Environment Texture Optimization for Augmented Reality

TIM SCARGILL, Duke University, USA
RITVIK JANAMSETTY, Duke University, USA
CHRISTIAN FRONK, Duke University, USA
SANGJUN EOM, Duke University, USA
MARIA GORLATOVA, Duke University, USA

Augmented reality (AR) platforms now support persistent, markerless experiences, in which virtual content appears in the same place relative to the real world, across multiple devices and sessions. However, optimizing environments for these experiences remains challenging; virtual content stability is determined by the performance of device pose tracking, which depends on recognizable environment features, but environment texture can impair human perception of virtual content. Low-contrast 'invisible textures' have recently been proposed as a solution, but may result in poor tracking performance when combined with dynamic device motion. Here, we examine the use of invisible textures in detail, starting with the first evaluation in a realistic AR scenario. We then consider scenarios with more dynamic device motion, and conduct extensive game engine-based experiments to develop a method for optimizing invisible textures. For texture optimization in real environments, we introduce MoMAR, the first system to analyze motion data from multiple AR users, which generates guidance using situated visualizations. We show that MoMAR can be deployed while maintaining an average frame rate > 59fps, for five different devices. We demonstrate the use of MoMAR in a realistic case study; our optimized environment texture allowed users to complete a task significantly faster (p=0.003) than a complex texture.

CCS Concepts: • **Human-centered computing** → **Mixed / augmented reality**; **Mobile devices**.

Additional Key Words and Phrases: Augmented reality, pose tracking, VI-SLAM, environment texture, visual perception

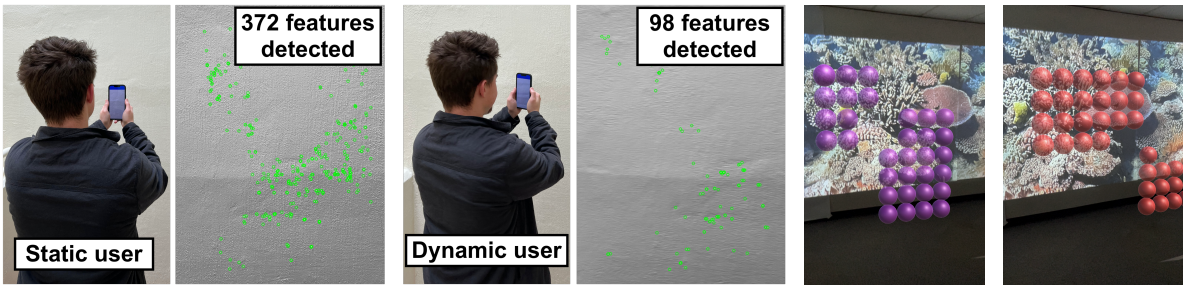
ACM Reference Format:
Tim Scargill, Ritvik Janamsetty, Christian Fronk, Sangjun Eom, and Maria Gorlatova. 2024. Environment Texture Optimization for Augmented Reality. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 8, 3, Article 121 (September 2024), 26 pages. https://doi.org/10.1145/3678510

1 Introduction

In markerless augmented reality (AR), virtual content is spatially registered with the real world by tracking natural environment features, i.e., the visual texture available in camera images of the real environment. Compared to marker-based AR, which relies on a view of predefined fiducial or image markers, markerless AR provides convenience and flexibility, and readily supports more mobile scenarios in which a user views many different environment regions. Moreover, modern AR platforms now support persistent AR experiences, in which virtual content appears at predefined locations across multiple devices and sessions, by anchoring it to a map of these natural environment features. These developments hold great promise for a wide range of persistent AR application scenarios, from factory and warehouse tasks to immersive museum exhibits.

Authors' Contact Information: Tim Scargill, Duke University, Durham, NC, USA, ts352@duke.edu; Ritvik Janamsetty, Duke University, Durham, NC, USA, ritvik.janamsetty@duke.edu; Christian Fronk, Duke University, Durham, NC, USA, christian.fronk@duke.edu; Sangjun Eom, Duke University, Durham, NC, USA, sangjun.eom@duke.edu; Maria Gorlatova, Duke University, Durham, NC, USA, maria.gorlatova@duke.edu.

ACM 2474-9567/2024/9-ART121
https://doi.org/10.1145/3678510



(a) (b) (c) (d)

Fig. 1. Fine, low-contrast environment textures (a) support good pose tracking while an AR device is moved slowly, but (b) result in poor performance with more challenging motions, due to e.g., motion blur. To guide environment texture optimization, MoMAR aggregates data from multiple AR users to generate situated visualizations that highlight the environment regions in view when users are (c) focused on virtual content, and (d) performing challenging device motions.

However, a key challenge in optimizing environments for AR is the conflicting requirements of machine and human perception. Virtual content stability relative to the real world is a critical aspect of user experience in AR, and in markerless AR this is dependent on device pose tracking, achieved through visual-inertial simultaneous localization and mapping (VI-SLAM). More complex visual textures in an environment generally result in better pose tracking performance [11, 32], and thereby more stable virtual content [52, 59]. On the other hand, complex environment textures can negatively impact human perception in AR, by distracting users or affecting virtual content visibility [57, 78]. Designers of spaces that host AR require guidance on how to optimize environment textures to support good pose tracking performance without impairing human perception.

To address this, the authors of [58] proposed the use of what they term 'invisible textures' – textures that have sufficient complexity to support good VI-SLAM performance, but low contrast to minimize their impact on human perception. However, the evaluation of these textures focused primarily on pose tracking performance. The impact of textures on human perception was assessed on the basis that less complex, low-contrast textures are likely to be beneficial [20, 24, 57, 78], and subjective texture complexity scores were obtained from a non-AR scenario [13]. To obtain a more accurate and complete understanding of whether the proposed low-contrast invisible textures benefit human perception compared to the complex environment textures generally recommended for virtual content stability [42], they must be tested in realistic AR scenarios, on state-of-the-art AR devices.

Furthermore, there has been limited consideration of the extent to which environment textures (including invisible textures), support content stability across different AR device movements. *Different tasks and virtual content naturally result in different motion patterns, which in turn place different requirements on environment texture properties.* For example, with a fine, low-contrast texture such as a rough concrete wall, many more visual features are detected in tracking camera images while a user is slowly inspecting virtual content (Figure 1a) than during more dynamic device motion, when camera images are corrupted by motion blur artifacts (Figure 1b). It is vital that environment designers are able to determine which types of motion users perform while facing different environment regions, and whether textures need to be adjusted accordingly. Ultimately, we require a solution which guides fine-grained environment texture optimization, such that sufficient features are present where needed, but extraneous texture does not impair human perception of virtual content.

The goals of this paper are twofold. Firstly, we wish to test the proposed invisible texture specification in a realistic AR scenario. We accomplish this via a user study on an AR headset, with a table-top toy assembly task that replicates a factory AR use case; in this setting one might adjust the textures on workbenches to support optimal AR-assisted product assembly or quality assurance. Secondly, we wish to assess how well invisible

textures support accurate pose tracking in more mobile scenarios. AR has promising applications in settings where virtual content persists in specific environment regions, e.g., an immersive museum exhibit, or situated patient notes in a hospital ward. However, these scenarios can introduce device motions that are challenging for pose tracking, particularly in mobile AR (smartphones and tablets) [59]. We conduct extensive game engine-based experiments to identify motions that result in poor tracking performance with invisible textures, and demonstrate how this can be addressed by adjusting texture only in regions associated with those motions. We use these insights to develop and evaluate MoMAR (**Mo**tion **M**apping for **AR**), *the first system that analyzes motion data from multiple AR users to provide environment texture optimization guidance*. Our contributions are as follows:

- We show how employing a low-contrast 'invisible' environment texture can reduce the perceived workload associated with an AR task, compared to a complex texture designed solely to minimize pose tracking error. We conduct a between-subjects experiment with 32 participants (two groups of n=16), in which participants complete an AR assembly task with the Microsoft HoloLens 2, in front of either a complex or an invisible texture; *workload reported on the effort scale of the NASA-TLX was significantly lower (p=0.034) with the invisible texture than with the complex texture* (Section 3).
- Through extensive game engine-based experiments (Section 4), we illustrate how comparable pose tracking performance is achieved with invisible and complex environment textures, but reveal the performance gaps that occur due to certain AR device movements. We identify two types of motion, *stationary rotation view change* and *focused orbital inspection*, that result in high pose tracking error with invisible textures, and demonstrate how adding texture patches in specific environment regions addresses this; for the SenseTime [32] A2 trajectory, *we reduce pose tracking error by 30% with texture patches covering <3% of the environment surface area.*
- We develop MoMAR, a multiuser motion mapping system for AR, that associates device motion with environment regions in the camera view, aggregates these data across multiple AR sessions, and displays the results using situated visualizations to inform environment texture optimization. We implement this system for iOS devices running ARKit, characterize system latency for up to five concurrent AR sessions, and show that, for five different devices with varying hardware and software configurations, *an average frame rate of greater than 59fps can be maintained while running MoMAR in addition to a resource-intensive AR session* (Section 5).
- We conduct a case study on the use of MoMAR in a realistic AR scenario, a multimedia museum exhibit (Section 6). We perform a between-subjects experiment with 30 participants (two groups of n=15), in which *participants completed a visual search and inspection task in significantly less time with our optimized environment texture than with a complex texture (p=0.003), with no significant difference in perceived virtual object stability.*

The code required to implement MoMAR is publicly available at https://github.com/timscargill/MoMAR/.

2 Related Work

Environment texture and VI-SLAM-based pose tracking. In modern markerless AR, feature-based VI-SLAM (e.g., [11, 50]) is used to concurrently map the environment and track the 6DoF device pose. VI-SLAM performance determines virtual content stability, a known issue in AR [52, 59]. Feature-based VI-SLAM extracts information from input camera images using a pixel intensity-based corner detection algorithm (usually a derivative of [55]), and relies on recognizable environment textures. Performance is poorer in low-texture environments [11, 32, 57, 58], while increased texture complexity along with sufficient contrast is associated with better performance [57, 58]. Previous work has shown how camera image artifacts arising from device motion, e.g., blur, negatively impact performance [41, 45, 57], and that the impact of texture is greatest in the presence of more dynamic device motion [57]. However, to the best of our knowledge, there is no work on defining challenging motions or associating environment regions in the camera view with device motion properties. We tackle this here, to inform designers of the texture required in specific environment regions to achieve good tracking performance.

Environment texture and human perception. How humans perceive and interact with objects in an environment (including AR users and virtual objects) is impacted by environment textures, due to the nature of human visual attention and perception. The Feature Integration Theory [68] and computational models based on this theory, e.g, [31], describe how elementary features such as contrast, color, shape, and movement are processed in a pre-attentive stage, followed by the serial application of focused attention to different salient regions in order to perceive different objects. Thus, the presence of visual features like texture or images in a user's real environment may direct attention away from virtual content, distracting users and degrading task efficiency. This was demonstrated for fully real environments in [20] and fully virtual environments in [57]. In AR scenarios where users have to locate virtual content in a real environment, identifying objects in visual search tasks can take longer in the presence of complex backgrounds [73] or low contrast between object and the surround [47], while the perception of a virtual object is affected by surrounding stimuli (e.g., a background environment texture) due to effects known as surround suppression and enhancement [12, 18, 60, 74]. Finally, environment textures have been shown to affect the perceived spaciousness of an environment [70], and texture properties have been associated with different emotional responses and cognitive attributes [28, 39, 76].

Environment texture and perception of virtual content in AR. The perception of virtual content by an AR user is impacted by environment textures due to the properties of human perception as described above. On AR headsets with optical see-through displays (e.g., the Microsoft HoloLens 2), background textures also affect visibility due to virtual content transparency. Multiple studies have shown that textured real environments degrade virtual text legibility [15, 23–25, 27], while the results in [78] indicate that greater background contrast results in greater perceived transparency. We contribute to this body of work with the first study on the effect of environment texture on an assembly task in AR. To address the negative impact texture can have on the perception of virtual content, the authors of [58] proposed the use of low contrast 'invisible' textures in environments hosting AR. This idea of extracting visual features from apparently homogeneous or random textures such as concrete or carpet is related to another work on localization using ground textures [77]. We are the first to evaluate the proposed invisible textures in a real AR setting and examine the effects of device motion on their efficacy.

