Robust and Autonomous Stereo Visual-Inertial Navigation for Non-Holonomic Mobile Robots

Hee-Won Chae, Ji-Hoon Choi, and Jae-Bok Song, *Senior Member, IEEE*

***Abstract*—Unlike micro aerial vehicles, most mobile robots have non-holonomic constraints, which makes lateral movement impossible. Consequently, the vision-based navigation systems that perform accurate visual feature initialization by moving the camera to the side to ensure a sufficient parallax of the image are degraded when applied to mobile robots. Generally, to overcome this difficulty, a motion model based on wheel encoders mounted on a mobile robot is used to predict the pose of a robot, but it is difficult to cope with errors caused by wheel slip or inaccurate wheel calibration. In this study, we propose a robust autonomous navigation system that uses only a stereo inertial sensor and does not rely on wheel-based dead reckoning. The observation model of the line feature modified with vanishing-points is applied to the visual-inertial odometry along with the point features so that a mobile robot can perform robust pose estimation during autonomous navigation. The proposed algorithm, i.e., keyframe-based autonomous visual-inertial navigation (KAVIN) supports the entire navigation system and can run onboard without an additional graphics processing unit. A series of experiments in a real environment indicated that the KAVIN system provides robust pose estimation without wheel encoders and prevents the accumulation of drift error during autonomous driving.**

***Index Terms*—Autonomous navigation, visual-inertial systems, keyframes, wheeled mobile robots.**

I. Introduction

In recent years, with the rapid advancement of computer vision technology, cameras have increasingly replaced expensive laser scanners in the field of navigation. In particular, visual-inertial odometry (VIO), which estimates the pose of an agent using a tightly-coupled combination of a camera and an inertial measurement unit (IMU), has recently been applied to many micro aerial vehicles (MAVs) and handheld mobile devices [1]–[3]. In addition, some research was conducted to improve the accuracy of feature extraction and reduce the amount of computation in VIO [4]. However, most VIO studies have been validated mainly through the drone-based navigation. While MAVs have become a major application of the visual inertial navigation systems, autonomous wheeled mobile robots have not been explored thoroughly. The ceiling-vision-based navigation system [5] is a representative example of improving the pose estimation of a mobile robot by observing the ceiling with a camera. However, it cannot estimate the accurate pose when the drift error of the wheel encoders is excessive or when the robot is kidnapped.

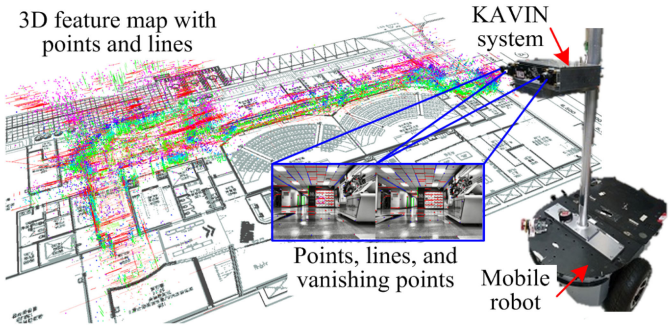


Fig. 1. The overall concept of the proposed KAVIN system mounted on the mobile robot, and the feature map being compared with the blueprint of the environment. The feature map consists of points, horizontal lines (red), and vertical lines (green).

Because a two-wheel differential drive robot cannot move sideways, owing to the non-holonomic constraints, the front camera mounted on the robot cannot secure sufficient image parallax during the navigation. Therefore, instead of the structure-from-motion algorithm that calculates the motion according to the camera images, it is more common to conduct feature initialization by calculating the motion through the wheel encoders. Wheel encoders can provide dead reckoning without noise but are very vulnerable to rotational motion and wheel slip. In particular, the kinematic model associated with wheel encoders is unreliable when the weight of the robot changes or when the robot encounters different floor conditions (e.g., carpeted areas). Additionally, if the two-wheel shafts of the mobile robot are not symmetrical to each other or if the diameters of the two wheels are measured incorrectly, the estimation of the robot's pose based on the wheel encoders involves a large drift error; thus, precise wheel calibration based on the hardware of the robot is required. Therefore, visual navigation systems that do not rely on wheel encoders are beneficial for wheeled mobile robots.

In this study, we propose a novel vision-based autonomous navigation solution that can be universally applied to various mobile robots by completely separating the navigation algorithm from the wheel encoders, as shown in Fig. 1. Simultaneous localization and mapping (SLAM) and autonomous navigation

Manuscript received August 13, 2019; revised January 28, 2020 and March 31, 2020; accepted June 16, 2020. Date of publication June 25, 2020; date of current version October 13, 2020. This work was supported by an IITP grant funded by the Korean Government (MSIT) (No. 2018-0-00622). The review of this article was coordinated by Dr. A. Chatterjee. *(Corresponding author: Jae-Bok Song.)*

Hee-Won Chae and Jae-Bok Song are with the School of Mechanical Engineering, Korea University, Seoul 02841, South Korea (e-mail: anakin722@korea.ac.kr; jbsong@korea.ac.kr).

Ji-Hoon Choi is with the School of Mechatronics, Korea University, Seoul 02841, South Korea (e-mail: johann7430@korea.ac.kr).

Digital Object Identifier 10.1109/TVT.2020.3004163

are highly dependent on the localization performance of the robot. Therefore, in this study, localization is performed by adopting line features based on vanishing-points, along with point features, so that a mobile robot robustly estimates the pose using only the camera and IMU. The line features classified through the three vanishing-points introduced in the projective geometry provide useful structural information in indoor environments and in outdoor environments with many buildings [6], [7]. However, these studies dealt only with VIO and SLAM, and not with the autonomous driving system of a mobile robot. Much consideration must be taken in order for a non-holonomic mobile robot to autonomously move on a map. Because the non-holonomic mobile robot moves in the forward direction, where the parallax of the image is relatively small, the Plücker line is supplemented through the vanishing-point and used for the feature initialization process of VIO. Additionally, to reduce the pose uncertainty of VIO, which increases with time, we designed a keyframe-based loop-closure algorithm to correct the pose and map when the robot revisits the same place. The map generated in this study is a topological map expressed by a keyframe, which provides information regarding the area that the robot can reach in a given environment. The proposed keyframe-based autonomous visual-inertial navigation (KAVIN) system, performs path planning and localization based on keyframes, so that mobile robots can perform robust autonomous navigation. A series of experiments were conducted to evaluate the performance of the KAVIN system. The main contributions of this study are as follows:

- The visual-inertial system employing a novel representation of three vanishing-points improves VIO performance for wheeled mobile robots compared with the existing methods.
- Using the proposed method, a vision-based system can be applied to various mobile robots regardless of the wheel kinematic model of a mobile robot.
- To our knowledge, this paper is the first to propose an entire navigation system that robustly executes autonomous navigation by applying a vision-based navigation system completely separated from the wheel encoders to the non-holonomic wheeled platform.

The remainder of this paper is organized as follows. The overall structure of the KAVIN system is presented in Section II. Section III covers the robust pose estimation of a robot using points and lines. In Section IV, the approach for keyframe-based localization and path generation is introduced. To evaluate the performance of KAVIN, several experiments were conducted, as described in Section V, and conclusions are drawn in Section VI.

II. Design of Navigation System for Mobile Robot

This section introduces the overall framework of the KAVIN system. From SLAM to motion control, the KAVIN system is designed by integrating the components necessary for the autonomous navigation of a mobile robot, as shown in Fig. 2. The KAVIN system consists of three nodes which are responsible for image feature extraction, localization and SLAM, and motion control, respectively. The whole system is designed to systematically share data between the nodes.

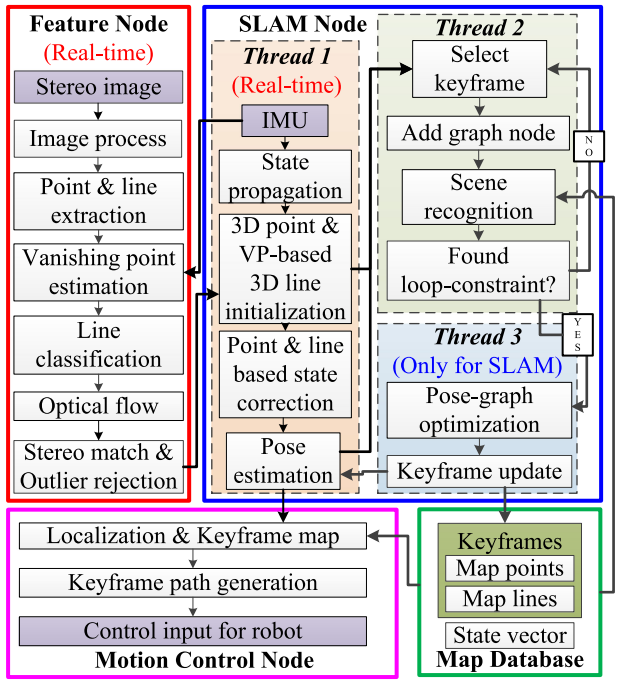


Fig. 2. Overall structure of the KAVIN system.

A. Feature Node

Corner features, which are robustly extracted in the textured environment, are typically used with an image descriptor or an optical flow scheme to find the corresponding feature pairs. The Feature Node in Fig. 2 tracks the Harris corner [8] using the optical flow scheme, which reduces the amount of computation required for feature matching. To cope with illumination changes, a contrast-limited adaptive histogram equalization algorithm [9] is applied to the original image before the feature extraction.

Line features can provide orientation information that cannot be provided by point features, but high computing power is required for line feature matching. Additionally, there is no guarantee that the endpoints of the corresponding line features match each other exactly. Therefore, in this study, both endpoints of a line feature extracted from the line-segment detector [10] are regarded as point features, and the optical flow is applied to these points to track line features.

