Real-Time Visual–Inertial SLAM Based on Adaptive Keyframe Selection for Mobile AR Applications

Jin-Chun Piao and Shin-Dug Kim, *Member, IEEE*

***Abstract*—Simultaneous localization and mapping (SLAM) technology is used in many applications, such as augmented reality (AR)/virtual reality, robots, drones, and self-driving vehicles. In AR applications, rapid camera motion estimation, actual size, and scale are important issues. In this research, we introduce a real-time visual–inertial SLAM based on an adaptive keyframe selection for mobile AR applications. Specifically, the SLAM system is designed based on the adaptive keyframe selection visual–inertial odometry method that includes the adaptive keyframe selection method and the lightweight visual–inertial odometry method. The inertial measurement unit data are used to predict the motion state of the current frame and it is judged whether or not the current frame is a keyframe by an adaptive selection method based on learning and automatic setting. Relatively unimportant frames (not a keyframe) are processed using a lightweight visual–inertial odometry method for efficiency and real-time performance. We simulate it in a PC environment and compare it with state-of-the-art methods. The experimental results demonstrate that the mean translation root-mean-square error of the keyframe trajectory is 0.067 m without the ground-truth scale matching, and the scale error is 0.58% with the EuRoC dataset. Moreover, the experimental results of the mobile device show that the performance is improved by 34.5%–53.8% using the proposed method.**

***Index Terms*—Adaptive keyframe selection, augmented reality, mobile applications, optical-flow-based tracking, simultaneous localization and mapping, visual–inertial odometry.**

I. Introduction

SLAM is a low-level technology to provide location and map information for applications that use it [1]. AR and VR are the prominent forms of immersive computing. AR is a technology that links the real world to the virtual world. In other words, AR is a technology between reality technology and VR technology. AR adds computer-generated information and objects to your everyday world, renders a digital object on a camera image such that the resulting image resembles an actual scene, estimates the motion (position and orientation) of the camera relative to an object, and envisions the placement effect of furniture before purchasing. Moreover, with AR, you can observe digital objects at their actual sizes and scales, in your world. Both VR and AR enable us to experience computing in a manner similar to experiencing the real world. Over the past year, the emergence of Apple's ARKit [2] and Google's ARCore [3] has made AR a reality in mobile devices.

Generally, the computing power and power consumption of mobile devices are limited to use SLAM system in mobile devices. However, high robustness and computational efficiency are still required in mobile SLAM environment. In AR/VR applications, the real-time camera motion and the actual distance between the camera and the target object are more important, and the accuracy of SLAM mapping and global locating is relatively low. Thus, there is a trade-off between performance and accuracy; hence, in mobile AR applications, in order to obtain real-time camera motion, the SLAM system must be optimized at the expense of losing partial accuracy.

In this paper, we propose a real-time visual–inertial SLAM based on adaptive keyframe selection for mobile AR applications. It incorporates the adaptive keyframe selection method and the lightweight visual–inertial odometry method. The IMU data is used to predict the motion state of the current frame at the pre-processing stage. Then, it is judged whether or not the current frame is a keyframe by learning and automatic setting method. Relatively unimportant frames are processed using lightweight methods for efficiency and real-time performance. Experimental results show that the mean translation RMSE of keyframe trajectory is 0.067 m without the ground-truth scale matching, and the scale error can be obtained as 0.58% with the EuRoC dataset.

The rest of the paper is organized as follows: In Section II, we introduce and explain the related work on monocular SLAM. In Section III, we propose a real-time visual–inertial SLAM based on adaptive keyframe selection for mobile AR applications, and present an implementation method based on Android platform in Section IV. In Section V, we try a variety of cases to demonstrate performance improvements. Finally, the conclusion and future work are described in Section VI.

II. Related Work

With the development of SLAM technology, we have begun to apply the technology in a wider range of fields. For example 3D object recognition and modeling, 3D reconstruction and deep learning [4]–[7].

Manuscript received August 6, 2018; revised December 14, 2018 and March 8, 2019; accepted April 10, 2019. Date of publication April 25, 2019; date of current version October 24, 2019. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Mea Wang. *(Corresponding author: Shin-Dug Kim.)*

J.-C. Piao is with the Department of Computer Science and Technology, Yanbian University, Yanji 133002, China (e-mail: jcpiao@ybu.edu.cn).

S.-D. Kim is with the Department of Computer Science, Yonsei University, Seoul 03722, South Korea (e-mail: sdkim@yonsei.ac.kr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2019.2913324

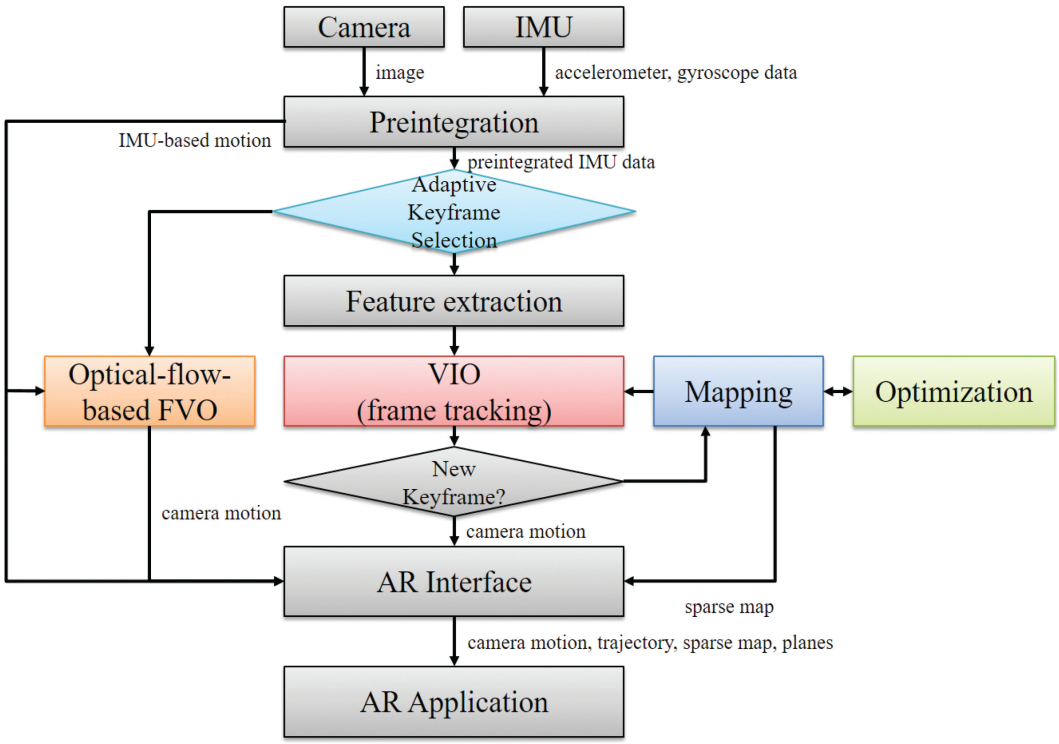

Fig. 1. Structural architecture of adaptive keyframe selection visual–inertial odometry based SLAM.

