optimization for keyframe pose and map-point mapping is also done in LSBA. The proposed 3D plane constraint for traffic sign is applied. LDBA consists of Object Motion BA (OMBA), Is Motion?

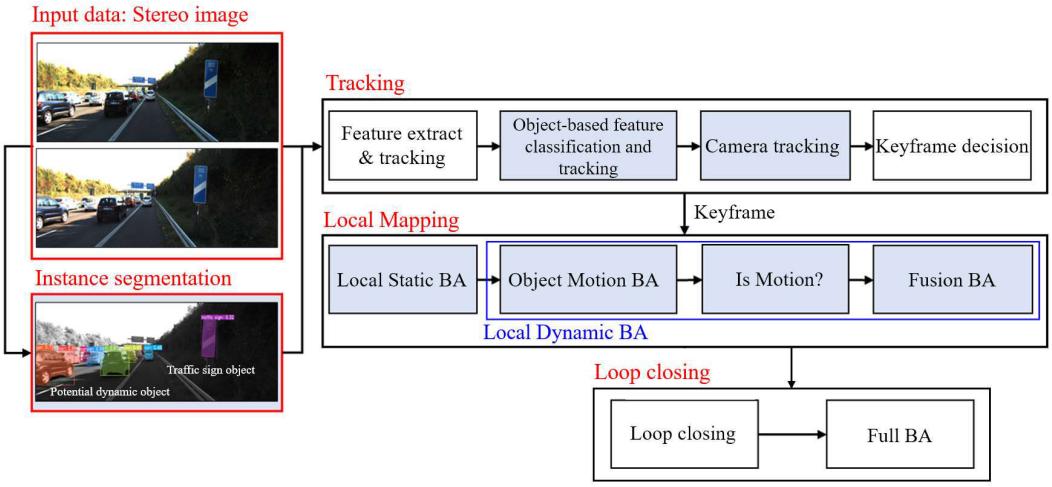

FIGURE 2. The architecture of TDO-SLAM. The baseline framework of TDO-SLAM is ORB-SLAM2. In addition to the ORB-SLAM2 architecture, the blue boxes are newly added in TDO-SLAM.

(IMQ), and Fusion BA (FBA). As the first step, OMBA fixes the static map point and camera pose and estimates only dynamic object motion. Then, the initially estimated object motion determines whether or not it is moving through a threshold value. In the next step, IMQ, a static object in a stationary state is optimized identically to the static object, and a moving object in a moving state is optimized considering the object motion. As the final step of LDBA, FBA performs BA simultaneously with camera pose and static map-point, dynamic map-point, and dynamic object motion. More details of local BA are presented in Section VI.

IV. INSTANCE SEGMENTATION

Instance segmentation thread detects and segments standing traffic signs and potential dynamic objects in an image sequence in real-time based on YolackEdge [13] optimized through TensorRT [34]. Total six dynamic object classes are defined, pedestrians, bicycles, cars, motorcycles, buses, and trucks. The background region is also defined if any region is not categorized into traffic sign or dynamic object. Since all regions of an image are segmented, the network's output is modified to output the segmentation mask of all regions from the image. Each pixel in the output segmentation mask contains class (ClassID) and local object (ObjectID) identification. The backbone network uses the R-101-FPN model [35], considering real-time and accuracy. YolackEdge's pre-trained weight was trained using the COCO 2017 [36] dataset containing standing traffic signs and potential dynamic objects. However, although standing traffic signs are included in COCO 2017, the KITTI odometry dataset [15] rarely contains standing traffic signs due to the small amount of the dataset. To overcome this problem, we have retrained the network to segment standing traffic signs and dynamic objects using the COCO 2017, Cityscape [37], and DFG [38] datasets. In addition, since these datasets do not have labeling information for continuous images, we trained the YolackEdge network without the partial feature transformation. Figure 3 shows an example of instance segmentation of a road image with dynamic moving objects. Image features on each segmentation cluster are handled by ObjectID in the data management graph descried in the next section.

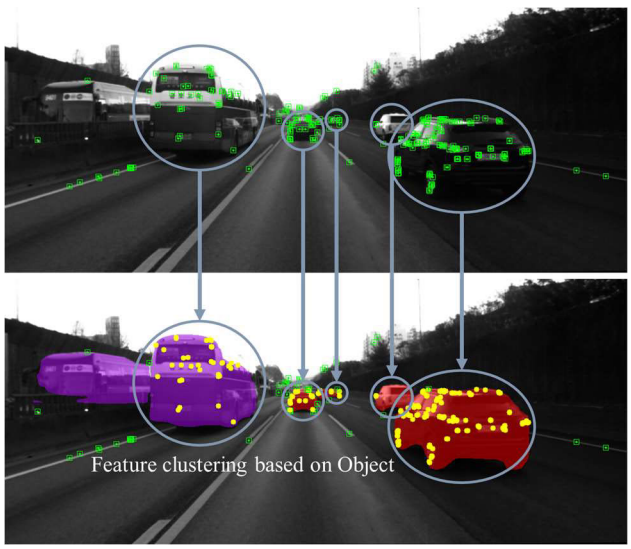

FIGURE 3. Instance segmentation for traffic signs and dynamic objects in a highway environment. Image features on each dynamic object is managed by identification number.

V. TRACKING

A. FEATURE EXTRACT AND TRACKING

The feature extraction and tracking step works similar with the process in ORB-SLAM2. The ORB features are extracted from a current image frame and feature matching is performed with the previous frame. Additionally, we assign identification attributes (ObjectID, ClassID) to every ORB feature using instance segmentation masks. ObjectID is used

to group ORB features to the same object. ClassID is used to determine if an object is static or potential dynamic state. ORB features included in the same object are clustered and managed based on ObjectID by the tree structure shown in Figure 4. ObjectID expresses a local object identification and the ID number is randomly assigned even though the same object is observed in consecutive image frames. This is because object tracking information is unknown at this stage. When classified as the background object, (ObjectID, ClassID) is assigned as $(-1, -1)$.

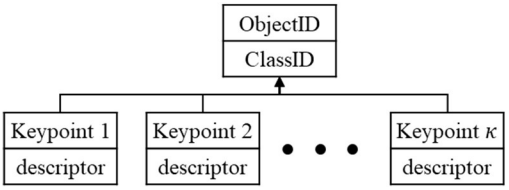

FIGURE 4. The graph structure of data management for the keypoints of an ObjectID cluster.

B. OBJECT LEVEL TRACKING

Object tracking information for several frames is required to perform bundle adjustment based on graph optimization for traffic signs and potential dynamic objects. The proposed Object Level Tracking (OLT) method aims to maintain and manage consistent global ObjectID for the same object in consecutive frames. An example is shown in Figure 5. In the right of the figure, a potential moving object in the current frame has a local ObjectID and five features. Then, the five features match with the features in the previous frame's features. At the same time, the matching count is also used to find the global ObjectID in the previous frame, who has the highest matching count. Finally, the global ObjectID is decided to the moving object in the current frame.

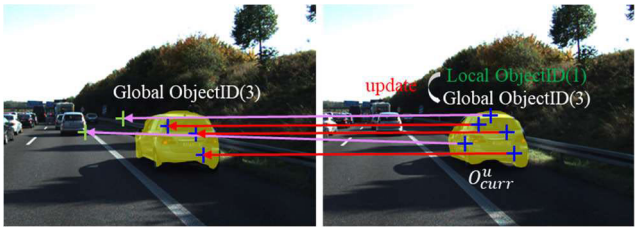

FIGURE 5. Description of Object Level Tracking. Left: ObjectID is assigned to features in the previous frame. Right: Local features in the current frame is matched with the global ObjectID features.

As shown in the figure, the local ObjectID of the moving vehicle is updated to the same global ObjectID with the previous frame. If the size of the matching count is less than 0.5 times the number of feature points in the object area of the current frame, it is regarded as a new object and a new global ObjectID is assigned. Global ObjectID is assigned sequentially in ascending order starting from 0.

If there is any occlusion on the moving object, in either the current or previous frame, the number of tracking feature maybe reduced. However, as mentioned in the previous paragraph, if the number of tracking features is more than 0.5 times the number of feature points in the object area of the current frame, the global ObjectID can be identified through the consecutive frames.

