Computer Vision - Theoretical Assignment 1

Wolf, Florian - 12393339 (UvA)
flocwolf@gmail.com

Biertimpel, David - 12324418 (UvA)
david.biertimpel@protonmail.com

Lindt, Alexandra - 12230642 (UvA)
alex.lindt@protonmail.com

Fijen, Lucas - 10813268 (UvA)
lucas.fijen@gmail.com

October 5, 2019

1 Introduction

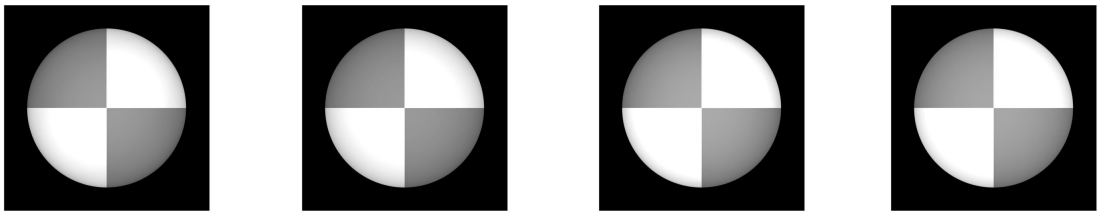
In the first practical assignment, we explore different approaches to photometric stereo as well as the analysis of colors in images. Regarding the former, we first look at the estimation of albedo reflectance and surface normals from a series of one image taken under different light sources. We then carry out the test of integrability as a sanity check on our results and for illustrating errors that occur in them. We continue to use the derivatives calculated for the test to derive a surface high-map using the row-major, column-major and average method. Finally, we apply the photometric stereo algorithm to three further data sets to gain a deeper insight into its behavior and abilities. The second part of the assignment focuses on the topic of color in digital images. We explore the properties of different color spaces, learn about intrinsic image decomposition and get to know two algorithms for color constancy.

2 Photometric Stereo

2.1 Estimating Albedo and Surface Normal

In this chapter we perform experiments and analysis first on gray-scale pictures of the same object under different lightning conditions. Then more complex objects and RGB color space are used for the experiments.

1. After implementing the code and calculating the albedo normal, we would expect the sphere to be completely free of shadows (uniform albedo). However, as depicted in figure 2a, the results still show slight shadows at the edge of the sphere.
2. The minimum number of images to perform the albedo estimate and calculate the surface normal is higher than 3 so that we can apply the least squares solution. By using the 25 images of the sphere (see figure 1c and 1d), the albedo gets estimated more accurately and we clearly observe less shadows around the edges of the sphere.
3. In our example the light source does not uniformly illuminate the sphere resulting in some areas being in the shadows. This can compromise the least squares result, as the corresponding pixel values (near 0) will be seen as outliers. The shadow-trick deals with this problem by multiplying both sides of the least squares equation with the diagonal matrix of the image-stack, which zeros out the contribution of the shadows. In figure 2 we visualize the effects of the shadow trick for the case of 5 and 25 images. From our rationale the shadow trick should have a stronger effect in the case of 5 images as these are more affected by outliers. In our results, however, the effects do not seem to differ significantly when comparing the 5 and 25 image case.



(a) (b) (c) (d)

Figure 1: Illustration of the albedo images of the spheres. (a) and (b) are computed with 5 images and (c) and (d) with 25 images. In (b) an (d) we used the shadow trick.

2.2 Test of Integrability

The test of integrability checks the parital derivatives of our normal computation. We need the derivatives to compute the surface height map later on in this assignment.

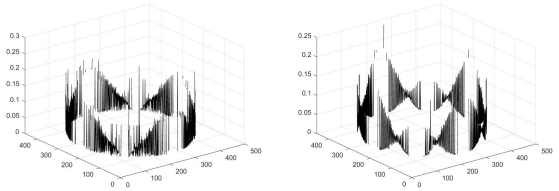
1. The partial derivatives are calculated as follows:

$$\frac{\partial f}{\partial x} = \frac{a(x,y)}{c(x,y)}\,; \qquad \frac{\partial f}{\partial y} = \frac{b(x,y)}{c(x,y)}$$

2. By taking the second derivative both terms should be equal. The test of integrability checks if this assumption holds:

$$\frac{\partial^2 f}{\partial x \partial y} = \frac{\partial^2 f}{\partial y \partial x}$$

   Since we use numerical computation, a small difference between the terms is accepted. The threshold value for this difference is set to 0.005. We consider that errors occur at points (x, y) where the difference exceeds this value. Using a higher number of images results in a smoother albedo and normal map. Errors appear at sharp geometrical edges, where the normal map makes sudden changes. This can be observed in the following plots, which illustrate the occurring errors using a set of 5 and 25 images.



