ShareAR: Communication-Efficient Multi-User Mobile Augmented Reality

Xukan Ran
University of California, Riverside
xran001@ucr.edu

Carter Slocum
University of California, Riverside
csloc001@ucr.edu

Maria Gorlatova
Duke University
maria.gorlatova@duke.edu

Jiasi Chen
University of California, Riverside
jiasi@cs.ucr.edu

ABSTRACT

Augmented reality is an emerging application on mobile devices. However, there is a lack of understanding of the communication requirements and challenges of multi-user AR scenarios. In this position paper, we propose several important research issues that need to be addressed for low-latency, accurate shared AR experiences: (a) Systems tradeoffs of AR communication architectures used today in mobile AR platforms; (b) Understanding AR communication patterns and adapting the AR application layer to dynamically changing network conditions; and (c) Tools and methodologies to evaluate AR quality of experience in real time on mobile devices. We present preliminary measurements of off-the-shelf mobile AR platforms as well as results from our AR system, ShareAR, illustrating performance tradeoffs and indicating promising new research directions.

CCS CONCEPTS

• **Networks**; • **Human-centered computing** → **Ubiquitous and mobile computing**;

KEYWORDS

Mobile Augmented Reality, Communication Efficiency, SLAM

ACM Reference Format:
Xukan Ran, Carter Slocum, Maria Gorlatova, and Jiasi Chen. 2019. ShareAR: Communication-Efficient Multi-User Mobile Augmented Reality. In *The 18th ACM Workshop on Hot Topics in Networks (HotNets '19), November 13–15, 2019, Princeton, NJ, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3365609.3365867

1 INTRODUCTION

Augmented and virtual reality (AR/VR) are forecast to be the next frontier of mobile devices and open up a $692 billion market by 2025 [8]. In AR, a user's perception of the world is augmented by overlaying virtual objects onto the real world. These virtual objects provide relevant information to the user and remain fixed with respect to the real world, creating the illusion of seamless integration. Widely used examples of AR apps today include Pokemon Go, Google Translate, and Snapchat face filters. However, these apps mainly focus on single-user experiences, and lack the ability for multiple, co-located users to see and interact with the same virtual objects in real-time.

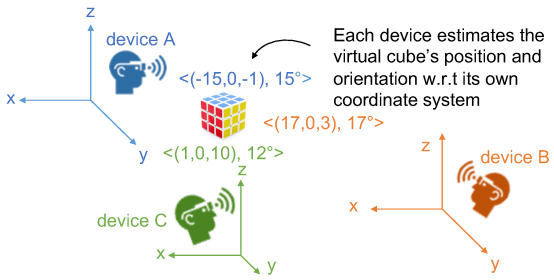


Figure 1: AR devices try to ensure consistent views of the virtual object despite different reference coordinate systems.

In this position paper, we explore several research directions to enable shared multi-user AR experiences. Multi-user AR will enable a new class of AR applications where users can interact with a common set of virtual elements to perform a common task or enjoy a common experience. For example, we envision multiple students wearing AR headsets in a classroom being able to see the same virtual chemistry molecule floating on table and manipulate it, with the virtual molecule remaining consistent across users. However, uncoordinated or laggy updates to the virtual molecule would break the illusion of seamless integration with the real world, and result in artifacts such as other users appearing to touch non-existent parts of the virtual molecule.

There are several challenges in realizing smooth communications between multiple AR users. Firstly, communication characteristics of existing mobile AR platforms are *under-explored*: there is a lack of understanding of what, how often, and to whom AR data transmissions are sent, and their impact on the user's AR-specific quality-of-experience. Secondly, AR involves *complex computation*: AR devices typically

HotNets '19, November 13–15, 2019, Princeton, NJ, USA

ACM ISBN 978-1-4503-7020-2/19/11…$15.00
https://doi.org/10.1145/3365609.3365867

utilize simultaneous localization and mapping (SLAM) and object detection techniques, which are initially computed on the AR device (or with the aid of edge/cloud), disseminated to other AR devices, and finally jointly processed to produce the final computation result. Thirdly, AR devices can have *differing fields-of-view*: the AR devices need to render the virtual objects at the correct locations and orientations in their current fields-of-view (FoVs), while maintaining a common understanding of the real and virtual worlds of other AR devices. Fourth, users *manipulate the virtual objects independently*: the changes made by one user to a virtual object cannot be directly observed by other devices (*e.g.,* through their cameras) because these manipulations do not act on real-world objects, and instead need to be explicitly communicated to ensure consistency between the users' displays.

