Here To Stay: A Quantitative Comparison of Virtual Object Stability in Markerless Mobile AR

Tim Scargill
Duke University
ts352@duke.edu

Gopika Premsankar
University of Helsinki
gopika.premsankar@helsinki.fi

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

Maria Gorlatova
Duke University
maria.gorlatova@duke.edu

***Abstract*—Mobile augmented reality (AR) has the potential to enable immersive, natural interactions between humans and cyber-physical systems. In particular *markerless AR*, by not relying on fiducial markers or predefined images, provides great convenience and flexibility for users. However, unwanted virtual object movement frequently occurs in markerless smartphone AR due to inaccurate scene understanding, and resulting errors in device pose tracking. We examine the factors which may affect virtual object stability, design experiments to measure it, and conduct systematic quantitative characterizations across six different user actions and five different smartphone configurations. Our study demonstrates noticeable instances of spatial instability in virtual objects in all but the simplest settings (with position errors of greater than 10cm even on the best-performing smartphones), and underscores the need for further enhancements to pose tracking algorithms for smartphone-based markerless AR.**

***Index Terms*—Markerless augmented reality, VI-SLAM, spatial stability, virtual object drift, relocalization, ARKit, ARCore.**

(a) Time 0 (b) Time 1 (c) Time 2

Fig. 1: Virtual object drift in markerless VI-SLAM-based AR (ARKit, iPhone 11, iOS 14.4, IKEA Place app [4]). A virtual lamp drifts from its original position (a), colliding with a real glass on the table (b), and moving off the table entirely (c).

I. Introduction

Augmented Reality (AR), the overlaying of virtual content onto a view of the real world, is a key enabler for interactions between humans and cyber-physical systems. It provides immersive, natural interactions, which can help users to efficiently exchange knowledge in a diverse range of application domains, from manufacturing [1] to surgery [2]. However, the effectiveness of an AR system, and the degree to which it is accepted by users, is inextricably tied to its ability to achieve *spatial stability*. This means that when a virtual object is placed, it should remain in exactly the same position relative to the real world, even if in the meantime the user moves around, walks away, pauses the AR app to send a text message, or performs any other action. When the virtual object does not, the illusion is immediately broken; users may become disengaged or frustrated as virtual object instability impedes or compromises task performance. If AR truly is "here to stay", then the virtual objects that we place within it must be as well.

In the last few years, *markerless* AR has become commonplace. Markerless AR enables the positioning of virtual content without predefined markers (recogizable textures such as fiducial markers, QR codes, or images); natural features within the environment are mapped and tracked instead. The convenience this markerless technique provides has led to its widespread use in AR apps, such as Pokemon GO [3] and IKEA Place [4]. The vast majority of modern AR platforms, such as ARCore [5] and ARKit [6], support markerless AR through *Visual-Inertial Simultaneous Localization and Mapping (VI-SLAM)*. VI-SLAM is used to concurrently map the environment and track the pose (position and orientation) of the AR device. If the correct pose is known in relation to the surrounding environment, virtual objects can be rendered in the correct position relative to the real world.

However, errors still frequently occur in the pose estimation process because of challenging visual conditions or device movement. This results in a virtual object being rendered in a different position from where it was originally placed, a problem often termed *drift* (i.e., the Euclidean distance between the original and resulting virtual object positions), as shown in Figure 1. At 'Time 0' (Figure 1a) the user places the virtual black and white lamp on the table; they then walk away from the area (perhaps to bring another person to come and look at the lamp) before returning to view it at 'Time 1'. The virtual lamp has drifted out of position and collided with a real-world glass (Figure 1b). After walking away and returning again, the drift is even more pronounced at 'Time 2', and the lamp has now moved off the table (Figure 1c). These positional errors can have a severe negative impact on a user's experience, especially when the virtual object violates real-world physics.

Our goal is to measure the current level of virtual object stability on different AR platforms and smartphones, for a variety of device movements, and thereby assess their suitability for practical cyber-physical-human system interfaces. We focus

here on the common scenario of 'single-session' markerless AR with stationary virtual content (e.g., IKEA Place [4]), in static indoor environments. We define two metrics for virtual object instability. We use *drift* to refer to the measurable displacement when device tracking is available, but pose estimation is inaccurate, and a virtual object is rendered in a different location. Sometimes an AR system may also lose tracking temporarily, and while the device attempts to relocalize, unwanted virtual object movement frequently occurs. We term this period when tracking is unavailable *relocalization time*.