Mapping human and mobile device motion properties. SLAM and other mapping technologies (e.g., LiDAR) generate an environment map that reflects where a device has moved, but these maps do not directly capture device motion properties in different environment regions. A recent study provides a solution for annotating the trajectories of unmanned aerial vehicles with environment images and motion data [33], but does not associate motion data with an environment map. Spatial heatmaps are widely used to overlay intensity data of human presence and eye gaze onto maps or images of an environment [2, 36, 65], but generally do not explicitly provide data on motion properties. Our work is related to maps of traffic speed generated from floating car data [38], which highlight where action needs to be taken. However, our approach is distinct in that we map the motion of device sensors (tracking cameras) with the end goal of improving the quality of data captured by those sensors.

Situated visualization in AR. To connect our motion map data with the real environment, we employ situated visualization [43, 71, 72], a concept of presenting data in the physical environment they refer to – a recent review can be found in [8]. Situated visualizations can be implemented through either AR [21], or ambient displays [9]; here we use AR-based situated visualization as it provides the spatial resolution we require for our map data, and is a convenient interface for designers already working with AR. Our work is similar to those that display AR device trajectory data in AR [10, 56]. However, unlike [10], our solution identifies the environment regions viewed by users, and unlike [56], it captures the motion of multiple users instead of a single administrator trajectory, which may not be representative of a diverse set of users. Examples of situated visualizations that overlay data onto environment surfaces using a heatmap include [35] and [14].

3 Evaluation of an Invisible Texture in a Realistic AR Scenario

First, we conduct an in-person study to evaluate whether employing an invisible texture is beneficial in a realistic AR scenario, compared to a complex texture chosen solely for high-quality pose tracking. We start with the motivation for our study (Section 3.1), and detail the AR assembly task we develop for the study (Section 3.2). We then present our study design (Section 3.3), results (Section 3.4) and discussion (Section 3.5).

3.1 Study overview

We wish to evaluate the invisible texture specification proposed in [58] in a realistic AR scenario. Additionally, we wish to expand the range of tasks for which the effect of environment texture has been studied – the vast majority of existing works focus on virtual text legibility [15, 23–25, 27]. To this end, we assess the impact of environment texture on an AR assembly task [6, 16, 22, 25, 66, 75], in which virtual guidance aids in the construction of a real object. We posit that even if a complex environment texture does not cause a user to make errors in an assembly task, it may still result in greater workload and lower efficiency than an invisible texture — vital considerations in the industrial or manufacturing scenarios in which AR assembly tasks will be performed.

3.2 AR assembly task

For our AR assembly task, we first defined the real objects which users would combine and created virtual representations to use in each step of the assembly task guidance. To create these digital twins of our real components, we chose objects that could be conveniently scanned with sufficient fidelity — large colored 3D Duplo blocks, a common toy for young children. We defined a sequence of 17 steps in which 18 of these blocks are combined to assemble a toy train. To scan the appearance of the train and the individual blocks used in each step we used the Polycam app for iOS [48], performed post-processing in Blender [7], and saved them as .glb files.

To implement our virtual assembly guidance, we used Unity 2021.3.1.14f [69] and the Mixed Reality Toolkit 2 [44] to develop an AR application for the Microsoft HoloLens 2 headset. This application included both the virtual assembly guidance itself and a mid-air user interface (UI) to control the application. The virtual assembly guidance at each of the 17 steps consisted of a 3D model of the current appearance of the assembled toy train and a 3D model of the next block to be added, along with a 2D arrow to indicate how to combine them. The UI consisted of four square buttons arranged in a square (see Figures 2c and 2d), used to navigate backwards or forwards in the guidance steps, rotate the virtual assembly guidance 180 degrees, or start the application. Additionally, when the application first started, a virtual white sphere was aligned with the point where the forward vector defined by the head pose intersected with a plane. When the start button was pressed on the UI, the guidance was instantiated at this point, such that the virtual guidance would appear at a specific point on the tabletop.

3.3 Study design

To test our hypothesis on the effect of environment texture on our AR assembly task, we conducted an Institutional Review Board (IRB)-approved user study. To avoid the learning effects associated with participants becoming more familiar with the assembly task over repeated trials, we opted for a between-subjects experimental design, in which participants performed the task on one of two tabletop textures. We chose to test one complex texture which has been shown to support high-accuracy pose tracking [57], along with one low-contrast invisible texture which meets the criteria specified in [58] for supporting good quality pose tracking but also having minimal impact on human perception. These two textures are shown in Figure 2a.

Participants: We recruited 32 participants (11 female, 21 male; aged 19 to 29) from our personal and professional networks. All participants had normal or corrected to normal eyesight. Participants were split into two groups of 16, with each group assigned to one of the two texture conditions.

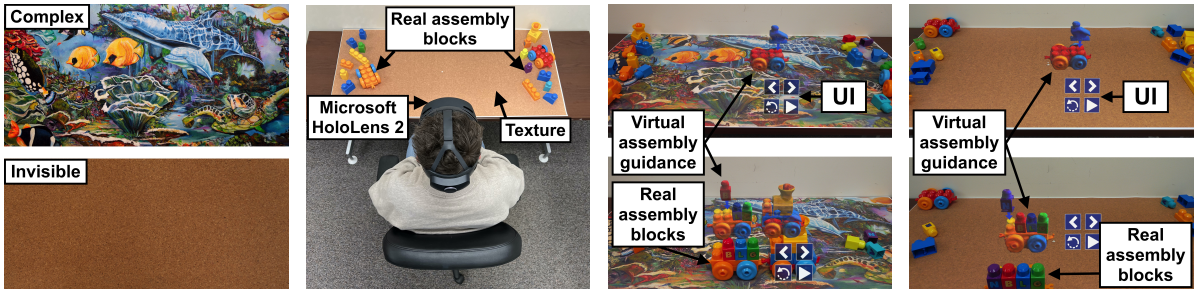


Fig. 2. Implementation details for our study on the impact of environment texture on an AR assembly task: (a) The complex and invisible environment textures tested; (b) The experiment setup for our study; (c) Example views through the AR headset at the (top) start of and (bottom) during the task with the complex environment texture; (d) Example views through the AR headset at the (top) start of and (bottom) during the task with the invisible environment texture.

Experiment setup and apparatus: The experiment setup is shown in Figure 2b. The study was performed under controlled conditions in a lab environment with no windows, and a fixed level of overhead lighting (illuminance in the study area was approximately 700 lux). The Microsoft HoloLens 2 was used as the AR headset. The chosen textures were printed on 1.22m×0.60m matte paper, and placed on a 1.83m×0.60m table top (height = 0.72m). Participants were seated next to the table and allowed to adjust the height of their chair such that they were comfortable, while confirming that the virtual guidance appeared in front of the texture. Before the start of the task, the real assembly blocks were placed in random positions on the left and right edges of the texture.
Task: After aligning their head pose with a predefined point in the middle of the texture, participants pressed the start button on the UI to place the virtual assembly guidance at that position. They then navigated the assembly guidance steps displayed on the AR headset using the UI buttons, and combined the real assembly blocks in the configuration and order illustrated by the guidance. Examples of a participant's view at the start of the task and during the task are shown for the complex texture in Figure 2c, and for the invisible texture in Figure 2d.
Dependent variables and operationalization: We measured the *workload* experienced by the user using the NASA-TLX [30] and the mean eye gaze fixation duration, and *efficiency* using the task duration and the total head rotation during the task. While not part of our main hypothesis, we also recorded the number of mistakes in the completed toy train model to capture task *accuracy*. Additionally, we evaluated *virtual content stability* through the post-experiment questionnaire. Participants were asked to rate on a 5-point Likert scale how much they agreed with the statement "*The guidance was stable (not moving or jittering etc.)*", using the following responses: 1=Strongly disagree, 2=Disagree, 3=Neither agree nor disagree, 4=Agree, 5=Strongly agree.
Procedure: First, we introduced participants to the study and asked them to sign a consent form. Participants were informed of the general purpose of the study (i.e., to evaluate their experience with the AR assembly app), but not about the environment texture variable or the experiment hypothesis. Participants answered a pre-experiment questionnaire via Qualtrics [51], then completed eye tracking calibration on the AR headset. After that, they completed a training period to become familiar with the AR assembly task; in this training period the study administrator provided task instructions, then the participant completed a short assembly task representative of but distinct from the main assembly task (using the same type of real assembly blocks in a different configuration). Once they were comfortable with the task, participants performed the main assembly task. Finally, participants completed the NASA-TLX on paper, and a post-experiment questionnaire using Qualtrics.

3.4 Study results

The results of our study for each of our dependent variables are detailed below. To test for statistical significance without the assumption that the data is normally distributed we used the Mann-Whitney U Test.

Workload: The results for the NASA-TLX are shown in Figure 3. Reported workload on the effort scale was significantly lower (p=0.034) with the invisible texture than with the complex texture, indicating that participants had to work harder to achieve their level of performance with the complex texture. No significant differences between the texture conditions were found for the other scales. No significant difference was found for fixation duration (Table 1), but it was lower for the complex texture, which is associated with greater perceptual load [40].
Efficiency: The results for efficiency, measured using task duration and head rotation are shown in Table 1. No significant differences were found between the two texture conditions for task efficiency.
Accuracy: A single mistake was made by one participant in each of the two texture condition groups; in one case, the orange and red train bases were mixed up and in the other, a yellow and orange brick were mixed up. As such we found no effect of environment texture on the accuracy of task completion for this assembly task.
Virtual object stability: We assigned numerical values from one to five to each point on our virtual object stability post-experiment question, with higher values representing greater stability. The mean responses were 4 for the complex texture and 4.125 for the invisible texture. No significant difference was found between the two texture conditions, indicating that a comparable level of pose tracking performance was achieved with the invisible texture and the complex texture.

3.5 Discussion

Our results show that employing an invisible environment texture instead of a complex texture can significantly reduce the workload associated with an AR assembly task, without compromising on other aspects of task performance or user experience. In general, even in cases where differences between the textures were not statistically significant, our results indicate benefits associated with an invisible texture. However, there were two exceptions to this; physical workload (Figure 3) was slightly higher for the invisible texture, as was task duration (Table 1). We discuss possible reasons for these results below.

For physical workload, one would not expect an invisible texture to be particularly beneficial in this scenario, because of the small amount of user motion required to complete the task. Any challenges associated with discerning instructions and objects are unlikely to result in greater movement. Our hypothesis is that this result arises from small individual differences between subjects rather than any effect of texture. In Section 6, we conduct a study with a task that requires greater user motion; for this task physical workload was greater for a complex texture than an optimized invisible texture.

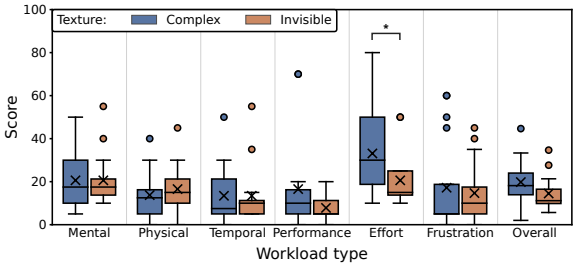


Fig. 3. NASA-TLX scores for each texture condition in our AR assembly task. Workload on the effort scale was significantly lower for the invisible texture than the complex texture (p=0.034).

Table 1. Mean values for eye movement, task performance, and head movement measures used to evaluate workload and efficiency associated with our AR assembly task, calculated for each texture condition (complex and invisible). No significant differences were found for these measures.

