A COMPARATIVE STUDY OF THE PERFORMANCE OF REAL-TIME INVERSE LIGHTING WITH MATTE, SEMI-GLOSS AND GLOSS SURFACES

By

SAPTAMI BISWAS

Bachelor of Technology in Computer Science and Engineering

West Bengal University of Technology

West Bengal, India

2018

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
July, 2020

A COMPARATIVE STUDY OF THE
PERFORMANCE OF REAL-TIME
INVERSE LIGHTING WITH MATTE,
SEMI-GLOSS AND GLOSS
SURFACES

Thesis Approved:

Dr. Blayne E. Mayfield

Thesis Adviser

Dr. Johnson P. Thomas

Dr. Christopher John Crick

Name: SAPTAMI BISWAS

Date of Degree: JULY, 2020

Title of Study: A COMPARATIVE STUDY OF THE PERFORMANCE OF REAL-TIME INVERSE LIGHTING WITH MATTE, SEMI-GLOSS AND GLOSS SURFACES

Major Field: COMPUTER SCIENCE

Abstract: Augmented Reality (AR) is the interactive process of introducing virtual objects or characters to real-world scenes. An effective way to increase the realism in AR is by mimicking real-world lighting conditions on the virtual objects. The process of gathering and analyzing real-world lighting information is called *inverse-lighting*. The surface textures of real-world objects may have different levels of glossiness. The goal of this research is to compare the effects that different glossiness levels have on the outcomes of the calculations. Several models of a regular dodecahedron were created using the Blender modeling software. These models were used to calculate and compare inverse-lighting on different levels of surface glossiness. Physical dodecahedrons also were created and used to see whether the Blender models accurately represent reality.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Augmented Reality (AR) is the process of introducing virtual objects or characters into real-world scenes. It has gained popularity from the masses with games such as *Pokémon Go* [1], *Temple Treasure Hunt* [2], *Harry Potter: Wizards Unite* [3] and so on. AR is not just restricted to games, it is also used as a tool for medical training of complex surgeries, classroom learning, and designing of architecture models.

AR is different from VR (Virtual Reality), which involves creating an entirely virtual environment for the user that is convincingly immersive. AR also is different from MR (Mixed Reality). Mixed Reality (MR) involves the creation of new environments and visualisations where physical and digital objects co-exist and interact in real time. To achieve the sense of immersion in AR, virtual object needs to blend well with objects in the real-world scenes. A prominent way to increase the realism in AR is by mimicking the real-world lighting conditions for the virtual objects. A common topic of research in the field of AR is *inverse-lighting*, which is the process of gathering real-world light information from images.

To achieve a solution to *inverse-lighting*, it is very important to have a good knowledge about the real-world environment. Some AR implementations do not have much foreknowledge, while others can provide detailed information of the environment where the process is to be carried out. The more that is known about the environment, the better the inverse-lighting results can be.

Often such cases have drawbacks such as expensive specialized equipment, the inability to analyze real-world lighting in real-time and so on. Our target is to do a comparative study to see if surface smoothness betters the real-time scene lighting evaluation.

CHAPTER II

LITERATURE REVIEW

2.1 Augmented Reality:

The process of superimposing virtual characters in real-world scenes is called Augmented Reality (AR). There are several ways to overlay virtual objects, including SLAM (Simultaneous Localization And Mapping), location-based, and the use of markers.

SLAM:

In the world of AR, SLAM is one of the most advanced technologies. To enable AR, it requires powerful hardware components. Moreover, to place virtual objects correctly, SLAM-based apps need the ability to map the real-world environment. Robots, autonomous vehicles, and drones use SLAM as a key driver. Point-based detection algorithms help these devices understand their surroundings.

Location-based:

A location or position-based AR app collects GPS data, mobile device built-in compass readings, accelerometer data, and gyroscope information to determine the position and orientation (also known as “pose”) of the device. Later, the app utilizes this information (such as textual location information or directional information) to place the virtual objects that fall within the field of view of the mobile device camera into real-world video. Such an AR app can help users to look for restaurants, gas stations, etc. situated in certain places.

Markers:

The most widespread method to perform AR is with the use of fiducial markers, commonly known simply as markers. They can be identified easily by software using computer-vision algorithms. When the camera of a mobile device detects a marker, it attaches a virtual object to the location of the marker. To achieve a unique pose of the virtual object, the markers must be asymmetric from every viewing angle.

Often, markers taken are in the form of 2D images, also called as 2D markers. Markers can also be presented as a 3D object. A simple way to create a 3D marker is by attaching 2D markers on each face of a 3D model such as a cube, cuboid and so on. An advantage of using 3D markers over 2D is to have more and better viewing angles. This serves as a great input to inverse-lighting. Light intensity information from each face can be used to improve real-world light direction estimation.

2.2 Inverse Rendering:

A primary objective in image processing research is to describe a scene in terms depth, shape, incident light, and reflection obtained from each visible surface point. These characteristics are called the *intrinsic* characteristics of a scene. Each intrinsic characteristic provides a valuable cue enabling better scene understanding. *Inverse Rendering* is the process of estimating these intrinsic scene characteristics from a photo, a set of photos, or a video.

A common field of research in inverse-rendering is *inverse-lighting.* Inverse-lighting is the process of estimating the real-world lighting conditions. Many researchers have contributed in the field of inverse-lighting. We will discuss a few of the approaches in the later sections.

2.3 OpenCV:

OpenCV (Open Source Computer Vision Library) [26] is an open source cross-platform

computer vision and machine learning library developed by Intel. This library contains programming functions that successfully solve computer vision problems in real time. OpenCV supports several programming languages, such as Python, Java, C++ and so on. This library is one of the most widely used packages for implementing video recognition, image recognition, motion detection, object recognition, and facial recognition applications.

2.4 Blender:

Blender [27] is a free and open source 3D computer modeling software. It is a cross-platform software and runs well on Linux, Windows and MacOS. Blender helps in creating 3D animations, 3D models, simulations, visual effects, motion graphics, product rendering, architectural visualization, game development, video editing and many more. These features make Blender much more advanced than many other 3D modeling software tools. Thus, Blender is referred as a 3D creation suite.

2.5 Other Related Work:

Kanbara and Yokoya [4] presented a novel, vision-based AR system to represent the attached and cast shadows on a virtual object in real time, where cast shadows are created due to light blockage coming from a light source by an object and attached shadows are surface patches away from the light source. The tracking system consists of a 2D square marker and a 3D mirrored ball placed at the center of the 2D marker. The 3D object is painted black to avoid dynamic range problem. The program first locates the 2D marker and then determines its pose (i.e. the position and orientation) by finding the position of the 3D mirrored ball. The relationship between the 2D square marker and the 3D mirrored ball is known to the system ahead of time.

The next step is to identify the position of the light source. For this, the camera detects the highlighted pixels. Once detected, the camera determines the angle of reflection of the highlighted points by taking the surface normal from the highlighted points with respect to its pose. The angle of reflection further helped to locate the light source that creates the highlight. The obtained information is stored in a light source map. Using this data, virtual objects are rendered with proper attached and cast shadows in real-world scenes.

Supan *et al.* [5] proposed a method to render appropriate shadows in an Augmented Reality application. This work included three different setups to test the approach. The first setup includes a single camera and a mirrored sphere such that both the real scene and the mirrored sphere are in the field view of the camera. The second setup includes two cameras and a mirrored sphere. One camera tracks the real scene, while the other tracks the mirrored sphere. The third setup consisted of a two-camera setup and a fisheye lens that captures the real environment directly.

After obtaining the images from one or two cameras (depending on the setup), the images are blurred to obtain the irradiance map. Further, this map is converted into a cube map that then is used to light while rendering virtual objects. The environmental maps are further

sampled to extract light information from each texture cell and set color to the texture cell. Shadows are cast using standard shadow mapping techniques. But rendering shadow maps each frame can be expensive. To avoid such a circumstance, they are updated in each frame. The system does not depend on any pre-processed data, which helps it to adjust to any change in light setting during runtime. However, this process increases the computational cost of the system.

Jensen *et al.* [6] developed a real-time image-based lighting system for outdoor AR. This method is sound in reciprocating the change in lighting conditions dynamically. To achieve this in real- time, a special environment called an *albedo* map stores the diffuse color of the surrounding environment along with a normal map to do so. To estimate the light, the scene is calibrated to a 3D model (environment map); later the user marks all visible diffuse surfaces. The sun intensity or *direct intensity* and *indirect intensity* are estimated from the light reflected by the surfaces in the scene. The proposed method requires many conditions to be considered; the use of outdoor scenes, the sun acting as the primary light source, diffuse surfaces in scene to be augmented, and a rough 3D model of the environment was required.

Later, Madsen *et al.* [7], extended his previous work [6] by describing an AR system that uses high range environment maps for representing the real-scene illumination. The novelty of this approach is rendering shadows created by virtual objects without disturbing the real shadows. They first obtain the environment map and determine the light sources with the *median-cut* algorithm. To cast shadows of the virtual objects, Madsen *et al.* model parts of the real-world scene in 3D. These 3D models occlude the virtual objects to achieve the virtual shadows.

