Colour Correction for Projection-Based Augmented Reality

Xinyun Fang, BSc

A Dissertation

Presented to the University of Dublin, Trinity College

in partial fulfilment of the requirements for the degree of

Master of Science in Computer Science (Augmented and Virtual Reality)

Supervisor: John Dingliana

August 2022

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

Xinyun Fang

August 19, 2022

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

Xinyun Fang

August 19, 2022

Colour Correction for Projection-Based Augmented Reality

Xinyun Fang, Master of Science in Computer Science
University of Dublin, Trinity College, 2022

Supervisor: John Dingliana

Projection-based Augmented Reality (AR) is a type of AR implementation. Instead of using wearable devices, it uses the projector to display the virtual object in the real world. It is the best solution for some specific scenarios like showing AR to a group of people. When using projection-based AR in real life, the environment illumination is not controllable, and the projected background is not always white. Therefore, it is necessary to find some way to make the darker part clear and correct the colour from the colour deviation on a non-white background. This dissertation provides a reasonable solution to analyse the darker part and correct the colour. For the darker part, we lighter its surrounding areas to make it clear. For the colour correction, we involved the colour transfer method to modify the display image based on the background. We tested several different background analyses and colour transfer ways, and find the best solution for different backgrounds and display images. Then create a map to record the best solution for different occasions. our colour correction solution is based on this map. We first recognize the background colour and the style of the display image, and then use the map to find the best algorithm. Use this algorithm to manage the display image, and the result is the solution we will give. This solution can works better on a solid non-white background, but still can reduce the colour deviation on texture background.

Acknowledgments

Thanks to my supervisor John, who is nice and helps me a lot.

Xinyun Fang

University of Dublin, Trinity College
August 2022

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Problem Overview

Projection-based Augmented Reality(AR) is a type of AR implementation which uses a projector to display the virtual object in the real world. This kind of AR implementation can eliminate many drawbacks of other AR implementations such as head-mounted displays (HMD) and has its specific usage scenario. For instance, in an industrial assembly line, AR could be used to provide guidance and assistance to reduce the cost of training temporary workers. Wearable devices like HMDs would limit the views of the users and makes people uncomfortable for long-time wearing due to their weight. Based on the research by Sand et al. (2016), projector-based AR can provide a better experience in this situation. In addition, the traditional AR with wearable devices has limited by the devices. We have to provide additional devices if we want to add users. In some place that needs to provide a simultaneous view to multiple people like theme parks, the projector-based AR is more convenient and have fewer latency issues, and be able to provide a more immersive environment to the audiences. So the theme parks always use the projector instead of mobile devices to create a dynamic interactive virtual environment to immerse the guests (Mine et al. (2012)). Moreover, the projector-based AR has also been used in surgery(Edgcumbe et al. (2018)), construction (Yeh et al. (2012)), and many other areas.

Several interesting areas can help to improve the performance of the projector-based AR. For instance, the shadow is one of the important spatial cues for the virtual object. HMDs and projection-based displays are classified as 'additive' displays which work by adding light to the scene and they cannot reduce illumination from the environment. So how to represent shadow properly is an important question in this area (Chun Wei Ooi (2022)). Besides, the relative position of the projector, the surface, and the spectator is also a problem. How to display the 3D object on a deformable surface and make it looks real (Fujimoto et al. (2014)), how to make the position of the object proper for a given

spectator's position (Ro et al. (2019)), and many other problems remain to be solved. Moreover, since the background is not always white or black, a colour mixing problem needs to be solved to display the colour we want.(Gabbard et al. (2010))

In this dissertation, we will focus on the colour problems for projection-based AR. Colour blending is the key problem that affects the result of the projected colour. A non-white background cannot reflect every colour, and different colours will reflect and absorb different colours. Thus if we project the same colour on different backgrounds, the result colour is different both numerically and visually (Sridharan et al. (2013)). Besides, the visual illusion is another important factor when measuring colour problems. In the very early years, scientists have proved that the surrounding colours will affect the visual effect of the colour inside it, which is a kind of visual illusion (Rossotti (1985)). Therefore, it is a critical problem to figure out the proper compensation colour projected on the chromatic background to attach the visual effect we want.

1.2 Objectives and Contributions

The objective of the project is to find a proper algorithm to make the display image look right on a non-white background. The data we have is the image that needed to be displayed, and the target result is the effect that the colour is like displaying this image on a white background, and all the objects in the image are displayed clearly. To achieve this goal, we will first capture and analysis the background, and then modify the display image by different algorithms based on the different background colours. In addition, we will discuss solutions for dealing with darker parts of images including shadows, which are problematic for projection-based AR.

This project has provided a colour correction algorithm for the projection-based AR with an achromatic background. It involved the colour transfer method when doing the colour correction. Besides, it also handles the darker part. This project will provide an API with the background image and display the image as the input parameter, and return the corrected image that can be displayed. based on the test, we can find that it works better on a solid non-white background with a colourful image. But it also provides a general solution for the textured background that is able to reduce colour diversion.

1.3 Methodology

The key method in this project is the colour transfer method. This method has been involved to modify the display image based on the background. Besides, the darker part has also been considered by lighting the surrounding areas. Visual illusions like colour

consistency and brightness induction will also be wildly researched and considered in this project. The image processing methods like convolution methods have been used, and we also use the interpolation function to smooth the change. Considering the colour space, The CIE LAB colour space has been used in the colour transfer method. Besides, the RGB colour space and the HSV colour space have also been involved to handle the background, the dark part, and doing other relevant management.

As for the implementation, the Python and OpenCV library has been used to process the image. Python is a commonly used scripting language that has many supporting libraries and is easy to start. The OpenCV is one of these libraries that is widely used in computer version and image processing. It is the most string library in image processing that contains many useful APIs like mask generation.

The devices we needed to do the experiments were only a camera, a projector, and a light meter on a mobile phone. The camera is used to read the background and the result, and the projector is used to display the result.In addition, we employed a digital version of the ColorChecker (McCamy et al. (1976)), which is a standard colour calibration device in photography and has wildly used in computer graphic areas. It is a rectangle that contains 4*6 50mm squires, with different colours. Since it has a variety of colours and the digital information of these colours is defined ahead, it has been used as the device to test the result.

1.4 Overview of this Report

In the rest of the dissertation, we will introduce our solution to this problem in detail. Chapter 2 provides the background of this problem, including a general overview of the projection-based AR and the colour correction problem. All the theories and methods we used will also be introduced in this part. Chapter 3 presents our design of the solution. It describes how we analyse the darker part, how we modify the colour, and how the final result works. The fourth section shows the implementation of the solution, and how we analyse the result digitally. Chapter 5 is the evaluation part. In this part, we use the real camera and projector to do the experiment, generate the result and analysis them. In the last part, several ideas have been listed that may improve this result in the future.

Chapter 2

Background

2.1 Projection-Based AR

The projection-based AR (also called spatial augmented reality) has been presented by Bimber and Raskar (2005a). They are inspired by the CAVE VR (Cruz-Neira et al. (1993)), which uses the projector to create an immersive virtual environment without any wearable devices. This idea provides a new AR type that many things and calculations can be prepared in a prepossessing stage. So it is convenient for the well-prepared multi-people scenario like classrooms, surgery rooms, or exhibition rooms. Furthermore, projection-based AR does not suffer from many of the occlusion problems faced in other forms of AR Kern et al. (2017).

In the beginning, the projection-based AR has been implemented on a standard white background with a fixed position of the projector and the user. The light condition has also been limited due to the limitation of the light intensity from the projector. After that, much research has been done to extend the use scenario of this technology. For instance, the background may not always be a flat surface. Fujimoto et al. (2014) have used an embedded pattern marker and computer vision methods to match the display image with a deformable material. They are able to project a geometrically correct texture on a deformable surface with a simple camera and projector. So it can be used to handle a more complex background.

Currently, most projectors are using the DLP (Digital Light Processing) or LCD technology. It has a narrow depth of field. So the defocus problem always happens if we do not place the project in an appropriate place, which is not a large area. Oyamada and Saito (2008) have presented a method that is able to reduce the defocus blur of the projector, so it is more flexible to place the projector and move it while it is in use. Besides, a Laser projector can also solve this problem. To implement the projection-based AR in a large space, the idea of the multi-projector system has been presented by Bimber and Raskar

(2005b). The key problem of this system is the complexity of the geometric calibration and light measurement, especially in the overlapping area.(Hainich and Bimber (2011))

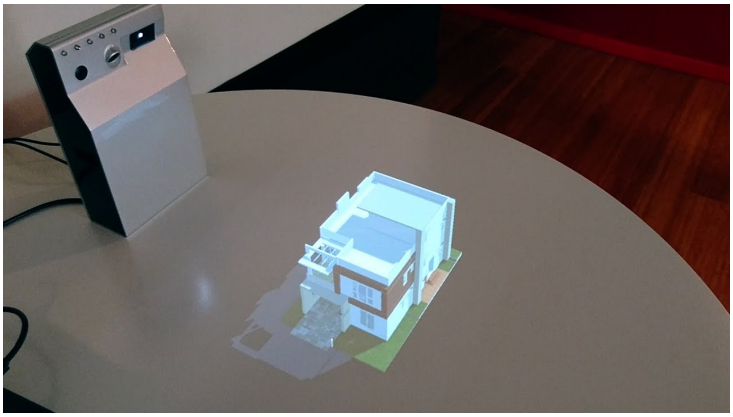

Figure 2.1: Projection-based AR example from Hololamp

2.2 Color Correction

How to display virtual things on a chromatic background has always been a complex problem. Nayar et al. (2003) has presented the problem that projects the complex image on an arbitrary surface. They create the projector-camera system to measure the process from the source image to the output. They also presented the radiometric compensation model to calculate the compensation image based on the background surface. Bimber et al. (2005) has introduced a smart projector which is able to project the image on some indoor arbitrary surfaces like papered walls or curtained windows. The texture neutralization has also involved in this scenario. To solve this problem, They use multi-cameras to measure the background colour. Then calculate the reluctance and absorption of the digital colour, and use shader to modify the display image based on it.

As for the colour correction problem in AR, Weiland et al. (2009) has introduced the colour correction problem into optical-see through AR. To solve this problem specifically for the outdoor environment, Gabbard et al. (2010) has created a testbed to simulate the outdoor background. In this implementation, the CIE LAB colour space (see section 2.3.3) has been used to calculate the colour deviation and modify the image. Based on this, Sridharan et al. (2013) has involved the colour profile method to solve this problem.

2.3 Colour Spaces

A colour space is a means of describing colours digitally. It maps the wavelength of the light to the digital values so that the computer will be able to recognise and simulate

them. Different colour spaces achieve this mapping in different ways. In this project, three colour spaces, RGB, HSV, and CIE LAB, are used in image analysis can colour transfer.

2.3.1 RGB

In RGB colour space, the colour has been separated into three channels: red, green, and blue. The value of these channels is 0 to 255. All the other colours have been represented by the mixing of these three colours and can be shown as a triangle like the left image in the figure 2.2. The way it represents colour is the closest one to the biological structure of human eyes. The human eye has three types of photoreceptors, which are approximately sensitive to red, green, and blue. In RGB colour space, the three channels are calculated by the sum of the respective sensitivity functions and the incoming light. (Tkalcic and Tasic (2003)) This colour space is a device-dependent colour space, any given RGB colour may vary across different devices. Besides, the colour change in this colour space does not follow the instinct. It is difficult to know the light and hue of the colour directly through these three channels.

