GLEAM: An Illumination Estimation Framework for Real-time Photorealistic Augmented Reality on Mobile Devices

Siddhant Prakash
sprakas9@asu.edu
Arizona State University

Alireza Bahremand
abahrema@asu.edu
Arizona State University

Linda D. Nguyen
ldnguye4@asu.edu
Arizona State University

Robert LiKamWa
likamwa@asu.edu
Arizona State University

ABSTRACT

Mixed reality mobile platforms attempt to co-locate virtual scenes with physical environments, towards creating immersive user experiences. However, to create visual harmony between virtual and physical spaces, the virtual scene must be accurately illuminated with realistic lighting that matches the physical environment. To this end, we design GLEAM, a framework that provides robust illumination estimation in real-time by integrating physical light-probe estimation with current mobile AR systems. GLEAM visually observes reflective objects to compose a realistic estimation of physical lighting. Optionally, GLEAM can network multiple devices to sense illumination from different viewpoints and compose a richer estimation to enhance realism and fidelity.

Using GLEAM, AR developers gain the freedom to use a wide range of materials, which is currently limited by the unrealistic appearance of materials that need accurate illumination, such as liquids, glass, and smooth metals. Our controlled environment user studies across 30 participants reveal the effectiveness of GLEAM in providing robust and adaptive illumination estimation over commercial status quo solutions, such as pre-baked directional lighting and ARKit 2.0 illumination estimation. Our benchmarks reveal the need for situation driven tradeoffs to optimize for quality factors in situations requiring freshness over quality and vice-versa. Optimizing for different quality factors in different situations, GLEAM can update scene illumination as fast as 30 ms by sacrificing richness and fidelity in highly dynamic scenes, or prioritize quality by allowing an update interval as high as 400 ms in scenes that require high-fidelity estimation.

CCS CONCEPTS

• **Human-centered computing** → **Mobile computing**; **Mixed / augmented reality**; **User studies**; • **Computing methodologies** → *Active vision*; Image processing; • **Software and its engineering** → *Real-time systems software*; • **Computer systems organization** → Distributed architectures.

MobiSys '19, June 17–21, 2019, Seoul, Republic of Korea

ACM ISBN 978-1-4503-6661-8/19/06…$15.00
https://doi.org/10.1145/3307334.3326098

KEYWORDS

augmented reality; light estimation; image-based lighting; light probe; lighting models; image processing; geometry

ACM Reference Format:
Siddhant Prakash, Alireza Bahremand, Linda D. Nguyen, and Robert LiKamWa. 2019. GLEAM: An Illumination Estimation Framework for Real-time Photorealistic Augmented Reality on Mobile Devices. In *The 17th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '19), June 17–21, 2019, Seoul, Republic of Korea.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3307334.3326098

1 INTRODUCTION

Light estimation is a critical component for photorealistic rendering of virtual scenes. For augmented reality (AR), a merging of virtual and physical worlds, accurate light estimation is especially important; inaccuracies in light estimation create noticeable visual inconsistencies between the virtual scene and physical environment, as shown in Figure 1. Lighting inconsistencies remove the user from the immersive experience, whether or not the user is consciously aware of the inaccuracy in the scene illumination [8, 22]. This is more problematic for certain virtual materials. While matte "Lambertian" surfaces suffice with most forms of virtual lighting, correct representation of specular surfaces requires a rich estimation of lighting [23, 28]. Without capabilities of illumination estimation, AR developers are forced to avoid the use of even partially reflective materials, such as glass, liquid, and polished metal.

Sufficient light estimation requires not only the intensity of light, but also the directionality of light. Furthermore, light estimation must be updated in real-time, adjusting to changes in the dynamic environment of a physical setting, e.g., people casting shadows, opening/closing doors, or turning on/off lights. Consequently, current approaches have thus far been inadequate. At this time, released Google ARCore implementations [13] provide coarse illumination estimation through ambient light sensing of average pixel values in a scene. Meanwhile, Apple ARKit [2] and academic research solutions sample light transmissions from the scene geometry [24, 25, 34] and use machine learning inference to estimate directional light intensity [9, 20]. These solutions can be computationally expensive and slow to update. We measure that ARKit updates its illumination every 3.7 sec. Furthermore, these techniques are prone to inaccuracy when filling in missing information.

Thus, towards real-time high-fidelity estimation, we design GLEAM, a software framework to ***G**enerate **L**ight **E**stimation for **A**R on **M**obile*

Figure 1: AR scene illuminated with ARKit (left) and GLEAM (right) illumination estimation, along with reflected environment in light probe (inset). GLEAM creates visual harmony between objects in virtual scenes and physical environments for improved photorealism.

systems in real time. We integrate *light probe estimation* from the graphics community, which uses physical reflections to estimate dynamic real-time illumination from several directions. GLEAM observes images of geometrically-tracked reflective light probes, which can be attached to hand-held controllers, game pieces, or other physical objects. Integrated with AR camera tracking engines for streamlined processing, GLEAM uses images of light probes to estimate incoming light in the physical environment.

We develop GLEAM through a set of computational modules for radiance sampling and cubemap composition. The construction of these modules is computationally efficient by design; radiance sampling leverages existing optimized game engine infrastructure for geometric processing, while cubemap composition involves only lightweight interpolation techniques. Furthermore, our Unity-based implementation takes care to not interrupt the main thread. Instead, GLEAM launches an auxiliary thread to minimize the influence of estimation overhead on the continuous runtime of the application.

We also design an optional network transfer module to allow nearby devices to share radiance sample information. A single camera viewpoint of a light probe will provide a basis of illumination estimation, but can be improved with information not captured in its viewpoint. Thus, when available – such as in a classroom or museum – multiple AR devices can share viewpoint-specific data to jointly improve illumination estimation.

The quality of estimation depends on multiple factors: coverage, resolution, freshness and update interval. We provide tradeoff mechanisms to balance the quality factors to the virtual and physical needs and expectations. If the scene demands high resolution, such as for reflective objects, developers can sacrifice update interval to prioritize resolution, updating only at 400 ms. However, if the environment is highly dynamic, such as with expected interaction or shadowing, GLEAM can sacrifice resolution to have update interval as low as 30 ms for adaptiveness to lighting changes.

Our 30-participant user study helps us evaluate perceptual effects of GLEAM. The study revealed the efficacy of GLEAM in capturing accurate directionality of environment lighting with 25 out of the 30 indicating that GLEAM captured light more accurately than ARKit. The participants also indicated that GLEAM is able to adapt to change in lighting faster than ARKit.

The main contributions of this paper are:

- A solution for integrating traditional image-based lighting estimation on mobile systems to achieve real-time illumination estimation.
- Situation-driven system tradeoffs to optimize for specific quality factors based on the needs of the virtual scene or target environment.
- A user study to evaluate (a) the effect of illumination estimation methods on human perception, comparing pre-baked lighting, ARKit, and GLEAM; and (b) situational quality preferences for guiding system tradeoff decisions.

In this paper, §2 covers the background and challenges of illumination estimation for mobile AR. §3 gives an overview of our GLEAM design. §4 describes trade-offs between visual quality and update interval. §5 describes our Unity-based implementation. §6 covers our user studies and system evaluations. §7 discusses previous related works. §8 imagines future research.

2 BACKGROUND & CHALLENGES

GLEAM builds on a rich history of illumination estimation from the graphics community. The aim has been to estimate illumination models that graphics renderers can use to illuminate virtual scenes realistically for enhanced visual appeal. Researchers have proposed different strategies to generate illumination models, including measured lighting [3–5] using physical light probes, sun-sky model estimation [18, 33] for outdoor scenes, and inferring perceptually plausible illumination [15, 16] from a single image.

Illumination models are often formulated under the "distant scene assumption": the intensity of incoming ray depends on the direction of incidence only. Thus, modeling illumination boils down to mapping *angular directions* in the 3D space to *light ray intensity*. Under the distant scene assumption, illumination models are usually represented in the form of environment maps, mapping incoming ray direction to ray intensity. One of the most commonly used representations for environment maps is a "cubic environment map" or "cubemap", shown in Figure 2b. Each spatial "texel" on a cubemap face maps to a discrete direction. Thus, mapping directions as the vector between the center of a cubemap and its texels (Figure 2c), a cubemap stores intensities spanning angular directions in 3D space.

