Assignment 1: Photometric Stereo & Colour

Vasiliki Vasileiou, Victor Kyriakou, Irene Papadopoulou

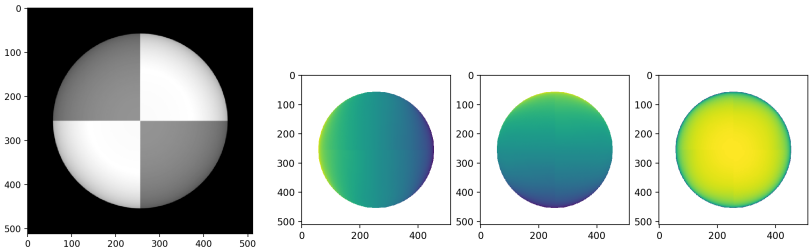
September 2021

1 Photometric Stereo

1.1 Estimating Albedo and Surface Normal

Question - 1

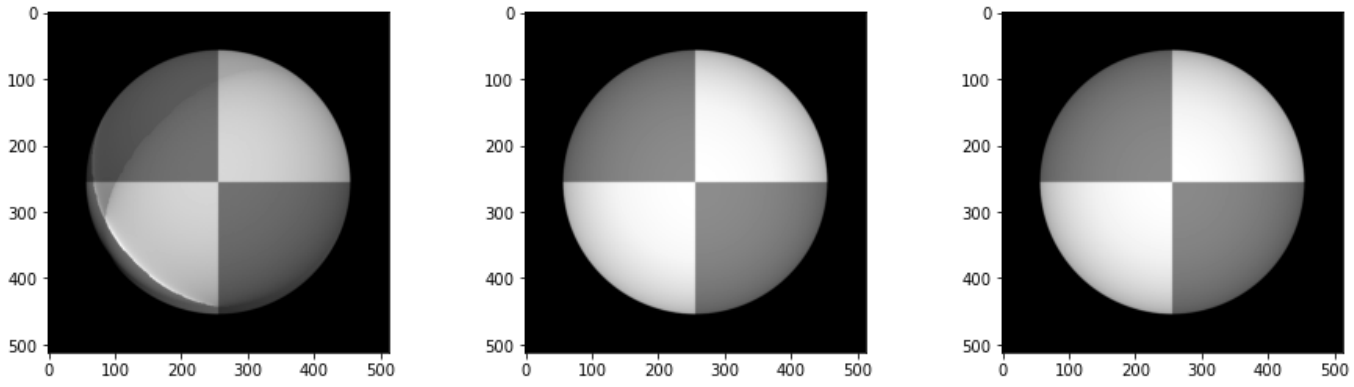
1. Using the photometric stereo method for the five sphere images with different known light sources, we get an estimation of albedo and the surface normal of the original image as shown in Figure 1. Looking at the albedo image, we notice that it is not uniformly distributed, which would be our initial expectation, instead there is shade around the boundaries of the sphere; making it a little bit darker.



(a) Albedo (b) Surface Normal

Figure 1: Albedo and Normal with shadow trick for SphereGray5

2. In principle, the minimum number of images (light sources) needed to estimate albedo and surface normal is 3; that way the solution of the least-squares in deterministic. Figure 2 shows that running the algorithm with more light sources, results to a more uniform and brighter albedo with the shade almost gone.



(a) 3 sources (b) 12 sources (c) 25 sources

Figure 2: Estimating albedo with different number of light sources

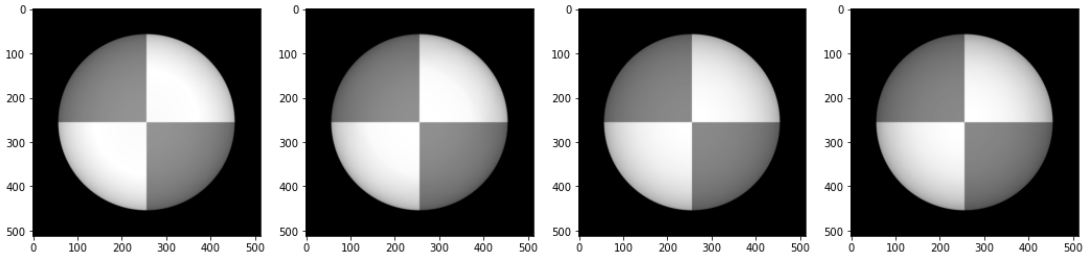
3. The shadow trick is a trick that deals with points that are in shadow. We know the photometric stereo equation:

$$i(x, y) = Vg(x, y)$$

and we want to zero out the equations from these points so they don't contribute in the calculation of $g(x, y)$. In order to do that, we form a diagonal matrix $I = diag(i)$ which we multiply from both sides of the equation so it becomes

$$Ii(x, y) = IVg(x, y)$$

Figure 3 shows the results we get with and without the shadow trick. We observe that with 5 sources the result of the trick is more noticeable, so the use of the trick is more necessary when we have a smaller number of sources.



(a) 5 sources, with trick (b) 5 sources, without trick (c) 25 sources, with trick (d) 25 sources, without trick

Figure 3: Albedo results, with and whithout the shadow trick

1.2 Test of Integrability

Question - 2

1. In order to compute the the partials derivatives $p = \frac{\partial f}{\partial x}$ and $q = \frac{\partial f}{\partial y}$ we compute

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

where $a(x, y), b(x, y)$ and $c(x, y)$ are the three values of the unit normal at some point (x, y). Since we already calculated the normal at each point, we can easily compute p and q.