2.3.2 HSV

HSV colour space also describes the colour in three channels: Hue, Saturation, and Value. Hue (H) represents the main characteristic of a colour, corresponding to different wavelengths of visible light. Saturation (S) represents the colour mixing with brightness. Value (V), also called brightness, represents the colour mixing with the darkness. This colour space is the one closest to the way the human brain recognizes colours. It means that humans tend to describe colour in hue, saturation, and brightness when they talk to each other. (Tkalcic and Tasic (2003)) This colour space can be described as the second image in figure 2.2. The circle outside is the hue channel, and the inside triangle represents the other two channels. It is also a device-dependent colour space but is convenient for measuring measure the brightness or the hue of the colour independently.

2.3.3 CIE LAB

The CIE LAB colour space (also called L$\alpha\beta$ colour space) is a device-independent colour space. Currently, the CIE LAB colour space is the most 'ideal' uniform colour space that has been used in recent research. (Kuehni (2001)) What is meant by 'uniform' is that, the perceptual colour change has a good match with the value change in the colour space. With CIE LAB colour space, it is convenient to measure the distance between colours.

Therefore, it is helpful to do the colour transfer or any other modification based on this colour space.

The last image in figure 2.2 describes how the CIE LAB colour space represents the colour. In this colour space, the L channel is the light of the colour. The α and β are the hues of the colour. The alpha represents the degree between red and green. The beta represents the degree of colour between yellow and blue. The brightness of the colour becomes darker from top to bottom, and the cross surface represents the hue.

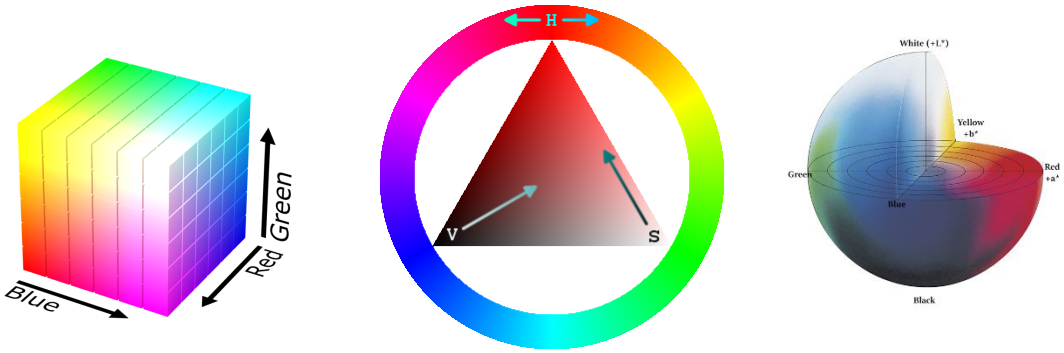

Figure 2.2: (1): RGB colour space, (2): HSV colour space, (3): CIE LAB colour space.

2.4 Color Transfer

Colour Transfer is a is an image processing technique that transfers the colour characters of a source image to a target image. This idea been presented by Reinhard et al. (2001), with the example shown as 2.3. Currently, there is four main technology that has been used to implement colour transfer: statistical information, user interaction, hybrid, and deep learning methods.(Liu (2022))

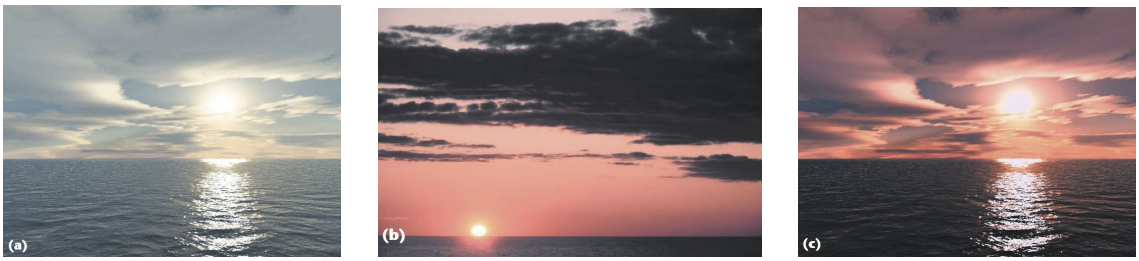

Figure 2.3: (a): source image, (b): target image, (c): result image. This is the colour transfer example from Reinhard et al. (2001). It transfer the colour style of the image (a) to image (b) and get the result (c).

The statistical information method uses statistical indexes like the mean value and the standard deviation of the image to do the calculation for three channels in the colour space. For instance, Reinhard et al. (2001) calculate the mean and the standard deviation

value of the three channels in the source image. Then apply the transform of the colour value in the target image based on it. This transform is based on the CIE LAB colour space, and Xiao and Ma (2006) have also applied it in RGB colour space. In general, this transfer does the same process for each pixel in the target image based on the statistical information from the source image and changes the overall colour gamut of the image.

The user interaction method uses strokes to mark the corresponding range between images manually. Then separate the image based on the mark, its colour, and its location. Then apply the transfer function for each corresponding range.(An and Pellacini (2010)) This method can switch parts of the image independently (like making the face whiter and making the hair darker).

Except for these two main methods, there are also many other colour transfer methods. For instance, the texture transfer is able to generate the given texture on the given surface feature. (Wei et al. (2009)), and the image tone transfers the approach style of the image. (HaCohen et al. (2011)) In recent years, deep learning methods becomes wildly used in many areas. He et al. (2017) uses Very Deep Convolutional Network (VGG), which is able to manage the distortion in edges, to do the transfer between images. Besides, convolutional Neural Networks(CNN) also perform very well in modifying the illumination of the image.(Lee and Lee (2016))

2.5 Perception of Colour

Optical illusion is one of the common phenomena that can be utilized when we think about colour problems. The perception of colour depends on more than the wavelength, intensity, and purity of the light. Most of the time, changing the surrounding colours, changing the overall scenario, and adding different occluded grids, will also affect the perceived colour.(Kitaoka (2010))

2.5.1 Lightness Constancy

When viewing objects under different lighting conditions, the human eye does not objectively perceive the absolute illumination of objects. Instead, it will automatically discount the illuminations or some other viewing conditions based on our "atmospheric transfer function". This phenomenon is called "lightness constancy". Based on this, we can easily recognize the difference between the grey surface and the white surface in a dark environment (Adelson (2000)). For instance, in figure 2.4, A and B have the same value if we represent it in colour space. However, due to our understanding of the surrounding colours and the overall scene, our eyes tells us that A is a dark block and B is a white

block. In some specific case, we can create this illusion manually, to abject the result and make the colour looks real.

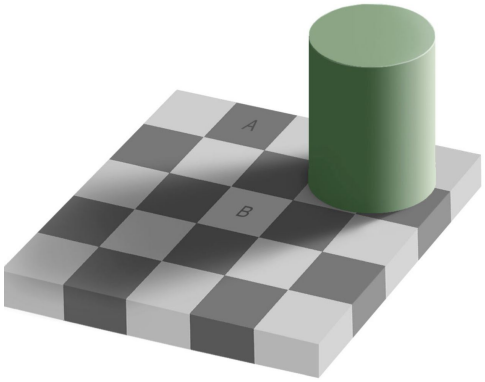

Figure 2.4: Adelson's Checker-shadow illusion(Adelson (1995))

2.5.2 Brightness Induction

Lightness constancy can be extended to the brightness induction phenomena, which describes the illusion that the perception of a grey part will be darker when the surrounding area is brighter. This phenomenon can be shown in the figure 2.5. In this image, the inner bar has a consistent colour and the surrounding areas become darker from left to right. However, the inner bar looks brighter on the left and darker on the right. This is due to the illusion of brightness induction by local contrast. (Reid Jr and Shapley (1988)) It is innately filling and does not need any inferential learning.Sinha et al. (2020)

Figure 2.5: Brightness induction effect. The inner bar is a consistent colour and the visual efficient change with the surrounding areas.

2.5.3 Colour Consistency

Colour consistency is one of these colour illusions that our eyes and brain are trying to find the "true" colour under the "filter". Figure 2.6 is one example of this phenomenon. If we measure the digital value of the pixels, The eyes colour of the three girls is the same, which is grey colour. However, due to the filter colour, the colour of these eyes looks different. The left one is light blue, the middle one is yellow, and the right one is red. This is because the human eyes and brain are trying to recover the colour under the filter. Kitaoka (2010)

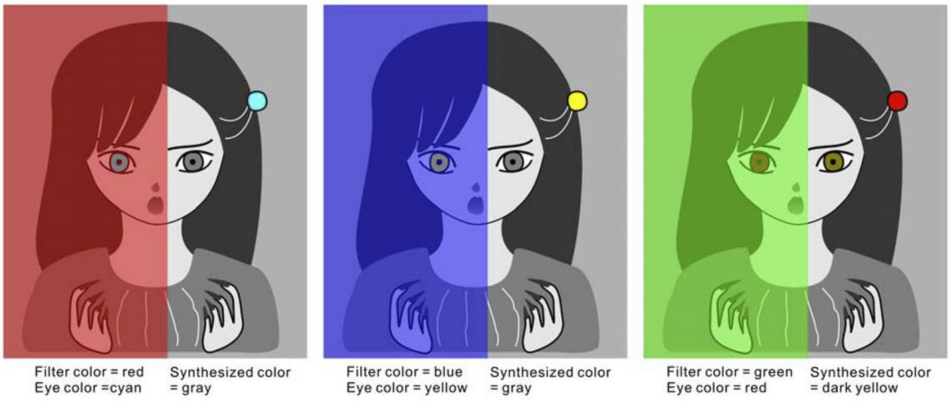

Figure 2.6: Colour consistency example form Kitaoka (2010)

2.6 Edge Detection

Edge detection is one of the most common operations in image processing and computer vision. The purpose of this opearation is to detect the huge change between the adjacent pixels. During the evolution of image process technologies, a variety of edge detection functions has been presented. (Ziou et al. (1998)) Correctly, several relevant functions are already being well prepared in some mature image process libraries like OpenCV.

The edge detection can be achieved by convolution, which combines two functions to generate another one. Normally, in the image process, one part is the matrix from the image and the other is the kernel.(Keys (1981)) The kernel will be applied to every pixel in some way, and the result will be expressed as another matrix.

The LoG (Laplacian of Gaussian) is one of the 2D kernels that can be used in edge detection. (BURT and ADELSON (1987)) The value in this kernel is approximate to the second derivative in the definition of Laplacian, which is

$$LoG(x,y) = -1/(\pi * \sigma^4) * (1 - (x^2 + y^2)/(2 * \sigma^2)) * e^{-((x^2+y^2)/(2*\sigma^2))}$$

IF we generate a kernel with size $3 * 3$, and consider 4 neighbors, the kernel is

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix} \tag{2.1}$$

And if we consider 8 neighbors, it will becomes

$$\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix} \tag{2.2}$$

Chapter 3

Design

The non-white background only reflect part of the light so they have different colours. Since the coloured background will absorb part of the light, projection on coloured or texture backgrounds leads to colour and brightness distortion and loss of perceived information. (Bimber et al. (2005)) The objective of this project is to take any input image, and apply some operations to get a result image. When project the optimal result image on the non-white background, the efficient will looks perceptually more similar to the original image. Note that although we deal with static images in this dissertation, the optimal solution, once determined could be applied to video, animations or interactive augmented reality applications

In general, our proposed image processing pipeline consists of three steps. First, process the dark part like shadows. Then, analyse the background conditions. Finally, modify the image that will be projected in the background. The reason why we choose these algorithms and how they solve the colour distortion problem will be explain in the following parts.