Measure	Complex	Invisible
Fixation duration (ms)	1.33	1.53
Task duration (s)	145	157
Head rotation (°)	2411	2335

For task duration, while again not statistically significant, the greater duration for the invisible texture is contrary to what one might expect, that challenges associated with discerning instructions and objects in front of a complex texture might delay users. One possibility is that the relatively straightforward nature of our task meant that any delays introduced due to texture were small, and the discrepancy is again related to small differences between subjects. Another possibility is that the color of the textures influenced participants; while we ensured that the average luminance of the complex and invisible textures were the same, the complex texture was much more colorful, and environment color has been shown to affect human productivity [1]. Future studies on this type of scenario should consider more complex tasks and control for factors such as texture color (see Section 7).

4 Optimizing Environment Texture in Mobile User Scenarios

We now evaluate the effect of environment texture on pose tracking performance in more mobile user scenarios. Because virtual content stability errors are a more prevalent issue on smartphones and tablets that run monocular VI-SLAM (with a single tracking camera) than on AR headsets [59], we perform our pose tracking evaluations with this type of SLAM algorithm. First, we detail the setup for our experiments (Section 4.1). Next, we illustrate the impact of texture in realistic environments (Section 4.2), then assess the performance gap between complex and invisible textures (Section 4.3). Finally, we identify challenging device motions that result in worse performance with invisible textures. We show how the targeted application of texture in specific environment regions helps to address this, while maintaining the benefits of invisible textures for human visual perception (Section 4.4).

4.1 Pose tracking texture experiments setup

Method: To study the effect of texture under repeatable and controlled conditions, we leveraged the game engine-based methodology from [57]. In this methodology, the ground truth trajectory data from existing VI-SLAM datasets is used to generate new camera images in a virtual environment. These images, the visual data, are then combined with the original inertial data from the dataset to create semi-synthetic input sequences. We executed sequences using ORB-SLAM3 [11], a state-of-the-art, open-source SLAM algorithm, in monocular VI-SLAM mode with default parameter settings, and performed 10 trials with each sequence configuration. We ran ORB-SLAM3 on a desktop PC (Intel i7-9700K CPU and an Nvidia GeForce RTX 2060 GPU), but used a virtual machine with 4 CPUs and 8GB RAM to replicate the computational resources available on a typical AR device.
Datasets: For our experiments we created custom virtual environments in Unity 2020.3.14f1, and generated new sequences in them using trajectories from the SenseTime [32] and TUM VI [61] handheld device datasets. The characteristics of these trajectories are shown in Table 2; we provide a measure of how challenging inertial data are for each trajectory with the mean and maximum magnitude of the accelerometer and gyroscope readings –

Table 2. Characteristics of the device trajectories we used from existing SLAM datasets to evaluate pose tracking performance. Higher accelerometer (Acc.) and gyroscope (Gyro.) magnitude values indicate more rapid motion and greater difficulty.

				Acc. Magnitude		**Gyro. Magnitude**	
Dataset	**Trajectory**	**Duration (s)**	**Length (m)**	**Mean**	**Max**	**Mean**	**Max**
SenseTime	A1	120.47	35.90	9.79	11.29	0.31	1.48
SenseTime	A2	76.01	24.83	9.77	11.83	0.27	1.24
SenseTime	A3	46.09	7.05	9.73	12.12	0.31	1.23
SenseTime	A4	98.85	30.67	9.79	11.86	0.30	1.07
SenseTime	A5	77.09	16.24	9.79	15.47	0.97	2.61
SenseTime	A6	72.89	11.45	9.73	12.33	0.22	1.13
SenseTime	A7	79.40	15.40	9.73	13.40	0.25	1.26
TUM VI	room5	142.32	131.64	10.03	16.93	1.53	5.10

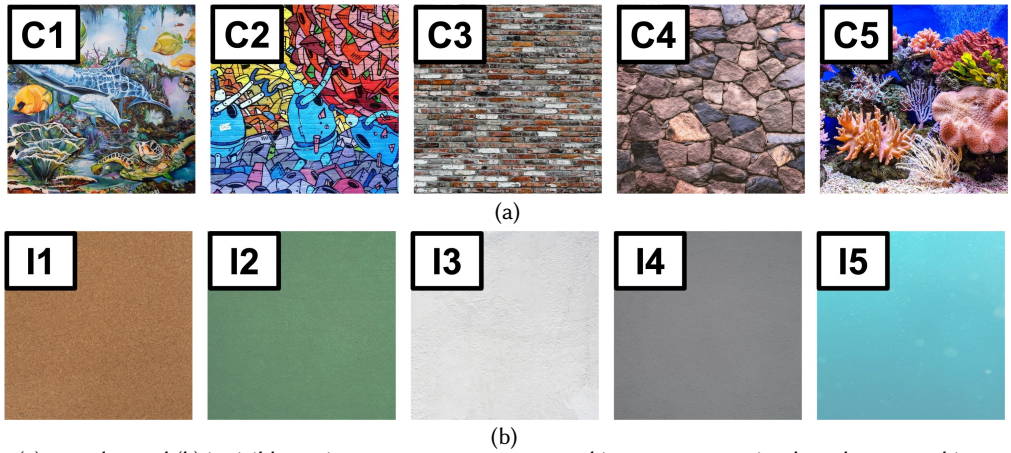


Fig. 4. The (a) complex and (b) invisible environment textures we tested in our game engine-based pose tracking experiments. Characterization metrics for these textures are provided in Table 3.

higher values indicate more dynamic device motion. The SenseTime trajectories are designed to replicate motion patterns in AR scenarios. In A1 and A2 the user walks around a room looking at different areas (described as 'inspect and patrol'). A3-7 each involve a user inspecting two different environment regions, but differ in device movement. In A4 and A7 surfaces are viewed from different angles but there is little device rotation. In A3 and A6 the user moves in a circle while remaining focused on a central region, and A5 incorporates rapid waving and shaking of the device. In TUM VI room5, the user walks around facing different environment regions, similar to A1 and A2, but device motion is more dynamic and rapid than in any of the SenseTime trajectories.

Performance and texture metrics: To quantify pose tracking accuracy, we used relative pose error, a standard measure of local trajectory consistency [34, 79]. This measure corresponds to the virtual content stability artifacts that AR users may perceive as they move around an environment. Specifically, we captured the translational (positional) component of relative pose error, on the basis that positional errors are more noticeable than

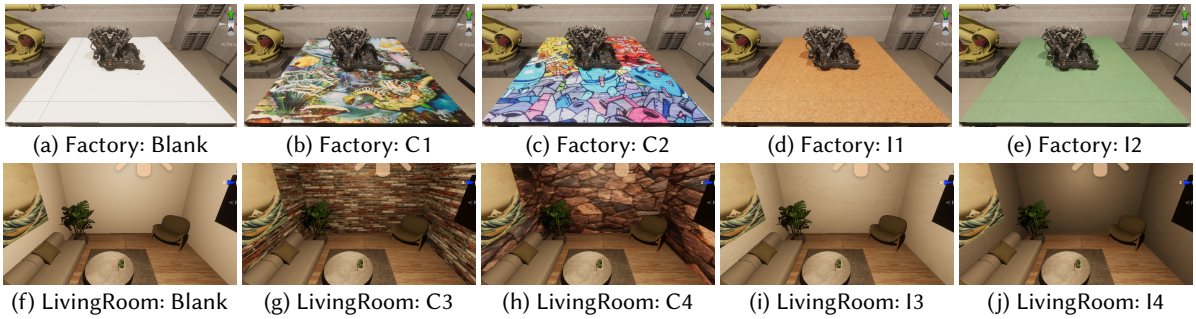
Table 3. Texture characterization metrics for our game engine-based pose tracking experiments: (a) metrics we used to characterize texture images, and (b) values for each of the complex and invisible textures (Figure 4) we tested.

(a)

Metric	Description
Brightness	Normalized mean pixel intensity
Contrast	Root mean square contrast
Entropy	Shannon entropy [63] of pixel intensities
Edge Density	Edge pixel density defined by Canny edge detector [54]

(b)

Texture	Brightness	Contrast	Entropy	Edge Density
C1	0.49	0.23	7.79	0.31
C2	0.46	0.26	7.67	0.21
C3	0.46	0.20	7.68	0.31
C4	0.44	0.25	7.66	0.33
C5	0.43	0.28	7.93	0.31
I1	0.50	0.05	5.64	0.27
I2	0.50	0.05	5.78	0.37
I3	0.87	0.04	5.13	0.16
I4	0.52	0.05	5.64	0.36
I5	0.67	0.05	5.48	0.00



(a) Factory: Blank (b) Factory: C1 (c) Factory: C2 (d) Factory: I1 (e) Factory: I2

(f) LivingRoom: Blank (g) LivingRoom: C3 (h) LivingRoom: C4 (i) LivingRoom: I3 (j) LivingRoom: I4

Fig. 5. The realistic environment settings (Environment: Texture) we tested in our game engine-based pose tracking experiments (Section 4.2); images are screenshots of the virtual environments we created in the Unity game engine.

orientation errors, and calculated the median error across all trials, the *median RE*. To quantify tracking *robustness*, we calculated the mean percentage of input camera frames tracked over all trials. We tested five complex environment textures (C1-5) and five invisible environment textures (I1-5) according to the definition proposed in [58]; these textures are shown in Figure 4. To characterize environment textures we used four metrics, shown in Table 3, which we calculated from the source image file of each texture using Python and OpenCV.

4.2 Impact of environment texture on pose tracking performance in realistic environments

First, we compared pose tracking performance with blank, complex, and invisible textures in two realistic environment settings: a factory representative of an AR assembly or repair scenario, and a living room representative of a home entertainment or educational scenario. In the factory environment, we varied the tabletop texture and tested a blank texture, our complex textures C1 and C2, and invisible textures I1 and I2. In the living room environment, we varied the wall texture and tested a blank texture, our complex textures C3 and C4, and invisible textures I3 and I4. These environments are illustrated in Figure 5. We used four environment plus trajectory configurations: the SenseTime A3, A5, and A6 trajectories in the factory environment, Factory_A3, Factory_A5, and Factory_A6, and the more dynamic SenseTime A1 trajectory in the living room environment, LivingRoom_A1.

The results for our realistic environment experiments are shown in Figure 6. Each data point within a box plot represents relative pose error for one trial, and the line plot represents robustness. The complex and invisible textures consistently outperformed the blank surface, with median RE greater for the blank surface than the textured surfaces for all configurations. *This result illustrates how the texture of an environment surface is a critical factor in achieving high-quality pose tracking, even for VI-SLAM algorithms that use inertial data, and even when other textured objects are present in the environment.* In general, the invisible textures (I1-5) resulted in comparable performance to the complex textures (C1-5), substantiating the approach proposed in [58]. However, there was still a performance gap between complex and invisible textures. In particular, for the Factory_A6 and LivingRoom_A1 configurations, median RE was greater for both of the invisible textures than either of the complex textures. This apparent limitation of invisible textures in certain scenarios motivated our deeper examination of this issue.

4.3 Pose tracking performance gap between complex and invisible textures

We evaluated the performance gap between complex and invisible textures by applying each texture to the walls, floor, and ceiling of an empty 6m×6m×6m virtual environment. We tested our five complex textures (C1-5) and our five invisible textures (I1-5) pictured in Figure 4, with the eight diverse trajectories in Table 2.

