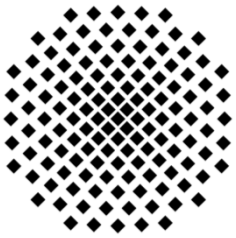

Institut für Parallele und Verteilte Systeme
Abteilung BildVerstehen
Universität Stuttgart
Universitätsstraße 38
D-70569 Stuttgart

Master Thesis Nr. 3477

Color Balance in LASER Scanner

Point Clouds

Umair Nasir

Studiengang:	INFOTECH
Prüfer:	Prof. Dr. rer. nat. habil. Levi, Paul
Betreuer:	PD Dr. rer. nat. Schanz, Michael
Begonnen am:	1st February 2013
Beendet am:	3rd August 2013
CR-Nummer	D.2.2, D.2.3, D.3.3, D.3.4. E.1, F.2.1, F.3.3, G.1.1, G.3, I.4.3, I.4.8, I.6.2

Abstract

Color balancing is an important domain in the field of photography and imaging. Its use is necessitated because of the color inconsistencies that arise due to a number of factors before and after capturing an image. Any deviation from the original color of a scene is an irregularity which is dealt with color balancing techniques. Images may deviate from their accurate representation because of different illuminant ambient conditions, non-linear behavior of the camera sensors, the conversion of file format from a wider color gamut of raw camera format to a file format with a narrower color gamut and so on.

Many approaches exist to correct the color inconsistencies. One of the basic techniques is to do a histogram equalization to increase the contrast in an image by utilizing the whole dynamic range of the brightness values. To remove color casts introduced due to false illuminant selection at the time of image capture many white balancing techniques exist. The white balancing can be employed before image capture right in the camera using hardware filters with dials to set illuminant conditions in the scene. A lot of research has been done regarding the effectiveness of white balancing after image capture. The choice of color space and the file format is quite important to consider before white balancing.

Another side to color balancing is color transfer whereby the image statistics of one image are transferred to another image. Histogram matching is quite widely used to match the histogram of a source image to that of a target. Other statistics for color transfer are to match the mean and standard deviation of a source image to a target image. These two approaches for color transfer are analyzed and tested in this thesis on images displaying the same scene but with different color casts. Color transfer matching the means and standard deviations is selected because of its superior color balancing and ease of implementation.

While a lot of color balancing work has been done in 2D images, no significant work is done in the 3D domain. There exist 3D scanners which scan a scene to build its 3D model. The 3D equivalent of the 2D pixel is a scan point which is obtained by reflecting a laser beam from a point in a scene. Hundreds of thousands of such points make up a single scan which displays the scene that was in the view of the 3D scanner. Because a single scan cannot capture scene behind obstructions or the scenes out of the scanners' range, more than one scans are undertaken from different positions and time. More than one scans grouped together make up a data structure called a point cloud. Due to these changes in position and time, luminance conditions alter. As a result the scan points from different scans representing the same scene show a considerably different color cast.

Color balancing by matching the means and standard deviation is applied on the point cloud. The color inconsistencies such as sharp color gradients between points of different scans, the presence of stray color streaks from one scan into another are greatly reduced. The results are quite appealing as the resulting point clouds show a smooth gradient between different scans.

Acknowledgements

First of all, I am very thankful to Dipl. Inf. Joachim E. Vollrath for his valuable time and suggestions during the supervision of my thesis. I would also like to thank PD Dr. Michael Schanz for giving me an opportunity of working in his department.

Finally, I would like to recognize the continuous support of my family members which has always been there throughout my studies.

Table of Contents

List of Figures

Acronyms

LASER	Light Amplification by Stimulated Emission of Radiation
CDF	Cumulative Distribution Function
CIE	Commission Internationale de l'Eclairage
HSV	Hue Saturation Value
PCA	Principal Component Analysis
LMS	Long, Medium, Short
STL	Standard Template Library
JPEG	Joint Photographic Experts Group

1. Introduction

1.1 LASER Scanner and Point Clouds

A LASER scanner is a device which uses laser beams to probe any object lying in its scope of view. Each laser beam that is reflected back from the object helps find the distance of the point which reflected it and thus calculate the distance information. A collection of millions of such points taken by changing the angle of the laser emitter help in the construction of 3D models of the subjects. These points are called scan points. Besides the distance information, the color information of the points can also be recorded.

A laser scanner has many uses. One use is in video games where the infrared laser scanner is known as Kinect and is a part of the Xbox 360 [**1**]. The scanner identifies the player and his movements in real time. It is also used in the field of robotics for the identification of obstructions and other objects in the path of a robot [**2**]. Another very common use for a laser scanner is in medicine for designing prosthetics. In industry the laser scanners facilitate high precision 3D measurement and 3D Image Documentation. One example of such a laser scanner is FARO Focus3D Laser Scanner [**3**].

A laser scanner such as FARO Focus3D can generate millions of scan points in a few minutes. A collection of all the scan points constitute a data structure called a point cloud [**4**]. Only those surfaces that lie in direct view of the scanner can be reconstructed. To get the other surfaces or obstructions into view, the position of the laser scanner is changed, new complete scans undertaken, and the points of the multiple scans are combined. Thus one point cloud can represent one or more individual scans.

1.2 Luminance and Chromaticity Artifacts in Scans

A problem arises when more than one scans are combined into a point cloud. While taking a scan at a different time and from a different position, the lighting conditions may have changed with the effect that the scan points representing the same object now vary in luminance and chromaticity. Besides external changes in the environment, there are internal effects as well. Other reasons can be variation in the laser scanner's internal camera signal response and time varying non-linear automatic gain control of the camera [**5**].

In a color model, luminance and chromaticity together make up a color. A color space comprises of at least three components such as the RGB, HSV, YUV, CMYK and so on. Luminance measures the brightness of light. The chromaticity component itself is made up of two further components which are hue and saturation [**6**].

Hue is the attribute with which the colors can be classified as red, blue, green, yellow and the colors that are formed by mixing them [**7**]. These are also called the tones of color or chromatic colors.

Saturation is the component of chromaticity which describes the purity of a hue or of a chromatic color. In the presence of any achromatic colors such as gray, the saturation is decreased and is eliminated when it becomes absolutely gray [**8**].

After the scans have been taken by the laser scanner and a point cloud is generated, objects representing the same scene consist of points that originate from different scans. Although this does not have any effect on the basic shape of any object, it does cause visual imperfections. Sometimes very sharp gradients appear in point clouds at the junctions where more than one scans meet. Sometimes the scan points of different luminance and chromaticity appear in small chunks or streaks between the scan points of another scan.

Figure 1.1 Screenshot of a 3D Point Cloud with color inconsistencies

Figure 1.1 shows a 3D point cloud which consists of a number of individual scans. We see sharp gradients between two scans on the floor. On the left side of the wall, there are some streaks from a scan with a brownish color cast. As a whole the complete wall is visually unappealing due to color inconsistencies between different scans.

1.3 Color Balancing

Color Balancing or color correction is a method which does adjustments to the chromatic and achromatic colors in order to reduce color inconsistencies that might appear during image capture. In other words, color correction is the recovery of the actual scene characteristics in an image. The

characteristics that need correction are color, contrast and sharpness [**9**]. Color inconsistencies can arise both in neutrals such as the gray as well as colors. The latter is said to have a color cast.

Adjustment of the neutrals is called white balancing or gray balancing. The goal of white balancing is that a neutral color in the original scene should also be neutral in a captured scene. For this purpose all the colors in an image are scaled by a value so that an effected neutral point or patch appears neutral again.

White balancing can be employed before capturing an image by estimating the illuminant followed by some filter so that a neutral object actually appears neutral. Von Kries Transform is one method that applies gains to the spectral sensitivity responses in the cones of the eyes. The gains are adaptable depending upon the illuminant [**10**] and are applied in the LMS color space. LMS stands for long, medium and short waves because the three types of cones in the eye are sensitive to these three wavelengths. Von Kries Adaptation is one feature of camera image processing.

In addition to the digital filtering within a camera, another method to apply white balancing is to physically put color filters over the lens of the camera. For a wide variety of illuminants, the filter wheels are used which can be rotated to adjust for different illuminants.

White balancing can also be performed in a later stage on monitor but this causes additional color inconsistencies. In [**11**] Stephen Viggiano studied six methods of white balancing at different stages and concluded that balancing in the camera's native RGB produces the best results for white balancing. This is because a raw image is minimally processed and exhibits a higher dynamic range compared to other formats which are obtained after processing the raw image. The higher dynamic range means that the color gamut is broad because each color channel is represented with 12, 14 or 16 bits compared to 8 bits in JPEG file format.

Color Balancing also removes color casts which may appear in images due to different reasons such as the filters used in the camera during capture, while printing and scanning and due to the way the original scene was illuminated [**12**]. Such color casts can be removed by first identifying the particular color casts and then using physical color correction(CC) filters of varying density depending upon the intensity of the color cast [**12**]. The color casts can also be removed later in software using color correction algorithms [**13**] [**9**] [**14**].

Contrast is the difference in the brightness level for any channel of a color space. The more the difference in brightness, the higher is the contrast and it is visually more aesthetic. Color correction by contrast adjustment is most widely done by histogram equalization which spreads the histogram of a color channel over the whole dynamic range of the brightness levels.

Another possibility of color balancing is by color transfer [**15**]. With color transfer the characteristics of one image are transferred to another image through some color transfer function. The characteristics can be the histogram of an image or some other image statistics such as mean, standard deviation or kurtosis. The color transfer function for modifying the histogram of the source image to match that of a target image is called the cumulative distribution function(CDF).

The luminance and chromaticity inconsistencies discussed in Section 1.2 have been known to be eliminated in 2D images [**13**] [**9**] [**16**] [**12**]. In this thesis report the removal of the color inconsistencies that arise in the point clouds is discussed.

Chapter 2 deals with the previous work that is done in the field of color balancing. It lists some of the color balancing algorithms and also explains the underlying concepts for color correction. The importance of a color space for color correction is also highlighted in this chapter along with some explanation of the formulae for color space conversions.

In Chapter 3, the experimental evaluation of two of the color balancing algorithms is described based on which one of these algorithms is picked. Chapter 4 details the implementation of the selected algorithm which is followed by the evaluation of the results of Chapter 3 and 4 in Chapter 5.

2. Background and Related Work

2.1 Color Balancing

2.1.1 White Balancing for Color Cast Correction

Under different light sources, a region in a scene that is actually white appears to have a color cast especially after an image has been captured. Humans are capable to interpret white objects as white under different illuminants such as tungsten, fluorescent light, direct sunlight, shady, cloudy weather etc. But for a camera this is not possible. White balancing is necessitated when objects that are actually white do not appear white in an image.

White balancing can be applied prior to or after image capture. If it is done prior to taking an image, it can be done as auto white balancing or manual white balancing within the camera. In auto white balancing the camera makes a guess on the most accurate description of the conditions. This 'auto' option is often shown in digital cameras. Prior to image capture, one can also manually set the light source where a user is presented with possible illuminants such as tungsten, shady, cloudy etc.

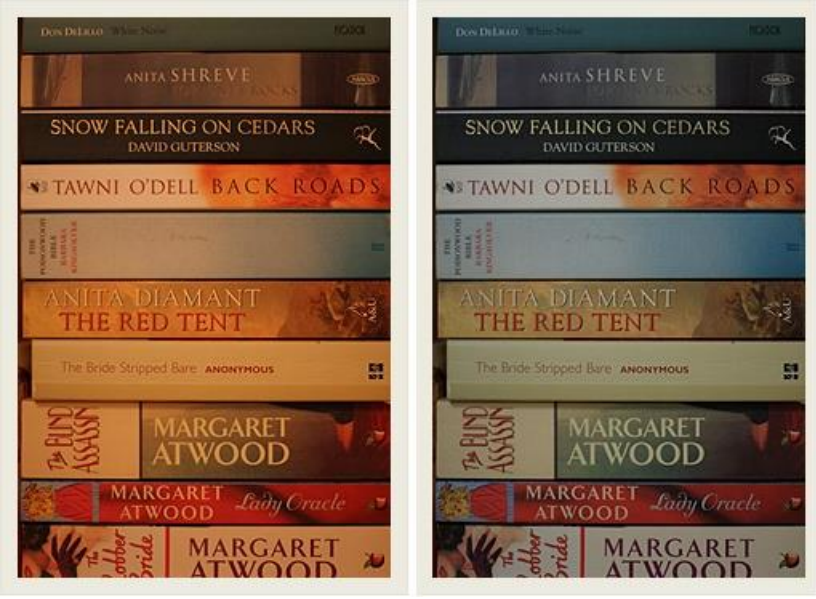

Figure 2.1 Images without and with White Balancing [17**]**

Figure 2.1 shows two images. The left image was taken under standard tungsten bulbs in a room without white balancing. It shows quite considerable yellow cast which is characteristic of tungsten bulbs. The right image was taken with white balancing after the camera was shown a white paper to adjust its white balancing settings [**17**]. The color cast reflects the type of illuminant [**18**]. When a reference white point is known then white balancing is a very simple scaling operation. Below a simple scaling is shown in RGB color space.

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 255/R'' & 0 & 0 \\ 0 & 255/G'' & 0 \\ 0 & 0 & 255/B'' \end{bmatrix} \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix}$$

R, G and B are the coordinates of a pixel which is the result of white balancing. R', G' and B' belong to the original pixel which has the color cast. In the 3x3 scaling matrix, R'', G'' and B'' are the average RGB values for the reference white point(s) effected by the color cast of the illuminant. The value of 255 in each of the diagonal entries of the scaling matrix represents the actual RGB value of a white point.

Generally white balancing occurs in two basic steps. The first step is to gather information about the illuminant. The illuminant can be known or it can be deduced by analysis of the image as well. The second step is to scale the values of the pixels with a matrix whose values differ for different illuminants.

A very simple method of white balancing which is also very common is the 'gray world assumption'. It states that any image contains equal amounts of red, blue and green. If an image doesn't live up to this assumption, then correction factors are applied for scaling the RGB values so that the final image holds on to the gray world assumption. In [**19**] Nguyen et al used the concept of gray world assumption for illuminant estimation. Images in a database were captured under different single light source and the scale factors to get the resulting images adhering to gray world assumption were recorded. These scale factors correspond to individual illuminants. The results were reasonable but the gray world assumption is not always true. In images where a particular color dominates such as sky, oceans, green grass, the gray world assumption fails [**18**].

In [**18**] Manuel Innocent describes another method for illuminant estimation. For an image a weighted average for the color values is calculated. With this value the illuminant is determined. Next based on the illuminant as well as using the color values of some reference white points in the image, the scaling factors for white balancing are calculated.

White balancing can be applied in various stages. As mentioned in Section 1.3, white balancing manifests best results when done on raw images in camera itself rather than later in monitor.

White balancing can also be applied in different color spaces. Xiao et al in [**20**] experimented with white balancing in various color spaces such as XYZ, Bradford, camera sensor RGB and sharpened RGB. They presented a general formula for white balancing

$$RGB_{out} = F_{out} * \mathrm{D} * F_{in} * RGB_{in}$$

RGB_{in} is a 3x1 matrix corresponding to the original pixel which needs to be white balanced.

F_{in} is the 3x3 matrix that converts RGB to any other space where white balancing is to be performed.
D is the 3x3 matrix with scale factors dependent upon the illuminant under which the image was taken.
$F_{out}$is a 3x3 color space conversion matrix which converts the result back to the RGB space.

Xiao et al concluded that XYZ and sharpened RGB performed almost equally well while Bradford and camera sensor RGB performed poorly.

2.1.2 Histogram Matching and Histogram Warping

Histogram matching or histogram specification is a method that maps the histogram of one image to the histogram of another image taken as a target. Histograms have many applications such as fast image retrieval where the images with a similar histogram are searched among a pool of images [**21**]. It is also used in gamut mapping techniques [**22**].