2.1 SfM-based illumination estimation

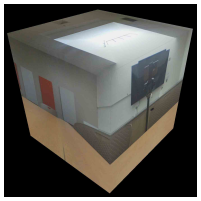
Advances in structure-from motion (SfM) and pose estimation on mobile devices have led to the development of systems that exploit these to estimate lighting for AR. Using 3D structural information about a physical environment, the accurate direction of incoming radiance samples can be associated with the correct radiance intensity. However, using such methods are expensive and require a pre-computation step to perform the SfM prior to generating the illumination model.

Apple's ARKit's illumination estimation uses structural information of the scene to create an environment map based on camera frames continually captured. The generated cubemap includes large missing parts because of the limited field-of-view of a single camera frame. ARKit handles this challenge by using "a machine learning algorithm to approximate the environment texture for parts of the scene it has not seen yet, based on a training model involving thousands of environments[1]."

[1]verbatim from Unity ARKit Plugin Documentation (Section: AREnvironmentProbeAnchor).

(a) A chrome ball placed on a marker serves as a GLEAM light probe.

(b) Cubic environment maps (cubemaps) are one of the most common representations of environment maps.



(c) Each texel on the cubemap represents a discrete direction in 3D space.

Figure 2: Illumination is modeled using physical light probes, such as a chrome ball (left), using which an environment map (middle) is generated that represents the intensity of light coming from discrete direction in 3D space (right).

On an Apple iPad 10.5", we have observed the resulting system to be inaccurate and slow to update, resulting in an average update interval of 3.7 sec. To compute the average update interval, we screen-recorded an application performing illumination estimation using ARKit and measured the interval between consecutive frames when illumination was updated. This delay can be seen in commercially released ARKit applications, such as JigSpace.

2.2 Physical light probes estimation

To perform **broad, accurate, and real-time** illumination estimation, we employ the technique of physical light probe estimation, capturing images of a light probe to reveal environmental lighting information. The light probe is a reflective object, such as a chrome ball (Figure 2a), placed at the location where the lighting needs to be sensed. By associating captured pixels with the angle of incoming light, systems can construct estimations of surrounding lighting.

This method was first explored by Debevec et al [4], who used a high-dynamic range (HDR) image of a reflective sphere captured in the environment to re-light a virtual scene with estimated physical illumination. Using physical light probe estimation can deliver high visual fidelity and richness since we physically measure the radiance at the location where the virtual scene is to be rendered. Thus, we choose this strategy and integrate it with current mobile systems to work towards achieving photorealistic illumination estimation for mobile AR.

2.3 Challenges/opportunities of integrating light probe estimation into mobile AR

The main contribution of this work studies the system integration of different tasks to sense and compose a cubemap from light probe estimation in real time. This approaches the following challenges and opportunities.

Integration with AR frameworks: To the best of our knowledge, we are the first to integrate physical light probe estimation into mobile AR systems in real time. Integration with standard game engines provides opportunities to leverage state-of-the-art tracking and positioning systems for accurate AR placement and also provides tools for geometric processing, e.g., raycasting and mesh collision. Successful integration onto standard game engines allows our illumination estimation to work across a variety of mobile systems, including smartphones, tablets, and headsets.

Flexibility to virtual scene and physical environment: Different virtual scenes and physical environments have different lighting and rendering needs, based on the reflectiveness of the virtual materials and the static or dynamic nature of the surrounding physical environment. The illumination estimation should be sensitive to such needs, and provide developers and/or users the ability to tune illumination estimation to prioritize different quality settings. We pursue this challenge by defining quality metrics and situation-driven tradeoff mechanisms to exchange quality prioritization.

Opportunity to leverage multiple viewpoints: Although curved reflective objects allow narrow camera viewpoints to observe wide environmental areas, a single perspective lacks a complete representation of the lighting from all angles. However, when more users are present, such as in a classroom or museum setting, there is an opportunity to integrate lighting estimation from multiple viewpoints with heightened richness and fidelity. To this end, we pursue challenges related to efficient distributed sampling among multiple devices, complete with strategies to integrate estimations from multiple viewpoints. This allows the benefits of the illumination estimation to scale to improve with additional users.

Computational efficiency: All mobile systems suffer from limited computational resources. Complicating the matter, image processing is notoriously computationally expensive. A chief victim of an overloaded system is a reduction in the frame rate of AR applications, considerably degrading user experience. Thus, so as to preserve user and developer expectation, we design our illumination estimation system with computational efficiency in mind. Furthermore, we implement our estimation system as an auxiliary thread, causing no interruption to the main thread and thereby preserving the high frame rate experiences of standard game engines.

3 GLEAM DESIGN

To provide real-time illumination estimation on AR devices, we propose our software framework GLEAM, which visually observes a reflective light probe object to accurately model scene illumination. Though built on established light probe estimation algorithms, GLEAM solves the systems integration challenges of tracking, sampling, and computational efficiency within a real-time mobile AR environment. To do this, we compose GLEAM with three key modules: (i) radiance sampling, (ii) optional network transfer, and (iii) cubemap composition, as illustrated in Figure 3. In this section, we describe these key modules in detail.

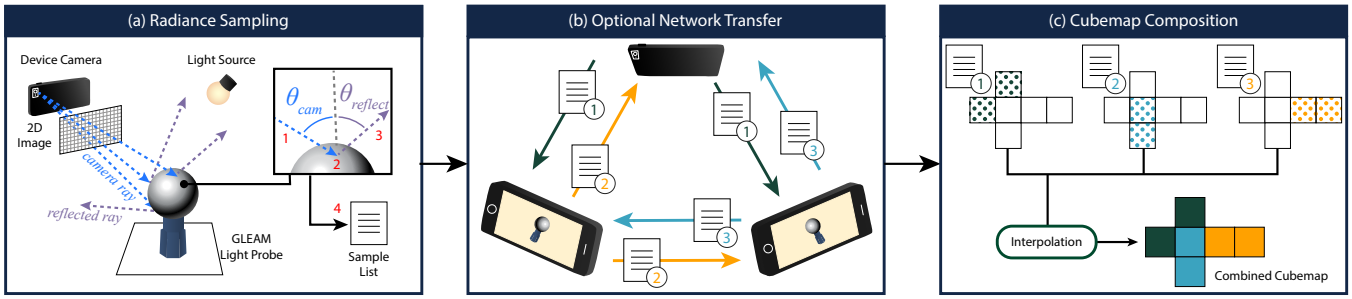


Figure 3: GLEAM illumination estimation system comprises three modules: (a) The radiance sampling module generates a collection of radiance samples by using raycasting to observe reflections off of a reflective light probe; (b) The optional network transfer module shares samples among multiple devices in the system; (c) The cubemap composition module interpolates collected radiance samples to create a combined high quality cubemap.

3.1 Radiance sampling

Cubemaps associate illumination radiance intensities and colors to the angular directions of the incoming light towards the scene. The intensity and color of incoming light associated with its angular direction becomes a "radiance sample." The goal of this module is to generate radiance samples to be used by subsequent modules for composing the cubemaps. Captured images of a reflective light probe with known shape and position can geometrically reveal such radiance information as the object surfaces reflect light into the camera. Thus, to capture radiance samples for an environment map, we spatially position the reflective light probe in the physical scene with respect to an AR positioning marker. Together, the reflective light probe and the AR positioning marker become a GLEAM light probe. Using standard marker-based pose estimation tools to geometrically track the position between the camera and the marker, the GLEAM system can indirectly calculate the position of the virtual camera, the specular object, and the physical scene. This forms the coordinate basis of the radiance sampling.

Specular reflection follows a strict geometric pattern: the angle of the reflected ray from a surface normal $\theta_{reflect}$ matches the angle of the incident ray from the surface normal θ_{cam}. As illustrated in Figure 3a, GLEAM leverages this principle in virtual AR engines to estimate radiance samples using the following process:

(1) Project a virtual ray from each pixel along its camera ray (θ_{cam}) into the virtual scene.
(2) Determine if and where a collision occurs between the ray and a specular surface in the virtual scene.
(3) Reflect the ray over the collision surface normal to generate an incoming ray vector ($\theta_{reflect}$).
(4) Associate the camera pixel color and intensity with the angle of the incoming ray. This association is a radiance sample.

We leverage the geometric raycasting and collision capabilities of the game engine to execute all four of these steps with optimized computational efficiency.

