applied system innovation

MDPI

Article

Visual SLAM Based Spatial Recognition and Visualization Method for Mobile AR Systems

Jooeun Song [1] and Joongjin Kook [2,*]

[1] Department of Electronics Information System Engineering, Sangmyung University, 31 Sangmyungdae-gil, Dongnam-gu, Cheonan-si 31066, Chungcheongnam-do, Korea; jueun5840@naver.com
[2] Department of Information Security Engineering, Sangmyung University, 31 Sangmyungdae-gil, Dongnam-gu, Cheonan-si 31066, Chungcheongnam-do, Korea
* Correspondence: kook@smu.ac.kr

Abstract: The simultaneous localization and mapping (SLAM) market is growing rapidly with advances in Machine Learning, Drones, and Augmented Reality (AR) technologies. However, due to the absence of an open source-based SLAM library for developing AR content, most SLAM researchers are required to conduct their own research and development to customize SLAM. In this paper, we propose an open source-based Mobile Markerless AR System by building our own pipeline based on Visual SLAM. To implement the Mobile AR System of this paper, we use ORB-SLAM3 and Unity Engine and experiment with running our system in a real environment and confirming it in the Unity Engine's Mobile Viewer. Through this experimentation, we can verify that the Unity Engine and the SLAM System are tightly integrated and communicate smoothly. In addition, we expect to accelerate the growth of SLAM technology through this research.

Keywords: mobile AR; markerless AR; SLAM; visual SLAM

Citation: Song, J.; Kook, J. Visual SLAM Based Spatial Recognition and Visualization Method for Mobile AR Systems. *Appl. Syst. Innov.* **2022**, *5*, 11. https://doi.org/10.3390/asi5010011

Academic Editor: Ondrej Krejcar

Received: 7 December 2021
Accepted: 4 January 2022
Published: 5 January 2022

1. Introduction

Simultaneous localization and mapping (SLAM) technology is gaining popularity in Drones and Augmented Reality (AR) with advancement in 3D Graphics technology and Machine Learning technology. As the demand for indoor applications increases, especially where there are no existing maps, positioning technologies such as SLAM help technology development in the robotics field. In addition, the use of SLAM in Visual SLAM algorithms and AR has promoted the growth of the SLAM market [1]. Currently, open source solutions for AR are not easily found in SLAM research fields. It burdens SLAM researchers as it requires them to conduct their own research and development with a deep understanding of similar systems when they build a SLAM-based AR system.

As shown in Figure 1, the SLAM system can be classified into direct SLAM and indirect SLAM. Direct SLAM systems such as LSD-SLAM [2] are mainly used in the AR field while Indirect SLAM systems such as ORB-SLAM [3] and Open-VSLAM [4] are being used in the robot or autonomous driving field. However, as these SLAM libraries are open source projects for general-purpose systems, it is somewhat difficult to implement AR contents on commercial devices such as mobile devices or AR headsets. The SLAM-based AR systems for commercial devices are provided in the form of platforms by companies such as Google, Apple, Maxst, and Vuforia, making it difficult to customize.

In this paper, we design a markerless AR and data pipeline via ORB-SLAM3, which can extract features in real time as it is the fastest among the existing open source SLAM libraries and offers a compromise between the quality and the performance among the modern SLAM systems, to implement a system that supports the development of AR contents on mobile devices or AR headsets through markerless SLAM [5].

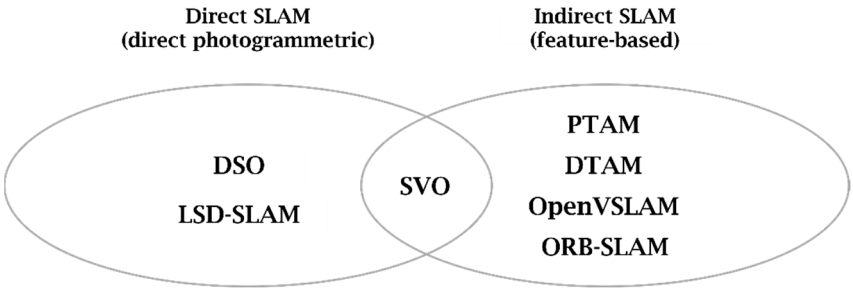

Figure 1. Direct SLAM and indirect (feature-based) SLAM.

