1 Introduction

In this assignment, we implement many common algorithms in computer vision field. Firstly, we focus on photometric stereo algorithm, which is a method to estimate object's surface and reflection. Different number of images, shape of objects, color of images and real life images are used to experiment algorithms' strength and limitation.

Secondly, we go into the properties of different color spaces and will compare results for an RGB image that is converted into different color spaces.

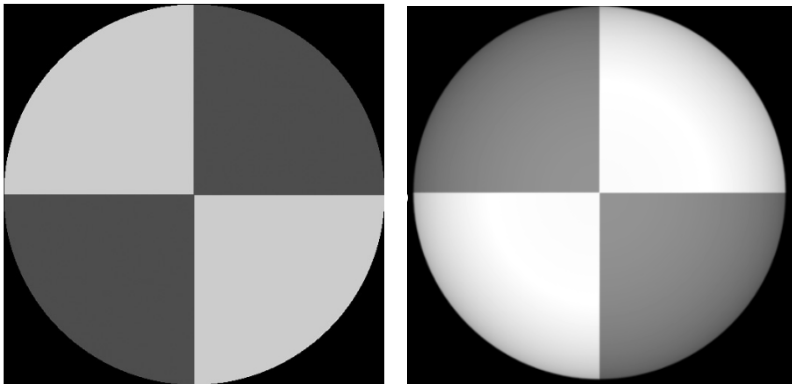
In the third experiment, we investigate how images can be reconstructed from their intrinsic components. Specifically, this experiment aims to reconstruct synthetic images from their albedo and shading components.

The last part covers the grey world algorithm, which is a color constancy method. The purpose of this algorithm is to correct the color in unnatural images that is caused by the light source.

2 Photometric Stereo

2.1 Estimating Albedo and Surface Normal

Question 1.1.1. After completing the code we use *SphereGray5* to calculate the albedo. Figure 1 shows our computed result together with Figure 5.11 from the book *Computer Vision: A Modern Approach* [9]. Besides a 90° rotation, we expect our result to be similar to the one from the book, i.e. within each quarter of the circle the albedo should be uniform. But as Figure 1(b) shows, our albedo is not smoothed and uniform as we expect. There is a color gradient within the four quarters of the circle. Besides, at the edges of sphere our result has a blurry and dark rim.

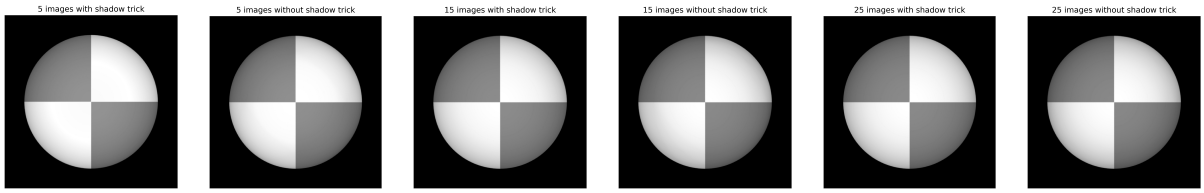


(a) Reference albedo (taken from [9]) (b) Computed albedo for five sources

Figure 1: Albedo comparison: The left image shows how the albedo should look like, besides from a 90° rotation. The right image was obtained from our own implementation.

Question 1.1.2. The minimum number of images to estimate the albedo and surface normal is three because we need to solve the linear system $i(x, y) = \nu g(x, y)$ to obtain $g(x, y)$, where $g(x, y) = \rho(x, y)N(x, y)$ describes the surface of the object. Since $g(x, y)$ is a three dimensional vector, there are three unknown variables in that linear system. With each image giving us one equation, we need at least three to solve for these three unknowns. Furthermore, we can use more than three images and use least square solution to get better estimation of albedo and surface normal.

We get a better estimation of albedo when using *SphereGray25*. Comparing with 5 images, as Figure 2(e) shows, we see the color gradient within the four quarters becomes weak and the dark rim of edge are thinner and clear. We also perform different number of images to see the influence. The great improvement of estimated albedo happens in the beginning especially between 13 and 15. Figure 2(c) shows the estimated albedo of 15 images. While after 15 images the results become consistent. Especially between 20 and 25 images difference among the results is small.



(a) 5 images with shadow trick (b) 5 without shadow trick (c) 15 images with shadow trick (d) 15 images without shadow trick (e) 25 images with shadow trick (f) 25 images without shadow trick

