Computer Vision: Photometric Stereo & Color

Sylke Gosen, Victor Zuanazzi, Simon Passenheim, David Speck

February 20, 2019

Introduction

In this assignment, we explore how we can use knowledge about how light and the human visual system work to dissect 2D images and extract meaningful information about color and shape. First, we estimate depth by using multiple images of the same object under different illumination conditions. So-called photometric stereo. This technique has several nice properties which binocular stereo, reconstructing depth using multiple viewpoints, does not have. Only set up with just one camera can be used, and it also does not require the matching of points in both images.

Next, we discuss how color is stored digitally using different color spaces and how these different representations are useful for various different applications ranging from humans editing images to compression and printing.

Following that we see how an image can be decomposed into albedo and shading components which allows us to manipulate them individually. Being able to do this sort of decomposition well is very powerful because it gives access to fundamental properties of the world captured by the image rather than just the pixel values captured by the camera.

Finally, we experiment with color constancy. The goal for color constancy is to ensure that colors of objects appear the same under different lighting conditions, something which humans have built in. It's worth noting that this is an ill-posed problem and there are no algorithms that can accurately infer the lighting conditions from an image. However, by making certain assumptions such as the *gray world*, it is possible to create algorithms that can perform reasonably well.

1 Photometric Stereo

1.1 Estimate Albedo and Surface Normal

Looking at the images of the gray sphere we would expect the albedo values to be of two different gray values shared by opposite quadrants of the sphere. Different to the expectation, the estimated albedo values decrease towards the edge of the visible sphere. This effect is weaker when the estimation is based on the richer image set *SphereGray25* instead of *SphereGray5*. This happens because the algorithm cannot distinguish, for some regions of the image, whether the pixel value comes from albedo or from the shape.

To properly estimate all 3 axes of the normals and the albedo value, each pixel has to be illuminated by at least 3 different light sources. To see significant changes we compare directly the 5 image data set to the 25 image data set. In figure 1 we see that the albedo estimates already look reasonable when using 5 images but the normal map shows clear shadows. The shadow trick helps to smooth out the normal estimates. This trick zeroes out equations in the linear system to ensure that black pixels do not contribute to the normal estimate.

The alternative to the shadow trick, more images with additional illumination angles improve the estimates as in figure 1. Given more images, the shadow trick does not bring noticeable improvements and it can be dropped for faster computation.

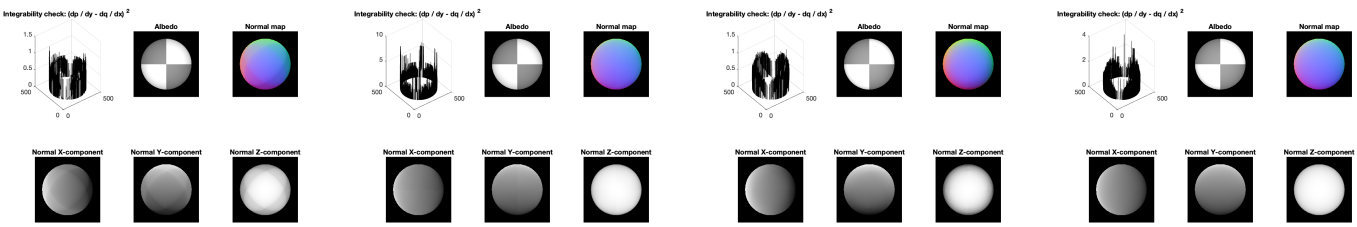


Figure 1: Albedo and normal esitmates, from left to right: 5 images, 5 images + shadow trick, 25 images, 25 images + shadow trick

(a) Threshold: 0.002 Outliers: 2022 (b) Threshold: 0.004 Outliers: 1028 (c) Threshold: 0.008 Outliers: 476 (d) Threshold: 0.016 Outliers: 205 (e) Threshold: 0.032 Outliers: 67

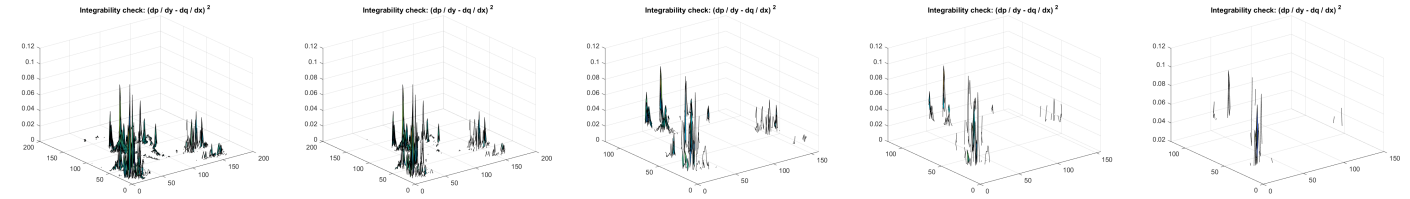


Figure 2: Number of outliers as the threshold is increased.Face data set was used.

