RealityCheck: A Tool to Evaluate Spatial Inconsistency in Augmented Reality

Carter Slocum, Xukan Ran, Jiasi Chen
Department of Computer Science and Engineering
University of California, Riverside
Riverside, CA, USA

***Abstract*—In augmented reality (AR), virtual objects can drift away from their original intended locations, significantly impairing a user's experience. Traditionally, a virtual object's drift is approximated by the device localization drift, which is measured using specialized hardware such as 3D scanners or laser-based positioning systems. However, with AR rapidly becoming more popular, there is a need for a lightweight, software-based approach to evaluate the drift of virtual objects. This software should be easy for researchers and developers to use, without requiring specialized hardware or extensive environment setup.**

Towards this, this paper presents RealityCheck, an open-source AR evaluation tool that reports the drift of AR virtual objects in the world coordinate system, requiring only paper printouts and minimal modifications to the AR app. RealityCheck is designed to measure the drift of a virtual object across time of a single user, as well as the positioning differences of the same virtual objects as seen by multiple users. Our prototype is implemented on an Android smartphone running the ARCore platform, and evaluated in indoor and outdoor scenarios under a variety of user mobility patterns with traces of different lengths. We compared the results of RealityCheck with the ground truth position of the virtual object, and showed that RealityCheck matches the ground truth within 1.5 cm on average.

***Index Terms*—augmented reality, spatial drift, evaluation tool**

I. Introduction

Mobile Augmented Reality (AR) is becoming increasingly popular, with the AR market estimated to grow to $61 billion by 2023 [1]. Many companies are developing AR platforms and integrating AR into their products. For example, Apple announced its mobile AR platform, ARKit, in 2017 and Google introduced ARCore for Android in 2018 [2], [3]. IKEA has developed a virtual furniture placement app to let users visualize virtual furniture at home [4].

In AR, virtual objects are rendered on the display and overlaid on top of a user's field of view (FoV). To provide a seamless integration with the real world, the AR app needs to have an understanding of the surrounding real-world environment [5]; for example, in order to place a virtual cup on a real table, rather than drawing the cup unrealistically floating in the air. AR algorithms to understand the environment can be categorized into three types: (1) object detection based AR (*e.g.*, Snapchat Lenses [6]) in which machine learning or computer vision is used to classify objects in the real world and overlay on top of them; (2) SLAM based AR (*e.g.*, Just A Line [7]) in which visual-inertial sensors are used to create a 3D map of the real world; and (3) marker based AR, which relies on fiducial markers placed in the scene.

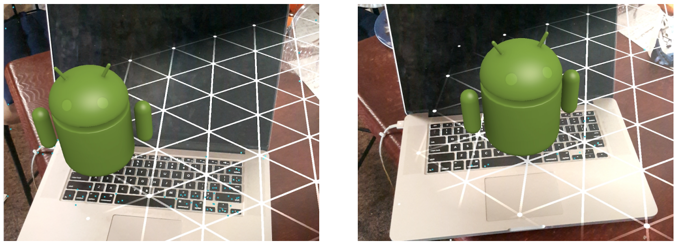

(a) Virtual object at time 1. (b) Virtual object at time 2.

Fig. 1: Example of a virtual object (Android character) inadvertently drifting to the right over time.

However, an AR app's understanding of the environment can be sometimes wrong or inconsistent, particularly if an AR user moves, and the virtual object may unintentionally drift away from its original position. This can significantly disrupt the connection between the virtual object and the real world and thus impair user experience. Fig. 1 shows an example from Android ARCore, where the virtual character drifted to the right on top of the keyboard between time 1 and time 2. A related problem happens when multiple co-located users participate in a shared AR experience and the virtual objects drift across users, appearing to have different locations in the displays of each user [8]. In our experience, such drifts manifest in popular mobile AR platforms, such as Google ARCore and Apple ARKit.

To improve AR applications, it is necessary to measure and evaluate the spatial drift of these virtual objects. However, current methods of doing so suffer from several limitations. Subjective evaluation through user questionnaires is time-consuming and cannot provide real-time feedback or exact quantitative numbers. Objective methods to measure spatial drift are time-consuming, rely on specialized hardware, and/or only work in constrained testing environments. For example, manual labeling of virtual object's position in the scene is time-consuming, requiring multiple seconds for a human to label every single frame when the AR app is running. The most relevant comparison is probably to SLAM performance evaluation, which typically uses Absolute Trajectory Error (ATE) [9], but which suffers from the following limitations. Firstly, ATE measures the difference between the *device's* estimated location and the ground truth location, which is different from the location drift of the *virtual object* (as shown in Section II). Secondly, computing ATE requires knowledge of the ground truth device location, which requires special equipment such as a 3D scanner or motion capture system that only works in a designated testing area, or offline datasets [10] that may not exemplify typical AR use cases. Thirdly, factors such as latency or graphical effects that can affect the final rendered AR object take place after the estimated device trajectory has been recorded, and thus are not accounted for by ATE. Finally, ATE is designed to measure performance of SLAM for an individual user, and does not provide information in the multi-user scenario. Such requirements pose great inconvenience to researchers who wish to experiment with and evaluate AR.