(a) (b)

Figure 2: The integration error with treshold 0.005 for a set of 5 and 25 images.

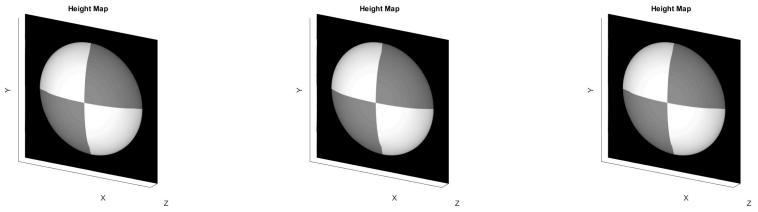
2.3 Shape by Integration

This section presents an algorithm to reconstruct the image height map. Therefore we integrate the partial derivatives over one of three path.

1. When using the column-major path (see figure 3a) we get a good 3D reconstruction of the sphere in horizontal axis, whereas on the vertical axis the sphere seems squeezed together. This is particularly evident when considering the sphere's upper and lower edge, which are dented

inwards. In the case of the row-major path (see figure 3a) we observe similar results but 90 degrees rotated. Now the vertical axis is reconstructed well and the horizontal axis shows artifacts at its edges. Note that we obtained both results by using 5 sphere images.

2. Averaging over the two previous paths gives a rather uniform result. Now we can observe that both axes seem to be squeezed together but not as strongly as before. Specially when it comes to preserving the symmetry of the sphere, this path is superior to the column-, and row-major paths. For the row and column path we can generally say that when using the 25 sphere images, the 3D reconstruction becomes more accurate and the artifacts reduce significantly. This causes the average path indentation at the edges to disappear almost completely, resulting in nearly flawless 3D reconstruction.



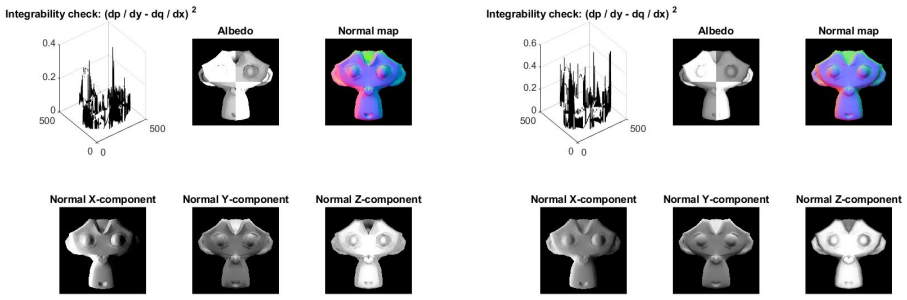
(a) (b) (c)

Figure 3: Depiction of the different effects of (a) row-major integration, (b) column-major integration and (c) the average of both, on the surface height map.

2.4 Experiments with different objects

The used sphere is a rather simple test object. In this section we use more complex geometries to generalize our results.

1. Question 4: Running the algorithm on the MonkeyGray set with 121 images returns a albedo with shadows around the eyes, nose and forehead. These areas hold shadows in most pictures and therefore are erroneously interpreted as surface coloring. This is also shown in the height map where these areas are not geometrically transformed.
   Using a set of only 20 images results in a worsened height map but surprisingly in an improved albedo.

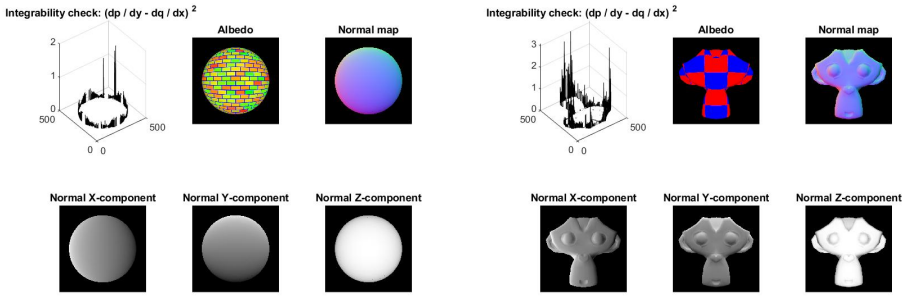


(a) (b)

Figure 4: Results for the Gray Monkey set reconstructed with 20 and 121 images.

