

ESTIMATION OF ENVIRONMENTAL LIGHTING FROM
HUMAN FACE FOR ILLUMINATION OF AUGMENTED
REALITY SCENES

Emre Koç

Submitted to the Graduate School of Sabancı University
in partial fulfillment of the requirements for the degree of
Master of Science

Sabancı University

August, 2011

ESTIMATION OF ENVIRONMENTAL LIGHTING FROM HUMAN
FACE FOR ILLUMINATION OF AUGMENTED REALITY SCENES

APPROVED BY

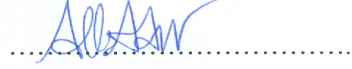
Assist. Prof. Dr. Selim BALCISOY
(Thesis Supervisor)



Assoc. Prof. Dr. Yücel SAYGIN



Assoc. Prof. Dr. Albert LEVI



Assoc. Prof. Dr. Erkey SAVAŞ



Assist. Prof. Dr. Gözde ÜNAL



DATE OF APPROVAL: ..09/08/2011.....

© Emre Koç 2011

All Rights Reserved

ESTIMATION OF ENVIRONMENTAL LIGHTING FROM HUMAN FACE FOR ILLUMINATION OF AUGMENTED REALITY SCENES

Emre Koç

EECS, M.Sc. Thesis, 2011

Thesis Supervisor: Assist. Prof. Dr. Selim BALCISOY

Keywords: Augmented Reality, Illumination, Real-time Rendering, Spherical Harmonics

Abstract

In this thesis, we propose a method to solve a common problem in augmented reality domain; estimating light sources in an outdoor scene and lighting virtual objects accordingly. As a basis of our method we developed a framework based on estimation of environmental lighting from well defined objects, specifically human faces. The method is tuned for outdoor use, and the algorithm is further enhanced to illuminate virtual objects exposed to direct sunlight.

In the first part of this thesis, we propose a novel lighting estimation technique where we assume a user is looking straight to mobile devices camera. This technique extracts information from input images to calculate possible light sources to pass to the rendering stage.

In the second part of this thesis, we propose a lighting model which uses the output from our lighting estimation in order to make objects appear as they are lit correctly by the sun light. This model uses a mathematical technique called Spherical Harmonics Lighting for real-time realistic rendering.

ESTIMATION OF ENVIRONMENTAL LIGHTING FROM HUMAN FACE FOR ILLUMINATION OF AUGMENTED REALITY SCENES

Emre Koç

EECS, Yüksek Lisans Tezi, 2011

Tez Danışmanı: Yrd. Doç. Dr. Selim BALCISOY

Keywords: Arttırılmış Gerçeklik, Işıklandırma, Gerçek Zamanlı Görselleme,
Küresel Harmonikler

Özet

Bu tez, arttırılmış gerçeklik ortamlarında ortak bir sorun olan, çevresel ışıkların analizi ve sahneye konulacak cisimlerin bu ışık bilgisine göre ışıklandırılmasına uygun bir metod sunuyor. Bu metodun temelinde arttırılmış gerçeklik uygulamalarında şekli belirli cisimlerden, özellikle insan yüzü, çevresel ışıkların analizini yapan bir yapı mevcuttur. Metod özellikle dış mekan kullanımı için düzenlenmiş ve doğrudan güneş ışığı tarafından aydınlatılan objeleri gerçeklemek için ayarlanmıştır.

Tezin ilk kısmında, mobil cihaza doğrudan bakan bir insanın yüzünden çevresel ışık bilgisinin nasıl çıkarılacağı anlatılmaktadır. Bu teknik ön kameradan gelen resimleri muhtemel ışık kaynaklarına bakarak analiz etmektedir.

Tezin ikinci kısmında ise ilk aşamada elde edilen ışık yön bilgisini kullanarak gerçek zamanlı ve görsel kalitesi yüksek bir biçimde sanal cisim görsellenme tekniği anlatılmaktadır. Bu model Küresel Harmonik Işıklandırması adında matematiksel bir teknik kullanılmaktadır.

Acknowledgements

I would like to express my deep and sincere gratitude to my supervisor, Selim Balcısoy for his patience, continuous support and excellent guidance. Our brainstorming sessions are very valuable for me where we created ideas for my academic papers and my thesis.

I have been honored to have Yücel Saygın, Albert Levi, Erkay Savaş and Gözde Ünal as members of my thesis committee. I am grateful for their valuable review and comments on the thesis.

I would like to thank all my colleagues in Computer Graphics Laboratory who contributed immensely to my personal and professional life. I specially thank Çağatay Turkey, Selçuk Sümengen, Mustafa Tolga Eren and Ceren Kayalar for their continuous support in my research. I would like to thank Uraz Cengiz Türker, Serdar Adali, Farhood Negin and Murat Çelik Cansoy for their great friendship and support.

Thanks to all my friends and colleagues for their support. I am very grateful for the time spent with the friends and memories. I also thank my friends from Gravi Ltd, Tolga Birdal, Emrah Bala and Bulut Karakaya for their inspirations during my studies.

Finally, I would like to thank my family for always loving and supporting me on my way through my 20 years of education. Their deep patience and guidance led me to my success in academic studies.

Contents

1	Introduction	1
2	Related Work & Background	3
2.1	Light Source Estimation	3
2.2	Lighting Virtual Objects	6
2.3	Mobile Technology	7
2.3.1	Mobile Devices	8
2.3.2	Augmented Reality	9
2.4	Motivation	12
3	Estimating Light Directions	14
3.1	Face Detection and Pose Estimation	14
3.1.1	faceAPI Tracking	14
3.1.2	OpenCV Face Detection	15
3.2	Estimating Azimuth and Zenith Angles	16
3.2.1	Clustering Intensity Vectors	19
3.3	Light Direction Projection	20
4	Lighting Virtual Objects	22
4.1	Spherical Harmonics	23
4.2	Skylight Model and Tone Mapping	29
5	Case Study	31
5.1	Controlled Environment Tests	31
5.2	Outdoor Tests	32
5.3	Discussion on System Performance	34
6	Conclusion and Future Work	37

List of Figures

2.1	Sony Ericsson W900i with two sided camera (left), NVIDIA GoGorce 4800 graphics processor running a real-time 3D Demo [?]	8
2.2	Using the Augmented Reality Interface (Left), Virtual Shared White Board (Right) [?]	11
3.1	faceAPI tracking in 3D space (left), face texture output (right)	15
3.2	HSL color space contains lightness information and RGB space defines a color by using three main colors [?] (left), Applying bilateral filter to lightness image gives very smooth lightness image without any high frequency texture details (right)	17
3.3	A spherical coordinate system with zenith direction Z and azimuth axis A . The point has radius $r = 4$, zenith angle $\theta = 70^\circ$, and azimuth $\varphi = 130^\circ$ [?]	18
3.4	Matching Normal map with captured and bilateral filtered image.	18
3.5	Clustering of k-means algorithm does not allow membership functions to have floating point values (Left), FCM allows a point to belong to one or more clusters.	19
3.6	Yaw, Pitch and Roll on a Virtual Reality Head Mounted Display	21
4.1	How High is The Tennis Player? [?]	22
4.2	Rendering outputs with Spherical Harmonics. Rendering output with no light in OpenGL (a), Diffuse lighting in OpenGL (b), Diffuse Lighting with Spherical Harmonics (c), Diffuse Lighting with Occlusion information (d)	28
5.1	Controlled Environment Test Image Sequence	32
5.2	Face Images taken in different sky conditions	33
5.3	Virtual Object rendered in different lighting conditions. Color values are similar to the grass colors which our object have similar color.	34

5.4 Rendering with only sun light configuration with Preetham sky-light model (a), Reference sun position and angle from camera direction (b), Reference face image for light direction extraction (c) 35

List of Tables

2.1	OpenGL ES supported device list	10
5.1	Real Measured Light Directions (R), value returned by our estimation (A) and absolute error (E) for the zenith ϕ and azimuth θ angles in the scene. Test images are marked with their respective letters in Figure 5.1	32
5.2	Light Source direction estimation for images in Figure 5.2	33

Chapter 1

Introduction

Virtual Object Lighting constitute a critical component in Augmented Reality (AR) applications. Today it is common to see many AR applications in web sites, desktop applications and mobile devices. In order to increase the feeling of reality in an AR scene, virtual objects should be lit realistically by using the information from the surrounding environment. In this thesis, we propose methods to solve two distinct issues in AR domain. First of these issues is extracting illumination information from human face in an outdoor scene and second one is the lighting of virtual objects realistically by using the extracted lighting information.