To address these issues, in this paper we propose an evaluation tool, called RealityCheck, for SLAM-based AR to directly and conveniently measure the drift of virtual objects in both single-user and multi-user cases. We propose a methodology that does not require specialized hardware or special lab testing environments, making AR evaluation easier and more accessible to general engineers and computer scientists. Our key idea is to temporarily replace the virtual object in the AR app with a virtual marker (*e.g.*, an ArUco marker), and use more accurate computer vision techniques (*i.e.*, PnP) to localize the virtual object/marker in 3D space. By recording the location of the virtual object over time and across users, RealityCheck can compute the spatial drift seen by a single user, or the spatial inconsistency of a virtual object seen by multiple users. We envision such a tool being used by researchers and developers to receive feedback from their AR apps on spatial drift, paving the way for corrections to be made. We focus on SLAM-based AR because it is the foundation of commercial off-the-shelf AR systems, such as Google ARCore, Apple ARKit, and Microsoft Hololens, although the methodology can be extended to other types of AR. We also focus on positioning errors as opposed to rotation errors because they tend to be larger in SLAM-based systems [11].

Overall, RealityCheck makes the following contributions:

- RealityCheck directly measures the spatial drift/inconsistency of a virtual object through a lightweight software-based approach that does not require specialized hardware or testing environments. It only needs a printed marker board, the device camera calibration parameters, and minimal changes to the AR app. This is contrast to measuring the ATE of the device, as typically done in SLAM evaluation, which cannot directly measure a virtual object's position.
- RealityCheck works in both single-user and multi-user scenarios. In a single-user scenario, RealityCheck evaluates the spatial drift of a virtual object over time. In a multi-user scenario, RealityCheck evaluates the spatial inconsistency of a virtual object when viewed by multiple users with different FoVs. The key idea in our methodology is to temporarily replace a virtual object with a virtual marker, and use computer vision techniques to accurately track the position of virtual object with respect to the real world.
- RealityCheck achieves 1.5 cm estimation error on average across 22 total trials, compared to the ground truth position of a virtual object. We perform these trials under various experimental conditions: indoor and outdoor environments, changing visibility of the virtual object, different user mobility patterns, and app usage length ranging from 30 seconds to 2.5 minutes.

The open-source code of RealityCheck and a video demonstration of its operation are provided through a website [12]. In the remainder of this paper, we discuss motivation (Section II), the design of RealityCheck (Section III), quantitative evaluations (Section IV), discussion (Section V), related work (Section VI), and finally conclude (Section VII).

II. Motivation: Issues with Manual Labeling and Absolute Trajectory Error

In this section, we provide insight into why manual labeling or ATE are insufficient for measuring drift of an AR virtual object. Note that neither manual labeling nor ATE measurements are needed for casual users of RealityCheck, but are only needed to evaluate RealityCheck in this paper.

Manual Labeling: In the case of manual labeling, in our experience, it took approximately 10-30 seconds to hand-annotate the position of a virtual object in each frame. For an AR app updating its display at 30 frames per second (FPS) running for 5 minutes, this could take up to 1.25 hours to measure a virtual object's drift for the duration of a user's experience. Clearly, this is infeasible and unwieldy, particularly if multiple users are participating in the AR experience and each of their frames need to be annotated.

Absolute Trajectory Error: In the case of ATE, as mentioned in Section I, ATE does not provide sufficient information about the position of the *virtual object*, since the device trajectory and ATE only have information about the position of the *device*. The device position is insufficient knowledge about the virtual object, because rendering the virtual object involves projecting the virtual object onto the AR display, which requires both device position and rotation provided by SLAM. Therefore, ATE alone cannot tell the AR device where the virtual object is rendered on the display, and hence what its spatial drift is.

We next describe an experiment we conducted to illustrate why ATE cannot be used to evaluate the spatial drift of an AR virtual object; *i.e.*, why ATE or the device trajectory does not accurately capture the spatial drift of an AR virtual object. This experiment is illustrated in Fig. 2. We use ARCore as the AR platform. We started the experiment by creating a virtual object (the tower of shapes) on the floor, and then move backward (in the y direction), without any left/right movement (in the x direction). The height of the device is also fixed so there is also no up and down movement (z direction). In the 3D plot of Fig. 2, we plot the SLAM-estimated device trajectory (blue

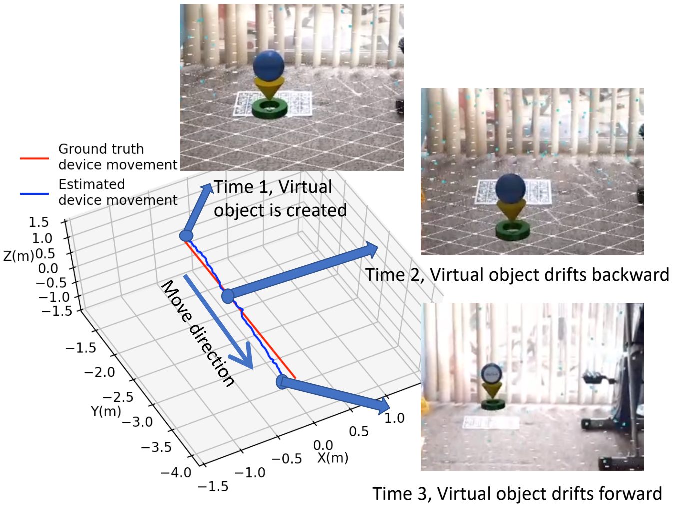


Fig. 2: The virtual object drifts back and forth as the user moves. However, simply looking at the device trajectory/ATE alone gives little information about how the virtual object drifts.

