PointAR: Efficient Lighting Estimation for Mobile Augmented Reality

Yiqin Zhao and Tian Guo

Worcester Polytechnic Institute, Worcester MA
{yzhao11, tian}@wpi.edu

Abstract. We propose an efficient lighting estimation pipeline that is suitable to run on modern mobile devices, with comparable resource complexities to state-of-the-art on-device deep learning models. Our pipeline, referred to as PointAR, takes a single RGB-D image captured from the mobile camera and a 2D location in that image, and estimates a 2nd order spherical harmonics coefficients which can be directly utilized by rendering engines for indoor lighting in the context of augmented reality. Our **key insight** is to formulate the lighting estimation as a learning problem directly from point clouds, which is in part inspired by the Monte Carlo integration leveraged by real-time spherical harmonics lighting. While existing approaches estimate lighting information with complex deep learning pipelines, our method focuses on reducing the computational complexity. Through both quantitative and qualitative experiments, we demonstrate that PointAR achieves lower lighting estimation errors compared to state-of-the-art methods. Further, our method requires an order of magnitude lower resource, comparable to that of mobile-specific DNNs.

Keywords: Lighting estimation, deep learning, mobile AR

1 Introduction

In this paper, we describe the problem of lighting estimation in the context of mobile augmented reality (AR) applications for indoor scene. We focus on recovering scene lighting with the image-based lighting model [10] which is widely used in modern computer graphics rendering. Accurate lighting estimation positively impacts realistic rendering, making it an important task in real-world mobile AR scenarios, e.g., furniture shopping apps that allow user to place a chosen piece in a physical environment.

At a high level, under the image-based lighting model, to *obtain* the lighting information at a given position in the physical environment, one would use a 360° panoramic camera such as ones from Matterport3D [7] that can capture incoming lighting from every direction. However, commodity mobile phones often lack access to such panoramic camera, making it challenging to directly obtain accurate lighting information and necessitating the task of lighting estimation. There are three key challenges when estimating lighting information for mobile

AR applications. **First**, the AR application needs to estimate the lighting at the rendering location, i.e., where the 3D object will be placed, based on the camera view captured by the mobile device. **Second**, as the mobile camera often only has a limited field of view (FoV), i.e., less than 360 degree, the AR application needs to derive or estimate the lighting information outside the FoV. **Lastly**, as lighting information is used for rendering, the estimation should be fast enough and ideally to match the frame rate of 3D object rendering.

Recently proposed lighting estimation approaches [12,13,23] are all learning-based but did not consider the aforementioned unique challenges of supporting AR on mobile phones. Gardner et al. proposed a simple transformation to tackle the spatial difference between observation and rendering positions [12]. However, the proposed transformation did not using the depth information and therefore can lead to image distortion. Garon et al. improved the spatial lighting estimations with a two-branches neural network that was reported to perform well on a laptop GPU but not on mobile devices [13]. Song et al. further improved the estimation accuracy by decomposing the pipeline into differentiable sub-tasks [23]. However, the overall network was large in both size and computational complexity which makes it ill-suited for running on mobile phones.

Our **key insight** is to break down the lighting estimation problem into two sub-problems: *(i)* geometry-aware view transformation and *(ii)* point-cloud based learning from limited scene. At a high level, geometry-aware view transformation handles the task of applying spatial transformation to an existing view, e.g., camera photo, with a mathematical model. In other words, we skip the use of neural networks for considering scene geometric and other parameters, unlike previous methods that approached the lighting estimation with a monolithic network [12,13]. This is crucial as it keeps the task simpler which in turns makes the models more efficient to run on mobile devices. Our **key idea** for learning lighting estimation spherical harmonics coefficients directly from point cloud, instead of image, is in part inspired by the use of Monte Carlo Integration in calculating the real-time spherical harmonics.

Concretely, We propose a *hybrid lighting estimation* for mobile AR with the promise of realistic rendering effects and fast estimation speed. We rethink and redefine the lighting estimation pipeline by leveraging an efficient mathematical model to tackle the *view transformation* and a compact deep learning model for point cloud-based lighting estimation.

Our method, referred to as PointAR, takes the input of a RGB-D image and a 2D pixel coordination (i.e., observation position) and outputs the 2nd order spherical harmonics coefficients (i.e., a compact lighting representation of diffuse irradiance map) of a world position for rendering virtual objects. We use irradiance map, a 360° panorama that represents the incoming irradiance at every direction of a 3D position, as it sufficiently describes the diffuse light of our interest. In summary, PointAR circumvents the hardware limitation (i.e., 360 degree cameras) and enables fast lighting estimation on commodity mobile phones.

Fig. 1. Lighting estimation workflow in AR applications. Lighting estimation starts from a camera view captured by a user's mobile phone camera. The captured photo, together with a screen coordinate (e.g., provided by the user through touch-screen), is then passed to a lighting estimation algorithm. The estimated lighting information is then used by a 3D rendering engine such as Unity3D for rendering virtual objects and combining with the original camera view into a frame. The final rendered image is displayed for the user.