Histogram matching can be applied in 1D grayscale or 3D color spaces. But the main idea is the same. The source histogram has to match the target histogram. In theory, the source histogram can match exactly to the target histogram in continuous domain, but in discrete domain with finite number of pixels and quantization levels this is not always the case.

In order to do color mapping through histogram matching, first of all the dynamic range of the intensity values of both the source(S) and target(T) images are divided into intensity ranges called bins. These bins act as the random variables denoted here with *x* for the source and y for the target images. The histograms for these random variables are calculated for both the images.

Let the histograms or the probability distributions for the random variable x be denoted as $p_s(x)$ for the source and $p_T(y)$ for the target. A cumulative distribution function(CDF) is determined for the two probability distributions denoted as $F_s(x)$ and $F_T(y)$ respectively. The CDF is a monotonically increasing function starting at 0 and ending at 1.

To find the value of y for each source pixel x, the following transformation T is used

$$\mathrm{y} = \mathrm{T}[\mathrm{x}] = {F_T}^{-1}(F_S[x])$$

where F_T^{-1} is the inverse function of F_T.

According to Neumann in [**21**] histogram matching generates results which lacks spatial coherences such as the gradients, neighborhoods and topological characteristics. The edges and overall structure is conserved nonetheless. The gradient can be preserved by variation in the histogram specification implementation [**14**] whereby Xiao et al apply an additional optimization step to preserve the source gradient map and apply it after the histogram specification has performed the color transfer.

Figure 2.2 Histogram Matching followed by Gradient Preservation [14**]**

Figure 2.2 shows the result in image 3 after Xiao's color transfer and gradient preservation is applied on image 1 to match image 2.

The problem with histogram matching is that it can lead to contouring effects with spikes and dips in the resulting histogram. The spikes are a result of many pixels in the neighborhood of a particular bin accumulate to the same bin and at the same time cause immediate dips in the neighbor bins. Histogram Warping is a modified version of histogram matching which divides the source and target histogram into quantiles with an adaptive matching function for each of the quantiles. If the number of quantiles are equal to the number of bins in both the source image and the target image, what we have is the original histogram specification [**23**]. By selecting different number of quantiles in the source and target histograms, one can match the target to any desired level of accuracy easily controlling histogram stretching and expansion.

Grundland et all in [**23**] broke down the 3D histogram matching into three 1D histogram matching in a space which is a derivation of CIELab referred in the paper as CIELa''b''. It is derived by applying principal component analysis on CIELab followed by independent component analysis so that the resulting space is maximally decorrelated. They also applied a monotonic interpolating spline to increase the contrast after the histogram warping has been performed.

Figure 2.3 Result of Color Histogram Warping [23**]**

Figure 2.3 shows the result of histogram warping where the top original images swap each other colors as shown in the bottom row.

2.1.3 Mean and Standard Deviation Matching

Another category of image enhancement is matching the image statistics of a source image to a target image. The source image and the target image may not be depicting the same scene. This has the effect to transfer a color cast of one image completely to another without changing the texture of the target image.

The algorithm basically computes the means and standard deviations for each of the three color channels of complete source and target images. Then each pixel of the source image is given a translational and scale effect based on the statistics computed. The formula is very simple and computationally inexpensive. Let a pixel be P whose old value is P_{old} and the new value to be calculated is P_{new}. Let the mean and standard deviations for the source and target be μ_s, σ_s and μ_t and σ_t respectively. Then the color transfer algorithm is

$$P_{new} = (P_{old} - \mu_s) * \frac{\sigma_t}{\sigma_s} + \mu_t$$

For each pixel, first the mean of the whole source image is subtracted and then a scale operation by a factor of σ_t/σ_s is applied which matches the standard deviation to the target. Lastly the addition of the term μ_t matches the mean to the target as well.

The use of this color transfer algorithm based on image statistics in different color spaces is found in [**13**], [**15**] and [**24**].

Figure 2.4 shows the result of color transfer in [**15**] in Lαβ space. The color cast that of sunset in image b is transferred to image a.

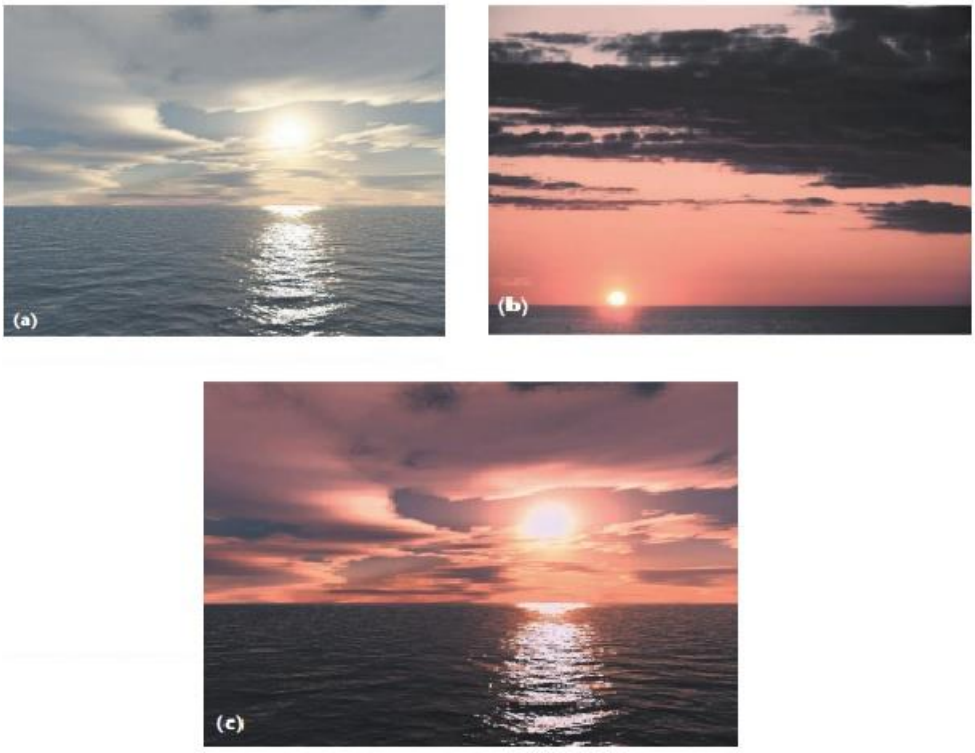

Figure 2.4 Result of Color Transfer from Image b to Image a. [15**]**

This color transfer algorithm matching the image statistics performs particularly well when the composition of both the images is more or less the same. In Figure 2.4, both the source image and the target image have similar composition showing sun, clouds, sea and sky. The results might not be appealing when the compositions differ. In such cases, separate swatches can be taken from different objects in the images, the means and standard deviations calculated for these swatches, and color transfer is applied separately for each of the swatches [**15**].

The report also deals with the cases where the scene is the same but with completely different color casts. Such is the case with the point clouds where two or more scans represent the same scene but have scan points of different colors. Point clouds are different from images in that images consist of pixels in 2D, but point clouds contain scan points that may fall anywhere in 3D space. There might be regions without any scan points at all. Furthermore in 3D the same scene might comprise of even more than two scans in which case the calculation of target means and standard deviations wouldn't be as trivial as seen in this section.

2.2 Color Spaces for Color Balancing

Color space is a way to express, specify and visualize colors [**25**]. A number of color spaces exists such as RGB, YUV, CMYK, CIELab, HSV and so on. All vary in certain characteristics such as the color gamut, device independence, independence of the chromaticity and luminance components, correspondence to human visual perception etc. Color gamut represents the range of colors that can be represented by a given color space. Each color space is suited for different domains. For example RGB is suitable for displays because of the ease of system design. CMYK (Cyan, Magenta, Yellow, Black) is used in color printing. CMYK is useful for color printing because it is a subtractive color space which means that each of the four pigments are applied to a white surface to create the final color [**26**]. The YUV color space is derived from RGB where Y stands for luminance and U and V channels are color difference channels. YUV color space is useful for analog television broadcasting because of its low bandwidth.

CIEXYZ is an international standard developed by CIE (Commission Internationale de l'Eclairage) and the major advantage of this color space is that it encompasses the gamut of most other color spaces such as RGB, CIELab etc. Furthermore it is device independent which means that this color space is independent of the image capturing device as well as the display device. All the color spaces that are derived from CIEXYZ are also device independent.

Then there are color spaces that have the luminance and chromaticity decoupled. Examples of such spaces are HSV, CIEXYZ, CIELab etc. The decoupling of luminance and chromaticity is particularly useful in image enhancement methods such as discussed in Sections 2.1.2 and 2.1.3, for rendering, noise removal, segmentation and object recognition [**27**].

In the subsections we discuss what exactly is decoupling, it's advantages and how to find the extent of decoupling through independent component analysis. Lastly the color space conversions that are used in the course of the report are discussed.

2.2.1 Decoupling of Chromaticity and Luminance

With image enhancement algorithms, where it is important to do computations on all the color channels, a problem arises when if one of the channels is adjusted, the overall luminance and chromaticity of an image is disturbed. Thus adjustment in one channel disturbs the other two channels. For example if one requires to reduce red cast in an RGB image, then this would also reduce the overall brightness in the image. This would necessitate that all the three channels are adjusted at the same time. Such color spaces are said to have correlated color channels [**13**]. The color spaces which have orthogonal color channels have zero correlation between them. For image enhancements it is a lot easier to work in color spaces that are uncorrelated to a considerable degree as it solves a 3D problem as three 1D problems.

An important work in the field of color transfer in a decoupled color space is done by Erik Reinhard et al who used a recently derived color space Lαβ [**15**]. Ruderman et al developed this perception based color space derived from LMS space where L, M and S stand for Long, Medium and Short wavelengths to

which the human eyes are sensitive. This color space shows very little correlation between the channels of any pixel [**28**].

Covariance is a method to compute the correlation between any pair of the three color channels. Figure 2.5 shows the extent of decorrelation for the input image shown in RGB and Lαβ color spaces after applying PCA. It is easily observable that the decorrelatoin for any pair in the RGB color space is absolutely nonexistent. For the Lαβ, it can easily be seen that the correlation is quite reduced.

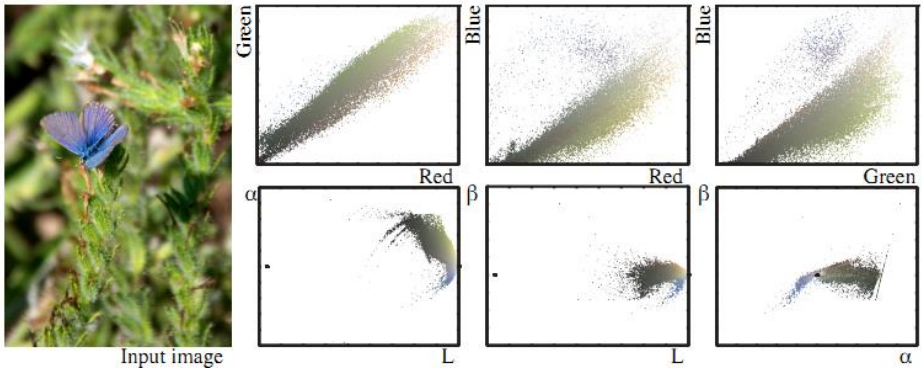

Figure 2.5 Showing Correlation between pairs of Channels in RGB and Lαβ for the input image [13**]**

Erik Reinhard et al in [**13**] studied a set of color spaces and used covariance between the pairs of channels to find out the decorrelation between the three color channels for a variety of image ensembles in natural day[ND], manmade day[MD], indoors[IN], night[N]. They also analyzed the success of these color spaces on the color transfer algorithm discussed in Section 2.1.3. The color spaces tested include Lαβ, CIELab with illuminant D65 and E, Yuv, HSV, XYZ, RGB etc as well as ensemble based maximally decorrelated spaces computed by Principal Component Analysis (PCA) [**29**].

The results of the covariance analysis showed that CIELab with illuminant E produced the most decorrelated channels for each of the category of input images. RGB and XYZ produced very poor results. The color transfer algorithm in Section 2.1.3 was applied to all the image ensembles in all the color spaces. Figure 2.6 shows the results as percentage success of each color space in each image category. The percentage success is a subjective result whereby the two authors of [**13**] independently analysed the color transferred images selecting the results that were believable and ignoring the ones that weren't visually appealing. The black bar shows the average result for all the image types.

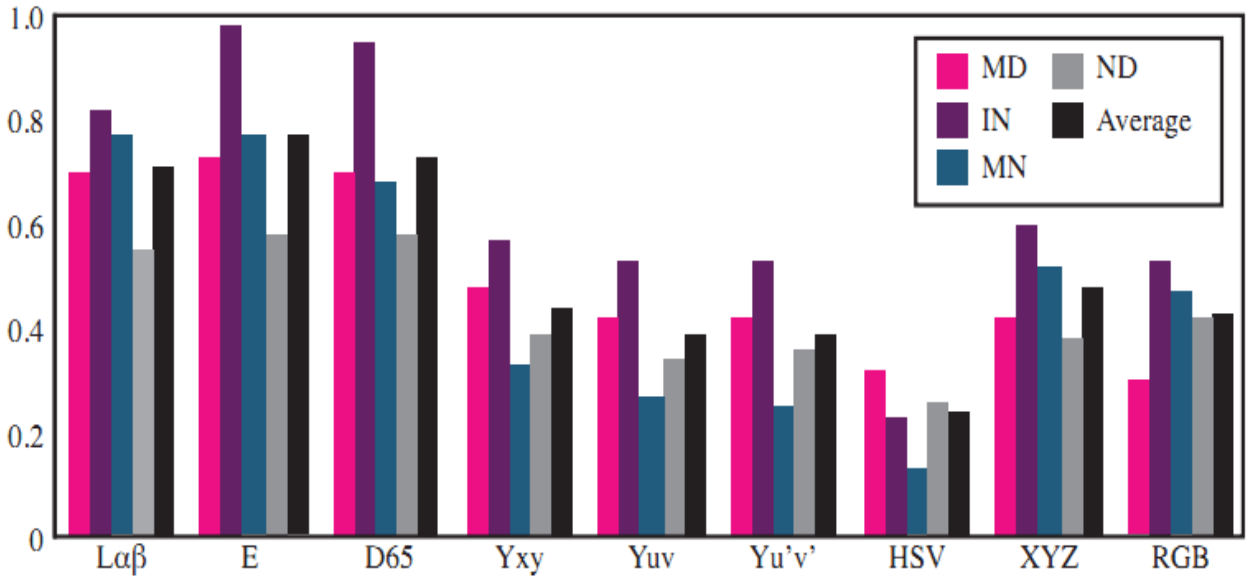

Figure 2.6 Performance of color transfer on different image ensemble in different color spaces [13**]**

It is easily observed that CIELab with both illuminants E and D65 does the best color transfer. The results of [**13**] also took into account the color spaces generated by PCA. PCA was applied on both source image and the target image separately. It was concluded that a color space derived through PCA on target image produced quite good results for color transfer. However CIELab with illuminant E still performed best. The fact that a color space is derived through PCA for each image in every run makes it an unfeasible approach for color transfer specially when CIELab with illuminant E already performs better.

2.2.2 Color Space Conversions

Based on the results of [**13**], the best color transfer with respect to color spaces occurs in the CIELab color space. The images and the scans available contain pixels and scan points respectively with colors in RGB color space. To apply the color transfer algorithm in [**15**], it is required that the color space be converted from RGB to CIELab. In the following is detailed the method to convert the color space to CIELab and then back to RGB after color transfer has been performed.

To get to the CIELab the color values must be converted to an intermediate space known as CIEXYZ. The human eye has three types of cones which are sensitive to long, medium and short wavelengths. These three components make up three tristimulus values. The tristimulus values are strongly related to color perception by the eyes. In 1931, CIEXYZ was introduced as the first mathematically defined color space by International Commission on Illumination (CIE). The CIEXYZ color space covers all the possible color perceptions for humans. Therefore this color space often acts as a building block for conversions to various color spaces such as CIELab. In CIEXYZ, the Y corresponds to the luminance, Z stimulates the blue hue, and X is the mixture of the three response curves of the cones. Thus X and Z together define the chromaticity [**26**].

