UNIVERSITEIT VAN AMSTERDAM

COMPUTER VISION

ASSIGNMENT 1

PHOTOMETRIC STEREO & COLOUR

Ahmet Taskale
Student Number:12344087
ahmet.taskale@student.uva.nl

Andreea Teodora Patra
Student Number:13365169
andreea.patra@student.uva.nl

Pan Pan Wu
Student Number: 12629642
pan-pan.wu@student.uva.nl

September 14, 2020

ABSTRACT

.

1 Introduction

To put the photometric stereo method down in black and white, objects can either be bright or dark. The reason for an object to be bright or dark depends on the albedo value and the amount of diffused light a source is capturing. Photometric stereo is a method for recovering the surface of an object from the collected image data. The principle of this method is based on collecting information from image intensity values from multiple images, illuminated by different light sources in a fixed view. By doing so, the height of the surface is captured at every point corresponding to the specific pixel. In this assignment, the photometric stereo method will be applied to different images. first of all, the albedo and the surface normals will be estimated. Secondly, the computed data needs to be tested for integrability. Finally, to reconstruct the surface height, the values of the normals need to be integrated.

2 Photometric Stereo

2.1 Estimating Albedo and Surface Normal

2.1.1 Question 1

1. A patch of surface is reconstructed from a series of images of the same surface taken under the different illuminations. We use a local shading model and we assume there is no ambient illumination. Therefore, using the radiosity for each (x,y) point in the image, we can derive $I(x, y) = g(x, y) \cdot V_1$, where g describes the surface and V_1 is a property of the illumination and of the camera. In order to construct the albedo and the normal, we solve the following system

$$\mathbf{i}(x, y) = V\mathbf{g}(x, y)$$

Using least squares would give as albedo as $\mid \mathbf{g} \mid$ and the normal as $\frac{\mathbf{g}}{|\mathbf{g}|}$. The vector $\mathbf{i}$ represent the measurements for each image point stacked into a vector and the matrix V is representative for the illumination sources.

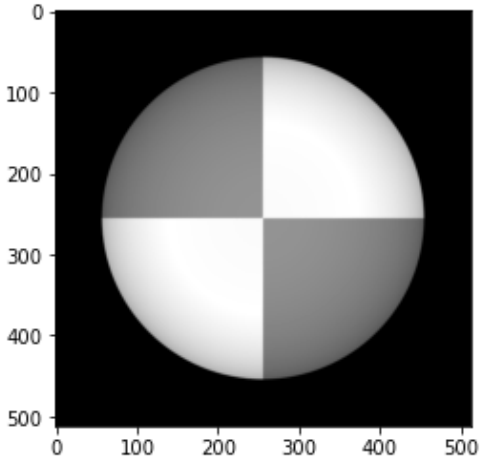

Figure 1. The albedo for 5 light sources

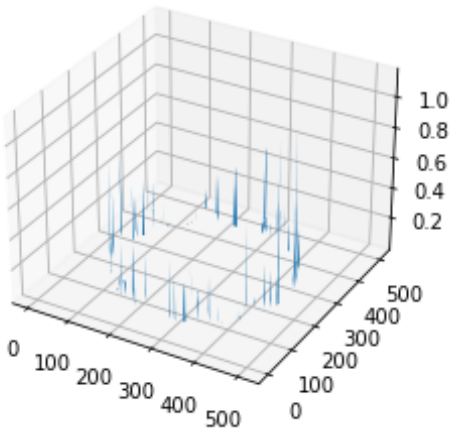

Figure 2. The error for 5 light sources - 2335 outliers

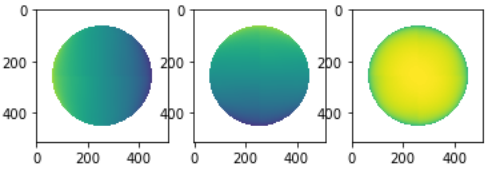

Figure 3. The normal channels for 5 light sources

The expectation is the illuminated sphere without any shadow, since the albedo recovers the image of the original object. The expectation is confirmed with the computed albedo, showing no signs of shadow.

2. In principal, there should be used at least 3 images so that the least squares solution is appropriate. When comparing SphereGray5 with SphereGray25, the albedo image computed from SphereGray25 (5v25) is slightly darker, meaning the albedo has a lower value. An explanation for this might be the angle of the light sources compared to the surface normal. When this angle value is too high, the intensity of the reflection is lower since reflected energy is proportional to cosine of angle between the surface normal (N) and the light direction (L). With more light sources, it is reasonable to assume that there are more light sources with an unfavourable angle with respect to the surface normal, resulting in a lower albedo value.