Among the extracted line features, there are lines representing the grid structure of the environment, known as structure lines [6]. To select only the lines with these characteristics and apply them to VIO, the KAVIN system classifies extracted line features by calculating the vanishing-points. The projective geometry has up to three vanishing-points, and their directions are orthogonal to each other. According to this principle, the line features can be classified into three types: vertical, lateral-horizontal, and forward-horizontal lines, as shown in Fig. 3. Before the equations are derived, the notation of the pinhole camera model

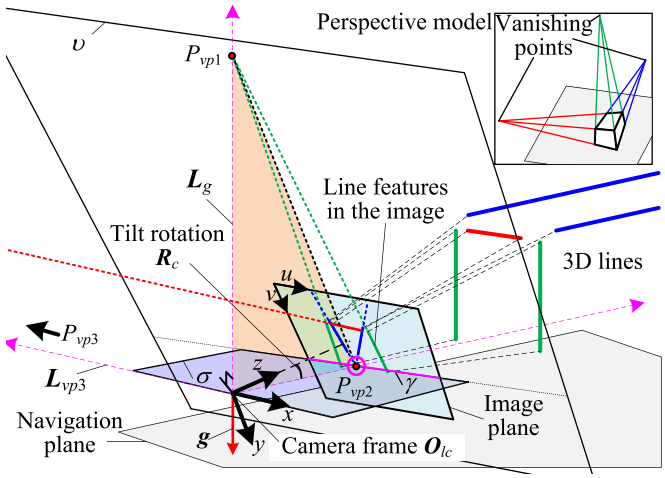


Fig. 3. The geometry of three vanishing-points and the classified lines from the pinhole camera model.

$\Pi : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ used in this study is introduced, as follows:

$$\boldsymbol{\Pi}(\boldsymbol{X}) = \frac{1}{\boldsymbol{z}} \begin{bmatrix} x \\ y \end{bmatrix}, \quad \text{where } \boldsymbol{X} = \begin{bmatrix} x \; y \; \boldsymbol{z} \end{bmatrix}^T, \; {}^\forall x, y, z \in \mathbb{R}. \quad (1)$$

Because the navigation plane on which a mobile robot travels is parallel to the floor, one of the three directions of the vanishing-points is parallel to the gravity vector and orthogonal to this plane. Therefore, the first vanishing-point $\boldsymbol{P}_{vp1}$ is on the straight line parallel to the direction of the gravity vector $\boldsymbol{g}$, which can be calculated using the IMU. Let the directions of the vanishing-points $\boldsymbol{P}_{vp1}$, $\boldsymbol{P}_{vp2}$, and $\boldsymbol{P}_{vp3}$ be $\boldsymbol{\mu}_{vp1}$, $\boldsymbol{\mu}_{vp2}$, and $\boldsymbol{\mu}_{vp3}$, respectively. Point $\boldsymbol{P}_{vp1}$ lies at the intersection of a line with the direction of $\boldsymbol{\mu}_{vp1}$ and a hyperplane υ extending from the image plane. If the stereo camera is inclined by the rotation matrix $\boldsymbol{R}_c$ from the direction facing the front, the line of gravity in the left camera frame $\{\boldsymbol{O}_{lc}\}$ is given as follows:

$$\boldsymbol{L}_g = t\boldsymbol{\mu}_{vp1} = t\boldsymbol{R}_c\hat{\boldsymbol{j}}, \; {}^\forall t \in \mathbb{R}. \quad (2)$$

The SO(3) rotation matrix, $\boldsymbol{R}_c$, can be calculated using the IMU when the robot is stationary. By rotating the standard basis vector $\hat{\boldsymbol{j}} = [0\ 1\ 0]^T$ of the left camera frame $\{\boldsymbol{O}_{lc}\}$, the vector $\boldsymbol{g}$ can be expressed from the frame $\{\boldsymbol{O}_{lc}\}$, as indicated by Eq. (2). Then, the vanishing-point $\boldsymbol{P}_{vp1}$ is given as

$$\boldsymbol{P}_{vp1} = \boldsymbol{\Pi}(\boldsymbol{K}_{lc}\boldsymbol{L}_g), \quad (3)$$

where $\boldsymbol{K}_{lc}$ is the intrinsic matrix of the left camera. The subscripts *lc* and *rc* denote the left and right cameras of the stereo rig, respectively. Among the extracted lines, the lines toward the vanishing-point $\boldsymbol{P}_{vp1}$ can be classified as the vertical lines.

The other two points $\boldsymbol{P}_{vp2}$ and $\boldsymbol{P}_{vp3}$ are observed alternately in the image according to the heading direction of the robot. As shown in Fig. 3, both vanishing-points can be found in the intersection γ of the planes υ and σ, where plane σ is given by

$$\boldsymbol{\mu}_{vp1}^T\boldsymbol{X} = 0, \; {}^\forall\boldsymbol{X} \in \mathbb{R}^3. \quad (4)$$

Plane σ is orthogonal to the first vanishing point $\boldsymbol{\mu}_{vp1}$ and the gravity vector $\boldsymbol{g}$ and passes through the origin. As shown in Fig. 3, since the second and third vanishing points lie on the intersection γ of plane σ and image plane υ, the intersection γ needs to expressed on the *uv* image coordinate system. Let $\boldsymbol{\mu}_{vp1}$ be $\boldsymbol{\mu}_{vp1} = [\mu_1\ \mu_2\ \mu_3]^T$. Since the image plane υ is represented by $z = 1$ in the frame $\{\boldsymbol{O}_{lc}\}$, substituting $z = 1$ in Eq. (4) yields

$$\mu_1 x + \mu_2 y + \mu_3 = 0. \quad (5)$$

Since x and y satisfying the above equation are the points ($z = 1$) on the image plane υ, they can be regarded as the normalized image coordinates (i.e., $\boldsymbol{X} = [x\ y\ 1]^T$). Therefore, the expression of x and y in the *uv* coordinate system using the camera intrinsic matrix $\boldsymbol{K}_{lc}$ is given by

$$\begin{bmatrix} u \\ v \end{bmatrix} = \boldsymbol{\Pi}\left(\boldsymbol{K}_{lc}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}\right) = \boldsymbol{\Pi}\left(\begin{bmatrix} fx + c_u \\ -\frac{f}{\mu_2}(\mu_1 x + \mu_3) + c_v \\ 1 \end{bmatrix}\right), \quad (6)$$

where c_u and c_v represent the camera centers with respect to the *uv* coordinates, and f represents the focal length of the camera. Since $u = fx + c_u$ is established, substituting this for x yields the relationship between u and v as follows:

$$\boldsymbol{P}_\gamma = \begin{bmatrix} u \; v(u) \end{bmatrix}^T = \begin{bmatrix} u \; -\frac{f}{\mu_2}\left(\frac{\mu_1(u - c_u)}{f} + \mu_3\right) + c_v \end{bmatrix}^T, \quad (7)$$

where $\boldsymbol{P}_\gamma$ is the intersection γ represented on the *uv* coordinate system. If either vanishing point $\boldsymbol{P}_{vp2}$ or $\boldsymbol{P}_{vp3}$ is observed in the camera image, it is necessarily lying on the line $\boldsymbol{P}_\gamma$ of the image. In addition, the vanishing point exists adjacent to the pixel coordinate where the previous vanishing point was extracted. By selecting only the line features that can converge to a single point that satisfies these assumptions, we can effectively estimate the coordinate of the vanishing point that is currently observed in the camera. To select the line features, the image is first divided into four quadrants, as shown in Fig. 4. Suppose that the vanishing point observed by the camera is $\boldsymbol{P}_{vp2}$. The image is divided up and down based on $\boldsymbol{P}_\gamma$ and left and right based on the u-coordinate of the previous vanishing point of $\boldsymbol{P}_{vp2}$. The line features converging to a point on the line $\boldsymbol{P}_\gamma$ have the negative slopes in the upper-right and lower-left quadrants (i.e., first and third quadrants) of the image, and positive slopes in the upper-left and lower-right quadrants (i.e., second and fourth quadrants). By selecting only these line features, the normalized equation can effectively estimate the vanishing point of the current image. Suppose that the selected line features are expressed as $a_i\, u + b_i\, v = c_i$; then, the point $\boldsymbol{P}_{vp2}$ can be

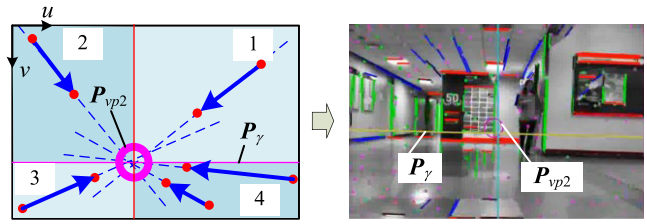


Fig. 4. Four quadrants in the image for the line selection to estimate the second vanishing-point.

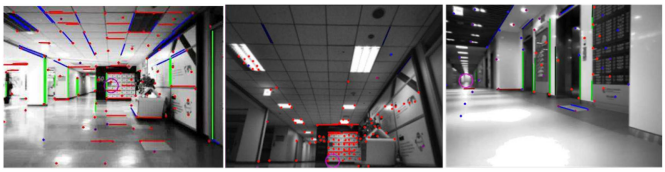


Fig. 5. Results for lines classified using vanishing-points from the (a) forward-looking image, (b) forward-looking image tilted upward, and (c) image during the rotation. The classified lines consist of vertical lines (green), horizontal lines in the lateral direction (red), horizontal lines in the forward direction (blue), and the vanishing-point (purple circle).