2. To perform the test of integrability, we need to compute the second derivatives. We expect that

$$\frac{\partial p}{\partial y} - \frac{\partial q}{\partial x}$$

is a number very close to zero since we compute these numerically; in principle, it should be zero. We want to compute these derivatives, in order to find the Squared Error (SE), which is given by:

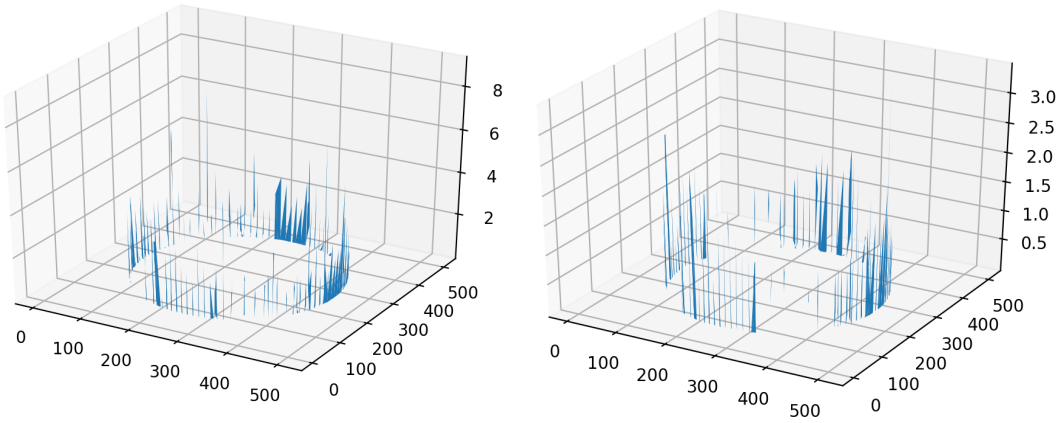
$$(\frac{\partial p}{\partial y} - \frac{\partial q}{\partial x})^2$$

The next step to complete the intergrability test, is to choose a reasonable threshold. By trying different thresholds, we observe that as the threshold decreases the the number of outliers increases. Noticeably, the errors mostly occur on the boundaries of the sphere, most likely because that is where the surface normal was wrongly estimated. Also we notice that as we use more images, the number of outliers decreases, so we know that the test performs correctly.

Table 1: Number of outliers based on different thresholds

Threshold	0.5	0.05	0.005	0.001
SphereGray5	1161	1670	3605	6219
SphereGray25	1040	1419	2788	4858

Based on out observations, we choose the threshold 0.05, which was also the default and we get the results shown Figure 4 for the two image sets.



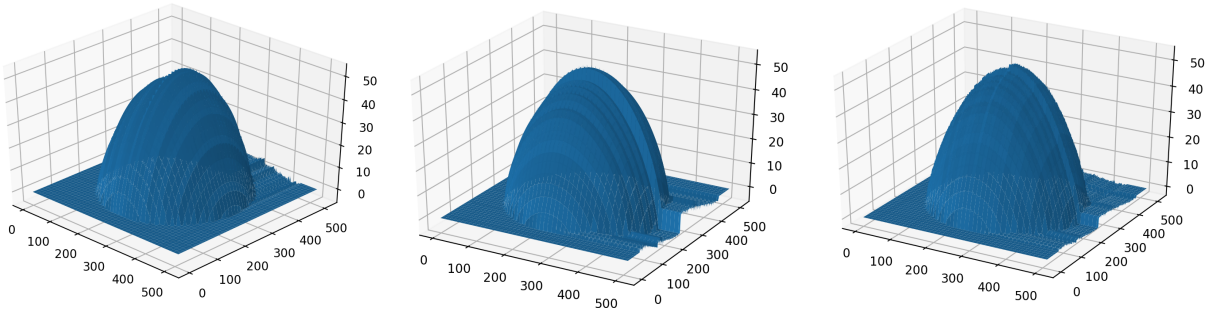
(a) SphereGray5 (b) SphereGray25

Figure 4: Squared Error with threshold = 0.005

1.3 Shape by Integration

Question - 3

1. In this part we are computing the surface height map based on algorithm 5.1 in chapter 5.4 of the book, using the column-major and row-major integration. In order to get the shape of the of the image we need to integrate the partial derivatives p and q, w.r.t. x and y, but since our surface normal function is discrete we can do it by summarizing.



