Computer Vision 1
Photometric Stereo & Color

Gongze Cao, Kai Liang, Xiaoxiao Wen and Zhenyu Gao
Faculty of Science, University of Amsterdam

1 Introduction

In this assignment, we cover different topics of image and color processing, which are namely 'Photometric Stereo' (see section 2), 'Color Spaces' (see section 3), 'Intrinsic Image Decomposition' (see section 4) and 'Color Constancy' (see section 5).

2 Photometric Stereo

2.1 Question - 1

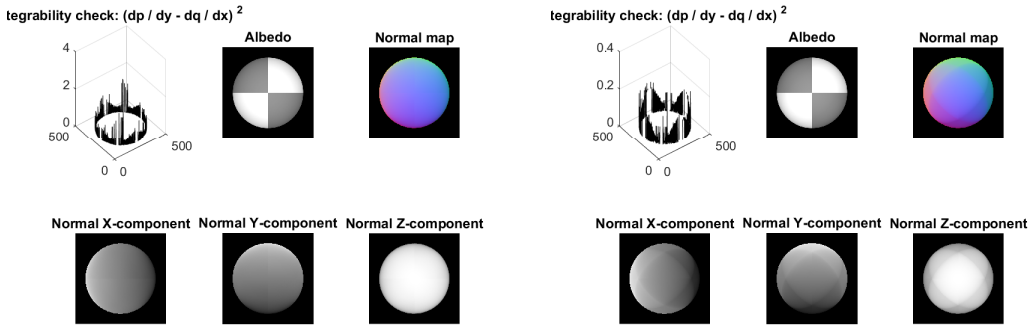
1. The implementations of estimating albedos and surface normals are strictly according to [1], where detailed descriptions of the method can be found. For each pixel (x, y) in the stack of illuminated images, the surface radiosity $\mathbf{g}(x, y)$ is computed by solving the linear system:

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

or solving:

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

in case of applying the shadow trick, where $\mathbf{i}(x, y)$ is the vector of brightness of the images at pixel (x, y) for the stack of illuminated images, $\mathcal{V}$ is the vector of properties of the illuminations and the cameras and $\mathcal{I}(x, y)$ is the matrix with $\mathbf{i}(x, y)$ as its diagonal. According to the descriptions of `linsolve` and `mldivide`, the difference between the two functions is that `linsolve` applies QR-factorization for matrix A in $Ax = b$ with column pivoting for rectangular matrices, which helps solving the potential rank deficient matrices. This is exactly the case for estimating albedo and surface normal as both $\mathcal{V}$ and $\mathcal{I}\mathcal{V}$ are of rectangular shape and are potentially rank deficient. Hence, the function `linsolve` is used in the implementation. From the observations of the 5 distinctly illuminated images in *SphereGray5*, and according

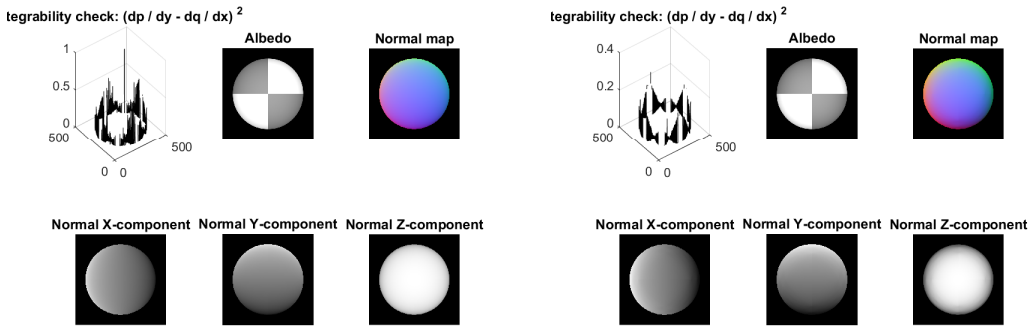


(a) **(b)**

Figure 1: *The general results of SphereGray5. The albedo and surface normal are estimated with (a) and without (b) applying the shadow trick.*

to the assumptions in [2], it is expected that the estimated albedo image should be uniform. As shown in Figure 1a, the albedo image meets the expectation to some extent, as the intensities on the circular object are mostly homogeneous, while towards the edge, the intensities decrease, which shows the limitations in the estimation.

