Computer Vision - Homework 1

Ioannis Gatopoulos 12141666, Philipp Ollendorff 11734078,
Konstantin Todorov 12402559, Vincent Roest 10904816

February 20, 2019

Introduction

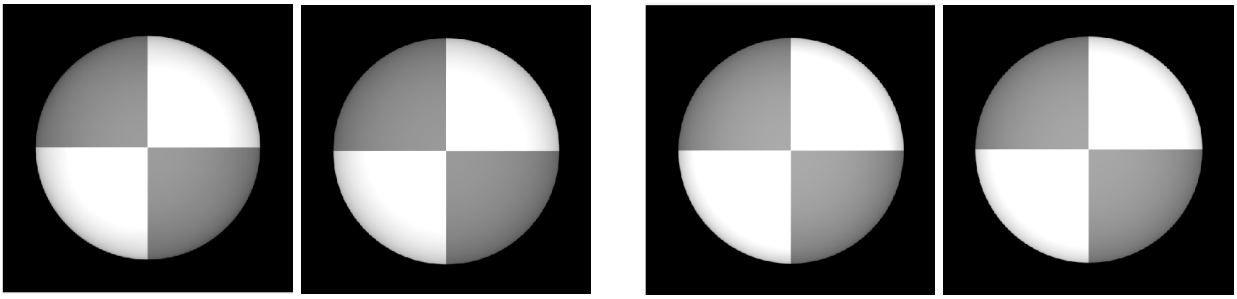
In this assignment we implement common algorithms used in Computer Vision, highlight differences and showcase their effects on example images. Specifically we look at *photometric stereo*, which is a method for recovering a three-dimensional representation from differently illuminated, two-dimensional image data [Forsyth and Ponce, 2002]. We present images with both simple and complex shadow contours as well as gray and colored images. Next we venture into different *color spaces*, compare their theoretical underpinnings and point out some real use cases for different representations. We are further investigating the *intrinsic decomposition of images* into two main components: The reflectance and shading. The theory is then applied on the problem of material recoloring with promising results on a simple ball object. Lastly, we adjust for different illuminants in light sources to achieve *color constancy* - the appearance of an image to be taken under white light. For this issue we implement and analyse the Grey-World algorithm while using the von Kries model.

1 Photometric Stereo

1.1 Estimating Albedo and Surface Normal

1.1.1 Firstly, we did use a least squares implementation for this problem, by calculating the Moore-Penrose inverse. Since our matrices were not squared, the latter solved the system in the least square error perspective, returning a solution with minimum Euclidean norm. We expect to see the albedo uniformly distributed over the input image. Namely, the reflectance is constant across the 2 colors in the circle. However, we can see in Figure 1a that using 5 images alone does not yield this perfectly smoothed and uniform albedo. Instead, we can see a slight shadow ring around the edges.

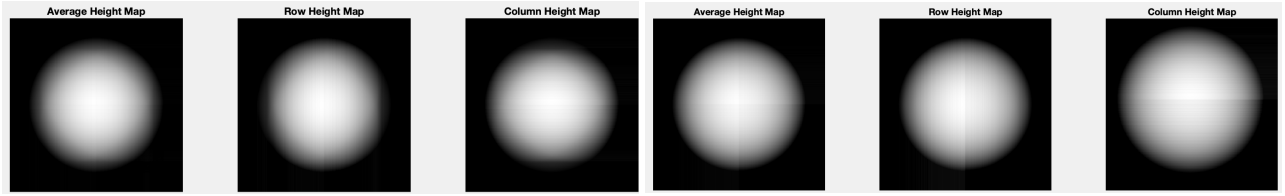
1.1.2 The book lists that in principle, the minimum number of images required is 3. There is exactly one linear system per point in the image, for which this system should be solved to recover the surface. The surface vector g is three-dimensional, and using 3 images would make the least-squares solution completely deterministic. This means that we would also not have a residual error from the least-square method that we could use to check the measurements. It should also be noticed that the more complex an object is, i.e. more height-differences or



(a) Using 5 images.

(b) Using 25 images.

Figure 1: Resulting Albedo images. Shadow tricks are applied on the right side images.



(a) Using 5 images. (b) Using 25 images.

Figure 2: Normalized Surface Height Maps of 3 Methods

curvy objects, one should consider more images to obtain satisfactory results. Figure 1b shows that the main difference with using more images is that the albedo appears to be more uniform, and hence looks brighter, and that the shadow ring has almost disappeared. Since the implementation runs very quickly as it is, we opted for processing all images at the same time, which also seemed the most straightforward implementation.

