Universiteit Utrecht

Realtime Environment Lighting Simulation for Augmented Reality Applications

Marco Antonio Salcedo Sanson

September 2017

Contents

1 Introduction **2**

2 Related Work **4**

3 Theoretical Framework **6**
3.1 Augmented Reality 6
3.2 Sensors 7
3.3 Standard Illuminant 7
3.4 Panoramic photography 8

4 Method **10**
4.1 Capture 360 panoramic image 11
4.2 Image processing 11
4.3 Calculate light properties 12
4.4 Calculate ambient light 13
4.5 Real-time Phase 14

5 Implementation **16**
5.1 Technologies 16
5.2 Implementation 17

6 Experiments **18**
6.1 Setup 18
6.2 Results 20
6.3 Gallery 22

7 Discussion **23**
7.1 Discussion 23
7.2 Known shortcomings 23
7.3 Conclusion 24
7.4 Future work 24

Chapter 1

Introduction

The rendering of virtual objects on current AR applications has a number of shortcomings in the state of the art. It is quite easy to differentiate a superimposed virtual object from a real one, even if they represent the same object. There are many factors that come into play and give away the lack of feasibility of an AR object, such as marker jiggle, inaccurate depth cues, lack of occlusion from real life objects and incompatible lighting. The method proposed in this work addresses the latter problem, the incompatible light conditions of the virtual and real world. The method is based on the idea that a 360 photograph of the environment can give us a lot of information about the lighting conditions and that such 360 photographs are now simple to create locally on an average mobile device. From this 360 photograph light sources are detected and filtered by intensity, dim lights or sources that are too far away and thus don't contribute to much to the local illumination are not taken into consideration. From a spherical 360 panoramic photograph the direction of the light sources with respect to the device can be calculated. The average color of the light source can also be inferred from the photograph, common light sources are within the light blue or yellow spectrum, but in special cases one could use colored lights with filters or special light bulbs, and having the AR objects reflect those colors would also contribute a lot to the illusion of reality of the overall scene.
The method's output is a set of lights whose properties are a rotation, color, and an intensity. It's important to stress the fact that the method is proposed for mobile devices. There have been previous works, detailed in the next chapter, that deal with the same problem but on computers. Mobile devices present advantages over computers for this task, but also the downside of generally lower processing power compared to computers. This decision comes from the ease of producing the 360 photographs locally, from the judgement that AR is better suited for mobile devices than computers (for mobility and ease of use reasons) and also to be able to exploit the devices geolocation hardware. Of course this doesn't mean that the method will only be applicable to mobile, there are also ways to adapt it to work on computers too.

The research question from which this method was originally envisioned is: "Can the accuracy of lighting in AR applications be improved using a mobile device to it's full potential?". What this means is that thanks to the many ways a smartphone or tablet can interact with the user and the environment we can know a lot about the surroundings. And this information could potentially be very useful to save computation time, which will make up for the lower computational capabilities and make for a more fair comparison to other methods.
The contributions of this method are:

- An image-based method with which the lighting conditions of the entire environment can be approximated
- The expansion of an already existing AR tracking library (ARToolkit) to add to the realisim of the rendering.

Chapter 2

Related Work

In terms of image-based methods to detect light sources, Laskowski[9] came up with a method to situate a pixel in an image that is determined to be the placement of a light source. It's a quite robust method that supports cases where there are no light sources visible and HDR images. The shortcomings of this method are that its response to outdoors photographs where there is only ambient light is not very accurate; and that his algorithm has a big trade-off with accuracy and converge time, there is an optional step involving a Bresenham method which makes results a lot more accurate but also presents a significant slow down. In this work the method will also go beyond, not just detecting the light sources in the static image, but also calculate their properties in 3D space.
Agusanto et al.[1] proposed a method similar to what is intended as a whole in the one presented here. The main differences in the method they propose have to do mostly with the technologies used. Being already an old paper, their method was conceived for Augmented Reality on a computer, rather than on a smartphone. The choice of a smartphone comes with the downside of lower processing power, but with the great advantage of the different sensors available on the device. The method in the Agusanto paper infers the lighting conditions using hand-crafted HDR photographs of the real scene using highly reflective metal spheres in contrast with my proposition of using the 360 degree photograph on the device itself, which is a lot more accessible to an average consumer.
Pessoa[11] expanded on Agusanto's work and created the Real-Time Photorealistic Rendering of Synthetic Objects into Real Scenes (RPR-SORS) toolkit as an extension of the ARToolkit Augmented reality SDK. Although they use more modern techniques, such as cubemaps instead of HDR photographs, their approach still has the same differences than the Agusanto method.
There have been several methods to insert virtual objects into static photographs automatically with highly realistic results. One such example is the method by Karsh [7] These kind of methods are not suitable for real- time applications, and they use a single view image to infer the lighting conditions. The method proposed here is more robust in the sense that it uses a 360 view of the environment to eliminate assumptions of what there is out of frame.