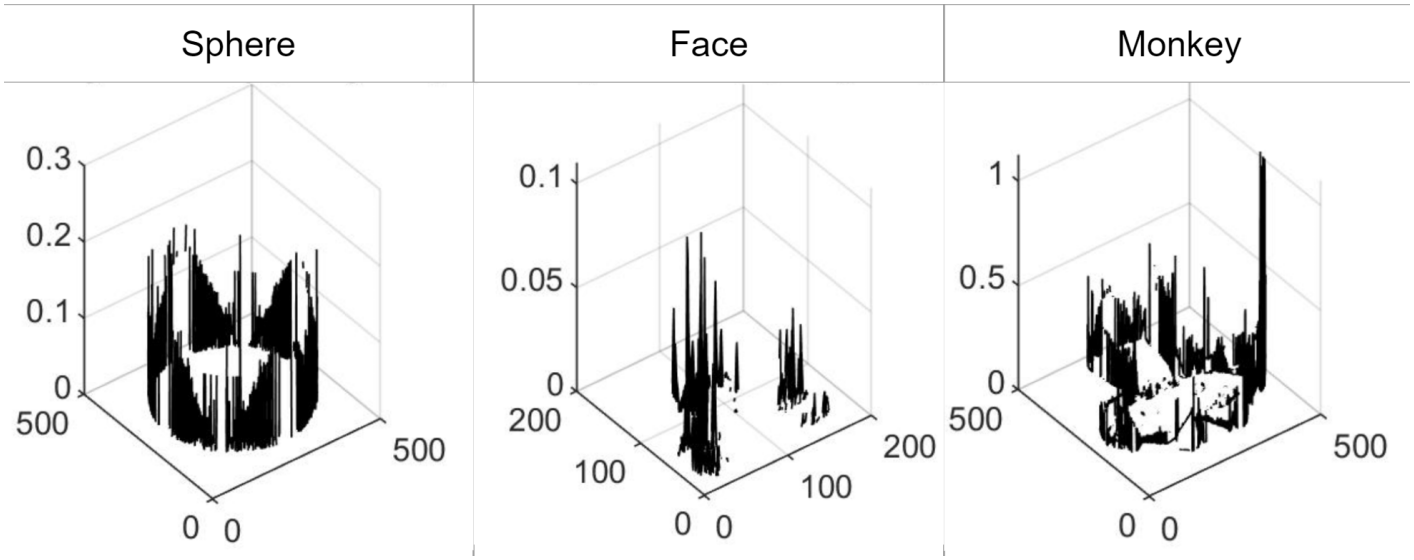


Figure 3: Errors are concentrated on the edges. Threshold = 0.005

1.2 Test of Integrability

First, we experiment with different threshold levels to understand how this value impacts the integrability check. As expected, there are fewer outliers when the threshold is increased. The figure 2 shows the value of the outliers for different thresholds for the image set of faces. For the rest of the experiments we kept the original threshold of 0.005, 3 shows the outliers for different sets of images.

Most outliers are located on edges of the image, figure 3. That is because those regions feature a more abrupt change in brightness. Those regions also show the highest variance between pictures with different light sources.

The number of outliers declines asymptotically approaching a lower bound as the number of images increases (figure 4a). The complexity of the images also plays an important role. The *Sphere* is the simplest image, and it does not have many outliers. The *Monkey* images have a lot of details, which requires a large number of images in order to recover the normal and the albedo with a relatively low number of outliers.

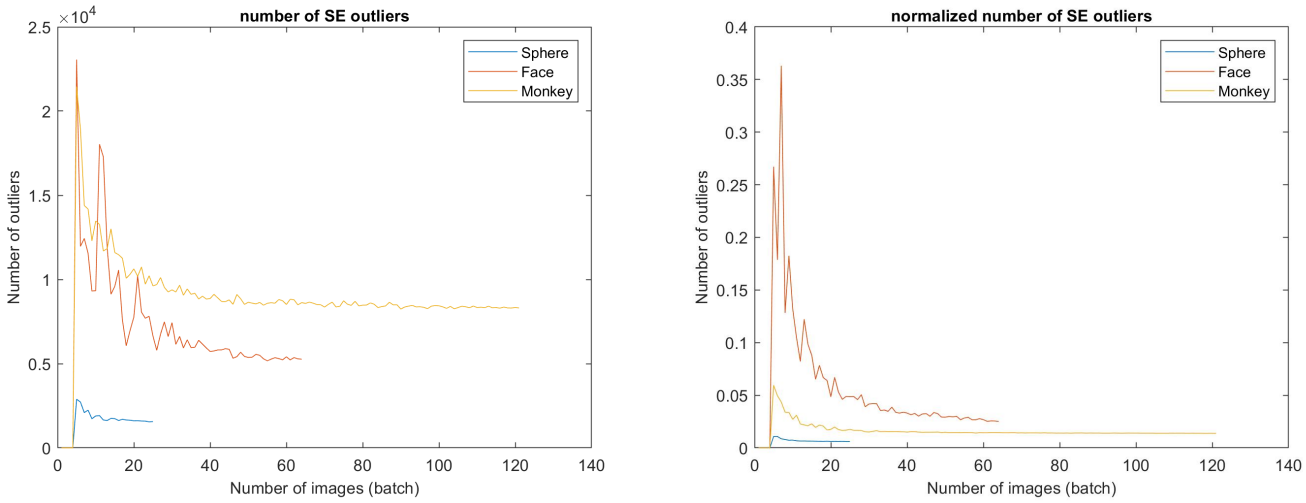
The *Face* is the one with highest number of outliers per pixel (figure 4b). That makes sense, the image was not generated synthetically. Those outliers come from noise in the pixel values as well as the texture of the man's face, especially around the eyes, eyebrows, and nose.

