Lab Report 1

Longxiang Wang, Jordan Earle, Ruihan Sun, Ard Snijders

Introduction

In this report, some basic methods of analyzing images were implemented to give practical experience and to see how the methods worked. In the first part photometric stereo was implemented on different images to deconstruct and reconstruct images both in color and in grayscale. Next colour spaces were studied, converting a given RBG image into a specific color space. After this, using a set of images that has been decomposed using intrinsic image decomposition, material recoloring was performed on a synthetic image. Finally, color constancy was examined by implementing the Grey-World algorithm.

Photometric Stereo

1. Estimating Albedo and Surface Normal

In this section, an albedo and surface normal map was created at first for the SphereGrey5 set of images and then for the SphereGray25 set of images. The intensity $i(x, y)$ can be represented as a function of the input radiance $g(x, y)$, and $V(x, y)$ which is the matrix which contains the properties of the illumination and of the camera where

$$\boldsymbol{i}(x, y) = \mathcal{V}\mathbf{g}(x, y) \tag{1}$$

The problem with this approach is that there can be substantial regions of the surface covered in shadow for one or more of the light sources, obscuring what the surface might be in that area with the overshadowed area. A so-called 'shadow trick' is introduced to overcome this issue:

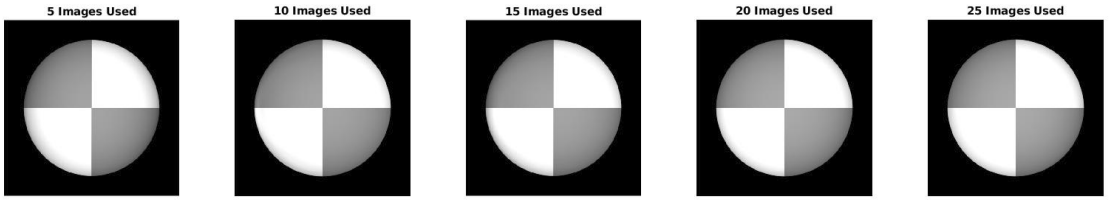
$$\mathcal{I}\boldsymbol{i} = \mathcal{I}\mathcal{V}\boldsymbol{g}(x, y) \tag{2}$$

where $\mathcal{I}$ is defines as diagonal matrix consisting of the values of $\boldsymbol{i}$ upon the diagonal and zero elsewhere. The matrix $\mathcal{I}$ has the effect of zeroing the contributions from shadowed regions, because the relevant elements of the matrix are zero at points that are in shadow. Solving the above equations for the input radiance $g(x, y)$, the second norm of the input radiance $|\mathbf{g}(x, y)|$ gives the albedo.

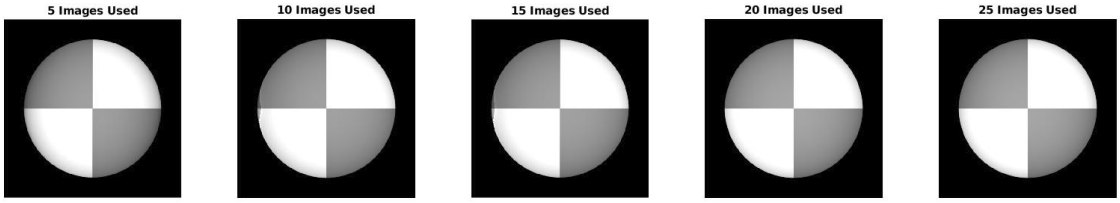
Initially, the SphereGray5 image set was analysed using the 'shadow trick'. The result of which can be seen in Figure 2d in the leftmost image. In an ideal albedo image you would expect to see uniform coloring to display what light is being reflected from the surface. From Figure 2d (even at higher numbers of images) it can be seen that there is some shading around the edges, which would indicate that there are not enough images to clearly define what the primary colours of the surface are. More images will be needed to improve the quality of the albedo calculation.