3.1 Dark Analysis

The dark analysis is inspired by the brightness induction rule. (Reid Jr and Shapley (1988)) In this rule, the dark part will look darker when the surroundings are lighter. Therefore, we light the surrounding areas to aggrandise the dark part. On this occasion, the shape of the displayed object and its shadow are not stable, and the darker part may locate on the border of the image. Therefore, if bright the whole image, the edge of the projected area will be obvious and affect the experience of immersion. Thus this project only lighter the edge of the darker part. The basic idea to implement this is to find the edges of the darker part and brighten them.

3.1.1 Edge Extraction

The first step to finding the edge of the darker part is to mark the area of this part. In this step, a white and black mask has been generated to mark the area. In this mask, the white part, which has the value 255, represents the darker area. The black part with the value 0 represents other areas. This mask can be generate automatically bu the given thresholds. If it cannot satisfy the user requirement, they can also generate it manually by PhotoShop and designate its pass in the computer.

Then apply the convolution function to do the edge detection of this mask. The kernel of this convolution is

$$\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix} \tag{3.1}$$

For this kernel, if all the surrounding pixels are equal to the middle one, the result will be $(-1 * x) * 8 + 8 * x = 0$, and otherwise, it will be another value. If apply this function to the mask image, the resulting image will mark the edge as white and others are black.

3.1.2 Light the Edge

The previous result is a white and black image which marks the edge of the darker part, and the goal of this step is to apply a smooth lighter part around the edges.

The idea of doing this is to modify the brightness(value) channel in HSV colour space based on the distance between the pixel and the edge. The pixels that are close to the edge will be lighter, and the pixels far away from the edge will be darker.

The ease-in and out function is an interpolation function that can be used to smooth this change. With this function, the value will change slower at the beginning and the end and change quicker in the middle. The equation of the ease-in and out function is like this:

$$Lightness = \begin{cases} l * (1 - 2 * d^2), & d < 0.5 \\ l * (-2 * d + 2)^2/2, & d \geq 0.5 \end{cases}$$

In this equation, l is the maximum brightness when brighter the edge. d is the proportion between the distance and the maximum distance that will be considered. The curve 3.1 shows how the value changes with d when $l = 1$.

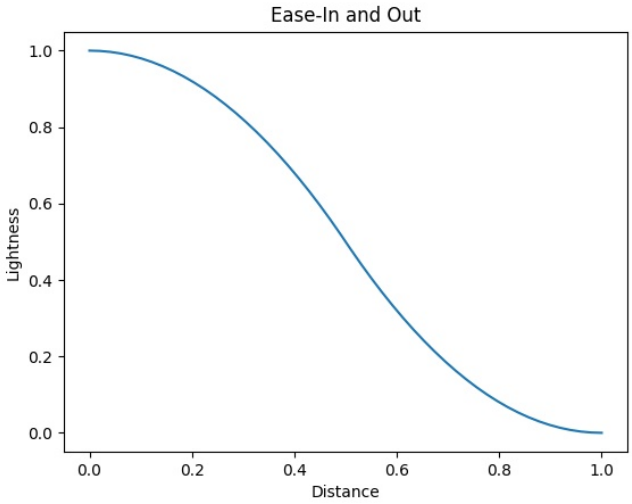

Figure 3.1: Ease-In and Out

3.2 Background Extraction

In order to adapt the projected imagery correctly, we need to have a precise understanding of how and where it will be projected. In other words, for every pixel of the input image, we essentially need to know where it will be projected in the real world. So we can know the underlying colour of the projection surface at that point, which can support pixel-to-pixel management. In the real experiment, even small changes in distance or angle of the projector can lead to a significant distortion in the projected image.Therefore we need a computational means of mapping the points in the real world environment to each pixel in the real world image.

With the current devices, the camera and the projector are independent. It means that the range that the photo contains cannot exactly match the range that the projector can project. Thus the received photo includes the needed background and some useless surrounding environment. Therefore, extracting the real background is important to do the correct analysis. The range that the projector need is defined by the projector and the size of the display image. After displaying the projector and camera in the target place, the first step is taking a photo. Then project a white image and take the photo again. The white image is the same size as the image that needed to be projected. So the part that the projector needed will be lighter than the others.

The background we need is the match the lighter part in the first image. At this time we assume that the projected image is a rectangle, so the lighter part must be a quadrilateral. The quadrilateral has been orientated by four corners. The position of these four points can be used to fund the target background in the first photo. After

that, in the case of the pixel-to-pixel analysis, we warp the background to the size of the image.

3.3 Colour Analysis

The way to analyse the colour is depend on both the background and the image itself. The first step is analysis the background that we extracted. Then there are several different algorithms to modify the display image. Based on the experiments, we can find the best match functions for different colours and analyse different parts independently.

3.3.1 Inverse Background

The inverse background is the image that has the offset colour of the background. In theory, if the background mixes with the inverse image, the result should be close to white. Then it is able to add the colour it as a white background. This step will be done in both RGB and HSV colour space and determine which one is better.

This inverse is a pixel-by-pixel operation. In RGB colour space, the value of the pixels in the inverse background is 255 minus the value in each channel. In HSV colour space, we only roll over the hue channel. The value of hue channel is $(h + (255/2))\%255$, where 255 is the maximum value in this channel. The $h + (255/2)$ is used to find the opposite hue in the circle, and then mod 255 to validate the data.

3.3.2 Pixel by Pixel Analysis

The first approach we attempted to try is simply adding the inverse background and the display image. Each pixel in the resulting image is equal to half of each pixel in the inverse background plus each pixel in the display image. The idea is combine the inverse background and the display object together. However, if we add them directly, the will be easy to above the maximum display value, which is 255 in RGB colour space. Thus we half the result to keep the value inside the range.

3.3.3 Colour Transfer

For testing purposes, we assume that the projector is always used in a stable space, so the general background texture will not change ate run time. The micro change of illumination happens at any time and anywhere. This change will still affect the result if we analyse the image pixel-by-pixel. Therefore, we invoke the statistical colour transfer methods (Reinhard et al. (2001)) to analyse the image in a general way.

There are two majority images in the colour transfer functions: source image and target image. In the statistical colour transfer algorithms, the source image provides the scene, while the target image provides the target colour distribution. The overall goal is to transfer the source image into the style of the target image. In addition, to avoid the influence of the devices, the colour transfer method is implemented by CIE LAB colour space.

The main colour transfer function we used can be shown as the equation:

$$I = (s - mean(s)) * (std(t)/std(s)) + mean(t)$$

This equation will be applied for each channel independently. In this equation, s means the value of the pixel in the source image, and t means the value in the target image. For each pixel, the value of the result is calculated by the value of the pixel in the source image and the mean and standard value in both the source image and target image. The result is equal to the source image minus its mean value, and then zoomed the deviation, map the standard deviation of the source image to the target image, and finally, add the mean of the target image.

The first and basic idea of the colour transfer is using the display image as the source image, and the inverse background as the target image. This operation transfers the overall style into the inverse background. It will cancel part of the influence of the background colour. However, based on our experiment, the inversed background image may not be a good target image. It is a solid image and its standard diversion is very small. Thus it will compress the difference between colours in the source image. The result can be shown as figure 3.2 So if the display image is colourful, it is necessary to find a better solution that can keep the colour distribution when transferring the image into the inverse colour of the background.

Figure 3.2: The colour transfer result with inverses background as the source image. image from left to right are: (1)target image, (2)source image, (3)result

There are two ways to implement this goal: change the equation and change the source image. The standard diversion is the key index that controls the colour difference. Therefore, ff changes the equation, instead of converting both the mean value and the

standard deviation value, we only convert the mean value. Otherwise, we can use the pixel-by-pixel analysis result, which is not as solid as the inverse background, as the source image to keep the colour distribution.

3.3.4 Result Mapping

In the previous steps, we already have four different algorithms to modify the display image: simple adding, basic colour transfer, modified colour transfer, and basic colour transfer with the simple adding result as the source image. Each algorithm will be implemented in both RGB and HSV colour space. To describes them easily in the following parts, we assign a simple name and an identifier number for each algorithm. They have been shown in the table 3.1

ID	Algorithm	Short Name
1	Adding the inverse background and the image in RGB	inverse RGB
2	Adding the inverse background and the image in HSV	inverse HSV
3	Basic colour transfer with inverse background in RGB	transfer RGB
4	Basic colour transfer with inverse background in HSV	transfer HSV
5	Modified colour transfer with inverse background in RGB	transfer RGB Fix
6	Modified colour transfer with inverse background in HSV	transfer HSV Fix
7	Basic colour transfer with inverse RGB as target	transfer RGB Inv
8	Basic colour transfer with inverse HSV as target	transfer HSV Inv

Table 3.1: The ID and the short mane for each algorithm.

These functions will have different performances in different backgrounds and different colour styles of the display image. With the result analysis, we can find the different best solutions for different occasions. Then, create a map to record the result. Since the illumination in the environment is uncertain, we use the hue channel in HSV colour space to categorize the background. We define the maximum and minimum hue value for each tested background colour. Then classify the background based on this range. For the display image, we will categorize it by red, green, or blue in the RGB colour space. We assume that a pixel is red if its red channel has the largest value. Count the number of pixels in each colour, and the image type is equal to the colour that the maximum pixel has.

To create this map, we need a standard colourful image to measure different colours. For this purpose we use the ColorChecker (shown in figure 3.3), a standard calibration tool used in photography, to test the algorithms. It contains abundant different colours

so it can be used to measure all three channels. Then create the baseline. To reduce the project diversion, the baseline should also be the projected result. Therefore, the baseline of these measurements is the result that projects the ColourChecker on the white background and projects the ColourChecker on the chromatic background directly.

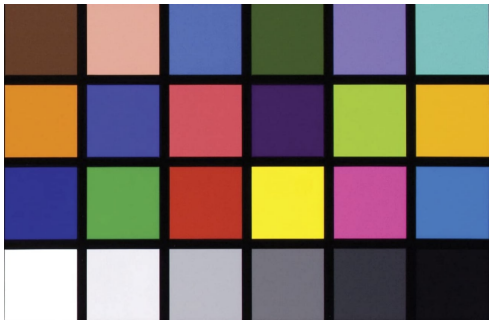

Figure 3.3: The colourChecker that is designed by McCamy et al. (1976)

The characteristics that we used to evaluate the quality of the result are the Mean Square Error (Wikipedial (2020)) and the colour distribution. The MSE is used to evaluate the difference between the two images. Currently, there are several different methods to compare two colours such as the Peak signal-to-noise ratio (Wikipedial (2021)) or the confusion matrix (School (2014)). However, some of them like psnr also considered the position of the pixels, and others like confusion matrix only counted the number of matched pixels. Therefore, the MES is the most appropriate way to evaluate the difference in the three channels independently. To compare the performance among algorithms, we create a bar chart to visualize the result. Besides, the colour distribution is also a characteristic to evaluate the degree of reduction of the image. Therefore, the distribution histogram will also be created for each result image.

3.4 Automatically Selection

In the previous step, we create a map that records the best algorithm for different occasions. With this map, we can create a new method to process the image. The first step is to recognize the background colour. Then categorize it with the existing background type in the map. Do the same operation to the display image. Access the map to find the algorithm they use, and return the process result of this algorithm.