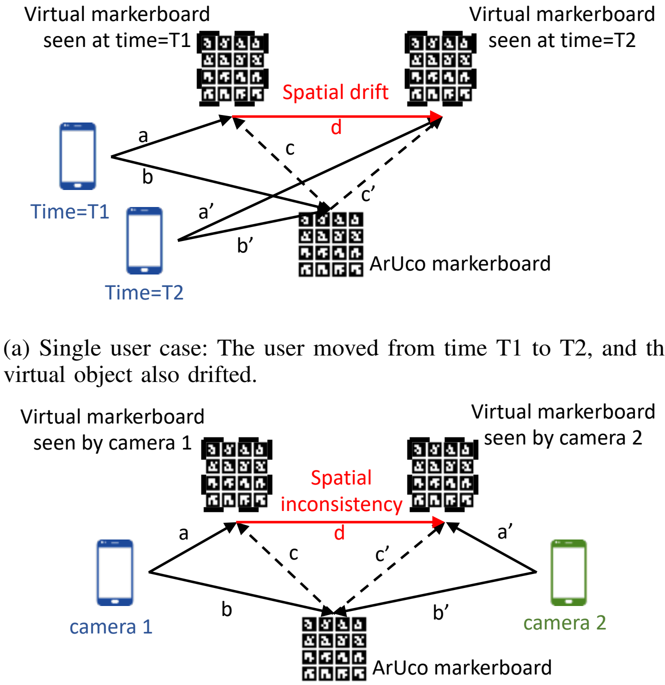
line), which is a straight line in the XY plane. The SLAM-estimated trajectory matches well with the ground truth device trajectory (red line). Intuitively, we might expect this accurate device position estimate to mean the virtual object won't drift.

However, the SLAM-estimated device trajectory gives us no information about the position of the virtual object on the display, and the virtual object does in fact drift, despite the accurate device position estimates. In Fig. 2, we show screenshots of the virtual object at 3 different times. At time 1, the virtual object is directly above the piece of paper. At time 2, the virtual object drifts backward from the paper (towards the user), and at time 3 it drifts forward (away from the user). We don't know where the virtual object is or how much the virtual object drifts by looking at the device trajectory alone, until we see the screenshots of the virtual object. In other words, the accuracy of the device trajectory estimation is not tightly correlated with the drift of the virtual object.

Moreover, in the multi-user scenario, each AR app can update its virtual object position, and then our tool can compute the position difference (spatial inconsistency) of the virtual object between any two users. However, we can't compute the spatial inconsistency across multiple users just from ATE or the trajectory because of the lack of time synchronization and rotation information from ATE alone.

III. Design of RealityCheck

The key idea behind the design of RealityCheck is that an AR virtual object can be any object that can be rendered to a screen. Thus it is possible to render a known, easy to detect object into the scene, such as an ArUco marker board [13], and accurately determine its location using computer vision techniques. Combined with a real marker board placed in the scene, these known objects allow measurement of the spatial consistency between the real world and the virtual objects rendered onto the screen. Fig. 3 illustrates the design of RealityCheckfor the single- and multi-user scenarios. In both cases, we place a real marker board in the physical world and create a virtual marker board as the virtual object for rendering. We next describe this general idea in further detail for each of the scenarios.



(a) Single user case: The user moved from time T1 to T2, and the virtual object also drifted.

(b) Multiple users case: Two users view a common virtual object, but see it at different positions with respect to the real world.

Fig. 3: Design of RealityCheck.

Single user scenario: The spatial drift in the single user scenario is determined as follows. As labeled on Fig. 3a, we define the vector from the device's physical position to the virtual marker board's designated physical position at time T1 as a, the vector from the device to the real marker board at time T1 as b, and the vector from the real marker board to the virtual marker board at time T1 as c. At time T2, due to registration errors in the AR app, the virtual marker drifted to another physical location. At time T2, we label the corresponding vectors a', b', c', analogous to a, b, c. Then to compute the spatial drift (the vector d in Fig. 3a), we can see that $d = c' - c$. Since $c = a - b$ and $c' = a' - b'$, we have $d = a' - b' - a + b$. The vectors a, a', b, b' can be relatively easily computed using the perspective-n-point (PnP) method to determine the pose of the camera with respect to the virtual/real marker boards. This method obtains the detected corners of the marker boards in the current FoV, and along with knowledge of their size and shape and the camera calibration matrix, solves the PnP problem to obtain the desired vectors a and b for the respective marker boards. Finally, we compute $d = a' - b' - a + b$, which is the spatial drift. The complete algorithm is summarized in Alg. 1.

As an illustration of our approach, we plot the vectors a and

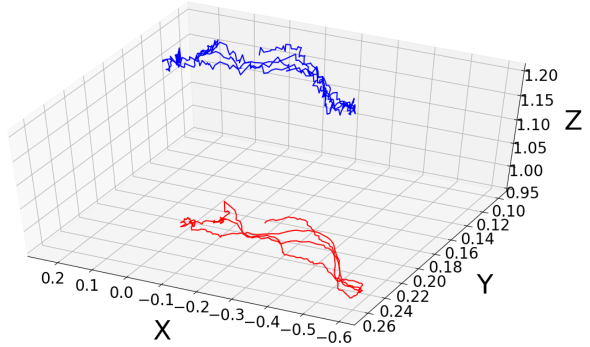


Fig. 4: Example paths of the real/virtual marker boards over time (meters), relative to a moving camera. The red (blue) line represents the position of the real (virtual) marker board over time, vector b (a). Any change in their difference (c) is the spatial drift.

b in Fig. 4 from a real trace captured by RealityCheck. The blue line represents the position of the virtual marker board (a) over time with respect to the camera. The red line represents the position of the real marker board (b) over time with respect to the camera. Their difference $c = b - a$ should ideally remain constant over time even as the camera moves, meaning that the real and virtual marker boards remain fixed with respect to each other, and there is no spatial drift. However, in practice there is spatial drift, which is reflected as changes in c. We see this in Fig. 4: while the general trajectory of the two lines are similar, they are not identical – their differences are the spatial drifts we are interested in.