DIY GLEAM light probe. Inspired by Google Cardboard and Nintendo Labo efforts, we similarly advocate for accessible hands-on development from engaged users and developers. Here, we provide instruction for Do-It-Yourself (DIY) construction of light probe. This involves three components: (*i*) a light probe, (*ii*) a makeshift stand, and (*iii*) an AR image marker.

A light probe can be purchased at hardware stores or online retailers, in the form of a 1.5-inch ball bearing. This typically costs $5 USD. To hold the light probe, a flat marker pen cap can be used to stabilize the probe. We use a Staples Remarx Dry-Erase marker cap to serve as our stand. The AR image marker can be printed on normal printer paper, or on cardstock for durability. Sufficient image marker guidelines are on Vuforia, ARKit, and ARCore websites.

The assembly is simple: place the stand in the center of the image marker, and place the light probe on the stand. The calibration for this DIY light probe is similarly straightforward. One must only measure/estimate the distance from the bottom of the stand to the center of the light probe and enter it into the game engine environment, e.g., Unity. Camera-marker correspondence is maintained through the image marker tracking. This automatically provides spatial synchronization among samples from different camera viewpoints. Generalization/standardization for DIY variability could be supported through configurable user interfaces, but the software engineering exercise is beyond the scope of this paper.

3.2 Optional network transfer

GLEAM presents an optional network transfer stage to collect radiance samples from multiple device viewpoints. Radiance samples generated from a single viewpoint will cover partial regions of the cubemap. The remaining regions can be coarsely estimated through interpolation, as we will discuss in the cubemap composition stage. However, in situations where multiple users view the same scene, e.g., classroom or museum scenarios, there is opportunity for radiance samples from multiple viewpoints to contribute to jointly populate the environment map. To leverage this, GLEAM uses a local network to share illumination information among multiple mobile devices, as illustrated in Figure 3b. The effective results of single- and multi-viewpoint GLEAM are shown in Figure 4.

The requirements for networking are simple: upon sample generation, a GLEAM device will transmit sets of samples to all other GLEAM devices that observe the same target. Local multiplayer game engines typically adopt a client-server model, using the server to synchronize information among multiple clients. To remove the need for a dedicated server, the server behavior is often hosted on

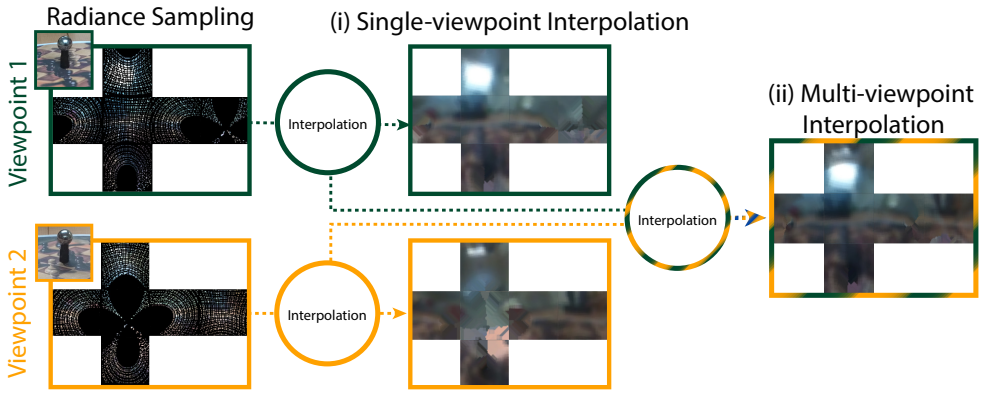


Figure 4: GLEAM computes radiance samples using a reflective object to compose cubemaps.

one of the client applications, which becomes a multiplayer "host". This satisfies the needs for transferring radiance samples among GLEAM devices with negligible latency. A full evaluation of the network requirements for different configurations of GLEAM is left for future work.

3.3 Cubemap composition

The generated and received radiance samples form a sparse estimation of illumination. To create a usable cubemap, GLEAM spatially interpolates the samples into the cubemap space, as illustrated in Figure 3c. While choosing interpolation algorithms, we need to consider not only interpolation quality, but also computational overhead. This is especially important because the cubemap updates on every newly processed list of samples, repeatedly incurring interpolation overhead.

To fill the cubemap, GLEAM uses a modified inverse distance weighting (IDW) interpolation [27]. Our IDW interpolation operates on each cubemap texel, computing a weighted average of nearby samples. We primarily weight each radiance sample by the inverse of its distance from the texel. For low complexity, we use Manhattan Distance as our distance function:

$$d(\mathbf{x}, \mathbf{x_i}) = ||\mathbf{x} - \mathbf{x_i}||_1 = \sum_{j=1}^{n} |\mathbf{x}_j - \mathbf{x_i}_j|$$

We also weight radiance samples on their reliability, depending on where the sample was captured from. Radiance samples collected on the outer rim of the light probe are subject to distortion from projection offset inaccuracies. This is well-documented in light probe estimation techniques [30], and due to the nature of reflective geometry. The angular inaccuracy is directly proportional to the angular deviation between the pixel's camera ray vector θ_{cam} and the reflected incoming ray vector $\theta_{reflect}$. Thus, we use the inverse of the angular deviation as a *reliability* score $r_i = 2\pi/\angle(\theta_{cam}, \theta_{reflect})$, weighting reliable samples stronger for cubemap consideration. Notably, multi-viewpoint GLEAM will allow low reliability samples from one viewpoint to be overridden by higher reliability samples from another viewpoint. The reliability score combines with the distance to form the sample weight:

$$w_i(\mathbf{x}) = \frac{r_i}{d(\mathbf{x}, \mathbf{x_i})}$$

We find the interpolated intensity u of texel x from nearby samples $u_i = u(x_i)$ for $i = 1, 2, ...N$ using the IDW function given by

$$u(\mathbf{x}) = \begin{cases} \frac{\sum_{i=1}^{N} w_i(\mathbf{x}) u_i}{\sum_{i=1}^{N} w_i(\mathbf{x})}, & \text{if } d(\mathbf{x}, \mathbf{x_i}) \neq 0 \ \forall \ i; \\ u_i, & \text{if } d(\mathbf{x}, \mathbf{x_i}) = 0 \end{cases}$$

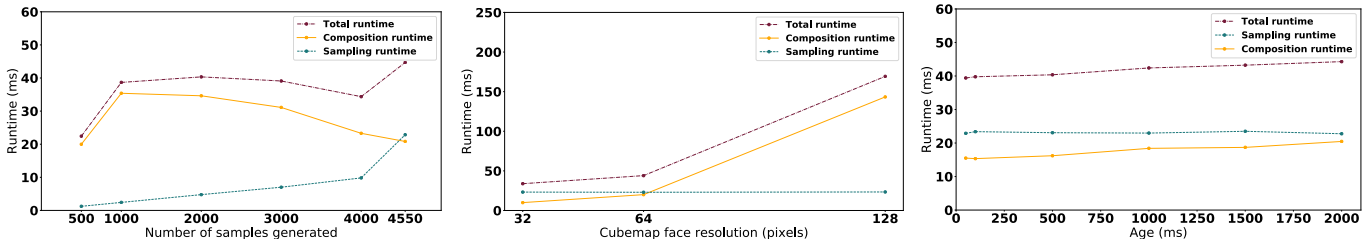
To computationally perform IDW interpolation, we iterate over our list of samples, adding each sample's weighted intensity value and weight to all cubemap texels within a neighborhood radius. We then iterate over the cubemap texels, dividing the sum of weighted pixel values by the sum of distance weights to generate the interpolated texels. IDW will leave cubemap gaps from texels that do not occupy any sample neighborhoods. To fill the remaining gaps, we use a nearest neighbor algorithm to assign missing cubemap texels.

IDW and nearest neighbor are two of many interpolation mechanisms that can satisfy the needs for cubemap interpolation. Other strategies, e.g., structural inpainting or neural network-based methods, are also viable solutions, with potentially higher quality at the expense of computational complexity. To prioritize for reduced computational complexity, we leave the full exploration of such algorithms as a future research avenue. By interpolating radiance samples into a full cubemap on a per-frame basis, the GLEAM system provides a dynamically updating scene illumination.