Section 2 of this paper describes the typical SLAM-based AR development tools and the characteristics of ORB-SLAM3 used as the basis of the system proposed in this paper. Section 3 describes the overall configuration of the mobile AR system designed based on Visual SLAM and the functions of each component, and Section 4 shows the results of the proposed SLAM system in a real mobile environment. Finally, Section 5 describes the future development plans and expected effects for this study.

2. Related Works

Markerless AR using SLAM technology has greatly contributed to improving the usability of AR contents, and in particular, various open source SLAM projects have made it possible to use AR contents on various types of computing platforms.

2.1. Markerless AR

In the early days of AR, markers were used to connect the real world and the virtual world, and German AR company Metaio has been leading the marker-based AR market for years. Marker-based AR recognizes a marker from an image input through a camera and displays an image or object matching the marker on an image output to a display device. When detecting a marker with a simple pattern such as a QR code from the input image of the camera, the position or direction of the marker is also calculated, and the digital 3D content is visualized in the AR application to interact with the user. However, marker-based AR clearly has technical limitations, such as requiring a promise for a marker in advance and making tracking impossible when the camera leaves the marker.

In order to overcome the limitations of marker-based AR technology, SLAM technology uses various sensors instead of markers to move in an arbitrary space and search for information to estimate one's location and create a map of the space has emerged [6]. Markerless AR recognizes objects by detecting feature points of objects or scenes without prior knowledge of the environment, such as walls or intersections, and through this, 3D content is placed in a specific location. The markerless AR technology has improved the accuracy of image analysis due to the development of SLAM, a simultaneous positioning and mapping technology, allowing various sensor data to detect surroundings and synchronize the current position of the object in real time [7,8]. Representative SLAM-based markerless AR development tools include Google's ARCore, Apple's ARKit, and Qualcomm's Vuforia. Google's ARCore can be linked to OpenGL, Unity, and Unreal, and motion tracking, plane detection, and lighting estimation are possible. Apple's ARKit is also capable of motion tracking and plane detection, and with support for Unity and Unreal, real-time video rendering is possible. Qualcomm's Vuforia can recognize a single image or multiple images and can recognize and track 3D objects.

2.2. ORB-SLAM3

Visual SLAM can be implemented using images acquired by either a camera or other image sensor. It can be implemented at low cost using a relatively inexpensive camera and support various cameras of Mono/Stereo/RGB-D method. Visual SLAM can be classified into sparse type SLAM with a low map density and dense type SLAM with a high map

density. Additionally, Visual SLAM is ideal for sophisticated AR projects because maps can be saved and later reloaded [9–11].

In this paper, the ORB-SLAM3 system proposed in 2020 was used, improving the existing ORB-SLAM and ORB-SLAM2 among various visual SLAMs as shown in Figure 2. It supports VIO (Visual Inertial Odometry) and multi-map [12,13].