1.1.3 The problem that is tackled by the so-called shadow trick is the detrimental effect of pixels completely covered by a shadow to the least-squares solution. These points can be considered outliers to the least-squares solution because here the values of $i(x, y)$ do not depend on $g(x, y)$ (the surface), because the relevant elements of the matrix i are zero in shadowy points. Consequently, it is an outlier to the least-square solution. From this also follows that the detrimental effect of a single point in I that is covered by shadow is mitigated as the number of measurements for that point increases; more images means less negative impact of one outlying point. The shadow trick constitutes of constructing a diagonal matrix $I(x, y)$, whose entries are those of the measurements. If we then multiply both sides of the least-square equation by this matrix, any equations from points that are in shadow are zeroed out. We can indeed observe that with 5 images (Figure 1a) there is a noticeable effect, whereas for 25 images (Figure 1b) the effect of this trick is negligible.

1.2 Test of Integrability

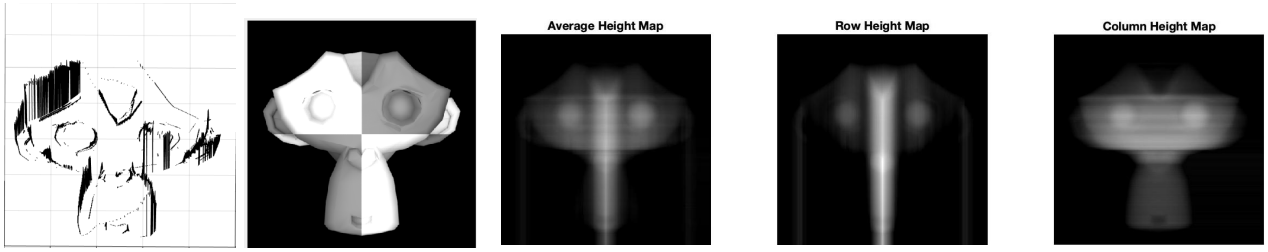
The error that results from this test is only meaningful at the immediate transition from the edge of the circle to the black surroundings. This high contrast makes the approximation difficult. However, the magnitude of the errors decreases as the number of images increases. This is not clearly visible in the ring plots from the exercise, but the number of outliers for 5 images is actually larger than that for 25 images (without the shadow trick: 1799 vs 1613). In an ideal scenario, images taken from each angle with infinitesimally small increments would completely eliminate this residual error.

1.3 Shape by Integration

Figure 2 shows the height maps constructed through the 3 different methods. Here, the greyness represents the normalized height; every pixel value is divided by the maximum pixel value of both the row and column construction paths. Quite clearly (especially in Figure 2b), one can distinguish the difference between the row and column integration. The resulting height maps show a clear top-bottom and left-right separation in the column and row maps respectively, and their merge in the average construction, which is completely logical. Averaging over both orders of path integration results in a better, smoother image. Again, the larger number of images used shows a more clear-cut result; it is reduced in vagueness around the edges and the separation is much more noticeable. There is also more contrast between individual paths.

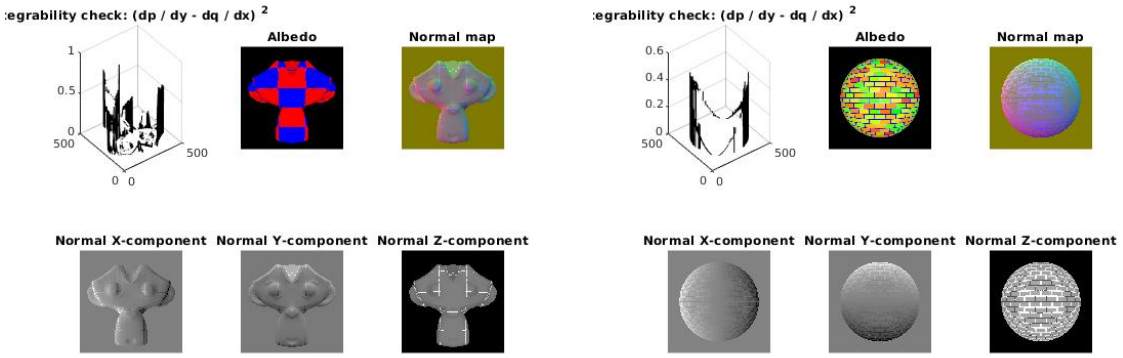
1.4 Experiments with different objects

1.4.1 Figure 3a shows the albedo and the errors of the integrability test. There are much more images in this dataset (121) and it can be seen that the albedo reconstruction is rather effective. However, compared to the spherical model, this model has much more depth and different shapes. This entails that the errors are also much more spread out over the entire area, as opposed to only around the outer edges. It also appears that the errors of the reconstruction (left image in Figure 3a) are larger on the left upper side. Another example is the area around the eyes, which is almost always shadowed from every angle. And indeed, it seems that the direct transition from the lighter areas of the monkey to the black background is less easily approximated