2. As described in the paper [2], the minimum number of images necessary to estimate albedo and surface normal is 3. With more than 3 light sources or illuminated images, the estimated albedo and surface normal are expected to be more accurate. Directly comparing the results of



(a) **(b)**

Figure 2: *The general results of SphereGray25. The albedo and surface normal are estimated with (a) and without (b) applying the shadow trick.*

SphereGray5 and *SphereGray25* ensures that the differences can be (relatively) easily recognized. As shown in Figure 1a and Figure 2a, even though the estimated surface normals are very similar, the albedo images have apparent differences. For *SphereGray5*, because of the few light sources, the estimated albedo has obvious decrease towards the edge of the sphere, while we can notice that the intensities in the albedo image of *SphereGray25* are relatively more uniform and homogeneous.

3. Shadows create zero intensity for pixels in the illuminated images which make it impossible to compute the corresponding radiosity $\mathbf{g}$, so further the albedo and surface normal. The shadow trick is implemented by first forming the diagonal matrix $\mathcal{I}(x, y)$ using the values of $\mathbf{i}(x, y)$ and then multiplying both sides of Equation 1 with $\mathcal{I}$ as given in Equation 2. This helps to eliminate the contributions of points in the shadow as they have an intensity of 0.

   For *SphereGray5*, by comparing Figure 1b and Figure 1a, we can observe that without applying the shadow trick, though the estimated albedos are similar, the surface normals are clearly distinct. The obvious artifacts (potential edges) in the surface normals displayed in Figure 1b are due to the pixels in the shadow region. Similarly, for *SphereGray25*, as shown in Figure 2b and Figure 1a, there are edge-shaped artifacts on the surface normals, though the effect of pixels in shadow are less obvious compared to the case for *SphereGray5*. Hence, we can conclude that it is necessary to apply the shadow trick both in the case of 5 and 25 images to mitigate the effects of the pixels in shadow region and to have a better estimate in the surface normals.

2.2 Question - 2

1. As described in [1], the surface normal is defined as:

$$N(x,y) = \frac{1}{\sqrt{1 + \frac{\partial f^2}{\partial x} + \frac{\partial f^2}{\partial y}}} \left\{ -\frac{\partial f}{\partial x}, -\frac{\partial f}{\partial y}, 1 \right\}^T$$

   where the surface is $(x, y, f(x, y))$. By the denotation of the components of N as N_1, N_2 and N_3 respectively, we can find:

$$\frac{\partial f}{\partial x} = \frac{N_1}{N_3} = p$$

   and:

$$\frac{\partial f}{\partial y} = \frac{N_2}{N_3} = q$$

which is implemented in the code.

Furthermore, with the help of the built-in function `gradient` as described in [3], the second order derivatives $\frac{\partial p}{\partial y}$ and $\frac{\partial q}{\partial x}$ are implemented.

2. In order to determine the threshold value for the test of integrability, an experiment is conducted for candidate threshold values, as shown in Figure 3. According to the graph, we can observe that for *SphereGray5*, when increasing the threshold from 0.00005 to 0.5, there is a drastic decrease in the percentage of outliers at 0.0005, and then the decrease becomes gradual; for *SphereGray25*, when increasing the threshold from 0.00005 to 0.5, there is a relatively large decrease in the percentage of outliers at 0.005, and then the decrease becomes gradual. Therefore, taken into consideration both *SphereGray5* and *SphereGray25*, the threshold is set to be 0.005. According to Figure 1a and Figure 2a, we can see that for both *SphereGray5* and *SphereGray25*,