Their work also addresses the *double-shadow* problem. To do this, they first consider the shadow appearance in the absence of irradiance. Using this information, they further identify the areas affected by the real-world shadows on the environment map. The real–world

irradiance is calculated using the 3D environment model simulations thereby obtaining the diffuse albedo of those surfaces. Finally, the virtual shadows are overlaid on real-world shadows using shadow mapping.

Ohta *et al.* [8], proposed an approach to enhance the genuineness of virtual shadows in a mixed-reality scene. The focus of their work is to simplify the light source models to generate convincing virtual shadows. This approach has a series of tests. The first test includes a proper relation between the size and arrangement of the light sources, and their impression on humans (volunteers) about the virtual shadows due to real light source is obtained. To do this, two cones, one a real object (Cone A) and the other a virtual object (Cone B), are used. Volunteers provide views on how closely matched the virtual cone shadow is to the real cone shadow in real-world lighting condition. The second test includes changes in the distance between two cones until a shadows obtained from both the cones is convincingly different.

The lighting conditions are captured using a fisheye lens. As a part of the test, AR images are rendered using six different AR light maps ranging from low to high resolution. Resolutions range from 8 x 8 pixels to 2,048 x 2,048 pixels. The four highest resolution maps represent identical solutions. They show more believable virtual light and shadow situations in a real-lit environment, unlike the lower resolution light-maps.

Aittala [9], developed a photorealistic rendering pipeline for augmented reality that responds to the real-world lighting conditions from the diffusions obtained from surfaces. The entire process is executed using a regular white ping pong ball and a 2D marker. The author gives two main reasons for choosing the ping pong ball. First, it has a known geometry and second, the matte surface of the ping pong ball. The ball being spherical makes the calculation of the surface normal simple and easy. The matte surface of the ball enables determining the direction of light falling on the ball much more easily.

In this approach, the system first looks for the marker. Once it was detected, a ping pong ball is used to calibrate the AR scene. The illumination on the ball surface is used further to

estimate the direction and intensity of the real-world lighting. For this process, the user either needs to show the regular ball, or rotate the marker for some number of seconds in front of the camera.

The approach of Noh and Sunar [10] generates soft shadows in an AR scene using a black reflective sphere placed over a 2D marker. Their approach is an extension to the work done by Kanbara *et al.* [3]. The initial stage of geometric registration executed by the former is same as the latter. The steps includes camera tracking the 2D marker, thereby determining the pose of the ball. Once that is complete, the information is utilized to create an environment map. Then it estimates the real-world light using the *median-cut algorithm* to create the virtual shadows.

Their contribution in extending the work of Kanbara *et al.* [3] was done by creating soft shadows. This is done by overlapping multiple hard shadows and then a soft shadow is created using the varying opacity information of each of the hard shadows. This method is named the *Heckbert and Herf* method. Though this method produces convincing soft shadows, it has a major drawback. High-quality soft shadows lead to a degradation of the system performance.

Pessoa *et al.* [11] proposed a few approaches to increase the realism of augmented reality. Briefly, they use a high-dynamic range environment map of the real scene to extract lighting information and render convincing virtual objects. The program starts by capturing images of the surrounding environment, further creating the environment map. Their approach to create the environment map was novel as compared to previous approaches. For each virtual object, four environment maps are calculated in every frame. An advantage of this was that it ensured the presence of all other virtual objects and phantom objects in the scene.

Once the first step is accomplished, the program performs a pre-filtering of the environmental illumination. This process is computationally expensive. Thus, to avoid this

expense, computations often are executed offline. Finally, with all the results obtained, virtual objects are rendered onto the real scene.

In 2010, Jensen *et al.* [12] proposed a method which started as an extension to Kanbara *et al.* [4] and Supan *et al.* [5]. Jensen *et al.* use a reflected-sphere marker to collect light information, thereby rendering convincing shadows of virtual objects. The marker used in their work includes a standard 2D *ARToolKit* marker with a black glossy ping pong ball placed at the center of the 2D marker. The specular properties of the ball are used to gather real-world lighting conditions using median-cut algorithm. Surface normal from those specular or highlighted points are taken to project light onto the environment map.

Next, the authors worked on rendering perceptually-correct shadows on virtual objects using the results obtained from previous collected data. To make the shadows convincing, they layered multiple shadows and applied various visual effects such as blurring to the shadows. To analyze the performance, the authors arranged an online survey. The results obtained from this survey were bit unexpected as the unaugmented control images received only 60% of positive views. From the survey results, the authors concluded a minimum of 64 overlapping of shadows were required to achieve a convincing shadow.

Jiang *et al*. [13] developed an algorithm to detect shadows for single images. In this approach, the program first performs a color segmentation algorithm, and then creates illumination environment maps for multiple light frequencies (high or low) and for different surface orientation. The color segmentation and illumination map are used to construct a shadow edge map. Once the shadow edge map is created, it goes through a couple of filtering steps. Once all the filtering steps are complete, the algorithm labels each pixel either “shadow” or “not shadow”.

To reduce the computational time, the algorithm looks only for color segment edges to find the shadow edges. The success of this is ensured by using three color spaces instead of one color space.

Chen, Wang and Jin [14] estimated illumination of a single image for lighting virtual objects. Their approach includes no prior knowledge of the 3D geometry or lighting information of the scene to light the virtual objects. The program first takes the input image to determine its 3D geometry of the image. To do this, there are two existing algorithms that were created using a common model called Markov Random Field (MRF) model. One is based on edge detection and the other on linear regression. They use one of the above algorithms to get the 3D geometry. Immediately after this, the program performs a reduction on the obtained model to get the intrinsic components (the shadow component and the reflective component). With all the above information, i.e. 3D geometry and intrinsic components, the program develops a *sparse radiance map*. The major contribution of this work is the sparse radiance map. This map contains *M* light sources placed at regular intervals inside a hemisphere. This map mimics real scene lighting conditions. So, each light source in the map has different light intensity. Thus, with all the information, the virtual object finally is rendered to the scene.

Gruber *et al.* [15] proposed a method to estimate environment lighting in real-time without

Fig-1: Adaptive radiance transfer for probeless light estimation in AR [15]

any *probe* such as reflective ball. A praiseworthy achievement of this approach is the

adaptive radiance transfer in real time. As shown in Figure-1, the system is designed with the ability to cast shadow on a virtual object if the user puts his or her hand between the real-world light and the virtual object. To implement this approach, a depth-sensor camera (Xbox Kinect) is required, such that both the scene geometry data and the visual information can be acquired. The process begins by creating an image of the real-world scene. This image then is used for extracting the color information, further comparing it with the real-world scene to determine the light positions of the scene. This is done using simple Lambertian reflection calculations. The use of a depth sensing camera plays a major part in this approach. Occlusion of the virtual objects due to real-world objects is determined using the depth information provided by the camera.

Kamboj and Liu [16] use the variance cut algorithm instead of medium cut algoithm for estimating real light sources in augmented reality. The process takes an input image through a fish-eye lens. The fish-eye camera is oriented upwards to help in obtaining a 180° environment map. The program then identifies the light sources from the environment map by implementing the variance cut algorithm. Using *MATLAB*, a script is generated to show the light directions. In the virtual environment, virtual light sources are placed in the estimated direction of real light sources using software called *3DS MAX* (a product somewhat similar to Blender). With the estimated light source positions, virtual objects are rendered onto the scene.

Michiels *et al.* [17] proposed a method based on omni-directional video. In this approach, the authors use a custom designed, omni-directional camera that was mounted on the top of a car. The custom camera was built with six cameras, each placed at an interval of 60° such that two adjacent cameras have an view overlap of 50%. This was designed to capture the image from all directions. The identification of the real scene lights is different than the previous approaches. The shadows in the surrounding environment are studied to estimate the direction of the light source. Virtual objects are made visible using voxel cone-tracing.

The alignment of the virtual objects is done using offline feature tracking, depending on the car being in motion. Feature tracking is done on the input from each camera and finally the results re filtered using bundle adjustment to finally obtain the car trajectory.

Rohmer *et al.* [18] developed a real-time differential illumination method for illuminating virtual objects on mobile devices consistently. This method is restricted only to indoor scenarios. The environment map is captured in real-time using HDR cameras fitted with fish-eye lenses. The approach of acquiring the map is different than the previous processes, which require the environment map to be captured beforehand or acquired immediately after the program begins to process.

Most of the data processing is done on the mobile device itself to prevent delays. Some of the operations are executed using a PC, such as the processing of environment data captured by the cameras. The PC interprets the captured environment data and also performs the radiance-transfer calculations. The results are sent to one or more mobile devices. The scene geometry models are created ahead of time. The PC projects surrounding environment radiance onto the model and the obtained results are stored in a *radiance atlas*. This map keeps track of both direct and indirect radiance. The direct radiance mimicks the colored point-light information inserted in the scene and indirect radiance mimics the pre-calculated radiance transfer function.

