CV Lab 1

Pieter Bouwman - ID 11904488 Tom Lieberum - ID 13253042 Erik Jenner - ID 13237896

0 Introduction

Many factors influence the appearance of objects, such as their reflectance at each point and for each wavelength, their shape and the position and spectrum of light sources. In images, a lot of that information is necessarily lost; for example, they are only two-dimensional and contain only three colour channels instead of the full continuous spectrum.

In this lab, we look at different aspects of image formation, as well as the inverse process of reconstructing shape and colour from images. Section 1 deals with *photometric stereo*, a particular method for reconstructing the shape and albedo of an object given images taken under different known lighting conditions. Section 2 is about colour spaces, i.e. how different ways to project the infinite-dimensional space of wavelength spectra to low-dimensional spaces are related. Section 3 covers intrinsic image decomposition and in particular the formation of an image from albedo and shading information, which we also use to recolour an object. Finally, we look at colour constancy in Section 4. We use the Grey-World algorithm to determine how a scene would look in white light, when only images under different lighting conditions are available.

1 Photometric Stereo

1.1 Estimating Albedo and Surface Normal

Albedo and surface normal reconstruction As explained in the supplementary material, photometric stereo is an algorithm for determining albedo and shape of a surface from n images taken from the same direction but with light sources in different (known) directions. For mathematical details we refer to the supplementary material.

Figure 1 shows the albedo and surface normal that were determined using this method with five sources on a synthetic image. Presumably, the underlying sphere used to generate the images had piecewise constant albedo. The recovered albedo is almost piecewise constant but a slight darkening (i.e. lower albedo) is visible towards the edges of the sphere.

Different numbers of light sources In principle, three different light directions would suffice to solve the system of linear equations. However, this is not the case in practice, as fig. 2 shows. The reason are likely shadows: if a pixel lies in shadow for a particular light source, the corresponding equation is zeroed out with the shadow trick, and this leads to fewer equations that constrain the solution. With three light sources, all the pixels that lie in shadow for one of them are thus underconstrained. This also explains why the problems occur at some of the edges of the sphere: this is where the shadows are.

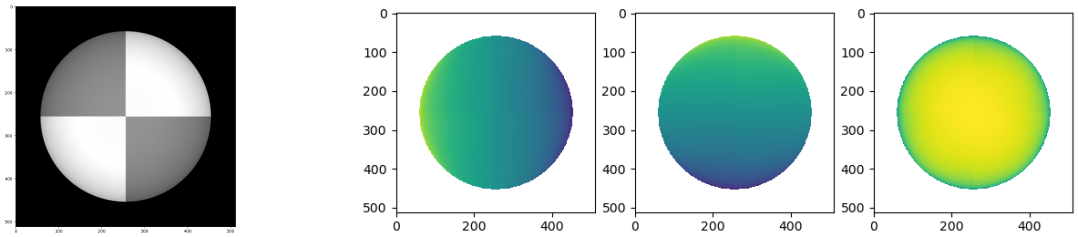
Using more than five sources yields only very slight improvements, shown in fig. 13 in the appendix. The darkening effect at the edges is less strong with 25 sources but the difference is barely visible. We also checked numbers between five and 25 without observing a large difference.

Shadow trick The photometric stereo equation will not be satisfied for those light sources for which the pixel lies in shadow. The *shadow trick* deals with this problem by zeroing out the relevant equations: we define the diagonal matrix $\mathcal{I} = \operatorname{diag}(i_1, \dots, i_n)$ and solve

$$\mathcal{I}i = \mathcal{I}\mathcal{V}g \tag{1}$$

instead of the original equation $i = \mathcal{V}g$.

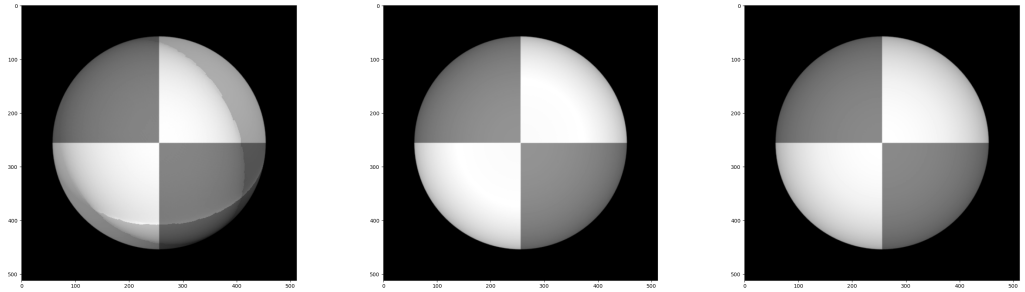
We can also calculate the albedo without the shadow trick, the results of which are also shown in fig. 2. In both cases, the results are slightly worse in terms of the darkening towards the edges. Now however, the result with five images is slightly better than with 25. This means the shadow trick is particularly important with more images, while its effect with only five images is relatively small.



(a) Albedo

(b) Surface Normal

Figure 1: Albedo and Normal as determined from five light sources



(a) 3 sources, with shadow trick

(b) 5 sources, no shadow trick

(c) 25 sources, no shadow trick

Figure 2: Degradation of the albedo reconstruction with fewer light sources or without shadow trick

1.2 Test of Integrability

Once we know the normal vectors, we can compute the first derivatives of the height function $f(x, y)$. As explained in the supplementary material, if $N = (a, b, c)$ and the sphere coordinates are $(x, y, f(x, y))$, we can compute the first derivatives as follows:

$$\frac{\partial f}{\partial x} = \frac{a}{c}, \quad \frac{\partial f}{\partial y} = \frac{b}{c} \tag{2}$$

The second mixed partial derivatives of f are approximated by computing the difference in the first derivatives between neighboring pixels.