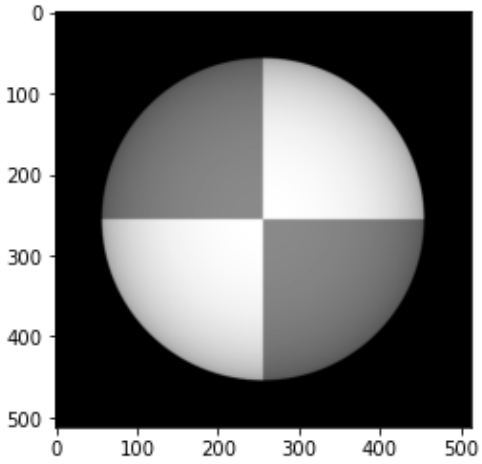

Figure 4. The albedo for 25 light sources

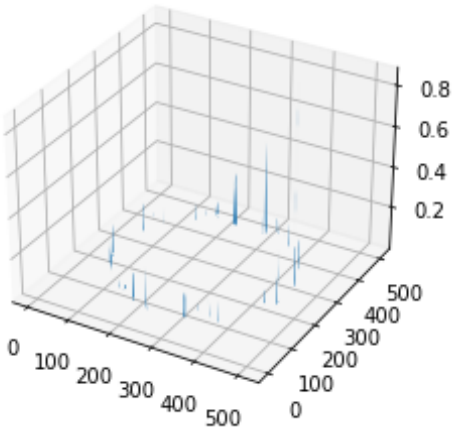

Figure 5. The error for 25 light sources - 1832 outliers

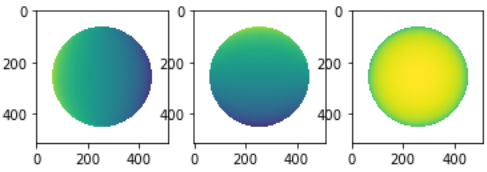

Figure 6. The normal channels for 25 light sources

3. Shadows occurs when a patch cannot see one or more sources of light. When not accounted, shadows can lead to distorted reconstructions. Assuming that there is no ambient illumination, a diagonal matrix for each pixel from the image vector is formed. Since pixels in shadows have a value of 0, multiplying the diagonal matrices by each side of the equation zeroes out any equation from points in shadows:

$$\mathbf{i}(x, y) = V\mathbf{(x, y)}$$

By doing so, we're left with one linear system per pixel in the image and each linear system needs to be solved to recover g at that specific point. As for applying the shadowtrick with 5 images, looking at the images it gets clear that images computed with the shadowtrick result in less shadow (higher albedo), thus looking brighter than the 5 images computed without the shadowtrick. This effect is the same in the case of 25 images, however it seems that with more light sources the effect of the shadowtrick gets less efficient and therefore resulting in minor increase of albedo.

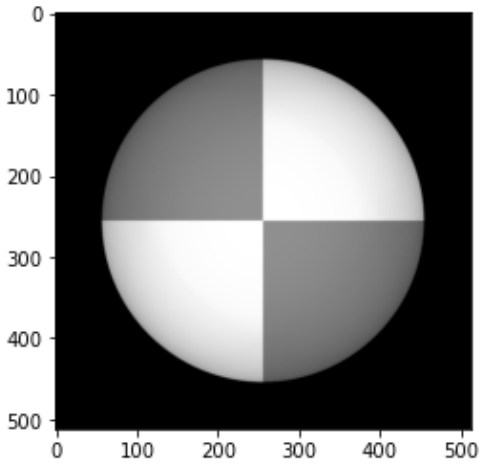

Figure 7. The albedo for 5 light sources with no shadow trick

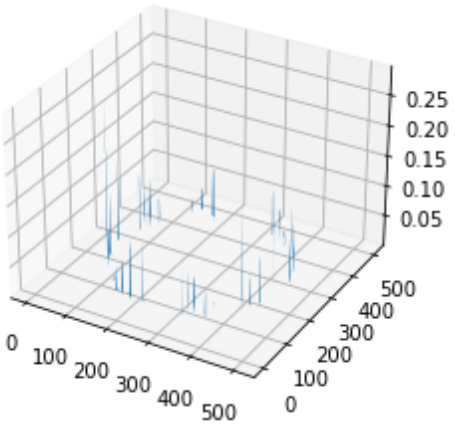

Figure 8. The error for 5 light sources - 1689 outliers with no shadow trick

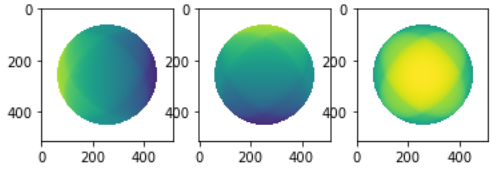

Figure 9. The normal channels for 5 light sources with no shadow trick

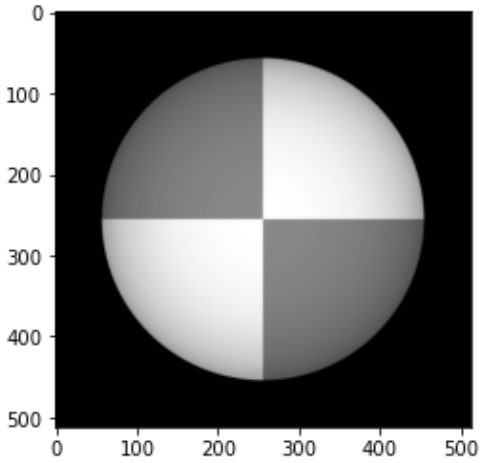

Figure 10. The albedo for 25 light sources with no shadow trick

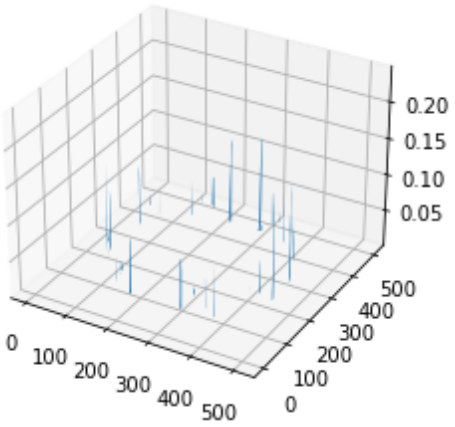

Figure 11. The error for 25 light sources - 1614 outliers with no shadow trick

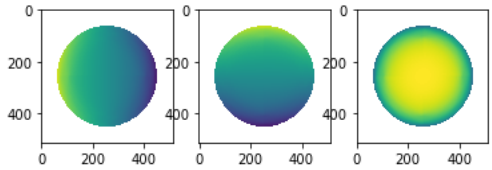

Figure 12. The normal channels for 25 light sources with no shadow trick

2.1.2 Question 2

2. The errors seem to arise around the borders of the object. An explanation might be that the difference in contrast between the object and it's background is too high, because the gradient is based on the difference between neighbouring pixels. What happens at this transition from background to object is that a high value of a derivative jumps to a lower value of a derivative. Naturally, a higher threshold results in less errors and a lower threshold results in more errors. As for the number of images; The test results in less number of errors with more images (25v5), returning 1831 outliers for 25 images and 2335 outliers for 5 images. It seems that providing the test with more images (more light sources), the method is less prone to errors. However, it is noteworthy to say that more lightsources produce a lower albedo value as mentioned in 1.1.2. It is important to find the right balance between these parameters. When constructing the surface, results show that the recontructing process with 25 images returns a smoother surface compared to 5 images regardless of the threshold value. As can be seen from the images, the reconstruction algorithm with 25 images returns the same shape with threshold values 0,05; 0,005 and 0,0005 and seems to have a smoother surface compared to the reconstruction algorithm with 5 images. As for the algorithm run with 5 images, the surface is less smoother. However, adjusting the threshold value (0,05; 0,005 and 0,0005) doesn't seem to affect the surface. Of course, as mentioned before, a higher threshold results in less errors and a lower threshold results in more errors.

