Rico Mossinkoff - 12805157 *Hannah Lim - 10588973*
Ewoud Vermeij - 11348860 *Yke Rusticus - 11306386*

Computer Vision - Assignment 1

Introduction

For this assignment we will focus on several fields in Computer Vision.

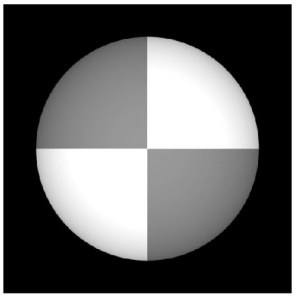
First, we will answer some questions about photometric stereo. This is a technique to estimate the depth and surface of an object by looking at it under different light conditions. By calculating the albedo and the normals, we can construct the surface. Second, we will look at 5 different color spaces and discuss the pros and cons of each space. In Matlab we will convert images to different color spaces partly by implementing our own code and partly by using built-in function from Matlab. Furthermore, we will examine intrinsic image decomposition. This is a technique to recover the formation components of an image. We will use the components albedo and shading of the image to reconstruct the image. Finally, we will focus on color constancy, which is the ability to recognize an objects real color, even when the light source has a different color. We will do this with the Grey-World algorithm. All the results and discussions will be included in this report and in the appendix the code in Matlab can be found. Throughout this assignment we expect that the questions will give us more insight in photometric stereo, possibilities with color spaces, consistency and decomposition of images.

1. Photometric Stereo

1.1 Estimating Albedo and Surface Normal

1.

See code estimate_alb_norm.m.



In the albedo image we expect to see the "true" color (in this case grayscale) of the image, without shadows or shading present. However, the presented albedo image shows a slight shade near the edges of the object, which might be due to an insufficient amount of input images.

2.

As the surface normal has 3 dimensions and albedo 1, there have to be at least 3 images for the surface normal and 1 for albedo. However, this would contain a lot of noise, so more images are recommended.

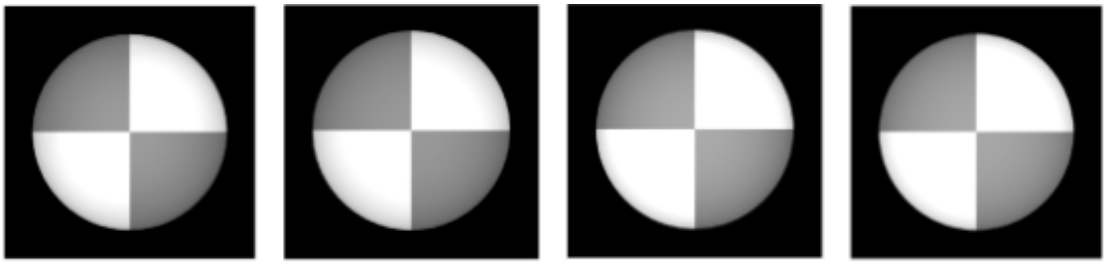
We ran the algorithm for a subset of n images (the first n images in the folder), where the presented images show from left to right the albedo images for n = 5, n = 15 and n = 25. The images show an improvement in the albedo images for higher n. The reason for this is that the subset of n = 5 images does not include images where the right-hand-side of the object is well illuminated. For n = 15 this problem reduces, and for n = 25 the set includes enough images where the object is well illuminated from each side. As the derivation of the normals depends on the albedo, the normals are also more accurate for larger n. A strategy would be to first select the images in which the entire (or most of the) object is illuminated, in order to use as few images as possible and still get a reliable result. One could then add images for more accuracy.

3.

See code estimate_alb_norm.m.