Monocular visual SLAM is low-cost and convenient to implement. Since the monocular camera cannot obtain depth information, the monocular SLAM shows the following problems: requirement of initialization, scale uncertainty and drift. In our previous study, we solved this problem by fusing image data with IMU sensor data by using a visual–inertial odometry (VIO) based SLAM [8]. By using various advanced hardware sensors, SLAM technology is evolving into multi-sensor fusion, using multiple complementary sensors to support higher accuracy and robustness. Mobile devices are generally equipped with a low-cost commercial camera and a microelectromechanical system (MEMS) IMU. Therefore, it is important to design an accurate, robust, and efficient algorithm by combining these sensors. The camera device can capture detail-rich scenes, and the IMU has a high frame rate and can accurately estimate the position in a relatively short period of time. By combining these two sensors, complementary results can be obtained.

Especially, the techniques to integrate visual information and IMU data can be classified into loosely coupled [9], [10] and tightly coupled [11], [12] techniques. This can be done by considering whether the visual information is added to the state vector. The loosely coupled methods typically are based on the execution of the vision-based SLAM and inertia-based modules separately. Also the combination of the results is used to estimate the measurements. However, in the loosely coupled method, the monocular SLAM drift problem is still present. In a tightly coupled method, the IMU deviation can be corrected by adding visual information to the state vector, and the depth information of the monocular SLAM can be estimated. This method is more accurate and robustness than the conventional single-vision-sensor-based SLAM. However, this method increases the computation and fusion of IMU data, resulting in higher computational load. Therefore, in order to be executed on a mobile device in real time, certain optimizations and improvements are required.

This paper focuses on optimize the monocular visual–inertial SLAM in mobile applications. The deficient computing power of mobile devices is a limitation on SLAM application in mobile environments. In addition, the accuracy and robustness of SLAM are reduced owing to the low accuracy of the sensors used in mobile devices. There must be a trade-off between accuracy and performance. In mobile AR applications, it is more important to obtain high-precision real-time camera motion.

III. Methodologies

This section describes the fundamental structure of the real-time visual–inertial SLAM based on adaptive keyframe selection for mobile AR applications and its operational implementation of the adaptive method in the SLAM system. Moreover, the implementation method based on the Android platform is presented in next section. These methods are suitable for commonly used feature-based visual–inertial SLAM.

The structural architecture of the adaptive keyframe selection visual–inertial odometry (AVIO) based SLAM is shown as in Figure 1. As in Figure 1, main architecture is constructed as several modules such as frame tracking, mapping, and optimization, consistent with the general visual–inertial SLAM. It also includes an adaptive keyframe selection module and an optical-flow-based fast visual odometry (FVO) module. The camera and IMU data are used as the input data, and the IMU data is integrated through the preintegration method in the preprocessing

step. The IMU-based motion is transmitted to the optical-flow-based FVO and the AR Interface modules, and are used for the camera and IMU data fusion. In Adaptive Keyframe Selection, the preintegrated IMU data is used to predict the motion state of the current frame and predict whether or not the current frame is a keyframe.

In most of typical feature-based SLAM methods, feature extraction and tracking are performed for all frames. In the feature extraction, the feature points are extracted and the description is calculated for each frame. This operation is a time-consuming process. Therefore, we propose an optical-flow-based FVO to rapidly calculate the relative motion state between the current frame and the previous frame without feature extraction. This method is much more efficient than the feature-extraction-based tracking. The optical-flow-based FVO is used for frames with relatively low significance and operations such as mapping of SLAM and creation of keyframes are omitted. After determining the keyframe according to certain conditions, the mapping module draws the map using the selected keyframe. In addition, the optimization module reduces the error through loop closing and graph optimization. Finally, the AR Interface provides information such as camara motion, trajectory, sparse map, and planes to the AR Application.

First, this paper introduces an adaptive keyframe selection method based on preintegrated IMU data. In general, not all frames are key frames that exert a significant impact on the SLAM system. Omitting a few low-importance frames to reduce the computation time and complexity does not affect the results. Feature-based SLAM requires feature extraction; however, it requires large amounts of computation and time. In the preprocessing step, the position change between the current frame and the reference frame can be estimated through the IMU preintegration method. In this manner, the importance of the current frame can be determined, and the less important frame is processed by the lightweight method described below.

Secondly, we present an optical-flow-based FVO module, which quickly estimates the camera motion between the current frame and previous frame. In the optical-flow-based FVO, the valid tracking keypoints of the previous frame are used for tracking without having to extract feature points from the input frame. Thus, no additional feature extraction time is required, and eventually the execution time of the tracking can be reduced. It is also a lightweight method for estimating camera motion because it uses an optical-flow-based feature point tracking method. We also propose a method to improve the performance of the optical-flow-based feature point tracking method using the value of the preintegrated IMU data. Specifically, the amount of calculation and the resource overhead can be reduced by simplifying the algorithm basically.

A. Adaptive Keyframe Selection

In the adaptive keyframe selection method, the value of the preintegrated IMU data is used to estimate the state of motion between the current frame and the reference frame. The reference frame takes a last valid tracking frame from the frame tracking (VIO) module. When the tracking is weak or a period of time has elapsed since the VIO module last inserted a new keyframe (the number of interval frames exceeds the threshold), the reference frame is retrieved. Thus, the keyframe can adaptively be selected from the original visual odometry or lightweight odometry for current-frame tracking according to the magnitude of the motion state.

The design goal for this module is to reduce the tracking time and also reduce computing resource requirement. A feature-based SLAM performs feature extraction and tracking for each input image notwithstanding whether the camera is not moving. That is, it needs to perform unnecessary calculations and depletes computing power. This is a critical issue and should be resolved in environments with limited power and computing resources, such as mobile devices.

1) IMU Preintegration: Typically, mobile devices are equipped with a camera and an IMU sensor. In a visual–inertial SLAM, the IMU sensors and camera are connected rigidly in general. The conversion mechanism is required between the IMU and the camera coordinate systems and it can be obtained by the multi-sensor system calibration [13], [14].