To the best of our knowledge, this work is the first to perform a direct, quantitative comparison of virtual object stability on both multiple smartphone models and an AR headset. We cover related work (Section II), highlight factors which can affect stability (Section III), and establish six common actions smartphone AR users may perform (Section IV). We compare performance across multiple platforms and devices (Section V), discuss our findings (Section VI), then present our conclusions (Section VII). Our key contributions are summarized as follows:

- We benchmark virtual object stability on an AR headset, the Microsoft HoloLens 2, demonstrating the high level of performance these devices are able to achieve, with mean drift of less than 2cm for all six user actions we tested.
- We conduct cross-platform experiments to compare virtual object stability on five smartphones, revealing noticeable and highly variable performance across all platforms, devices and actions. For example, mean drift after walking away and returning was 3.4cm on a Nokia 7.1 running ARCore 1.23 and 25.1cm on an iPhone 11 running ARKit 4.

II. Related Work

AR platform tracking state: ARCore and ARKit both provide indicators of current tracking quality [7], [8], though granularity is limited and neither include the magnitude of virtual object instability likely to occur. ARKit indicates whether tracking is 'not available', 'limited' (results are questionable), or 'normal', while ARCore describes the tracking state as 'stopped', 'paused' (which may indicate device tracking has been lost), or 'tracking'. The methods by which these tracking states are derived are proprietary. *To the best of our knowledge, no data on tracking accuracy achieved on these platforms for markerless AR has been made public.*

Virtual content positional errors: While virtual content positional errors are a known problem in AR [9], until recently quantitative assessment of them has been hampered by a lack of a suitable measurement method. One small scale study, using an optically-tracked stylus, found drift on the HoloLens 1 of 0.5–0.6cm [10], but only in one environment: this method is impractical for many settings. We have developed an alternative methodology using fiducial markers and measured a virtual object position error of 6cm on a Samsung Galaxy S20 [11]; however, this technique requires modifications to the real-world environment. Here we use a method based on placing virtual objects near a real reference point, that we introduced in [12].

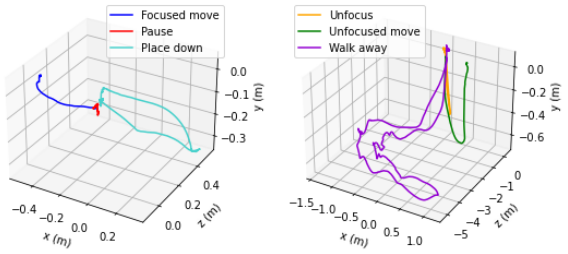
Pose tracking accuracy: Another option is to assess virtual content positional errors indirectly by measuring the pose tracking accuracy of an AR system. The standard method for measuring VI-SLAM performance is to compare pose estimates with ground truth pose obtained using a motion capture system such as OptiTrack [13] or Vicon [14], as in [15], [16], [17]. In [18], a motion capture system is used to compare device localization errors on ARCore, ARKit and the HoloLens 1. However, methods such as this are limited because they do not allow quantifying virtual object position errors directly, and any use of a motion capture system is impractical for evaluations in more than a small number of different environments, due to the logistics associated with setup and calibration.

III. Factors Affecting VI-SLAM Performance

Virtual object stability on AR platforms such as ARCore [5] and ARKit [6] is dependent on the pose tracking accuracy achieved by the VI-SLAM system employed. While the exact VI-SLAM algorithms on commercial devices are proprietary, a number of factors are known to affect performance:

Device hardware: The resolution of the visible-light cameras on modern smartphones is sufficient for current SLAM algorithms, but other factors such as the rolling shutter effect and light sensitivity may affect performance [19]. The use of a stereo camera pair (stereo SLAM) has been shown to reduce error [20], while headsets can also benefit from an additional pair of cameras that face outward at a different angle to increase the feature tracking field of view (not possible on a smartphone due to form factor). Because an iterative solver is used in pose estimation, and computation time is limited due to real time constraints, more powerful devices are able to complete more iterations and thereby achieve greater accuracy. Finally, a few smartphones are equipped with time-of-flight (ToF) depth sensors that measure distances to different points in the environment by emitting near-infrared light [21]. The depth measurements from these sensors allow virtual object placement without device motion. ToF sensors are categorized into two types: indirect ToF (iToF) sensors are found on a few Android smartphones and the Microsoft HoloLens 2, and direct ToF (dToF) sensors (marketed as LiDAR scanners) are found on high-end Apple devices.