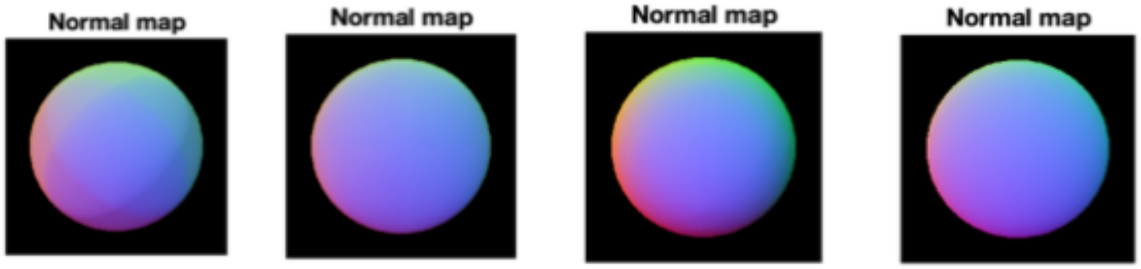
Multiple shapes on an object could cast their own shadow. This makes it harder to identify reflections from the surface of the object. If there really is no ambient illumination, then we can form a matrix from the image vector and multiply both sides by this matrix; this zeroes out any equations from points that are in shadow. [p. 79, Computer Vision: a Modern Approach]

Albedo images



5 images, without shadow trick | *5 images, with shadow trick* | *25 images, without shadow trick* | *25 images, with shadow trick*

Normal maps



5 images, without shadow trick | *5 images, with shadow trick* | *25 images, without shadow trick* | *25 images, with shadow trick*

In the case of the sphere, the own shadow is only dependent on the direction of the light source. Since there are multiple images with different direction of the light source, the full

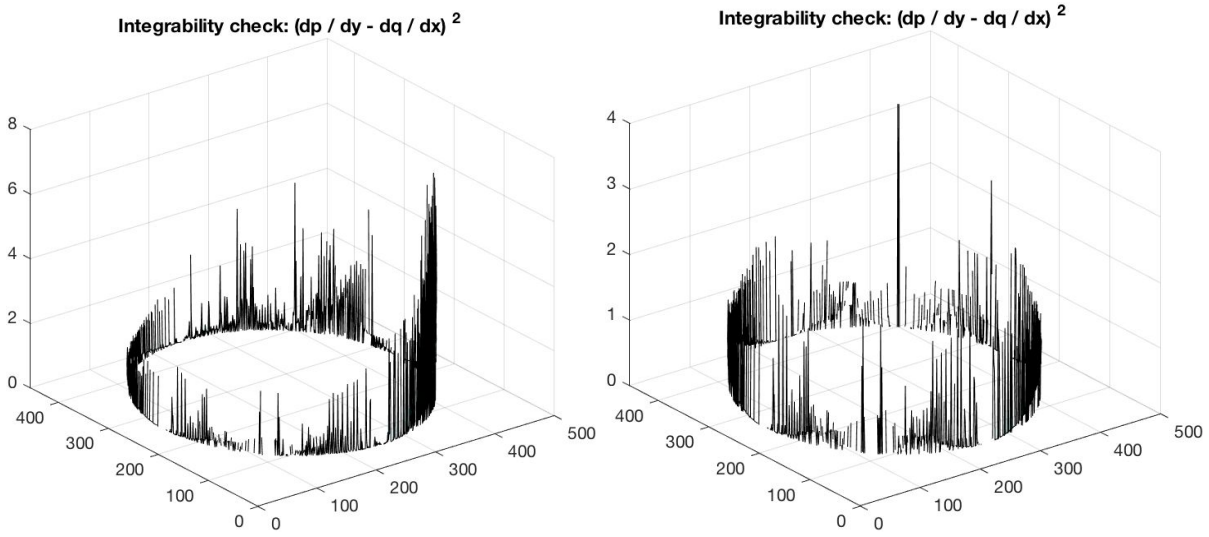
surface of the sphere can be perceived and therefore the shadow trick does not make much difference as can be seen from the presented images.

However, a small difference can be noticed in the normal maps. For n = 5, the result where the shadow trick is used shows a more uniform color distribution than if the shadow trick is not used. For n=5, there are not enough images where certain parts of the sphere (mainly around the edge of the sphere) have sufficient illumination. In the normal maps with 5 images, without the shadow trick you can see the influence of the shadow. The shadow also influences the albedo, but this is not noticeable for our eyes but in the normal maps this is visible. For n = 25, there are enough images, therefore the influence of the shadow trick is almost unnoticeable.

1.2 Test of Integrability

1. See code check_integrability.m.

2.

