Integrated and scalable augmented reality multiplayer robotic platform

Tandianus, Budianto; Seah, Hock Soon; Wang, Li; Quah, Chee Kwang

2020

Tandianus, B., Seah, H. S., Wang, L., & Quah, C. K. (2020). Integrated and scalable augmented reality multiplayer robotic platform. Proceedings of SPIE - International Workshop on Advanced Imaging Technology (IWAIT) 2020. doi:10.1117/12.2566243

https://hdl.handle.net/10356/142569

https://doi.org/10.1117/12.2566243

Integrated and scalable augmented reality multiplayer robotic platform

Tandianus, Budianto, Seah, Hock Soon, Wang, Li, Quah, Chee Kwang

Budianto Tandianus, Hock Soon Seah, Li Wang, Chee Kwang Quah, "Integrated and scalable augmented reality multiplayer robotic platform," Proc. SPIE 11515, International Workshop on Advanced Imaging Technology (IWAIT) 2020, 1151516 (1 June 2020); doi: 10.1117/12.2566243

Event: International Workshop on Advanced Imaging Technologies 2020 (IWAIT 2020), 2020, Yogyakarta, Indonesia

Integrated and Scalable Augmented Reality Multiplayer Robotic Platform

Budianto Tandianus*[a], Hock Soon Seah[a], Li Wang[b], Chee Kwang Quah[c]
[a]School of Computer Science and Engineering, Nanyang Technological University, Singapore; [b]State Key Laboratory of Robotics and System Harbin Institute of Technology, China; [c]ST Electronics (Training & Simulation Systems) Pte Ltd Singapore

ABSTRACT

We propose a scalable AR (Augmented Reality) multiplayer robotic platform, which enables multiple players to control different machines (a drone and a robot) in shared environments, i.e virtual and real environments. We use state-of-the-art visual SLAM (Simultaneous Localization and Mapping) algorithms for tracking machine poses based on camera and IMU (Inertial Measurement Units) inputs. Players will observe consistent AR objects between them thanks to our backend system, which synchronizes the AR objects between players. Moreover, the system is scalable in term of hardware (e.g. IMU, camera, machine type) and software (SLAM algorithm) as we utilize ROS for communication between modules. We demonstrate our system on a game developed in Unity, a robust and widely used popular game engine. We present some statistics of the game such as its frames-per-second performance.

Keywords: SLAM, ORB-SLAM2, VINS-Mono, ROS, Unity, Drone, Multiplayer

INTRODUCTION

In this multiplayer robotic platform, each player controls a machine (drone or robot) remotely from a GCS (Ground Control Station), which is a laptop, over a wireless network. Video feed from the machines are streamed and viewed in the respective GCS. AR targets are inserted into the video feed in the GCS. In order to accurately track the AR targets, we use SLAM algorithms, which compute the machine pose (position and orientation) based on the video feed. Our system supports multiplayer thanks to a backend server that synchronizes all players. Our system has the following features:

1. Accurate tracking: we incorporate state-of-the-art SLAM algorithms, i.e. VINS-Mono [1] and ORB-SLAM2 [2]. Tracking is essential in order to be able to place an AR object correctly.
2. Heterogeneous and scalable: it supports more than one type of machines such as aerial drones and ground robots. Moreover, it can be extended to other machine types such as underwater drone, customized robots based on Lego Mindstorms bricks, and so on. This is because we utilize ROS in our system.
3. Modular system: thanks to ROS that we are using, the system is modular. We can easily switch various components with minimal changes (e.g. SLAM algorithms, camera devices, IMU devices, etc.) as long as they support ROS.
4. Flexible game engine: we use Unity, a flexible and portable game engine. Applications developed in Unity can be compiled to various platforms. Moreover, it is good for fast prototyping and it has intuitive UI, in contrast to using C++ or OGRE3D, which needs to build the game from scratch.
5. Complete backend solution: the server in our system consists of two components: database server for handling non-real-time requests (e.g. player and map information) and simulation server for handling real-time requests (e.g. syncing players' poses and dynamic objects' states).

