Shantanu Chandra (12048461)
Rick Halm (10651098)
Phillip Lippe (12182451)
Mahsa Mojtahedi (12166510)

Practical assignment 1
Computer Vision 1, 2019

2019-03-07

1 Introduction

In this assignment we carry out experiments to estimate the albedo and surface normals of an object from a series of its images taken under different lighting conditions. We explore the effects of shadows and how it can be used for estimation of the image components. We carry out these experiments for objects of different shapes and using different number of its images for the task to study the effects. This helps us understand how the shape of an object, different lighting condition and our assumptions of the world can affect our estimates. We then familiarize ourselves with the different color spaces that can be used to represent an image. We explore the properties and advantages of each one of them (normalized-RGB, HSV, Opponent, YCbCr, grey scale). We carry out experiments to understand the image formation from its decomposed components (albedo and shading) and manipulate color of an image to finally reconstruct it. Lastly, we experiment with color correction of an image under the Grey world assumption (using the Von Kries model) to reduce the effects of noise caused by the light source in an image.

2 Photometric stereo

The following section evaluates the photometric stereo algorithm using different trials and experiments on artificial and real images.

2.1 Estimating Albedo and Surface Normal

Complete the code in `estimate-alb-norm.m` to estimate albedo and surface normal map for the `SphereGray5` folder. What do you expect to see in albedo image and how is it different with your result?
We expect that the resulting albedo image would be a perfect copy from [3] Figure 5.11. However, the resulting image as shown in Figure 1 still contains some shade, especially on the borders of the sphere to the black background. We experienced that these artifacts are given due to a higher amount of shaded examples and higher inaccuracy of the normal estimation at points with a lower z component.

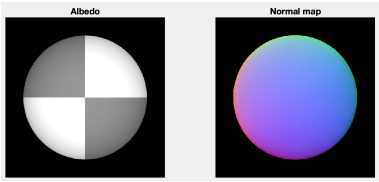
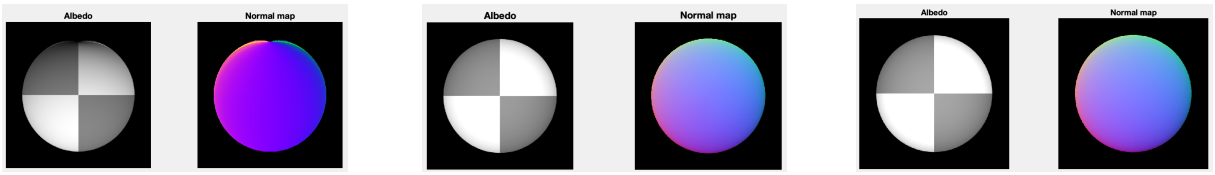


Figure 1: Albedo and surface normal estimation for the `Sphere5` model. The surface normals are encoded by using the red channel for the x component, green for y and blue for z respectively.

In principle, what is the minimum number of images you need to estimate albedo and surface normal?
In general, the minimum number of images you need to estimate the surface normal and albedo image is three. In this case, the linear system $\boldsymbol{i}(x, y) = \mathcal{V}\boldsymbol{g}(x, y)$ contains three variables (represented by $\boldsymbol{g}$) and three equations. Note that if we have only three images, the rank of the matrix $\mathcal{V}$ must be three. Otherwise, no unique solution can be found. This property can be ensured by observing the object with three orthogonal light source positions.
If we have more than three images, a least squares solution is appropriate for this over-identified linear system [3]. We expect that more images will lead to a higher accuracy of our albedo and surface normal estimation. Note that we still require a rank of 3 for the matrix $\mathcal{V}$.

Run the algorithm with more images by using `SphereGray25` *and observe the differences in the results. You could try all images at once or a few at the time, in an incremental fashion. Choose a strategy and justify it by discussing your results.*
As our hypothesis is that the quality of the estimations mainly rely on the rank of the matrix $\mathcal{V}$, we test different amount of images regarding the rank of $\mathcal{V}$. In Figure 2a, the estimation is based on 5 images and a rank of 2 for $\mathcal{V}$. As expected, the albedo and normal map contain many errors and do not capture the model well (e.g.

upper part of sphere). By increasing the rank of $\mathcal{V}$ to 3 (7 images), we significantly improve the estimation (see Figure 2b). This goes along with our hypothesis, as we now can accurately estimate the normals by the given equation. Adding even more images as visualized in Figure 2c shows to de-noise the image further which might be introduced by discretising the image (integer values between 0 and 255). The albedo contains less shade artifacts at the border.



(a) 5 images (rank 2) (b) 7 images (rank 3) (c) 25 images (rank 3)

Figure 2: Comparing albedo and surface normal estimation with different amount of images for the `SphereGray25` model. Images were sampled by taking every n-th image from the folder. For the examples, (a) every 5th image, (b) every 4th image and (c) every image was used. The rank of the matrix $\mathcal{V}$ is 2 for (a) and 3 (max) for the others.

