Demo: Will it Move? Indoor Scene Characterization for Hologram Stability in Mobile AR

Tim Scargill
Duke University
ts352@duke.edu

Shreya Hurli
Duke University
shreya.hurli@duke.edu

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

Maria Gorlatova
Duke University
maria.gorlatova@duke.edu

ABSTRACT

Mobile Augmented Reality (AR) provides immersive experiences by aligning virtual content (holograms) with a view of the real world. When a user places a hologram it is usually expected that like a real object, it remains in the same place. However, positional errors frequently occur due to inaccurate environment mapping and device localization, to a large extent determined by the properties of natural visual features in the scene. In this demonstration we present SceneIt, the first visual environment rating system for mobile AR based on predictions of hologram positional error magnitude. SceneIt allows users to determine if virtual content placed in their environment will drift noticeably out of position, without requiring them to place that content. It shows that the severity of positional error for a given visual environment is predictable, and that this prediction can be calculated with sufficiently high accuracy and low latency to be useful in mobile AR applications.

CCS CONCEPTS

• **Computing methodologies → Mixed / augmented reality**; **Tracking**; **Scene understanding**.

KEYWORDS

Augmented reality, scene characterization, VI-SLAM.

ACM Reference Format:
Tim Scargill, Shreya Hurli, Jiasi Chen, and Maria Gorlatova. 2021. Demo: Will it Move? Indoor Scene Characterization for Hologram Stability in Mobile AR. In *The 22nd International Workshop on Mobile Computing Systems and Applications (HotMobile '21), February 24–26, 2021, Virtual, United Kingdom.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3446382.3449026

1 INTRODUCTION

Over the past few years, mobile augmented reality (AR) apps that do not require images or fiducial markers to position virtual content have increasingly become standard, enabled by platforms such as Google's ARCore [6] and Apple's ARKit [1]. However, these markerless AR apps rely on natural visual features in the environment that can be mapped and tracked accurately, which is not always the case. Errors resulting from inaccurate tracking cause virtual content (holograms) to be rendered at a different location in the real world from where they were originally placed, often termed *drift*. Examples of hologram drift in the commercial AR app 'Avo!' are shown in Figure 1. These errors can degrade a user's subjective experience and her task performance in AR.

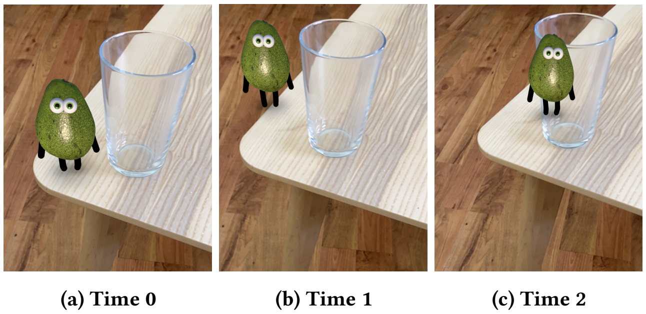
(a) Time 0 **(b) Time 1** **(c) Time 2**

Figure 1: Hologram drift examples in commercial AR (ARKit, iPhone 11, iOS 13.1.3, "Avo!" application)

In modern mobile AR, accurate environment mapping and device localization is dependent on Visual-Inertial SLAM (Simultaneous Localization and Mapping) [7]. The more conducive visual input data are to accurate Visual-Inertial SLAM pose estimation, the lower the resulting drift. Our goal is to guide users, developers and designers of spaces that host AR toward optimal experiences, by determining and detecting the conditions that cause virtual content positional errors. Though this is a known issue in AR, recent works only quantify either drift [9, 16] or environmental characteristics [2], not both. ARKit and ARCore can identify unfavorable conditions based on whether tracking results are unavailable or questionable, along with possible high-level causes, but these solutions lack the granularity required to guide users toward more suitable visual environments. *Our work is the first to formally examine the relationship between visual scene characteristics and drift magnitude.*

