PointAR: Efficient Lighting Estimation for Mobile Augmented Reality

Yiqin Zhao(✉) and Tian Guo(✉)

Worcester Polytechnic Institute, Worcester, USA
{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 mobile deep learning models. Our pipeline, PointAR, takes a single RGB-D image captured from the mobile camera and a 2D location in that image, and estimates 2nd degree spherical harmonics coefficients. This estimated spherical harmonics coefficients can be directly utilized by rendering engines for supporting spatially variant indoor lighting, in the context of augmented reality. Our key insight is to formulate the lighting estimation as a point cloud-based 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, from a partial view, to a representation within the widely used image-based lighting model [10]. 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.

In 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 that can capture incoming lighting from every direction. However, commodity mobile phones often lack such panoramic camera, making it challenging to directly obtain accurate lighting information and necessitating the task of.

A. Vedaldi et al. (Eds.): ECCV 2020, LNCS 12368, pp. 678–693, 2020.
https://doi.org/10.1007/978-3-030-58592-1_40

Fig. 1. Rendering examples of PointAR in spatially variant lighting conditions. Row 1 shows the Stanford bunnies that were lit using the spherical harmonics coefficients predicted by PointAR. Row 2 shows the ground truth rendering results.

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, from 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°, 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 learning-based lighting estimation approaches [12,13,25] did not consider the aforementioned unique challenges of supporting lighting estimation for Mobile AR. Gardner et al. proposed a simple transformation to tackle the spatial difference between observation and rendering positions [12]. However, the proposed transformation did not use 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 [25]. However, the overall network is large in size and has high 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 a camera view with a mathematical model. In other words, we skip the use of neural networks for considering scene geometry, unlike previous methods that approached the lighting estimation with a monolithic network [12,13]. Stripping down the complexity of lighting estimation is crucial as it makes designing mobile-oriented learning models possible. Our key idea for learning lighting information directly from point clouds, instead of images, is in part inspired by the use of Monte Carlo Integration in the real-time spherical harmonics lighting calculation.

Concretely, we propose a mobile-friendly lighting estimation pipeline PointAR that combines both physical knowledge and neural network. We rethink and

Fig. 2. 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 to render 3D objects, which are then combined with the original camera view into a 2D image frame.

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 two-stage lighting estimation for mobile AR has the promise of realistic rendering effects and fast estimation speed.

PointAR takes the input of an RGB-D image and a 2D pixel coordinate (i.e., observation position) and outputs the 2nd degree spherical harmonics (SH) coefficients (i.e., a compact lighting representation of diffuse irradiance map) at a world position. The estimated SH coefficients can be directly used for rendering 3D objects, even under spatially variant lighting conditions. Figure 1 shows the visually satisfying rendering effects of Stanford bunnies at three locations with different lighting conditions. In summary, PointAR circumvents the hardware limitation (i.e., 360° cameras) and enables fast lighting estimation on commodity mobile phones.

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 [7,25]. 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 AR Lighting Estimation and Its Challenges

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 component will be used while a human user is interacting with a 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 2 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 3D virtual object, such as a couch, on the detected surface. To achieve a realistic rendering of the 3D object, i.e., seamlessly blending to the physical environment, the mobile device leverages lighting estimation methods such as those provided by AR frameworks [1,5]. The estimated lighting information will then be used by the rendering engine to relit the 3D object.

In this work, we target estimating indoor lighting which can change both spatially, e.g., due to user movement, and temporally, e.g., due to additional light sources. To provide good end-user experiences, 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 33 ms (assuming 30 fps).

However, lighting estimation for mobile AR comes with three inherent challenges that might benefit from deep learning approaches [11–13,25]. 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). Second, as indoor lighting often varies spatially, directly using lighting information at the observation location to render a 3D object can lead to undesirable visual effect. Third, battery-powered commodity mobile devices, targeted by our work, have limited hardware supports when comparing to specialized devices such as Microsoft HoloLens. This further complicates the network designs and emphasizes the importance of mobile resource efficiency.

3 Problem Formulation

We formulate the *lighting estimation in mobile augmented reality* as a SH coefficients 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 stages to achieve the goal of fast lighting estimation on commodity mobile phones.

The first stage starts with an operation $f(C, D, I)$ that generates a point cloud P_o at observation position o. This operation takes three inputs: *(i)* an RGB image, represented as C, *(ii)* the corresponding depth image, represented as D; and *(iii)* the mobile camera intrinsic I. Then $g(P_o, r)$ takes both P_o and the rendering position r, and leverages a linear translation T to generate a point cloud P_r centered at 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 Sect. 4.1.