Figure 2: Albedo of 5, 15 and 25 images with or without shadow trick

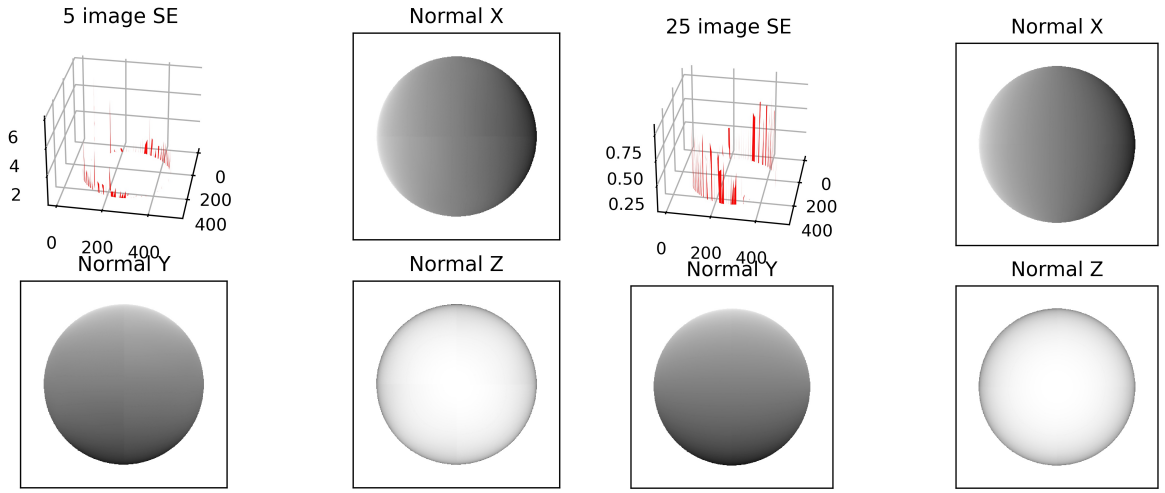
Question 1.1.3. Shadow brings problem in photometric stereo, which covers the true pixel value in different light condition. For example, pixel (x, y) of image i is shadow because of light source, which means $i(x, y) = 0$. Thus, the equation becomes $0 = \nu^T g(x, y)$. When solving the least square equation we add a constraint on this pixel because the true value is shaded by shadow.

Shadow trick is a method to solve this constraint. Shadow trick generates a diagonal matrix I according to $i(x, y)$ and the equation becomes: $Ii(x, y) = IV^T g(x, y)$. For example, if a pixel is still under same condition we listed above, the corresponding row in diagonal matrix I will become 0. Thus, the left side of equation is 0 and right side of equation becomes $0 * V^T g(x, y)$ which is also 0. Thus the equation remain $0 = 0$ and the constrain is reduced.

As Figure 2(a) and 2(b) show, there is no significant difference between with and without shadow trick. While for 2(e) and 2(f) we see that image with shadow trick is slightly better than image without shadow trick, which is intuitive because 25 images have many shadow because of different light sources. Thus, for 5 images the shadow trick is negligible and for 25 images we can use shadow trick.

2.2 Test of integrability

Question 2. Using 0.005 threshold we can see there are errors at edge of sphere for both 5 and 25 images. These errors can be find in both normal X and normal Y maps as figure 3 shows. Errors are caused by the fast transition from edge of sphere to dark environment. This quick transition makes approximation difficult and causes errors. However, the errors decreases as the number of images increases. From 3D SE plot we can see the error of 5 images are significantly bigger than 25 images according to Z axis. And the number of outliers decreases from 3297 to 2672 as the number of images increases from 5 to 25.



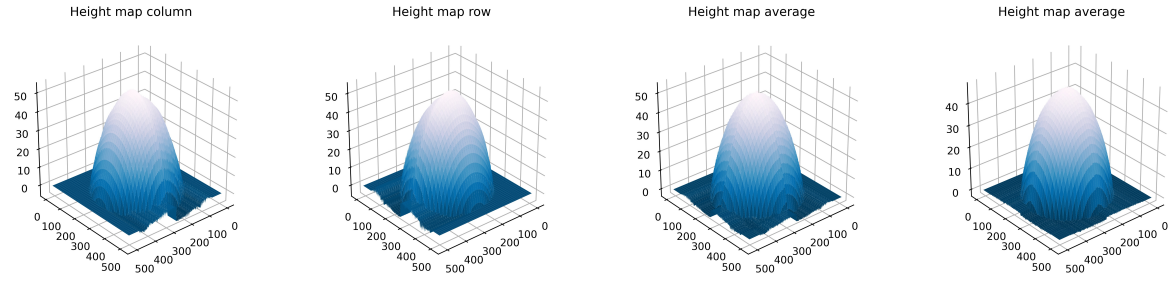
(a) 5 images SE and corresponding normal heat maps (with shadow trick) (b) 25 images SE and corresponding normal heat maps (with shadow trick)

Figure 3: The 3D plots on the top left show the squared error of 5 and 25 images. The normal heat maps for the X, Y and Z dimensions are on the top right, bottom left and bottom right respectively.

2.3 Shape by Integration

Question 3.1: From Figure 4(a) we can see that in the row direction the height map is distorted, while in Figure 4(b) the height map is distorted in column direction. That's because when using row or column method the errors on the edge accumulate over columns or rows.