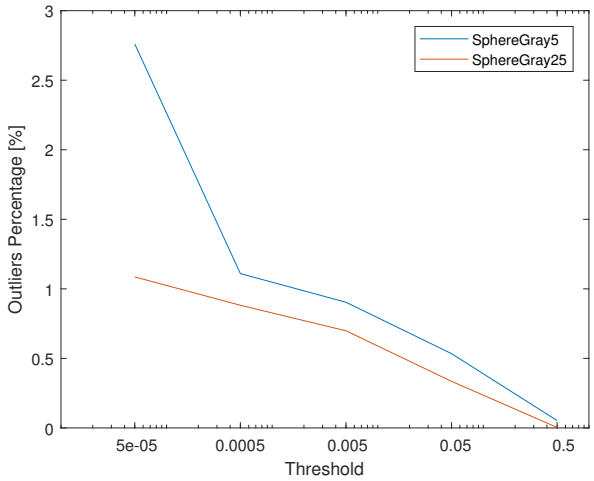


Figure 3: *Plot of outliers percentage (#outliers / #pixels) w.r.t. candidate threshold values.*

the errors occur on the edge of the sphere. Thus, it can be speculated that the errors are due to incorrect/inaccurate estimations of the surface normals on the edge between the sphere and the background. Also, when comparing *SphereGray5* and *SphereGray25*, the overall magnitude of the errors and the percentage of the errors are lower for *SphereGray25*. Hence, we can conclude that with more illuminated images, the accuracy of estimations of albedo and surface normals increases, and the test of integrability performs better.

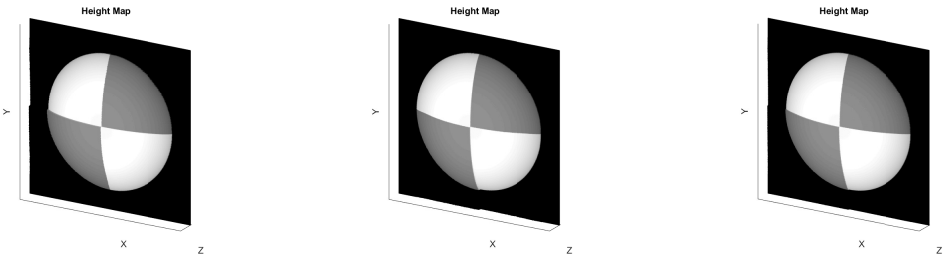
2.3 Question - 3

1. In order to reconstruct the surface height map, the partial derivatives of the surface $f(x, y)$ with respect to x and y directions need to be integrated along a selected path. According to the descriptions in [1], `construct_surface` is implemented with three different path integrations: column-major, row-major and average. In this case, as the surface normals are discretized for each pixel, the integration requires cumulatively adding the previous height values along the path with the partial derivatives p or q at that pixel.

   For column-major order, assuming the top left corner is zero, the leftmost column is first integrated from top to bottom with q. Then, based on the computed leftmost column, each row is integrated from left to right with p. As for row-major order, assuming the top left corner is zero, similarly, the top row is first integrated from left to right with p, and then each column is integrated from the top to bottom with q. The average path integration is implemented by applying both column-major and row-major path integrations and taking the average values for each pixel.

   When comparing the two different paths for *SphereGray5*, as shown in Figure 4a and Figure 4b, the artifacts are distinctly created. When using the column-major path, it can be observed that lower half of the sphere is higher than the upper half, and the artifacts are especially obvious in the middle row of the image. During integration, each row is integrated along the path from

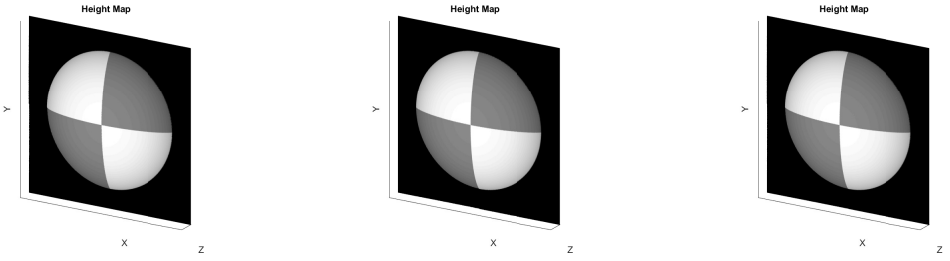
left to right, and as the errors in the estimated surface normals lie on the edge of the sphere, the errors are accumulated starting from the leftmost edges. Since only the partial derivatives of the surface w.r.t. the horizontal direction are integrated along the path, the partial derivatives w.r.t. the vertical direction are not taken into account, leading to discontinuities between proximate rows. Similarly, while using the row-major path, the right half of the sphere is higher than the left half, caused by the same reason stated above but only in the vertical direction. In this case, the discontinuities occur between proximate columns.