C. CAMERA TRACKING

In this step, camera pose is predicted using feature points tracked between the previous and the current frames. Similar to a previous investigation [39], map initialization or initial camera pose prediction is performed using the PnP (Perspective-n-Point) algorithm based on photometry reprojection error for all feature points tracked in all input images. In this step, all potential dynamic object features are considered as outliers to maintain real-time and stability, and the camera pose is estimated using static object features as shown in Figure 6.

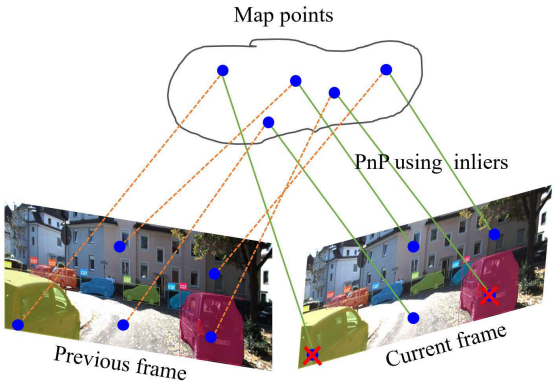

FIGURE 6. In the camera tracking module, all features in potential dynamic objects are considered as outliers, marked X on the dynamic object segmentations in the current frame. Only inlier features are used for PnP algorithm.

VI. LOCAL MAPPING

The local mapping module of TDO-SLAM is similar with ORB-SLAM2, and the original Local BA part is replaced with the proposed Local BA method, which estimates camera poses for keyframes. The proposed Local BA is divided into Local Static BA (LSBA) and Local Dynamic BA (LDBA) sub-modules. LSBA performs BA using background and traffic sign features. It simultaneously reconstructs 3D map points and estimates camera pose by minimizing the residual for both static map points and the traffic sign constraint using a nonlinear optimization algorithm. The residual for the static map point is an error function for fine-tuning the camera pose and the static map point. Residual for traffic sign constraint is an error function that includes the constraint of the 3D planar surface of the traffic sign, which reconstructs the traffic sign to be a 3D planar surface. If no traffic sign exists, LSBA works exactly same as the ORB-SLAM2's Local BA [5].

LSBA is followed by LDBA. The optimized map points and camera poses are used as initial information of LDBA. The LDBA sub-module can be skipped if there is no potential dynamic object. LDBA includes the BA method

for estimating static map point, dynamic map point, dynamic object motion, and camera pose, constituting the factor graph structure shown in Figure 7. LDBA consists of three steps: Object motion BA (OMBA), Is Motion? (IMQ), and Fusion BA (FBA). OMBA estimates the initial motion information of a potential dynamic object. IMQ determines whether the motion state of the potential dynamic object is moving state or static state. FBA is an error function that combines LSBA and OMBA, and fine-tunes the camera poses and all map points. All nonlinear optimization solutions use the Levenberg-Marquardt [40] (LM) algorithm.

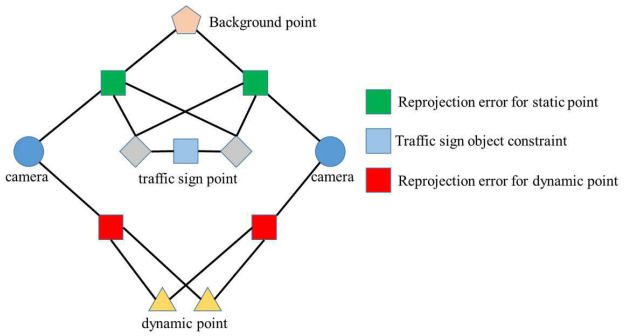

FIGURE 7. BA factor graph representation of traffic signs and dynamic objects.

A. LOCAL STATIC BA

The residual of OMBA consists of two terms; residual for static point and residual for traffic sign constraint. First, the residual for static point is residual to minimize the photometric reprojection error of the static point to the observed keyframes. The static points include the points of both the background and traffic signs. Keyframes used in optimization consist of the current keyframe and other keyframes included in the covisibility graph [41], [42]. Nodes in the covisibility graph are keyframes, and the edges between the nodes are created when the number of map points observed between two keyframes is fifteen or more. In addition, the covisibility is determined according to the number of map points shared by two keyframes. This means that the sizes of edges for all nodes are determined differently. It creates a powerful graph for keyframes, including the case where the covisibility is one hundred or more, and it is defined as an essential graph. The essential graph is used in the loop closure module.

The second term is the proposed residual for traffic sign constraint. The face of a traffic sign is a planar 3D surface in the real world. Considering this geometric condition, map points belong to a traffic sign are reconstructed on a 3D planar surface in the BA process. Because the camera pose estimation and map point reconstruction are performed simultaneously in the BA process, the improvement of map point reconstruction performance is directly related to the camera pose estimation performance. The proposed residual with traffic sign constraint is also used in the residual for static points because the traffic sign is a static object.

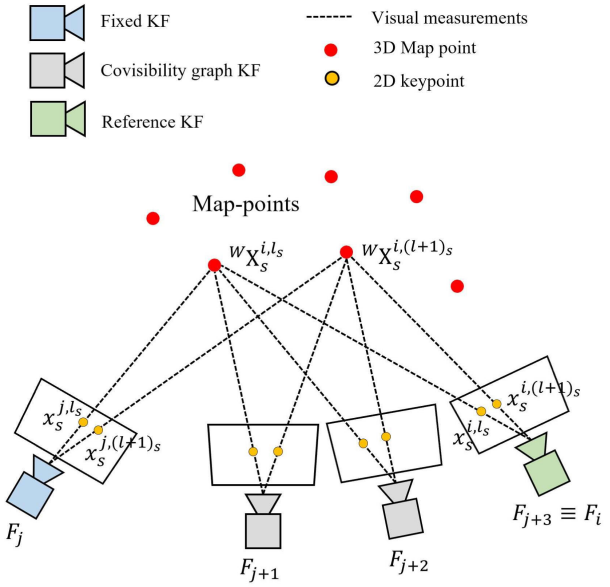

FIGURE 8. Static point residual based on the projection of map points to keyframes.

B. RESIDUAL FOR STATIC POINT

As for the static point, the photometric reprojection error is used as a residual for the keyframe (KF) included in the covisibility graph. Let $\mathcal{P}_i^b$ be the set of map points of the background represented in the world coordinate system and observed in the current keyframe F_i. And let $\mathcal{P}_i^t$ be the set of map points of the t-th traffic sign represented in the world coordinate system and observed in the current keyframe F_i. As a method of reconstructing the initial map points, the stereo matching and triangulation algorithm is used through the observation of multiple keyframes. The reconstructed map points are robust enough because the stereo matching and triangulation is used across multiple keyframes.

In (1), a set of map points, $\mathcal{P}_i^s$ is defined and it represents an element of the sets, $\mathcal{P}_i^b$ and $\mathcal{P}_i^t$. In the world coordinate system, 3D map points $\mathrm{X}_i^{b,l_b} \in \mathcal{P}_i^b$ and $\mathrm{X}_i^{t,l_t} \in \mathcal{P}_i^t$ refer to the l_b-th background map-point observed in F_i and the l_t-th map point included in the t-th traffic sign, respectively. Then, X_i^{s,l_s} is a static feature observed as X_i^{b,l_b} or X_i^{t,l_t} in F_i as shown in (2). The map point is represented with the homogenous coordinate system as shown in (3).

$$\mathcal{P}_i^s \in \left\{\mathcal{P}_i^b, \mathcal{P}_i^t\right\} \tag{1}$$

$$X_i^{s,l_s} \in \left\{X_i^{b,l_b}, X_i^{t,l_t}\right\} \tag{2}$$

$$X_i^{s,l_s} = \begin{bmatrix} x & y & z & 1 \end{bmatrix}^T \tag{3}$$

The residual for static point uses an optimization method to minimize the photometric projection error, as presented in (4) and Figure 8. X_i^{s,l_s} means l_s-th static map-point found in F_i. $\mathcal{F}_c$ is a set of keyframes connected to F_i in the covisibility graph. $\mathcal{F}_v$ is the set of keyframes that can observe X_i^{s,l_s} but are not connected to F_i. The keyframe pose of $\mathcal{F}_v$ is fixed as a

constant parameter (blue color KF in Figure 8). Keyframe F_j means a vector as in (5), and x_j^{s,l_s} represents the 2D keypoint for F_j corresponding to X_i^{s,l_s}. $T_{w,j}^c \in SE(3)$ is the pose of F_j, and T is the homogeneous transformation matrix as shown in (6).