4 SITUATION-DRIVEN TRADEOFFS

The realism of the virtual scene rendered by AR engines depends on the quality of illumination estimation. Multiple quality factors contribute to a high quality environment map, including coverage, freshness, resolution, and a fast update interval. Under a fixed set of computational resources, these quality factors compete with one another, necessitating tradeoffs to sacrifice some factors for others. However, not all quality factors are needed for all situations. Specifically, depending on the virtual scene materials and the dynamic nature of the physical environment, various quality factors can be promoted over others. We discuss this situation dependence as we define quality factors in §4.0.1. Leveraging this fact, GLEAM can tradeoff quality factors through parameterized policies (§4.1).

4.0.1 Quality factor definitions. ***Coverage*** defines the angular spread of the radiance samples over the cubemap. Covering larger regions of the cubemap allows for accurate representation of lights and



(a) GLEAM update interval is limited by total runtime on increasing number of samples generated. (Cubemap Face res. = 64; Age = 2000 ms)

(b) Computational cost of cubemap composition module increases by an order-of-magnitude with cubemap face resolution. (#samples = ∼ 4500 (max); age = 2000 ms)

(c) There is no significant effect on GLEAM module runtimes with change in age. (#samples generated = ∼ 4500 (max); Cubemap Face res. = 64)

Figure 5: Characterizing runtime performance of GLEAM single-viewpoint prototype.

shadows from more angles. The optional network transfer stage of GLEAM assists with coverage by collecting radiance samples from multiple viewpoints.

Resolution defines the amount of detail the illumination estimation can represent. Higher resolution is beneficial in virtual scenes with smooth reflective materials, in which the surrounding environment is visible. This includes glass materials, polished metals, and liquid surfaces. For non-smooth materials, illumination estimation resolution is less perceptible; in virtual scenes with rough or matte materials, the resolution can be reduced without detriment.

Freshness defines how long ago the illumination estimation information was sampled. Higher freshness allows the estimation to adapt quicker to changes in the environment by discarding older estimation information. Lower freshness accumulates estimation information to build estimations with higher coverage and resolution, but blurs environmental changes over time. Thus, freshness is useful to capture the needs of the dynamically changing physical environments, but can be sacrificed to assist in other quality factors, especially in static physical environments.

Update Interval defines the rate at which the illumination estimation is refreshed. While freshness indicates the age of the *oldest* radiance samples used in an estimation, the update interval indicates the recency of the *newest* radiance samples to be included in an estimation. For dynamically changing environments, a fast update interval will allow the illumination estimation to quickly incorporate changes in the physical environment. However, to allot time to collect radiance samples and compute cubemaps, GLEAM may sacrifice update interval to ensure other quality factors.

4.1 Tradeoffs

Figure 5 shows the trends in runtime of the GLEAM system and its modules based on the 3 parameters that control quality factors. So as not to constrain performance, we perform the characterization on a Windows desktop, hosting a 3.5 GHz Intel Xeon Processor with 128 GB RAM and an Nvidia GeForce GTX 1060 GPU.

4.1.1 Number of radiance samples generated. GLEAM can collect a different number of radiance samples to balance the coverage, resolution, and update interval. Spiraling outward from the center of the image of the light probe, a larger collection of samples will span a broader set of angles to populate the cubemap. If the distance between the probe and device camera is increased, we observe the amount of samples generated decrease. This is because at farther distances, fewer pixels capture the reflective object.

The number of samples captured also has an effect on the runtime performance of the radiance sampling runtime and the cubemap composition runtime as shown in Figure 5a. Together, the total runtime limits the update interval of the estimation. With an increasing number of samples, the sampling workload increases, raising the sampling runtime. However, at and above 1000 samples, raising the number of samples *reduces* the composition runtime. This is because the interpolation workload decreases as angular coverage increases. Altogether, this creates a relatively constant update interval of 34 –45 ms between 1000 and 4000 samples.

With 500 samples, however, both sampling and composition runtimes are low. The composition workload decreases, as samples are interpolated over fewer cubemap faces, using average pixel value to populate the missing faces. This improves the update interval to a lower 22 ms, allowing for rapid adaptation to dynamic environment at the expense of resolution and coverage.

While performing characterization experiments, the system captured up to 4500 samples for the fixed distance and FullHD resolution scenario. The variation in number of samples is due to the radiance sampling algorithm, which checks if all possible samples are extracted for every frame. The algorithm takes additional time near the edges of the reflective object to check the same which also contributes to the non-linear behavior in Figure 5a when sampling at max-capacity.

4.1.2 Cubemap face resolution. The resolution of the cubemap allows a tradeoff between detail capture and runtime performance. As shown in Figure 5b, higher resolutions degrade the composition runtime performance, limiting update interval. For a cubemap resolution of 64 pixels, GLEAM achieves an update interval of 44 ms, which increases to 170 mswhen the resolution is doubled to 128.

The rise in computational cost on increasing the face resolution is due to an increase in the number of texels that need to be filled in the cubemap. Doubling the face resolution increases the number of texels in the cubemap by four times, increasing the composition workload.

However, higher cubemap face resolutions will allow an improvement in the fidelity and richness of the appearance of smooth

materials in virtual scenes. This is contingent on having enough radiance samples to fill the dense cubemap space. For improved resolution, the sacrifice in update interval may be justified for scenes with glass, metals, liquids, and other smooth surfaces.

4.1.3 Age of samples. GLEAM can maintain the freshness of the estimation by discarding samples above a given age threshold. Lower thresholds will allow the estimation to only keep samples that adapt to changing illumination in the physical environment. However, higher thresholds will allow the estimation to accumulate samples to improve the resolution and coverage of the cubemap as the user moves around the scene. Notably, as shown in Figure 5c, the age does not significantly affect the runtime performance of the GLEAM modules, and therefore has little effect on the update interval.

Thus, the threshold parameter for the age of samples creates a tradeoff between freshness, resolution and coverage. As mentioned earlier, freshness is useful in expected dynamic lighting, while other quality factors should be prioritized for static lighting.

4.1.4 Summary. These three tradeoffs allow developers to prioritize (or compromise) qualities of coverage, resolution, freshness, and update interval of the GLEAM estimation. These tradeoffs can work with single viewpoint GLEAM for a single device or multi viewpoint GLEAM with networked devices. Optimal prioritization becomes situation dependent; virtual scenes with smooth surfaces need high resolution, and dynamic lighting benefits from high freshness and low update interval. Developers can make decisions to tune GLEAM to their users' needs.

To further study these tradeoffs, we design five configurations of GLEAM as described in Table 1. Our evaluation studies their effect on visual perception (§6.1.3) and system performance (§6.2).

5 IMPLEMENTATION

We develop software through the Unity 3D game engine [7] and PTC Vuforia [6] to provide the graphics rendering and AR tracking infrastructure for our GLEAM implementation. GLEAM uses Unity's UNet API to transfer samples between devices. Unity supports cross-platform deployment, which allows us to harness the versatile design of GLEAM for Android, iOS, macOS, and Windows deployments. As of this writing, GLEAM has been designed and tested on Nvidia Shield K1 Tablet, Apple iPhone X, Apple iPad 10.5", Samsung Galaxy S8, OnePlus 3T, Apple MacBook Pro, and Windows 10 desktop computers. Additionally, our GLEAM prototype is backward compatible on Android and iOS. Our evaluation uses the DIY light probe.

GLEAM environment mapping. In our implementation, GLEAM uses Vuforia SDK's marker-based pose estimation to track reflective objects. Vuforia, in addition to generating pose estimates, also provides the camera frame that is used to obtain the correspondences between the camera and an image marker. When generating radiance samples, we extract the pixel intensities of the reflective objects from this frame.

GLEAM represents environment maps as Unity's `Cubemap` objects. `Cubemap` objects in Unity are *Texture2D* objects indexed using a `CubemapFace` value and two floating-point values for spatial location on each cubemap face. The intensity of each sample in the `Cubemap` is stored as Unity's `Color32` object, which stores the red, blue, green and alpha channel intensities as a byte value within the range of 0-255. The color and intensity of the corresponding pixel obtained from the camera frame is used as the intensity of the radiance sample generated. To store the direction, samples store a single `CubemapFace` value and two spatial floating-point values, which are directly used as `Cubemap` indices.

