Journal of Network and Systems Management (2024) 32:40
https://doi.org/10.1007/s10922-024-09809-9

An Edge Cloud Based Coordination Platform for Multi-user AR Applications

Balázs Sonkoly[1,3,4] · Bálint György Nagy[1,3,4] · János Dóka[1,3,4] · Zsófia Kecskés-Solymosi[1,3] · János Czentye[1,3,4] · Bence Formanek[2] · Dávid Jocha[2] · Balázs Péter Gerő[2]

Received: 30 November 2023 / Revised: 7 February 2024 / Accepted: 9 February 2024 /
Published online: 1 April 2024

Abstract

Augmented Reality (AR) applications can reshape our society enabling novel ways of interactions and immersive experiences in many fields. However, multi-user and collaborative AR applications pose several challenges. The expected user experience requires accurate position and orientation information for each device and precise synchronization of the respective coordinate systems in real-time. Unlike mobile phones or AR glasses running on battery with constrained resource capacity, cloud and edge platforms can provide the computing power for the core functions under the hood. In this paper, we propose a novel edge cloud based platform for multi-user AR applications realizing an essential coordination service among the users. The latency critical, computation intensive Simultaneous Localization And Mapping (SLAM) function is offloaded from the device to the edge cloud infrastructure. Our solution is built on open-source SLAM libraries and the Robot Operating System (ROS). Our contribution is threefold. First, we propose an extensible, edge cloud based AR architecture. Second, we develop a proof-of-concept prototype supporting multiple devices and building on an AI-based SLAM selection component. Third, a dedicated measurement methodology is described, including energy consumption aspects as well, and the overall performance of the system is evaluated via real experiments.

Keywords Augmented reality · Edge computing · SLAM

✉ Balázs Sonkoly
sonkoly.balazs@vik.bme.hu

[1] HSN Lab, Department of Telecommunications and Media Informatics, Faculty of Electrical Engineering and Informatics, Budapest University of Technology and Economics, Budapest, Hungary

[2] Ericsson Research, Budapest, Hungary

[3] MTA-BME Network Softwarization Research Group, Budapest, Hungary

[4] HUN-REN-BME Cloud Applications Research Group, Budapest, Hungary

1 Introduction

Extended Reality (XR) applications, including Augmented Reality (AR), Mixed Reality (MR) and Virtual Reality (VR), were launched from science fiction and have already landed in "reality". These applications can reshape our society enabling novel services, new ways of interactions and immersive experiences in many fields. Online gaming [1], Industry 4.0 [2], healthcare [3] and architecture [4] are just highlighted examples with special importance and business potential. The features and requirements of the use cases vary widely, however, the systems under the hood and the applied concepts share common components and properties.

One of the key components mainly affecting the user experience is the AR device, including the AR glasses (e.g., Microsoft HoloLens 2, Magic Leap 1) or head-mounted VR displays (e.g., Oculus Quest 2, Sony PlayStation VR, HTC Vive Cosmos) or AR-capable mobile phones (e.g., iPhone 12 Pro, Samsung S10). The other essential constituents are the special purpose, compute-intensive functions, running in the background, which enable the real-time operation. For example, 3D rendering, simulation of the virtual 3D environment, continuous localization of the devices, detecting and tracking of objects, users, gestures are crucial tasks, and the performance characteristics of the respective implementations significantly affect the quality of the immersive experience. The heterogeneity of these services indicates different hardware requirements [5], and we have multiple options where to run the corresponding tasks. A straightforward strategy is to implement all functions in the device itself. For example, HoloLens 2 follows this approach and encompasses a "gamer PC" in the device with the needed GPU/CPU capacity and the versatile software stack. As a result, these glasses are very expensive ones. Another option is to split the rendering function from the headset and deploy to a local PC. VR headsets usually follow this approach and require dedicated connection to a local rendering engine.

But why should we stop there? Today, several providers operate cloud based remote rendering platforms and the customers can rent the resources on demand instead of buying expensive hardware devices. Of course, this setup requires stable, high speed and low latency network connections. These services are often combined with online gaming offerings, such as in NVIDIA GeForce Now or PlayStation Now. Furthermore, we can make more steps towards more utilization of cloud platforms. Other functions can also be offloaded to the cloud if the end-to-end latency is controlled carefully and kept within a predefined bound. Edge computing is an emerging concept bringing compute resources closer to the users and end devices. Thus, it provides a suitable platform and execution environment for the backend components of latency sensitive and compute intensive applications, and it can help a lot with the energy consumption of mobile AR devices. If we manage to offload XR functions, the battery life can be multiplied. Fortunately, the networking requirements are (or more precisely, will be) fulfilled by 5G systems or a new generation of WiFi networks [6].

In an earlier version of this work [7] we presented a novel edge cloud based AR platform for running distributed, multi-user AR applications. We investigated

the benefits and also the limits of the approach and evaluated the performance of a proof-of-concept prototype which offloads the Simultaneous Localization And Mapping (SLAM) function.

In this paper, we extend that work with study of collaboration of multiple AR devices, energy consumption measurements and feasibility check. Furthermore, we improved the DNN-based SLAM selector module (SIA), which selects the best from the results of different SLAMs according to the current conditions.

The rest of the paper is organized as follows. In Sect. 2, the related works are presented. Section 3 is devoted to the architecture including the main design goals and concepts. Section 4 describes the relevant implementation details and the revealed issues. Section 5 presents our measurement methodology, while in Sect. 6, we evaluate the performance of the overall system and summarize our main findings. Finally, Sect. 7 concludes the paper.

2 Related Work

In this section, the relevant SLAM algorithms and the challenges of collaboration among the participant devices are summarized. In addition, the research works on offloading the SLAM functions to cloud or edge infrastructures are also presented.

2.1 SLAM Algorithms

Image-based camera localization is a key task in many of today's hot research fields, such as robotics [8], autonomous vehicles [8] and also virtual and augmented reality [9]. During the last decades, several methods and approaches have been proposed by researches to determine the pose, i.e., position and orientation, of the camera in real time (called visual odometry) and to build the map of the discovered environment. Such algorithms, called Simultaneous Localization and Mapping (SLAM), aim at creating the map of the surrounding environment and locating the device within them. The performance of visual SLAM algorithms heavily relies on the sensors exploited for tracking the devices. The most important one is the basic camera providing RGB color information for image processing purposes, and the advanced RGB-D camera extending the data with per-pixel depth information [10, 11]. Camera state acquisition can be refined in other steps of the algorithms based on additional sensors. Inertial Measurement Unit (IMU) is the source of augmented data for visual-inertial SLAMs [12], which consists of the gyroscope measuring angular velocity and orientation, and the accelerometer that is in charge of tracking the change of linear acceleration.

The majority of existing techniques rely on visual geometric models, called model-based SLAM methods [13]. On the one hand, monocular SLAM algorithms, using a single camera, implicitly estimate camera ego-motion, while incrementally building the map of the environment. In [9] a novel benchmarking method is defined for this type of algorithms and several proposals are evaluated quantitatively. On the other hand, multi-ocular SLAM methods, use two

or more cameras in order to acquire the color and depth information, as well [14]. Model-based solutions can be divided into *i*) feature-based algorithms that aim to find certain features or key points of the environment, making use of e.g. filtering techniques, and compute the camera pose information and the map based on these observations and *ii*) direct SLAM methods that use image intensities to estimate the location and surroundings [15]. For example, ROVIOLI [16] is a feature-based algorithm, while LSD-SLAM [17] is a direct method. Furthermore, keyframe-based SLAM approaches separate localization and mapping steps [18, 19]: camera localization takes place on regular frames over the subset of the map, while optimization takes place on certain keyframes. Different techniques can also be combined in hybrid algorithms to achieve better performance (see e.g., ORB-SLAM3 [20–24]). Besides the model-based SLAM techniques, data-driven, deep learning based methods have also been proposed in recent years [25–30].

2.2 Collaboration

Other challenges arise when multiple AR devices are collaborating [31] and the research community has just recently targeted this field [32–35]. For example, when multiple AR devices collaborate in the same physical environment, a joint coordinate system for the virtual 3D world has to be shared among the users. There are known methods for coordinate system synchronization, such as mechanisms based on well-known points (e.g., QR code), anchor-based synchronization (e.g., ARAnchor) and distributed SLAM algorithms [35–37]. Well-known points and anchor-based methods are typically capable of only offline synchronization and during the online operation the device local SLAMs and the local maps are used. Distributed SLAM algorithms follow a distributed approach to build a Global Map where the map data is shared among the devices and the server side. However, this operation can raise severe security issues because each AR device can access data recorded by the camera of any other AR devices.