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

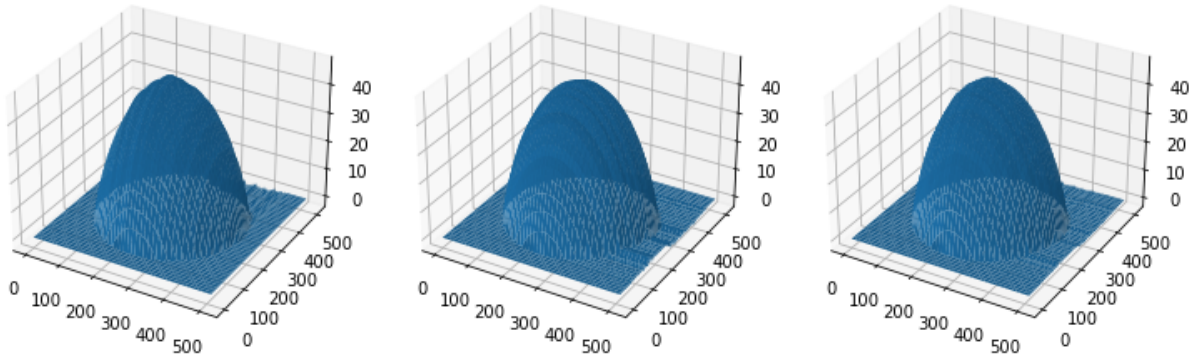
Figure 5: Height maps of column-major and row-major integration with their average on SphereGray5

Specifically, in the column-major integration, we compute the leftmost column from top to bottom by summarizing the corresponding q value with the previous height value. Then we compute the row values from left to right by summarizing the corresponding p value with the previous height value. The row-major integration operates in a same manner, by computing the top row first and then the other columns from left to right.

Figure 5 shows the results of the two integrations on SphereGray5. By comparing the two results, it is noticeable, that in both cases the error in the derivative estimates is away from the starting point. To be specific, in the column-major case the error is located at the bottom row, where in the row-major case it is located at the rightmost column.
2. By summarizing each value from the previous two height maps, and dividing by 2, we get the average height map as shown at 5(c). We notice that now the error in the derivative estimates is not concentrated in one place, but it spreads around the whole shape, yielding an improved result.

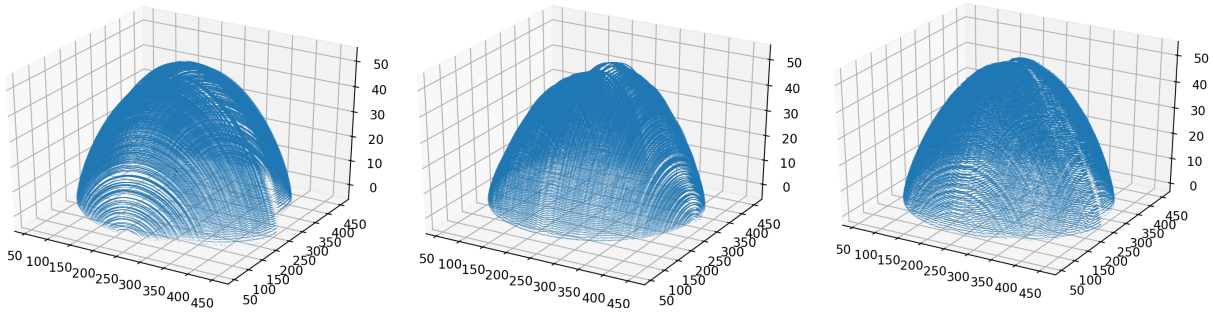
By comparing the results in Figure 5 and 6 we notice that the height maps differ majorly in each integration case, with the results of the SphereGray25 images being much more accurate.

We also implemented the quiver plot for each of the 3 height maps, using the plt.quiver() function. By observing 7, we can confidently say that the normal directions are following the way each height map is filled.



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

Figure 6: Height maps of column-major and row-major integration with their average on SphereGray25



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

Figure 7: Quiver plots of column-major and row-major integration with their average on SphereGray5

1.4 Experiments with different objects

Question - 4

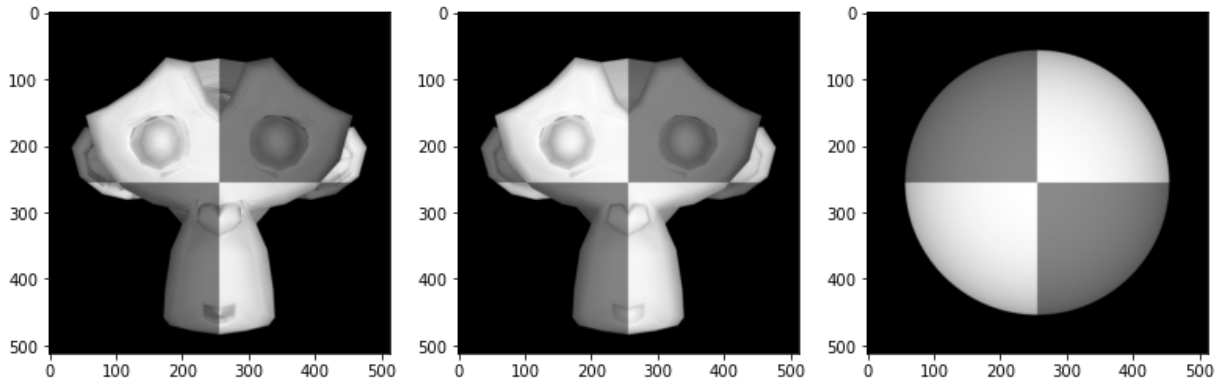
We experimented with different image ranges in the Monkey dataset and we observe behavior similar to that of the Sphere dataset. To be more precise, the greater the range of the dataset, the more robust the albedo result, as seen on Figure 8 (a) and (b). The rate of improvement of results is much lower in the case of monkey dataset. This conclusion makes sense as the monkey has a more complex shape with many edges and different depths. These points are the most defective with the most shading being there during the extraction of the albedo. Comparing the results in the total dataset (Figure 8 (b) and (c)), we observe that even in this case, where the size of the dataset of the monkeys is much larger than this of the spheres, the albedo of the sphere has far fewer shadings. Therefore, we conclude that the more complex the image of which we calculate albedo, the more images that make up the dataset must be in order to deal with the aforementioned problems.