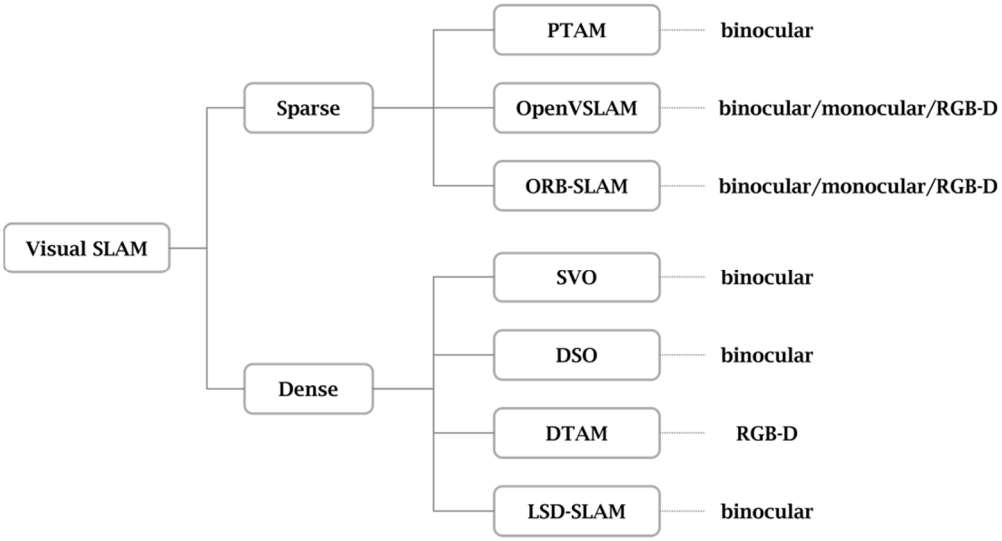

Figure 2. Classification of the Visual SLAM Technologies.

ORB (Oriented FAST and Rotated BRIEF) feature is a representative real-time image feature that improves the problem of the undirected FAST detector and greatly accelerates the overall image feature extraction by using a very fast binary descriptor, BRIEF. ORB-SLAM3 enables fast Edge extraction, and the main direction of the feature point is calculated by the ORB, and the rotation invariant property is added by the BRIEF descriptor. In addition, a simple descriptor that well expresses the surrounding image area is generated at the location where the feature point is extracted, and most of the feature extraction has excellent parallelism. Accordingly, it is suitable for the SLAM system that requires a lot of real-time demand because it maintains the data characteristics and the scale invariance and improves the speed.

Compared to ORB-SLAM2 proposed in 2017, ORB-SLAM3 as shown in Figure 3, the most advanced library among current ORB-SLAMs, adds a model that uses IMU values and a fisheye camera model to Mono/Stereo cameras. In addition, a multi-map management function has been added, and by creating a new map when the tracker is lost, the possibility of tracking failure is reduced, enabling continuous tracking, improving the loop detection algorithm, and detecting previously visited locations with higher probability.

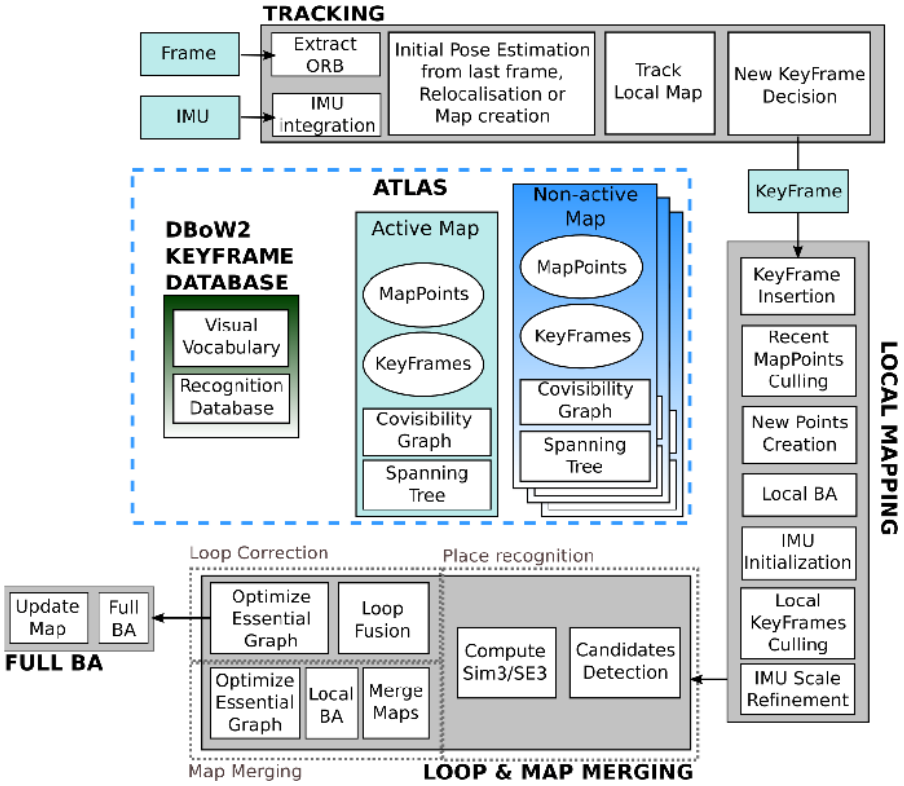

Figure 3. ORB-SLAM3 System Structure.