In [38] a novel collaborative SLAM system is proposed where autonomous agents run visual-inertial odometry algorithms locally on the devices while sharing map information with a central server which is responsible for merging and optimizing the global map. The solution works only with keyframe-based VIO systems. Another client–server based collaborative SLAM framework is proposed for service robots in [39]. Tracking and local mapping take place on the devices (clients) and after processing, filtered data (keyframes and landmarks) are exchanged with the server which is in charge of the loop detection and map merging. Based on the client's movement, selected part of the global map is sent to the client on-demand. Both solutions [38, 39] require processing on the devices and lack a full remote SLAM at the server side. In contrast, our proposal offloads the full SLAM stack to the edge infrastructure, it supports arbitrary SLAM algorithms and additional AR services as the camera and IMU data are available at the edge cloud.

2.3 Offloading to Edge/Cloud

Computation-intensive algorithms in SLAM systems require proper equipment and fast, reliable processing solutions. Embedded systems, such as smartphones and smart glasses are poorly equipped in the long run as they are still dependent on battery constraints. Edge and Cloud Computing offer the possibility to offload excessive computational tasks, e.g., the SLAM algorithm, from the devices to a server in the cloud or at the edge of the network close to the user in terms of latency [40–43]. Thus, future augmented and mixed reality (AR/MR) applications can leverage the additional resources and dedicated hardware support (GPU) provided by cloud/edge servers for improved SLAM services, such as enhanced localization scalability and performance [44], improved response time based on parallel processing [45], collaborative localization by a globally synchronized map [46] or multi-user support [47], along with reduced power consumption of the AR device. Besides the SLAM service, other AR functions can also be offloaded to the cloud/edge infrastructure, including e.g., the rendering [48] or object tracking [49] functions. Of course, the capabilities and the characteristics of the underlying networks significantly impact the feasibility of these techniques.

We have two options how to offload the SLAM algorithms from the devices: partially or fully. For example, CloudSLAM, proposed in [50], partitions different workflows of ORB-SLAM and as a result, tracking and local mapping are executed on the device (vechicle) and loop closure is executed on the edge. Similarly, authors of [51] proposes a functional split of the ORB-SLAM2 architecture between the edge and the mobile device. Their solution keeps the tracking computation on the device and moves the rest, i.e., local mapping and loop closure, to the edge. Both proposals are tightly coupled to ORB-SLAM. In addition, the modifications of the original algorithms are needed. On the other hand, our framework supports arbitrary off-the-shelf SLAM algorithms which can be incorporated into the system and combined during the operation. In [52], the impact of processing power of different edge cloud systems on odometry and map generation is analyzed in detail. In this scenarios, the device executes only tasks related to camera data processing. Authors of [53] address fully remote SLAMs and present a novel buffering method to mitigate the impact of data losses in unreliable networks. ORBBuf optimizes the performance of the SLAM module by discarding frames with the least impact on SLAM's quality.

2.4 Edge/Cloud Platforms and 5G/6G

Different cloud platforms with several relevant services are available today delivering the computing infrastructure for future XR applications. On the one hand, the three giants operating the leading public cloud platforms, i.e., Amazon Web Services [54], Google Cloud Platform [55], Microsoft Azure [56], provide services in a wide range. At the end of the day, developers and application providers can compose and run virtual machines or software containers implementing the business logic, while the burden

of operational tasks, including resource management, on demand resource scaling or fault management, is delegated to the cloud providers. On the other hand, open-source technologies are also available to establish private cloud or edge platforms provisioning "arbitrary" amount of virtual resources on demand For example, OpenStack [57] and Kubernetes [58] are de facto industry standards for virtual machine and container management, respectively. These platforms and services together with emerging network technologies, such as beyond 5G and 6G systems, enable a novel architecture supporting both cloud and edge based deployments of AR applications where the computation intensive functions can be offloaded from the devices.

3 Proposed Architecture

This section is devoted to the main goals driving our architecture design and to the details of the relevant components of the proposed system. In addition, the feasibility of the concept is also discussed.

3.1 Design Goals

We target a novel edge cloud based AR platform supporting the coordination of multiple users in a common geographical space. The user experience requires precise pose (position and orientation) information to be calculated for each AR device in real-time. Moreover, due to the joint space, the coordinate systems of different users/devices need to be synchronized continuously in order to display the virtual objects in the right place on each device during the whole game. To meet these requirements, we propose an edge cloud based solution, where the camera images and sensor data are streamed towards a dedicated coordination service. The SLAM function is offloaded from the device and arbitrary algorithms can be invoked in the remote environment (even in parallel). By these means, we can prolong the battery life of the AR devices. In order to enable the coordination among the users, the per-device coordinate systems have to be synchronized which requires to build and use a joint global map. This is a core task of the coordination service and the details are hidden from the users. For example, they cannot access images sent from other devices. Built-in local SLAM algorithms of current AR devices (e.g., ARKit [59], ARCore [60]) operate with proprietary map formats which cannot be shared among different devices. Making use of open-source SLAM libraries, we can provide cross-platform solutions. Another important goal of our design is the extensibility. As we stream raw data to the edge cloud system, other AR services can easily be added as distinct software components. Real-time detection and tracking of objects, users and gestures are important examples which can be implemented atop the platform.

3.2 Main Components

The architecture of the proposed system is depicted in Fig. 1. Currently, we assume mobile phones as AR devices but the concept is valid for AR glasses

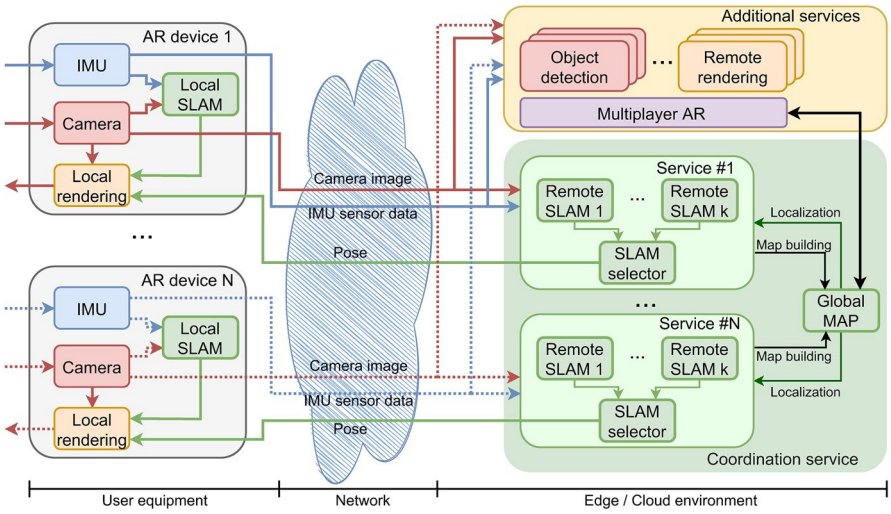

Fig. 1 Proposed architecture

as well. On the devices, camera images and IMU data are collected and sent to the coordination service. Since the user experience is greatly affected by the response time of the SLAM function, it is crucial to have a proper network connection between the devices and the edge cloud infrastructure. It is worth noting that local SLAM algorithms can also be run on the devices optionally. Following this scenario, for example, the faster but not coordinated local operation can be improved periodically based on the remote service.

At the cloud side, each device has its own service instance including one or multiple SLAM modules and a selector component. SLAM modules with different approaches can achieve different results regarding accuracy and robustness. Moreover, the performance is also affected by the surrounding environment and the user behavior in a diverse way. Therefore, we run several SLAM algorithms side by side, so that we can choose the best regarding mapping and tracking results. This is done by the SLAM selector module, which decides which algorithm to use based on the camera image data and the availability of pose estimations. Analyzing different SLAM algorithms in different environments, we can construct policies to choose the appropriate algorithm dynamically during the operation.