The contribution of this paper is a research agenda highlighting several key components to support networked, multi-user AR applications in the near future:

- *Who to send to?* We examine common communication architectures used by Google ARCore and Apple ARKit, two mobile AR platforms, and find they use either cloud-based or peer-to-peer architectures. To the best of our knowledge, we are the first to clearly illustrate the flow of information exchange in multi-user AR. Since these platforms are closed source, we cannot perform an "apples to apples" comparison of the performance tradeoffs (*e.g.,* scalability, latency). Therefore, we propose extending an existing open-source single-user AR system [26] with multi-user functionality (which we call ShareAR) and use it to measure the performance tradeoffs, which can help provide guidelines on which architecture to use for a given AR application, environmental context, and network conditions (§3.1).
- *What and how often to send?* We collected network traces of several existing multi-user AR apps, and found that AR data transmissions consist of relatively large map data (more than 20 Mb) along with smaller chunks of user interaction data. We propose methods to adapt the AR application layer data to time-varying network conditions, while maintaining good AR quality. Our key idea is to tune the available "control knobs" of the AR application data (*e.g.,* number of keyframe features, number of transmissions, point cloud spatial granularity), taking into account their tradeoffs with AR quality, and choose the right configuration (§3.2).
- *How to measure multi-user AR quality?* Tools to measure AR quality are needed in order to quantitatively evaluate any proposed modification to multi-user AR systems. AR quality is defined in terms of the accuracy, jitter, and drift of a virtual object's position and orientation (§2.2). However, computing a virtual object's position and orientation error is challenging because there is no common "ground truth" point of reference (there exist only relative observations made by each device). We propose an AR quality evaluation tool that relies on markers placed in the environment, serving as points of reference from which to compute the virtual object's position/orientation error. The tool collects logs from each AR device and computes the placement and orientation error of the virtual objects seen by the AR users (§3.3).

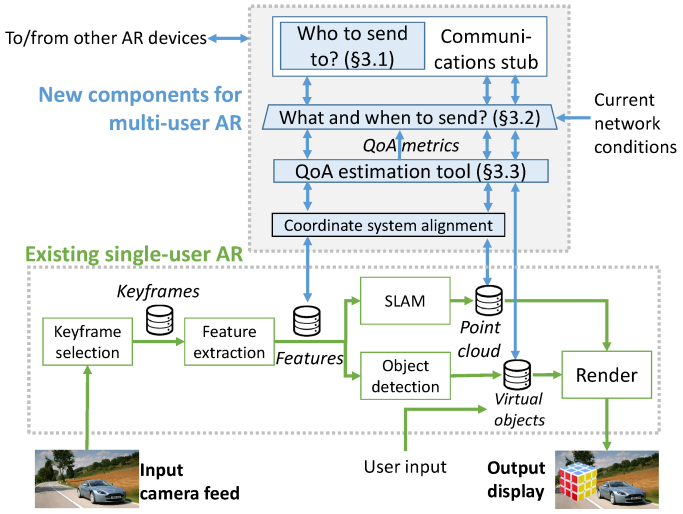


Figure 2: ShareAR integrates new components for multi-user AR with existing single-user AR.

Underlying the above ideas is the ability to experiment with the internal computations and communications of multi-user AR. Because existing multi-user AR platforms [3, 13, 21] are closed source, we are currently working on integrating the above ideas into our Android prototype, ShareAR, as shown in Fig. 2. The purpose of our research prototype is not to compete with existing commercial platforms, but rather to provide a research prototype for experimentation and testing. ShareAR implements multi-user AR functionality, including sharing environmental maps and virtual object information with other AR devices, as well the proposed quality measurement tool, and provides a platform on top of which other AR applications can be built.

In the remainder of this paper, we discuss relevant AR background (§2.1), performance metrics (§2.2), related work (§2.3), our research agenda (§3), and conclude in (§4).

2 BACKGROUND

2.1 Multi-User AR & SLAM

Simultaneous localization and mapping (SLAM) is a fundamental component of modern mobile AR systems, including those from Google, Apple, and Microsoft [5, 14, 23]. In a typical single-user AR scenario, the AR device uses SLAM to construct a point cloud representing the real world, and also estimates its own location and orientation (known as *pose*). To compute the point cloud, SLAM selects a subset of

camera frames (known as keyframes), extracts features from the keyframes, runs SLAM algorithms on the features (*e.g.,* bundle adjustment), and outputs the 3D coordinates of the features (*i.e.,* the point cloud) and the estimated device pose. This pipeline is shown in the bottom half of Fig. 2. The 3D coordinates of the point cloud are relative to an origin point, typically set as the pose of the device when the AR app was first launched. The device's coordinate system is called its *world frame*.

To place virtual objects for AR, the device records the pose of the virtual object (the pose of the virtual object is defined as its location and orientation, and can be provided by user input, or an object detector). The device runs SLAM continuously to update its own pose estimate and the point cloud, and then draws the virtual object on the display when its FoV overlaps with the virtual object's pose.