3. Visual SLAM Based Mobile AR System

The mobile SLAM system proposed in this paper uses ORB-SLAM3, one of the open source SLAMs, to data image-based spatial information, and uses Unity, which supports cross-platform, as a means to visualize it. In order for them to be integrated into one on Android to form a Mobile SLAM system, we need to provide some support.

3.1. Markerless AR System Design Using ORB-SLAM3

The overall structure of the mobile AR system based on Visual SLAM proposed in this paper is shown in Figure 4, and it is divided into mobile devices and ORB-SLAM3 libraries.

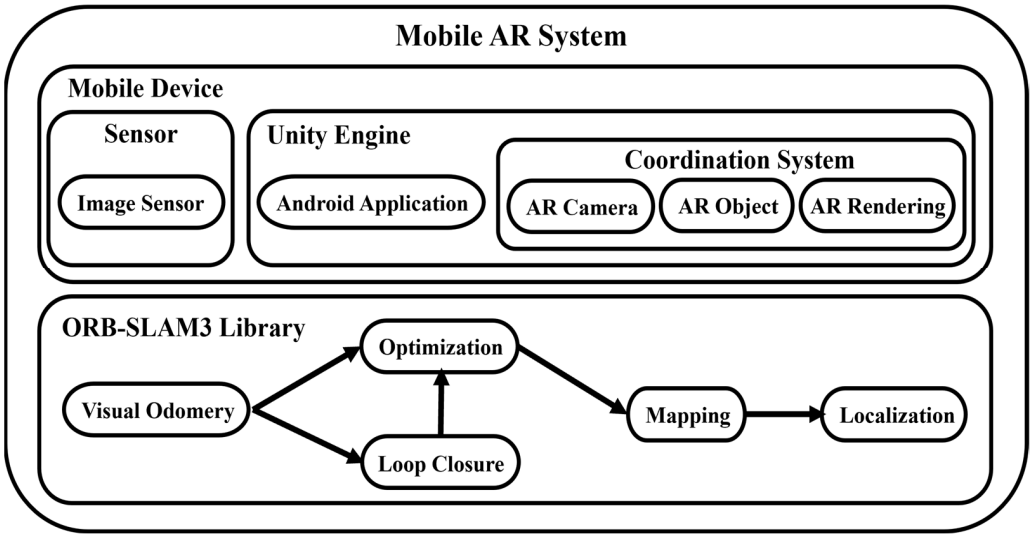

Figure 4. Proposed Mobile AR System Structure.

The mobile device uses the built-in cameras of the Android device to collect image data and uses the Android build module supported by the Unity Engine and the application using the Android SDK and NDK to show the SLAM operation results. The opensource ORB-SLAM3 library implemented in C++, was processed in the form of a plug-in through Clang build of LLVM to use it in the C#-based Unity Engine, providing basic SLAM functions such as mapping and localization.

3.2. Mobile Application with Unity

The existing ORB-SLAM3 uses Pangolin to visualize SLAM's key frames, camera poses, and point clouds, showing SLAM running with a pre-prepared data set. Pangolin is a lightweight and fast 3D visualization library for managing OpenGL displays and abstracting image input, which is widely used in computer vision as a means to eliminate the platform-specific boilerplate and easily visualize data [14]. In this paper, to design a SLAM system targeting the Android platform, the Unity Engine is used instead of Pangolin, and the API of the Unity Engine is used to control the camera and render the AR contents. Unity Engine is an authoring tool that provides a development environment for 3D and 2D video games and an integrated development tool for the creation of the interactive contents such as 3D animation, architectural visualization, virtual reality, and AR. It supports various platforms such as Windows, MacOS, iOS and Android. The Android application acting as an AR viewer in this system, can be built and used targeting Android by installing the Android build module, Android SDK and NDK tool supported by the Unity Engine.