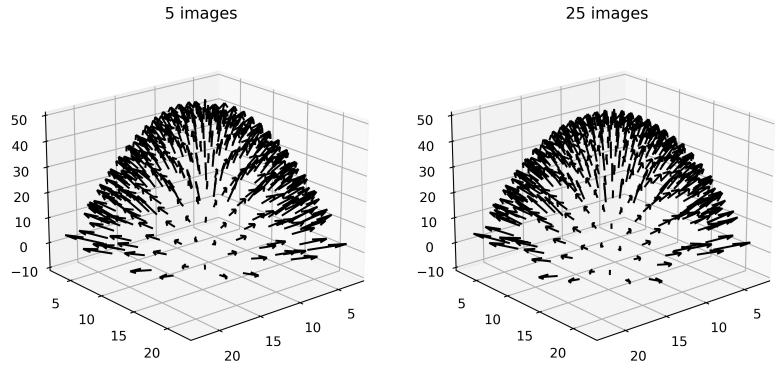
Question 3.2: Figure 4(c) shows a slightly different outcome for the average method. We can see figure 4(c) is distorted in both column and row direction. But the magnitude of fluctuation is smaller than only column or row method since it calculates the mean of two methods. We expect the errors to decrease as we use more images. Indeed, we can see from Figure 4(d) that there is a smaller distortion in 25 images for the average method when compared to 5 images. The accumulated errors become smaller and the height map become smoother. Thus, average method brings smaller errors and result of 25 images outperforms 5 images.



(a) 5 images column method (b) 5 images row method (c) 5 images average method (d) 25 images average method

Figure 4: The three leftmost plots are height maps for the column, row and average method for five source images, respectively. The rightmost height map uses the average methods for 25 images.

Figure 5 shows the surface normal recovery of 5 images and 25 images by **Quiver** function. It is obvious that we do sub-sampling for X and Y axis and sub-sampling ratio is 20. Additionally, the X and Y axes were scaled as well. We can see there is no significant difference between two images. The only difference occurs at edge of surface. We can see the arrows on the top of surface are compact and smooth for 25 images. While for 5 images the arrows seem random and loose. Thus, the large number of images indeed improve the surface normal reconstruction.



(a) 5 images surface normal construction (b) 25 images surface normal construction

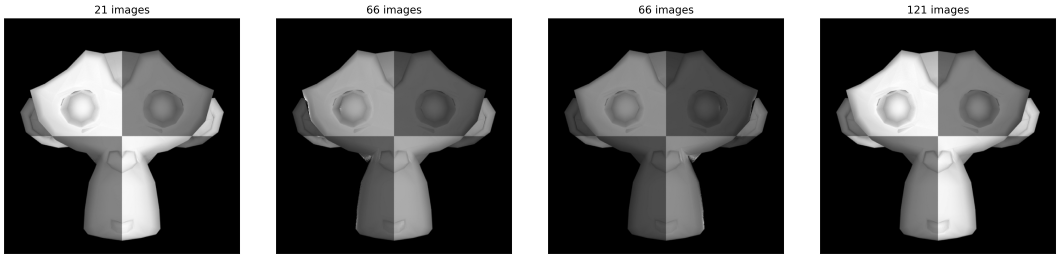
Figure 5: Visualization of the normal vector field for 5 (a) and 25 (b) source images.

2.4 Experiments with different objects

Question 4: Furthermore, we experiment different objects to test photometric stereo algorithm. Firstly, we do experiment on MonkeyGray dataset. Apart from the common error of albedo along the edge of monkey, we can see new arises from ears, eyes and nose. We can see the inside albedo of these places is a little bit shading but these places should have same albedo as outside. The possible explanation is the ambient light can not reach these place which causes the difference of albedo. The results show as Figure 6(d).

Additionally, we investigate the influence of varying number of source images and position of light source on albedo. Figure 6(a) shows the albedo for 21 source images with all the light sources being positioned on the centered axes ($x = 0$ or $y = 0$). We can see that there is no significant difference when compared to Figure 6(d) which uses 121 images. The color gradient from center to edge is more obvious in Figure 6(a) than in Figure 6(d). While comparing with Figure 6(b), the result shows great difference. It is visible that Figure 6(b) is darker than in Figure 6(a) and we can also see that there are some extremely bright areas around left head, neck and body. That is because we use only left side light sources ($x \leq 0$). The same effects also happen in right side. Figure 6(c) shows opposite side albedo of 6(b). In short, the number of images show less significant

influence on albedo. But the light source position shows important impact on albedo. Thus, in order to solve the errors more light sources should be put near the centered axes.