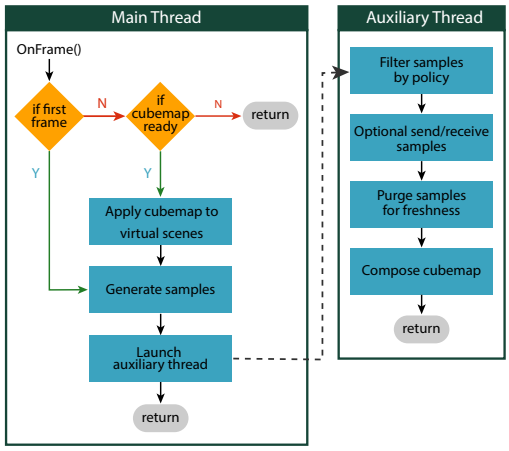


Figure 6: Multi-threaded GLEAM implementation. An auxiliary thread keeps the main thread free for interactive display frame rates.

Cubemap composition. After the transfer of samples is complete, GLEAM uses collected sample lists to compose an environment map. All samples which are to be used in the environment map are composed into a Unity `Cubemap` object. GLEAM uses a Unity material which has a "skybox shader" as the environment map to light the scene. By setting the main texture of the material to the cubemap, GLEAM achieves realistic lighting in every frame.

Multi-threading. To achieve interactive display frame rates and smooth cubemap updates, we employ multi-threading, as shown in Figure 6. Unity's *main thread* includes operations to compute the game state and render frames to the screen. Thus, to preserve fast frame rates, we aim to minimize operations performed on the main thread. Sample generation requires main thread operation to perform game physics raycasting. Applying the cubemap requires the main thread operation to influence rendering operations. All other GLEAM operations, e.g., sample network transfer, environment map composition, are performed on an auxiliary thread so as not to block the main thread during operation. As we later show in our microbenchmarking, this sufficiently allows fast frame rates, limited only by the overhead of Vuforia tracking.

6 EVALUATION

Our evaluation aims to answer the following:

- "How does illumination estimation advance the realism of AR past pre-baked lighting solutions?"
- "How does GLEAM compare against ARKit (the state-of-the-art illumination estimation framework)?"
- "How does variation in GLEAM parameters impact visual perception and system performance?"

(a) Kitchen Scene (b) Living Room Scene (c) User study environment

Figure 7: Using different AR scenes, we evaluate perceptual effects via a controlled environment user study.

Table 1: GLEAM configurations used for evaluation

Policy	#Samples per List	Age (ms)	Cubemap Face res. (pixels)
Single-view SV	4000	200	128
Multi-view MV1	2000	1500	128
Multi-view MV2	500	1500	128
Multi-view MV3	2000	300	128
Multi-view MV4	500	300	128
Multi-view MV5	2000	200	64

To evaluate these questions, we conducted a user study with 30 participants and our GLEAM single- and multi-viewpoint prototypes. We also perform benchmark analysis on the 5 multi-viewpoint GLEAM configurations to monitor the runtime performance of GLEAM modules.

Each multi-viewpoint GLEAM configuration used two devices that contributed to the process of generating samples and creating the environment map. In the user-study (§ 6.1), users were paired with one another for objective evaluation of single and multi-viewpoint configurations. For benchmarking (§ 6.2), the devices were positioned on tripods at different locations and orientations from the physical probe.

Unfortunately, we are unable to generate objective quantitative measures, e.g., SSIM, for image quality comparisons. This is because there is a lack of means to estimate ground truth lighting at the precise estimation location, and comparing rendered images would be subject to material inaccuracy and bias of scene selection.

6.1 Perceptual user study

We designed our user study with different scenes and lighting conditions to test the following hypothesis: *"Illuminating a virtual object with accurate environmental illumination imparts realism to the scene."*

6.1.1 Study design. We used 9 virtual scene illumination methods for our studies. For the first question, we asked the participants to compare 3 virtual lighting environments: (i) no light, (ii) pre-baked directional light, and (iii) GLEAM estimated light. To compare GLEAM against ARKit, we provided users with scenes enabled with (iv) ARKit's illumination estimation module and GLEAM single viewpoint estimation. Finally, we ran the study comparing 5 different configurations of GLEAM multi-viewpoint estimated lighting (v-ix) as described in Table 1 against single-viewpoint estimated lighting.

We used 2 scenes for the study to test user perception across a range of materials. Each scene required 1 GLEAM light probe. We used the "Standard Unity shader" on all models in our scenes. To vary the material properties, we changed the *albedo*, *metallic* and the *smoothness* properties of the shader. Since GLEAM is built for close-range mobile AR applications, such as multi-user AR games and training, AR exhibits, and AR shopping, the scenes were designed to depict similar use-cases. The kitchen scene (Figure 7a) used objects with predominantly specular materials such as reflective utensils *(metallic: 1, smoothness: 0.85)*. The living room scene (Figure 7b) used diffuse materials with low smoothness *(metallic: 0-0.3, smoothness: 0-0.26)*. Real objects similar to the virtual objects in the scenes (utensils, couch, etc.) were provided for objective comparison.

The study was designed to measure user perception in both static and dynamic environmental lighting conditions. There were two types of light sources: overhead DMX lights and incandescent lamps were placed around the scenes. To simulate dynamic lighting, we switched the lights on/off in a fixed loop pattern. Figure 7c shows the user study environment.

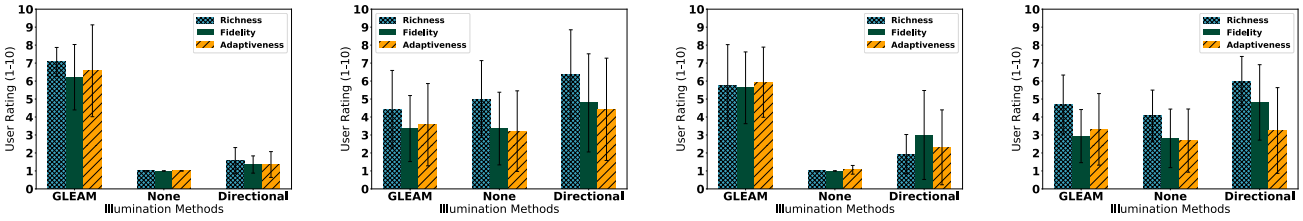
Participants observed the scenes through their handheld mobile devices. The participants were **free and encouraged to move around the scenes**, casting shadows by blocking the light sources intentionally or unintentionally. The participants used an Apple iPhone X (A1901) or an Apple iPad 10.5" (A1709). Both devices were connected via a local WiFi hotspot.

The study took place in 2 parts: First, the participants were asked to report their observations on one of the 8 lighting methods (No lighting, Directional lighting, GLEAM single-view, and 5 GLEAM multi-view) at random. The observations were made as an integer rating from 1-10 on 3 visual perception metrics: richness, fidelity, and adaptiveness as described in Table 2.

Second, participants were asked to compare GLEAM single-viewpoint estimation and ARKit estimation by means of a post-study survey. In the survey, the participants were asked to indicate their preferred method on 2 criteria: i) directionality of source lights, and ii) objects blend with surrounding, after observing an application performing light estimation in the same scene on same device with ARKit and GLEAM one after another. For this comparison, the participants were informed about the method they were observing beforehand.

The participants were encouraged to provide their subjective comments on ARKit and GLEAM estimations post-survey. The study was also recorded on video.

6.1.2 User recruitment and statistics. The study was approved by the institutional IRB of Arizona State University and recruitment was done via email and social media. We recruited 30 participants. 36.7% of the participants reported themselves as working with images and videos in a professional capacity (computer graphics/vision researcher, photographers, digital artist, game developers, etc.), while 56.7% of the participants used smartphones for casual photography. 66.7% reported that they were familiar with AR/VR content, having used either technology more than 3 times. 30% of the participants were females and 80% of the participants were in



(a) Kitchen, Dynamic Lighting (b) Living Room, Dyn. Lighting (c) Kitchen, Static Lighting (d) Living Room, Static Lighting

Figure 8: To evaluate the need for illumination estimation, we compare GLEAM against pre-baked methods: static directional lighting and no lighting.

Table 2: Metrics for user ratings on a scale of 1-10.

Metric	Definition
Richness	The visual quality or appeal of the scene due to illumination. If the scene looked rich and detailed, they got a high rating on richness.
Fidelity	The measure of correctness of illumination. If the light came from the right direction in the scene making the object look accurately illuminated, the rating for fidelity was high.
Adaptiveness	The ability to adapt to illumination changes. An instantaneous response to changing illumination warranted a high adaptiveness rating.

the age group of 18-25 years. None of the participants reported being diagnosed with color blindness.