In this study, the notation for transformation $T_{a,b}^*$ is defined as follows. $'*'$ means the target object in T^*. If $'*'$ is c, it means transformation for camera, and it is d for dynamic object. $T_{a,b}$ means the coordinate system transformation from the reference coordinate system b to a. $T_{a,b}^{*-1}$ can be expressed as $T_{b,a}^*$. The camera intrinsic matrix is defined by (7) and determined through camera calibration in advance. (f_x, f_y) is the focal length for the pinhole camera model, and (c_x, c_y) is the principal point. Function $\Pi(\cdot)$ means the perspective projection model function as in (8). Symbol s means the scale factor to normalize the z value projected onto the image to one in (8).

$$e_j^{s,l_s} = x_j^{s,l_s} - \prod \left({T_{w,j}^c}^{-1} X_i^{s,l_s} \right) \tag{4}$$

$$F_j \in \{F_i, \mathcal{F}_c, \mathcal{F}_v\} \tag{5}$$

$$T = \begin{bmatrix} r_{11} & r_{12} & r_{12} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \tag{6}$$

$$K = \begin{bmatrix} fx & 0 & cx \\ 0 & fy & cy \\ 0 & 0 & 1 \end{bmatrix} \tag{7}$$

$$\begin{bmatrix} I_x \\ I_y \\ 1 \end{bmatrix} \approx \frac{1}{s} \Pi(T, X) = \frac{1}{s} \times \begin{bmatrix} fx & 0 & cx \\ 0 & fy & cy \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{12} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{8}$$

C. RESIDUAL WITH TRAFFIC SIGN CONSTRAINT

Since the surface of a traffic sign in the real world is flat as shown in Figure 9, this paper proposes a residual with traffic sign constraint to reconstruct the map points of a traffic sign on a 3D plane. The proposed method allows the map points of the traffic sign to be located on a 3D plane when mapping them on the world map. The proposed traffic sign constraint only applies if the number of map points contained in the traffic sign is four or more. When the number of map points contained on a traffic sign is three or less, these map points can form a single plane without applying the residual with the traffic sign constraint.

The residual with the traffic sign constraint is defined by (9). Equation (9) consists of three vectors and creates the constraint for four map points on the traffic sign. Each of the three vectors represents the vector from the central point to other three map points as shown in Figure 10. If four map points lie on a 3D plane, Equation (9) has a scalar value of zero. Function $\mathcal{N}(\mathcal{P}_i^t)$ returns the number of elements in $\mathcal{P}_i^t$. l_t stands for the index of 3D map points for t-th object. In the traffic sign constraint, the scope of l_t is limited by (10). This means the number of constraint residuals for the map point of the traffic sign. For example, if there are $\mathcal{N}(\mathcal{P}_i^t)$ map points on a traffic sign, then $(\mathcal{N}(\mathcal{P}_i^t) - 3)$ residual with traffic sign constraints are created. In summary, Equation (9) is the constraint to maintain the 3D planar property of the map-point of the traffic sign as shown in Figure 10.

FIGURE 9. Examples of detecting traffic sign features. There are multiple image features on various traffic signs.

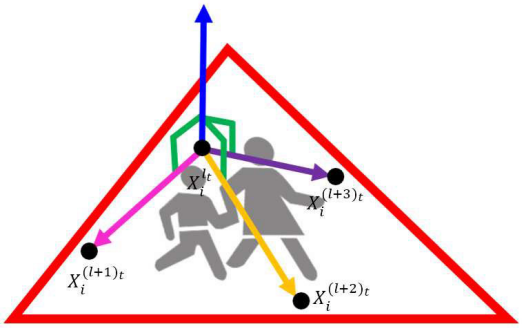

FIGURE 10. Model of a standing traffic sign and four map points on the planar surface.

$$e_t^{l_t} = \left(\left(X_i^{t,(l+1)_t} - X_i^{t,l_t} \right) \times \left(X_i^{t,(l+2)_t} - X_i^{t,l_t} \right) \right) \cdot \left(X_i^{t,(l+3)_t} - X_i^{t,l_t} \right) \tag{9}$$

$$\{ l_t | 0 \le l_t \le (\mathcal{N}(X_i^t) - 3) \} \tag{10}$$

The error value in (11) is minimized by the optimization algorithm for fine tuning $\mathcal{P}_i^s = \{\mathcal{P}_i^b, \mathcal{P}_i^t\}$, the keyframes pose,

and map-points for local window $\mathcal{C}$.

$$min \sum_{j \in C} \left(\sum_{l_s \in \mathcal{P}_j^s} \rho \left(\left\| e_j^{s,l_s} \right\|^2_{\sum_j^{l_s}} \right) \right) + \sum_{t \in Q_t} \sum_{l_t \in X_i^t} \rho \left(\left\| e_t^{l_t} \right\|^2 \right) \tag{11}$$

where $\rho(\cdot)$ is the Huber robust norm function [43], [44], $\sum_i^{l_s}$ is covariance matrix, and $\mathcal{Q}_t$ is a set of indexes for traffic signs. In this study, all non-linear optimization is performed using the LM algorithm.

If there is no potential dynamic object, the result of LSBA is used as the final output of the pose and map points of the current keyframe. However, if there is a potential dynamic object, the result of LSBA is used as the initial camera pose and initial map points of LDBA. Equation (11) is performed by non-linear optimization of the LM algorithm in LSBA. The Jacobian matrices for the residuals for static points and traffic sign constraints are described in the following paragraphs.

The residual for static points defined by (4) has as unknown parameters of static map points and keyframe poses. The residual for static points needs to calculate for each Jacobian matrix for static map points and keyframe poses, respectively. First, the Jacobian matrix for static map points is defined by (12). $\mathcal{J}_\theta$ is the partial derivative of e_j^{s,l_s} with respect to T_θ, and the chain rule of differentiation holds. T_θ is an unknown keyframe pose, $\tilde{m}$ is the partial derivative of the perspective projection function $\Pi(\cdot)$, and $\mathcal{K}$ denotes the camera intrinsic parameter matrix. Computation efficiency can be obtained by minimizing the number of parameters when calculating the solution of non-liner optimization. If T_θ is composed of twelve parameters for a 3×3 rotation matrix and a 3×1 translation matrix as in (6), it can result in very complicated calculations. To minimize the rotation matrix parameters, T_θ follows Lie algebra SO(3). The translation parameters are defined as $\left(t_x^\theta, t_y^\theta, t_z^\theta\right)$. The rotation parameter of w_θ is defined as $\left(w_x^\theta, w_y^\theta, w_z^\theta\right)$ as in (14). This can be logarithm mapped from $\mathbb{R}^3$ to $\mathbb{R}^{3\times3}$ by a skew-symmetric matrix as in (15). Conversely, $\mathbb{R}^{3\times3}$ can be exponentially mapped to the Lie group as $\mathbb{R}^3$ in (16). The Jacobian matrix for the keyframe pose is defined in (17). X_ξ indicates an unknown parameter for map points.

$$\mathcal{J}_\theta \left(e_j^{s,l_s} \right) = \frac{\delta e_j^{s,l_s}}{\delta T_\theta} = \frac{\delta e_j^{s,l_s}}{\delta \tilde{m}} \cdot \frac{\delta \tilde{m}}{\delta \mathcal{K}} \cdot \frac{\delta \mathcal{K}}{\delta T_\theta} \tag{12}$$

$$T_\theta = \left[w_x^\theta \; w_y^\theta \; w_z^\theta \; t_x^\theta \; t_y^\theta \; t_z^\theta \right] \tag{13}$$

$$w_\theta = \left[w_x^\theta \; w_y^\theta \; w_z^\theta \right] \tag{14}$$

$$exp\left(w^\theta\right) = exp\left(\begin{bmatrix} 0 & -w_z^\theta & w_y^\theta \\ w_z^\theta & 0 & -w_x^\theta \\ -w_y^\theta & w_x^\theta & 0 \end{bmatrix} \right) = R_{3\times3}^\theta \in SO(3) \tag{15}$$

$$ln\left(R_{3\times3}^\theta\right) = w^\theta \in SO(3) \tag{16}$$

$$\mathcal{J}_\xi \left(e_j^{s,l_s} \right) = \frac{\delta e_j^{s,l_s}}{\delta X_\xi} = \frac{\delta e_j^{s,l_s}}{\delta \tilde{m}} \cdot \frac{\delta \tilde{m}}{\delta K} \cdot \frac{\delta K}{\delta T_\theta} \cdot \frac{\delta T_\theta}{\delta X_\xi} \tag{17}$$

