Assignment 1

Ragger Jonkers - 10542604
Joeri Sleegers - 10631186

February 2018

1 Photometric Stereo

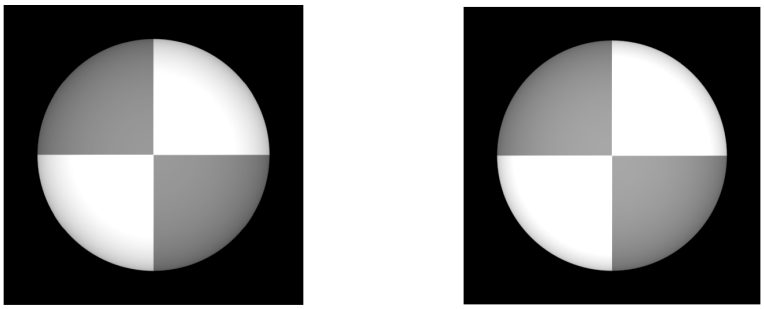
If we want to recover the depth of a surface from only 2D images, we essentially need to somehow find the Z coordinate while only having the X and Y coordinates. This can be described by the Monge patch which proposes the desired depth image as $(X, Y, f(x, y))$. Photometric stereo is a technique that tries to find this $f(x, y)$ by combining information of the same image, but with different illumination (Forsyth & Ponce, 2002).

1.1 Estimating Albedo and Surface Normal

The albedo in an image is found by calculating the diffuse reflection in each pixel. Now we have an illumination value for each different image per pixel. We stack these values in a vector. Then we can solve a linear equation for each vector to obtain the height value per pixel.

We expect the albedo to be quite uniform, since every image in the folder has a different area shaded, thus giving a good estimate to the illumination of all the pixels in the result image.

It is difficult to see the differences when varying the amount of processed images, therefore we choose to observe the largest difference of 5-25 images. When comparing the albedo results side by side, we observe a smoother border using 25 images than with only 5. The comparison can be seen in figure 1.
The shadow trick is used to deal with shadowed parts of an image. Putting each pixel in a large matrix and making shadowed regions 0, doesn't take them into equation. We performed an experiment that measured the squared difference of the pixel values between albedo's with the shadow trick and without it and did this for both the 5 and 25 image stack. The results showed a squared difference of 57.14 for 5 images and 33.10 for 25 images. This shows that the shadow trick is more useful when you have fewer images.



(a) 5 images (b) 25 images

Figure 1: Albedo constructed by processing images of different shading angles using the shadow trick

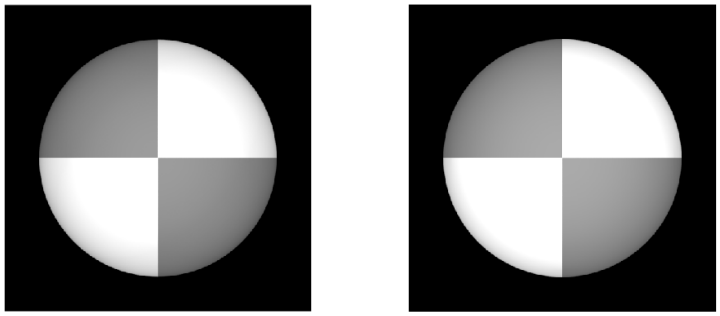


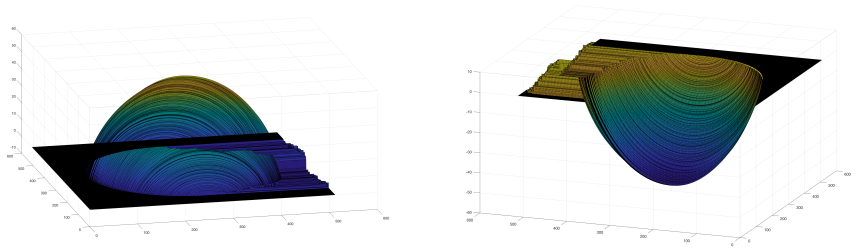
Figure 2: Albedo constructed by processing images of different shading angles omitting the shadow trick, with on the left: 5 images and right: 25 images

1.2 Test of Integrability

Because the third unknown coordinate stays the same when taking the derivative over X and then over Y and vice-versa, both second derivatives should be equal. Only we can not do a hard equal comparison as we have only estimated these second derivatives numerically. This is why we use a low threshold to determine "equality". It appears that, surprisingly, most outliers are found in the normal of 25 images. We assume that this is because there are some images in the stack of 25 that have conflicting information (e.g. the 6th image changes the "perfect albedo" that was formed by the first 5 images that do not violate the assumptions of no ambient illumination or reflection). Consequently this means there are no conflicting images in the 5 stack.