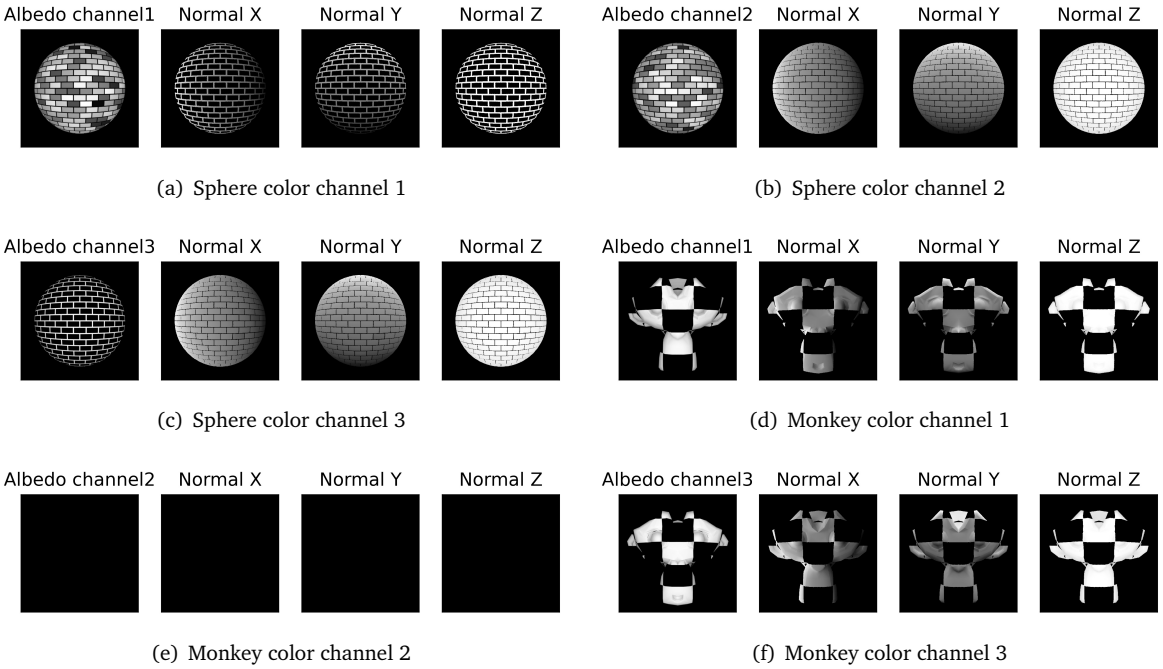


(a) 21 images with middle light sources (b) 66 images with left light sources (c) 21 images with right light sources (d) 121 images

Figure 6: MonkeyGray albedo with different number of images and light sources (with shadow trick).

Question 5 Furthermore, we experiment with colored images. To work with 3-channel images, a channel value is added to the *load_syn_images* function in the code and we stack all channels into image stack variable. Then, when calculating albedo and normals we treat each channel separately. Then every channel has one albedo and three surface normals. In then end, albedo has three channel and normals have nine surface normals. The code are listed in *photometric_stereo.py*. And because the datatype is *unit8*, which is difficult to visualize, we change it into *float32* and normalize normals. The corresponding code is listed in *utils.py*

The normals and height maps for the different channels are shown in Figure 7 and 8. The normal map for channel 2 of MonkeyColor.jpg is (almost) completely black. Also the height map for channel 2 of MonkeyColor.jpg is completely flat. This indicates the problem with treating the three channels individually: when a pixel has value 0 for one of the channels, it seems that there is no reflection for that pixel. And when solving a pixel with value 0 the equation will return NaN values for albedo. However that is not the case as the pixel has values for the other two channels. A way to solve this is to not treat the channels individually but to combine them by, for example, taking the mean of the channels. Another option is to replace the NaN values with zero.