3.3. SLAM Design on Android

The overall work procedure in the mobile SLAM system is shown in Figure 5.

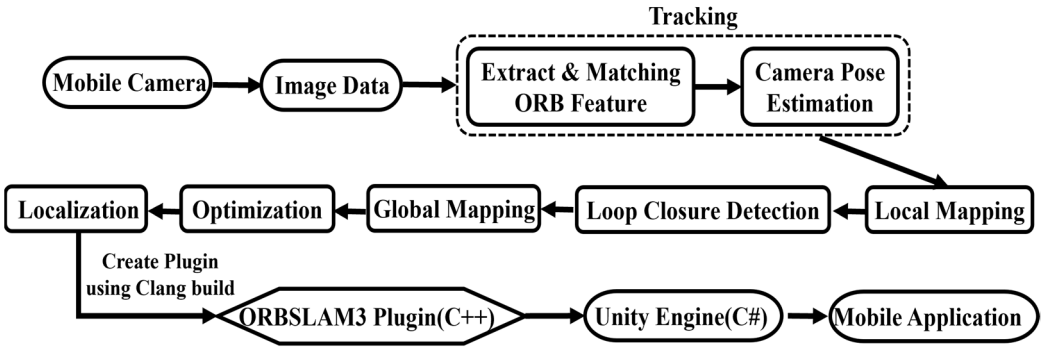

Figure 5. System Workflow.

Continuous image data is collected using the camera of the mobile device. The camera pose is approximately estimated by extracting the ORB feature for each image data and by calculating the position of the feature point compared to the nearest key frame. To create a map as similar as possible to the real environment by determining the correlation with the real environment using the camera pose information, BA (Bundle Adjustment) co-visibility graph, which indicates the relationship between key frames, was used, and a regional map, which is the map data at the time the camera is looking at, is drawn. Loop Closure Detection is performed on the local map to remove the accumulated drift; there are too many map points in the global map obtained by adding all the local maps, so localization is performed only with the camera pose, excluding the map points through optimization. BA (Bundle Adjustment) uses a non-linear optimization library called g2o to calculate the pose of the current frame as Motion-only BA, the pose of the recent key frame as Local BA, and the overall optimization as Loop Closing's pose graphing optimization and Full BA. All four optimizations are performed independently on different threads. Motion-only BA optimizes only the current pose with the key points of the map fixed, and Local BA optimizes the pose of the public visible key frame and the key points visible in the corresponding pose when a new key frame occurs. Loop Closing performs pose graph

optimization in a way of optimizing only with the pose perspective, excluding key points from the co-visibility graph, and then, the key frames and key points added through Full BA are updated [12,13].

To make SLAM available in the Unity Engine where only dynamic libraries are available, the SLAM system is made into a library through Clang build of LLVM and processed as a Unity plug-in. The complier-based LLVM helps to easily implement optimizations regardless of the programming language at compile time, link time, and runtime [15]. Clang is a compiler frontend for C, C++, Objective-C, and Objective-C++ programming languages that use LLVM as a backend [15]. It increases the convenience of development by provides faster compilation speed and faster execution speed than GCC compiler, and by providing more accurate error location and information. In order to use the function that delivers the image along with the initialization function and setting values of the SLAM system in the Unity Engine through the SLAM plugin created by Clang build, in this paper, the Unity Engine's [D11Import ("PluginName.d11")] function is used, and the required SLAM function is controlled by calling it. In this way, it is possible to implement SLAM-based AR technology in the Unity Engine through the built-in SLAM plug-in.

Continuous image data collected from Android mobile devices is delivered to the C++ SLAM library to provide pose data and point cloud data to the Unity Engine. Camera pose data can represent camera movement in a 4×4 matrix, and point cloud data is 1×3 vector data representing the position in space. This makes sure that implementing visualization of the current camera pose and surrounding environment are stored inside the SLAM.