Alhajhamad *et al*. [19] proposed an algorithm for fetching information about multiple light sources from images of indoor scenes. Figure-2 shows an illustration of the setup. The color images are converted from RGB to HSV (Hue, Saturation, Value) and finally converted to a black-white image using Otsu's threshold algorithm. This result is filtered further using Gaussian blur for smoother approximations. This filtered image goes through contour detection to find the shadowed regions. Light direction is estimated by comparing the center of mass of the real scene light and image shadow to obtain a vector. Light intensity is

determined by comparing the angles between the shadows. To determine which light caused the shadow, the nearest light is considered as the source of the shadow.

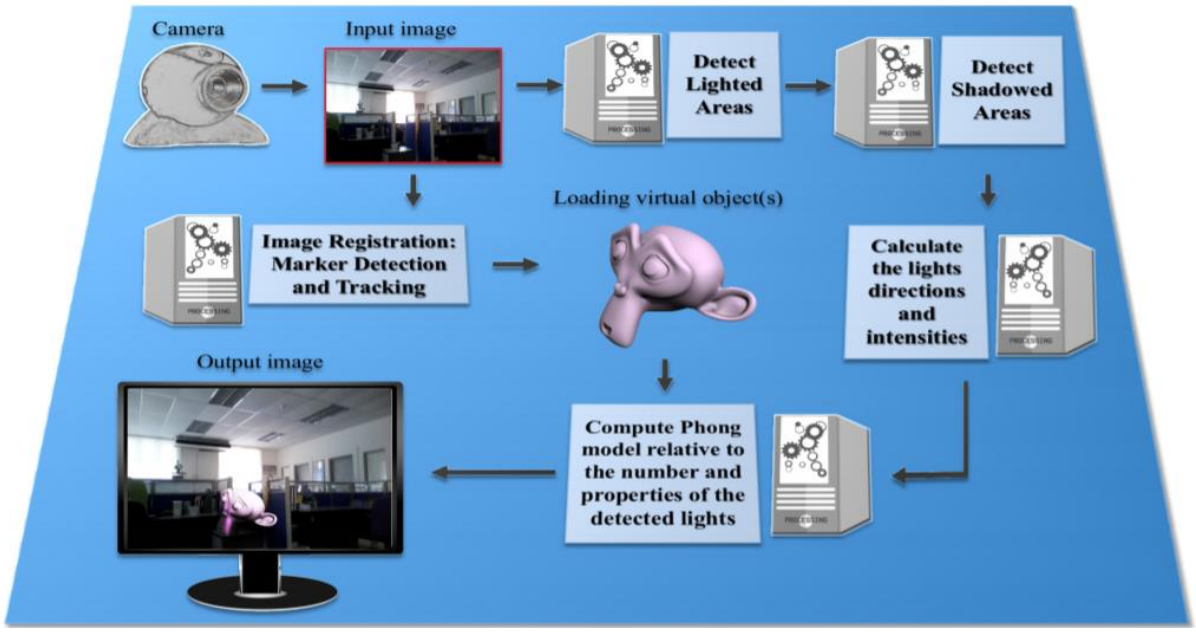

Fig-2: The pipeline of the illumination sources detection model. [19]

This algorithm is computationally effficient. It maintains above 40 frames per second throughput. This method also is dynamic in estimating shadows with changing scene lights. This approach has many exemplary achievements based on a major constraint or assumption – a room of specific size – to achieve all results.

In 2016, Soulier *et al.* [20] used a low-cost light sensor to estimate the illumination direction for augmented reality. In this method, the author's use eight light-dependent resistors (LDRs). Figure-3 shows the arrangement of the LDRs, placed at an intervals of 45° along the greatest diameter of a sphere and one additional LDR on the top of one hemisphere. Each LDR is connected to an Arduino microcontroller. This entire system is connected to a PC that computes the light data, further feeding the results to a mobile device that is used to control the AR program. The AR app was designed using the *Unity* game engine and the Vuforia plugin for Unity.

Fig-3: Low-cost sensor [20]

Kasper *et al.* [21] proposed a method to estimate the light sources in a real-world scene using path-tracing. The process first generates a 3D virtual geometry of the environment using depth camera and triangulation algorithm. The albedo with respective vertex of the mesh is required to render the RGB image of light estimation. An environment map is used to establish scene lighting conditions. Light sampling is performed using ray tracing and finally, the weights of per pixel sampled lights are adjusted using Monte Carlo filtering.

Glen Straughn [22] used a dodecahedron marker (DM) to determine the light direction of the primary light source. The obtained results are used further to achieve real-time inverse lighting for augmented reality. In this approach, light sample results of each visible face of the DM are used to determine if the face is exposed to the real light source or in shadow. Determination of the face being exposed to light, or not, was done by taking its surface normal vector. If the face is found to be in shadow, then the surface normal of the face is reversed. An average of all the surface normal vectors was used to determine the direction of the primary light source in the real environment. Figure-4 shows the setup used for in this experiment.

The research makes an assumption of a single light source in the real-world scene for simplicity of testing. This work was conducted using a DM covered with matte material.

Jiddi, Robert and Marchand [23] proposed a method to estimate the 3D position and intensity of multiple light sources. This method required no light probe or user interaction to estimate light intensity and cast shadows. The process took the following as inputs: a) a rough 3D model of the environment captured by an *Intel R200 sensor*, b) simulation of ambient lighting and finally, c) color pictures of the scene that were used to detect the real light in the scene.

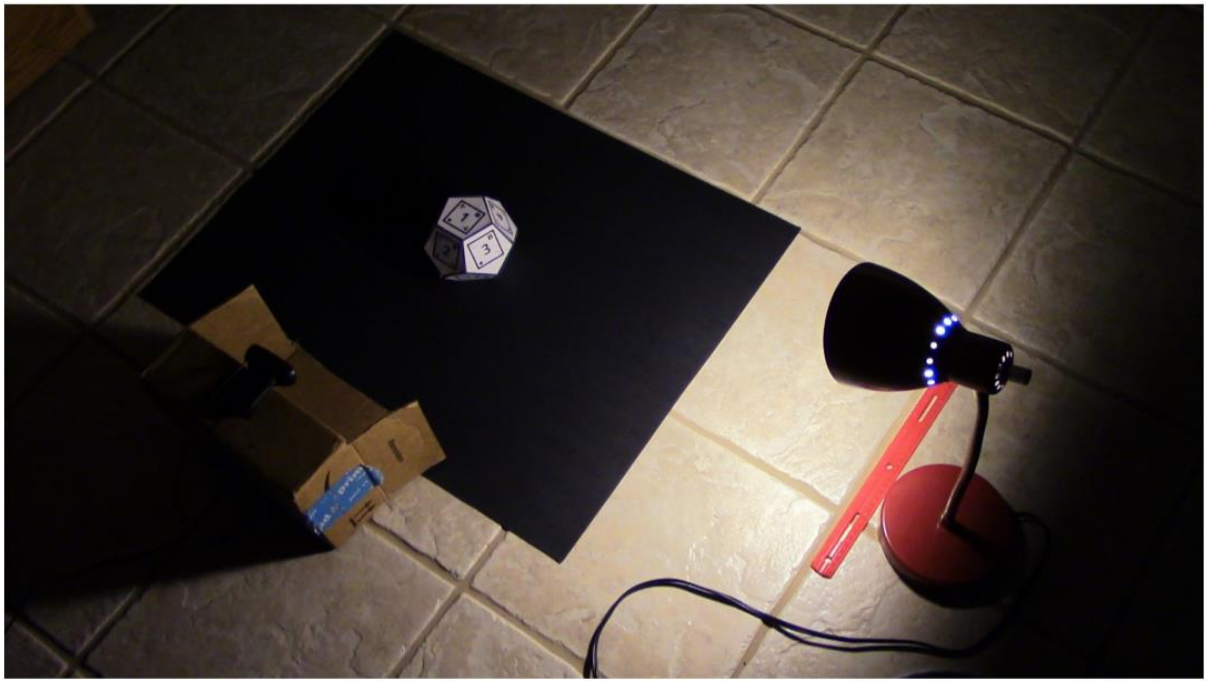

Fig-4: Straughn's experimental setup. Web camera mounted on the top of the cardboard box. [22]

Straughn made a couple of assumptions: a static scene geometry that contains a primary planar surface onto which shadows are cast, and scene reflectance described by the Lambertian reflection model.

This method first computes the scene geometry (this is done once in the entire computation). Then the texture/albedo is separated from the illumination in the current frame. After this, the program proceeds to detect the position of the 3D position of the light sources. A subset of point lights is extracted, provided that the shadow map and estimated *illumination ratio map*

are same for each frame. Finally, the intensity of the light sources are estimated using the *Phong reflection model*.