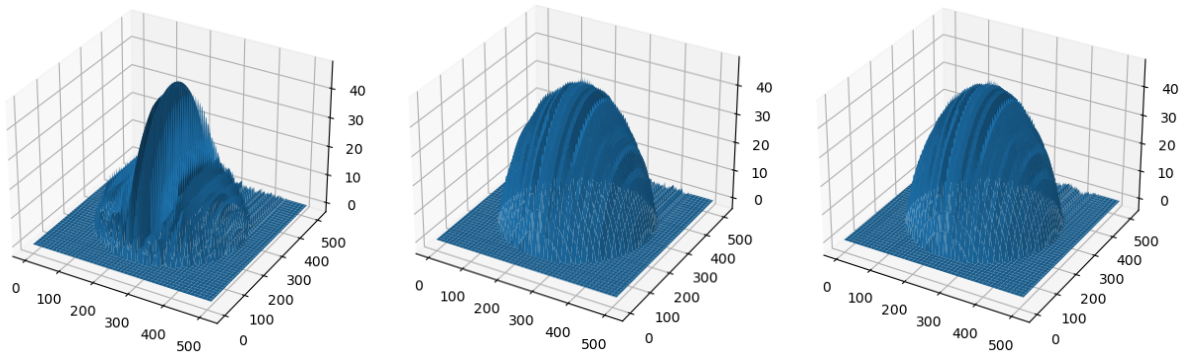


(a) Sphere color channel 1 (b) Sphere color channel 2

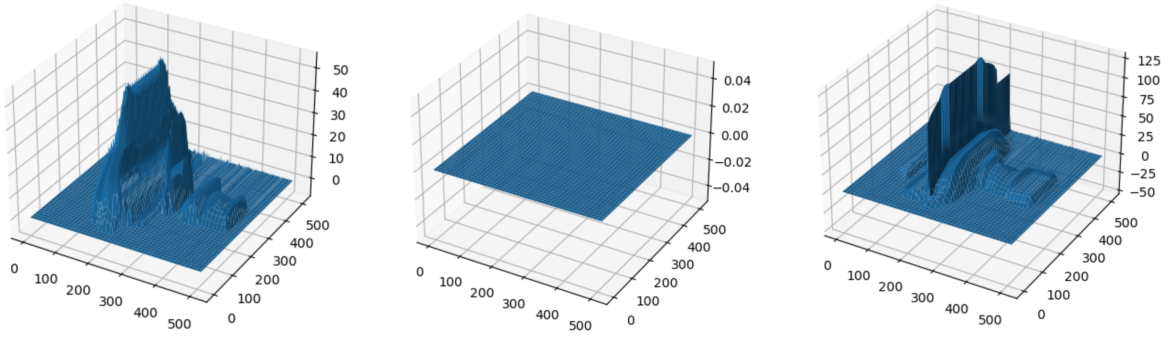
(c) Sphere color channel 3 (d) Monkey color channel 1

(e) Monkey color channel 2 (f) Monkey color channel 3

Figure 7: Albedo and surface normal for separate channels. (a)-(c) are estimated albedo and surface normals for each channel of SphereColor. (d)-(f) are estimated albedo and surface normals for each channel of MonkeyColor. All albedo and normals are calculated without shadow trick.



(a) channel 1 (b) channel 2 (c) channel 3

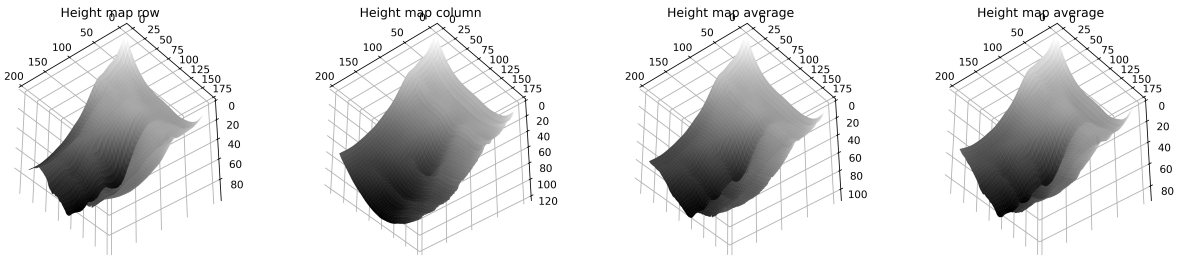


(d) channel 1 (e) channel 2 (f) channel 3

Figure 8: Height map for separate channels(average method without shadow trick

Question 6: Finally, we investigate a dataset with complex structure and light sources. Figure 9(a), 9(b) and 9(c) show height maps for the different integration paths. As we discussed before, the row method outperforms column method. We can see that column method loses many structural details, especially around the nose and the image seems flat, while the row method has sharp structural details and the image seems squeezed. The average method reduces these two effects and shows decent outcome because it compensates errors of the two previous methods.

When we carefully investigate the images in Yale Face files, we find some images violate the assumption which is that there is a single distant point light source. We find that some images are lighted by long specular light. Especially around the nose, there is a bright highlight which doesn't match a point light source. Another problem is that when the light source is behind the face, we can still see some structure of the face. This indeed violates the assumption because the face should be completely dark if there is no ambient light source. The possible explanation is there are surrounding surfaces which reflect light onto the face. Thus, we remove 7 images and Figure 9(d) shows the final average method. There is a slight improvement when compared to Figure 9(c). The structure becomes sharper and clearer and the map become smoother.



(a) Height map of row method (b) Height map of column method (c) Height map of average method (d) Height map of average method after correction

Figure 9: Yale Face height map. The three left plots are height maps of column, row and average method respectively. The rightmost plot is height map of average method after correction.

3 Colour spaces

Question 2.1: RGB Colour Model
We use RGB colour model as basis for digital cameras because it closely mimics the way our eye/brain system responds to light[11]. Different colours in our eyes are detected by cones. Humans normally have three different types of cones. The first type responds most to light with longer wavelengths (peaking at about 585 nm, corresponding with red-ish light), the second type responds most to light with medium wavelengths (peaking at about 540 nm, corresponding with green-ish light) and the third type responds most to light with short wavelengths (peaking at about 447 nm, corresponding with blue-ish light). The RGB filter in digital cameras mimic this behaviour[13].

Digital cameras capture RGB colors by using three types sensors, each of which is designated for a specific color. The sensors are preceded by a color filter that only let either red, green or blue light pass. A widely used choice for filter patterns is the so-called Bayer filter[3]. In this method, sensors are arranged in a checkered pattern, with half of the sensors capturing green light, one quarter capturing red light and the last quarter capturing blue light. In order to get complete images for each color, missing values are interpolated from neighbouring sensor values.

Question 2.2: Colour Space Properties
For code implementation, see provided python files. The output for the different colour spaces are shown in figure 10.
Color spaces and their properties:

- **Opponent**: The opponent theory is a different major theory about how colours are detected by the human eye/brain-system. It states that colour is coded in opponent pairs: black-white, yellow-blue and red-green. The opponent method implicates that the opponent colours are never experienced at the same time e.g., we never experience yellowish-blue or greenish-red[12]. The opponent method also explains the experience of afterimages. Both the trichromatic (RGB) theory and the opponent theory explain the way in which human perceive colours: the trichromatic theory mimics the way the retina of our eyes behaves and the opponent theory mimics the way our brain interprets the signal transmitted by our eyes[5]. Opponent colour space is well suited for computational applications while still containing the qualitative concept op color warmth as required by some artistic applications.