1.3 Shape by Integration

The height map of an image is constructed by summing over the partial derivatives in either the X or Y direction. These two ways are visible in figure 3 for the 5-image stack. The average is taken of the X in Y in figure 4 for 5 and 25 images. There is not much difference between the row and column-wise approach, but the average looks a lot better than either one, even more when using 25 images.



(a) Row-wise

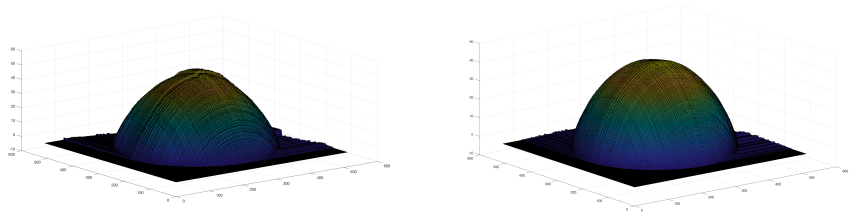
(b) Column-wise

Figure 3: Height map construction summing over either the X derivative(column-wise) or Y derivative(row-wise)

1.4 Experiments with different objects

We ran the algorithm on all MonkeyGray images and on only 5 of them, hoping that these 5 images would aid better than all of them in which some may be counterproductive as described in section 1.2. The outlier count was slightly lower with 5 images, but will not result in visible difference of the height map.

To construct a height map from RGB images, we seperated them into their respective channels and combined their height maps later on. Zero values are a

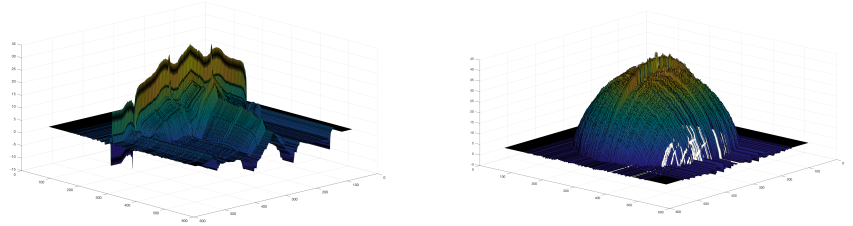


(a) 5 images

(b) 25 images

Figure 4: Height map construction by taking the average of row and column-wise method

problem, because they are most likely very inconsistent with the average over all three channels, therefore it is possible to use a gaussian convolution. You could also just convert the RGB to gray and then run everything on the gray image. Our results on the RGB sphere and monkey are visible in figure 14



(a) RGB monkey

(b) RGB sphere

Figure 5: Height maps constructed by averaging the height maps of the different RGB channels

The Yale face dataset violates local shading assumption. This assumption states that there can be no ambiant illumination an that there should be only one light source. It was difficult to spot which specific images violate this assumptions since a face has a lot of surface reflecting points.

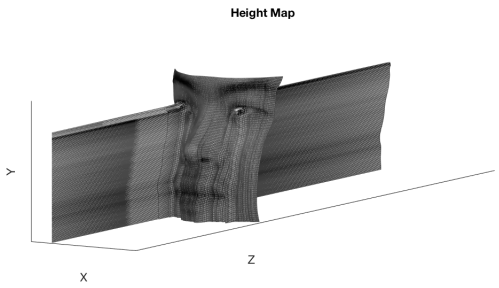


Figure 6: Yale face reconstruction

2 Color Spaces

We use RGB model as the basis of our digital cameras and photography, because humans don't perceive the full spectral distribution but rather a integral on a point in the distribution (Ohta & Robertson, 2006).

Color Space Conversion Below are the results of doing the Color Space Conversion for HSV, normalized RGB and blabla repsectively.

Hue Saturation and Value Hue Saturation and Value is a color system that can be obtained by doing a non-linear transformation to the RGB values (Forsyth & Ponce, 2002). HSV is a way of decomposing the RGB value into color(Hue), saturation, and brightness(Value).

YCbCr Color Space YCbCr is a way to compress the luminance signal with a higher fidelity than the chrominance signal (Szeliski, 2010).

Opponent Color Space Opponent color space is the space on the opposite of the border in the uniform color space model.