The core component of the coordination service is the global map which is constructed on-the-fly based on the received information from the connected devices. Making use of the global map, novel AR applications can be realized that are not possible with the current technologies. Furthermore, the platform can be extended with additional AR services (e.g., object detection and tracking) enriching the feature set provided to developers and customers.

3.3 Feasibility

The feasibility of the design concept is mainly determined by the capabilities of the underlying network infrastructure and the specific bandwidth and latency requirements of the proposed platform. On the one hand, the delay bound stems from the characteristics of the applications and the used AR devices. Assuming mobile (handheld) AR, the video content on the display is typically delayed with a few (e.g. three) frames which yields a well-defined, manageable upper bound. Head-mounted displays realizing video pass-through technology (similar concept to mobile AR) also allow a slight offset in projection but the head-pose cannot be delayed. In contrast, optical see-through devices pose more stringent latency limits and call for additional mechanisms, such as precise prediction on the movements. On the other hand, the bandwidth requirement of a client primarily depends on the parameters of the camera stream, such as the resolution, FPS and the encoding bitrate. In our framework, the targeted bitrate of the video encoder is a configuration parameter that controls the ultimate bandwidth usage. According to our preliminary analysis, HD video with 30 FPS frame rate encoded into 3.5 Mbps or 7 Mbps video streams result in similar and acceptable SLAM's accuracy. The architecture supports automatic rate adaptation as well which is an essential feature in bandwidth constrained radio network environments adjusting the targeted bitrate according to the varying network characteristics.

The available bandwidth of a cell in the Radio Access Network of a 5G (later 6G) network depends on multiple parameters, configuration settings and geographical properties. Assuming 3–7 Mbps upstream load per user, current systems can support several (but of course not hundreds of) AR clients which is a good starting point e.g. for multi-player gaming scenarios. But, we van expect much more uplink capacity from future radio networks [6]. We believe that besides cloud-based gaming scenarios, cloud-based multi-user AR applications will also be important ones in the "5G timescale" and will become a killer application during the era of 6G. Today's online gaming platforms typically require minimum 2–3 Mbps downlink and 0.5–2 Mbps uplink speeds, but with remote rendering the downstream easily jumps up to 10–15 Mbps. We argue that it is crucial to understand the specific requirements (uplink/downlink bandwidth, delay, jitter) of different AR related services and based on the assessments we can provide input for beyond 5G and 6G design and standardization activities.

4 Implementation

To validate our approach and the proposed architecture, we have created a proof-of-concept prototype including the relevant components. This section presents the details of the implemented system in two steps: first, the client side is discussed and second, the main parts of the platform are reviewed. The overall system is depicted in Fig. 2.

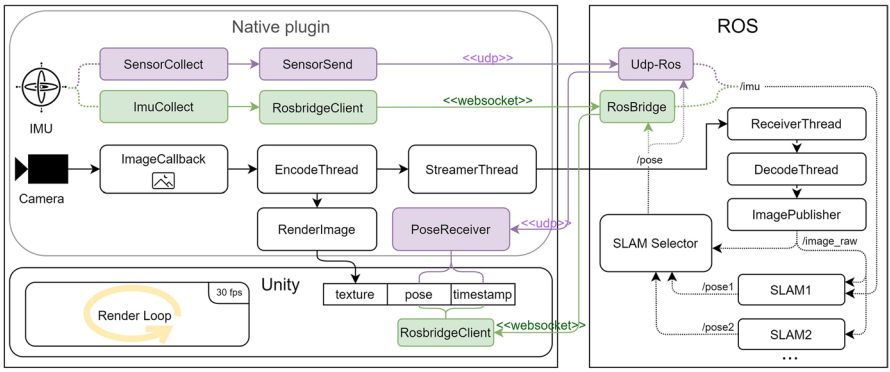

Fig. 2 Proof-of-concept prototype. Clients implemented on Android and iOS (left); Coordination platform (right)

4.1 Client Side: AR Application

We have implemented a multi-player AR application (game) making use of our edge cloud based coordination service. As it is shown in Fig. 2, the game is built on the Unity 3D engine [61], which is a widely used game engine for developing AR applications. In addition, we have implemented a native library in C++ *i*) to stream the camera images, *ii*) to stream the IMU data, and *iii*) to render the camera image to a Unity texture. On the one hand, this solution makes it easier to reuse our program code in other projects, and on the other hand, it is an efficient solution providing fast operation due to the C++ implementation. We have addressed two different mobile platforms, namely, Android and iOS platforms, with slightly different implementation details.

On the one hand, the streaming of the camera images is implemented in the same way on the two platforms. Invoking a listener, we get the images continuously in a callback method. The images are then passed to a H.264 encoder running in a separate thread. The targeted bitrate of the H.264 encoder is a configuration parameter, and in our experiments we used 3.5 Mbps and 7 Mbps, respectively. When encoding is complete, the original camera image is forwarded for rendering, and the encoded image is sent in a separate thread to the remote service. On the other hand, IMU data is sent differently on the two platforms. Both cases are shown in Fig. 2, where the components corresponding to the iOS implementation are indicated by purple boxes, while the Android related elements are shown by green boxes. In case of the Android implementation, a dedicated module, called Rosbridge [62], is used for data transmission over a websocket. In contrast, on iOS, the IMU data is sent in a raw UDP stream. (This difference stems from a dependency issue of Rosbridge.) We use the same client time for timestamping both the IMU data and video frames and they are synchronized at the server side. The IMU data is sent at a higher frequency (100 or 200 Hz in our experiments, depending on the device capability) and the image frame is combined with the closest one which has been received.

Our main focus was on an application which uses only the remote SLAM service for positioning, but we have also created a hybrid solution on both platforms. In this scenario, a local SLAM function also runs on the device, which is ARKit and ARCore, respectively, in our case, and the final pose information is calculated based on the local and the remote data. This solution requires the transformation between the local coordinate system and the one used by the global map in the platform. Following this approach, the positioning of the application is not fully lost during network errors because the local service can provide data continuously, and the coordination service can be used to improve the accuracy.

4.2 Platform Side: Coordination Service

Our coordination service has been implemented on top of the Robot Operating System (ROS) [63] and the components are packaged into distinct software containers. Building on ROS is beneficial for several reasons. First, most open source SLAM implementations run with ROS so these modules can be changed easily. Second, the topic based communication in ROS following the publish/subscribe model provides an efficient messaging bus where extra components, implementing additional AR features (e.g., object detection), can be connected to. The required input data, such as the image stream, or IMU can easily be broadcast to the involved entities. A dedicated stream receiver is in charge of receiving the UDP stream from the phone, decoding the camera images, and publishing it to the corresponding ROS topic. For each AR device, a new module is created which is responsible for receiving and sending the raw data and the calculated pose estimations and map. With ROS and Docker containers this can be managed in a flexible way. As we have made use of available open-source components, the global map is currently built on ROVIOLI and maplab [16], and the map is built based on explicitly merged dedicated missions. Later, additional algorithms and map handlers can be added.

As described in Sect. 3, multiple SLAM instances are running in parallel. Each module is subscribed to the incoming camera image messages and the IMU messages (if the latter is also required by the algorithm). In the current configuration, we have incorporated two different open-source SLAM libraries: ROVIOLI [16] and LSD-SLAM [17]. Additional implementations can easily be added later. The output of the SLAMs is monitored by the SLAM selector component and the "better" (or faster) pose response is selected for transmission towards the device. The coordinate systems are transformed accordingly. In our proof-of-concept prototype, a Deep Neural Network (DNN) model was constructed and trained with the publicly available EuRoC benchmark dataset [64] which includes the ground truth values as baselines. The model learned the accuracy of the selected SLAM modules for different environments and movement sequences and it is able to predict which algorithm will provide the more accurate pose value for the subsequent series of frames. The module is called SLAM Image Analyzer (SIA) and it is able to switch between SLAM outputs making use of different heuristics (e.g., it checks periodically the accuracy and changes if the predicted deviation exceeds a given threshold). For example, environments with varying light conditions or motion blur typically impact the accuracy of tracking, and particular techniques (e.g., feature-based

and direct SLAM algorithms) react differently which we can exploit during the online operation.