In this demo we present SceneIt, an edge-based visual environment rating system for mobile AR, based on predictions of hologram positional error magnitude. An illustration of SceneIt is shown in Figure 2. Unlike existing commercial solutions that identify challenging environments using simple indicators (e.g., light level, number of feature points), SceneIt uses a set of custom scene characterization metrics, and takes into account the complex interactions between scene properties. As well as detecting conditions known to cause severe degradation or loss of tracking, it demonstrates how more subtle cases can be identified; in which sufficient visual features are detected, but feature mismatching is likely to occur. The nature of the design supports future work on automatic

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
HotMobile '21, February 24–26, 2021, Virtual, United Kingdom
© 2021 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-8323-3/21/02…$15.00
https://doi.org/10.1145/3446382.3449026

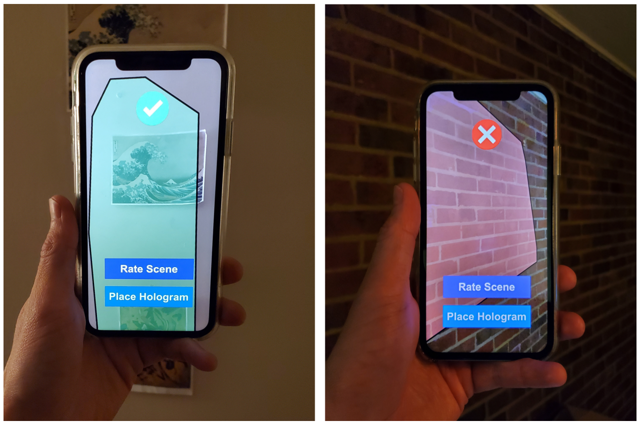

Figure 2: SceneIt in action: examples of good (left) and bad (right) scene quality ratings.

improvements to AR environments, and the consideration of other factors which affect quality of experience in AR.

2 SYSTEM DESIGN

An overview of the SceneIt system architecture is shown in Figure 3. We implement an edge computing-based architecture to serve online predictions. This architecture allows us to calculate the visual scene characterization metrics required for accurate drift prediction, a task which is too computationally expensive to perform on a smartphone.

We identify three key sources of data in an AR application that may be used to characterize a visual scene: the RGB images from the rear-facing camera, the point cloud generated by the AR platform, and additional data supplied by the AR platform (e.g., light estimation, plane detection). These data are captured within the AR application, converted to JSON format, and transferred over a one-hop wireless local area network connection via an HTTP POST request (constructed using a UnityWebRequest). The request is received and handled on the edge server by a high performance Python web framework, FastAPI [4]. This converts the data into Python objects, from which our scene characterization metrics (detailed below) can be computed.

RGB Image Properties. At the lowest level, we can extract information from the overall distribution of pixel values about light level and the distinguishable elements of a scene (**Img. Brightness**, **Img. Brightness RSD** (Relative Standard Deviation), **Contrast**, **Entropy** [14]). However, we should also take into account the spatial organization of those pixel values (**Spatial Information** [19], **Laplacian Variance** [5], **Corners** [13]). A more complex image is generally beneficial, but this complexity should relate to discernable features rather than chaos, measured using gradient orientation entropy (**GO Entropy** [17]). More sophisticated image processing can also recognize situations likely to cause pose estimation errors; we identify repetitive textures using gradient orientation self similarity (**GO SelfSim** [10]), bright, transient spots of light (**Specular Highlights** [8]), metal or glass household objects that cause reflections (**Challenging Objects** [11]), and mirrors (**Mirrors** [18]).

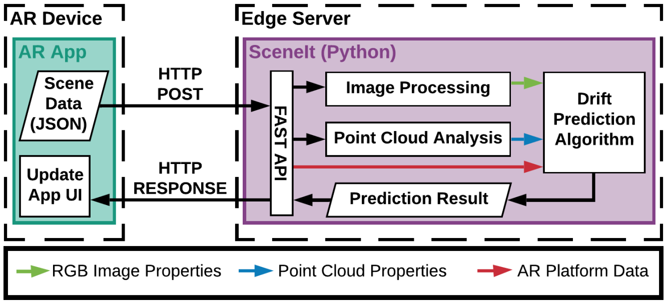

Figure 3: SceneIt system architecture: Data collected about the visual environment is sent from the AR app to the edge server. SceneIt calculates scene characterization metrics and the hologram drift prediction result, and the user can be notified with check and cross symbols, updated plane coloring or sound effects.