2. Question 5: In order to extract information from all 3 color channels, we stacked each channel of each images in a third dimension. The images are stacked in rising channel order. We calculate the albedo as color image and the normals by taking the means of all 3 dimensions. The albedo image shows a much better result than the gray image set. The normals and height map become

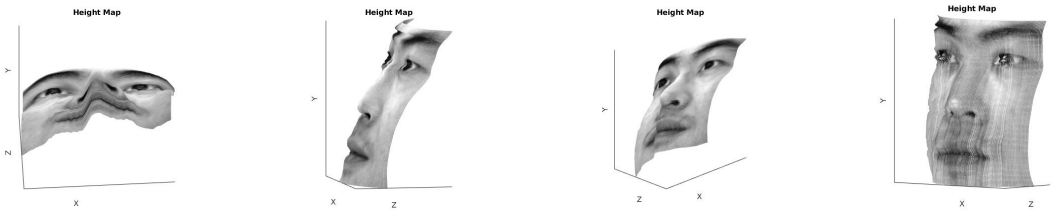
worse and does barley represent the object shape. Zero pixel are a big problem since we get NaN errors when taking the derivative. This results in empty normal and height maps. We can avoid this problem by omitting NaN values in our computation of the normals.



(a) (b)

Figure 5: Results for the Color Sphere and Color Monkey set.

3. Question 6: Looking at the results for the different integration paths (see figure 6), we notice that the observations from section 2.3 are here even more evident. While the face constructed with the row-major path seems to be squeezed on the horizontal axis, the same is true for the column-major path on the vertical axis. The average of both paths reduces both effects. Methods based on the shape-from-shading assumption, assume only scenes with diffuse (Lambertian) reflectance and therefore struggle to model specular highlights. The faces with obvious specular highlights on the skin violate this assumption and can therefore be considered problematic (see figure 15e-15h). When we remove such images from our data and repeat the calculations, we obtain the results shown in figure 14. When considering the face contructed with the average integration path (figure 6d), we see that the geometry of the face tends to be more accurate than with the complete data set. However, we also notice that the z-value of the pixels show a higher variance. This is likely due to the fewer data available for estimating the face.



(a) (b) (c) (d)

Figure 6: Illustration of the resulting height maps of the faces across the different integration paths. (a) row-major integration, (b) column-major integration and (c) the average of both. (d) shows the average path when only considering favorable face images.

3 Color Spaces

3.1 RGB Color Model

Color spaces are used to represent colors in digital formats. The basis of most color spaces is the RGB color model, standing for Red Green Blue. In this model a color is represented in 3 values that each represent the intensity of one of these colors. This is an intuitive way to represent colors and the combinations of the three colors can create a very wide range of colors.

With a digital sensor one can capture the intensity of a specific color. Filters in front of the sensors allow only one of the colors red, green or blue through. This way the sensors, which only registers the light intensity, thus measure the amount of presence of one of these specific color bases. In digital color capturing devices -such as cameras- these color sensors are put very close to each other in a grid form with altering of the 3 different color filters. The light source is projected on this grid of sensors, resulting into an image with RGB values for specific area's of this grid.

One of the advantages of this color model is that reproducing these colors can be quite easy. A modern monitor exists out of a matrix of small leds that each only have one of the 3 colors. Each led can be controlled in intensity. This way the RGB can be directly used for the intensities of the leds, which together will show 1 on 1 the original picture.

3.2 Color Space Properties

In Figure 7 the picture of Peppers is converted into 4 different color spaces. The first picture shows the 3 channels combined as if the 3 channels represent an RGB value. Although this method doesn't show the original colors of the pictures, this combined picture does show clearly the sort of differences a color model can distinguish within the picture. Also the 3 channels are plotted separately.