Visual environment and device motion: If the input image contains few distinguishable elements (due to e.g., dark or textureless environments, or motion blur [22]), then a limited number of feature points can be detected, which affects the rest of the pose estimation process. Errors may also occur in instances of repeated textures. The measurement accuracy of ToF depth sensors is dependent on the reflectance properties of surfaces in the environment; depth estimates for highly absorptive (black) or reflective regions, at greater distances or oblique angles can be inaccurate [23]. dToF sensors may measure depth incorrectly when surfaces are too close to the sensor (e.g., less than 20cm) [21]. Finally, fast device acceleration or rotation reduces pose estimation accuracy by introducing noise into the inertial input data [17], and because integration errors accumulate over time, longer trajectories also tend to result in larger errors.



(a) (b)

Fig. 2: Representative device trajectories for 6 common actions that users may perform after virtual object placement: *Focused move*, *Pause*, and *Place down* (a), *Unfocus*, *Unfocused move*, and *Walk away* (b).

IV. Device Movement in Smartphone AR

We aim to investigate the effect of device movement on virtual object stability. To this end we define six common actions users may perform after placing a virtual object:

- *Focused move*: user moves to a different viewing angle (to inspect another side of the virtual object), while keeping the camera focused on the virtual object.
- *Unfocus*: user lets hand holding device hang at their side (as if distracted), then raises it to focus on the virtual object.
- *Unfocused move*: user lets the hand holding the device hang at their side, moves to a different viewing angle, then raises the device to focus on the virtual object.
- *Walk away*: user lets the hand holding the device hang at their side, walks away (to e.g., perform another action), returns and raises the device to focus on the virtual object.
- *Pause*: user pauses the AR app (to e.g., send a text), then resumes it while focused on the virtual object position.
- *Place down*: user places device down on a nearby surface so that the rear camera is covered (to e.g., pick up another object), then raises the device to focus on the virtual object.

Pause and *Place down* are challenging for VI-SLAM systems because they involve an interruption to input sensor data (visual input data for *Place down* when the camera is covered, and both visual and inertial input data for *Pause* when the app is paused). Because this results in a temporary loss of tracking, we examine relocalization time as well as drift in these cases.

For all subsequent experiments, the change in viewing angle for *Focused move* and *Unfocused move* was set to approximately 45° (on a horizontal plane), the distance walked away in *Walk away* to 10 steps (approximately 7m), and the *Pause* and *Place down* interruption duration to approximately 5s. Representative trajectories for each of these actions, captured in our experiments, are shown in Figure 2.

V. Virtual Object Stability Experiments

This section presents our virtual object stability experiments, conducted using the method we introduced in [12], on commercial AR platforms: the Microsoft HoloLens 2 AR headset (Section V-A) and five different smartphones (Section V-B).

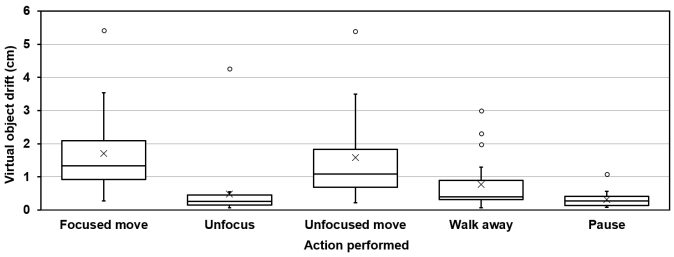


Fig. 3: Virtual object drift on the HoloLens 2 after performing different actions. Mean drift was under 2cm for all actions.

A. AR Headset Baseline: Microsoft HoloLens 2

To start, we measure virtual object stability on a state-of-the-art AR headset, the HoloLens 2, which has four synchronized 30Hz grayscale global-shutter tracking cameras (two configured as a stereo rig, two facing outward at a wider angle to track features in a wider field of view), and a 1MP iToF depth sensor. This should represent the best performance currently available on headsets, against which smartphones can be compared.