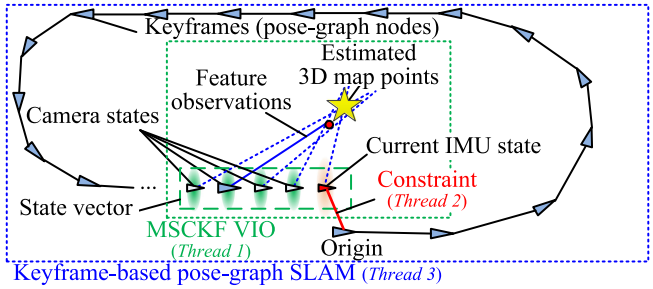


Fig. 6. Keyframe-based visual-inertial SLAM implemented in SLAM Node.

computed using the following linear equation:

$$\begin{bmatrix} a_1 & b_1 \\ : & : \\ a_N & b_N \end{bmatrix} \begin{bmatrix} u_{vp2} \\ v_{vp2} \end{bmatrix} = \begin{bmatrix} c_1 \\ : \\ c_N \end{bmatrix}, \tag{8}$$

where u_{vp2} and v_{vp2} are the pixel coordinates of the point $\boldsymbol{P}_{vp2}$. After $\boldsymbol{P}_{vp1}$ and $\boldsymbol{P}_{vp2}$ are computed, the third vanishing-point $\boldsymbol{P}_{vp3}$ can be obtained using the following equation:

$$\boldsymbol{L}_{vp3} = \boldsymbol{t}\boldsymbol{\mu}_{vp3}, \ {}^{\forall}\boldsymbol{t} \in \mathbb{R}, \tag{9}$$

where $\boldsymbol{\mu}_{vp3}$ is the vector obtained from the cross product of $\boldsymbol{\mu}_{vp1}$ and $\boldsymbol{\mu}_{vp2}$. The point $\boldsymbol{P}_{vp3}$ is the intersection of the line $\boldsymbol{L}_{vp3}$ and the plane υ, which is given by

$$\boldsymbol{P}_{vp3} = \boldsymbol{\Pi}(\boldsymbol{K}_{lc}\boldsymbol{\mu}_{vp3}). \tag{10}$$

As $\boldsymbol{P}_{vp2}$ approaches the image center (i.e., $u_{vp2} \rightarrow c_u$), the point $\boldsymbol{P}_{vp3}$ is placed at infinity in the x-direction of the coordinate $\{\boldsymbol{O}_{lc}\}$, and the vector $\boldsymbol{\mu}_{vp3}$ becomes parallel to the horizon of the image. Fig. 5 shows the results for the classified lines. Because the gravity vector is always orthogonal to the ground (i.e., the navigation plane) in the navigation of a mobile robot, the vector $\boldsymbol{\mu}_{vp1}$ always indicates the upward direction of the robot. Therefore, the other two vectors, $\boldsymbol{\mu}_{vp2}$ and $\boldsymbol{\mu}_{vp3}$, are always parallel to the ground.

B. Map Database and SLAM Node

Keyframes are databases generated by storing only the useful sensor data and are widely used in the field of visual SLAM. In this study, the keyframe is created when the following conditions are satisfied.

- There are over twenty three-dimensional (3D) points that have an ORB descriptor extracted in the image.
- The distance from the latest keyframe to the current keyframe is larger than 50 cm.

It is recommended that keyframes are generated as periodically as possible, but if multiple keyframes are allowed to be generated in the adjacent area, redundant keyframes are likely to be created. Therefore, it is desirable to generate them based on the appropriate criteria. In this study, if at least twenty 3D points with image descriptors appear in the image, it is determined that they are appropriate as keyframes. The keyframes are used to perform loop-closures, and less than twenty 3D points are disadvantageous for calculating loop constraints. However, when a robot travels in an environment with poor image features, keyframes may not be generated continuously due to the small number of features. Therefore, the additional conditions are set so that the distance between the keyframes does not exceed 50 cm at maximum. These parameters can vary depending on the environment, so it is desirable to determine them experimentally.

A set of keyframes generated by the KAVIN system is used as a topological map for robot navigation. These keyframes contain sensor data that are beneficial to the navigation and are stored in the Map Database in Fig. 2. To generate the keyframe-based topological map, the SLAM Node of the KAVIN system selects the keyframes satisfying the aforementioned conditions and performs loop-closure based on graph optimization. Loop-closure is a technique for modifying the topological map by correcting the poses of the graphs (i.e., keyframes). In this study, only the keyframes are defined as the graph nodes, and the features are not included in the graph; thus, simple and fast graph optimization is performed. Fig. 6 shows the idea of the keyframe-based SLAM adopted in KAVIN.

The SLAM Node consists of three threads, which are responsible for localization, loop-closure, and graph optimization, respectively. Fig. 6 shows the role of three threads of the SLAM Node for mapping. In the case of localization, we use the proposed VIO, which estimates the pose of the robot by adopting the point features and line features extracted from the Feature Node. This operation is conducted in Thread 1, as shown in Fig. 2, in real-time. The details of the VIO are presented in Section III.

Keyframes are selected from Thread 2. When the visual features extracted from the Feature Node and the current pose of a robot calculated in Thread 1 are passed to Thread 2, keyframes are selected according to the foregoing conditions. The selected keyframes are defined as the pose-graph nodes and added to the Map Database. Thread 3 requires a loop-constraint that connects a pair of graph nodes located in the same place to conduct graph optimization. The KAVIN system performs place recognition based on DBoW2 [11] to find a pair of keyframes created at the same place. DBoW2 is an algorithm for classifying ORB descriptors [12], and KAVIN acquires the ORB descriptor according to the Harris corners. When two keyframes are successfully paired through the place recognition, the relative pose between them is calculated through the PnP (perspective-n-points) algorithm [13] to form a loop-constraint. PnP is an

algorithm that estimates a rotation matrix and a translation vector that minimizes the reprojection error between the pixel coordinates of the features extracted from the current image and those calculated by projecting the corresponding 3D features onto the current image. When Thread 2 successfully calculates the loop-constraint, Thread 3 corrects the poses of all keyframes stored in the Map Database through the incremental smoothing and mapping (iSAM) method [14].

C. Motion Control Node

The current pose and path of the robot are provided by the SLAM Node and Map Database, respectively. The path is generated by connecting only the keyframes necessary for the robot to reach the destination among the keyframes existing in the topological map, as discussed in Section IV. To control the wheel actuators, the current pose of the robot and the desired pose to guide the robot are required. The desired pose is selected on the generated path and transmitted to the vector field histogram algorithm [15] of the Motion Control Node along with the current pose of the robot.

III. Pose Estimation Using Visual-Inertial Sensor

This section introduces the VIO algorithm implemented in Thread 1 of KAVIN. The point and line features extracted from the Feature Node are used in the Kalman filter observation model and are significant for estimating the current pose of the robot. The point features are applied in the same way as most VIO algorithms, but in the case of line features, a novel line observation model based on a vanishing-point is introduced in this study, considering that a mobile robot travels on the ground.

A. Extended Kalman Filter for VIO

The main filter used in KAVIN's VIO is a multi-state constraint Kalman filter (MSCKF) and has a structure similar to that reported in [2], [16]. The MSCKF is one of the representative Kalman filters used to integrate an IMU with a camera. The state vector $\boldsymbol{X}_s$ of the MSCKF consists of the IMU state and the camera state, as follows:

$$\hat{\boldsymbol{X}}_I = \begin{bmatrix} \hat{\boldsymbol{p}}_I^T & \hat{\boldsymbol{v}}_I^T & \hat{\boldsymbol{q}}_I^T & \hat{\boldsymbol{b}}_{I,a}^T & \hat{\boldsymbol{b}}_{I,\omega}^T \end{bmatrix}^T, \tag{11}$$

$$\hat{\boldsymbol{X}}_{lc,i} = \begin{bmatrix} \hat{\boldsymbol{p}}_{lc,i}^T & \hat{\boldsymbol{q}}_{lc,i}^T \end{bmatrix}^T, \text{ and} \tag{12}$$

$$\hat{\boldsymbol{X}}_s = \begin{bmatrix} \hat{\boldsymbol{X}}_I^T & | & \hat{\boldsymbol{X}}_{lc,1}^T & \dots & \hat{\boldsymbol{X}}_{lc,i}^T \end{bmatrix}^T, \tag{13}$$

where $\boldsymbol{p}_I$, $\boldsymbol{v}_I$, $\boldsymbol{q}_I$, $\boldsymbol{b}_{I,a}$, and $\boldsymbol{b}_{I,\omega}$ represent the position, velocity, quaternion, and two biases of the IMU, respectively. The vectors $\boldsymbol{p}_{lc,i}$ and $\boldsymbol{q}_{lc,i}$ represent the positions and quaternions, respectively, of the i-th camera state. The symbol '^' above the variable denotes the estimated vectors of the state. To avoid the singularity of the rotation, the estimated vectors, which are the original form of the state, express the rotation using the quaternion, but the number of quaternion variables must be reduced from 4 to 3 to derive the Jacobian matrix to be used in the Kalman filter. Therefore, we define the error-state vectors

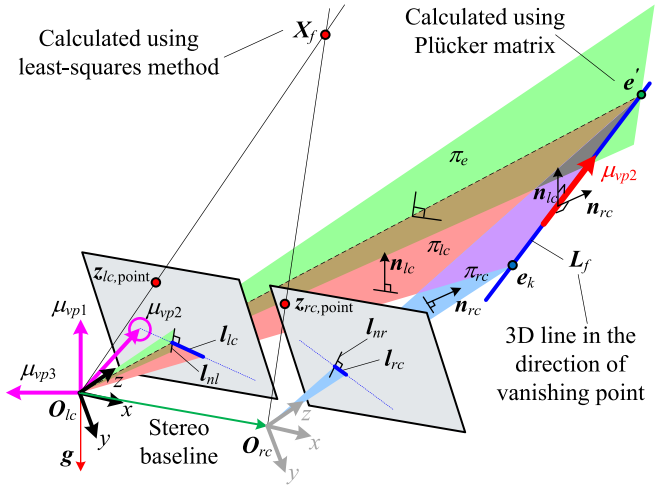


Fig. 7. Observation model for the point and line features.