Xing[13] also devised a method to compose virtual objects into static photographs, focusing on outdoors scenes. The method is quite sophisticated, but a lot of the information fed to the light calculation functions involve manual input by the user. The proposed method is intended to work both for indoors and outdoors scenes, the main difference with the way outdoors scenes will be managed is that the method works automatically, without user intervention. This is accomplished by the usage of the mobile device's dedicated geolocation hardware. By using it, there is no need to assume anything or ask for any user input, the device will be able to tell a good approximation of the sun position using only the sensor's information.
This method is also based on Kanbara's work [8] to a certain extent. This 2004 paper proposes a method to calculate direct lighting on an AR object using a 2D/3D marker. One part is a regular fiducial marker and the other is a small chrome sphere. The light reflections on the sphere are detected and from them the light direction is approximated. The need of a physical chrome sphere is the main shortcoming of this method. The method proposed in this work builds on this idea but using a virtual analogy of the chrome sphere consisting of a spherical panoramic image mapped on a sphere, this has all the advantages of Kanbara's method without the disadvantage of needing an actual reflective sphere.

Chapter 3

Theoretical Framework

In this chapter a few technical concepts will be explored in order to make the proposed method and what it aims to accomplish more clear.

3.1 Augmented Reality

As of yet, the definition of Augmented Reality that is most widely accepted is the one presented by Azuma[2]. In this survey Augmented Reality (or AR for short) is defined as a software system that features a blend of real and virtual objects, the user can interact with it in real time and presents a 2D representation of a 3D world for both the real and virtual sides. A feature film that integrates real and virtual characters or objects is therefore strictly speaking not AR for example, because it's static content, the consumer cannot interact with it at all, let alone in real time. In order to combine real and virtual objects both worlds must be aligned, in the sense that they must share a common origin for both position and rotation. If the virtual objects present in the real scene are not aligned the feasibility of the whole composition is compromised. The tracking of real world features as a guideline for alignment in the virtual world is one of the challenges in the field of AR. There are many techniques that have been used, such as sensor-based tracking, that uses dedicated hardware to feed the software application the position and rotation information it needs; vision-based tracking, in which the position and orientation origin of the virtual world are determined by images recognized by a computer vision algorithm. These images are named fiducial markers and can be an artificial black and white image, a "natural" images (points, lines, shapes and/or textures) or even 3D objects. Other approaches involving both sensors and vision algorithms have also been used, if the reader is interested in learning more about tracking, Zhou's survey[14] on tracking is a good resource. This method does not aim in any way to improve tracking for AR, current state-of-the-art methods will be used for that, but it's important to explain the concept a bit in order to settle the scope and expectations of the end result.

3.2 Sensors

Most smartphones and tablets in the market nowadays have built-in sensors to measure many things that are helpful for application developers, such as motion, orientation and other environmental conditions. The ones that are relevant for this method are the following:
Accelerometer: This device measures proper acceleration relative to gravity. Its application in mobile development is to measure motion changes and to measure the device orientation relative to Earth's surface. It usually consists of 3 orthogonal axes. The output is given in m/s^2
Gyroscope: A sensor that is capable of measuring the rate of rotation around a particular axis. It serves the same purpose as the accelerometer, but mobile devices usually have both for more robust measurements. The output is given in rad/s. In the method these will be used to acquire the 360 panoramic photograph, making sure that the device is within the same pitch while capturing all the images that will be stitched together to make the panorama.
Photometer: There is a wide variability in terms of capacity of photometers across different devices, but the only measurement that is guaranteed is the ambient illuminance expressed in lx. In the method it will be useful for detection of a noticeable change in ambient lighting, leading to a re-capture of the panorama. What exactly constitutes a noticeable change in ambient lighting will be determined through parameter tuning.
Magnetometer: In terms of mobile devices it serves the purpose of a compass, it measures the device's orientation with respect to the Earth's magnetic poles. In mobile development it is useful to get the heading of the device expressed in degrees.
GPS: Measures the raw position of the device in three-dimensional Cartesian coordinates with origin on the center of the Earth. It can be used to determined where on planet Earth the device is located (Country, city, neighborhood, etc). It needs to have line of sight with no electromagnetic interference with at least 4 out of the 24 satellites in orbit that are used for GPS.