Question - 5

For the case of RGB images, we repeat the algorithm followed in the previous queries three times, once for each channel. Then we compute the average values for normal, height map and squared error and plot them. While displaying the monkey results, a NaN value problem occurs due to the absence of green in the image. To solve this problem we used the np.nan_to_num function to remove the NaN values.

Question - 6

We ran the *photometric_stereo.py* for the *Yale Face* and without using the shadow trick, we were able to see that near the eyes of the faces there was high square error as shown in Figure 11a. After we removed the images where the eyes reflect the light the error has been decreases as shown in Figure 11b. This is because the pictures used, violate the assumption that the surface is a Lambertian diffuser. In some images light reflects directly from the eyes of the face causing major distortions in the estimation of the albedo and the normals of the image.



(a) 25 Monkey Images (b) All Monkey Images(121) (c) All Sphere Images(25)

Figure 8: Albedo experiments with different objects

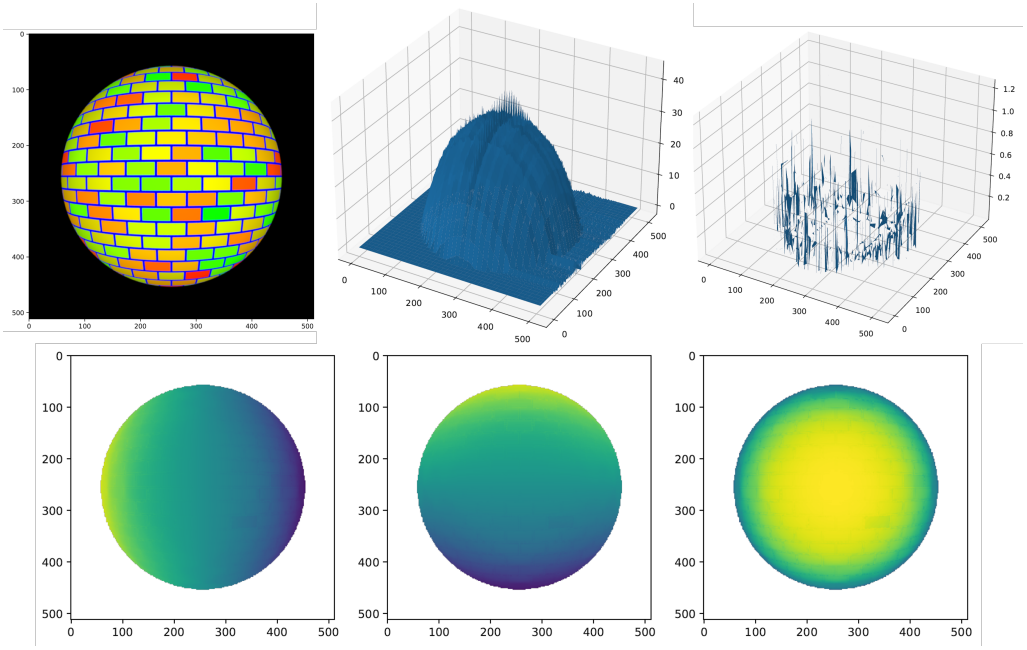
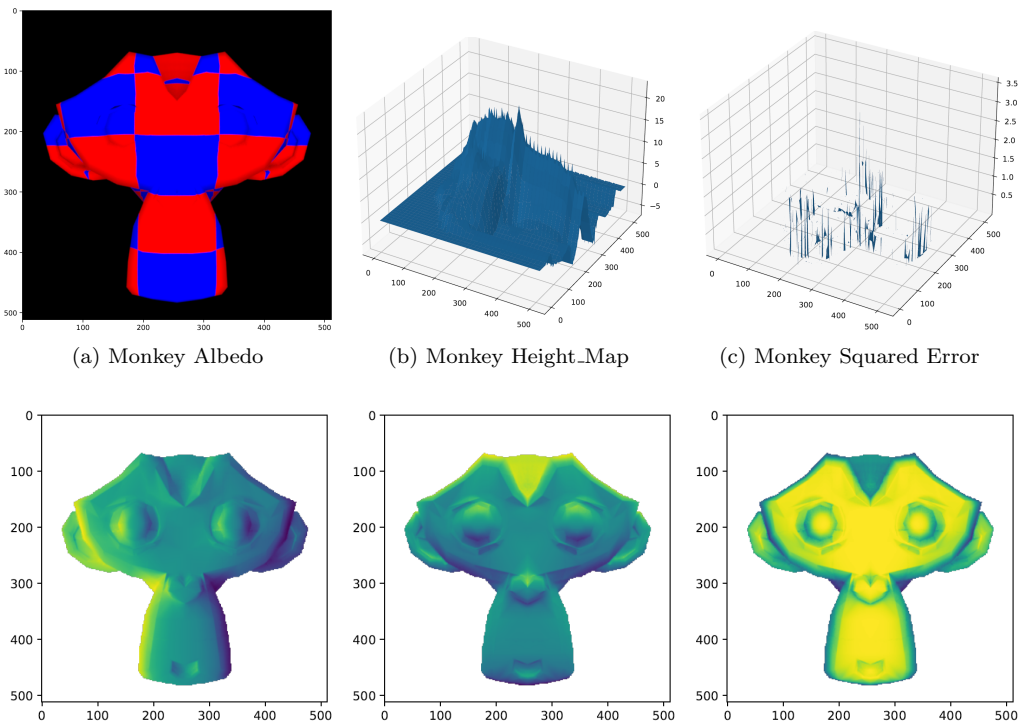