of the state as follows:

$$\tilde{\boldsymbol{X}}_I = \begin{bmatrix} \tilde{\boldsymbol{p}}_I^T & \tilde{\boldsymbol{v}}_I^T & \delta\tilde{\boldsymbol{\theta}}_I^T & \tilde{\boldsymbol{b}}_{I,a}^T & \tilde{\boldsymbol{p}}_{I,\omega}^T \end{bmatrix}^T \text{ and} \tag{14}$$

$$\tilde{\boldsymbol{X}}_{lc,i} = \begin{bmatrix} \tilde{\boldsymbol{p}}_{lc,i}^T & \delta\tilde{\boldsymbol{\theta}}_{lc,i}^T \end{bmatrix}^T. \tag{15}$$

The symbol '$\sim$' represents the error-state vectors of the state. For a small-angle rotation, the quaternion $\delta\boldsymbol{q} \in \mathbb{R}^4$ can be expressed as $\delta\theta \in \mathbb{R}^3$, which is the error quaternion. The relationship between $\delta\theta$ and $\delta\boldsymbol{q}$ is as follows:

$$\delta\tilde{\boldsymbol{q}} \cong \begin{bmatrix} \frac{1}{2}\delta\tilde{\boldsymbol{\theta}}^T & 1 \end{bmatrix}^T. \tag{16}$$

When the Kalman filter is predicted and updated, each element of the state vector (except for the quaternion) is applied to the filter through the standard additive error (i.e., $\tilde{\boldsymbol{u}} = \boldsymbol{u} - \hat{\boldsymbol{u}}$), while the quaternion is applied through the quaternion multiplication: $\boldsymbol{q} = \delta\boldsymbol{q} \otimes \hat{\boldsymbol{q}}$.

To estimate the position, velocity, and orientation of the IMU from the IMU input, the discrete data of the accelerometer and gyroscope are numerically integrated using a fourth-order Runge–Kutta algorithm. In contrast to MAVs, a mobile robot is exposed to various external forces because it is in contact with the ground. These external forces can increase the velocity of the state by exciting the accelerometer of the IMU. To mitigate this, low-pass filtering is applied to the accelerometer. The remaining prediction processes related to the propagation of the state and covariance are omitted here (refer to [2], [16]).

B. Feature Initialization

To establish an observation model, 3D points and 3D lines must be initialized. For the point features, the initialization is performed using least-squares triangulation. This scheme can quickly estimate the coordinates of the 3D point $\boldsymbol{X}_f$. For the line features, the direction of the line is expressed using the vanishing-point directions $\boldsymbol{\mu}_{vpi}$, $(i = 1, 2, 3)$ calculated from the Feature Node as shown in Fig. 7. Let the line feature extracted from the left image be $\boldsymbol{l}_{lc} = [\boldsymbol{z}_{lc,1}\ \boldsymbol{z}_{lc,2}]$, where $\boldsymbol{z}_{lc,1}$ and $\boldsymbol{z}_{lc,2}$ represent the endpoints of the line feature extracted from

the left image. Let π_{lc} be the plane defined by the line feature $\boldsymbol{l}_{lc}$ and the origin of the left camera frame $\{\boldsymbol{O}_{lc}\}$. Similarly, $\boldsymbol{l}_{rc} = [\boldsymbol{z}_{rc,1}\ \boldsymbol{z}_{rc,2}]$ and π_{rc} represent the line and plane obtained when observing the same 3D line as $\boldsymbol{l}_{lc}$ in the right image, respectively. First, the same procedure as point feature initialization is applied to both endpoints to express both endpoints of a line in the 3D space. Let these two 3D endpoints be $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$. Among these, the endpoint having the larger parallax between the left and right images of the stereo camera is selected. The 3D line $\boldsymbol{L}_f$ obtained through the selected endpoint $\boldsymbol{e}_k$ ($k = 1$ or 2) and the vanishing-point direction $\boldsymbol{\mu}_{vpi}$ is given by

$$\begin{aligned}&\boldsymbol{L}_f(t) = t\boldsymbol{\mu}_{vpi} + \boldsymbol{e}_k\\ &\text{where}\\ &k = \underset{k=1,2}{\operatorname{argmax}} \left\| \boldsymbol{z}_{lc,k} - \boldsymbol{z}_{rc,k} \right\|,\ ^{\forall}t \in \mathbb{R}.\end{aligned} \tag{17}$$

The endpoint on the opposite side of $\boldsymbol{e}_k$ is not located on the line defined in Eq. (17) due to the measurement noise and must be newly calculated. For example, if the previously selected endpoint is $\boldsymbol{e}_1$, the other endpoint $\boldsymbol{e}_2$ is not on the line $\boldsymbol{L}_f$ and must be replaced with a new endpoint $\boldsymbol{e}'$. To obtain $\boldsymbol{e}'$, we must express the line $\boldsymbol{L}_f$ as the Plücker matrix $\boldsymbol{L}_p \in \mathbb{R}^{4\times 4}$.

Let the two planes π_{lc} and π_{rc} in Fig. 7 be $\boldsymbol{\pi}_{lc} = [\boldsymbol{n}_{lc}^T\, d_{lc}]$ and $\boldsymbol{\pi}_{rc} = [\boldsymbol{n}_{rc}^T\, d_{rc}]$, where $\boldsymbol{n}_{lc}, \boldsymbol{n}_{rc} \in \mathbb{R}^3$ and $d_{lc}, d_{rc} \in \mathbb{R}$. In general, a Plücker matrix $\boldsymbol{L}_p$ is obtained from a dual Plücker matrix $\boldsymbol{L}_p^*$ that can be computed from any two planes that are not parallel to each other, which is given by

$$\begin{aligned}\boldsymbol{L}_p^* &= \boldsymbol{\pi}_{lc}\boldsymbol{\pi}_{rc}^T - \boldsymbol{\pi}_{rc}\boldsymbol{\pi}_{lc}^T = \begin{bmatrix}\boldsymbol{n}_{lc}\\ d_{lc}\end{bmatrix}\begin{bmatrix}\boldsymbol{n}_{rc}^T & d_{rc}\end{bmatrix}\\ &\quad - \begin{bmatrix}\boldsymbol{n}_{rc}\\ d_{rc}\end{bmatrix}\begin{bmatrix}\boldsymbol{n}_{lc}^T & d_{lc}\end{bmatrix}\\ &= \begin{bmatrix}\boldsymbol{n}_{lc}\boldsymbol{n}_{rc}^T - \boldsymbol{n}_{rc}\boldsymbol{n}_{lc}^T & d_{rc}\boldsymbol{n}_{lc} - d_{lc}\boldsymbol{n}_{rc}\\ d_{lc}\boldsymbol{n}_{rc}^T - d_{rc}\boldsymbol{n}_{lc}^T & d_{lc}d_{rc} - d_{rc}d_{lc}\end{bmatrix} = \begin{bmatrix}\lfloor \boldsymbol{v}_p^* \rfloor_\times & \boldsymbol{n}_p^*\\ -\boldsymbol{n}_p^{*T} & 0\end{bmatrix}.\end{aligned} \tag{18}$$

According to the above equation, the dual Plücker matrix $\boldsymbol{L}_p^*$ contains the direction vector $\boldsymbol{v}_p^*$ and the normal vector $\boldsymbol{n}_p^*$ of the intersecting line of the two planes. It is common to find the Plücker matrix $\boldsymbol{L}_p$ using the calculated $\boldsymbol{L}_p^*$ but in this study, instead of obtaining the dual Plücker matrix $\boldsymbol{L}_p^*$, the vanishing directions are used to represent the Plücker matrix $\boldsymbol{L}_p$.

First, since the vanishing direction $\boldsymbol{\mu}_{vpi}$ represents the direction of the line feature, the vector $\boldsymbol{v}_p$ of the Plücker matrix $\boldsymbol{L}_p$ can be defined as $\boldsymbol{v}_p = \lambda\boldsymbol{\mu}_{vpi}$. The scalar λ is a compensation value to preserve the properties of the Plücker matrix. The length of $\boldsymbol{v}_p^*$ represented in Eq. (18) is not equal to 1. Therefore, in order to employ the unit vector $\boldsymbol{\mu}_{vpi}$ in the Plücker matrix $\boldsymbol{L}_p$, the vector $\boldsymbol{\mu}_{vpi}$ should be multiplied by λ to be equal to the length of $\boldsymbol{v}_p^*$. According to Eq. (18), since $\boldsymbol{v}_p^* = \boldsymbol{n}_{lc} \times \boldsymbol{n}_{rc}$, the scalar λ becomes the length of the vector $\boldsymbol{v}_p^*$ and is given by

$$\begin{aligned}\boldsymbol{v}_p &= \lambda\boldsymbol{\mu}_{vpi} = \left\|\boldsymbol{v}_p^*\right\|\boldsymbol{\mu}_{vpi} = \frac{\left\|\boldsymbol{n}_{lc}\times\boldsymbol{n}_{rc}\right\|}{\left\|\boldsymbol{n}_{lc}\right\|\left\|\boldsymbol{n}_{rc}\right\|}\boldsymbol{\mu}_{vpi}\\ &= \left\|\boldsymbol{n}_{lc}\times\boldsymbol{n}_{rc}\right\|\boldsymbol{\mu}_{vpi},\end{aligned} \tag{19}$$