The wearable nature of an AR headset meant that the exact movements for some of the actions performed (Section IV) on the HoloLens 2 were slightly altered compared to a smartphone. However, we kept the motivation for these actions consistent, and replicated what a real user might do in these scenarios. For example, in *Unfocus* we simulated the same scenario with the user turning their head 90° to face in a different direction. Similarly for *Unfocused move* and *Walk away*, the device did not drop to the user's side, and the environment sensors faced in the same direction the user was moving. Because taking off and placing down a headset does not involve covering the environment tracking sensors like on a smartphone, we do not evaluate *Place down* in our baseline.

We performed each of the five actions in 20 environments (in six rooms in two buildings) for a total of 100 trials. Our choice of environments was designed to cover normal usage conditions; 16 of the 20 were consistent with AR platform guidelines [24], [25] for well-lit and textured environments, and none were extremely challenging (e.g., dark conditions, or completely blank surfaces). The drift after performing each action is shown in Figure 3. As expected, the HoloLens 2 achieved a high degree of virtual object stability, with mean drift less than 2cm for all actions, with all drift values less than 6cm. Lowest mean drift was observed for the two actions with the shortest trajectories, *Pause* (0.3cm) and *Unfocus* (0.5cm). The most challenging actions for the HoloLens 2 were those which involved a change in viewing angle between hologram views, *Focused move* (1.7cm) and *Unfocused move* (1.6cm).

B. Cross-Platform Smartphone Experiments

We now move on to comparing virtual object stability on five smartphones: a Nokia 7.1 and a Samsung Galaxy Note 10+, both running Android 11 and ARCore 1.23, an iPhone 11 running iOS 14.4 and ARKit 4, an iPhone 11 running iOS 15.1 and ARKit 5, and an iPhone 13 Pro Max running iOS 15.0 and ARKit 5. The Samsung Galaxy Note 10+ has an iToF depth

TABLE I: Hardware and software comparison of the smartphones used in our experiments (Section V-B).

Device Model	RAM (GB)	World-facing depth sensor	AR Platform
Nokia 7.1	4	None	ARCore 1.23
Samsung Galaxy Note 10+	12	iToF	ARCore 1.23
iPhone 11	4	None	ARKit 4
iPhone 11	4	None	ARKit 5
iPhone 13 Pro Max	6	dToF (LiDAR)	ARKit 5

sensor and the iPhone 13 Pro Max has a dToF depth sensor, which enables us to evaluate their respective impact on virtual object stability. Relevant hardware and software specifications are shown in Table I, and these devices are pictured (along with the HoloLens 2) in Figure 4. Conducting our experiments on the same iPhone 11 running ARKit 4 and then ARKit 5 allows us to assess the impact of any updates to mapping and tracking algorithms in ARKit 5, released in June 2021.

To ensure a fair comparison, we performed the six actions between virtual object views on each of the five smartphones in the same 20 environments as the HoloLens 2, for a total of 600 trials. We performed actions consecutively from the same positions, under fixed artificial lighting, and no apps were run concurrently with our measurement app ARStats [12]. Below we analyze our results for virtual object drift and relocalization time, exploring the factors that impacted performance.

Virtual object drift: Our cross-platform results for virtual object drift are shown in Figure 5a. As expected, mean drift was higher for all actions on all smartphones than on the AR headset, the HoloLens 2 (with the exception of *Focused move* on the iPhone 13 Pro Max, 1.6cm compared to 1.7cm on the HoloLens 2). There was also some consistency across different smartphones in terms of the level of drift caused by specific actions. For example *Unfocus*, involving a relatively short device trajectory, no change in viewing position and no interruption to input data, resulted in the lowest mean drift on the Nokia 7.1 (2.5cm), the Samsung Galaxy Note 10+ (4.0cm) and the ARKit 4 iPhone 11 (1cm), and the second-lowest mean drift on the ARKit 5 iPhone 11 (2.2cm) and iPhone 13 Pro Max (1.1cm). In contrast, *Unfocused move*, with the addition of a change in viewing position and a longer trajectory, was consistently the most or second-most challenging action, resulting in mean drift greater than 8cm on all devices (Nokia 7.1: 12.6cm, Samsung Galaxy Note 10+: 12.1cm, iPhone 11 ARKit 4: 10.1cm, iPhone 11 ARKit 5: 8.9cm) except the iPhone 13 Pro Max (2.3cm). However, we also find important differences in virtual object stability performance across platforms and devices, as we now detail.