3.3 Standard Illuminant

In order to describe a color of a light source it is important to have detailed knowledge of the illuminant used. The International Commission on Illumination (CIE) have defined a number of spectral power distributions, referred to as CIE standard illuminants, to provide reference spectra for colorimetric issues. The illuminants are denoted by a letter or a letter-number combination. Their spectral power distributions (SPD) are normalized to a value of 100 at a wavelength of 560 nm in following figures. Illuminants series A through D exist, all of them specializing on a different kind of light source. Fo this method the interesting one is series D, because it describes a sort of average light source, typically associated to daylight global illumination.
D65 corresponds roughly to the average midday light in Western Europe /

Northern Europe (comprising both direct sunlight and the light diffused by a clear sky), hence it is also called a daylight illuminant. It has a correlated colour temperature of approximately $6500K$. As any standard illuminant is represented as a table of averaged spectrophotometric data, any light source which statistically has the same relative spectral power distribution (SPD) can be considered a D65 light source. It is important to note that D65 is purely theoretical, there are no actual artificial D65 light sources.[10]

3.4 Panoramic photography

Panoramic photography is a technique of photography, using specialized equipment or software, that captures images with horizontally, and sometimes also vertically, elongated fields of view. While there is no formal definition of the minimum field of view required for a photograph to be considered panoramic, in this work the interest is focused upon those that capture the entire 360 degrees of horizontal field of view.
There are different kinds of panoramic images, the most intuitive is the one called wide format photograph. It basically consists of a series of photos taken from the same height and later stitched together to appear as a single wide image. The image can be then projected on to a cylinder. This kind of panoramic photograph captures 360 degrees of horizontal field of view, while the vertical field of view is dependant entirely on the lens used to capture the individual photographs. Figure 3.1 shows an example of a cylindrical panoramic image.

Figure 3.1: A cylindrical panorama of Trafalgar Square

The type of panoramic image used in this method is called equirectangular panoramic image, or spherical panoramic image, due to the fact that the resulting image can be projected on to a sphere. The projection maps meridians

to vertical straight lines of constant spacing, and circles of latitude to horizontal straight lines of constant spacing. This kind of projection introduces the types of distortion often seen in spherical projections, such as Mercator, the poles of the sphere appear bigger than they really are. The process used by Google to create equirectangular panoramic images on mobile devices involves taking several pictures of the environment and using the device's sensors prematurely project them onto a sphere in their position relative to Earth's north. Once all pictures are captured they are stitched together in a similar manner as for cylindrical mapping, for each individual image features are identified, those same features are searched on neighboring images and aligned and the overlap is blended, resulting in a seamless, yet distorted representation of all the images as one. After this, the projection is applied as follows:

$$\lambda = \frac{x}{cos(\phi_l)} + \lambda_0 \tag{3.1}$$

$$\phi = y + \phi_l \tag{3.2}$$

Where:
λ is the longitude of the location to project.
ϕ is the latitude of the location to project.
ϕ_l are the standard parallels (north and south of the equator) where the scale of the projection is true.
λ_0 is the central meridian of the map.
x is the horizontal coordinate of the projected location on the map.
y is the vertical coordinate of the projected location on the map.

The kind of expected result can be seen on figure 3.2

Figure 3.2: An average indoors equirectangular panorama

Chapter 4

Method

This chapter covers the steps needed to produce the wanted results. The input needed is a fiducial marker M to provide the object position in the real world; and a 3D mesh O which will be rendered in Augmented Reality. The output is a set of light sources L_i, each one having a position, orientation, intensity, color, type and size. In order to complete the luminance ($L()$) analysis a 360 panoramic image is required, but it will be generated in a previous step with a separate application as explained in the Theoretical Framework chapter. The entirety of the process is layed out in the flowchart in Figure 1, and each step is described in more detail afterwards.

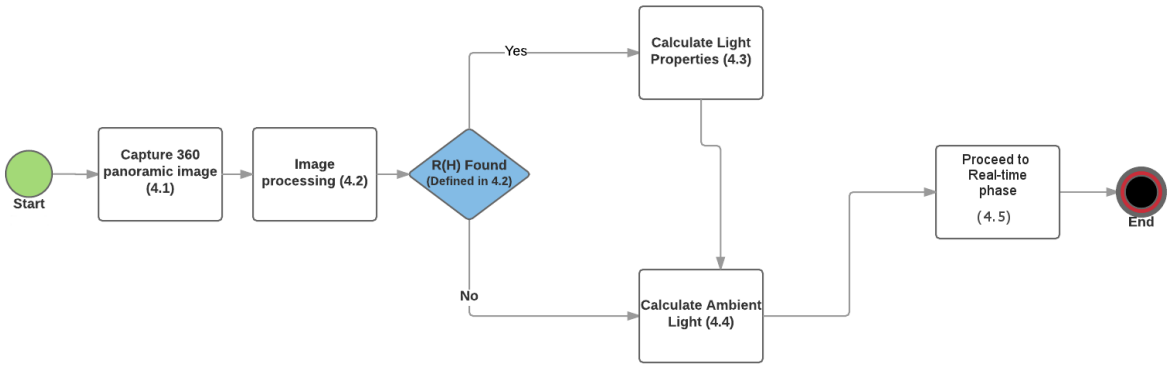

Figure 4.1: The method in a nutshell

4.1 Capture 360 panoramic image

In this work panoramic images are used as a tool, therefore it's not within the scope of the method to define a new way of capturing 360 images. The suggested application to procure a spherical panoramic image within the device itself is Google Street View. The origin of the virtual world for placing and orienting O within it is asked to the user by rotating the virtual reflective sphere so that the texture offset matches the device orientation within the real world. This will also simplify calculations of light orientations later on.