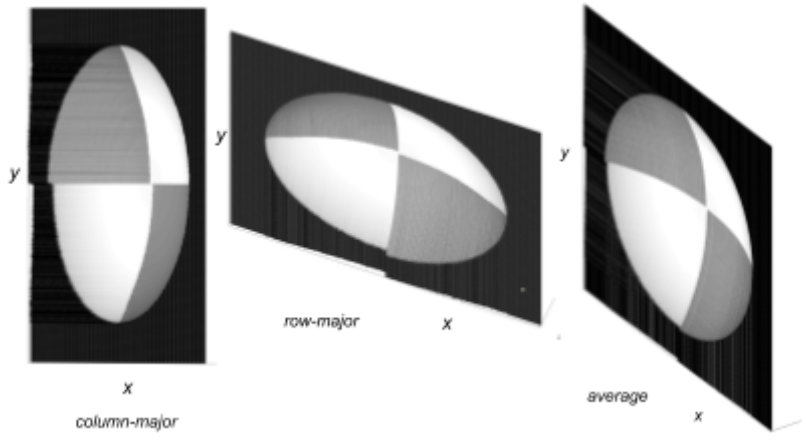


Left: number of images n = 15, outliers = 1853. Right: n = 25, outliers = 1457. The threshold is set to 0.005, because this value gives (in general) an amount of outliers reasonable for visualization. As can be seen from the integrability check, the outliers all lie in a circle, i.e. the edge of our object. For n = 15, both the number of outliers and the values of the errors are larger compared to n = 25. The reason for the errors could be the fact that at the edge of our object, the slope is relatively large as opposed to the center of the object, which looks flat from our perspective. A large slope could easily result in large errors in the derivatives, since we are taking derivative estimates based on discrete values. However, the more images we have, the smaller the errors and the smaller the amount of outliers, because the albedo calculation is more accurate (see 1.1.3) and therefore the normals and the derivatives are better estimated.

1.3 Shape by Integration

1. See code construct_surface.m.

Rico Mossinkoff - 12805157
Ewoud Vermeij - 11348860
Hannah Lim - 10588973
Yke Rusticus - 11306386

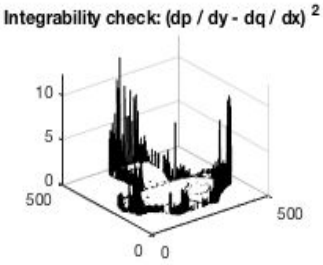


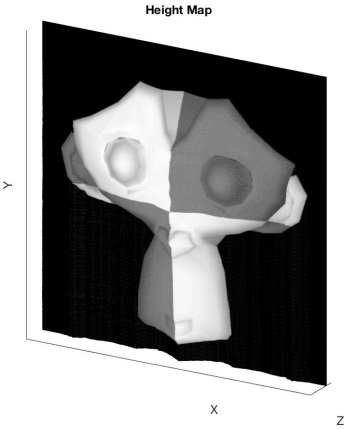
The images show constructed surfaces of the gray sphere based on 5 input images. If the surface is constructed using column-major path integration, then we see a small abnormality, or edge, along the y-axis. If we use row-major order instead, the edge becomes visible along the x-axis. The abnormality is shifted by 90 degrees, which is intuitively explained by the fact that we shifted our integration order by 90 degrees.

2. See code.
Averaging between the two methods gives a better result than in both cases: both abnormalities are still present in the image, however small enough to make the black background look flat, which is what we want. In the image presented in 1.3.1, where n = 5, the abnormalities are already hard to recognize. In the case of n = 25, the abnormalities are hardly recognizable anymore.

1.4 Experiments with different objects

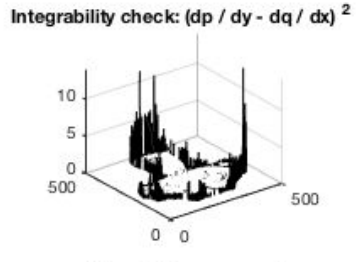
1.