where $\boldsymbol{n}_{lc}$ and $\boldsymbol{n}_{rc}$ represent the unit normal vectors of the planes π_{lc} and π_{rc}, respectively. Next, in the case of $\boldsymbol{n}_p$ of the Plücker matrix $\boldsymbol{L}_p$, it can be represented as $d_{rc}\boldsymbol{n}_{lc}$ - $d_{lc}\boldsymbol{n}_{rc}$ according to $\boldsymbol{n}_p^*$ in Eq. (18). The two planes π_{lc} and π_{rc} can be represented by the vector equations $\boldsymbol{n}^T{}_{lc}\boldsymbol{X} + d_{lc} = 0$ and $\boldsymbol{n}_{rc}^T\boldsymbol{X}\boldsymbol{X} + d_{rc} = 0$, respectively. According to Fig. 7, since the left camera frame $\{\boldsymbol{O}_{lc}\}$ is the reference frame, the term d_{lc} in the equation of the plane π_{lc} passing through the origin of $\{\boldsymbol{O}_{lc}\}$ becomes 0. Therefore, the $d_{lc}\boldsymbol{n}_{rc}$ term of $\boldsymbol{n}_p$ in Eq. (18) is eliminated establishing $\boldsymbol{n}_p = d_{rc}\boldsymbol{n}_{lc}$. Now, by substituting the equation $\boldsymbol{n}_{rc}^T\boldsymbol{X} + d_{rc} = 0$ into $\boldsymbol{n}_p^*$ of Eq. (18), $\boldsymbol{n}_p$ can be newly expressed as follows:

$$\begin{aligned}\boldsymbol{n}_p &= d_{rc}\boldsymbol{n}_{lc} - d_{lc}\boldsymbol{n}_{rc} = d_{rc}\boldsymbol{n}_{lc} = -\left(\boldsymbol{n}_{rc}^T\boldsymbol{X}\right)\boldsymbol{n}_{lc}\\ &= -\left(\boldsymbol{n}_{rc}^T\boldsymbol{e}_k\right)\boldsymbol{n}_{lc},\end{aligned} \tag{20}$$

where $\boldsymbol{e}_k$ is the endpoint obtained in Eq. (17). Since the only known $\boldsymbol{X}$ that satisfies the equation $\boldsymbol{n}_{rc}^T\boldsymbol{X} + d_{rc} = 0$ is the point $\boldsymbol{e}_k$, it can be substituted into $\boldsymbol{X}$ in the above equation. Using Eqs. (19) and (20), the Plücker matrix $\boldsymbol{L}_p$ of the line $\boldsymbol{L}_f$ is given by

$$\begin{aligned}\boldsymbol{L}_p &= \begin{bmatrix}\lfloor \boldsymbol{n}_p \rfloor_\times & \boldsymbol{v}_p\\ -\boldsymbol{v}_p^T & 0\end{bmatrix}\\ &= \begin{bmatrix}\left\lfloor -\left(\boldsymbol{n}_{rc}^T\boldsymbol{e}_k\right)\boldsymbol{n}_{lc}\right\rfloor_\times & \left\|\boldsymbol{n}_{lc}\times\boldsymbol{n}_{rc}\right\|\boldsymbol{\mu}_{vpi}\\ -\left\|\boldsymbol{n}_{lc}\times\boldsymbol{n}_{rc}\right\|\boldsymbol{\mu}_{vpi}^T & 0\end{bmatrix}.\end{aligned} \tag{21}$$

If the selected endpoint $\boldsymbol{e}_k$ is $\boldsymbol{e}_1$, the vectors $\boldsymbol{n}_{lc}$ and $\boldsymbol{n}_{rc}$ are calculated through the cross products $\boldsymbol{z}_{lc,1} \times \boldsymbol{z}_{lc,2}$ and $\boldsymbol{z}_{rc,1} \times \boldsymbol{z}_{rc,2}$, respectively. If the selected endpoint is $\boldsymbol{e}_2$, the property of the Plücker matrix can be maintained by reversing the arguments of each cross product. Applying the conventional Plücker line method in Eq. (18) to two images that do not have sufficient image parallax results in inaccurate 3D line initialization due to the pixel noises, but by adopting Eq. (21), more accurate line coordinates can be acquired by directly applying the vanishing directions into the Plücker matrix. The new endpoint $\boldsymbol{e}'$ is given by the Plücker matrix $\boldsymbol{L}_p$, as follows:

$$\begin{aligned}&\boldsymbol{e}' = \boldsymbol{L}_p\boldsymbol{\pi}_e = \boldsymbol{L}_p\begin{bmatrix}\boldsymbol{z}_{lc,j}\times\boldsymbol{n}_{lc}\\ 0\end{bmatrix},\\ &\text{where}\\ &\boldsymbol{z}_{lc,j} = \begin{cases}\boldsymbol{z}_{lc,1} & \boldsymbol{e}_k = \boldsymbol{e}_2\\ \boldsymbol{z}_{lc,2} & \boldsymbol{e}_k = \boldsymbol{e}_1\end{cases}.\end{aligned} \tag{22}$$

Here, $\boldsymbol{\pi}_e$ is a plane that is orthogonal to the plane $\boldsymbol{\pi}_{lc}$ and the vector $\boldsymbol{z}_{lc,j}$. Calculating $\boldsymbol{L}_p\boldsymbol{\pi}_e$ gives a vector of $\mathbb{R}^4$; thus, each element of the $\mathbb{R}^4$ vector should be divided by the fourth element of $\boldsymbol{L}_p\boldsymbol{\pi}_e$ to express $\boldsymbol{e}'$ as $\mathbb{R}^3$. The initialized point and line features are represented in the global coordinate system and stored in the Map Database shown in Fig. 2.

C. Observation Models

To update the Kalman filter, an observation model is needed to represent the initialized 3D points and lines in the current robot frame. The residuals between the feature coordinates obtained through the observation model and the feature coordinates actually observed in the current image are important information

for updating the filter. First, the residual of the point feature is given by

$$\boldsymbol{r}_{\text{point}} = \begin{bmatrix} \boldsymbol{z}_{lc,\text{point}} - \boldsymbol{h}_{lc}({}^{G}\boldsymbol{X}_f) \\ \boldsymbol{z}_{rc,\text{point}} - \boldsymbol{h}_{rc}({}^{G}\boldsymbol{X}_f) \end{bmatrix}, \tag{23}$$

where $\boldsymbol{z}_{lc,\text{point}}$ and $\boldsymbol{z}_{rc,\text{point}}$ represent the corner features observed in the left and right images, respectively, and the model $\boldsymbol{h}$ represents the observation model of the point feature given by

$$\begin{aligned} \begin{bmatrix} \boldsymbol{h}_{lc}({}^{G}\boldsymbol{X}_f) \\ \boldsymbol{h}_{rc}({}^{G}\boldsymbol{X}_f) \end{bmatrix} &= \begin{bmatrix} \boldsymbol{\Pi}({}^{R}\boldsymbol{X}_f) \\ \boldsymbol{\Pi}({}^{R}\boldsymbol{X}_f) \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{\Pi}(\text{Rot}(\boldsymbol{q}_{lc,i})({}^{G}\boldsymbol{X}_f - \boldsymbol{p}_{lc,i})) \\ \boldsymbol{\Pi}(\text{Rot}(\boldsymbol{q}_{rc,i})({}^{G}\boldsymbol{X}_f - \boldsymbol{p}_{rc,i})) \end{bmatrix}, \end{aligned} \tag{24}$$

where ${}^{G}\boldsymbol{X}_f$ and ${}^{R}\boldsymbol{X}_f$ are the coordinates representing the 3D point feature in the global frame and the robot frame, respectively. Here, Rot(.) is an operator that converts a quaternion into a SO(3) rotation matrix. The Jacobians for the observation model of Eq. (24) are given as follows:

$$\boldsymbol{H}_{\text{point},s} = \begin{bmatrix} \boldsymbol{H}_{lc,s} \\ \boldsymbol{H}_{rc,s} \end{bmatrix} = \begin{bmatrix} \frac{\partial \boldsymbol{h}_{lc}}{\partial {}^{R}\boldsymbol{X}_f} \cdot \frac{\partial {}^{R}\boldsymbol{X}_f}{\partial \boldsymbol{X}_{lc}} \\ \frac{\partial \boldsymbol{h}_{rc}}{\partial {}^{R}\boldsymbol{X}_f} \cdot \frac{\partial {}^{R}\boldsymbol{X}_f}{\partial \boldsymbol{X}_{lc}} \end{bmatrix} \text{ and} \tag{25}$$

$$\boldsymbol{H}_{\text{point},f} = \begin{bmatrix} \boldsymbol{H}_{lc,f} \\ \boldsymbol{H}_{rc,f} \end{bmatrix} = \begin{bmatrix} \frac{\partial \boldsymbol{h}_{lc}}{\partial {}^{R}\boldsymbol{X}_f} \cdot \frac{\partial {}^{R}\boldsymbol{X}_f}{\partial {}^{G}\boldsymbol{X}_f} \\ \frac{\partial \boldsymbol{h}_{rc}}{\partial {}^{R}\boldsymbol{X}_f} \cdot \frac{\partial {}^{R}\boldsymbol{X}_f}{\partial {}^{G}\boldsymbol{X}_f} \end{bmatrix}, \tag{26}$$

where $\boldsymbol{H}_s$ and $\boldsymbol{H}_f$ are the Jacobians obtained by partially differentiating the model $\boldsymbol{h}$ with respect to the camera state $\boldsymbol{X}_{lc}$ and the 3D point $\boldsymbol{X}_f$, respectively.