(a) Errors and Albedo (b) Height Maps

Figure 3: Results for the MonkeyGray Model



(a) MonkeyColor (b) SphereColor

Figure 4: Test with 3-channel input

than the other sides of the monkey, and the same goes for the bottom right side, but here there is a bit more contour than on the top left side, which makes the integration a bit less difficult. Decreasing the number of images would negatively impact the albedo reconstruction, as it would provide even less varied datapoints for the "difficult" pixels (such as those on the upper right). The height maps are shown in Figure 3b, which are also very indicative of the differences in integration approaches.

1.4.2 In order to make our implementation work on 3-channel RGB inputs we simply ran the 1-channel algorithm once for each channel. The resulting temporary albedos were concatenated to form one final albedo, while we averaged over the three normals to create one 3-channel normal. The results can be seen in Figure 4 and 5. A zero pixel can be an issue with this technique, as it results in NaN values for some of the albedos, which are then impossible to use numerically in subsequent calculations such as averaging or concatenation. It is quite simple to overcome this by setting any value that is NaN to 0.

1.4.3 Disabling the shadow trick was the first thing on the to-do list, which could be easily done in the script by passing the value "false" to the estimate_alb_nrm function. This was priority number one, because some parts of the face images are actually black, which means they are black regardless of the position of the light source. As explained in Section 1.1.3, the shadow trick removes the contribution of these black images through a pixel-value multiplication. As a result, the least-square solution will be either undetermined or containing errors. From the height maps in Figure 6, it can be seen that the shape-from-shading method for this dataset is not overly accurate. The key issue in this dataset is that the images in the Face dataset violate the basic assumption that the surface is Lambertian, which is lighted by a single distant light source. However, faces exhibit some intrinsic specular highlights, which simply cannot be modelled effectively through this method. There are two easily identifiable problems in the dataset. One is the previously mentioned specular highlights, which is exemplified by the pictures in Figure 7a. Another problem is the fact that although the light comes from behind a face, it still manages to cast itself on the faces. By our assumptions, this should not be the case. However, in the photography room there was probably some reflection that caused the illumination of the front of the face; examples are shown in Figure 7b. We added a script that removed these images of azimuths > 90 or < -90, and the resulting height maps are shown in Figure 6. Although it looks like there are much more spikes in the row part, the resulting average picture (spikes are cancelled out) shows a better resistance against

(a) MonkeyColor (b) SphereColor

Figure 5: Test with 3-channel input

that unnatural chin blob that was present in the uncorrected version above. However, this is just a very crude removal of images that might have been very informative regardless of the light source being behind the subject.