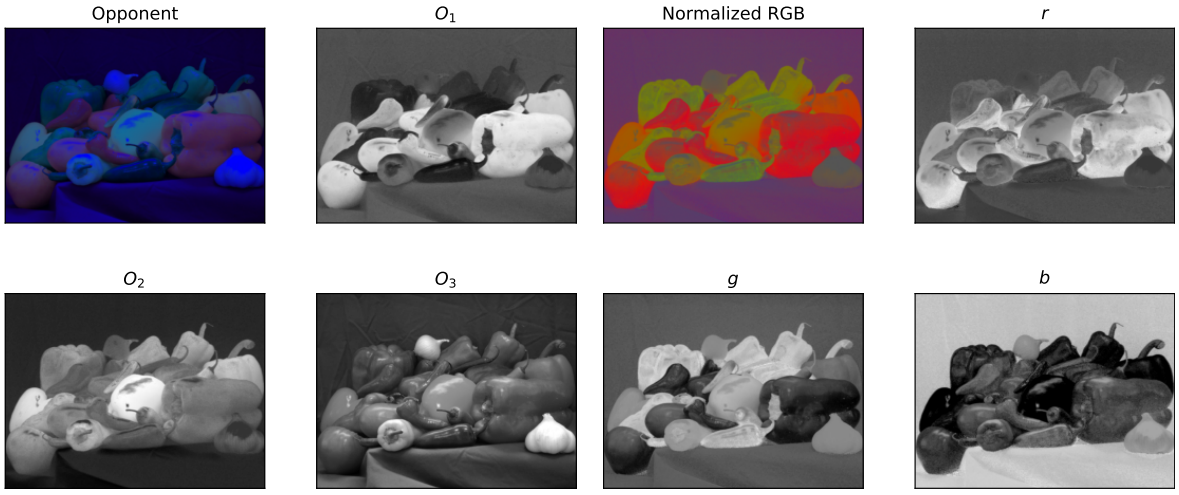
- **Normalized RGB**: By normalizing each channel by the sum of all channels, one eliminates the luminance property of an image. Visually, we can see that the normalized RGB image isn't as detailed as the original image, e.g. minor differences in color, caused by shadows, are evened out. This is very noticeable when comparing the purple background of the image. This color space might be useful when performing object detection tasks like segmentation[14].

- **HSV**: The HSV model that uses Hue, Saturation and Value. Hue is the main colour, the Saturation appoint the 'amount' or 'brightness' of the colour (from pale to full colour) and the value appoints the brightness of the colour (from black to white). This model aligns more closely to the way humans perceive colours with all it's attributes. The HSV model is used in a lot of computer based graphics. One of the most know uses is the 'colour picker'[4].

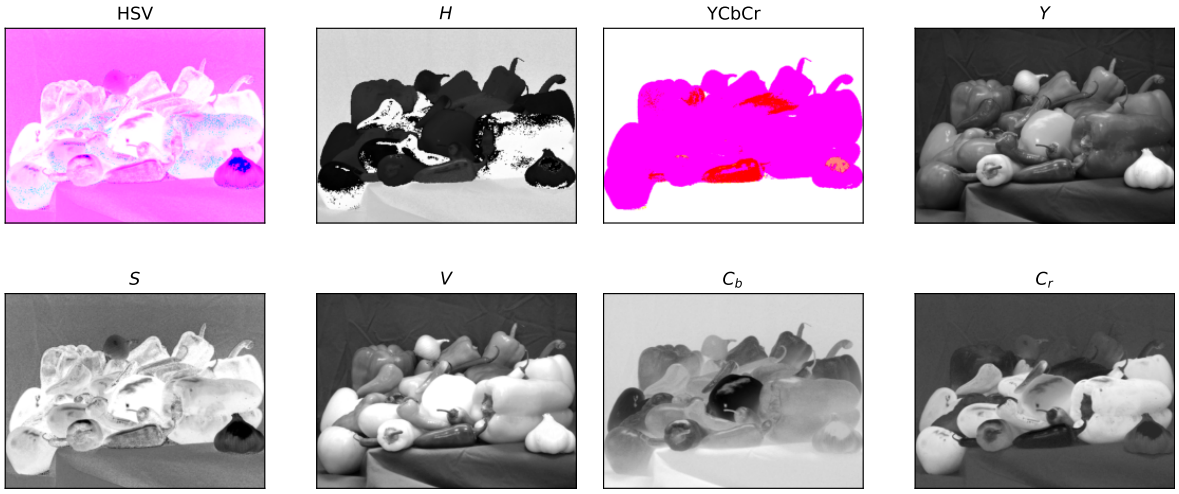
- **YCbCr**: YCbCr is a colour space model that describes colours in three components: the luma signal (Y) which represent the brightness of the colour, the blue-difference chroma component (Cb) and the red-difference chroma component (Cr). This is colour space model can efficiently be used for chroma subsampling: reducing file size by implementing less resolution for chroma information than for the luma information, taking advantage of the human's lower sensitivity for color differences than for luminance[6].