Let $\boldsymbol{m}_{lc}$ and $\boldsymbol{m}_{rc}$ be vectors that are orthogonal to the line segments $\boldsymbol{l}_{lc}$ and $\boldsymbol{l}_{rc}$ and are contained in the image plane. Here, $\boldsymbol{m}_{lc}$ and $\boldsymbol{m}_{rc}$ can be computed by applying Eq. (1) to $\boldsymbol{n}_{lc}$ and $\boldsymbol{n}_{rc}$, respectively. Then the residual of the line feature is given by

$$\boldsymbol{r}_{\text{line}} = \begin{bmatrix} (\boldsymbol{z}_{lc,\text{mid}} - \boldsymbol{h}_{lc}(\boldsymbol{e}_k)) \cdot \boldsymbol{m}_{lc} \\ (\boldsymbol{z}_{rc,\text{mid}} - \boldsymbol{h}_{rc}(\boldsymbol{e}_k)) \cdot \boldsymbol{m}_{rc} \\ (\boldsymbol{z}_{lc,\text{mid}} - \boldsymbol{h}_{lc}(\boldsymbol{e}')) \cdot \boldsymbol{m}_{lc} \\ (\boldsymbol{z}_{rc,\text{mid}} - \boldsymbol{h}_{rc}(\boldsymbol{e}')) \cdot \boldsymbol{m}_{rc} \end{bmatrix}, \tag{27}$$

where $\boldsymbol{h}_{lc}$ and $\boldsymbol{h}_{rc}$ represent the point feature observation models introduced in Eq. (24), and $\boldsymbol{z}_{lc,\text{mid}}$ and $\boldsymbol{z}_{rc,\text{mid}}$ represent the center points of the line segments $\boldsymbol{l}_{lc}$ and $\boldsymbol{l}_{rc}$, respectively. Accordingly, the Jacobian of the line feature is expressed as

$$\begin{aligned} &\boldsymbol{H}_{\text{line},s} \\ &= \begin{bmatrix} \boldsymbol{H}_{lc,s,1}^{T}\boldsymbol{m}_{lc} & \boldsymbol{H}_{rc,s,1}^{T}\boldsymbol{m}_{rc} & \boldsymbol{H}_{lc,s,2}^{T}\boldsymbol{m}_{lc} & \boldsymbol{H}_{rc,s,2}^{T}\boldsymbol{m}_{rc} \end{bmatrix}^{T}, \end{aligned} \tag{28}$$

$$\begin{aligned} &\boldsymbol{H}_{\text{line},f} \\ &= \begin{bmatrix} \boldsymbol{H}_{lc,f,1}^{T}\boldsymbol{m}_{lc} & \boldsymbol{H}_{rc,f,1}^{T}\boldsymbol{m}_{rc} & \boldsymbol{H}_{lc,f,2}^{T}\boldsymbol{m}_{lc} & \boldsymbol{H}_{rc,f,2}^{T}\boldsymbol{m}_{rc} \end{bmatrix}^{T}, \end{aligned} \tag{29}$$

where the Jacobians $\boldsymbol{H}_{lc,s}$, $\boldsymbol{H}_{lc,f}$, $\boldsymbol{H}_{rc,s}$, and $\boldsymbol{H}_{rc,f}$ can be obtained through the same procedure as Eqs. (25) and (26), and the subscripts 1 and 2 represent the Jacobians related to the endpoints $\boldsymbol{e}_k$ and $\boldsymbol{e}'$, respectively. The remainder of the Kalman filter updating process is omitted in this paper (refer to [2]).

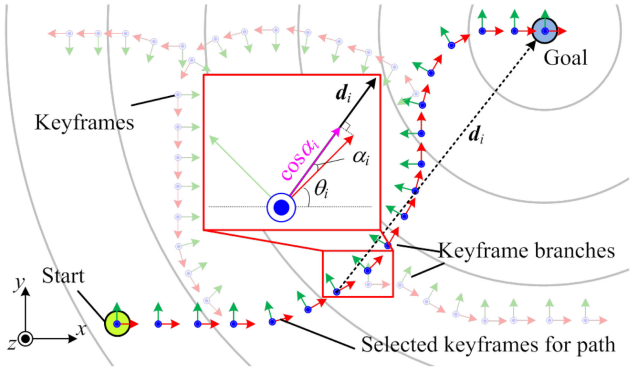


Fig. 8. Selection of the keyframes that are relevant to the desired path.

IV. Keyframe-Based Path and Localization

In the autonomous navigation of a mobile robot, the path is essential information for the robot to successfully reach the target points. Most path planners require the occupancy grid map because they generate paths by calculating cost functions according to the structure of the environment. However, to generate an accurate grid map, high-quality range data should be acquired from the laser scanners. The KAVIN system, which does not employ a laser scanner, adopts a keyframe-based topological map for path generation.

A. Global Path Generation

The keyframes are the paths that a robot traveled when generating a map. In the absence of the grid map, these keyframes indicate the reachable areas of the environment. The path is generated by selecting only the keyframes that are likely to be passed through when the robot moves from the starting point to the goal point. Each keyframe is assigned an ID, according to the order in which it was created. A path can be generated by selecting the keyframes whose IDs increase sequentially, starting from the keyframe closest to the current robot pose. When a robot follows the keyframes, the keyframes may be divided into several branches, as shown in Fig. 8. Branches are naturally created when the robot repeatedly revisits places to perform a loop-closure during mapping. The user should generate keyframes by manually controlling a robot to visit all the reachable spaces within the environment at least once. During this process, the robot should also visit the potential destinations to perform its tasks and store the goal points in the form of keyframe. In the case of Fig. 8, among the first keyframes of each branch, a branch to which the keyframe that points closest to the goal belongs is selected. Let the vector $\boldsymbol{d}_i$ be a vector pointing to the goal point at the position of the i-th keyframe. Then, the first keyframe of each branch is assigned a score given by

$$\boldsymbol{\eta}_{\text{branch}} = \cos\alpha_i, \tag{30}$$

where α_i represents the angle difference between the vector $\boldsymbol{d}_i$ and the heading vector $\boldsymbol{h}_i$ of the i-th keyframe. When Eq. (30) is applied to the first keyframe of each branch, the branch to which the keyframe with the highest $\boldsymbol{\eta}_{\text{branch}}$ belongs is selected as the

path. Once the branch is selected, the path is created in sequence according to the keyframe IDs until the robot encounters the next branch. This procedure is repeated until the robot reaches the target. On this path, the robot selects the desired pose required for the motion control.

B. Localization

The KAVIN system provides the robot's current pose for mapping and autonomous driving through VIO, but the drift error of the pose can increase gradually. The keyframe-based path introduced above not only provides the desired pose required to control the wheel actuators of a robot but also allows the accumulated pose error of a robot to be corrected continuously. In the autonomous navigation of a robot, the pose-to-pose constraint calculated for loop-closure is used to determine the relative coordinates between a keyframe and a robot. If the keyframe of the place most similar to the current camera image is selected through place recognition, the relative coordinates between the selected keyframe and the current robot can be computed using the P*n*P algorithm. The current pose of a robot is corrected through this relative coordinate.

V. Experiments and Discussion

First, the pose-estimation performance of VIO processed by KAVIN was evaluated using a public dataset [18] and a stereo visual-inertial sensor mounted on a mobile robot. Then, to evaluate the autonomous driving performance of a mobile robot equipped with KAVIN, a robot was sent to various points in a real environment via autonomous driving. All experiments in this study were conducted using a main i5-8259U CPU equipped with 4 GB of RAM.

A. Evaluation on Pose Estimation

1) MVSEC Outdoor Car Datasets: The objective of this study was to design a robust visual-inertial navigation system for a wheeled mobile robot. Thus, the Multi Vehicle Stereo Event Camera (MVSEC) dataset [18], which was generated by collecting the sensor data from a car, was suitable for this study. The MVSEC dataset provided stereo images with a resolution of 752×480 and IMU data at cycles of 20 and 200 Hz, and the two sensors are synchronized with each other. No other sensor data from the dataset were used. To evaluate the VIO performance of KAVIN, VINS-Fusion [19] and S-MSCKF [2], which are two representative VIO open-source algorithms adopting stereo cameras, were executed with the same dataset to compare the pose error. For a fair comparison of the VIO, the loop-closure function of KAVIN was disabled. Fig. 9 shows the alignment of the estimated trajectories for each algorithm used in the experiment. The global positioning system (GPS) provided in the MVSEC dataset was used as the ground truth. As shown in Fig. 9, the root-mean-square (RMS) error of the trajectory estimated by KAVIN was 3.97 and 4.21 m in the x and y directions, respectively. VINS-Fusion, which was developed in a follow-up study to VINS-Mono [3], exhibited RMS errors of 4.9 and 3.64 m in the x and y directions, respectively, which is

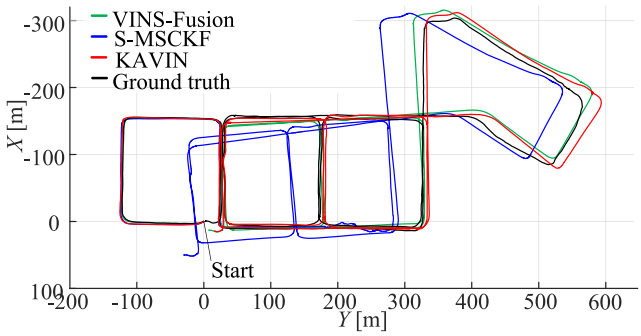


Fig. 9. Experimental comparison of two representative state-of-the-art open-source algorithms and KAVIN for the MVSEC dataset.

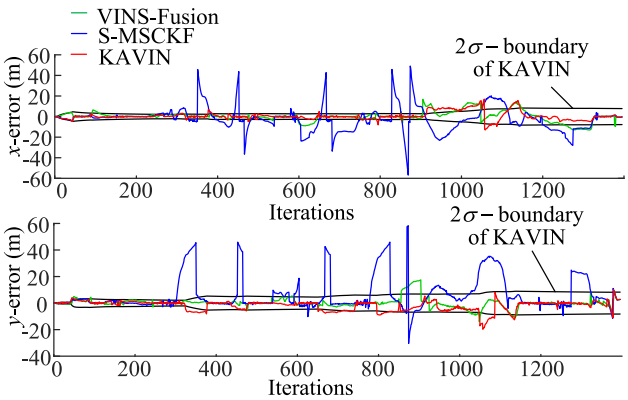


Fig. 10. Quantitative analysis on the accumulated pose-error in the MVSEC dataset.

similar to that of KAVIN. S-MSCKF exhibited a relatively large drift error compared with the other algorithms. The MVSEC dataset contained sensor data collected from a car traveling approximately $\geq$3 km. Considering this, the VIO of KAVIN as well as VINS-Fusion was able to estimate the pose of the vehicle robustly without loop-closure. Since the experiments were conducted based on the performance of pure VIO without loop-closure, errors of all algorithms are accumulated naturally. To discuss this more quantitatively, the cumulative errors of each algorithm are plotted in Fig. 10. Fig. 10 presents the quantitative analysis results for the pose error estimated by each algorithm relative to the ground truth. In Fig. 10, the boundary of the 2σ-confidence level, based on the standard deviation of the accumulated error of KAVIN, is plotted on the graph. In Fig. 10, the cumulative errors of KAVIN and VINS-Fusion were relatively small even after a long-distance trip. Therefore, KAVIN could perform vehicle pose estimation more robustly based on the line features than S-MSCKF that is a filter-based scheme similar to KAVIN. VINS-Fusion is an optimization-based algorithm that showed excellent performance, and the results were similar to those of KAVIN.