While calculating the albedo, it is important to consider the theoretical minimum number of images needed to find the albedo and the surface normal. Note that the input radiance is an $(n \times n \times 3)$ matrix, and is required to calculate both the surface normal (with (x, y, z) components) and the albedo. In order for the linear set of equations to be completely defined, a minimum of 3 images is required. These images also need to be complete, in order to ensure that for each pixel the 3 dimensions can be found.

In order to test how the number of images affects the albedo and the surface normal both with and without the 'shadow trick', the SphereGray25 image set was used with varying numbers of images. First, the numbers of images were selected according to order of appearance in the folder, the results of which can be seen in Figure 1.



(a) Albedo map created without the 'shadow trick'



(b) Albedo map created with the 'shadow trick'

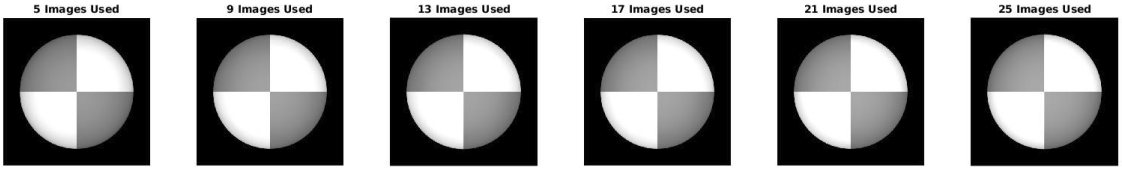
Figure 1: Albedo map for images from SphereGray25 selected in the order of appearance from the SphereGray25 image set.

From the figures it can be seen that when using the 'shadow trick' there is an artifact in the left which does not appear in the non 'shadow trick' images. This appears to be because the shadow trick requires the shadows in the images to placed symmetrical around the object to give an accurate representation. In the images used for this set, they had a shadow in the left side for most of the images, which appears to have caused this. The map created without the 'shadow trick' seems to be less sensitive to the placement of the shadows.

Once this was discovered, a new set was curated for each of the sizes to ensure that the shadows were evenly represented in the sets. The results can be seen in Figure 2. From the figures showing the albedo it can be seen that in both cases, as the number of images increases, the smoothness and the uniformity of the image increases. However, for a low number of images such as 5, the shadow trick is needed and this does cause the image to be smoother with the shadows at the edge to be smaller. The real difference between the two can be observed in the normal map. In the figures without the shadow trick it can be seen that the normals are not as smooth and have large jumps in values, which is not representative of the shape. As the number of images is increased the normals become smoother, as can be seen in the 21 and 25 image sets, as there is not much difference between the 2 methods smoothness. In order to closely examine what happens when less images are used, the normals were broken into their primary components. This can be observed in Figure 3 for both the shadow trick and without.

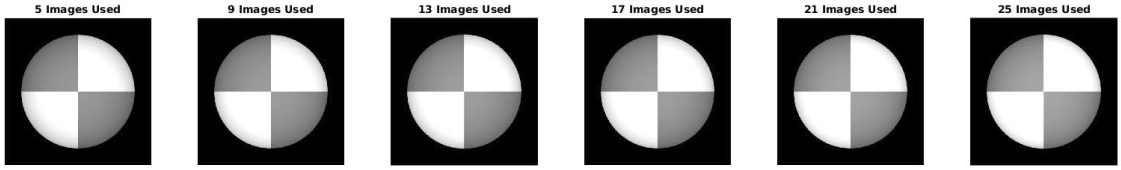
2. Test of Integrability

After the normals and albedo is found, a test of integrability has to be performed in order to construct a surface map. As can be seen in Figure 4, the error is largest around the edge, which is likely due to the darker edges blending in with the background at the end of the object. Since the colour is similar to both the black background and the darker pixels around the edge of the object, the error is higher as it's hard to determine if the location is in the background or the edge of the object in shadow. When increasing the number of images, the edge becomes clearer, lowering the overall error in locations around the object, leaving it almost zero in some. It would follow that increasing the number of images would make the edge



(a) Albedo map created without the 'shadow trick'.