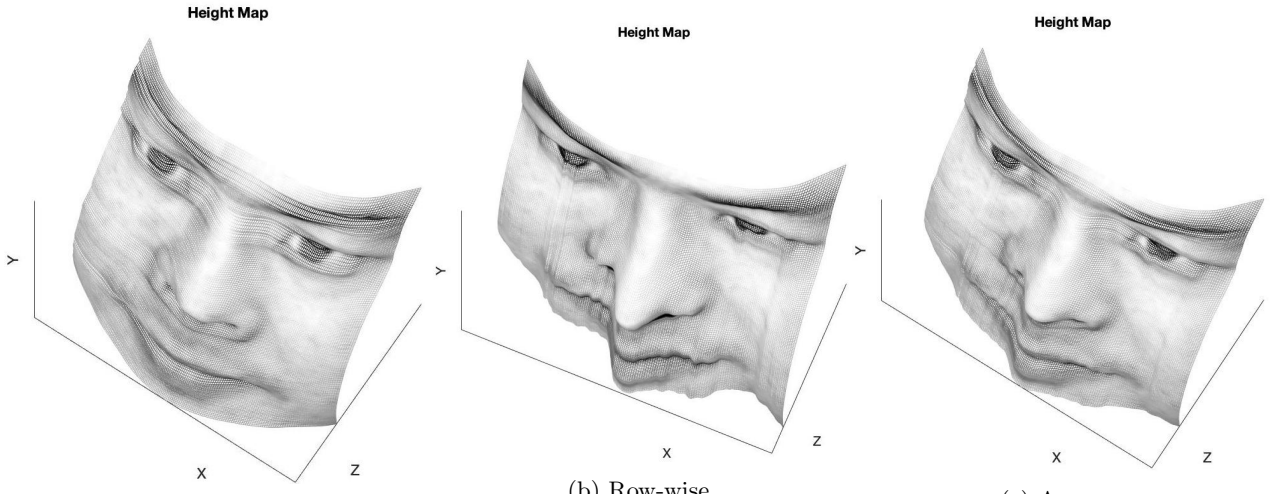
2 Color Spaces

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

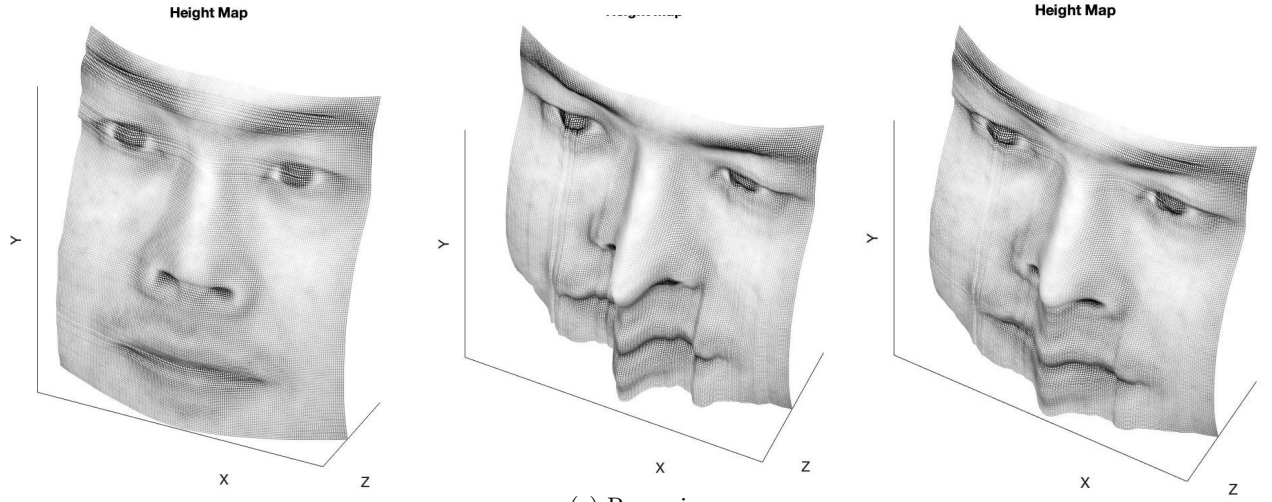
The human eye uses a *trichromic* color model which encodes colors with *red*, *green* and *blue* components (sensors). Thus we construct cameras that can capture the world in a similar way as to how humans experience it. There are other color systems that are three-dimensional (HSV, YCbCr, etc.), but we are not using them, not because of intrinsic physical properties of color, but because of the very way the visual system of a human works. Therefore the light signals have to be encoded in these terms. It is important that we note that the RGB is an *additive color system* because when the three colors are combined in equal amounts they form white. Digital cameras use an image sensor which is essentially a photo sensitive layer that measures how much light has hit its certain position. This is arranged in a grid form, where each position represents a pixels of the image. As doing this, we do not know the proportion of red, green and blue colors, filters are installed (physical elements that intersects certain wavelength) of either red, green or blue on top of each of these pixels. A famous technique - color filter array (CFA) for arranging RGB color filters on a square grid of photosensors is the *Bayer Filter*; every 2x2 pixel matrix has 2 green filters, 1 red and 1 blue. This makes sense as the human eyes are more sensitive to green and can disguise luminance (brightness) with much more intensity in the green channel. Because we want to give each pixel a value of all three colors (to get the RGB format) but we only have one color value per pixel, we interpolate their value by looking at the nearby pixels (the process is called *demosaicing*). A simple approach would be taking a 2x2 pixel grid where the red and blue color would be the amount of the pixels that have red and blue filter and the green one would be the average between the two green measurements.

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

Opponent Color Space Here the RGB signals are transformed to three channels; one hypothesized achromatic (white-black) channel O_3, and two opponent colour channels, O_1 (red-green) and O_2 (yellow-blue). The motivation for this color space was the separate luminance from chrominance and the assumption that red is the 'opposite' colour to green and likewise blue must be 'opposite' to yellow. Opponent colour spaces have found numerous applications in different fields of image processing: real-time colour edge detection, interactive computer graphics and detection of colour contrast.