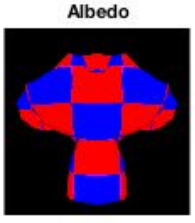




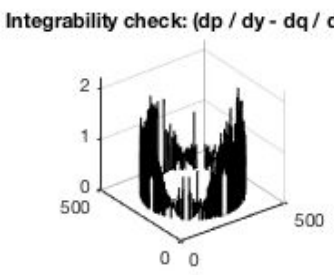
The sphere is convex, so there are almost no distortions. This is not the case for the MonkeyGray model, in which there are more shadows. This results in more outliers (~4500 for the monkey, ~1500 for the sphere). Moreover, some outliers now lie around the center of the object as well, not only near the outer edges. This is because the features of the monkey (eyes, nose) have steep slopes, and are therefore prone to errors in reconstruction. Taking more images in which important edges are well illuminated could reduce this problem.

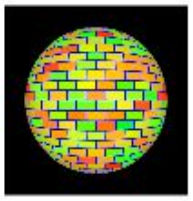
2. See code photometric_stereo.m.

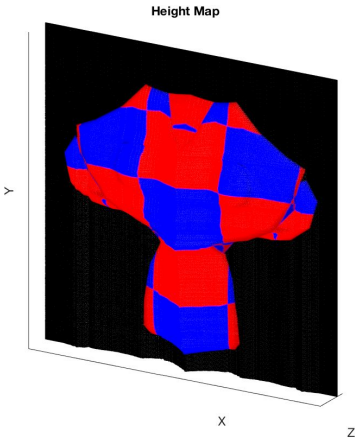




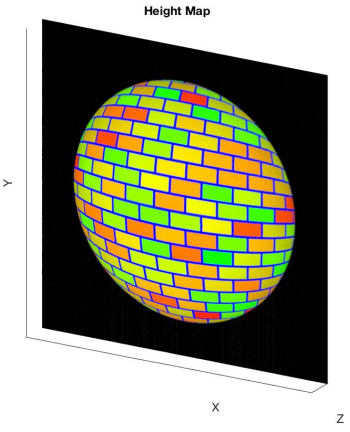
Rico Mossinkoff - 12805157
Ewoud Vermeij - 11348860
Hannah Lim - 10588973
Yke Rusticus - 11306386



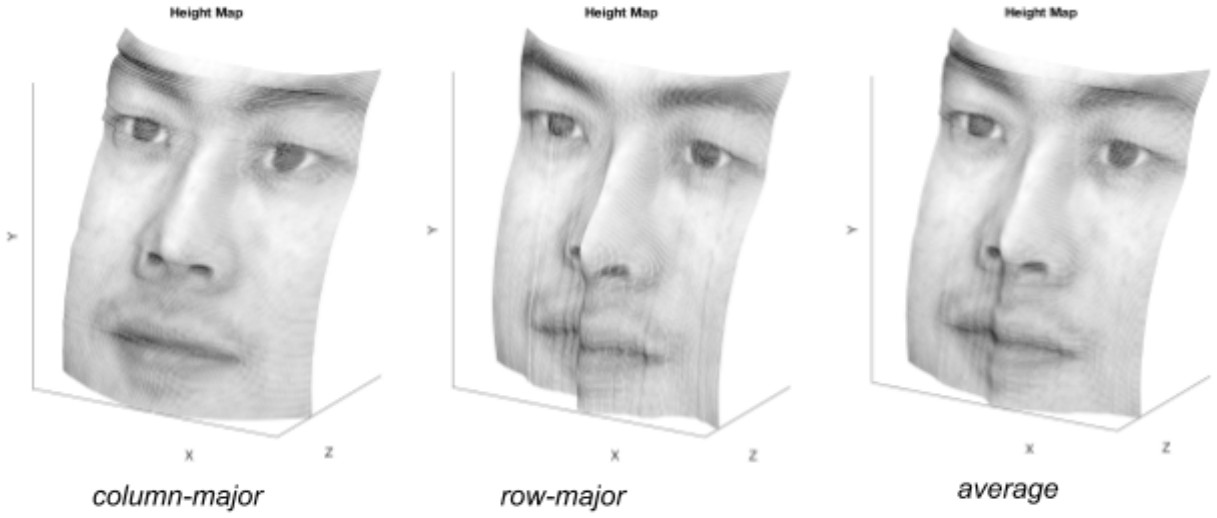




In the case of three channels, we run the algorithm for each channel individually. By combining albedo images from each channel we get an RGB albedo image as presented above. A problem arises when the normals are calculated based on the individual albedo images. In that case, the program cannot "see" most parts of the image and simply does not calculate (correct) normals in those regions where the albedo for a specific channel is zero. Therefore, calculation of the normals is done in the same way as for grayscale images. The RGB images are first converted to grayscale and are then used for calculating the normals. Another problem arose for the monkey, because there was no green color present in the images, which resulted in errors when running the program. To overcome this problem, we set all values in the green channel to zero (instead of NaN).