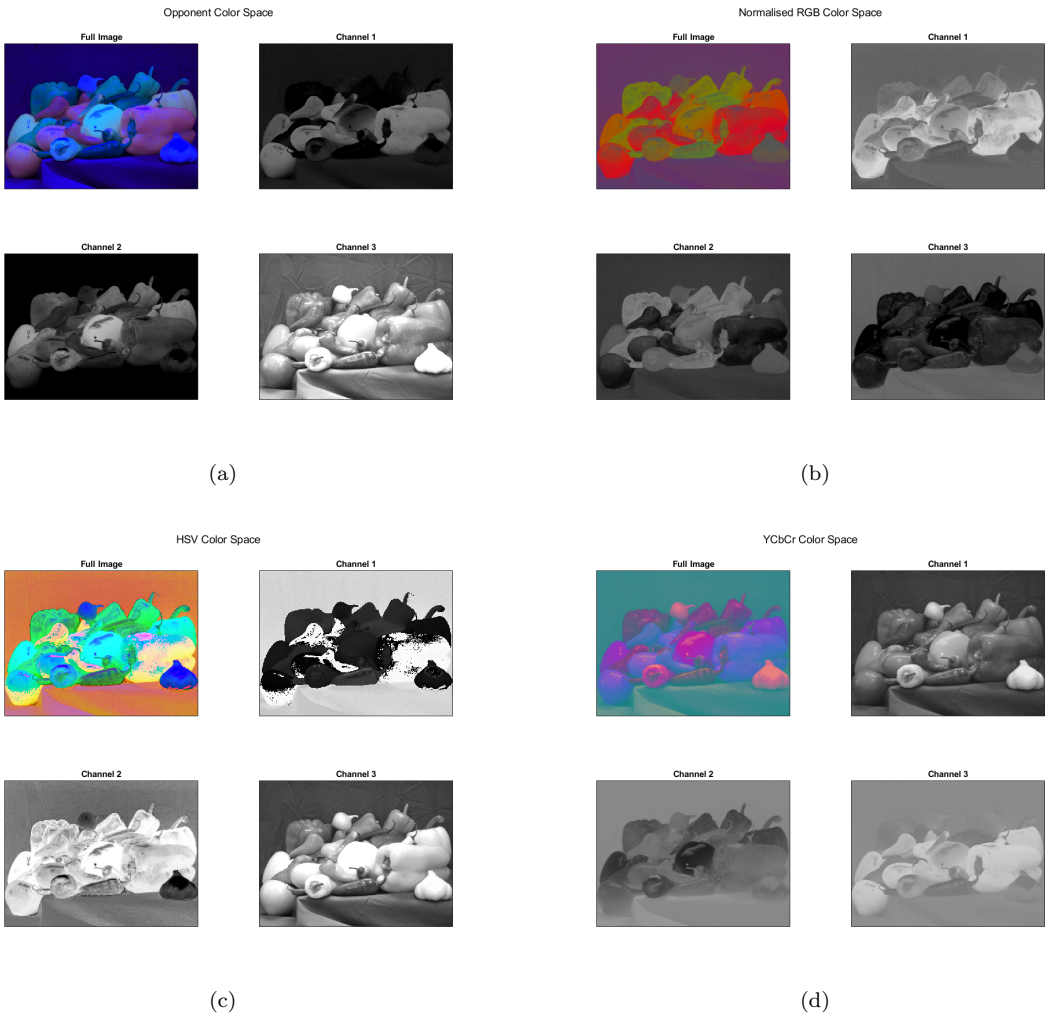


Figure 7: Color spaces: Opponent, Normalised RGB, HSV and YCbCr

3.2.1 Opponent Color Space

In Figure 7a the picture of peppers is converted into the opponent color space. The intuition behind this Color Model is that there are 3 coloring opponents. These represent the 3 Channels in the figure. Channel 1 represents red versus green, channel 2 represents blue versus yellow, and the third channel represents black versus white. As we see in the combined picture it is well possible to distinguish the different colors of the peppers and the background. Also the white of the reflection of the light source is well visible as dark spot on the peppers. This method is mainly used when there is a need to get clear distinctions between colors.

3.2.2 Normalised RGB Color Space

In Figure 7b the picture is converted into a normalized color space. In this method each color is divided by the sum of the 3 color values. The idea behind this normalization is to filter away distortion/noise such as that created by the light source. For example shadows and bright parts of a picture are coming closer to each other with this normalization. As we can see in the combined picture, the bright parts of the peppers are less present than they were in the opponent color scale. Also the background seems equally purple, in contrast with the original picture where some shadows fell on the background curtain. Thus this approach can filter out some forms of distortion caused by the light source.