The core element of our methods is a framework which uses Lambertian surface properties to extract lighting information from a predefined geometry, which is specifically human face in our solution. This framework contains a face detection solution in order to place face model to the input image. After placing the model we use the framework to extract light directions and process sensor data so that real world light directions and light intensity information can be calculated correctly.

Environmental lighting is essential for realistic perception of augmented reality scenes; therefore extracting lighting information from predefined objects has always been an interesting problem for the graphics community. A recent survey by Swan and Gabbard [?] discusses the effects of lighting and shadows on users

perception of reality and depth. Augmented reality applications that provide only tracking a coded marker or tracking a surface, do not provide a real depth feeling to users. It is also hard to extract lighting information by using predefined objects like a mirrored ball in outdoor applications. Hence, we need a tool which will allow us to get information about the environmental lighting which neither depends on any special object, which users do not carry mostly with themselves, for light source detection nor depends on the visibility of sky, a large area of ground or some nearby objects in order to get a reference lighting and shadow estimation. To help users who are investigating their surrounding environment from a screen, an AR system should be able to detect lighting information dynamically and update the virtual objects *lighting and shadow information* in real-time. Our motivation is to find a light estimation technique that can extract lighting information for a scene while user investigates the given outdoor area.

Outline of the rest of the thesis is as follows: In Chapter 2, we give an overview of previous research on both light direction estimation and virtual object lighting techniques. In Chapter 3, we define how our light estimation system work and its basis. Chapter 4 explains how we efficiently light objects on a mobile system with estimated light directions. Chapter 5 shows test results for the proposed system. Finally on Chapter 6, we summarize our results and propose some future work on the framework.

Chapter 2

Related Work & Background

Both of light source estimation from a given geometry and lighting in augmented reality involve extensive literatures. Therefore, we will review these both fields under two main titles.

2.1 Light Source Estimation

Several light source estimation techniques have been proposed in the literature. We will try to review them according to their technique on how to capture light information from the environment.

Light source detection from a predefined geometry is the main technique for most of the previous works. Previously, there were extensive studies about shape and reflectance recovery by using the method shape from shading [?, ?, ?, ?]. But these studies did not extend to recovering illumination information from an object with known shape and reflectance but focus on light direction. Sato et al. [?] proposed a technique to handle this problem with the help of cast shadows of an object to a scene. This method works on scenes with cast shadows but there were no extension to extraction of light information only from a non-convex object. Due to the difficulty of the problem, most studies involve several assumptions. Jensen et al. [?] proposed a solution to solve real-time image based lighting problem for outdoor augmented reality scenes where the illumination conditions

are dynamically changing. They created some constraints and assumptions like no precipitation as they heavily depend on surface reflectance in the environment. Apart from that what makes this system hard to adapt in different environments is that they need a 3D model of the scene with an HDR environment map recorded in the centre of the scene. Even though they do not require a detailed 3D model, even a rough model for a new environment is hard to create in mobile augmented reality scenarios.

Different group of researchers are interested in resolving the photometric problem, estimating light source direction in real-time by using specific shiny object, most popularly mirrored balls [?]. Debevec [?] proposed a technique to capture scene radiance and global illumination in order to have correct lighting for virtual objects. Even though the technique presented have realistic visual quality renders, using a mirror ball and correctly placing it in the center of a scene to capture HDR images takes a long time to setup and cannot accommodate lighting changes in real-time. A recent and interesting study by Tominaga et al. [?] estimate the illuminant spectra of an omni-directional light distribution from the images of a camera aiming at a mirrored ball. They proposed algorithms to measure *spectral radiance distributions* in an outdoor scene. By extracting radiance distributions, they achieve a realistic representation of both sky and sun but again the problem of using a hard-to-find mirrored ball arises in mobile outdoor augmented reality. Kanbara et al. [?] proposed a technique for *real-time estimation of light source environment* in an augmented reality system. They used a marker tracking system that contains a small black mirrored ball. Even though this method allows real-time tracking of the light changes it is not feasible in a large augmented reality setup like excavation sites. Since light source estimation depends on the small mirrored ball, environment effects such as dust or mud might occlude the clear reflections on its surface.

Another group of researchers studied light source estimation from a scene by using fish-eye lens cameras. Yoo et al. [?] captures light information by using a 185 degree fish-eye lens and Neutral density filter (NDF) to have no changes in hue of color rendition by modifying intensity of all wavelengths or colors of light equally. They stated that lighting maps can be extracted precisely by using fish eye lenses with NDF. Frahm et al. [?] extended the method of using fish-eye lens camera for light estimation to use in a multi-camera system of a TV studio. They proposed a fish-eye camera image segmentation system that defines 3D camera positions in a stage so that they can lit virtual objects correctly as they really appear in the scene. Both of these fish-eye camera models can give very detailed information about lighting information in a scene but they are hard to setup and hard to maintain since the main perquisite is they should always face light sources. This constraint makes them impractical for rapidly changing mobile augmented reality applications.

Estimating light information from human eye is also a well studied topic. Tsumura et al. [?] determines where the light source is by using the reflections in the image of the eye. They propose a method that uses reflectivity of human eye to capture up to three light sources. In [?], they present a detailed analysis about the characteristics of the cornea image of an eye which was taken by a catadioptric (mirror + lens) imaging system. They show that geometric parameters of the corneal system are suitable for environment map extraction from a single image. Wang et al. [?] improves this technique by separating corneal reflections in an image of human irises. They estimate illumination from the surrounding scene by using human iris features such as chromaticity and illumination correspondence between two irises. Even though extracting environment map from eye seems feasible in indoor scenes, it becomes infeasible in outdoor scenes where direct sunlight comes into the eye.

Estimating lighting information from human face was studied for cancelling

out lighting variations in order to enhance the performance of face detection/recognition applications [?, ?]. Basri et al. [?] shown a low-dimensional linear subspace which is effective for face recognition by using spherical harmonics basis. They used spherical harmonics for representing lighting information on human face. Lee et al. [?] proposed a technique for estimating directional lighting in uncalibrated images of faces in frontal pose. They estimated the principal lighting direction by using least-squares formulation with Lambertian illumination model. Their technique uses surface normals as light direction vectors and they assume light direction does not change with intensity information in each pixel, which is same as our assumption.

2.2 Lighting Virtual Objects

Rendering realistic light in real-time has been a long challenge in computer graphics community. There have been several methods for calculating shading like Phong interpolation [?], normal-vector interpolation shading [?], per-fragment lighting [?]. Most of these techniques can be implemented faster than real-time in current graphics hardware. Even though these models can shade virtual objects correctly for a viewer to have a depth perception, they are unable to simulate occlusion of light and shadow casting. Williams proposed a solution to shadow calculations [?] called *shadow mapping*. This method allowed graphics community to have complex shadows even for dense geometries or curved surfaces. [?] extended this technique by proposing Percentage-Closer Soft Shadows. They used a filtering technique to make shadow maps look more natural with soft edges. Although calculating soft shadows runs in real-time, having more realistic scenes requires powerful display cards and programmable pipeline capabilities. When the lighting requirements extend to area lights and inter-reflections between objects, we might need to take off-line rendering methods into consideration. Monte Carlo ray tracing [?] [?], radiosity [?], or multi-pass rendering [?] that calculates lighting information by summing up intensities from multiple point light sources.

These methods are able to provide higher visual quality than standard lighting methods in computer graphics but they are costly and their complexity increase proportionally with included lights in the scene.

Spherical Harmonics Lighting (SHL) was introduced to graphics community by Sloan et al. [?]. They introduced a method that transforms low-frequency incident lighting to transferred radiance which includes shadows, occlusions and inter-reflections. They defined these transferred functions in low-order spherical harmonics. They even extended this work by compressed per-point transfer matrices from a high-dimensional surface signal by *clustered principal component analysis* [?]. Green explained the underlying mathematics of the proposed method and expressed several details about spherical harmonics lighting that the original paper did not cover and this paper was the main guide of our rendering system development [?]. These methods opened a new way of defining global illumination in computer graphics. Previous techniques allowed realistic renderings with several assumptions but they mostly capture light information from a limited angle. SHL allows dynamic changes in light and even dynamically moving objects.