6.1.3 Results and analysis. At the onset of the study we predicted the following outcomes: (*i*) Estimated lighting carries higher accuracy than pre-baked directional or no lighting. Hence, GLEAM estimated lighting should be rated higher than pre-baked methods on the 3 visual perception metrics to indicate heightened realism; (*ii*) ARKit illumination should receive higher ratings on richness than GLEAM estimates, but the adaptiveness and fidelity of lighting will be better with GLEAM; and (*iii*) For multi-viewpoint GLEAM estimations, configurations with higher coverage should achieve better richness and fidelity rating than adaptiveness rating, while configurations with high freshness should do better on adaptiveness.

Pre-baked vs. estimated lighting. Figure 8 shows the mean rating reported by the participants for the 3 visual perception metrics when the estimation methods were changed between no lighting, pre-baked directional lighting, and GLEAM. We observe a preference towards GLEAM in the kitchen scene across all metrics. For the living room scene, directional lighting was rated slightly higher on richness and fidelity, while we monitor mixed responses on adaptiveness.

These outcomes indicate the dependency of human visual perception on material of objects. The virtual objects in the kitchen scene had a highly specular surface which made it easier for the participants to observe the effect of environmental illumination reflected in the virtual scene. In the living room scene, the diffuse materials did not reflect the nuances of environmental lighting changes on the objects' surface.

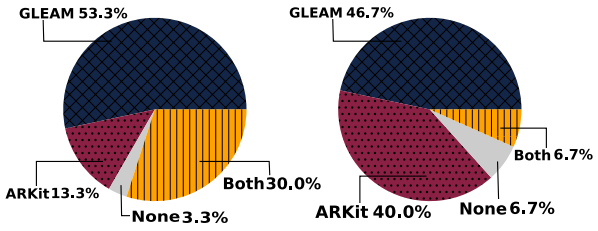
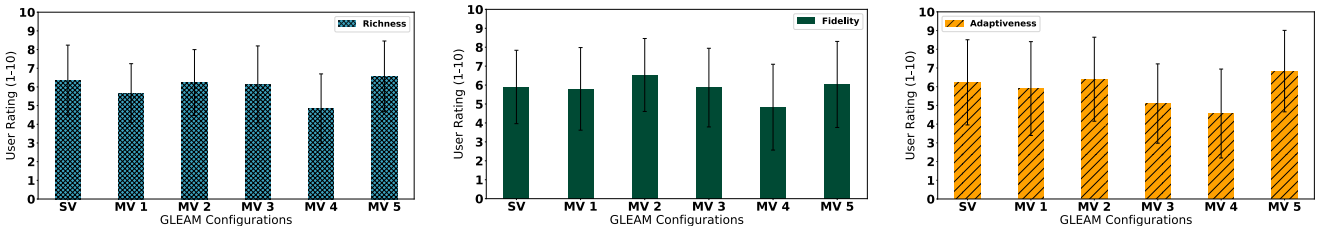


Figure 9: User preference of GLEAM (single-viewpoint) against ARKit for correct directionality of lighting (left) and match with surrounding environment (right).

ARKit vs. GLEAM single-viewpoint estimation. As mentioned in §2, ARKit uses the camera view to estimate its cubemap, filling in missing portions with machine learning. The comparison between ARKit and GLEAM (single-viewpoint) turned out to be highly in favor of GLEAM, as reported in Figure 9.

25 of the 30 participants favored GLEAM's estimation of the directionality of light. Moreover, 16 of them indicated that ARKit failed to detect or incorrectly detected light sources. This result demonstrates the efficacy of GLEAM in providing *accurate* illumination estimation that *adapts* to the environment lighting. Here we attach some subjective comments:

- Participant 6 (GLEAM): *"The lights were changing to different levels, and GLEAM seem to keep up with the lighting changes. If there was any change on ARKit it was very slow."*
- Participant 20 (Both): *"They both seemed to come from the correct direction. However, at times GLEAM seemed to do it faster than the other."*
- Participant 11 (GLEAM): *"GLEAM provided better reflection as it adapted with the lights pretty accurately."*
- Participant 18 (ARKit): *"The reflection was more clear, and there was more of a recognizable environment, but I saw more random blue lights, but also some yellow lights in the pots and pans."*



(a) Mean user rating on richness. **(b) Mean user rating on fidelity.** **(c) Mean user rating on adaptiveness.**

Figure 10: User visual perception of GLEAM single- and multi-viewpoint configurations used in our study.

User perception of GLEAM configurations. We report the mean user ratings for different configurations of GLEAM estimated lighting on the 3 metrics as perceived on the kitchen scene in Figure 10. For this comparison, we disregard ratings from the living room scene because of the participants' lack of ability to distinguish illumination changes.

In general, multi-viewpoint estimation garnered better ratings than single-viewpoint estimation across all metrics, proving the benefit of multi-viewpoint estimation over single-viewpoint estimation.

If the number of samples generated is increased, we observe boosts in richness and fidelity when age is low (MV3 vs. MV4), while these factors are not affected when age is high (MV1 vs. MV2). This is because with high age, the resolution and coverage is already being improved due to accumulation of samples. As predicted in §4.1.1, increasing samples generated (from MV1 to MV2) reduces perceived adaptiveness.

Surprisingly, increasing resolution has no apparent effect on fidelity. But with reduced resolution we see a significant increase in adaptiveness as MV5 is rated better in adaptiveness than all other configurations. We suspect the lack of fidelity improvement to be due to the lack of dense samples to fill the larger cubemap.

For age, with a low number of samples, it becomes clear that higher age provides higher richness and fidelity (MV2 vs. MV4). However, with a high number of samples (MV1 vs MV3), we observe user perception do not follow the trends predicted in §4.1.3 for either coverage, resolution, or fidelity. This may be because with high number of samples, coverage and resolution are already high, while with a higher age of 1500 ms, the system was sufficiently adaptive to the changes in illumination. This was an unexpected (but welcome) outcome.

6.2 Micro-benchmarks

We execute micro-benchmarks on GLEAM single- and multi-viewpoint configurations to evaluate the runtime performance of different modules and the GLEAM system. The data was collected across devices with multiple constrained and unconstrained configurations. Figure 11 reports the runtime performance of the three modules: radiance sampling, network transfer, and cubemap composition along with the total GLEAM runtime.

6.2.1 Computational overhead per module. The computational overhead of the cubemap composition module dominates overall execution time, consuming over 86% of GLEAM's execution time on average across all devices and policies. This is due to the iterative computational expense of interpolation across cubemap texels. Meanwhile, the overhead of the radiance sampling module only consumes up to 6% of GLEAM's execution time on average. Radiance sampling only requires a constant set of geometric operations for each image pixel destined to be a radiance sample.

The execution time of the network transfer module is negligible. The Unity engine handles network transmissions on a separate non-blocking thread, allowing minimal execution overhead. However, while it does not block execution, the transfer itself is not instantaneous and may vary based on network bandwidth.

