CS 492 BTP I
Report

Coherent Rendering for Augmented Reality

Submitted in partial fulfillment of
the requirements for the award of the degree of

Bachelor of Technology
in
Computer Science and Engineering

Submitted by

Sneha Bhakare
150050040

Under the guidance of
Prof. Parag Chaudhuri

Department of Computer Science and Engineering
Indian Institute of Technology Bombay
Mumbai, Maharashtra, India – 400076
Autumn Semester 2018

Acknowledgments

I would like to express my gratitude to Prof. Parag Chaudhuri for his valued guidance and continuous support during the entire course of work.

Sneha Bhakare
IIT Bombay
20 November, 2018

Abstract

Coherent Rendering for Augmented Reality is concerned with seamlessly blending the virtual world and real world in real time. We are interested in applying real-world light to virtual objects. This implies the measurement of the real-world lighting, also known as photometric registration. In this project we implement a method to estimate high quality illumination using convolutional neural networks (CNNs). Our represention of light model of real world is based on spherical harmonics representation. The aim of this project is to make augmentations photorealistic while removing any constraints in the environment.

Contents

Chapter 1

Introduction

1.1 Augmented Reality

Augmented reality is an attractive and exciting way to present virtual content in a real context for various application domains, like architectural visualizations, virtual prototyping, marketing and sales of not yet existing products. With such wide range of applications, AR has developed into a topic of broad and current interest in computer graphics. With the advent of GPUs and other improvements in computation speeds, Augmented Reality has become feasible in realtime.

1.2 Coherent Rendering

Many of the applications require AR to be convincing enough for a spectator to not be able to differentiate between real and virtual. This requirement demands both the real and virtual worlds to be lit with same light models. This is what is also known as visual coherence. A key step for coherent rendering is to estimate the incident lighting of a scene. Common approaches use active lightprobes to capture the light model. The major disadvantage of having a lightprobe is that it interferes with the user's experience.

Gruber et al. [3] demonstrated a probe-less method that estimates the incident light purely from reflections in the unmodified scene, scanned in real-time with a commodity RGB-D sensor. This method solves inverse rendering equation to get spherical harmonics coefficients and thus render the scene using the obtained light model. This approach forgoes artificial lightprobes, but suffers from high computational demands and cannot be used on mobile devices.

Another viable learned lightprobes method is proposed by Mandl et al. [7] The learned lightprobes work in an unmodified scene, providing unobtrusive user experience, while still retaining the computational benefits of a normal lightprobe. The incident lighting is estimated with a pre-trained convolutional neural network (CNN), which is orders of magnitude cheaper to evaluate at runtime than inverse rendering.

1.3 Spherical Harmonics

The Spherical Harmonic (SH) series of functions is the analogue of the Fourier Series for functions on the surface of a unit sphere centered at origin. The complete set of functions is an infinite-dimensional basis for functions on the sphere, but in practical use the series is truncated to give an approximation of an arbitrary function by a finite weighted sum of basis functions.

In computer graphics, spherical harmonics are used as a form of compression, which in turn greatly accelerates computations. For instance, the incoming light at a point in space is a spherical function (since it varies with direction), but using SH its approximation can be compactly represented by a handful of coefficients (the weights for the first few basis functions). This allows computations to be performed on a vector of these SH coefficients, rather than the spherical functions themselves.

The Spherical Harmonic Series is composed of separate bands of functions, usually denoted L_0, L_1, L_2 etc. The L_0 band is a single constant function; the L_1 band consists of three linear functions; the L_2 band contains five quadratic functions; and so on. Higher band indices account for finer details.

1.4 Problem Definition

The problem being attempted is to estimate the light model of real world without introducing a lightprobe, the input is constrained to be just RGB images and output rendering is expected to be calculated in realtime without any constraints in the environment. The main challenge is to use the objects in real world to act like a lightprobe. Thus, we need to learn a lightprobe from the scene. A great advantage of the learning lightprobe is that practically anything can be a lightprobe. The only practical requirement is that the lightprobe should have a good distribution of surface normals.