2.3 Mobile Technology

Technology drives the innovation, so what is very expensive few years before is easy to buy for a few hundred dollars. This progress increased the availability of common hardware needed for computer graphics applications. Also the commonness of sensors, gyroscopes and GPS chips increased so much that they are very easy to find in most of the recent mobile devices. Even though the technology renews itself rapidly users had problems with systems that contain new technologies that lack user guidance.

2.3.1 Mobile Devices

Processing power is the main need for having extensive graphics quality both for games and visual applications. Since the power comes from both CPU and GPU, they play a very crucial role in developing advanced visualizations. Even though the mobile phones are not a very brand new technology, which is first introduced by Dr Martin Cooper of Motorola in 1973 [?], transition from ordinary phones that can just send and receive phone calls, to smart phones that have extensive processing power, both from their CPU and graphics processor, changed how end-users use their mobile devices. For instance Sony Ericsson W900i, launched in 2005, was the first mobile phone to have a real graphics processing chip named NVIDIA GoForce 4800.



Figure 2.1: Sony Ericsson W900i with two sided camera (left), NVIDIA GoForce 4800 graphics processor running a real-time 3D Demo [?]

Having mobile graphics processor allowed developers to work on applications that use special hardware features such as auxiliary buffers or specialized API calls. Also having a camera built in a phone emerged very later than the first mobile phone that allow just phone calls. Since initial internal cameras had a very poor quality, they were only used as a MMS (Multi Media Messaging) sending tool which cannot capture much detail from a scene. With the advancements in CMOS technology phone manufacturers integrated more megapixels in to mobile devices. As the technology got cheaper phone manufacturers are able to place two cameras in one device one points to user and other one pointing opposite direction, ex. Nokia 6280 in 2005. Even though this new camera setup

opened a new era for mobile phones to have video conferencing, there are still problems in software side that make developing applications for mobile phones not easy. Starting from smart phones manufacturers integrated OpenGL capabilities, however the initial versions did not cover extensive features. (See Table 2.1 on page 10)

Even after having several smart phone models in the market, without a community for developers and a robust SDK , mobile software market stayed as a niche topic that only some researchers worked on to build advanced applications. Of course, there has been research going on especially for windows mobile platforms like HP iPAQ but the applications are mostly limited to academic community and even though the innovation in those applications are extensive, they could not find any distribution in market because of the platform [?]. With the release of iPhone in 2007, smart phone market got into a new era of large displays with multi-touch capabilities. In the same year Google Inc. announced Android platform to several mobile phone manufacturers and the first device hit on market was HTC Dream on 22 October 2008 [?]. With the help of new development platforms and application stores, mobile software development became popular as never before and users started to investigate new applications that use available hardware extensively. For instance Layar which is a company found in 2009 developed a mobile augmented reality browser that partner companies are able to put their location aware content to real world coordinates. Users are now able to browse even real estate posts around them with help of AR.

2.3.2 Augmented Reality

Having a virtual reality system was first introduced by Sutherland in 1966 by the invention of head mounted display [?]. Later in 1990, Tom Caudell expressed the phrase 'Augmented Reality' in Boeing when helping workers to assembly cables into aircraft. Rosenberg [?] developed the first functioning AR system called "Virtual Fixtures" and demonstrated its benefits to human performance. Most of

<p>OpenGL ES 1.0</p> <ul style="list-style-type: none"> • Official 3D graphics API of the operating systems Android and Symbian • Supported by the PlayStation 3 as one of official graphics APIs (the other one being low level libgcm library), the PlayStation 3 also includes several features of OpenGL ES 2.0 • Supported by QNX
<p>OpenGL ES 1.1</p> <ul style="list-style-type: none"> • Supported by Android 1.6 • Supported by iOS for iPad, iPhone, and iPod Touch • Supported by the BlackBerry 5.0 operating system series, however, only BlackBerry Storm 2, BlackBerry Curve 8530 and later models have the needed hardware. • Supported for Palm webOS, using the Plug-in Development Kit • Supported by the Nintendo 3DS
<p>OpenGL ES 2.0</p> <ul style="list-style-type: none"> • Supported by the iPad, iPhone 3GS or later, and iPod Touch 3rd generation and later • Supported by the Android platform since Android 2.2 • Supported by the Android platform NDK since Android 2.0 • Supported by the BlackBerry PlayBook • 3D Library of the Pandora console • Chosen for WebGL: OpenGL for web browsers • Supported by some new Nokia mobile phones, such as the Maemo based Nokia N900 and the Symbian based Nokia N8 • Supported by various Samsung mobile phones, including the Galaxy S and Wave • Supported for Palm webOS, using the Plug-in Development Kit • Supported by the Archos Internet tablets: Archos 70 IT, Archos 101 IT

Table 2.1: OpenGL ES supported device list

the previous AR systems are based on information display in a spatial domain. Having extensive graphics in real-time was a challenge with the low processing power in hand. With the technical developments in graphics processors they started to be able to render 3D scenes faster. Also pose estimation in a real scene was a major problem. Knowledge about the camera calibration parameters and position tracking of the object in 3D Euclidean space was needed to correlate target position between subsequent frames [?] [?]. Durlach et al. [?] concluded that position tracking with magnetic markers are prone to noise and may not be enough for real correlation of virtual and real images. That's why there is a need for an image based method for the final fine tuning. Kato et al. introduced a system that allows tracking of fiducial markers and a calibration method for optical see-through HMD based on the marker tracking [?].



Figure 2.2: Using the Augmented Reality Interface (Left), Virtual Shared White Board (Right) [?]

2.4 Motivation

There are several methods proposed both for exporting light source direction and illumination of virtual objects in a real scene. These methods motivated us to develop an augmented reality framework that works on a specific scenario where a user points a mobile device, which has double sided camera, to a field where user wants to inspect any kind of virtual content. Primarily we needed a light source estimator that will work on a specific object, mainly human face, and do not depend on expensive math calculations. Secondly, since most of the previous work on light source estimation works on a single image or a series of images which are consequently taken, they did not need a light space projection between two different cameras. So we needed sensor information on how our mobile device is placed in real world in order to relate extracted light information with the augmented view. Third, in out-door scenes sun is the main light source, so we wanted to know in which direction the sun is coming to our scene. Fourth, there is a need to know where we will put our virtual objects in the view of the back facing camera, so there should be a tracking mechanism involved. Last, we need to illuminate virtual objects in an augmented reality setup to make them seem as if they are in the same scene.

In order to suffice these requirements in a framework, we eliminated light source estimation techniques that take very long time and need extensive geometry information about the scene. We focused on estimating light source automatically, without having any user interaction or requesting any information from user. Since we defined our system to work on mobile environment, we put constraint on the mobile device hardware to have Digital Compass, Gyroscope and two-sided camera setup which are very common in recent end-user phones and tablets. Even though we made some tests in indoor controlled environment, there is a need to assume that a light source is placed distantly. Our assumption is that our face and virtual object that will be placed has the same illumination prop-

erties. So we defined our system to work only for daylight configuration in an outdoor scene and we used a tracking system that works with square markers.

In practical terms we wanted to build a framework that can be used in scenarios where users navigate through outdoor environment for information retrieval. For instance using this framework in an excavation site can help archaeologists to investigate that site with correct depth perception with the help of realistic lighting of virtual objects. Having a depth perception is a crucial feature in virtual object rendering in augmented reality which we explained in detail 4. By having a correct depth perception archaeologists can use this framework for virtual object placement on the excavation site. Usage of this system can also be extended to historical site visitors to show how a demolished historical building would be seen in that time of the year.

Chapter 3

Estimating Light Directions

In this section we will introduce the light source estimation engine which constitutes the core of our outdoor lighting framework. First, we will describe how we place our face model to front-facing camera view and then we will continue with explaining the light direction estimation procedure.

3.1 Face Detection and Pose Estimation

Even though detecting faces is straight forward with in an outdoor scene, light variations can make this process an ill-posed problem. We investigated two different systems for tracking of human face.