We demonstrate our system by applying it to a simple multiplayer AR game, i.e. a shooting game in which players shoot AR targets within a time limit. Our proposed system has various training applications, such as military, search and rescue, and delivery training.

* btandianus@ntu.edu.sg

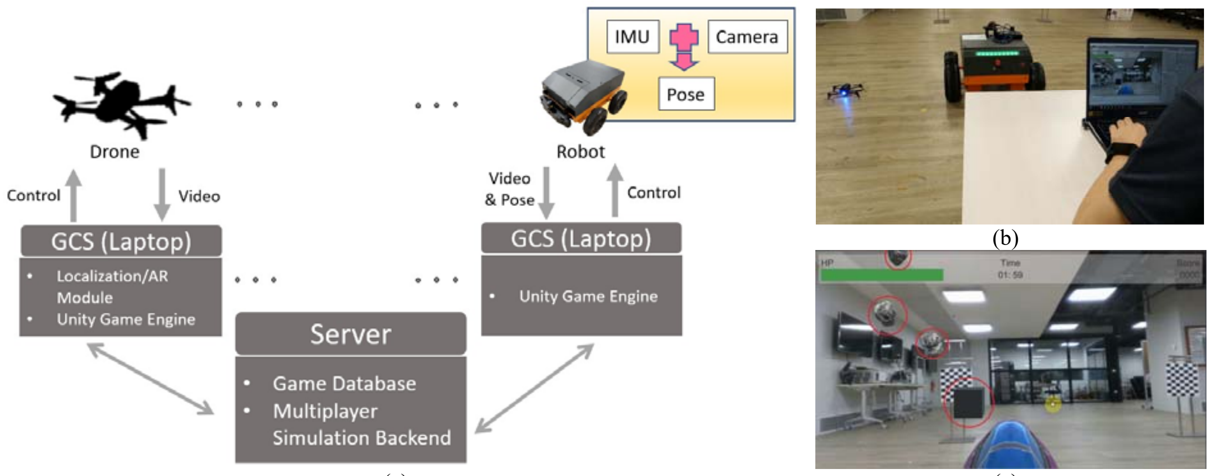
International Workshop on Advanced Imaging Technology (IWAIT) 2020, edited by Phooi Yee Lau, Mohammad Shobri, Proc. of SPIE Vol. 11515, 1151516 · © 2020 SPIE
CCC code: 0277-786X/20/$21 · doi: 10.1117/12.2566243

RELATED WORK

One main component in the proposed system is SLAM. Hence, we provide a brief overview of state-of-the-art SLAM algorithms. Visual-Inertial SLAM is a family of methods that computes pose (position and orientation) and surrounding map based on inputs from camera(s) and an IMU. Visual SLAM methods can be categorized into two categories: filtering-based [3, 4] and optimization-based [1, 5] methods. Filtering-based methods generally use Extended Kalmann Filter (EKF) method that is based on Bayesian filter. Filtering-based methods are generally considered as an online SLAM methods as current estimation is augmented and refined by incorporating new measurements during runtime. Optimization-based method, on the other hand, performs optimization on all or a set of frames simultaneously by minimizing a set of predefined cost functions. Thus, optimization-based methods are generally called full SLAM methods. As Visual Inertial SLAM method computes by combining both camera and IMU inputs, there are two approaches in doing so: loosely-coupled [3, 5] and tightly-coupled [1, 4] methods. In loosely coupled methods, the system estimates the states (e.g. poses) from visual and inertial data separately and combine them later. On the other hand, in tightly-coupled methods, the system estimates the states by directly combining raw measurement of visual and inertial data.

Optimization-based methods obtain pose by minimizing cost functions over several keyframes in a window. As a result, it will produce more accurate result as it keeps refining previous keyframes. The downside of this approach is that it requires higher computational power and runtime memory. Filter-based methods, on the other hand, is a well-established SLAM approach. It was developed when the computing power was limited, and thus they are more efficient in resource requirement at the expense of accuracy. This is due to recursive nature that considers the only latest keyframe instead of several keyframes simultaneously. The disadvantage of recursive approach is that it is less accurate compared to optimization approach as errors will build up over the time. Moreover, generally filter-based methods do not have loop closure mechanism. For more comprehensive state-of-the-arts review of SLAM solutions, readers can refer to existing publications [6–8].