(a) **(b)** **(c)**

Figure 4: *The reconstructed height maps of SphereGray5 with column-major order (a), row-major order (b) and average (c).*

2. The result of taking the average is shown in Figure 4c. As can be observed from the graph, the artifacts of elevated half spheres in horizontal and vertical directions are mitigated and, hence, there is an apparent improvement in the result. Comparing the results for *SphereGray5* and *SphereGray25*, as displayed in Figure 4 and Figure 5, with an increased number of illuminated images, the construction results are greatly improved. The artifacts of the elevated half spheres are hardly observable for *SphereGray25* when applying either of the integration paths, and after taking the average of the two paths, the artifacts are further addressed.

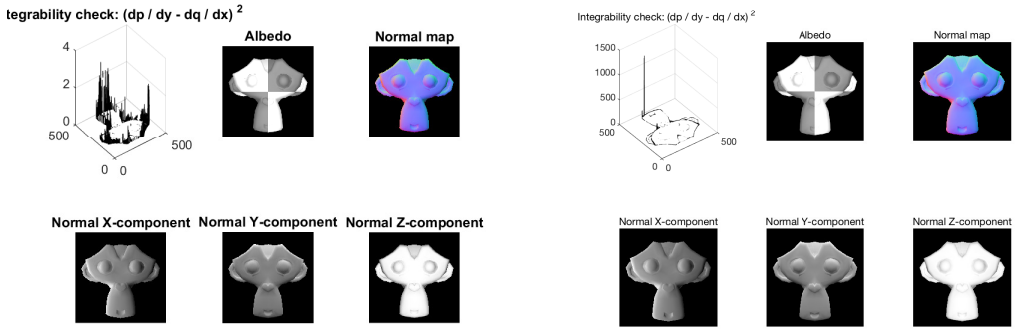


(a) **(b)** **(c)**

Figure 5: *The reconstructed height maps of SphereGray25 with column-major order (a), row-major order (b) and average (c).*

2.4 Question - 4

By comparison of Figure 2a and Figure 6a, we can see the albedo errors taken place in the reconstruction of monkey model are more severe than those of sphere model, as well as the integrability errors. With careful examination on Figure 6b, the albedo is not uniform in a particular area and showing some minor artifacts. The errors of albedo reconstruction in the monkey model mainly come from the irregular shape of the object. Because the margin of the object is very complex, it requires more different light sources to display all shadows and further determine the radiosity accurately. As shown in Figure 6a and Figure 6b, the solution to this problem might be to use images under more illumination directions to capture more kinds of shadow conditions, or to focus on using the light sources that are representative to the object shape in the current situation.

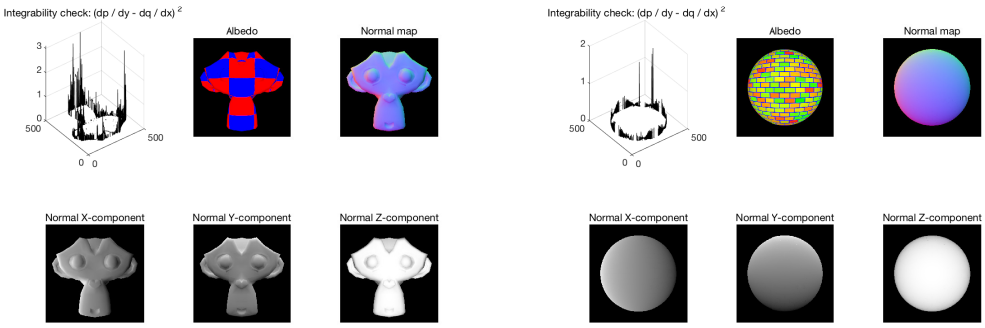


(a) **(b)**

Figure 6: *The results of MonkeyGray with all images (a) and MonkeyGray with 17 images (b). One can see that not only the albedo becomes more accurate by using more images, but the integrability error also drops drastically.*