Figure 9: Sphere RGB Experiments

2 Colour Spaces

2.1 RGB Colour Model

The human visual system uses a trichromatic colour model where colours are encoded in red, green and blue. The RGB colour model is an additive colour model where different proportions of red, green and blue light wavelengths are mixed in order to produce a certain colour. The inclusion of all three wavelengths at their maximum value produces white, while the absence of all three of them produces black. Because of the human anatomy, and the way humans perceive colours, RGB is the colourspace where colours are presented as close to what humans see as true-colour and that is why is udsed as a basis of our digital cameras and photography. A standard digital camera to records the full RGB colour, by passing the incoming light through filters where only a narrow spectrum of the light wavelength passes the filter in order to look at the light in its three primary colors. Then the camera sensor records the intensity values of each primary colour for each pixel resulting in three matrices of intensity values, one for each colour. The addition of the three matrices result in the RGB image.



(a) Monkey Albedo (b) Monkey Height_Map (c) Monkey Squared Error

Figure 10: Monkey RGB Experiments

2.2 Colour Space Conversion

2.2.1 Description pf the implementation

Firstly, the images are read in BGR from the OpenCV method *imread* and converted to RGB with the given code. For the conversion the values of the image channels are converted from integers to float numbers. After each conversion to the corresponding colourspace the images are again normalized in order to take values in range of $[0, 255]$. Finally, we convert the values in *uint8* type in order to plot the using *Matplotlib's imshow*.

2.2.2 Opponent Colour Space

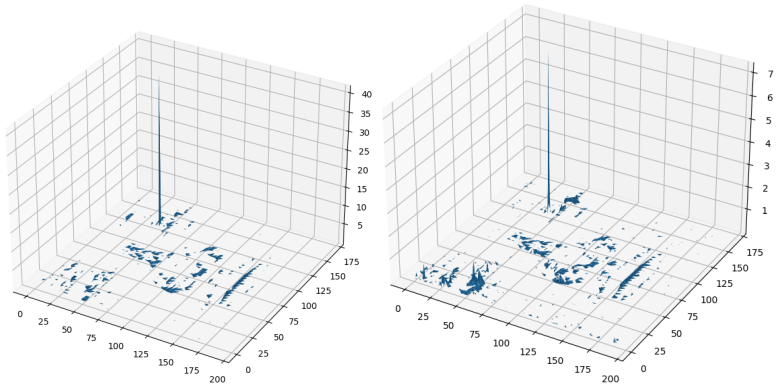
The colourspace consists of three components, namely O_1, O_2 and O_3. Firstly O_1, red-green channel, is obtain by $\frac{R-G}{\sqrt{2}}$. The second component O_2, blue-yellow channel, is obtained by $\frac{R+G-2B}{\sqrt{6}}$ and lastly O_3, luminance channel, is obtained by $\frac{R+G+B}{\sqrt{3}}$, where R, G, B the channels of the RGB colourspace. The main aim of this colourspace is the separation of luminance and chrominance, by assuming that colours are opposite to each other. More specifically, it assumes that red is opposite to green and blue is opposite to yellow.

2.2.3 Normalized RGB Colour Space

In this colourspace the image values of the RGB colourspace are normalised in the range of $[0, 1]$. The main aim is to remove the distortions caused by lights and shadows in an image. It has application in object recognition on color images, when it is important to remove all intensity values from the image while preserving color values.