4. Experiments and Results

Table 1 shows the experiment environment and device specifications of this system. Using the EuRoC MH03 dataset, the existing Pangolin viewer and the mobile viewer of the Unity engine of this paper were compared. The EuRoC MH03 dataset is a visual inertial data set collected from a Micro Aerial Vehicle (MAV), including stereo images, synchronized IMU measurements, and accurate motion and structural measurements and it is recorded as two pinhole cameras and an inertial sensor of Asctec Firefly hex-rotor drone [16]. The existing Pangolin viewer was run in the Ubuntu environment, and the Android viewer of the Unity engine of this paper was run in the Galaxy Note 20 Ultra 5G.

Table 1. Experiment Environment and Device Specifications.

System Type	Specification		
	OS	**CPU**	**Memory**
Desktop Computer	Ubuntu 21.04	AMD Ryzen 5 3600X 8-Core Processor 3.79 GHz	16 GB
Mobile Device	Android 10	Qualcomm Snapdragon 865+ ARM Cortex-A77 MP1 3.09 GHz ARM Cortex-A77 MP3 2.42 GHz ARM Cortex-A55 MP4 1.80 GHz	12 GB

Since the Pangolin viewer and the mobile viewer of the Unity engine used the same EuRoC MH03 data set to visualize the SLAM system, the number of point clouds is the same at about 32,000. The existing ORB-SLAM3 uses the Pangolin library to visualize the SLAM system as shown in Figure 6a, and in this paper, the Unity engine is used to visualize the SLAM system in the mobile environment. Compared with Figure 6a, Figure 6b shows that the SLAM system can be visualized in the Android viewer the same as Figure 6a, which is the existing viewer, although there is a difference in the size of the point cloud and the shape of the entire map. If the point clouds of Figure 6a,b are represented by *Meshlab* and *RTABMap*, the 3D visualization tools, they are expressed in 3D space as shown in Figure 6c,d.