The residual with traffic sign constraint defined by (9) has static map points included in the traffic sign as an unknown parameter. The Jacobian for traffic sign constraint is defined by (18). X_ξ^{t,l_t} indicates an unknown parameter of a static map point included in the traffic sign. If X_ξ^{t,l_t} is the same static map point as X_i^{s,l_s}, it is shared equally with X_ξ. ψ_1, ψ_2, ψ_3, and ψ_4 mean the indices of each unknown parameter for the four traffic sign map points for $e_t^{l_t}$.

$$\mathcal{J}_{\psi_1} \left(e_t^{l_t} \right) = \frac{\delta e_t^{l_t}}{\delta X_\xi^{t,l_t}}, \quad \mathcal{J}_{\psi_2} \left(e_t^{l_t} \right) = \frac{\delta e_t^{l_t}}{\delta X_\xi^{t,(l+1)_t}},$$
$$\mathcal{J}_{\psi_3} \left(e_t^{l_t} \right) = \frac{\delta e_t^{l_t}}{\delta X_\xi^{t,(l+2)_t}}, \quad \mathcal{J}_{\psi_4} \left(e_t^{l_t} \right) = \frac{\delta e_t^{l_t}}{\delta X_\xi^{t,(l+3)_t}} \tag{18}$$

D. LOCAL DYNAMIC BA

The LDBA module is performed as the next step of LSBA when a potential dynamic object is detected in the instance segmentation module. If the potential dynamic object is not detected, LDBA is not performed. Instead, the optimized keyframes poses and static map points are calculated in LSBA as the final result. This BA consists of three steps, OMBA, IMQ, and FBA and each step runs sequentially. OMBA uses keyframe poses optimized from LSBA as input. OMBA estimates the relative motion matrix of the potential dynamic object, which minimizes the reprojection error for the potential dynamic object. IMQ calculates the rotation and translation scale of the estimated motion matrix and determines the moving state through a threshold comparison. If the state is '*stop*', the motion matrix is initialized as the identity matrix. If the state is '*moving*', the motion matrix is refined in the FBA step. FBA optimizes all of static and dynamic map points, keyframe poses, and dynamic object motion matrices.

1) OBJECT MOTION BA

In this step, the 3D transformation of a potential dynamic object is estimated. The keyframe pose of $\{F_i, \mathcal{F}_c\}$ is used and set as constant values. $\mathcal{P}_i^d$ denotes a map point set of d-th potential dynamic object observed at F_i. $X_i^{d,l_d} \in \mathcal{P}_i^d$ denotes the l_d-th map point of the d-th potential dynamic object observed at F_i. Reprojection error using the constant keyframe poses and motion of the potential dynamic object is defined by (19). Keyframe $F_{\bar{j}}$ denotes the frame at which the d-th potential dynamic object map point is created. $\Delta T_{w_{\bar{j}},w_j}^d \in SE(3)$ represents the object transformation matrix from $F_{\bar{j}}$ to F_j in the world coordinate system of d-th potential dynamic object. If $F_j \equiv F_i$, $\Delta T_{w_j,w_{\bar{j}}}^d$ is initialized to the identity matrix as in (20). This is because there is no movement of the potential dynamic object in the current keyframe if the potential dynamic object is observed for the first time.

$$e_j^{d,l_d} = x_j^{d,l_d} - \prod \left({T_{w,j}^c}^{-1} \Delta T_{w_j,w_{\bar{j}}}^d X_i^{d,l_d} \right) \tag{19}$$

$$Init\left(\Delta T^{d}_{w_j,w_{\bar{j}}}\right) = I_{4\times4} \tag{20}$$

Figure 11 shows an example case where the camera motion and the object motion occur simultaneously. If there is no motion of the object, the corresponding line between the map points and the feature points between multiple keyframes are represented as blue and green dotted lines. Map points of motionless objects can be regarded as static map points. However, if the dynamic object moves simultaneously with the camera motion, the correspondences between the map points and feature points of the keyframe cannot be explained without knowing the object motion.

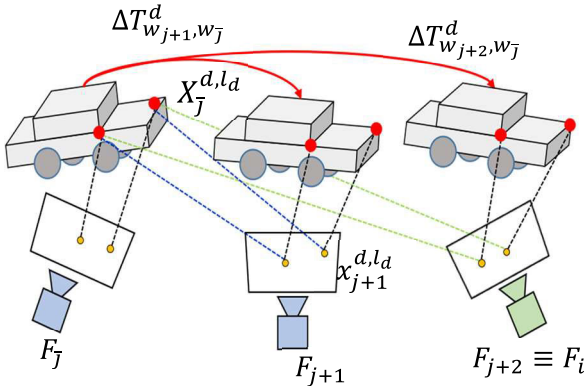

FIGURE 11. Dynamic object reprojection error for the camera poses and moving dynamic objects. The blue and green dotted lines represent the correspondence points between map points and feature points without object motion. The black dotted line represents the correspondences between the map points and the feature points with simultaneous camera and object motion.

In order to reconstruct consistent map points of a dynamic object, the motion of the dynamic object must be considered from the keyframe when the object is first observed. If the motion of the moving object is not considered, the 3D-2D correspondence is not correct, and the method for residual for the static points may cause errors. In this reason, we propose a residual for dynamic object motion by considering the motion of the dynamic object as shown in (21). This represents the reprojection error to the keyframe considering the motion of the dynamic object. In other words, OMBA estimates the relative motion of a potential dynamic object with respect to a fixed camera pose included in the local sliding window(C). This means, $T^{c}_{w,j}$ in (19) is not included in this optimization and is used as a constant value. Residual in (21) is minimized by the LM algorithm, and the dynamic object map points and the object motion transformation are optimized.

$$min \sum\nolimits_{j\in C}\left(\sum\nolimits_{l_d\in P^{d}_{j}} \rho\left(\left\| e^{d,l_d}_{j}\right\|^{2}_{\sum^{k,l_d}_{j}}\right)\right) \tag{21}$$

Equation (21) requires Jacobian matrix for the LM non-linear optimization. Projection error e^{d,l_d}_{j} defined by (19) has unknown parameters of dynamic map point, dynamic motion transformation, and keyframe pose. In OMBA step, the keyframe pose is set as constant parameters, but in FBA step, it is reset as unknown parameters. The Jacobian matrix for keyframe poses is defined by (22). The Jacobian matrix for the dynamic motion matrix is defined in (23). Here, T_ϕ is an unknown parameters for the dynamic motion transformation in (24), and the rotation part follows Lie algebra SO(3). The rotation parameters are defined as $\left(w^{\phi}_{x}, w^{\phi}_{y}, w^{\phi}_{z}\right)$, and the translation parameters are defined as $\left(t^{\phi}_{x}, t^{\phi}_{y}, t^{\phi}_{z}\right)$ on the T_ϕ. The Jacobian of the map points for the dynamic object are defined in (25). X_ϖ represents unknown parameters for a dynamic map point.

$$\mathcal{J}_{\theta}\left(e^{d,l_d}_{j}\right) = \frac{\delta e^{d,l_d}_{j}}{\delta T_\theta} = \frac{\delta e^{d,l_d}_{j}}{\delta \tilde{m}} \cdot \frac{\delta \tilde{m}}{\delta \mathcal{K}} \cdot \frac{\delta \mathcal{K}}{\delta T_\theta} \tag{22}$$

$$\mathcal{J}_{\phi}\left(e^{d,l_d}_{j}\right) = \frac{\delta e^{d,l_d}_{j}}{\delta T_\phi} = \frac{\delta e^{d,l_d}_{j}}{\delta \tilde{m}} \cdot \frac{\delta \tilde{m}}{\delta \mathcal{K}} \cdot \frac{\delta \mathcal{K}}{\delta T_\theta} \cdot \frac{\delta T_\theta}{\delta T_\phi} \tag{23}$$

$$T_\phi = \begin{bmatrix} w^{\phi}_{x} & w^{\phi}_{y} & w^{\phi}_{z} & t^{\phi}_{x} & t^{\phi}_{y} & t^{\phi}_{z} \end{bmatrix} \tag{24}$$

$$\mathcal{J}_{\varpi}\left(e^{d,l_d}_{j}\right) = \frac{\delta e^{d,l_d}_{j}}{\delta X_\varpi} = \frac{\delta e^{d,l_d}_{j}}{\delta \tilde{m}} \cdot \frac{\delta \tilde{m}}{\delta \mathcal{K}} \cdot \frac{\delta \mathcal{K}}{\delta T_\theta} \cdot \frac{\delta T_\theta}{\delta T_\phi} \cdot \frac{\delta T_\phi}{\delta X_\varpi} \tag{25}$$

2) IS MOTION?