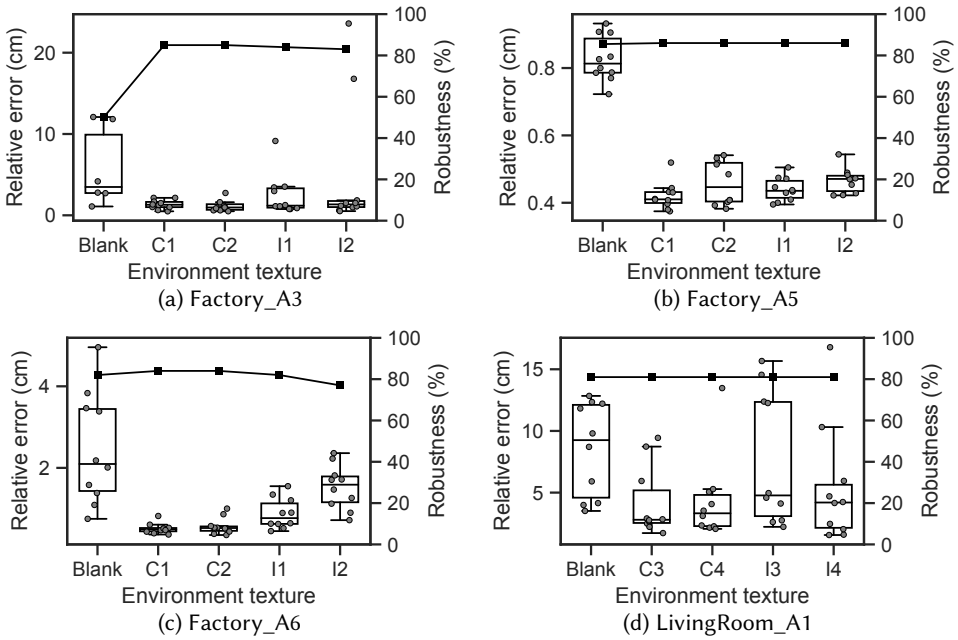


Fig. 6. Results from our game engine-based experiments on the effect of environment texture on pose tracking performance in realistic environments. The addition of either complex (C1-5) or invisible (I1-5) textures to blank regions such as table tops or walls improves pose tracking performance; invisible textures provide comparable performance to complex textures, but median RE (relative error) is consistently greater with invisible textures in some scenarios, e.g., Factory_A6, LivingRoom_A1.

The results of our experiments on the performance gap between complex and invisible textures are shown in Figure 7 and Table 4. The values for each texture class – complex and invisible – for each trajectory are the combination of results for the individual textures (5 textures × 10 trials for each texture class). The greatest increases in median RE for invisible textures compared to complex textures were for the A2 and A6 trajectories. For A2 median RE was 2.58cm for the complex textures but 4.42cm for the invisible textures, an increase of 71%. For A6 median RE was 0.84cm for the complex textures and 2.08cm for the invisible textures, an increase of 148%.

We also observed drops in robustness when using invisible textures for room5 and A5, caused by the failure of pose tracking to initialize during rapid device motion – as shown in Table 2, the room5 and A5 sequences contain the greatest accelerometer and gyroscope readings. However, the persistent AR scenario we target requires sufficient texture to localize with a previously created environment map (see Section 5.1), and this texture will also provide sufficient texture for initialization. Therefore, we focus on the challenging device motions in the A2 and A6 trajectories that result in greater relative error, rather than those which result in lower robustness.

Importantly, *the greatest increases in pose tracking error when using invisible textures instead of complex ones did not occur with the most dynamic device motion; rather, they arose due to specific types of challenging motion.* In analyzing possible reasons for greater pose tracking error with A2, we note that this trajectory contains multiple instances of the user stopping and turning to view a different area of the environment, during which the camera position remains relatively constant — we term this type of motion *stationary rotation view change.* During the A6 trajectory, the camera remains focused on an area of the floor, while the user walks around it to view it from different angles — we refer to this type of challenging motion *focused orbital inspection* (this motion is also present in A3, another trajectory for which we observed an increase in median RE for invisible textures).

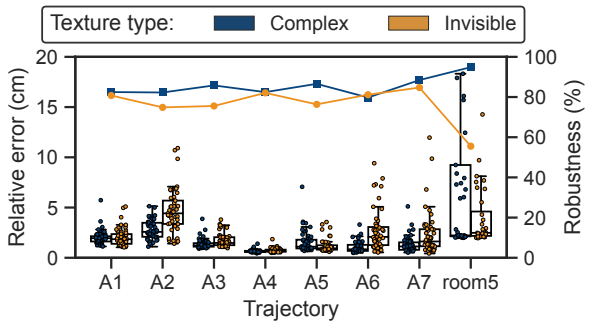


Fig. 7. The results of our game engine-based experiments on pose tracking performance with complex and invisible environment textures. For some device trajectories, the use of invisible instead of complex textures causes an increase in relative error or a drop in robustness, which is problematic for environment designers looking to employ invisible textures.

Table 4. The median RE and robustness values for the complex and invisible (Inv.) textures we tested with each trajectory; trajectories A2 and A6 resulted in the greatest performance gap for median RE, while room5 resulted in the greatest drop in robustness.

	Median RE (cm)		**Robustness** (%)	
	Complex	**Inv.**	**Complex**	**Inv.**
A1	1.95	1.85	82.48	80.76
A2	2.58	4.42	82.27	74.85
A3	1.22	1.48	85.78	75.55
A4	0.62	0.68	82.44	82.07
A5	1.10	0.94	86.56	76.35
A6	0.84	2.08	79.75	81.15
A7	1.13	1.62	88.44	84.72
room5	2.22	2.48	94.85	55.53

4.4 Motion-informed environment texture optimization

We now describe the device motion metrics which enable us to isolate the *stationary rotation view change* and *focused orbital inspection* motions within a trajectory, specify the conditions for each motion, and test the hypothesis that we can improve pose tracking performance for invisible textures by adding targeted texture patches to environment regions in the camera view during these motions.

4.4.1 Device motion metrics. We use three device motion metrics to identify specific types of device motion. The first two are the **rate of device position change**, v_d, and the **rate of environment view change**, v_e. The 3D device position is obtained directly from the pose tracking output, and the environment view is obtained by raycasting along the forward vector defined by the camera pose, and recording the 3D point at which that vector first intersects with an environment trackable (e.g., a plane or mesh). To obtain the change in the device position and environment view we calculate the 3D Euclidean distance between consecutive frames. Since we want to capture device motion characteristics over a greater time period than a single frame, we window each signal using a centered moving average. This smoothing function helps to eliminate spikes in device position due to loop closures, and high-frequency components in the environment view caused by small or irregularly shaped objects. For all experiments, we set the window length to 1s.

Our third device motion metric is **accumulated device rotation**, θ_d. This is the cumulative sum of device rotation changes around one axis during a specified time period. Since device orientation in the world frame obtained via pose tracking will likely contain errors during challenging device motions (because challenging motions typically involve device rotation), we instead obtain device rotation in the body frame via the raw gyroscope readings for an axis. We calculate the rotation change between consecutive frames from the gyroscope readings, then perform a cumulative sum of these values to approximate device rotation magnitude.

4.4.2 Environment texture patches. Our motion-informed texture optimization approach aims to improve the pose tracking performance of an invisible environment texture such as a rough concrete wall, by applying texture patches to specific regions. This represents an accessible technique in which a designer adds texture to walls

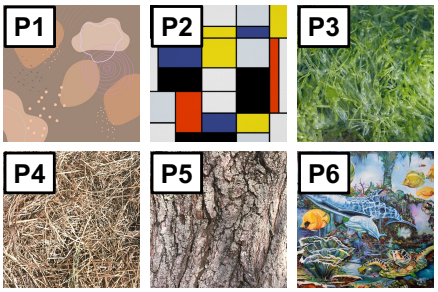


Fig. 8. The different texture patches we applied to invisible textures in the evaluation of our motion-informed environment texture optimization method (Section 4.4).

Table 5. Characterization metrics for the texture patches (Figure 8) we applied to invisible textures in our motion-informed environment texture optimization experiments (Section 4.4). Condition P7 in our experiments was one instance each of P3-6 (Section 4.4.3).

Texture	Brightness	Contrast	Entropy	Edge Density
P1	0.61	0.08	3.52	0.05
P2	0.52	0.36	5.32	0.03
P3	0.44	0.14	7.17	0.29
P4	0.55	0.29	7.63	0.37
P5	0.53	0.24	7.75	0.37
P6	0.49	0.23	7.76	0.33

using a poster, or to the floor using a mat. To evaluate the efficacy of this approach, we tested six different textures (P1-6) for these patches, covering a range of **Entropy** (complexity) and **Contrast** values. These textures are shown in Figure 8, and our characterization metrics for each are provided in Table 5. Testing a variety of textures allows us to determine which types of textures are most effective in improving pose tracking performance, thereby better informing practical environment texture optimization.

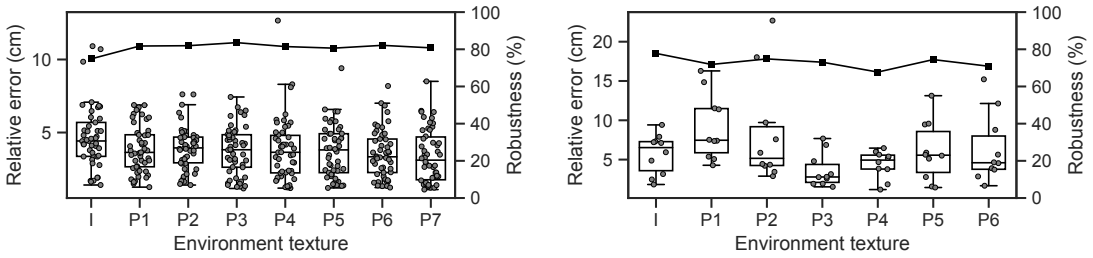
4.4.3 Challenging motion 1: Stationary rotation view change. This type of motion occurs when an AR user turns (i.e., rotates around the vertical axis) to face a different environment region, such as a different wall, without moving to a different position. To isolate this type of motion in a trajectory, we identify three conditions that must be satisfied:

- The device must be relatively stationary, i.e., the **rate of device position change** v_d must be below a threshold v_d^{th}.
- The device must be being moved such that it faces a different environment region, i.e., the **rate of environment view change** v_e must be above a threshold v_e^{th}.
- The motion must involve rotation of the device, i.e., the **accumulated device rotation** θ_d during the period in which the above two conditions are satisfied must be above a threshold θ_d^{th}.

For all experiments in this paper, we set $v_d^{th} = 0.4m/s$, $v_e^{th} = 0.4m/s$, and $\theta_d^{th} = 1rad$.

During this type of motion, the camera view covers a large area of the environment. However, when optimizing the environment using additional texture, we wish to target the region in view around the midpoint of that motion, where rotation speed and motion blur are typically greatest. For these experiments, we calculate the 3D point on an environment surface that the camera is facing (by raycasting along the forward vector defined by the camera pose) halfway through the duration of the motion. If this environment point is a 3D Euclidean distance of at least 1m from any existing texture patches (from other instances of challenging motions), we then add a 1m×1m texture patch to the environment surface, centered at that point.

Because the drop in performance with stationary rotation view change was observed for all the invisible textures we tested, we evaluate our set of texture patches, P1-6, with all of our invisible textures, I1-5. Applying our motion-informed texture optimization method results in four texture patches being added to the environment, and for P1-6, the same texture was applied to all four patches. We also tested an additional condition, P7, in which one instance of textures P3, P4, P5, and P6 was applied to each patch, to evaluate whether texture patch diversity provided any performance benefits.



(a) A2 (with all invisible textures) (b) A6 (with invisible texture I4)

Fig. 9. Results of our motion-informed texture optimization experiments, for (a) the A2 trajectory containing the stationary rotation view change motion, and (b) the A6 trajectory containing the focused orbital inspection motion. For A2 all texture patch conditions (P1-7) resulted in lower median RE than invisible textures alone (I), while for A6 all texture patch conditions (P1-6) except the low **Entropy** P1 texture resulted in lower median RE than invisible textures alone (I).

The results of our texture optimization experiments for the stationary rotation view change motion are shown in Figure 9a. All the texture patch conditions we tested resulted in a reduction in median RE compared to invisible environment textures alone, illustrating the efficacy of our motion-informed texture optimization. The texture which achieved the best performance was P7; after using this texture for the four patches median RE was 3.11cm, compared to 4.42cm with the invisible textures alone. As such, *our texture optimization method reduced pose tracking error by 30% using texture patches covering less 3% of the environment surface area.* This result also indicates that when using multiple texture patches, designers should use diverse textures for best performance.