The color conversion formulae are taken from [**30**]. The conversion from RGB to CIELab and back is shown as pseudocode below.

RGB to CIEXYZ

First of all the values of RGB are normalized between 0 and 1.

```
var_R = (R/255)
var_G = (G/255)
var_B = (B/255)

if (var_R> 0.04045)
        var_R = ((var_R + 0.055) / 1.055)^2.4
else
        var_R = var_R / 12.92

if (var_G> 0.04045)
        var_G = ((var_G + 0.055) / 1.055)^2.4
else
        var_G = var_G/ 12.92

if (var_B> 0.04045)
        var_B = ((var_B + 0.055) / 1.055)^2.4
else
        var_B = var_B/ 12.92

var_R =var_R * 100
var_G = var_G * 100
var_B = var_B * 100
```

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.4124 & 0.3576 & 0.1805 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9505 \end{bmatrix} * \begin{bmatrix} var\_R \\ var\_G \\ var\_B \end{bmatrix}$$

Once in CIEXYZ, the CIELab is derived by nonlinearly compressing(cube root) the CIEXYZ values. The L stands for luminance and a and b channels are color opponent channels. CIELab encompasses the gamut of RGB.

CIEXYZ to CIELab

```
var_X = (X/ref_X)
var_Y = (Y/ref_Y)
var_Z = (Z/ref_Z)
```

The values of ref_X, ref_Y and ref_Z depend upon the illuminant. For illuminant E, the values are 100 each. For D65, the values are 95.047, 100 and 108.883 respectively

```
if (var_X> 0.008856)
        var_X = (var_X)^1/3
else
        var_X = (7.787 * var_X) + 16/116

if (var_Y> 0.008856)
```

var_Y = (var_Y)^1/3
else
*var_Y = (7.787 * var_Y) + 16/116*

if (var_Z> 0.008856)
var_Z = (var_Z)^1/3
else
*var_Z = (7.787 * var_Z) + 16/116*

*CIE–L = (116 *var_Y) - 16*
*CIE–a = 500 * (var_X – var_Y)*
*CIE–b = 200 * (var_Y – var_Z)*

Now the algorithm in [**15**] can be applied on the CIELab values followed by conversion of the CIELab back to RGB via CIEXYZ which is basically the inverse to the RGB to CIELab conversion.

CIELab to CIEXYZ

var_Y = (CIE–L + 16) / 116
var_X = (CIE-a/500) + var_Y
var_Z = var_Y - CIE-b/200

if (var_X> 0.008856)
var_X = (var_X)^3
else
var_X = (var_X – 16/116) / 7.787

if (var_Y> 0.008856)
var_Y = (var_Y)^3
else
var_Y = (var_Y– 16/116) / 7.787

if (var_Z> 0.008856)
var_Z = (var_Z)^3
else
var_Z = (var_Z – 16/116) / 7.787

*X = var_X * ref_X*
*Y = var_Y * ref_Y*
*Z = var_Z * ref_Z*

CIEXYZ to RGB

var_X = (X/100)
var_Y = (Y/100)
var_Z = (Z/100)

$$\begin{bmatrix} var\_R \\ var\_G \\ var\_B \end{bmatrix} = \begin{bmatrix} 3.2406 & -1.5372 & -0.4986 \\ -0.9689 & 1.8758 & 0.0415 \\ 0.0557 & -0.2040 & 1.0570 \end{bmatrix} * \begin{bmatrix} var\_X \\ var\_Y \\ var\_Z \end{bmatrix}$$

if (var_R> 0.0031308)
*var_R = 1.055 * (var_R ^ 1/2.4) – 0.055*
else

```
        var_R = var_R* 12.92

if (var_G>0.0031308)
        var_G = 1.055 * (var_G ^ 1/2.4) – 0.055
else
        var_G= var_G* 12.92

if (var_B>0.0031308)
        var_B = 1.055 * (var_B^ 1/2.4) – 0.055
else
        var_B = var_B * 12.92

var_R  =var_R * 255
var_G = var_G * 255
var_B = var_B * 255
```

These color space formulae work on one pixel at a time.

3. Experimental Evaluation in MATLAB

This chapter deals with selecting two approaches to color balancing, applying these approaches on 2D panoramic images generated from LASER Scanners, and selecting the suitable algorithm for final integration in 3D.

The two approaches that would be simulated in MATLAB are

- Histogram Matching
- Means and Standard Deviations Matching.

3.1 Data Acquisition

Using FARO's FOCUS 3D LASER Scanner, some scans were generated. The scanner was placed at the same position throughout the experiment so as to capture the same scene in all the scans. However because the scans were taken at different times i.e. in the morning, afternoon and before sunset, the contrast of the chromaticity and luminance was significant to the extent that the auto white balancing of the camera couldn't compensate this effect.

Figure 3.1 depicts the variation in color and luminance in scans taken in a room.

The algorithms were also applied on other data-sets which were taken from scans in different locations such as museums etc. In this chapter, only the scans that of the room shown in Figure 3.1 are considered.

Figure 3.1 Scans of the same scene taken in the morning(above) and the evening(below)

3.2 Experiments in MATLAB

Both the algorithms aim to match a source image to a given target but the matching occurs in two different ways. Histogram matching algorithm tries to match the cumulative distributive function of the source to the target. On the other hand the mean and standard deviation matching as the name implies matches the source mean and standard deviation to that of the target.

3.2.1 Histogram Matching

For histogram matching, any of the two images is taken as the source and the other as the target. In the subsequent discussion, the morning image is taken as the source and the evening image as the target. Histogram matching matches the source to the target in such a way so that the cumulative distribution function (CDF) of the matched image is as close to the CDF of the target as possible. This also means that the resulting histogram of the matched image is as close to the histogram of the target as possible.

In RGB color space, the intensity values range from 0 to 255, and these intensity values act as the 256 random variables. The histograms are generated by counting the number of pixels for each of the

intensity values and then displaying the results in the form of a graph. A normalized histogram follows the identity

$$\sum_{i=0}^{255} \frac{X_i}{N} = 1$$

Where Xi represents the number of pixels with intensity i and N is the total number of pixels in the image. Each intensity value is called a bin. In RGB, this bin represents discrete integral values of the intensities. In other color spaces, the values of the color channels might take non-integer values in which case a bin is a range of values.

In MATLAB, an algorithm for histogram matching was implemented which took the number of bins as a user-specified input. The more the number of bins, the smaller the range of each bin and the higher is the resolution. For RGB, making more number of bins than 256 makes no sense because all the intensity values are integral values. Histogram matching was also performed on the images after a color space conversion was carried out from RGB to CIELAB.

Once in CIELAB, the values range for the three color channels are now different and can possibly take any real value. The L channel for luminance, ranges from 0 - 100. The a and b channels are basically unbounded but are encoded within a range of -127 to +128. Here in this color space, because the channel values are infinite but the number of pixels is finite, it's normal that a lot of bins do not have any pixel falling in that range.

Figure 3.2 shows the histograms of the red channel of two panoramic images of the room in RGB.

As one would expect, the morning image has more pixels with higher intensity values compared to the darker evening image.

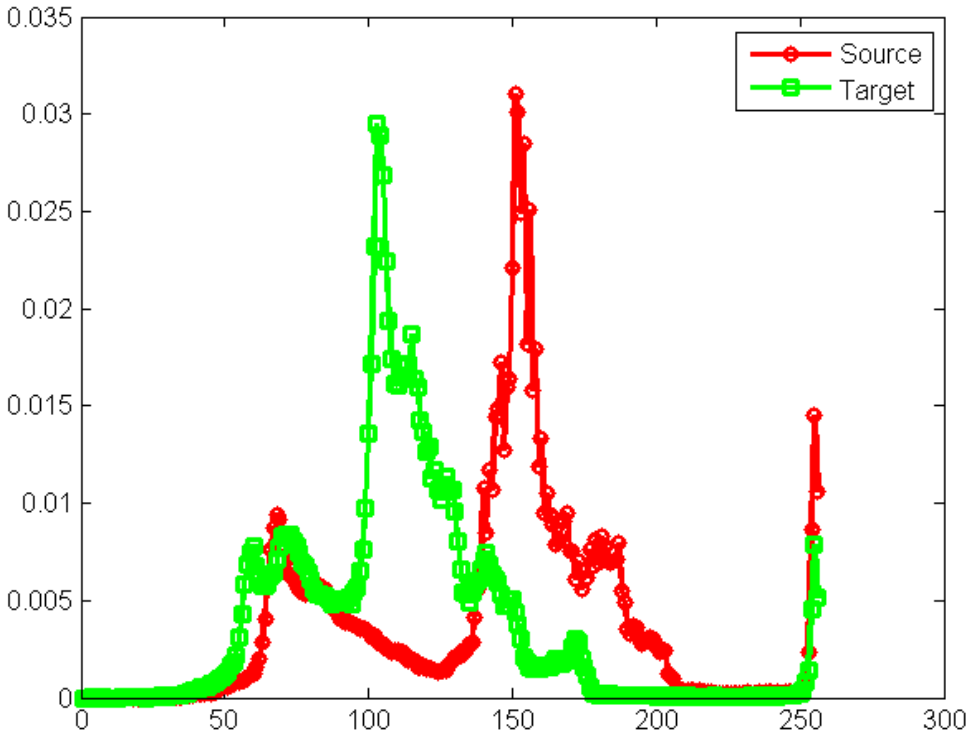

Figure 3.2 Red channel of the morning(source) and evening(target) images of the room

For source image, let the cumulative distributive function be called F_S and for the target let it be called F_T. The functions F_S and F_T are calculated using the histograms of both the source and the target respectively. In general, the formula for CDF is

$$F = \mathrm{P}(\mathrm{X} \leq \mathrm{x}) = \sum_{k=0}^{x} \mathrm{p}(\mathrm{x})$$

In this formula P is the distribution function (histogram) as in Figure 3.2 above and the expression represents the probability that the random variable X (iterated over by 'k') takes on values less than or equal to x.

The CDF is a monotonically increasing function starting from value of 0 till a maximum of 1. For the source and target images above, the CDFs are shown below in Figure 3.3.

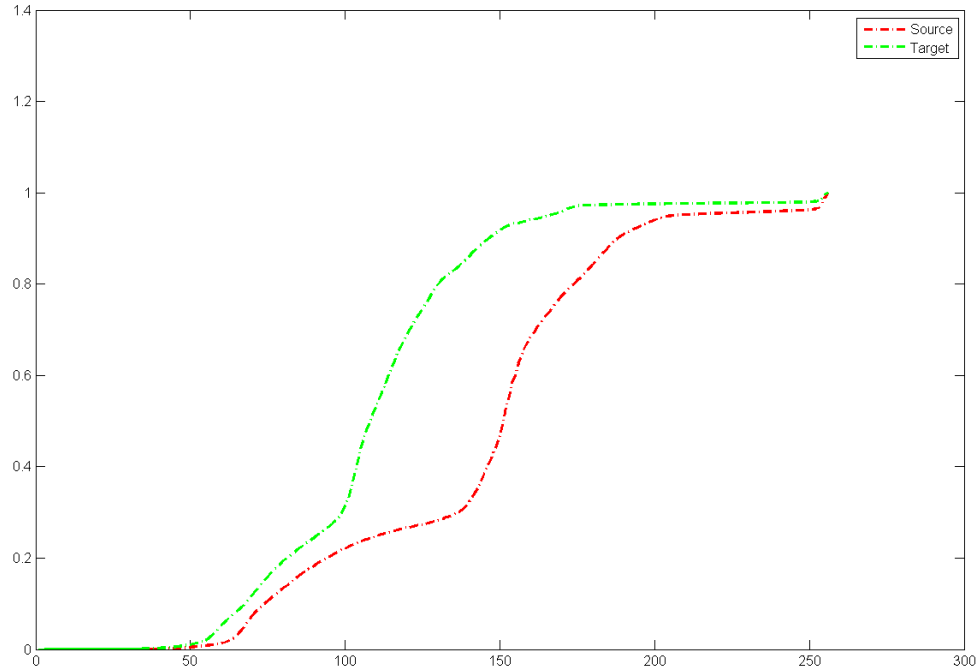

Figure 3.3 Displaying the CDFs for red channel of the source and target images

Once the CDFs F_S and F_T are found, a lookup table is generated between the intensity values and the CDF for both source and the target. The process of histogram matching repeats itself for each and every pixel and works the same way for any pixel.

Let's say a pixel value from source image has an intensity level of s. The matching occurs in 3 steps.

1. For intensity s the value of $F_S(s)$ is read from the table.
2. The value of $F_S(s)$ and $F_T(t)$ are on the same level(see Figure 3.4). The value '$F_T(t)$' is searched in the lookup table of the target.
3. Then an inverse lookup occurs by finding the intensity value t from the table against this $F_T(t)$.

The three steps are shown in the figure below.

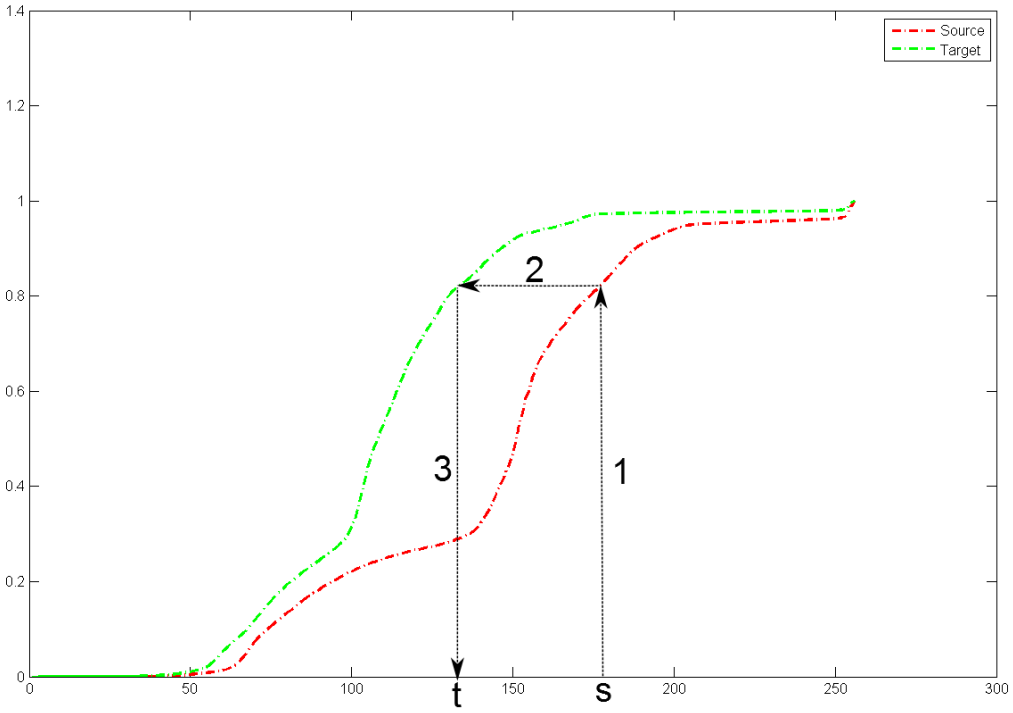

Figure 3.4 Histogram matching. Pixel with value s transformed to t

3D histogram matching in RGB color space can also be undertaken where by the input 3D histogram is being matched to a joint cumulative distribution function [**31**]. However it is not pursued for implementation.

3.2.2 Mean And Standard Deviation Matching

This color transfer algorithm is based on matching the mean and standard deviation of the source to that of the target image. Let the mean and standard deviation of the source be represented as μ_s and σ_s and the mean and standard deviation of the target be represented as μ_t and σ_t.