For the SIA module, different types of neural networks, prediction horizon parameters, and target error metrics were systematically examined for each SLAM implementation to find the best-performing model configuration and hyperparameters. The intrinsic problem of SLAM's accuracy prediction has been formulated as a regression that relies on a specific blur metric of received image frames combined with synchronized (closest received) IMU values as the main input. The Variance of Laplacian (VoL) [65] value is leveraged as the blur metric, which can be calculated quickly and it is efficient in characterizing the quality of an image frame regarding either the number of feasible feature points or the pixel intensity's distribution [66]. For the model's target, the relative pose error (RPE) has been chosen that is calculated for the range between poses of the current and one following frame according to the prediction horizon, i.e., for consecutive frames the model needs to look ahead and infer a prediction. This local error metric is suitable to our problem since it can be calculated for each frame in a flexible manner and does not depend on previously calculated error values as it is the case of cumulative metrics, such as the absolute trajectory error (ATE). Moreover, these predicted RPE values can be easily utilized to specify the magnitude of expected drifts suffered in the trajectories of the different SLAMs, despite the absence of ground truth data. Beside the standard deep neural network, recurrent neural models were also trained and evaluated to tackle the occurring hidden correlations between the pose errors of consecutive frames by taking the time factor into consideration. Two different recurrent structures were examined: a stacked Long Short-Term Memory (LSTM) [67] based model with two stacked layers and an encoder-decoder model [68] that applies an LSTM layer for both the encoding and decoding parts along with a single hidden layer for the output calculation.

For the model training and evaluation steps, prerecorded measurements of the EuRoC MH 01-04 missions were used with randomized shuffling and cross-validation, while the evo [69] tool was used for calculating the pose errors. Our automated training and tuning processes ultimately resulted that the simpler DNN model has superior performance over the tested recurrent models, using a 10-frame prediction horizon. The best result is provided by a two-layer DNN that we have incorporated into our prototype implementation. It consists of two hidden layers, each with 512 neurons and the rectified linear unit (ReLU) activation function. To avoid overfitting, L2 bias regularizers ($l2 = 0.2$) are defined for each layer along with a separate dropout layers with a 30 % drop rate. For the model training, the Adam optimizer [70] is used with reduced learning rate ($1e-4$) and the specific Huber loss [71] that can robustly handle target values with frequent changes by behaving quadratically for small residuals and linearly for large residuals.

5 Measurement and Evaluation Methodology

In regular AR application, the pose information is provided by the local SLAM (e.g., ARCore or ARKit) periodically at certain time instants, estimating how far the device moved from the position since the application started. The quality of

the user experience is significantly affected by the accuracy of this estimation. This holds for remote SLAM solutions, as well. In order to measure the accuracy of a SLAM algorithm, we need to know the real physical position for each time instant, which is called the *ground truth* in the literature [72–74]. This is a baseline trajectory for measuring the accuracy of a SLAM. There are some open-source databases [64, 75, 76] that contain the camera images for each timestamp and also include the ground truth for the datasets. However, this tool set cannot be used to characterize our own application in our physical environment including also the network infrastructure. Therefore, a novel measurement methodology is required to be able to analyze the impact of all components and mechanisms.

5.1 Baseline Trajectory

We have elaborated an appropriate measurement methodology which can leverage a wide range of baseline trajectories. We use a robotic arm to move the AR device on preliminary defined 3D trajectories. The real physical position is captured from the robot software and it can be used as a ground truth value.

This methodology is capable of providing high precision ground truth data series, however, the size of the trajectories is limited by the moving range of the robotic arm. In our measurements, we used an Universal Robot 5E robotic arm (shown in Fig. 3) with a mounted mobile device. Within the range, arbitrary trajectories can be configured with programmable velocity and acceleration. Recording the ground truth and the estimated positions provided by the SLAM algorithms (of course, both with timestamps), we can easily compare how close the estimated position is to reality.

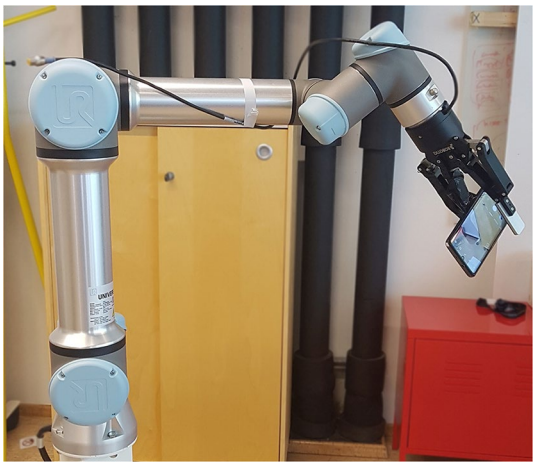

Fig. 3 Measurement setup with an UR5e robotic arm

5.2 Evaluation Method

A crucial step of the evaluation is the synchronization of the coordinate systems of the robotic arm and the AR device. SLAMs measure the distance and the rotation from the AR application's starting point, which is the origin of that coordinate system. In contrast, the ground truth is defined in the robotic arm's coordinate system where the origin is at the base of the robot stand. Furthermore, some SLAM algorithms use right-handed coordinate systems (e.g. ARCore), but others assume left-handed ones. Therefore, we need to apply shifting and matrix multiplication to synchronize the two worlds. The magnitude of the shifting is calculated in the starting position and applied throughout the measurement. The positions obtained in this way and the ground truth values are shown together in the plots in the next section allowing a visual comparison. Besides, we use objective metrics to characterize the targeted SLAM algorithms, such as Absolute Trajectory Error (ATE) and Relative Pose Error (RPE) [75, 77]. The RPE shows the drift of the trajectory over a fixed time interval and the ATE measures the global difference compared to the baseline trajectory.

Another important aspect which should be considered, especially in case of a remote SLAM, is the response time of the service. If the latency is too large, the provided pose is outdated and unusable. During the measurements, we put a timestamp on each image as soon as we get it from the camera of the mobile phone and we also put a timestamp when we get back the estimated pose for that image from the SLAM. Hence, we specifically measure end-to-end latency which includes image encoding, network latency, image decoding, SLAM runtime, and all platform overheads.

Besides the robotic tests, we addressed human motion experiments as well, where the trajectory is not limited. Inside our office building, we have accomplished several missions with different complexities. The recorded datasets can be replayed for the platform running with chosen configurations. Here, the exact ground truth is not available, however, the results of the local SLAM algorithms can be captured and used as an approximation for the baseline. This approach is validated by our measurements and ARCore and ARKit follow the real trajectories with acceptable precision.

6 Evaluation

This section is devoted to the evaluation of our proof-of-concept prototype following the methodology described in the previous section. In our experiments, the coordination service platform was established both in an AWS cloud region (Frankfurt, eu-central-1) and in the local premises (edge setup), while the robotic arm and the AR devices were operated from the local premises connected via a WiFi network. The average round-trip time between the device and the coordination platform was 26 ms and 2 ms, respectively. We targeted experiments where the network was not a bottleneck of the system and the number of clients was limited. Our testbed environment encompassed WiFi and wired networks connecting the AR devices to the edge/

cloud services, while the interference in the radio channels was minimized. By these means, we analyzed the limits of the approach and revealed the performance characteristics under unconstrained network capacity.

6.1 Latency Characteristics

We conduct experiments to measure the runtime of each component in order to analyze the usability of the system and to reveal potential bottlenecks. Both Android (Huawei P30 Pro) and iOS (iPhone 12 Pro) phones are tested with the coordination service. Detailed analysis has been carried out for a wide range of scenarios and as an illustration, a selected cloud setup and an edge scenario is presented here, respectively. In the chosen cloud experiment, the coordination service uses ROVIOLI as the SLAM module (shown in Fig. 4), while the edge scenario is equipped with LSD-SLAM (presented in Fig. 5). We use HD videos (Huawei P30 Pro: 1280x960, iPhone 12 Pro: 1280x720 resolution) with 30 FPS encoded into streams targeting 3.5 Mbps bitrate. Four separate operation phases are measured: the camera image encoding time and the streaming time on the phone, the processing time of the given SLAM module (including queueing delay if it appears), and the network delay together with all platform overheads (e.g., ROS based communication, virtualization overhead).