Ideally the difference between the mixed derivatives should be zero (this is a general property of twice continuously differentiable functions). We compute the pixel-wise squared error between those two and use that as a basis to identify outliers where the estimation is too bad to be used in subsequent steps.

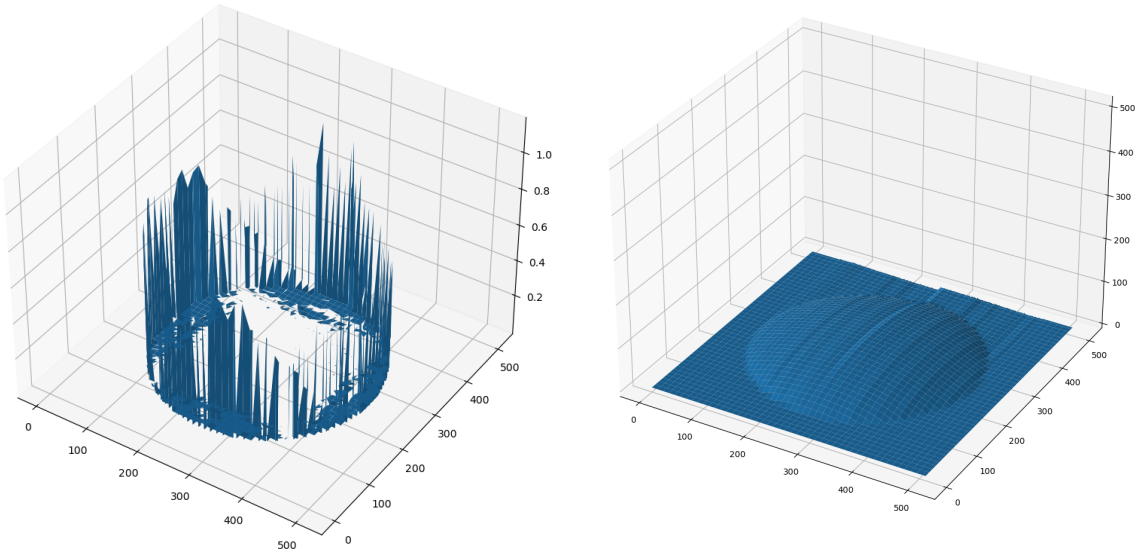
We used a threshold of 0.001, which resulted in roughly 2% of the pixels being outliers (exact values listed in table 1). The number of outliers is highest with very few sources but afterwards it increases again. This behavior is robust against the choice of different thresholds.

One source of the errors is of course the discretization. The equality of second derivatives does not hold precisely if the derivatives are replaced with finite differences. This approximation of the second derivatives as finite differences would become exact if the first derivatives were linear or in other words, if the second derivatives were constant. For a sphere however, the second derivatives diverge[1] at the edges instead of being approximately constant which makes the discrete approximation worse and creates larger errors. As fig. 3(a) shows, this is exactly what we observe.

# of sources	5	10	15	20	25
# of outliers	9084	4703	3892	4118	4858

Table 1: Number of outliers with different numbers of sources (threshold = 0.001)

[1] WolframAlpha with input `d/dx d/dy sqrt(1 - x^2 - y^2)`



(a) Squared errors (25 light sources, lower threshold of 1e-6 used for better visibility)

(b) Height map with column-major integration path

Figure 3: Violations of the equality of second derivatives (a) and height map (b)

1.3 Shape by integration

The height map of an object can be reconstructed using the estimated first derivatives, by calculating the path integral

$$f(x, y) = \int_{\gamma} \nabla f(x', y') \cdot d\mathbf{s} \tag{3}$$

with γ a continuous path: $\gamma : [0, 1] \to \mathbb{R}$ with $\gamma(0) = (0, 0)$, $\gamma(1) = (x, y)$.

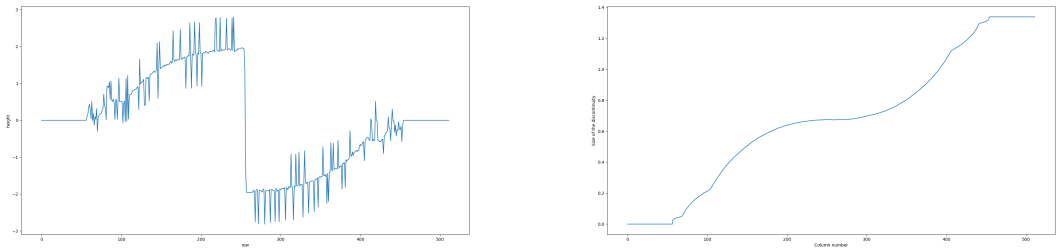
Since we only have a discrete approximation of the gradient, we need to sum over the pixels and multiply it with the step-size (1 in this case). In theory we can use any path we want, but in practice we either sum over the left most column first and then along the rows ('column-major') or over the top most row first and then along the columns ('row-major'). As a third option, we can compute the height maps using both of these methods and average them ('average'). We implemented all versions efficiently using `numpy`'s `cumsum` but also checked the result against a double-loop solution.