As discussed in the Section 2.1.3, the mean and standard deviation matching works better when the chromaticity and the luminance are decoupled. Thus, this color transfer algorithm is then applied in the CIELAB space with illuminant E which gives the best results according to [**13**].

Figure 3.5 below depicts the flow of the color transfer algorithm.

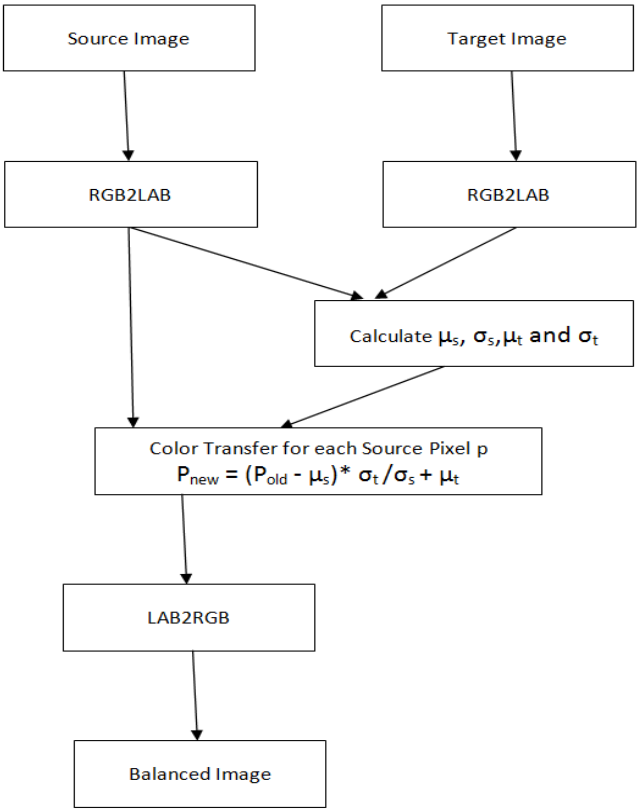

Figure 3.5 Color Transfer by matching means and standard deviations

Figure 3.5 shows color transfer by matching means and standard deviations. The values of μ and σ are calculated for each channel independently. Once the means and standard deviations are known, then the color transfer algorithm is applied on each and every pixel of each channel of the source image. The formula for color transfer is:

$$L_{new} = (L_{old} - \mu_s) * \frac{\sigma_t}{\sigma_s} + \mu_t$$

Here it shows the color transfer only for the L channel of CIELab color space. The means and standard deviation of the source and target are also shown only for the L channel. However this formula is true for each channel independently with their own set of means and standard deviations.

The first term, which is a multiplication of the offset (L_{old} - μ_s) and the ratio of standard deviations σ_t/σ_s, in effect, matches the standard deviation of the source to the target while the addition of μ_t matches the source to the target mean.

After the color transfer has been done, the image is again converted from the CIELab space to RGB. In the process, due to the color transfer algorithm, some values might overshoot and undershoot from the RGB gamut especially if the target mean lies near the upper and lower limits of the RGB gamut. The out of gamut values can either be compressed towards the original source value as in [**23**] or clipped which

is a more favorable option [**32**] . In our implementation, the intensity values are clamped at 255 and 0 respectively.

3.2.3 Color Transfer with Multiple Targets

It is to be noted that in the above explanation any one of the images could be taken as the target. As seen in Figure 1.1, a scene can be represented with more than two scans or images. In such cases it is not wise to only take one of the images as the target. Without prior knowledge of the actual scene, it is very difficult to know which of the images represents the true colors of the scene. Therefore in the MATLAB implementation, all the images of a scene were taken and an average of the pixel values was calculated for all the images combined. Then using these average pixel values, a single mean and standard deviation was calculated. Then the color transfer formula of Section 2.1.3 was applied. This way it is expected that the source would match to a target which is more likely to be the real depiction of the scene.

3.2.4 Interpolation of Means and Standard Deviation

In order to simulate the 3D environment as closely as possible, the images were divided into smaller portions called cells. Then the same process of color transfer algorithm was followed as shown above in Figure 3.5. Only in this case, we worked on each cell separately, with each cell having its own mean and standard deviation in both the source and the target images.

After the local color transfers cell by cell and the conversion of the balanced image back to RGB resulted in seams along the cell boundaries. This was expected because the pixel values in different cells were matched to different target means and standard deviations. This is in comparison to a complete image color transfer where the source image was matched to a single target value for the whole image as in Section 3.2.2.

In order to remove the seams we made use of bilinear interpolation algorithm [**33**] to interpolate the source and target means of the immediate neighbor cells before the color transfer algorithm is applied. This was done by introducing a mask with a size equal to that of a cell but centered on the junction of four cells. By centering it on the cells' junction we made sure that the mask covers one quadrant of four different cells at a time. Then the means and standard deviations of the four different cells were picked up for both the source and the target image. For all the pixels of a cell that fall in the same quadrant of the mask, the set of four neighbors remains the same. The only factor that differs for different pixels is the distance of the pixel from the edges of the mask. The farther the position of the pixel from the neighbor, the lesser the effect of the neighbor's mean and standard deviation for calculation of an interpolated value.

Let Figure 3.6 represent an image with 12 cells, 3 rows and 4 columns of cells. The figure shows that for all the pixels of the top left quadrant of the cell number 6, there is a constant mask which also overlaps cell number 1,2 and 5. Thus for all the pixels in the top left quadrant of cell 6, we would use the means and standard deviations of cells 1, 2, 5 and 6, pass them through an interpolation algorithm(discussed below), and finally get a single interpolated value of the mean and standard deviation. This interpolation

occurs for both the source and target images separately. Similarly if we move to the top right quadrant of the cell number 6, then the neighbor set would be cells 2, 3, 6 and 7.

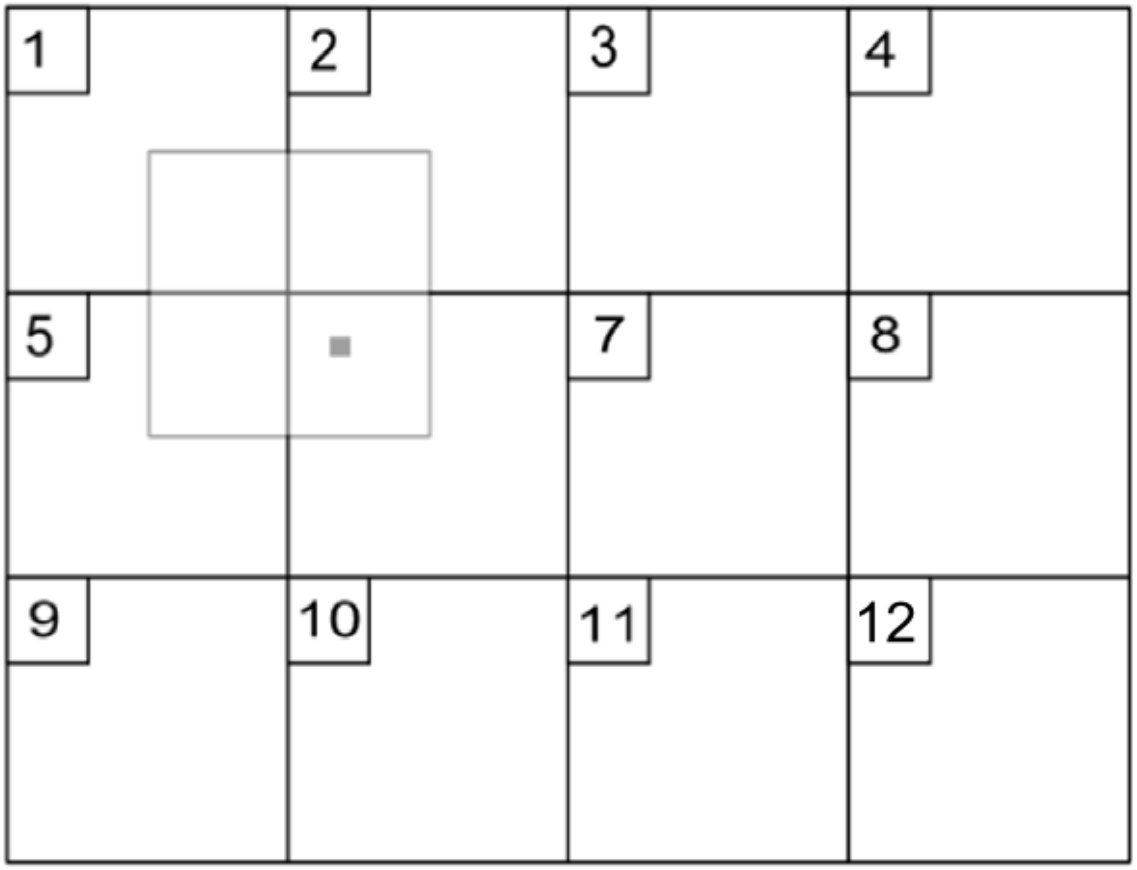

Figure 3.6 Showing a mask overlapping a quadrant each from cells 1, 2, 5 and 6

Let's say that the means of cell 1,2, 5 and 6 be μ_1, μ_2, μ_3, μ_4 and the standard deviations be σ_1, σ_2, σ_3, σ_4 respectively. We need to find a single interpolated mean μ_i, and single interpolated standard deviation σ_i at the location of the pixel shown. Figure 3.7 shows the mask of Figure 3.6 more closely and clearly. The means μ_1, μ_2, μ_3, μ_4 for the cells 1, 2, 5 and 6 are shown at the corners of the mask. μ_{12} and μ_{34} are the interpolated values along the x-axis while μ_i is obtained by interpolating μ_{12} and μ_{34} along the y-axis. The normalized distances to the pixel are shown as dx and dy.

The interpolated values of μ_i is found below as:

First interpolating in the x direction

$$\mu_{12} = \mu_1 * (1 - dx) + \mu_2 * dx$$

$$\mu_{34} = \mu_3 * (1 - dx) + \mu_4 * dx$$

Now interpolating in the y direction.

$$\mu_i = \mu_{12} * (1 - dy) + \mu_{34} * dy$$

Similarly the interpolated value for σ_i is also calculated.

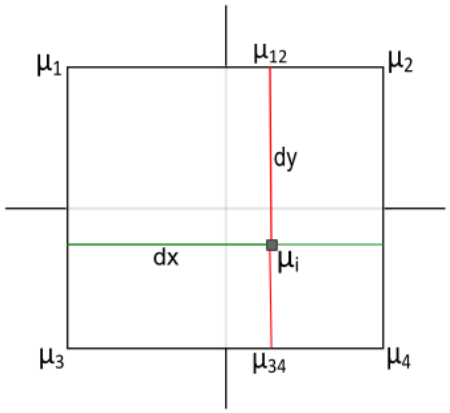

Figure 3.7 A Pixel in the top left corner of Cell 6

The interpolated mean and standard deviation are found for both the source and target. Let these be represented as μ_{is}, and σ_{is} for the source and as μ_{it}, and σ_{it} for the target. Then the color transfer formula for a pixel say L will be

$$L_{new} = (L_{old} - \mu_{is}) * \frac{\sigma_{it}}{\sigma_{is}} + \mu_{it}$$

Now the color transferred pixel value is converted back to RGB.

If in Figure 3.6, a pixel in consideration lies at the boundary of the image, then there would be neighbor cell(s) that would not be a part of the image. In such cases, the empty cells are given the means and standard deviation of the cell in which the pixel lies.

4. Implementation on 3D Point Clouds

Chapter 3 discussed the color transfer algorithms applied on 2D images. The mean and standard deviation matching algorithm was selected to be implemented in 3D. Chapter 4 begins by discussing some basic concepts about the 3D point clouds generated through scans take by LASER scanners. The underlying concepts of a point cloud such as its existence in the world coordinates, and then transformation to a construction space is discussed.

The 3D world is compared to the 2D environment to see in what aspects is the implementation of the selected color transfer is similar to that in 3D as well as what are the differences. Finally the implementation details are explained.

4.1 LASER Scanner Point Clouds

A LASER scanner such as the one shown in Figure 4.1 is capable of generating a point cloud containing millions of points [**3**]. The point density can be adjusted and can provide extremely detailed 3D images.

Figure 4.1 FARO LASER Scanner [3]

A mirror in the scanner rotates so as to reflect the laser beams in all the directions to get a single scan that covers 360 view (except a small region directly below the scanner) around the scanner.

Not all scenes can be captured by a single scan due to obstructions in the path of laser beams or because the target scene may not be in the range. So multiple scans are taken at different positions and all the scans are combined to generate a very large set of points making up a 'Point Cloud'.

4.1.1 Scan point

Each and every point in the point cloud is called a scan point. This scan point consists of two important pieces of information. It carries its own RGB value as well as its coordinates in 3D. The RGB values are modifiable in software after the scans have been taken and this is what is done during the color transfer algorithm.

4.1.2 World Space and Construction Space

A point cloud consists of a hierarchical structure in the form of a workspace tree. Each of the scans in a point cloud have their own space with origin being the position of scanner represented as a scan node. A world space is the concatenation of the scan nodes using their individual transformation matrices towards a root node in the workspace tree to build a common space called world space.

In the software this world space with all the scans and points are converted and reconstructed to another representation called the construction space. This construction is called octree construction which recursively divides a cubic volume into eight cubic sub-volume cubes to a specified depth of the octree. The smallest cube obtained by this decomposition is called a construction cell. All the cells are allotted a unique ID and unique integral coordinates. These coordinates increment by one over each cell in each of the three rectilinear axes. This completes the construction space.

4.1.3 Point Group

A point group is a data structure that holds the scan points in a volume equal to that of a construction cell. There is one point group per scan per cell as long as at least one point from that scan exists in that cell. A point group also holds other information such as the number of scan points it contains, the ID of the scan to which the scan points belong, total space in memory the points hold, the coordinates of the point group etc. The coordinates of the point groups are in Cartesian coordinates and are used to derive a single unique number which acts as the unique cell-ID of the cell the point groups belongs to. The use of the cell-ID will be seen in Sections 4.3.1 and 4.3.2.

At this point it is important to know that a cell might contain more than one point groups if that cell contains point(s) from more than one scan. Similarly there can be cells that only contain one point group because only one scan contributed towards this cell. Some cells in the construction space might not contain a point group at all. This means that this particular cell wasn't in the range of any of the scans in the point cloud.

4.2 Comparison between 2D and 3D

In order to implement the color transfer algorithm in 3D as seen in Section 3.2.2 and Section 3.2.3, it is important to know the possible differences that need to be considered.

Each scan point is equivalent to a pixel with a certain RGB value. Each scan is equivalent to an image. In a 2D image the pixel density is constant throughout the image, which is not true in 3D. In the 2D implementation, there was always a corresponding pixel in the two images(source and target) with the same coordinates. In 3D, there may be cells that might not contain a single scan point from a particular scan. Some cells might contain more scan points from one scan than the others.

In Section 3.2.3, we made use of the neighbors to calculate the interpolated mean and standard deviation. The neighbors always existed except for those cells that lied on the boundaries of a 2D image. In 3D, there will be empty cells irrespective of the position of the scan points. In 2D the mask overlapped quadrants of 4 different cells while in 3D, a mask in the form of a 3D cell covers octants from 8 different cells. This means that in 3D, tri-linear interpolation would be used to calculate interpolated means and standard deviations which is discussed in Section 4.3.3.

Furthermore, because the cases where the neighbor cells are empty appear so often in 3D, and because the empty cells may exist anywhere in the construction space(not only on the borders as in Section 3.2.3), we assign these empty cells with means and standard deviations which are a weighted average of all the non-empty cells around this empty cell. This would be discussed in more detail in Section 4.3.3.

In 2D, while calculating an average value of mean and standard deviation for individual cells of the target images, we always had the same number of pixels in all the target images. In 3D, one cell might contain two or more point groups, where the number of scan points in the point groups might vary significantly.

