Visual-Inertial RGB-D SLAM for Mobile Augmented Reality

Williem[1], Andre Ivan[1], Hochang Seok[2], Jongwoo Lim[2], Kuk-Jin Yoon[3], Ikhwan Cho[4], and In Kyu Park[1] ⋆

[1] Department of Information and Communication Engineering, Inha University, Incheon, Korea
{williem.pao@gmail.com, andreivan13@gmail.com, pik@inha.ac.kr}
[2] Division of Computer Science, Hanyang University, Seoul, Korea
{kaha0707@naver.com, jlim@hanyang.ac.kr}
[3] School of Electrical Engineering and Computer Science, GIST, Gwangju, Korea
{kjyoon@gist.ac.kr}
[4] Media Experience Lab, Corporate R&D Center, SK Telecom, Seoul, Korea
{ikhwan.cho@sk.com}

Abstract. This paper presents a practical framework for occlusion-aware augmented reality application using visual-inertial RGB-D SLAM. First, an efficient visual SLAM framework with map merging based relocalization is introduced. When the pose estimation fails, a new environment map is generated. Then, a map merging is performed to merge the current and previous environment maps if a loop closure is detected. The framework is then integrated with the inertial information to solve the missing environment map problem. Camera pose is approximated using the angular velocity and translational acceleration value when the pose estimation fails. Experimental results show that the proposed method can perform well in the presence of missing pose. Finally, an occlusion-aware augmented reality application is built over the SLAM framework.

1 Introduction

For the last decades, visual simultaneous localization and mapping (visual SLAM) has become an active research topic in robotics and computer vision. Visual SLAM utilizes a set of images to estimate the camera pose and generate the environment map simultaneously. Recently, visual SLAM plays an important role to provide geometric information for augmented reality (AR) and virtual reality (VR) applications. With the extracted camera pose and environment map, AR application can augment virtual graphic objects to it. Then, viewers can experience the augmented objects aligned in the real world which generates emerging experiences to the viewers. There are various kinds of visual SLAM based on the input type, such as monocular visual SLAM [2, 12, 13], visual RGB-D SLAM [3, 5, 6], visual-inertial SLAM [1, 9, 15, 19], etc.

In this paper, we focus on mobile visual-intertial RGB-D SLAM for AR application. Mobile visual SLAM often fails when there are motion blur, occlusion and distortion. Thus, it cannot estimate the relative pose between the previous and current frames. Conventional method [12] performs relocalization until it finds camera pose with enough

⋆ Corresponding author

inliers. Otherwise, we need to build the environment map from the beginning. To solve the failed pose estimation, we develop an efficient feature-based visual SLAM method. Instead of using conventional relocalization method, a map merging based relocalization method is utilized. We generate a new environment map whenever the pose estimation fails, and merge the environment maps when a loop closure is detected across the environment maps. However, it is not suitable for AR application because the current camera pose is reset and previous environment map information is lost. Thus, we integrate the system by employing the inertial information from the mobile device. While the pose estimation fails, the visual-inertial SLAM system exploits the inertial information to approximate the camera pose. Experimental results show that the proposed method can handle the missing pose problem with better visualization. The contribution of this paper is summarized as follows.

- The visual-inertial SLAM framework that is aware of failed pose estimation.
- Map merging based relocalization to correct the accumulated error in the visual-inertial SLAM framework.
- An occlusion-aware augmented reality application over the SLAM framework.

2 Related Works

Henry *et al.* [5] introduced a keyframe based system that utilized a RGB-D camera to generate dense 3D models. Sparse features were extracted to find the correspondence between consecutive frames. Each feature was projected into 3D space and then the camera pose was estimated using RANSAC algorithm. The camera pose was refined using iterative closest point (ICP) algorithm. A global pose optimization was performed to handle the camera drift. They exploited a loop closure detection for a subset of previous keyframes to look for the possible loop closure. The global optimization was done when the loop closure is detected. Fioraio and Stefano [3] introduced SlamDunk that employed a similar system as in [5] with more efficient computation. Instead of performing RANSAC for all possible corresponding keyframes, they utilized feature pool for fast and efficient feature matching. On the other hand, Kerl *et al.* [6] proposed a dense RGB-D SLAM instead of using sparse features. They focused on improving several components to reduce the drift.