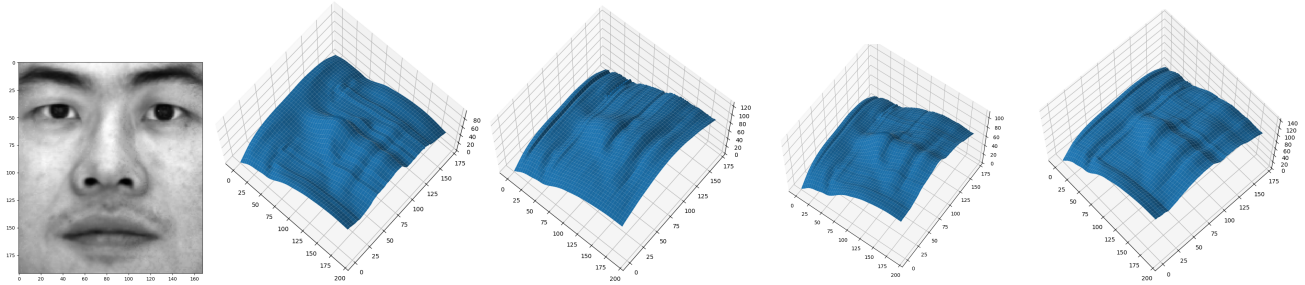
2.2.4 HSV

This colourspace can be depicted as a cylinder where it maps the RGB values in a more convenient way for visual interpretation and consists of three elements, namely *Hue*, *Saturation* and *Value*. *Hue* is the component that specifies the angle on the RGB circle with red, green and blue located on 0, 120 and 240 degrees, respectively. *Saturation* defines the amount of colour and it takes values between 0 and 100%. Colours with 100% *saturation* are pure colours and are located in the bounds of the cylinder while colors with 0% saturation are located in the centre and are grayscale. Lastly, *Value* defines the amount of black contained in the colour and takes values between 0 and 100%. Colours with 0% *value* are total black. In contrast, value of 100% denotes that there is no black in the colour.



(a) Square Error with all images (b) Square Error after removing images

Figure 11: Squared error for the photometric stereo on Yale Face images.



(a) Albedo (b) Height map row major (c) Height map column major (d) Height map average (e) Height map average for less images

Figure 12: Albedo and height maps for the photometric stereo of Yale face.

2.2.5 YCbCr

The YCbCr color space consists of three channels, namely Y, Cr and Cb. The main purpose is to separate the luminance and chrominance components into different channels. The Y channel indicates the Luminance (brightness) of the image and is obtained from RGB after gamma correction. Cb (blue-difference) and Cr (red-difference) are both chromatic channels obtained by $Cb = B–Y$ and $Cr = R–Y$, respectively. These channels indicate how far the blue (B) and the red (R) channel is from Y, in each case. This colourspace is mostly used in compression (of Cr and Cb components) for TV Transmission.

2.2.6 Grayscale

Gray-scale colourspace consists of only one channel, which indicates the intensity value of each pixel It is used in cases where colour information is unnecessary, leading also in compression since we are dealing with only one channel instead of three. The built-in method used in OpenCV to convert images from RGB to grayscale is:

$$Y = 0.299R + 0.587G + 0.144B$$

2.2.7 CMYK

The CMYK colourspace is a subtractive colour mixing model, composed of three colours, namely Cyan (C) Magenta (M) and Yellow (Y). In contrast with the RGB model, which is applied in mediums that can emit light, such as monitors, CMYK is used for mediums like paper where light can be emitted but only reflected off of i. In a subtractive model to produce a colour we add portions of cyan, magenta and yellow, in order to subtract the amount of the reflected light which is . More specifically, if none of these colours is added the produced colour is white and if all these colours are at their maximum value, all the light is absorbed and thus producing the colour black. To transform

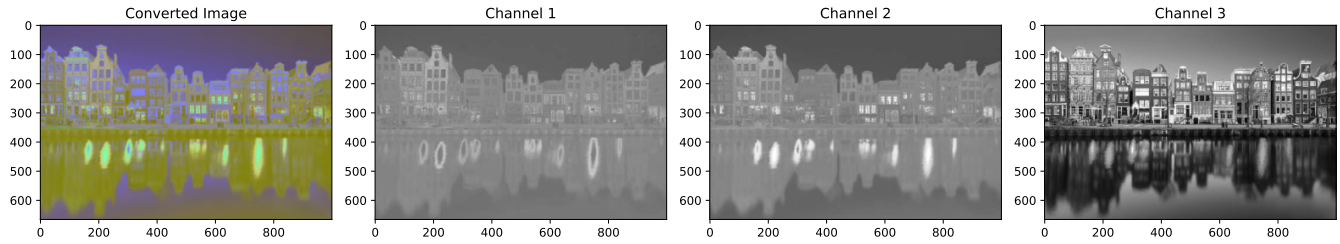


Figure 13: The conversion of the original image from the RGB colourspace to the opponent colourspace. The resulting image is shown along with its channels separated. Channels 1, 2, and 3 correspond to the values of the normalized red - green, blue - yellow and luminance channels respectively.

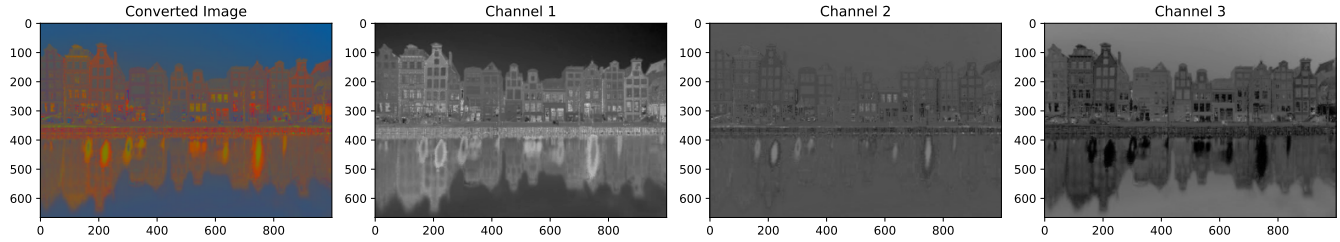


Figure 14: The conversion of the original image from the RGB colourspace to the normalized RGB. The resulting image is shown along with its channels separated. Channels 1, 2, and 3 correspond to the values of normalized Red, Green and Blue channels respectively.

the RGB colourspace values into CMYK values we need the following equations:

$$C = \frac{1-R-K}{1-K},\ M = \frac{1-G-K}{1-K},\ Y = \frac{1-B-K}{1-K},\ where\ \ K = 1 - max(R, G, B).$$

One of its application as mentioned is on printing machines.