Multi-user scenario: The multi-user scenario, shown in Fig. 3b, is analogous to the single user scenario, except that drift is measured across space, rather than across time. a is the defined as the vector from the phone 1's camera to the virtual object, and a' is defined as the vector from phone 2's camera to the same virtual object. b is defined as the vector from phone 1's camera to the real marker board, and b' is defined analogously for phone 2. Then $c = a - b$ is the vector from the real marker board to the virtual marker board as seen by device 1, and $c' = a' - b'$ is the vector from the real marker board to the virtual marker board as seen by device 2. Their difference, $d = c' - c$, represents the spatial inconsistency, *i.e.*, the difference in the position of the virtual object as seen by the two devices. We follow the same computation method using PnP as in the single-user case. Thus, the same overall method is general enough to be used for the single and the multi-user scenarios.

IV. Experimental Evaluation

A. Implementation and Test Methodology

1) Implementation

A picture of our implementation is shown in Fig. 5, with the major components described below.

Devices. The AR test application is built on the Android ARCore platform [3] and run on a Samsung Galaxy S20 mobile phone. RealityCheck itself runs on a separate laptop with an Intel Core i7 2.8Ghz CPU.

Algorithm 1: Computation of spatial drift

```
Inputs: camera frame $f_1$ and $f_2$;
Outputs: spatial drift $d$;
Compute $a$ and $b$ in $f_1$ using PnP;
$c \leftarrow a - b$;
Compute $a'$ and $b'$ in $f_2$ using PnP;
$c' \leftarrow a' - b'$;
Return $d \leftarrow c' - c$;
```

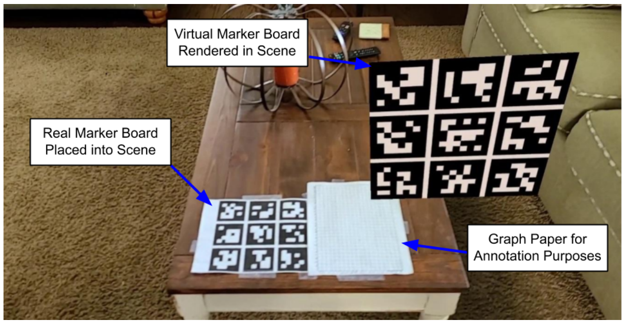

Virtual Marker Board Rendered in Scene
Real Marker Board Placed into Scene
Graph Paper for Annotation Purposes

Fig. 5: Evaluation setup. The position of the virtual object (the virtual marker board) is evaluated with respect to the real marker board on the table. The graph paper aids in hand-annotating the ground truth position of the virtual object.

Real marker board. An ArUco marker board comprised of 9 unique markers arranged in a 3×3 grid is printed on standard printer paper. The real marker board is 0.2 m × 0.2 m in our experiments so it could fit on the paper, but in general, it may be of any size small enough to fit in the FoV of the device camera, and large enough for the marker patterns to be recognized.

Virtual object (marker board). We create our virtual marker board similarly to the real marker board. The virtual board is textured using 9 ArUco markers (different from the 9 in the real marker board) and arranged in a 3x3 grid. The virtual marker board is created in Blender [14] and imported into Android Studio. We set the size of virtual marker as 0.4 m × 0.4 m. It is important that the virtual marker board is not so large as to obstruct the real marker board, although this could be solved by simply recording the image both before and after the virtual marker board is drawn to the screen. Special attention is paid to make sure no additional graphics effects are drawn, such as drop shadows or specular maps, as they may affect the visibility of the real or virtual markers.

Graph paper. To compare RealityCheck's measurements to the ground truth, a piece of paper of the same dimensions as the real marker board is placed adjacent to it in the scene. This paper has printed on it a grid of 1 cm × 1 cm squares for use in hand annotating the position of the virtual object offline (as described in the next subsection).

2) Test Methodology

AR app recording. The AR test application is run on the mobile device, and the user taps the screen to place the virtual marker board in the scene. The user then moves around the environment and captures what is shown on the display, either using screen capture software [15] or simply taking screenshots at regular intervals to obtain the data needed to run RealityCheck.

Mobility patterns. To examine the efficacy of RealityCheck with respect to user mobility, we evaluated three different walking patterns.

- *Side-to-side:* The user moved side to side a distance of approximately 1.5 meters without rotating, keeping the real marker board and the virtual object in the device's FoV.
- *Look-Away-and-Back:* The user started the trial with the real marker board and virtual object in the FoV, then walked away about 15 meters keeping them out of the FoV, and finishing the trial by returning to them.
- *Around-in-a-Circle:* The user walked a complete 1.15 meter radius circle around the real marker board and virtual object, keeping them in view.