One standout result was that the iPhone 13 Pro Max was able to achieve lower mean drift than any other smartphone for all six actions, less than 3cm for all actions. The level of improvement was particularly marked for the challenging *Unfocused move* action, with mean drift just 2.3cm on the iPhone 13 Pro Max compared to 8.9cm on the next best-performing device, the iPhone 11 running ARKit 5. *The extent to which the iPhone 13 Pro is able to achieve lower drift than the iPhone 11, running on the same AR platform version (ARKit 5), is striking, and illustrates the advantages of obtaining more accurate environment mapping through the dToF depth sensor*. It should be noted though, that drift of 5–15cm did occur for every action except *Focused move*, which would still be frustrating for a user in many AR applications.

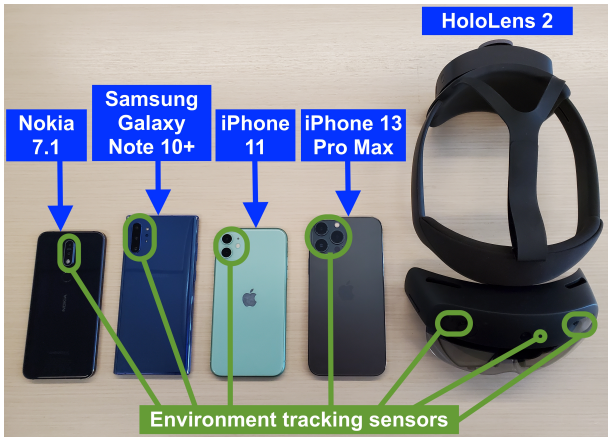
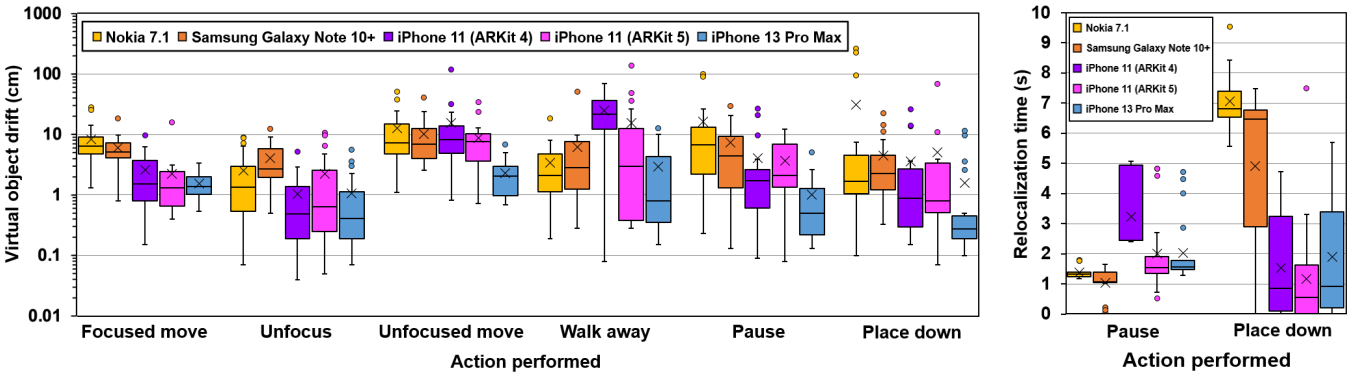


Fig. 4: The AR devices used in our experiments. On the HoloLens 2 tracking cameras are spaced farther apart, with the depth sensor just off-center.

Next we compare the two AR platforms, ARCore and ARKit. All three ARKit devices achieved lower mean drift than the two ARCore devices for the three actions involving the least movement, *Focused move*, *Unfocus* and *Pause*. For the three actions involving greater movement, *Unfocused move*, *Walk away* and *Place down*, the two lower-spec ARKit smartphones sometimes performed worse than ARCore. For example, for *Walk away*, mean drift was 3.4cm on the Nokia, 6.1cm on the Samsung, but 25.1cm on the ARKit 4 iPhone 11. Apple evidently took steps to address this in the next ARKit release, with mean drift down to 15.5cm on the ARKit 5 iPhone 11. However, only the iPhone 13 Pro Max achieved comparable performance with ARCore for this challenging action. Given the iPhone 11 and iPhone 13 Pro Max have similar camera specs (12MP resolution with optical image stabilization, apertures $f/1.8$ and $f/1.5$ respectively) and computational resources (4GB and 6GB RAM respectively), we again posit that the dToF depth sensor enables this improvement in virtual object stability.