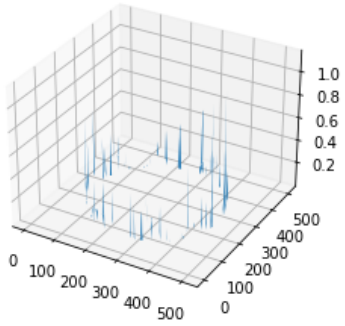

Figure 13. The error with a threshold of 0.005 - 2335 outliers - 5 images

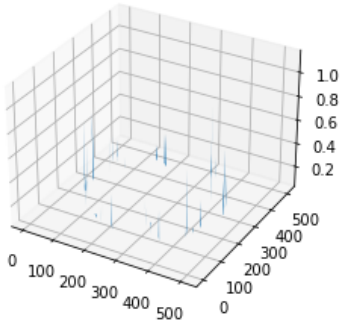

Figure 14. The error with a threshold of 0.05 - 1335 outliers - 5 images

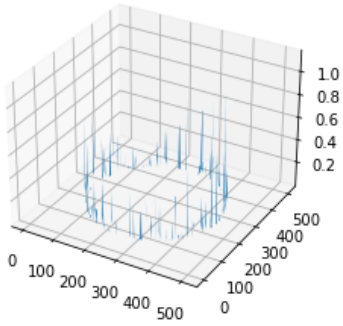

Figure 15. The error with a threshold of 0.0005 - 2900 outliers - 5 images

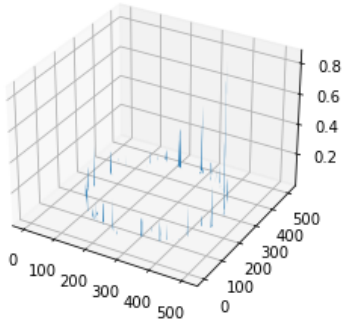

Figure 16. The error with a threshold of 0.0005 - 2312 outliers - 25 images

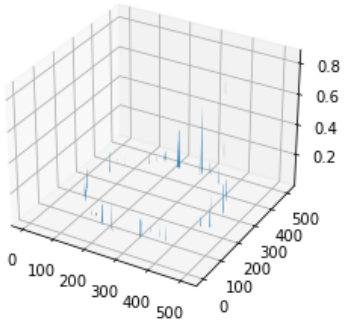

Figure 17. The error with a threshold of 0.005 - 1831 outliers - 25 image

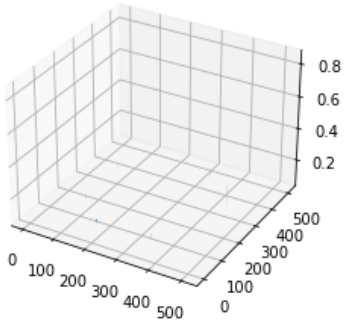

Figure 18. The error with a threshold of 0.05 - 878 outliers - 25 images

2.1.3 Shape by integration

1. As seen in the images in the Appendix, there is no particular difference between the methods that uses either column path or row path.

2.1.4 Experiments with different objects

4. The surface of the MonkeyGray model is more complex compared to the sphere. It has more depth at certain points, which results in a high value of angle between certain light sources and and surface normals. This is verified by the table below, which shows the number of images(number of light sources) and the corresponding outliers. As mentioned before, more light sources from different angles illuminating the model decreases the number of light sources with a high value angle of incoming light with respect to the surface normal. Having only a handful of light sources is limited in the favourable (lower) angles they can cover. Indeed, more light sources means more favourable angles to cover which result in a higher albedo value. (See appendix)

5. Looking at the images, the number of channels (BGR) for the input of images seem to be of paramount importance. For instance, the image of MonkeyColor with channel=0 (blue) show high albedo values for patches where the model originally is blue and dark patches for the red parts. This is verified by the image MonkeyColor with channel=2, showing high albedo values for the parts that are red in the original model. The problems encountered with the zero pixels can be overcome by adding a small value 1e-7. (See appendix)

3 Colour Spaces

In this section, we are going to investigate different color spaces and how we can transform our RGB images into them.

3.1 RGB Colour Model

The existence of 3 primaries (Red, Green and Blue) is a result of the tri-stimulus nature of the human systems since we have 3 different types of cones, each responding selectively to a different portion of the color spectrum. Therefore, just using additive colors, the color spectrum can be created.

Color camera integrates light according to the spectral response function of it red, green and blue sensors:

$$R = \int L(\lambda) S_R(\lambda) d\lambda$$
$$G = \int L(\lambda) S_G(\lambda) d\lambda$$
$$B = \int L(\lambda) S_B(\lambda) d\lambda$$

where $L(\lambda)$ is the incoming spectrum of light at a given pixel.
The majority of digital color cameras nowadays use the Bayer pattern to have valid RGB values for all the pixels. Green filters are placed over half of the sensors and red and blue filters over the remaining ones. The luminance signal is mostly determined by green values and the visual system is more sensitive to high frequency detail in luminance. That is why the green filters are twice as many. Another step that cameras take is performing color balancing in an attempt to move the white point to pure white.

3.2 Color Space Conversion

In this part, we are going to use as input an RGB image and convert it into 5 different color spaces. In this way, we can analyse their efficiency in particular use cases.

