Computer Vision Assignment 1

Frederik Harder
10986847
frederikharder@gmail.com

Maartje ter Hoeve
10190015
maartje.terhoeve@student.uva.nl

February 2016

1 Photometric Stereo

Photometric stereo is a method that can be used to construct a 3D object given 2D images of this object. By using different images of the object that are taken under the same view, but under different illuminations, one is able to reconstruct the 3D shape.

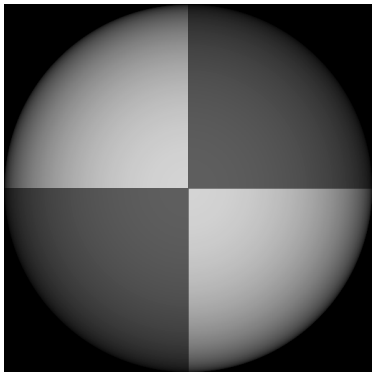
For this assignment we five images of a sphere, shown in figure 1. In order to be able to compute the surface normal we first construct five source vectors (the vector from the source to the object), one for each image. The exact positions of the vectors can be experimented with. We obtain the best results placing the sphere in the center of a 3D grid at $(0; 0; 0)$, with the frontal light source at $(0; 0; 1)$ and the lateral light sources placed around it with x- and y-coordinates equal to ± 1 (to make sure the light could come from the bottom and the top of the image) and the z-coordinate equal to 0. The next step of the algorithm is to read the pixel values of all images. For each pixel a matrix I is constructed with on the diagonal the values for this pixel in the five images. Now, as we know that

$$\mathbf{I}\vec{i} = \mathbf{IV}\vec{g}(x, y) \tag{1}$$

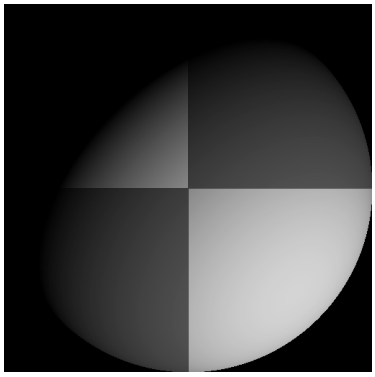
we can solve for $\vec{g} = (\mathbf{IV})^{-1}\mathbf{I}\vec{i}$ and get the surface normal at this point by

$$\frac{\vec{g}}{|\vec{g}|} \tag{2}$$

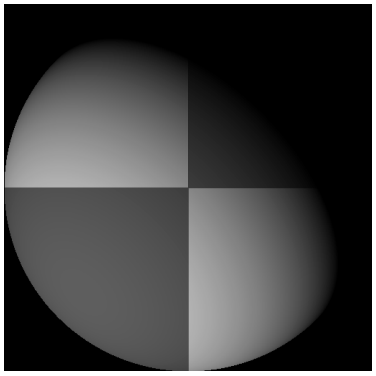
From the surface normals we have now computed, we can construct a height map, which gives a height estimate for each normal vector. Following the algorithm in **forsyth2003modern**, we construct $p = \frac{N_1}{N_3}$ and $q = \frac{N_2}{N_3}$ values from