4.2 Image processing

Once the panoramic image is loaded and oriented the first step in order to be able to analyze the luminance is to get rid of the chromatic information. This is achieved with the following equation:

$$L(R_{ij}, G_{ij}, B_{ij}) = 0,2126 \times R_{ij} + 0,7152 \times G_{ij} + 0,0722 \times B_{ij}; \quad \forall\ P_{ij} \quad (4.1)$$

Where L is the luminance obtained at D65 white point and P is the pixel in the i, j position of the image
The contrast ratio has to be adjusted, so that the regions with high luminance are more clearly differentiable.

$$g(i, j) = \alpha \times f(i, j) + \beta \quad (4.2)$$

Where $g(i, j)$ is the adjusted image, $f(i, j)$ is the original grayscale image and α and β are the brightness and contrast constants, determined by parameter tuning. The parameter tuning will be carried out in a trial and error way, propose initial extreme values and run the program, changing the values in order to achieve the best result. The values that worked in the implementation were $\alpha = 220$ and $\beta = 255$. In order to prevent outlier pixels and noise from causing false positives, let $g(i, j)$ be normalized like so:

$$L(P_{ij}) = \frac{2 \times P_{ij}}{min(L) + max(L)} \quad (4.3)$$

Where $min(L)$ and $max(L)$ are the overall minimum and maximum luminance values in the image.
The result of these steps is a black and white image with the rough shape of the light source. If the image ends up completely black it means there are no important visible light sources and ambient illumination alone would give a good enough result on the rendered model. A region of high luminance is defined as follows:

$$R(H) = \{p_{00}, p_{01}, ..., p_{mn}\} \quad (4.4)$$

Where p_{ij} is the pixel in the i,j position of the image. Such that

$$L(p_{ij}) > 0.9 \times max(L(p_{ij}))$$

The region must also be continuous, so the pixels must be adjacent.
If there are regions of high luminance in the image, they would be discovered with a fair amount of noise and artifacts, caused by clear objects, reflections of light sources on highly reflective surfaces, or even light sources that are so far away in the distance that don't really contribute an important amount of light to the area of interest. Therefore it's necessary to also add a relative size constraint to the definition. What constitutes an acceptable region size to deem the light source is not an easy question to answer. The best approach to face this problem is to define it as a percentage of the width and height of the overall image and tune the parameter in search for the best solution. The size constraint would therefore be:

$$width(R(H)) \times height(R(H)) \geq k \times W \times H \tag{4.5}$$

Where k is the parameter to be tuned, in the implementation the value that yielded the best results was $k = 0.004$; W and H are the total image width and height.

These constraints also help keep the amount of lights to be processed within an acceptable range for a real-time application, even if the real environment has many light sources a simplification of them is necessary when modelling them to keep the application feasible. In the implementation the maximum amount of lights was capped to 8 because the overall performance of the application started to suffer with more lights. However, in order to identify the most relevant light sources it's necessary to calculate their parameters first.
Once a set of light sources have been identified their properties need to be calculated.

4.3 Calculate light properties

The properties needed to calculate for each light are orientation, intensity and color. In the real world it is not trivial to calculate the position from just a single point of view as input, so even though the simulation would be a lot more robust if the actual position with respect to the camera was simulated there is just not enough information to calculate it in this method and all the lights are assumed to be at unit distance from the camera. With this in mind, all the other properties can be calculated.

1. Position:The x, y and z components will be those of the unit orientation vector, to ensure that the light is in the correct direction at unit distance.

2. Orientation: The pixel coordinates (x,y) of the centroid of each light within the luminance panorama are saved. The image is projected on

a sphere. The camera casts rays onto the sphere from 4 different points of view, 0°, 90°, 180°and 270°. If the ray hits a white pixel and the pixel coordinates of the hit are approximately those of the centroid the light direction is calculated as follows:

$$O(L_p) = -2 \times (N_h \cdot C_r)N_h + C_r \tag{4.6}$$

Where $O(L_p)$ is the orientation of the pth light source, N_h is the sphere normal on the hit point, C_r and is the camera ray direction.

3. Color: Storing both versions of the panorama, one in full color and another one after processing will allow us to have both the color and the luminance information. Once a light source is detected, the equivalent area in the color image can be averaged to determine the color of the light source.

4. Intensity: There are two factors that influence the intensity of a light as perceived by a camera, the light size and the color temperature. The light's color temperature in Kelvin can be calculated as follows:

$$T(C_p) = -949.86315 + 6253.80338\,e^{\frac{-n}{0.92159}} + 28.70599\,e^{\frac{-n}{0.20039}} + 0.00004\,e^{\frac{-n}{0.07125}} \tag{4.7}$$

$$n = \frac{0.23881 \times R + 0.25449 \times G - 0.58291 \times B}{0.11109 \times R - 0.85406 \times G + 0.52289 \times B} \tag{4.8}$$

Where R, G, B are the red, green and blue components of the light color. The size analogy is given by the integral of the region of high luminance with respect to the full image. The light intensity is finally expressed by:

$$I(L_p) = T(C_p) \times \int_R L(i)\,dx \tag{4.9}$$

Where $L(i)$ is the output image of the luminance analysis and R is the pth region of high luminance.