Starting from the right in Figure 60, you can see the conversion of the RGB image into HSV, Opponent, Normalised RGB and YCbCr color spaces. Their respective color channels decomposition can be observed in Figure 2, in the same order as before. The analysis of their performance and use cases can be found in part 3.3 Color Space Properties.

Figure 19. Initial RGB image used for conversions

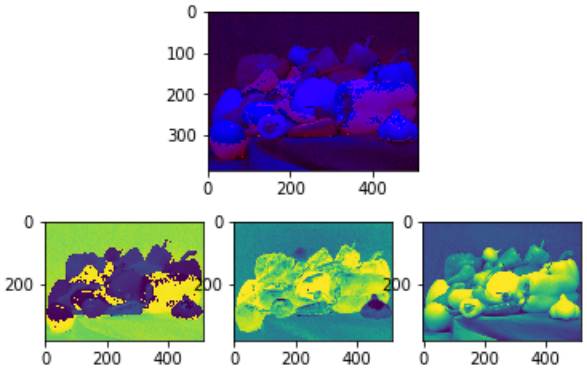

Figure 20. HSV color space

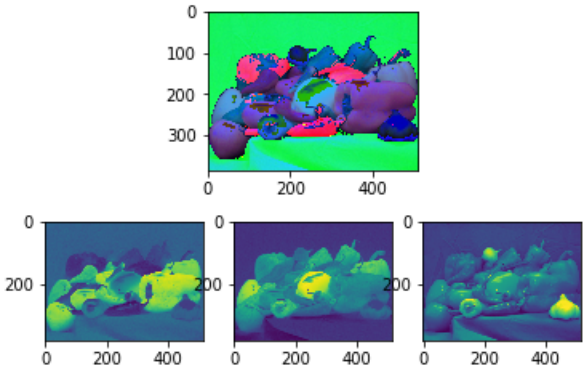

Figure 21. Opponent Color Space

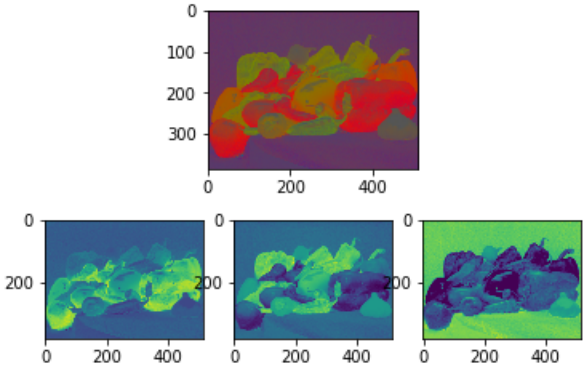

Figure 22. Normalised RGB color space

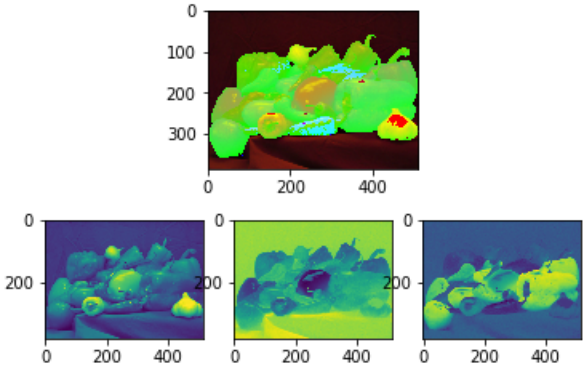

Figure 23. YCbCr color space

3.3 Color Space Properties

The HSV color is a projection of the RGB onto a non-linear chroma angle, a radial saturation percentage and a luminance value. It separates the intensity from the color information. The luminance value is defined as the mean or maximum color value, saturation is defined as scaled distance from the diagonal and hue is defined as the direction around a color wheel. HSV proves to be more useful when doing histogram equalisation as our goal is to make the intensities uniform. Looking at the saturation channel we can see how vivid each color and by looking at the third channel we can understand their brightness.

In the YCbCr color space, Y is the luma component and C_B and C_R are the blue-difference and red-difference chroma components. Luma is defined as the brightness, whereas chroma referes to the color information. It it particular useful in image compression. The Y component is more sensitive to the human eye than C_B or C_R and in image compression

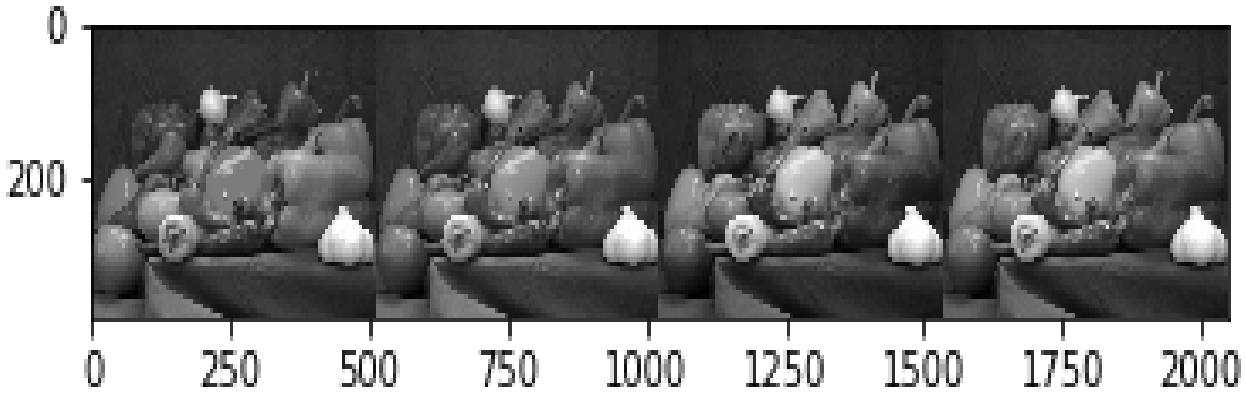

Figure 24. Grayscale conversion - from right to left Lightness, average, luminosity, OpenCV function

we want to compress the luminance signal with higher fidelity than the chrominance signal. Therefore, as a result, YCbCr is used more often. As seen in the images of the channels, the output of the luma is particular similar to the output we would get from a grayscale algorithm which agrees with the idea that the luma component is expressed with more precision.