The IMU preintegration [15] method is used to solve the computational complexity problem of the visual–inertial SLAM based on optimization method, where the integration is repeated when the bias estimate changed. The constraints between the two frames can be represented by only IMU data; moreover, they can be expressed according to the motion states of the two frames. Thus, we can define the residual between the observation and state. And then, we can construct a least-squares solution and optimize the motion.

If the value of the preintegrated IMU data is small, it shows that the camera movement changes marginally and the parallax between the two images is also small. Therefore, the new input image can be considered as relatively unimportant in the SLAM module. When the value of the preintegrated IMU data – Δ_{ij} between the current frame *j* and the previous frame *i* is calculated and also it turns out to be less than the threshold, the new input frame (current frame) *j* is marked as unimportant and the camera motion using our lightweight optical-flow-based FVO module is subsequently estimated. This frame is not processed by the main SLAM module. Thus, we can estimate the state of motion between the current frame and the reference frame, and select the keyframe through learning and automatic setting methods.

2) Adaptive Selection Method Based on Learning and Automatic Setting: The adaptive selection module uses IMU preintegration measurements to compare the current frame with the reference frame each time a frame arrives, to determine whether the current frame is an important frame. The adaptive selection module automatically updates the learning model by continuously learning whether the value of the preintegrated IMU data and the result of the input frame is keyframe or not.

In the IMU preintegration, we have to estimate the changes in velocity, position, and rotation between the current frame and the reference frame, i.e., $\Delta\mathbf{v}$, $\Delta\mathbf{p}$, and $\Delta\mathbf{R}$, respectively. If the value of the IMU measurement is smaller, the motion change becomes smaller. That is, the parallax between the current frame and the reference frame is small. In order to straightforwardly determine the magnitude of the change in the IMU measurement

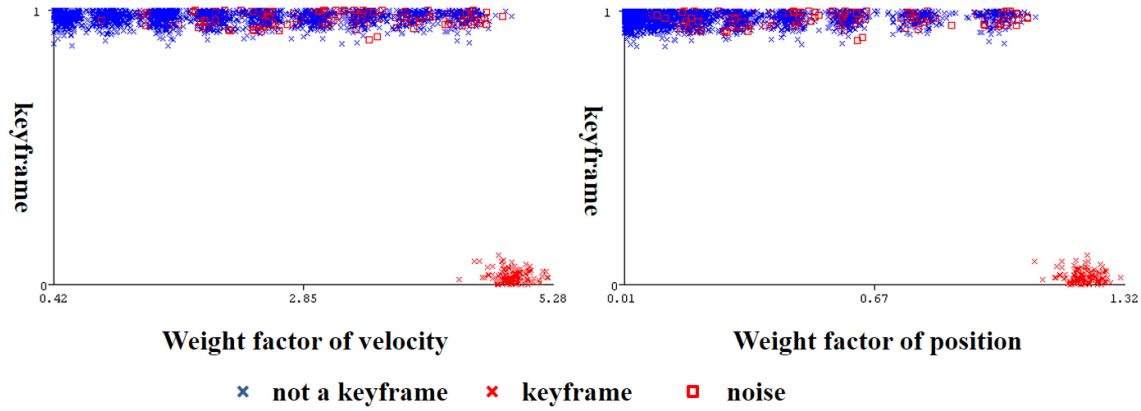

Fig. 2. Performance of proposed adaptive keyframe selection learning model.

value, we calculated the velocity and position as the norm, as shown in the following equation, and the rotation angle $\boldsymbol{\theta}$ using the Rodrigues' rotation formula.

In order to compare the magnitude of the velocity and position of preintegrated IMU data, the norm of the velocity and position can be described as follows:

$$\text{Norm of velocity: } |\boldsymbol{V}| = \sqrt{V_x^2 + V_y^2 + V_z^2}, \tag{1}$$

$$\text{Norm of position: } |\boldsymbol{P}| = \sqrt{P_x^2 + P_y^2 + P_z^2}, \tag{2}$$

where $\mathbf{V} = (V_x, V_y, V_z)$ is the velocity and $\mathbf{P} = (P_x, P_y, P_z)$ is the position, which are decomposed into the x-, y-, z-coordinate system variation.

The Rodrigues' rotation formula is expressed in a more compact form as follows:

$$\boldsymbol{\theta} = \sqrt{a^2 + b^2 + c^2}, \boldsymbol{v} = \left[\frac{a}{\theta}, \frac{b}{\theta}, \frac{c}{\theta}\right]. \tag{3}$$

where the rotation transformation has three degrees of freedom; it is compactly represented as a three-dimensional (3D) vector $[a, b, c]$. $\boldsymbol{\theta}$ and $\boldsymbol{v}$ represent the rotation angle and unit vector, respectively. We use the rotation angle $\boldsymbol{\theta}$ to compare the magnitude of rotation.

As the motion of the velocity and position values in the x-, y-, and z-coordinate axes influences the result differently, we assign different weights to the values of the x-, y-, and z-coordinate axes. That is, the movement in the x- and y-coordinate axes indicates the left-and-right and up-and-down movement of the screen, and the zoom-in and zoom-out of the screen of the movement in the z-coordinate axis. $P_{velocity}$, $P_{position}$, and $P_{rotation}$ represent the weight factors of velocity, position, and rotation, respectively.

There are several methods of learning; however, here we use the proposed formula to calculate the weight factor and compare this value with the threshold value to determine the importance of the frame.

$$\text{The weight factor: } \boldsymbol{V} = aP_{velocity} + bP_{position} + cP_{rotation}. \tag{4}$$

Here, $P_{velocity}$, $P_{position}$, and $P_{rotation}$ represent the weight factors of velocity, position, and rotation, respectively, and parameters a, b, and c are designated through learning.

Each time a new frame is input, the existing weight factor model parameters are updated using the following formula:

$$\begin{aligned} a &= (1-k)a + ka_t, \\ b &= (1-k)b + kb_t, \\ c &= (1-k)c + kc_t. \end{aligned} \tag{5}$$

where a, b, and c are the current parameters and a_t, b_t, and c_t are the parameters of the newly input frame. Here, k is a weight having a value between zero and one. The smaller the value of k is, the higher is the weight of the existing parameter; moreover, the larger the value of k is, the higher is the weight of the new parameter. k starts at an initial value of one, decreases each time learning is performed, and has a minimum value of 0.1.

The most accurate result can be obtained by performing the learning each time the frame is input. However, in this case, the procedure for performing the learning is performed too frequently, which is likely to adversely affect real-time accessibility and the power consumption. Therefore, if the learning is stabilized for a certain period of time, the learning is stopped, and if the result of the learning model continues to be erroneous, the learning procedure is resumed.