Results We expected to see a half-sphere as the reconstructed surface but instead got a much flatter shape (see fig. 3). The reason is probably closely related to the darkening of the reconstructed albedo towards the edges: if part of the darkening in the images themselves is attributed to the albedo, this leads to normal vectors that are closer to the camera direction, and thus to a flatter surface after integration.

We also observed an unexpected ridge in the height map. This ridge extends even to the background and is therefore clearly an artifact rather than the true surface. The ridge is always parallel to the second direction of integration, i.e. along the rows for a column-major path. With averaging, both ridges are visible, though less strong. Averaging also washes out other smaller artifacts that are visible in the other two height maps. Figure 14 in the appendix shows the results for all three integration paths.

Figure 4(a) shows the height profile of this ridge. The slice is taken along the last column which completely belongs to the background. Ideally, the profile should therefore be completely flat. Instead we can see that the height increases slightly towards the middle and then suddenfly flips sign – this is the ridge we observe. To understand the origin of this discontinuous jump at the middle, fig. 4(b) shows the size of that jump as a function of the column. In the first column, there is no ridge at all and it then becomes more and more pronounced until it reaches its maximum in the last column (the one shown in fig. 4(a). This suggests a gradual accumulation of small errors during integration. These errors are apparently largest towards the edges of the sphere which fits with our observations from above.

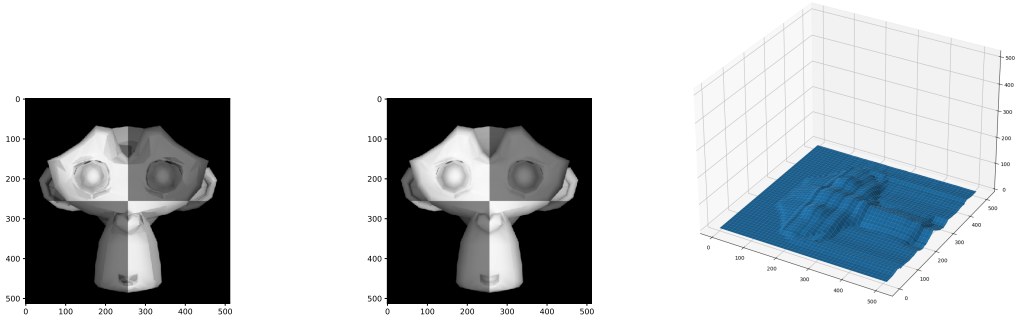
The ridge then arises because apparently these errors have a different sign in the top half and bottom half of the image. This must be related to the albedo of the sphere: the sign of the errors seems to depend on whether the integration path enters the sphere at the light albedo and leaves at the dark one or the other way around. However, we don't know what the precise mechanism is that leads to these different signs.



(a) Slice of the height map along the last column

(b) Height of the discontinuous jump along the ridge

Figure 4: Analysis of the ridge for column-major integration path



(a) Albedo for 5 images

(b) Albedo for 121 images

(c) Height map for 121 images

Figure 5: Reconstruction results on the gray monkey without the shadow trick

1.4 Experiments with different objects

Gray monkey With the gray monkey images (see fig. 5), there is a similar kind of albedo error as with the sphere. It is however much more clearly visible now, in part because there are more edges where the surface curves away. As before, this effect diminishes a bit with an increasing number of images. But even more so than in the sphere experiments, it doesn't go away entirely. Another difference is that abrupt changes in the albedo coming from shadows arise even with five images. This illustrates that the number of images needed to ensure that the problem isn't underconstrained depends on the scene and lighting directions.

Using the shadow trick worsens the results for the albedo significantly, especially when we only use few images. The results in fig. 5 are thus without the shadow trick.

Colour images An arguably more principled approach would be to directly solve a different optimization problem where the normal vectors are constrained to be the same for all channels.

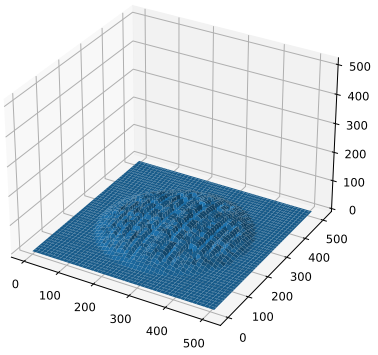
Faces For the Yale Face images, we disabled the shadow trick because results were much worse otherwise. The integration path has an important influence here. For example, a row-major integration path picks up the nose better, but it creates additional artifacts which also show up in the averaged height map, see fig. 7.

The results can be improved by removing those images with the strongest shadows (i.e. largest angles of the light sources). This is not surprising: for one thing, we removed the shadow trick which means we are not well equipped to deal with shadows. But another important consideration is that some of the shadows are not entirely black because they are illuminated indirectly by other parts of the face. This violates our model assumptions and thus distorts the reconstructed albedo and normal vectors. For the last result in fig. 7(e), we worked with 38 out of the 64 images. Figure 7(f) shows the least shadowy image we removed from the dataset.