Reference : `http://refbase.cvc.uab.es/files/AVL2011b.pdf`
The opponent color space has the first component as the luminance, the second as the red-green channel and the third as blue-yellow channel. It is based on the opponent color theory that was first introduced in the 19th century by Ewald Hering. He states that red, green, blue and yellow are fundamental hues in such a way that they cannot be described by mixtures of other hues. Yellow-blue and red-green information is represented by 2 parallel channel in the visual system. Rao Muhammad Anwer, David Vazquez and Antonio M. Lopes obtained better results using opponent color space when conduction human detection. When looking at the different channels of the image, we can see that the red-green channel appears more faded than the blue-yellow one.

The normalised color space is specifically useful when it is required to remove all the intensities whilst still preserving the color values. It is particularly used in surveillance cameras when it is important to remove shadows or lighting changes. If we look at the resulting image from the algorithm, we can see that the shadows and the intensities have been removed. Removing the intensities also meant removing the white patches from the image. The blue channel appears to have had the least influence, whilst red the most.

The grayscale space carries only information about the intensity values. In the previous part, we are implementing a function that takes as input an RGB image and transforms it into a gray image. We are using 3 algorithms as specified here `https://www.johndcook.com/blog/2009/08/24/algorithms-convert-color-grayscale/`. As seen in the images displayed in the previous section the lightness method produces a brighter version than the previous ones. This method is an extension of the average method, but it weights the green channel more heavily as we are more sensitive to it. In terms of computation, the average method is the fastest, but less complex than the other ones. It does not represent well shades of gray relative to the way humans perceive luminosity. We can see that in the 'average' method image, the colors appear more faded. The lightness method works by converting the image to the HSL color space and forcing the saturation to be zero. From the images, we can see that this method results in the darkest and flat image. As expected, OpenCV uses the luminosity method, since being the more accurate to the human visual system.

3.4 More on Color Spaces

Another color space from the literature is $L * a * b$ which represents a non-linear remapping of the XYZ space. Differences in luminance or chrominance are more uniform. This color space defines colors indipendently of how they are created and is widely used in printing. The forward transformations that can be applied to get this color space

are:

$$L^* = 116f\left(\frac{Y}{Y_n}\right)$$
$$a^* = 500\left[f\left(\frac{X}{X_n}\right) - f\left(\frac{Y}{Y_n}\right)\right]$$
$$b^* = 200\left[f\left(\frac{Y}{Y_n}\right) - f\left(\frac{Z}{Z_n}\right)\right]$$

4 Intrinsic Image Decomposition

4.1 Other intrinsic components

In my opinion, an other intrinsic components of an image could be noise. Inevitably, images are contaminated by noise during acquisition or compression. Therefore, in order to recover the true original image, we need to take into consideration the noise that it might have. The technique is nowadays called image denoising. Reference:`https://link.springer.com/content/pdf/10.1186/s42492-019-0016-7.pdf`
In this paper, multiple techniques are presented and it states that the biggest challenge with this issue is that it is an inverse problem and its solution is not unique. Another paper states that texture can be considered an intrinsic component of an image. Within their decomposing procedure, texture was removed in the first step, and then, the smooth picture was decomposed into shading and reflectance. Texture was in the end added back to the image based on the material of each pixel. Reference: `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5161468/`

4.2 Synthetic images

To solve the intrinsic decomposition problem, required prior knowledge on the reflectance and shading. A single input image can be explained by a multitude of reflectance and illumination combinations. By using synthetic images, researchers are able to reproduce many of the challenges of the real-world scene. One of the synthetic images dataset that was used by Qifeng Chen and Vladlen Koltun was MPI-Sintel. This was appreciated because it had similar statistics as natural images and provided images with complex shapes, complex lighting and occlusions. Reference: `https://cqf.io/papers/Intrinsic_Decomposition_ICCV2013.pdf`

4.3 Image Formation

We know that an image can be decomposed into its intrinsic components, the albedo and shading. Therefore, we are going to show that this holds and that if we have the intrinsic components images, we can easily construct the original image. In order to recuperate the original image, we perform a element wise multiplication of the albedo and the shading images. That is performed using the builtin function provided by OpenCV.

Figure 25. The original image

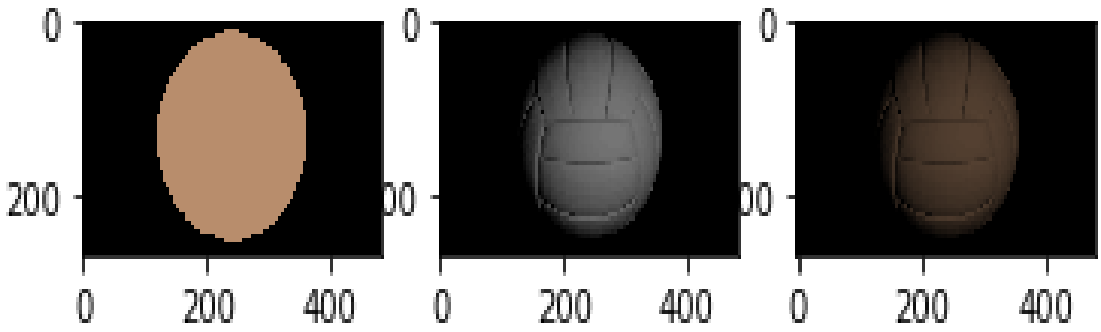

Figure 26. The albedo, shading and reconstructed image

4.4 Recoloring

1. To find out the true color of the ball, we first chose the pixel where the values of three channels are not all 0, then summed up the values of the three channels individually then took the average. The resulting values are (56, 43, 33). This color in the RGB space corresponds to the sequence # 382B21 which is deep dark orange visually.

Figure 27. The original ball color and the ball recolored with pure green (0,255,0).