Chapter 4

Implementation

4.1 Overview of the Solution

The main technologies we use to implement this project are Python and OpenCV. Python is a programming language that has been commonly used in many cutting-edge technologies like machine learning and computer version. It has a simple algorithm and large supporting libraries to simplify the complex code. Python is responsible for the overall process, and the image processing part relies on OpenCV. OpenCV (Open Source Computer Vision Library) is the most commonly used computer version library. It provides many useful functions that are necessary for the background analysis in AR. For instance, it can invoke the camera to read the background, converse the colour space for more complex calculations, or generate the mask with given thresholds.

The implementation has been well encapsulated, and all we need are only the code and the devices. The code is able to take the photo, do the colour transfer, calculate the relevant information to analyse the result, and automatically process the image after we define the map.

4.2 Dark Analysis

In general, the dark analysis includes generating the mask, detecting the edge, and changing the lightness.

4.2.1 Mask Generation

The mask generation can be implemented as shown in the code listing 4.1. If using HSV colour space to detect the dark space, the only channel that needed to be considered is the lightness, which is more convenient than the RGB colour space. Therefore, the first

step is using the *cvtColor*() function to convert it from RGB to HSV colour space. Then define the maximum and minimum threshold in three channels. This threshold should be defined manually since the colour of the darker range is different for different display images. Then use the *inRange*() function to generate the mask. The white part (with the value 255) is the darker range we want, and the others are the black part (with the value 0).

```
imageHSV = cv.cvtColor(image, cv.COLOR_BGR2HSV)
lower = np.array([0, 0, 10])
upper = np.array([255, 255, 120])
shadowMask = cv.inRange(imageHSV, lower, upper)
```

Listing 4.1: Generate the mask

4.2.2 Edge Detection

The OpenCV library has excellent support for convolution functions (Keys (1981)). See the code listing in 4.6, after define the kernel, the $filter2D()$ function can process the convolution function and return the result image directly.

```
kernel = np.array([[-1, -1, -1], [-1, 8, -1], [-1, -1, -1]])
result_image = cv.filter2D(shadowMask, -1, kernel)
```

Listing 4.2: Detect the edge of the image based on convolution.

4.2.3 Modify Lightness

The code 4.3 is the majority part of lighting the edge of the darker part. If we want to apply the ease-in and out function, the first step is to define the number of pixels that will be lighter manually. This value is necessary to measure the ratio between the distance and the maximum distance. Then, traverse all the pixels in the edge detection result. If the pixel is on the edge, modify the saturation and the brightness channel of its surrounding pixels. For each pixel, we modify 5 pixels around it, and the value is calculated by the ease-in and out functions.

```
numOfPixel = 5
for i in range(0, height):
  for j in range(0, width):
    if (result_image[i][j] > 0):
      for x in range(0, numOfPixel):
```

```
        result = x / numOfPixel
        if result < 0.5:
          result = 1 - 2 * result * result
        else:
          result = (-2 * result + 2) **2 / 2
        result = result * 200
        if(result > 50):
          if(imageHSV[i + x][j][2] < result):
            imageHSV[i + x][j][1] = 0
            imageHSV[i + x][j][2] = result
          if(imageHSV[i - x][j][2] < result):
            ......
```

Listing 4.3: Modify the lightness of the image with ease in and out.

4.3 Camera Invocation

The camera has been invoked by OpenCV. OpenCV provides the video capture function to invoke the camera, including both the built-in camera and the external camera. In this implementation, a photo is enough to do the analysis. Therefore, we invoke the camera by the video capture function and only get one frame. The code segment in 4.4 shows how the photo has been taken, including defining a function to return this frame and using the *imwrite*() function to store the image in the target place.

```
def readBackground():
        vid = cv.VideoCapture(1)
        ret, frame = vid.read()
        vid.release()
        return frame

image = readBackground()
cv.imwrite(path + "name.jpg", image)
```

Listing 4.4: Code to invoke the camera

4.4 Background Extraction

Two photos are needed to be taken before invoking the background extraction functions. One is take the photo to the background directly, and the other is the background with the projector projecting the white image. The goal of this function is to fund the corresponding area of the projected place in the pure background photo and generate an analysable image of the background that has the same size as the projected image.

The background extraction function also needs a mask to mark the lighter part. It has a similar process to the mask generation in the previous step. The only difference is that the lower and upper thresholds are generated by the image. Assume that the corner of the photo is the darker part and the centre of the photo is the lighter part. The threshold is the middle brightness of these two points.

Since the shape of the lighter part is approximate to a quadrangle, the way to find the coordinate of the four corners. The main idea of the function can be shown in code segment 4.5. The first step is finding the center of the minimum bounding rectangle by *findContours*() function and *minAreaRect*() function. These points must be inside the lighter part. Then, use a loop to find the furthest light points from the centre in four directions. These four points are the four corners of the quadrangle so we return the list of these points.

```
def extractMask(mask):
    contours, hierarchy =
        cv.findContours(
            mask, cv.RETR_TREE, cv.CHAIN_APPROX_SIMPLE)
    rect = cv.minAreaRect(np.vstack(contours).squeeze())
    # top left point
    topLeft = [(int)(rect[0][1]), (int)(rect[0][0])]
    while(mask[topLeft[0] - step][topLeft[1]] == 255 or
          mask[topLeft[0]][topLeft[1] - step] == 255):
        if(mask[topLeft[0] - step][topLeft[1]] == 255):
            topLeft[0] = topLeft[0] - 1
        if(mask[topLeft[0]][topLeft[1] - step] == 255):
            topLeft[1] = topLeft[1] - 1
    # top right point
    # bottom left point
    # bottom right point
    ......
    points = np.array(
```

```
            [topLeft, topRight, bottomRight, bottomLeft])
    return points
```

Listing 4.5: Find the position of the four corners

Then use the functions in OpenCV to warp the background. First, match the position of the four corners, and the target place that we want to display in the new image. This target place is equal to the size of the display image on this occasion. Then use the *getPerspectiveTransform*() function to calculate the projection matrix, and use this matrix to transfer the pixels to their new coordinates. The code has been shown in code segment 4.6.

```
def extractRange(image, points, width, height):
    points = np.float32(points)
    target = np.float32([
                [0, 0], [width, 0], [width, height], [0, height]
            ])
    matrix = cv.getPerspectiveTransform(points, target)
    imgOutput = cv.warpPerspective(
                    image, matrix, (width, height))
    return imgOutput
```

Listing 4.6: Detect the edge of the image based on convolution.

4.5 Image Modification

The essence implementation of all the image processing algorithms is image modification. All these image modifications have the same process: convert the colour space into the type we want, traverse each pixel and apply the equation for each channel, check the boundary, and convert back to the RGB colour space.

The colour transfer function 4.7 is a standard example of this process. First, convert the image into LAB colour space. Then traverse each pixel and each channel in the image. Doing the calculation, and if the value is lower than 0 or larger than 255, limit the value to 0 and 255. Finally, convert it back to RGB colour space and return the image.

```
def colorTransfer(source, target):
    source = cv.cvtColor(source, cv.COLOR_BGR2LAB)
    target = cv.cvtColor(target, cv.COLOR_BGR2LAB)
    s_mean, s_std = get_mean_and_std(source)
    t_mean, t_std = get_mean_and_std(target)
```

```
    height, width, channel = source.shape
    for i in range(0, height):
        for j in range(0, width):
            for k in range(0 ,channel):
                value = source[i, j, k]
                value = ((value - s_mean[k]) *
                    (t_std[k] / s_std[k])) + t_mean[k]
                value = int(x)
                value = 0 if value < 0 else value
                value = 255 if value > 255 else value
                source[i,j,k] = value
    return cv.cvtColor(source,cv.COLOR_LAB2BGR)
```

Listing 4.7: Colour transfer function

4.6 Result Mapping

After all these algorithms have been implemented, the last step is to generate the final map. To do this, we need to analyse the result for different algorithms and different backgrounds and find the best algorithm for the specific occasion.

To analyse this, we first project our result on the background, take the photo, and extract the range we need. The result extraction is similar to the background extraction. Then we calculate the mean squire error and the colour distribution, draw the diagram based on it, analyse it manually, and create the map to match the situation and the algorithm. Then, when receiving a new background or a new display image, we can use this map to find the best algorithm and modify the display image.

4.6.1 Mean square Error

The first measure we use to analyse the result is the mean square error. We will calculate it in R, G, and B channels independently. The standard image is the result that projected the colour checker on a white background. Compare the results and the standard image, calculate the sum of the squire of the difference for each pixel, and divide by the number of the pixel. This process has been encapsulation as 4.8 function, processing it for all the result images independently, and using the python *matplotlib* library to draw the bar chart for them.

```
def compairImage(image1, image2):
```

```
    r1 = image1[:,:,2]
    r2 = image2[:,:,2]
    ......
    diffR = 0
    ......
    height, width = b1.shape
    for i in range(0, height):
        for j in range(0, width):
            diffR = diffR + (int(r1[i][j]) - int(r2[i][j])) ** 2
            ......
    MSER = diffR/(height * width)
    ......
    return [MSER, MSEG, MSEB]
```

Listing 4.8: Mean Squire Error

4.6.2 Colour Distribution

The colour distribution has been represented by a colour histogram. The x-axis of the histogram is the colour value from 0 to 255, and the y-axis is the number of pixels that have this value. The way to implement it can be shown as code 4.9. The *calcHist*() function is able to count the number of pixels in each value, and the *matplotlib* library has been used to draw and store the image.

```
hist_b = cv.calcHist([image], [0], None, [256], [0, 256])
hist_g = cv.calcHist([image], [1], None, [256], [0, 256])
hist_r = cv.calcHist([image], [2], None, [256], [0, 256])
plt.plot(hist_b, color='b')
plt.plot(hist_g, color='g')
plt.plot(hist_r, color='r')
plt.legend(['blue', 'green', 'red'])
plt.savefig(path + 'Barchart.jpg')
plt.show()
```

Listing 4.9: Colour Distribution Histogram

4.6.3 Automatically Selection

The last step, also the main step of this part is the map generation and invoke. With the analysis of the MSE and the colour distribution, we can find the best solution for a specific background colour and a specific style of the display image. Thus the only thing to do before the image process is to recognize the background type and the image style.

The first step is background recognition. This step recognizes the category of the colour of the background. Since the illumination is floating and unable to control, the only character that will be considered is the hue. Therefore, the HSV colour space has been used on this occasion.

The implementation of background recognition has been shown in listing 4.10. First we define the range of the hue value for different colours. For instance, the hue value for yellow is 25 to 45. Then, generate the mask to mark the pixels inside this range. Then count the number of pixels that have this colour. Do the same process for all the colours of the background we tested. If two-thirds of the pixels are the same colour, we can confirm that the background is this type. If the background is not followed any type of colour range, we will define it as a chromatic background.

```
# yellow part
lower = np.array([25, 0, 0])
upper = np.array([45, 255, 255])
yellowMask = cv.inRange(background, lower, upper)
list = [i for j in yellowMask for i in j]
yellowCount = list.count(255)
# green part
......
# purple part
......
if(yellowCount > (2 * size / 3)):
    backgroundType = 1
elif(greenCount > (2 * size / 3)):
    ......
```

Listing 4.10: Mark the type of the background

Then analyse the display image. Since the previous analysis is in RGB colour space, this step also uses this colour space and will define the image as red style, green style, or blue style. The way to define the colour style of the image is based on the number of pixels in each colour. For each pixel, find which channel has the largest value. Assume that the style of the pixel belongs to the colour that has the largest value. Then count the

number of pixels in each channel. The style of the image has been defined as the colour which has the maximum pixels. The way to implement this has been shown in listing 4.11.

```
BGRvalue = [0, 0, 0]
for i in range(0, height):
    for j in range(0, width):
        colour = maxColor(image[i][j])
        BGRvalue[colour] = BGRvalue[colour] + 1
imageType = maxColor(BGRvalue)
```

Listing 4.11: Analysis the style of the display image.