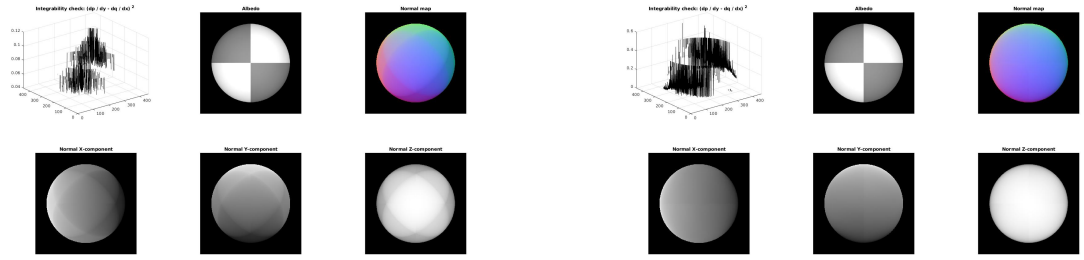
(b) Normal map created without the 'shadow trick'.



(c) Albedo map created with the 'shadow trick'

(d) Normal map created with the 'shadow trick'.

Figure 2: Albedo and normal maps for images from a curated SphereGray25 with even shadow representation in each subset



(a) without shadow trick.

(b) with shadow trick.

Figure 3: SE, albedo, sufrace normal, surface normal X, surface normal Y, surface normal Z (from left to right, top to bottom) generated using 5 images from the SphereGray5 with and without shadow trick.

even clearer, lowering the error further.

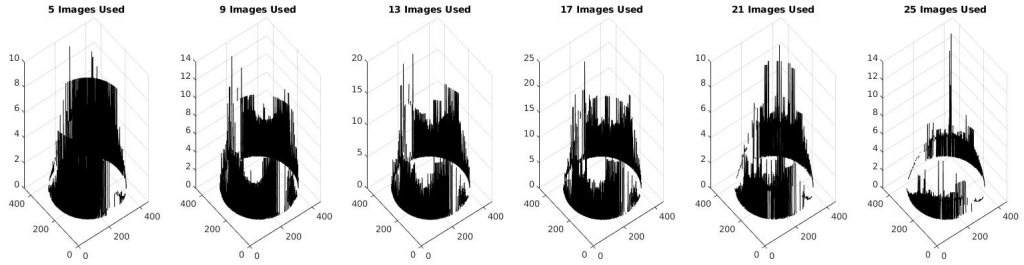
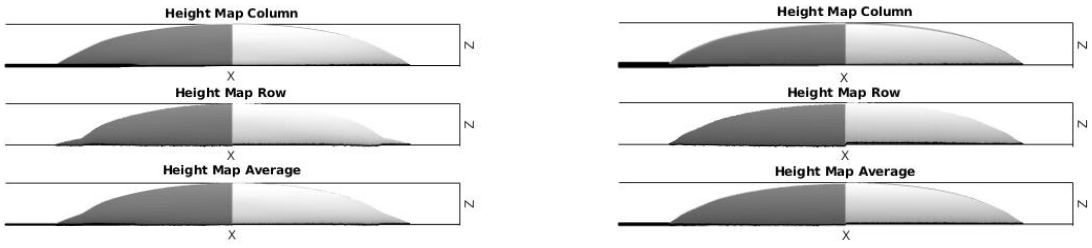


Figure 4: SE created using the shadow trick with the curated GraySphere25 image set with even shadow representation.

3. Shape by Integration

Once the integrability check was performed, a surface height map of the image was constructed. This was done through use of the shadow trick over various numbers of images. In Figure 5 it can be seen that using the row/column method, error will compound along the direction the integration is occurring, causing a rise in the profile as the path continues. Using the average method, the error along the edges are reduced and the overall shape is more well defined, as can be clearly seen in Figure 5a with 5 images. As the number of images increases, the distortion from the path method is decreased, and the overall shape is more well defined as can be seen comparing the 5 image and 25 image height maps seen in Figure 5.