1.3 Shape by Integration

In theory, we should reconstruct the same height map regardless of the path we take. That is because integrals (sums, in the discrete case) are commutative operations. However, the imperfections of the real world add noise that is propagated over the sums. We can clearly see in figure 5 how the end result changes depending on the sum path we take. That is because the error propagation happens across columns or across rows.

One could use the average of both paths to dampen those errors. However, as illustrated in the figure 5, this method does not really help if both height maps are heavily distorted.

As the image 6 suggests, a greater number of images reduces the reconstruction errors, which is in line with the findings



(a) Total number of outliers

(b) Relavitve number of outliers

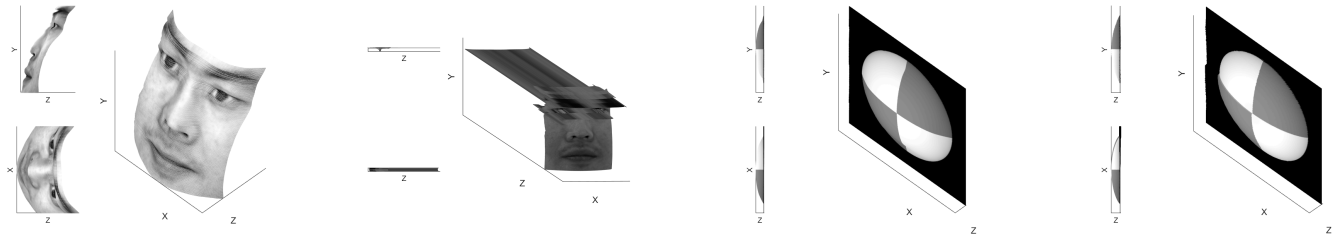
Figure 4: Number of outliers and outliers per pixel versus number of images. Thereshold kept at 0.005.

(a) Column

(b) column and shadow trick

(c) Column

(d) column and shadow trick

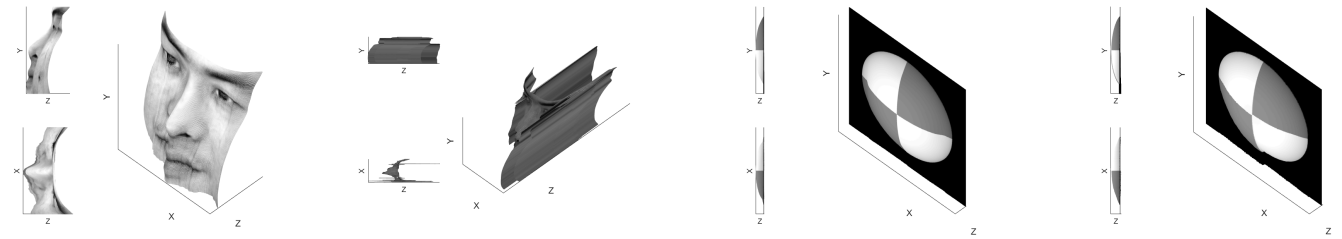


(e) Row

(f) row and shadow trick

(g) row

(h) row and shadow trick



(i) Average

(j) Average and shadow trick

(k) Average

(l) Average and shadow trick

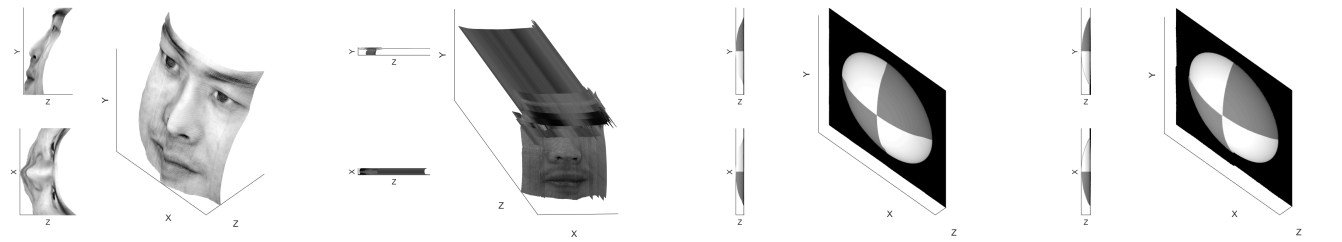


Figure 5: Comparison of height maps reconstructed using columns (a, b, c, d), rows (e, f, g, h) and the average of the two (i, j, k, l). The images in the first and third columns (a, e, i, c, g, k) do not use the shadow trick, the images in the second and forth columns (b, f, j, d, h, l) made use of the shadow trick.