In a multi-user scenario, each device runs SLAM to build up its own point cloud of the environment, with coordinates stored relative to its own world frame, as shown in Fig. 1. One device can add a virtual object to its own world frame and desire these objects to be reflected on the displays of the other AR devices. Since the devices have different world frames, they need to *align* their maps (*i.e.,* point clouds) using map fusion techniques. However, the point clouds estimated by each AR device are likely not identical, making map alignment noisy and potentially resulting in inconsistency of the virtual objects' locations and orientation across users.

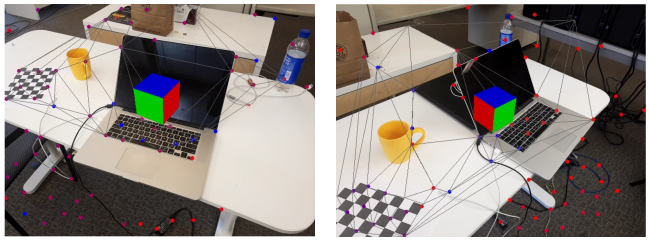
For this paper, we define a SLAM map[1] as comprised of several data structures: a 4×4 homogeneous transformation matrix representing the device's pose when each keyframe was captured [16], the 3D coordinates of the keyframe features (*i.e.,* the point cloud), feature descriptors, and a Distributed Bag Of Words [9] (DBoW) database enabling fast keyframe matching for map alignment.

2.2 Metrics for Multi-User AR

Single-user AR quality can be measured through a number of quantitative metrics, including rendering quality, FoV size, registration error, and task completion [30]. In this work, we focus on quantitative metrics relevant to a multi-user, collaborative AR experience. We call these multi-user Quality of Augmentation (QoA) metrics. Specifically, the metrics we focus on are:

- **Virtual Object Pose Accuracy (m,°):** The pose accuracy measures how far the virtual object is from the correct position/orientation [33]. We assume that the correct pose is given by the initiating device (either from user input, or by an object detector). For example, Fig. 3 shows screenshots of an inaccurately placed virtual cube seen by device B, relative to a virtual cube initially placed by device A.

[1]The CloudAnchor, ARWorldMap, and SpatialAnchor objects used in Google [13], Apple [3], and Microsoft's [21] AR platforms, respectively, are some version of SLAM maps.



(a) Virtual cube as seen by device A. **(b) Virtual cube as seen by device B.**

Figure 3: Example of virtual object pose inaccuracy.

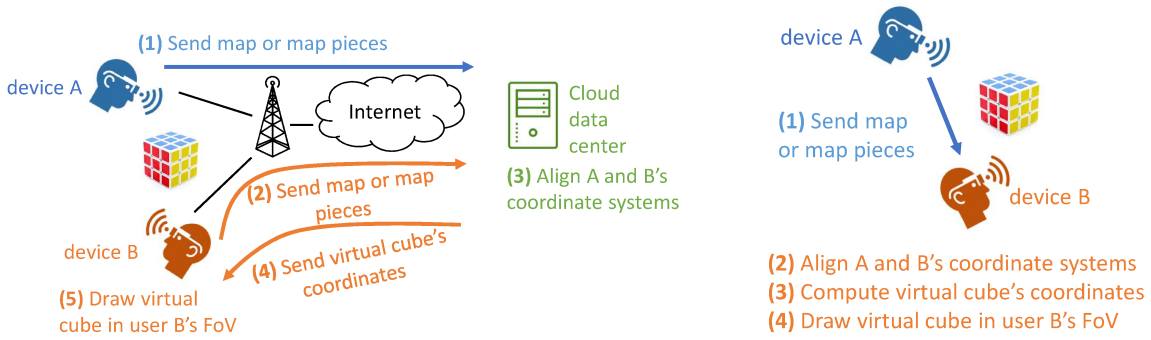
- **Virtual Object Pose Jitter and Drift (m/s,°/s):** The pose jitter measures the motion of the virtual object between subsequent frames [22]. The pose drift measures the accumulation of position and orientation errors over time [18]. Low jitter and drift means that the virtual object stays in place with respect to the real world over time, even if the user moves.
- **End-to-end Latency (s):** The end-to-end latency measures the time from when device A places a virtual object, to when it is drawn in the FoV of device B. The components of latency can include communication, computation, rendering, I/O, OS time, etc.

We note that while pose accuracy, jitter, and drift have been considered in the robotics community, "pose" in that setting typically refers to the pose of the device. Instead, for AR, we are interested in the pose of the virtual objects displayed in the user's FoV. Virtual object pose estimation differs from real object pose estimation in that there is no external ground truth observable by each device.

2.3 Related Work

Mobile AR systems: Several works focus on object detection for AR [2, 7, 15, 19, 20, 27, 33], sometimes with the help of the cloud/edge. While object detection is one component of AR that can generate virtual objects, we consider communicating general pose information about the virtual objects to other users. OverLay [17] labels real-world objects with the help of odometry sensors, whereas we additionally incorporate continuous camera frames, and can display virtual objects with 3D location/orientation in the world. MARVEL [6] focuses on the energy-efficiency of mobile AR, and GLEAM [24] discusses lighting rendering for virtual objects, which are orthogonal directions to this work.