Experiments have shown that the streaming time of a frame on a phone is most significantly affected by the phone's encoder. In case of Android, the transcoding time of a frame (1280x960 pixels) takes around 72 ms yielding an average total transmission time of 76 ms (including the encoder and streamer threads). This large encoding time cannot be reduced due to the limitation of the available libraries (and the underlying drivers) which do not allow to enforce individual frame processing (instead three subsequent frames are handled together). For iOS devices, the encoding function shows much better performance due to per-frame processing and it needs around 7.27 ms to encode an image (1280x720

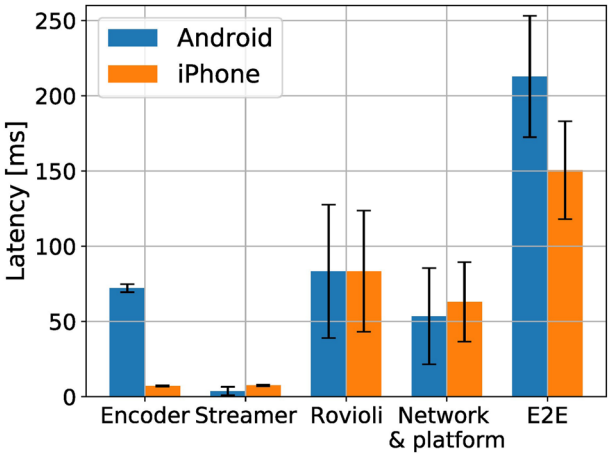

Fig. 4 Performance of the coordination service with Rovioli running in AWS

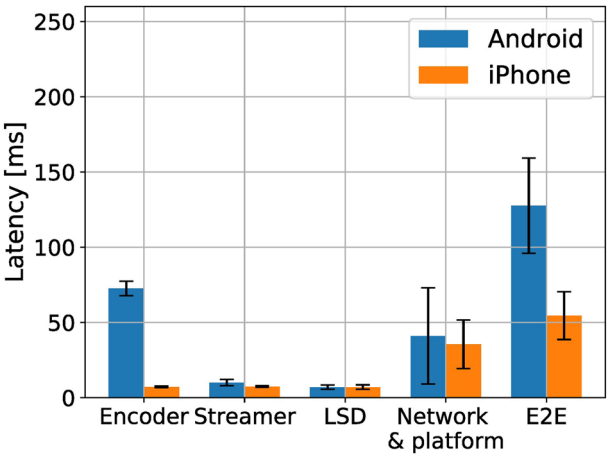

Fig. 5 Performance of the coordination service with LSD running at the edge site

pixels). Approximately, additional 7.5 ms is required by the streamer thread which yields around 15 ms as the total streaming time. The response time of the SLAM modules show the same characteristics for both devices. The average runtime of ROVIOLI, including the queuing delays, is above 80 ms with large variance, while LSD-SLAM exhibits much better performance calculating the pose within 7.5 ms in all scenarios. The network delays and also the overheads are different for the cloud and edge scenarios depending on the underlying hardware architecture, the used network technologies and the physical distance between the platform and the AR devices. The average delay is tolerable for our test scenarios, however the results show significant jitter which can have impact on the user experience. We expect that moving to 5G (and 6G) environment will result in reduced jitter and more deterministic delays in the network part. However, in order to mitigate the jitter introduced by the edge/cloud platforms, additional mechanisms are needed.

We also present the total end-to-end (E2E) latency in Figs. 4) and 5) which measures the time between grabbing the camera image and the arrival of the pose. Besides, Fig. 6 summarizes the E2E statistics for all scenarios in a violin plot. On the one hand, in case of Android, the E2E delay is above 200 ms with ROVIOLI even in edge scenarios, which unfortunately has a noticeable effect on the user experience. With LSD-SLAM, a slightly better operation is achieved (above 100 ms but with large deviation) which also has an impact on the experience. On the other hand, the results with iOS (iPhone 12 Pro) are very promising, especially together with LSD-SLAM. In case of the edge setup, the average E2E latency is kept around 50 ms which means that the pose information can be updated within 2-frame delay for most of the frames (assuming 30 FPS frame rate).

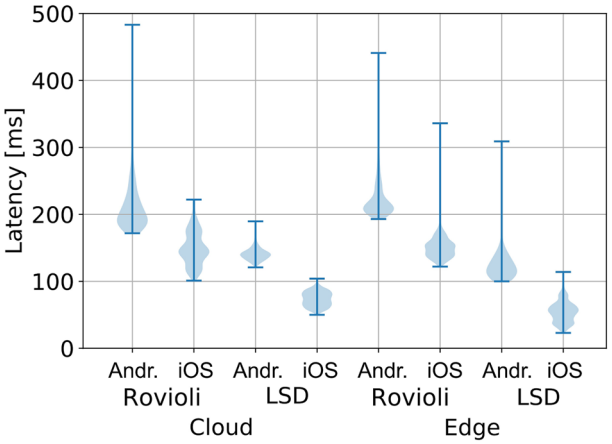

Fig. 6 End-to-end latency statistics

6.2 Accuracy

Besides the strict latency characteristics which is essential for the real-time operation, accuracy is the other dimension which directly impacts the user experience. The accuracy of our coordination service is affected by the incorporated SLAM libraries and also the distributed operation of the overall system. We have conducted several experiments to assess the performance characteristics, here we highlight some relevant results. The results of the robotic accuracy measurements are shown in Fig. 7 and Fig. 8 for two different trajectories. The first one is a simple one lifting

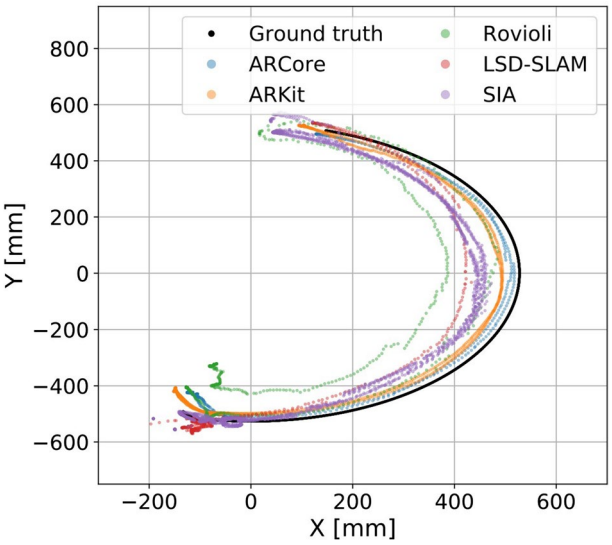

Fig. 7 The performance of different SLAMs using simple trajectory (top view)

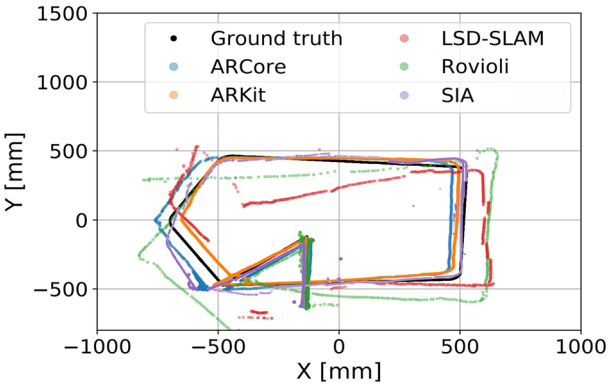

Fig. 8 The performance of different SLAMs using complex trajectory (top view)

the AR device and following an arc there and back. In Fig. 7, the top view of the trajectory is plotted presenting the results for different SLAMs. The black curve indicates the ground truth, and as baselines, the results of ARCore and ARKit are also depicted. Besides single SLAMs, we show the results of our ensemble module (SIA) combining ROVIOLI and LSD-SLAM. ARCore and ARKit outperform the open-source methods, however, the errors are not extreme and in certain applications, this performance can be accepted (position coordinates are in millimeters).

The second experiment applies a complex trajectory, where the robotic arm makes several changes of direction and draws the shape of a house after a synchronization phase (bottom center part of Fig. 8 which shows the top view of the movements). Here, ARCore and ARKit strictly follow the ground truth values but the accuracy of the remote SLAMs shows non-negligible divergence. They follow straight movements very well, but the turning angles are often incorrect. Although sometimes the remote SLAMs make corrections and these cause some minor jumps in the estimated positions. Moreover, at the corners of the "house", the altitude values also show deviation.