(a) 5 images (b) 30 images (c) 65 images (d) 105 images (e) 121 images

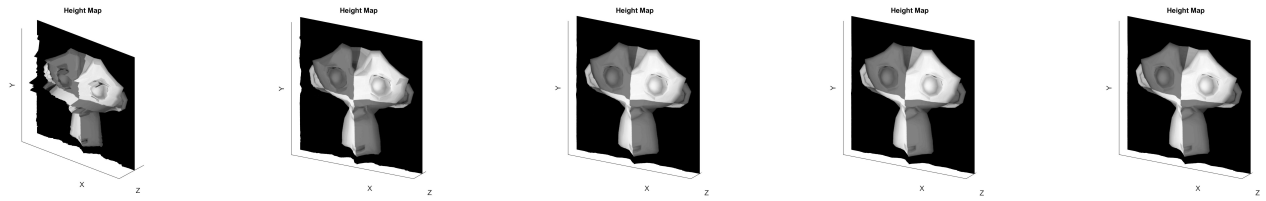


Figure 6: Height map reconstruction for 5, 30, 65, 105 and 121 images used to produce the normal and albedo.

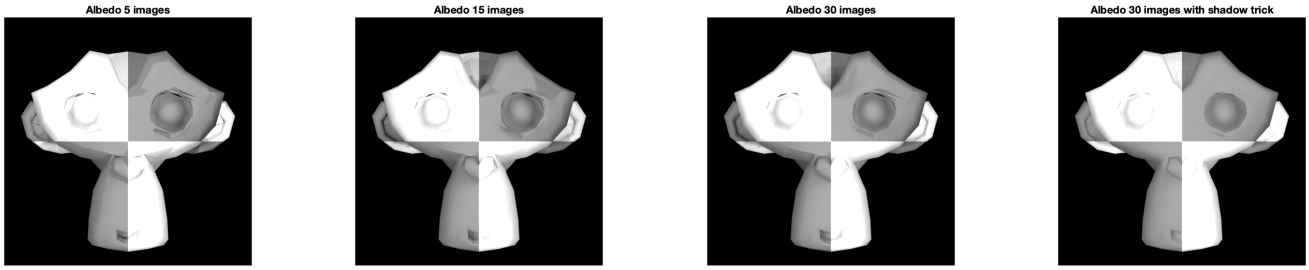


Figure 7: The geometry of the monkey face stays visible even with more data and shadow trick.

of 4a. That is expected, the more images we have the better our normal decomposition is.

1.4 Experiments with different objects

1.4.1 MonkeyGray model

When estimating the albedo of the MonkeyGray model we get darker areas where the geometry in the image points away from the camera. The shape of the face stays visible in the albedo estimate even though the albedo is constant for most of the surface. This may be caused by these areas being in the shadow for many images leading to under constrained linear equation systems.

While the shadow trick and a higher number of images helps to get more stable albedo estimates, the errors described above persist. See figure 7.

1.4.2 Estimation for RGB

To handle 3-channel RGB inputs we run the albedo estimate for each channel separately as if they where grayscale images and interpret the result as the channels of the combined RGB estimate as seen in figure 8.

Points valued at zero cannot have an normal estimate for that channel. This makes it harder to combine three normal estimates to one. Instead, we chose to convert the image into grayscale and estimated the normals on the gray image.

1.4.3 Yale Face Database

In figure 5 we saw the estimation for the Yale Face data set. As discussed in section 1.3 the reconstructed height map shows an increasing error along its reconstruction direction, whereas in theory, the path does not matter.

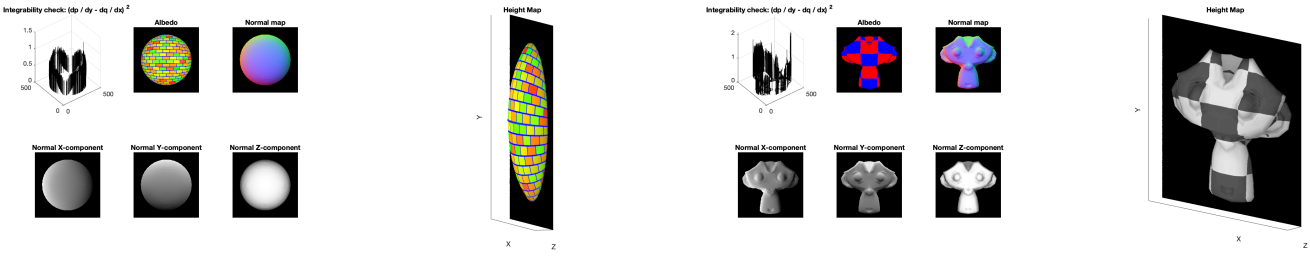


Figure 8: Results from albedo estimate for seperate RGB channels; normals are estimated from grayscale image