2.5 Question - 5

The way we deal with RGB image is to view the three channel separately and compute the corresponding albedo and normal map respectively. For the albedo, we concatenate the three resulting albedo together in channel dimension as the final result. For the normal map, we first average the obtained three normal maps and then normalize each vector in the map to norm 1. The final results are presented in Figure 7a and Figure 7b. After we obtain the averaged and normalized normal map,



(a) **(b)**

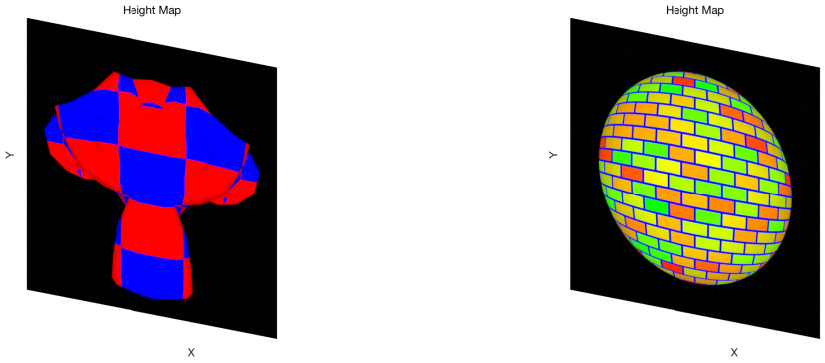
Figure 7: *The results of MonkeyColor (a) and SphereColor (b). The albedo and normal map are obtained from all channels of original images.*

we can reconstruct the height map from it, see Figure 8a and Figure 8b. We construct the height map using three different methods as mentioned in subsection 2.3. Here we adopt the column way, because the differences between these three methods are marginal in our given image model.

2.6 Question - 6

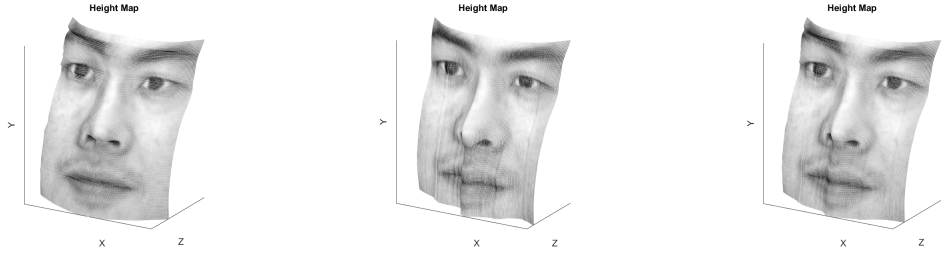
As displayed in Figure 9, as like the results in subsection 2.3, when integrating in the column-major order, discontinuities between proximate rows are raised and the heights of the lower face are inaccurately elevated. When integrating in the row-major order, the artifacts occur column-wise, which is obviously shown in Figure 9b. When taking the average, both artifacts are mitigated and the face is constructed in a relatively normal way. We find empirically there are four main factors that affect the quality of the height map generated:

- There are several images showing stripe-like artifacts that should be removed.
- The person in the images does minor movement across all angles, which is against the assumption of having a still object in shape-from-shading reconstruction.



(a) **(b)**

Figure 8: *The height maps reconstructed from the mean and normalized normal maps of MonkeyColor (a) and SphereColor (b).*



(a) **(b)** **(c)**

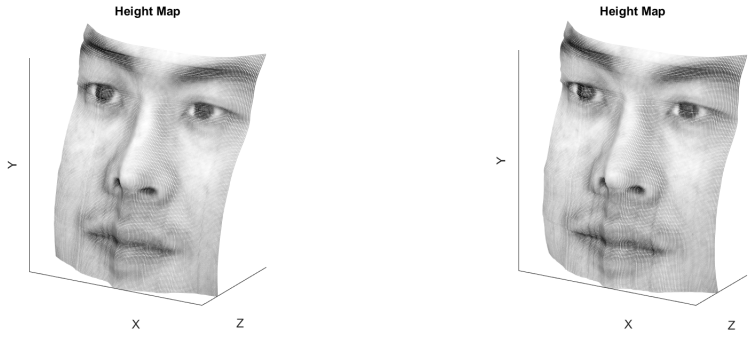
Figure 9: *The reconstructed height maps of YaleB05 with column-major order (a), row-major order (b) and average (c).*