4.3 Implementation of Color Transfer Algorithm

Having discussed concepts such as the construction space, point groups and scan points, we are ready to understand how the color transfer algorithm is performed on individual scan points.

4.3.1 Populating required data for Color Transfer

The formula for color transfer is the same as discussed in Section 3.2.2 which is:

$$L_{new} = (L_{old} - \mu_s) * \frac{\sigma_t}{\sigma_s} + \mu_t$$

Here we first convert a scan point's color from RGB to CIELab. This formula shows the color transfer applied on only the L channel of CIELab, whereby the values of μ_s, σ_s, μ_t and σ_t are means and standard deviation values for the L channel of the source and of the target respectively. This means that before

we can apply the color transfer formula, we need to get the values of μ_s, σ_s, μ_t and σ_t for both the source and the target.

In 3D, the source is one point group which occupies a particular 3D cell. One point group can have a variable number of scan points in it ranging from 1 to several hundreds or even thousands of scan points. The source mean μ_s and source standard deviation σ_s are calculated by visiting each scan point in the point group one by one, converting the RGB value of the scan point to CIELab space, and then calculating the mean and standard deviation of each of the L, a, and b channels independently. As discussed in Section 4.1.3, that a point group holds information such as number of scan points it contains, scan ID etc., so when the mean and standard deviation for the scan points have been computed, the point group also holds this information. This procedure is repeated for each point group of each scan in the point cloud as shown in Figure 4.2.

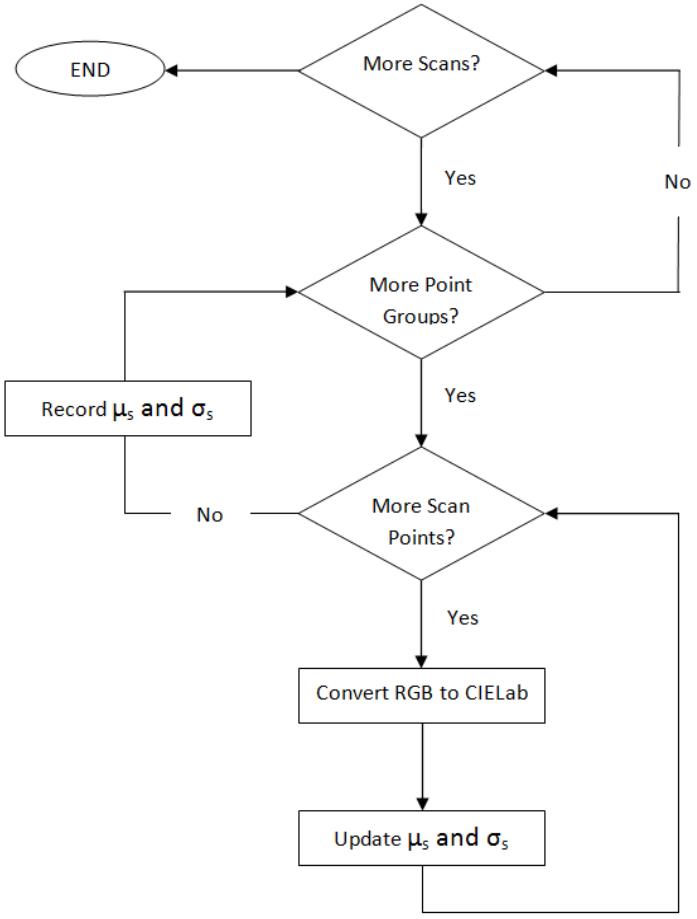

Figure 4.2 Calculation of Source's means and standard deviations

Now we have the source means and standard deviations for all the point groups belonging to a particular scan. We also need the target means and standard deviations. The target for the color transfer algorithm is the cell as a whole where one or more point groups may fall. In 3D, the target

values are calculated by the weighted average of the individual means and standard deviations of the point groups falling in that cell. Given that we have two point groups falling in the same cell called PG1 and PG2. Let the means of PG1 and PG2 be μ_1 and μ_2 respectively. The total number of points in PG1 and PG2 are n1 and n2 respectively. Then the combined mean of the cell (also the target mean) would be μ_t and will be calculated as follows.

$$\mu_t = \frac{(\mu_1 * n_1) + (\mu_2 * n_2)}{n_1 + n_2}$$

Similarly the weighted standard deviation is also calculated. The same procedure is carried out for all the cells where at least one point group lies.

Here it is important to note that each cell has a unique ID. The target values of mean and standard deviations as calculated above are then stored in an ordered associative STL container 'map'. The map holds pairs of values for each cell whereby a single pair consists of <cell-ID ,μ_t and σ_t>. The map is ordered with respect to the cell-IDs.

We also have one map each for individual scans where the map's key correspond to the cell-ID of the point groups and the value of the map corresponds to the means and standard deviation of the point group in that cell. There is only one point group per cell per scan. The pair for each entry of this map is <cell-ID ,μ_s and σ_s>. This map is used to provide the source values for the color transfer algorithm of means and standard deviation matching as will be discussed in section 4.3.3. There is one such map for each scan in the point cloud.

4.3.2 Implementation Flow Graph

Now we have the means and standard deviations for the source and target ready. The source means and standard deviations exist as members of the point groups. The target means and maps exist in the map container as discussed in Section 4.3.1.

Next step is to apply the color transfer algorithm on every scan point in the point cloud. All the point groups that exist in the point cloud are collected into a vector of point groups. Each point group is visited in the vector and then each scan point of this point group is visited one by one. The RGB values of the scan points are first converted to CIELab color space. The color transfer algorithm is applied whereby the values of μ_s and σ_s are obtained from the point group itself whereas the values of μ_t and σ_t are obtained from the map as described in Section 4.3.1. Next the new color transferred Lab values are converted back to RGB and the scan point's color is updated. Graphical representation of the flow is shown in Figure 4.3.

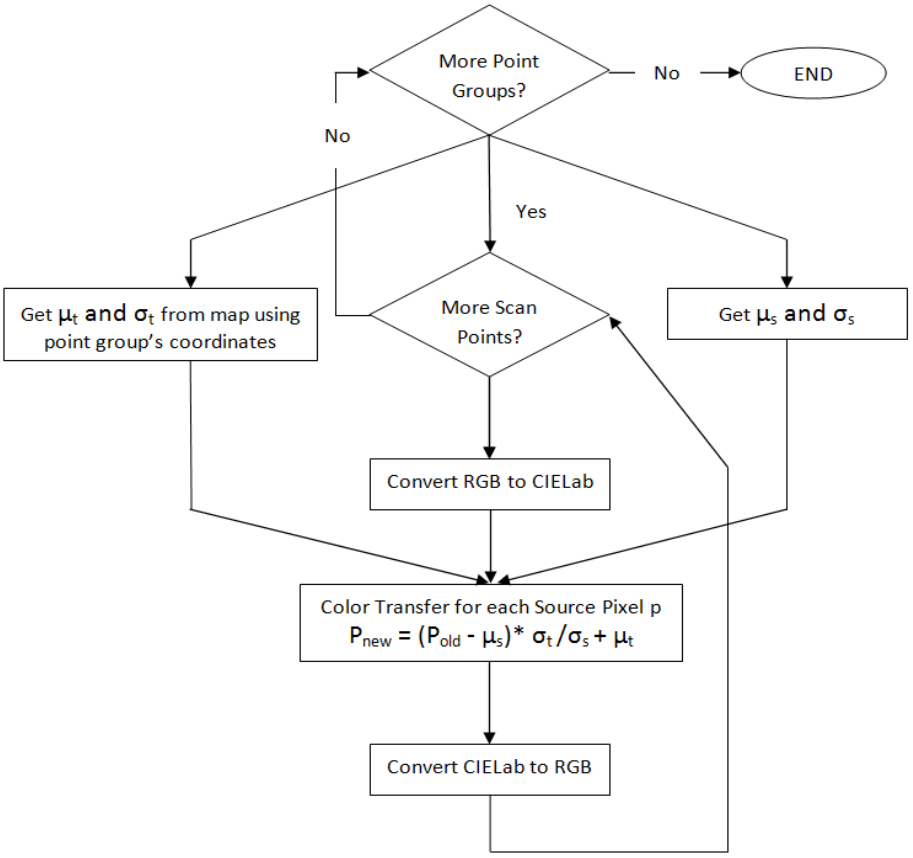

Figure 4.3 Flow Graph of Color Transfer Algorithm in 3D

4.3.3 Interpolation of Neighbor's Means and Standard Deviations

In Section 4.3.2 we applied the color transfer algorithm where the means and standard deviations all belonged to the same cell. Similar to the simulation in MATLAB in Section 3.2.2, we would expect seams between the 3D cells. Thus we need to interpolate the means and standard deviations of both the source and the target.

For the source, the neighbor cells also belong to the same scan to which the current point group belongs.

The 3D implementation in C++ is shown as pseudo code in the table below. The vector holding the point groups is called pGVector. Using an iterator to iterate over all the point groups in the vector, each point group (pG) is dereferenced from the vector one by one in a while loop. Using the coordinates of the point group, the IDs of all the cells surrounding the cell of this point group and the current cell itself are found. In 3D there are 26 cells surrounding the central cell where the point group falls. This is in the form of a rubik cube as shown in Figure 4.4

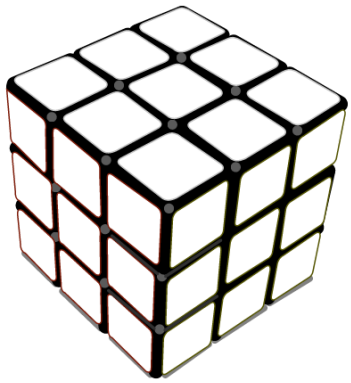

Figure 4.4 Rubik Cube: The point group lies in the center most cell

Once the IDs of all the 27 cells are found, then the maps populated in section 4.3.1 for both the source and the target are used to find the means and standard deviations of all these cells. One of the members of the point group is the scan-ID which holds a unique number against all the scans in the point cloud. This scan-ID helps to identify and pick the correct map for the source. Using the cell IDs and this map, the means and standard deviations of the 27 cells are extracted from the map and stored in arrays called meansOfNeighbors and stdDevOfNeighbors respectively.

Similar procedure is applied to extract the 27 means and standard deviations of the target using the target map. When the arrays of means and standard deviations for both the source and target are known, these arrays remain constant for all the scan points (totScanPoints) in the point group.

Color Transfer Algorithm	
1:	**Function colorBalance()**
2:	*pGVector::iterator iter = pGVector.begin()*
3:	**while** *(iter != pGVector.end())*
4:	*pG = *it*
5:	*neighborIDs= pG.findNeighbors(pG.coordinates)*
6:	*meansOfSourceNeighbors = pG.calcMeansOfNeighbors(neighborIDs)*
7:	*stdDevOfSourceNeighbors = pG.calcStdDevOfNeighbors(neighborIDs)*
8:	*meansOfTargetNeighbors = pG.calcMeansOfNeighbors(neighborIDs)*
9:	*stdDevOfTargetNeighbors = pG.calcStdDevOfNeighbors(neighborIDs)*
10:	*totScanPoints = pG.getTotalScanPoints()*
11:	*i = 0*
12:	**while** *(i != totScanPoints)*

```
13:  scanPoint = pG.getScanPoint(i)
14:  colorTransfer(scanPoint)
15:  i++
16:  end
17:  iter++
18:  end
19:  end
20:
21:  Function colorTransfer(scanPoint)
22:  rgb = scanPoint.getColor()
23:  lab = rgb2Lab(rgb)
24:  μs = interpolate(meansOfSourceNeighbors, scanPoint)
25:  σs = interpolate(stdDevOfSourceNeighbors, scanPoint)
26:  μt = interpolate(meansOfTargetNeighbors, scanPoint)
27:  σt = interpolate(stdDevOfTargetNeighbors, scanPoint)
28:  lab = (lab -μs) *σt/σs +μt
29:  scanPoint.rgb = lab2rgb(lab)
30   end
```

Next all the scan points of the current point group are visited one by one. For each scan point the target is to transfer its rgb color to a new value by using the color transfer algorithm, shown as colorTransfer(scanPoint) in the pseudo code.

In this function, first of all the rgb color value is extracted from the scan point. Then the rgb value is converted to the Lab color space. Then we apply the tri-linear interpolation algorithm to find μ_s, σ_s, μ_t and σ_t which are all interpolated values.

The interpolation algorithm is applied in the same manner as in section 3.2.3 in the 2D case. The difference here is that we have three axes instead of two. Therefore we need eight values instead of four, on which the tri-linear interpolation algorithm is applied. Similarly as in 2D a cell is divided into four quadrants, here the cell is divided into eight octants. Figure 4.5 shows one 3D cell divided into eight octants, one of which is colored gray.

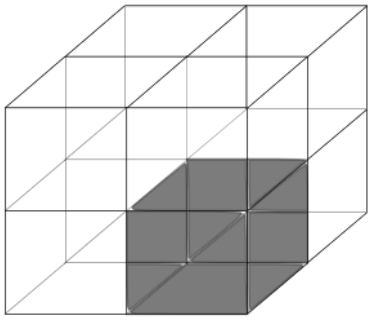

Figure 4.5 Showing 8 octants of a 3D cell

Depending upon the coordinates of the scan point, it would be determined which of the eight octants the scan point falls into. Once the correct octant is known for the scan point, a mask the same size as a 3D cell, is applied on this octant in such a way that only one of the octants of the mask overlap with the scan point's octant. The rest of the seven octants of the mask overlap one octant from seven different neighboring cells. One can visualize it using the rubik cube as well, where the mask size is equal to any one of the twenty-seven smaller cubes. For each of the eight octants of the central cell of the rubik cube, the mask will overlap one octant of eight smaller cubes. For each octant of the central cell, the identity of the eight cells are always constant. Figure 4.6 shows that a mask shown in red color overlaps one octant of blue cell. The scan point lies in the overlapping octant shown in gray. The other seven octants of the mask overlap one octant of seven different cells.

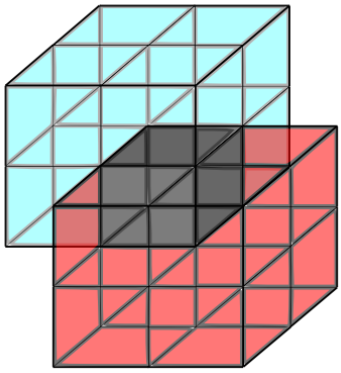

Figure 4.6 Red Mask overlapping one Octant(gray) of the Blue Cell

When visiting each scan point, the coordinates of the scan point help in finding the octant where it lies, which in turn helps to find the eight cells that the mask overlaps. The means and the standard deviation corresponding to these eight cells are picked up from the arrays which were populated in lines 6-9 of the pseudo code. The interpolation algorithm is applied on the means and standard deviations of these eight cells, here forth referred as μ_1, μ_2, μ_3, μ_4, μ_5, μ_6, μ_7, μ_8 and σ_1, σ_2, σ_3, σ_4, σ_5, σ_6, σ_7, σ_8 respectively.

The normalized distance of the scan point from the center of its cell, is found using the scan point's coordinates and referred to as dx, dy and dz.

The tri-linear interpolation algorithm is basically a combination of bilinear interpolations in x and y axes followed by a linear interpolation in z axis [**34**]. As seen in the algorithm, the interpolate() method receives the scan point itself and the arrays of means and standard deviation of the source and target. The coordinates of the scan point helps to find and extract the values of $\mu_{1\text{-}8}$ and the $\sigma_{1\text{-}8}$ as described above. The interpolation algorithm works in the same way in each of the lines 24-27 in the pseudo code. Below only the calculation of μ_s is described.