2. To recolor the ball with pure gree color (0, 255, 0), we first picked the pixels where one of the channel values is not 0, then recolor it with the pure green color. The recolored ball image is not uniformly colored with the new color (figure 27 (right)).

3. From the original ball image (figure 27 (left)) we can observe that there are several dark-colored lines where these lines are served to increase the friction, these lines are the ones that don't have the new pure green color. These lines are the main reason of having the uneven distributed green color. When recoloring the image with the script Recolor.py, the pixels forming these lines which are pure black are not chosen, causing these lines remain the original color (black). The light source and the angle of the light source are important when recoloring the object, they should both be considered.

5 Colour Constancy

Grey-World Algorithm

1. The assumption of the grey-world algorithm is that under the white light source, the average of all the colors in the same image is normal grey. In the example of *awb.png* (figure 28 (left)), the reddish color is scaled by the average of the three colors, which gives a more natural color (figure 28 (right)).
Steps of the Grey-World algorithm

(1)

$$Gray = \frac{\overline{R} + \overline{G} + \overline{B}}{3}$$

where $\overline{R}, \overline{G}, \overline{B}$ are the average values for the three channels respectively.

(2) Calculate the von Kries coefficients:

$$k_r = \frac{Grey}{\overline{R}}, k_g = \frac{Grey}{\overline{G}}, k_b = \frac{Grey}{\overline{B}}$$

(3) Adapt each pixel color according to von Kries hypothesis [2]:

$$P(R') = P(R) * k_r$$
$$P(G') = P(G) * k_g$$
$$P(B') = P(B) * k_b$$

where P means the pixel value.

Figure 28. Color correct using Grey-World algorithm. (Left) The original image. (Right) The color-corrected image using Grey-World algorithm. The reddish color is removed.

2. When the variety of the colors in the same image is not high enough (i.e. single color or only a few kind of colors), the grey-world algorithm might fail due to the imbalance of the RGB composition.
From figure 29, the original image is composed of mainly two colors, blue and white, which is one of the example with very few color varieties. The resulting image (figure 29 (right) shows unnatural colors where the sky is dyed orange and over-exposed. This example proves that with insufficient kinds of colors, the grey-world algorithm would fail.

3. [1] compared several color constancy algorithms. One of the algorithms is very similar to the gray-world algorithm, which is the Scale By Max approach. According to this approach, the estimate of the illuminant is the maximum of the pixel values of each channel. It is a subset of the Bayesian approach under the assumption that the reflectance is independent and uniform [3].

$$l_r = mean(E_R), l_g = mean(E_G), l_b = mean(E_B)$$

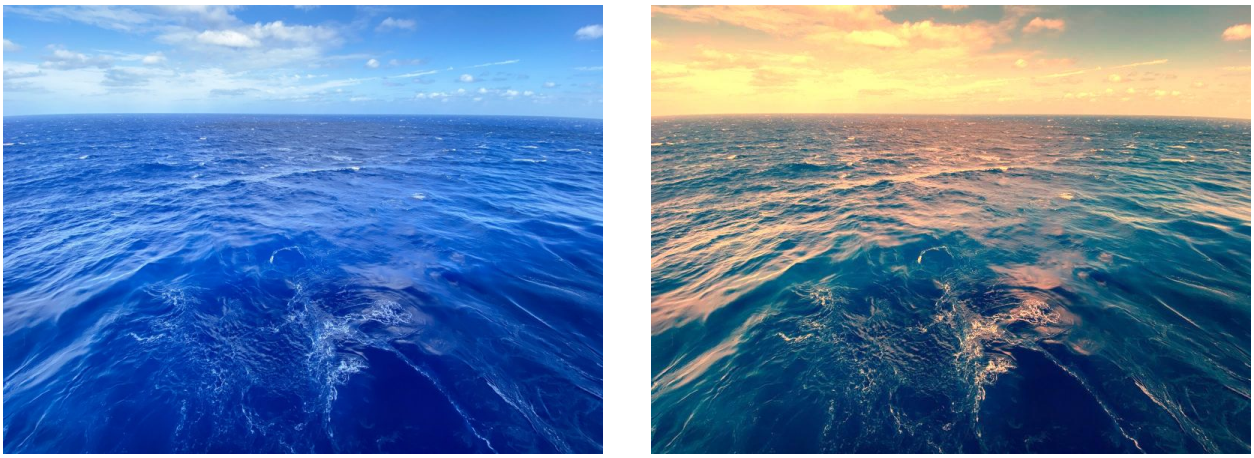

Figure 29. Color correct using Grey-World algorithm. (Left) The original image. (Right) The color-corrected image using Grey-World algorithm. The natural blue ocean, sky and the cloud are too reddish.

where l_r, l_g, l_b are the mean value of each channel respectively and E_R, E_G, E_B are individual image channel.

6 Bibliography

[1] AGARWAL, V., ABIDI, B. R., KOSCHAN, A., AND ABIDI, M. A. An overview of color constancy algorithms. *Journal of Pattern Recognition Research 1*, 1 (2006), 42–54.

[2] KRIES, J. Die gesichtsempfindungen. *Nagel's Handbuch der Physiologie des Menschen 3* (1905), 109.

[3] ROSENBERG, C., LADSARIYA, A., AND MINKA, T. Bayesian color constancy with non-gaussian models. In *Advances in neural information processing systems* (2004), pp. 1595–1602.