We evaluated our method by training on a point cloud dataset generated from large-scale real-world datasets called MatterPort3D and the one from Neural Illumination [7,23]. Compared to recently proposed lighting estimation approaches, our method PointAR achieved up to 31.3% better irradiance map $l2$ loss with one order of magnitude smaller and faster model. Further, PointAR produces comparable rendering effects to ground truth and has the promise to run efficiently on commodity mobile devices.

2 Mobile Lighting Estimation and its Challenges

In this section, we describe how lighting estimation, i.e., recovering scene lighting based on limited scene information, fits into the mobile augmented workflow from an end-user's perspective. The description here focuses on how lighting estimation module will be used while a human user is interacting with an mobile AR application such as a furniture shopping app. Understanding the role played by lighting estimation module can underscore the challenges and inform the design principles of mobile lighting estimation.

Figure 1 shows how a mobile user with a multi-cameras mobile phone, such as iPhone 11 or Samsung S10, interacts with the mobile AR application. Such mobile devices are increasingly popular and can capture image in RGB-D format, i.e., with depth information. The user can tap the mobile screen to place the virtual object, such as a couch, on the detected surface. To achieve a realistic rendering of the virtual object, i.e., seamlessly blending to the physical environment, the mobile device leverages lighting estimation methods such as those provided by AR software development kit [1]. The estimated lighting information will then be used by the rendering engine to relit the virtual object.

In this work, we target at estimating indoor lighting which can change spatially, e.g., due to user movement, and temporally, e.g., due to additional light sources. To provide good end-user experience, the mobile AR application often needs to re-estimate lighting information in a rate that matches desired fresh rate measured in frame per second (fps). This calls for fast lighting estimation method that can finish execute in less than 33ms (assuming 30fps).

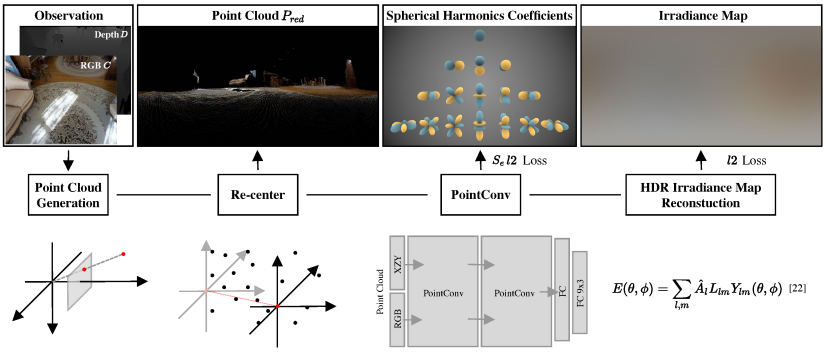

Fig. 2. PointAR pipeline composition and components. We first transform the camera view into a point cloud and then apply transformations described in Figure 4. Then we use a compact neural network described in Section 4.2 to estimate spherical harmonics coefficients at the rendering position. We use $l2$ distance loss on both estimated spherical harmonics coefficients and HDR irradiance map reconstructed from spherical harmonics coefficients for evaluation.

However, lighting estimation for mobile AR comes with three inherent challenges that might require sophisticated deep learning approaches [12,13,11,23]. First, obtaining accurate lighting information for mobile devices is challenging as it requires access to the 360° panorama of the rendering position; mobile devices at best can obtain the lighting information at the device location, also referred to as observation position, through the use of ambient light sensor or both front- and rear-facing cameras to expand limited field-of-view (FoV). [1] Second, as indoor lighting often varies spatially, leveraging observation location lighting information to lit a virtual object in the rendering location can lead to undesirable visual effect. Third, battery-powered commodity mobile devices, targeted by our work, have rather limited hardware supports when comparing to specialized devices such as HoloLens. This further complicates the network designs and emphasizes the importance of resource efficiency.

3 Problem Formulation

We formulate the **lighting estimation in mobile augmented reality** as a spherical harmonics coefficients (SHC) regression problem as $h : h(g(f(C, D, I), r)) = S_r$ where $g : g(P_o, r) = P_r$ and f: $f(C, D, I) = P$. Specifically, we decompose the problem into two stage to achieve the goal of fast lighting estimation on commodity mobile phones.

For the first stage, we formulate the geometric-aware transformation as a linear translation problem $g(f)$. $f(C, D, I)$ takes the inputs of a RGB-D image,

[1] It is possible to achieve accurate lighting information by requiring user cooperation such as spinning in circle or walking to rendering location and back while holding the device. However, this often hinders the usability.