Chapter 2

Related Work

A large body of work on both photorealistic and non-photorealistic rendering for AR exists. Coherence requires estimation of light models. Many other issues need to be seen for photorealistic AR like occlusion, shadows due to virtual objects which are not discussed as part of this report. Here, we focus on methods that provide an estimate of the current lighting in the users environment only. Work related to major light model estimation approaches relevant to our problem statement is discussed here.

2.1 Active lightprobes

The seminal work of Debevec demonstrated how to make use of a light probe in order to directly measure the incident light [1]. Light probes are placed inside an environment and provide light measures as high-dynamic range images, which are subsequently used to compute an environment map. Active light probes obtain the environment map directly. A camera with fish-eye lens or an omni-directional camera is placed directly in the scene to acquire images of all directions in one step [2]. Passive light probes make use of a reflective object, commonly a sphere, which is placed within the scene and observed by a camera. While light probes deliver environment maps at full frame rates, they are invasive and not suitable for casual use.

2.2 Inverse Rendering Equation

As an alternative Gruber et al. [3] demonstrated a probe less method that estimates incident light purely from reflections scanned in real time using a RGB-D sensor. This method estimates radiance transfer and solves inverse rendering equation to get spherical harmonics coefficients and thus render the scene using the obtained light model. Further, this approach was improved to Gruber et al. [4] [5]. These methods forgoe artificial light probes, but suffer from high computational demands that currently cannot be run of mobile devices.

2.3 Deep outdoor illumination estimation

Hold-Geoffroy et al. [6] wherein they described use of natural available data to train a CNN to predict lighting model in outdoor environment. The idea was that to use parameters from sky luminance distribution model as the output to be learned from images via a CNN. So the neural network served as a complex mapping between input image pixels and output illumination parameters. But this approach was constrained on outdoor environments. Such sky models wont be very much useful for indoor illumination.

2.4 Learning Lightprobes

Mandl et al. [7] described an innovative approach to use CNNs to approximate the scene lighting in form of spherical harmonics (SH). The idea was that of using any known object in scene as a light probe and then creating synthetic data for the neural net using that probe and predict spherical harmonic coordinates to light the scene. The only problem was that using a known object adds constraints on the environment. The only practical requirement is that the lightprobe should have a good distribution of surface normals. The learned lightprobes work in an unmodified scene, providing unobtrusive user experience, while still retaining the computational benefits of a normal lightprobe. The incident lighting is estimated with a pre-trained convolutional neural network (CNN), which is orders of magnitude cheaper to evaluate at runtime than inverse rendering.

Chapter 3

Method

3.1 Synthesis of training images

A key aspect for CNN training is rich training data. Using image synthesis to supply sizeable amounts of training data without additional cost in human labor is an obvious approach. We first reconstruct real-world exemplars and then synthesize many variations of the exemplars appearance. We use an SH representation with four bands. This SH representation consists of 16 coefficients, in the range $[1, 1]$. 32768 illumination variations are created by setting the ambient light $C_0 = 1$ and quantizing the higher-order SH coefficients C_1 to C_15 with two different values in the range of -1 to 1. To provide the training data, we repeatedly render the known object, while systematically varying the incident illumination. For a camera pose, we render a set of training images $\mathbf{T}$, each with size 224×224. The resulting database consists of 32768 images.