Schwandt, Kunert and Broll [24] studied the glossy reflections in an AR/MR environment to achieve realistic reflections in real-time on a mobile device. The team had performed a prior work to identify light sources from all directions (360°). This approach combines the camera image obtained in each frame to construct the environment map. Each camera image is reduced to a cube map. This cube map is used as the environment illumination of the scene.

A drawback of the above discussed method is the amount of transparency in the reflections. In their paper, the team addresses a way to mend it with a process known as *stitching*. The existing images inside the cube map are combined with the current image stream. This helps to achieve a more realistic environmental map. Depending on all the evaluated values, virtual objects are rendered to the real-world scenes.

In 2019, Alhakamy and Tuceryan [25] presented a three step method to extract the physical illumination to render virtual objects to the scene in real-time. The system begins by capturing a 360° live camera video with an AR device such as a head-mounted display, phone or web cam. This video serves as an input to the first phase of the entire process. The direction and angle of incidence of the incoming light is estimated from the input video. The second phase includes extracting information of the reflected light or *indirect illumination*. The 2D texture of the region over which the virtual element to be projected is captured using the image-based lighting model.

The result obtained from the second phase is used further for the processing of the third phase. Shader language and the CG technique are used to estimate the light and shadow conditions on the virtual objects to be rendered. Both GLSL (Open GL Shader Language) and the Cg programming language were used to help identify numerous features of a virtual object before and during rendering. To obtain complete shading information, both direct and indirect influences on illumination are required. These were achieved using surface shaders

and unlit shaders, respectively, on the virtual objects. Though this approach is convincing to make the entire system interactive, it has the incapability to differentiate between white area and the light reflected areas from the light-sources.

CHAPTER III

PROPOSED METHOD

3.1 Problem statement review:

As discussed in Section 2.1, augmented reality (AR) is the process of introducing virtual objects into an image or video feed of a real-world environment. One of most common research challenges in AR concerns introducing virtual objects seamlessly into real-world scenes. To achieve this, virtual objects must have enough realism so that they blend well into the real-world scene. Immersion experience in AR can be increased by enhancing the real-world environment lighting conditions over virtual objects. This process of gathering real-world lighting conditions is called *inverse-lighting* (IL).

As described in section 2.5, Straughn [22] used a dodecahedron marker (DM) covered with 12-unique 2D matte markers, one on each face of the DM. The light sampling information from each visible face of the DM was used to determine the amount of light and shadow on the visible faces. An average of the normal light vectors from the faces was used to calculate a vector indicating the direction toward the primary light source in the real environment.

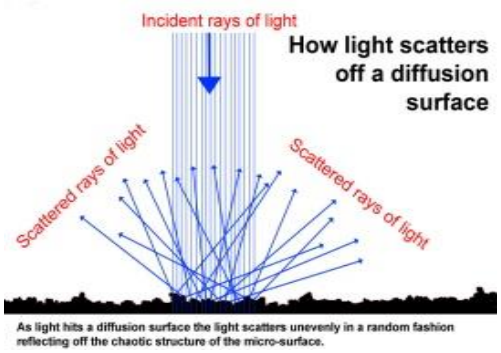

Fig-5: Reflection of a diffused surface

This method had great efficiency in speed, ease of use, and the ability to work in real-time; but it achieved a low accuracy in estimating light intensity. A possible reason, apart from the variability in marker detection and tracking, could be the use of matte material for the

markers. Light rays on a matte surface suffer diffusion (Fig 5), which might lead to loss of intensity of the incident rays. Introduction of glossier material markers might help to better the performance. The factors affecting the intensity of light reflected from a glossy surface are light absorption of the surface material and the angles of incidence and reflection.

3.2 Overview of the proposed method:

A promising solution to problems encountered in Straughn's method could be the use of glossy surface instead of matte materials for the markers. In this research, we analyse whether the accuracy gets enhanced on introducing gloss surfaces. The proposed method comprises three primary stages: identifying visible faces of the dodecahedron marker (DM), luminance sampling, and estimation of direction of the real-world light source. This process assumes a single light source. The results were tested on both virtual and physical DMs. The virtual DMs was created using *Blender* [27] while the physical DMs were created by painting the faces of a DM using gloss paints thereby testing for different levels of glossiness: gloss, semi-gloss and matte.

During the first stage of DM face identification, an image (both for virtual and physical DM respectively) is obtained from the camera and the position and orientation (also known as pose) of the DM is determined. This step is followed by luminance sampling and light direction estimation stages.

In the luminance sampling stage, the mean RGB value of each face is evaluated to find the best lit DM face. The RGB color model has three color parameters: Red, Green, and Blue. The each parameter (red, green, and blue) defines the intensity of a color as an integer between 0 and 255. When each parameter of RGB has at its maximum integer value of 255, i.e. rgb(255, 255, 255), the color obtained is pure white. Similarly, for a perfect black, each parameter of RGB must have the lowest integer value of 0, i.e. rgb(0, 0, 0). So, to determine the best lit face, the average RGB of each face is evaluated and ranked from highest to lowest RGB values. In the later sections, the process has been discussed in detail.

3.3 Dodecahedron Marker and Face Detection:

3.3.1 Designing of the virtual and physical DM:

In this research, a 3D dodecahedron model was used to determine the lighting of real-world scenes. As discussed above, the virtual dodecahedron was made using *Blender*. In the newer versions of Blender (2.75+ versions) the regular solids (such as cube and sphere) have been extended to permit the creation of higher polyhedrons (tetrahedron and above) with ease. The glossiness of the dodecahedron is controlled using the *Glossy BSDF node*. The maximum surface glossiness was obtained with roughness parameter of 0.05 and minimum glossiness with 1.

For the glossy, physical DMs, card stock paper faces were painted with gloss paints and attached to each face of the DM; the paints used were of different level of glossiness: semi-gloss and gloss. For a matte surface, colored card stock paper was used to make the faces.

3.3.2 Reasons for choosing dodecahedron over other 3D models as a marker:

In section 2.1, different ways to overlay virtual objects in real-world scenes are discussed briefly. The most common approach to perform AR is with the use of markers. In this research, a 3D marker (dodecahedron marker) was used because of several reasons. 3D markers can be viewed from any angle unlike 2D markers. Moreover, a 3D marker can provide information about the real-world lighting conditions from the direction it is facing. In this research, the marker was required to serve as a light probe. So, 3D marker was chosen over 2D marker because of the above stated reasons.

Now, a dodecahedron was chosen over most simple 3D models like a cube or cuboid because of a hitch. Standard 3D models of cube or cuboid can be viewed in such a way that only one face is visible at a time. Thus, both polyhedrons disqualify the above mentioned advantage of 3D models. A regular polyhedron with greater number of faces was a better solution. A regular dodecahedron is a polyhedron composed of 12 regular pentagonal faces.

Thus, dodecahedron was chosen as it qualifies the advantage of 3D marker and easier geometric pentagonal face.

3.3.3 Determining pose and faces:

To detect the position and orientation (pose) of the DM, Straughn's method used a marker identification to find the 2D markers printed on the faces of the DM. Once any face marker was identified, the pose if the DM relative to the camera could be determined using the AR software. In our method, the DM does not contain any 2D markers attached to the faces. Thus, to determine the visible faces of the DM, the program first takes the image of the DM. For the virtual DM, a high quality image was obtained by setting the rendering samples at 500 in blender and then performing a screen capture from the computer. Images of the physical DM were taken with a OnePlus 7 Pro mobile camera.

The next target is to detect the edges of the RGB image which was done using *Sobel operator* (also known as Sobel-Feldman operator). Sobel operator is a differentiator operator. It computes an approximation of the gradient of an image. It marks the gradient changes along the horizontal (G_x) and vertical (G_y) direction, where G_x, G_y can be represented by 3 X 3 kernels as follows:

$$G_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * Original\ Image$$

and

$$G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * Original\ Image \qquad (3.1)$$

and the common value of G is given by:

$$G = \sqrt{{G_x}^2 + {G_y}^2} \qquad (3.2)$$

$$=> G = |G_x| + |G_y| \qquad (3.3)$$

The obtained RGB image is converted to a grayscale image and passed through the laplacian operator, which helps finding the derivative of the image thereby giving the high

and low intensity of the image. In other words, it gives the gradient of the image. This gradient value is further differentiated with respect to the horizontal and vertical axis to give the G_x and G_y values. The G_x and G_y were combined together to achieve sharp edges**.** Fig 6 shows determined edges for respective DMs.

The determination of the edges is the basis for identifying the faces of the DM. For this, all possible shapes that can be obtained from the DM edge image are identified. This was achieved by drawing contours on the edge image. This step is performed to get all vertices of the DM. Instead of finding the contours of the RGB image straight away, the additional step of edge detection is performed to ensure that only the edges contributing to the pentagonal face is processed during this step.