What is the impact of shadows in photometric stereo? Explain the trick that is used in the text to deal with shadows.
The shadow trick simply zeros out any part of the image that lies in the shadow [3]. This requires however that for every pixel, the set of images contains at least one image without shadow. By multiplying by the diagonal matrix $\mathcal{I}$ and solving the linear system $\mathcal{I}i = \mathcal{I}\mathcal{V}g(x, y)$, relatively more weight is given to high intensity pixels compared to the linear system without shadow trick.

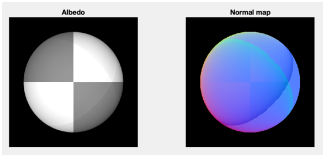
Remove that trick and check your results. Is the trick necessary in the case of 5 images, how about 25 images?
We experienced that we can see a great effect of the shadow trick in case of three images. If we take the first three images of the model `SphereGray5`, using the shadow trick results in a linear system with only two equations (as the equation in the shadow is dropped) with harsh normals as shown in Figure 3. With more than three images, the under-identification is less likely to occur and the problem fades. Therefore, in the case of 5 images, we retrieve a smooth surface normal map when using the shadow trick (see Figure 3c). In contrast, not using the shadow trick leads to some artifacts on the surface normals as visualized in Figure 3d. We can see a clear separation of the regions that were shaded at least once, and the part in the middle for which the intensity was always greater zero. Thus, the shadow trick helps to remove shadow artifacts.
For 25 images, the two results are not distinguishable anymore (see Figure 3e and 3f). However, for the `MonkeyGray` model, using the shadow trick has more benefits even on a set of more than 100 images as parts of the shadow come from the object itself as seen in Figure 6 (around the eyebrows). Using the equation $\boldsymbol{i}(x, y) = \mathcal{V}\boldsymbol{g}(x, y)$ results in insufficient estimations as we cannot properly deal with those problems. We will continue the discussion on this model in section 2.4.

2.2 Test of Integrability

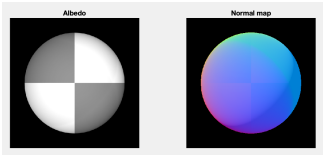
Implement and compute the second derivatives according to the algorithm and perform the test of integrability by choosing a reasonable threshold. What could be the reasons for the errors? How does the test perform with different number of images used in the reconstruction process in Question-1?
The estimations of the normals is especially inaccurate at the corners/boundaries of the object. The reason for that is the high gradient at these pixels. Small changes in the z-component of the normals can significantly influence the gradient at the corners as p and q are anti-proportional to z ($p, q \sim 1/z$), and z is close to 0. This results in high integrability errors at these pixels as depending on which normal is used in this region, we get different height estimates for pixels within the sphere.
Figure 4 summarizes the error distribution for different amount of images. We can see that more images lead to lower errors and therefore more accurate gradients. Note that also the number of outliers are reduced (2335 for `Sphere5`, 1831 for `Sphere25` with a threshold of 0.005). More complex examples like the monkey result in higher integrability errors than the sphere as there are more sharp edges.

2.3 Shape by Integration

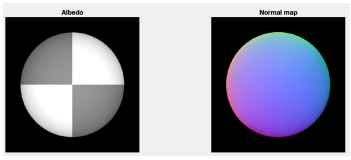
Construct the surface height map using column-major order as described in the algorithm, then implement row-major path integration. What are the differences in the results of the two paths?



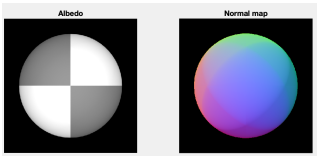
(a) Three image estimation with shadow trick



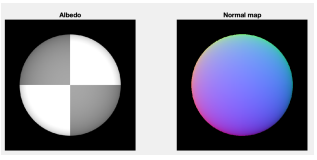
(b) Three image estimation without shadow trick



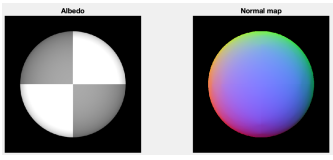
(c) Five image estimation with shadow trick



(d) Five image estimation without shadow trick

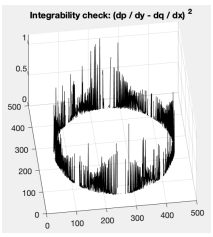


(e) 25 image estimation with shadow trick

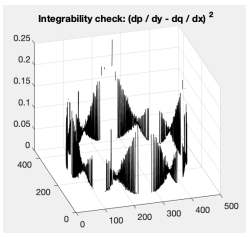


(f) 25 image estimation without shadow trick

Figure 3: Estimations on the (a) - (d) `SphereGray5/` (e) - (f) `SphereGray25` model by varying the usage of the shadow trick and number of images. For each subfigure: albedo on the left, surface normals on the right.