3.2 Training the CNN

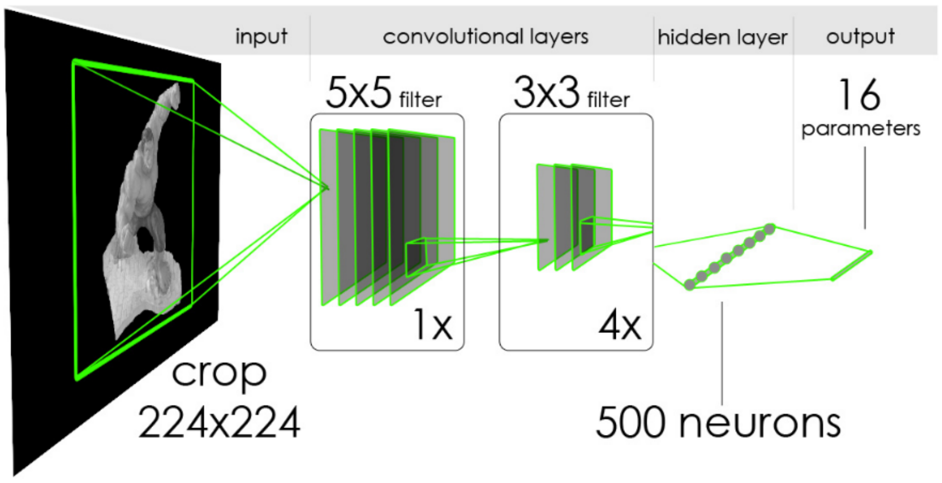

Figure 3.1: The CNN input are images with 224x224 pixels. The input images pass through five convolution layers, one hidden layer with 500 neurons and arrive at an output layer, which combines the result to 16 SH coefficients used to render the input image.

We train the CNN directly on the SH coefficients, using **T** as training data and **I** lighting as reference for the corresponding loss function. Note that this approach does not require an intermediate image storage, since only one single synthetic image is considered for training at a time. Our CNN is implemented using PyTorch, using the layout shown in Figure 3.1. Each image passes through five convolution-and-pooling layers with Rectifide Linear Unit (ReLU) activation. The first layer is using a 5×5 filter kernel, while the remaining four layers use a 3×3 kernel size. After each convolution, we downsample the image to half of its size with max-pooling, i.e., max-pooling with a 2×2 kernel with a stride of 2, which is then followed by ReLU. At the end of the pipeline, we use a standard fully connected layer with 500 neurons and tanh activation. We use ReLUs in the convolution layers to enhance the training speed, and use tanh at the end to keep the output of the network within the range of SH coefficients.

The loss function is a standard mean squared error function, thus solving a linear regression problem. For validation during training, we use one sixth of the training data as the validation set to have both rich training and validation data. For validation, we observe the Euclidean distance of the estimate to the ground-truth. We also follow the standard practices when training the CNNs. We use Stochastic Gradient Descent (SGD) and iterate the CNN with a maximum of 200 epochs, using a learning rate = 0.1.

3.3 Implementation Details

All the coding has been done in C++. Synthesis of training dataset is done using OPENGL methods by varying illumination based on SH representation. The CNN model is implemented using PyTorch framework. Training is performed on a desktop PC using an i5-7400 processor, 32 GB RAM and a GTX 1080Ti GPU with 11 GB VRAM. MATLAB is used for visualization.

Chapter 4

Results and Inference

4.1 Results

The 16 SH coefficients were quantized with 2 different values of 0.5 and -0.5 to generate a training set of 32768 images. The training time required was around 11 hours. An average mean square error of 0.01 was observed in the predicted SH coefficients.