7 Appendix

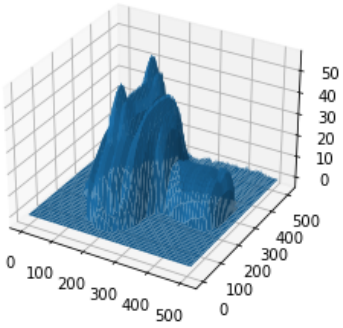

Figure 30. Surface reconstruction 121 images

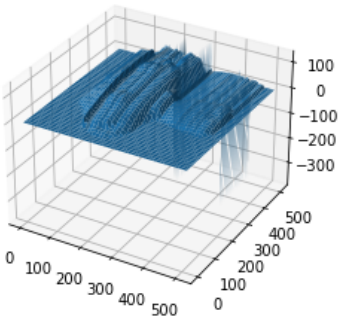

Figure 31. Surface reconstruction 30 images

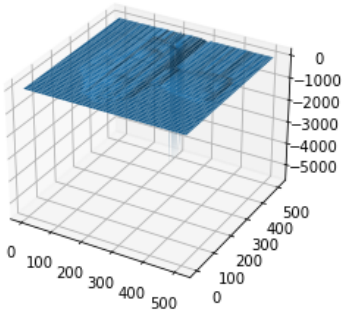

Figure 32. Surface reconstruction 60 images

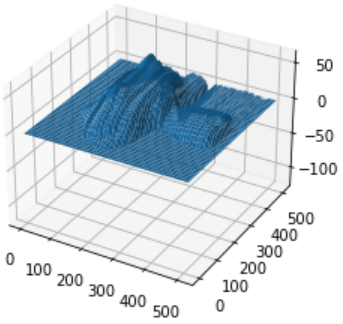

Figure 33. Surface reconstruction 90 images

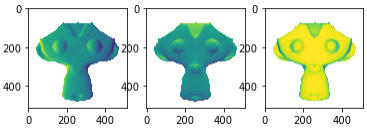

Figure 34. Normals 121 images

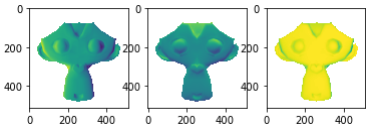

Figure 35. Normals 60 images

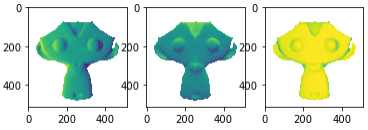

Figure 36. Normals 30 images

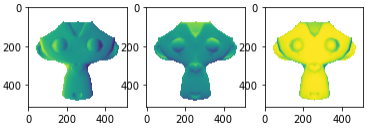

Figure 37. Normals 90 images

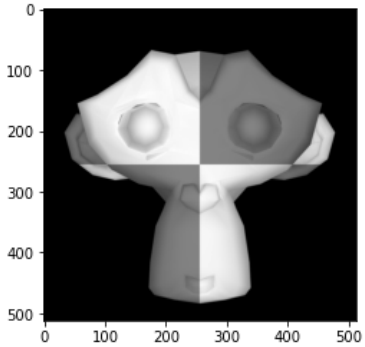

Figure 38. Albedo 121 images

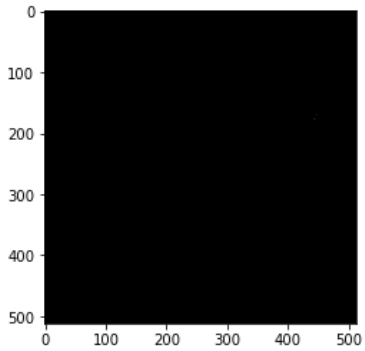

Figure 39. Albedo 30 images

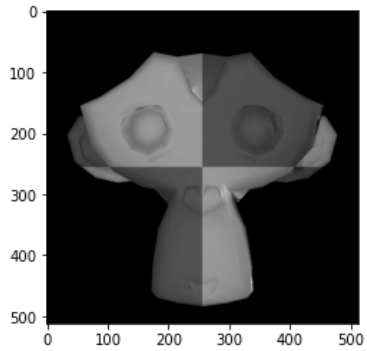

Figure 40. Albedo 60 images

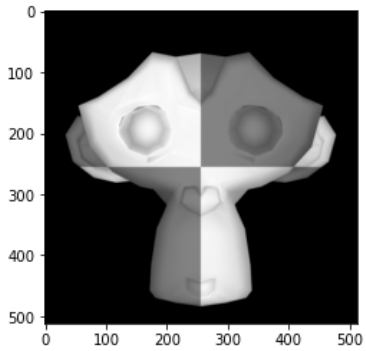

Figure 41. Albedo 90 images

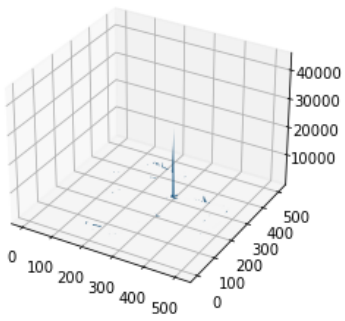

Figure 42. The Error 30 images - 8246 outliers

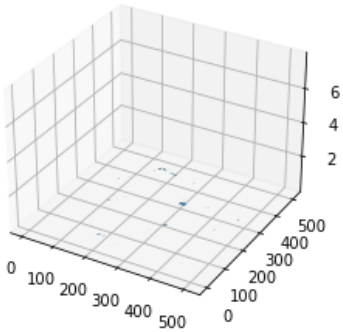

Figure 43. The error 60 images - 6635 outliers

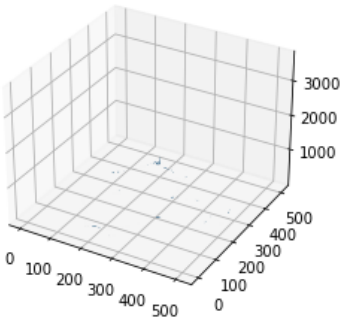

Figure 44. The error 90 images - 5318 outliers

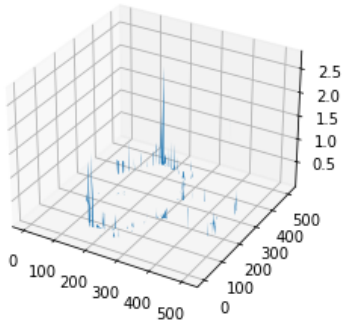

Figure 45. The error 121 images - 4546 outliers

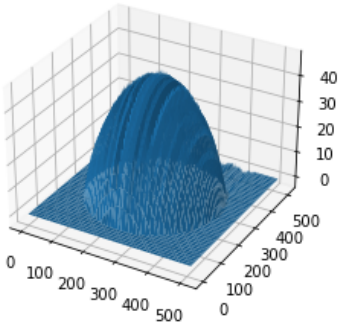

Figure 46. Colored sphere surface map, red channel