Multi-user AR: CARS [34] discusses sharing object detection results between multiple users, whereas we consider more general virtual objects and integration with SLAM-based AR. In industry, Google ARCore, Apple ARKit, and Microsoft HoloLens are recent mobile AR platforms [3, 13, 21]; our focus is on measurements and comparative evaluation



(a) Cloud-based architecture, similar to Google ARCore. **(b) P2P architecture, similar to Apple ARKit.**

Figure 4: Current mobile AR platforms use peer-to-peer or cloud-based architectures.

of the communication architectures and strategies of such systems.

Multi-user SLAM: Since AR is closely tied to SLAM, there has been some work on multi-user SLAM in the robotics context. These works mainly focus on coordinate system alignment algorithms, with less attention paid to the communication aspects. For example, Zou *et al.* [35] hardcodes transmitting the SLAM data up to every 5 frames, while Schmuck *et al.* [31] transmits SLAM information whenever it is updated. Furthermore, these works focus on communicating information about the real world, while AR also requires communicating information about virtual objects.

3 RESEARCH AGENDA

3.1 AR Communication Architectures

3.1.1 Current P2P and Cloud-based Architectures. There are two primary communication architectures in current SLAM-based mobile AR systems: cloud-based and P2P. For the remainder of this paper, for ease of exposition, we will refer to two devices, A and B. Device A places a virtual object in its environment, and wishes to share this information with a newly joined device, B. In our current prototype, we focus on the two-device scenario for simplicity, but we intend to scale up our experiments in the near future.

Cloud-based: In a centralized architecture, the cloud collects device pose information from the AR devices, performs processing, and returns results as needed. Cloud-based architectures are used, for example, by Google ARCore [13] and MARVEL [6]. The information exchange is illustrated in Fig. 4a, and described below:

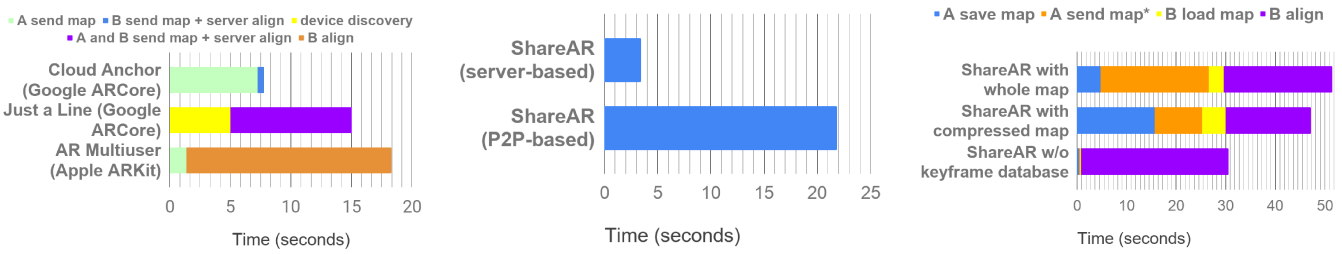
(1) *Device A sends*: A sends its SLAM map (or the related camera frames), and the virtual object's coordinates to the cloud.
(2) *Device B sends*: B sends a piece of its map corresponding to its current location (or the related camera frames) to the cloud.
(3) *Cloud aligns coordinate systems*: The cloud runs SLAM (if camera frames only were sent), then aligns A's map and B's map piece, and computes the virtual object's pose in B's coordinate system.
(4) *Cloud sends virtual object's coordinates*: The cloud sends the computation result to device B.
(5) *B draws virtual object*: B draws the virtual object in its world coordinate system.

P2P-based: In a de-centralized or P2P architecture, AR devices communicate directly with each other, without the assistance of a central entity. Such an architecture is followed, for example by Apple ARKit [3]. The process is illustrated in Fig. 4b and described below:

(1) *Device A sends*: A sends its SLAM map (or related camera frames) and the virtual object's coordinates to B.
(2) *Device B aligns coordinate systems*: B runs SLAM (if only camera frames were sent), then aligns A and B's coordinate systems, and computes the virtual object's pose in B's coordinate system.
(3) *B draws virtual object*: B draws the virtual object in its world coordinate system.

In summary, these two architectures require similar types of computation, but at different locations (*i.e.*, on the device or in the cloud), which impacts the information exchange between the devices.