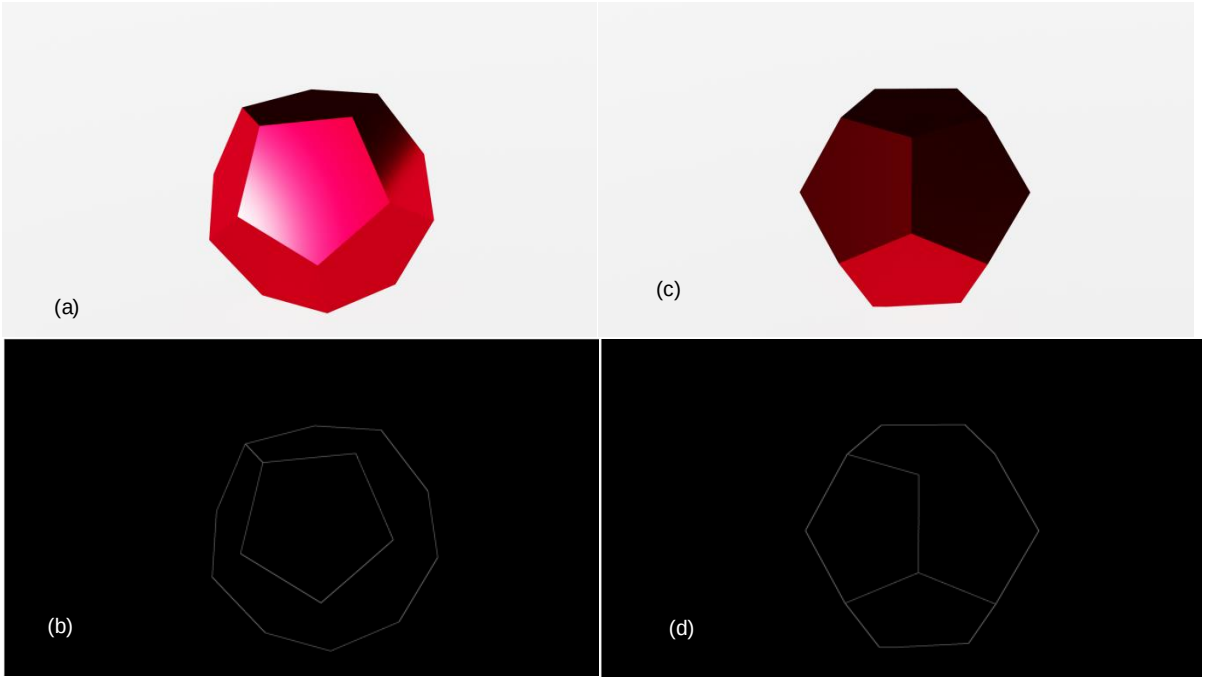

Fig-6: (a), (c): Virtual DMs placed with different pose.
(b), (c): Determined edges of the respective DMs.

The above mentioned process was tested on many DM poses. Notably, this method could successfully identify the extreme outer polygon and at least one pentagon. For instance, for the DM pose as shown in Fig 6(a), the program could identify of the extreme outer polygon and the central pentagon. While for the DM pose as shown in Fig 6(c), the program could identify the outer polygon and the two of the pentagons. This was implemented with OpenCV

library function *approxPolyDP*, which draws contours on the image to identify their geometric shape. This function returns the coordinates of the vertices of the identified polygons. In any polyhedron, each face shares its edge with the neighbouring faces. So, with the vertex information of the pentagon(s), the other visible pentagons can be identified easily. Once the DM faces has been identified, the program performs luminance sampling.

3.3.4 Luminance Sampling:

After the information of each face has been yielded from the previous step, the program determines the brightest face by evaluating its mean RGB value. This step would be essential to DMs whose brightest face has a consistent RGB value throughout. Considering merely the mean color, might not provide sufficient information to estimate the direction of the real-world light source for a DM pose in which the face contains uneven color distribution (as shown in Fig 6(a)). Thus, the aforementioned step is extended to find the mean color around the corners and the centre of that face. Fig 7 shows the division of the brightest face along the vertices (corners) and the centre.

To achieve RGB information around the corners, triangles were chosen for the ease of calculation, while for the centre – circles. To find the vertices of the corner triangles respectively of the brightest face, a vertex was chosen as pivot and an equidistant point was evaluated on the line connecting the neighbouring points. A point along a line at a certain distance can be found by:

$$(x_t, y_t) = (((1-t)x_0 + tx_1), ((1-t)y_0 + ty_1)) \qquad (3.4)$$

where, (x_0, y_0) : start point

(x_1, y_1) : end point

(x_t, y_t) : point at distance d_t

$t = d_t/d$

$d = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2}$: distance between start and end point

This step was repeated unless all the corners of the pentagonal face has been identified. The RGB value of each corner triangle and the centre is evaluated and surface normal from the brightest section is taken. This normal gives the estimated direction of light source,

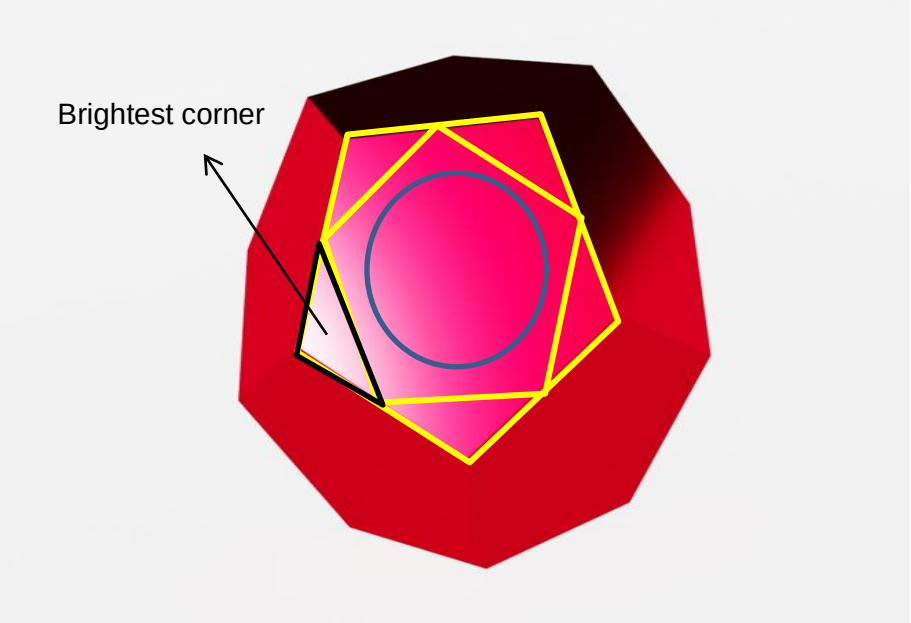

Fig-7: Dividing the faces into sections of the DM face with greatest RGB value

3.3.5 Estimate light direction:

The final step of our approach is determining the real-world light source direction. The evaluation of the light vector is dependent on one of the following discussed cases:

Case – 1: *More than one bright faces*

When there are more than one visible bright faces with same (or approximately same) RGB mean values, than an average of the surface normal from each face is evaluated to achieve the final light vector.

Case – 2: *Significantly bright face amongst all visible faces*

When there is exactly one face with the highest RGB value, the corners and the centre of the face is reconsidered to find the brightest sections (discussed in section 3.2.3). The surface normal is calculated from the highest RGB value section.

Case – 3: *Average RGB is close to color black*

When the average face mean color is close to darker region, then the surface vector is rotated at a 180°. This case would happen when the incident light is shining behind the DM such that the camera can capture the faces in shadow.

3.3.5.1 Calculation of surface normal:

In this research, the polygons that are considered include triangle, circle and pentagon. If the program found the situation as mentioned in case-2 has been satisfied, it further checked whether the section is a triangle. This was easily implemented as the vertices of the triangles were stored in an array. The next step was to find the surface normal of the triangle. The cross product of two sides of the triangle gives the surface normal [29]. So, if vector $V = P_2 - P_1$, vector $W = P_3 - P_1$, and surface normal vector N would be:

$$N_x = (V_y * W_z) - (V_z * W_y), \quad (3.5)$$

$$N_y = (V_z * W_x) - (V_x * W_z), \quad (3.6)$$

$$N_z = (V_x * W_y) - (V_y * W_x) \quad (3.7)$$

such that $(N_x)^2 + (N_y)^2 + (N_z)^2 = 1$ (3.8)

For the brightest section being about the centre of the pentagon (circular section), the program computes surface normal for the inner pentagon, that is formed by connecting the mid-points of the pentagonal face. This was done as computing surface normal of any polygon is simpler compared to circle.

Surface normal for any polygon can be computed by:

$normal(v_0, v_1, ..., v_{n-1}) = \sum v_i \times v_{i+1}$, where: v_i with i = {0, 1, 2, …, n-1} (3.9)

The surface normal information gives the estimated direction of the real-world light source.