(a) Sphere 5 images

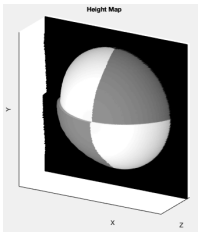


(b) Sphere 25 images

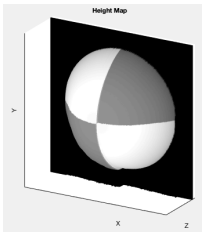
Figure 4: Comparing integrability errors over number of images for the sphere (threshold set to 0.005). In general, the error decreases with the number of images. Note the different scales for the z dimension.

Starting in the right upper corner (based on visualization in Figure 5), the algorithm sums the partial derivatives which it encounters along the path. Whenever a large error is encountered, this error persists over the next values. Hence, column major would result in horizontal errors whilst the row major results in vertical errors as can be seen in Figure 5a and 5b respectively. The errors are especially large in the corner points as shown in the test of integrability.

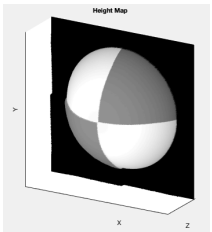
Now, take the average of the results. Do you see any improvement compared to when using only one path? Are the construction results different with different number of images being used?
Using the average of column and row major smooths the results and halves the errors of the column and row major. However, averaging only over these two paths for 5 images does not reduce the noise significantly, so that the mistakes can be still easily spotted in Figure 5c. The errors can be reduced by using more images. Figure 5d shows the averaging over column and row major for the model `SphereGray25`. Here, almost no errors can be seen at the borders.



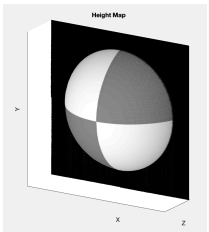
(a) Column major



(b) Row major



(c) Average `SphereGray5`



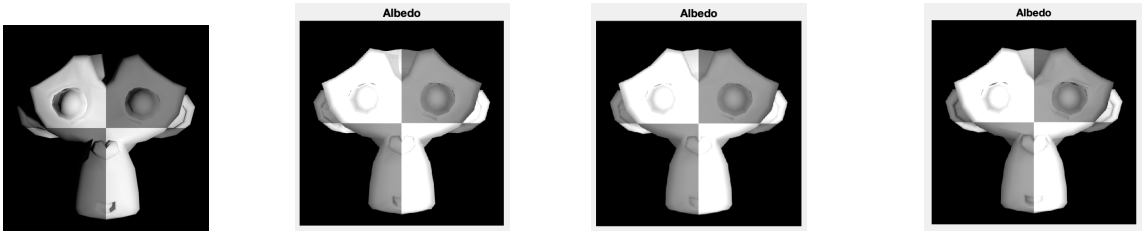
(d) Average `SphereGray25`

Figure 5: Visualization of the surface height map with different path integrations for the `SphereGray5` and `SphereGray25` model.

2.4 Experiments with different objects

Run the algorithm and show the results for the `MonkeyGray` model. The albedo results of the monkey may comprise more albedo errors than in case of the sphere. Observe and describe the errors. What could be the reason for those errors?

The `MonkeyGray` model shows an object with a more complex shape than the `Sphere` model. Due to different height levels of the object, some parts can shade others as visualized in Figure 6a. Thus, we partially get shadows that are independent of the angle between surface normal and light source. When performing the estimation of the surface normal, we have to neglect these shadows.



(a) Example image (b) Albedo 7 images (c) Albedo 121 images (d) Albedo without shadow trick

Figure 6: (a) Example of the monkey and the shadows as mentioned above (see forehead or left ear). Light source shines from the slightly bottom, right position. (b) - (c) Albedo estimation for different number of input images while applying the shadow trick. (d) Albedo estimation for all images without using the shadow trick.

One aspect that can help is using more images. When comparing the albedo estimation with only seven images (see Figure 6b) and the estimation based on all 121 images (see Figure 6c, we see that the albedo with less images includes more shadows. Especially the left side of the forehead includes a shadow part which is removed in the estimation which is based on all images. An even greater difference can be observed when not applying the shadow trick (see Figure 6d). The shadows of the ear, nose and mouth are clearly visible in the albedo, which emphasizes the importance of the shadow trick for the `MonkeyGray` model.
However, there are same shadows like in the region of the eye and ears in Figure 6a which are not totally black. Thus, the shadow trick does also not mask out those but may just weight them less when performing the least squared error approximation. This is why the albedo image partially contains shadows as well although we use the shadow trick. Still, we can clearly see an improvement compared to the estimation without the shadow trick in Figure 6d.