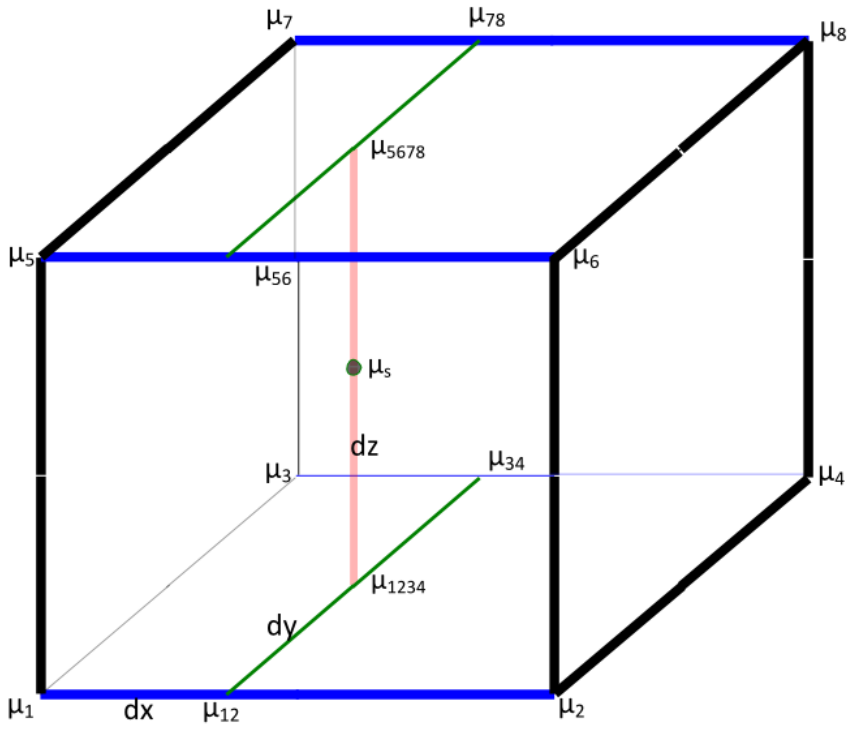

Figure 4.7 Mask with eight mean values at its corner

Figure 4.7 shows a mask with a scan point shown as black dot. It is at a distance of dx, dy and dz from the front left-most corner in each of the x, y and the z axes. The values of the eight neighbor's means are shown on the corners as $\mu_{1\text{-}8}$.

First we interpolate in the x direction shown in blue color to find the values of μ_{12}, μ_{34}, μ_{56} and μ_{78}.

$$\mu_{12} = \mu_1 * (1 - dx) + \mu_2 * dx$$

$$\mu_{34} = \mu_3 * (1 - dx) + \mu_4 * dx$$

$$\mu_{56} = \mu_5 * (1 - dx) + \mu_6 * dx$$

$$\mu_{78} = \mu_7 * (1 - dx) + \mu_8 * dx$$

Now we interpolate the values of μ_{12}, μ_{34}, μ_{56} and μ_{78} in the y direction to find the values of μ_{1234} and μ_{5678}. The distance is shown as dy on the green line in the y direction.

$$\mu_{1234} = \mu_{12} * (1 - dy) + \mu_{34} * dy$$

$$\mu_{5678} = \mu_{56} * (1 - dy) + \mu_{78} * dy$$

Now we need one another linear interpolation in the direction of z shown as red line to find μ_s

$$\mu_s = \mu_{1234} * (1 - dz) + \mu_{5678} * dz$$

Now we have the interpolated value of source mean. Similarly we get the interpolated source standard deviation as well as the target mean and standard deviation.

When the values of μ_s, σ_s, μ_t and σ_t are known, then the color transfer formula is applied on each of the three lab channels as shown below

$$\text{lab} = (\text{lab} - \mu_s) * \frac{\sigma_t}{\sigma_s} + \mu_t$$

Next the lab values are converted back to rgb and the scan point's color is modified.

4.3.4 Color Balancing Artifacts - Stains

While color balancing, a scenario can arise when structures such as roof, floor or walls are intersected by an edge of a construction cell. If there are two scans for instance with scan points covering a wall, it can happen that part of the scan points for each scan lie in two different cells on either sides of the cell boundary which is intersecting the wall.

Figure 4.8 shows a cross section of a wall whose surface consists of scan points from two scans shown in red and green. The boundary between cell 1 and cell 2 intersects the wall. This has the effect that the scan points are now distributed in two different cells. As seen before, color balancing occurs cell by cell. For cell 1, almost all the points are red in color and therefore, the target mean and standard deviation is more inclined towards red cast. For cell 2, there is a uniform mixture of red and green scan points and the target is more inclined to a yellow cast.

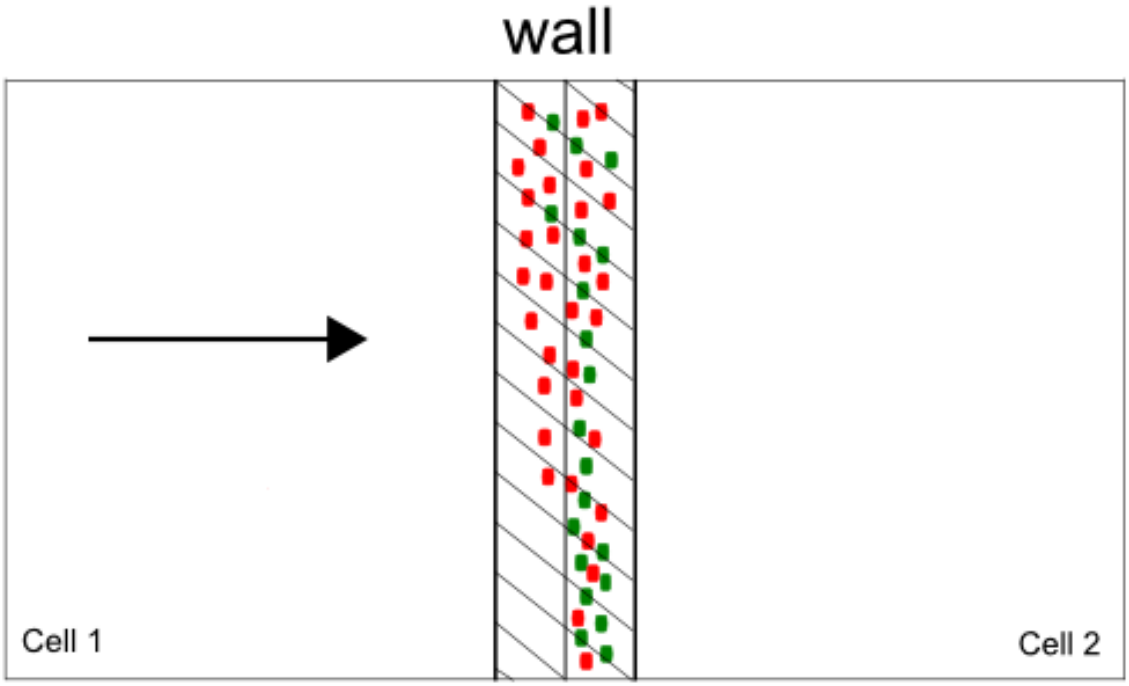

Figure 4.8 Cross section of a wall manifesting the Stain Artifact

Now after color balancing, if the same wall is seen in the direction of the arrow shown, what would be observed is shown in Figure 4.9. In general, with many more scans contributing to the points on the wall, this artifact causes some stains to appear in the view. This is shown in Figure 4.10.

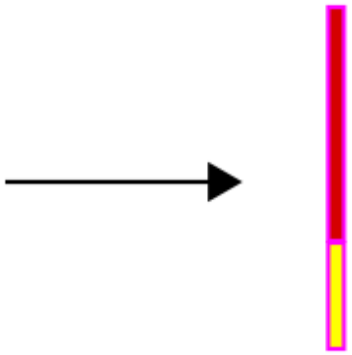

Figure 4.9 Red and Yellow cast on the wall after color balance

Figure 4.10 A wall showing a brownish stain

It is important to note that this artifact is not due to the color balancing algorithm itself but due to the spatial arrangement in the construction space. Interpolation discussed in Section 4.3.3 does try to remove the stains but interpolation has little effect here because most of the cells around cell 1 and 2 are empty as one would expect in the front and back of the wall. However interpolation does diminish the effect to only a little extent by spreading the stain. A workaround is to incorporate a new procedure to find the values of $\mu_{1\text{-}8}$ and the $\sigma_{1\text{-}8}$ for the neighbors before the interpolation algorithm is applied as in Section 4.3.3. For any cell where a scan point lies, the values of μ and σ for the neighbors of this cell are now calculated differently. Let Figure 4.11 show N as one of the neighbors. The calculation of μ_N for a neighbor cell is depicted in 2D for clarity where the number of neighbors for cell-N are 8. In 3D, the neighbors are 26.

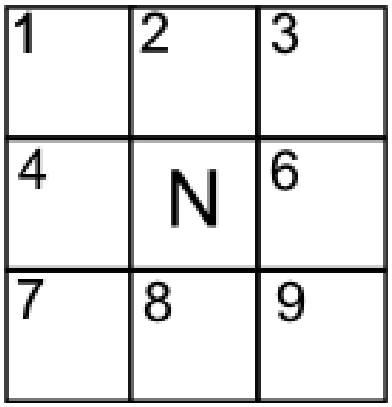

Figure 4.11 Showing 8 neighbor cells of the Cell N

The value of μ_N for the cell N is calculated by taking the weighted mean of all the cells in the neighborhood of N. The formula below shows the calculation for μ_N .

$$\mu_N = \frac{\sum_{i=1}^{9}(\mu_i * n_i)}{\sum_{i=1}^{9}(n_i)}$$

μ_i and n_i refer to the individual mean and the number of scan points of the i^{th} cell. The formula iterates over the 9 cells shown in Figure 4.11. Some of the cells 1-9 might be empty in which case the summation term in the formula also results in zero. In 3D, the formula remains the same except that i iterates from 1 to 27.

5. Evaluation

In this chapter the results of the implementation of Chapters 3 and 4 are discussed. Chapter 3 involved the implementation of two color balancing algorithms in 2D. The two algorithms are Histogram Matching and color transfer by matching means and standard deviations. After discussing the results of Chapter 3, the results of the color transfer by matching means and standard deviation in 3D are discussed.

5.1 Histogram Matching in 2D

Figure 3.1, reproduced here from chapter 3, shows two images of the same scene taken in the morning and evening respectively using FARO Focus 3D LASER Scanner.

Figure 5.1 Morning and Evening images of the same scene

As described in section 3.2.1, the histogram matching aims to match the histogram of one image(the source) to the histogram of another image(the target). The histogram matching was performed in both

the RGB space as well as CIELab space. The morning image was histogram matched to the evening image and vice versa.

Figure 5.2 Morning image matched to Evening image.

Figure 5.2 shows the effect of matching the morning image to the evening image by histogram matching. As a whole the source image took the color of the target image quite well, but some artifacts appeared near the brighter white pixels. This is known as contouring effect [**35**] and happens when the nearby pixel values in a histogram all saturate at a certain brightness value.

Figure 5.3 shows the histogram of the 3 color channels RGB in three rows. Each row shows the effect of matching one of the color channels of the source to the target. The result of histogram matching is then shown in blue color. In the histograms as well we see pretty good matching of the source to the target. The blue lines follow the general flow of the green lines. However the contouring effect is also evident. It can be observed easily for the red and green channels in the gray range of 105 to 125. Here the blue lines don't follow the green lines exactly. Instead some spikes followed by dips are visible. This is because the number of pixels at the dips are accumulated at the gray level of the spikes exhibiting the contouring effect. The same happens near the end at the brighter pixel values whose effect is seen in Figure 5.2 near the windows.

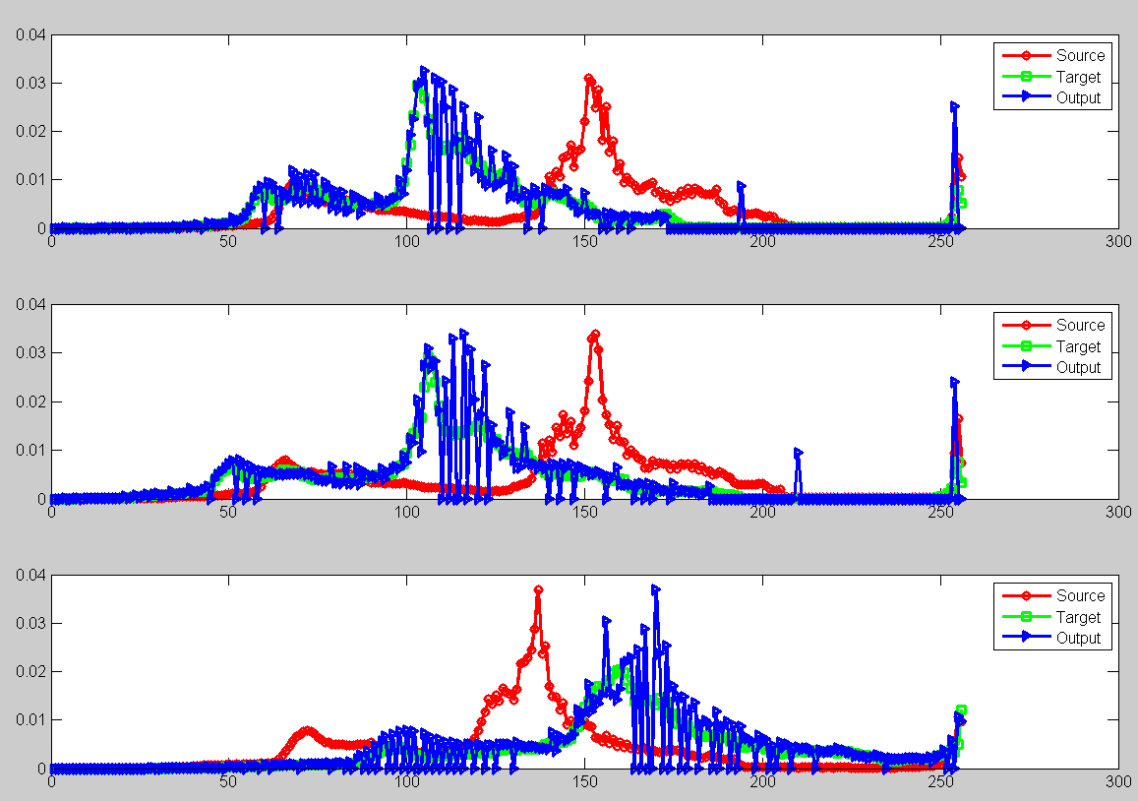

Figure 5.3 RGB channels of morning image matched to target image

As discussed in section 3.2.1, the histogram matching was also performed in the CIELab color space. This was done to see whether the histogram matching gets better if the brightness and chromaticity are decoupled. However one problem in CIELab is that there exist infinite pixel values in a closed range. In RGB we had fixed number of levels in each of the R, G and B channels i.e. 256 levels each. In CIELab, the chromaticity channels of a and b remain close to zero and it seldom happens that we get pixel values near to the end of the dynamic range.

Figure 5.4 shows the luminance(L) and the chromaticity(a and b) channels of the source, target and the matched images. The number of bins were high i.e. 2000 compared to 255 in RGB to accommodate smaller differences in pixel values. But as can be observed, most of the bins in the whole dynamic range are empty. Overall matching looks satisfying but the artifacts increase further.