Finally, we match the occasion and the algorithm. In the previous steps, the type of the background and the image has already been defined. Thus the only thing left is to invoke the algorithm. Since the previous code is already well encapsulated, the only thing left is to invoke the algorithm with the defined parameters. See code 4.12, use *if* sentence to invoke the algorithm based on the type of background and the display image.

```
if(backgroundType == 0):
    if(imageType == 0):
        result = inverseAndObject(image, RGBinverse)
    elif(imageType == 1):
        ......
elif(backgroundType == 1):
    if(imageType == 0):
    ......
```

Listing 4.12: Map the occasion and the algorithm.

Chapter 5

Evaluation

5.1 Experiments Overview

The experiment included five steps: dark analysis, background extraction, image process, result analysis, and automatically selection. The dark analysis is an independent part of the analysis the dark part. The background extraction, image process, and result analysis are three steps that analyse the colour and define the map. The last part, automatically selection, using the map the generate ahead and display the result generated from our solution.

5.2 Dark Analysis

The test image for dark analysis is a text with shadow. The figure 5.1 is the test image we used. Then we remove the white background of the image. Since the transparent background is difficult to do the following process, and project black has the same effect as project transparent, we switch the background to black.

Figure 5.1: The image to test shadow.(1):image with white background, (2):image with black background

Then do the image processing. With the proper threshold, the shadow range has been

recognized properly. This mask has been shown in the left image of the figure 5.2. Due to the occluded, the shadow does not have the same shape as the text itself. The right image is the edge of this area.

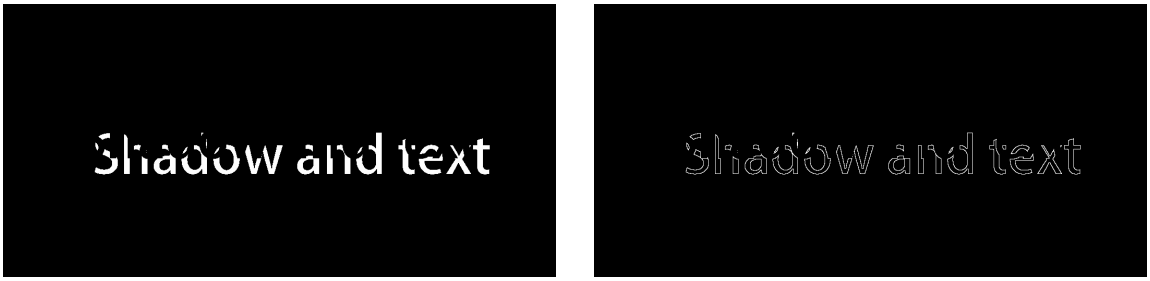

Figure 5.2: (1):the generated mask for the shadow part, (2):the edge of the shadow that generated from the mask

After all the processes, the result will be like 5.3. Here we

Figure 5.3: The shadow analysis result. The shadow has a lighter surroundings.

The figure 5.4 shows the effect of this process. The left image is the photo that projects the original image on the white background, and the right one is projecting the analysis result. The shadow in the left image is weak but it can be found clearly in the right image.

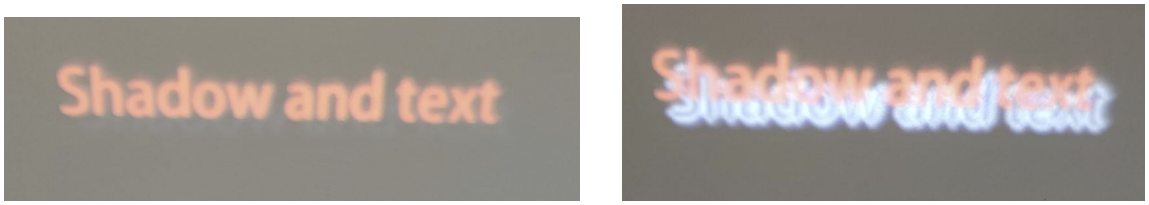

Figure 5.4: (1):project the original shadow image, (2) project the modified shadow image

5.3 Colour Analysis

In this experiment, four different backgrounds: purple, yellow, green, and chromatic have been used. In the background extraction and image processing part, the purple one will be used as an example to explain the process of the whole experiment. However, in the last part, all the results will be considered, and give a table to map all different background colours.

5.3.1 Background Extraction

The process of background extraction has been shown in figure 5.9. The first image is the photo that projects the white image. The second one is the mask that marks the lighter part. The third image marks the centre of this range, the four corners, and the quadrangle that will be extracted. Then apply it to the fourth image, which is the photo of the background. The actuarial range that will be extracted from the background has been shown in the fifth image. After warping it to the size we want, image six is the final background we received.

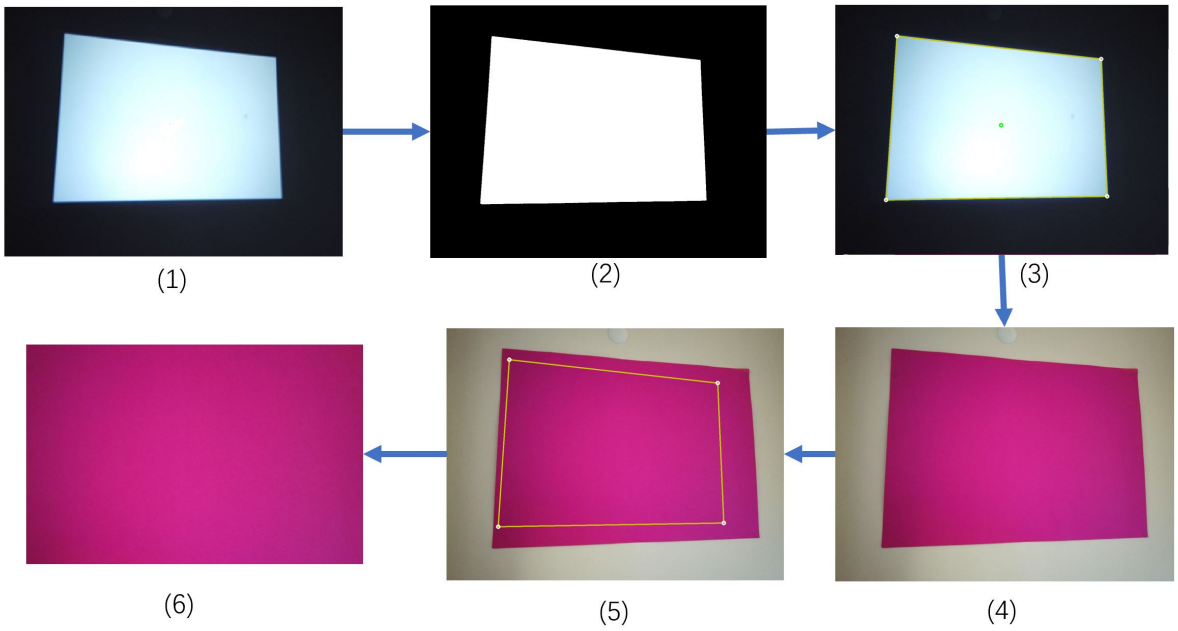

Figure 5.5: The flow of the background extraction. (1):background with the lighter part, (2):mask for the lighter part, (3):center and corner of this range, (4):background, (5):the selected range on the background, (6):the result

5.3.2 Image Processing

The first step in the image process is to calculate the inverse image. For the purple background like the first image in 5.6, the inverse image that is calculated by RGB colour space is the next green image in the figure 5.6, and the third. one is calculated in HSV colour space. Due to different computations and colour space, two different but reasonable inverse image has been received and used in the following process.

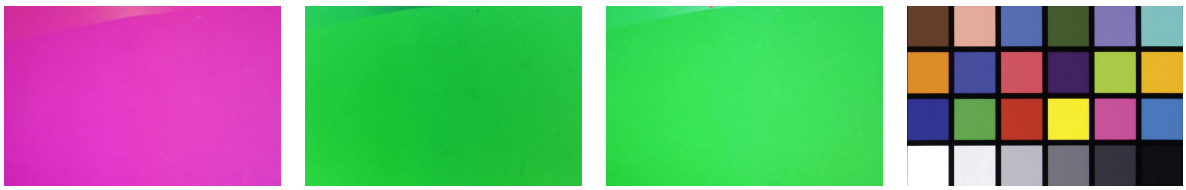

Figure 5.6: (1):background, (2):the inverse background in RGB colour space, (3):the inverse background in HSV colour space, (4):the test image, which is colourChecker now

The last image in 5.6 is the colour checker. This is the test image we use this time, so it is the based image that needed to be modified.

The figure 5.7 and 5.8 are the modified image. The figure 5.7 are using the colour checker and the RGB inverse background, and the 5.8 are using the HSV inverse image. The image from left to right are using: a simple adding algorithm, basic colour transfer algorithm, fixed colour transfer algorithm, and colour transfer which uses the first image as the source image.

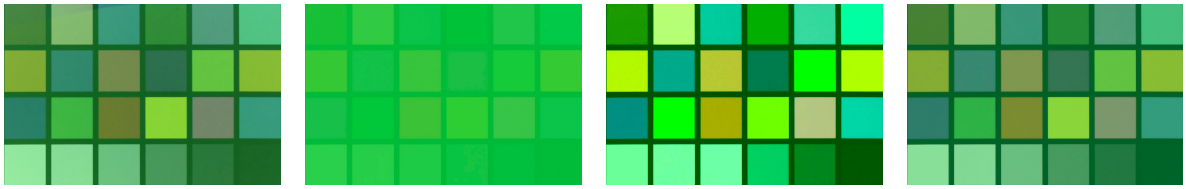

Figure 5.7: Colour process with RGB colour space in four algorithms. (1):inverse RGB result, (2):transfer RGB result, (3):transfer RGB fix result, (4): transfer RGB Inv result

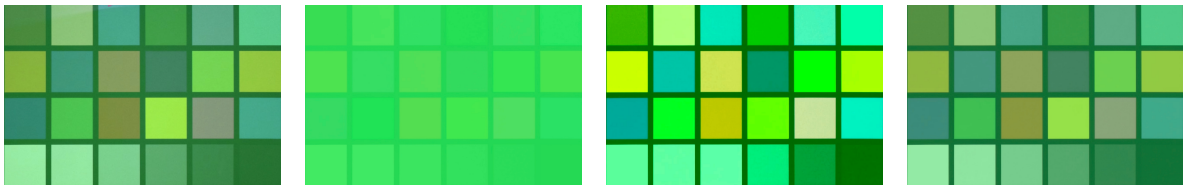

Figure 5.8: Colour process with HSV colour space in four algorithms. (1):inverse HSV result, (2):transfer HSV result, (3):transfer HSV fix result, (4): transfer HSV Inv result

5.3.3 Result Analysis

The first step of result analysis is result extraction. Since the photo is not the real projected space, we need to extract the projected range before doing the analysis. This process can be done as the same operation as the background extraction. Figure 5.9 shows the photo we that when project the colour checker on white background, extract the range and wart it into the size we want.