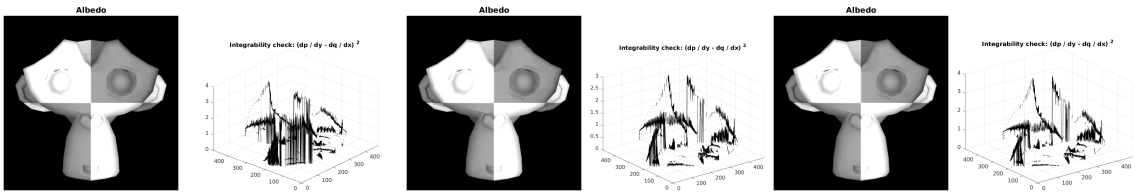
(a) 5 images.

(b) 25 images.

Figure 5: Surface height map from same view point for different number of images with different path methods.

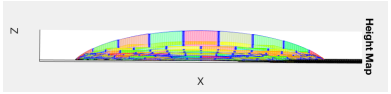
4. Experiments with Different Objects

When applying the photometric model to the MonkeyGray image set it was seen that at small numbers of images, the albedo and surface noramls could not be computed well, since the surface is so complicated and there are multiple areas in shadow for a number of the images. Looking at the plots in Figure 6, it can be seen that for larger numbers of images, the albedo does improve slightly, but there is still significant shading in the albedo plots, due to the complex shapes and recessed areas seen around the eyes, mouth and ears. This can also be confirmed in the associated SE plots. In order to solve these errors, more images that have the problem areas lit from multiple angles strongly could be use in order to provide a clearer image. Also removing some of the more shadowed images which obscure those areas from the existing set could allow for a more complete albedo map.

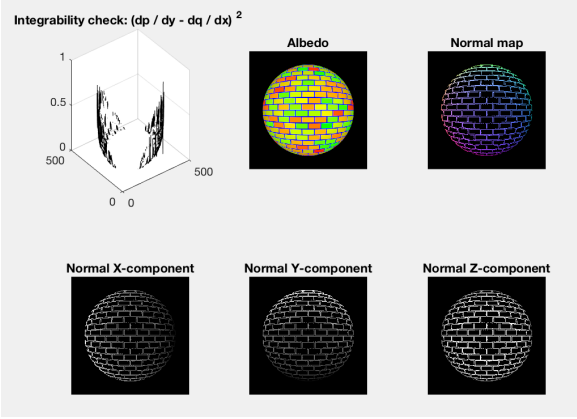


(a) Albedo 41 images (b) SE 41 images. (c) Albedo 81 images (d) SE 81 images (e) Albedo 121 images (f) SE 121 images

Figure 6: Albedo and SE maps created using shadow trick on MonkeyGray dataset with various number of images.



(a) Height map for color sphere



(b) Normal and albedo for color sphere

Figure 7: Results for color sphere

For color images, first the albedo is computed for the separate color channels, which are then concatenated together to form a colored albedo image. Similarly, to determine the normals, normals are computed for each channel, and then averaged. For color spheres, the result seems good. However, for color monkeys, the normal map and height map are black due to zero pixels. The reason for this is that for zero pixels, undefined divide by zero operations arise when constructing height map and normal map. One possible solution is applying some average filters to the image to average out zero pixels beforehand.

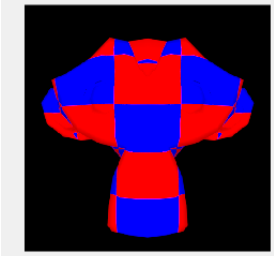
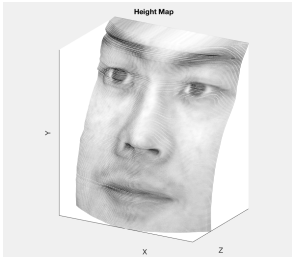


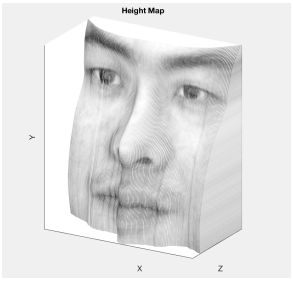
Figure 8: Albedo constructed for color monkey

The column constructed face is more flat compared to the one with row constructed, while the row constructed face collapses a lot, especially the mouth. Averaged face spreads out the errors from column and row faces, making it better than row constructed one.