4.4.4 Challenging motion 2: Focused orbital inspection. This type of motion occurs arises when a user remains focused on the same area of virtual content on the floor (and hence remains focused on the same environment region), but moves around it in a circular or elliptical path to view it from different directions. To isolate this type of motion within a trajectory we identify two conditions which must be satisfied:

- The device must be being moved such that it remains focused on the same environment region; i.e., the **rate of environment view change** v_e must be below a threshold v_e^{th} (here $v_e^{th} = 0.4m/s$).
- The motion must involve rotation of the device around the vertical axis, such that the environment region is viewed from sufficiently different directions; i.e., the **accumulated device rotation** θ_d during the period in which the above condition is satisfied must be above a threshold θ_d^{th} (here $\theta_d^{th} = 1rad$).

During the motion, the AR device camera remains focused on the same environment region. To optimize the environment texture, we add a patch to that region. We calculate the centroid of the points on the environment surface that are hit by raycasting along the camera forward vector each frame during the motion, and add a 1m×1m texture patch centered at that centroid. For these experiments, we evaluate adding our patches P1-6 to the invisible texture I4, which accounted for by far the greatest drop in performance for this motion.

The results of our texture optimization experiments for the focused orbital inspection motion are shown in Figure 9b. All the texture patches we tested except the low **Entropy** P1 texture resulted in a reduction in median RE compared to the use of the invisible environment textures alone, again illustrating the efficacy of our motion-informed texture optimization. The texture patch which resulted in the best performance was P3. When this texture patch was added to invisible texture I4, the median RE was 2.79cm, compared to 6.53cm with I4 alone. The fact that P3 has lower **Entropy** than P4, P5, or P6, but also lower **Edge Density**, indicates that choosing a texture patch with medium rather than high **Edge Density** is more effective for this motion, and is consistent with our analysis of why this motion is challenging. It is also promising that a relatively low **Contrast** texture P3 can improve performance in an environment region a user is focused on, because it shows that designers can add texture patches which will have a limited effect on user perception of virtual content placed above it.

5 A Multiuser Motion Mapping System for Mobile AR

In this section, we present MoMAR, a system for identifying and communicating environment regions where texture patches need to be applied to support the use of invisible textures. First, we describe the motivation and implementation scenario for our system (Section 5.1). Next, we detail our system design, including both the recording of motion data from AR users, and visualizing these data (Section 5.2). Finally, we characterize our system in terms of both latency and user experience, for heterogeneous devices and numbers of users (Section 5.3).

5.1 Motivation and implementation scenario

Our goal is to enable environment designers to optimize environment textures as we demonstrated in Section 4.4, by adding texture patches to an invisible texture. This scenario is compatible with many of the persistent AR use cases we target, in that users may also need to extract information from elements of the real environment; the real objects or features which are required to communicate information can be positioned where texture patches are needed. For example, in the museum scenario we highlighted in Section 1, real artifacts or informational posters could be placed where texture patches are required. In warehouses or hospitals, safety notices or navigational cues could be placed in those regions. Once we communicate the regions in view during challenging device motions, designers then have the flexibility to select texture patches most suited to a specific environment.

However, when adding texture to mitigate the effect of challenging device motion on VI-SLAM performance, it is desirable that where possible, designers avoid regions that are in the camera view when users are inspecting virtual content, because adding texture in these regions may impair user perception of that content. This is more feasible for the stationary rotation view change motion than for the focused rotation inspection motion, because the latter often results from focusing on a virtual object. However, even for the focused rotation inspection, this information can prompt designers to use a low-contrast texture patch, or one that is hidden behind the virtual content. As such, our motion mapping system should communicate both the environment regions in the camera view during challenging motions, and the environment regions in the camera view while viewing virtual content.

In Section 4.4, we identified environment regions in the camera view during different types of motion for a single device trajectory in a game engine-based emulator. A key requirement for the practical deployment of this method is the ability to aggregate device motion characteristics from *multiple* AR users; to obtain a true sense of device motion in an environment our system must capture and synchronize the motion data from all AR sessions that take place there. We highlight four core challenges associated with implementing this *multiuser motion mapping system* for AR devices in a real environment, and how we address them:

- **Unifying coordinate frames:** 3D environment points are in a different coordinate frame for each AR session (with the origin defined by its starting position). However, to aggregate motion map data across sessions, we must combine them into a single coordinate frame. To address this, we leverage the spatial anchors that can be generated and localized within an environment map on modern AR platforms. All users start their experience by localizing against the spatial anchor in a world map generated by the environment administrator (a step that is already required for loading persistent virtual content), and subsequently viewed environment regions are recorded in the coordinate frame defined by that spatial anchor.
- **Minimizing overhead on AR devices:** We must facilitate the analysis of device motion data while incurring minimal overhead on the AR devices already running intensive tasks, such that user experience is not compromised. To this end, we employ a distributed system architecture, in which computation is offloaded from user AR devices to a centralized server.
- **Managing mapping and tracking inconsistencies:** Environment regions recorded from multiple user sessions must be aggregated in a manner that handles mapping and tracking inconsistencies among those sessions. For example, an environment plane may be detected at different depths, but the final motion map should combine these points back into a single plane for visualization purposes. We address this by

calculating the density of points in a grid of spatial regions during motion analysis, and projecting these data to the environment plane detected during data visualization (see Section 5.2).
- **Communicating motion map data:** While raw 3D coordinates can easily be associated with environment regions in a game engine, we require a solution to effectively associate our motion map data with a real-world environment. In MoMAR, we employ situated visualizations [43, 71, 72] to render the motion analysis output directly on top of the real environment, allowing designers to quickly and easily discern regions where texture needs to be added.

Addressing these challenges informs our system design for MoMAR, which we describe below. The code required to implement MoMAR is publicly available at https://github.com/timscargill/MoMAR/.

5.2 System design and implementation

MoMAR is designed for environment texture optimization in spaces that host persistent AR experiences, i.e., those in which the position and orientation of virtual content relative to the real world is consistent across AR devices and sessions, through the saving and loading of a shared world map and spatial anchors. Additionally, we target smartphone or tablet-based AR devices running monocular VI-SLAM (as opposed to AR headsets with multiple tracking cameras), because virtual content stability errors are a prevalent issue on these devices [59], and so we developed our methods for identifying challenging motions for them (Section 4). Because ARKit provides the flexibility to implement a custom networking solution for sharing world maps and spatial anchors, specifically the option for an edge computing-based architecture (ARCore requires cloud connectivity to use Google Cloud Anchors or Azure Spatial Anchors), we choose ARKit for this implementation of MoMAR. However, MoMAR may be implemented for any AR platform that supports spatial anchors, using either an edge or cloud server.

System overview: The MoMAR system architecture is shown in Figure 10. To create a persistent AR experience for which an environment will be optimized, an environment administrator uses the map creation module on an admin AR device to generate a world map and place one or more spatial anchors within that space. These data are then transferred to and stored on the server. When a user starts a new session, the map retrieval module on the user AR device requests the map and anchor data from the map provisioning module on the server. These data are then used to localize the new session within the saved world map. Upon successful localization, the motion logging module on the user AR device is activated, which periodically sends device motion data to the server while the session is active. The motion analysis module on the server can be run on demand or periodically to analyze all user motion data, or those from a specified time range, to produce the motion map data. Finally, the data visualization module on the admin AR device is used to request motion map data from the server and display a motion map using situated visualizations. Below, we provide further details on the operation of each system module.

Map creation: An admin AR device creates the map consisting of two objects: the *world map* itself, an 'ARWorldMap' [3] in ARKit (proprietary .worldmap format, contains feature points, planes, anchor data), and an *anchor dictionary* (.txt format), which associates each unique anchor identifier with the virtual content attached to that anchor. The AR experience administrator performs the standard mapping procedure by moving the device slowly to view all environment regions from different distances and angles. The ARKit ARWorldMappingStatus [4] is used to display whether the space has been sufficiently mapped. The administrator then places spatial anchors on planes detected in the environment as required, by pointing the device towards the appropriate region and pressing UI buttons associated with different types of virtual content. Finally, they press a 'Save' UI button to transfer the world map and anchor dictionary files to the server via an HTTP POST request.

Map provisioning: The map provisioning module is implemented in a Python app on the server. It handles the receipt of the world map and anchor dictionary files from the map creation module using the FastAPI [19]

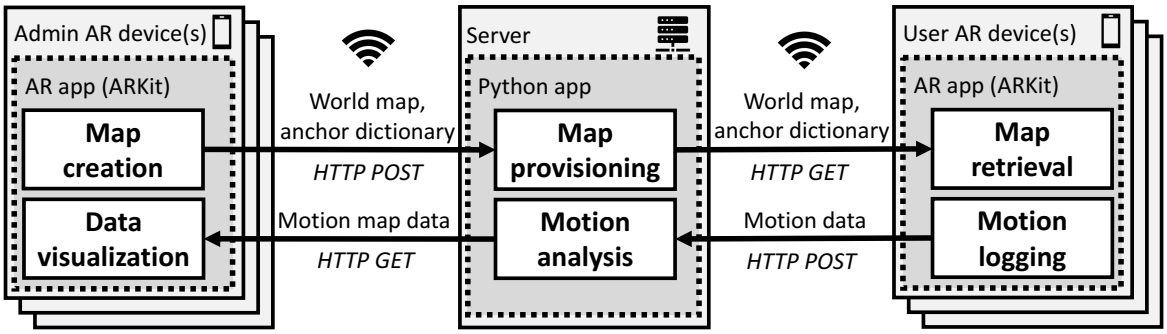


Fig. 10. The architecture of our multiuser motion-mapping system for mobile AR, MoMAR.

Python framework. This module writes them to storage on the server and uses the FastAPI framework to handle HTTP GET requests from user AR devices. The world map and anchor dictionary are then transmitted to those devices accordingly.

Map retrieval: The map retrieval module is implemented within an AR app (the same app that generates the main AR experience) on the user AR device. Upon starting the app, this module generates HTTP GET requests to obtain the world map and anchor dictionary from the server. Because the vast majority of the latency associated with map retrieval and localization is associated with deserializing the world map file, this file is retrieved first, so that the world map deserialization and anchor dictionary retrieval can be performed in parallel. Once the map files have been retrieved and the user has localized within the world map (by focusing the device camera on an environment region containing a spatial anchor), the motion logging module on the user AR device is activated.

Motion logging: The motion logging module is also implemented in the AR app on user AR devices. Once a user has localized within the retrieved map (and any virtual content has been rendered), this module records the following data each frame, which is required for motion analysis:

- **Timestamp:** the timestamp associated with the frame.
- **Device pose:** the estimated position and orientation of the device provided by the AR platform.
- **Environment hit points:** the environment regions in the camera view, in the coordinate frame defined by one of the spatial anchors. Obtained by raycasting from each point in a grid of screen space points and recording the 3D point (in meters to one decimal place, for a map resolution of 0.1m) where each ray intersects with a detected plane in the environment.
- **Gyroscope readings:** the raw gyroscope readings, used to obtain device rotation in the body frame.
- **Virtual object in view:** which virtual object (if any) is in the camera view, obtained by recording the virtual content intersected by screen space raycasts.

These data are associated with a unique device identifier (on ARKit, the UUID [5]) such that the motion logs for individual devices can be distinguished in the case of multiple concurrent users. All the above data – collectively referred to as the *motion data* – recorded in a predefined time window are combined (into one .txt file) and periodically transferred to the server via HTTP POST requests.

Motion analysis: The motion analysis module is implemented in a Python app on the server. This module performs the challenging motion identification we described in Section 4.4, and outputs the environment regions in the camera view during these motions. It also identifies the environment regions in the camera view during virtual object inspection. The conditions for this state are that a virtual object is in the camera view and that the rate of environment viewpoint change (as defined in Section 4.4.1) is lower than a specified threshold v_e^{th}.