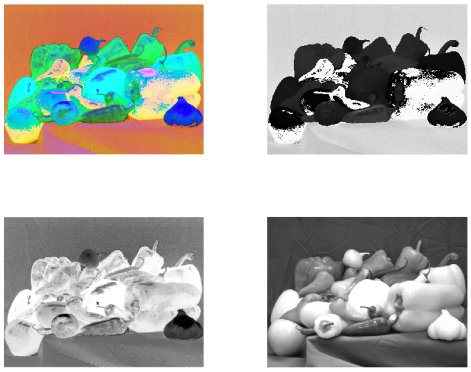


Figure 7: Hue, Saturation and Value decomposition

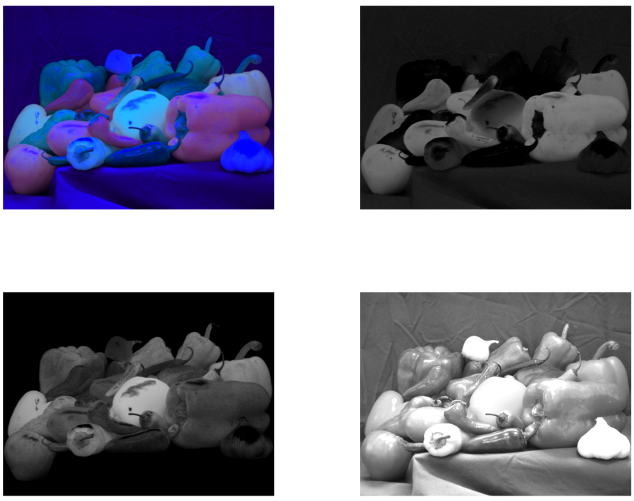


Figure 9: Opponent Color Space

Normalized RGB

Normalized RGB is a way to get rid of distortions caused by light and shadows in an image. Deviding the R, G and B values respectively by the saturation is a way to achieve this.

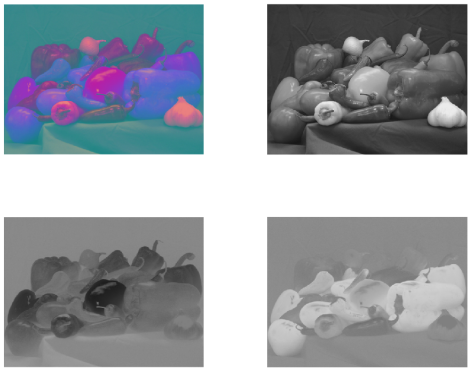


Figure 8: ycbcr

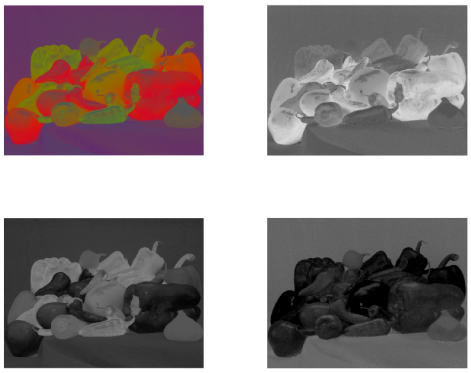


Figure 10: Opponent Color Space

Grayscale

The pepper image was converted to grayscale using three different method's, namely luminosity, averaging and lightness (d. Cook, n.d.) and matlabs build-in *rgb2gray* function. It is apparant that the top left(luminosity) uses the same conversion method as matlab's buildin function which is the bottom right.

TSL color space

This color space seperates the color into tint, Saturation and lightness it was developed for face detection (Terrillon, David, & Akamatsu, 1998).

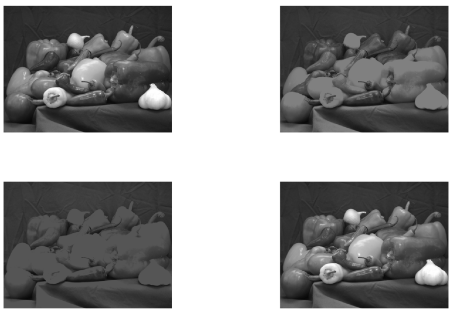


Figure 11: gray color conversion.

3 Intrinsic Image Decomposition

Images can be decomposed into several things. As an example we have reflectance. Objects in an image reflect incoming light. This changes how we perceive the color of the object. For example surface reflection of a red car can make us perceive it as orange. Shading makes a particular object appear darker because the light going to that object is occluded by another object. Normally these components are calculated as a per pixel product of shading S and reflectance R, which means there is not one solution, but multiple approaches are possible that do not necessarily require multiple images with different illumination like we used or some kind of user input. An option is to **cluster** the image on similar reflectance (Garces, Munoz, Lopez-Moreno, & Gutierrez, 2012).
We can also look into other components like **motion** and **boundaries**, depending on the application. Boundaries or edges are for example indicators of reflectance changes, and an object in motion has as similar blurry reflection for that object (Barrow & Tenenbaum, 1978).