Measurements: We conducted measurements of several AR applications utilizing the above architectures in a controlled lab setting. We used Samsung Galaxy S7 smartphones with WiFi connectivity (50 Mbps download and upload speed), unless otherwise mentioned. Each measurement was repeated 3 times, with the averages plotted. In Fig. 5a, we show end-to-end latency measurements of three AR apps: CloudAnchor [11] and Just a Line [12] (Google ARCore demo apps), and AR MultiUser [4] (Apple ARKit demo app). We observe that latencies between device A placing a virtual object and device B drawing the virtual object are quite long, from 7-18 s. The cloud-based apps tend to have longer communication times and shorter coordinate system alignment times, because of cloud compute resources. On the other hand, the P2P app has shorter communication time



(a) End-to-end latency of current mobile AR apps.

(b) Time to align coordinate systems on a server versus mobile device in ShareAR.

(c) ShareAR **(P2P version) reduces latency by optimizing data transmissions.**[2]

Figure 5: Latency breakdown starting from device A placing a virtual object to device B rendering it on the display.

but longer coordinate system alignment time due to running the joint computations on the mobile device (in this case an iPad), leading to longer end-to-end latency overall.

3.1.2 Comparing Architectures in ShareAR. Given our understanding of the above architectures, which architecture is more suitable in different scenarios? The P2P architecture has advantages in terms of scalability (there is no central bottleneck link), and privacy (information doesn't need to be sent to the cloud). However, updates to virtual objects can take time to propagate across the devices, potentially resulting in inconsistent information. The cloud architecture has advantages in terms of compute power, and can synchronize updates about the virtual objects across devices, but relies on Internet connectivity. Another possibility is a hybrid architecture, where the devices share information only amongst themselves, but one device acts as a "master" node that undertakes communication and computation efforts. Such an approach essentially assigns one of the devices to take the role of the cloud, but places heavy computation and communication demands on the master node.

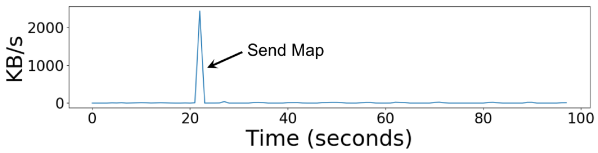
Given that these architectures are currently implemented in different mobile OSes (iOS and Android) and are closed source, an apples-to-apples comparison of their system performance metrics cannot be made between them, nor can modifications be made. Our idea is to develop an open-source reference system that allows comparison of the different communication architectures. To do this, we build on an open-source state-of-the-art SLAM system for single users [26], and add multi-user capabilities and the ability to switch between the communication architectures observed in the commercial mobile AR platforms. This will enable accurate measurements of the performance of each component of the AR computation and communication pipeline. Our system, ShareAR, will provide researchers and developers with insight into the system requirements of multi-user mobile AR, guidelines on architectural decisions, and understanding of which parts of the AR pipeline can be optimized.

We have implemented an initial prototype of ShareAR with two Android devices and an edge server. ShareAR allows device A to place a virtual cube and device B to receive and render the cube in its FoV, with full control over all components of the system, including SLAM algorithms, communication protocol, communication frequency, coordinate system alignment frequency, etc. In Fig. 5b, we show some initial measurements of ShareAR, comparing the computation time of coordinate system alignment of P2P and server-based architectures. We can see that the edge server-based computation time is lower, suggesting that an edge-server based architecture may be ideal as communication latency is also low in edge scenarios [28]. Specifically, the map alignment time with the P2P architecture is seven times longer than in the server-based architecture, which suggests great potential for both the edge- and cloud-based architectures. In fact, the server-based architecture can still tolerate an additional 10 seconds of communication latency and still have lower end-to-end latency than the P2P architecture (Fig. 5b).

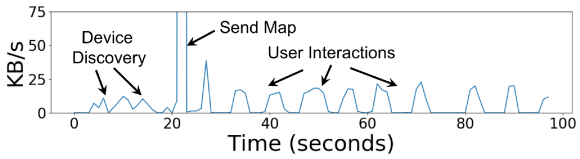
Black box testing: To ensure that the results produced ShareAR reflect existing AR systems, we will perform black box testing. We will choose a set of sample AR apps and scenarios, and tune ShareAR until its results are similar to those observed in commercial platforms (*e.g.,* Google ARCore). While we cannot have perfect reproduction of commercial platforms, due to their opaqueness, we explicitly try to match their performance for a given set of test cases. In our initial results with ShareAR, the computation latency of the server-based architecture (upper bar in Fig. 5b) is roughly comparable to computation latency of Google ARCore's server-based architecture (the "B send map + server align" bar in Fig. 5a). (ShareAR is slightly slower because its computation latency includes the DBoW generation time, which we were not able to measure in ARCore because it is closed-source.) Similarly, ShareAR's P2P computation latency (lower bar in Fig. 5b) is comparable to ARKit's P2P computation latency ("B align" in Fig. 5a).

3.2 Adaptive AR Communications

A good understanding of the network traffic sent and received by the AR devices is needed in order to effectively manage the traffic and ensure good user QoE. While low latency is a key requirement for mobile AR [28], our initial measurements show that bandwidth requirements can also



(a) Bandwidth trace.