In this step, the object motion state (S_d) is determined using the motion information of the d-th potential dynamic object estimated by OMBA in F_i. The estimated transformation matrix of a dynamic object contains 6-DoF rotation and translation. We determine the state of the dynamic object as S_d shown in (26), depending on the magnitude of the rotation and translation. S_d represents the motion state of $\Delta T^{d}_{w_j,w_{\bar{j}}}$, the *stop* state is set to zero, and the *moving* state is set to one. The *moving* state of the estimated potential dynamic object is decided by threshold values, $\left\{\tau^{d}_{R}, \tau^{d}_{t}\right\}$ for rotation and translation, respectively. Dynamic object motion can be divided into a rotation matrix ($R^{d}_{w_j,w_{\bar{j}}}$) and a translation vector($t^{d}_{w_j,w_{\bar{j}}}$) in which $\Delta T^{d}_{w_j,w_{\bar{j}}}$ is represented as (27). $R^{d}_{w_j,w_{\bar{j}}}$ and $t^{d}_{w_j,w_{\bar{j}}}$ are the 3D rotation matrix and translation vector of the dynamic object observed in the j-th frame with respect to the $\bar{j}$-th frame where the d-th dynamic object was first observed. If the d-th potential dynamic object is stationary, the ideal values of the rotation matrix and translation vector are shown in (28). To compare the threshold values for object motion, each scale for rotation and translation is defined as in (29). Definition of the Norm L2 function for the matrix is defined in (30). Matrix M represents an arbitrary matrix for Norm L2 description, and m represents an element for M as in (30). Here, r and c represent the indices for row and column, respectively.

$$S_d \in \{stop\,(0)\,, moving\,(1)\} \tag{26}$$

$$T^{d}_{w_j,w_{\bar{j}}} = \begin{bmatrix} R^{d}_{w_j,w_{\bar{j}}} & t^{d}_{w_j,w_{\bar{j}}} \\ 0_{1\times3} & 1 \end{bmatrix} \tag{27}$$

$$R^{d}_{w_j,w_{\bar{j}}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, t^{d}_{w_j,w_{\bar{j}}} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{28}$$

$$S_R^d = \left\| R^d_{w_j,w_{\bar{j}}} - I_{3\times 3} \right\|_2, S_t^d = \left\| t^d_{w_j,w_{\bar{j}}} \right\|_2 \tag{29}$$

$$M = \begin{bmatrix} m_{(1,1)} & \cdots & m_{(1,c)} \\ \vdots & \ddots & \vdots \\ m_{(r,1)} & \cdots & m_{(r,c)} \end{bmatrix}, \|M\|_2 = \sum_{r=1}^{N_r} \sum_{c=1}^{N_c} \left(m_{(r,c)}\right)^2 \tag{30}$$

The motion state S_d is decided by comparing the thresholds for rotation and translation as shown in (31). In this study, the threshold values are set as $\tau_R^d = 3.0$ and $\tau_t^d = 2.0$. The threshold values of $\left(\tau_R^d, \tau_t^d\right)$ was experimentally set by considering the scale values S_R^d and S_t^d for the rotation matrix and translation vector of stationary dynamic objects from the KITTI odometry dataset. In the case of a parked stationary car, 2.827 for S_R^d and 1.857 for S_t^d were estimated in most cases. S_R^d and S_t^d for each vehicle are calculated with higher values of all vehicles which are moving in the highway environment of Sequence 01 in the KITTI odometery dataset. The threshold values $\left(\tau_R^d, \tau_t^d\right)$ are set as constants for all test datasets in the paper.

$$S_d = \begin{cases} 1, & if \left(S_R^d > \tau_R^d\right) or \left(S_t^d > \tau_t^d\right) \\ 0, & otherwise \end{cases} \tag{31}$$

3) FUSION BA

Fusion BA (FBA) simultaneously minimizes reprojection errors for background points, traffic sign points, dynamic object points, dynamic object motions, and keyframe poses. In other words, as the last step of LDBA, both static and dynamic map points, dynamic object motion, and keyframe poses are simultaneously optimized to reconstruct map points and estimate the motions of dynamic objects and keyframe poses. FBA combines three residuals for static point, traffic sign constraint, and dynamic object motion as in (32). The keyframe pose $\{F_i, \mathcal{F}_c\}$ intitialized as constant parameters in the OMBA step is optimized in this step. In addition, if the motion state of $\Delta T^d_{w_j,w_{\bar{j}}}$ for a potential dynamic object is decided as *stop* state in IMQ, it is set as constant in this optimization step.

$$\begin{aligned} min \sum_{j\in C} \left(\sum_{l_s \in P_j^s} \rho \left(\left\| e_j^{s,l_s} \right\|^2_{\sum_j^{l_s}} \right) \right) + \sum_{t\in Q_t} \sum_{l_t \in X_i^t} \rho \left(\left\| e_t^{l_t} \right\|^2 \right) \\ + \sum_{j\in C} \left(\sum_{l_d \in P_d^j} \rho \left(\left\| e_j^{d,l_d} \right\|^2_{\sum_j^{d,l_d}} \right) \right) \end{aligned} \tag{32}$$

VII. EXPERIMENTAL RESULTS

To evaluate the performance of camera trajectory estimation of the proposed TDO-SLAM, quantitative and qualitative evaluations are performed on three benchmark datasets, KITTI odometry dataset, KITTI raw dataset, and Complex Urban dataset. Each dataset contains stereo RGB images synchronized at 10Hz and high-precision GPS data. GPS data is used as Ground-Truth (GT) data of the camera pose. The KITTI Odometry dataset contains twenty-one sequences, however GT pose data is provided for only eleven sequences. The KITTI raw dataset is collected in various locations, but has fewer images and shorter playtime than the KITTI odometry dataset. We evaluate using seven sequences with dynamic environments in the KITTI raw dataset. As the third dataset, the stereo images from the Complex Urban dataset are used for evaluation. The Complex Urban dataset consists of a very dynamic and long-term dataset compared to the KITTI dataset. We choose the Complex Urban dataset to demonstrate robust and accurate visual odometry results in very dynamic road environments.

TABLE 1. KITTI odometry dataset evaluation (APE, unit: meter).

Seq.	ORB-SLAM2	ORB-SLAM3[25]	ORB-SLAM3[47]	DynaSLAM	TDO-SLAM
00	0.918	0.929	4.244	0.891	0.802
01	10.777	×	12.242	9.234	2.991
02	2.865	3.819	6.756	2.904	2.550
03	0.328	0.370	1.275	0.395	0.272
04	0.200	0.202	0.242	0.154	0.153
05	0.398	0.388	2.069	0.372	0.337
06	0.844	0.646	1.982	0.431	0.788
07	1.033	0.436	1.287	0.437	0.436
08	3.568	3.481	3.895	2.940	2.787
09	3.552	0.964	3.014	3.281	0.930
10	1.360	1.518	1.261	1.243	1.239
Avg.	2.349	1.275*	3.479	2.026	1.208

*Sequence 01 is excluded in the average calculation.

In this study, we use RMSE Absolute Trajectory Error (ATE) and RMSE Relative Pose Error (RPE) as evaluation metrics. The average execution time of TDO-SLAM is evaluated including the instance segmentation module. The evaluation tools [45], [46] are used for fair quantitative measurement. The proposed method is compared with ORB-SLAM2, ORB-SLAM3, and DynaSLAM provided via open-source code and evaluation results from two recent papers. All algorithms are performed on desktop PC with intel i7-9700K (3.60 GHz), TITAN RTX 24GB, and 16GB RAM. ORB-SLAM2 and TDO-SLAM's operating systems run on Ubuntu 18.04, while DynaSLAM runs on Ubuntu 16.04.