- From the image with an azimuth of 95 degrees, which is purely dark, to the image of 85 degrees, the face illumination changes greatly. Also, though the image of 95 degrees is purely dark, the images taken from behind the people's head has illumination. It might be the reason that there are some other objects in the fields reflecting the light to the front face of the person. So it violates the assumption of a single infinitely distant and known light source.
- The nose, forehead and right cheek show some glaring reflections that should belong to specular reflections. It is again opposite to our assumptions that only Lambertian reflectence exists.

(a) **(b)** **(c)** **(d)** **(e)** **(f)**

Figure 10: *Samples of the filtered images: glaring cheek (a), glaring cheek and forehead (b), overly shadowed image with glaring nose (c), overly shadowed image (d), purely dark image (e) and image with striped artifacts (f).*

Some example images falls upon these three categories are shown in Figure 10. After removing these images by hand and redo the reconstruction of height map, we found out that the height map indeed shows better behavior, see Figure 11, where the human face is reconstructed to a better estimation of the natural face curves and detailed textures.



(a) **(b)**

Figure 11: *The reconstructed height maps of YaleB05 with average of two paths integration by unfiltered images (a) and filtered images (b).*

3 Color Spaces

3.1 RGB Color Model

1. The RGB color space is a linear color space, which means that each color can be considered as a linear combination of the three primary colors. RGB is very similar to the human visual system and it is very convenient to use in digital cameras. An LCD display can be thought of as a grid of millions of little red, green, and blue lamps.

2. The digital cameras use three separate sensors, each with a different filter. A beam splitter directs light to the different sensors. Each sensor gets an identical looking at the image, but because of the filters each sensor only responds to one of the primary colors.

3.2 Color Space Conversion

We use five different color spaces for image representations, which are namely Opponent color space (see Figure 12), Normalized RGB color space (see Figure 13), HSV color space (see Figure 14), YCbCr color space (see Figure 15) and Grayscale (see Figure 16), which will be elaborated in subsection 3.3.

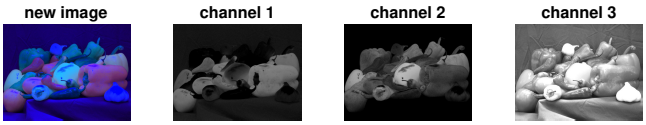


Figure 12: *Opponent color space.*

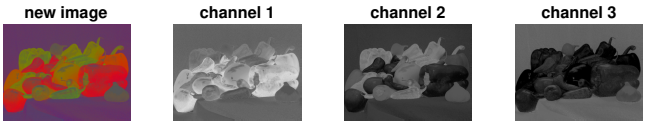


Figure 13: *Normalized RGB color space.*

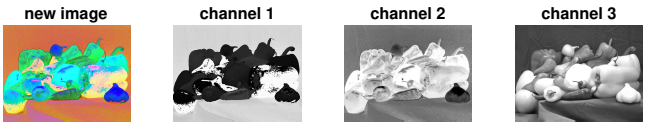


Figure 14: *HSV color space.*

3.3 Color Space Properties

1. Opponent Color Space: Opponent color decomposes color into three 'dimensions' of opposing properties: dark or light, yellow or blue, and red or green. One could speculate that the opponent color perception is more 'meaningful' in real-world environments. The dark/light

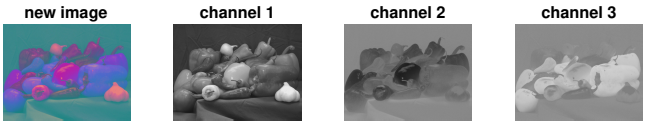


Figure 15: *YCbCr color space.*

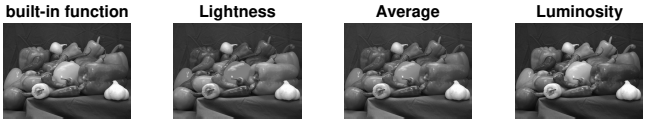


Figure 16: *Grayscale.*

axis represents the overall amount of light (see channel 3 in Figure 12), the yellow/blue axis indicates which end of the visible spectrum is predominant (channel 2), and the red/green axis provides a finer distinction for low-frequency light (channel 1). Edge detection, in particular color contrast perception, can be enhanced under opponent color space [4].