For the second stage, we formulate the lighting estimation as a point cloud based learning problem h that takes an incomplete point cloud P_r and outputs 2nd degree SH coefficients S_r. We describe the end-to-end point cloud learning in Sects. 4.2 to Sect. 4.4.

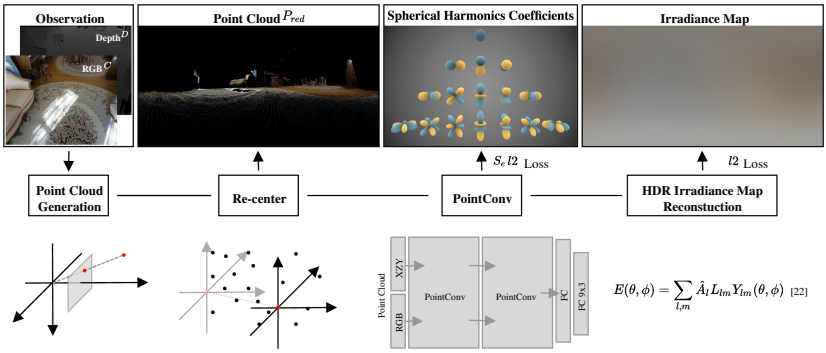

Fig. 3. PointAR pipeline composition and components. We first transform the camera view into a point cloud centered at the observation position, as described in Sect. 4.1. Then we use a compact neural network described in Sect. 4.2 to estimate SH coefficients at the rendering position. We use $l2$ loss on both estimated SH coefficients and HDR irradiance map reconstructed from spherical harmonics coefficients for evaluation.

4 PointAR Pipeline Architecture

In this section, we describe our two-stage lighting estimation pipeline PointAR, as shown in Fig. 3, that is used to model h. The first stage includes a point cloud generation and a geometry transformation modules (Sect. 4.1). The second stage corresponds to a deep learning model for estimating lighting information, represented as SH coefficients (Sect. 4.2). Compared to traditional end-to-end neural network designs [12,13,25], PointAR is more resource efficient (as illustrated in Sect. 5) and exhibits better interpretability. We describe our dataset generation in Sect. 4.3 and our design rationale in Sect. 4.4.

4.1 Point Cloud Generation

The transformation module takes an 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.

Figure 4 compares the warped results between traditional sphere warping [12] 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 only requires a simple linear matrix operation, and therefore has the promise to achieve fast lighting estimation for heterogeneous mobile hardware, during inference.

Figure 5 shows example video frames when transforming an RGB-D image at observation position by recentering and rotating the generated point cloud,

Fig. 4. 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 is not impacted by distortion and therefore is more accurate than spherical warping that only considers RGB image [12].

Fig. 5. Example video frames showing point cloud transformation 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.

to the rendering position. We detail the process of generating point cloud and its corresponding SH coefficients from a large-scale real-world indoor dataset in Sect. 4.3.

4.2 From Point Cloud to SH Coefficients Estimation

Our second component takes the point cloud P_r at rendering position r and estimates the SH coefficients S_r which is a compact representation of lighting information at location r. Our choice of learning SH coefficients S_r *directly*, instead of other representations such as image style irradiance map (Fig. 6), is largely driven by our design goal, i.e,. efficient rendering in commodity mobile phones. Formulating the illumination as a pixel-wise regression problem [25] often requires complex neural network designs. As mobile augmented reality applications often have a tight time budget, e.g., 33 ms for 30 fps 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 support rendering 3D objects directly with SH coefficients.

Fig. 6. Irradiance map comparison between ground truth and predicted ones. Row 1 shows the observation captured by mobile camera. Row 2 shows environment map generated from the dataset. Row 3 shows the irradiance map generated from the environment map using spherical harmonics convolution. Row 4 shows the reconstructed irradiance map from SH coefficients predicted by PointAR.

To train $h : h(P_r, r) = S_r$, we chose the PointConv [28] model, 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 SH coefficients $l2$ loss L_S as defined in Eq. (1), similar to Garon et al. [13].

$$L_S = \frac{1}{27} \sum_{c=1}^{3} \sum_{l=0}^{2} \sum_{m=-l}^{l} (i_{l,c}^{m*} - i_{l,c}^{m}), \tag{1}$$

where c is the color channel (RGB), l and m are the degree and order of SH coefficients. We chose to target 2nd degree SH coefficients as it is sufficient for our focused application scenarios, i.e., diffuse irradiance learning [24]. We envision that PointAR will run in tandem with existing techniques such as environment probe in modern mobile AR frameworks to support both specular and diffuse material rendering.