We conducted 22 total trials. First, we conducted 5 indoor trials of each mobility pattern, for a total of 15 trials. Each trial lasts for 12-36 seconds. These indoor trials were performed without direct sunlight with the real marker board and graph paper placed on a short table. An additional 4 trials were performed in an outdoor environment in direct sunlight with one of each mobility pattern, plus an additional circular walk trial. Finally, 3 trials lasting 2.5 minutes each were performed in the same indoor environment with the 3 mobility patterns.

Running RealityCheck. To process the results, the recorded video is partitioned into its individual frames. RealityCheck performs the steps described in Section III for each frame or frame pair, and saves the resulting c vectors to a file for analysis.

Ground truth via manual labeling. Finally, in order to check the results of RealityCheck against the ground truth, it is necessary to hand annotate the position of the virtual object (specifically, the virtual marker board relative to the real marker board). The graph paper assists in this by allowing easy visual counting of the distance to the virtual marker, with a resolution of 0.5 cm. For every frame we wish to evaluate, we measure the Euclidean coordinates in terms of grid squares, then convert the grid squares to cm in order to obtain the ground truth vector between the virtual and real marker boards, which we call c_{true}. This is then compared against the estimated c vectors output by RealityCheck. We then repeat this measurement over multiple frames to compute spatial drift d_{true}. A similar approach is followed for the multi-user scenario. Note that this manual labeling is only performed by us to evaluate RealityCheck, and does not need to be performed by general users of the tool.

Using this method, we hand annotate every 30 frames ($\sim$1 s) across the 15 indoor trials, resulting in a total of 191 annotated frames, plus 77 annotated frames for the outdoor trials. Since the hand-annotated virtual object positions are in the real marker board coordinate system, but RealityCheck's output is in camera space, to compare their results, a transformation (obtained from the PnP method) is performed to convert RealityCheck's output from camera space to real marker board space, in order to make the vectors comparable.

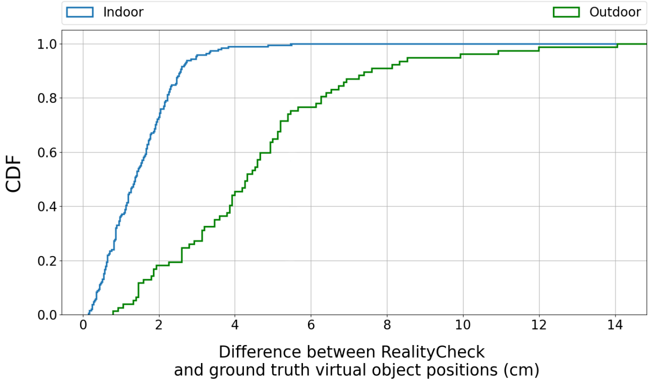


Fig. 6: Cumulative Distribution Function of the difference between RealityCheck's estimated position of the virtual object and the ground truth ($||c - c_{\text{true}}||$) of a single user.

B. Evaluation Results

In this section, we discuss RealityCheck's performance in terms of how accurately it reports the drift of a virtual object, compared to the ground truth.

Position of the virtual object. We first report the distance between where RealityCheck reports the virtual object is, versus the ground truth ($||c - c_{\text{true}}||$ in the notation of Section III). A low distance indicates that RealityCheck matches more closely with the ground truth. The Cumulative Distribution Function (CDF) of the distances for the indoor and outdoor trials are shown in Fig. 6. The mean of the indoor errors is 1.5 cm, whereas the outdoor errors average at 4.6 cm. The median distance in the indoor scenarios is 1.36 cm, and 90% of the samples fall within 2.5 cm. RealityCheck tends to perform slightly worse outdoors, possibly due to noise, and were biased by one particular trial that performed particularly badly. For example, outdoors, the lighting changes more frequently than indoors, or a strong wind shaking leaves on a tree can break the static environment assumption of SLAM-based AR, giving rise to larger virtual object drift and resulting in a slightly larger difference between RealityCheck's estimate and the ground truth. Overall, though, the distance between the ground truth estimates and RealityCheck's estimates are 2.4 cm on average across all indoor and outdoor scenarios, suggesting that RealityCheck is reporting accurate results.

RealityCheck can accurately report the position of a virtual object both when the virtual object drifts slightly, or when it jumps significantly. To illustrate this, in Fig. 7 we show an example time series of the distance between the virtual object and the real marker board, as output by RealityCheck (blue dots, $||c||$) and by the ground truth hand annotations (red dots, $||c_{\text{true}}||$). The point of interest is just past the 455th frame, where the virtual object “jumps” nearly 6 cm, as a more extreme example of spatial drift. RealityCheck is able to detect the

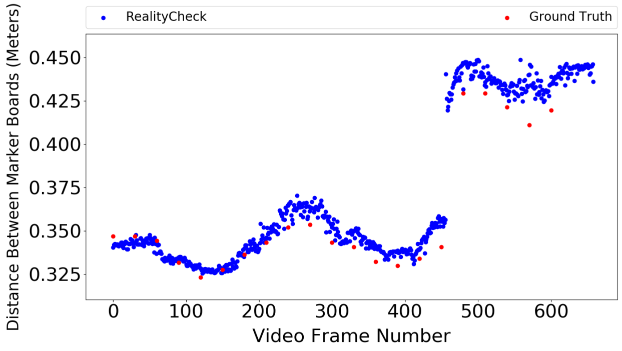


Fig. 7: Example time series of the distance of the virtual object from the real marker board, as estimated by RealityCheck (blue, $||c||$) and the ground truth (red, $||c_{\text{true}}||$). RealityCheck is able to capture the large "jump" of the virtual object between frame 455 and 456.