3 Intrinsic Image Decomposition

Other Intrinsic Components: Several interesting methods for decomposing image have been proposed. The most widespread is the process of separating an image into its albedo and shading components as we do in this laboratory exercise. In other methods, noise removal from images (e.g. weather conditions, lighting), wavelets or even addition of other parameters in the basic case (albedo + shading) which mentioned above are used. But what has intrigued us is the use of depth information and the structure texture separation in combination with Surface Normals. Regarding the first work [1] (use of depth information), the commoditization of RGB-D imaging sensors provides an opportunity to re-examine the intrinsic image decomposition problem and a chance to obtain highly accurate decompositions of complex scenes without human assistance. In particular, the decomposition should approximately satisfy the equivalence $I_p = A_p S_p$, where the product $A_p S_p$ is performed separately in each color channel. Specifically, in this work, Chen et. al factorize I into four component images: an albedo image A, a direct irradiance image D, an indirect irradiance image N, and an illumination color image C. The albedo image A encodes the Lambertian reflectance of surfaces in the scene. The direct irradiance image D encodes the irradiance that each point in the scene would have received had there been no other objects that occlude or reflect the radiant flux emitted by the illuminants. The image D is thus intended to represent the direct irradiance that is modeled by local shading algorithms in computer graphics, which do not take shadows or inter-reflections into account. The indirect irradiance image N is the complement of D, intended

[1] Q. Chen and V. Koltun, "A Simple Model for Intrinsic Image Decomposition with Depth Cues," 2013 IEEE International Conference on Computer Vision, 2013, pp. 241-248, doi: 10.1109/ICCV.2013.37.

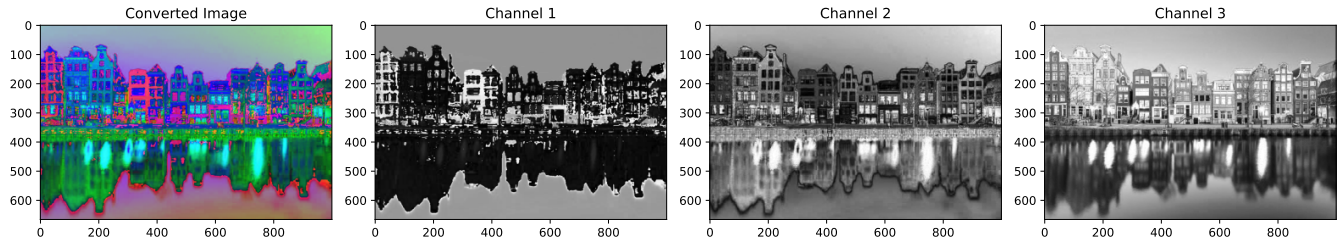


Figure 15: The conversion of the original image from the RGB colourspace to the HSV colourspace. The resulting image is shown along with its channels separated. Channels 1, 2, and 3 correspond to the values of Hue, Saturation and Value respectively.

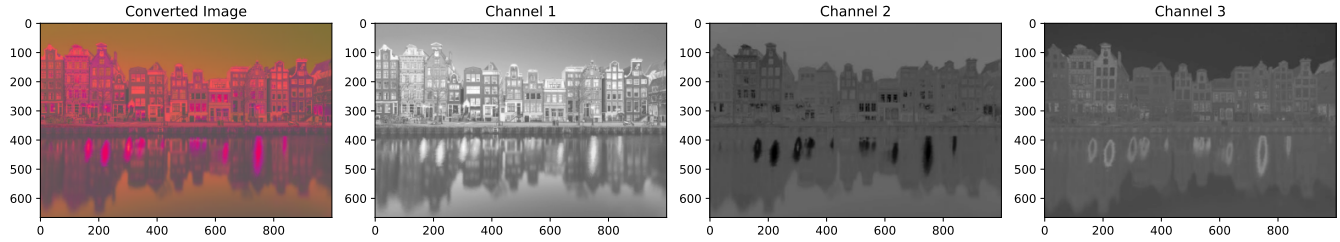


Figure 16: The conversion of the original image from the RGB colourspace to the YCbCr colourspace. The resulting image is shown along with its channels separated. Channels 1, 2, and 3 correspond to the values of Luminance, Blue difference (Cb) and Red difference (Cr) channels respectively.

to absorb the contribution of shadows and indirect illumination. For every pixel p, their factorization approximately satisfies $I_p = A_p D_p N_p C_p$. On the second approach [2], they define a novel model for intrinsic image decomposition as: $I(p) = B(p)T(p) = S_B(p)R_B(p)T(p)$, where $B(p) = S_B(p)R_B(p)$ is a base layer, and $S_B(p), R_B(p)$ and $T(p)$ are shading, reflectance and texture components at a pixel p, respectively. Note that S_B and R_B are different from S and R in basic equation that we use in our lab assignment as S_B and R_B contain no textures. In this work they also propose a novel constraint based on surface normal vectors obtained from an RGB-D image. The overall process of their method, consists of two steps with the first being the decomposition of an RGB input image into a base layer B and a texture layer T while the second being the further decomposition of the base letter B into a reflectance layer R_B and a shading layer S_B based on the surface normal constraint and other simple constraints. This work explicitly models textures for intrinsic image decomposition to avoid confusion on the smoothness property caused by textures.