CHAPTER IV

RESEARCH METHODOLOGY

4.1 Implementation:

The first step is designing the algorithm that would perform light estimation using the technique discussed in Chapter 3: *Proposed Method*.

The code was written in Python, using Python 3.8.5 specifications. In this research, PyCharm 2020.1.2 IDE was used to compile and debug the code. The code used several libraries for achieving the steps discussed in Chapter 3.

The tasks like edge detection, shape detection, mean color value estimation was performed using OpenCV [26]. OpenCV is an open-source cross-platform computer vision and machine learning library. The plot of the detected edges was plot with *matplotlib*. Matplotlib is the plotting library for the Python programming language. This research includes many large mathematical operations on arrays. Performing each of these calculations would be tedious. Thus, for these critical operations, the *NumPy* library of the Python programming language was used.

4.2 Experimental Design:

In this research, we designed a virtual DM and a physical DM. The inclusion of the physical DM was to verify whether virtual models created using any 3D modeling software could mimic the nature of a real or physical model. Thus, the virtual model was created using 3D modeling software, *Blender*. In section 3.2.1, a detailed discussion on designing for both the DMs has been illustrated in detail. In this section, we shall discuss the setup for both the

DMs (Fig 8(a)). In both the setup, we considered for a single light. The physical DM was placed 38 cm away from the light source. The picture of the setup and the DMs was taken using by mobile phone camera (OnePlus 7 Pro).

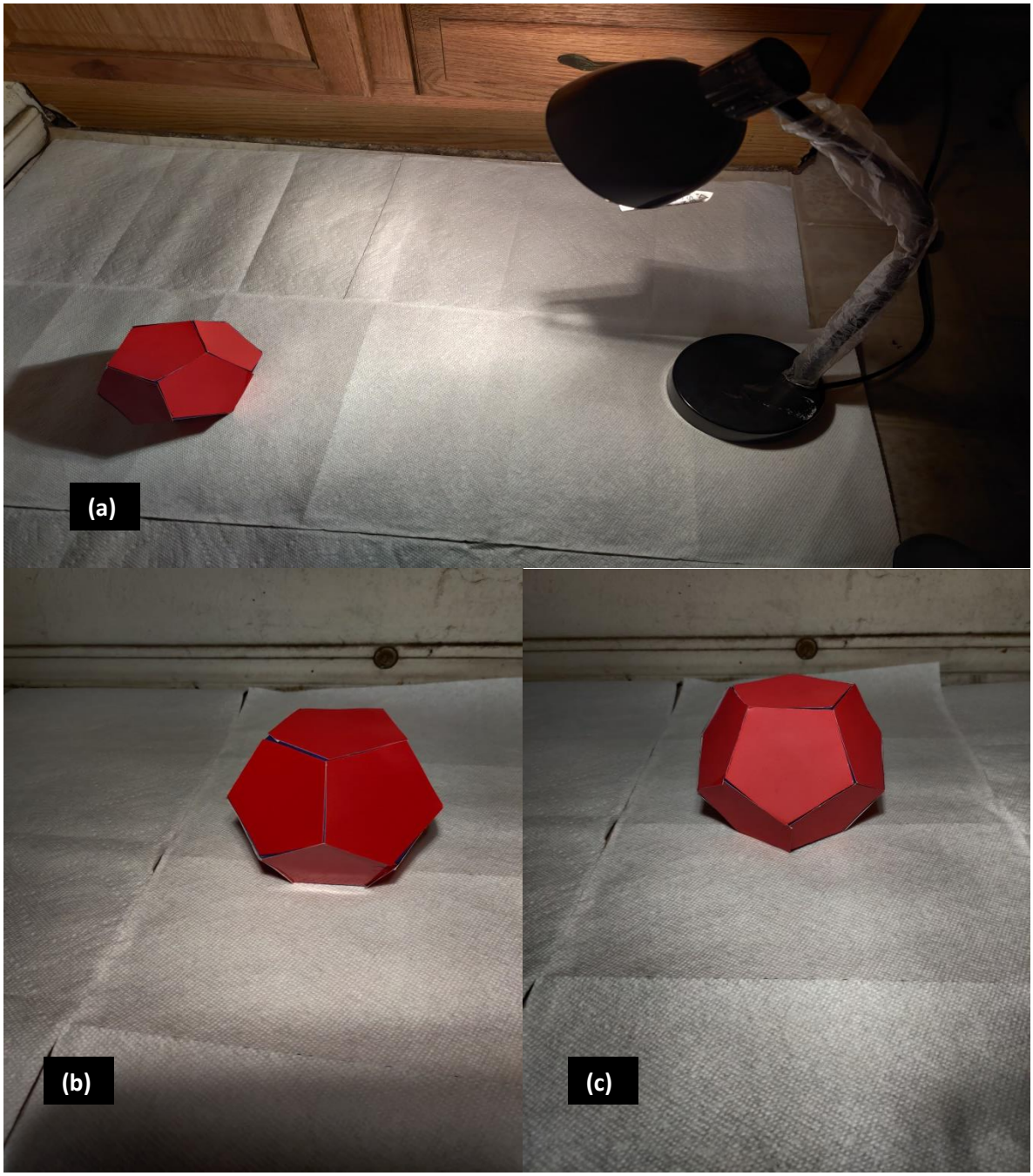

Fig 8: (a) – Shows the entire setup of the physical DM; (b), (c): Gloss and matte DMs respectively

The virtual DM was created for using math function for regular solids in Blender. The virtual model was tested for several angles to understand the performance of the gloss, semi-gloss

and matte surfaces. The roughness of the DM surface was adjusted accordingly to match the gloss, semi-gloss and matte surface. For the gloss surface, the roughness level was set to 0.1, for the semi-gloss surface – roughness = 0.5 and for the matte surface the roughness level was at 1.0. Fig 9 shows the DMs with different surface roughness. These DMs is rotated at 40° relative to the initial position.

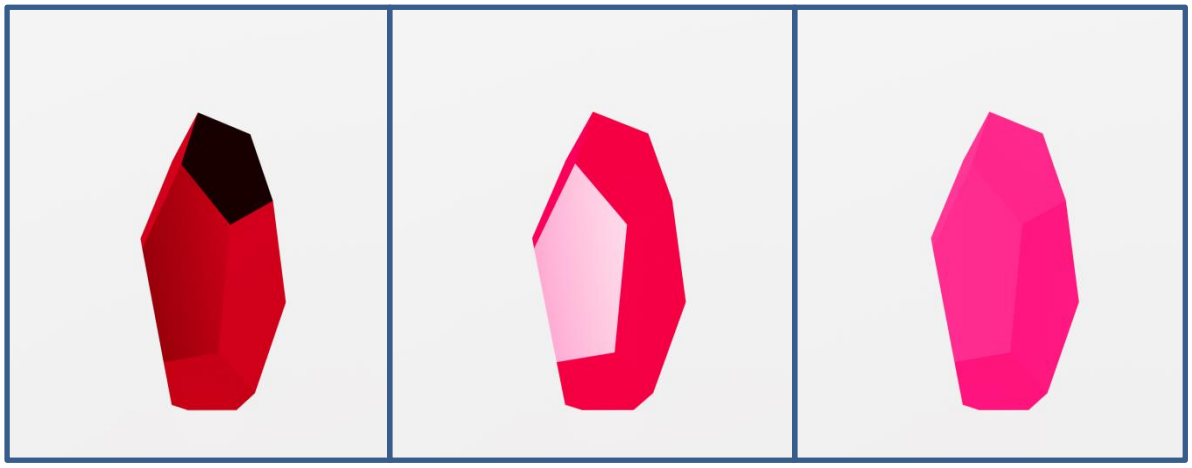

Fig 9: Virtual DMs of different surface roughness. Gloss (roughness = 0.1) (left); Semi-gloss (roughness = 0.5) (center); Matte (roughness = 1) (right). The DMs have been rotated at 40° from the initial position such that it facing the light directly relative to the camera.

The accuracy of estimation of light direction was determined by the angle of incident light to the surface of the DM and the light direction vector estimated by the proposed method. Fig 9 shows the percentage of error due for every 20° rotation made to the object with respect to previous pose along the horizontal axis.

4.3 Testing for the accuracy of the method:

The goal of this research is to estimate the light source direction. Hence, the accuracy of this method would be to find how the estimate direction was to the real light source. For this, the angle between the real light source vector and vector for the estimate direction of light source is calculated. For the virtual DMs, vector for the real light source could be found easily. In the recent versions of blender, a node property called *Lamp Data Node* has been added. This property helps in obtaining all information related to a specified lamp object [28].

It provides with information such as color of the lamp, light vector – a unit vector in the direction of the lamp and the shading point, distance - for obtaining the distance between lamp and shading point, to name a few. Thus, solving both the vector – gives the required angle difference between the two vectors. Lesser the angle difference between the two vectors, lesser the percentage of error and vice–versa. Further, these angle information is plotted (Fig 10) in a graph.