3.2.3 HSV Color Space

In Figure 7c we see the picture of peppers converted into the Hue Saturation Value (HSV) color space. The intuition behind this model is to represent colors in such a way as humans experience the color. The first channel represents the hue. The hue represents the degree/direction of the color. In this channel the actual color is thus described. The second channel describes the saturation of the color. The best way to describe this channel is a measure how much the color of a certain hue is 'expressed'. If the value is high, the color is very much expressed. A low value means the color is close to black/white depending on the third channel. This third channel describes the value of the color. This value is the highest value of the RGB values. It is a measure for the amount of light that is captured.

In the Channels we see that especially the peppers have higher hue values. Also the peppers have a strong saturation, and the background and the garlic in front of the peppers both have lower saturation. The peppers also have a higher value compared to the background. As we can see the garlic has a high value together with a low saturation that results into the white color of the actual garlic. In comparison, the background is more dark and gray due to a low saturation, combined with a low value. This color model is thus very intuitive and similar to human understanding of colors. It is therefore possible to make good guesses for the 3 channel values given a certain picture.

3.2.4 YCbCr Color Space

Figure 7d shows the picture using the YCbCr color model. This model is often used in video systems. The Y channel represents the Luma, which is a measure for brightness in the picture. The Cb and Cr describe color differences with blue and red. The combination of these two measures decide the color. This in combination with the brightness of Y describes the measured color. One of the original advantages of this measure is that the Y component is a form of a grayscale, which could originally be used to represent black-white videos. On top of that the Cb and Cr could be added to give color to these images, allowing the coloring of black and white video footage towards color videos.

3.2.5 Grayscales

As we shortly described the gray-scale component of the YCbCr color space, there exist more models to represent footage in black and white. In Figure 8, 4 different methods are shown to represent color images in a gray-scale. The first method is Lightness. This method calculates the average between the highest and lowest value of the RGB input. However one of the drawbacks of this method is that very bright colors seem extremely bright. Therefore it doesn't seem to represent the way humans would see an image in black and white.

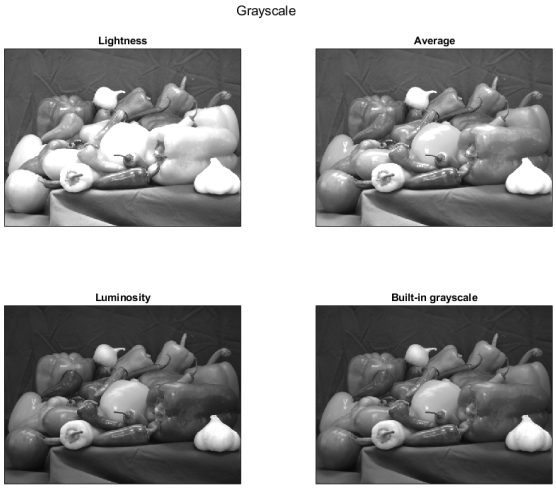


Figure 8: Grayscales

The second method is the average between the 3 values of RGB. This seems to show the differences between the bright parts of the image better. The third method is Luminosity. This measure represents the color in a similar way as the human eyem, which is more sensitive to for example red than to blue. By following these sensitivities, the resulting image seems to represent the original picture well. The last method is the built in gray-scale of Matlab. This method seems to be very similar or even equal to the Luminosity gray-scale.

3.3 More on Color Spaces; HSI

A variation of HSV that is commonly used is the HSI color space. The first two channels are equal to the earlier explained HSV measure, however the last value is in this case Intensity instead of Value. Contrarily to the Value scale, this measure is calculated by taking the average over the 3 values of RGB. This third channel equals the average gray-scale in figure 8. This measure has a slightly more realistic representation of the brightness of the picture this way and can be preferred over HSV in some complicated lighting environments.

4 Intrinsic Image Decomposition

4.1 Other Intrinsic Components

In addition to decomposing an image into reflectance (albedo) and shading, we could also consider depth information and the shape of the objects in the scene. In a scenario, each object has a certain distance to the camera that captures the scene. Equipping each pixel of the image with meaningful

depth information provides a representation of this distance. These depth information can then be used to create an intrinsic image of the scene [1].

Further, the shape of the objects present in the scene can be seen as an intrinsic image [6, 1]. The decomposition of the image only into reflection and shading leaves the problem of extracting expressive image features under-constrained. For example, considering the ball in the figure 9, the most likely explanation for its color is that its shape is round and the darker areas around the edges are shadows caused by the lighting. However, another explanation would be a flat shape and darker colors around the ball's edges. Therefore, the shapes of objects present in a scene can provide an intrinsic image.