Synthetic Images Since the ground truth reflectance, R, and shading, S, images for a synthetic dataset are available, we can provide a direct supervision for the network by minimizing the mean squared error (MSE) between the ground truth and the estimated reflectance and shading layers. The instrict components of real-world images need to be estimated by other models such as CNNs. The errors of these models could prevent us from assessing the accuracy of our task. At the same time, there is a lot of research in the direction of creating synthetic data that resembles real data so that it can be reduced to gap between the distribution of synthetic and real-world data.

Image Formation To reconstruct the original "ball.png" image from its intrinsics, initially we normalize the values to [0,1], using our normalize function, to prevent overflow during the multiplication. We perform element wise multiplication and then we re-normalize the values back to the range [0,255], using cv2.normalize function. At this point, is useful to note that the reconstruction image is a little bit lighter from the original, due to normalizations and same with the original one if we apply the same normalization on it.

[2] Jeon, Junho & Cho, Sunghyun & Tong, Xin & Lee, Seungyong. (2014). Intrinsic Image Decomposition Using Structure-Texture Separation and Surface Normals. 218-233.

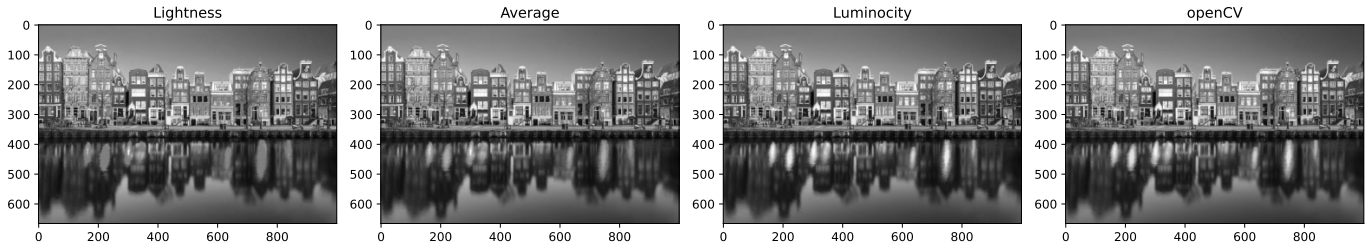


Figure 17: This figure illustrates the conversion of the original image from RGB to grayscale using three different methods and the built-in method from *openCV*.

Figure 18: Reconstruction

Recoloring To find out the true material colour we apply the np.unique function in each 2d channel array of albedo. Then we set to zero all red and blue elements and set to pure green (255) all the green elements in order to calculate the green albedo. Finally we apply the same algorithm with the previous query to recolor the ball. Although we have recoloured the object with pure green, the reconstructed images do not seem to display those pure colors and thus the colour distributions over the object do not appear uniform. This uniformity is due to the uniformity of the shading as shown by the equation $I(\vec{x}) = R(\vec{x}) \times S(\vec{x})$.

Figure 19: Recoloring

4 Colour Constancy

4.1 Grey-World Algorithm

The Grey-World algorithm is a method of Color Constancy which assumes that the average colour in a scene should be achromatic (grey, [128, 128, 128]). To apply this algorithm, we need to multiply each channel (R, G, B) by the average colour of the whole image, in this case by 128 (grey), and then divide by the mean of each colour separately.

We compute:

$$(R_{GW}, G_{GW}, B_{GW}) \rightarrow (\frac{R \times 128}{mean(R)}, \frac{G \times 128}{mean(G)}, \frac{B \times 128}{mean(B)})$$

After running the algorithm, we get the following, resulting in a more natural image without the reddish colour tones:

The Grey-World algorithm can fail to show the natural colours of the image if the original image has a large dominant colour patch. That is because the algorithm works on the assumption that there is a good distribution of

Figure 20: Results after applying the GW algorithm

Figure 21: Results of GW algorithm failure

colour in the image, that way the image's average colour can be achromatic. For example, if we have an image which has mostly green tones, the algorithm will fail as shown in the following figure:

Some colour constancy algorithms are:

- White patch The White-Patch Retinex (WP) algorithm was proposed by Land and McCann. This algorithm considers the highest value in each color channel as the white representation for the image. Computationally, such values are found from the maximum intensity in each channel.
- Scale max This algorithm is very similar to the Grey-World algorithm, except for the fact that the mean of each channel is replaced by the maximum intensity of each channel. The assumption now is that there is at least one white pixel in the image (white, [255, 255, 255]).
- Von Kries model The von Kries coefficient rule rests on the assumption that color constancy is achieved by individually adapting the gains of the three cone responses, the gains depending on the sensory context, that is, the color history and surround. The von Kries coefficient law in color adaptation describes the relationship between the illuminant and the human visual system sensitivity. The law accounts for the approximate color constancy in the human visual system. It compensates for the illumination change using a purely diagonal scaling of the cone absorptions.