Figure 2 shows the performance of our proposed learning model. The importance of the frame is determined by the weight factor of the velocity and position. The precision and F-measure of the proposed learning model are 0.915 and 0.956, respectively, and we can obtain higher accuracy by adjusting the threshold according to the situation. For example, if you are testing accuracy on the EuRoC benchmark [16], set the threshold to a lower value to improve the classification results for less important frames. Furthermore, low-importance frames operate as lightweight modules.

The adaptive selection module automatically sets the adaptive execution strategy in real-time based on the learning method and the results of feedback (such as performance, learning accuracy, and number of frames between last keyframe).

3) Adaptive Execution Strategies: For different scenarios, different levels of adaptive execution strategies are designed as

TABLE I
TABLE OF ADAPTIVE EXECUTION STRATEGIES

Mode	Modules	Descriptions
Level 0	VIO	Accuracy, no adaptive method
Level 1	VIO and optical-flow-based FVO	Balance, applies to all scenes
Level 2	VIO and optical-flow-based FVO	Performance, applies to easy and medium scenes
Level 3	VIO and optical-flow-based FVO	Performance, applies to only easy scenes

follows. First, we analyzed the performance of our proposed SLAM system via experiments on the EuRoC dataset. A straightforward way to reduce the average tracking time is to reduce the number of feature points used in each image. Concurrently, the translation RMSE [17] value of the keyframe trajectory increases and the number of lost tracking frames increases. There is a trade-off relation between the number of features used in the SLAM system and its accuracy and robustness.

Analysis results obtained using the EuRoC dataset show that when the number of features is less than 500, the accuracy and stability are significantly reduced. To ensure the efficiency of the calculation, we use 500 features in the adaptive execution strategy approach. Subsequently, as shown in Table I, four modes were designed for additional execution strategies. The adaptive execution strategies can be automatically selected according to the performance of the mobile device in the system initialization phase, or the user may preset the mode depending on the circumstances.

When the computational resources are sufficient, accuracy and stability are prioritized; therefore, Level 0 uses only VIO modules. According to the change of the IMU sensor value, Levels 1–3 are divided into different strategies, and the VIO module and the optical-flow-based FVO module are adaptively selected and used. Level 1 supports all scenes with high accuracy. Level 2 is similar to Level 1; however, it has performance priority and supports easy-level and medium-level scenes. Level 3 is the fastest performance-oriented setting and supports easy-level scenes. Moreover, the proposed method can measure the change value of the IMU sensor for a certain period of time and compare it with the threshold value so that it can be executed as a strategy that automatically changes in runtime.

B. Lightweight Visual–Inertial Odometry

In the feature-based SLAM method, the feature extracted for each frame. This operation is very time-consuming. For instance, the ORB feature extraction operation in ORB-SLAM [18] requires at least about 20 ms per frame to run on a computer, and this time exceeds 100 ms on a mobile device. Moreover, SLAM performs tasks such as keypoints matching, motion estimation, and local map update for each frame.

Hence, we propose a lightweight visual–inertial odometry based on optical-flow-based FVO module to rapidly estimate the relative motion between the current frame and the previous frame. By combining the IMU, the motion change between the two frames is estimated in the IMU preintegration step. The FVO module is used for frames with relatively low importance and processes such as local map update and keyframe insertion are skipped. This method effectively reduces the computational complexity of SLAM, making it suitable for the rapid estimation of location information, such as in AR applications. Since the FVO module is performed when the value of the preintegrated IMU data is marginal, it shows a higher effect when the camera moves relatively slowly.

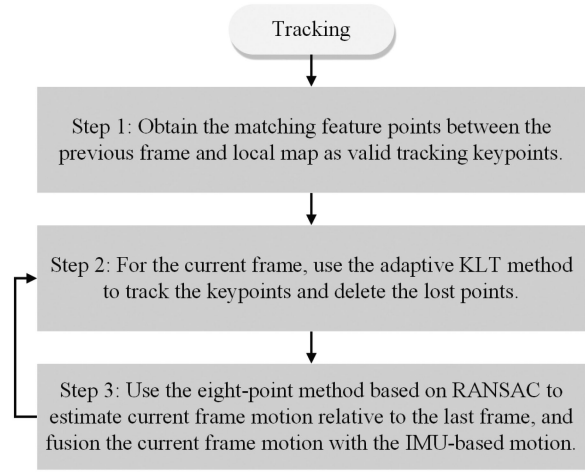

Fig. 3. Flowchart of optical-flow-based FVO.

Figure 3 shows a flow chart of the optical-flow-based FVO, which can be classified into the following three steps:

In the first step, when the optical-flow-based FVO is used in the tracking module, the matching feature points between the previous frame and the local map are acquired. These points are used as valid tracking keypoints, and the previous frame is set as the reference frame.

In the second step, a tracking process based on an adaptive KLT (Kanade–Lucas–Tomasi) feature tracking method is executed on the current frame. KLT [19], [20] is a typical sparse-optical-flow feature tracking method. However, if the image resolution is high or the camera moves rapidly, the motion of the points in the window may be different; this will result in a larger calculation error. Therefore, we propose the adaptive KLT feature tracking method to reduce the computational error and improve tracking accuracy and robustness.

In the third step, the eight-point [21] method is used to estimate the relative motion between the two frames, each time using the valid matched keypoints. If the subsequent input frame continues to execute the optical-flow-based FVO module, then only steps 2 and 3 are performed.

1) Adaptive KLT Feature Tracking Method: Figure 4 illustrates the adaptive KLT feature tracking method. We designed an adaptive Gaussian pyramid KLT method with IMU data. The Gaussian pyramid KLT method involves the use of Gaussian blur and subsamples to detect images of different resolutions; starting with the image with the lowest resolution at the top, gradually increasing the resolution to the original image at the bottom. To consider the impact of real-time problems, we use a

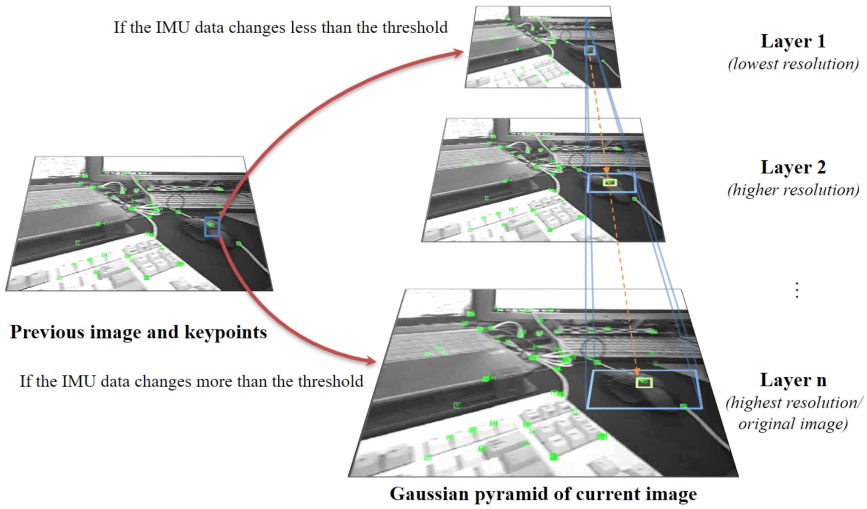

Fig. 4. Diagram of adaptive KLT feature tracking method.