2. Normalized RGB Color space: Normalizing the RGB values can at times be a simple and effective way to help us get rid of distortions caused by lights and shadows in an image. The normalized image can be represented using only 2 bytes per pixel (as opposed to 3 bytes per pixel in RGB), and the loss of information can be regarded as 'removing' the lighting information from the image. As an example, we cannot clearly identify the garlic near the lower right corner of Figure 13, but we can easily distinguish the green and red peppers from the background in a way.

3. HSV Color Space: HSV is a cylindrical color model which remaps the RGB colors into dimensions that are easier for humans to interpret, because humans normally do not think about colors as mixtures of red, green, and blue lights. Similar to the Munsell Color System, the dimensions are hue (see channel 1 in Figure 14), saturation (channel 2) and value (channel 3).

4. YCbCr Color Space: YCbCr represents colors in terms of one luminance component/luma (Y) and two chrominance components/chroma (Cb and Cr). As the Y component (see channel 1 in Figure 15) is more sensitive to human eyes, it needs to be more accurate. On the other hand, Cb (channel 2) and Cr (channel 3) are less sensitive to human eyes, thus, they do not need to be that accurate. In some compression methods, it uses these sensitivities of the human eyes to eliminate the unnecessary details of the image.

5. Grayscale: Grayscale is the collection or the range of monochromic shades, ranging from pure white on the lightest end to pure black on the opposite end. Grayscale only contains luminance information and no color information. Most digital imaging software applications, even the most basic ones, are able to convert images to grayscale. This is also very important for printing, since it only consumes black ink, as opposed to printing in color which consumes all three print colors as well as black. In Figure 16, we use 4 methods to convert a color image to grayscale. The built-in function works best overall and it is based on Equation 3.

$$I = 0.2989 \times R + 0.5870 \times G + 0.1140 \times B \tag{3}$$

3.4 More on Color Spaces

1. Another color space can be the CMYK Color Space, which is a subtractive color model used for color printing. CMYK refers to the four inks used in some color printing: cyan, magenta, yellow, and key. White is the natural color of the paper or other backgrounds, while black results from a full combination of colored inks. To save cost on ink, and to produce deeper black tones, unsaturated and dark colors are produced by using black ink instead of the combination of cyan, magenta and yellow.

2. As the black color generated by mixing commercially practical colored inks is unsatisfactory and expensive, four-color printing uses black ink in addition.

4 Intrinsic Image Decomposition

4.1 Other Intrinsic Components

We can decompose the image to structure and texture, but the decomposition is not theoretically well-defined. The structures are the main skeleton and large object present in the image, and the texture is the local detailed pattern. Decomposing images like this enables us to manipulate the texture or structure of a given image while keeping the other component unchanged.

4.2 Synthetic Images

Because we have the ground truth albedo and shading for the synthetic images, we could compare the results with the ground truth accurately. Inverting the real image back to albedo and shading is an ill-posed problem in which there might be multiple solutions. On the other hand, by using the synthetic image we could restrict the solution to a plausible one implicitly.

4.3 Image Formation

The reconstruction image is displayed in Figure 17, which is obtained by multiplying the provided shading and albedo together.

Figure 17: *Illustrations of the reconstruction results.*

4.4 Recoloring

1. The true RGB color can be obtained by taking the maximum value of each color channel, as the color is uniform by hypothesis. The true RGB value is $[0.7216, 0.5529, 0.4235]$.
2. To recolor the image we first find the nonzero part of the albedo image, then replace all the nonzero pixel with $[0, 1, 0]$ (pure green). After the replacement, we multiply the resulting image with shading image element-wise. The final result is shown in Figure 18.