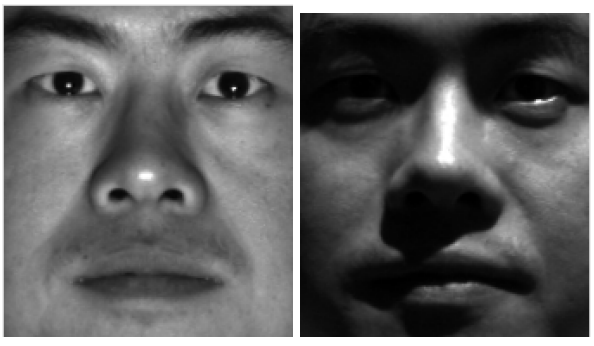


(a) Column-wise (b) Row-wise (c) Average

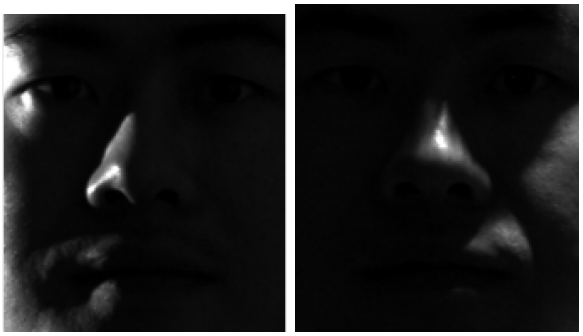


(d) Column-wise (e) Row-wise (f) Average

Figure 6: Height Maps for the Face Data with the uncorrected maps on the top, and the corrected ones at the bottom



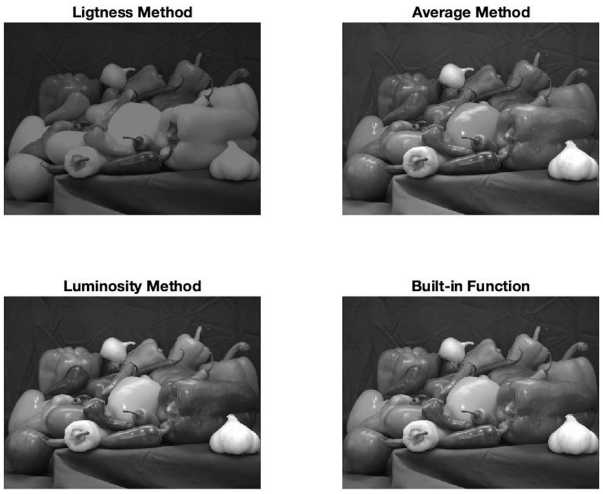
(a) Two Images with azimuth 0 degrees and positive elevation (20 & 35) that exhibit specular noses from a light source from the front



(b) Two Images with light sources from behind (azimuth < -90 or > 90 degrees).

Figure 7: Problematic Images. A positive azimuth means that the light source was from the right of the subject, negative on the left. Elevation > 0 means that the source of light was above the horizon, < 0 below it.

(a) Original image



(b) Different Grayscale Color Space

Figure 8: Original image (left) and Grayscale color spaces (right)

Normalized RGB Color Space Converting an RGB image into a normalized RGB removes the effect of any intensity variations. It is used to get rid of distortions caused by lights and shadows in an image. From the Figure 8b, by looking at the background of the image, we can observe that the background is one solid colour and the white onions (which have high light reflection rate) now don't stand out as much.