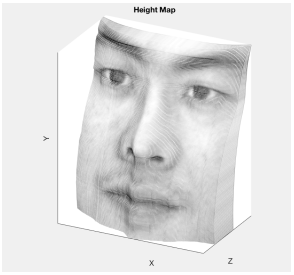
These faces violate shape-from-shading since the skin is not Lambertian reflective and is not a smooth surface. Furthermore, the light in those images is not a single distant source, as there is a lot of shadow. Given the effect of the light source, some ill-illuminated images were removed. Unfortunately, due to the complicated structure of faces, it was difficult to achieve an improvement in the results.



(a) Column constructed faces



(b) Row constructed faces



(c) Average constructed faces

Color Spaces

1. RGB Color Model

The RGB color space is the dominant color space for digital cameras and photography, since it bears similarity to the way the human brain perceives color and images.

Human eyes have three cone cells that can sense three different light waves based on their wavelengths, long (around 560nm), medium (around 530nm) and short (around 420nm), corresponding to red, green and blue. Filters are placed over camera pixel sensors, allowing certain wavelength passes through. A very well-known RGB filter is Bayer filter.

2. Color Space Conversion

The detailed implementation, which is quite straightforward, is given in the Matlab files.

3. Color Space Properties

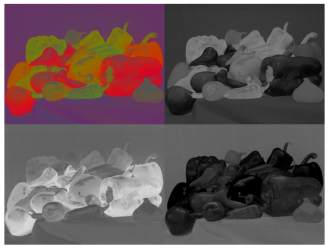
3.1 Opponent Color Space

The opponent color space introduced by Bratkov et. al.[3] is based on opponent process theory proposed by Ewald Hering. The idea is that instead of three hue primaries (red, green and blue), there are actually four of them, including yellow. In this space, there are two chromatic axes: one from red to green and another one from yellow to blue, and one luma channel. It is easy to transform from RGB space and designed very suitable for computational purposes. Moreover, it achieves better results on many applications than RGB space, like color adjustment, cool to warm shading, color transfer and so on.

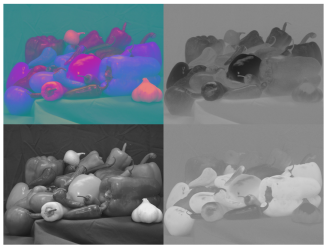
3.2 Normalized RGB Color Space

An important effect to be considered concerns the distortions and shadows caused by light. In order to get rid of those effects, it is neccesary to normalize the RGB values. Without the effect of lights, images shown in a normalized RGB space have clear boundaries among different objects and thus easier to do object/edge detection.

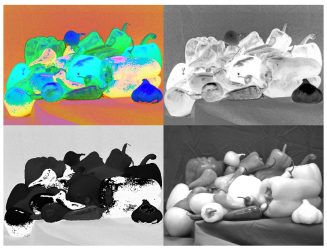
(a) Opponent color space



(b) Normal RGB color space



(c) Ycbcr color space



(d) HSV color space

(e) Gray color space

Figure 10: Different color spaces

3.3 HSV Color Space

HSV color space also has three components, hue, saturation and value. It is a cylindrical system, where hue (from 0 to 360) indicates different colors, saturation (from 0 to 1) indicates the amount of gray in a color and value (from 0 to 100) indicates the intensity of color. Compared to RGB and other color spaces, it is more close to how human perceive colors and is more expressive since it also contains information for shadow and brightness. Thus HSV color space is commonly used in many image editing softwares to show variety color effects.

3.4 YCbCr Color Space

YCbCr color space is defined by three components, where Y represents the intensity of color, Cb represents the blue difference (blue component relative to green component) and Cr represents the red difference. This color space works well when compressing images - since human eyes are more sensitive to intensity, only the Y part needs to be accurate and Cb, Cr can be less accurate. This lowers the amount of required storage while minimizing the loss of information.