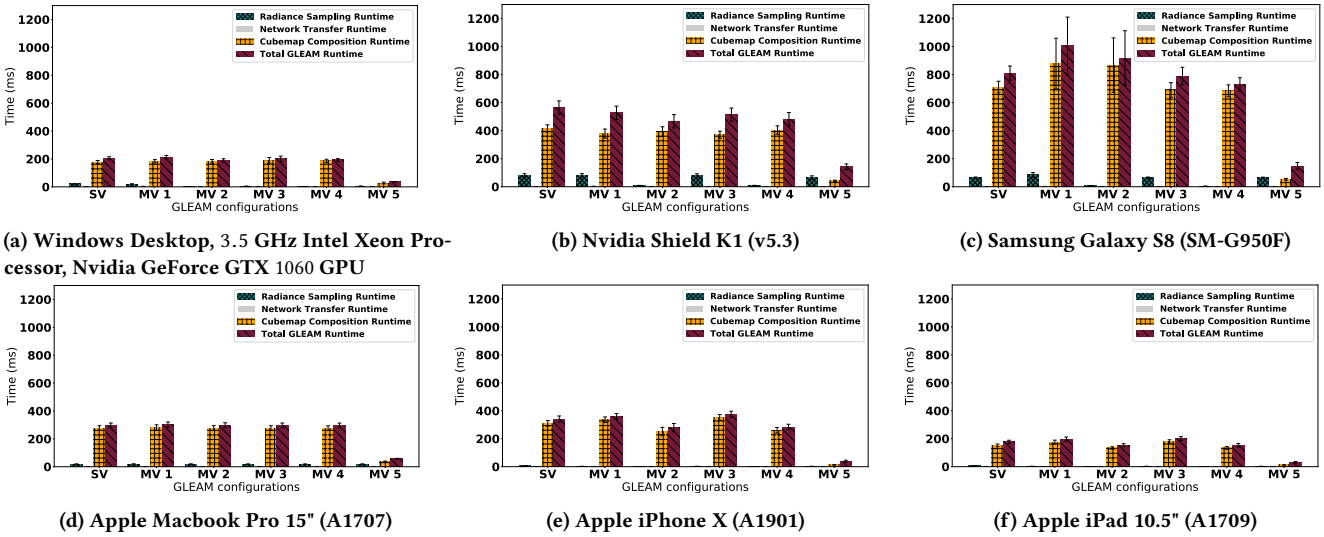
6.2.2 Policy implications on execution time. The execution time of the radiance sampling module and cubemap composition module are both related to the number of samples, as shown in §4.1. Policies that prioritize number of samples increase execution time. To fully minimize update interval, the policies that prioritize low age retain very few samples, further reducing execution time. This noticeably reduces the overhead of cubemap composition, allowing for rapid update cycles.

Post-evaluation take-aways

Our perceptual study comparing GLEAM with pre-baked lighting confirms the need for estimated lighting in AR applications. The dependence of human perception for realistic virtual materials was also revealed as an extended outcome of the experiment. The results support our claim that **GLEAM enables developers to use a wider palette of materials** with realistic appearance through illumination estimation.

In comparison with ARKit, GLEAM was perceived to be more accurate and adaptive to changes. However, ARKit was perceived to be better in richness despite being inaccurate and slow. ARKit fills in the environment map texels with pixels obtained from camera frame, which may result in very incorrect information being captured. This is observed in Figure 12 because the camera is facing downward, sampling a very small portion of the entire space. Clearly, the machine learning algorithms used are not sufficient to fill-in the maps acceptably. **We envision our GLEAM system to work in tandem with ARKit's estimation module** to provide GLEAM's essential boosts in fidelity and adaptiveness while adopting ARKit's sampling techniques for cubemap richness when necessary.

Finally, our perceptual study and benchmarking of different GLEAM configurations revealed the improvement achieved with



(a) Windows Desktop, 3.5 GHz Intel Xeon Processor, Nvidia GeForce GTX 1060 GPU

(b) Nvidia Shield K1 (v5.3)

(c) Samsung Galaxy S8 (SM-G950F)

(d) Apple Macbook Pro 15" (A1707)

(e) Apple iPhone X (A1901)

(f) Apple iPad 10.5" (A1709)

Figure 11: Runtime performance of different modules on GLEAM configurations

GLEAM with multi-viewpoint configurations. One of the prominent outcomes of the study was GLEAM single-viewpoint being outperformed by MV2 and MV5. MV2 performed better than all other configuration generating the same resolution cubemaps which can be attributed to low sampling runtime and hence lower update interval. This result convincingly proves the significance of low update interval due to reduced sampling in achieving smooth user experience. Meanwhile, MV5 – with its order-of-magnitude lower update interval – was rated as the better configuration between all GLEAM configurations as it provided sufficient resolution, coverage, and freshness while maintaining a low update interval.

7 RELATED WORKS

Illumination estimation is a well-studied computer graphics problem. Here, we discuss various facets of illumination estimation.

Measuring and estimating lighting As described in §2, image-based lighting [3–5] uses reflection probes to measure illumination of the scene. Alternative explorations through temporal image based lighting [10, 12, 31] led to HDR or RGBD videos of reflection probes being captured and used for illumination estimation over time. Sparse [1, 26] as well as dense [29, 32] sampling were employed through a combination of custom light probe devices and capture techniques to improve estimation for different situations. Various works use image features to regress an illumination model [11, 17, 21]. Others have exploited perception to approximate illumination [9, 14–16], which is not always accurate to the physical environment. Estimating illumination for outdoor scenes with perceptual or implicit methods can give very wrong results, and was addressed by [18, 19, 33]. We study their method of evaluating perceptual illumination, especially in the design of our user studies. Beyond these works, we solve challenges in integrating lillumination estimation in mobile systems for real-time high quality lighting.

Vision for mixed-reality lighting Some of the recent SfM-based methods use multiple-viewpoints and depth sensors on mobile devices for illumination estimation. Fish-eye cameras can model illumination from multiple viewpoints and offloads computation to a PC server for estimating illumination [25]. Removing the need for fish-eye cameras, commercial RGB-D Kinect sensors can estimate illumination from 3D reconstruction of everyday objects [24]. Towards more dynamic estimation, GLEAM is designed for commercial mobile devices and uses marker-based pose estimation to surpass the need for 3D reconstruction.

8 DISCUSSIONS

Through our GLEAM prototype we provide accurate and real-time illumination estimation for mobile AR. GLEAM is only an early step to photorealistic AR on mobile systems, serving as a framework to open the door for several future opportunities.

Denser sampling: We observe that increasing cubemap face resolution does not by itself lead to richer estimation. Constant capture resolution limits the angular coverage of the samples generated. One way to ensure denser coverage is increasing capture resolution which results in more pixels translating to samples, resulting in more detailed cubemaps. Moreover, generating samples from the camera frame (a la SfM) in addition to the samples generated from reflective geometry can provide a denser sampling at the cost of slower update, while still providing real-time estimation. We need to explore such tradeoffs for situations that demand a high sampling density for better visual quality.

Distributed sensing: To improve runtime efficiency, we envision that GLEAM could use the network transfer module in a distributed framework to reduce computational overhead by eliminating redundant processing. For example, if the viewpoints of two or more devices coincide, only one device needs to perform reflective geometry and share the samples with all other devices. We can further exploit this redundancy for other qualities. For *dynamic range*, devices

Figure 12: ARKit (left) and GLEAM SV (right) illuminated scenes composed of different materials. The environment is shown as reflected on a chrome ball in inset.

can capture radiance at different exposure settings, collectively revealing illumination details at a wider range of intensities. Deeper investigation into GLEAM workload distribution across devices and infrastructure, e.g., edge computing, could reveal interesting opportunities towards distributed estimation and rendering.

9 CONCLUSION

We present GLEAM, a system that estimates environmental lighting to illuminate a virtual scene in real-time with accurate scene illumination. The GLEAM system comprises three major modules: (*i*) radiance sampling, to generate radiance samples using reflective geometry; (*ii*) optional network transfer, to share samples among multiple participating devices; and (*iii*) cubemap composition, to interpolate an environment map from the accumulated samples. We presented trade-offs between update interval and visual fidelity to optimize for quality factors of coverage, resolution, and freshness based on situational need. A supplementary video showing the working system can be found here: https://youtu.be/2gPrfCJyCoc.

ACKNOWLEDGMENTS

We would like to thank NVIDIA for providing us with the NVIDIA Shield-K1 tablets used in this work. We also extend our thanks to Dr. Sha Xin Wei, Connor Rawls, Brandon Mechtley, Alexia Lopez Klein and the rest of the team at The Synthesis Center at ASU for providing us access to the iStage for the user studies. Finally, we thank our shepherd Dr. Inseok Hwang for helping us address reviewers' comments and guidance during the camera-ready process.

REFERENCES

[1] Francesco Banterle, Marco Callieri, Matteo Dellepiane, Massimiliano Corsini, Fabio Pellacini, and Roberto Scopigno. 2013. EnvyDepth: An interface for recovering local natural illumination from environment maps. In *Computer Graphics Forum*, Vol. 32. Wiley Online Library, 411–420.

[2] M Buerli and S Misslinger. 2017. Introducing ARKit-Augmented Reality for iOS. In *Apple Worldwide Developers Conference (WWDC'17)*. 1–187.

[3] Massimiliano Corsini, Marco Callieri, and Paolo Cignoni. 2008. Stereo light probe. In *Computer Graphics Forum*, Vol. 27. Wiley Online Library, 291–300.