The third type of experiments addresses human movements. The result of a selected mission using ROVIOLI is shown in Fig. 9. Here, we use the trajectory of ARCore or ARKit as the ground truth. For this complex human motion, the accuracy of the remote service is very close to the performance of the local solutions and it can provide acceptable user experience. As quantitative metrics, ATE and RPE values are also calculated and shown in Table 1 for selected representative experiments from each test group. LSD-SLAM shows better performance than ROVIOLI, while the ensemble solution is able to outperform both SLAMs in terms of the absolute error (ATE). However, the relative drift (RPE) indicates performance issues for SIA which can be the result of sudden changes in pose values at switching events. This is an important future research to identify the trade-off between sudden corrections vs. slowly accumulated drifts when we optimize for the experience.

To sum up, the revealed and experienced accuracy is acceptable for simple games, however it is currently not enough for critical applications, such as a remote

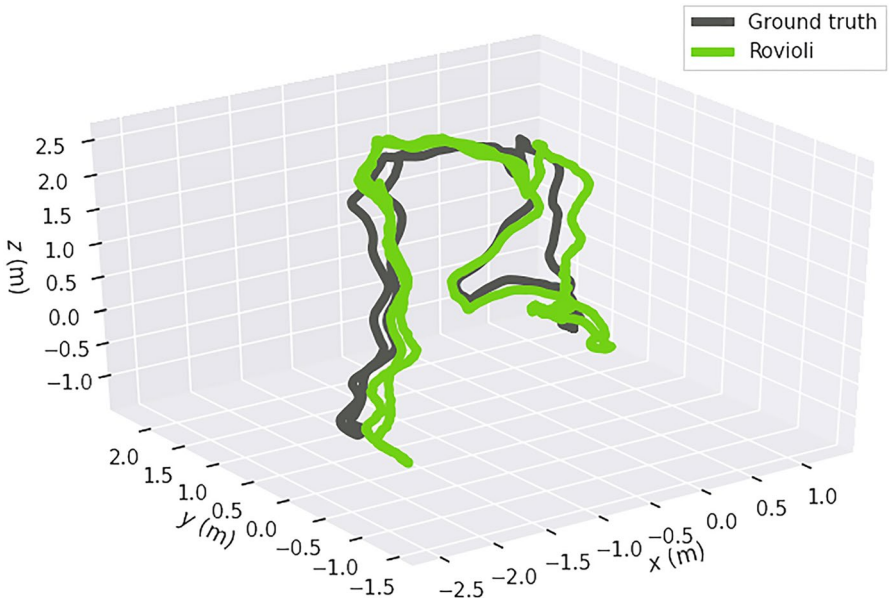

Fig. 9 The performance using ROVIOLI in a human motion experiment inside the office building

Table 1 ATE and RPE for different SLAMs [m]

		ARCore	ARKit	Rovioli	LSD	SIA
ATE	Rob. 1	0.104	0.104	0.198	0.220	0.198
	Rob. 2	0.109	0.102	0.368	0.360	0.355
	Hum	–	–	0.430	0.393	0.372
RPE	Rob. 1	0.008	0.007	0.013	0.014	0.012
	Rob. 2	0.004	0.004	0.012	0.009	0.016
	Hum	–	–	0.017	0.016	0.021

surgery support system. If we target high-quality AR experiences, the coordination service has to be combined with local enhancements (e.g., sophisticated predictions or filtering mechanisms). But we think that the platform can be extended towards this direction.

6.3 Energy Consumption

It is also important to consider the power consumption for client devices used in the system because as already mentioned, the battery life of a mobile AR device can be multiplied by offloading certain XR functions [6]. We have conducted several experiments to compare the consumption of a baseline application invoking local AR libraries with a novel application using our coordination service. The measured energy consumption of the device while running the application for 15 minutes,

Table 2 Energy consumption for different SLAMs

iPhone 12 Pro	ARKit	223 mAh
	Remote SLAM	159 mAh
Huawei P30 Pro	ARCore	252 mAh
	Remote SLAM	248 mAh

respectively, is shown in Table 2. In case of iPhone 12 Pro, the *energy consumption is reduced with 30%* when the coordination service is invoked, which is an important advantage of this approach. However, in case of Android platform, the results do not show that gain. This stems from an implementation issue; currently our rendering codes are not optimized and some tasks are implemented in CPU instead of GPU.

7 Conclusion

In this paper, we have proposed a novel edge cloud based coordination platform for multi-user AR applications. An extensible architecture has been described and a proof-of-concept prototype were developed. We have focused on the latency sensitive and computation intensive Simultaneous Localization And Mapping function which was offloaded from the device to the edge cloud infrastructure. Our solution has been built on open-source SLAM libraries and the Robot Operating System (ROS) and it inherently supports further extensions, including additional AR services, such as object/face detection and tracking, semantic understanding of the environment or remote rendering. In order to be able to evaluate the concept, we have defined a dedicated measurement methodology providing the necessary ground truth information. Following the methodology, we analyzed the latency characteristics and the accuracy of the platform via real experiments. The current version of the proof-of-concept prototype provides sufficient services for simple AR games or applications, however further improvements and additional components are required to enable high-quality AR experiences.

Challenging future works have also been identified. We are working on an extension to the encoder/streamer part which supports automatic rate adaptation. It is crucial in bandwidth-constrained radio network environments to be able to adjust the targeted bitrate of the encoder dynamically to follow the varying characteristics of the radio network. Besides, the trade-off between the encoder's bitrate and SLAM's accuracy for different SLAM implementations is also an interesting research question. Our future work addresses the migration to an internal experimental 5G testbed where the full system is under our control and dedicated experiments with our platform can be carried out. And finally, we target performance enhancements in two parallel tracks. On the one hand, more recent SLAM libraries (e.g., ORB-SLAM3), will be added to the framework. On the other hand, novel mechanisms, such as precise movement prediction and filtering algorithms will be integrated into the platform in order to mitigate delay and jitter issues.

Acknowledgements This work was supported by the ÚNKP-23-5-BME-435 New National Excellence Program of the Ministry of Innovation and Technology of Hungary from the National Research, Development and Innovation Fund through projects no. 135074 under the FK_20 funding scheme and 2021−1.2.4-TÉT-2021-00058 under the 2021−1.2.4-TÉT funding scheme. B. Sonkoly was supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences.

Funding Open access funding provided by Budapest University of Technology and Economics.

References

1. Thomas, B.H.: A survey of visual, mixed, and augmented reality gaming. Comput. Entertain. **10**(1), 1–33 (2012)
2. Masood, T., Egger, J.: Augmented reality in support of Industry 4.0-implementation challenges and success factors. Robot. Comput. Integr. Manuf. **58**, 181–195 (2019). https://doi.org/10.1016/j.rcim.2019.02.003
3. Al-Hiyari, N.N., Jusoh, S.S.: Healthcare Training Application: 3D First Aid Virtual Reality. In: International Conference on Data Science, E-Learning and Information Systems 2021. DATA'21, pp. 107–116 (2021). https://doi.org/10.1145/3460620.3460741
4. Chi, H.-L., Kang, S.-C., Wang, X.: Research trends and opportunities of augmented reality applications in architecture, engineering, and construction. Automat. Construct. **33**, 116–122 (2013). https://doi.org/10.1016/j.autcon.2012.12.017
5. Quandt, M., Knoke, B., Gorldt, C., Freitag, M., Thoben, K.-D.: General requirements for industrial augmented reality applications. Proc. CIRP **72**, 1130–1135 (2018). https://doi.org/10.1016/j.procir.2018.03.061
6. Alriksson, F., Phillips, C., Pradas, J.L., Zaidi, A., et al.: Xr and 5g: extended reality at scale with time-critical communication. Ericsson Technol. Rev. **2021**(8), 2–13 (2021)
7. Sonkoly, B., Nagy, B.G., Dóka, J., Kecskés-Solymosi, Z., Czentye, J., Formanek, B., Jocha, D., Gerő, B.P.: Towards an edge cloud based coordination platform for multi-user ar applications built on open-source slams. In: NOMS 2023–2023 IEEE/IFIP Network Operations and Management Symposium, pp. 1–6 (2023)
8. Bresson, G., Alsayed, Z., Yu, L., Glaser, S.: Simultaneous localization and mapping: a survey of current trends in autonomous driving. IEEE Trans. Intell. Vehicles **2**(3), 194–220 (2017). https://doi.org/10.1109/tiv.2017.2749181
9. Jinyu, L., Bangbang, Y., Danpeng, C., Nan, W., Guofeng, Z., Hujun, B.: Survey and evaluation of monocular visual-inertial SLAM algorithms for augmented reality. Virtual Reality Intell. Hardw. **1**(4), 386–410 (2019). https://doi.org/10.1016/j.vrih.2019.07.002
10. Endres, F., Hess, J., Engelhard, N., Sturm, J., Cremers, D., Burgard, W.: An Evaluation of the RGB-D SLAM Systems. In: 2012 IEEE International Conference on Robotics and Automation (2012). https://doi.org/10.1109/icra.2012.6225199
11. Zhang, S., Zheng, L., Tao, W.: Survey and evaluation of RGB-D SLAM. IEEE Access **9**, 21367–21387 (2021). https://doi.org/10.1109/ACCESS.2021.3053188
12. Servières, M., Renaudin, V., Dupuis, A., Antigny, N.: Visual and visual-inertial SLAM: state of the art, classification, and experimental benchmarking. J. Sens. **2021**, 1–26 (2021). https://doi.org/10.1155/2021/2054828