4.4 Calculate ambient light

Environment mapping is an image-based technique to approximate the appearance of the overall light conditions of an environment. This is accomplished by means of a precomputed texture image mapped as a far-away environment surrounding. Said surrounding is usually a geometric surface, when Blinn first introduced the method[3] a sphere was used. Nowadays there are other alternatives, such as cube, paraboloid, pyramid or cylinder maps. The principle for each surface is the same, but the way to map a planar image onto the surface varies per surface.
Since a panoramic image of the environment is already available using it to implement environment mapping would be an adequate use of resources. In order

to generate an environment map it is necessary that the panorama is made into a High Dynamic Range image. This is because the Low Dynamic Range image captured directly from the device camera fails to capture the information necessary to simulate correct color balance, shadows, and highlights of the lighting environment; ultimately producing both inaccurate and less visually pleasing results. This has been illustrated by Paul Debevec.[5]
A relatively easy and effective way to make an image into an HDR version is a technique called Tone Mapping, in which versions with different exposure values of the same image are blended together to include the full range of highlights and shadows of the overexposed and underexposed versions in a single image. Since asking the user to capture the environment more than once would have a bad impact on user friendliness, and also it's highly unlikely that the produced image would have the exact same framing every time, the different exposure values for the Tone Mapping will be produced altering the brightness and contrast values of the base image using equation 2 once again. After that the HDR image is produced using Debevec's weighting algorithm[6].
It's important to disclaim that the Tone Mapping process will not yield an actual HDR image. In the first place, it will be a standard 24 bit image in the 0...255 range, with highlight and shadow valued clipped. The upside to still going through this process nonetheless is that the environment will be described in a richer way, capturing the bright areas and the shadows better than the standard exposure image.
After the HDR version of the panorama is created it can be used to have an actual ground truth about the environment light color and intensity at any given point in the virtual space. This will be detailed in the Real-time Phase step subsection.

4.5 Real-time Phase

The original sphere panoramic photograph is widely used in the real time phase. It is used to create a cubemap of the real environment to be used as ambient lighting, to calculate the ambient contribution of the diffuse shaders and to fake reflections for the specular shaders.

1. Cubemap: The image coordinates are first converted to polar and divided into four regions by latitude $-\Pi/4 < \theta < \Pi/4, \Pi/4 < \theta < 3\Pi/4, 3\Pi/4 < \theta < 5\Pi/4, 5\Pi/4 < \theta < 7\Pi/4$. These represent either one of the four faces of the cube, top or bottom. The projected coordinate is given by:

$$P = (1, tan(\phi), \frac{cot(\theta)}{cos(\phi)}) \tag{4.10}$$

   The projected point is always on the top if $\frac{cot(\theta)}{(1/\sqrt{2})} > 1$ or $tan(\theta) < \frac{1}{\sqrt{2}}$

   The projected points for each face of the cube are composed into an image and each image stitched together to create a cross cube map.

2. Ambient contribution: The environment lighting contribution is done via shader. The ambient contribution is reduced to a single value in the range of 0 and 1 and then the albedo color component of the object's shader is multiplied by this value. The result is a dimmer color when the ambient light is low. The ambient contribution value itself is obtained by calculating the mean of the luminance panoramic image, since this already contains the lighting information of the room and the different intensities and distributions.

3. Reflections: Another way the panoramic image is used to enhance the realism of the composted scene is by generating accurate reflections of the real scene. The cubemap from a previous step is also used to fake reflections. A vector is cast from every vertex of the object along its normal and intersected with the cubemap generated from the panoramic image. The color is mixed with the albedo according to the specularity defined for the material to simulate reflection, this information is calculated in advance and used at runtime. This works well enough because we know beforehand that the scene is static, there will only be one non-animated object throughout the full execution.

4. Shadows: During early experiments it became clear that one of the bigger differences between real and virtual object were a product of the shadows as well as the light. There are two main problems when it comes to casting shadows for virtual objects in AR, one is the fact that there has to be an object underneath the 3D model to cast the shadow on, but this object has to be as unobtrusive as possible. If the goal is to achieve realism having the object appearing on a pedestal for the sake of casting shadows is counter productive. This issue is worked around by using a transparent material that receives shadows. The other main problem is that the shadows of real objects are deformed when cast on other objects, this problem is beyond the scope of this method, as it would require the program to have a notion of the geometry of the real space.
   The shadows work quite well, but are not as soft as real shadows. A masking scheme using a gradient texture was tried to make softer shadows but it didn't work, because the color was contaminated and it made apparent that there was a shadow catching plane.