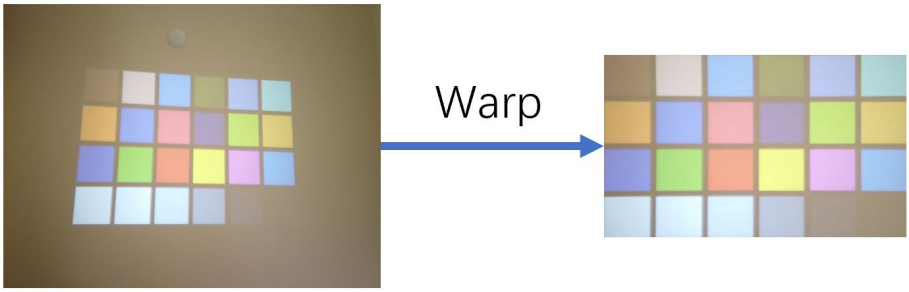

Figure 5.9: Extract the real space before analysis

Then, project all the results on its background, take the photo, and warp the result part. All the warped results have been shown in figure 5.10. The top two images are projecting the colour checker on white background, and projecting the colour checker directly on the purple background. The white one is the target, and the purple one is the baseline. The following two rows are projecting the result images on the purple background. The first line are using RGB colour space and the second one are using HSV colour space.

Then analysis the results we received for all four backgrounds. The backgrounds we used have been shown in 5.11.

Figure 5.11: The backgrounds that have been used in this experiment. (1):purple background, (2):green background, (3):yellow background, (4)texture background

Then we use the mean squire error to evaluate the difference between the target and the results. The R, G, and B channels are calculated independently. For every diagram, the most left bars are the baseline. The following bars are: the simple adding function in

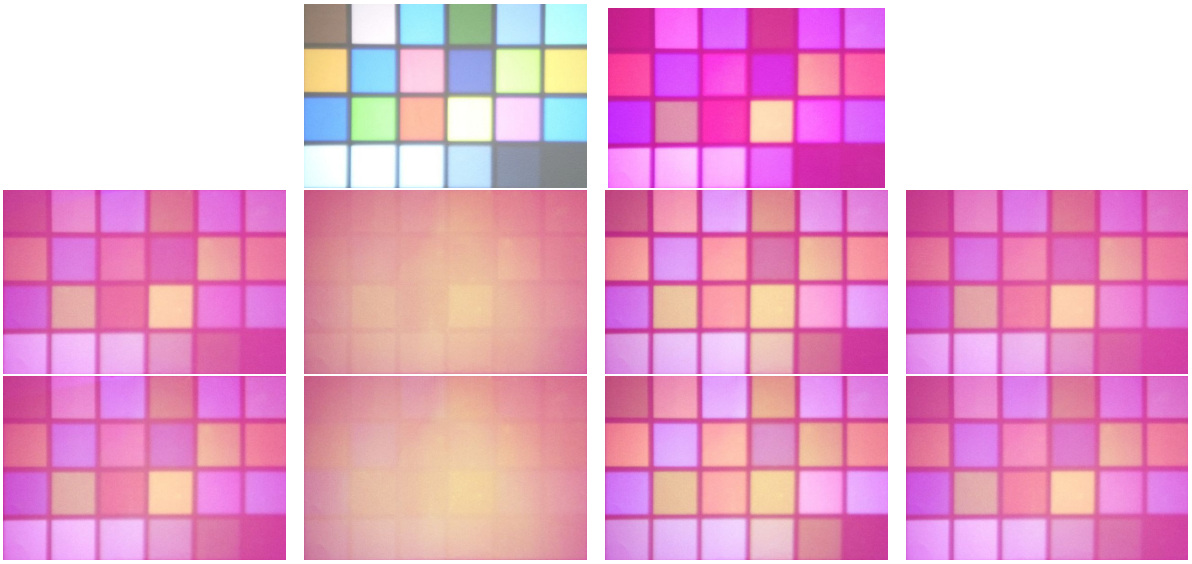

Figure 5.10: The image that projects the colour checker on the white background, and the images that project the baseline and all the results on the purple background. Extract the range we needed, and warp them into the same size. Images on the first line: target result and baseline. Image on the second line: (1):inverse RGB, (2):transfer RGB, (3):transfer RGB fix, (4):transfer RGB Inv. Image on the third line: (1):inverse HSV, (2):transfer HSV, (3):transfer HSV fix, (4):transfer HSV Inv

RGB and HSV colour space, the basic colour transfer function in RGB and HSV colour space, the colour transfer function which only considers the mean value in RGB and HSV colour space, and the colour transfer function with the adding result as the source image in RGB and HSV colour space,

The figure 5.12 shows the mean squire error for the purple background. Based on this diagram, we can find that the most affected channel is the green channel. The purple colour is absorbing the green and reflects the red and blue so it is a reasonable result. The result for red and blue channel are similar as the baseline in all the algorithms except the basic function for both RGB and HSV colour space. As for the green channel, the modified transfer function in HSV colour space works the best. Therefore, the modified transfer function in HSV colour will be used in any display image if the background is close to purple.

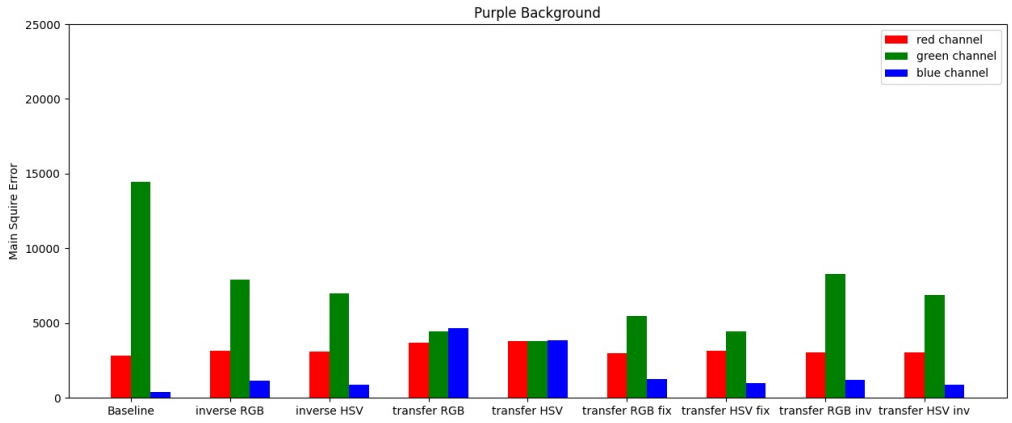

Figure 5.12: The bar chart of MSE for purple background

The figure 5.13 shows the mean squire error for the green background. For this colour, most of the algorithms work well in the green channel because the background reflects most of the green wave. Besides, the modified transfer function in HSV colour space works well in both red channel and blue channel.

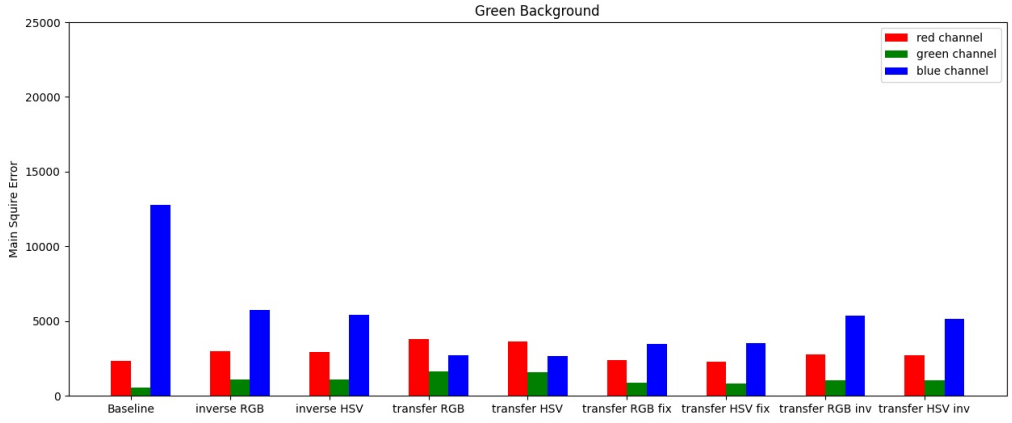

Figure 5.13: The bar chart of MSE for green background

The result for yellow has been shown in figure 5.14. The blue channel is the worst channel for yellow background. All the algorithms make the red and green channel worst than the baseline but greatly improve the blue channel. Among all these algorithms, the modified colour transfer function in HSV colour space works best on blue channel, and the RGB colour space for same algorithm is better for red and green channel.

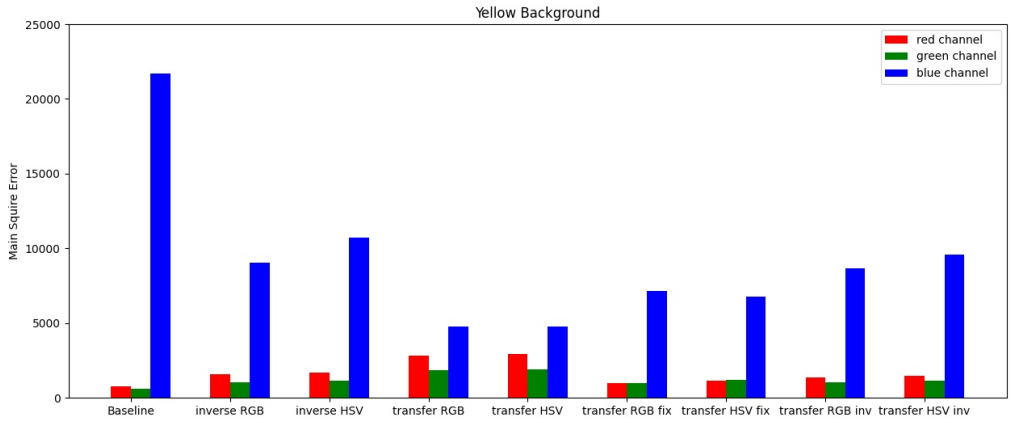

Figure 5.14: The bar chart of MSE for yellow background

For the chromatic background as figure 5.15, only adding function works better than the baseline. This is probably because the mean value has been affected by different colours, and there does not have the overall colour style of the background. So the colour transfer function cannot work well.