3.1.1 faceAPI Tracking

Seeing Machines is an award winning company with a focus on vision based human machine interfaces [?]. One of their successful products is the faceAPI which tracks human face in real-time by option to export face image that is being tracked. Even though this system works correctly when light is evenly distributed to face, it stops tracking when light conditions change. For instance, in the case of a user rotating around him for 360 degrees, even though the tracker which is based on image features continues to function, face texture output is suspended.

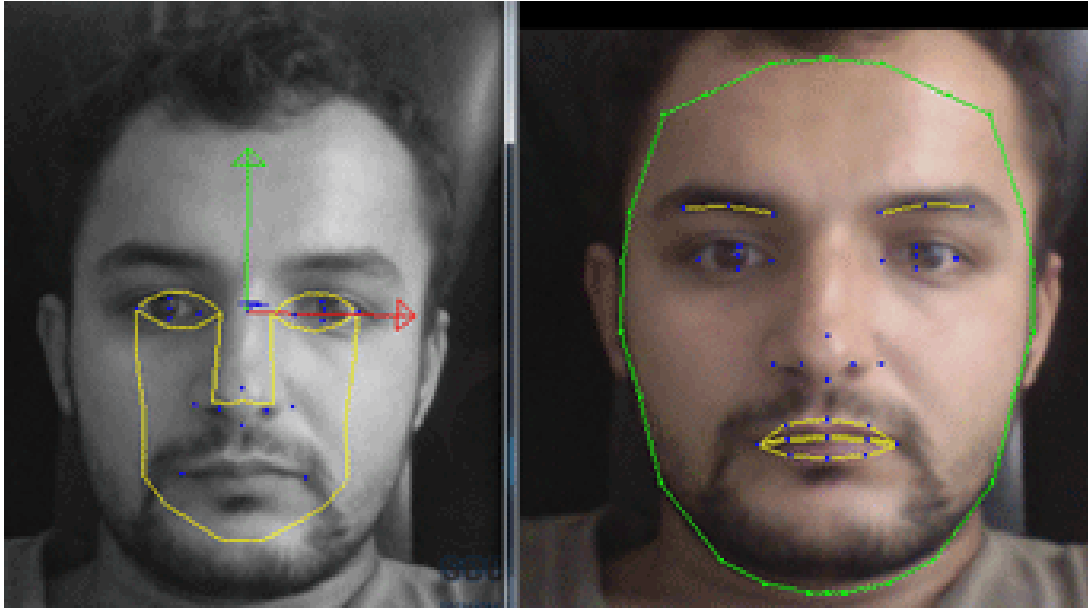


Figure 3.1: faceAPI tracking in 3D space (left), face texture output (right)

3.1.2 OpenCV Face Detection

OpenCV is a library of programming functions for real time computer vision [?]. Object detector of OpenCV works with a classifier that can detect any type of object with the given training set. Classifier is trained with a few hundreds of sample views of a particular object like face, car etc. and it can find the trained object if it exists in the image. It is of course hard for this system to work on all kinds of configurations. But classifier works well with precomputed data for a human face, since we limited our system to specific constraints. These constraints are:

- User should be guided to look at a specified position on the screen
- Only rotation on z axis is allowed. (axis that is perpendicular to screen)
- Distance between user's face and mobile device should be limited

We do not have any other assumptions for this system. We know that our user is directly looking at mobile device screen and front-facing camera is placed very close to the screen. So we assume that the distance of a user face from the camera is between 20 cm 80 cm. Cascaded Haar classifier also allows capturing left and

right eyes position separately. We defined a condition to get face rotation from the image as follows by using Equation 3.1.

$$\beta_{Head} = \begin{cases} \arctan 2(P_{R_y} - P_{L_y}, P_{R_x} - P_{L_x}) & \text{if } P_{R_x} > P_{C_x} \text{ and } P_{L_x} < P_{C_x} \\ \arctan 2(P_{L_y} - P_{R_y}, P_{L_x} - P_{R_x}) & \text{if } P_{R_x} < P_{C_x} \text{ and } P_{L_x} > P_{C_x} \\ 0 & \text{otherwise} \end{cases} \quad (3.1)$$

Where β represents 'Roll' of user head, P_C is the center position of face region and P_R and P_L are right and left eye positions respectively. In this definition as the classifier can output rotations up to 25 degrees, from the conducted experiments, we can say that in most cases the system can detect roll angles up to 15 degrees.

3.2 Estimating Azimuth and Zenith Angles

In this section we will show how we compute azimuth and zenith angles from a given input image by using our prior knowledge of human face geometry. In an outdoor setup we know that in a daylight configuration, we expect to see the direct sunlight's specular component on a user's face. One problem can be the occlusion of clouds which will have a direct effect on the specular component's visibility. We defined a similar approach to [?] by defining each light with an unknown luminance L_j and unknown unit direction $\omega_j, j = 1 \dots N$ and we also assume human face as a Lambertian surface. We start by converting color domain of our image from RGB color space to HLS color space (See Figure 3.2 left side) where HLS stands for hue, saturation and lightness. Now we have a single channel image that contains lightness information invariant from hue and saturation information.

Even though the image is invariant from the color information, it may contain high frequency variations on its texture. We apply bilateral filtering to remove any

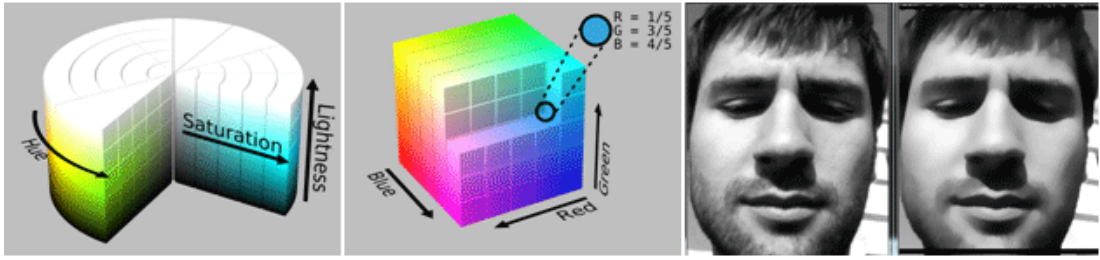


Figure 3.2: HSL color space contains lightness information and RGB space defines a color by using three main colors [?] (left), Applying bilateral filter to lightness image gives very smooth lightness image without any high frequency texture details (right)

high frequency variations on the face texture [?]. Since we have a smooth face model that contains lighting information and our assumption about the surface is Lambertian. From the Lambertian Cosine Law, a lambertian surface will have the same apparent brightness from any angle that it is viewed. But we know that our camera is placed as parallel as possible to user's face since face detection will not work otherwise, so we propose a technique for this specific scenario where a ray traced from our pinhole camera to users face reflected by that surface normal approximates light direction more accurately. This scenario hold especially when light is received from angles in the range $[\pi..2\pi]$, which are azimuth angles where light is coming from back side of user's head. If we were to assume N to show the direction of light, azimuth angles would be in range $[0..\pi]$ which will not be enough to approximate light vectors coming from back side.

For the lighting of virtual objects we use light directions in spherical coordinates so we can define light direction by our reflection vector for each pixel with the following formula

$$\begin{aligned}
 r &= \sqrt{x^2 + y^2 + z^2} \\
 \theta &= \cos^{-1} \left(\frac{z}{r} \right) \\
 \varphi &= \tan^{-1} \left(\frac{y}{x} \right)
 \end{aligned} \tag{3.2}$$

where θ denotes angle of light from the z up axis, φ defines the angle of light

in projected to horizontal x-y plane. x , y and z are the values of a vector that is the reflection of vector coming from the camera by the surface normal at given pixel. $r = 1$ since we use normalized vectors.