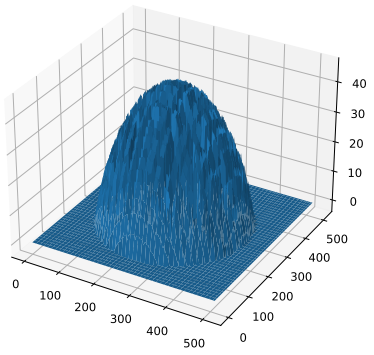
2 Colour Spaces

2.1

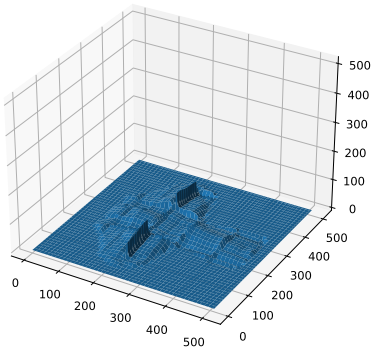
In the physical world, light is an electromagnetic spectrum of different wavelengths. If we want to capture an image, we need to work with the intensity of the different wavelengths as our raw data. In theory we could measure many different wavelengths with nanometer precision and store the spectrum with high fidelity. However, such measurement devices are very expensive and it is sufficient for the human perception to store the intensity values for red, green and blue, which is also why we only need to use these three primary colours in



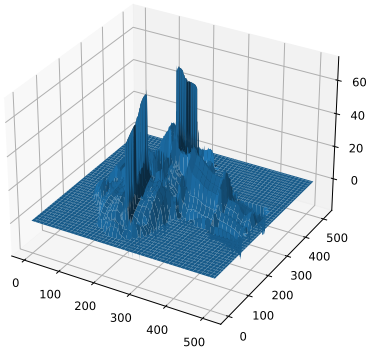
(a) Sphere



(b) Sphere (different aspect ratio)



(c) Monkey



(d) Monkey (different aspect ratio)

Figure 6: Results on colour images. We show the results first with the correct aspect ratio and then an exaggerated version where the artifacts are more visible.

our digital screens to present images. Another, less important reason is that image files would be significantly larger, if we were to save the complete spectrum, since we would not be working with three channels, but with hundreds.

Digital cameras record the three primary colours by sending the incoming light through filters which only transmit light of a certain wavelength (or, more precisely, a narrow spectrum.). They act as narrow band-pass filter. The sensor of the camera then records the intensity values for each pixel under that filter. Doing this for each of red, green and blue gives three images which can then be stored as a $(H \times W \times 3)$ tensor and displayed accordingly, e.g. via activating LEDs in proportion to the intensity value of that colour.

2.2

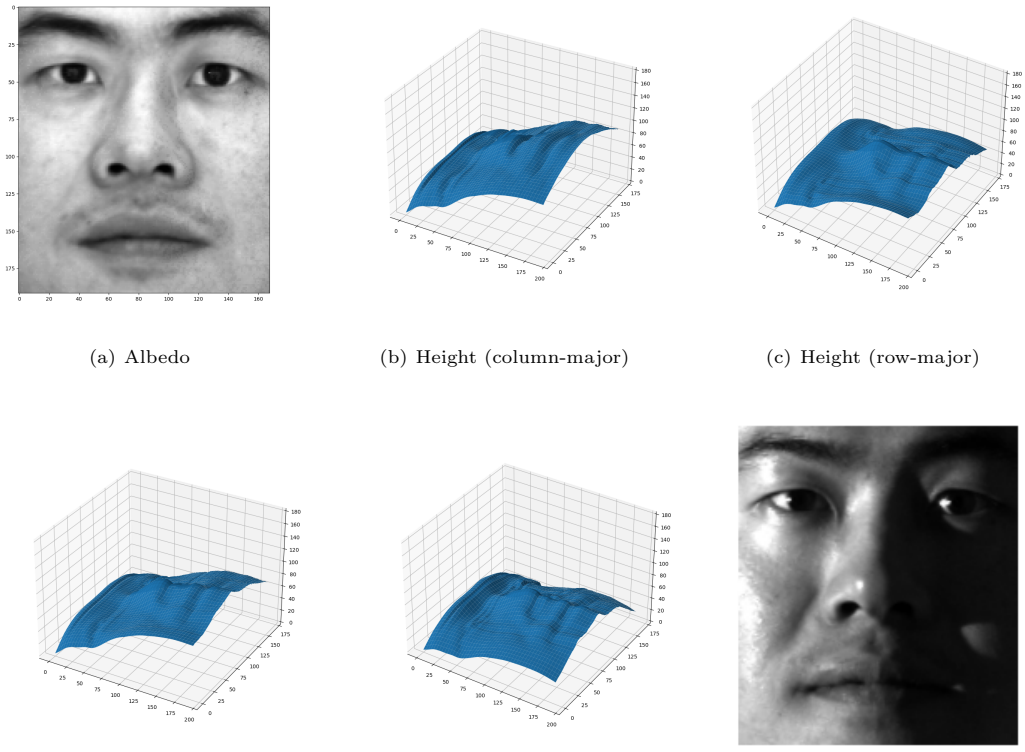
Details of implementation and flow of the program Before converting the images, the data type of the representation is changed to float32 to allow for numbers outside the interval of [0,255] and avoid rounding errors in subsequent steps.

Then, depending on the specified target colour space, the respective conversion sub-routine is called. We convert to the following colour spaces: Opponent, Normalised RGB, HSV, YCbCr, and Grayscale. They are explained below and the mathematical conversion rules are taken from the exercise sheet. The results are displayed in fig. 8. The results for the other colour spaces can be found in appendix B.

We note that the converted image is not very informative as the target colour spaces do not lend themselves to an RGB interpretation. Their channels are being clipped by matplotlib's imshow function, resulting in very distorted pictures, as can be seen in the results.

2.3 colour Space Properties

In some cases other colour representations can be more convenient than RGB. For example, having a representation of colour in which brightness is separated from colour information (chroma) can be useful when one wants to make changes to the brightness of an image without changing the colours, or vice versa. It can even



(a) Albedo (b) Height (column-major) (c) Height (row-major)

(d) Height (average) (e) Height (average, cleaned dataset) (f) Example for removed image

Figure 7: Results on the Yale Face dataset

be that we don't want or need colour information at all. We will go over the implemented colour spaces and shortly discuss their properties, benefits and possible application.