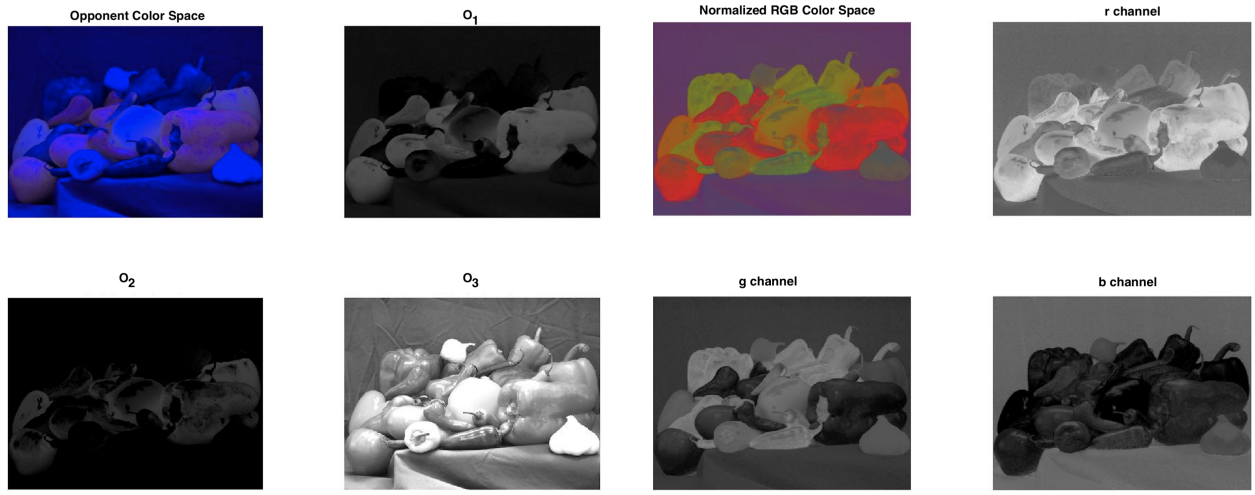
HSV Color Space Its components are: *Hue*, the dominant raw color as perceived by an observer, *Saturation*, the amount of white light mixed with Hue (how concentrated the dominant color is) and *Value*, describes the brightness or intensity of the color (the closest to 0, the more similar to black, and the closest to 1, the more similar to the raw full color). Expressing color in this manner allows us to adjust the color of an image, without altering the brightness, which can not be done via RGB.

YCbCr Color Space Its components are: *Y'*, the brightness (luma), *Cb*, which is blue minus luma ($B - Y$) and *Cr*, which describes red minus luma ($R - Y$). YCbCr is another way of looking at color, where color is divided into one brightness channel (Y') and two color difference signals. It is used for TV broadcasting, where a stream of data is sent to a television. Because in this case the compression of data is crucial, YCbCr is down-sampling the color components but not the luminous component, which makes no difference for human vision.

Grayscale The objects with the *lightness* method, are becoming smoother, loosing their illumination properties; the brightness level seems to be constant. The *average* and the *luminosity* methods are very similar, but the latter seems to contribute more brightness to the colors that can reflect light and less to those which can absorb (red reflects a lesser proportion of light than green, and we can observe from the figure that the green peppers appear brighter than the red ones). So it forms a weighted average to account for human perception. Lastly, we can see that the Built-in Matlab function uses the luminosity method.

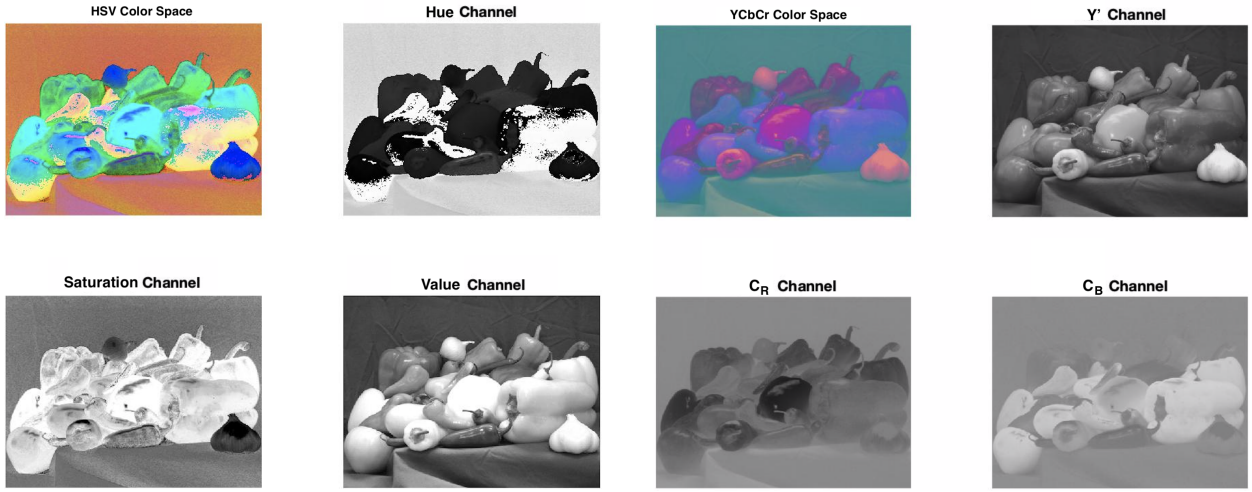
Find one more color space from the literature and simply explain its properties and give a use case.