3.



As can be seen from the height maps, the integration path has a large influence on the reconstructed surface. The face is best reconstructed if we use column-major order.

However, the eyebrows are reconstructed as dents in the surface in the case of column-major order. This could be due to the reflection properties of the eyebrows and the influence the column-major order has on the following rows. For example, if an error is made in the first column during reconstruction, then this error is the starting point for the reconstruction of the corresponding row. Eyebrows consist of hairs and absorb a lot of light. The abnormal reflectivity could be the reason for errors around the eyebrows in the first column, resulting in a dent in the heightmap over the entire corresponding rows. The eyes are also not well reconstructed. Eyes are highly specular, so they reflect a lot of light under specific angles. This property is not well handled by the algorithm, which works for diffuse materials. Another thing that could have had a negative effect on the results is that there are two images in the given dataset with pixel-like noise. However, removing these images from the dataset did not give different results.

2 Color Spaces

1. RGB Color Model

RGB colors form the basis to create all visible colors to the human eye. This makes the color model great for a basis to represent digital images. A standard digital camera has a Charge-Coupled Device (CCD). This device contains an array of light sensors and converts a received photon into electric charges accumulated in each sensor unit. Each electric charge is translated to a corresponding RGB value which represents the received color.

2. Color Space Conversion

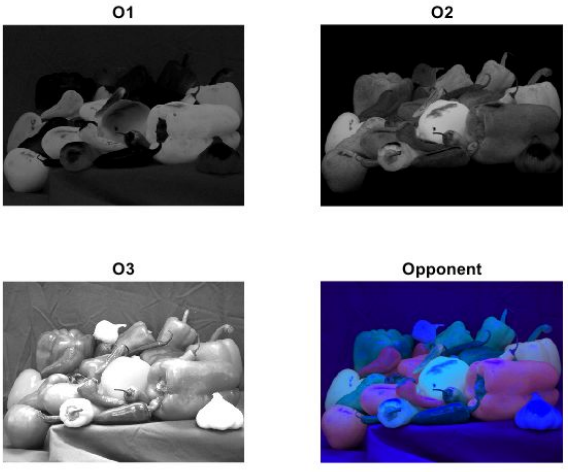
See code ConvertColorSpace.m (images are shown in the next question).

3. Color Space Properties

For each color space, we showed each channel in a gray scale for consistency to have a clear view on the differences between channels and color spaces.

Opponent

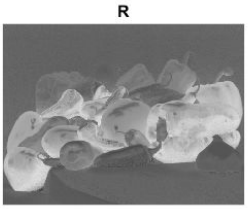
The opponent color space consists of three components: O1, O2 and O3. Each component consists of a combination or subtraction of two or more RGB components divided by the square root of 2, 6 or 3. We can see that O3 has more influence on the image compared to O1 and O2. Therefore the color Blue is predominantly present. A benefit of the opponent color scheme is sharpening the color contrast between opponent colors.

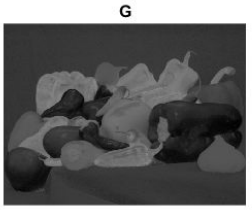


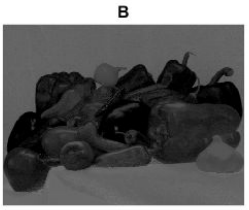
Normalized RGB

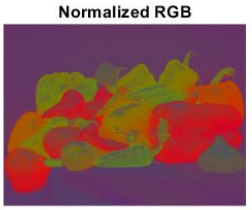
Normalized RGB divides each color by the sum of R, G and B. This makes smaller details like shadows disappear. Therefore objects are more distinguishable. For example, we can

see a clear difference between the peppers and the background. An algorithm for object recognition can benefit from this since the smaller details are left out.