Opponent colour Space The first component of the three-channel colour space as defined in the lab assignment is the red-green channel which is obtained subtracting G from R. The second component is the blue-yellow channel. Yellow is obtained by adding R and G, which can be seen in the nominator of the formula (R+G-2B). The third component is the luminance channel: R,G and B are added to get information about the brightness of that point.

Normalised RGB colour Space The values of the 3 channels lie between 0 and 1 due to the normalisation. The normalised RGB colour space can be used when one wants to remove or equalize small local differences in brightness from an image while preserving colour information. In Figure 9[2] it can be seen how in a robo-soccer setting normalised RGB can be used to get a new image in which task-irrelevant information from shadows and illumination differences over the field can be removed.

HSV colour Space HSV represented as a cylindrical geometry. On the angular dimension is the hue, on the vertical axis the value (lightness), and on the radial dimension the saturation. Where RGB has three channels corresponding to a colour, HSV has one channel for brightness and two for colour. This way HSV separates luma (brightness or intensity) from chroma (colour information). Representing colour in this fashion allows for relatively easy equilization of image intensity while preserving the colour information.

YCbCr colour Space In this colour space there are 3 channels. The Y channel is the luma (brightness) component, while Cb and Cr are chroma components. Chromaticity is a representation of perceptual difference from grayscale (colourless). Low chromaticity means more towards how we perceive gray, and higher chromaticity appears more colourful to us. Cb is the so called blue-difference, it is proportional to the difference between blue and luma. Cr is the red-difference and is proportional to the difference between red and luma. While the human eye is sensitive to small changes in the Y component, the brightness, it is not to the Cb and Cr components. In image compression, information in these channels can be compressed (by subsampling) without the human eye noticing too much of a difference.

[2]This image was obtained from https://aishack.in/tutorials/normalized-rgb/ on Sept. 14th 2020

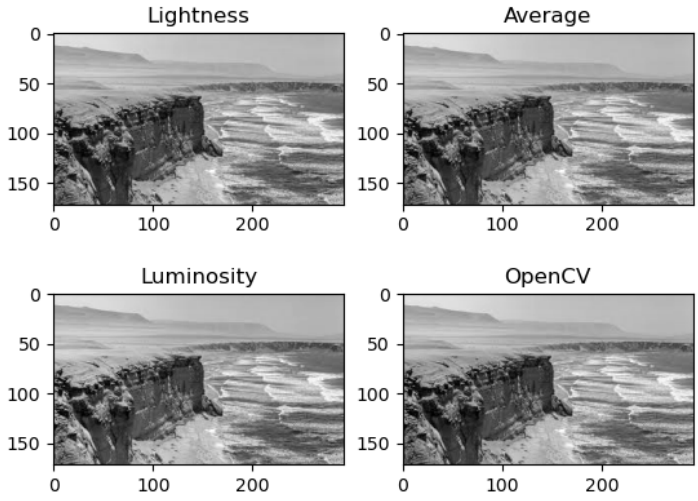


Figure 8: Original image converted to Grayscale, showing all 4 channels corresponding to lightness, average, luminosity and opencv standard function respectively.

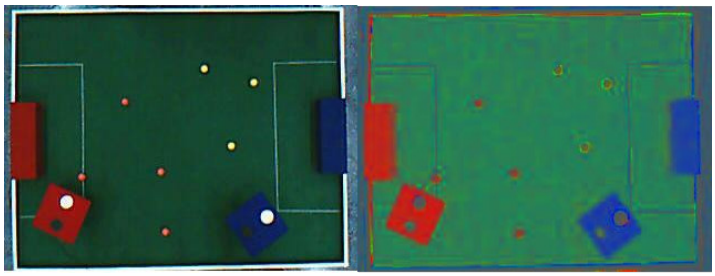


Figure 9: Left: original RGB colour space image. Right: original image converted to normalised RGB colour space.

Grayscale Grayscale images can be used when information about colour of objects in the scene is irrelevant to the task at hand. Grayscale images only have one channel, this can save a lot of memory when dealing with many or large images.

2.4

YUV As explained by Szeliski in 'Computer Vision: Algorithms and Applications' on pp. 94-95, the YUV space was used in Europe during the early age of colour television. Conceptually, it is similar to YCbCr.

The Y or *luma* channel encodes the luminance and can be computed via

$$Y' = 0.299R' + 0.587G' + 0.114B',$$

where the apostrophe denotes a compressed channel.

The two colour channels capture blue and red difference from the luma channel respectively.

$$U = 0.492111(B' - Y') \text{ and } V = 0.877283(R' - Y')$$

The colour channels were transmitted via a lower frequency, taking up less band-width, whereas the luma channel used a higher frequency. Black-and-white TVs could then effectively ignore the colour channels while colour TVs could make use of all three channels.