Cyan Magenta Yellow Black (CMYK) In a medium like screens, we do additive color mixing (RGB), where mixing different proportions of the colors result in different colors (white is the full value from each of the three primary colors of light and black would be the absence of all light). In contrast, printers represent colors the other way around, by subtracting color mixing to project images on a white paper. This is because paper can not produce light (in contrast to the screen) and therefore, it has to have light reflected off of it. In CMYK, a subtractive model, the idea is when all these colors are added together (cyan, magenta and yellow), the amount of light that is reflected back is subtracted. Therefore, if all the CMYK values are zero, which represents white, the amount of color that will come out of the printer is zero. The formulas for transitioning from RGB to CMYK are given below. Note that cyan, magenta and yellow are the opposites of red, green and



(a) Opponent Color Space

(b) Normalized RGB (rgb) Color Space



(c) HSV Color Space

(d) YCbCr Color Space

Figure 9: Different color spaces for an image representation, followed by their color channels representations

blue respectively.

$$\begin{pmatrix} C \\ M \\ Y \end{pmatrix} = \begin{pmatrix} \frac{1-R-K}{1-K} \\ \frac{1-G-K}{1-K} \\ \frac{1-B-K}{1-K} \end{pmatrix} \qquad where \quad K = 1 - \max(R, G, B)$$

3 Intrinsic Image Decomposition

3.1 What components can an image be decomposed other than albedo and shading?

There are many ways in which one image can be decomposed. For example, we can have an image which is corrupted by noise. We can then decompose it into signal parts and noise parts. This method is also known as "denoising" and helps us when we want to clear out a noisy image. ([Ghimpeeanu et al., 2016])

We can also decompose an image using wavelets. Those are used to represent an image in multiple resolutions which are then contrasted. Useful information can be gathered and used to develop interesting image representations leading to detection of features and anomalies such as edges or corners for example. ([Treil et al., 2015], [Xizhi, 2008])

Additionally, wavelets can be used for denoising as well.

3.2 Why use synthetic images?

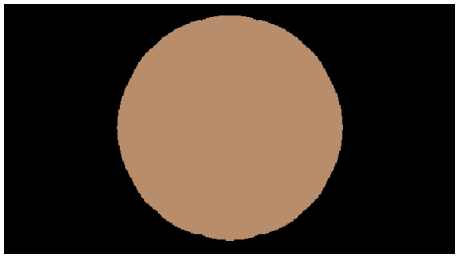
Synthetic images are very convenient for scientific projects. They allow us to test exactly what we need in our algorithm. For example, if we are developing an edge detecting algorithm we can easily create a synthetic image which contains exactly one edge which we can then observe using our algorithm. There will be no interference from e.g. multiple light sources, partial obstruction of the area of interest, or otherwise from parts of the image.

Real images are very hard to be taken exactly as we need them to be. Moreover, in the present days, synthetic images are beginning to look more and more real and exactly because they require far less setup to be configured for the current experiment, ultimately, they are preferred by scientists and researchers.

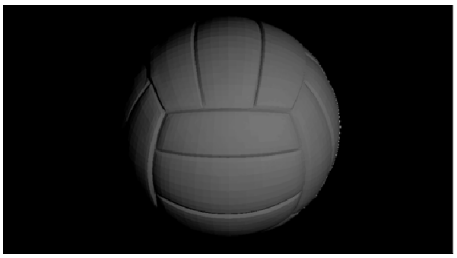
3.3 Image formation

We can see that by combining the images of the albedo and the shadow, we can reconstruct the original image. You can see the result in Figure 10d

(a) Original image



(b) Original albedo image



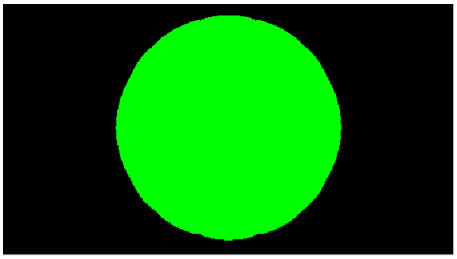
(c) Original shadow image

(d) Reconstructed image

Figure 10: Image decomposition and reconstruction

3.4 Recoloring

For the recoloring, we first needed to extract the pure material color. We did this by simply taking the first pixel from the albedo image which was not pure black. After this, we recolor the albedo image by substituting all pixels which are of this material color with our new [0, 255, 0] green color. Finally, we combine the albedo and shadow image the same way as in the last step. The recolored albedo image and the final result can be seen in Figure 11



(a) Recolored albedo image

(b) Recolored reconstructed image

Figure 11: Image recoloring

3.5 Non-uniform recoloring

We can see from the images in Figure 11a that our recolored albedo is indeed pure green. But what we also need to notice in Figure 10c is that the original shadow picture is not one pure color like the albedo. It contains different layers of shadowing and more precisely - different variations of blackness. Therefore when merging the recolored albedo picture with the shadow, we can never receive pure recoloring.