Servant *et al.* [15] used inertial sensors to improve the performance of plane based SLAM. They performed the sensor fusion in Extended Kalman Filter (EKF) based framework. Leutenegger *et al.* [9] introduced a non-linear optimization for combining the visual and inertial information. Inertial error terms and landmarks reprojection error terms were utilized in the joint optimization framework. Tiefenbacher *et al.* [19] proposed an off-the-shelf sensor integration for monocular SLAM on mobile devices. They introduced a Unscented Kalman Filter (UKF) based sensor fusion and a UKF motion model. They employed PTAM algorithm [7] as the SLAM approach and used the estimated pose as the measurement input for UKF. Brunetto *et al.* [1] extended SlamDunk framework [3] and integrated it with inertial sensors. There were two filters used for filtering the camera orientation and camera position, separately. First, an orientation EKF was used to integrate the camera rotation and gyroscope information. Then, a position KF was used to integrate the camera position and accelerometer information.

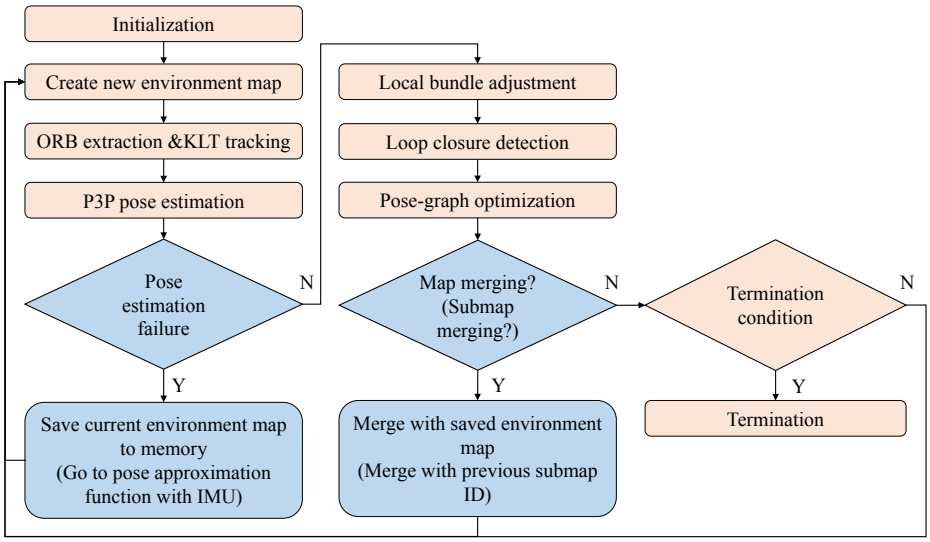


Fig. 1: Overview of the proposed visual (-inertial) SLAM system. It consists of motion estimation, local and global optimization, loop closure detection, and map merging. Pose estimation failure detection and map merging process is highlighted by blue-colored component. The functions used in visual-inertial system are described in the parentheses.

3 Proposed Method

3.1 Visual SLAM

Framework overview We propose a feature-based visual SLAM system which operates robustly even when the camera pose estimation fails. First, feature points are extracted and tracked to estimate the camera pose, and the environment map is constructed with the 3D landmarks obtained using the depth information. When the loop closure occurs, the pose-graph is optimized to minimize the accumulated pose errors, thereby improving the accuracy of the entire camera postures and 3D landmark positions. However, if sufficient number of feature points cannot be extracted or tracked due to severe motion blur and occlusion, the estimated camera pose may have large error and it may fail in the worst case. To alleviate this problem, we propose an efficient algorithm to maintain multiple environment maps and merge them if their relative poses are known in loop closure detection. Fig. 1 illustrates the overview of the proposed algorithm.

Pose estimation In our framework, ORB feature points [12] are used as the feature points for pose estimation. Extracted ORB feature points are tracked using the Kanade-Lucas-Tomasi tracker [16]. As the camera moves, the number of tracked ORB feature points decrease. When the number of tracked feature points is less than the threshold, new ORB feature points are extracted and added for tracking. The depth of feature

points are given from the depth images. The relative pose of the camera is calculated by the P3P RANSAC algorithm [8] using the pairs of tracked 2D feature points and 3D landmarks. If the relative translation or rotation of the current camera pose to the most recent keyframe is larger than thresholds, the current frame is registered as a new keyframe, a visual odometer edge links the previous and the new keyframes, and new features are added to the environment map as landmarks. Then local bundle adjustment (LBA) optimizes the poses of recent keyframes and the positions of landmarks.