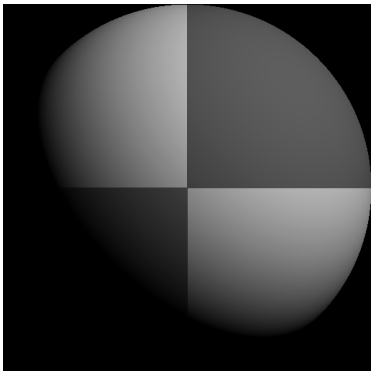
(a) sphere 1



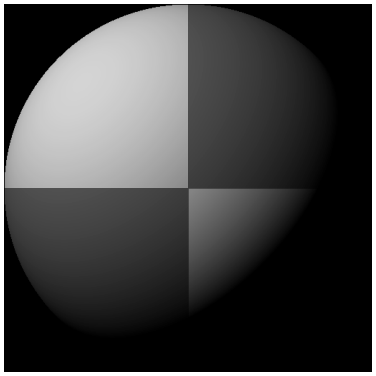
(b) sphere 2



(c) sphere 3



(d) sphere 4



(e) sphere 5

Figure 1: Used images - five spheres under different illuminations

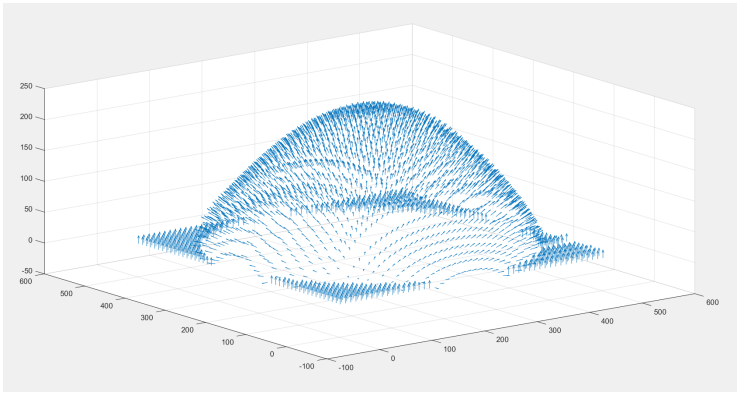


Figure 2: Reconstructed surface normals

the normal vectors N. Going down the leftmost column of the height map we compute height differences with respect to the top left pixel using q-values and after that, use p-values to compute heights for each row, following the algorithm in Forsyth and Ponce (2012).

Once we have all this, we can plot the surface normals using Matlab's *quiver3* function. Given that the coordinate system used so far places the top left pixel at maximum x and y, which is inverse to the x and y direction used in the plot function, the x and y values of all surface normals must be inverted as well. Figure 2 shows the plot with the surface normals. Using Matlab's *surf* function we plot the entire surface, which can be seen in figure 3.

Before plotting the surface we need to check whether

$$(\frac{\delta p}{\delta y} - \frac{\delta q}{\delta x})^2 \tag{3}$$

is small. However, as we do not know the complete function, we need to compute the derivatives slightly differently. In order to compute the derivatives, we go through the p-value matrix row by row and through the q-value matrix column by column (as opposed to what we do to construct the height matrix). For every matrix we compute the difference between the current value and the previous value and the difference between the next value and the current value. We take the average of these values. Like this we approximate the gradients. After that we can fill in the values in equation 3. Having computed this, we still need to decide on what values are considered 'small'. By looking at the matrix we decide to set the boundary at 1. All p- and q-values for which the corresponding value in the gradient matrix are larger than 1 are replaced by 0. One might expect that interpolating values (by taking the previous and next p- or q-values and averaging these two values) give better results. Yet, these values might also be too large. Our attempts at interpolation do not give a smoother plot. For this reason we have favour the use of replacement.

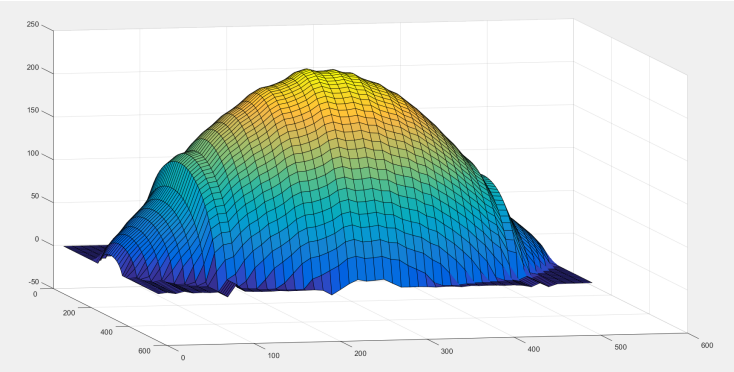


Figure 3: Reconstructed surface

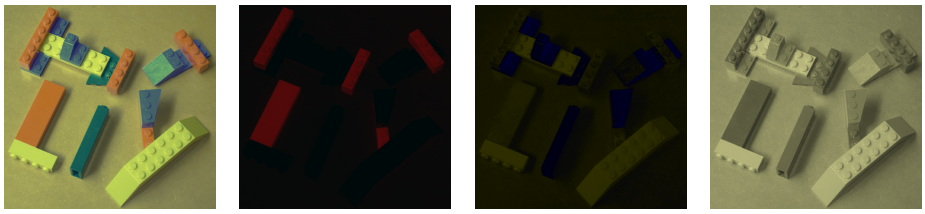
2 Colour spaces

In a colour space a colour is described as a number of colour channels. For example, in the RGB colour space a colour is composed of the intensity of red, green and blue it contains. For this assignment we have visualised the colour channels of the image in different colour spaces. Loading a picture with Matlab's *imread* gives access to the R, G and B components of the image. We have used these components to compute the channels in the different colour spaces. The results are shown in figures 4, 5 and 6.