Fig. 3. Comparison of different warping methods. To transform mobile camera view spatially, i.e., from observation to rendering location, we leverage the point cloud which was generated from the RGB-D image. Our approach does not subject to distortion problem and therefore is more accurate than spherical warping that only considers RGB image [13].

represented as (C, D), and the mobile camera intrinsic I and transforms (C, D) at observation position o (i.e., mobile phone position) into a point cloud representation P_o. $g : g(P_o, r)$ takes another input of the rendering position r and leverages a linear translation T to generate a point cloud P_r centered at the rendering position r. In essence, this transformation simulates the process of re-centering the camera from user's current position o to the rendering position r. We describe the geometric-aware transformation design in Section 4.1. For the second stage, we formulate the lighting estimation as a point cloud based learning problem h that takes a partial point cloud P_r and outputs 2nd order spherical harmonics coefficients S_r. We describe the end-to-end point cloud learning in Sections 4.2 to Section 4.4.

4 PointAR Pipeline Architecture

In this section, we describe our hybrid lighting estimation pipeline that is used to model h. As shown in Figure 2, we decompose the lighting estimation into two sequential modules: *(i)* a point cloud generation module for handling geometry transformation (Section 4.1); *(ii)* and a deep learning model for estimating lighting information, represented as spherical harmonics coefficients (Section 4.2). Compared to traditional end-to-end neural network designs, our PointAR is more resource efficient (as illustrated in Section 5) and exhibits better interpretability. We also describe our dataset generation in Section 4.3 and our design rationale in Section 4.4.

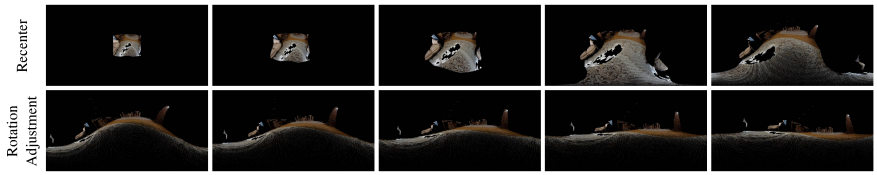

Fig. 4. Example video frames showing point cloud transforming process. Row 1 shows the recenter process that simulates a linear camera movement in 3D space. **Row 2** shows the rotation adjustment on the recentered point cloud.

4.1 Point Cloud Generation

The transformation module takes a RGB-D image, represented as (C, D), and the rendering location r and outputs a point cloud of the rendering location. Our **key insights** for using point cloud data format are two-folds: *(i)* point cloud can effectively represent the environment geometry and support view transformation; *(ii)* point cloud resembles the Monte Carlo Integration optimization used for real-time spherical harmonics calculation.

Compared to prior work that addressed the challenge of spatially-variant lighting with complex neural network architecture [12,13,23], our proposed approach has the benefits of better interpretability and robustness in handling complex indoor geometries. Figure 3 compares the warped results between traditional sphere warping and our point cloud based approach. Our approach achieved better warping effect by circumventing the distortion problem associated with the use of RGB images. More importantly, our point cloud transformation is faster as it only requires a simple linear matrix operation, and has the promise to achieve fast lighting estimation for heterogeneous mobile hardware, during inference.

Figure 4 shows example video frames when transforming a RGB-D image at observation location to a point cloud at rendering location. In Section 4.4, we detail the process of generating point cloud and its corresponding spherical harmonics coefficients from a large-scale real-world indoor dataset.

4.2 Point Cloud to SHC Estimation

Our second component takes the point cloud P_r at rendering position r and estimates the spherical harmonics coefficients S_r which is a compact representation of lighting information at location r. Our choice of learning spherical harmonics coefficients S_r *directly* instead of other representations such as image style irradiance map is largely driven by our design goal, i.e,. efficient rendering in commodity mobile phones. Formulating the illumination as a pixel-wise regression problem [23] often require complex neural network designs. As mobile augmented reality applications often have a tight time budget, e.g., 33 ms for 30fps UI update, it can be challenging to support the use of such neural networks directly on mobile devices. Additionally, popular rendering engines such as Unreal Engine and Unity3D support rendering virtual objects directly with spherical harmonics coefficients.

Fig. 5. Irradiance map comparison between ground truth and predicted ones. Row 1 shows the observation captured by camera. **Row 2** environment map from dataset. **Row 3** shows the irradiance map generated from the environment map using spherical harmonics convolution. **Row 4** shows the reconstructed irradiance map from spherical harmonics coefficients predicted by PointAR.

To train $h : h(P_r, r) = S_r$, we chose the PointConv [26] model that is an efficient implementation for building deep convolutional networks directly on 3D point clouds. It uses multi-layer perceptrons to approximate convolutional filters by training on local point coordinates.

This component is trained with supervision from a spherical harmonics coefficients $l2$ loss L_S as defined in Equation (1), similar to Garon et al. [13].

$$L_S = \frac{1}{9} \sum_{c=1}^{3} \sum_{l=0}^{2} \sum_{m=-l}^{l} (i_{l,c}^{m*} - i_{l,c}^{m}), \tag{1}$$

where c is the color channel (RGB), l and m are the degree and order of spherical harmonics coefficients. We chose to target 2nd order spherical harmonics coefficients as it was demonstrated by Ramamoorthi et al. [22] to be sufficient for diffuse irradiance learning which aligns with our focus application scenarios.