13. Taketomi, T., Uchiyama, H., Ikeda, S.: Visual SLAM algorithms: a survey from 2010 to 2016. IPSJ Trans. Comput. Vis. Appl. **9**, 1 (2017)
14. 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). Proc. Comput. Sci. **76**, 174–179 (2015). https://doi.org/10.1016/j.procs.2015.12.336
15. Li, R., Wang, S., Gu, D.: Ongoing evolution of visual SLAM from geometry to deep learning: challenges and opportunities. Cogn. Comput. **10**, 1–15 (2018). https://doi.org/10.1007/s12559-018-9591-8
16. Schneider, T., Dymczyk, M., Fehr, M., Egger, K., Lynen, S., Gilitschenski, I., Siegwart, R.: Maplab: an open framework for research in visual-inertial mapping and localization. IEEE Robot. Automat. Lett. (2018). https://doi.org/10.1109/LRA.2018.2800113
17. Engel, J., Schöps, T., Cremers, D.: LSD-SLAM: Large-Scale Direct Monocular SLAM. In: Computer Vision—ECCV 2014, pp. 834–849 (2014). https://doi.org/10.1007/978-3-319-10605-2_54
18. Younes, G., Asmar, D., Shammas, E., Zelek, J.: Keyframe-based monocular SLAM: design, survey, and future directions. Robot. Autonom. Syst. **98**, 67–88 (2017). https://doi.org/10.1016/j.robot.2017.09.010
19. Rosinol, A., Abate, M., Chang, Y., Carlone, L.: Kimera: an Open-Source Library for Real-Time Metric-Semantic Localization and Mapping. In: 2020 IEEE International Conference on Robotics and Automation (ICRA), pp. 1689–1696 (2020). IEEE
20. Campos, C., Elvira, R., Rodríguez, J.J.G., Montiel, J.M., Tardós, J.: ORB-SLAM3: an accurate open-source library for visual, visual-inertial, and multimap SLAM. IEEE Trans. Robot. (2021). https://doi.org/10.1109/TRO.2021.3075644
21. Krombach, N., Droeschel, D., Behnke, S.: Combining Feature-Based and Direct Methods for Semi-Dense Real-Time Stereo Visual Odometry. In: Intelligent Autonomous Systems 14, pp. 855–868 (2017)
22. Ait-Jellal, R., Zell, A.: Outdoor Obstacle Avoidance Based on Hybrid Visual Stereo SLAM for an Autonomous Quadrotor MAV. In: 2017 European Conference on Mobile Robots (ECMR) (2017)
23. Younes, G., Asmar, D., Zelek, J.: FDMO: Feature Assisted Direct Monocular Odometry. In: Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (2019)
24. Hess, W., Kohler, D., Rapp, H., Andor, D.: Real-Time Loop Closure in 2D LIDAR SLAM. In: 2016 IEEE International Conference on Robotics and Automation (ICRA), pp. 1271–1278 (2016)
25. Bruno, H.M.S., Colombini, E.L.: LIFT-SLAM: a deep-learning feature-based monocular visual SLAM method. Neurocomputing **455**, 97–110 (2021)
26. Schonberger, J.L., Pollefeys, M., Geiger, A., Sattler, T.: Semantic Visual Localization. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (2018). https://doi.org/10.1109/cvpr.2018.00721
27. Izquierdo-Domenech, J., Linares-Pellicer, J., Orta-Lopez, J.: Supporting interaction in augmented reality assisted industrial processes using a CNN-based semantic layer. In: 2020 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR) (2020)
28. Yu, C., Liu, Z., Liu, X.-J., Xie, F., Yang, Y., Wei, Q., Fei, Q.: DS-SLAM: A Semantic Visual SLAM Towards Dynamic Environments. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2018)
29. Li, R., Wang, S., Gu, D.: DeepSLAM: a robust monocular slam system with unsupervised deep learning. IEEE Trans. Ind. Electron. **68**(4), 3577–3587 (2021). https://doi.org/10.1109/TIE.2020.2982096
30. Stenborg, E., Toft, C., Hammarstrand, L.: Long-term visual localization using semantically segmented images. In: 2018 IEEE International Conference on Robotics and Automation ICRA (2018). https://doi.org/10.1109/icra.2018.8463150
31. Sereno, M., Wang, X., Besancon, L., Mcguffin, M.J., Isenberg, T.: Collaborative work in augmented reality: a survey. IEEE Trans Visual Comput Graph **1**, 1–1 (2020). https://doi.org/10.1109/TVCG.2020.3032761
32. Mahmood, T., Fulmer, W., Mungoli, N., Huang, J., Lu, A.: Improving Information Sharing and Collaborative Analysis for Remote GeoSpatial Visualization Using Mixed Reality. In: 2019 IEEE International Symposium on Mixed and Augmented Reality (ISMAR), pp. 236–247 (2019). https://doi.org/10.1109/ISMAR.2019.00021