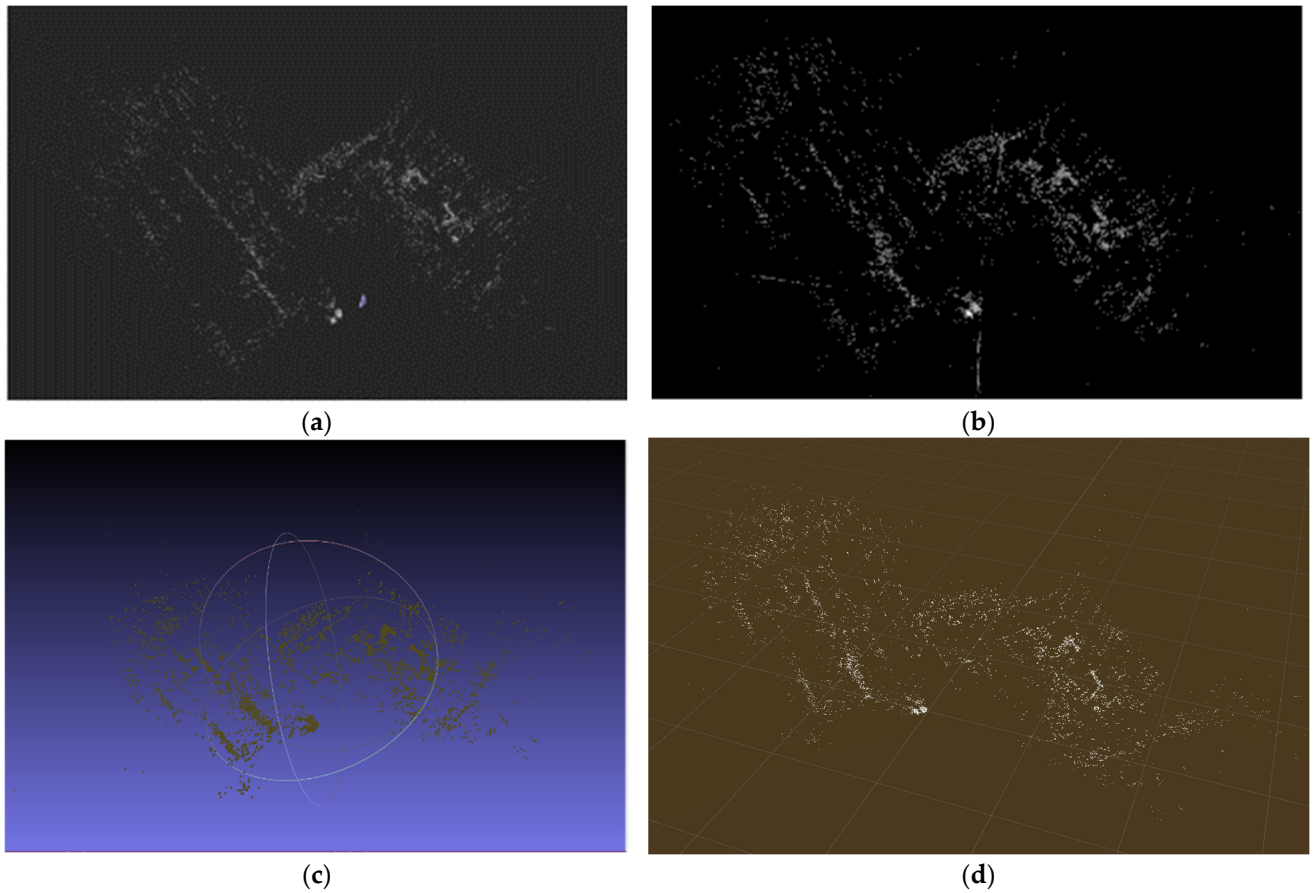

Figure 6. Visualization of SLAM; (**a**) Pangolin Viewer; (**b**) Mobile device; (**c**) 3D Visualization with Meshlab, (**d**) 3D Visualization with RTABMap.

As shown in Table 2, the two viewers were specifically compared based on CPU usage, memory usage, battery consumption, resolution, and fps. When running the Pangolin viewer, CPU usage was measured at 99% and memory usage was 23.1%, while when running the Android viewer of the Unity Engine, CPU and memory usage were measured to be less, 28% and 8.3%, respectively. The battery consumption of the device was also low. The resolution of each viewer was 1920 × 960 and 1280 × 720, and the fps was measured as 30 fps and 25 fps, respectively.

Table 2. Comparison of Pangolin Viewer and Unity Engine's Mobile Viewer.

Item	Pangolin Viewer	Mobile Device (Android)
CPU Usage	99%	28%
Memory Usage	23.1%	8.3%
Power Consumption	High (149 W)	Very low (1.3 W)
Resolution	1920 × 960	1280 × 720
Framerate	30 fps	25 fps

Through an experiment, it was confirmed that when the Android viewer of the Unity Engine was used instead of the Pangolin viewer, CPU and memory usage was small, but it took more time to visualize the SLAM system. In addition, it was confirmed that the Unity Engine and the SLAM system are closely integrated to enable smooth communication, and a SLAM-based AR system that can track camera poses and receive Mapping and Localization information was implemented. This shows that the foundation has been laid to accelerate further SLAM research and development.

5. Conclusions

In this paper, we designed a SLAM-based AR system in a mobile environment with good accessibility for the purpose of open source in order to reduce the burden of researchers' and to contribute to faster growth in the SLAM field. To this end, image data was obtained using the camera on mobile devices, creating a d11 plug-in to use Tracking, Local and Global Mapping, Loop Closure Detection, and Localization in the Unity engine, and enabling SLAM function to be checked in the mobile applications.

In future works, we plan to expand to the area of recognizing objects through learning of point cloud data and expressing 3D contents that match it and expand by adding content related to the actual expression of AR content. In addition, we would like to build a server based on this system to implement Collaborative SLAM among multiple clients to increase efficiency by reducing development costs such as labor costs and development time, and to improve the performance of SLAM in a mobile environment by implementing computational functions such as Mapping and Optimization on servers and by showing them in a mobile application. In addition, it is expected that more accurate localization will be possible in real time by obtaining IMU data from the mobile camera and by drawing a precise map.