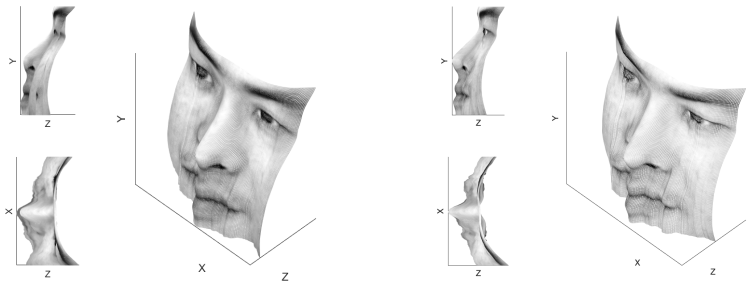


Figure 9: On the left row surface construction with the full data set, on the right after removing 22 problematic images. Top and side view noticeable improve but details around cheeks suffer

This error can be explained by noisy images and imprecision of estimates for big shadow areas. If these problematic images are removed from the data set for the row construction as seen in figure 9. To reduce the accumulating error images with shadows along the reconstruction direction seem to be most harmful. In your improvement its visible that we were able to remove the raising error from left to right but suffer from the noise around the cheeks due to the smaller data set.

2 Color Spaces

RGB Color Model

The choice for using RGB is based on the way the human eye works. The human eye contains cone cells which are responsible for color perception. There are three different types of cone cell that each responds to a different range of wavelengths. The brain combines the signals from these cone cells into what we perceive as color. It is possible to recreate the sensation for many of the visible colors simply by mixing three different colors of light with varying intensity. This is known as the trichromatic theory. Red, green and blue are the primary colors because together they create a large gamut, that covers a sufficient part of the colors we can perceive while still being easy to implement in hardware.

Standard digital cameras make use of photosensors that are able to detect light intensity but cannot distinguish colors. Color information is extracted by placing filters over these sensors. These filters are mosaics of tiny color filters that select specific wavelengths. The most common filter is the Bayer filter which consists of 25% red and blue filters and 50% green filters. The larger number of green filters is an attempt to mimic the human eye which is also more sensitive to green light. The signals from these three different filters are then combined into a digital RGB format.

Color Space Properties

Before discussing all the color spaces, it is important to understand that light has both a color component, which is determined by the wavelength spectrum and an intensity or brightness component. Images also contain these two types of information and these two components are usually referred to as chrominance and luma respectively. Luma is captured by the size of the RGB values while chrominance is captured by the differences in RGB values. Three of the color spaces we discuss make use of this idea by separating these components. The other two color spaces also separate luma, but they mainly capitalize on the fact that, while RGB is useful for the storage and recreation of color using a display, it is not a very intuitive color space for editing images. This is why there are many other color spaces that have channels that correspond to more intuitive concepts such as hue/warmth etc.

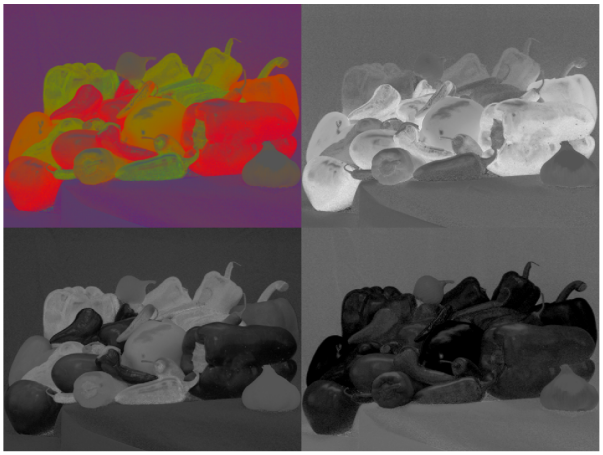
Normalized RGB (rgb) Color Space

By normalizing the RGB values we effectively remove any information about light intensity and only keep information about color. This can be very useful for image processing because we may want to remove distortions caused by light and shadows. We can see this effect in figure 10 where we see that the background tablecloth is now a more or less even color which captures the fact that, despite all the folds and shadows, it is the same object.