A. KITTI ODOMETRY DATASET

The quantitative and qualitative results are compared and analyzed using sequence number from 00 to 10 in the KITTI odometry dataset. The KITTI odometry dataset includes urban and highway environments. Most of the dataset consists of static environment and includes dynamic objects in some parts. Table 1 presents the quantitative results for the KITTI odometry dataset. TDO-SLAM is compared with the baseline method ORB-SLAM2 and DynaSLAM. RMSE ATE of ORB-SLAM3 is also compared by referring two papers [25], [47].

In Table 1, the proposed TDO-SLAM shows the best results in almost all sequences. The KITTI odometry dataset contains mostly static environments and many scenes with parked cars. Given that stationary and parked cars contain static feature points, they can be useful for estimating the camera's pose. DynaSLAM does not use all of the image

TABLE 2. KITTI raw dataset evaluation (APE, unit: meter).

Seq.	ORB-SLAM2	DynaSLAM	TDO-SLAM
0926-0009	0.731	1.148	0.648
0926-0013	0.290	0.312	0.270
0926-0014	0.828	0.517	0.532
0926-0051	0.294	0.293	0.294
0926-0101	8.670	9.877	6.731
0929-0004	0.331	0.247	0.267
1003-0047	8.563	0.481	1.377
Avg. error	2.815	1.839	1.446

features contained in stationary road objects such as parked vehicles. In contrast, TDO-SLAM determines the motion state of the potential dynamic objects and actively utilize the features included in the dynamic objects.

Especially, sequence 01 contains several dynamic objects. In the case of ORB-SLAM2 and ORB-SLAM3, a large error occurs in this sequence because of many dynamic objects. In sequence 06, the error of ORB-SLAM3 and DynaSLAM is less the proposed method. In this sequence, there is no dynamic object and only one traffic sign. Therefore, the average error must be similar to ORB-SLAM2 or ORB-SLAM3 because almost all frames consist of static scenes. Except the sequence 06, the proposed method shows better performance than ORB-SLAM3 and DynaSLAM. This means that the proposed method provides stable performance in either static or dynamic environment.

B. KITTI RAW DATASET

The KITTI Raw dataset is used as the second evaluation dataset. The KITTI Raw dataset consists of City, Residential, Road, Campus, and Person categories and contains various category sequences. However, in our experiments, we only evaluate sequences that are representatively used for dynamic SLAM evaluation [9], [10], [11], [12]. These sequences contain many moving dynamic objects, thus, it is a useful dataset for performance evaluation. However, the data playback time is shorter than that of the KITTI Odometry dataset. Table 2 shows quantitative evaluations for the KITTI Raw dataset. TDO-SLAM achieves the lowest ATE. It also provides stable results in all sequences. Sequence 1003-0047 continuously detects moving dynamic objects from the first frame. In the initial stage of the SLAM system, a moving dynamic object can cause a large drift error. In this reason, ORB-SLAM2 suffers from a large error. In contrast, DyanSLAM and TDO-SLAM show relatively lower errors. This experimental result shows that the initialization of the SLAM system in the dynamic environment is also critical point.

C. COMPLEX URBAN DATASET

Most previous dynamic SLAM studies [9], [10], [11], [12] have been tested and evaluated on simple environments such as the KITTI dataset. The KITTI dataset contains moving dynamic objects, however the number of moving objects is small and even their motion is not large. Furthermore, because there is little change in lighting condition, many strong static features can be detected, thus track loss does not occur.

TABLE 3. Complex Urban dataset evaluation with loop closure (APE, unit: meter).

Seq.	Location	Length (km)	ORB-SLAM2	TDO-SLAM
39	Pangyo	11.06	×	28.049
38	Pangyo	11.42	68.399	28.541
37	Seoul	11.77	784.695	58.991
35	Seoul	3.2	×	23.246
34	Yeouido	7.8	110.385	149.070
33	Yeouido	7.6	36.934	16.720
32	Yeouido	7.1	20.011	20.190
31	Gangnam	11.4	×	402.882
30	Gangnam	6.0	245.088	11.497
29	Pangyo	3.6	53.037	34.528
28	Pangyo	11.47	24.967	15.504
27	Dongtan	5.4	18.083	14.023
26	Dongtan	2.5	24.993	11.653
Avg. error			138.659	62.684

TABLE 4. Complex Urban dataset evaluation without loop closure (APE, unit: meter).

Seq.	Location	Length (km)	ORB-SLAM2	TDO-SLAM
39	Pangyo	11.06	×	133.618
38	Pangyo	11.42	169.21	124.328
37	Seoul	11.77	854.37	41.281
35	Seoul	3.2	×	14.364
34	Yeouido	7.8	334.182	90.607
33	Yeouido	7.6	36.01	16.979
32	Yeouido	7.1	38.211	35.322
31	Gangnam	11.4	×	456.154
30	Gangnam	6.0	248.328	12.314
29	Pangyo	3.6	54.760	26.684
28	Pangyo	11.47	38.115	49.201
27	Dongtan	5.4	39.963	27.134
26	Dongtan	2.5	21.581	11.852
Avg. error			183.473	79.988

In order to evaluate TDO-SLAM using a long-term sequence with changing light and many dynamic objects, we use the Complex Urban dataset. Thirteen sequences of urban areas in the Complex Urban dataset are used. The Complex Urban dataset is provided to be read in real-time within Robot Operating System (ROS). However, DynaSLAM cannot run in real-time, it is not possible to use in this complex urban dataset. Therefore, ORB-SLAM2 and TDO-SLAM are only evaluated with this dataset.

Table 3 shows quantitative results, including the loop closure module, while Table 4 shows results excluding the loop closure module. TDO-SLAM outperforms ORB-SLAM2 in all sequences. In addition, ORB-SLAM2 results in track loss in Sequences 31, 35, and 39. Sequences 31 and 35 cause large error in the initialization phase because a large number of moving dynamic objects are observed from the beginning of the sequences. In Sequence 39, the

TABLE 5. Runtime analysis of the segmentation, tracking, and Local BA part compared to ORB-SLAM2, DynaSLAM in KITTI odometry dataset (unit: millisecond).

Seq.	ORB-SLAM2		DynaSLAM			TDO-SLAM		
	Tracking	Local BA	Seg.	Tracking	Local BA	Seg.	Tracking	Local BA
00	45.782	101.226	737.264	68.433	122.712	24.205	46.407	185.203
01	48.291	61.997	719.201	76.532	62.988	22.489	46.796	97.117
02	47.105	80.869	706.256	68.473	94.425	23.066	47.27	182.420
03	54.477	126.222	708.785	65.631	144.066	23.419	50.544	196.421
04	62.107	133.237	704.362	67.956	175.512	22.291	53.686	162.459
05	52.273	107.902	723.357	70.609	113.473	23.110	52.373	201.591
06	59.496	119.896	741.366	75.297	119.59	22.914	53.685	198.135
07	54.255	83.158	753.301	62.994	89.523	24.632	49.584	194.826
08	53.085	78.604	739.246	70.139	84.368	23.903	51.035	197.756
09	49.363	69.542	712.859	64.597	82.087	22.985	50.714	178.146
10	47.075	67.07	739.673	70.823	64.358	23.537	47.686	153.471
Avg.	52.119	93.611	725.97	69.226	104.827	23.322	49.98	177.049

map points are initialized in a static environment. However, many dynamic objects appear in the middle of sequence, which causes drift errors and eventually the camera tracking fails. However, TDO-SLAM shows that camera tracking can be successfully performed in complex urban environments without track loss.

Figure 12 shows estimated trajectories of ORB-SLAM2 and TDO-SLAM in Sequence 38 of the Complex Urban dataset. Because ORB-SLAM2 fails in Sequence 39, we show a similar sequence for visual comparison of estimated trajectories. Compared with the ground truth, TDO-SLAM shows better estimation result. At the bottom right corner of the trajectory as shown in Fig. 12, the proposed method yields a larger error than ORB-SLAM2, because of feature tracking errors due to pure black or white colored sedans running across the camera.