Our results from the two Android smartphones, both running ARCore 1.23, also demonstrate the impact of different hardware. The Samsung, released more recently and targeted at a higher price point, outperformed the Nokia in the majority of cases. For instance, we observed higher mean drift on the Nokia after *Pause* and *Place down* than on the Samsung, 16.2cm compared to 7.4cm for *Pause* and 31.1cm compared to 4.5cm for *Place down*. These differences are likely due to a combination of greater computational resources and a higher-spec camera on the Samsung. Resolution is not a factor (both devices have 12MP cameras, and the ARCore backend appears to only use 640x480 pixel images), but the primary camera of the Samsung has two aperture modes ($f/1.5$ and $f/2.4$), which allows it to better adjust to different light levels, and the Nokia camera has



(a) Virtual object drift (log scale)

(b) Relocalization time

Fig. 5: Virtual object drift and relocalization times for five AR device and platform configurations after performing different actions. We observe significant differences between platforms and devices: for example, for *Walk away*, mean drift ranges from 2.9cm on the iPhone 13 Pro Max and 3.4cm on the Nokia 7.1 to 25.1cm on the iPhone 11 running ARKit 4.

a fixed aperture (f/1.8). The Samsung camera also has optical image stabilization, which should result in less blur and hence better feature detection, especially at low light levels.

While in general the smartphones equipped with a depth sensor were able to achieve lower drift than those without one (that ran the same AR platform), this was not always the case. In fact, these instances illustrate some of the limitations of different types of depth sensors (Section III). For example, higher drift was recorded on the Samsung Galaxy Note 10+ than the Nokia 7.1 for black or reflective surfaces (e.g., a stovetop), known to pose challenges for the iToF depth sensor [26]. We also found instances when the iPhone 13 Pro Max recorded large drift values when surfaces were too close. For *Unfocus* on the iPhone 13 Pro Max, we recorded the largest drift (5.6cm) when, while the phone was by the user's side, the environment tracking sensors were facing a surface in close proximity ($<$ 30cm). Similarly, the largest and most frequent outliers on the iPhone 13 Pro Max occurred after *Place down*; we hypothesize that while the iPhone 13 Pro Max is close to the surface it is put down on, the dToF sensor produces inaccurate depth readings, in some cases resulting in these larger errors.

Relocalization time: Relocalization time is presented in Figure 5b for *Pause* and *Place down*, the two actions which incurred an interruption to environment tracking sensor data. The comparative performance of ARCore and ARKit depended greatly on the type of interruption. Mean relocalization time after *Pause* was greater on all ARKit devices (iPhone 11 ARKit 4: 3.2s, iPhone 11 ARKit 5: 2.0s, iPhone 13 Pro Max: 2.0s), than either ARCore device (Nokia 7.1: 1.4s, Samsung Galaxy Note 10+: 1.0s). This indicates ARCore recovers tracking more quickly after an interruption to both visual and inertial input data. However, for an interruption to visual data alone (*Place down*), all ARKit devices regained tracking faster (iPhone 11 ARKit 4: 1.5s, iPhone 11 ARKit 5: 1.2s, iPhone 13 Pro Max: 1.9s) than either ARCore device (Nokia 7.1: 7.1s, Samsung Galaxy Note 10+: 4.9s). These results suggest the VI-SLAM algorithms employed on each platform take different approaches to handling the recovery of tracking.

Relocalization time was in general comparable for devices running the same platform version, especially for *Pause*. This indicates that it is the relocalization algorithm employed, rather than device hardware, which has the greatest impact on relocalization speed. An exception to this was observed however in our results for *Place down*; *when an AR platform relocalizes after the smartphone is picked up off a table, an onboard depth sensor can either speed up or slow down recovery of tracking, depending on its type*. For example, the Samsung Galaxy Note 10+, equipped with an iToF depth sensor which should provide accurate results at short distances, regained tracking much faster (4.9s) than the Nokia 7.1 (7.1s). In contrast, the iPhone 13 Pro Max, equipped with a dToF depth sensor which may provide inaccurate results at short distances, took longer to relocalize (1.9s) than the ARKit 5 iPhone 11 that does not have a depth sensor (1.2s). Similar to the large drift values that sometimes occurred for *Place down* on the iPhone 13 Pro Max, we posit that while the dToF sensor is close to the surface it is put down on, it produces inaccurate depth readings, making relocalization more challenging.