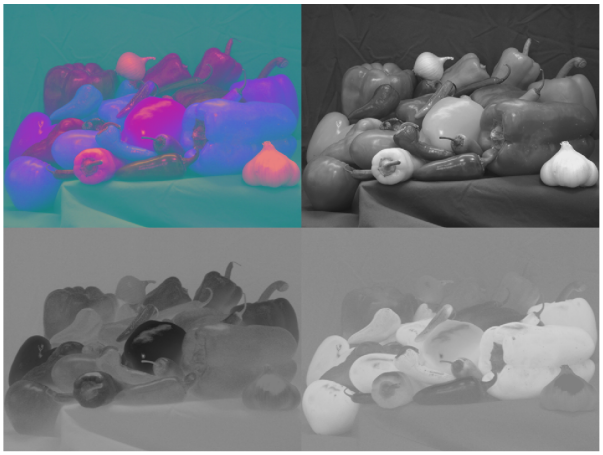
Grayscale

Grayscale color spaces instead single out the brightness component. Old images are grayscale because back then we had not yet developed techniques such as filters and could only capture light intensity. One big reason to use this color space is that it only has one channel which means that it can be stored and processed more efficiently. It is also important to note that humans are more sensitive to brightness than to color differences and that a grayscale image arguably still contains

(a) Normalized RGB



(b) YCbCr



(c) Opponent

(d) HSV

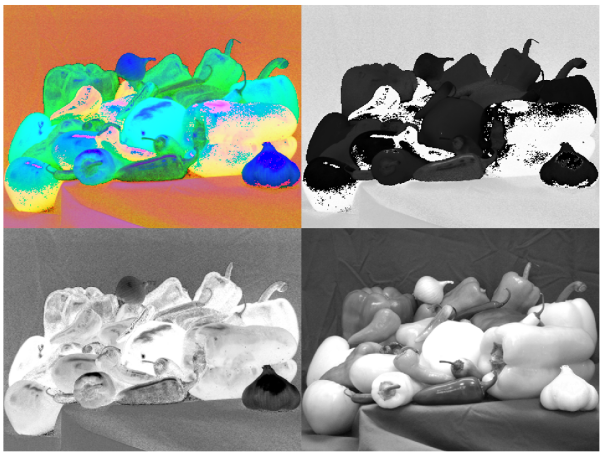


Figure 10: 4 different color spaces. For each subplot the full image is in the top left and channels are displayed in clock wise order.

Figure 11: 4 grayscale variants and original image.

most of the relevant information in an image. After all, we can still easily recognize objects in a grayscale image. For certain image processing tasks such as, for example, edge detection a grayscale image also contains enough information.

YCbCr Color Space
As we discussed above, humans are more sensitive to luma than to chrominance. This color space makes use of that factor for compression purpose by splitting the image into two chrominance channels, Cb (blue relative to green) and Cr (red relative to green), and one luma channel Y. We can store the chrominance channels at a smaller resolution to compress the image without losing a lot of relevant information. In 10 we can also see that the Y channel contains most of the relevant information.

Opponent Color Space
This color space is based on the opponent process theory of color which is a different interpretation of how we perceive color than the trichromatic theory we discussed earlier although both theories are used in conjunction nowadays. This theory is based on the idea that there are opposing colors, that is colors for which we cannot perceive an intermediate color. The two pairs of opposing colors that were identified are green/red and blue/yellow. While we, for example, can perceive color in between red and yellow, namely orange, that is not the case for these two pairs. The way this is used to create a color model is by having one channel that scales from red to green (O_1), one channel that scales from blue to yellow (O_2) and one channel that captures the brightness (O_3). This color space is useful because both of the opponent channels correspond to the intuitive notion of warm and cold colors. Red and Yellow are both perceived to be warm while blue and green are both perceived to be cold. This allows us to very easily change the perceived warmth of an image when editing images.

HSV Color Space
The Hue Saturation Value color space is another attempt at creating a color space that is more aligned with human intuition about colors. Hue, saturation, and value are some of the components contained in common color appearance models that seek to mathematically model how humans perceive color. It is hard to describe exactly what hue, saturation, and value are, but everyone that has used image editing software will have some understanding of what they represent. Hue relates to the dominant wavelength, saturation relates to the sharpness of the peak for the dominant wavelengths which we perceive as color vibrancy and value relates to perceived brightness (we can also see this in figure 2 where the value channel looks similar to a grayscale image).

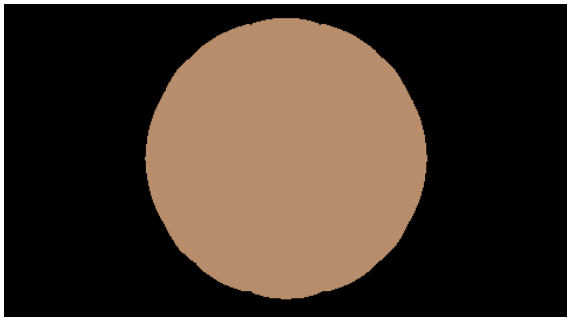
More on Color Spaces
Another interesting color space is CMYK which is used for printing and refers to four inks that are often used for color printing: Cyan, Magenta, Yellow, and Key (black). Printing requires its own color space because ink cannot emit light and has to reflect light instead. This is a subtractive color model similar to the models that painters have used for a long time. Cyan, Magenta, and Yellow can together be used as primary colors that can be mixed to create a wide range of colors including Black. Black ink is added to create a good looking black color and to prevent the waste of the more expensive colored inks if we just want to print a black and white image.