Data sets mostly use synthetic images, because the local shading model we adopted, assumes there is no influence of other light created by things like surface reflection or even another light source that occur in natural images. Synthetic images allow for these properties of shading and reflection to be controlled.

We show how we can decompose an image of a ball into its shading and reflectance properties. Consequently we can change the color of the reflectance component to assemble a ball with a different color. Figure 12 shows the components, figure 13 the recolored ball.

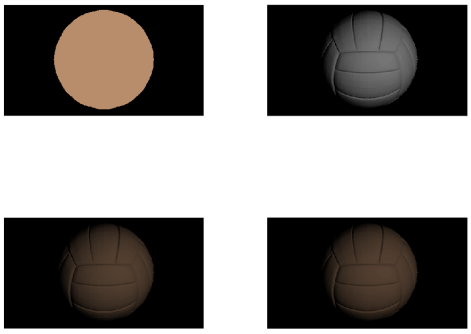


Figure 12: Top left: reflectance. Top right: shading. Bottom left: original image. Bottom right: recontructed image

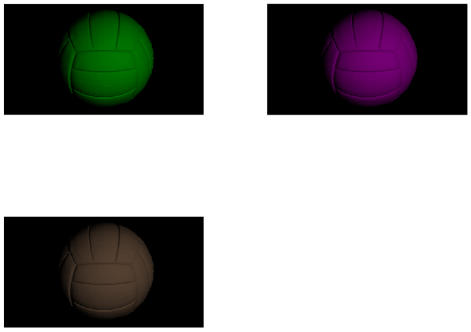


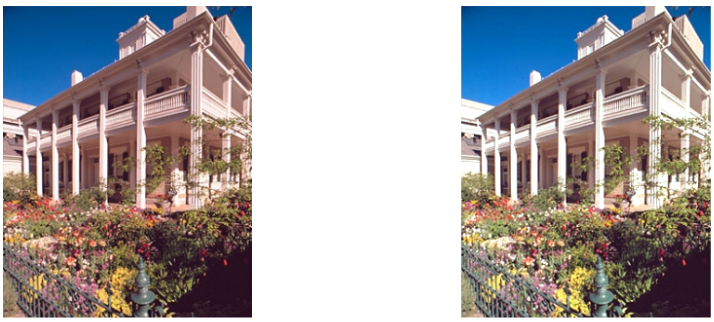
Figure 13: Recoloring of the ball

4 Color Constancy

Gray world algorithm is an algorithm to perceive colors of an object invariant of the light source. It assumes that the pixels in an image average out over a grey value in this particular case (128, 128, 128). The algorithm fails when this assumptions shouldn't hold, for instance when you take a picture in the jungle where there would be alot of green colors, you wouldn't want the picture's pixel values to average out over a gray value.

Another algorithm for color balancing is Simplest Color Balancing. This algorithm tries to make the brightest pixel in the image white and the darkest pixel in the image black.

This is the color balancing technique used in Adobe Photoshop's "auto levels" command. The idea is that in a well balanced photo, the brightest color should be white and the darkest black. Thus, we can remove the color cast from an image by scaling the histograms of each of the R, G, and B channels so that they span the complete 0-255 scale. In contrast to the other color balancing algorithms, this method does not separate the estimation and adaptation steps



(a) original (b) Grey World

Figure 14: Grey World algorithm

References

Barrow, H., & Tenenbaum, J. (1978). Recovering intrinsic scene characteristics. *Comput. Vis. Syst*, *2*.

d. Cook, J. (n.d.). *Three algorithms for converting color to grayscale.*

Forsyth, D. A., & Ponce, J. (2002). *Computer vision: a modern approach.* Prentice Hall Professional Technical Reference.

Garces, E., Munoz, A., Lopez-Moreno, J., & Gutierrez, D. (2012). Intrinsic images by clustering. In *Computer graphics forum* (Vol. 31, pp. 1415–1424).

Ohta, N., & Robertson, A. (2006). *Colorimetry: fundamentals and applications.* John Wiley & Sons.

Szeliski, R. (2010). *Computer vision: algorithms and applications.* Springer Science & Business Media.

Terrillon, J.-C., David, M., & Akamatsu, S. (1998). Automatic detection of human faces in natural scene images by use of a skin color model and of invariant moments. In *Automatic face and gesture recognition, 1998. proceedings. third ieee international conference on* (pp. 112–117).