In addition to the accuracy of the pose, the computation times of the three algorithms were compared. The computation time of KAVIN was measured for Feature Node and Thread 1, which should operate in real-time. In addition, we measured

TABLE I
COMPARISON OF COMPUTATION SPEEDS FOR MVSEC DATASET

Algorithm	Function	Average computation speed (sec)	Number of poses per second (Hz)
S-MSCKF	Image process	0.016	19.99
	Pose estimation	0.05	
VINS-Fusion	Image process	0.047	9.98
	Pose estimation	0.1	
KAVIN	Image process	0.066	10.5
	Pose estimation	0.088	

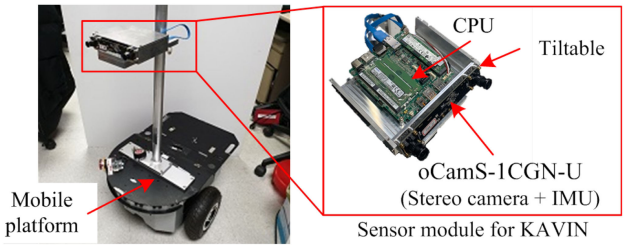


Fig. 11. Experimental setup for the real environment test using a sensor module installed on the mobile platform.

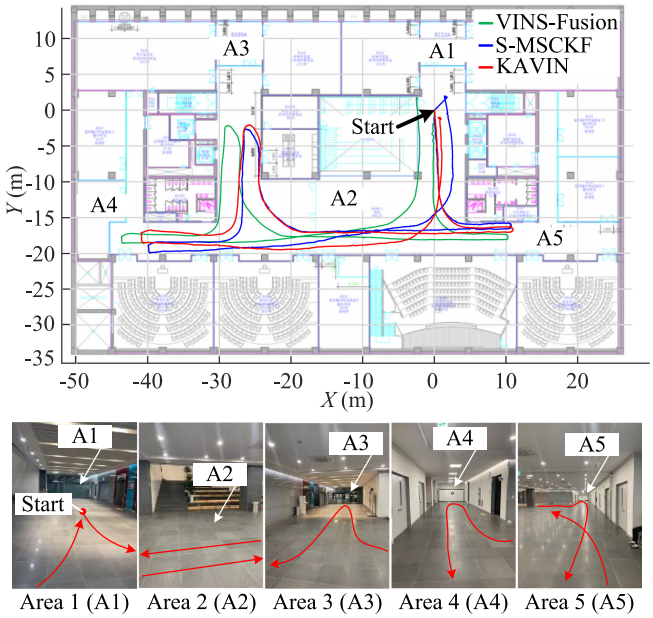


Fig. 12. Experimental comparison of two representative state-of-the-art open-source algorithms and KAVIN in the real environment. The five images at the bottom are the actual scenes of the experimental environment, and the red path shows the path that a robot traveled.

the number of poses per second in Hz to check real-time performance. As indicated by Table I, the computation time of S-MSCKF was the fastest. S-MSCKF was developed for a MAV moving at a high speed; thus, it had the fastest calculation speed, although its accuracy was low. VINS-Fusion and KAVIN exhibited computation speeds of approximately 0.1 sec and 0.088 sec, respectively. VINS-Fusion showed the slowest pose estimation speed since the optimization-based scheme requires high computational burden. However, the computation speed of KAVIN's image processing (0.066 sec) showed the slowest speed due to the line extraction. Nevertheless, KAVIN provided the robot's pose at 10.5 Hz, showing that real-time operation is possible.

2) Real Environment: To evaluate the VIO performance of the KAVIN system in a real environment, a self-contained module with KAVIN installed, as shown in Fig. 11, was mounted on MRP-NRLAB 02, which is a two-wheel differential mobile robot. This sensor module was equipped with an oCamS-1CGN-U stereo visual-inertial sensor that provided VGA stereo images and IMU data at 30 and 100 Hz, respectively. In this experiment, the loop-closure function of KAVIN was deactivated for a fair comparison of the VIO. The experiment was conducted in the lobby of Korea University, which contained a wide-open space, as shown in Fig. 12, where visual features could be extracted from distant walls. Fig. 12 shows the trajectories of the robot estimated by the algorithms in comparison with those for KAVIN. Because the point features were not extracted in some areas, S-MSCKF and VINS-Fusion exhibited homing errors of approximately 2.5 and 2.98 m, respectively, owing to the drift error. In comparison, KAVIN exhibited a smaller homing error of 1.38 m. As shown in Fig. 12, the environment in which this experiment was conducted was precisely compared with the blueprint and displayed in the figure. The pose of the robot estimated by VINS-Fusion deviated from the environment,

TABLE II
COMPARISON OF COMPUTATION SPEEDS IN REAL ENVIRONMENT

Algorithm	Function	Average computation speed (sec)	Number of poses per second (Hz)
S-MSCKF	Image process	0.01	30.00
	Pose estimation	0.033	
VINS-Fusion	Image process	0.029	15.01
	Pose estimation	0.067	
KAVIN	Image process	0.045	19.57
	Pose estimation	0.049	

and S-MSCKF exhibited unstable performance because of the insufficient features when the robot returned to the starting point. The computation times of the algorithms in this experiment were compared, as shown in Table II. Because the MVSEC dataset and oCamS-1CGN-U had different speeds of sensor acquisition, the computation speeds of the three algorithms were higher than those in the previous experiment. S-MSCKF was the fastest, but its accuracy was low. KAVIN and VINS-Fusion exhibited accurate pose estimation, but the computation speed of KAVIN (0.049 sec) was faster than that of VINS-Fusion (0.067 sec) which adopts the optimization-based scheme. However, the computation speed of KAVIN's image processing (0.045 sec) was the slowest due to line extraction. Nevertheless, KAVIN provided the robot's pose at 19.57 Hz, showing that real-time operation is possible.

B. Mapping Experiments of KAVIN in Real Environments

Based on KAVIN, the mapping experiments were conducted in various environments, as shown in Fig. 13. By activating the loop-closure function of KAVIN, we visualized all the 3D point and line features stored in each keyframe, and then verified

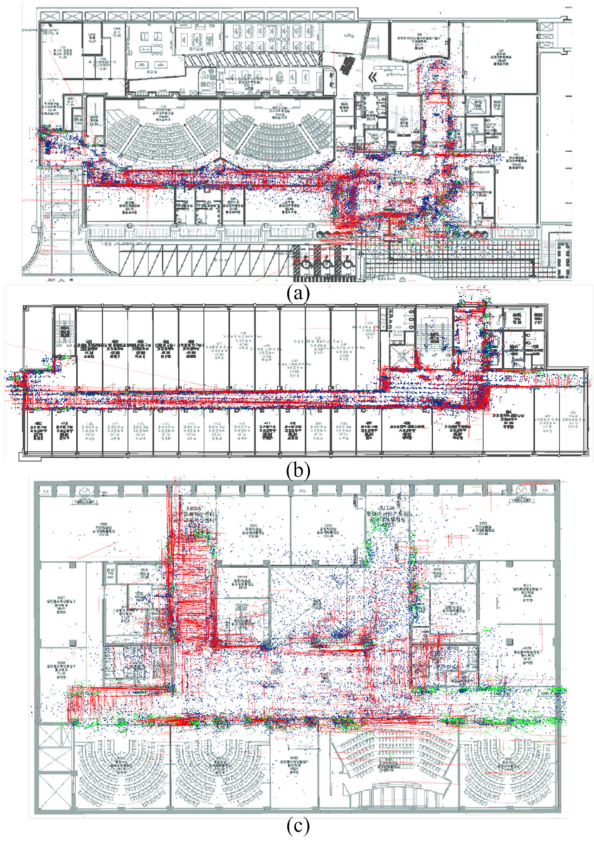


Fig. 13. The mapping results of KAVIN (with loop-closure) in (a) the texture-rich lobby, (b) the office-like environment with the long corridor, and (c) the plaza with wide-open areas. The results of the mapping in each environment are being compared with the blueprints. The keyframe map consists of the points (dark blue), vertical lines (green), and horizontal lines (red).

whether they match the blueprints of the real environments. In Fig. 13(a), where there are many glasses and reflectors, some outliers of the features are seen, but the overall point and line features are consistent with the blueprint. Fig. 13(b) shows the result of mapping in the long corridor. In Fig. 13(b), loop-closure is rarely performed in the hallway. This is because the directions that the front camera faces are opposite when the robot moves to the end of the hallway and returns from the end. Nevertheless, the 3D features are well aligned with the blueprint. Fig. 13(c) is a large open space area, which is a challenging environment. Even in this environment, keyframe features correspond well to the blueprint.