Chapter 5

Implementation

In this chapter, the specifics of the implementation made for the demo application are discussed. In general it's a direct translation of the theoretical method proposed there were a few adaptations needed because of the way light is modelled in raster graphics.

5.1 Technologies

The idea of the method is that after the experiments and the eventual would-be finished product can be used within a mobile device with no help from computers. To that end, the implementation uses different technologies. For the capturing of spherical panoramic images the Google Street View app is used, it's available for both iOS and Android and it's a ready-made solution to use panoramic images as a tool for the experiments. The base of the implementation is done in Unity, but a native code library is used within Unity. Said native code library is written in C++ and uses OpenCV for all the image processing tasks. OpenCV is also available as a library for Unity, but it's a paid library, it doesn't have all the functionality that its C++ counterpart has and offloading the heavy image processing parts of the method to native code results in faster execution.
All of these technologies offer platform agnosticism by themselves, however adjustments do have to be made in order to target either iOS or Android. For instance, to be able to use the panoramic images created with Street View another native code plugin was necessary to read images from the device's file system, this plugin would be a completely different implementation from platform to platform. The ARToolKit library is used for the Augmented Reality tasks of tracking markers and providing the virtual world's point of reference. The device used for this particular implementation is an iPad Air tablet running iOS 11, however adjusting the application to support Android devices as well would be a simple task.

5.2 Implementation

The system consists of three components, the Google Street View app to capture the panoramic image, the C++ and OpenCV library to that takes care of most of the pre-calculation steps defined in the method and the Unity application that puts everything together and takes care of the real-time phase of the method.
The OpenCV API has all the necessary functions implemented to apply the mathematical functions proposed in the method section so this part of the implementation was a direct translation into code. In order to achieve the desired graphic quality the implementation uses several tools at Unity's disposal. Reflections are implemented using Reflection Probes for example. Shadows are also baked using Unity, but a custom shader was necesary for the "shadow catcher" plane.
In an attempt to homologate the look of both the real and virtual camera, a full screen effect was applied to the composted scene, in an attempt to make the clean virtual camera match the noisy device's camera. This was done through a Film Grain camera effect.

Chapter 6

Experiments

In terms of scope the aim of the method is to produce plausible lighting for Augmented Reality applications, and thus increase the sense of realism of the composed graphics. This does not mean that imagery that would for all practical purposes be indistinguishable from reality will be achieved. As said earlier, the incompatible lighting is only one of the problems with current AR graphics. In order to achieve complete realism all the other problems would have to be tackled as well. The method also does not attempt to improve other aspects of AR, such as tracking. And so the experiment is designed only to evaluate the similarity of a real object and a virtual representation of the same object in the same controlled lighting conditions.

6.1 Setup

In order to test the results a real object with a variety of materials was chosen, in this case it was an Xbox 360 controller with a custom paint job. A 3D model of the same Xbox 360 controller was modified to match the custom paint job and the materials were replicated as closely as posible to the real object using Unity's built-in shaders. In the end 4 main shaders were used, a highly reflective plastic for the borders and some of the buttons, a more matte plastic for the main body, a completely specular chrome for the Xbox button and a semi-transparent and glossy plastic for the colored buttons. The choice of this motif was based both on the ease to find a reliable 3D counterpart for a real object and on the already wide variety of materials present in the object.
In order to provide a ground truth for a reliable size by side comparison a screen capture of the application running while both the real and virtual object are in the frame, in similar positions and orientations and affected by controlled lighting that is also simulated using the method for the virtual counterpart. Another experiment will be a direct substitution, leave the controller on the table, remove it from frame and place the marker instead, in a way such that

the virtual controller will appear in a similar position and orientation in order to provide a clear comparison, everything remains the same and both objects can be appreciated in the exact same setting.
As for benchmarking how this method stands in comparison to other similar ones, the conditions of the experiments presented in the results section for the methods in [8], [7] and [11] are replicated in terms of similar setting and virtual objects used. The resulting images are compared to the ones from their methods. In order to replicate these settings additional 3D models are needed, namely a teapot and the dragon and Buddha from the Stanford 3D Scanning Repository. The actual way in which the experiment is conducted is defined in the following scenario:

- **Setting:** A room with consistent and invariable lighting is used and some kind of flat, matte surface to lay the objects on. An Xbox 360 controller with the characteristics described previously and a marker to track the virtual object.
- **Requirements:** A mobile device running the developed demo application, a marker for virtual objects. A real object and its corresponding virtual counterpart, modelled as close as possible. Common 3D models from the Stanford 3D Scanning Repository.
- **Goals:** Obtaining a set of images that will enable readers and experimenters alike to make a fair comparison of the method application, side by side with a real object counterpart.
- **Actions:** Place the controller on the surface, and capture and image. Then remove the real controller and substitute it with the marker in a way that the virtual controller appears in the most similar position and orientation possible; capture an image in the end as well. Replicate the scenarios in [8], [7] and [11] using teapots, Buddahs and dragons and capture images of each.
- **Benchmark:** The image result yielded by the method, as well as the time per frame will be captured and used for comparison's sake and benchmarking head to head with the results from similar methods.