Loop closure detection and pose-graph optimization Loop closure is tested when a keyframe is registered. First, N candidate keyframes are searched in a vocabulary tree [14]. Then, the feature points of the current keyframe and the candidate keyframes are matched using the P3P RANSAC algorithm. If the total reprojection errors and the inlier ratio satisfy the loop closure criteria, a loop closure edge is added in the environment map [4]. When a loop closure occurs, a pose graph composed of visual odometer edges and loop edges is constructed, and pose graph optimization is performed to minimize the accumulated keyframes pose errors [17].

Pose estimation failure detection We hypothesize that two cases of pose estimation failure.

1. Pose cannot be estimated - P3P RANSAC failure
2. Pose with very large error is calculated

To detect the case 2, we use the statistical difference of the number of tracked feature points and the relative pose between previous keyframes. The condition for failure of pose estimation is given as follows.

$$n > \mu_n + 2\sigma_n \quad | \quad T > \mu_T + 2\sigma_T \tag{1}$$

where n and T are the number of features and the relative translation of current keyframe, μ_n, σ_n and μ_T, σ_T are the mean and the standard deviation of the number of features and the relative translation in the previous five keyframes, respectively.

Map merging process When the pose estimation fails, the pose of the current keyframe cannot be estimated. In our system, the previous environment map is kept in background, and a new environment map is initialized with the current keyframe, into which the subsequent keyframes are added. Since the vocabulary tree contains all keyframes in existing environment maps, the loop closures not only in the current map but also the previous maps are tested using the P3P RANSAC algorithm whenever a new keyframe is added. If the best candidate keyframe is in a previous environment map, it is merged with the current map by adding a loop closure edge between the matching keyframes. In this process the keyframe poses and landmark positions in the previous map need to be adjusted using the estimated relative pose between the keyframes. Note that for a new keyframe at most one loop closure edge can be added regardless of a between-map or within-map loop closure.

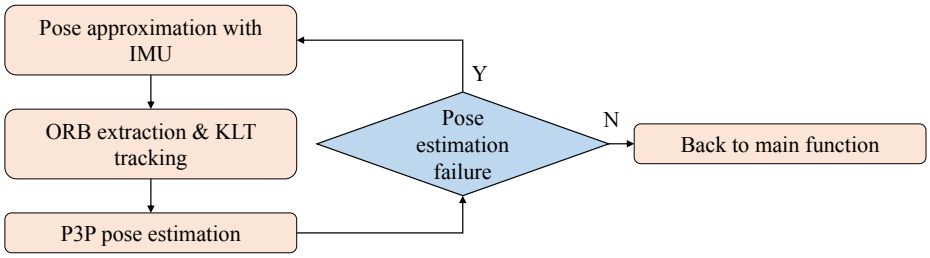


Fig. 2: Overview of the pose approximation function using inertial information.

3.2 Visual-Inertial SLAM

Framework overview The visual SLAM has a limitation that the current camera pose does not have relation with previous camera poses until the map merging process is done. This is a disadvantage when visual SLAM information is used by AR application because the camera pose is reset to initial and become inconsistent. To ameliorate this problem, we utilize inertial information when the pose estimation fails. Note that we do not integrate the visual and inertial information together, but use each of them alternatively. As inertial information is very noisy which can lead to large drift error, we propose a system to correct the camera pose when the visual SLAM can obtain reliable camera pose.

Instead of maintaining multiple environment maps as performed in visual SLAM, we generate only a single environment map with additional group identifier number (submap ID) to manage the keyframes, landmarks, and camera poses. The submap ID separates the keyframes, landmarks, and poses into different groups until they are merged later. Each optimization (local bundle adjustment, loop closure detection, and pose graph optimization) is performed only for keyframes with the same submap ID. Submap merging process, which shares similar concept as map merging, is done when a loop closure occurs across groups with different submap ID. The overview of the proposed visual-inertial SLAM is shown in Fig. 1 with the updated functions described inside the parentheses.

When the pose estimation fails, the proposed system operates the pose approximation function using inertial sensors as shown in Fig. 2. Instead of utilizing global 3D landmarks, we generate and utilize temporary 3D landmarks in this function to check the pose estimation. We execute the pose approximation function iteratively until the pose estimation does not fail. When it succeeds, the application goes back to the main function and execute other processes.

Inertial sensors usage First, we calibrate the low cost IMU using an IMU-calibration techniques by Tedaldi [18] to estimate the gyroscope and accelerometer biases. Using the angular velocity, we update the camera orientation by the following equation.

$$\theta^k = \theta^{k-1} + \omega^k \Delta t \tag{2}$$

where θ is the camera orientation and ω is the angular velocity. k is the frame index and Δt is the time interval between consecutive frames.