C. Autonomous Navigation Performance Without Encoders

1) Autonomous Navigation Scenario: The purpose of this study was to develop a vision-based navigation system for autonomous mobile robots. Therefore, in this experiment, when the mobile robot was autonomously sent to the target points in the environment, the pose error of the point where the robot arrived was measured, and the performance was evaluated. Before the experiment, we used KAVIN's loop-closure to generate a topological map of the environment. Fig. 14 shows the environment at Korea University where the experiment was conducted which is the same environment as in Fig. 13(b). Since the maps of 3D

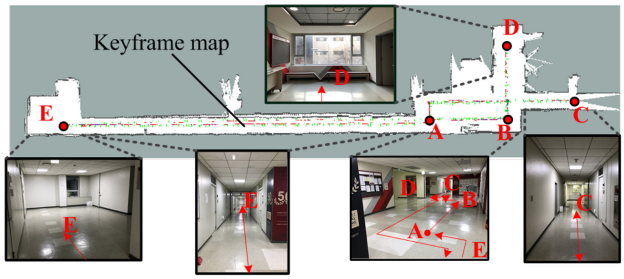


Fig. 14. Test scenario for autonomous navigation and the keyframe map (the red, green, and blue markings on the grid map) with visualized occupancy grid map of the test environment.

TABLE III
RESULTS OF AUTONOMOUS NAVIGATION OF A MOBILE ROBOT USING KAVIN

Point	Errors when reaching the target points (m)					
	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Trial 6
A	0.09	0.08	0.09	0.09	0.04	0.14
B	0.04	0.11	0.15	0.21	0.10	0.09
C	0.05	0.06	0.18	0.07	0.22	0.06
D	0.09	0.03	0.05	0.27	0.20	0.25
E	0.09	0.09	0.06	0.02	0.20	0.13

features were crowded as shown in Fig. 13, from this experiment, the laser scan data in each keyframe was stored to visualize the environment as a grid map. The laser scanner was used only for the environmental visualization and the verification of the keyframe map accuracy, but not for the overall KAVIN system including autonomous navigation. The robot was commanded to visit five target points A, B, C, D, and E on the topological map shown in Fig. 14. When the robot reached each target point, the pose error between the actual location where the robot arrived and the target point (i.e., ground truth) was measured, and the repeatability of KAVIN was examined by repeating this procedure six times for each point. Table III presents the error between each target point and the location where the robot actually has reached.

2) Kidnap Recovery Scenario: Kidnap situations occur frequently in service-robot navigation. In such cases, the robot should find its current pose on the map through global localization. This experiment was performed to determine whether a mobile robot could overcome a kidnap situation using the KAVIN system. Fig. 15 shows the results of global localization performed in the same environment as the previous experiment. When the robot started to move from the starting location A, as shown in Fig. 15(a), the robot was kidnapped and moved to an arbitrary location on the map while the sensor was covered, as shown in Fig. 15(b). Consequently, the robot found a keyframe with a similar place on the map and correctly recovered its lost pose. As shown in Fig. 15(c), the laser scanner installed on the robot indicated that KAVIN found the correct pose.

VI. CONCLUSION AND FUTURE WORK

This study demonstrates a VIO-based autonomous navigation system (i.e. KAVIN) that works stably by initializing the line features based on the vanishing points, even when a camera is mounted on a non-holonomic wheeled platform that generates

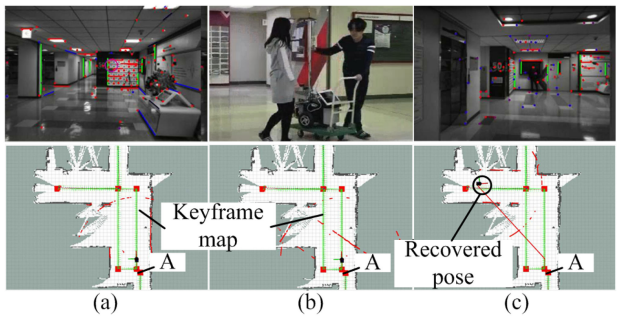


Fig. 15. Global localization test in the kidnap recovery scenario. (a) When the robot starts to move from the origin, (b) two persons cover the sensor, kidnap a robot, and place it to an arbitrary location. (c) After placing a robot to an unknown location, a robot uses a keyframe map to restore its current location.

relatively limited motion compare to a drone. In a series of experiments, KAVIN's VIO exhibited robust pose estimation performance when applied to a wheeled mobile robot. Additionally, a mobile robot equipped with KAVIN visited five target points six times each via autonomous driving, with an average pose error of 11 cm. As the pose error was not accumulated even when the robot repeatedly visited the same target point, the high repeatability of KAVIN was verified. Therefore, KAVIN ensures robust pose estimation of a mobile robot without wheel encoders and can prevent the accumulation of drift error of VIO by using the given keyframes. Because the KAVIN system does not depend on wheel encoders, its performance is not affected by the type of mobile robot; thus, the KAVIN system is suitable for various applications.

As with many visual-inertial navigation systems, the performance of KAVIN highly depends on the number of features that can be extracted from the camera image. Since the vehicles move in contact with the ground, they are exposed to various external forces that can easily excite the IMU. If both point features and vanishing points are poorly extracted, it is difficult to prevent the VIO from diverging due to the external forces. Therefore, various features and observation models should be proposed to ensure continuous feature initialization. In a subsequent study, various geometric constraints will be applied to the low-level visual features (e.g., points, lines, and planes) of the image to enable robust localization in various environments with extreme changes even when a camera is mounted on a non-holonomic wheeled platform.

REFERENCES

[1] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual-inertial odometry using nonlinear optimization," *Int. J. Robot. Res.*, vol. 34, no. 3, pp. 314–334, Jun. 2015.

[2] K. Sun *et al.*, "Robust stereo visual inertial odometry for fast autonomous flight," *IEEE Robot. Autom. Lett.*, vol. 3, no. 2, pp. 965–972, Apr. 2018.

[3] T. Qin, P. Li, Z. Yang, and S. Shen, "VINS-Mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 1004–1020, Aug. 2018.

[4] X. Qiu, H. Zhang, W. Fu, C. Zhao, and Y. Jin, "Monocular visual-inertial odometry with an unbiased linear system model and robust feature tracking front-end," *Sensors (Basel)*, vol. 8, no. 19, Apr. 2019, Art. no. 1941.

[5] S. Y. Hwang and J. B. Song, "Monocular vision-based SLAM in indoor environment using corner, lamp, and door features from upward-looking camera," *IEEE Trans. Ind. Electron.*, vol. 58, no. 10, pp. 4804–4812, Oct. 2011.

[6] H. Zhou, D. Zou, L. Pei, R. Ying, P. Liu, and W. Yu, "StructSLAM: Visual SLAM with building structure lines," *IEEE Trans. Veh. Technol.*, vol. 64, no. 4, pp. 1364–1375, Apr. 2010.

[7] F. Camposeco and M. Pollefeys, "Using vanishing points to improve visual inertial odometry," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2015, pp. 5219–5225.

[8] J. Shi and C. Tomasi, "Good features to track," in *Proc. IEEE Conf. Comput. Vision Pattern Recogn.*, Jun. 1994, pp. 593–600.

[9] A. M. Reza, "Realization of the contrast limited adaptive histogram equalization (CLAHE) for real-time image enhancement," *J. VLSI Signal Process.*, vol. 38, no. 1, pp. 35–44, Aug. 2004.

[10] R. Grompone von Gioi, J. Jakubowicz, J. Morel, and G. Randall, "LSD: A line segement detector," *Image Process. Line*, vol. 2012, no. 2, pp. 35–55, Mar. 2012.

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

[12] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Proc. IEEE Int. Conf. Computer Vision*, Nov. 2011, pp. 2564–2571.

[13] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge, U.K.: Cambridge University Press, 2000.

[14] M. Kaess, H. Johannsson, R. Roberts, V. Ila, and F. Dellart, "ISAM2: Incremental smoothing and mapping using the Bayes tree," *Int. J. Robot. Res.*, vol. 32, no. 2, pp. 216–235, Dec. 2012.

[15] J. Borenstein and Y. Koren, "The vector field histogram—Fast obstacle avoidance for mobile robots," *IEEE Trans. Robot. Autom.*, vol. 7, no. 3, pp. 278–288, Jun. 1991.

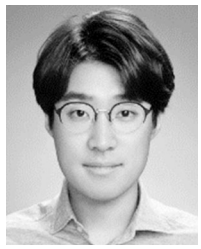
[16] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint kalman filter for vision-aided inertial navigation," in *Proc. IEEE Intl. Conf. Robot. Autom.*, Apr. 2007, pp. 3565–3572.

[17] F. Zheng, G. Tsai, Z. Zhang, S. Liu, C. C. Chu, and H. Hu, "Trifo-VIO: Robust and efficient stereo visual inertial odometry using points and lines," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2018, pp. 3686–3693.

[18] A. Z. Zhu, D. Thakur, T. Ozaslan, B. Pfrommer, V. Kumar, and K. Daniilidis, "The multi vehicle stereo event camera dataset: An event camera dataset for 3D perception," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 2032–2039, Feb. 2018.

[19] T. Qin, J. Pan, S. Cao, and S. Shen, "A general optimization-based framework for local odometry estimation with multiple sensors," Jan. 2019. [Online]. Available: https://arxiv.org/abs/1901.03638v1

Hee-Won Chae received the B.S. degree in mechanical engineering, in 2013 from Korea University, Seoul, Korea, where he is currently working toward the Ph.D. degree in the School of Mechanical Engineering. His research interests include robot navigation, computer vision, machine learning, and visual SLAM.



Ji-Hoon Choi received the B.S. degree in mechatronics engineering from Kangwon National University, Gangwon-do, Korea, in 2017, and the M.S. degree from the Department of Mechatronics, Korea University, Seoul, Korea, in 2020. His research interests include robot navigation, computer vision, mobile manipulator, and software engineering.

Jae-Bok Song (Senior Member, IEEE) received the B.S. and M.S. degrees in mechanical engineering from Seoul National University, Seoul, Korea, in 1983 and 1985, respectively, and the Ph.D. degree in mechanical engineering from MIT, Cambridge, MA, in 1992. He joined the Faculty of the Department of Mechanical Engineering, Korea University, Seoul, Korea, in 1993. His current research interests are the design and control of robot arms and robot navigation systems.