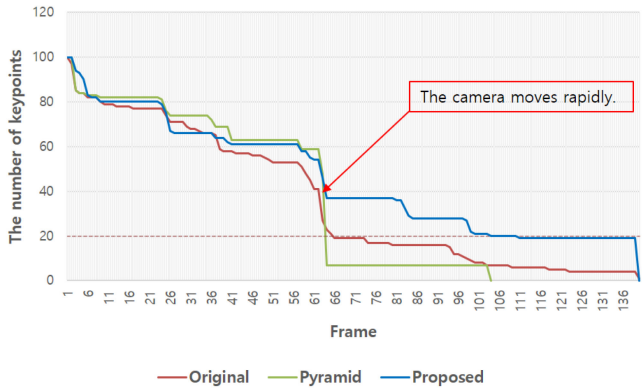

Fig. 5. Comparison of performance of proposed method and original KLT method.

three-layer Gaussian pyramid and adaptive search method based on IMU data. If the IMU data changes less than the threshold, then the KLT method is used to estimate the motion of the pixel from the top layer; moreover, after the scale conversion, it is used as the initial value of the subsequent layer. Higher accuracy is obtained at the subsequent layer; and finally, the motion of the pixel is estimated in the original image. If the IMU data changes more than the threshold, search is performed directly in the original image.

Figure 5 shows the performance of the proposed adaptive KLT feature tracking method in comparison with the original method and the pyramid method. It shows the number of feature points tracked in the same image frame sequence with 100 feature points at the start. We can verify that the proposed method is more effective at tracking feature points, particularly when the camera moves rapidly. As a result, the tracking performance was improved by 26.9% and 30.8% compared to the original and pyramid methods, respectively.

The optical flow method is very fast and efficient when using a small number of tracking keypoints. As in Figure 3, we use the keypoints obtained in step 1 to track the current frame, thereafter delete any keypoints in the current frame that cannot be tracked. The keypoints that have been successfully tracked are stored with the current frame, and the current frame is updated to the new reference frame. If the subsequent input frame continues to perform the optical-flow-based FVO module, step 2 is performed immediately without performing step 1.

If the number of tracked feature points is smaller than a certain value, tracking is considered to fail because it does not provide adequate matching points to estimate the camera motion. In this case, the current frame drops out the optical-flow-based FVO process and attempts to trace itself back to the VIO process.

When the grayscale vary with the ambient environment and camera exposure parameters change, the result of the optical flow method is less than ideal. Since the proposed method uses only a few consecutive frames for tracking at a time, it is not significantly affected by this problem.

2) Motion Estimation: In the second step, the displacement of the target object on the 2D image can be obtained; however, in order to obtain the 3D motion, we estimate the essential matrix $\mathbf{E}$ through the matching points of the two frames. We estimate the relative motion between two frames by the epipolar geometry [22]. First, the feature points of the two images are matched, and then an eight-point method based on random sample consensus (RANSAC) is used to estimate the essential matrix of the two images. Eventually, the camera rotation and translation matrix $\mathbf{T} = [\boldsymbol{R}|\boldsymbol{t}]$ is recovered from the estimated essential matrix $\mathbf{E}$ by using singular value decomposition (SVD) [23].

We obtain the current frame motion and IMU-based motion. Then, we use the Ethzasl multi-sensor fusion (MSF) framework [10] to integrate the camera and IMU data. Finally, we obtain the output of the camera motion.

IV. Implementation on Android Platform

In this section, we propose an AR application solution using SLAM in an Android environment. The structural architecture of the Android SLAM-based AR application is shown in Figure 6. As shown in Figure 6, it is mainly divided into the

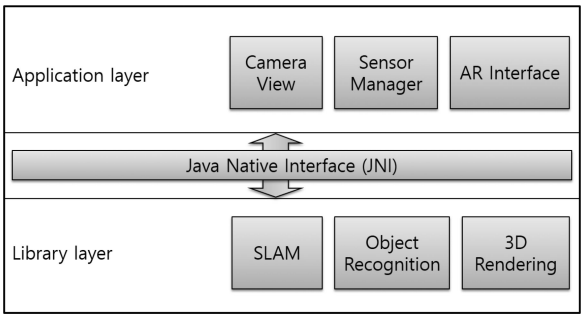

Fig. 6. Structural architecture of Android SLAM-based AR application.

application layer and library layer. In the application layer, the Camera View instantiates the Android Camera2 class in real-time from the front-end and displays the camera image on the user's screen. In the library layer, the SLAM module and the object recognition module provide the necessary information for the AR application in the back-end.

The application layer is implemented in Java and consists of the Camera View, Sensor Manager, and AR Interface. The Camera View is used to display the camera sensor image of the mobile device, the Sensor Manager permits you to access the device's sensors, and the AR Interface converts the information such as the camera motion and sparse map obtained from the SLAM module to the information required by the AR application. The library layer includes the SLAM module, object recognition module, and 3D rendering module. The SLAM module and the object recognition module are implemented in C++ using Android Native Development Kit (NDK), and the 3D rendering module is implemented in OpenGL ES language. Between the application layer and the library layer is the Java Native Interface (JNI), which acts as a link between the Java code and C++ code and calls a function. In the JNI layer, because the camera image is stored in the memory buffer and the address value is transmitted, the time required to transfer the image between the application layer and the library layer can be reduced.

We use the previously proposed AVIO-based SLAM to dynamically select relatively important frames and perform odometry and mapping.

V. Experiments and Results

First, we implemented and evaluated the proposed system in a PC environment. We compared the keyframe trajectory of the SLAM system with the ground-truth data and calculated the translation RMSE [24], [25] of the keyframe trajectory for each sequence. Finally, we implemented the SLAM-based AR application in mobile systems and tested on an Android smartphone.