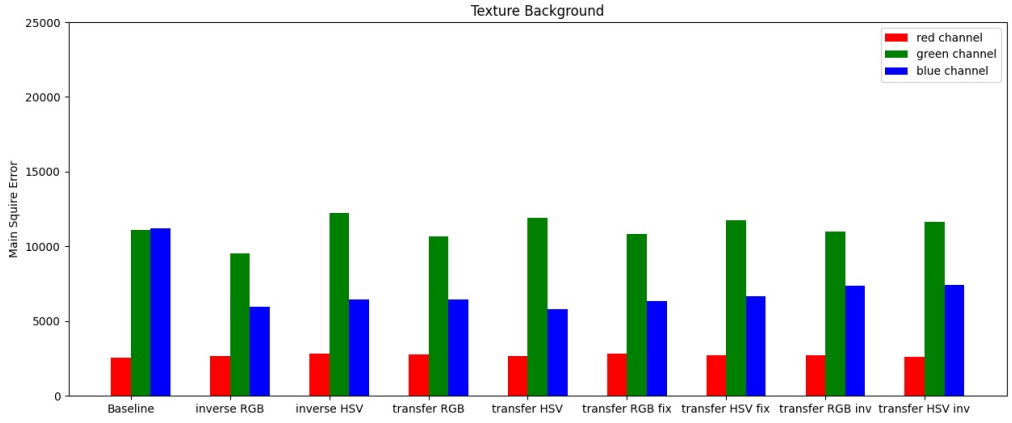

Figure 5.15: The bar chart of MSE for texture background

Then we consider the colour distribution. The figure 5.16 shows the colour distribution for the projected result and purple image. The more it is close to the baseline, the better result is. For the purple background, the RGB colour space is better than the HSV colour space. Besides, the basic transfer function will reduce the colour difference since it reduces the floating of the colour distribution.

In figure 5.17, we can find that the RGB and HSV colour space have similar performance. The basic transfer function will reduce the colour difference. This conclusion is

also shown in the yellow result in figure 5.18. However, for chromatic background 5.19, the colour distribution is more close to the baseline rather than the target.

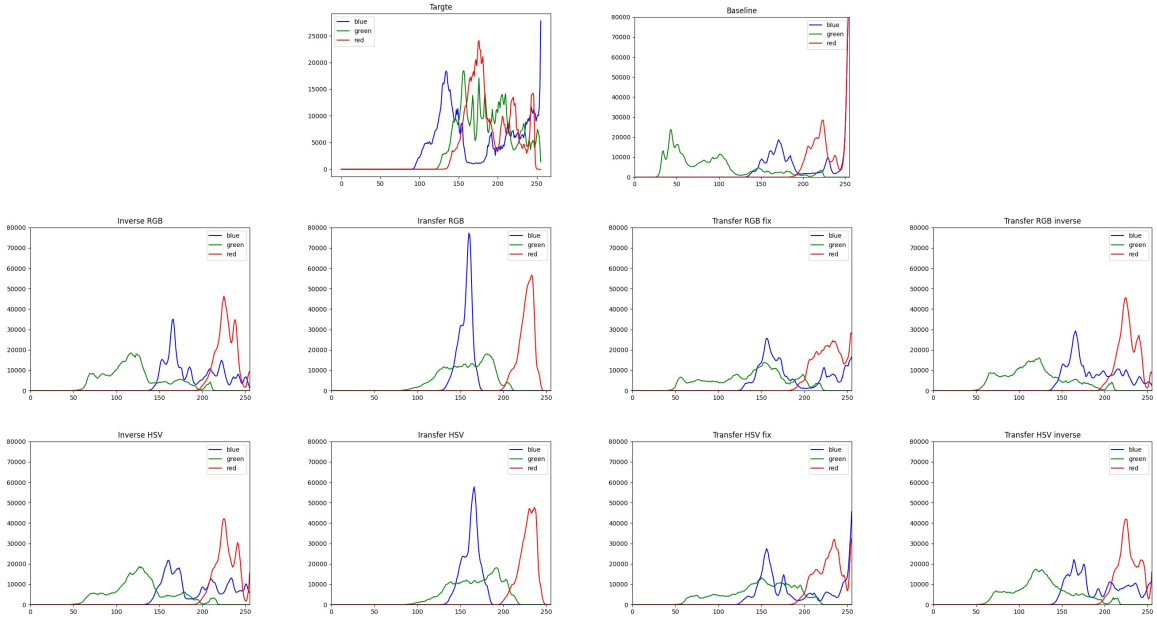

Figure 5.16: Colour distribution for purple background

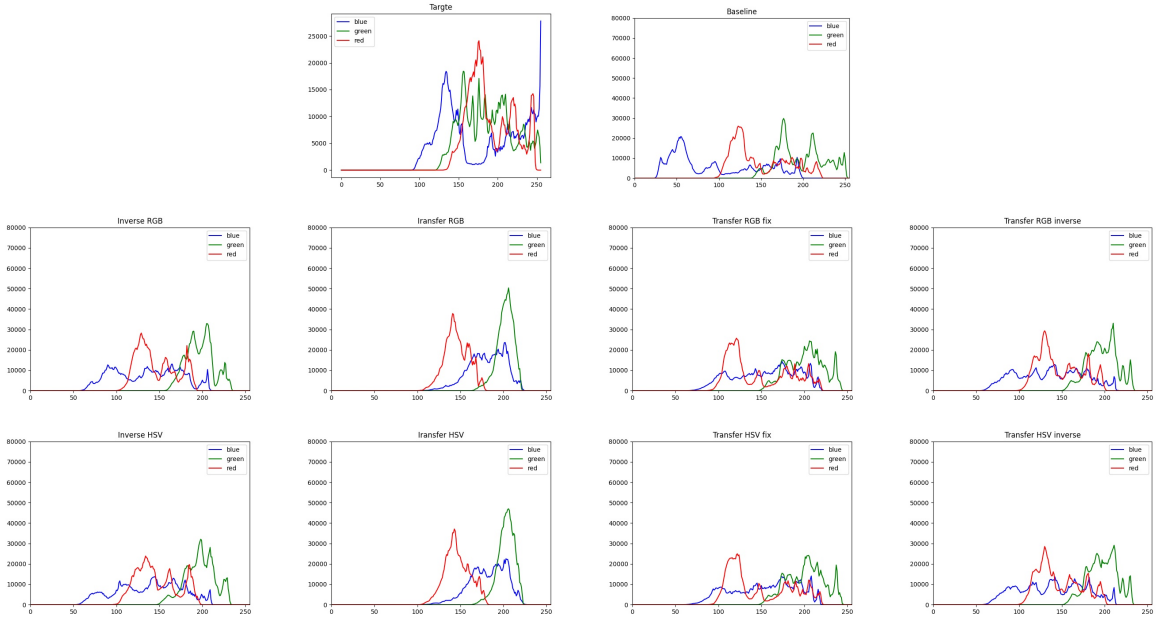

Figure 5.17: Colour distribution for green background

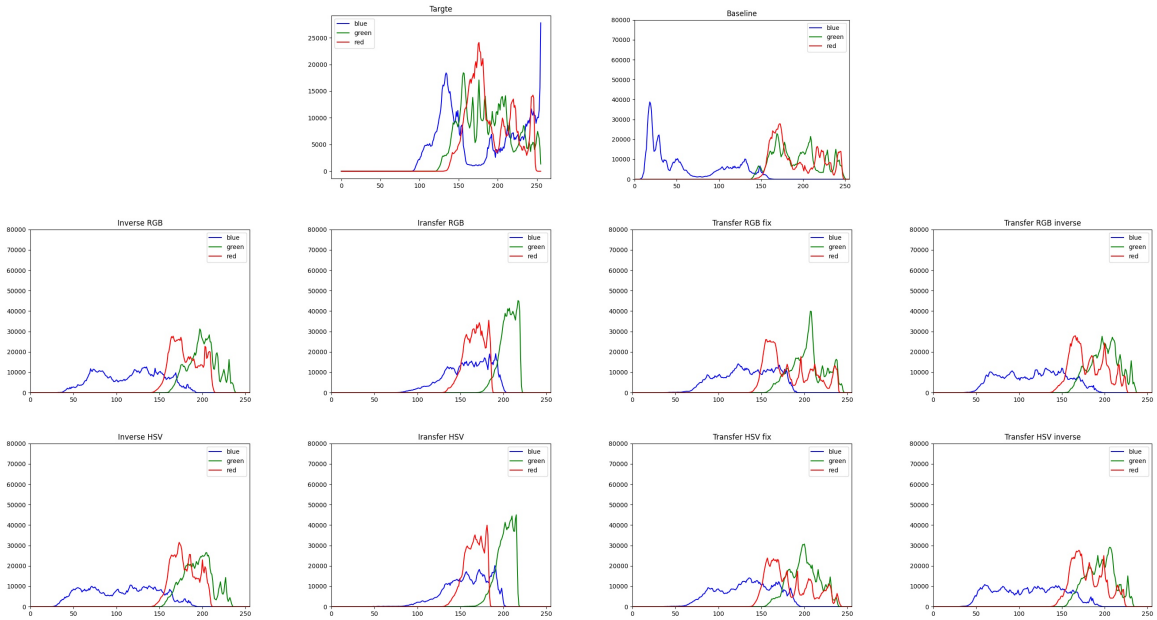

Figure 5.18: Colour distribution for yellow background

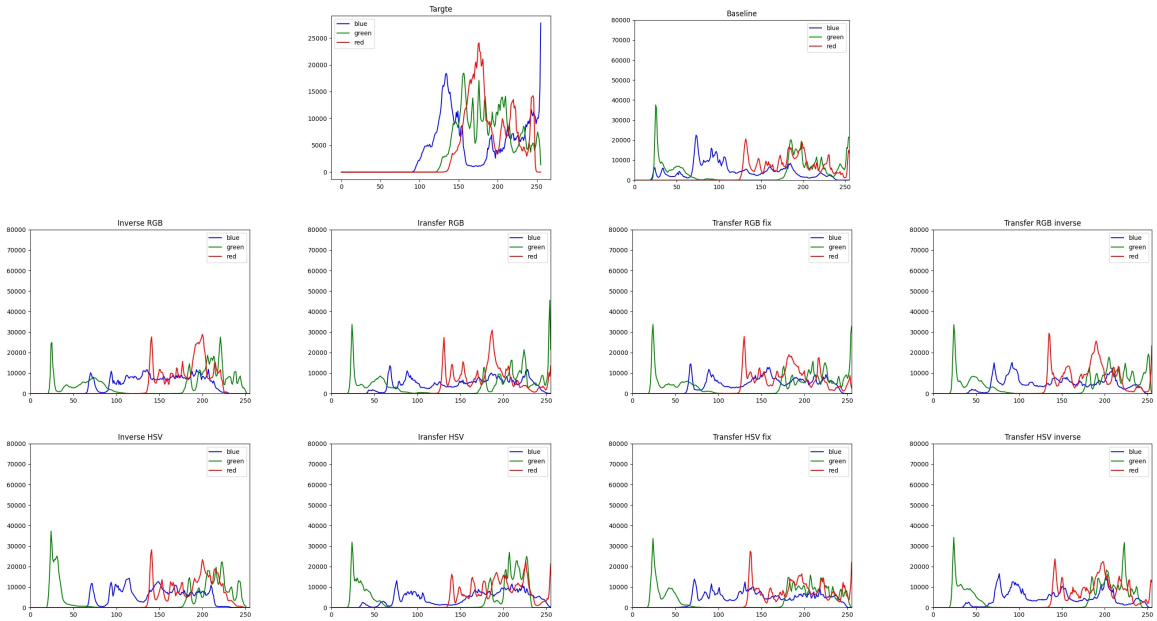

Figure 5.19: Colour distribution for chromatic background

After all these analyse, the occasion and the corresponding algorithms have been shown in the table 5.1. The ID of each function has been shown in table 3.1. The modified colour transfer function is the best solution for most solid backgrounds, and the pixel-by-pixel analysis works better for the texture background.