CHAPTER V

OBSERVATIONS AND ANALYSIS

5.1 Notable Issues Encountered During Research:

The virtual dodecahedron marker (DM) was designed using 3D modeling software *Blender*. These models were extremely flexible so perform actions like surface roughness and rotations of the DM relative to the camera and so on. This was not the situation with the physical DM. The images of the physical DM irrespective of having different roughness level looked almost the same. The key factor of this method was to do with RGB values. In theory, with a light shining brightly on the surface of the DM makes the color on the region exposed to light would have lighter color compared to the area that is not exposed to light. Since, the incident light could not bring any possible change to the appearance of the image, physical DM did not respond to the system like the virtual DMs did.

5.2 Light Estimation Results:

The graph in Fig 10, shows the light estimation achieved by different surface roughness for different angles. In the following graph, the angle at 0° indicates the DM in its original position. The angles in negative indicate the conditions when the light source is facing the DM relative to the camera. Contrarily, positive angles indicate for the conditions in which the

Finally, after all the estimation is complete, this information is applied on to a virtual object to see its performance (as shown in Fig 11 and Fig 12).

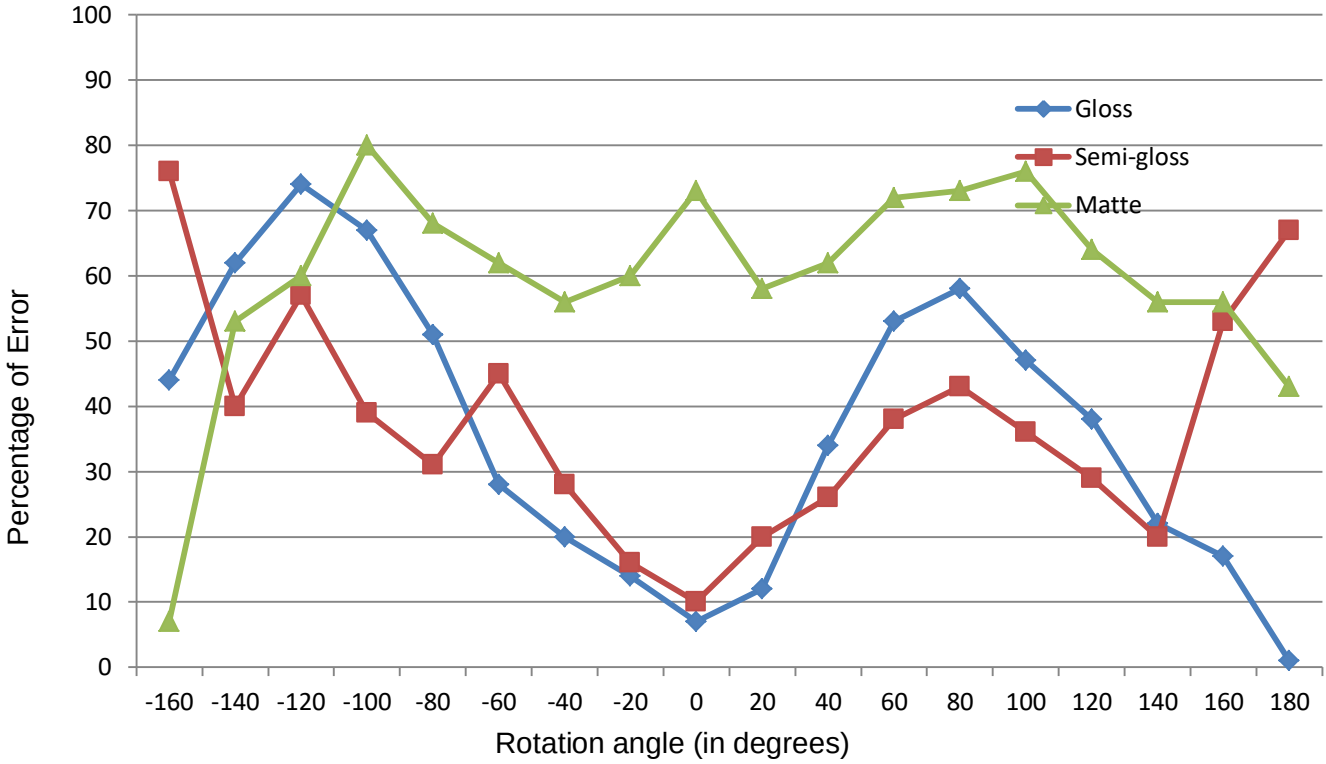

Fig 10: The above graph shows the percentage error due to change of the pose of the virtual DM for every 20°. The negative degree indicates that the light source facing the DM while positive degree indicates that the light source is behind the DM relative to the camera pose.