Table II enumerates the specifications of the PC and mobile devices environment we utilized. We experimented in the PC environment and conducted the experiments using the EuRoC dataset [16], and tested on an Android smartphone. The EuRoC dataset has a total of 11 sequences and is recorded using a micro aerial vehicle. Sequences are measured in a machine hall and two vicon rooms and classified into easy, medium, and difficult levels according to the illumination, texture, and motion speed. The

TABLE II
Experiment Environments

PC specification	
CPU	Intel Core i7-6700K (4.0 GHz × 4)
RAM	SDRAM 16 GB (8 GB × 2)
OS	Ubuntu 14.04 (64-bit)
Mobile specification	
Product name	ASUS ZenFone AR (ZS571KL)
Android version	7.0 Nougat
CPU	Snapdragon 821 (2.35 GHz × 4)
Camera	SONY IMX 318 (22.5 Megapixel)
IMU	InvenSense ICM20602 (˜200 Hz)

TABLE III
Comparison of Average Execution Time Between Optical-Flow-Based FVO Module and VIO Module

Optical-flow-based FVO	**Average time (ms)**	**VIO**	**Average time (ms)**
Obtain valid keypoints	0.16	ORB feature extraction	20.85
Adaptive KLT tracking	2.53	Initial motion estimation with IMU	5.09
Motion estimation	6.13	TrackLocalMap with IMU	8.02
Total	**8.82**		**33.96**

EuRoC dataset provides stereo images, MEMS IMU, ground-truth data, etc. The imaging system uses a global shutter camera that supports a size of 752 × 480 and an image data frequency of 20 FPS, and the IMU data supports a data frequency of 200 Hz. The ground-truth data is used for comparison when analyzing the accuracy of the SLAM estimated trajectory. Therefore, the EuRoC dataset is widely used for benchmarking SLAM systems.

A. Simulation in PC Environment

1) AVIO-Based SLAM Evaluation: First, the average execution time required for tracking by the optical-flow-based FVO module and that by the VIO module was measured and compared in the AVIO-based SLAM. Table III shows the average execution time of the two odometry modules of AVIO-based SLAM for the MH_01_easy sequence when the Level 1 adaptive execution strategy is applied. The experimental results reveal that the optical-flow-based FVO module is 3.85 times faster than the VIO module. That is, the higher the proportion of the optical-flow-based FVO among the total tracking process, the faster the average tracking time.

Figure 7 shows the average tracking time and translation RMSE of the keyframe trajectory with ground-truth scale matching of all the sequences for the EuRoC dataset when the features of four adaptive execution strategies are applied in the AVIO-based SLAM. The unit of average tracking time is measured in milliseconds, and the unit of mean translation RMSE is meters. If the RMSE value is marginal, the accuracy of SLAM system is high. If the number of lost tracking frames is smaller than the limited threshold (a certain percentage of the total number of

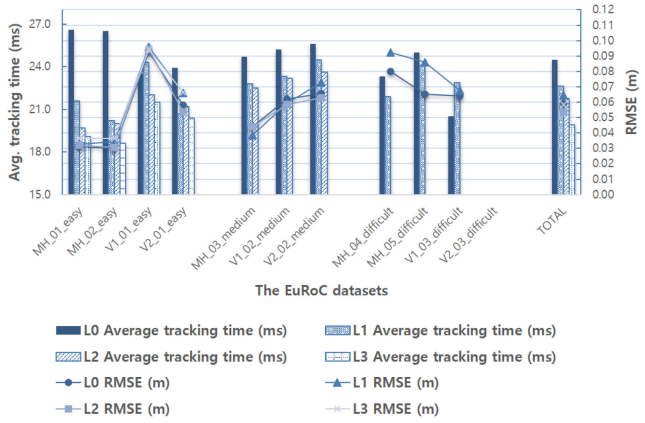

Fig. 7. Comparison of average tracking time and translation RMSE of keyframe trajectory with different level-sets of AVIO-based SLAM.

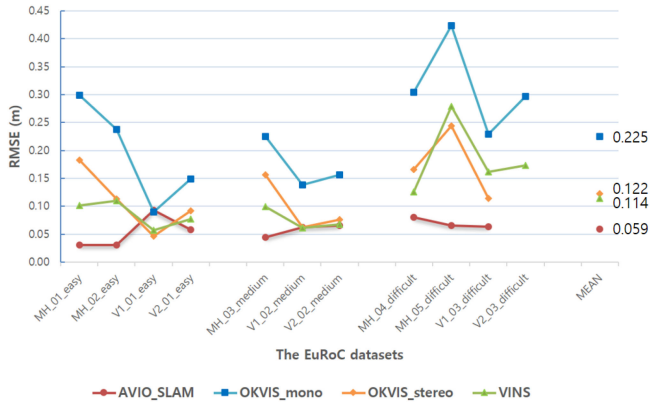

Fig. 8. Evaluation of AVIO-based SLAM and comparison with the OKVIS [26] and the VINS [12] under the same conditions.

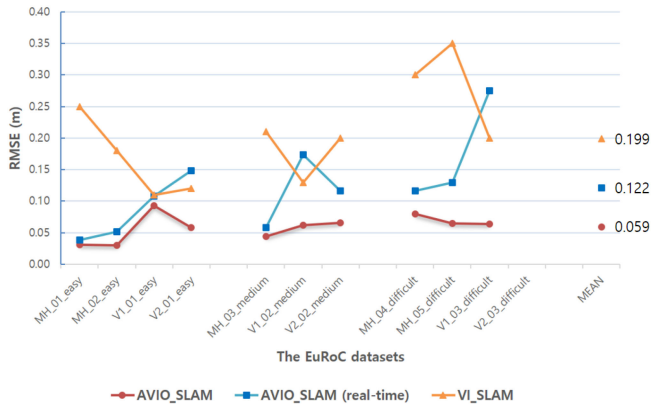

Fig. 9. Comparison of accuracy of AVIO-based SLAM and VI_SLAM [27].

frames), the robustness is high. If this value is higher than the threshold, it indicates that the tracking has failed.

Figure 7 shows the results of Level 0-3 of the adaptive execution strategies. Level 1 successfully passed the test of all datasets. Level 2 passed the test of easy dataset and medium dataset, and the tracking failed in the difficult dataset test. Level 3 only passed the test of easy dataset. We observe that using the adaptive selection method can be significantly reduce the average tracking time. For the Level 1 strategy, the average tracking time is reduced by 7.8% compared to that of Level 0, whereas the RMSE value increased by 8.5%. For the Level 2 strategy with the easy dataset and medium dataset, the average tracking times are reduced by 17.5% and 8.3%, respectively, compared to those of Level 0. For the Level 3 strategy with the easy dataset, the average tracking time is reduced by 18.8% compared to that of Level 0.