4.3 Dataset Generation of Point Clouds and SH Coefficients

Next, we describe how we generated a large-scale real-world training dataset by leveraging two existing datasets, i.e., Matterport3D [7] and Neural Illumination [25] datasets. Briefly, Matterport3D contains 194,400 RGB-D images forming 10,800 panoramic views for indoor scenes. Each RGB-D panorama contains aligned color and *depth* images (of size 1028×1024) for 18 viewpoints. The Neural Illumination dataset was derived from Matterport3D and contains additional 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 do so, we leveraged the pinhole camera model [9] and camera intrinsics of each photo in the dataset. For each RGB-D photo, we first recovered a small portion of missing depth data by preprocessing the corresponding depth image with the cross bilateral filter. Then we calculated the 3D point cloud coordinates (x, y, z) as:

$$x = \frac{(u - cx) * z}{fx}, \quad y = \frac{(v - cy) * z}{fy},$$

where z is the depth value in the 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. 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. To represent the rendering position for a 3D object, we used a scale factor. This allows us *(i)* to compensate for the position difference between the placement and the ground truth locations; *(ii)* to account for the potentially inaccurate depth information that was introduced by IR sensors. Currently, we used 0.95 based on empirical observation of point cloud projection.

We also used a rotation operation that aligns the recentered point cloud P_o with ground truth environment maps in our dataset. Both the recenter and rotation operations are needed during the inference to achieve spatially-variant lighting estimation and to account for the geometry surface and camera pose.

Finally, for each panorama at the rendering position, we extracted 2nd degree SH 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 in the ablation study.

4.4 Design and Training Discussions

Learning from Point Cloud. Our choices to learn lighting information *directly* from point cloud and estimating diffuse environment map are driven by mobile-specific challenges and inference time requirement. For example, 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. Furthermore, 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.

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 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 image distortion and missing pixel values. We compare this formulation to PointAR in Sect. 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. proposed the use of spherical harmonics convolution to speedup the irradiance calculation [24]. 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 [24]. 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.

Directly learning from point cloud representation can be difficult due to the sparsity of point cloud data [21]. We chose a recently proposed PointConv [28] architecture as an example point cloud-based neural network for lighting estimation (the second stage of PointAR). Other point cloud learning approaches might also be used [19,23,30].

Training Dataset. Training a neural network to accurately estimate lighting requires a large amount of real-world indoor 3D scenes that represent complicated indoor geometries and lighting variances. Furthermore, in the context of mobile AR, each training data item needs to be organized as a tuple of *(i)* a RGB-D photo (C, D) captured by mobile camera at the observation position to represent the user's observation; *(ii)* a 360° panorama 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 location to the ones at the rendering position, as well as the distance between these pixel coordinates.

Existing datasets all fall short to satisfy our learning requirements [6,7,25]. 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 Neural Illumination dataset has the format that is closest to our requirements but is still missing information such as point clouds.

Fig. 7. Rendering example from PointAR. Comparison between rendering a 3D 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 rendered objects. Row 3 and 4 show the comparison between rendered objects in original observations. Note we did not cast shadow to highlight lighting-related rendering effects to avoid misleading visual effects.

In this work, we leveraged both Matterport3D and the Neural Illumination datasets to generate a dataset that consists of point cloud data P and SH coefficients S. Each data entry is a five-item tuple represented as $((C, D), E, R, P, S)$. However, training directly on the point clouds generated from observation images are very large, e.g., 1310720 points per observation image, which complicates model training with can be inefficient as each observation image contains 1310720 points, requiring large amount of GPU memory. In our current implementation, we uniformly down-sampled each point cloud to 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 [28]. Similar to what was demonstrated by Wu et al. [28], we also observed that reducing the point cloud size, i.e., the number of points, does not necessarily lead to worse prediction but can reduce GPU memory consumption linearly during training.

5 Evaluation

We trained and tested PointAR on our generated dataset (Sect. 4.3) by following the train/test split method described in previous work [25]. Our evaluations include end-to-end comparisons (Fig. 7) to recent works on lighting estimation, ablation studies of our pipeline design, and resource complexities compared to commonly used mobile DNNs.