3 Intrinsic Image Decomposition
In intrinsic image decomposition an image is separated into several components, for example in a reflectance (albedo) and a shading (illumination) component.

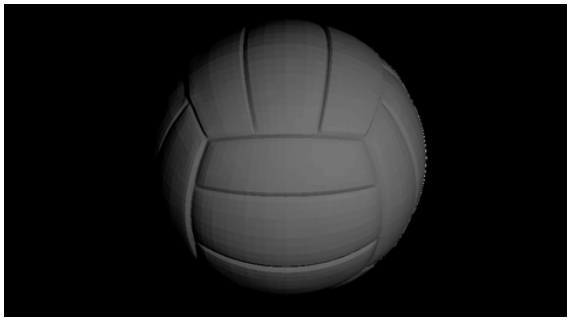
3.1 Other Intrinsic Components
An image could not only be decomposed in an albedo and a shading component but also in a specular component $C(\vec{x})$ such that the image can be reconstructed via $I(\vec{x}) = R(\vec{x})S(\vec{x}) + C(\vec{x})$. The specular component accounts for the viewpoint, the geometry and illumination, see [4]. Another approach is to consider noisy images f that can be decomposed in a true image component u and a noisy component v. The aim is then to denoise the image and reconstruct the true image u. As proposed by Rudin-Osher-Fatemi [7] u can be obtained by an energy minimization approach and by solving for $\inf_u\{\int_\Omega |\nabla u| + \lambda \int_\Omega |f-u|^2 dx\}$, where Ω is the domain of the image. See also [3] and [6].

3.2 Synthetic Images
As it was mentioned in the lecture the process of obtaining a ground truth intrinsic image decomposition is a hard and generally ill-posed problem. If for example $S(\vec{x})$ and $R(\vec{x})$ are a decomposition, then also $\alpha S(\vec{x})$ and $\frac{1}{\alpha}R(\vec{x})$ reconstruct

(a) Original Ball



(b) Albedo Image



(c) Shading Image

(d) Reconstructed Image

Figure 12: The Original Image, the Albedo and Shading Image and the Reconstructed Image

the image. Therefore it is not clear which of the possible decompositions reflect the absolute ground truth unless further measurements, like a light meter, are available. That is why in practice often synthetic images are used for which decompositions are available. See [1] for a reference.

3.3 Image Formation

See figure 12.

3.4 Recoloring

The original color of the ball is (184,141,108).

(a) Original Ball

(b) Recolored Ball

Figure 13: The Original Image and the Recolored Ball via Image Decomposition

As it can be seen in figure 13b the green color distribution is not uniform. There are spots with darker and lighter green. The reason for this is that after recoloring the albedo image with pure green, it is multiplied with the shading image to obtain the recolored image. The color distribution in the shading image is not uniform because it reflects the natural shadows of the object. Therefore the color distribution in the resulting image is not uniform as well and corresponds to the distribution of the shadows.

4 Color Constancy

The following section deals with color constancy correction algorithms. They are first estimating the real illumination and then changing and then changing the image such that it appears to be taken under a canonical illumination. Color constancy itself is the ability of the human color perception system that ensures that the colors of objects appear constant under different illumination conditions.

4.1 Gray World Algorithm

The underlying assumption of the Gray World algorithm is that the average color for each channel in a natural scene should be grey, that is on average all channels have the value 128. Therefore we transform each channel $C \in \{R, G, B\}$ according to

$$C_i \mapsto \frac{128}{255} \frac{C_i}{\frac{1}{n}\sum_i C_i},$$

where $C_i \in [0, 1]$ is the normalized double precision representation of the respective color in pixel i. After that transformation all channels have an average color of 128, that is grey. The result of such a a transformation can be seen in figure 14. Note that the in the corrected picture 14b the reddish tint vanished and the resulting picture looks more natural.