Update the implementation to work for 3-channel RGB inputs and test it with 2 models `SphereColor` and `MonkeyColor`. Explain your changes and show your results. Observe the problem in the constructed surface normal map and height map, explain why a zero pixel could be a problem and propose a way to overcome that.

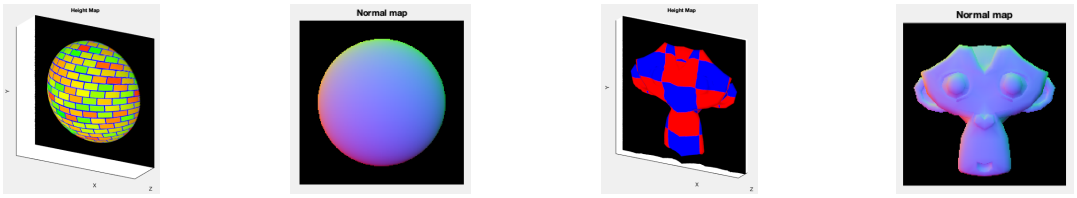
In order to include the RGB channels, the normals $\boldsymbol{g}(x, y)$ are calculated for each channel separately (denoted by $\boldsymbol{g}_R$, $\boldsymbol{g}_G$ and $\boldsymbol{g}_B$). The albedo image is an RGB image consisting of the length of normals $\boldsymbol{g}$ (representing intensity) so that a pixel $a(x, y)$ in the albedo image is determined by $a(x, y) = (|\boldsymbol{g}_R(x, y)|, |\boldsymbol{g}_G(x, y)|, |\boldsymbol{g}_B(x, y)|)$. For the surface normals, we sum up all normals of the different channels and divide it by the norm to get a unit vector. Formally, we can write our estimation by:

$$\boldsymbol{N}(x, y) = \frac{\boldsymbol{g}(x, y)}{|\boldsymbol{g}(x, y)|} \quad \text{where} \quad \boldsymbol{g}(x, y) = \boldsymbol{g}_R(x, y) + \boldsymbol{g}_G(x, y) + \boldsymbol{g}_B(x, y)$$

Note that this combination can be seen as an average over the surface normals that are given by the different channels. The surface normal based on the red channel is $\frac{\boldsymbol{g}_R(x,y)}{|\boldsymbol{g}_R(x,y)|}$, and similar for blue and green. In order to get the equation above, we weight them by their intensity $|\boldsymbol{g}_R(x, y)|$ as seen in the experiments before (and the idea of the shadow trick), that we should focus on examples with a high intensity.
A problem occurs when one of the channels is missing, as is the case in the monkey picture. This is solved by setting all values of the missing channel to zero. As the normals are calculated with the weighted average over intensity, this zero matrix does not influence the results. Results are shown in Figure 7.

Run the algorithm for the Yale Face dataset. Observe and discuss the results for different integration paths. Discuss how the images violate the assumptions of the shape-from-shading methods. Remember to include specific input images to illustrate your points. How the results would improve when the problematic images are all removed? Show the results in your report.

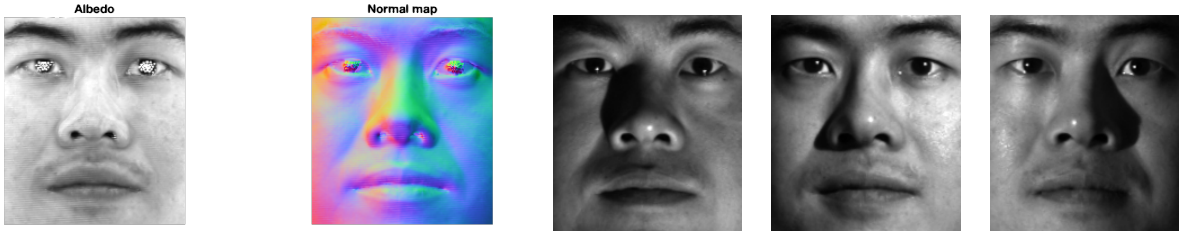
The Yale Face dataset violates the assumption of shape-from-shading methods by not adhering to the Lambertian reflectance law. Here it is assumed that the surface reflects light equally in all directions. However,



(a) Sphere height map (b) Sphere surface normals (c) Monkey height map (d) Monkey surface normals

Figure 7: RGB height and surface normals of the models `SphereColor` and `MonkeyColor`

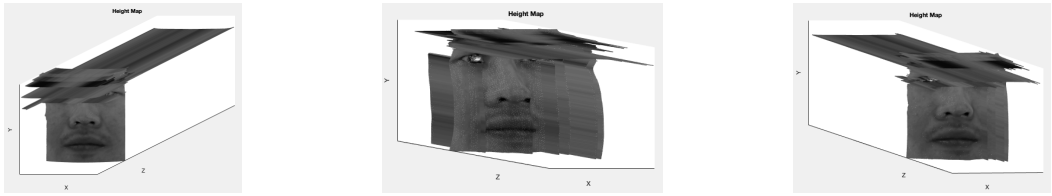
the example input images in Figure 8 (b) - (d) show that e.g. the eyes differ in their reflectance (position of white spot) depending on the light source. For example, in Figure 8b, the light source shines from below on the face so that the white spot in the eyes can also be found in the lower part. The albedo image contains values greater than one for these pixels as in most other images, the corresponding pixels are black and are therefore neglected by applying the shadow trick. Furthermore, the same effect can be observed on the nose, the cheek and most other parts of the skin. Overall, these effects lead to a very noisy albedo and surface normal estimation for the face as visualized in Figure 8a.