First, the environment hit points associated with challenging motion or inspection are calculated for each user session. These points are then combined into a single set of 3D points, with two binary flags that indicate

whether a point is associated with challenging motion and virtual object inspection. Next, we partition the 3D space defined by the extents of the points into a grid of cells (we use cell dimensions of 0.25m×0.25m×0.25m for our implementation), and count the number of points in each cell to calculate the density of points in each environment region (for challenging motion and inspection separately). We select the top N cells with the highest density for challenging motion and inspection (we use $N = 50$ for challenging motion and $N = 200$ for inspection), and for each cell, save the center coordinates and the associated motion type to a .csv file. This file, the *motion map data*, is stored on the server and transferred on request to an admin AR device.

Data visualization: The data visualization module is implemented in an AR app on the admin AR device. When an environment designer wishes to view the motion map generated by MoMAR, they focus on the environment region containing the spatial anchor and press a UI button. This localizes the session within the world map saved on the device and sends an HTTP GET request to the server for the motion map data. When the data visualization module receives the motion map data, it generates the situated visualization using these data. For each of the points defined in the motion map data, a virtual sphere (with a diameter that matches the cell dimensions chosen in the motion analysis module) is rendered at the specified coordinates, in the coordinate frame defined by the spatial anchor. To ensure visualizations appear attached to the real environment in the current session, the spheres are projected to the closest point on a detected plane. The spheres for points associated with inspection are colored purple, and those associated with challenging motion are colored red. The app UI can then be used to display either the regions associated with inspection, as shown in Figure 1c, or those associated with challenging motion, as shown in Figure 1d.

5.3 System characterization

We evaluate MoMAR under realistic conditions by performing two types of system characterization. Firstly, we measure the latency of the motion logging element of MoMAR with varying numbers of user AR devices to test scalability to multiple synchronous users. Secondly, we evaluate user experience by measuring the frames per second that different devices can achieve while MoMAR is running, in addition to resource-intensive tracking and rendering tasks.

5.3.1 System characterization setup. To evaluate MoMAR under realistic conditions, we used the environment and AR app we developed for our case study, described in Section 6. This consisted of an 8m×6m×3m room, which users moved around to view eight virtual objects, including high-resolution and animated 3D models. Standard ARKit mapping and tracking were run on the user AR devices in the background. We tested five user AR devices covering a variety of hardware and software configurations: an iPhone 13 running iOS 16, an iPhone 13 Pro Max running iOS 15, an iPhone 14 Pro Max running iOS 17, an iPad Pro 2nd generation running iPad OS 17, and an iPad Pro 4th generation running iPad OS 16. We used a desktop PC with an Intel i7-9700K CPU and an Nvidia GeForce RTX 2060 GPU as an edge server, placed in the same room. The user AR devices and the server were all connected to a 5GHz wireless local area network, with the router also in the same room.

5.3.2 Motion logging latency. We measured the end-to-end latency of the motion logging component of MoMAR by calculating the amount of time between sending an HTTP POST request with the motion data from a user AR device and the user AR device receiving a response to that request from the server. We tested one, two, three, four, and five devices running concurrently, with the latency logged for two minutes for each setting. The results of our motion logging latency experiments are shown in Table 6a (calculated from the latency recorded for all sessions). Mean latency remained less than 100ms even when five devices ran MoMAR concurrently. This shows that MoMAR is suitable for practical multiuser AR scenarios such as collaborative tasks or shared experiences.

5.3.3 User Experience. We evaluated user experience by recording the current number of frames per second (fps) rendered by a device, an important metric for AR [37, 53]. We tested each of the five AR devices listed

Table 6. Results of our system characterization for MoMAR: The mean and standard deviation (SD) of (a) the motion logging latency for multiple concurrent AR sessions running MoMAR, and (b) the frames per second rendered on various AR devices while running MoMAR in addition to a resource-intensive AR session.

(a) Motion logging latency

Concurrent sessions	Mean (ms)	SD (ms)
1	63.8	37.8
2	86.2	128.7
3	99.5	264.6
4	92.0	136.8
5	76.0	155.5

(b) User experience (frames per second)

Device	Mean (fps)	SD (fps)
iPhone 13 (iOS 16)	59.2	9.9
iPhone 13 Pro Max (iOS 15)	59.2	9.4
iPhone 14 Pro Max (iOS 17)	59.6	8.5
iPad 2nd Gen. (iPad OS 17)	59.6	4.9
iPad 4th Gen. (iPad OS 16)	59.9	4.8

in Section 5.3.1, and recorded the fps values throughout the realistic AR task in our case study, described in Section 6.3. The results of our user experience experiments are shown in Table 6b. While running MoMAR in addition to resource-intensive tracking and rendering tasks, all five iOS devices we tested maintained an average frame rate greater than 59fps. This demonstrates how MoMAR can be deployed on devices with a variety of hardware and software capabilities without compromising user experience.

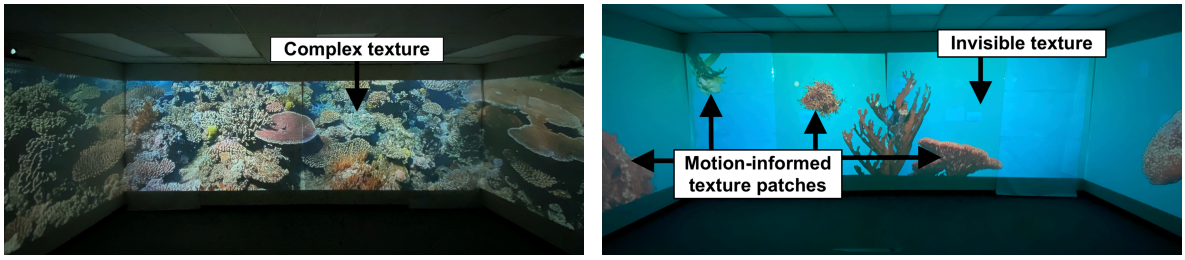
6 Case Study: Environment Texture Optimization for an AR Museum Exhibit

Finally, we demonstrate the use of MoMAR in a realistic persistent AR scenario, optimizing environment texture for an immersive museum exhibit, and conduct a study to compare user experiences with a complex environment texture and an optimized environment texture. First, we outline the motivation and overview for this case study (Section 6.1), and provide details of the complex and optimized environment textures we used for our study (Section 6.2). We then describe our study design (Section 6.3), and present our results (Section 6.4).

6.1 Case study motivation and overview

The goal of this case study was to demonstrate the use of MoMAR to optimize an environment for a specific persistent AR experience, in which virtual content appears at consistent locations relative to the real world. We targeted a highly mobile AR scenario in which users move around and interact with virtual content in different areas of an environment, with the potential for the challenging device motions we identified in Section 4. To create a more informative and impactful case study, we designed an experience representative of a real AR use case, one that encourages natural user motion and meaningful interaction with virtual content — i.e., one in which users behave as they might in an industrial, commercial, or educational setting. This allowed us to better capture the issues that environment designers may face in real deployments.

To this end, we developed a multimedia persistent AR experience, *Dive*, which is a representative of an immersive museum exhibit (e.g., [29, 46, 49, 64, 67]). *Dive* allows users to explore a virtual coral reef, which combines 3D marine animals displayed on a mobile AR device, 2D projections of an underwater background on the walls, and underwater audio effects. Consistent with our museum setting, we based our study on an educational context, in which users learn about different marine animals by answering questions about them in a mobile AR app. Users were required to locate the animal in each question in the environment, then inspect the animal to answer a question about their appearance. Examples of these marine animals and associated questions are shown in Figure 12. This design allowed us to study the effect of environment texture on both a visual search task [17] and a visual inspection task [62], both of which are applicable to many other AR scenarios, such as factory or warehouse settings.



(a) Complex environment texture (b) Optimized environment texture

Fig. 11. The two environment textures we tested in our case study on an AR museum exhibit. The complex texture was designed solely to maximize virtual content stability (pose tracking performance), while the optimized texture was created using the guidance generated by MoMAR, taking into account both virtual object stability and visibility.

6.2 Case study environment textures

In this case study, we compared the efficacy of an environment texture optimized using MoMAR to a complex environment texture, the latter being a naive approach designed solely to support good tracking performance. To remain consistent with the underwater theme of our *Dive* AR experience, we chose a dense collection of coral as the complex texture and an image of water that met the specifications for an invisible texture, with texture patches of seaweed and coral. To design the optimized texture – i.e., to inform the placement of those texture patches – we used MoMAR to capture and analyze the motion data from 15 users who experienced *Dive* with the complex texture. Guided by the situated visualizations it generated, we added texture patches to the invisible water texture in environment regions associated with challenging motion, e.g., the red regions in Figure 1d, while avoiding those associated with virtual content inspection, e.g., the purple regions in Figure 1c. We used four projectors to display a texture (i.e., 2D projections) in four segments (each segment being a 1000px×600px image) on three of the four walls in the study environment. The surface area covered by the projections consisted of four areas of approximately 4m×2.4m. The projections of the complex and optimized environment textures are shown in Figure 11, and the textures as viewed behind the animals on the AR device are shown in Figure 12.

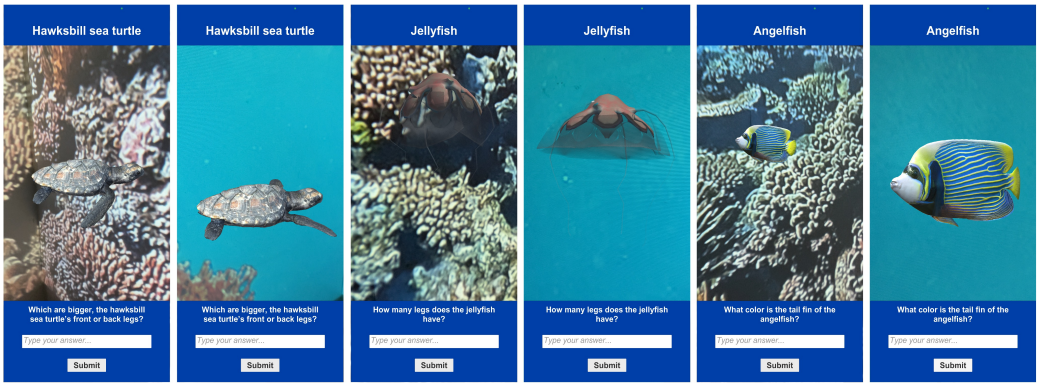
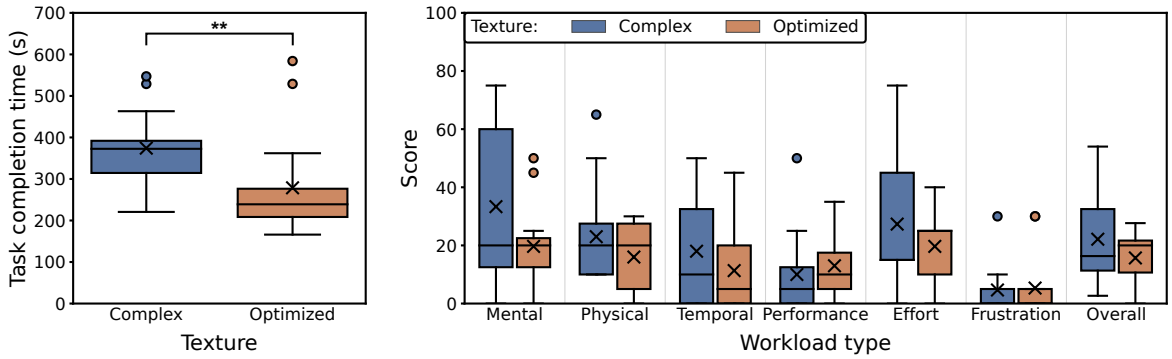


Fig. 12. Representative user views through the mobile AR device during our immersive museum experience, *Dive*, for three animals, with both the complex and optimized environment texture for each. Study participants were tasked with locating different animals, then answering questions about their appearance in the app UI.

6.3 Case study design

To evaluate the optimized environment texture we created using the MoMAR guidance, we conducted an IRB-approved user study with a between-subjects design; participants experienced the immersive exhibit with either the complex environment texture or the optimized environment texture (detailed above in Section 6.2).