Point Cloud Properties. Next, we can extract information from the 3D point cloud generated by the AR platform while mapping the scene. From this we calculate **Feature Point Count, Feature Point Density, Mean Depth, Feature Point Proximity**, and how evenly distributed the points are across the mapped space (**Spatial Heterogeneity** [12]). We hypothesize that dense, evenly spread point clouds, with higher numbers of points and a low mean depth will decrease drift, and vice versa.

AR Platform Data. Finally, the AR platform itself (e.g., ARCore, ARKit) may provide useful data on light estimation and detected planes. We can extract measures of light level (**Brightness**) and appearance (**Color Temperature**), extreme values of either we expect to indicate challenging conditions. Estimates of **Main Light Intensity** and **Light Direction** are sometimes provided; a high horizontal component of the light direction vector, combined with high light intensity, may result in shadows and dynamic (unreliable) feature points. More planes (**Plane Count**) should aid accurate positional tracking, but a window (**Window Count**, ARKit only) may impede it, as the feature points within that plane are often unreliable.

Drift Prediction Algorithm. To inform the design of our drift prediction algorithm we developed a custom AR app for both ARKit and ARCore (using Unity's AR Foundation framework [15]), that can gather scene data and measure drift. Keeping device movement as similar as possible across trials, we collected data for 141 diverse scenes in 13 rooms (6.6m^2–40.4m^2), in 6 residential or office buildings, using both ARCore and ARKit. Drawing inspiration from indoor datasets such as Matterport3D [3], we constructed a wide range of conditions, including a flashlight-lit basement, a brick accent wall, and a living room at sunset. ARKit scenes were captured using an iPhone 11 and ARCore scenes using a Google Pixel 3a or Nokia 7.1. We observe that *for normal indoor conditions drift magnitude is determined by a complex interaction between scene properties*, rather than a single metric such as brightness or feature point count.

Due to this complex interaction between variables, we use machine learning to predict the amount of drift in a scene. To meet our end goal of interpretability (to inform users how to improve their scenes) with a limited number of data points, we use a decision tree. For data gathered on ARKit, we train a binary classifier using scikit-learn, to predict if a scene is 'Good' (drift < 3cm) or 'Bad' (drift ≥ 3cm). This classification boundary is based on both the distribution of drift values and our own observations on how drift magnitude impacts user experience. We use an 80/20 train/test split and to limit overfitting set the max tree depth to 5. We achieve 82.6% accuracy and an F1 score of 82.4% for our classifier. Future work will include more sophisticated feature selection techniques to gain further insights and improve performance.

Once SceneIt has calculated the prediction result, FastAPI returns the result to the app via an HTTP RESPONSE over the same wireless connection, and the app user interface is updated accordingly. Over 10 trials in unseen visual environments (not part of our dataset), we classify 8 correctly and achieve a mean end-to-end system latency of 1.74 seconds.

3 INTERACTIVE DEMONSTRATION

The demonstration utilizes the same architecture as in Figure 3. It is performed using an iPhone 11 running iOS 13.1.3 and ARKit 3, and an AR application built with Unity 2019.3.9f1. SceneIt runs on a high-end desktop computer with an Intel i7-9000 3GHz CPU and Intel UHD Graphics 630 GPU. A video of the demonstration is available online.[1]

As the user moves around, ARKit generates a map of the environment in the form of a sparse 3D point cloud. When a plane is detected in the environment, it is represented by a transparent hologram with a black border. The presence of a plane allows the user to attach a hologram to the real-world surface. When the user presses the 'Rate Scene' button, the application captures the current RGB camera image and the point cloud, as well as the lighting estimation and plane detection data gathered by ARKit.