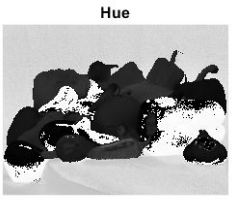


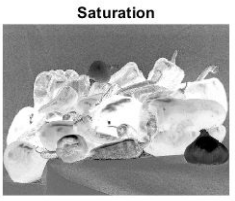




HSV

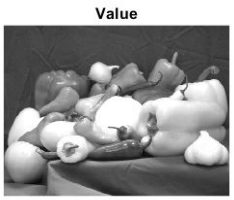
Hue describes the actual color. Saturation describes the amount of gray in a color and Value describes the amount of white or black. The benefits of this color schema is that it is more intuitive. Altering one of the components to, for example, adjust the brightness of an image can be done with the Value component. With RGB a combination of the three components need to be altered.

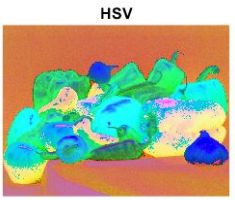




YCbCr

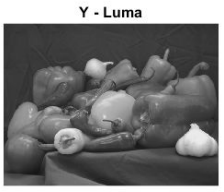
Y describes the brightness, Cb is blue minus brightness and Cr is red minus brightness. A benefit of YCbCr is that the same image in YCbCr takes less space in memory compared to that image in RGB.

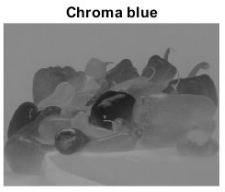


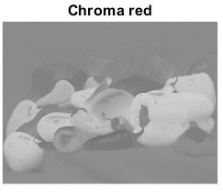


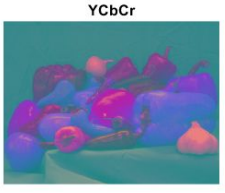
Gray

The lightness method has been applied to the first image and averages the most and least prominent colors. The second image averages the values of R, G and B. The third image uses the luminosity method which averages the RGB colors as well, but each color differently. Green is weighted most heavily since the human eye is more sensitive to green colors. The last image uses the same method as the third image, having each color weighted slightly different.
Image processing can benefit from using a gray scheme. Color does not necessarily help finding edges or features in an image. It is also simpler to process, since a gray color scheme consists of only one channel.



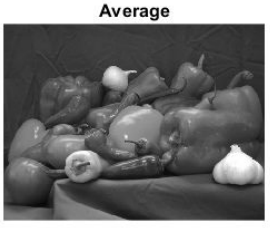


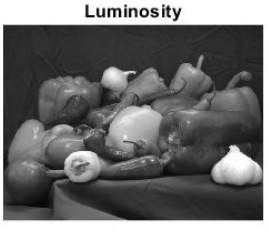




4. More on Color Spaces

The XYZ color space, developed by the CIE, can be used to obtain the chromaticity coordinates. The chromaticity diagram represents all colors perceivable by humans.





Rico Mossinkoff - 12805157 *Hannah Lim - 10588973*
Ewoud Vermeij - 11348860 *Yke Rusticus - 11306386*

3. Intrinsic Image Decomposition

1. Other Intrinsic Components

Shape and texture. The texture of an image can be quantified using a set of metrics (such as albedo and illumination) to give information about the spatial arrangement of color or intensity of an image. Given the texture of an image, one could perform edge detection.

2. Synthetic Images

Synthetic images can be generated more easily compared to real images. It is easier to generate shaded images with different light source directions for the same surface. Large datasets are preferable when training a machine learning or deep learning model. Since synthetic images can be generated with some computer algorithm, most intrinsic image decomposition datasets are composed of synthetic images.

3. Image Formation

See code iid_image_formation.m.