(b) Zoomed in version of (a).

Figure 6: Bandwidth trace from AR app, showing large map and small user interaction transmissions.

be significant, due to the exchange of large amounts of map data. We collected network traces from several AR applications [4, 11, 12], and show a representative example from the Just a Line app in Fig. 6. The trace shows a large burst of traffic (more than 26 Mb) around $t = 21$ s, and smaller periodic bursts starting at $t = 35$ s. The ARCore documentation and code [13] suggest that the large traffic burst contains SLAM map data or camera frames, while the small bursts contain the coordinates of the virtual line drawn by user A, who is drawing periodically. Moreover, the large data burst needs to be repeated if a new virtual object is placed in significantly different location by a user, which we have observed experimentally (not shown).

3.2.1 Adapting AR Communications to the Network. Based on the above observations of large amounts of SLAM/visual update data, we propose adapting the size and frequency of data transmissions in order to improve communication efficiency and reduce user-perceived latency. We propose both lossy and lossless adaptation methods to do so.

Lossless methods: Lossless methods involve suppressing data that is not needed; for example, device A does not need to send keyframe features corresponding to areas far away from device B. In our initial work with ShareAR, instead of sending the entire map for alignment between device A and B, device A omitted sending its DBoW, which consumed a large fraction of the map size. Instead, device B used its own locally generated DBoW to perform matching with A's keyframes. Fig. 5c shows that this simple optimization drastically reduced the communication time (to nearly zero). However, the alignment time increased because B needs to query all of A's keyframes in B's DBoW (which grows continuously as B captures more frames), while in the default case, B only needs to query its current keyframe in A's DBoW (which is frozen after transmission). A naive approach of simply applying the standard Boost gzip compressor [29] to the entire map, while saving communication time compared to the baseline of full map transmission, did not result in as much latency savings overall because it did not intelligently select which data structures to send.

Lossy methods: Lossy methods involve tradeoffs between the amount of communicated data and the QoA metrics. For example, sending frequent map updates can improve the virtual object's pose accuracy by re-aligning the devices' coordinate systems, but also increase the end-to-end latency due to longer communication time and use up system resources for frequent data processing. Essentially, for lossy methods, there are a set of "control knobs" that can be tuned (*e.g.,* frequency of updates/map alignment, number of features, spatial granularity or compression of the point cloud [25]), each of which has some impact on QoA. We are inspired by auto-tuning works in the multimedia community [1, 32], where the control knobs/parameters are tuned to the current network conditions to maximize quality-of-experience. In the context of AR, these operating conditions include network bandwidth, and the performance metrics are the QoA metrics. Our proposed method first involves performing an offline characterization of the tradeoffs between application-layer parameters and the QoA. This will provide valuable rules of thumb on how to set these parameters. Then, given an understanding of these tradeoffs, we will formulate this problem as an optimization problem, the output of which are the optimal configuration parameters. We envision our adaptation method implemented as an API, which a developer can incorporate into her AR app to improve its communication efficiency.

3.3 Quality of Augmentation Tool

A good estimate of multi-user AR performance metrics (Sec. 2.2) is needed to evaluate the performance of multi-user AR systems. Such a tool should take as input information from device A on the correct location of each virtual object, and information from device B on the rendered location of each virtual object. The outputs of the tool are the QoA metrics defined earlier. We envision such a tool as an overlay in our ShareAR system, collecting logs from the AR devices, and running on a device or an external server. While we will evaluate our measurement tool in ShareAR, the tool will also be usable in any multi-user AR system that exposes the appropriate information (poses of virtual objects, certain coordinates in the point cloud, and timing information; details below).

While latency is relatively easy to compute by instrumenting timers in the code, the main challenge is measuring the virtual object pose accuracy (and its derivatives, drift and jitter). This problem is illustrated in Fig. 3a, 3b, where the virtual cube is drawn at different locations, with respect to the real world, in device A and B's FoVs (*e.g.,* in A's FoV, the cube hovers above the center of the laptop, while in B's FoV,

[2] In this set of experiments, send time is estimated as $\frac{\text{file size}}{5\ \text{MBps}}$.

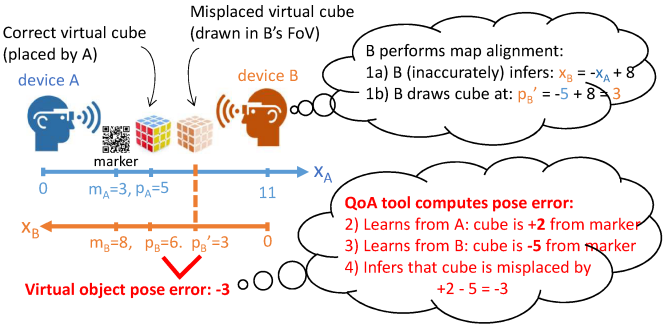


Figure 7: The QoA tool uses markers placed in the environment to measure the pose accuracy of virtual object. Above is a simplified 1D example.

the cube hovers to the upper left of the laptop). How can B measure the misplacement and angle of its virtual cube?