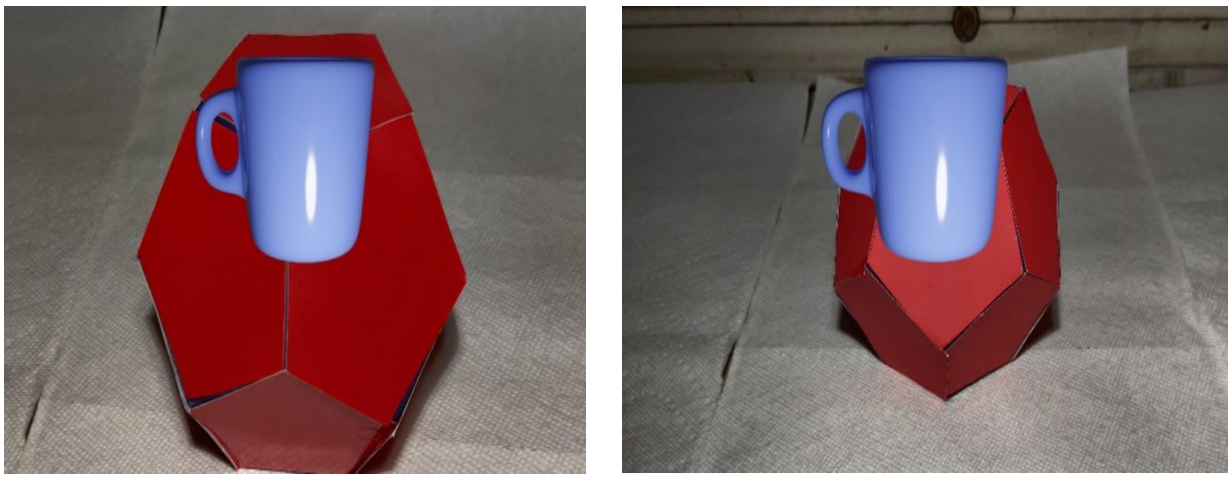

Fig 11: Inverse lighting effects on the virtual objects on a physical DM

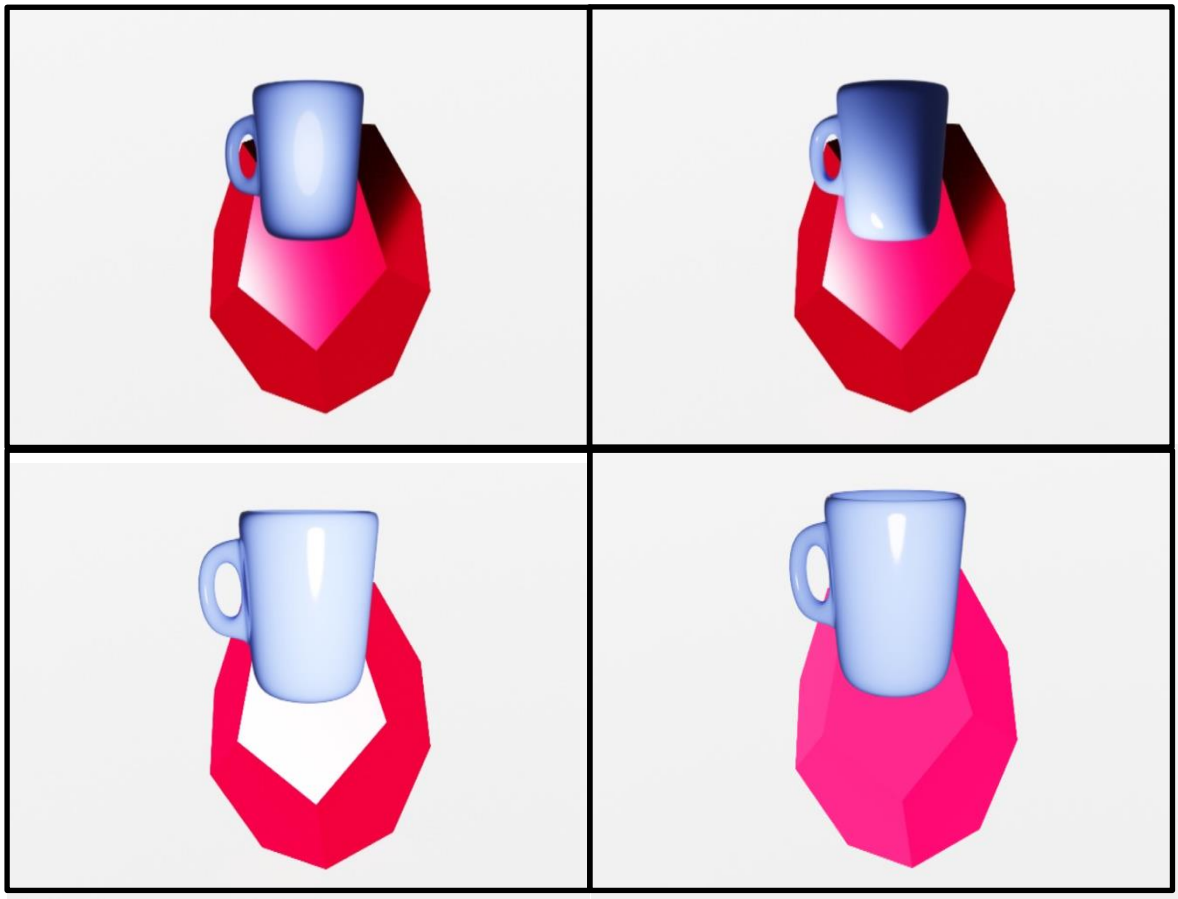

Fig 12: Inverse-lighting effects on virtual objects using virtual DM, where: (a) shows the virtual object without applying the inverse-lighting algorithm, (b) shows the effect of the IL due to a gloss DM, (c) IL on semi-gloss DM and (d) shows IL effects due to complete matte DM.

CHAPTER VI

CONCLUSION AND FUTURE WORK

In this research, the goal was to compare the performance of inverse-lighting effects for different surface: matte, semi-gloss, and gloss. Also for the testing of the viability of the model, we used both the physical and virtual DM. Firstly, from the above results we can conclude that gloss produces more impressive results for inverse-lighting than the surfaces of higher roughness. This is because in theory, gloss allows incident light to reflect back at the same angle. The test results for other surface roughness also support for the same. From Fig 10, we can clearly see that with the increase in surface roughness the average percentage of error goes high. Therefore, gloss surfaces would help in getting better results for inverse-lighting. Secondly, Blender is not efficient in creating virtual that can mimic physical models enough convincingly. In future, this research can be extended to develop virtual model that would actually mimic real models.

REFERENCES

[1] *Pokémon Go*. Niantic, 2016.

[2] *ARrrrgh.* Warpin Media, 2017.

[3] *Harry Potter: Wizards Unite.* Niantic, Warner Bros. Interactive Entertainment, WB Games, Inc., 2019.

[4] M. Kanbara and N. Yokoya, "*Real-time Estimation of Light Source Environment for Photorealistic Augmented Reality*." IEEE, 2004.

[5] P. Supan, I. Stuppacher, and Michael Haller, "Image Based Shadowing in Real-Time Augmented Reality," *International Journal of Virtual Reality*, vol. 5, no. 3, pp. 1–10, 2006.

[6] C.B. Madsen, T. Jensen, and M.S. Andersen, "*Real-Time Image-Based Lighting for Outdoor Augmented Reality under Dynamically Changing Illumination Conditions*," presented at the International Conference on Graphics Theory and Applications, Setúbal, Portugal, 2006, pp. 364–371.

[7] C. Madsen and R. Laursen, "*Scalable GPU-Based Approach to Shading and Shadowing for Photo-Realistic Real-Time Augmented Reality*," presented at the International Conference on Computer Graphics Theory and Applications, Barcelona, Spain, 2007, pp. 252–261.

[8] G. Nakano, I. Kitahara, and Y. Ohta, "*Generating Perceptually-Correct Shadows for Mixed Reality*," in Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality, Cambridge, United Kingdom, 2008.

[9] M. Aittala, "*Inverse lighting and photorealistic rendering for augmented reality*," The Visual Computer, vol. 26, no. 6, pp. 669–678, Apr. 2010.

[10] Z. Noh and M.S. Sunar, "*Soft Shadow Rendering based on Real Light Source Estimation in Augmented Reality*," Advances in Multimedia-An International Journal (AMIJ), vol. 1, no. 2, p. 26, 2010.

[11] S. Pessoa et al, "*Photorealistic Rendering for Augmented Reality: A Global Illumination and BRDF Solution*," presented at the Virtural Reality Conference, Boston, MA, US, 2010, pp. 3–10.

[12] B.F. Jenson et al., "*Simplifying Real-Time Light Source Tracking and Credible Shadow Generation for Augmented Reality using ARToolkit*," Medialogy, 2010.

[13] X. Jiang, A.J. Schofield, and J.L. Wyatt, "*Shadow Detection based on Colour Segmentation and Estimated Illumination*," presented at the BMVC, Dundee, UK, 2011.

[14] X. Chen, K. Wang, and X. Jin, "*Single image based illumination estimation for lighting virtual object in real scene*," presented at the 12th International Conference on Computer Aided Design and Computer Graphics, San Jose, CA, USA, 2011.

[15] L. Gruber, T. Langlotz, P. Sen, T. Hollerer, D. Schmalstieg, "*Efficient and Robust Radiance Transfer for Probeless Photorealistic Augmented Reality*," presented at the Virtual Reality 2014 IEEE, Minneapolis, Minneapolis, MN, USA, 2014.

[16] D. Kamboj and W. Liu, "*Improved Variance Cut Algorithm for Light Source Estimation in Augmented Reality*," International Journal of Computer Applications, vol. 85, no. 19, Jan. 2014.

[17] N. Michiels et al., "*Interactive Augmented Omnidirectional Video with Realistic Lighting*," presented at the International Conference on Augmented and Virtual Reality, 2014, pp. 247– 263.
[18] K. Rohmer, W. Buschel, R. Dachselt, T. Grosch, "*Interactive near-field illumination for photorealistic augmented reality on mobile devices*," presented at the 2014 IEEE International Symposium on Mixed and Augmented Reality (ISMAR), 2014, pp. 29–38.
[19] H. Alhajhamad, M. S. Sunar, and H. Kolivand, "*Automatic Estimation of Illumination Features for Indoor Photorealistic Rendering in Augmented Reality*," presented at the International Conference on Intelligent Software Methodologies, Tools, and Techniques, Naples, Italy, 2015.
[20] K.E. Soulier, M.N. Selzer, and M.L. Larrea, "*Real-Time Estimation of Illumination Direction for Augmented Reality with Low-Cost Sensors*," presented at the XXII Congreso Argentino de Ciencias de la Computación, 2016.
[21] M. Kasper, N. Keivan, G. Sibley, C. Heckman, "*Light Source Estimation with Analytical Path-tracing*," arXiv, Jan. 2017.
[22] Glen K. Straughn, "*Real-time inverse lighting for augmented reality using a dodecahedral marker*", 2018. https://pqdtopen.proquest.com/pubnum/10810575.html . [Accessed on Feb-12-2020]
[23] Salma Jiddi, Philippe Robert, Eric Marchand, "*Estimation of position and intensity of dynamic light sources using cast shadows on textured real surfaces*". ICIP 2018-25th IEEE International Conference on Image Processing, Oct 2018, Athens, Greece, pp.1063-1067.
[24] Schwandt, T., Kunert, C., Broll, W, "*Glossy reflections for mixed reality environments on mobile devices. In: Cyberworlds*" 2018. Institute of Electrical and Electronics Engineers (IEEE) (2018).
[25] A. Alhakamy and M. Tuceryan, "*AR360: Dynamic Illumination for Augmented Reality with Real-Time Interaction*," *2019 IEEE 2nd International Conference on Information and Computer Technologies (ICICT)*, Kahului, HI, USA, 2019, pp. 170-174.
[26] "OpenCV". [Online]. Available: https://opencv.org/. [Accessed on July-4-2020]
[27] "Blender". [Online] Available: https://www.blender.org/about/ [Accessed on July-4-2020]
[28] "Blender 2.79 Manual". [Online] Available: https://docs.blender.org/manual/en/2.79/render/blender_render/materials/nodes/types/input/lamp_data.html [Accessed on June-29-2020]
[29] "How to find surface normal of a triangle?" [Online]. Available: https://math.stackexchange.com/questions/305642/how-to-find-surface-normal-of-a-triangle [Accessed on June-20-2020]

VITA

Saptami Biswas

Candidate for the Degree of

Master of Science

Thesis: A COMPARATIVE STUDY OF THE PERFORMANCE OF REAL-TIME INVERSE LIGHTING WITH MATTE, SEMI-GLOSS AND GLOSS SURFACES

Major Field: Computer Science

Biographical:

Education:

Completed the requirements for the Master of Science in Computer Science at Oklahoma State University, Stillwater, Oklahoma in July, 2020.

Completed the requirements for the Bachelor of Technology in Computer Science and Engineering at West Bengal University of Technology, West Bengal, India in 2018.