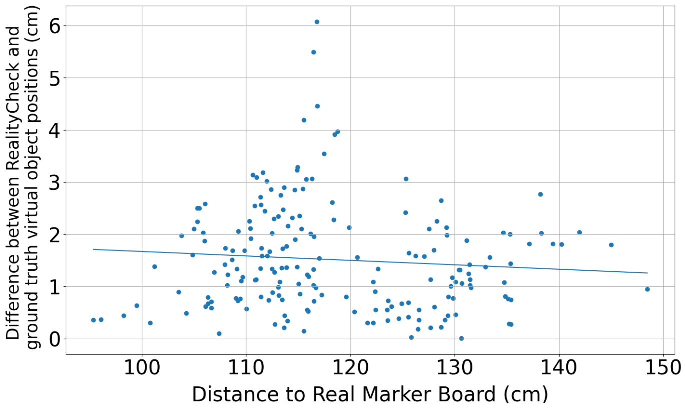


Fig. 8: Distance error ($||c - c_{\text{true}}||$) as a function of a user/camera's distance to the real marker board. RealityCheck works even when the camera is further away.

virtual object's sudden change in position quite accurately (as the blue dots line up with the red dots).

Finally, we ask whether RealityCheck's output has low error even as the camera moves farther away from the virtual object and real marker board. In Fig. 8, we plot the difference between RealityCheck and the ground truth's estimate of the virtual object's position ($||c - c_{\text{true}}||$), as a function of the distance between the camera and the real marker board. The trend in Fig. 8 is essentially flat, indicating that RealityCheck is robust to distance from the virtual object within typical AR application ranges.

Position drift over time. We are not only interested in RealityCheck's ability to estimate the location of a virtual object at a single point in time, but also in its ability to track the change in that position over time. To evaluate this, we define the drift as the distance that a virtual object moves during one second (d). We also compute the ground truth drift (d_{true}). Fig. 9 shows the difference in the drift measurement from RealityCheck and the ground truth ($||d - d_{\text{true}}||$), for each user mobility pattern. Overall, the average difference across all

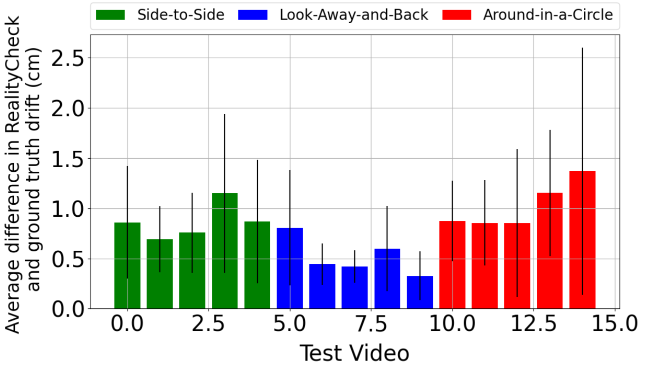


Fig. 9: Average difference between RealityCheck and the ground truth, with standard deviation, when computing the per-second drift of a single user ($||d - d_{\text{true}}||$).

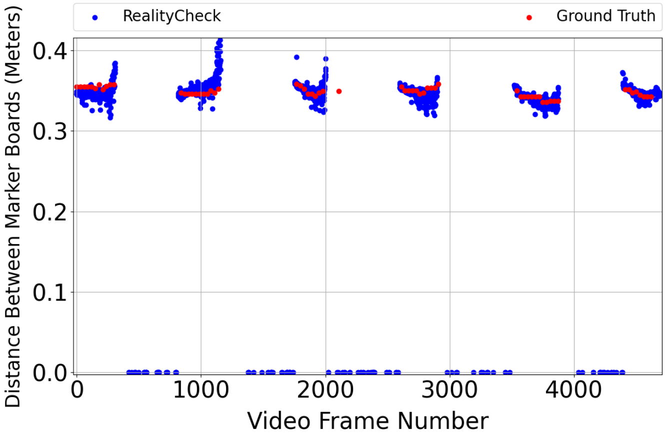


Fig. 10: Example time series from a longer 2.5-minute trial of the distance of the virtual object from the real marker board, as estimated by RealityCheck (blue, $||c||$) and the ground truth (red, $||c_{\text{true}}||$) for a single user.

trials was 0.86 cm. The trials where the user moved and looked away from the virtual object and returned later (look-away-and-back) had the least drift difference (0.52 cm on average), because there was usually only one large drift as the user returned, but nearly no drift for the rest of the video, and hence little drift difference. On the other hand, in the trials where the user faced the virtual object from different angles (side-to-side), the average drift difference was higher (0.87 cm). The trials where the user moved in a complete circle around the markers (around-in-a-circle) experienced the most extreme angles and, correspondingly, the greatest drift error (1.02 cm on average).