3 Intrinsic Image Decomposition

3.1 Other Intrinsic Components

The decomposition $I(\vec{x}) = S(\vec{x}) \times R(\vec{x})$ only accounts for an ideal diffuse (Lambertian) model of reflection. In particular it does not account for specular highlights that are due to smooth surfaces which reflect light

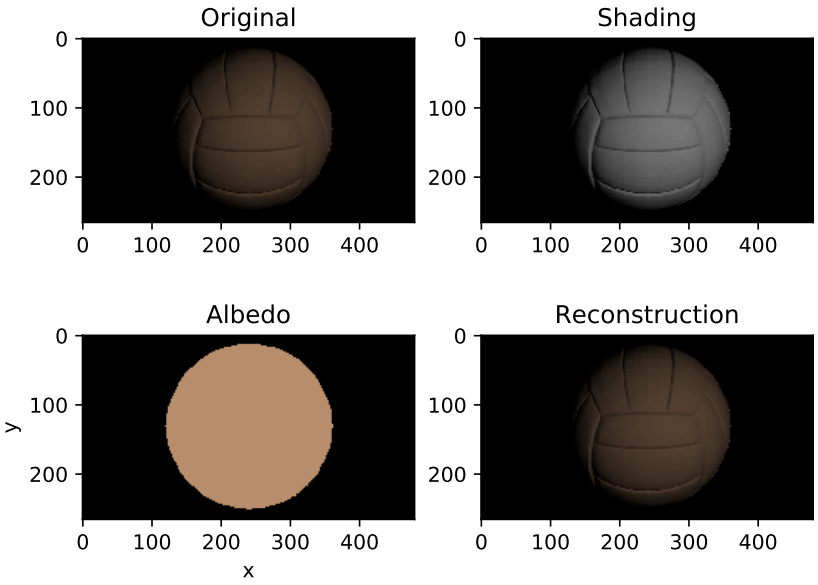


Figure 10: Original image, decomposition and reconstructed image of a ball. Normalized by 255.

preferentially in a particular direction. We can model this effect by introducing an additive specular component $C(\vec{x})$ in the term for the intensity:

$$I(\vec{x}) = S(\vec{x}) \times R(\vec{x}) + C(\vec{x})$$

This decomposition was proposed by Grosse et al. in their MIT data set paper[3].

3.2 Synthetic Images

When building a data set with synthetic instead of natural images, one can control the intrinsic components of the image precisely. This means that an algorithm that does intrinsic image decomposition can be tested against a 'ground truth', namely the parameters with which the image was synthetically constructed. In case of using natural images in the data set, the intrinsic components are not necessarily easy to determine. They might have to be estimated, which is already what the data set was intended for: testing your intrinsic component estimator, or they have to be measured (physically) in some way that might be more costly than when images are constructed synthetically.

3.3 Image Formation

To reconstruct the image, we load the albedo and shade in python and multiply them element-wise and then re-normalise by 255. To prevent overflow issues, we have to change the datatype from `unit8` to `int32`. The original image, albedo, shading as well as the reconstructed image can be found in fig. 10.

Calculating the root mean squared error between the reconstructed image and the original, we obtain $RMS_{255} = 0.2129$, so it is not a perfect reconstruction. Changing the normalization factor to 256 leads to a perfect reconstruction, i.e. $RMS_{256} = 0$. This is quite surprising and could be explained by an off-by-one error in the original decomposition. With the human eye those two different reconstructions are indistinguishable.

3.4 Recolouring

True material colour: The true colour of the material was obtained by taking the RGB value of the uniform albedo component. The RGB value is (R,G,B)=(184,141,108). This was obtained by looking at the R, G and B component of a coloured pixel in *ball_albedo.png*.

Green Sphere: In order to recolour the ball with pure green, we first find the points in the albedo image that have the original colour, and then replace the colour on these points with pure green. This gives us a pure green circle in the albedo image, that can now be combined with the shading again to get the reconstruction.

[3]Grosse et al., 2009 in ICCV: Ground-truth data set and baseline evaluations for intrinsic image algorithms

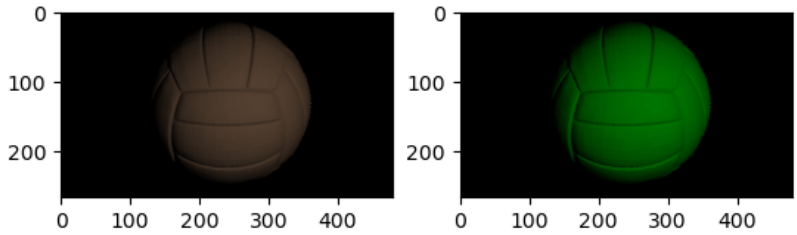


Figure 11: Original reconstruction from albedo and shading information (left). recoloured version (right).

(a) Original (b) Clip (c) Re-scale (d) Re-scale Channel-wise

Figure 12: Original image (a) and results of applying the GW algorithm with (b) clipping, (c) re-scaling to [0,255], or (d) re-scaling channel-wise afterwards.

Uniformity: What we observe is $I(\vec{x}) = R(\vec{x}) \times S(\vec{x})$ for all positions $\vec{x}$ in the image. Although the albedo ($R(\vec{x})$) is uniform, the shading $S(\vec{x})$ is not uniform. This means that $I(\vec{x})$, what we observe, will also not be uniform, even though the albedo is uniform.

4 Colour Constancy

4.1

The Grey-World algorithm assumes that under a white light the mean of an image will be $[128, 128, 128]$(grey). In the von Kies model, we model the colour of the illuminant as being represented by three scaling factors α, β, and γ, such that the image's values can be described as $[\alpha R_i, \beta G_i, \gamma B_i]$. Here, the R_i, G_i, and B_i are the values the image would have under white lighting conditions.