Author Contributions: Conceptualization, J.S. and J.K.; methodology, J.S. and J.K.; software, J.S.; validation, J.S. and J.K.; formal analysis, J.S.; investigation, J.S.; resources, J.S.; data curation, J.S. and J.K.; writing—original draft preparation, J.S. and J.K.; writing—review and editing, J.K.; visualization, J.S.; supervision, J.K.; project administration, J.K.; funding acquisition, J.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by a 2021 research Grant from Sangmyung University.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. PR Newswire. Simultaneous Localization and Mapping Market Worth $465 Million by 2023-Exclusive Report by MarketsandMarkets(TM). 2018. Available online: https://search.proquest.com/docview/2149675668 (accessed on 20 December 2021).
2. Mur-Artal, R.; Montiel, J.M.M.; Tardos, J.D. ORB-SLAM: A Versatile and Accurate Monocular SLAM System. *TRO* **2015**, *31*, 1147–1163. [CrossRef]
3. Sumikura, S.; Shibuya, M.; Sakurada, K. OpenVSLAM: A versatile visual slam framework. In Proceedings of the 27th ACM International Conference on Multimedia, New York, NY, USA, 21–25 October 2019; pp. 2292–2295.
4. Engel, J.; Schöps, T.; Cremers, D. *LSD-SLAM: Large-Scale Direct Monocular SLAM*; Computer Vision—ECCV 2014; Springer International Publishing: Cham, Switzerland, 2014; pp. 834–849.
5. SLAM for Unity. Available online: https://github.com/songjueun/SLAM-For-Unity.git (accessed on 27 December 2021).
6. Cadena, C.; Carlone, L.; Carrillo, H.; Latif, Y.; Scaramuzza, D.; Neira, J.; Reid, I.; Leonard, J.J. Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age. *TRO* **2016**, *32*, 1309–1332. [CrossRef]
7. Chong, T.J.; Tang, X.J.; Leng, C.H.; Yogeswaran, M.; Ng, O.E.; Chong, Y.Z. Sensor Technologies and Simultaneous Localization and Mapping (SLAM). *Procedia Comput. Sci.* **2015**, *76*, 174–179. [CrossRef]
8. Chen, C.; Chen, W.; Peng, J.; Cheng, B.; Pan, T.; Kuo, H.; Hu, M. A Real-Time Markerless Augmented Reality Framework Based on SLAM Technique. In Proceedings of the 2017 14th International Symposium on Pervasive Systems, Algorithms and Networks & 2017 11th International Conference on Frontier of Computer Science and Technology & 2017 Third International Symposium of Creative Computing (ISPAN-FCST-ISCC), Exeter, UK, 21–23 June 2017; IEEE: New York, NY, USA, 2017; pp. 127–132.
9. Merzlyakov, A.; Macenski, S. A Comparison of Modern General-Purpose Visual SLAM Approaches. 2021. Available online: https://arxiv.org/abs/2107.07589 (accessed on 20 December 2021).
10. Di, K.; Wan, W.; Zhao, H.; Liu, Z.; Wang, R.; Zhang, F. Progress and Applications of Visual SLAM. *Acta Geod. Cartogr. Sin.* **2018**, *47*, 770–779. [CrossRef]
11. Gao, X.; Zhang, T. *Introduction to Visual SLAM: From Theory to Practice*; Springer: Singapore, 2021.
12. Campos, C.; Elvira, R.; Rodriguez, J.J.G.; Montiel, J.M.M.; Tardos, J.D. ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial, and Multimap SLAM. *TRO* **2021**, *37*, 1–17. [CrossRef]

13. Mur-Artal, R.; Tardos, J.D. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. *TRO* **2017**, *33*, 1255–1262. [CrossRef]
14. Pangolin. What is Pangolin; Retrieved 3 October 2021. Available online: https://github.com/uoip/pangolin (accessed on 24 January 2018).
15. Hsu, M. *LLVM Techniques, Tips, and Best Practices Clang and Middle-End Libraries*; Packt Publishing, Limited: Birmingham, UK, 2021.
16. Burri, M.; Nikolic, J.; Gohl, P.; Schneider, T.; Rehder, J.; Omari, S.; Achtelik, M.W.; Siegwart, R. The EuRoC micro aerial vehicle datasets. *Int. J. Robot. Res.* **2016**, *35*, 1157–1163. [CrossRef]