(a) Estimated alebdo and surface normals (b) Input image (c) Input image (d) Input image

Figure 8: (a) The albedo (left) and normal's image (right) of a face. It can clearly be seen that the normal vectors in the region of the eyes differ a lot as the eyes are highly reflective. Also the nose is brighter than one would expect due to the non-Lambertian reflectance. (b) - (d) Example input images of different light source directions showing the non-Lambertian reflectance in the eyes and on the nose.

Again, comparing the different integration paths in Figure 9 the differences occur at the points where the errors are made. Especially around the eyes, where there is a lot of specularity, the errors distort the height map. Even the averaging method is not able to smooth the height map due to the huge errors as shown in Figure 9c. In addition, the face is not a completely steady model and slightly changes its positions over images which further introduces noise.



(a) Column-major (b) Row major (c) Average

Figure 9: Surface height map with different path integrations for the Yale Face dataset.

3 Color spaces

3.1 RGB Color Model

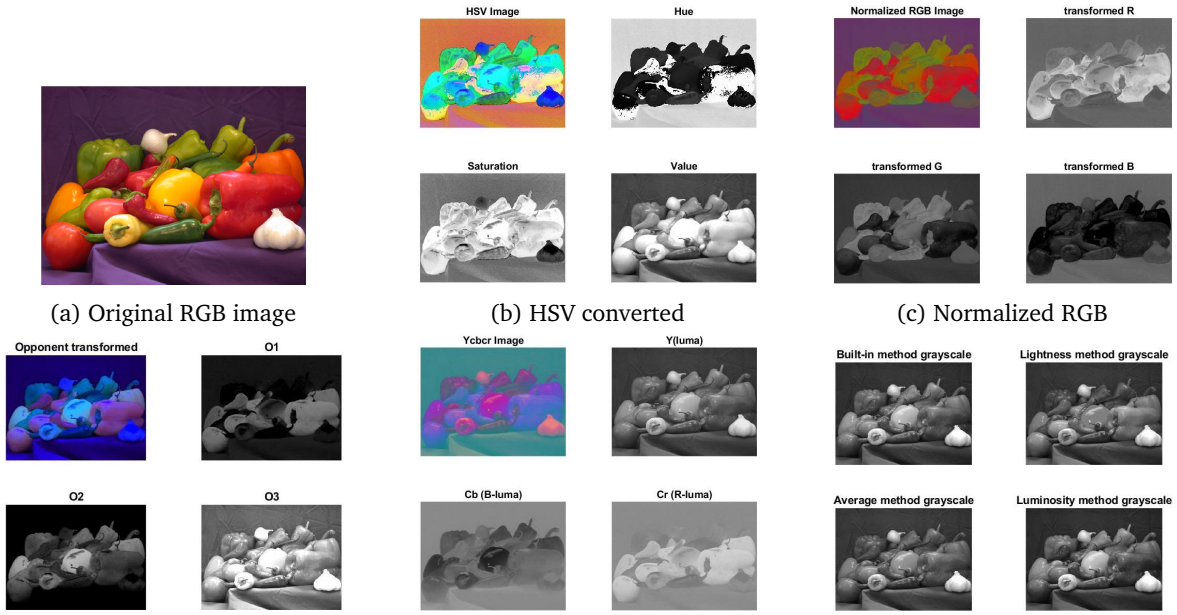
Why do we use RGB color model as a basis of our digital cameras and photography? How does a standard digital camera capture the full RGB color image?

Digital Cameras, as opposed to film cameras, store images as numerical data. Color models provide us with an efficient way of representing colors with a few numbers. RGB color model uses the three primary colors of

light to describe the color of each pixel. It should be noted that red, green and blue are chosen because they are the primary 'additive' colors; and since light is additive, any color can be recreated by combining lights of these colors. RGB is a convenient model to use in a digital camera, since it's different components can be derived by applying filters in front of the lens that only let one color pass through. Finding the components of other color models, such as HSV, is not as easily done with only a filter. That is why although RGB model is not as intuitive for humans, it is predominantly used to represent images in digital cameras and photography. Digital cameras have many sensors that are capable of capturing light, and the final picture is crated by putting the results of these sensors together. In order to capture color, they apply filters that only lets light of a certain color through, either red, green, or blue. In order to get the correct color of each pixel, they can take 3 successive pictures and apply a different filter each time. But this is not practical. Another more common approach is to divide the sensors into a color grid, so each part of the grid has only sensors that can capture one color. The colors in the grid are then distributed in a way that the colors of the scene are best captured.