4.2 Synthetic Images

Creating a dataset for intrinsic image decomposition with real world scenarios is a tedious task. Not only is the effort for creating a single scene high, the whole process is also associated with many constraints (e.g. no interreflections, separating specular and lambertian reflectance) [1]. Thus, obtaining precise ground truth from complex scenes is barely possible and real world datasets only construct simple scenarios (one object, one illuminant). These limitations make it difficult to generalize to unknown scenes with potentially multiple objects and light sources.

Since 3D rendering techniques have become more advanced in recent decades, they are now able to produce sufficiently realistic scenes. At the same time they allow the trivial generation of ground truth data, without the need of a manual labeling process. This enables the implementing of more complex and diverse scenes with multiple objects and lighting sources.

4.3 Recoloring

For determining the true material color we consider the uniform color of the ball's albedo component. This is reasonable as this albedo component stems from the decomposition of the original ball image. This color turns out to be (184, 141, 108) in the rgb colorspace.

In order to recolor the ball to a pure green, we change every colored pixel in the albedo image to the rgb values(0, 255, 0). If we now calculate the reconstructed image, we get a green colored ball, as shown in the second column of figure 9.

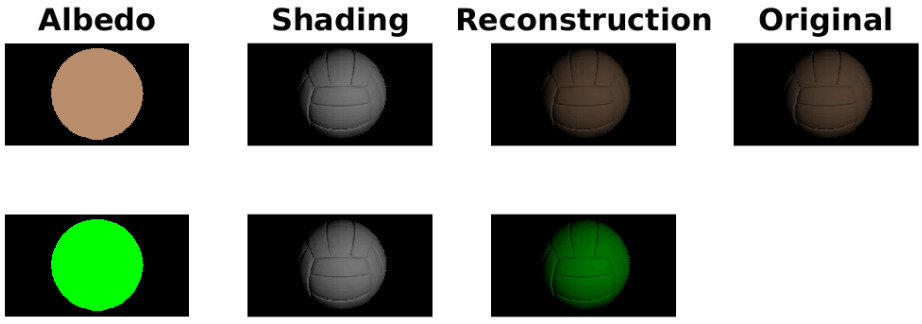


Figure 9: Illustration of image reconstruction and image coloring using intrinsic images. In the first row the reflectance (albedo) and shading components are combined to reconstruct the original image. The second rows shows the recoloring of the image by coloring the reflectance component to pure green and then performing the reconstruction as in the first row.

However, we observe that the reconstructed image does not solely display the pure color green, but rather a non-uniform green pattern. The reason for this is that the pure green colored albedo component is combined with the shading component. Depending on the position of the light source in

the image, the pure color interacts with different levels of shading, resulting in variations in color distribution.

5 Color Constancy

Color constancy is the ability to determine the true color of an object, regardless of the lighting conditions under which it is perceived. In computer vision, the purpose of color constancy algorithms is to transform images in such a way that a scene is represented in its true colors (i.e. as perceived under neutral light source).

5.1 Gray World Algorithm

One of the best-know algorithms for color constancy is the *gray world* (GW) algorithm proposed by Buchsbaum [2]. The algorithm's underlying *gray world assumption* is that the colors of the displayed objects in an image are evenly distributed over the entire color range. In this case, the average color in an image is gray. According to Buchsbaum, the average image color corresponds to the image's illuminant and can be used to compute the true colors of an image. The adjustments that the basic GW algorithm performs can be described by the formula

Figure 10: Transformation performed by the gray world algorithm.

$$[r_{ij}, g_{ij}, b_{ij}] \longmapsto [\frac{r_{ij}}{\frac{1}{n}\sum_{ij\in I} r_{ij}}, [\frac{g_{ij}}{\frac{1}{n}\sum_{ij\in I} g_{ij}}, [\frac{b_{ij}}{\frac{1}{n}\sum_{ij\in I} b_{ij}}]$$

for every pixel I_{ij} within an image of n pixels [3]. This transformation shifts the average pixel color towards the neutral gray RGB value $(128, 128, 128)$, which results in an image that appears as taken under neutral light conditions (see figure 10). In our implementation we have to linearize the pixel colors before applying the gray world algorithm, because our test images are stored in *jpg* format (see [4]). We use the von Kries model [5] to perform the color correction on the input image.