33. Vidal-Balea, A., Blanco-Novoa, O., Fraga-Lamas, P., Vilar-Montesinos, M., Fernández-Caramés, T.M.: Creating Collaborative Augmented Reality Experiences for Industry 4.0 Training and Assistance Applications: Performance Evaluation in the Shipyard of the Future. Applied Sciences **10**(24) (2020) https://doi.org/10.3390/app10249073
34. Platinsky, L., Szabados, M., Hlasek, F., Hemsley, R., Pero, L.D., Pancik, A., Baum, B., Grimmett, H., Ondruska, P.: Collaborative Augmented Reality on Smartphones via Life-long City-scale Maps. In: 2020 IEEE International Symposium on Mixed and Augmented Reality (ISMAR), pp. 533–541 (2020). https://doi.org/10.1109/ISMAR50242.2020.00081
35. Zou, D., Tan, P., Yu, W.: Collaborative visual SLAM for multiple agents: a brief survey. Virtual Reality Intell. Hardw. **1**(5), 461–482 (2019). https://doi.org/10.1016/j.vrih.2019.09.002
36. Egodagamage, R., Tuceryan, M.: A Collaborative Augmented Reality Framework Based on Distributed Visual Slam. In: 2017 International Conference on Cyberworlds (CW), pp. 25–32 (2017). https://doi.org/10.1109/CW.2017.47
37. Egodagamage, R., Tuceryan, M.: Distributed monocular visual SLAM as a basis for a collaborative augmented reality framework. Comput. Graph. **71**, 113–123 (2018). https://doi.org/10.1016/j.cag.2018.01.002
38. Schmuck, P., Ziegler, T., Karrer, M., Perraudin, J., Chli, M.: Covins: Visual-inertial slam for centralized collaboration. In: 2021 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct), pp. 171–176. IEEE Computer Society, Los Alamitos, CA, USA (2021). https://doi.org/10.1109/ISMAR-Adjunct54149.2021.00043 . https://doi.ieeecomputersociety.org/10.1109/ISMAR-Adjunct54149.2021.00043
39. Ouyang, M., Shi, X., Wang, Y., Tian, Y., Shen, Y., Wang, D., Wang, P., Cao, Z.: A Collaborative Visual SLAM Framework for Service Robots. In: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 8679–8685 (2021). https://doi.org/10.1109/IROS51168.2021.9636798
40. Siriwardhana, Y., Porambage, P., Liyanage, M., Ylianttila, M.: A survey on mobile augmented reality with 5G mobile edge computing: architectures, applications, and technical aspects. IEEE Commun. Surv. Tutor. **23**(2), 1160–1192 (2021). https://doi.org/10.1109/COMST.2021.3061981
41. Qiao, X., Ren, P., Dustdar, S., Liu, L., Ma, H., Chen, J.: Web AR: a promising future for mobile augmented reality - state of the art, challenges, and insights. Proc. IEEE **107**(4), 651–666 (2019). https://doi.org/10.1109/JPROC.2019.2895105
42. Dey, S., Mukherjee, A.: Robotic SLAM. In: Adjunct Proceedings of the 13th International Conference on Mobile and Ubiquitous Systems: Computing Networking and Services (2016). https://doi.org/10.1145/3004010.3004032
43. Hu, P., Dhelim, S., Ning, H., Qiu, T.: Survey on fog computing: architecture, key technologies, applications and open issues. J. Netw. Comput. Appl. **98**, 27–42 (2017)
44. Xu, J., Cao, H., Li, D., Huang, K., Qian, C., Shangguan, L., Yang, Z.: Edge Assisted Mobile Semantic Visual SLAM. In: IEEE INFOCOM 2020—IEEE Conference on Computer Communications (2020). https://doi.org/10.1109/infocom41043.2020.9155438
45. Benavidez, P., Muppidi, M., Rad, P., Prevost, J.J., Jamshidi, M., Brown, L.: Cloud-Based Realtime Robotic Visual SLAM. (2015). https://doi.org/10.1109/syscon.2015.7116844. ieeexplore.ieee.org/document/7116844
46. Opdenbosch, D.V., Oelsch, M., Garcea, A., Aykut, T., Steinbach, E.: Selection and Compression of Local Binary Features for Remote Visual SLAM. In: 2018 IEEE International Conference on Robotics and Automation (ICRA) (2018). https://doi.org/10.1109/icra.2018.8463202
47. Karrer, M., Schmuck, P., Chli, M.: CVI-SLAM—collaborative visual-inertial SLAM. IEEE Robot. Automat. Lett. **3**(4), 2762–2769 (2018). https://doi.org/10.1109/lra.2018.2837226
48. NVIDIA CloudXR. https://developer.nvidia.com/nvidia-cloudxr-sdk. Accessed 29 Nov 2023
49. Rambach, J., Pagani, A., Schneider, M., Artemenko, O., Stricker, D.: 6DoF object tracking based on 3d scans for augmented reality remote live support. Computers (2018). https://doi.org/10.3390/computers7010006
50. Wright, K.-L., Sivakumar, A., Steenkiste, P., Yu, B., Bai, F.: Cloudslam: Edge offloading of stateful vehicular applications. In: 2020 IEEE/ACM Symposium on Edge Computing (SEC), pp. 139–151 (2020). https://doi.org/10.1109/SEC50012.2020.00018

51. Ali, A.J.B., Hashemifar, Z.S., Dantu, K.: Edge-SLAM: Edge-Assisted Visual Simultaneous Localization and Mapping. In: MobiCom '20: The 26th Annual International Conference on Mobile Computing and Networking (2020). https://doi.org/10.1145/3372224.3417326
52. Sossalla, P., Rischke, J., Hofer, J., Fitzek, F.H.P.: Evaluating the Advantages of Remote SLAM on an Edge Cloud. (2021). https://doi.org/10.1109/etfa45728.2021.9613415.ieeexplore.ieee.org/document/9613415
53. Wang, Y.-P., Zou, Z.-X., Wang, C., Dong, Y.-J., Qiao, L., Manocha, D.: ORBBuf: A Robust Buffering Method for Remote Visual SLAM. (2021). https://doi.org/10.1109/iros51168.2021.9635950 . https://arxiv.org/abs/2010.14861
54. Amazon Web Services. https://aws.amazon.com. Accessed 23 Nov 2023
55. Google Cloud Platform. https://cloud.google.com. Accessed 23 Nov 2023
56. Microsoft Azure. https://azure.microsoft.com. Accessed 23 Nov 2023
57. OpenStack: The Most Widely Deployed Open Source Cloud Software in the World. https://www.openstack.org. Accessed 29 Nov 2023
58. Kubernetes: Automated Container Deployment. https://kubernetes.io. Accessed 29 Nov 2023
59. ARKit. https://developer.apple.com/augmented-reality/arkit/. Accessed 29 Nov 2023
60. ARCore. https://arvr.google.com/arcore/. Accessed 29 Nov 2023
61. Unity 3D. https://unity.com/. Accessed 29 Nov 2023
62. Crick, C., Jay, G., Osentoski, S., Pitzer, B., Jenkins, O.C.: Rosbridge: ROS for Non-ROS Users. In: Robotics Research, pp. 493–504 (2017). https://doi.org/10.1007/978-3-319-29363-9_28
63. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y., et al.: ROS: An Open-Source Robot Operating System. In: ICRA Workshop on Open Source Software (2009)
64. 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. **35**(10), 1157–1163 (2016). https://doi.org/10.1177/0278364915620033
65. Pech-Pacheco, J.L., Cristobal, G., Chamorro-Martinez, J., Fernandez-Valdivia, J.: Diatom autofocusing in brightfield microscopy: a comparative study. In: Proceedings 15th International Conference on Pattern Recognition. ICPR-2000, vol. 3, pp. 314–317 (2000). https://doi.org/10.1109/ICPR.2000.903548
66. Czentye, J., Gerö, B.P., Sonkoly, B.: Managing Localization Delay for Cloud-Assisted AR Applications Via LSTM-Driven Overload Control. In: 2021 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR), pp. 92–101 (2021). https://doi.org/10.1109/AIVR52153.2021.00023
67. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735
68. Wang, Z., Su, X., Ding, Z.: Long-term traffic prediction based on LSTM encoder-decoder architecture. IEEE Trans. Intell. Transp. Syst. **22**(10), 6561–6571 (2021). https://doi.org/10.1109/tits.2020.2995546
69. Grupp, M.: evo: Python package for the evaluation of odometry and SLAM. https://github.com/MichaelGrupp/evo. Accessed 24 May 2022
70. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization (2014). https://doi.org/10.48550/ARXIV.1412.6980
71. Huber, P.J.: Robust estimation of a location parameter. Ann. Math. Stat. **35**(1), 73–101 (1964). https://doi.org/10.1214/aoms/1177703732
72. Filatov, A., Filatov, A., Krinkin, K., Chen, B., Molodan, D.: 2D SLAM Quality Evaluation Methods. In: 2017 21st Conference of Open Innovations Association (FRUCT), pp. 120–126 (2017). IEEE
73. Nardi, L., Bodin, B., Zia, Z., Mawer, J., Nisbet, A., Kelly, P., Davison, A., Luján, M., O'Boyle, M., Riley, G., Topham, N., Furber, S.: Introducing SLAMBench, a Performance and Accuracy Benchmarking Methodology for SLAM, vol. 2015 (2014). https://doi.org/10.1109/ICRA.2015.7140009
74. Muller, M., Surmann, H., Pervolz, K., May, S.: The Accuracy of 6D SLAM using the AIS 3D Laser Scanner. In: 2006 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, pp. 389–394 (2006). https://doi.org/10.1109/MFI.2006.265647
75. Sturm, J., Engelhard, N., Endres, F., Burgard, W., Cremers, D.: A Benchmark for the Evaluation of RGB-D SLAM Systems. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 573–580 (2012). https://doi.org/10.1109/IROS.2012.6385773

76. Li, W., Saeedi, S., McCormac, J., Clark, R., Tzoumanikas, D., Ye, Q., Huang, Y., Tang, R., Leutenegger, S.: InteriorNet: Mega-scale Multi-sensor Photo-Realistic Indoor Scenes Dataset. In: British Machine Vision Conference (BMVC) (2018)
77. Prokhorov, D., Zhukov, D., Barinova, O., Anton, K., Vorontsova, A.: Measuring Robustness of Visual SLAM. In: 2019 16th International Conference on Machine Vision Applications (MVA)