In the code, we first compute the mean of each colour channel over the whole image, resulting in

$$\mu = [\alpha 128, \beta 128, \gamma 128] \tag{4}$$

if we assume the grey world assumption to be true. To obtain the corrected image, we now simply divide each pixel by $\mu/128$. Under our assumptions, this will return the original pixel values $[R_i, G_i, B_i]$ for each pixel.

However, this might result in some pixels taking values higher than 255, which is why we (a) clip the values between 0 and 255, or (b) re-scale the values by the maximum over the image. We also try to re-scale it by the channel-wise maximum instead of the over overall maximum, which can be seen in (c). The three approaches can be compared in fig.12. Of the three approaches, the channel-wise scaling seems to produce the best result. For example, the roof of the building in the background is closer to the original image than it is in the clipped image is. In the clipped image the roof seems to be too bright. Also, for the channel-wise scaled image, the very front column of the building in the front has a higher contrast than in the image that was scaled by the total maximum. Generally the intensity is lower in the rescaled image than in the clipped image, which makes sense given that we are re-scaling by the maximum intensity rather than just cutting it off.

4.2

The Grey-World algorithm will fail if its core assumption is violated, i.e. if the image exhibits a noticeably different colour distribution than the assumed average grey. For example, this will occur if a significant part of the image is a blue sky or ocean, or a green forest. In those cases the algorithm will assume that the illuminant is more blue (or green) than is actually the case, resulting in an over-correction of the image.

4.3

Scale-by-max algorithm

In the 'Scale-by-max' algorithm, the assumption is that there is at least one white pixel in your image, which should be $[255, 255, 255]$ under white lighting conditions. By taking the channel-wise maximum $[R_m, G_m, B_m]$ one can re-scale the image accordingly by computing for each pixel:

$$[R_i, G_i, B_i] \mapsto 255 \cdot [R_i/R_m, G_i/G_m, B_i/B_m] \tag{5}$$

5 Conclusion

In this lab, we experimented with several methods for reconstructing the true shape and colour of objects and for image formation, and familiarized ourselves with different ways of representing colours. We also investigated under which circumstances these methods work well and when they can fail.

In part 1, we investigated the efficacy of the photometric stereo algorithm. The reconstructed albedo was reasonable but still contained some shading in all cases, so the algorithm did not manage to separate shape and albedo entirely. We also observed artifacts such as a noticeable ridge when reconstructing the images for a gray sphere. Photometric stereo can also deal with color images but with our averaging method, the results obtained for those were much worse. When reconstructing faces, the assumptions of the algorithm were violated, so we had to clean the data set to obtain more or less decent results.

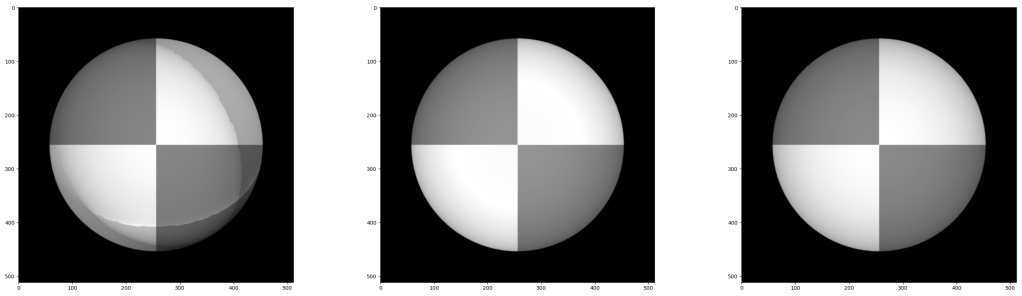
Part 2 dealt with different color spaces. We visualized one image in several different spaces by splitting up the channels. This allowed us to get a sense of the information they carry.

In part 3, we could successfully reconstruct an image that was decomposed into its shading and its albedo components. For this we assumed a Lambertian model of reflection. We also used this decomposition to recolour the displayed object.

Part 4 applied the simple Gray-World algorithm successfully in order to perform automatic white balancing on an image and reduce the effect of a non-white illuminant.

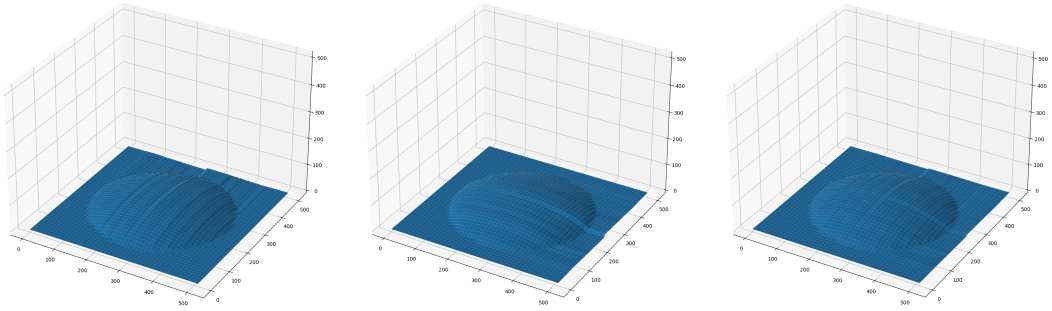
Compared to part 1, we got very good results in these last parts but that may also have been influenced by the easy images. For example, there are plenty of cases where the Gray-World algorithm would have failed, and recoloring the object was particularly easy because it had a uniform albedo which was already given separately from the shading.

A Photometric Stereo



(a) 3 sources (b) 5 sources (c) 25 sources

Figure 13: Albedo reconstruction with different numbers of light sources (with shadow trick)



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

Figure 14: Height map of the gray sphere with 5 images and different integration paths. We observe a ridge along the center of the sphere, which is visible as a bright line (or as a cross for (c)).