5.2 Limits of the Gray World (GW) Algorithm

* The GW algorithm does not work in cases where the grey world assumption does not hold, i.e. if not all colors are balanced within an image. For example, if we try to correct an image of an ocean (see figure 11). Since there is no presence of red within the image and the GW algorithm shifts the colors in a way to achieve an average gray pixel, this results in an image with unnatural red tint.

Figure 11: The input image misses the color red, so the GW algorithm shifts it towards an unnatural red tint to achieve an average color of gray.

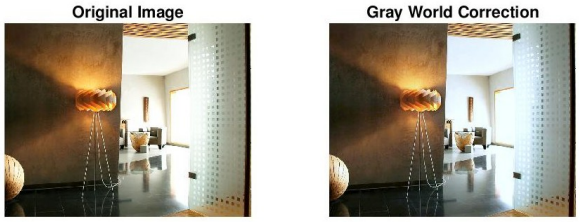


Figure 12: The input image includes two light sources. However, the GW algorithm can only estimate a single illuminant per image and therefore fails to correct the colors.

Another case in which the GW algorithm fails is if we have more than one light source within an image (see figure 12). By averaging over all pixels to estimate the illuminant, it is implicitly assumed that there is only one illuminant for the whole image. Finally, the gray world algorithm is only valid for static scenes. This means, it does not work anymore as soon as a new object enters the scene and the color distribution shifts.

5.3 White Patch Retinex Algorithm

Another algorithm for color constancy is the *white patch retinex* (WPR) algorithm [4]. Here, we look for the most white pixel within a scene and assume that it represents the maximum light possible and therefore represents the illuminant of the image. This maximum light pixel is subsequently used to scale each color band of each pixel back to the range $[0, max]$. The WPR algorithm shares one drawback with the GW algorithm: it assumes a single illuminant in an image. Therefore, it does also not perform well for scenes with multiple light sources. Furthermore, a highlight that does not reflect the overall illuminant of the image can cause the WPR algorithm to fail, because the WPR illuminant estimate will be far off the actual one.

6 Conclusion

This assignment gave us the opportunity to experiment with various methods for digital image manipulation and photometric stereo. We have seen that the albedo and surface normal (and therefore the true color) for each visible point of an object can be estimated from a series of images of the same object under differently angled light sources. For this process, we need at least three images but the estimate gets better the more images are considered. When only a few images are available, the shadow trick can help here. The albedos and surface normals for all points on a 2D image can be used for constructing a 3D high-map of the depicted object. Here, the average of row-major and column-major integration gives the best result across different datasets. We analyzed different color spaces. The opponent color space is used for color distinction, normalised RGB can be used to reduce noise by a light source, HSV is used to have a human readable metric and YCbCr is useful for videos. Also 3 versions of gray-scale were analyzed of which luminosity appeared to show the most accurate image. By performing intrinsic image decomposition, we can disassemble features of light source and entity within an image. We can modify the single components and then put them together again to receive a changed image. Finally, we saw that simple approaches to color constancy, such as the gray world algorithm and the white patch retinex algorithm, can only be applied to very specific images. This is due to the strong assumptions about their input images that both algorithms implicitly make.

References

[1] S. Beigpour, M. Serra, J. van de Weijer, R. Benavente, M. Vanrell, O. Penacchio, and D. Samaras. Intrinsic image evaluation on synthetic complex scenes. In *2013 IEEE International Conference*

on Image Processing, pages 285–289, Sep. 2013.

[2] Gershon Buchsbaum. A spatial processor model for object colour perception. *Journal of the Franklin institute*, 310(1):1–26, 1980.

[3] José M Buenaposada and Luis Baumela. Variations of grey world for face tracking. *Image Processing & Communications*, 7(3-4):51–61, 2001.

[4] Marc Ebner. *Color constancy*, volume 6. John Wiley & Sons, 2007.

[5] Mark D Fairchild. *Color appearance models*. John Wiley & Sons, 2013.

[6] Michael Janner, Jiajun Wu, Tejas D. Kulkarni, Ilker Yildirim, and Joshua B. Tenenbaum. Self-supervised intrinsic image decomposition. *CoRR*, abs/1711.03678, 2017.