Longer length trials. To evaluate performance over longer runs of the AR app, to see if the tool performs correctly, 3 additional 2.5-minute trials were performed, using the same three movement strategies as in the shorter videos. Overall, the results were similar to those of the shorter-length trials discussed so far. Fig. 10 shows an example time series of one of the longer-length trials. The y-axis is the distance between

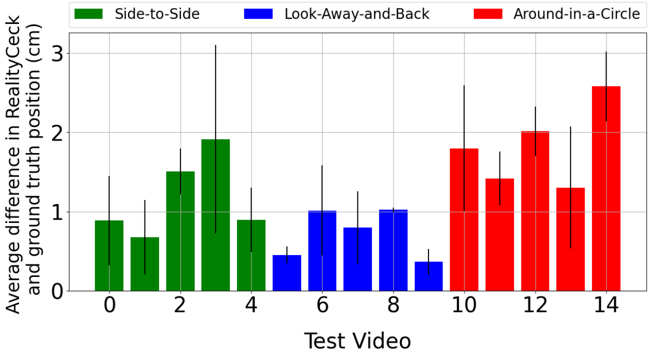


Fig. 11: Average difference between RealityCheck and ground truth, with standard deviation, when measuring the position of a virtual object as viewed by two emulated users ($||d - d_{\text{true}}||$).

the virtual object and the real maker board, as output by RealityCheck (blue dots, $||c||$) and the ground truth (red dots, $||c_{\text{true}}||$. Despite the virtual object coming in and out of view, RealityCheck is capable of maintaining accuracy even during longer AR sessions, as RealityCheck and the ground truth match up well. The average difference between the vectors received from RealityCheck and the ground truth was 1.6cm across all the longer-length trials.

Spatial inconsistency across multiple devices. RealityCheck is able to take in any recording of an AR app along with the board configurations and camera parameters and perform the estimations, regardless of when or what device the images came from, as long as the image shows both marker boards and the camera calibrations are known. To emulate a multi-user scenario, we consider the first half of a video trace as originating from one device, and the second half as originating from a second device, then compare the drift across these two halves of the video. Fig. 11 shows the average difference in the position of the virtual object as measured by RealityCheck and the ground truth, as seen by each of our two emulated users. The difference is 1.3 cm on average. As the change in the virtual object's position is larger in this scenario than in the 1 second scenario previously considered (Fig. 9), we therefore see the average error per video increase accordingly, but stay within a reasonable 1-3 cm range.

V. Discussion

In this section, we briefly discuss several assumptions of RealityCheck. Firstly, RealityCheck relies on marker detection and pose estimation (via PnP), which have been shown to have good accuracy [16]. We also use a marker board, which consists of 9 markers, to increase the detection accuracy [17]. RealityCheck can fail to output measurements if visibility is poor; for example due to insufficient lighting, motion blur, poor resolution, or extreme viewing angles of the virtual object (*e.g.*, 90° to the side, causing the virtual marker board is too thin to be detected).

Secondly, the addition of a marker board to the scene, as required for RealityCheck to work, can add additional features to the scene for the AR system to use, thus impacting the spatial drift of a virtual object. However, since the amount of added features can be quite small compared to the natural visual features available elsewhere in the scene, we believe the effect on the AR application to be small, if any. Moreover, our experiments show that even with these few additional features, the AR app still experiences spatial drift (see Fig. 7).

Finally, while all tests were performed using SLAM-based AR on mobile devices, RealityCheck generalizes beyond SLAM-based AR, because it only needs the video of the AR app's operation, camera calibration parameters, and known markers as inputs. Recording screenshots of a non-SLAM AR framework (a machine learning-based framework, for example), will not prevent RealityCheck from measuring the spatial consistency of the virtual objects. RealityCheck is agnostic to the internals of the AR platform, and only needs the final rendered images to run successfully.

VI. Related Work

Specialized hardware: Yagfarov *et al.* [18] evaluate SLAM performance against results from a laser tracker in a static indoor environment. However, such method requires extra equipment(laser), and can't evaluate camera rotation, whereas our method can work in indoor, outdoor, or even dynamic environments provided that the ArUco marker in our system remain stationary. Other evaluation methods include constructing a 3D model of the real scene, requiring a 3D scanner [19].

Pixel differences: Several works [20], [21], [22] measure the pixel difference between where the virtual object is and where it should be on the device screen (*e.g.*, screen-space error). We note that RealityCheck also projects the virtual object to the device screen, and could calculate the drift in terms of pixels. However, pixel drifts can tell us how much drift the virtual objects have on the device screen, but they can have significantly different meanings in 3D space.

Statistical analysis: Faion *et al.* [23] compute camera translation and rotation multiple times using different set of markers, and then use the standard deviation of the results as the estimation reliability. However, such methods provide only camera pose but not the drift of virtual objects. MacIntyre *et al.* [24] propose a statistical method to estimate the error bounds of 3D points and then calculate the mean and covariance of the drift in the screen coordinates. However, this requires knowledge of the individual sensor errors, which can be difficult to obtain for heterogeneous AR hardware. Ran *et al.* [25] design a marker-based method using mobile devices, but require modification to the AR app, while our method requires only screenshots and camera parameters. Scargill *et al.* [26] develop an alternative methodology requiring more extensive user interaction.

User participation: Some AR evaluation methods involve user participation, rather than the objective feedback studied in this paper. Peillard *et al.* [27] and Rosales *et al.* [28] measure the difference between the distance the user perceives from the device, and the distance designed by the program. While RealityCheck is not evaluating the perceived position by users, the drift of virtual object we measure will affect the perception of the observer. Lehman *et al.* [29] and Bork *et al.* [30] evaluate AR performance by asking users to give feedback or fill out questionnaires. Such methods can provide direct feedback from users, which is complementary to our approach, but can be time-consuming.