PROPOSED SYSTEM



Video 1. Figure 1. (a) System architecture. (b) The physical setting of the game, with the drone on the left, the robot on the right, and a laptop controlling the robot. (c) Screenshot of the robot GCS in (b). The red circles show the AR objects. http://dx.doi.org/10.1117/12.2566243.1

In this project, we develop an Augmented Reality (AR) multiplayer system incorporating state-of-the-art SLAM algorithms. One of the main requirements of an AR system is accurate tracking. We could achieve this by incorporating the latest sensor fusion algorithms, VINS-Mono [1] and ORB-SLAM2 [2]. We show our system architecture in the Video 1. **Figure 1.**

The system consists of two types of machines, i.e. robot and drone. Each machine is controlled by a laptop, which we call GCS (Ground Control Station). Both the drone and robot are able to capture video feed from the camera installed on their systems. The robot has an additional input, i.e. an IMU that helps in performing more accurate SLAM computation compared to SLAM computation, which is based solely on a camera input. Our system uses ROS (Robotic Operating system) middleware in order to cater for communications between components in the system and hence it is modular and scalable. For instance, we can replace the robot and drone with other type of machines. In the lower level, we could replace each component (e.g. camera, IMU, and so on) in the robot with ease, as long as the replacement components are using ROS. Due to differences in specification and capability of robot and drone, the implementations between the robot and drone differ slightly which we explain in the following paragraphs.

The robot body is where the system receives input from an environment, i.e. video feed (from camera installed on the robot body) and orientation & acceleration (from IMU installed on the robot body), performs SLAM, and sends the calculated pose to the GCS. We incorporate VINS-Mono [1], a state-of-the-art sensor-fusion-based Visual-Inertial SLAM algorithm which computes pose (i.e. position and orientation) based on camera and IMU inputs. The GCS is where player interacts with the system, e.g. sees video feed and gives input command to the robot. The main component in the GCS is a game application developed in Unity. As the laptop is using Microsoft Windows environment and the robot is using Ubuntu and mainly ROS, we use Ros# to bridge the communications between the robot and the GCS.

The drone architecture is largely similar to the architecture of the robot. The main difference between the drone and robot is that SLAM computation is pushed to the GCS. This is because the drone we used does not have capability in performing heavy computation, i.e. SLAM. Moreover, the drone also does not have an IMU and therefore we use ORB-SLAM2 [2], a monocular SLAM algorithm. As the GCS is using Microsoft Windows platform and the ORB-SLAM2 is in Linux platform, we install a Virtual Machine so that we can run the ORB-SLAM2 in a Linux environment in the GCS.

In order to cater for multiplayer, we have a backend server, which is responsible for syncing and distributing information among the machines. There are two servers in our system. The first server is a database server for non-real-time purpose. It is used to store map data, player data, and so on. We design and develop endpoints using typescript for MongoDB. Using endpoints, we are able to provide CRUD services to developers around the world. In order to ease functional and load testing, JMeter is introduced to ease testing. The second server is a simulation server for real-time updating. During gameplay, each player updates its real-time position to the simulation server that will broadcast each player's position to all players. It uses Node.js script. In addition to receiving and broadcasting each player's status (i.e. pose), it also broadcasts any status of any dynamic objects in the scene, such as destroyed objects.

As mentioned, we use VINS-Mono in robot and ORB-SLAM2 in drone. Compared to ORB-SLAM2, VINS-Mono can achieve better accuracy in term of accuracy as it fuses camera and IMU data as ORB-SLAM2 uses only data feed from a single camera. As SLAM tracking uses visual feed, the tracking performance depends on how the algorithm can extract features from camera feed. If the visual feed does not contain enough visual features, SLAM algorithms will generally unable to track anymore. By incorporating IMU data, sensor fusion-based algorithm can continue the tracking, hence the superiority of VINS-Mono over ORB-SLAM. One of the inputs to VINS-Mono are extrinsic parameters between a camera and an IMU (i.e. physical spatial and orientation differences between camera and IMU hardware). Hence, VINS-Mono can track with more accurate scale compared to ORB-SLAM2 (whose input is only a single camera feed).