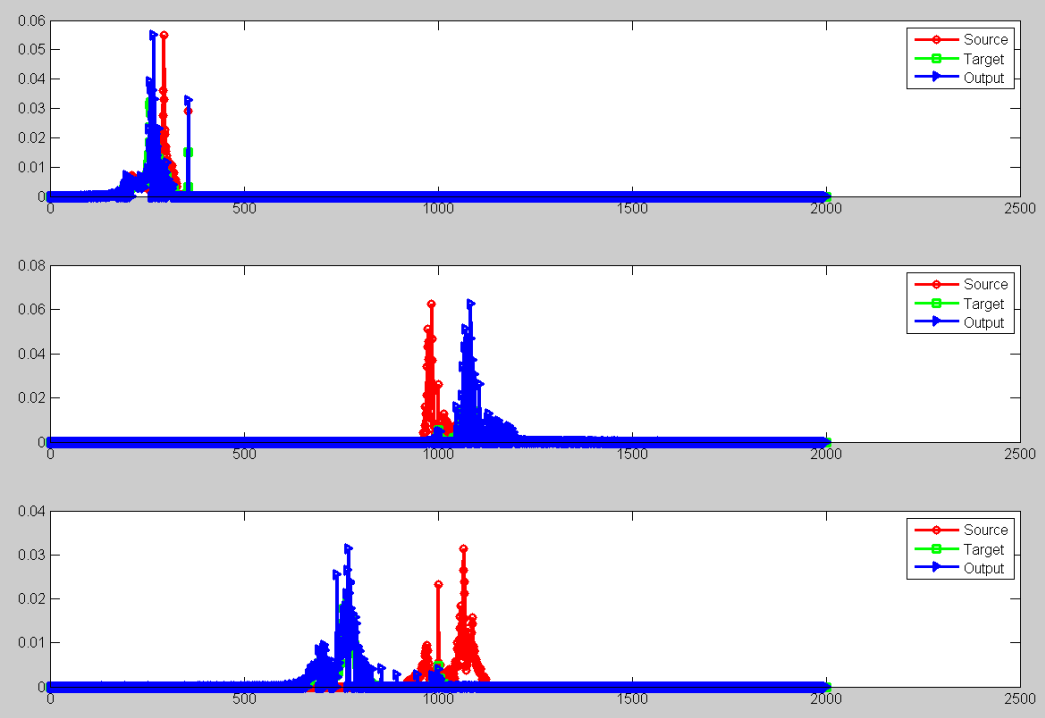

Figure 5.4 Histograms for the L, a, and b channels of Source, Target and Matched Image

Figure 5.5 Morning Image matched to Evening in CIELab space

It can be seen in Figure 5.5 that the contouring effect has further increased. This is because the pixel values specially in the a and b chromaticity channels are so near to each other that there are more chances to saturate to a particular value.

We observed that the general tendency to match to the target existed in histogram matching algorithm but the artifacts introduced were not acceptable. Next the statistical parameters of means and standard deviation of the source image were matched to the target and the results are shown in the next section.

5.2 Color Transfer algorithm in 2D

In this section, first the effect of the color transfer will be seen on the images shown in Figure 5.1. Then, in order to simulate the 3D point cloud which consists of 3D cells holding scan points, the effect of color transfer would be seen on images divided into cells. Then the effect of interpolation algorithm would be seen. At last the performance of the algorithm would be shown on other data sets as well where the targets are a combination of many source images.

The rather simple color transfer algorithm of means and standard deviation matching performs really well without any contouring artifacts or others. Figure 5.6 shows four images. Image 1 and 2 are the morning and evening images. Image 3 is the result of matching the means and standard deviations of image 1 to image 2 while image 4 is the result of matching image 2 to 1.

It is to be noted that the mean and standard deviation of the complete image were matched to the target. It is easily observed that both the original images match to the other very well.

Compared to histogram matching, there aren't any contouring effects as well specially near the windows. Once the performance of this color transfer algorithm was seen as impressive, the histogram matching approach was dropped and the color transfer by means and standard deviation matching was adopted. No further color transfer algorithms were investigated because this method already provided excellent results.

The next task was to divide the image into smaller portions called cells. The original description of the color transfer algorithm in [**15**] was to transfer the characteristics of one image to another irrespective of the scene they display. However it is also mentioned that the transfer of color is more accurate if the two images represent the same scene. In the point clouds this is always the case that we have the same scene but different color statistics.Therefore the more the images already match each other; the better is the color transfer. Thus it was easy to deduce that if the matching of the mean and standard deviation is performed cell by cell, the statistics of the individual objects in the images would be better conserved. Therefore instead of matching the mean and standard deviation of the entire source image to the target, color transfer was performed by matching the means and standard deviations of individual cells of the corresponding source and target images.

This division into cells is more importantly driven by the underlying cell structure in 3D point clouds.

In Figure 5.6, if we compare image 2 with 4, it is noticeable that the building behind the windows have faded after color transfer.

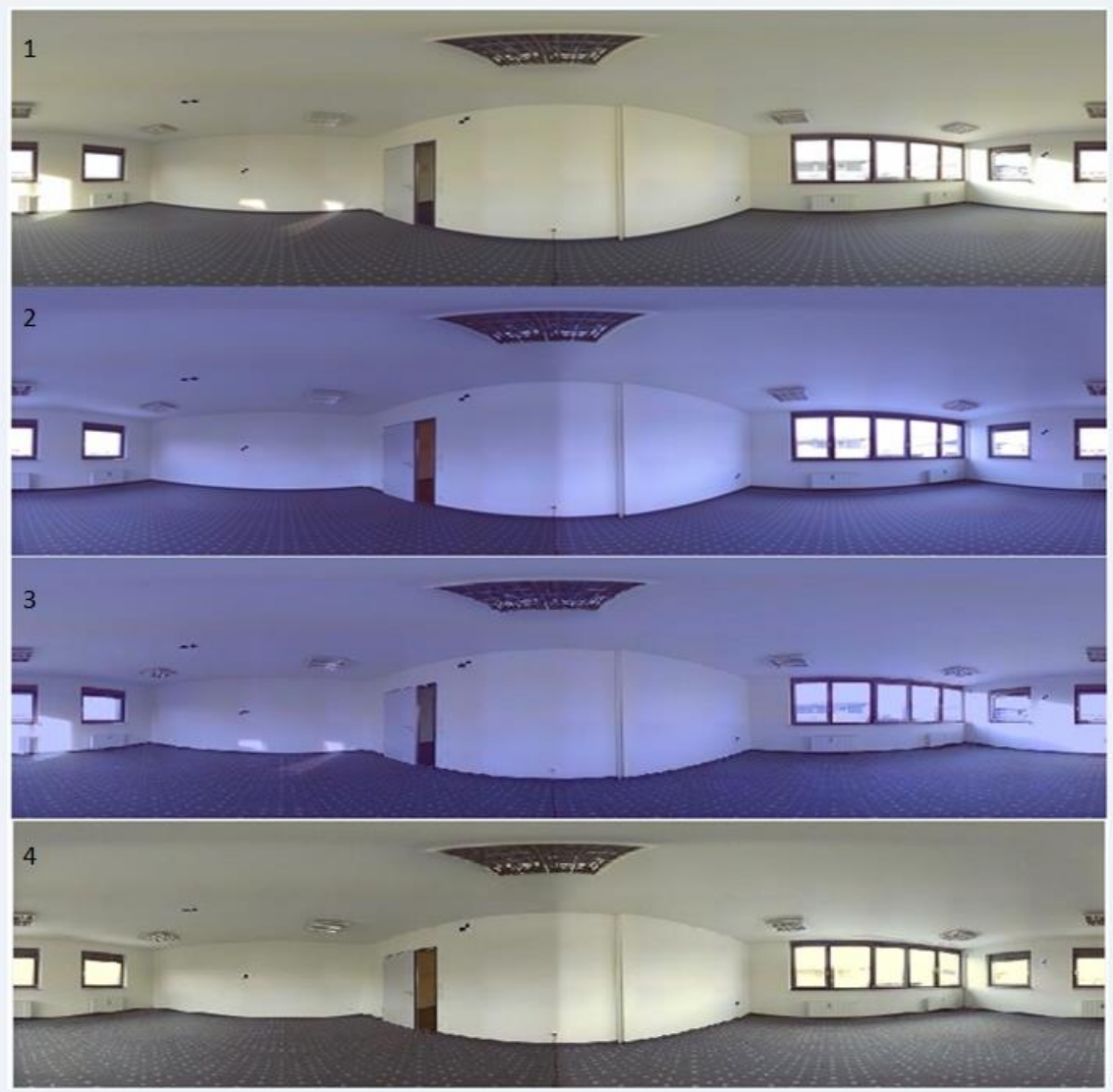

Figure 5.6 Image 1-2: original images. Image 3-4: matched images

Figure 5.7 shows the result of matching the evening image to the day image by first dividing the images into cells, and then matching the means and standard deviation cell wise. The cell size can be observed due to the seams that have appeared due to cell-wise color transfer. The seams are an undesirable effect but it is also observed that the color transfer is sharper in this case. If the same building is observed behind the right windows in Figure 5.7 and in image 4 of Figure 5.6, we see that the texture of the image is retained.

Figure 5.7 Cell to Cell Color Transfer

Section 3.2.3 discussed the process of interpolation of the means and standard deviation to remove the seams as seen in Figure 5.7. We still do the same cell by cell color transfer however the values chosen for the means and standard deviation for both the source and the target are not solely the values obtained in the particular cell in which the color transfer is performed. Instead the mean and standard deviation are calculated in such a way so as also to get the effect of the means and standard deviations of the immediate neighbors. This is what the interpolation algorithm does in general as it gives different weights to the neighbors' values depending upon the distance of the individual pixels from the neighbors. The neighbors' means and standard deviations values remain constant, but the distance between the neighbors and the pixels changes gradually and this is what brings in a smoothing effect and thus removes the seams between the cells.

Thus it can be said that for any cell in 2D, there is effect of the statistics of eight more cells that surround this cell. The comparison of the region right under the left window in Figure 5.7 with the image four of Figure 5.6, we observe that there is a very sharp gradient in the brightness levels between the cells. The interpolation algorithm has the effect of spreading this effect between the cells that are close to each other.

Figure 5.8 shows the same image as in Figure 5.7 but after interpolation. We observe that there are no seams at all. Also the sharp gradients between the brightness levels between individual cells have been spread out and also we conserve the texture of the objects by virtue of cell by cell color transfer as we still see the building behind the right windows.

Figure 5.8 Result after Interpolation

So far we have seen the viability of the color transfer. It certainly does match the target as closely as possible. The next step is to find the right statistics of the target or in other words the right target itself. In Figure 5.6 image 1 was correctly matched to image 2 but image 2 is not the actual color of the room.

For this purpose another test was done in the same room as shown in the figures above. A high power lamp and some colored foils were used to produce colored shades in the room. The colors of the foils were the red, green and blue. The aim was to combine the effects of the three shades into a common target which might be a more accurate representation of the actual color of the scene.

Figure 5.9 shows three images of the room with red, green and blue shades respectively. It is to be noted that the light was focused towards the corner from where it was expected to reflect and diffuse in the room as much as possible. However the ambient effect as that with the sunlight couldn't be created. Also, it is not expected that the region of direct focus of the lamp light will be correctly color transferred. The shades on the walls around the focus point are expected to be color transferred to a new target which is common for the three images and more plausibly represent the actual color of the room.

Figure 5.9 Showing Red, Green and Blue shades

The combined target would have characteristics such that it would have means and standard deviations which will be the weighted average of the means and standard deviations of the individual images. The

weights correspond to the number of pixels each of the source images have. In each cell in 2D, the number of pixels is always equal. However in 3D one of the scans can have more number of scan points than the other. In that case, the color of the scan that has a higher number of scan points dominates.

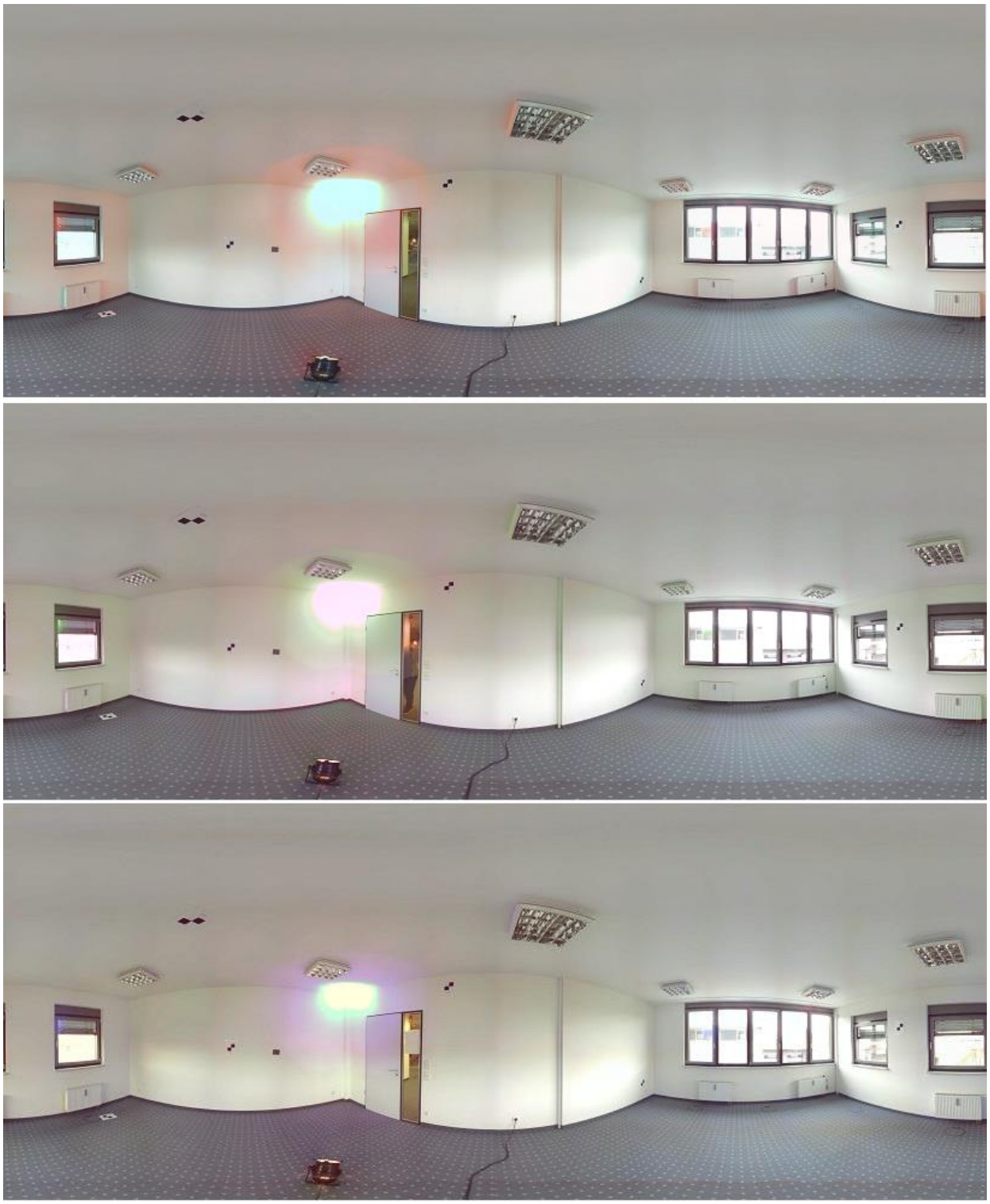

Figure 5.10 Red, Green and Blue Images matched to a combined Target

Figure 5.10 shows the effect of color transfer whereby the red, green and the blue images were matched to a common target statistics. We observe that around the lamp focus point, the three images tend to move towards a common target. The individual shades of red, green and blue have diminished significantly. However since the red light was the strongest of the three, its effect still remains and is seen in all three of the images. Even at the corner where the lamp light was focused the effects of the individual colors have been reduced. Availability of any ambient source would have shown even better and clearer results.

5.3 Color Transfer Algorithm on 3D Point Clouds

In this section first of all the artifacts caused by the color inconsistencies are shown. These inconsistencies are the driving factor for the color balancing. Then the effect of color transfer on the scans is shown without interpolation. Then the final results are shown with the application of tri-linear interpolation.

For experimentation in 3D, a point cloud consisting of two scans was taken. The two scans contain scan points with purple and blue color casts as seen in Figure 5.11. The scans show a wall in a museum.

Figure 5.11 Point Cloud showing two Scans

The difference in the color is because the scanner was positioned in different locations and the scans were taken at different times. This resulted in different luminance and different chromaticity. The

intermingling of the scan points from the two scans in Figure 5.11 gives a very unpleasant effect to the viewer. Thus color balancing was employed to match the two scans to a common target so that the resulting image looks smooth and a better representation of the actual scene.