The robotics community typically tackles this problem by collecting ground truth pose measurements (*e.g.,* using a laser system) that pinpoints the exact location of all the physical objects in a space. However, such solutions do not apply in the AR context because we are interested in the pose of virtual objects, not the pose of real-world objects. Other work [33] in the multimedia community manually labels the ground truth locations of the real-world objects on which the virtual object was placed. However, this is not scalable over many frames.

Our idea, in a nutshell, is to place easily recognizable markers in the environment (*e.g.,* ArUco markers [10]), whose location and orientation can be accurately estimated by the devices using SLAM or PnP methods [16], and used as reference points to measure the virtual object's pose accuracy. A toy 1D example is shown in Fig. 7, and described below:

(1) After (inaccurately) aligning A and B's coordinate systems, device B draws the cube at $p_B\prime = 3$. This is an error of -3.
(2) Device B accurately estimates the marker's position $m_B = 8$, and reports $m_B, p_B\prime$ to the tool.
(3) Device A accurately estimates the marker position $m_A = 3$, and reports $m_A, p_A = 5$ to the QoA tool.
(4) The QoA tool computes the cube's pose error as $(p_A - m_A) - (m_B - p_B') = (5 - 3) - (8 - 3) = -3$.

We plan to extend this technique to all 6-DoF (position, orientation). The virtual object's pose jitter and drift will be computed based on the pose accuracy.

We will place these markers at various locations in a controlled environment. Having more markers requires more setup time, but can improve the measurement accuracy, as there are more reference markers to compare against. Having fewer markers is a simpler setup, but results in fewer reference points to calculate the virtual object's pose accuracy against. We note that the markers do not need to be placed at regular, fixed locations in the environment; they can be moved between experiments, as the QoA tool only requires that they be spaced out. In our initial experiments, we investigated whether the presence of the markers impacts the pose of the virtual object (*i.e.,* whether the measurement setup affects the quantity we are trying to measure); however, we found that adding/removing the marker from the scene had little impact on the virtual object's pose.

While we mainly propose a marker-based setup to estimate the virtual object's pose, a marker-less setup may also be possible, for ease of deployment but potentially with lower measurement accuracy. For example, we could use natural features in the scene as the "markers". However, even with the ArUco markers, which are specifically designed to be easily detectable in the environment, we find that multiple AruCo markers (*i.e.,* an ArUco board) are needed in the scene to achieve good measurement accuracy. Using natural features would likely further degrade the measurement accuracy. Furthermore, while extracting the 3D locations of the natural features to use as markers is possible as an intermediate output of SLAM, this would couple the measurement tool with the quantity being measured (the virtual cube's pose), which is undesirable.

4 CONCLUSIONS

Our proposed research dissects and enhances the communication capabilities of multi-user AR. We first examine current mobile AR platforms to understand their communication architectures, end-to-end latency, and bandwidth requirements. We propose ShareAR, which allows fine-grained control of the AR processing pipeline, in order to conduct fair comparisons between different communication architectures and strategies. We propose optimizations of the large data transmissions arising from exchanging SLAM map data or other visual information, accounting for AR-specific quality metrics and current network conditions. We also propose a simple mechanism using markers to measure the AR quality in controlled environments. The outcome of this research will be communication-efficient support for multi-user AR applications in the near future.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their valuable comments, from which this paper greatly benefited. This work has been supported in part by NSF grants CNS-1817216 & CSR-1903136.

REFERENCES

[1] Akhtar, Z., Nam, Y. S., Govindan, R., Rao, S., Chen, J., Katz-Bassett, E., Ribeiro, B., Zhan, J., and Zhang, H. Oboe: Auto-tuning video abr algorithms to network conditions. *ACM SIGCOMM* (2018).
[2] Apicharttrisorn, K., Ran, X., Chen, J., Krishnamurthy, S., and Roy-Chowdhury, A. Frugal following: Power thrifty object detection and tracking for mobile augmented reality. *ACM SenSys* (2019).
[3] Apple. Creating a multiuser ar experience. https://developer.apple.com/documentation/arkit/creating_a_multiuser_ar_experience.

[4] Apple. Swiftshot. https://developer.apple.com/documentation/arkit/swiftshot_creating_a_game_for_augmented_reality.

[5] Apple. Understanding arkit tracking and detection - wwdc 2018. https://developer.apple.com/videos/play/wwdc2018/610/, 2018.

[6] Chen, K., Li, T., Kim, H.-S., Culler, D. E., and Katz, R. H. MARVEL: Enabling mobile augmented reality with low energy and low latency. *ACM Sensys* (2018).