(a) Original image

(b) Color corrected image

Figure 14: The AWB demo picture before and after automatic white balancing (AWB)

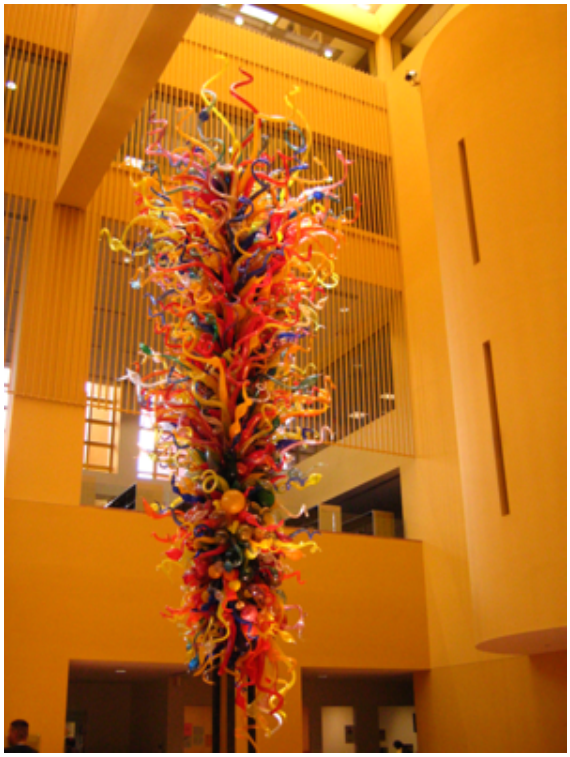
4.2 Negative Example for Gray World Algorithm

The gray world assumption states that the colors of the original setting are randomly distributed in each channel and average to gray, that is 128 in RGB color decoding. As can be seen in the figure 14, it is reasonable to assume that this is the case in the original setting, due to the even distribution of most colors. However, if that assumption is violated and the image has a dominant color, the Gray World Algorithm will fail to achieve good results, which is often the case in the real world.

In figure 15 an example is given where the dominant color is orange. As it can be seen, the 'corrected' image in 15b overcompensates the orange in the original image and the image gets a bluish tint. The picture was taken from `https://web.stanford.edu/~sujason/ColorBalancing/gallery.html`.

4.3 White patch algorithm

The white patch algorithm as proposed in [2] is a simplified version of the Retinex algorithm, see [5]. It estimates the illuminant by the maximum value in each channel. Therefore each channel $C \in \{R, G, B\}$ is transformed by dividing



(a) Original image

(b) Color corrected image

Figure 15: Example of color dominant original picture and failing Gray World algorithm

each pixel value C_i by it's maximum, that is $C_i \mapsto \frac{C_i}{\max\{C_i\}}$. Here C_i is again taken to be normalized double precision representation. The intuition behind that is that the brightest point of the image should represent the color of the illuminant. Therefore by rescaling, the brightest point becomes pure white and the image appears to be taken under this (canonical) white illumination. By doing so the other colors are corrected as well.

Conclusion

To conclude we will briefly summarize our findings.

We found that, for photometric stereo, it is crucial to have enough images, especially for modeling more complex objects such as the rendered monkey or a human face. If enough images are available, the shadow trick does not help a lot anymore and it becomes possible to get reasonable height maps even for face images that violate ambient light assumptions and contain noise.

All color spaces we covered separate the image into chrominance and luma components. This allows us to analyze these components separately, to eliminate light based distortions for example, and it also allows for more efficient storage by either removing chrominace information altogether (grayscale) or by storing the chrominance at a lower resolution. Then there are also many color spaces such as HSV and Opponent that attempt to align more closely with aspects of human color perception such as saturation or warmth to make editing images easier.

We found that it is very useful and easy to manipulate images if we have access to the intrinsic components such as albedo and shading which are easily obtained for rendered images, but unfortunately it is very hard to make these decompositions for real images because it is an even more ill-posed problem than color constancy and requires a lot of assumptions about the world or additional information, for example in the form of multiple lighting conditions for the same scene like the ones we used for photometric stereo.

By experimenting with the gray world algorithm we confirmed that color constancy is indeed an ill-posed problem and that assumptions that need to be made can also be broken easily. This is why color constancy in practice is usually done using human supervision. Humans can then provide information about the lighting conditions or adjust images themselves using more flexible color constancy algorithms until the images look right.

References

[1] Nicolas Bonneel, Balazs Kovacs, Sylvain Paris, and Kavita Bala. Intrinsic decompositions for image editing. In *Computer Graphics Forum*, volume 36, pages 593–609. Wiley Online Library, 2017.

[2] Jonathan Cepeda-Negrete and Raul E. Sanchez-Yanez. Color constancy algorithms in practice.

[3] Jérôme Gilles. Noisy image decomposition: a new structure, texture and noise model based on local adaptivity. *Journal of Mathematical Imaging and Vision*, 28(3):285–295, 2007.

[4] Roger Grosse, Micah K Johnson, Edward H Adelson, and William T Freeman. Ground truth dataset and baseline evaluations for intrinsic image algorithms. In *2009 IEEE 12th International Conference on Computer Vision*, pages 2335–2342. IEEE, 2009.

[5] Edwin H Land and John J McCann. Lightness and retinex theory. *Josa*, 61(1):1–11, 1971.

[6] Stanley Osher, Andrés Solé, and Luminita Vese. Image decomposition and restoration using total variation minimization and the h. *Multiscale Modeling & Simulation*, 1(3):349–370, 2003.

[7] Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*, 60(1-4):259–268, 1992.