VII. Conclusions

Tools to measure spatial inconsistency and other metrics of interest are necessary for AR to improve. RealityCheck is an accurate, fast, and cheap way to check the spatial inconsistency of an AR system. Our evaluation of RealityCheck showed that it can measure the spatial drift of a virtual object with 1.5 cm error on average, compared to the ground truth. We release the code as open-source in the hope that it will be useful to other researchers and developers. In the future, RealityCheck could be extended into a suite of tools to measure additional AR-relevant metrics such as appropriate rendering of virtual objects in different real world lighting conditions, appropriate shadow placement, and proper occlusion of virtual objects.

Acknowledgements

This work is supported in part by NSF CAREER 1942700 and a U.S. Department of Education GAANN fellowship.

References

[1] G. Goswami, "Council Post: Augmented Reality's Applications And Future In Business," https://www.forbes.com/sites/forbescommunicationscouncil/2020/10/15/augmented-realitys-applications-and-future-in-business, year=2020.

[2] Apple, "Arkit - apple developer," https://developer.apple.com/arkit/.

[3] Google, "Arcore overview," https://developers.google.com/ar/discover/.

[4] IKEA, "Ikea place," https://apps.apple.com/us/app/ikea-place/id1279244498.

[5] R. L. Holloway, "Registration error analysis for augmented reality," *Presence: Teleoperators & Virtual Environments*, vol. 6, no. 4, pp. 413–432, 1997.

[6] Google, "Snapchat lenses," https://www.snapchat.com/.

[7] Google Creative Labs, "Just a Line - Draw Anywhere, with AR," https://justaline.withgoogle.com/.

[8] K. Apicharttrisorn, B. Balasubramanian, J. Chen, R. Sivaraj, Y.-Z. Tsai, R. Jana, S. Krishnamurthy, T. Tran, and Y. Zhou, "Characterization of multi-user augmented reality over cellular networks," in *IEEE SECON*, 2020.

[9] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *IEEE/RSJ IROS*, 2012.

[10] 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.

[11] 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.

[12] "Realitycheck project website," https://sites.google.com/view/arrealitycheck/home.

[13] R. M.-C. Francisco J.Romero-Ramirez, Rafael Muñoz-Salinas, "Speeded up detection of squared fiducial markers," in *Image and Vision Computing*, vol. 76, 2018, pp. 38–47.

[14] Blender, "Blender overview," https://www.blender.org/.

[15] I. Inc., "Xrecorder," https://play.google.com/store/apps/details?id=videoeditor.videorecorder.screenrecorder&hl=en_US&gl=US.

[16] D. F. Abawi, J. Bienwald, and R. Dorner, "Accuracy in optical tracking with fiducial markers: an accuracy function for artoolkit," in *IEEE ISMAR*, 2004.

[17] "Opencv aruco marker board," https://docs.opencv.org/master/db/da9/tutorial_aruco_board_detection.html.

[18] R. Yagfarov, M. Ivanou, and I. Afanasyev, "Map comparison of lidar-based 2d slam algorithms using precise ground truth," in *International Conference on Control, Automation, Robotics and Vision*, 2018.

[19] F. Zheng, D. Schmalstieg, and G. Welch, "Pixel-wise closed-loop registration in video-based augmented reality," in *IEEE ISMAR*, 2014.

[20] W. A. Weliamto, H. S. Seah, T. Feng, and L. Li, "Enhancement of aligning accuracy on zooming camera for augmented reality," in *International Conference on Advances in Computer Entertainment Technology*, 2005.

[21] F. Zheng, R. Schubert, and G. Welch, "A general approach for closed-loop registration in ar," in *IEEE Virtual Reality*, 2013.

[22] S. Petrangeli, G. Simon, H. Wang, and V. Swaminathan, "Dynamic adaptive streaming for augmented reality applications," in *IEEE ISM*, 2019, pp. 56–567.

[23] F. Faion, A. Zea, B. Noack, J. Steinbring, and U. D. Hanebeck, "Camera- and imu-based pose tracking for augmented reality," in *IEEE International Conference on Multisensor Fusion and Integration*, 2016.

[24] B. MacIntyre, E. M. Coelho, and S. J. Julier, "Estimating and adapting to registration errors in augmented reality systems," in *IEEE Virtual Reality*, 2002.

[25] X. Ran, C. Slocum, Y.-Z. Tsai, K. Apicharttrisorn, M. Gorlatova, and J. Chen, "Multi-user augmented reality with communication efficient and spatially consistent virtual objects," in *ACM CoNEXT*, 2020.

[26] T. Scargill, J. Chen, and M. Gorlatova, "Here to stay: Measuring hologram stability in markerless smartphone augmented reality," *arXiv preprint arXiv:2109.14757*, 2021.

[27] E. Peillard, F. Argelaguet, J. Normand, A. Lécuyer, and G. Moreau, "Studying exocentric distance perception in optical see-through augmented reality," in *IEEE ISMAR*, 2019.

[28] C. S. Rosales, G. Pointon, H. Adams, J. Stefanucci, S. Creem-Regehr, W. B. Thompson, and B. Bodenheimer, "Distance judgments to on- and off-ground objects in augmented reality," in *2019 IEEE VR*.

[29] S. M. Lehman, H. Ling, and C. C. Tan, "Archie: A user-focused framework for testing augmented reality applications in the wild," in *IEEE Virtual Reality*, 2020.

[30] F. Bork, R. Barmaki, U. Eck, K. Yu, C. Sandor, and N. Navab, "Empirical study of non-reversing magic mirrors for augmented reality anatomy learning," in *IEEE ISMAR*, 2017.