It's important to also mention the need of replicating the previous work's settings and the reason why. In some cases the product is not a runtime application, and thus the only applicable comparison is image-based. The authors of other runtime methods were approached to evaluate the possibility of using their binaries to produce images and they either declined or gave no answer. And so the only possible comparison is image to image and taking their word for the performance indications in the form of frames per second.
All of the experiments were recorded on video by connecting the tablet device to a laptop, and are available to watch. The images shown here are select still images from said videos and the framerates discussed per experiment are averages from the on-screen counter seen on each video. It's also important to say

that the recorded framerates are about 10% lower than when the application is not being recorded.
.

6.2 Results

Experiment 1: Real object substitution
This experiment produced quite satisfactory results, the materials and the position and orientation on the virtual object are not exactly the same, and so the highlights shining off the surface are not in the same places. But the shadow shows that the light is placed in a very acceptable approximation of the real light. The application is running at 24 FPS on this screen capture and it's a good measure of the average of performance.

Figure 6.1: Real custom painted Xbox 360 controller

Figure 6.2: Virtual counterpart of the same object

Experiment 2: Karsch's scenario
This scenario was hard to replicate, due to the unusual framing and placing of the objects. The image produced by this method is plenty different, and the method it's being compared with is not a runtime method, but even so there are conclusions that can be drawn. In the image produced by Karsch's application it's hard to say if the lighting in the real and virtual parts of the image really correspond, the light sources, except for one, are not present in the frame and there are no similar real objects to the virtual ones that would serve as ground truth. In this methods result there is enough evidence to prove that the lighting in both the real and virtual worlds is similar, with cues such as the presence of real objects and their shadow directions.
In this methods application the framerate at 15 FPS is still quite acceptable considering that there are 6 virtual dense models, but given the fact that Karsch's method is not interactive there's no possibility to compare in this regard.

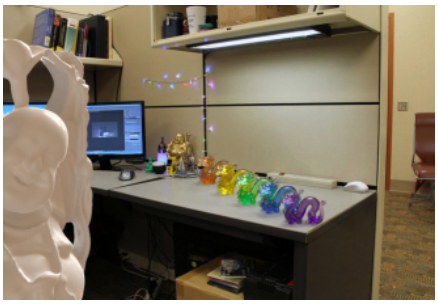

Figure 6.3: Karsch's method results

Figure 6.4: This methods's results

Experiment 3: Kanbara and Pessoa's scenarios
These two scenarios will be grouped together due to the fact that the steps to replicate the experiment are the same. They both presented ceramic material teapots with a lighting setting that they did not specify.

Figure 6.5: Pessoa's method results

Figure 6.6: Kanbara's method results

Both results look well blended into the environment, however in Pessoa's case it also must be said that the lighting setting is not disclosed and is not obvious from just looking at the results. For the aforementioned method there's also a considerable amount of pre-production required for the method to actually work, and the performance has a rather bad scaling, ranging from 180 FPS with no objects to 5 FPS with seven objects. The method in this work doesn't have those problems, the only pre-step required is to take a spherical panoramic image with the same device and no technical knowledge necessary; and performance scales better going from 24 FPS with a single object to 15 with six of them.
In the case of Kanbara's results they look really good, the downsides to their method that the one here resolves are the need for a physical 3D marker to probe lighting and the performance, they report 20 FPS on a computer for the experiment they published. It's admittedly a more than 10 year old computer

at the time of this writing, but the method proposed here is achieving slightly better framerates on a mobile device.

Figure 6.7: Pessoa's comparison results

Figure 6.8: Kanbara's comparison results

6.3 Gallery

Here are a few more screen captures from the application that show the potential of the method for the reader to draw their own conclusions.

Chapter 7

Discussion

In this chapter the results, discussed and a conclusion is formulated. The shortcomings of the method and the implementation are also mentioned and taken into consideration for future work.

7.1 Discussion

From the resulting images a very close resemblance among both virtual and real object can be appreciated, however it's still not a perfect fit. Some of the factors cannot be worked around trivially, for example one of the differences is that there are small imperfections present in the real object that are not there in the virtual one, in order to account for them, a 3D scan of the specific real object would be necessary. 3D scans are often extremely dense meshes that would pose a problem to maintain interactive framerates.
Despite the film grain filter, it's still evident that the virtual camera has better resolution than the real one. The fact that in the virtual world everything looks "too perfect" also breaks the illusion of realism to a certain extent. However, in a real application of the method the user would ideally use AR to view something that is not there in the real world, instead of a side by side comparison to a real object. So by not having a clear ground truth of how such an object would look through the camera feed this problem is mitigated.