- **Grayscale**: Grayscale is a colour space where every pixel is only representing the amount light captured at that pixel. The contrast of the pixels ranges from black to white. Grayscale images were widely used before the invention (and wide adaptation) of color photography. Nowadays it is commonly used for reducing the file size compared to coloured images. OpenCV uses a similar method to the luminosity method, which is computing a weighted sum of the RGB channels. Their transformation is given by $0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$. [1].

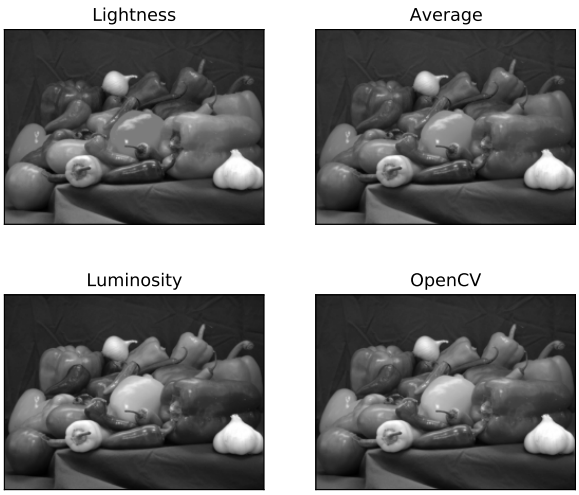
Another colour space is CMYK, which is primarily used in printing. In contrast to RGB, this color space combines cyan, magenta, yellow and key (black) colours. The CMYK model is a subtractive model: it 'subtracts' the reflection of light from a lighter (usually white) background. White light minus red gives us cyan, white light minus green gives us magenta and white light minus blue give us yellow[7].



(a) Opponent (b) Normalized RGB



(c) HSV (d) YCbCr



(e) Grayscale

Figure 10: Different colour spaces and their channels

4 Intrinsic Image Decomposition

Image decomposition tackles the task of seperating a source image into different components. In the following experiment we will investigate the decomposition of an image into its albedo and shading components. Another method is the decomposition into low and high frequency features of the image, i.e. structures (low-frequency features) and textures (high-frequency features) are seperated into individual images. The original image can be reconstructed from these image components by computing a linear combination over all extracted features [8].

Intrinsic image decomposition datasets usually contain synthetic images since it's difficult to obtain the real albedo and shading components for real images. While the reconstruction $I = R \times S$ is fairly easy to compute from given albedo and shading images, it is very difficult to solve the factorization problem and obtain R and S from I. Furthermore, the problem becomes even more complex due to the high number of pixels in an image for which we would have so solve this.

Figure 11 shows the reconstruction of the ball image from the given albedo and shading. We can observe that the color of the ball has been reconstructed succesfully by incorporating the shading. Compared to the albedo image, the new image has depth to it.

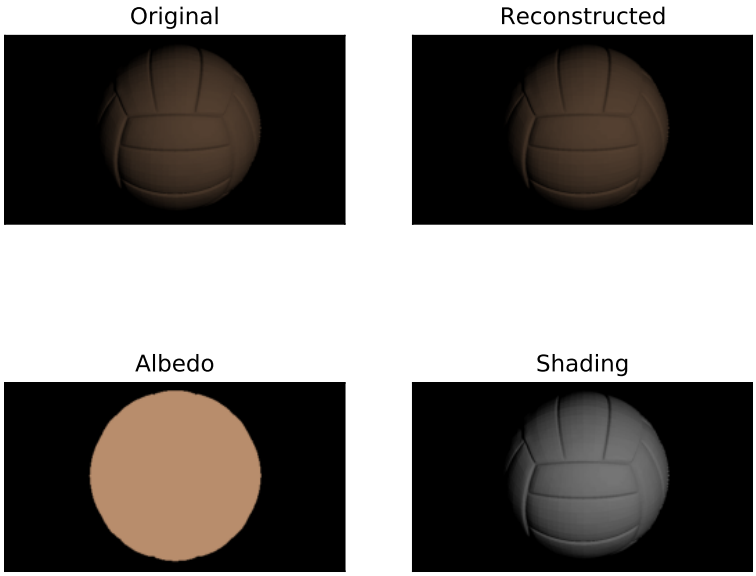


Figure 11: Intrinsic image decomposition: The reconstructed image was constructed by multiplying the albedo with the shading image.

In order to recolor the ball, we first obtained its color in the given albedo image. Using the **numpy.unique** function, we derived the color of the ball to be $[184, 141, 108]$. We proceeded to set all albedo values of 184 and 108 to zero and values of 141 to 255, which gave us the recolored albedo. The recolored reconstructed image can be seen in Figure 12. Even though we used an albedo image which had a pure green ball, the reconstructed image doesn't contain that green color. This is because of the shading image which is multiplied with the albedo image and basically acts as a scaling factor. Only a value of 255 in the shading image would show the real color of the ball in the reconstruction, but the maximum value in the given shading image is only 149, i.e. we aren't able to observe the true color of the ball in the reconstruction.