These data are transferred to the edge server, and SceneIt calculates the prediction result based on these inputs. For subsequent rating requests the image- and lighting-based metrics are averaged across the session, while the most recent point cloud and plane detection data are used each time. If SceneIt returns a positive rating to the app (indicating a prediction of low drift), the AR application user interface is updated to show a green check symbol. The user can then place their virtual content (in the demo a Mondrian painting) on the plane using the 'Place Hologram' button, with confidence that it will not move significantly out of position.

On the other hand, if SceneIt returns a negative rating, the user interface is updated to show a red cross symbol. This indicates that drift of 3cm or over is predicted to occur, and if the user places the hologram at this time then this can be observed. In future work we plan to instruct the user how to improve the quality of a bad scene, or automatically do so using IoT devices available in the environment, to adjust lighting conditions or display additional textures.

[1] https://sites.duke.edu/timscargill/sceneit-prototype/

4 ACKNOWLEDGEMENTS

This work was supported in part by the Lord Foundation of North Carolina and by NSF awards CSR 1903136, CNS 1908051 and CAREER 1942700.

REFERENCES

[1] Apple. 2021. ARKit. https://developer.apple.com/augmented-reality/

[2] Lillemor Blom. 2018. *Impact of light on augmented reality.* Master's thesis. Linköping University.

[3] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. 2017. Matterport3D: Learning from RGB-D data in indoor environments. In *International Conference on 3D Vision (3DV) 2017.*

[4] FastAPI. 2021. FastAPI. https://fastapi.tiangolo.com/

[5] James Garforth and Barbara Webb. 2019. Visual appearance analysis of forest scenes for monocular SLAM. In *IEEE ICRA 2019.*

[6] Google. 2021. ARCore. https://arvr.google.com/arcore/

[7] Li Jinyu, Yang Bangbang, Chen Danpeng, Wang Nan, Zhang Guofeng, and Bao Hujun. 2019. Survey and evaluation of monocular visual-inertial SLAM algorithms for augmented reality. *Virtual Reality & Intelligent Hardware* 1, 4 (2019), 386–410.

[8] Alexandre Morgand and Mohamed Tamaazousti. 2014. Generic and real-time detection of specular reflections in images. In *VISAPP 2014.*

[9] Xukan Ran, Carter Slocum, Maria Gorlatova, and Jiasi Chen. 2019. ShareAR: Communication-efficient multi-user mobile augmented reality. In *ACM HotNets 2019.*

[10] Christoph Redies, Seyed Ali Amirshahi, Michael Koch, and Joachim Denzler. 2012. PHOG-derived aesthetic measures applied to color photographs of artworks, natural scenes and objects. In *ECCV 2012.*

[11] Joseph Redmon and Ali Farhadi. 2018. YOLOv3: An incremental improvement. *arXiv preprint arXiv:1804.02767* (2018).

[12] Brian D Ripley. 1976. The second-order analysis of stationary point processes. *Journal of Applied Probability* 13, 2 (1976), 255–266.

[13] Edward Rosten and Tom Drummond. 2006. Machine learning for high-speed corner detection. In *ECCV 2006.*

[14] C. E. Shannon. 1948. A mathematical theory of communication. *The Bell System Technical Journal* 27, 3 (1948), 379–423.

[15] Unity. 2021. Unity AR Foundation. https://unity.com/unity/features/arfoundation/

[16] Reid Vassallo, Adam Rankin, Elvis C. S. Chen, and Terry M. Peters. 2017. Hologram stability evaluation for Microsoft HoloLens. In *SPIE Medical Imaging 2017: Image Perception, Observer Performance, and Technology Assessment.*

[17] Yichin Wu, Liwei Chan, and Wen-Chieh Lin. 2019. Tangible and visible 3D object reconstruction in augmented reality. In *IEEE ISMAR 2019.*

[18] Xin Yang, Haiyang Mei, Ke Xu, Xiaopeng Wei, Baocai Yin, and Rynson WH Lau. 2019. Where is my mirror?. In *IEEE ICCV 2019.*

[19] Honghai Yu and Stefan Winkler. 2013. Image complexity and spatial information. In *IEEE QoMEX 2013.*