7 Appendix

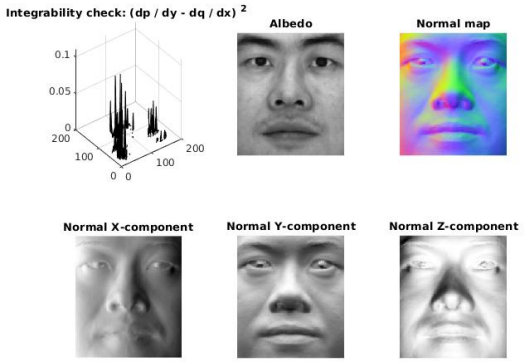


Figure 13: Results obtained with the *Yale Face Database*.

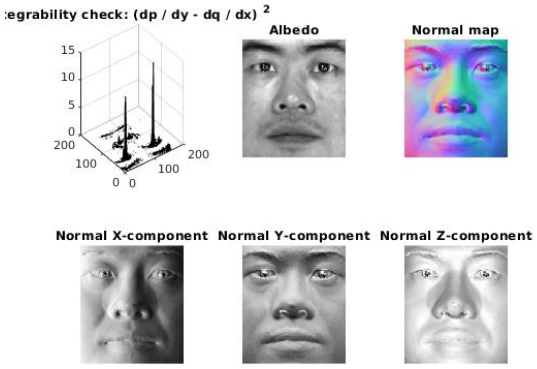
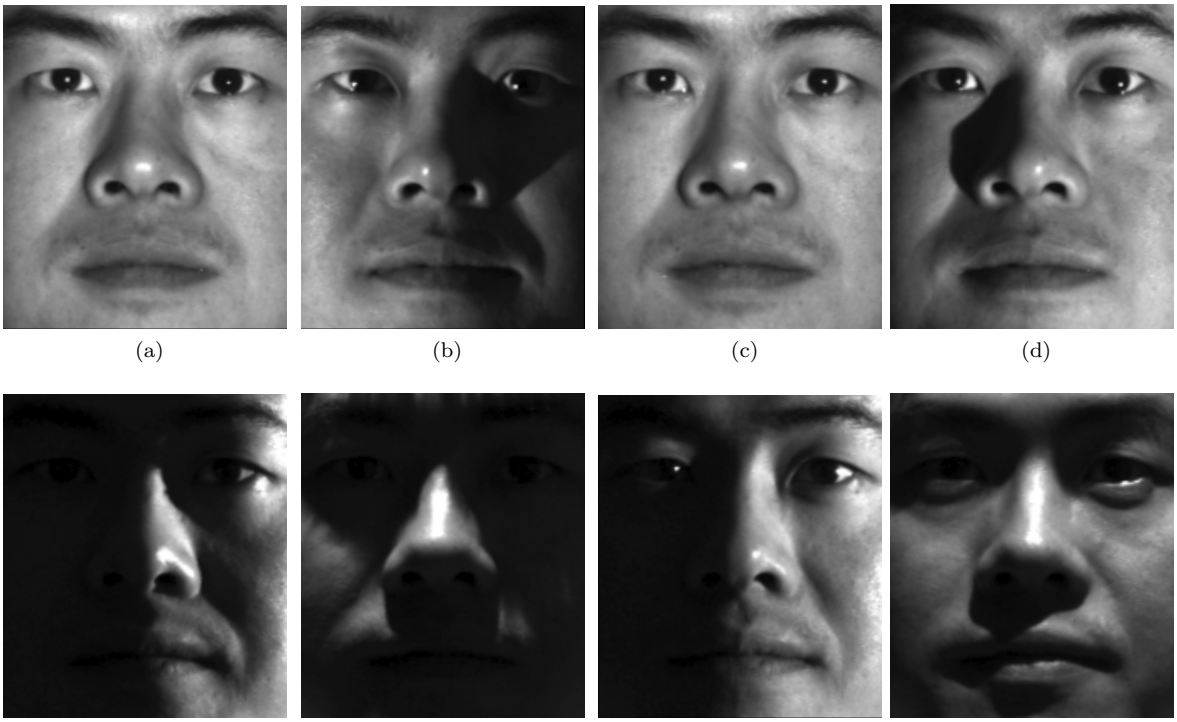


Figure 14: Results obtained with the *Yale Face Database*, with only selecting favorable images that satisfy the shape-from-shading assumption.



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 15: Examples of favorable and problematic input images in the *Yale Face Database*. (a)-(d) show favorable input faces. (e)-(h) depict rather problematic input faces.