3.2 Color Space Conversion

We have an image in RGB space, and this image is mapped to five other color spaces: normalized RGB, opponent space, HSV, YCbCr, and gray-scale. Different channels of these spaces as well as the final image after combining these channels are shown in Figure 10.
It should be noted that after the conversion, the meaning of the numbers in each channel has changed. However, if we depict the original image resulting from them, we will get the same picture. In order to better show the difference between these color spaces we let MATLAB assume that the values given to it are still from RGB space. Now, for example, in the HSV space, blue represents Hue, and green saturation; therefore, the places with a higher color saturation look greener. In the normalized RGB, we can see that all the colors have become darker, that is because in normalization, the intensity of every pixel is set to one. Also, since we only have one value for each channel in the new space, MATLAB depicts the separated channels as gray-scale.



(a) Original RGB image (b) HSV converted (c) Normalized RGB

(d) Opponent converted (e) YCBCR converted (f) Grayscale converted

Figure 10: Color Space conversions to HSV, Opponent, YBCBR, Grayscale and Normalized-RGB

3.3 Color Space Properties

Explain each of those 5 color spaces and their properties. What are the benefits of using a different color space other than RGB? Provide reasons for each of the above cases. You can include your observations from the visualizations.

In order to represent images as numerical data for storing or analyzing, we need to have a model to represent colors with numbers. Representing each color with a separate number is not feasible, therefore, color models are created. These models use certain features of colors to represent them in a color space. These color models defer in the features they use for representation. Here, we have examined 5 color spaces, which are explained bellow.

- **Normalized RGB color space**: This space is simply made by normalizing each of the RGB channels by the sum of all channels at that pixel, or intensity. As a result, the sum of all normalized RGB (rgb) channels will be one at each pixel, and therefore, we only need to store two values. Before normalizing, the RGB values would have a wide range, so the values for different shades of a color would significantly differ. By normalizing, this effect is mitigated. In other words, normalizing with respect to intensity will make the picture invariant with respect to it. It is useful in sharpening the edges and makes discerning between objects easier.

- **Opponent color space:** While the RGB model focuses on the three additive primary colors as color features, this space is represented by different axis. The opponent model is based on the opponent process theory of colors, which states that human color perception uses two chromatic axis: red-green and blue-yellow[2]. These two, after normalization, represent the first and second channels of the opponent space. The third channel is the luminance component. This model, as opposed to RGB, does not focus on primary additive colors, and chooses features that are more intuitive for humans.

- **HSV color space:** This space consists of three features. The first one, hue, determines different colors, and is represented by an angle on a color wheel. The second component, saturation, is the amount of gray in the picture. Reducing saturation will fade the image to gray. The last feature is color brightness. As was the case with the opponent color model, HSV represent colors in a more intuitive way. Adjusting colors is therefore easier with HSV, since we have a sense of what its values mean.

- **YCbCr color space:** This space also consists of three features: Y, C_b and C_r. Y determines the luminacity. The other two channels are the blue difference and the red difference. These are the chroma components. This space is mostly used in tasks containing storing and transmission, such as video streaming; since the last two channels can be compressed.

- **Gray-scale space:** In this space, contrary to other spaces, the value of a pixel is only represented by one number. The only information this space conveys is the intensity of light at each pixel. Therefore, only the luminance information remain and not the chromatic information. However, since luminance is more important for the human eye in distinguishing shapes, they would still be clearly discernable. This helps to reduce the amount of necessary information significantly. Also, by simplifying the picture, it makes edge detection easier.

3.4 More on Color Spaces

Find one more color space from the literature and simply explain its properties and give a use case.

Another color space worth mentioning is the CMYK space. It stands for Cyan, Magenta, Yellow, and Key (black). These are the four ink colors used in printing. In a sense, it is using subtractive primary colors plus black to represent an image. This representation is better for printers that represent an image by mixing ink colors (which are subtractive), and not by light (which is additive).

4 Intrinsic Image Decomposition

4.1 Other Intrinsic Components

What other components can an image be decomposed other than albedo and shading? Give an example and explain your reasoning.