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 $l2$. Note Song et al. used traditional diffuse convolution to generate irradiance map and did not use SH coefficients.

Method	SH coefficients $l2$ loss	Irradiance map $l2$ loss
Song et al. [25]	N/A	0.619
Garon et al. [13]	1.10 ($\pm$ 0.1)	0.63 ($\pm$ 0.03)
PointAR (Ours)	**0.633 ($\pm$ 0.03)**	**0.433 ($\pm$ 0.02)**

Evaluation Metrics. We use the following two metrics to quantitatively evaluate the accuracy on our predicted SH coefficients S_r: *(i) SH coefficients $l2$ distance loss* is the average of all SH coefficients $l2$ distance, which is calculated as the numerical difference between the S_r predicted by PointAR and the ground truth. *(ii) 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 Eq. 7 in Ramamoorthi et al. [24]) and then compare to the ground truth extracted directly from our dataset, to calculate the $l2$ loss. We compare this metric to the diffuse loss, defined by Song et al. [25], as both representing the reconstruction quality of irradiance map.

Hyperparameter Configurations. We adopted a similar model architecture used by Wu et al. [28] for ModelNet40 classification [29]. 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.

Comparisons to State-of-the-Art. We preformed quantitative comparison experiments with two state-of-the-art end-to-end deep learning model architectures: *(i)* Song et al. [25]; and *(ii)* Garon et al. [13]. Table 1 shows the comparison on two loss metrics.

Song et al. 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 SH coefficients $l2$ loss. Garon et al. estimates the SH coefficients represented lighting and a locale pixel coordinate of a given input image by training a two-branch convolutional neural network with end-to-end supervision. We reproduced the network architecture and trained on our dataset, excluding point clouds and relation E.

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 downsampled input point clouds.

Method	SH coefficients $l2$ 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	0.668 (± 0.02)	0.479 (± 0.02)
768 points	0.660 (± 0.02)	0.465 (± 0.02)
1024 points	0.658 (± 0.03)	0.441 (± 0.02)
1280 points (PointAR)	**0.633 (± 0.03)**	**0.433 (± 0.02)**

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

Comparisons to Variants. To understand the impact of neural network architecture on the lighting estimation, we further conducted two experiments: *(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, which was used in Song et al. [25]. In this experiment, we observed that learning from projected point cloud resulted 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, 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 high computational complexity still makes them infeasible to directly run on mobile phones. In summary, our experiment shows that learning lighting estimation from point cloud achieved better performance than traditional image-based learning.

In the second experiment, we evaluate the performance difference on a serial of down-sampled point clouds. 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

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. [25]) and mobile-oriented DNNs to that of PointAR. Row 5 to 8 compare the complexity with different downsampled input point clouds.

Model	Parameters (M)	MACs (M)
ResNet50 [16]	25.56	4120
MobileNet v1 1.0_224 [17]	4.24	569
SqueezeNet 1_0 [18]	1.25	830
PointAR (Ours)	**1.42**	**790**
512 points	1.42	320
768 points	1.42	470
1024 points	1.42	630
1280 points (PointAR)	**1.42**	**790**

convolution layers in PointConv block do not change based on input data size, while the number of MACs depends on input size. Furthermore, we observe comparable prediction accuracy using down-sampled point cloud sizes 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 larger batch sizes can be used. In summary, our results suggest the potential 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 approach, we compare the resource requirements of PointAR to state-of-the-art mobile neural network architectures [17,18]. We chose the number of parameters and the computational complexity as proxies to the inference time [26]. Compared to the popular mobile-oriented models MobileNet [17], our PointAR only needs about 33.5% memory and 1.39X of multiple accumulates (MACs) operations, as shown in Table 3. Further, as MobileNet was shown to produce inference results in less than 10 ms [2], it indicates PointAR's potential to deliver real-time performance. Similar observations can be made when comparing to another mobile-oriented model SqueezeNet [18].

6 Related Work

Lighting estimation has been a long-standing challenge in both computer vision and computer graphics. A large body of work [11,12,15,25,31] has been proposed to address various challenges and more recently for enabling real-time AR on commodity mobile devices [13].