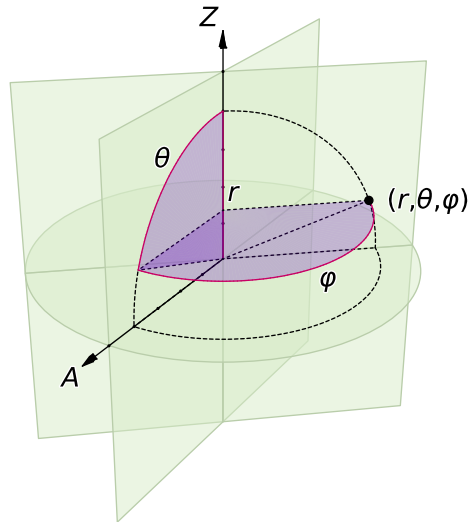


Figure 3.3: A spherical coordinate system with zenith direction Z and azimuth axis A . The point has radius $r = 4$, zenith angle $\theta = 70^\circ$, and azimuth $\varphi = 130^\circ$ [?]

As we have two spherical coordinates in the face plane and intensity for each pixel, we cluster these spherical coordinates with respect to their magnitude and see if we have any light source other than sun light.



Figure 3.4: Matching Normal map with captured and bilateral filtered image.

In order to simplify clustering of vectors we will assume that our base intensity value is the average intensity in the face texture. This assumption may fail in a perfect diffuse lighting environment during an overcast day but in that condition we can only use a dim ambient light for the object. Masking the input vectors by thresholding the input image leaves only the vectors that have higher

intensity from the specific direction and we call these as *Intensity Vectors* defined in spherical coordinates.

3.2.1 Clustering Intensity Vectors

The notion of *k-means clustering* is suitable for grouping a set of points but we assume that in the case where we have more than one light source or a high intensity reflection from a different direction than sun light, any pixel in the image may be lit by the two light sources. We borrow the idea of *Fuzzy C-Means (FCM)* which was developed by [?] [?] allows one piece of data to belong to two or more clusters.

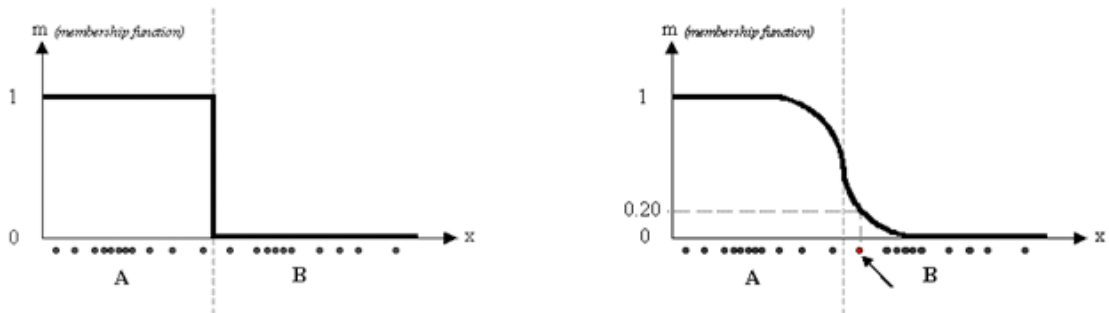


Figure 3.5: Clustering of k-means algorithm does not allow membership functions to have floating point values (Left), FCM allows a point to belong to one or more clusters.

We do not know how many light sources that are reflecting through the user's face so we assume there is only sun light in the scene and intensity vectors should point close to that direction. Calculating standard deviation of azimuth angle gives how diverse are the light vectors distributed in the scene. Azimuth angle defines the light direction rotated in z axis; in our conducted tests with our mobile platform, by using 85 different images we concluded that having a standard deviation larger than 20 degrees shows user's face is lit by another strong light source. During the initial calculation of standard deviation on azimuth angle, in the case where the value is large, we initialize FCM with cluster count two. We repeat this step for each cluster till we get an acceptable standard deviation.

Diversity in zenith angle is not a straightforward feature to extract as azimuth

angle. In the case where we have one sun light and a very strong reflection from a floor it would be very hard to distinguish the large intensity distribution of sun light from reflection of the floor in the same azimuth angle. In order to capture any extreme lighting conditions where there is a strict distinction between intensity vectors, we apply a separate FCM process by starting with 2 clusters as explained with azimuth clustering. In the first iteration, we check for mean angle differences between two clusters. If the mean value difference is larger than defined angle limit parameter of our algorithm, which we defined as 30 degrees during our tests, it is possible to conclude that we have a separate light source coming through a different angle, it is left to the application programmer to define the limits of cluster count and angle difference limit for zenith angles.

3.3 Light Direction Projection

Projection of light direction from front-facing camera to back-facing camera is one of the fundamental features of our system. In previous chapter we have shown that we can find azimuth and zenith angles from the face image but these values are in the coordinate space of mobile devices front-facing camera. In order to map these spherical coordinates to real world coordinates, we need to use some sensor information available as a requirement in our mobile device.

Sensor initialization plays a crucial role in determining how the mobile device is positioned in world coordinates. Digital compass streams information about device rotation with respect to world's true north. Even though the sensor is accurate, there is a need to make sure that device is positioned perpendicular to the vector coming from the center of our planet. In this stage we can continue with the initialization of gyroscope. Gyroscope is a device for measuring orientation that we can use it to track rotations in three dimensions Yaw, Pitch and Roll (See Figure 3.6). Since the only available data from the gyroscope is amount of change in orientation, there is a need to initialize Yaw axis by fixed rotation information which is the compass direction. We need to inform users about

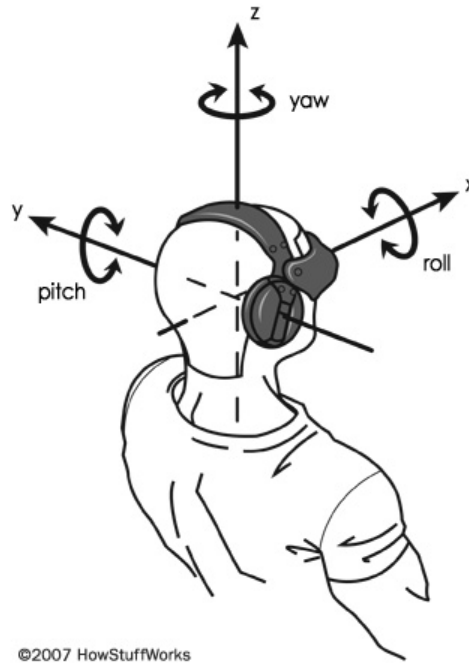


Figure 3.6: Yaw, Pitch and Roll on a Virtual Reality Head Mounted Display

the initialization process and guide them to place the device perpendicularly to world's axis by using the accelerometer information. As this process is done once for each execution of the software it does not affect usability of the system.

Knowing the device orientation in real world space allows us to project lighting information captured from the front-facing camera to real world coordinates with the predicted intensities.

Chapter 4

Lighting Virtual Objects

Working on a mobile platform requires further optimizations for having real-time rendering rates as a result of lower CPU and GPU power. In previous augmented reality applications most of the work has been done on how people can use this new technology in different environments [?]. In those applications researchers defined how the interaction with augmented reality system should be defined [?].

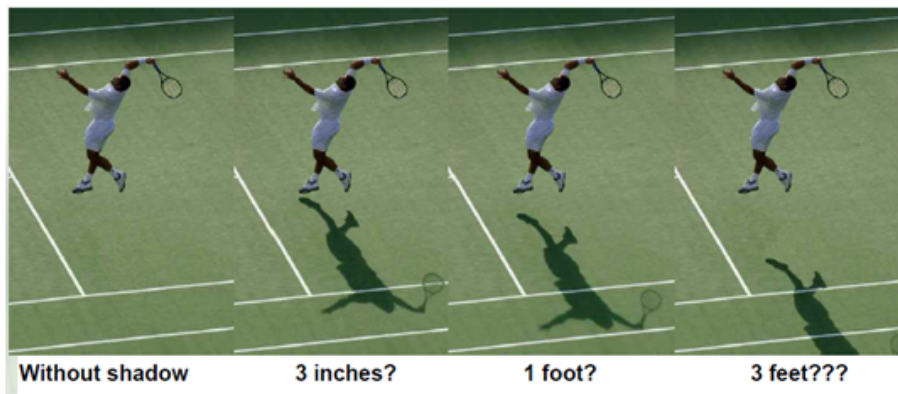


Figure 4.1: How High is The Tennis Player? [?]