The white spots in the wall mean that there existed no scan point from either of the scans. This is a low resolution point cloud where the number of scan points from each scan has been significantly reduced. For testing, the low resolution scans were used because of faster point cloud generation. The higher resolution point cloud doesn't show any white spots.

It is also to be noted that in Figure 5.11, in the regions on the wall where only purple scan points are visible, it doesn't mean that the blue scan points do not exist there. A 3D cell is a cube whereby the scan points can lie anywhere. The 2D snapshot of the wall doesn't show the blue points that lie behind the purple points.

Figure 5.12 shows color balancing by matching the individual means and standard deviations of the scans cell by cell to the weighted mean and standard deviation target value for all the scan points existing in the point cloud. The weight corresponded to the number of scan points each scan contributed in that cell.

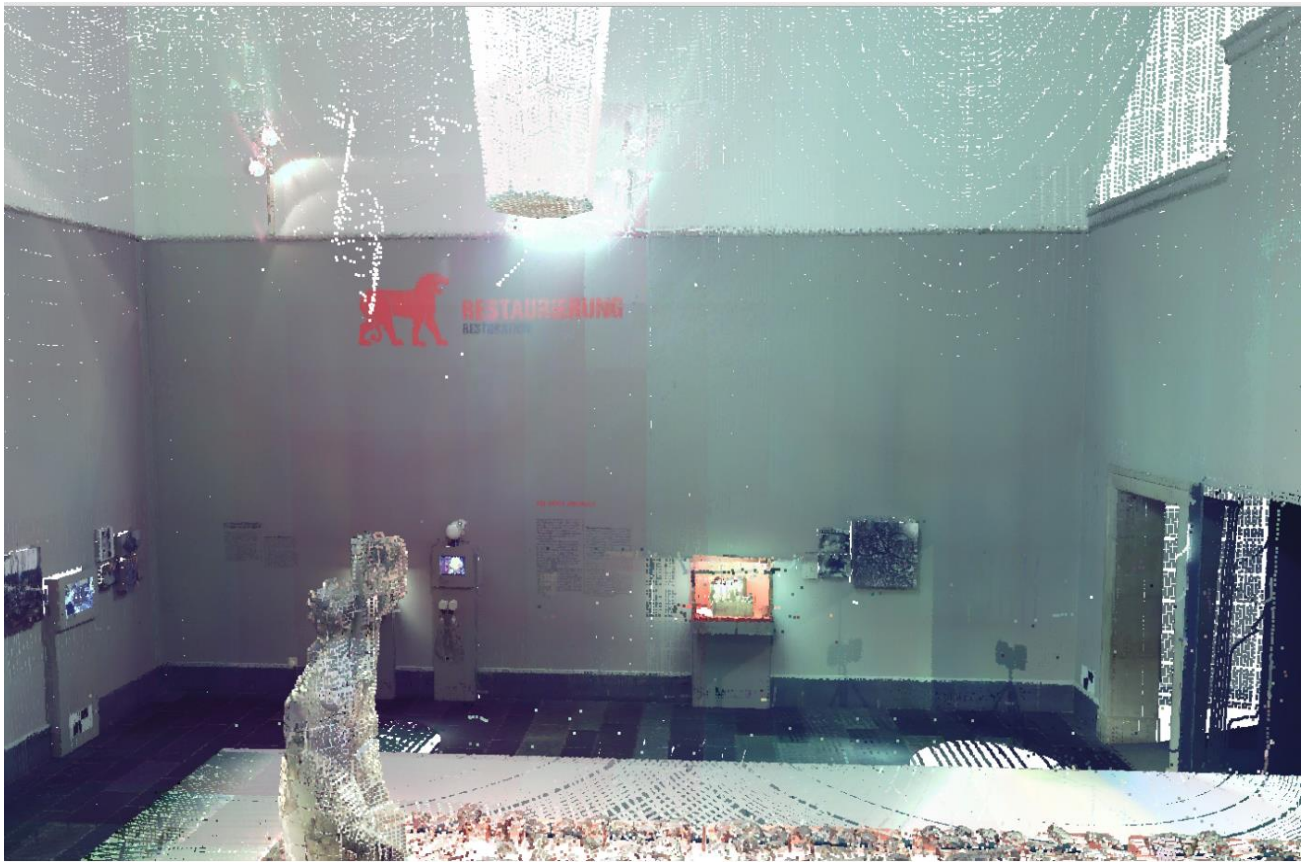

Figure 5.12 Color Balanced without Interpolation

It is observable that in the portion of the wall where one could see only the purple scan points in Figure 5.11, now a bluish effect can also be seen. This is the contribution by the blue scan points that also existed in that cell. Overall one can see a smoothness in the wall which wasn't there in the point cloud without color transfer.

The streaks of blue scan evident in the Figure 5.11 are now removed. Towards the right side of the wall, the effect of the purple scan points is removed and the cell shows more tendency towards the blue scan. This is because in this region the purple scan points were too few as compared to the blue scan points to have much effect in the calculation of a combined mean and standard deviation.

Because the color transfer was done cell by cell where each cell had different statistics with respect to the source and target means and standard deviations, as expected we see vertical and horizontal seams at the cell boundaries.

Figure 5.13 shows the final image of the point cloud after color transfer with interpolation. Now no seams exist between the cell boundaries.

Figure 5.13 Final Point Cloud after Color Transfer with Interpolation

A smooth gradient exists between the two scans. As discussed in Section 4.2 that for any cell in 2D the neighbor cells always exist except on the borders. In 3D, many cells might be empty and the value of the means and standard deviations for these empty cells is calculated by taking the weighted average of the means and standard deviations of its own existing neighbor cells. Thus, after the interpolation in 3D, the color transfer algorithm in any cell would exhibit the influence of not only its own neighbors but also the neighbors of the neighbors. This results in even more smoothness in the gradient as is evident in Figure 5.13.

Figure 5.14 shows a snapshot of another point cloud which is a mixture of several scans. There are some scans which aren't visible in the figure because they are being overlapped by the ones visible in the figure. However the scan points of those scans do contribute to the color balancing. The construction space (see Section 4.1.2) is constructed in such a way that by chance in this point cloud the wall and the roof of Figure 5.14 are intersected by a common boundary of two cells. This phenomenon is shown in Figure 4.8.

Figure 5.14 A Point Cloud with several scans

When color balancing is applied, we get the result as in Figure 5.15. One of the scans which had an orange cast has caused some stains on the roof and the wall. The remaining part of the figure shows very good color balancing. On the floor the effect of orange scan points has been removed. The wall and the roof too have the same single color. But the stains present an unwanted artifact.

Figure 5.15 Color Balanced Image with Stains

The technique as described at the end of Section 4.3.4 was applied so that the values of means and standard deviation for any cell are a result of the cells lying on either side of the cell-boundary that intersects the walls and roof. As a result we get the point cloud as in Figure 5.16. The stains have vanished.

Figure 5.16 Color Balanced Point Cloud with no Stains

There is one scenario when both sides of the walls, roof or floor lie in the same construction cell. Both the sides of a wall might not necessarily be of the same color. Because color balancing occurs cell by cell the scan points of both the surfaces are combined to calculate one single mean and standard deviation. These values of mean and standard deviation may be inclined towards one of the surfaces.

Figure 5.17 shows a wall after color balancing and interpolation. The rather white wall shows quite substantial shades of brown. This is because of the presence of an open brown door on the other side of the wall. The application of interpolation with color balancing further spread this on the wall's surface.

Figure 5.17 Color Balancing Artifact

6. Summary and Future Work

During color transfer algorithm, we calculate the values of means and standard deviations of targets as a combined weighted effect of all the scans present in the point cloud. This combined effect is expected to represent the actual scene as accurately as possible. However this is not always true. All the individual scans of the point clouds might be deviated from the actual scene in terms of chromaticity which means that the combined target itself would be deviated from the actual scene. However if white balancing or gray balancing is also incorporated as a preceding step before color transfer algorithm, the results can be highly improved. White balancing tries to reverse the color shifts that result due to different illuminants and ambient conditions [**16**].

As a result the true color of the scene is achieved by reversing the color of all the pixels/scan points by the same amount with which the known white/gray points were affected. Thus for a more robust color transfer, all the individual scans can be passed through a white/gray balancing filter before the color transfer which will further refine the results. Known gray or white points can be made a part of every scan that a scanner takes. A possible way is to install a gray card on the scanner itself which appears in the view of the scans. In such a case, however, the light received by reflection from the gray cards may not represent the ambient light in the overall scene around the scanner.

Secondly, there is one inherent problem with the color transfer algorithm in conjunction with the construction space 3D view. As every world space is converted into construction space with a 3D cell as the smallest unit, a problem arises when both the sides of a wall, roof or floor lie in the same cell (see Figure 5.17) and the two surfaces have different colors. This is because the color transfer algorithm works cell by cell using the statistics of the whole cell. In the real world, both the sides of a wall might comprise of different color, however the color transfer algorithm matches all the points to a common target. The effect can be reduced if interpolation weights are also a function of distance along with function of the number of scan points. This would reduce the spread. This artifact can be avoided possibly by identifying the wall and roof like structures and make a partition between the two sides.

7. Bibliography

[1] Microsoft. XBox. [Online]. http://www.xbox.com/en-US/kinect

[2] R. B. R. a. S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, 2011.

[3] FARO. http://www.faro.com/products/3d-surveying/laserscanner-faro-focus-3d/overview#main.

[4] OpenSource. Point Cloud Library. [Online]. http://pointclouds.org/about/

[5] I. J. Cox, P. N. U. NEC Res. Inst., S. Roy, and S. L. Hingorani, "Dynamic histogram warping of image pairs for constant image brightness," in *Image Processing, 1995. Proceedings., International Conference on (Volume:2)*, Washington, DC, 1995, pp. 366-369vol2.

[6] D. A. Kerr. (2003) Chromaticity and Chrominance in Color Definition. [Online]. http://www.ma.utexas.edu/users/davis/reu/ch1/signals/chroma.pdf

[7] L. T. TROLAND, "REPORT OF COMMITTEE ON COLORIMETRY FOR 1920–21," JOSA, Vol. 6, Issue 6, pp. 527-591 (1922), 1922.

[8] D. B. JUDD, "Hue Saturation and Lightness of Surface Colors with Chromatic Illumination," *Journal of the Optic Society*, vol. 30, no. 1, pp. 2-32, 1940.

[9] R. S. Francesca Gasparini, "Color balancing of digital photos using simple image statistics," *Pattern Recognition*, vol. 37, no. 6, p. 1201–1217, 2004.

[10] M. H. Brill, "The relation between the color of the illuminant and the color of the illuminated object," *Color Research & Application*, vol. 20, pp. 70-76, 1995.

[11] J. A. S. Viggiano, "Comparison of the accuracy of different white-balancing options as quantified by their color constancy," *Proc. SPIE 5301, Sensors and Camera Systems for Scientific, Industrial, and Digital Photography Applications*, 2004.

[12] J. Sachs. Color Balancing Techniques. [Online]. http://www.dl-c.com/color%20balance.pdf

[13] T. P. Erik Reinhard, "Color Spaces for Colour Transfer," in *Third International Workshop,*

CCIW, Milan, Italy, pp. 1-15.

[14] L. M. Xuezhong Xiao, "Gradient-Preserving Color Transfer," *Computer Graphics Forum*, vol. Volume 28, no. 7, p. 1879–1886, Oct. 2009.

[15] E. Reinhard, S. L. C. U. U. Utah Univ., M. Adhikhmin, B. Gooch, and P. Shirley, "Color transfer between images," in *Computer Graphics and Applications, IEEE (Volume:21 , Issue: 5)*, 2002, pp. 34-41.

[16] Y.-C. Liu, H. T. Ind. Technol. Res. Inst., W.-H. Chan, and Y.-Q. Chen, "Automatic white balance for digital still camera," in *Consumer Electronics, IEEE Transactions*, 2002, pp. 460-466.

[17] D. Rowse. Digital Photography School . [Online]. http://digital-photography-school.com/introduction-to-white-balance

[18] M. Innocent, "White balance correction using illuminant estimation," USA Cypress Semiconductor Corporation US20080143844 A1, Jun. 19, 2008.

[19] M. T. T. Nguyen and S. o. M. &. M. Eng, "Estimating image illuminant color based on gray world assumption," in *Image and Signal Processing (CISP), 2011 4th International Congress on (Volume:2)*, Shanghai, 2011.

[20] F. Xiao, J. E. Farrell, J. M. DiCarlo, and B. A. Wandell, "Preferred color spaces for white balancing," in *SPIE 5017, Sensors and Camera Systems for Scientific, Industrial, and Digital Photography Applications IV*, 2003.

[21] A. N. László Neumann, "Color style transfer techniques using hue, lightness and saturation histogram matching," in *Computational Aesthetics'05 Proceedings of the First Eurographics conference on Computational Aesthetics in Graphics, Visualization and Imaging*, Switzerland, 2005, pp. 111-112.

[22] M. J. SUN P., "The influence of image histograms on cross-media colour image reproduction.," in *PICS*, 2001, pp. 363-367.

[23] M. Grundland and N. A. Dodgson, "Color histogram specification by histogram warping," in *PIE 5667, Color Imaging X: Processing, Hardcopy, and Applications*, San Jose, 2005.

[24] L. M. Xuezhong Xiao, "Color transfer in correlated color space," in *Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications*, New York, 2006, pp. 305-309.

[25] A. R. Adrian Ford. (1998) Color Space Conversions. [Online]. http://www.poynton.com/PDFs/coloureq.pdf

[26] Intel. Color Models and Color Space Conversions. [Online]. http://software.intel.com/sites/products/documentation/hpc/ipp/ippi/ippi_ch6/ch6_color_models.html#fig6-4

[27] H. K. F. C. M. Emre Celebi, "Fast Color Space Transformations Using Minimax Approximations," *IET Image Processing 4*, pp. 70-80, 2010.

[28] T. W. C. ,. C.-C. C. Daniel L. Ruderman, "Statistics of Cone Responses to Natural Images: Implications for Visual Coding," *Journal of the Optical Society of America A*, 1998.

[29] V.-E. Neagoe, "Decorrelation of the Color Space, Feature/Decision Fusion, and Concurrent Neural Classifiers for Color Pattern Recognition," in *Proceedings of the 2008 International Conference on Image Processing, Computer Vision, & Pattern Recognition, IPCV 2008*, vol. 2, Las Vegas, 2008.

[30] EASYRGB. [Online]. http://www.easyrgb.com/index.php?X=MATH&H=07#text7

[31] A. N. V. P.E. Trahanias, "Venetsanopoulos, Color image enhancement through 3-d histogram equalization," in *11th IAPR Conf. Pattern Recogn*, 1992.

[32] J. Morovic and M. R. Luo, "The Fundamentals of Gamut Mapping: A Survey," *Journal of Imaging Science and Technology*, vol. 8, no. 3, pp. 283-290, May 2001.

[33] scratchapixel. [Online]. http://www.scratchapixel.com/lessons/3d-advanced-lessons/interpolation/bilinear-interpolation/

[34] P. Bourke. (1997) http://paulbourke.net/. [Online]. http://paulbourke.net/miscellaneous/interpolation/

[35] X. .-D. Yang, R. U. ,. S. ,. C. Dept. of Comput. Sci., Q. Xiao, and H. Raafat, "Direct mapping between histograms: an improved interactive image enhancement method," in *Systems, Man, and Cybernetics, 1991. 'Decision Aiding for Complex Systems, Conference Proceedings., 1991 IEEE International Conference*, Charlottesville, VA, 1991, pp. 243-247vol1.

Declaration

I hereby declare that the work presented in this thesis is entirely my own.
I did not use any other sources and references that the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations.
Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before.
The electronic copy is consistent with all submitted copies.

(Umair Nasir)