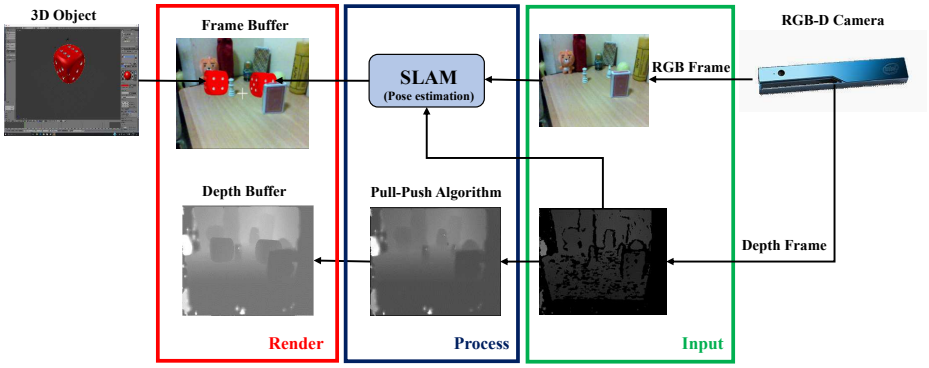


Fig. 3: Overview of the occlusion-aware augmented reality application.

We need to remove the gravity $g = [0, 0, 9.8]$ which is included in the obtained acceleration value a. Thus, we utilize a method in [10] to estimate the orientation of the IMU sensor. Using the obtained orientation, we can measure the linear acceleration a_l as the following equation.

$$a_l^k = \exp(\theta_d) a^k - g \tag{3}$$

where θ_d is the IMU device orientation and $\exp(\theta_d)$ is the function to generate a rotation matrix from an orientation θ_d. Using the linear acceleration, we perform double integration to estimate the position while the pose estimation is failed as follows.

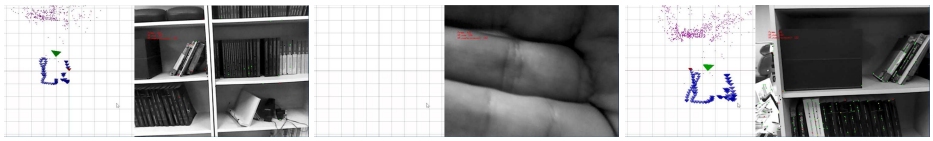
$$v^k = v^{k-1} + a_l^k \Delta t \tag{4}$$

$$p^k = p^{k-1} + v^k \Delta t \tag{5}$$

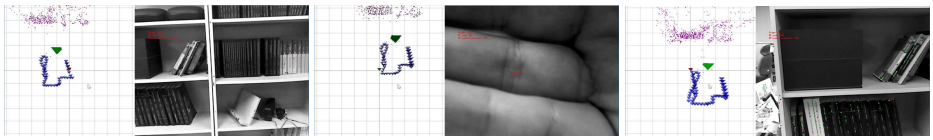
where v is the velocity and p is the camera position.

3.3 Occlusion-Aware AR

Using the obtained camera pose, we develop an occlusion-aware AR application under OpenGL rendering pipeline. Fig. 3 shows the overview of the application. The application receives a sequence of RGB and depth frames from RGB-D camera. Then, pose estimation process is run for each input frame. The estimated camera pose is utilized to augment the 3D graphic objects aligned in the real world. To build an occlusion-aware AR application, the depth frame is copied to the depth buffer every time. Pull-push algorithm [11] is utilized to fill the holes in raw depth frame. The 3D objects are loaded in advance and augmented to the environment map when needed. For each frame, the augmented objects are drawn in the world coordinates.



(a)



(b)

Fig. 4: Sample results of (a) visual SLAM and (b) visual-inertial SLAM. (Left) Frame 323; (Middle) Frame 423; (Right) Frame 523.

Table 1: Maximum and mean angular error.

Start Frame	Total Frame	Maximum error	Mean error
300	50	2.60°	0.02°
	100	8.08°	0.09°
	200	8.08°	0.07°
600	50	4.70°	0.03°
	100	5.97°	0.07°
	200	6.16°	0.07°

4 Experimental Results

The proposed algorithm is implemented on Microsoft Surface 4 Pro with its internal inertial sensors. For the camera, we utilize Intel RealSense R200 and use its SDK to perform the calibration between depth and color cameras. Fig. 4 (a) shows the qualitative results of the environment maps generated by the visual SLAM algorithm. Frame 323, 423, and 523 are selected to show the performance when the pose is missing. It confirms that the visual SLAM can find the previous environment map and merge it with the current environment map.