4.3 Dataset Generation of Point Clouds and SHC

Next, we describe how we generate a large amount of real-world training data including transformed point clouds and spherical harmonics coefficients by leveraging two existing datasets, i.e., Matterport3D [7] and Neural Illumination [23] datasets. Briefly, Matterport3D is a large-scale dataset that contains RGB-D images and panoramic views for indoor scenes. Each RGB-D panorama contains aligned color and *depth* images (of size 1028x1024) for 18 viewpoints. The Neural Illumination dataset was derived from Matterport3D and consists of additional

information that associate images, 360° panoramas, and the relationship between images at observation and rendering locations.

The first step of our dataset creation process is to transform the observation RGB-D images, i.e., captured at user locations, in Matterport3D dataset into point cloud format. To transform the 2D image plane to 3D representation, we leveraged the the pinhole camera model [9] and camera intrinsics of each photo in the dataset. For each RGB-D photo, we first pre-processed the depth image with distance transform to remove a small portion of pixels that are missing depth data during capturing. Then we calculated the 3D point cloud coordinates (x, y, z) as:

$$x = \frac{(u - cx) * z}{fx}, \tag{2}$$

$$y = \frac{(v - cy) * z}{fy}, \tag{3}$$

Where z is the depth value in RGB-D photo, u and v are the photo pixel coordinates, fx and fy are the vertical and horizontal camera focal length, cx and cy are the photo optical center.

With the above transformation, we generated the point cloud P_o for each observation position. Then, we applied a linear translation T to P_o to transform the view at observation position to the rendering position. This is an important step as it will allow our model to learn and estimate spatially-variant lighting.

Specifically, T is determined by using the pixel coordinates of each rendering position on observation image from the Neural Illumination dataset in order to calculate a vector to the locale point. We scaled the distance with a factor of 0.95 to represent the rendering position for virtual objects. We found and demonstrated in Section 5 that this approach achieved good spatial transformation, even though it only represents an estimate for the rendering location.

Then, we applied rotation U on the recentered point cloud P_O for: *(i)* aligning with ground truth environment map in dataset for training purpose; and *(ii)* rotating point cloud according to geometry surface and camera orientation during inference time.

Finally, for each panorama at rendering position, we extracted 2nd order spherical harmonics coefficients as 27 float numbers to represent the irradiance ground truth. On generated point clouds, we also performed projection and consequently generated respective 2D panorama images which will be used for conducting variants evaluation.

4.4 Design and Training Discussions

Learning From Point Cloud Our choice to learn lighting information *directly* from point cloud is driven by mobile AR use cases and their performance requirement of fast inference.

After obtaining the transformed point cloud, one intuitive approach is to formulate the learning process as an image learning task by projecting the

transformed point cloud into a panorama. This is because image-based lighting graphics models [10] commonly use 360° panoramic view to calculate lighting from every direction of the rendering position. However, learning from projected point cloud can be challenging due to potential distortion caused by projecting 360° data to a 2D image plane and missing pixel values due to point cloud being stored sparsely. We compare this formulation to PointAR in Section 5.

Our idea to learn diffuse lighting from the point cloud representation is in part inspired by how real-time diffuse lighting calculation has been optimized in modern 3D rendering engines. Ramamoorthi et. al [22] proposed the use of spherical harmonics convolution to speedup the irradiance calculation. Compared to diffuse convolution, spherical harmonics convolution is approximately $O(\frac{T}{9})$ times faster where T is the number of texels in the irradiance environment map [22]. However, as spherical harmonics convolution still includes integral operation, performing it directly on large environment maps might hurt real-time performance. Consequently, Monte Carlo Integration was proposed as an optimization to speed up lighting calculation through spherical harmonics convolution by uniformly sampling pixels from the environment map.

In short, Monte Carlo Integration demonstrates the feasibility to calculate the incoming irradiance with enough uniformly sampled points of the environment map. In our problem setting of mobile AR, we have limited samples of the environment map which makes it nature to formulate as a data-driven learning problem with a neural network.

Although learning from point cloud representation can be a difficult task due to the sparsity of point cloud data [20], we adopted a recently proposed PointConv [26] architecture as the main component of our neural network design. We chose PointConv due to its good learning performance and efficient implementation; other point cloud learning approaches [21,28,18] might also be used.

Finally, we choice of estimating diffuse environment map instead of more detailed environment map is again driven by our application scenarios. First, it is challenging to construct detailed environment map from a limited scene view captured by the mobile phone camera. This in turns can lead to more complex neural networks that might not be suitable to run on mobile devices. Second, neural network generated environment maps may be subject to distortion and unexpected shape. This might lead to reflective textures during rendering and can significantly affect the AR end-user experience.