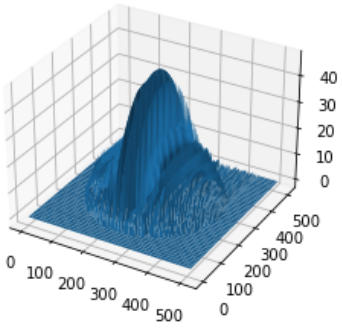

Figure 47. Colored sphere surface map, blue channel

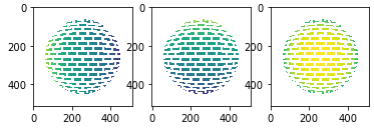

Figure 48. Colored sphere normals, red channel

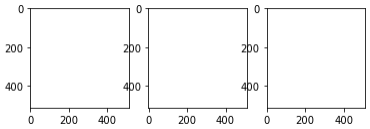

Figure 49. Colored sphere normals, blue channel

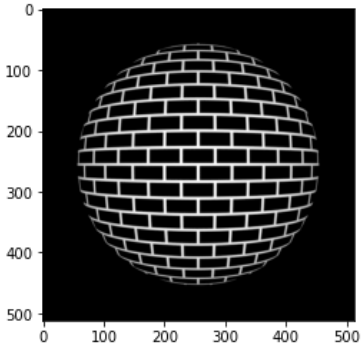

Figure 50. Colored sphere albedo, blue channel

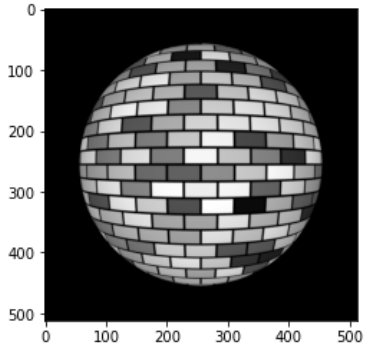

Figure 51. Colored sphere albedo, red channel

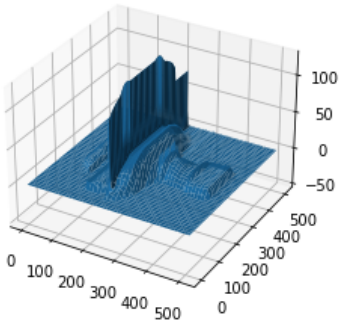

Figure 52. Colored monkey surface map, red channel

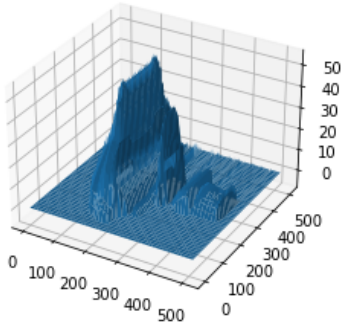

Figure 53. Colored monkey surface map, blue channel

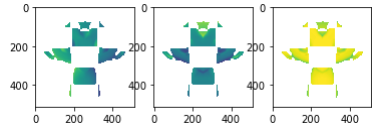

Figure 54. Colored monkey normals, red channel

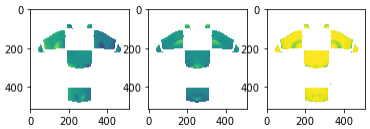

Figure 55. Colored monkey normals, blue channel

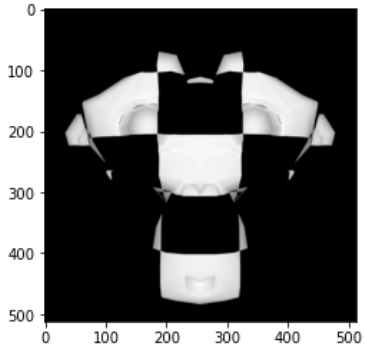

Figure 56. Colored monkey albedo, blue channel

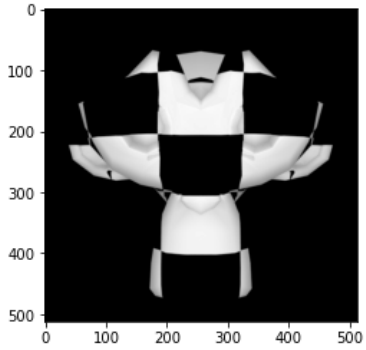

Figure 57. Colored monkey albedo, red channel

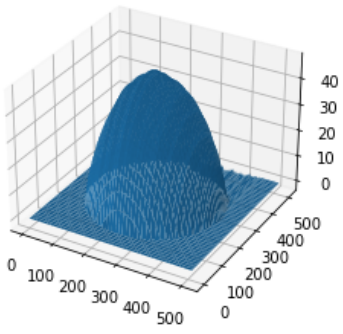

Figure 58. The reconstruction of the sphere using 25 images using columns path

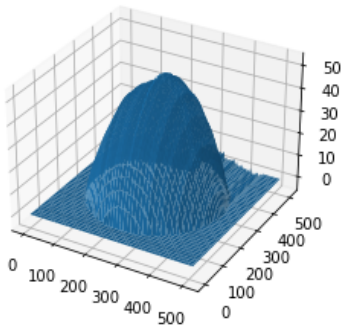

Figure 59. The reconstruction of the sphere using 5 images using column path

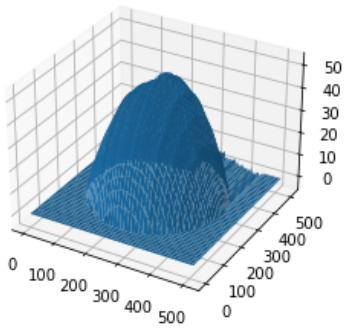

Figure 60. The reconstruction of the sphere using 5 images using row path