Shadow mapping is an essential component for a user to understand where an object is placed and at what height and depth it resides. As stated in [?], shadows are effective for recognizing spatial relationships in the depth direction. The effect was especially significant in the case of monocular display where there are no stereo spatial cues. Such a result underlines the importance of shadow casting for depth perception. Since we are working on a single display mobile

device, users can only understand the depth information by using shadows (See Figure 4.1).

4.1 Spherical Harmonics

Real-time lighting in a mobile device requires an efficient method for realistic rendering. There are implementations that can visualize any type of graphics realistically but they work very slowly due to low frequency of mobile processors [?]. Lighting equation is formed by a complex integral that calculates all light formation in continuous domain.

$$L(x, \vec{\omega}_0) = L_e(x, \vec{\omega}_0) + \int_S f_r(x, \vec{\omega}_i \rightarrow \vec{\omega}_0) L(x', \vec{\omega}_i) G(x, x') V(x, x') d\omega_i \quad (4.1)$$

Where

$L(x, \vec{\omega}_0)$ = the intensity reflected from position x in direction ω_0

$L_e(x, \vec{\omega}_0)$ = the light emitted from x by this object itself

$f_r(x, \vec{\omega}_i \rightarrow \vec{\omega}_0)$ = the BRDF of the surface at point x , transforming incoming light ω_i to reflected light ω_0

$L(x', \vec{\omega}_i)$ = light from x' on another object arriving along ω_i

$G(x, x')$ = the geometric relationship between x and x'

$V(x, x')$ = a visibility test, returns 1 if x can see x' , 0 otherwise

Bidirectional reflectance distribution function which is abbreviated as BRDF in Equation 4.1 is a four-dimensional function that defines how a light vector is reflected on an opaque surface. It's parameters are incoming light direction ω_i , outgoing light direction ω_o defined in surface normal space for each azimuth angle φ and zenith angle θ which makes it four dimensional. For realistic rendering of objects BRDF property can be used to include detailed surface properties but

it adds a complex pre-calculation routine to rendering system. Memory requirements increase as we need to store detailed reflectance properties for each vertex. Since we work on mobile environment, we left integration of BRDF as a future work.

Calculation of the lighting equation integral in a mobile real-time augmented reality environment is time consuming and we need to make some simplifications. For shiny scenes it is convenient to use cube maps for reflections that seem realistic. However, this type of rendering still accounts a simple diffuse lighting model which only outputs a color depending on the incoming light direction, surface normal and surface's material diffuse color. In order to have shadows in a virtual scene, geometry information needs to be used. As described in [?] casting shadows of curved geometries can be accomplished by using a multi-pass algorithm where in each step a depth texture taken from the light's point of view needs to be transformed to camera's view space. After the transformation, shadows are calculated by comparing each depth value of current view with the transformed depth values and marking pixels which do not have equal depths as shadowed.

Shadow mapping techniques have been improved after the first proposition of the algorithm but mobile device hardware and OpenGL ES specification did not develop as fast as graphics hardware available for personal computers. For a brief definition of OpenGL ES support on mobile devices please see Table 2.1. In our system we assumed that our system will support OpenGL ES 1.1. Since we have a limited computing power we need to use techniques that are fast to calculate and require as little special feature of the hardware as possible.

Spherical Harmonics Lighting (SHL) presented [?] allows us to compress lighting information for a given model in an efficient way. Spherical harmonics (SH) are special types of basis functions which can be used to reconstruct any function. This type of feature is analogous to Fourier transform which works over the unit circle for one-dimensional functions (See Equation 4.2) where a) shows reconstructing with basis functions in one dimension and b) shows approximation

using spherical harmonics projection. SH contains a large set of mathematical definitions which we will cover only the crucial points in this chapter, for more information about the details of SHL please refer to [?].

$$f(x) = \lim_{n \rightarrow \infty} \sum_{i=0}^n c_i B_i(x) \qquad \tilde{f}(s) = \sum_{l=0}^n \sum_{m=-l}^l c_l^m y_l^m(s) = \sum_{i=0}^{n^2} c_i y_i(s)$$

where where (4.2)

$$c_i = \int_D f(x) B_i(x) dx \qquad c_l^m = \int_s f(s) y_l^m(s) ds$$

a) b)

where $y_l^m(s)$ represents Associated Legendre Polynomials with two arguments l and m . l defined as band index and takes positive integer values. We can also define reconstruction equation with 1D indexing and rewrite summation in range $[0..N^2]$. These polynomials defined over range $[-1, 1]$ and they return real numbers. For each band polynomials are orthogonal w.r.t. a constant term and between bands they are orthogonal with a different constant. This allows us to calculate coefficients required for real-time reconstruction by multiplying Legendre polynomials with our lighting function $f(s)$.

Reconstruction process is a time consuming computation defined continuously over a sphere. In order to compute SH coefficients we will use a mathematical tool called Monte Carlo Estimator (MCE). Integral estimation of any function f can be computed by summing the product of the value of that function at sample points by the value of the probability density function p (Equation 4.3.a). If we also assume that sample points that we take around a sphere are uniform, equation will be as simple as the function f sampled at uniform sample points divided by a weight (Equation 4.3.b).

$$\begin{aligned}
E[f(x)] &= \int f(x)dx = \int \frac{f(x)}{p(x)}p(x)dx \\
&\approx \frac{1}{N} \sum_{i=1}^N \frac{f(x_i)}{p(x_i)} & \text{(a)} \\
&\approx \frac{1}{N} \sum_{i=1}^N f(x_i)w(x_i) & \text{(b)} \\
&= \frac{4\pi}{N} \sum_{i=1}^N f(x_i) & \text{(c)}
\end{aligned} \tag{4.3}$$

As we are considering a set of integration over a surface of a unit sphere, sample points should be distributed over the surface uniformly. Since we know that the surface area of unit sphere is 4π our weighting function becomes constant as equal to the area of the unit sphere (Equation 4.3.c). Now we can define Monte Carlo integration for SH coefficients as follows

$$c_i = \frac{4\pi}{N} \sum_{j=1}^N f(x_j)y_i(x_j) \tag{4.4}$$

here c_i represents a coefficient that we can use to approximate our lighting function. These coefficients are stored for each vertex in our model. For N bands we have N^2 coefficients.

So we can define a simplified version of the rendering equation as

$$L(x) = \frac{\rho_x}{\pi} \int_{\Omega} L_i(x, \omega_i) V(\omega_i) \max(N_x \cdot \omega_i, 0) d\omega_i \tag{4.5}$$

Where

$L_i(x, \omega_i)$ = incoming light at point x along vector ω_i

ρ_x = the surface albedo at point x

N_x = the surface normal at point x

$V(\omega_i)$ = a visibility test, returns 1 if ray is not blocked by self, 0 otherwise

So the transfer function for each vertex becomes

$$M^{DS} = V(\omega_i) \max(N \cdot \omega_i, 0) \quad (4.6)$$

Algorithm 1 Pseudo code for calculating vertex color values by using coefficients

```

for  $i = 0 \rightarrow NumOfSamples$  do
   $Hs \leftarrow DOT(sample[i].vec, normal)$ 
  if  $Hs > 0$  then
    if  $SelfShadow(pos, sample[i].vec) = 0$  then
      for  $j = 0 \rightarrow NumOfCoefficients$  do
         $Intensity \leftarrow Hs * sample[i].coeff[j]$ 
         $ResultRed[j] = Intensity * MaterialRed$ 
         $ResultGreen[j] = Intensity * MaterialGreen$ 
         $ResultBlue[j] = Intensity * MaterialBlue$ 
      end for
    end if
  end if
end for
 $Factor \leftarrow Area/NumOfSamples$ 
for  $i = 0 \rightarrow NumOfCoefficients$  do
   $CoeffRed[i] = ResultRed[i] * Factor$ 
   $CoeffGreen[i] = ResultGreen[i] * Factor$ 
   $CoeffBlue[i] = ResultBlue[i] * Factor$ 
end for

```

The main property of SH Lighting which allows us to use it on most of the mobile platforms available today is that, it is very easy to define global lighting, shadow and occlusion information by using SH coefficients. We only need to store required coefficients within each vertex so that for each rendering step lighting information can be constructed by multiplying the computed light coefficients with vertex coefficients. Since we multiply all coefficients as much as the samples that are calculated along a sphere, it is straightforward to have light intensity information for all light directions by using the orthogonality feature of SH. Rotation in SH domain is not straightforward as in Cartesian coordinate system and we want to rotate SH coefficients to define light direction. One downside of having a need to rotate SH coefficients is, for each object in the scene we need to rotate light coefficients specifically to that object. Also having more than one light source increases the computation need for each object but there is an oppor-

tunity to cache rotated coefficients in case of a static scene for each object that is being rendered is mostly static. (See Figure 4.2).