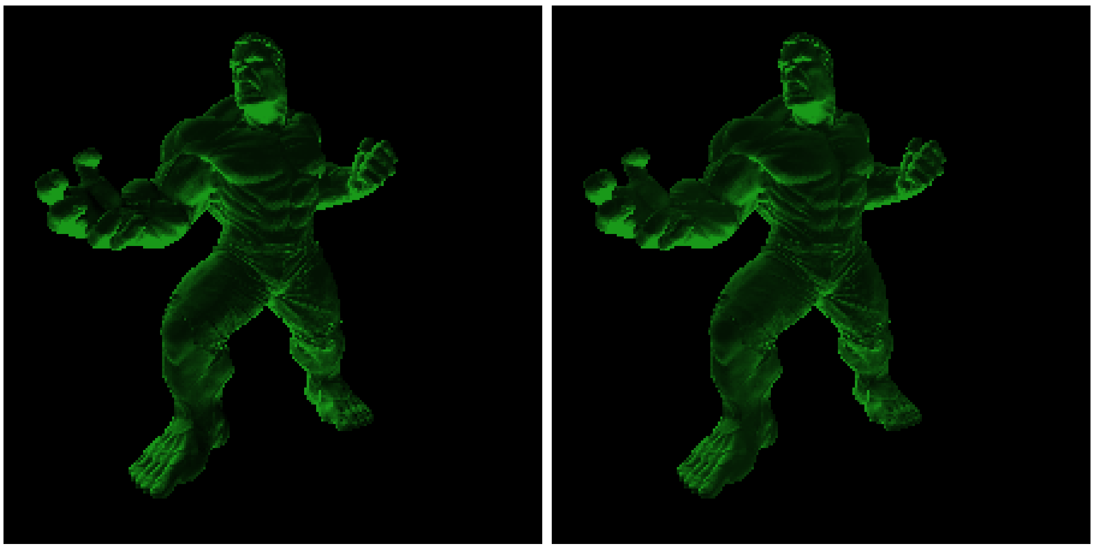

Figure 4.1: (a) Input Image, (b) Output Image

Orignal SH coefficients	1	0.5	-0.5	0.5	0.5	0.5	-0.5	-0.5
	0.5	0.5	-0.5	-0.5	0.5	0.5	0.5	0.5
Predicted SH coefficients	0.999	0.402	-0.428	0.672	0.442	0.15	-0.41	-0.434
	0.399	0.424	-0.521	-0.564	0.487	0.553	0.402	0.489

Table 4.1: Input image is rendered using the original SH coefficients, Output image is rendered using the predicted SH coefficients

Visually, image (a) and (b) in Figure 4.1 are fairly similar in terms of illumination. For better analysis, we compare these two images as shown in Figure 4.2.

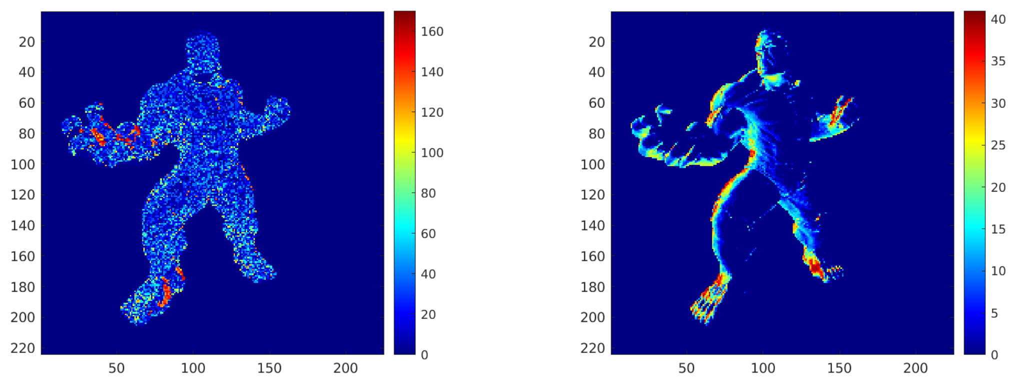
Figure 4.2: (a) Absolute pixel-wise color coded difference, (b) Model illuminated with difference between used and estimated lighting

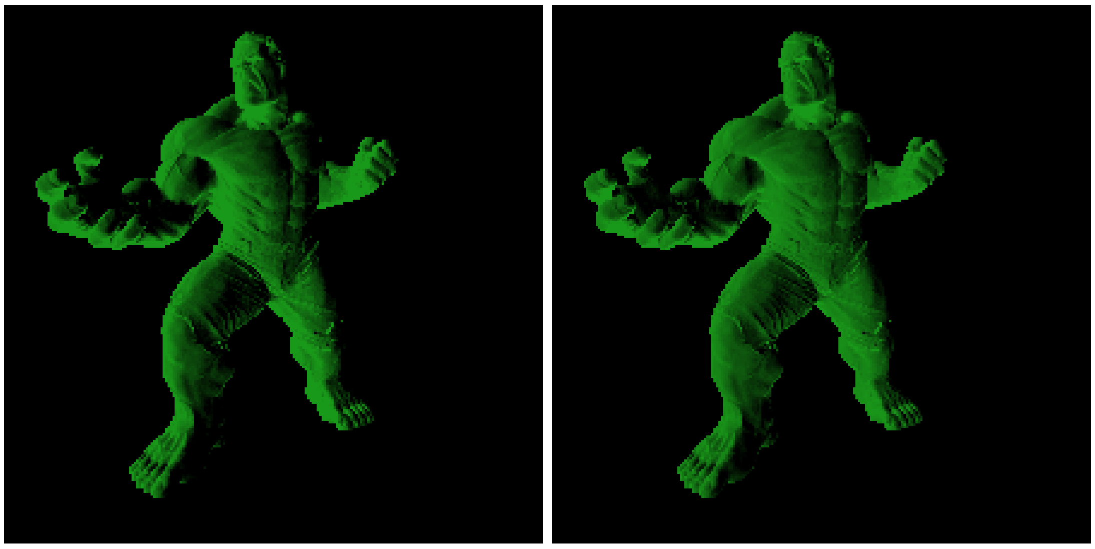
Figure 4.3: (a) Input Image, (b) Image rendered with predicted SH coefficients