D. RUNTIME ANALYSIS

Processing time is also measured as shown in Table 5 using Sequences from 00 to 10 of the KITTI odometry dataset. The measured module is the Local BA step of the segmentation module, Tracking module, and Local mapping module. ORB-SLAM2 is excluded because it does not have a segmentation module. The Loop-closure and Full BA steps are also excluded from the analysis because they act as backends and are completely identical to those of ORB-SLAM2. Because the input of the tracking module is the instance segmentation mask, the total computation time of the segmentation module and the tracking module represents the estimated time of camera tracking. As the speed of the stereo camera of all datasets is 10Hz, it can operate in real time if the total processing time of the segmentation module and tracking module is less than 100ms. TDO-SLAM operates in real-time at approximately 13 fps, as presented in Table 5.

Most of the time in the local mapping module is spent in the Local BA step. In addition, all the other steps except Local

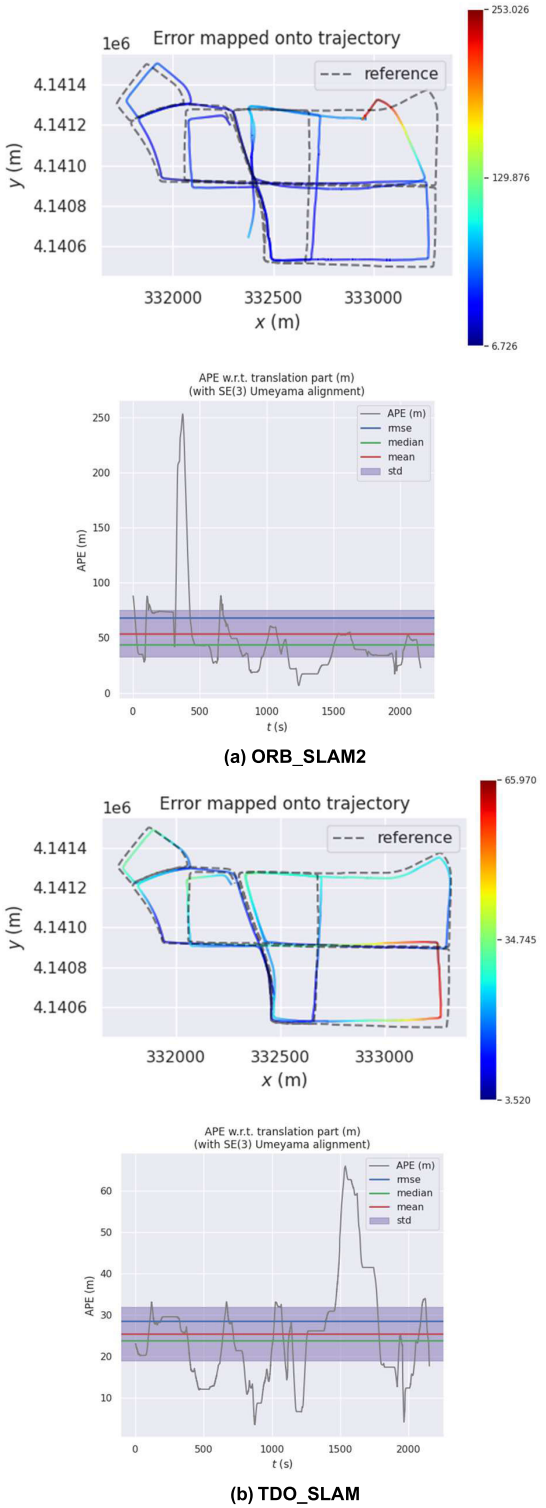

FIGURE 12. Comparison of each trajectory in Sequence 38 from the Complex Urban dataset.

BA in the local mapping module take time as the same as in ORB-SLAM2. Therefore, we measure the computation time for the Local BA step. Local BA of TDO-SLAM contains the traffic sign constraint. The local mapping module only

processes keyframes, not all input frames. Therefore, the proposed method can operate in real-time at five keyframes per second as shown in Table 5. For example, if the keyframes are five frames apart, then, our method can run in 25 fps.

VIII. DISCUSSION AND CONCLUSION

This study proposes TDO-SLAM, a real-time visual SLAM system with the consideration of dynamic objects and standing traffic sign constraint. The proposed system uses only a stereo camera without any IMU or GPS positioning sensor. Thus, the proposed system can be employed any autonomous vehicle equipped with only a stereo camera. TDO-SLAM utilizes a real-time instance segmentation method to detect potential dynamic objects and traffic signs. We consider various types of vehicles and pedestrians as potential dynamic objects. In the tracking thread module, TDO-SLAM based on the ORB-SLAM2 system extracts and matches ORB features for successive image frames. The extracted ORB feature is assigned attributes for ClassID and ObjectID using the instance segmentation mask. We also propose the Object Level Tracking method to track objects across successive frames and manage the global ObjectID. In the steps of map initialization and initial pose estimation of the camera, all ORB features included in the mask area of the potential dynamic object are removed as outliers, and initialization is performed with the remaining inlier features.

In the Local Mapping module, the Local BA part of ORB-SLAM2 is modified to LSBA and LDBA. LSBA calculates the poses and map points of both background keyframes and traffic signs through optimization. The standing traffic sign constraint is designed such that the map points of the traffic sign lie on a planar surface while simultaneously improving the performance of pose estimation. The result of LSBA is used as the initial value of LDBA. LDBA consists of OMBA, IMQ, and FBA. OMBA estimates motion of a potential dynamic object using a constant keyframe pose. In the IMQ step, the moving state is determined by calculating the rotation and translation scale of the estimated potential dynamic object motion. FBA performs optimization for background, traffic sign, map-point of the dynamic object, dynamic object motion, and keyframe pose.

To analyze the performance of the proposed method, we use three datasets, KITTI Odometry dataset, KITTI Raw dataset, and Complex Urban dataset. Experimental results show that the proposed TDO-SLAM is more robust and accurate compared with ORB-SLAM2 and DynaSLAM. TDO-SLAM runs with very small drift error in both static and dynamic environments. Especially, TDO-SLAM yields robust and accurate odometry estimation performance in the Complex Urban dataset which consists of very dynamic and long-term road image sequences.

REFERENCES

[1] S. Saeedi, "Navigating the landscape for real-time localization and mapping for robotics and virtual and augmented reality," *Proc. IEEE*, vol. 106, no. 11, pp. 2020–2039, Nov. 2018, doi: 10.1109/JPROC.2018.2856739.

[2] J. P. M. Covolan, A. C. Sementille, and S. R. R. Sanches, "A mapping of visual SLAM algorithms and their applications in augmented reality," in *Proc. 22nd Symp. Virtual Augmented Reality (SVR)*, Nov. 2020, pp. 20–29, doi: 10.1109/SVR51698.2020.00019.

[3] C. Theodorou, D. V. Velisavljevic, D. V. Dyo, and F. Nonyelu, "Visual SLAM algorithms and their application for Ar, mapping, localization and wayfinding *,**," *SSRN Electron. J.*, vol. 15, Mar. 2022, Art. no. 100222, doi: 10.2139/ssrn.4058786.

[4] U. Frese, R. Wagner, and T. Röfer, "A SLAM overview from a User's perspective," *KI Künstliche Intelligenz*, vol. 24, no. 3, pp. 191–198, Sep. 2010, doi: 10.1007/s13218-010-0040-4.

[5] 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, doi: 10.1109/TRO.2017.2705103.

[6] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 6, pp. 1052–1067, Jun. 2007, doi: 10.1109/TPAMI.2007.1049.

[7] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "DTAM: Dense tracking and mapping in real-time," in *Proc. Int. Conf. Comput. Vis.*, Nov. 2011, pp. 2320–2327, doi: 10.1109/ICCV.2011.6126513.

[8] 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, doi: 10.1109/TRO.2015.2463671.

[9] I. Ballester, A. Fontán, J. Civera, K. H. Strobl, and R. Triebel, "DOT: Dynamic object tracking for visual SLAM," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 11705–11711, doi: 10.1109/ICRA48506.2021.9561452.

[10] M. Gonzalez, E. Marchand, A. Kacete, and J. Royan, "TwistSLAM: Constrained SLAM in dynamic environment," *IEEE Robot. Autom. Lett.*, vol. 7, no. 3, pp. 6846–6853, Jul. 2022, doi: 10.1109/LRA.2022.3178150.

[11] B. Bescos, C. Campos, J. D. Tardos, and J. Neira, "DynaSLAM II: Tightly-coupled multi-object tracking and SLAM," *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 5191–5198, Jul. 2021, doi: 10.1109/LRA.2021.3068640.