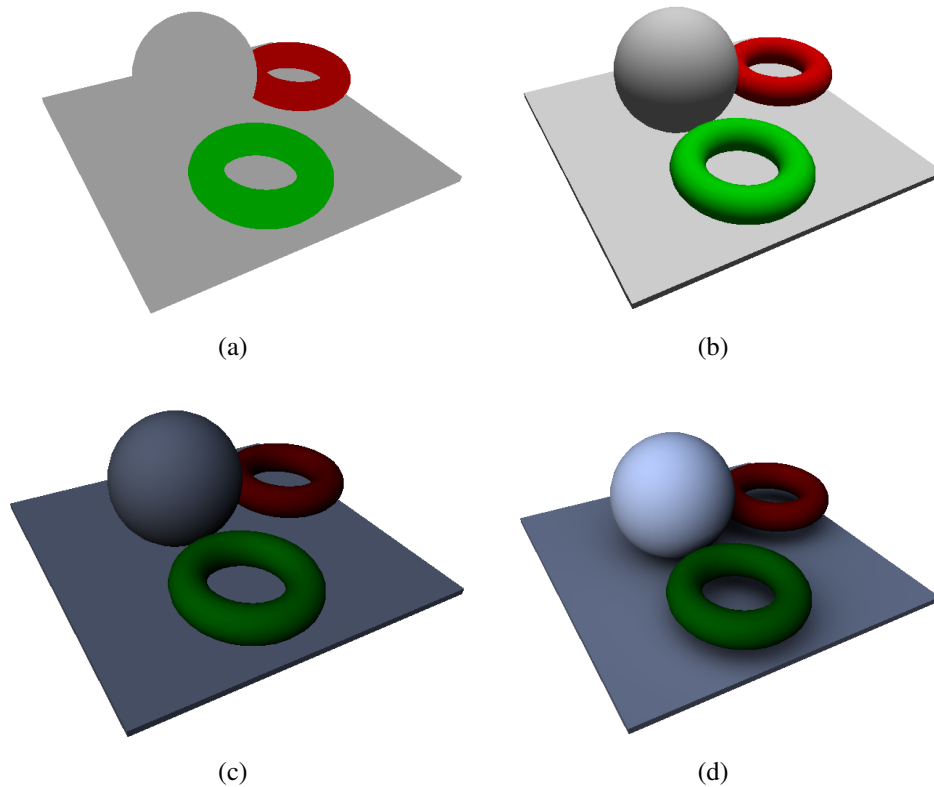


Figure 4.2: Rendering outputs with Spherical Harmonics. Rendering output with no light in OpenGL (a), Diffuse lighting in OpenGL (b), Diffuse Lighting with Spherical Harmonics (c), Diffuse Lighting with Occlusion information (d)

Lighting in computer graphics can be achieved with different types of methods. We tried to show the difference between SH Lighting and standard OpenGL lighting in Figure 4.2. In standard lighting in OpenGL with only ambient lighting present, we can only see the outlines of objects. By adding diffuse lighting, we can perceive the shading and surface information of objects correctly as compared to Figure 4.2a. Calculating diffuse lighting without any shadow calculation in SHL is more efficient than using default OpenGL lighting since we reuse the calculated parameters from SH coefficients in each frame. Even though we see the surface structure, we cannot be sure about the depth of objects. As we can see in Figure 4.1, shadow information allows us to perceive depth information correctly. So we use spherical harmonics coefficients which are calculated by

considering the parameter $V(\omega_i)$ in 4.5. This gives us both shadow and occlusion information for objects in the scene (See Figure 4.2d).

4.2 Skylight Model and Tone Mapping

Lighting from sunlight needs a robust skylight model. We used Preetham [?] skylight model for which was explained in detail for spherical harmonics lighting in [?]. Output of Preetham skylight model gives High Dynamic Range (HDR) color. We implemented Reinhard tone mapper [?] for tone mapping HDR values to visible range. Most of HDR calculation is done in image domain where a virtual scene is rendered to a render target that supports 16 bit Floating point values. Then algorithm for tone mapping is applied to pixels. We implemented this system for per-vertex color calculation.

We approximate the key of the scene by using Equation 4.7.

$$\bar{L}_\omega = \frac{1}{N} \exp\left(\sum_i \log(\delta + L_\omega(i))\right) \quad (4.7)$$

Where $L_\omega(i)$ is the "world" luminance for vertex i , N is the total number of pixels in the image and δ is a small value to avoid the singularity that occurs if vertices exist that do not receive any light. As mentioned in [?], if the scene has a normal-key we would like to map this to middle-grey of the displayed image by using Equation 4.8.

$$L(i) = \frac{\alpha}{\bar{L}_\omega} L_\omega(i) \quad (4.8)$$

Where $L(i)$ is a scaled luminance and $\alpha = 0.18$ called exposure. For low-key and high-key vertex colors they allowed α to be changed as proportional to 0.18. We used this average luminance value to tone-map HDR colors with Equation 4.9.

$$L_d(i) = \frac{L(i)(1 + \frac{L(i)}{L_{whitepoint}^2})}{1 + L(i)} \quad (4.9)$$

We did not implement automatic dodging-and-burning feature explained in the paper, which controls how very high luminance appear as blurry. From our tests in an outdoor environment with diffuse object above formula gives acceptable results. Implementing automatic dodging-and-burning also requires convolution operations which Reinhard implemented by using multiplication in FFT domain. Even though this gives better results in highlights it adds more complexity to each frames render routine.

Chapter 5

Case Study

We run our defined methods on several input images which are taken various times of the day. Development of our methods started with a Sony VAIO UX Micro PC which had double sided camera setup but no sensors attached built-in. Since we needed sensor information along the input images and recently the most common tablet platform emerged as iPad, we decided to collect data by using it. Switching target platform which has a totally different development environment during our research had a side effect that we did not have libraries to be compiled for that platform. So we gathered all images by using two sided camera built in to iPad and read sensor information in order to process them on a PC. In order to make sure our system is cross-platform compatible we only used open-source libraries which can be ported to any mobile platform.

5.1 Controlled Environment Tests

In order to see our system's performance clearly we captured several images in a controlled environment where we specify the light direction with only a single light source without any reflections from other surfaces.

These tests are performed on Macbook Pro with the built in i-Sight camera. OpenCV face detector runs at 25 fps and applying bilateral filtering to face region detected slows down the capture rate to 15-20 fps where capturing gets slower as



Figure 5.1: Controlled Environment Test Image Sequence

the face image regions occupies a larger area. Results show that for controlled light environments where light source is not infinite as sun light, highlights on the image can mislead the algorithms since there are two separate peak points in the image as a result of very close lighting. From the algorithmic perspective we can treat these types of differences as different light sources or we can limit FCM clustering and take the average of the light directions for a single light.

		ϕ	θ
Image A	R	110°	160°
	A	88.54°	175.2°
	E	22.54°	15.2°
Image B	R	10°	90°
	A	21.3°	101.50°
	E	11.3°	11.50°
Image C	R	110°	140°
	A	95.48°	144.7°
	E	14.52°	4.7°

Table 5.1: Real Measured Light Directions (R), value returned by our estimation (A) and absolute error (E) for the zenith ϕ and azimuth θ angles in the scene. Test images are marked with their respective letters in Figure 5.1

5.2 Outdoor Tests

We also made tests in outdoor environment in order to measure the performance of the algorithm with sunlight. In outdoor, our system works as expected since the

sunlight comes from an infinitive direction. The main problem with this approach is having an overcast day where clouds cover the intensity of the sun. Cameras available in mobile environments use auto-adjustments for exposure setting and for now we can only obtain the camera parameters via EXIF information embedded in JPEG files. As a parameter for global light intensity we use exposure value from embedded image information and define exposure in HDR lighting in inverse-proportion. Mapping between camera and HDR exposure level are not exact representations. So it is expected that we have some inconsistencies in object lighting due to overcast weather conditions.