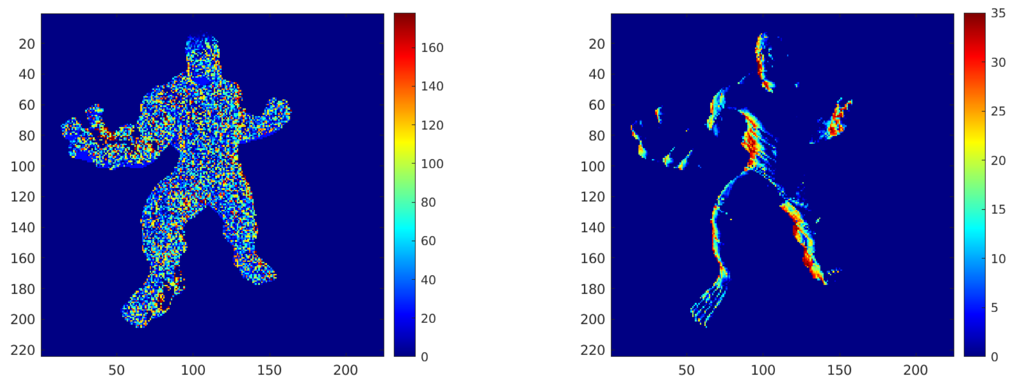
Figure 4.4: (a) Absolute pixel-wise color coded difference, (b) Model illuminated with difference between used and estimated lighting

4.2 Inference

- The average absolute error in prediction of SH coefficients is around 0.1 i.e. 5%. The input image and output image are very similar visually. Thus, this method can be used for coherent rendering.

- In Figure 4.2 (b) and Figure 4.4 (b), the difference is more on edges as compared to the remaining body. This may be a result of insufficient samples of the normal on edges.

- Images in Figure 4.2 (a) and Figure 4.4 (a) are pixelated with large variations in absolute difference values. There is no smooth pattern. The variation in case of Figure 4.4 (a) is much more than in case of Figure 4.2 (a). This is because the lighting in the second case causes saturation in some areas.

Chapter 5

Conclusion and Future Work

At present, our CNN estimates light model fairly close to the original light model. Our experimental results show that our method can render high quality photorealistic images. The accuracy of SH coefficients can be improved by further improving the CNN heuristics. We plan to extend this approach to estimate light model from objects present in environment thus getting rid of any constraints. To this aim, we will investigate combination of planes from the environment which has good distribution of surface normals and use them effectively as a plausible lightprobe. Then we plan to integrate our model with an AR rendering application to render in real environment.

References

[1] P. Debevec., 1998 Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography.

[2] M. Knecht, C. Traxler, O. Mattausch, W. Purgathofer, and M. Wimmer., 2010 Differential instant radiosity for mixed reality.

[3] Gruber, L., Richter-Trummer, T., and Schmalstieg, D., 2012, Real-time photometric registration from arbitrary geometry

[4] L. Gruber, T. Langlotz, P. Sen, T. Hoherer, and D. Schmalstieg., 2014, Efficient and robust radiance transfer for probeless photorealistic augmented reality.

[5] L. Gruber, Jonathan Ventura, and D. Schmalstieg., 2015, Image-Space Illumination for Augmented Reality in Dynamic Environments

[6] Hold-Geoffroy, Y., Sunkavalli, K., Hadap, S., Gambaretto, E., and Lalonde, J.-F., 2017, Deep outdoor illumination estimation

[7] Mandl, D., Yi, K. M., Mohr, P., Roth, P. M., Fua, P., Lepetit, V., Schmalstieg, D., and Kalkofen, D., 2017, Learning lightprobes for mixed reality illumination