Visual-inertial SLAM refines the performance by keeping the environment map when the pose estimation fails. It is shown in Fig. 4 (b) which selects the same frame index. It confirms that the environment map is not missing in frame 423 and can approximate the camera pose because of the inertial information usage. To evaluate the performance of each inertial sensor (gyroscope and accelerometer) in the device, we capture two data which consists of only rotation and only translation, separately. Missing pose is synthetically generated by selecting a portion of frames as the missing frames. Then,

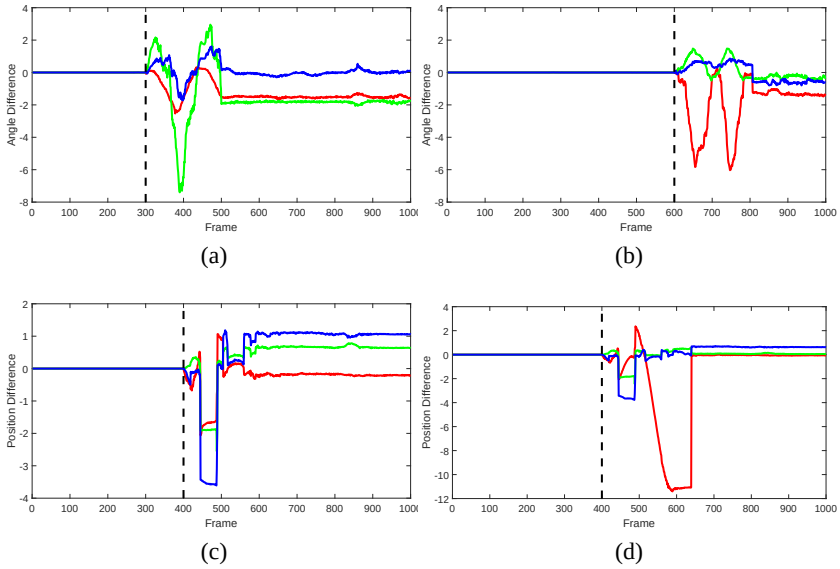


Fig. 5: Plot of angular error with 200 missing frames which start from (a) Frame 300 and (b) Frame 600; Plot of position error which starts from frame 400 with (c) 50 missing frames and (d) 200 missing frames. Red, green, and blue colors denote the difference in x, y, and z directions. The black dash line denotes the starting point of missing frame.

we compare the camera poses generated by visual SLAM and visual-inertial SLAM algorithms.

Table 1 shows the maximum and mean angular error between both algorithms. When the pose is missing, visual-inertial algorithm gain at most 8.08° error. This amount of error is acceptable as we do not have any information other than the inertial information. Fig. 5 (a) and (b) show the plot of angular error in all captured frames. It demonstrates that the camera orientation is corrected after the missing frames. This is due to the proposed submap merging process.

While the gyroscope can provide acceptable measurements for the rotation, the accelerometer has severely noisy measurement. It is difficult to provide accurate camera position by only using the accelerometer. Thus, accelerometer is only used for the rough position approximation and the error is corrected by the proposed submap merging process. In the only translation data, the missing frames start from frame 400. The total missing frames are 50, 100, and 200 and the maximum position error of each scenario is 3.4686, 3.7782, and $11,5271$, respectively. It confirms that the accelerometer is acceptable for short period but contains large error for long period. Fig. 5 (c) and (d) show the plot of position error in all captured frames. It proves that the proposed submap merging process can correct the position error after the missing frames.

Finally, the proposed SLAM framework is tested on the AR application. There are two scenarios to show the usability of the application. The first scenario is by moving the camera with static occlusion and the second scenario is by moving the occlusion (dy-

(a)

(b)

Fig. 6: Sample results of (a) static occlusion; (b) dynamic occlusion. Dices and globes are the augmented 3D graphic objects.

namic occlusion) with static camera. Fig. 6 shows the sample results of each scenario. It is confirmed that the proposed AR application shows correct occlusion handling by updating the depth buffer and camera pose in each frame.