Learning-Based Approaches. Recent works all formulated the indoor lighting estimation problem by learning directly from a single image, using end-to-end neural networks [12,13,25]. 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 [12]. Similarly, Cheng et al. proposed to learn a single lighting estimate in the form of SH coefficients for an image by leveraging both the front and rear cameras of a mobile device [8]. In contrast, Garon et al. proposed a network with a global and a local branches for estimating spatially-varying lighting information by training on indoor RGB images [13]. However, the model still took about 20 ms to run on a mobile GPU card and might not work for older mobile devices. Song et al. [25] 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. In contrast, our PointAR not only can estimate lighting for a given locale (spatially-variance), but can do so quickly with a compact point cloud-based network. Further, our work generated a dataset of which each scene contains a dense set of observations in the form of (point cloud, SH coefficients).

Mobile AR. Companies are providing basic support for developing AR applications for commodity mobile devices in the form of development toolkits [1,5]. However, to achieve seamless AR experience, there are still a number of mobile-specific challenges. For example, it is important to detect and track the positions of physical objects so as to better overlay the virtual 3D objects [3,20,27]. Apicharttrisorn et al. [3] proposed a framework that achieves energy-efficiency object detection by only using DNNs as needed, and leverages lightweight tracking method opportunistically. Due to its impact on visual coherence, lighting estimation for AR has received increasing attention in recent years [4,14,22]. GLEAM proposed a non-deep learning based mobile illumination estimation framework that relies on physical light probes [22]. 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 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 two-stage lighting estimation pipeline PointAR that consists of an efficient mathematical model and a compact deep learning model. PointAR provides spatially-variant lighting estimation in the form of SH coefficients at any given 2D locations of an indoor scene.

Our current focus is to improve the lighting estimation accuracy for each camera view captured by mobile devices, given the real-time budgets.

However, mobile AR applications need to run on heterogeneous resources, e.g., lack of mobile GPU support, and have different use cases, e.g., 60 fps instead of 30 fps, which might require further performance optimizations. As part of the future work, we will explore the temporal and spatial correlation of image captures, as well as built-in mobile sensors for energy-aware performance optimization.

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

References

1. ARCore. https://developers.google.com/ar. Accessed 3 Mar 2020
2. TensorFlow Mobile and IoT Hosted Models. https://www.tensorflow.org/lite/guide/hosted_models
3. 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, SenSys 2019, pp. 96–109. ACM, New York (2019)
4. Apple: adding realistic reflections to an AR experience. https://developer.apple.com/documentation/arkit/adding_realistic_reflections_to_an_ar_experience. Accessed 10 July 2020
5. Apple Inc: Augmented reality - apple developer. https://developer.apple.com/augmented-reality/. Accessed 3 Mar 2020
6. Armeni, I., Sax, A., Zamir, A.R., Savarese, S.: Joint 2D–3D-semantic data for indoor scene understanding. ArXiv e-prints, February 2017
7. Chang, A., et al.: 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., et al.: Learning to predict indoor illumination from a single image. ACM Trans. Graph. **36**(6), 14 (2017). https://doi.org/10.1145/3130800.3130891. Article No. 176
13. Garon, M., Sunkavalli, K., Hadap, S., Carr, N., Lalonde, J.: Fast spatially-varying indoor lighting estimation. In: CVPR (2019)
14. Google. https://developers.google.com/ar/develop/unity/light-estimation/developer-guide-unity
15. 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)
16. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385 (2015)

17. Howard, A.G., et al.: MobileNets: efficient convolutional neural networks for mobile vision applications (2017). http://arxiv.org/abs/1704.04861, cite arxiv:1704.04861
18. 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
19. 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)
20. 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 2019) (2019)
21. 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)
22. Prakash, S., Bahremand, A., Nguyen, L.D., LiKamWa, R.: GLEAM: an illumination estimation framework for real-time photorealistic augmented reality on mobile devices. In: Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services, MobiSys 2019, pp. 142–154. Association for Computing Machinery, New York, June 2019
23. 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)
24. 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 2001, pp. 497–500. ACM Press, Not Known (2001). https://doi.org/10.1145/383259.383317. http://portal.acm.org/citation.cfm?doid=383259.383317
25. Song, S., Funkhouser, T.: Neural illumination: lighting prediction for indoor environments. In: CVPR (2019)
26. 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
27. Tulloch, A., et al.: Enabling full body AR with mask R-CNN2Go - facebook research, January 2018. https://research.fb.com/blog/2018/01/enabling-full-body-ar-with-mask-r-cnn2go/. Accessed 3 Mar 2020
28. 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
29. Wu, Z., et al.: 3D ShapeNets: a deep representation for volumetric shapes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1912–1920 (2015)
30. 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)
31. 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)