Training Dataset Training a neural network that can accurately estimate lighting requires a large amount of real-world indoor 3D scenes that represents the complicated indoor geometries and lighting variance. Furthermore, in the context of mobile AR, each training data item need to be organized as a tuple of *(i)* a photo (C, D) captured by mobile camera at the observation position to represent the user's observation; *(ii)* a 360° panorama photo E at the rendering position for extracting lighting information ground truth; *(iii)* a relation R between (C, D) and E to map the pixel coordinates at the observation phone to

the ones at the rendering position, as well as the distance between these pixel coordinates.

Existing datasets such as Matterport3D [7], Stanford 2D-3D-S [6] and Neural Illumination [23] all fall short to satisfy our learning requirement. For example, Matterport3D provides a large amount of real-world panorama images which can be used to extract lighting information to serve as ground truth. However, this dataset does not include either observation nor relation data. The dataset derived by Neural Illumination has the format that is closest to our requirement but is still missing information such as point clouds.

In this work, we leveraged both Matterport3D and the Neural Illumination datasets to generate a dataset that consists of point cloud data P and spherical harmonics coefficients S. Each data item is a five-item tuple with $((C, D), E, R, P, S)$. However, the point clouds generated from observation images are very large, e.g., 1310720 points per observation image, which complicates model training with large amount of GPU memory requirement.

In our current implementation, we down-sampled each point cloud with 1280 points to reduce resource consumption during training and inference. Our uniform down-sampling method is consistent with the one used in the PointConv paper [26]. Similar to what was demonstrated by Wu et al. [26], we also observe that reducing the point cloud size, i.e., the number of points, do not necessarily lead to worse classification results but can reduce GPU memory consumption linearly during training.

5 Evaluation

We trained and tested our PointAR neural network on our generated dataset (Section 4.3) by following the train/test split method described in previous work [23]. Our evaluations include end-to-end comparisons to recent works on lighting estimation, ablation studies of our pipeline design, and resource complexities compared to commonly used mobile DNNs.

Result Highlights We briefly summarize our evaluation results.

- Our proposed hybrid pipeline achieves better, i.e., lower, spherical harmonics coefficients $l2$ loss and irradiance map $l2$ loss, than recently proposed works by Song et al. [23] and Garon et al. [13] as shown in Table 1.
- When comparing to ground truth, the 3D objects rendered by our approach have good rendering results as shown in Figure 6.
- Directly training from point cloud instead of from projected point cloud achieved better SHC $l2$ and irradiance map $l2$ losses. Further, reducing the number of sampled points slightly impacted the losses, e.g., by up to 5% for spherical harmonics coefficients $l2$ loss, but proportionally reduced multiply accumulates (MACs).
- Our proposed PointAR has comparable model size and MACs when compared to mobile-specific DNNs, indicating its ability to efficiently run on commodity mobile devices.

Fig. 6. Rendering example from PointAR. Comparison between rendering a virtual object with ground truth irradiance map and irradiance map generated by PointAR. **Row 1 and 2** show the comparison between a closeup look of the rendering objects. **Row 3 and 4** show the comparison between rendered objects in original observation. Note we did not cast shadow to highlight the lighting rendering results and to avoid misleading visual effects.

Evaluation metrics. We use the following two metrics to quantitatively evaluate the accuracy on our predicted spherical harmonics coefficients S_r:

- **Spherical harmonics coefficients $l2$ distance loss** is the average of all spherical harmonics coefficients $l2$ distance. This metric is calculated as the numerical difference between the S_r predicted by our PointAR and the ground truth.
- **Reconstructed irradiance map $l2$ distance loss** is defined as the average of pixel-wise $l2$ distance loss on reconstructed irradiance map. In our PointAR pipeline, we need to first reconstruct an irradiance map (see Equation 7 in Ramamoorthi et al. [22]). This irradiance map is then used to calculate the $l2$ loss by comparing to the ground truth irradiance map which can be calculated directly from our dataset. We compare this metric to the diffuse loss, defined by Song et al. [23], as both representing the reconstruction quality of irradiance map.

Hyperparameter configurations. We adopted a similar model architecture used by Wu et al. [26] for ModelNet40 classification [27]. We used a set of hyperparameters, i.e., 2 PointConv blocks, each with multilayer perceptron setup of (64, 128) and (128, 256), based on accuracy and memory requirement.

Table 1. Comparison to state-of-the-art networks. Our approach PointAR (highlighted in green) achieved the lowest loss for both spherical harmonics coefficients $l2$ and irradiance map $l2$, demonstrating the efficiency of our point cloud transformation and directly learning from point clouds for indoor lighting estimation. Note Song et al. [23] used traditional diffuse convolution to generate irradiance map and did not use spherical harmonics coefficients.