According to existing tests, VINS-Mono can yield the most accurate localization at the expense of more CPU resource usage [8, 9] compared to its competitors. Moreover, the benefit (i.e. accuracy) of VINS-Mono outweighs its drawback (i.e. high CPU and RAM consumption) in our system as we install VINS-Mono in a robot, which is very modular and extendable (we can always upgrade its CPU and RAM). Note that based on the tests [8, 9] generally optimization-based SLAM methods such as VINS-Mono can yield better accuracy compared to filtering-based due to error accumulation in filtering-based SLAM.

GAME DESIGN

We develop a simple multiplayer game in order to demonstrate our system. In this game, players need to navigate their machines in an unknown environment and destroy all AR targets while avoiding them within a given time limit. Before the game start, they will be asked to calibrate their machines. Calibration involving initializing SLAM tracking (ORB-SLAM2 for drone and VINS-Mono for robot) and resetting players' position in game world to origin (for both drone and robot). Moreover, in the calibration we also obtain scaling factor for ORB-SLAM2 used by the drone. If the calibration is not done, SLAM algorithm might not give correct tracking and players may be spawned at random positions. Thus, VINS-Mono calibration consists of several random movements in order to initialize its position and mapping and ORB-SLAM2 involves an additional calibration step, i.e. manually moving the drone to any predefined distance (1.5 meters in our experiment) after initialization in order to calibrate its scaling factor. We show a photograph and a screenshot of the game in Figure 1.

We tested the game on two laptops with one laptop controlling the robot (Intel i7-8550U 1.8GHz, 12 GB RAM, NVIDIA GeForce MX150) and another laptop (Intel i7-8700K 3.7 GHz, 64 GB RAM, NVIDIA GTX 1080) controlling the drone and hosting the server. The drone GCS performance was around 72 frames-per-second and the robot GCS was around 152 frames-per-second).

CONCLUSION AND FUTURE WORK

We have demonstrated our multiplayer AR system to be heterogeneous and scalable as it supports various machines, SLAM algorithms, and components. There are still rooms for improvement in our system. For instance, enabling overrides, i.e. the server takes over a player's machine during an emergency such as shutting it down during an imminent physical collision event. We could also expand the system to Unreal game engine, another game engine similar to Unity that is also widely used.

ACKNOWLEDGEMENT

The research was partially supported by the ST Engineering-NTU Corporate Lab through the NRF corporate lab @ university scheme, the China Scholarship Council (201706120137), and the Centre for Augmented and Virtual Reality.

REFERENCES

[1] T. Qin, P. Li, and S. Shen, "VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, Aug. 2018.

[2] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017.

[3] N. de Palézieux, T. Nägeli, and O. Hilliges, "Duo-VIO: Fast, light-weight, stereo inertial odometry," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 2237–2242.

[4] Z. Zhang, S. Liu, G. Tsai, H. Hu, C.-C. Chu, and F. Zheng, "PIRVS: An Advanced Visual-Inertial SLAM System with Flexible Sensor Fusion and Hardware Co-Design," *arXiv:1710.00893 [cs]*, Oct. 2017.

[5] J. M. Falquez, M. Kasper, and G. Sibley, "Inertial aided dense & semi-dense methods for robust direct visual odometry," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 3601–3607.

[6] F. Hidalgo and T. Bräunl, "Review of underwater SLAM techniques," in *2015 6th International Conference on Automation, Robotics and Applications (ICARA)*, 2015, pp. 306–311.

[7] C. Cadena *et al.*, "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, Dec. 2016.

[8] C. Chen, H. Zhu, M. Li, and S. You, "A Review of Visual-Inertial Simultaneous Localization and Mapping from Filtering-Based and Optimization-Based Perspectives," *Robotics*, vol. 7, no. 3, p. 45, Sep. 2018.

[9] J. Delmerico and D. Scaramuzza, "A Benchmark Comparison of Monocular Visual-Inertial Odometry Algorithms for Flying Robots," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 2502–2509.