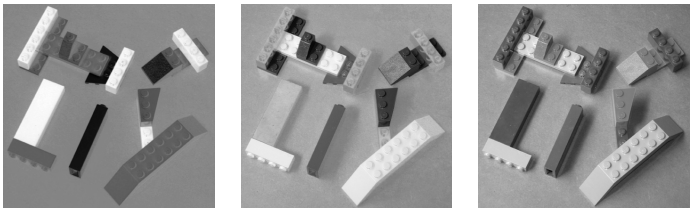
The opponent colour space encodes images with three channels which represent the green versus red colouring, the blue versus yellow colouring and the luminosity value of each pixel respectively. While this may not be apparent in grey scale, it can clearly be seen when individual channels are mapped back to RGB as in 4. A comparison with the original image shows that our mapping back is not completely accurate, which is likely owed to the renormalization of the channels. As this was not part of the assignment, we are still quite satisfied with the way it works right now.

In the normalised RGB colour space the R, G, and B values are divided by the sum of these three channels. This means that the values for r, g and b show the proportion of red, green and blue in the image. The three values sum to 1, thus removing the effect of changes in intensity from the image. As the bricks have reflective surfaces, the normalization does not entirely map them to the object colour, but the effect is still visible, shown in figure

The HSV colour c cylinder representation of this colour space. On the different angles of the HSV colour space, hue (H) is set out. The vertical axis of the cylinder represents the value (V) and the radius represents the saturation (S). As can be seen in the bottom left corner of both the coloured (with $S = V = 1$) and the greyscale hue plots, hue increases in uncertainty for small saturation values.

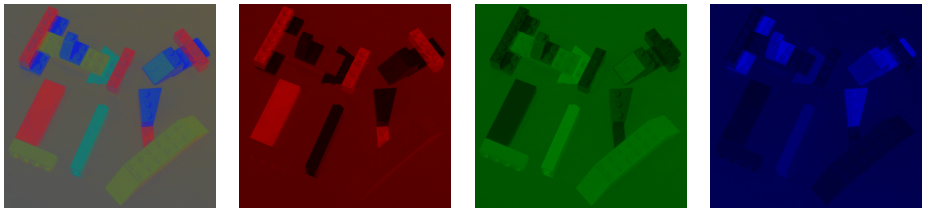


(a) reconstruction (b) green-red (c) blue-yellow (d) luminosity

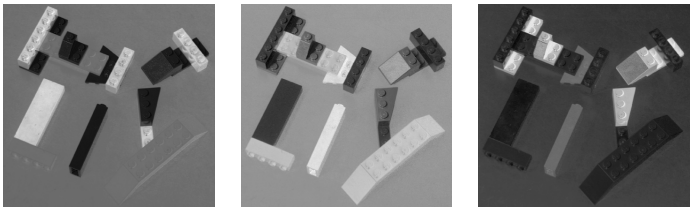


(e) green-red (f) blue-yellow (g) luminosity

Figure 4: opponent colour space in mapped back to RGB and in greyscale

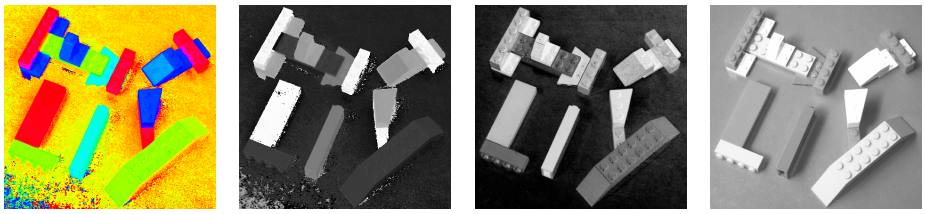


(a) image in rgb (b) red channel (c) green channel (d) blue channel



(e) red channel (f) green channel (g) blue channel

Figure 5: rgb colour space colour and greyscale



(a) coloured hue (b) greyscale hue (c) saturation (d) value

Figure 6: hsv colour space RGB and greyscale

References

Forsyth, D. and J. Ponce (2012). *Computer Vision: A Modern Approach*. Always learning. Pearson. ISBN: 9780136085928. URL: https://books.google.nl/books?id=gM63QQAACAAJ.