	Purple	**Green**	**Yellow**	**Texture**
Red	6	6	5	1
Green	6	6	5	1
Blue	6	6	6	1

Table 5.1: The algorithms we used on different occasion

5.4 Automatically Selection & Result

Finally, we use the rule we find to process the images. We use the cake image as an example. The left image in 5.20 is the original image, and the right one is the modified result on purple background. The 5.21 shows how this result has been displayed on the purple background.

Figure 5.20: (1):original cake image, (2) result cake image

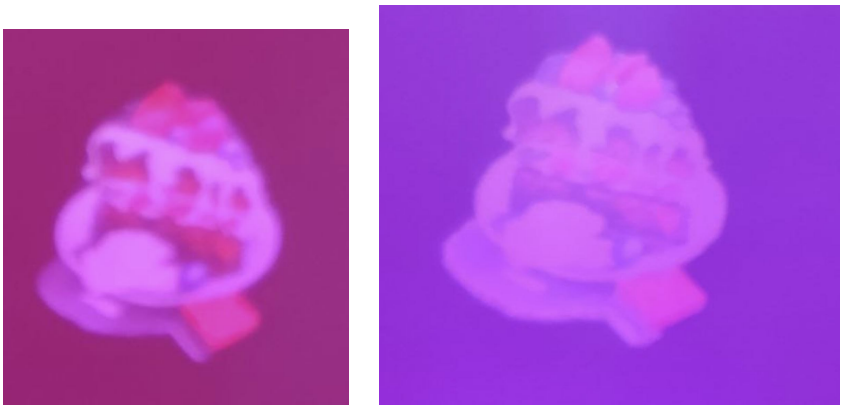

Figure 5.21: Projected cake image

5.5 Performance Analysis

The solution was tested on a Thinkpad X1 laptop with a Intel(R) Core(TM) i7-8750H 2.20GHz CPU, running windows and Python version 3.6 with OpenCV version 2.0. The table 5.2 shows the running time for the preprocess and each algorithm.

Algorithm	**Time**
preprocess	6.2s
inverse RGB	24.5s
inverse HSV	22.9s
transfer RGB	18.4s
transfer HSV	17.3s
transfer RGB Fix	20.2s
transfer HSV Fix	22.5s
transfer RGB Inv	35.2s
transfer HSV Inv	34.4s

Table 5.2: Time cost for different algorithms.

Chapter 6

Conclusions & Future Work

6.1 Conclusion

In conclusion, this project provides a colour correction solution for projection-based AR. This solution contains the shadow analysis and colour correction. For the shadow, we lighter its edge to make the dark part clear. To implement, a mask for the shadow is needed. Then, use the convolution function to detect the edge, and use the ease in and out interpolation function to smooth the change of the brightness. This implementation has been done in HSV colour space.

For the colour correction algorithm, the colour transfer methods have been involved. The colour transfer methods use the statistic indexes of the image like the mean value to modify the colour style of the source image to the target image. For the given background, we first calculate its inverse image pixel by pixel in both RGB and HSV colour space. Then generate the simple adding between the object and the inverse image. Try different target image and colour transfer algorithms, and calculate the mean squire error and colour distribution to measure the result.

In the beginning, the colour checker has been used as the test object. Using the colour checker to test different backgrounds and different algorithms. Then figure out the best algorithm for different backgrounds, and create the map to record it. Finally, when doing it in a new environment, we can recognize the background colour and the condition of the display image, find the best algorithm on the map and manage the display image.

6.2 Future Work

The environment illumination is one of the important conditions that influence projection-based AR. In this project, we do not have the condition to control the environment

illumination. The only thing we can do is use the natural light and test the illumination before doing the experiment, try to keep it to about 40 lx when doing the experiment. Besides, this illumination is also not a completely parallel light. However, there are many relevant problems that are affected by the illumination. For instance, will the best algorithm change in a lighter or darker environment? Is the algorithm still work in a glare environment or a gloomy environment? Will the uneven light affect the result? Due to the limitation of time and devices, these problems remain to be solved.

Due to the visual illusion, especially the colour consistency, the actual visual effects may be different from the digital analysis result. However, due to the COVID-19 and time limitations, we cannot execute the in-person test with the interview to detect the real visual effects.

Bibliography

Adelson, E. H. (1995). Adelson's checker-shadow illusion. `http://persci.mit.edu/people/adelson/checkershadow_illusion`.

Adelson, E. H. (2000). 24 lightness perception and lightness illusions. *The new cognitive neurosciences*, page 339.

An, X. and Pellacini, F. (2010). User-controllable color transfer. In *Computer Graphics Forum*, volume 29, pages 263–271. Wiley Online Library.

Bimber, O., Emmerling, A., and Klemmer, T. (2005). Embedded entertainment with smart projectors. *Computer*, 38(1):48–55.

Bimber, O. and Raskar, R. (2005a). *Spatial augmented reality: merging real and virtual worlds*. CRC press.

Bimber, O. and Raskar, R. (2005b). *Spatial augmented reality: merging real and virtual worlds*. CRC press.

BURT, P. J. and ADELSON, E. H. (1987). The laplacian pyramid as a compact image code. In Fischler, M. A. and Firschein, O., editors, *Readings in Computer Vision*, pages 671–679. Morgan Kaufmann, San Francisco (CA).

Chun Wei Ooi, J. D. (2022). Perceptually enhanced shadows for ost ar. *International Workshop on Immersive Mixed and Virtual Environment Systems (MMVE'22)*.

Cruz-Neira, C., Sandin, D. J., and DeFanti, T. A. (1993). Surround-screen projection-based virtual reality: the design and implementation of the cave. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 135–142.

Edgcumbe, P., Singla, R., Pratt, P., Schneider, C., Nguan, C., and Rohling, R. (2018). Follow the light: projector-based augmented reality intracorporeal system for laparoscopic surgery. *Journal of Medical Imaging*, 5(2):021216.

Fujimoto, Y., Smith, R. T., Taketomi, T., Yamamoto, G., Miyazaki, J., Kato, H., and Thomas, B. H. (2014). Geometrically-correct projection-based texture mapping onto a deformable object. *IEEE Transactions on Visualization and Computer Graphics*, 20(4):540–549.

Gabbard, J. L., Swan, J. E., Zedlitz, J., and Winchester, W. W. (2010). More than meets the eye: An engineering study to empirically examine the blending of real and virtual color spaces. In *2010 IEEE Virtual Reality Conference (VR)*, pages 79–86. IEEE.

HaCohen, Y., Shechtman, E., Goldman, D. B., and Lischinski, D. (2011). Non-rigid dense correspondence with applications for image enhancement. *ACM Trans. Graph.*, 30(4).

Hainich, R. R. and Bimber, O. (2011). Displays: Fundamentals and applications.

He, M., Liao, J., Yuan, L., and Sander, P. V. (2017). Neural color transfer between images. *arXiv preprint arXiv:1710.00756*, 2.

Kern, J., Weinmann, M., and Wursthorn, S. (2017). Projector-based augmented reality for quality inspection of scanned objects. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, 4.

Keys, R. (1981). Cubic convolution interpolation for digital image processing. *IEEE transactions on acoustics, speech, and signal processing*, 29(6):1153–1160.

Kitaoka, A. (2010). A brief classification of colour illusions. *Colour: Design & Creativity*, 5(3):1–9.

Kuehni, R. G. (2001). Color space and its divisions. *Color Research & Application: Endorsed by Inter-Society Color Council, The Colour Group (Great Britain), Canadian Society for Color, Color Science Association of Japan, Dutch Society for the Study of Color, The Swedish Colour Centre Foundation, Colour Society of Australia, Centre Français de la Couleur*, 26(3):209–222.

Lee, J. and Lee, S. (2016). Hallucination from noon to night images using cnn. In *SIGGRAPH ASIA 2016 Posters*, pages 1–1.

Liu, S. (2022). An overview of color transfer and style transfer for images and videos. *arXiv preprint arXiv:2204.13339*.

McCamy, C. S., Marcus, H., Davidson, J. G., et al. (1976). A color-rendition chart. *J. App. Photog. Eng*, 2(3):95–99.

Mine, M. R., van Baar, J., Grundhofer, A., Rose, D., and Yang, B. (2012). Projection-based augmented reality in disney theme parks. *Computer*, 45(7):32–40.

Nayar, S. K., Peri, H., Grossberg, M. D., and Belhumeur, P. N. (2003). A projection system with radiometric compensation for screen imperfections. In *ICCV workshop on projector-camera systems (PROCAMS)*, volume 3. Citeseer.

Oyamada, Y. and Saito, H. (2008). Defocus blur correcting projector-camera system. In *International Conference on Advanced Concepts for Intelligent Vision Systems*, pages 453–464. Springer.

Reid Jr, R. C. and Shapley, R. (1988). Brightness induction by local contrast and the spatial dependence of assimilation. *Vision research*, 28(1):115–132.

Reinhard, E., Adhikhmin, M., Gooch, B., and Shirley, P. (2001). Color transfer between images. *IEEE Computer Graphics and Applications*, 21(5):34–41.

Ro, H., Park, Y. J., Byun, J.-H., and Han, T.-D. (2019). Display methods of projection augmented reality based on deep learning pose estimation. In *ACM SIGGRAPH 2019 Posters*, pages 1–2.

Rossotti, H. (1985). *Colour: Why the world isn't grey*, volume 3. Princeton University Press.

Sand, O., Büttner, S., Paelke, V., and Röcker, C. (2016). smart. assembly–projection-based augmented reality for supporting assembly workers. In *International Conference on Virtual, Augmented and Mixed Reality*, pages 643–652. Springer.

School, D. (2014). Simple guide to confusion matrix terminology. `https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/`.

Sinha, P., Crucilla, S., Gandhi, T., Rose, D., Singh, A., Ganesh, S., Mathur, U., and Bex, P. (2020). Mechanisms underlying simultaneous brightness contrast: Early and innate. *Vision Research*, 173:41–49.

Sridharan, S. K. a., Hincapi'e-Ramos, J. D., Flatla, D. R., and Irani, P. (2013). Color correction for optical see-through displays using display color profiles. In *Proceedings of the 19th ACM Symposium on Virtual Reality Software and Technology*, pages 231–240.

Tkalcic, M. and Tasic, J. (2003). Colour spaces: perceptual, historical and applicational background. In *The IEEE Region 8 EUROCON 2003. Computer as a Tool.*, volume 1, pages 304–308 vol.1.

Wei, L.-Y., Lefebvre, S., Kwatra, V., and Turk, G. (2009). State of the art in example-based texture synthesis. *Eurographics 2009, State of the Art Report, EG-STAR*, pages 93–117.

Weiland, C., Braun, A.-K., and Heiden, W. (2009). Colorimetric and photometric compensation for optical see-through displays. In *International Conference on Universal Access in Human-Computer Interaction*, pages 603–612. Springer.

Wikipedial (2020). Mean squared error. `https://en.c.org/wiki/Mean_squared_error`.

Wikipedial (2021). Peak signal-to-noise ratio. `https://en.wikipedia.org/wiki/Peak_signal-to-noise_ratio`.

Xiao, X. and Ma, L. (2006). Color transfer in correlated color space. In *Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications*, pages 305–309.

Yeh, K.-C., Tsai, M.-H., and Kang, S.-C. (2012). On-site building information retrieval by using projection-based augmented reality. *Journal of Computing in Civil Engineering*, 26(3):342–355.

Ziou, D., Tabbone, S., et al. (1998). Edge detection techniques-an overview. *Pattern Recognition and Image Analysis*, 8:537–559.

Appendix

...