Participants: We recruited 30 participants (8 female, 22 male, aged 19 to 67) from our personal and professional networks. All participants had normal or corrected to normal eyesight. Participants were split into two groups of 15, with each group assigned to one of the two texture conditions.
Experiment setup and apparatus: The study was conducted in controlled conditions in a lab environment with no windows and white walls. We covered any textures on the walls (e.g., power outlets) with white paper. We projected the environment textures onto the walls using four AuKing 2023 1080P mini projectors, each connected via HDMI to a Raspberry Pi 3B which supplied the texture source image. No environment lighting was used aside from the projected textures. An iPhone 13 Pro Max running iOS 15 was used as the AR device. For the main task, participants started by standing 1.5m away from the middle of the environment textures (where the reference spatial anchor was placed, the middle of the images in Figure 11), with the AR device facing the wall. We connected a JBL Flip 4 speaker to the server via Bluetooth to play the underwater audio effects.
Task: Prior to starting the main quiz task, participants were required to localize the AR device in the environment; after positioning themselves in the starting pose, with the AR device facing the center of the environment textures, participants opened the *Dive* AR app, which automatically started the download of the world map and anchor dictionary files from the server. Participants were guided by UI text to move the AR device slowly while remaining focused on the same environment region, to achieve tracking initialization and localization within the downloaded world map. Once localization was completed, the UI was updated to show the first quiz question, and motion logging was activated – this marked the start of the main task. Participants were then free to move around the environment however they wished to answer the quiz questions about the marine animals; prior to the task they were instructed that animals would only appear in front of the projected textures. Participants answered 16 questions about eight animals in a consistent order, with the two questions about each animal always consecutive. Participants entered their answers to the questions in a text box, navigated to the next question using a 'Submit' UI button with an audio effect, and were notified with UI text and an audio effect when the task was complete.
Dependent variables and operationalization: We measured *efficiency* using the task duration (not including localization), trajectory length (the sum of position differences between consecutive device pose estimates), and total device rotation (the sum of rotation magnitudes between consecutive device pose estimates). We also evaluated *virtual content stability* through a post-experiment questionnaire, and the *workload* experienced by the user using the NASA-TLX. For virtual content stability, participants were asked to rate on a 5-point Likert scale how much they agreed with the statement "*The guidance was stable (not moving or jittering etc.)*", using the following responses: 1=Strongly disagree, 2=Disagree, 3=Neither agree nor disagree, 4=Agree, 5=Strongly agree. Because the answers to some of the quiz questions had a subjective element, e.g., the color of some of the animals, we did not analyze task accuracy (i.e., correct answers) as a dependent variable.
Procedure: We began by introducing participants to the study and asking them to sign a consent form if they were willing to take part. Participants were informed of the general purpose of the study, which was to evaluate their experience with the *Dive* AR experience. However, they were not informed about the environment texture variable or the experiment hypothesis. Participants answered a pre-experiment questionnaire using Qualtrics to capture demographic data, which included confirmation of vocabulary knowledge required to answer questions about the anatomy of the marine animals. They then completed a training period to become familiar with the localization process and the quiz task. The study administrator provided a verbal overview of the task, demonstrated the localization and question-answering processes (in a separate environment area and with a



(a) Task duration

(b) NASA-TLX

Fig. 13. Results from our case study, which compared the use of a complex environment texture with an optimized texture in an AR museum setting. Participants completed the task significantly faster in the optimized environment (p=0.003).

different animal and question from the main task), and allowed the participant to complete this training task until they were comfortable with performing it independently. Participants then performed the main task, with the study administrator available for technical support or to answer process clarification questions if needed. After completing the main task participants completed the NASA-TLX on paper and a post-experiment questionnaire using Qualtrics.

6.4 Case study results

The results of our study for each of our dependent variables are detailed below. We used the Mann-Whitney U Test to test for statistical significance without the assumption that the data is normally distributed.

Efficiency: The results of our study for task duration are shown in Figure 13a. *Participants completed the task significantly faster with our optimized environment texture than with the original complex texture (p=0.003, complex texture mean = 374.25s, optimized texture mean = 278.73s).* Similarly, less device movement was performed with the optimized texture than with the complex texture. Trajectory length was significantly shorter with the optimized texture than with the complex texture (p=0.011, complex texture mean = 66.11m, optimized texture mean = 50.53m). Total device rotation was also significantly lower with the optimized texture (p=0.011, complex texture mean = 6999.35°, optimized texture mean = 5478.21°).
Virtual content stability: The mean virtual content stability perceived by users (after assigning the values 1 to 5 to responses on the 5-point Likert scale) was 4.6 for the complex texture and 4.5 for the optimized texture. We found no significant difference between the two texture conditions, showing that *the environment texture optimized using MoMAR was able to achieve the same level of perceived virtual object stability as the complex texture.*
Workload: The results of the NASA-TLX for each texture condition are shown in Figure 13b. We found no significant difference between the complex and optimized texture on any of the workload scales.

6.5 Case study discussion

Overall, our results show that an optimized texture created using the guidance from MoMAR can support the same level of perceived virtual content stability as a complex texture, while allowing users to complete an AR task faster at the same time. Participants spent significantly less time inspecting virtual content with the optimized texture, indicating that locating and perceiving details of virtual objects takes longer with more complex background

textures. This improvement in task efficiency with the optimized texture was also reflected in our device motion measures; our results indicate that less movement was required to complete the task with an optimized texture.

Interestingly, we did not find a significant reduction in workload for the optimized texture. This was somewhat surprising given that the amount of device motion required was significantly lower, and that we had observed significantly lower workload for an invisible texture in our previous study (Section 3). Mean workload was lower for the optimized texture on every scale though, except for the performance and frustration scales which had low ratings overall. One possible reason for the lack of significance between texture conditions is the varying degree to which participants found the task challenging; in general we observed a greater variance in workload ratings compared to our previous study. It may be useful for future studies with a similar number of participants to consider this effect, when designing the task that AR users will perform.

7 Limitations and Future Work

Our current evaluation of MoMAR is based on an optimized texture created using the MoMAR guidance. Future work on MoMAR or similar systems that provide texture optimization guidance should also be evaluated from a system usability perspective, with participants from the target audience, environment designers. Different types of situated visualizations should also be tested; there are opportunities to evaluate and improve the usability and aesthetic quality of the current motion map by experimenting with different types of virtual objects as well as spheres to highlight environment regions. Our study also only covers one type of AR task and environment, so evidence for the efficacy of MoMAR and texture optimization in general would be strengthened by studies with different types of AR tasks and diverse real environments, e.g., moving objects in a warehouse.

One limitation of the MoMAR system as it stands is that it was developed and evaluated specifically with devices running monocular VI-SLAM, i.e., smartphones and tablets (because accurate pose tracking is particularly challenging on these devices). AR headsets with multiple tracking cameras are increasingly being deployed as well, especially in industrial settings, so future work should evaluate the use of these devices in scenarios with dynamic motion. It will be important to establish whether invisible textures alone are sufficient to support good virtual content stability on these devices, and if not, what types of device motions result in greater pose tracking error. There is also the possibility of heterogenous co-located AR devices, so future work should assess the challenges associated with designing textures that support different types of tracking systems and displays.

Additionally, the game engine-based experiments we used to inform the development of MoMAR did not replicate the localization step that is required at the start of persistent AR experiences. The choice of a texture with sufficient visual features for reliable and fast spatial anchor-based localization is essential, and our choice of texture for an anchor region in our case study (Section 6) was based on experience and experimentation, which is not efficient for designers. Future work should investigate how to incorporate localization using spatial anchors into game engine-based simulations, such that ideal texture properties for those anchors can be defined. Similarly, more investigation of the textures required specifically for recognizing previously visited places will be beneficial, especially for working with larger environments. We anticipate that determining the environment regions users return to most frequently, or visit after navigating a particularly challenging area for pose tracking, will also be areas where texture patches could improve performance.

Finally, the MoMAR system could be extended to not only guide the positioning of texture patches, but also the *content* of texture patches. In the current system we largely leave decisions on patch content to the discretion of designers, but future systems could display virtual content with visual properties representative of a suitable texture patch, or even evaluate the suitability of a texture adjustment the designer makes. A tantalizing prospect specific to projected environment textures is automatic texture optimization, with texture adjusted directly using the output of user motion analysis (using, for example, deep learning-based texture generation [26]), without the need for human intervention.

8 Conclusion

This article evaluated the use of low-contrast 'invisible' environment textures for AR. First, we conducted a user study that compared the effect of a complex and an invisible environment texture in a realistic scenario, an AR assembly task. We then performed extensive game engine-based experiments that revealed the impact of environment texture on pose tracking performance with more dynamic device motion, and developed a method to optimize environment textures accordingly. To support real deployments we presented MoMAR, the first system that aggregates the motion data from multiple synchronous or asynchronous AR users to guide environment texture optimization for persistent markerless AR experiences. We implemented MoMAR for ARKit, and showed it can be deployed with a variety of AR devices without impacting user experience. Finally, through a realistic case study we demonstrated that using MoMAR to guide environment texture optimization, instead of using a complex texture, benefits user perception of virtual content without compromising virtual object stability.

Acknowledgments

We thank our study participants for their assistance in this research. This work was supported in part by NSF grants CNS-1908051, CNS-2112562, CSR-2312760 and IIS-2231975, NSF CAREER Award IIS-2046072, a Meta Research Award and a CISCO Research Award.

References

[1] Al Horr, Y., Arif, M., Kaushik, A., Mazroei, A., Katafygiotou, M., and Elsarrag, E. Occupant productivity and office indoor environment quality: A review of the literature. *Building and Environment 105* (2016), 369–389.

[2] Alavi, H. S., Verma, H., Mlynar, J., and Lalanne, D. On the temporality of adaptive built environments. *People, Personal Data and the Built Environment* (2019), 13–40.

[3] Apple. ARWorldMap. https://developer.apple.com/documentation/arkit/arworldmap, 2024.

[4] Apple. ARWorldMappingStatus. https://developer.apple.com/documentation/arkit/arworldmappingstatus, 2024.

[5] Apple. UUID. https://developer.apple.com/documentation/foundation/uuid, 2024.

[6] Baird, K. M., and Barfield, W. Evaluating the effectiveness of augmented reality displays for a manual assembly task. *Virtual Reality 4* (1999), 250–259.

[7] Blender. Blender. https://www.blender.org/, 2024.

[8] Bressa, N., Korsgaard, H., Tabard, A., Houben, S., and Vermeulen, J. What's the situation with situated visualization? A survey and perspectives on situatedness. *IEEE Transactions on Visualization and Computer Graphics 28*, 1 (2021), 107–117.

[9] Bressa, N., Vermeulen, J., and Willett, W. Data every day: Designing and living with personal situated visualizations. In *Proceedings of ACM CHI* (2022).

[10] Büschel, W., Lehmann, A., and Dachselt, R. MIRIA: A mixed reality toolkit for the in-situ visualization and analysis of spatio-temporal interaction data. In *Proceedings of ACM CHI* (2021).

[11] Campos, C., Elvira, R., Rodríguez, J. J. G., Montiel, J. M., and Tardós, J. D. ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap SLAM. *IEEE Transactions on Robotics* (2021), 1–17.

[12] Cannon, M. W., and Fullenkamp, S. C. Spatial interactions in apparent contrast: Inhibitory effects among grating patterns of different spatial frequencies, spatial positions and orientations. *Vision Research 31*, 11 (1991), 1985–1998.

[13] Corchs, S. E., Ciocca, G., Bricolo, E., and Gasparini, F. Predicting complexity perception of real world images. *PloS One 11*, 6 (2016), e0157986.

[14] De Amicis, R., Riggio, M., Shahbaz Badr, A., Fick, J., Sanchez, C. A., and Prather, E. A. Cross-reality environments in smart buildings to advance STEM cyberlearning. *International Journal on Interactive Design and Manufacturing (IJIDeM) 13* (2019), 331–348.