4 Color Constancy

4.1 Color correction using the Grey-World algorithm

We created a function to apply color correction by using the Grey-World algorithm. First we determine the mean color value for each RGB channel and calculate three multipliers to adjust each channel to grey $(128, 128, 128)$. Then we multiply each pixel's color value with the respective multiplier. The result is an achromatic scene.

(a) original awb.jpg

(b) After applying Grey-World

Figure 12: Removal of reddish color cast.

4.2 Where does Grey-World fail?

The Grey-World algorithm assumes that any image's average color should be achromatic. This assumption is flawed in some cases, especially when the image naturally consists of mostly one color. In this case the algorithm is skewed towards the intensity of that color. In Figure: 13 we present three examples to highlight this failure in color correction: An image of strawberries, an image of a wave and an image of a forest. Each image has been chosen to reflect a natural, correct tendency towards one color. We can clearly observe an unnatural change after Grey-World correction. This is most pronounced in the forest image, where the ground becomes tinted in surreal violet. Strawberries receive a neon-like color adjustment, which is due to the introduction of more blue. Lastly, after applying Grey-World the splashing wave is covered in red fog.

4.3 Retinex as an alternative to Grey-World

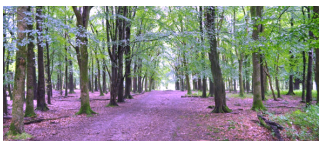
Retinex algorithms are another method to achieve color constancy in images. They were first introduced by Land and McCann [Land and McCann, 1971]. Several variants have since been proposed, but the general idea is as follows: We determine the maximal value for each color in the entire image. This way we obtain the triplet $(R_{max}, G_{max}, B_{max})$ which describes the color of our light source. Now we can transform every pixel i in the original: $\left(\frac{R_i}{R_{max}} \frac{G_i}{G_{max}} \frac{B_i}{B_{max}}\right)$. We have now adjusted each pixel's color based on the existing maximal color values. This should account for different levels of brightness and does not assume an achromatic color average.

(a) Original strawberries

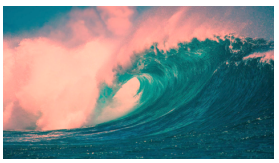
(b) Original forest

(c) Original wave

(d) Color corrected strawberries



(e) Color corrected forest



(f) Color corrected wave

Figure 13: Three examples highlighting color correction failures in images with high focus on single colors.

Conclusion

In this assignment we have learned the basics of image manipulation: Recreation of three-dimensional surfaces from multiple images under different lighting conditions, common color representations, how to decompose images into their components and account for varying light sources. Unsurprisingly our results include an increased effectiveness for the photometric stereo algorithm when used with more data, as well as a decreased effectiveness when used on images with more complex shadow effects. Decomposing images into their albedo and shading components has shown very promising results, even in the case of recoloring to arbitrary values. However, this was not yet tested on images of more complicated shapes than a spherical ball. The Grey-World algorithm works conveniently on images which fulfill the underlying assumption that the mean pixel value must be grey. We have proved that the algorithm fails in cases where this assumption does not hold, e.g. images of a forest, which heavily favor the color green.

References

[Forsyth and Ponce, 2002] Forsyth and Ponce (2002). In *Computer Vision: A Modern Approach*.

[Ghimpeeanu et al., 2016] Ghimpeeanu, G., Batard, T., Bertalmo, M., and Levine, S. (2016). A decomposition framework for image denoising algorithms. *IEEE Transactions on Image Processing*, 25(1):388–399.

[Land and McCann, 1971] Land, E. and McCann, J. (1971). In *Lightness and Retinex Theory*.

[Treil et al., 2015] Treil, N., Mallat, S., and Bajcsy, R. (2015). Image wavelet decomposition and applications.

[Xizhi, 2008] Xizhi, Z. (2008). The application of wavelet transform in digital image processing. In *2008 International Conference on MultiMedia and Information Technology*, pages 326–329.