Method	SHC $l2$ Loss	Irradiance Map $l2$ Loss
Song et al. [23]	N/A	0.619
Garon et al. [13]	1.10 (± 0.1)	0.63 (± 0.03)
PointAR (Ours)	0.633 (± 0.03)	0.433 (± 0.02)

Comparisons to state-of-the-art. To evaluate the lighting estimation performance of PointAR, we preformed quantitative comparison experiments with two state-of-the-art end-to-end deep learning model architectures: *(i)* Song et al. [23]; and *(ii)* Garon et al. [13]. Table 1 shows the comparison on two loss metrics.

Song et al. [23] estimates the irradiance by using a neural network pipeline that decomposes the lighting estimation task into four sub-tasks: *(i)* estimate geometry, *(ii)* observation warp, *(iii)* LDR completion, and *(iv)* HDR illumination. As we used the same dataset as Song et al., we obtained the corresponding irradiance map $l2$ loss from the paper. However, since Song et al. used the traditional diffuse convolution to obtain irradiance map, the paper did not include spherical harmonics coefficients $l2$ loss. Garon et al. [13] estimates the spherical harmonics coefficients represented lighting and a locale pixel coordinate of a given input image by training a two branches convolutional neural network with end-to-end supervision. We reproduced the network architecture and trained on the same dataset as ours without point clouds and relation E.

Table 1 shows that our PointAR achieved 31.3% and 30% lower irradiance map $l2$ loss compared to Garon et al. [13] and Song et al. [23], respectively. We attribute such improvement to PointAR's ability to handle spatially variant lighting with effective point cloud transformation. Further, the slight improvement (1.7%) on irradiance map $l2$ loss achieved by Song et al. [23] over Garon et al. [13] is likely due to the use of depth and geometry information by Song et al. [23].

Comparisons to variants. To understand the impact of neural network architecture on the lighting estimation, we further conducted two experiments that consist of *(i)* learning from projected point cloud and *(ii)* learning from point clouds with different number of points. Table 2 and Table 3 compare the model accuracy and complexity, respectively.

In the first experiment, we study the learning accuracy with two different data representations, i.e., projected point cloud and point cloud, of 3D environment. We compare learning accuracy between learning from point cloud directly with PointConv and learning projected point cloud panorama with ResNet50,

Table 2. Comparison to variants. Row 1 and 2 compare the lighting estimation accuracy with two input formats: point cloud and projected point cloud panorama image. **Row 3 to 6** compare the lighting estimation accuracy with different input point cloud downsampling level.

Method	SHC $l2$ Loss	Irradiance Map $l2$ Loss
Projected Point Cloud + ResNet50	0.781 (± 0.015)	0.535 (± 0.02)
PointAR (Point Cloud + PointConv)	0.633 (± 0.03)	0.433 (± 0.02)
512 points (40%)	0.668 (± 0.02)	0.479 (± 0.02)
768 points (60%)	0.660 (± 0.02)	0.465 (± 0.02)
1024 points (80%)	0.658 (± 0.03)	0.441 (± 0.02)
1280 points (PointAR)	0.633 (± 0.03)	0.433 (± 0.02)

which was used in Song et al. [23]. In this experiment, we observed that learning from projected point cloud result in lower accuracy (i.e., higher $l2$ losses) despite that ResNet50 requires an order of magnitude more parameters (i.e., memory) and MACs (i.e., computational requirements) than PointConv. The accuracy difference is most likely due to the need to handle image distortion that was caused by point cloud projection. Even though there might be other more suitable convolution kernels than the one used in ResNet50 for handing image distortion, the computational complexity still makes them infeasible to directly run on mobile phones. In summary, our experiment shows that learning lighting estimation from point cloud is a better way than traditional image-based learning.

In the second experiment, we evaluate the performance difference on a serial of down sampled point clouds at 40%, 60%, 80% of PointAR. From Table 3, we observe that the multiply accumulates (MACs) decreases proportionally to the number of sampled points, while parameters remain the same. This is because the total parameters of convolution layers in PointConv block do not change based on input data size, while the number of MACs depends on input size. Despite the size decrease during down sampling, we do not observe noticeable prediction accuracy decrease compared to PointAR, as shown in Table 2. An additional benefit of downsampled point cloud is the training speed, as less GPU memory is needed for the model and more can be used for loading input data. In summary, our results suggest the potential and benefit for carefully choosing the number of sampled points to trade-off goals such as training time, inference time, and inference accuracy.

Complexity comparisons. To demonstrate the efficiency of our point cloud based lighting estimation for real-time AR applications on mobile phones, we compare the resource requirements of PointAR to state-of-the-art mobile neural network architectures [16,17]. Table 3 compares the number of parameters and the computational complexity of different deep learning models. We chose these two metrics as a proxy to the inference time because they were demonstrated to be important factors [24]. Compared to the popular mobile-oriented models MobileNet [16], our PointAR only needs about 33.5% memory and 1.39X of multiple accumulates (MACs) operations. Further, as MobileNet was

Table 3. Comparison of model complexities. Row 1 to 4 compare resource complexity of ResNet50 (which is used as one component in Song et al. [23]) and efficient mobile-oriented DNNs to that of PointAR. **Row 5 to 8** compare the complexity with different input point cloud downsampling level.