[15] Debernardis, S., Fiorentino, M., Gattullo, M., Monno, G., and Uva, A. E. Text readability in head-worn displays: Color and style optimization in video versus optical see-through devices. *IEEE Transactions on Visualization and Computer Graphics 20*, 1 (2013), 125–139.

[16] Drouot, M., Le Bigot, N., Bolloc'h, J., Bricard, E., de Bougrenet, J.-L., and Nourrit, V. The visual impact of augmented reality during an assembly task. *Displays 66* (2021), 101987.

[17] Eckstein, M. P. Visual search: A retrospective. *Journal of Vision 11*, 5 (2011), 14–14.

[18] Ellemberg, D., Allen, H. A., and Hess, R. F. Investigating local network interactions underlying first-and second-order processing. *Vision Research 44*, 15 (2004), 1787–1797.

[19] FastAPI. FastAPI. https://fastapi.tiangolo.com/, 2024.

[20] Fisher, A. V., Godwin, K. E., and Seltman, H. Visual environment, attention allocation, and learning in young children: When too much of a good thing may be bad. *Psychological Science 25*, 7 (2014), 1362–1370.

[21] Fleck, P., Calepso, A. S., Hubenschmid, S., Sedlmair, M., and Schmalstieg, D. RagRug: A toolkit for situated analytics. *IEEE Transactions on Visualization and Computer Graphics* (2022).

[22] Funk, M., Kosch, T., Greenwald, S. W., and Schmidt, A. A benchmark for interactive augmented reality instructions for assembly tasks. In *Proceedings of ACM MUM* (2015).

[23] Gabbard, J. L., Swan, J. E., and Hix, D. The effects of text drawing styles, background textures, and natural lighting on text legibility in outdoor augmented reality. *Presence 15*, 1 (2006), 16–32.

[24] Gabbard, J. L., Swan, J. E., Hix, D., Kim, S.-J., and Fitch, G. Active text drawing styles for outdoor augmented reality: A user-based study and design implications. In *Proceedings of IEEE VR* (2005).

[25] Gabbard, J. L., Swan, J. E., Hix, D., Schulman, R. S., Lucas, J., and Gupta, D. An empirical user-based study of text drawing styles and outdoor background textures for augmented reality. In *Proceedings of IEEE VR* (2005).

[26] Gan, Y., Chi, H., Gao, Y., Liu, J., Zhong, G., and Dong, J. Perception driven texture generation. In *Proceedings of IEEE ICME* (2017).

[27] Gattullo, M., Uva, A. E., Fiorentino, M., and Gabbard, J. L. Legibility in industrial AR: Text style, color coding, and illuminance. *IEEE Computer Graphics and Applications 35*, 2 (2015), 52–61.

[28] Groissboeck, W., Lughofer, E., and Thumfart, S. Associating visual textures with human perceptions using genetic algorithms. *Information Sciences 180*, 11 (2010), 2065–2084.

[29] Hamburger Kunsthalle. Augmented reality application on Emil Nolde's painting technique. https://www.hamburger-kunsthalle.de/en/augmented-reality-application-emil-noldes-painting-technique-0, 2024.

[30] Hart, S. G., and Staveland, L. E. Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. In *Advances in Psychology*, vol. 52. Elsevier, 1988, pp. 139–183.

[31] Itti, L., Koch, C., and Niebur, E. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence 20*, 11 (1998), 1254–1259.

[32] Jinyu, L., Bangbang, Y., Danpeng, C., Nan, W., Guofeng, Z., and Hujun, B. Survey and evaluation of monocular visual-inertial SLAM algorithms for augmented reality. *Virtual Reality & Intelligent Hardware 1*, 4 (2019), 386–410.

[33] Kotis, K., and Soularidis, A. ReconTraj4Drones: A framework for the reconstruction and semantic modeling of UAVs' trajectories on MovingPandas. *Applied Sciences 13*, 1 (2023), 670.

[34] Kümmerle, R., Steder, B., Dornhege, C., Ruhnke, M., Grisetti, G., Stachniss, C., and Kleiner, A. On measuring the accuracy of SLAM algorithms. *Autonomous Robots 27* (2009), 387–407.

[35] Kurzhals, K., Becher, M., Pathmanathan, N., and Reina, G. Evaluating situated visualization in AR with eye tracking. In *Proceedings of IEEE BELIV* (2022).

[36] Lanir, J., Kuflik, T., Sheidin, J., Yavin, N., Leiderman, K., and Segal, M. Visualizing museum visitors' behavior: Where do they go and what do they do there? *Personal and Ubiquitous Computing 21* (2017), 313–326.

[37] Li, M., Arning, K., Vervier, L., Ziefle, M., and Kobbelt, L. Influence of temporal delay and display update rate in an augmented reality application scenario. In *Proceedings of ACM MUM* (2015).

[38] Li, Q., Zhang, T., Wang, H., and Zeng, Z. Dynamic accessibility mapping using floating car data: A network-constrained density estimation approach. *Journal of Transport Geography 19*, 3 (2011), 379–393.

[39] Liu, J., Lughofer, E., and Zeng, X. Could linear model bridge the gap between low-level statistical features and aesthetic emotions of visual textures? *Neurocomputing 168* (2015), 947–960.

[40] Liu, J.-C., Li, K.-A., Yeh, S.-L., and Chien, S.-Y. Assessing perceptual load and cognitive load by fixation-related information of eye movements. *Sensors 22*, 3 (2022), 1187.

[41] Liu, P., Zuo, X., Larsson, V., and Pollefeys, M. MBA-VO: Motion blur aware visual odometry. In *Proceedings of the IEEE/CVF ICCV* (2021).

[42] Microsoft. HoloLens environment considerations. https://learn.microsoft.com/en-us/hololens/hololens-environment-considerations, 2024.

[43] Moere, A. V., and Hill, D. Designing for the situated and public visualization of urban data. In *Street Computing*. Routledge, 2016, pp. 24–45.

[44] MRTK2. Mrtk2. https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/, 2024.

[45] Mustaniemi, J., Kannala, J., Särkkä, S., Matas, J., and Heikkilä, J. Fast motion deblurring for feature detection and matching using inertial measurements. In *Proceedings of IEEE ICPR* (2018).

[46] Muséum national d'Histoire naturelle. Revivre, extinct animals in augmented reality. https://www.mnhn.fr/en/experience/revivre-extinct-animals-in-augmented-reality, 2024.

[47] Näsänen, R., Ojanpää, H., and Kojo, I. Effect of stimulus contrast on performance and eye movements in visual search. *Vision Research 41*, 14 (2001), 1817–1824.

[48] Polycam. Polycam. https://poly.cam/, 2024.

[49] Pérez Art Museum Miami. New Realities. https://www.pamm.org/en/new-realities/, 2024.
[50] Qin, T., Li, P., and Shen, S. VINS-Mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics 34*, 4 (2018), 1004–1020.
[51] Qualtrics. Qualtrics. https://www.qualtrics.com, 2024.
[52] Ran, X., Slocum, C., Gorlatova, M., and Chen, J. ShareAR: Communication-efficient multi-user mobile augmented reality. In *Proceedings of ACM HotNets* (2019).
[53] Ren, J., Gao, L., Wang, X., Ma, M., Qiu, G., Wang, H., Zheng, J., and Wang, Z. Adaptive computation offloading for mobile augmented reality. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies 5*, 4 (2021), 1–30.
[54] Rosenholtz, R., Li, Y., and Nakano, L. Measuring visual clutter. *Journal of Vision 7*, 2 (2007), 17–17.
[55] Rosten, E., and Drummond, T. Machine learning for high-speed corner detection. In *Proceedings of ECCV* (2006).
[56] Scargill, T., Chen, Y., Hu, T., and Gorlatova, M. SiTAR: Situated trajectory analysis for in-the-wild pose error estimation. In *Proceedings of IEEE ISMAR* (2023).
[57] Scargill, T., Chen, Y., Marzen, N., and Gorlatova, M. Integrated design of augmented reality spaces using virtual environments. In *Proceedings of IEEE ISMAR* (2022).
[58] Scargill, T., Hadziahmetovic, M., and Gorlatova, M. Invisible textures: Comparing machine and human perception of environment texture for ar. In *Proceedings of ACM ImmerCom (co-located with ACM MobiCom)* (2023).
[59] Scargill, T., Premsankar, G., Chen, J., and Gorlatova, M. Here to stay: A quantitative comparison of virtual object stability in markerless mobile AR. In *Proceedings of IEEE/ACM CPHS Workshop (co-located with CPS-IoT week)* (2022).
[60] Schallmo, M.-P., and Murray, S. O. Identifying separate components of surround suppression. *Journal of Vision 16*, 1 (2016), 2–2.
[61] Schubert, D., Goll, T., Demmel, N., Usenko, V., Stückler, J., and Cremers, D. The TUM VI benchmark for evaluating visual-inertial odometry. In *Proceedings of IEEE/RSJ IROS* (2018).
[62] See, J. E., Drury, C. G., Speed, A., Williams, A., and Khalandi, N. The role of visual inspection in the 21st century. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* (2017).
[63] Shannon, C. E. A mathematical theory of communication. *The Bell System Technical Journal 27*, 3 (1948), 379–423.
[64] Smithsonian. Critical distance. https://naturalhistory.si.edu/exhibits/critical-distance, 2024.
[65] Sundstedt, V., and Garro, V. A systematic review of visualization techniques and analysis tools for eye-tracking in 3D environments. *Frontiers in Neuroergonomics 3* (2022), 910019.
[66] Tang, A., Owen, C., Biocca, F., and Mou, W. Comparative effectiveness of augmented reality in object assembly. In *Proceedings of ACM CHI* (2003).
[67] The Dalí Museum. Visual magic: Dalí's masterworks in AR. https://thedali.org/exhibit/visual-magic-dalis-masterworks-in-augmented-reality/, 2024.
[68] Treisman, A. M., and Gelade, G. A feature-integration theory of attention. *Cognitive Psychology 12*, 1 (1980), 97–136.
[69] Unity. Unity. https://unity.com/, 2024.
[70] Wang, C., Lu, W., Ohno, R., and Gu, Z. Effect of wall texture on perceptual spaciousness of indoor space. *International Journal of Environmental Research and Public Health 17*, 11 (2020), 4177.
[71] White, S., and Feiner, S. SiteLens: Situated visualization techniques for urban site visits. In *Proceedings of ACM CHI* (2009).
[72] Willett, W., Jansen, Y., and Dragicevic, P. Embedded data representations. *IEEE Transactions on Visualization and Computer Graphics 23*, 1 (2016), 461–470.
[73] Wolfe, J. M., Oliva, A., Horowitz, T. S., Butcher, S. J., and Bompas, A. Segmentation of objects from backgrounds in visual search tasks. *Vision Research 42*, 28 (2002), 2985–3004.
[74] Xing, J., and Heeger, D. J. Measurement and modeling of center-surround suppression and enhancement. *Vision Research 41*, 5 (2001), 571–583.
[75] Yang, Z., Shi, J., Jiang, W., Sui, Y., Wu, Y., Ma, S., Kang, C., and Li, H. Influences of augmented reality assistance on performance and cognitive loads in different stages of assembly task. *Frontiers in Psychology 10* (2019), 1703.
[76] Ze-Yuan, C., and Jeong-Won, H. Exploring the cognitive attributes of visual texture in architectural materials-focused on the vein, grain, pattern. *Journal of the Architectural Institute of Korea 39*, 8 (2023), 67–78.
[77] Zhang, L., Finkelstein, A., and Rusinkiewicz, S. High-precision localization using ground texture. In *Proceedings of IEEE ICRA* (2019).
[78] Zhang, L., and Murdoch, M. J. Perceived transparency in optical see-through augmented reality. In *Proceedings of IEEE ISMAR-Adjunct* (2021).
[79] Zhang, Z., and Scaramuzza, D. A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry. In *Proceedings of IEEE/RSJ IROS* (2018).