3.5 GrayScale

Unlike previous color spaces, GrayScale color space only has one channel to represent the brightness of pixels. Whiter means a higher intensity and darker means a lower intensity. As a result, computer just needs one byte to store each pixel other than three. If someone just cares about the brightness of different parts of an image, GrayScale color space is a good choice.

4. More on Color Spaces

Another color space found in the literature [5] is the CMYK color space, where C = Cyan, M = Magenta, Y = Yellow and K = Key Black. It is a subtractive color space and can be transformed back to RGB color

space, as follows:
C + Y = G , C + M = B , M + Y = R , C + M + Y = Black

Furthermore, by modifying ratios of the four colors, CMYK color space can represent different color shades and values. Today, CMYK color space is mostly used for printers.

Intrinsic Image Decomposition

1. Other Intrinsic Components

From a physics point of view, every single image can be decomposed into a reflectance component (albedo) and an illumination component (shading) such that $I(x) = (R(x)S(x)$ [2]. However, some studies suggest elaborating the shading component with an additional specular component $C(x)$.[1]

Alternatively, an image can also be decomposed into separate wavelet components in frequency domain using the Fourier Transform.

2. Synthetic Images

Image decomposition remains largely unsolved due to the ill-posed nature of the problem. Researchers have only been able to produce approximate solutions rather than absolute, ground-truth intrinsics. Resultantly, creating these ground-truth images from real-world environments is costly and non-synthetic data-sets are of limited supply. Using synthetically generated data circumvents this issue as it allows for simulating different lighting environments whilst retaining the ability to easily generate reliable image intrinsics for model calibration/validation.

3. Image Formation

Figure 11a displays how the image can be reconstructed through the use of its albedo and shading intrinsics. Reconstruction is achieved by simply multiplying the reflectance and shading matrices, according to the formula described in 1.

4. Recoloring

It is assumed that the albedo has a single, uniform color. The true color in RGB space can be obtained by finding the value of the non-zero element for each color space. This yields a color of [184, 141, 108].

The ball can be recolored with a different hue through the use of a simple algorithm which iterates over all the pixels in the image. If a pixel has color information, it can be recolored by multiplying it with a vector that reflects the desired color combination - for instance, for pure green, the vector [0, 255, 0] is used.

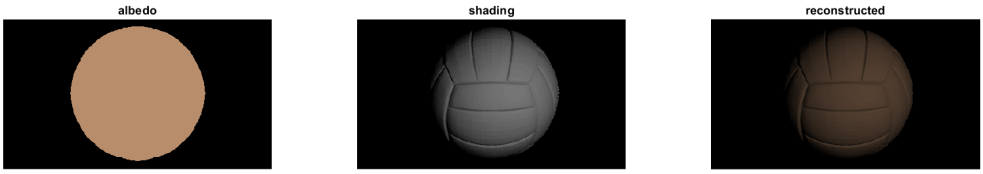
The reconstructed ball does not display any pure green because of the shading component. The severity of this shading depends on two things: the geometry of the object and the overall intensity of the illumination of the object.

Color Constancy

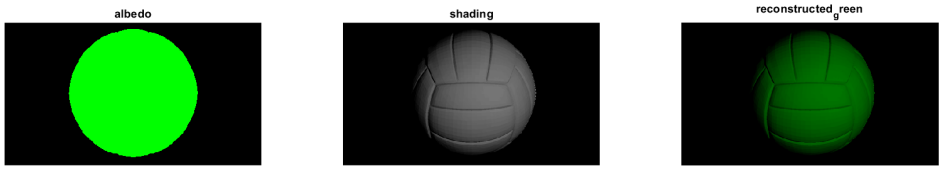
1. Grey World Algorithm

The Grey-World algorithm (GWA) assumes that the average color of a scene is neutral grey. This algorithm consists of 3 steps:

1. Obtain grey value in one of the two ways:
   (a) fixed value of $\overline{Grey} = [128, 128, 128]$ in 8-bit RGB representation.
   (b) mean of the sum of mean of RGB channels, $\overline{Grey} = \frac{\overline{R}+\overline{G}+\overline{B}}{3}$.