[12] B. Bescos, J. M. Fácil, J. Civera, and J. Neira, "DynaSLAM: Tracking, mapping, and inpainting in dynamic scenes," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 4076–4083, Oct. 2018, doi: 10.1109/LRA.2018.2860039.

[13] H. Liu, R. A. R. Soto, F. Xiao, and Y. J. Lee, "YolactEdge: Real-time instance segmentation on the edge (Jetson AGX Xavier: 30 FPS, RTX 2080 Ti: 170 FPS)," 2020, *arXiv:2012.12259*.

[14] J. Jeong, Y. Cho, Y.-S. Shin, H. Roh, and A. Kim, "Complex urban dataset with multi-level sensors from highly diverse urban environments," *Int. J. Robot. Res.*, vol. 38, no. 6, pp. 642–657, May 2019, doi: 10.1177/0278364919843996.

[15] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, Sep. 2013, doi: 10.1177/0278364913491297.

[16] R. Scona, M. Jaimez, Y. R. Petillot, M. Fallon, and D. Cremers, "StaticFusion: Background reconstruction for dense RGB-D SLAM in dynamic environments," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 3849–3856, doi: 10.1109/ICRA.2018.8460681.

[17] T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison, "ElasticFusion: Dense SLAM without a pose graph," in *Robotics: Science and Systems*, 2015, doi: 10.15607/RSS.2015.XI.001.

[18] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 2, pp. 386–397, Feb. 2020, doi: 10.1109/TPAMI.2018.2844175.

[19] H. Guan, C. Qian, T. Wu, X. Hu, F. Duan, and X. Ye, "A dynamic scene vision SLAM method incorporating object detection and object characterization," *Sustainability*, vol. 15, no. 4, p. 3048, Feb. 2023, doi: 10.3390/su15043048.

[20] 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, doi: 10.1109/TRO.2021.3075644.

[21] G. Jocher. (2022). *Ultralytics/YOLOv5: V7.0 YOLOv5 SOTA Real-time Instance Segmentation (v7.0)*. [Online]. Available: Https://Github.Com/Ultralytics/Yolov5/Tree/V7.0

[22] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 573–580, doi: 10.1109/IROS.2012.6385773.

[23] Z. Xing, X. Zhu, and D. Dong, "DE-SLAM: SLAM for highly dynamic environment," *J. Field Robot.*, vol. 39, no. 5, pp. 528–542, Aug. 2022, doi: 10.1002/rob.22062.
[24] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520, doi: 10.1109/CVPR.2018.00474.
[25] L. Chen, Z. Ling, Y. Gao, R. Sun, and S. Jin, "A real-time semantic visual SLAM for dynamic environment based on deep learning and dynamic probabilistic propagation," *Complex Intell. Syst.*, vol. 9, no. 5, pp. 5653–5677, Oct. 2023, doi: 10.1007/s40747-023-01031-5.
[26] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6230–6239, doi: 10.1109/CVPR.2017.660.
[27] Y. Liu and J. Miura, "RDS-SLAM: Real-time dynamic SLAM using semantic segmentation methods," *IEEE Access*, vol. 9, pp. 23772–23785, 2021, doi: 10.1109/ACCESS.2021.3050617.
[28] S. Thrun, B. Wolfram, and D. Fox, *Probabilistic Robotics* (Intelligent Robotics and Autonomous Agents). Cambridge, MA, USA: MIT Press, 2005.
[29] P. Su, S. Luo, and X. Huang, "Real-time dynamic SLAM algorithm based on deep learning," *IEEE Access*, vol. 10, pp. 87754–87766, 2022, doi: 10.1109/ACCESS.2022.3199350.
[30] S. Yang and S. Scherer, "CubeSLAM: Monocular 3-D object SLAM," *IEEE Trans. Robot.*, vol. 35, no. 4, pp. 925–938, Aug. 2019, doi: 10.1109/TRO.2019.2909168.
[31] J. Huang, S. Yang, T.-J. Mu, and S.-M. Hu, "ClusterVO: Clustering moving instances and estimating visual odometry for self and surroundings," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 2165–2174, doi: 10.1109/CVPR42600.2020.00224.
[32] J. Zhang, M. Henein, R. E. Mahony, and V. Ila, "VDO-SLAM: A visual dynamic object-aware SLAM system," 2005, *arXiv:2005.11052*.
[33] X. Hu and J. Lang, "DOE-SLAM: Dynamic object enhanced visual SLAM," *Sensors*, vol. 21, no. 9, p. 3091, Apr. 2021, doi: 10.3390/s21093091.
[34] *NVIDIA TensorRT*, NVIDIA Developer, NVIDIA, Santa Clara, CA, USA, 2021.
[35] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 936–944, doi: 10.1109/CVPR.2017.106.
[36] T. Y. Lin, "Microsoft COCO: Common objects in context," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 740–755, doi: 10.1007/978-3-319-10602-1_48.
[37] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 3213–3223, doi: 10.1109/CVPR.2016.350.
[38] D. Tabernik and D. Skocaj, "Deep learning for large-scale traffic-sign detection and recognition," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 4, pp. 1427–1440, Apr. 2020, doi: 10.1109/TITS.2019.2913588.
[39] V. Lepetit, F. Moreno-Noguer, and P. Fua, "EPnP: An accurate O(n) solution to the PnP problem," *Int. J. Comput. Vis.*, vol. 81, no. 2, pp. 155–166, Feb. 2009, doi: 10.1007/s11263-008-0152-6.
[40] J. J. Moré, "The Levenberg–Marquardt algorithm: Implementation and theory," Argonne Nat. Lab., IL, USA, Tech. Rep. CONF-770636-1, 1978, doi: 10.1007/bfb0067700.
[41] C. Mei, G. Sibley, and P. Newman, "Closing loops without places," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2010, pp. 3738–3744, doi: 10.1109/IROS.2010.5652266.
[42] H. Strasdat, A. J. Davison, J. M. M. Montiel, and K. Konolige, "Double window optimisation for constant time visual SLAM," in *Proc. Int. Conf. Comput. Vis.*, Nov. 2011, pp. 2352–2359, doi: 10.1109/ICCV.2011.6126517.
[43] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G^2O: A general framework for graph optimization," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 3607–3613, doi: 10.1109/ICRA.2011.5979949.
[44] E. Malis and E. Marchand, "Experiments with robust estimation techniques in real-time robot vision," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2006, pp. 223–228, doi: 10.1109/IROS.2006.282572.
[45] H. Zhan, C. S. Weerasekera, J.-W. Bian, and I. Reid, "Visual odometry revisited: What should be learnt?" in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 4203–4210, doi: 10.1109/ICRA40945.2020.9197374.
[46] M. Grupp. *Evo: Python Package for the Evaluation of Odometry and SLAM*. Accessed: Aug. 5, 2023. [Online]. Available: https://github.com/MichaelGrupp/evo
[47] D. Hug, P. Bänninger, I. Alzugaray, and M. Chli, "Continuous-time stereo-inertial odometry," *IEEE Robot. Autom. Lett.*, vol. 7, no. 3, pp. 6455–6462, Jul. 2022, doi: 10.1109/LRA.2022.3173705.

SOON-YONG PARK received the B.S. and M.S. degrees from the School of Electronics Engineering, Kyungpook National University, Daegu, South Korea, in 1991 and 1993, respectively, and the Ph.D. degree from Stony Brook University, NY, USA, in 2003. He was with the Korea Atomic Energy Research Institute, from 1993 to 1999, and the Electronics and Telecommunications Research Institute, from 2004 to 2005. He was a Professor with the School of Computer Science and Engineering, Kyungpook National University, from 2005 to 2019, where he is currently a Professor with the School of Electronics Engineering. His research interests include 3-D scanning, 3-D registration, and robot vision.

JUNESUK LEE received the B.S. degree from the Department of Computer Science and Engineering, Hanbat National University, Daejeon, South Korea, in 2017, the M.S. degree from the School of Computer Science and Engineering, Kyungpook National University, Daegu, South Korea, in 2019, and the Ph.D. degree from the School of Electronics and Electrical Engineering, Kyungpook National University, in 2023. He is currently a Research Staff with 42dot, Seoul, South Korea. His research interests include robot vision, deep learning, and visual-inertial localization and mapping.