Model	Parameters(M)	MACs(M)
ResNet50 [15]	25.56	4120
MobileNet v1 1.0_224 [16]	4.24	569
SqueezeNet 1_0 [17]	1.25	830
PointAR (Ours)	1.42	790
512 points (40%)	1.42	320
768 points (60%)	1.42	470
1024 points (80%)	1.42	630
1280 points (PointAR)	1.42	790

shown to produce inference results in less than 10ms [3], it indicates PointAR's potential to deliver real-time performance. Similar observations can be made when comparing to another mobile-oriented model SqueezeNet [17].

6 Related Work

Estimating lighting has been a long-standing challenge in both computer vision and computer graphics. A large body of work [12,23,11,14,29] has been proposed to address various challenges in lighting estimation and more recently for enabling real-time AR on commodity mobile devices [13].

Learning-based approaches. Recent works all formulated the indoor lighting estimation function using end-to-end neural networks [12,23,13]. Gardner et al. trained on an image dataset that does not contain depth information and their model only outputs one estimate for one image [12]. Consequently, their model lacks the ability to handle spatially varying lighting information. Similarly, Cheng et al. [8] proposed to learn a single lighting estimate in the form of spherical harmonics coefficients for an image by leveraging both the front and rear cameras of a mobile device. In contrast, Gardner et al. [12] proposed a network with a global and a local branches for estimating spatially-varying lighting information. Their model is trained on indoor RGB images and is demonstrated to generate lighting estimation quickly. However, the model still took about 20ms to run on a mobile GPU card and might not work for older mobile devices. Song et al. [23] proposed a fully differential network that consists of four components for learning respective subtasks, e.g., 3D geometric structure of the scene. Although it was shown to work well for spatially-varying lighting, this network is too complex to run on most of the mobile devices.

Despite the differences in the neural network architectures, these works all formulated the learning problem as one **directly from a single image**. Consequently, the resulting neural networks have high computational complexities and therefore might not be suitable to run on resource-constraint mobile devices, e.g., devices without GPU and powered by battery. In contrast, our PointAR

not only can estimate lighting for a given locale, but can do so quickly by directly learning from the transformed point cloud. Further, our work generated a dataset of which each scene contains a dense set of observations in the form of (point cloud, spherical harmonics coefficients).

Mobile AR. Today we are observing an increasingly rich support for developing AR applications for commodity Mobile devices such as by Google's ARCore and Apple's ARKit [2,5]. These development toolkits currently provide basic support for building the end-to-end AR pipeline and lower the barriers for AR applications development. To achieve seamless AR experience, blending virtual and physical worlds, there are still a number of challenges that are particular to mobile devices. For example, it is important detect and track the positions of physical objects so as to better overlay the virtual 3D objects [19,25,4]. Apicharttrisorn et al. [4] proposed a framework that achieves energy-efficiency in object detection and tracking by only using DNNs as needed, and leverages lightweight tracking method opportunistically. Our work shares similar performance goals, i.e., being able to run AR tasks on mobile devices, and design philosophy, i.e., by reducing the reliance on complex DNNs. Unlike prior studies, our work also focuses on rethinking and redesigning the lighting estimation, an important AR task for realistic rendering, by being mobile-aware from the outset.

7 Conclusion and Future Work

In this work, we described a hybrid lighting estimation pipeline that consists of an efficient mathematical model and a compact deep learning model for predicting lighting information at 2D locations for indoor scenes.

Our current focus is to improve the lighting estimation effect for each camera view captured by mobile devices within the real-time budgets. However, mobile AR applications might encounter heterogeneous resources, e.g., lack of mobile GPU support, and application scenarios, e.g., 60fps instead of 30fps, that might require further optimizations to meet the rigorous time budget. As part of the future work, we will explore the temporal and spatial correlation of captured image views as well as built-in mobile sensors for further improving the time efficiency of lighting estimation in an energy-aware manner.

Acknowledgement This work was funded in part by NSF Grants # 1755659 and #1815619.