2) Comparison of Performance of AVIO-Based SLAM With State-of-the-Art Work: We compared the proposed AVIO-based SLAM with OKVIS [26] and VINS [12], which are the state-of-the-art VIO-based SLAMs that operate with monocular camera. Our proposed method is different from the above SLAMs in terms of a number of implementation details, such as adaptive keyframe selection method, lightweight visual–inertial odometry method, and the combination of heavy and lightweight odometry modules. The IMU data is used to predict the motion state of the current frame and it is determined whether or not the current frame is a keyframe. Relatively insignificant frames are processed using lightweight visual–inertial odometry for efficiency and real-time performance. More details can be found in the Methodology section. In particular, the learning-based adaptive keyframe selection method has advantages that can be applied to various feature-based SLAM systems.

Figure 8 illustrates the comparison of the accuracy of the proposed AVIO-based SLAM, the OKVIS, and the VINS under identical conditions. When testing on the EuRoC dataset, use the mean translation RMSE of the keyframe trajectory with ground-truth scale matching to compare the accuracy of each SLAM system. The mean translation RMSE values of the AVIO-based SLAM, OKVIS_mono, OKVIS_stereo, and VINS are 0.059 m, 0.225 m, 0.122 m, and 0.114 m, respectively. The scale errors of the AVIO-based SLAM, OKVIS_mono, OKVIS_stereo, and VINS are 0.58%, 1.63%, 0.95%, and 1.04%, respectively. Overall, the AVIO-based SLAM outperforms the other SLAMs in accuracy.

Figure 9 shows the comparison of the accuracy of the proposed AVIO-based SLAM and that of the VI_SLAM [27]. In this study, because we did not obtain the source code for VI_SLAM, we compared using the results of execution of the EuRoC dataset. The average RMSE values of the AVIO-based SLAM, the AVIO-based SLAM (real-time), and the VI_SLAM are 0.059 m, 0.122 m, and 0.199 m, respectively. Therefore, the accuracy of our proposed the AVIO-based SLAM is higher than that of the VI_SLAM.

3) Experiments of AVIO-Based SLAM With Actual Mobile Device Sensor Data: First, we use the OpenCV library to resize and convert the acquired camera image to RGB format on an Android smartphone. The image data and IMU data are transferred to the PC in real time under a WiFi connection using the ROS toolkit. We obtained a 30 fps image of size 640 $\times$ 480 pixels and 200 Hz IMU data on the ASUS ZenFone AR smartphone. After the camera–IMU calibration, the proposed method was tested in a PC environment. Figure 10 shows a screenshot of the proposed

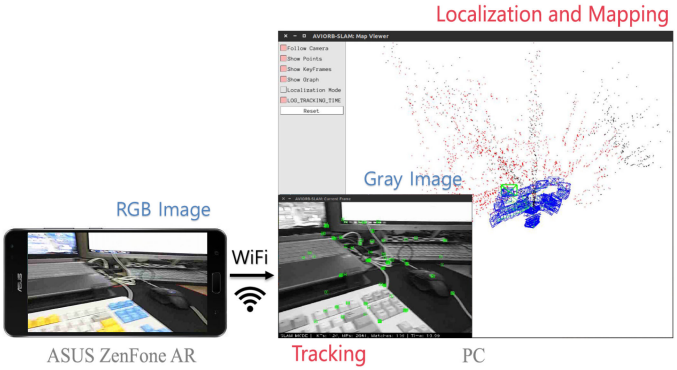

Fig. 10. Screenshot of proposed AVIO-based SLAM with actual mobile device sensor data in PC environment.

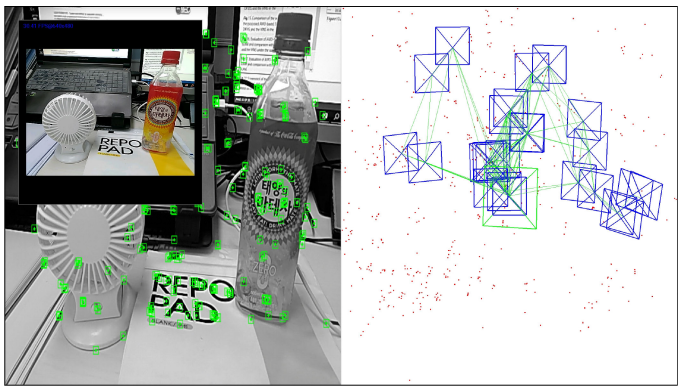

Fig. 11. Screenshot of Android SLAM-based AR application in mobile device.

AVIO-based SLAM with actual mobile device sensor data in a PC environment.

When we tested the proposed system using actual mobile device sensor data, the average tracking time was 21.5 ms. However, because the smartphone used in the experiment utilizes a low-priced rolling shutter camera and the image frequency is relatively low, moving it marginally faster will cause the image to be blurred and tracking to be lost. Although it can be solved to a certain extent with the VIO method, this system still has not achieved the desired robustness.

B. Implementation on Android Platform

The primary SLAM-based AR application for implementation in an Android mobile environment was developed based on the SLAM system and AR interface. The screenshot of the Android SLAM-based AR application implemented in a mobile device is shown as in Figure 11.

In order to reduce the computational complexity, we use a camera image stream with 640 × 480 pixels at 30 fps and IMU data at 200 Hz on the ASUS ZenFone AR smartphone. Moreover, we use derived 3D sensors to directly acquire the values of gravity, linear acceleration, and rotation vector. In mobile devices, the physical center position and coordinate axes of the IMU sensor and those of the camera sensor are different.

TABLE IV
COMPARISON OF PERFORMANCE BETWEEN ORB-SLAM AND AVIO-BASED SLAM IN ANDROID MOBILE DEVICE

Number of features	**ORB-SLAM** (FPS)	**AVIO-based SLAM** (FPS)	**Speed up**
500	5.8	7.8	1.345
1000	3.9	6.0	1.538

Therefore, it is necessary to align the center of the camera sensor and that of the IMU sensor and synchronize the timestamp.

We implemented ORB-SLAM and AVIO-based SLAM modules in an Android device and compared their performance. The frame per second (FPS) value of the SLAM-based AR application in the Android environment is evaluated. The results of the experiment are shown in Table IV. In the SLAM module, the FPS values of ORB-SLAM and AVIO-based SLAM are 5.8 and 7.8, respectively, when 500 feature numbers are applied. When 1000 features are applied, the FPS values of ORB-SLAM and AVIO-based SLAM are 3.9 and 6.0, respectively. Therefore, AVIO-based SLAM can improve performance by 34.5% and 53.8%, respectively, compared to ORB-SLAM. The accuracy is marginally reduced, and the difference is not evident. The loop closing function mainly determines if the camera has reached the previously captured scene. Loop closing solves the problem of drifting of the estimated positions over time and reduces the error. When this function is turned off, the FPS can reach 10.1.