The difference between intensity images in a strong day light and a full overcast sky shown in Figure 5.2. Rendering with the input light directions using SHL can be seen in Figure 5.3.

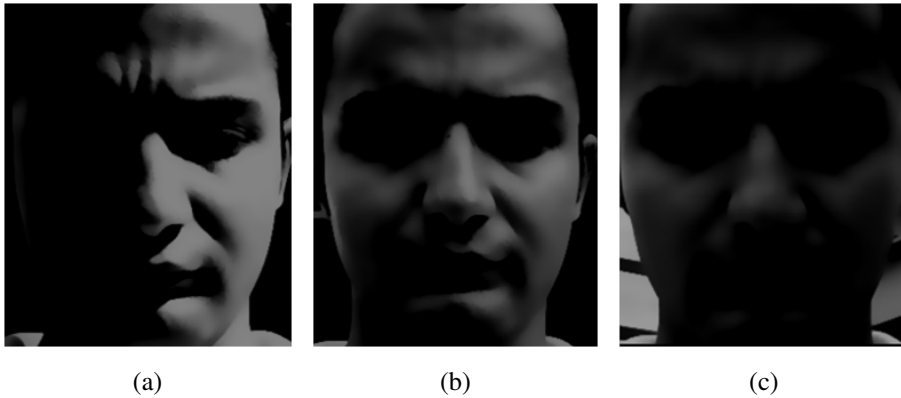


Figure 5.2: Face Images taken in different sky conditions

	Image A		Image B		Image C	
	ϕ	θ	ϕ	θ	ϕ	θ
R	51.81°	99.96°	51.81°	99.96°	52.56°	241.55°
A	75.4°	105.7°	87.9°	125.23°	32.2°	182.8°
E	23.59°	5.74°	36.09°	25.27°	19.61°	59.47°

Table 5.2: Light Source direction estimation for images in Figure 5.2

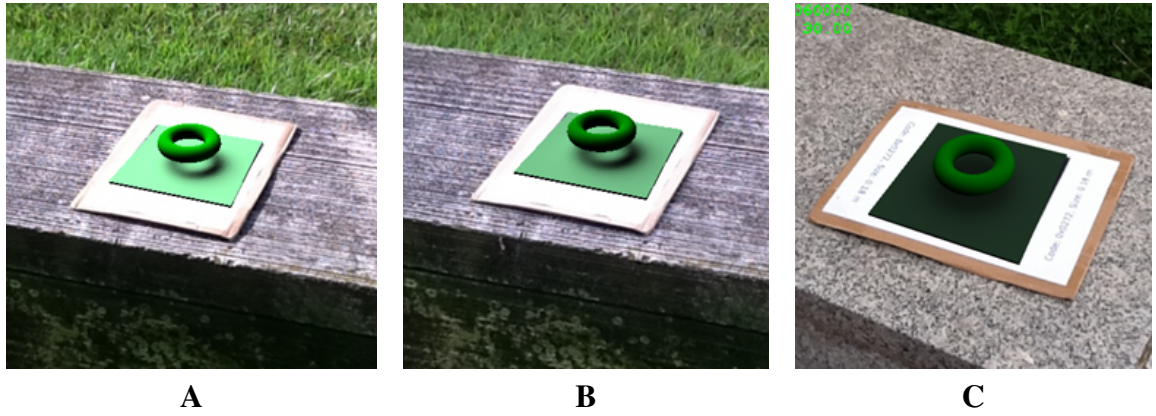
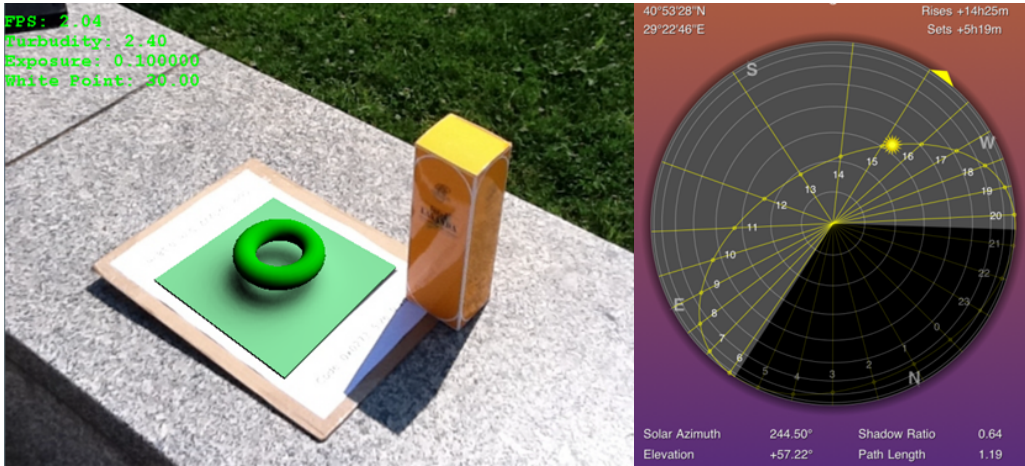


Figure 5.3: Virtual Object rendered in different lighting conditions. Color values are similar to the grass colors which our object have similar color.

5.3 Discussion on System Performance

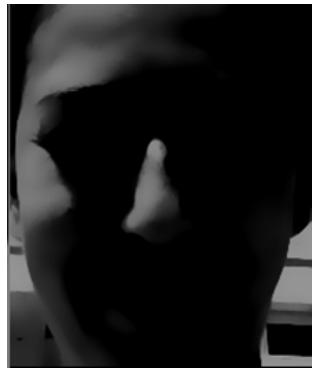
Detecting outdoor lighting from a single image can have many side effects depending on the weather conditions. We have tested and demonstrated our method in strong, mid and low direct sun light. Performance in different weather conditions like snow, storm or sky covered with dense clouds are not evaluated so there is no data available for those scenarios. However our system is capable of extracting lighting direction in a mobile augmented reality scenario. Sensor results are measured very precisely for testing purposes and any kind of drifting in sensor results are neglected during tests. Since our assumption is that user look directly to our mobile device screen, software cannot detect a correct direction in case if there is a head rotation in other z-axis. As can be seen from 5.2c in a totally overcast weather our software's rate of error increases dramatically in azimuth direction (See Table 5.2 Image C θ value).

This is a result of full diffuse lighting in sky which distributes light evenly so that extracting a direction is generally impossible. Exposure level is an indication of such scenarios where we detect a large standard deviation in light directions and the exposure level is slower than a direct sun light scene. In the case where we have a large standard deviation and intensities in the image are not high as direct sunlight configuration we decrease the used SH coefficient bands to 2 where first



(a)

(b)



(c)

Figure 5.4: Rendering with only sun light configuration with Preetham skylight model (a), Reference sun position and angle from camera direction (b), Reference face image for light direction extraction (c)

band defines ambient color and second band defines diffuse color. Having more bands allows us to capture more detail for both shadow and lighting but in full overcast scenes the detail needed for lighting coefficients is less.

Chapter 6

Conclusion and Future Work

Our contributions in this thesis can be summarized as:

- We proposed a new technique on how to extract lighting information from a real scene by using human face.
- We proposed a complete system to handle augmented reality illumination in outdoor scenes.
- Our system is build on top of most common open-source libraries like OpenCV and ARToolkit. These libraries have been ported to different mobile platforms which allows our system to be integrated easily to different hardware.
- Our system allows mobile application programmers to use recent hardware in mobile devices for real-time augmented reality applications.

Our work is a novel method to capture a simplified illumination model in real-time. This model can be extended to more complex lighting calculations by using a robust 3D face tracker system. Future works may include:

- Developing a complete mobile application that contains all proposed features built-in on an Android or iPad device.
- Implementing Spherical Harmonics Lighting for dynamically changing objects.

- Conducting several user studies in various locations and different times of the year in order to detect lighting errors in different sky and weather conditions.

In this thesis we aimed to propose a real-time light extraction method specifically tuned for mobile devices. This system can be easy to integrate into different platform while requiring minimal hardware.