7.2 Known shortcomings

There are a few known cases where false positives are found by the light detection algorithm. When a light source is directed towards a glossy surface, such as a varnished desk for example, the incidence of the light on such a surface is detected as a light source as well. A similar thing happens sometimes for intense white objects that are not light sources. Although in the first scenario it can be argued that the light bouncing off a glossy surface could indeed be considered a light source. The system is closed once the runtime phase starts,

therefore it lacks adaptation to light changes, due to the nature of the method, having a pre-calculation phase and a runtime phase, it wouldn't be possible to change the light setup in the same way as it was originally captured.
The system limits the amount of lights it simulates to 8 for performance reasons, but even at 8 light sources the performance of the application is low. Between 15 and 24 FPS, it can still be considered interactive, but it's not ideal, specially when taking into account that the application is only doing rendering, a real application of the method, such as a game, would require other layers of complexity that would need processing time.

7.3 Conclusion

All in all, despite the shortcomings of the method the objective was achieved. The main goals of the method are to approximate the lighting conditions of the environment via a panormaic image and to use this knowledge of the environment to improve the realisim of the rendering. Both objectives are met, in the understanding that improvement is not an absolute measure, meaning that as long as the method makes the composted scene more realistic the objective is fulfilled, regardless of being far from perfection still.
There's room for improvement, both on the technical side, on the user experience side and in the results. This will be further developed in the next section.

7.4 Future work

During the development of this method there were several changes to the Augmented Reality landscape through the introduction of new SDK's by Apple and Google. ARKit by Apple and Tango/ARCore by Google are much more robust solutions than the ARToolKit used in this method. While both ARKit and ARCore have an ambient light estimation feature, and ARCore supposedly even has direct illumination estimation, the overlap with what was proposed here is not complete. In a future iteration of the system it would be interesting to see the method working with these new SDK's. Google Tango would be even more interesting, due to the fact that it has a depth sensing camera, so it would even be possible to roughly reconstruct the scene for accurate shadow casting and determine the distance to light sources precisely.
Other than that, a vast improvement would be support for changing light conditions. Most real-world applications for such AR graphics would require interaction outdoors for more than a few minutes, so the lighting conditions are bound to change in the middle of a session.
Just for the sake of maintaining a more uniform user experience it would be a nice addition to implement something similar to the Google Street View spherical panorama capture module within the app. This is something completely off-topic for the method and something not trivial to implement, but if it was there it would be a nice addition nonetheless.

Bibliography

[1] Kazuma Agusanto, Li Li, Zhu Chuangui, and Ng Wan Sing. Photorealistic rendering for augmented reality using environment illumination. *Proceedings of the Second IEEE and ACM International Symposium on Mixed and Augmented Reality*, 2003.

[2] Ronald T. Azuma. A survey of augmented reality. *Presence*, 6(4), 1997.

[3] James F. Blinn and Martin E. Newell. Texture and reflection in computer generated images. *Commun. ACM*, 19(10):542–547, October 1976.

[4] Aaron Chuang, Tsung-Lin Yang, and Yon-Way Chee. 360 degree panorama using an android phone. `http://www.cs.cornell.edu/courses/cs4670/2010fa/projects/final/results/group_of_acc269_ty244_yc563/cs4670_final.html`, 2010.

[5] Paul Debevec. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *ACM SIGGRAPH 2008 Classes*, SIGGRAPH '08, pages 32:1–32:10, New York, NY, USA, 2008. ACM.

[6] Paul E. Debevec and Jitendra Malik. Recovering high dynamic range radiance maps from photographs. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, pages 369–378, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[7] Karsch K., Sunkavalli K., Hadap S.and Carr N., Jin H., Fonte R., Sittig M., and Forsyth D. Automatic scene inference for 3d object compositing. *ACM Transactions on Graphics*, 2014.

[8] M. Kanbara and N. Yokoya. Real-time estimation of light source environment for photorealistic augmented reality. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 2, pages 911–914 Vol.2, Aug 2004.

[9] Maciej Laskowski. Detection of light sources in digital photographs. *In 11th Central European Seminar on Computer Graphics*, 2007.

[10] International Commission on Illumination. Cie standard illuminants for colorimetry. `http://www.cie.co.at/publ/abst/s005.html`, 1998.

[11] Saulo A. Pessoa, Guilherme de S. Moura, Joao Paulo S. do M. Lima, Veronica Teichrieb, and Judith Kelner. Rpr-sors: Real-time photorealistic rendering of synthetic objects into real scenes. *Elsevier*, 2011.

[12] Ibrahim Reda and Afshin Andreas. Solar position algorithm for solar radiation applications. *Midwest Research Institute, National Renewable Energy Laboratory*, 2008.

[13] Guanyu Xing, Xuehong Zhou, Qunsheng Peng, Yanli Liu, and Xueying Qin. Lighting simulation of augmented outdoor scene based on a legacy photograph. *Computer Graphics Forum*, 32(7), 2013.

[14] Feng Zhou, Henry Been-Lirn, and Mark Billinghurst. Trends in augmented reality tracking, interaction and display: A review of ten years of ismar. *IEEE International Symposium on Mixed and Augmented Reality*, 15(18), 2008.