[7] Chen, T. Y.-H., Ravindranath, L., Deng, S., Bahl, P., and Balakrishnan, H. Glimpse: Continuous, real-time object recognition on mobile devices. *ACM SenSys* (2015).

[8] Citibank GPS: Global Perspectives and Solutions. Virtual and augmented reality. https://www.citibank.com/commercialbank/insights/assets/docs/virtual-and-augmented-reality.pdf, October 2016.

[9] Gálvez-López, D., and Tardós, J. D. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics 28*, 5 (October 2012), 1188–1197.

[10] Garrido-Jurado, S., Muñoz Salinas, R., Madrid-Cuevas, F., and Marín-Jiménez, M. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recogn.* (2014).

[11] Google. Cloudanchor. https://developers.google.com/ar/develop/java/cloud-anchors/quickstart-android.

[12] Google. Just a line. https://justaline.withgoogle.com/.

[13] Google. Share ar experiences with cloud anchors. https://developers.google.com/ar/develop/java/cloud-anchors/cloud-anchors-overview-android, May 2018.

[14] Google. Fundamental concepts | arcore | google developers. https://developers.google.com/ar/discover/concepts, May 2019.

[15] Ha, K., Chen, Z., Hu, W., Richter, W., Pillai, P., and Satyanarayanan, M. Towards wearable cognitive assistance. *ACM MobiSys* (2014).

[16] Hartley, R., and Zisserman, A. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[17] Jain, P., Manweiler, J., and Roy Choudhury, R. Overlay: Practical mobile augmented reality. *ACM MobiSys* (2015).

[18] Kerl, C., Sturm, J., and Cremers, D. Dense visual slam for RGB-D cameras. *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2013).

[19] Liu, L., Li, H., and Gruteser, M. Edge assisted real-time object detection for mobile augmented reality. *ACM MobiCom* (2019).

[20] Liu, Q., and Han, T. Dare: Dynamic adaptive mobile augmented reality with edge computing. *IEEE ICNP* (2018).

[21] Microsoft. Shared experiences in Unity. https://docs.microsoft.com/en-us/windows/mixed-reality/shared-experiences-in-unity, March 2018.

[22] Oskiper, T., Sizintsev, M., Branzoi, V., Samarasekera, S., and Kumar, R. Augmented reality binoculars. *IEEE Transactions on Visualization and Computer Graphics 21*, 5 (2015), 611–623.

[23] Pollefeys, M. Microsoft hololens facilitates computer vision research by providing access to raw image sensor streams with research mode. https://www.microsoft.com/en-us/research/blog/microsoft-hololens-facilitates-computer\-vision-research-by-providing-access-to-raw\-image-sensor-streams-with-research-mode/, June 2018.

[24] Prakash, S., Bahremand, A., Nguyen, L. D., and LiKamWa, R. GLEAM: An Illumination Estimation Framework for Real-time Photorealistic Augmented Reality on Mobile Devices. *ACM MobiSys* (2019).

[25] Qian, F., Han, B., Pair, J., and Gopalakrishnan, V. Toward practical volumetric video streaming on commodity smartphones. In *Proceedings of the 20th International Workshop on Mobile Computing Systems and Applications* (2019), ACM, pp. 135–140.

[26] Qin, T., Li, P., and Shen, S. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics 34*, 4 (Aug 2018), 1004–1020.

[27] Ran, X., Chen, H., Liu, Z., and Chen, J. Deepdecision: A mobile deep learning framework for edge video analytics. *IEEE INFOCOM* (2018).

[28] Satyanarayanan, M., Bahl, V., Caceres, R., and Davies, N. The case for vm-based cloudlets in mobile computing. *IEEE Pervasive Computing* (2009).

[29] Schäling, B. *The boost C++ libraries*. Boris Schäling, 2011.

[30] Schmalstieg, D., and Hollerer, T. *Augmented reality: principles and practice*. Addison-Wesley Professional, 2016.

[31] Schmuck, P., and Chli, M. Multi-uav collaborative monocular slam. *IEEE International Conference on Robotics and Automation* (2017).

[32] Zhang, H., Ananthanarayanan, G., Bodik, P., Philipose, M., Bahl, P., and Freedman, M. J. Live video analytics at scale with approximation and delay-tolerance. *USENIX NSDI* (2017).

[33] Zhang, W., Han, B., and Hui, P. Jaguar: Low Latency Mobile Augmented Reality with Flexible Tracking. *ACM Multimedia* (2018).

[34] Zhang, W., Han, B., Hui, P., Gopalakrishnan, V., Zavesky, E., and Qian, F. CARS: Collaborative augmented reality for socialization. *ACM HotMobile* (2018).

[35] Zou, D., and Tan, P. Coslam: Collaborative visual slam in dynamic environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence 35*, 2 (2012), 354–366.