Figure 12: Recolored image reconstruction: The right image was created by recoloring the albedo image of the corresponding original image.

5 Color Constancy

The grey-world algorithm is based on the assumption that the means of each channel in RGB images are equal if the scence is illuminated by a white light source. According to the Von Kries model, the change in color, due to a non white light source, is given by

$$C = \frac{\bar{R} + \bar{G} + \bar{B}}{\bar{C}} C_{\text{nat}} \tag{1}$$

with $C \in \{R, G, B\}$ and $\bar{R}, \bar{G}$ and $\bar{B}$ being the averages of the respective channel. C_{nat} denotes the channel value in the natural scene, while C is the measured channel value. This transformation results in an image that has channels with equal means, which corresponds to the assumption of the grey-world method [10].

The result of the color corrected example image is visualized in Figure 13. We can clearly see that the reddish color cast is removed in the corrected image. As mentioned above, the outcome of the grey-world algorithm is an image that has channels with equal means. In the example shown in Figure 13, the corrected image looks more natural having equal mean channels, but this is not always the case. The image that was used in the second task looks perfectly natural, but has channels with different means, namely $\bar{R} \approx 121$, $\bar{G} \approx 66$ and $\bar{B} \approx 58$. Therefore, the grey-world algorithm only works for images that have near equal mean channels under natural conditions.

Another color constancy algorithm is the scale-by-max method. In contrast to the gray world algorithm, this method estimates the color of the light source by computing the maximum over each channel. The algorithm is based on the assumption that the maximum value in each channel corresponds to the full reflection of the light source. It is implemented in a similar way as the gray world algorithm, but it's using the maximum over each channel instead of the mean [2].

Figure 13: Grey-world algorithm: The image on the right shows the color corrected version of the left image. The corrected image appears more natural.

6 Conclusion

In this assignment, we have learned many common computer vision algorithms. Photometric stereo shows decent results with large number of images but fails when the objects are too complex or images violate the assumptions.

Furthermore, we learned about the properties of five different color spaces and implemented the conversions between these color models.

Intrinsic image decomposition was covered as well, where we implemented a method to reconstruct synthetic images from their albedo and shading components sucessfully. The decomposition method also allowed us to recolor a specific image by varying its albedo component.

Finally, we saw how color errors in images can be corrected using the grey-world algorithm. We implemented and applied the algorithm to succesfully obtain a corrected version of an image that looked unnatural originally.

References

[1] Opencv. https://docs.opencv.org/3.4/de/d25/imgproc_color_conversions.html.

[2] BARNARD, K., CARDEI, V., AND FUNT, B. A comparison of computational color constancy algorithms. i: Methodology and experiments with synthesized data. *IEEE Transactions in Image Processing, Vol. 11* (2002).

[3] BAYER, B. E. Patent us3971065 - color imaging arrays.

[4] BEAR, J. H. The hsv color model in graphic design. https://www.lifewire.com/what-is-hsv-in-design-1078068.html.

[5] CAFASSO, J. Opponent process theory. https://www.healthline.com/health/opponent-process-theory.html.

[6] DIAS, D. What is ycbcr ? (color spaces). https://medium.com/breaktheloop/what-is-ycbcr-964fde85eeb3.html, 2017.

[7] DOUGHTY, M. Graphics color models. http://www.sketchpad.net/basics4.htm, 2016.

[8] EVANGELOPOULOS, G., AND MARAGOS, P. Image decomposition into structure and texture subcomponents with multifrequency modulation constraints. *In: CVPR* (2008).

[9] FORSYTH, D. A., AND PONCE, J. *Computer vision: a modern approach*. Prentice Hall Professional Technical Reference, 2002.

[10] GUANGHUA, C., AND XIAOLONG, Z. A method to improve robustness of the gray world algorithm. *In: ICCMCEE* (2015).

[11] HUNT, R. W. G. *The Reproduction of Colour (6th ed.)*. John Wiley Sons Inc., 2004.

[12] MARGARITA BRATKOVA, SOLOMON BOULOS, P. S. orgb: A practical opponent color space for computer graphics. *IEEE Computer Graphics and Applications 29*, 1 (2008), 42–55.

[13] VAN GINKEL, D. G. Color vision, brightness, resolution and contrast in binocular images. a review of published data. https://www.houseofoutdoor.com/testrapporten/COLOR_VISION_BRIGHTNESS_RESOLUTION_AND_CONTRAST_IN_BINOCULAR-IMAGES_highres.pdf.

[14] VANRELL, M., LUMBRERAS, F., PUJOL, A., BALDRICH, R., LLADOS, J., AND VILLANUEVA, J. J. Colour normalisation based on background information. In *Proceedings 2001 International Conference on Image Processing (Cat. No.01CH37205)* (2001), vol. 1, pp. 874–877 vol.1.