References

1. `https://developers.google.com/ar/develop/unity/light-estimation/developer-guide-unity`
2. ARCore - google developers. `https://developers.google.com/ar`, accessed: 2020-3-3
3. TensorFlow Mobile and IoT Hosted Models. `https://www.tensorflow.org/lite/guide/hosted_models`

4. Apicharttrisorn, K., Ran, X., Chen, J., Krishnamurthy, S.V., Roy-Chowdhury, A.K.: Frugal following: Power thrifty object detection and tracking for mobile augmented reality. In: Proceedings of the 17th Conference on Embedded Networked Sensor Systems. pp. 96–109. SenSys '19, ACM, New York, NY, USA (2019)
5. Apple Inc: Augmented reality - apple developer. https://developer.apple.com/augmented-reality/, accessed: 2020-3-3
6. Armeni, I., Sax, A., Zamir, A.R., Savarese, S.: Joint 2D-3D-Semantic Data for Indoor Scene Understanding. ArXiv e-prints (Feb 2017)
7. Chang, A., Dai, A., Funkhouser, T., Halber, M., Niessner, M., Savva, M., Song, S., Zeng, A., Zhang, Y.: Matterport3d: Learning from rgb-d data in indoor environments. arXiv preprint arXiv:1709.06158 (2017)
8. Cheng, D., Shi, J., Chen, Y., Deng, X., Zhang, X.: Learning scene illumination by pairwise photos from rear and front mobile cameras. Comput. Graph. Forum **37**(7), 213–221 (2018), http://dblp.uni-trier.de/db/journals/cgf/cgf37.html#ChengSCDZ18
9. Chuang, Y.Y.: Camera calibration (2005)
10. Debevec, P.: Image-based lighting. In: ACM SIGGRAPH 2006 Courses, pp. 4–es (2006)
11. Gardner, M.A., Hold-Geoffroy, Y., Sunkavalli, K., Gagne, C., Lalonde, J.F.: Deep parametric indoor lighting estimation. In: The IEEE International Conference on Computer Vision (ICCV) (October 2019)
12. Gardner, M., Sunkavalli, K., Yumer, E., Shen, X., Gambaretto, E., Gagné, C., Lalonde, J.: Learning to predict indoor illumination from a single image. ACM Transactions on Graphics (2017)
13. Garon, M., Sunkavalli, K., Hadap, S., Carr, N., Lalonde, J.: Fast spatially-varying indoor lighting estimation. CVPR (2019)
14. Gruber, L., Richter-Trummer, T., Schmalstieg, D.: Real-time photometric registration from arbitrary geometry. In: 2012 IEEE international symposium on mixed and augmented reality (ISMAR). pp. 119–128. IEEE (2012)
15. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385 (2015)
16. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications (2017), http://arxiv.org/abs/1704.04861, cite arxiv:1704.04861
17. Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J., Keutzer, K.: Squeezenet: Alexnet-level accuracy with 50x fewer parameters and ¡0.5mb model size (2016), http://arxiv.org/abs/1602.07360, cite arxiv:1602.07360Comment: In ICLR Format
18. Li, Y., Bu, R., Sun, M., Wu, W., Di, X., Chen, B.: Pointcnn: Convolution on x-transformed points. In: Advances in neural information processing systems. pp. 820–830 (2018)
19. Liu, L., Li, H., Gruteser, M.: Edge assisted real-time object detection for mobile augmented reality. In: The 25th Annual International Conference on Mobile Computing and Networking (MobiCom'19)
20. Liu, W., Sun, J., Li, W., Hu, T., Wang, P.: Deep learning on point clouds and its application: A survey. Sensors **19**(19), 4188 (2019)
21. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. arXiv preprint arXiv:1612.00593 (2016)

22. Ramamoorthi, R., Hanrahan, P.: An efficient representation for irradiance environment maps. In: Proceedings of the 28th annual conference on Computer graphics and interactive techniques - SIGGRAPH '01. pp. 497–500. ACM Press, Not Known (2001). https://doi.org/10.1145/383259.383317, http://portal.acm.org/citation.cfm?doid=383259.383317
23. Song, S., Funkhouser, T.: Neural illumination: Lighting prediction for indoor environments. CVPR (2019)
24. Sze, V., Chen, Y., Yang, T., Emer, J.S.: Efficient processing of deep neural networks: A tutorial and survey. CoRR (2017), http://arxiv.org/abs/1703.09039
25. Tulloch, A., Gkioxari, G., He, K., Reblitz-Richardson, O., Zhang, P., Vajda, P., Dollar, P., Chatterjee, P., Girshick, R., Tsai, S., Cheung, V., Wang, Y., Jia, Y., He, Z.: Enabling full body AR with mask R-CNN2Go - facebook research. https://research.fb.com/blog/2018/01/enabling-full-body-ar-with-mask-r-cnn2go/ (Jan 2018), accessed: 2020-3-3
26. Wu, W., Qi, Z., Fuxin, L.: Pointconv: Deep convolutional networks on 3d point clouds. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)
27. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3d shapenets: A deep representation for volumetric shapes. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1912–1920 (2015)
28. Xu, Y., Fan, T., Xu, M., Zeng, L., Qiao, Y.: Spidercnn: Deep learning on point sets with parameterized convolutional filters. arXiv preprint arXiv:1803.11527 (2018)
29. Zhang, E., Cohen, M.F., Curless, B.: Discovering point lights with intensity distance fields. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6635–6643 (2018)