(a) Image intrinsics



(b) Recoloring the ball with pure green

Figure 11: Image reconstruction

2. Calculate the scaling coefficient for each channel $k_r = \frac{\overline{R}}{\overline{Grey}}, k_g = \frac{\overline{G}}{\overline{Grey}}, k_b = \frac{\overline{B}}{\overline{Grey}}$.

3. For each pixel, the RGB components are adjusted. $C(R') = k_r * C(R), C(G') = k_g * C(G), C(B') = k_b * C(B)$

(a) Working example

(b) Failure example

Figure 12: Original image vs converted image

As can be seen in Fig. 12a, the reddish color cast is removed by AWB.

2. Failures

GWA is fast and efficient due to its simplicity. However, this approach will fail when the dominant illumination is not neutral grey. For example, when the scene has large patch of mono color, such as a blue

sky scene. The GWA is applied on a blue sky scene, and the result is shown in Fig. 12b. As can be observed, GWA casts a unrealistic color to the original blue sky.

3. Other Color Constancy Algorithm(s)

The perfect reflector algorithm [4] is based on the assumption that the brightest pixel in an image corresponds to the white point. Specular or glossy surfaces reflect the actual color of the light source because their reflectance functions are constant over a wide range of wavelengths.

The perfect reflector algorithm locates the brightest pixel and assigns it as the reference white point.

Conclusion

In this assignment, basic photometric stereo techniques were used to construct shapes from different illumination conditions. It was found that in order to get a good result, the quality and quantity of data set is important (with good illumination and Lambertian). If the set of images contains a large number of images with difficult features well illuminated from different lighting positions, then the output should be more accurate and smooth. If the images have shadows obscuring the complex features, the regions will be more difficult to map. For coloured images, the normals should be averaged over the different channels to have a smooth height map.

In part two, various color spaces were discussed. The different color spaces are suitable for variety of purposes and one of the main reasons that the RGB color space is the most commonly used one due to its close resemblance to human vision system.

In part three, intrinsic image decomposition was explored. As was shown, it is possible to reconstruct the image of a ball by combining its albedo and shading intrinsics. It should be noted that this reconstruction concerned a synthetic image: When dealing with the many complexities of real-world images, it can be difficult to successfully extract the albedo and shading intrinsics due to the ill-posed nature of the problem. It is also possible to recolor the image of the ball, given that the shading intrinsic is available. In this particular instance, the shading slightly altered the color distribution of the recolored ball, insofar that its color green was less intense compared to the intended pure green.

Finally, some white balance algorithms were explored. It was found that the Grey-World algorithm only works when the assumption that the light source is close to grey is met - Otherwise, the algorithm will fail. The perfect reflector algorithm was briefly discussed to cast light on other alternative color constancy algorithms.

References

[1] Shi et al. “Learning Non-Lambertian Object Intrinsics Across ShapeNet Categories”. In: (2017). (Accessed on 09/18/2019).

[2] H.G. Barrow and J.M. Tenenbaum. “Recovering intrinsic scene characteristics from images”. In: *Computer Vision Systems.* (1978).

[3] Margarita Bratkova, Solomon Boulos, and Peter Shirley. “oRGB: A Practical Opponent Color Space for Computer Graphics”. In: *IEEE computer graphics and applications* 29 (Mar. 2009), pp. 42–55. DOI: 10.1109/MCG.2009.13.

[4] Ching-Chih Weng, H. Chen, and Chiou-Shann Fuh. “A novel automatic white balance method for digital still cameras”. In: *2005 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2005, 3801–3804 Vol. 4. DOI: 10.1109/ISCAS.2005.1465458.

[5] PM Nishad. “Various colour spaces and colour space conversion”. In: *Journal of Global Research in Computer Science* 4.1 (2013), pp. 44–48.