5 Conclusion

In this paper, we proposed a visual-inertial RGB-D SLAM system that was aware of failed pose estimation problem. A new environment map was generated whenever the camera pose was missing. Then, a map merging process was performed to combine the previous and current environment maps when a loop closure occurred. A visual-inertial SLAM framework was introduced to handle the missing environment map problem. Inertial information was used to approximate the camera pose while the camera pose was missing. Both visual SLAM and inertial information were used alternatively to perform continuous pose estimation. Experimental results confirmed that the proposed framework could solve the failed pose estimation problem. Finally, an occlusion-aware AR application was developed over the proposed SLAM framework.

Acknowledgement

This work was supported by SK Telecom. This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (2017-0-00142)

References

1. Brunetto, N., Salti, S., Fioraio, N., Cavallari, T., Stefano, L.D.: Fusion of inertial and visual measurements for RGB-D SLAM on mobile devices. In: Proc. of IEEE International Conference on Computer Vision Workshops. pp. 148–156 (2015)
2. Davison, A.J., Reid, I.D., Molton, N.D., Stasse, O.: MonoSLAM: Real-time single camera SLAM. IEEE Trans. on Pattern Analysis and Machine Intelligence 29(6), 1052–1067 (2007)
3. Fioraio, N., Stefano, L.D.: Slamdunk: Affordable real-time RGB-D SLAM. In: Proc. of European Conference on Computer Vision Workshops. pp. 401–414 (2014)
4. Galvez-Lopez, D., Tardos, J.D.: Real-time loop detection with bags of binary words. In: Proc. of IEEE International Conference on Intelligent Robots and Systems. pp. 51–58 (2011)
5. Henry, P., Krainin, M., Herbst, E., Ren, X., Fox, D.: RGB-D mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments. International Journal of Robotics Research 31(5), 647–663 (2012)
6. Kerl, C., Sturm, J., Cremers, D.: Dense visual SLAM for RGB-D cameras. In: Proc. of IEEE International Conference on Intelligent Robotics and Systems. pp. 2100–2106 (2013)
7. Klein, G., Murray, D.: Improving the agility of keyframe-based SLAM. In: Proc. of European Conference on Computer Vision. pp. 802–815 (2008)
8. Kneip, L., Scaramuzza, D., Siegwart, R.: A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation. In: Proc. of IEEE Conference on Computer Vision and Pattern Recognition. pp. 2969–2976 (2011)
9. Leutenegger, S., Lynen, S., Bosse, M., Siegwart, R., Furgale, P.: Keyframe-based visual-inertial odometry using nonlinear optimization. International Journal of Robotics Research 34(3), 314–334 (2015)
10. Mahony, R., Hamel, T., Pflimlin, J.M.: Nonlinear complementary filters on the special orthogonal group. IEEE Trans. on Automatic Control 53(5), 1203–1217 (2008)
11. Marroquim, R., Kraus, M., RCavalcanti, P.: Efficient point-based rendering using image reconstruction. In: Proc. of Eurographgics Symposium on Point-Based Graphics. pp. 101–108 (2007)
12. Mur-Artal, R., Montiel, J.M.M., Tardós, J.D.: ORB-SLAM: A versatile and accurate monocular SLAM system. IEEE Trans. on Robotics 31(5), 1147–1163 (2015)
13. Newcombe, R.A., Lovegrove, S.J., Davison, A.J.: DTAM: Dense tracking and mapping in real-time. In: Proc. of IEEE Conference on Computer Vision and Pattern Recognition. pp. 2320–2327 (2011)
14. Nister, D., Stewenius, H.: Scalable recognition with a vocabulary tree. In: Proc. of IEEE Conference on Computer Vision and Pattern Recognition. pp. 2161–2168 (2006)
15. Servant, F., Houlier, P., Marchand, E.: Improving monocular plane-based SLAM with inertial measures. In: Proc. of International Conference on Intelligent Robots and Systems. pp. 3810–3815 (2010)
16. Shi, J., Tomasi, C.: Good features to track. In: Proc. of IEEE Conference on Computer Vision and Pattern Recognition. pp. 593–600 (1994)
17. Sünderhauf, N., Protzel, P.: Towards a robust back-end for pose graph SLAM. In: Proc. of IEEE International Conference on Robotics and Automation. pp. 1254–1261 (2012)
18. Tedaldi, D., Pretto, A., Menegatti, E.: A robust and easy to implement method for IMU calibration without external equipments. In: Proc. of IEEE International Conference on Robotics and Automation. pp. 3042–3049 (2014)
19. Tiefenbacher, P., Schulze, T., Rigoll, G.: Off-the-shelf sensor integration for mono-SLAM on smart devices. In: Proc. of IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 15–20 (2015)