VI. Discussion

In our study of markerless smartphone AR, *we observed measurable virtual object drift for all devices and all user actions we examined*, mean drift ranging from 1.0–4.0cm for the 'easy' *Unfocus* action to as high as 25.1cm and 31.1cm for more 'challenging' actions (*Walk away* on an ARKit 4 iPhone 11, *Place down* on the Nokia 7.1). While there were noticeable improvements in a recent AR platform release (ARKit 5), and mean drift was much lower on one high-end device (the iPhone 13 Pro Max), *robustness remains a key unsolved problem, with multiple instances of drift greater than 10cm, even on the best-performing smartphones*. It is clear that in these cases virtual objects are not yet 'here to stay', and high-precision applications (e.g., medical or industrial scenarios) are not yet supported in markerless AR on smartphones.

We observed greater virtual object stability on the HoloLens 2, with mean drift of less than 2cm for all actions

and no drift exceeding 6cm. This is partly due to its specialized tracking cameras, and their wider field of view on a headset compared to a smartphone. However, the camera views on a handheld device are also inherently more challenging, and more frequently obstructed, because these devices are often held by a user's side when they move, potentially in close proximity to walls or low light areas. Further enhancements to SLAM algorithms, specific to the challenges of handheld devices, are needed to address this problem on smartphones.

Our experiments also revealed considerable differences in both drift and relocalization time between the five smartphones we tested. A variety of reasons exist for this, including different VI-SLAM algorithms employed, as well as differences in cameras, inertial sensors, depth sensors, and computational capabilities across different models. This implies that *while tools such as Unity's AR Foundation [27] facilitate cross-platform app development, virtual content stability testing needs to be conducted separately on different platforms*. Our results also demonstrate that there are instances in which a high-end smartphone (e.g., the Samsung Galaxy Note 10+) does not out-perform a mid-range one (e.g., the Nokia 7.1), due to the inherent limitations of different types of depth sensors in certain environments, or at certain distances. This indicates that developers cannot rely on testing a lower-end device to establish a minimum level of virtual object stability performance.

VII. Conclusions and Future Work

In this work we assessed the current suitability of markerless AR as an interface between humans and cyber-physical systems, through a direct quantitative comparison of virtual object stability on five smartphones, on two different AR platforms, as well as the Microsoft HoloLens 2. In systematic experiments, we show that while the specialized cameras and form factor of headsets facilitate a high degree of spatial stability, robustness remains a key unsolved issue on smartphones.

Beyond the effects of device hardware and motion, in future we will examine how environment characteristics impact virtual object stability, and VI-SLAM performance in general, by studying more environments, and introducing characterization metrics, which we explored in preliminary work [28]. We will also investigate further performance metrics for virtual object stability, by analyzing the distributions of drift and relocalization time. Alongside experiments on real devices, we will develop simulation-based evaluation methods, using open-source VI-SLAM algorithms in virtual environments, to support systematic environment alterations. We will demonstrate how knowledge of the relationship between environment characteristics and virtual object stability can be used to improve AR experiences in practice, such as by enhancing the visual environment using Internet of Things lights and displays.

Acknowledgments

This work was supported in part by NSF grants CSR-1903136 and CNS-1908051, NSF CAREER Award IIS-2046072, NSF CAREER Award CNS-1942700, the Academy of Finland grant number 338854, and an IBM Faculty Award.