B Colour Spaces

In figures fig. 15 - fig. 18, we show the results for the additional colour spaces, not included in the main text.

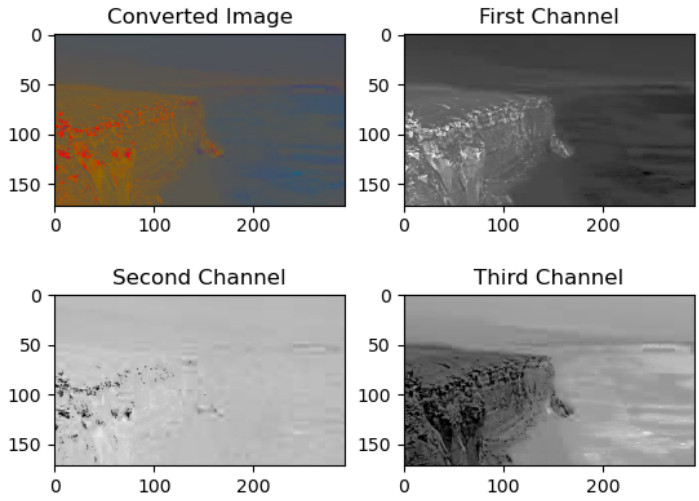


Figure 15: Original image converted to Normalised RGB colour Space, showing the converted image and its three channels separate. First channel is normalised R channel, Second channel the normalised B channel and Third the normalised G channel.

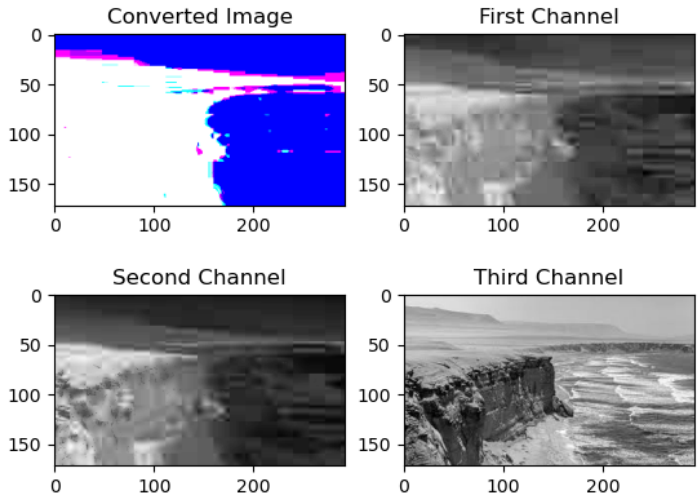


Figure 16: Original image converted to Opponent colour Space, showing the converted image and its three channels separate. First channel is the red-green channel, Second channel the blue-yellow channel, and Third channel the luminance channel.

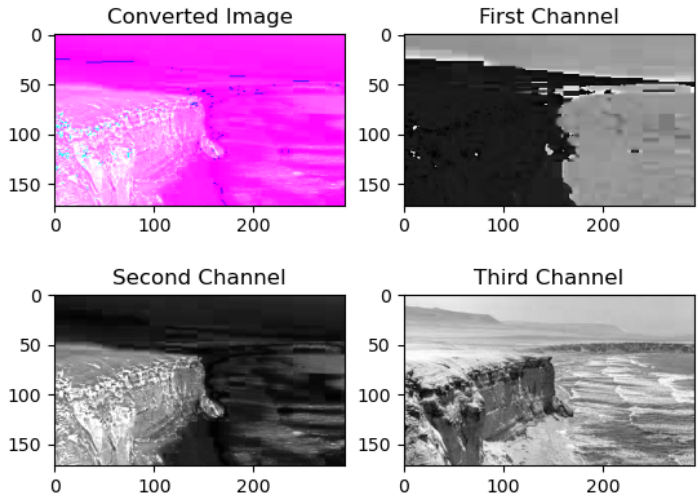


Figure 17: Original image converted to HSV colour Space, showing the converted image and its three channels separate. First channel is the hue channel, Second channel the saturation channel, and Third channel the value channel.

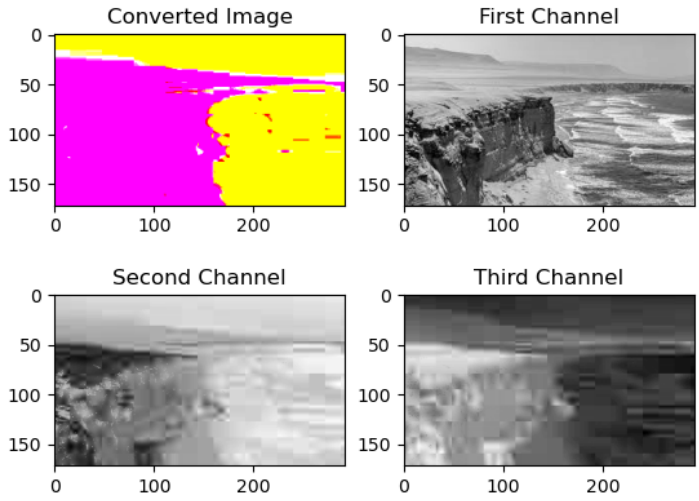


Figure 18: Original image converted to YCbCr colour Space, showing the converted image and its three channels separate. First channel is the luma channel (brightness), Second channel the blue difference, and Third channel the red difference.