One another important component of image decomposition is ***Specularity***. Reflectance (albedo) captures the uniform (i.e., independent of the direction of incident light) reflection from the surface (rough or smooth) and is attributed to the object's intrinsic properties (i.e., the color) irrespective of its shape or relative position of the light source from it. Shading (illumination) captures the interaction of the light with the object due to the relative position of the light source and varies with the direction/shape of the surface of the object. Specularity not only captures the information of the position of the light source but also the position of the viewer. It is observed as parts of the incident white light hitting the surface and getting reflected (at the same angle to the surface normal as the incident light) to directly reach the viewer. Depending on the angle of incidence and shape of the surface, specularity can cause different parts of the object to appear white to the viewer. This information of relative position of the light source can be used to model 3D renditions of objects from their decomposed components for various angles. This is helpful in applications like designing of computer graphics for games or animations with real world models.

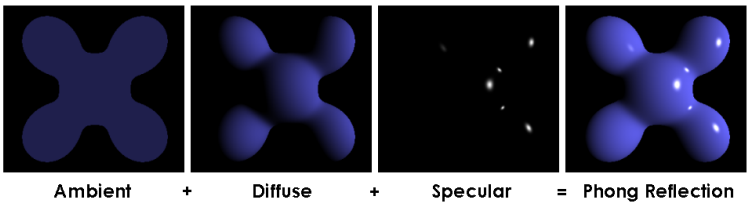


Figure 11: The Phong reflection model. Here, *Ambient* = Albedo, *Diffuse* = Shading and *Specular* = Specularity [5]. We see spots of white on the object surface when the viewing angle is the same as the reflected angle of the incident white light. Here, the direction of incident light is from the right while the point of observation is from infront/top of the object. This phenomenon is Specularity.

4.2 Synthetic Images

If you check the literature, you will see that almost all the intrinsic image decomposition datasets are composed of synthetic images. What might be the reason for that?

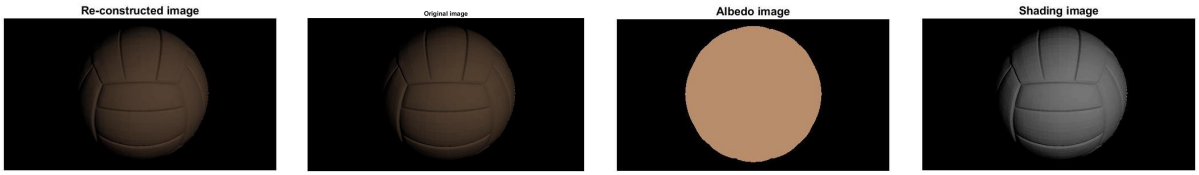
The task of capturing the true and unbiased intrinsic components of an image is very hard in the real world. Factors like nature of the light source (presence of many mixed light sources with varied energy distributions), presence of objects near by (reflection of light from these objects causes inaccurate estimations/readings of the object under observation) and properties of the viewing source (the filters used can cause difference in readings based on the dye/material used to make them, the color functions of the viewer, etc) usually significantly alter the true readings of the decomposed components. Hence, the intrinsic image decomposition datasets are usually synthetic that are carefully created in controlled environments with no sources of noise where all the factors are known and measured accurately.

4.3 Image Formation

To reconstruct the image, we use the decomposed image components of the original image, namely albedo $R(x)$ and shading $S(x)$. We reconstruct the new image by :

$$I(x) = R(x) * S(x)$$

where, $*$ is element-wise multiplication and R is the original image albedo with RGB channels, and S is the gray-scale shadow information. Using this technique we reconstruct the *ball.png* image in Figure 12 below.



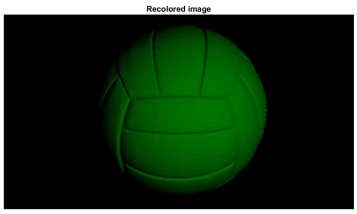
(a) HSV converted (b) Normalized RGB (c) Opponent converted (d) YCBCR converted

Figure 12: The re-constructed image of *Ball.png* using the decomposed intrinsic components, albedo and shading.

4.4 Recoloring

Find out the true material color of the ball in RGB space (which is uniform in this case).
The true material color of the ball in the RGB space is $R = 0.7216 \quad G = 0.5529 \quad B = 0.4235$.

Recolor the ball image with pure green (0,255,0). Display the original ball image and the recolored version on the same figure. Name your script as recoloring.m.
The recolored *ball.png* image is shown in Figure 13. To reconstruct the image with new color, we use the decomposed image components of the original image and the technique described previously.
Although you have recolored the object with pure green, the reconstructed images do not seem to display those pure colors and thus the color distributions over the object do not appear uniform. Explain the reason.



(a) Recolored and Reconstructed

(b) Original

Figure 13: The re-constructed image of *Ball.png* using the decomposed intrinsic components, albedo and shading.

Since the recolored image is not just a function of the albedo (pure green color) of the object but also the original shading component (the appearance of object due to its shape and presence of a light source at a relative position), it does not have a uniform color distribution as one would expect. On the contrary, the shading patters lead to non-uniform (and more realistic) color patterns in the recolored-reconstructed image.