[4] Paul Debevec. 1998. Rendering Synthetic Objects into Real Scenes: Bridging Traditional and Image-based Graphics with Global Illumination and High Dynamic Range Photography. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '98)*. ACM, New York, NY, USA, 10. http://doi.acm.org/10.1145/280814.280864

[5] Paul Debevec, Paul Graham, Jay Busch, and Mark Bolas. 2012. A single-shot light probe. In *ACM SIGGRAPH 2012 Talks*. ACM, 10.

[6] Vuforia Developer. [n. d.]. SDK, Unity extension Vuforia–7.1 (2018).

[7] Unity Game Engine. 2018. Unity game engine-official site. *Online[Cited: August 6, 2018.] http://unity3d.com* (2018), 1534–4320.

[8] Y. Feng. 2008. Estimation of Light Source Environment for Illumination Consistency of Augmented Reality. In *2008 Congress on Image and Signal Processing*, Vol. 3. 771–775. https://doi.org/10.1109/CISP.2008.87

[9] Marc-André Gardner, Kalyan Sunkavalli, Ersin Yumer, Xiaohui Shen, Emiliano Gambaretto, Christian Gagné, and Jean-François Lalonde. 2017. Learning to Predict Indoor Illumination from a Single Image. *ACM Trans. Graph.* 36, 6, Article 176 (Nov. 2017), 14 pages. https://doi.org/10.1145/3130800.3130891

[10] Thorsten Grosch, Tobias Eble, and Stefan Mueller. 2007. Consistent interactive augmentation of live camera images with correct near-field illumination. In *Proceedings of the 2007 ACM symposium on Virtual reality software and technology*. ACM, 125–132.

[11] L. Gruber, T. Langlotz, P. Sen, T. Höherer, and D. Schmalstieg. 2014. Efficient and robust radiance transfer for probeless photorealistic augmented reality. In *2014 IEEE Virtual Reality (VR)*. 15–20. https://doi.org/10.1109/VR.2014.6802044

[12] Vlastimil Havran, Miloslaw Smyk, Grzegorz Krawczyk, Karol Myszkowski, and Hans-Peter Seidel. 2005. Interactive System for Dynamic Scene Lighting using Captured Video Environment Maps.. In *Rendering Techniques*. 31–42.

[13] Google Inc. 2018. ARCore API. *Online[Cited: August 6, 2018.]: https://developers.google.com/ar/discover/* (2018).

[14] Kevin Karsch, Varsha Hedau, David Forsyth, and Derek Hoiem. 2011. Rendering Synthetic Objects into Legacy Photographs. In *Proceedings of the 2011 SIGGRAPH Asia Conference (SA '11)*. ACM, New York, NY, USA, Article 157, 12 pages. https://doi.org/10.1145/2024156.2024191

[15] Kevin Karsch, Kalyan Sunkavalli, Sunil Hadap, Nathan Carr, Hailin Jin, Rafael Fonte, Michael Sittig, and David Forsyth. 2014. Automatic Scene Inference for 3D Object Compositing. *ACM Trans. Graph.* 33, 3, Article 32 (June 2014), 15 pages. https://doi.org/10.1145/2602146

[16] Erum Arif Khan, Erik Reinhard, Roland W. Fleming, and Heinrich H. Bülthoff. 2006. Image-based Material Editing. In *ACM SIGGRAPH 2006 Papers (SIGGRAPH '06)*. ACM, New York, NY, USA, 654–663. https://doi.org/10.1145/1179352.1141937

[17] Natasha Kholgade, Tomas Simon, Alexei Efros, and Yaser Sheikh. 2014. 3D Object Manipulation in a Single Photograph Using Stock 3D Models. *ACM Trans. Graph.* 33, 4, Article 127 (July 2014), 12 pages. https://doi.org/10.1145/2601097.2601209

[18] J. Lalonde, A. A. Efros, and S. G. Narasimhan. 2009. Estimating natural illumination from a single outdoor image. In *2009 IEEE 12th International Conference on Computer Vision*. 183–190. https://doi.org/10.1109/ICCV.2009.5459163

[19] Jean-François Lalonde, Alexei A. Efros, and Srinivasa G. Narasimhan. 2009. Webcam Clip Art: Appearance and Illuminant Transfer from Time-lapse Sequences. In *ACM SIGGRAPH Asia 2009 Papers (SIGGRAPH Asia '09)*. ACM, New York, NY, USA, Article 131, 10 pages. https://doi.org/10.1145/1661412.1618477

[20] D. Mandl, K. M. Yi, P. Mohr, P. M. Roth, P. Fua, V. Lepetit, D. Schmalstieg, and D. Kalkofen. 2017. Learning Lightprobes for Mixed Reality Illumination. In *2017 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 82–89. https://doi.org/10.1109/ISMAR.2017.25

[21] Ko Nishino and Shree K. Nayar. 2004. Eyes for Relighting. In *ACM SIGGRAPH 2004 Papers (SIGGRAPH '04)*. ACM, New York, NY, USA, 704–711. https://doi.org/10.1145/1186562.1015783

[22] D. Nowrouzezahrai, S. Geiger, K. Mitchell, R. Sumner, W. Jarosz, and M. Gross. 2011. Light factorization for mixed-frequency shadows in augmented reality. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality*. 173–179. https://doi.org/10.1109/ISMAR.2011.6092384

[23] Ravi Ramamoorthi and Pat Hanrahan. 2001. An efficient representation for irradiance environment maps. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. ACM, 497–500.

[24] T. Richter-Trummer, D. Kalkofen, J. Park, and D. Schmalstieg. 2016. Instant Mixed Reality Lighting from Casual Scanning. In *2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 27–36. https://doi.org/10.1109/ISMAR.2016.18

[25] K. Rohmer, W. Büschel, R. Dachselt, and T. Grosch. 2014. Interactive near-field illumination for photorealistic augmented reality on mobile devices. In *2014 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 29–38. https://doi.org/10.1109/ISMAR.2014.6948406

[26] Imari Sato, Yoichi Sato, and Katsushi Ikeuchi. 1999. Acquiring a radiance distribution to superimpose virtual objects onto a real scene. *IEEE transactions on visualization and computer graphics* 5, 1 (1999), 1–12.

[27] Donald Shepard. 1968. A Two-dimensional Interpolation Function for Irregularly-spaced Data. In *Proceedings of the 1968 23rd ACM National Conference (ACM '68)*. ACM, New York, NY, USA, 517–524. https://doi.org/10.1145/800186.810616

[28] Peter-Pike Sloan, Jan Kautz, and John Snyder. 2002. Precomputed Radiance Transfer for Real-time Rendering in Dynamic, Low-frequency Lighting Environments. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '02)*. ACM, New York, NY, USA, 527–536. https://doi.org/10.1145/566570.566612

[29] Jonas Unger, Stefan Gustavson, Per Larsson, and Anders Ynnerman. 2008. Free form incident light fields. In *Computer Graphics Forum*, Vol. 27. Wiley Online Library, 1293–1301.

[30] Jonas Unger, Stefan Gustavson, and Anders Ynnerman. 2007. Spatially varying image based lighting by light probe sequences. *The Visual Computer* 23, 7 (2007), 453–465.

[31] Jonas Unger, Joel Kronander, Per Larsson, Stefan Gustavson, and Anders Ynnerman. 2013. Temporally and spatially varying image based lighting using hdr-video. In *Signal Processing Conference (EUSIPCO), 2013 Proceedings of the 21st European*. IEEE, 1–5.

[32] Jonas Unger, Andreas Wenger, Tim Hawkins, Andrew Gardner, and Paul Debevec. 2003. *Capturing and rendering with incident light fields*. Technical Report. University of Southern California Marina Del Rey CA Institute for Creative Technologies.

[33] Guanyu Xing, Xuehong Zhou, Qunsheng Peng, Yanli Liu, and Xueying Qin. 2013. Lighting simulation of augmented outdoor scene based on a legacy photograph. In *Computer Graphics Forum*, Vol. 32. Wiley Online Library, 101–110.

[34] Edward Zhang, Michael F. Cohen, and Brian Curless. 2016. Emptying, Refurnishing, and Relighting Indoor Spaces. *ACM Trans. Graph.* 35, 6, Article 174 (Nov. 2016), 14 pages. https://doi.org/10.1145/2980179.2982432