References

[1] C. Liu, S. Cao, W. Tse, and X. Xu, "Augmented reality-assisted intelligent window for cyber-physical machine tools," *Journal of Manufacturing Systems*, vol. 44, pp. 280–286, 2017.
[2] L. Qian, A. Deguet, Z. Wang, Y.-H. Liu, and P. Kazanzides, "Augmented reality assisted instrument insertion and tool manipulation for the first assistant in robotic surgery," in *IEEE ICRA 2019*, 2019.
[3] Niantic, "Pokemon GO app," https://www.pokemongo.com/, 2022.
[4] IKEA, "IKEA Place app," https://www.ikea.com/us/en/customer-service/mobile-apps/, 2022.
[5] Google, "ARCore," https://arvr.google.com/arcore/, 2022.
[6] Apple, "ARKit," https://developer.apple.com/augmented-reality/, 2022.
[7] Google, "ARCore TrackingState," https://developers.google.com/ar/reference/java/com/google/ar/core/TrackingState, 2022.
[8] Apple, "ARKit ARCameraTrackingState," https://developer.apple.com/documentation/arkit/arcamera/trackingstate, 2022.
[9] X. Ran, C. Slocum, M. Gorlatova, and J. Chen, "ShareAR: Communication-efficient multi-user mobile augmented reality," in *ACM HotNets 2019*, 2019.
[10] R. Vassallo, A. Rankin, E. C. S. Chen, and T. M. Peters, "Hologram stability evaluation for Microsoft HoloLens," in *SPIE Medical Imaging 2017: Image Perception, Observer Performance, and Technology Assessment*, 2017.
[11] C. Slocum, X. Ran, and J. Chen, "RealityCheck: A tool to evaluate spatial inconsistency in augmented reality," in *IEEE ISM 2021*, 2021.
[12] T. Scargill, J. Chen, and M. Gorlatova, "Here to stay: Measuring hologram stability in markerless smartphone augmented reality," *arXiv preprint arXiv:2109.14757*, 2021.
[13] OptiTrack, "OptiTrack," https://optitrack.com/, 2022.
[14] Vicon, "Vicon," https://www.vicon.com/, 2022.
[15] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The EuRoC micro aerial vehicle datasets," *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1157–1163, 2016.
[16] D. Schubert, T. Goll, N. Demmel, V. Usenko, J. Stückler, and D. Cremers, "The TUM VI benchmark for evaluating visual-inertial odometry," in *IEEE/RSJ IROS 2018*, 2018.
[17] L. Jinyu, Y. Bangbang, C. Danpeng, W. Nan, Z. Guofeng, and B. Hujun, "Survey and evaluation of monocular visual-inertial SLAM algorithms for augmented reality," *Virtual Reality & Intelligent Hardware*, vol. 1, no. 4, pp. 386–410, 2019.
[18] T. Feigl, A. Porada, S. Steiner, C. Löffler, C. Mutschler, and M. Philippsen, "Localization limitations of ARCore, ARKit, and HoloLens in dynamic large-scale industry environments," in *VISIGRAPP 2020*, 2020.
[19] N. Yang, R. Wang, X. Gao, and D. Cremers, "Challenges in monocular visual odometry: Photometric calibration, motion bias, and rolling shutter effect," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 2878–2885, 2018.
[20] M. K. Paul, K. Wu, J. A. Hesch, E. D. Nerurkar, and S. I. Roumeliotis, "A comparative analysis of tightly-coupled monocular, binocular, and stereo VINS," in *IEEE ICRA 2017*, 2017.
[21] R. Horaud, M. Hansard, G. Evangelidis, and C. Ménier, "An overview of depth cameras and range scanners based on time-of-flight technologies," *Machine vision and applications*, vol. 27, no. 7, pp. 1005–1020, 2016.
[22] Z. Zhu, F. Xu, M. Li, Z. Wang, and C. Yan, "Challenges from fast camera motion and image blur: Dataset and evaluation," in *ECCV 2020*, 2020.
[23] P. Hübner, K. Clintworth, Q. Liu, M. Weinmann, and S. Wursthorn, "Evaluation of HoloLens tracking and depth sensing for indoor mapping applications," *Sensors*, vol. 20, no. 4, p. 1021, 2020.
[24] Apple, "Human Interface Guidelines: Augmented Reality," https://developer.apple.com/design/human-interface-guidelines/ios/system-capabilities/augmented-reality/, 2022.
[25] Google, "ARCore design guidelines: Environment," https://developers.google.com/ar/design/environment/definition/, 2022.
[26] M. Hansard, S. Lee, O. Choi, and R. Horaud, *Time-of-flight cameras: Principles, methods and applications*. Springer, 2012.
[27] Unity, "AR Foundation," https://unity.com/unity/features/arfoundation, 2022.
[28] T. Scargill, S. Hurli, J. Chen, and M. Gorlatova, "Demo: Will it move? Indoor scene characterization for hologram stability in mobile AR," in *Proceedings of ACM HotMobile 2021*, 2021, demo video available at https://sites.duke.edu/timscargill/sceneit-prototype/.