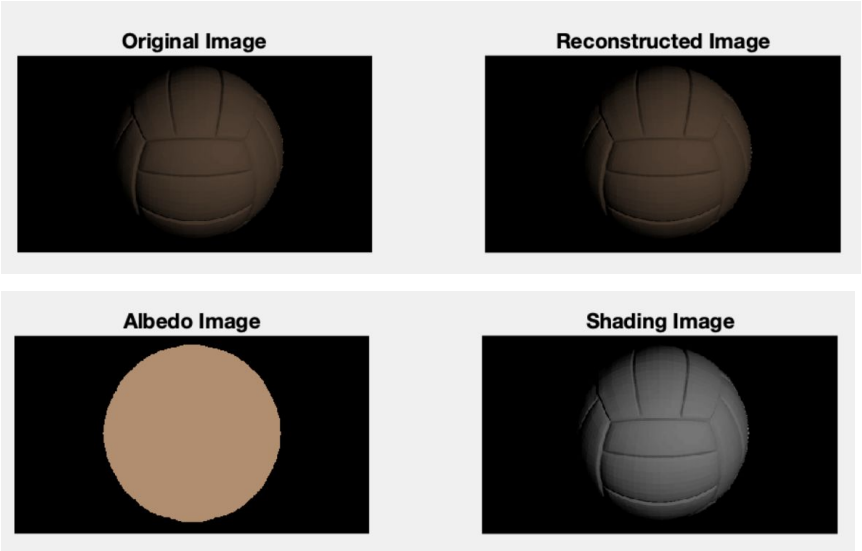


Image displaying the original image, it's intrinsic images (i.e. the albedo and shading image) and the reconstructed image.

4. Recoloring

1. The true material color of the ball is [184,141,108] in RGB.

2. See code recoloring.m.

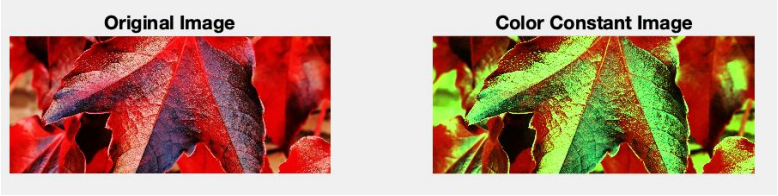
3. To recolor the ball image, we changed the color of the albedo image (True material color) into RGB(0, 255, 0). When reconstructing the ball image with the new albedo image and

shading image, we get a similar shaped image compared to the original image, but now containing the color green. the color distribution in the recolored image are not uniform due to the shading image which makes the green ball darker at some locations in the image.

4 Color Constancy

1. See code AWB.m.
2.

The Grey-World Algorithm would fail if we do not have a good distribution of colors in the image. If an image has a dominant color, for instance orange, then the color distribution of the image will not meet the Grey-World assumption (i.e. that the average reflectance in a scene is achromatic) and so the algorithm will fail. In the example below we can see that in the original image there is a very obvious dominant color. Therefore you can see in the right image that the Grey-World Algorithm fails.



3.

Gamut mapping is a color constancy method that makes use of the observation that only limited colors in the RGB spectrum can be observed for a specific illuminant. First the set of possible RGB values is identified for a base illuminant, which is normally white. Then the observed values are mapped to the base values. These mapping transformations are then used to derive the illuminant color. (“Edge-Based Color Constancy”, 2007, van de Weijer et al.)

Conclusion

With this assignment we learned a lot about various concepts in computer vision.
Matlab gave us some interesting results for photometric stereo images. We learned that the shadows present in an image influence the albedo and the normals and therefore influences the surface structure. Using more images to calculate the albedo and the normals will have a more accurate construction. Also by implementing the shadow trick the albedo is smoother. Our expected results did coincide with the actual results most of the times. However, when computing the surface structure of the face images, we would have expected that the quality of the face reconstruction would be a lot worse because of the very divergent images. This was not the case, as the quality turned out to be quite good (for column-major order integration). In further elaboration we could use multiple paths to compute the surface structure.
In addition, we learned about different color spaces and how to implement these. Each color space gives different information about the image which can come in handy by adjusting different image properties. Information such as the albedo, different grayscales and HSV (Hue, Saturation, Value) can be extracted with specific algorithms performed on RGB represented images. Furthermore, we reconstructed images with the components albedo and shading of an image. Finally, we have learned to correct the color of an image to the true color despite the color of the light. We did this using the Grey-World Algorithm. This algorithm fails if it the color distribution doesn't meet the assumption that the average reflectance is achromatic. Therefore another color constancy algorithm could be used, like Gamut mapping.