VI. CONCLUSION AND FUTURE WORK

In this paper, we presented an AVIO-based SLAM for mobile AR applications. We designed an AVIO-based SLAM that includes the adaptive keyframe selection and the lightweight visual–inertial odometry methods. In order to verify the performance of the proposed method, we simulated it in a PC environment, compared it with other SLAM technologies, and ported it to the Android platform to measure actual performance.

The experimental results reveal that the mean translation RMSE of keyframe trajectory with the EuRoC dataset is approximately 0.067 m. Using different level set adaptive strategies, the average tracking time of the SLAM system was reduced by 7.8%, 12.9% and 18.8%. The visual–inertial SLAM based on adaptive keyframe selection is relatively stable and robust, and the scale error of the EuRoC dataset is typically less than 1.5%. Consequently, our proposed method is suitable for improving the efficiency and real-time performance of SLAM with mobile devices. The AVIO-based SLAM outperforms the state-of-the-art VIO-based SLAMs that operate with monocular camera in accuracy.

In addition, we conducted experiments using a mobile device and the results reveal a significant performance improvement with the use of the proposed method. The proposed learning-based adaptive keyframe selection method in this paper has advantages that can be applied to various feature-based SLAM systems. This is also a direct way to improve the efficiency of SLAM system on mobile platforms with limited resources such as smartphone, robot and drone.

Currently, our system runs only on the CPUs and does not use parallelization or hardware acceleration methods. In future work, we will consider using a GPU-based parallelized computational acceleration method and a more lightweight structure based on optimization.

REFERENCES

[1] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: Part I," *IEEE Robot. Autom. Mag.*, vol. 13, no. 2, pp. 99–110, Jun. 2006.

[2] Apple. ARKit, 2017. [Online]. Available: https://developer.apple.com/arkit

[3] Google. ARCore, 2017. [Online]. Available: https://developers.google.com/ar

[4] X. Suau, J. Ruiz-Hidalgo, and J. R. Casas, "Real-time head and hand tracking based on 2.5 d data," *IEEE Trans. Multimedia*, vol. 14, no. 3, pp. 575–585, Jun. 2012.

[5] S. Pillai and J. Leonard, "Monocular SLAM supported object recognition," 2015. arXiv:1506.01732.

[6] P. Wu, Y. Liu, M. Ye, J. Li, and S. Du, "Fast and adaptive 3d reconstruction with extensively high completeness," *IEEE Trans. Multimedia*, vol. 19, no. 2, pp. 266–278, Feb. 2017.

[7] K. Tateno, F. Tombari, I. Laina, and N. Navab, ''CNN-SLAM: Real-time dense monocular slam with learned depth prediction," in *Proc. IEEE Conf. Comput. Vis. Pattern Recogn.*, 2017, pp. 6565–6574.

[8] J.-C. Piao and S.-D. Kim, "Adaptive monocular visual–inertial slam for real-time augmented reality applications in mobile devices," *Sensors*, vol. 17, no. 11, 2017, Art. no. 2567.

[9] S. Weiss, M. W. Achtelik, S. Lynen, M. Chli, and R. Siegwart, "Real-time onboard visual-inertial state estimation and self-calibration of MAVs in unknown environments," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2012, pp. 957–964.

[10] S. Weiss, M. W. Achtelik, M. Chli, and R. Siegwart, "Versatile distributed pose estimation and sensor self-calibration for an autonomous MAV," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2012, pp. 31–38.

[11] M. Li, *Visual-Inertial Odometry on Resource-constrained Systems*. Ph.D. dissertation, Univ. California, Riverside, CA, USA, 2014.

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

[13] Z. Yang and S. Shen, "Monocular visual–inertial state estimation with online initialization and camera–IMU extrinsic calibration," *IEEE Trans. Autom. Sci. Eng.*, vol. 14, no. 1, pp. 39–51, Jan. 2017.

[14] P. Furgale, J. Rehder, and R. Siegwart, "Unified temporal and spatial calibration for multi-sensor systems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2013, pp. 1280–1286.

[15] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual–inertial odometry," *IEEE Trans. Robot.*, vol. 33, no. 1, pp. 1–21, Feb. 2017.

[16] M. Burri *et al.*, "The EuRoC micro aerial vehicle datasets," *Int. J. Robot. Res.*, vol. 35, no. 10, pp. 1157–1163, 2016.

[17] T. Chai and R. R. Draxler, "Root mean square error (RMSE) or mean absolute error (MAE)?–arguments against avoiding RMSE in the literature," *Geoscie. Model Develop.*, vol. 7, no. 3, pp. 1247–1250, 2014.

[18] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, ''ORB-SLAM: A versatile and accurate monocular slam system," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, Oct. 2015.

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

[20] B. D. Lucas *et al.*, "An iterative image registration technique with an application to stereo vision," in *Proc. 7th Int. Joint Conf. Artif. Intell.*, 1981, pp. 674–679.

[21] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[22] P. Rives, "Visual servoing based on epipolar geometry," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2000, pp. 602–607.

[23] G. H. Golub and C. Reinsch, "Singular value decomposition and least squares solutions," *Numerische Mathematik*, vol. 14, no. 5, pp. 403–420, 1970.

[24] B. K. Horn, "Closed-form solution of absolute orientation using unit quaternions," *J. Opt. Soc. Amer. A*, vol. 4, no. 4, pp. 629–642, 1987.

[25] 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.*, 2012, pp. 573–580.

[26] 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, 2015.

[27] A. Kasyanov, F. Engelmann, J. Stückler, and B. Leibe, "Keyframe-based visual-inertial online slam with relocalization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 6662–6669.

Jin-Chun Piao received the B.S. degree in computer science and technology from Beijing Forestry University, Beijing, China, in 2010 and the Ph.D. degree in computer science from Yonsei University, Seoul, South Korea, in 2018. He is currently an Instructor of computer science and technology with Yanbian University, Yanji, China. His research interests include AR-based ubiquitous computing, mobile computing, and graphics.

Shin-Dug Kim received the B.S. degree in electronic engineering from Yonsei University, Seoul, South Korea, in 1982, the M.S. degree in electrical and computer engineering from the University of Oklahoma, Norman, OK, USA, in 1987, and the Ph.D. degree from the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA, in 1991. He is currently a Professor of computer science with Yonsei University, Seoul, South Korea. His research interests include advanced computer systems, intelligent memory system design, and ubiquitous computing platforms. Prof. Kim is a member of the IEEE Computer Society.