Figure 18: *Illustrations of the recoloring results.*

3. As the shading is not uniform, after multiplying the shading with the pure uniform albedo, the resulting image is not uniform either.

5 Color Constancy

5.1 Grey-World

1. The implementation of the function can be found in `AWB.m`. Briefly, the Grey-World Algorithm (GWA) assumes that given a white light source, the average color of a scene should be achromatic [5, 6] (i.e. grey). Here we take 50% of the maximum RGB value to represent the grey color (i.e. $[128, 128, 128]$). To correct an input image, we'll first separate it into its color channels and compute the mean of each channel. Then, we compute the ratios of the RGB value of grey to the mean of each channel, which are the coefficients of the Von Kries model. We use these coefficients to scale the image pixels by multiplying each of the three channels with the corresponding coefficient. As a result, the image becomes white-balanced (see Figure 19).

Figure 19: *Comparison of the original image and corrected image with the Grey-World Algorithm.*

2. An important assumption of the Grey-World Algorithm is that the image should have sufficient color variations [5], which thus means that it might fail when there are large blocks in the image with uniform color, or when the image lacks color variations [6]. As an example, we take a picture of the desert as an input image and feed in the Grey-World Algorithm. The original and corrected images are displayed in Figure 20. Since that the original image has large blocks of uniform color (i.e. 'yellow') and lacks color variations (e.g. 'blue'), obviously it does not meet the assumption of GWA that the average color of a scene should be achromatic. Therefore, the resulted image becomes 'bluish' which is unnatural.

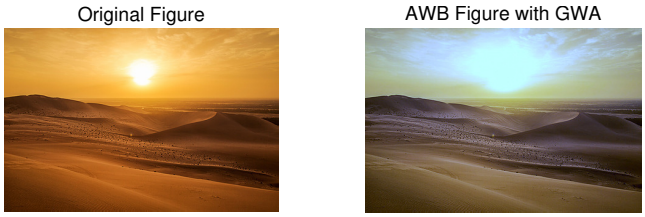


Figure 20: *Comparison of the original image of the desert and corrected image with the Grey-World Algorithm.*

3. Another color constancy algorithm we found in the literature is the Perfect Reflector Algorithm [6]. As specular surfaces can fully reflect the color of the illuminant, this algorithm assumes that the brightest pixel in a scene is from specular reflection and can be taken as the illuminant estimate. Similar to the case of the Grey-World Algorithm, the Perfect Reflector Algorithm also scales the image pixels by multiplying some coefficient of each color channel with the original channel values. Differently, here the coefficients are the ratios of the RGB value of pure white (i.e. $[255, 255, 255]$) to the maximum value of each channel. A limitation of this algorithm is that it might fail when the brightest pixel in a scene is not in white color.

6 Conclusion

In this assignment, we distribute the workload evenly during the code implementation and report writing. Specifically, in section 2, we make use of a series of pictures taken under different illuminates to reconstruct a patch of surface, and we discuss the differences with different reconstruction methods, different number of pictures and different models. In section 3, we discuss several color spaces and their properties. We experiment with intrinsic image components to perform material recoloring in section 4. In section 5, we implement a simple color constancy algorithm: Grey-World Algorithm, discuss its shortcomings and possible variations.

References

[1] D. Forsyth and J. Ponce, *Computer vision: a modern approach*. Pearson, nov 2012.

[2] R. J. Woodham, "Photometric Method For Determining Surface Orientation From Multiple Images," *Optical Engineering*, vol. 19, p. 191139, feb 1980.

[3] MathWorks, *MATLAB Primer R2018b*. MathWorks, 2018.

[4] L. M. Hurvich and D. Jameson, "An opponent-process theory of color vision.," *Psychological review*, vol. 64, no. 6p1, p. 384, 1957.

[5] K. Barnard, *Practical Colour Constancy*. PhD thesis, Simon Fraser University, Burnaby, BC, Canada, Canada, 1999.

[6] C.-C. Weng, H. Chen, and C.-S. Fuh, "A novel automatic white balance method for digital still cameras," in *2005 IEEE International Symposium on Circuits and Systems*, pp. 3801–3804 Vol. 4, IEEE, 2005.