5 Color Constancy

Create a function to apply color correction to an RGB image by using Grey-World algorithm. Display the original image and the color corrected one on the same figure. Name your script as AWB.m. Use awb.jpg image to test your algorithm.

The greyworld assumption works under the premise that the average of all the pixels in an image would be that of grey color (i.e., $R_{avg} = G_{avg} = B_{avg} = 128$). Hence, for color correction of an image (assumed to be diverse in colors), we need to linearly transform its RGB matrix such that the average over all pixels for each color channel $= 128$. We achieve this by the following transformation:

$$Ch_{corrected} = \frac{Ch_{old}}{Ch_{old_{mean}}} * 128$$

where $Ch_{corrected}$ is the new color corrected values of the channel (here, RGB), Ch_{old} are the channel values in the original image and $Ch_{old_{mean}}$ is the mean of channel values of the original image. We observe in Fig 14 that the new color corrected image looks more natural without the reddish tinge that was present in the old image.

(a) AWB original

(b) Corrected

Figure 14: The corrected image of `AWB.png` using the Grayworld algorithm. We see that the corrected image has reduced red tinge.

Give an example case for Grey-World Algorithm on where it might fail. Remember to include your reasoning.

The gray world algorithm is based on the assumption that the average colors in any scene with sufficient color diversity would be 'gray'; and any deviation is caused by the light source. This assumption does not hold where there actually is a predominating color in the scene. As an extreme example, if we apply the gray-world algorithm on a red plain, we will get a gray plain in return. This can also happen when a picture is naturally biased towards a color, and then the gray-world algorithm will over compensate for that color.

Find out one more color constancy algorithms from the literature and explain it briefly.

Another color constancy algorithm is the White Patch method. In this algorithm we assume that there is at least a pixel that is white in the scene. If there is no white pixel in the scene, then the difference is caused by the light source, and the picture can be corrected by changing all values so that this pixel is white again. The said pixel would have the maximum RGB values in our image. After locating this pixel, the RGB values of each pixel in the image is corrected by dividing them by these maximum values, thereby correcting the effect of the light source.

OPTION 2: Rosenberg et al.[4] presented a Bayesian color constancy algorithm using non-Gaussian models (in contrast to the Gaussian model earlier proposed by Brainard et al.[1]). Here, they replaced the independent and Gaussian distribution of reflectance with the exchangeable reflectance distribution defined by a Dirichlet-multinomial model. They modelled the surface reflectances of the scene and illuminations as random variables separately and retrieved the illumination color using Bayes rule. However, they formulated their theory on assumptions like one light source per image, approximately uniform brightness over the entire image and Lambertian image model so that the diagonal lightning model holds.
It is interesting to note that the greyworld algorithms are a sublclass of the Bayesian model corresponding to simple reflectance distributions which is channel independent.

$$p(x) = \Pi_{c=r,g,b} p(x_c)$$

6 Conclusion

In the first part of this assignment, we have demonstrated that with the help of multiple images containing different light source positions, we are able to determine the shape and the albedo of a 3D object. The estimations get more accurate by increasing the number of images. Furthermore, we experienced that shaded areas can be challenging, especially if the estimation is performed without applying the shadow trick to mask those examples. However, the whole process underlies the assumption of having a pure Lambertian world where the light intensity only depends on the angle between light source and surface normal. Thus, real images containing specularity might result in much worse estimations as seen for the Yale face dataset.
Different color spaces and their attributes were explored by converting an image into different spaces. By decomposing each space into its channels, we can have a better idea of how each representation works. This is useful in choosing the best representation for an application, for example, HSV is better than RGB when a more intuitive representation is needed.
Images also have intrinsic properties, such as albedo and shading. Since each component contains information about a separate attribute of the image, certain changes or corrections on these attributes, such as color, can be done after decomposing the image. After applying the desired changes, the image will be reconstructed by combining the decomposed intrinsic components.
An important correction that is usually done on images is correction for the light source color. The human brain can apply color constancy correction on the image it receives. In order to apply this correction on digital images, we need to have an assumption about the original color distribution, so we can omit the effect of the light source. One such assumption is gray-world assumption. It corrects the image by setting the average color of the scene to gray, and is applicable as long as the image does not contain a predominant color.
All members contributes equally.

References

[1] BRAINARD, D. H., AND FREEMAN, W. T. Bayesian color constancy. *JOSA A 14*, 7 (1997), 1393–1411.

[2] BRATKOVA, M., BOULOS, S., AND SHIRLEY, P. orgb: A practical opponent color space for computer graphics. *IEEE computer graphics and applications 29* (03 2009), 42–55.

[3] FORSYTH, D. A., AND PONCE, J. *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference, 2002.

[4] 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.

[5] WIKIPEDIA, THE FREE ENCYCLOPEDIA. Phong reflection model, 2006.