1 Introduction

In this assignment, we will implement and discuss a basic algorithm for photometric stereo. Given a set of images of the same object taken from the same angle, but under different lighting directions, we will compute a depth map that shows a 3D representation of the object.

2 Photometric Stereo

The goal of photometric stereo is to compute the surface normal map (depth map) of a given object in a series of images. The images are all of the same object from the same angle – only the direction of light is changed. The resulting images will be shaded in different areas. Calculating the difference between them allows for computing the depth of the object.

To solve the depth map N, we used the Linearized Lambertian model:

$$I(p) = \rho(x)s(x) \cdot n(x)$$

Multiplication is commutative, so:

$$I(p) = \rho(x)s(x) \cdot (\rho(x)n(x))$$

$$N = norm\left(S^{-1} \cdot I\right)$$

The first step is to calculate the direction that the light is coming for. In this case, we are already given a set of light vectors S representing the direction of light for each image. With this, we can extract the albedo:

$$A = \|M\|$$

where $M = \rho N$. The albedo value represents light reflection from a surface: the higher the albedo, the more light is reflected. The albedo is represented as a value between 0 and 1, where 1 is total reflectance and 0 total absorption. This specific albedo map has been extracted from the albedo modulated normal field, which is calculated by normalization of M, which we can obtained by:

$$M = S^{-1} \cdot J$$

where J is the array containing intensity values within the applied mask, and S is the array containing light vector. Once M is obtained, we get the vector length at each pixel by calculating the Euclidean norm.

3 Beethoven

The Beethoven dataset consists of three synthetic images of a Beethoven bust as seen in Figure 1 on Page 3.

3.1 Albedo Map

The matrix achieved from calculating the Euclidean norm, as explained in section 2, is the Albedo map representation, where the bright areas reflect high albedo and dark areas reflect low albedo, as can be seen in Figure 2. To meet the 'normal' representation of Albedo, we have normalised the vectors in range 0 to 1 in the final produced Albedo map. The values within this matrix are therefore weighted relative to the initial range of the vector length matrix, such that the pixel with value 1 reflects the point of most light reflectance in this particular image setup.

Figure 1: Beethoven Images

When we look at the albedo map in Figure 2 below, we see the Beethoven figure without the highlights and shadows from the original images. The values from this map will help us compute the depth map.

Figure 2: Beethoven Albedo

3.2 Depth Map

Figure 3 shows the computed depth map for the Beethoven Image. As can be seen, the result is pretty good: The details of the face show well, as well as the structures of the hair and the shirt. It definitely looks like a 3D version of the Beethoven bust. This is probably largely due to the fact that the dataset is synthetic and clean, without noise.

The visualisation of the depth map has been done, not with the included `display depth` function, but with module `graphs objs` from `plotly`. We found that this package resulted in quicker visualisation process and an included interface for easy exploration of the 3D object.

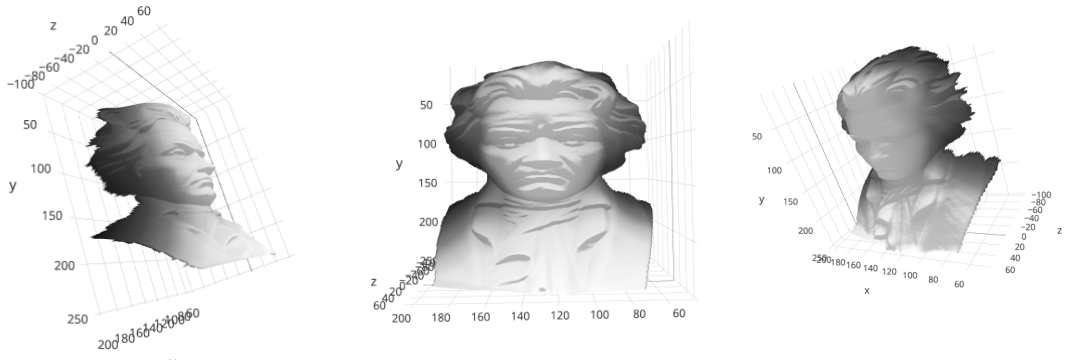

Figure 3: Beethoven Depth Map

4 Buddha

The Buddha dataset consists of ten real images. Since they're not synthetic as the Beethoven images, we expect the resulting albedo and depth maps to be not quite as good.
A selection of three images from the dataset can be seen in Figure 4.

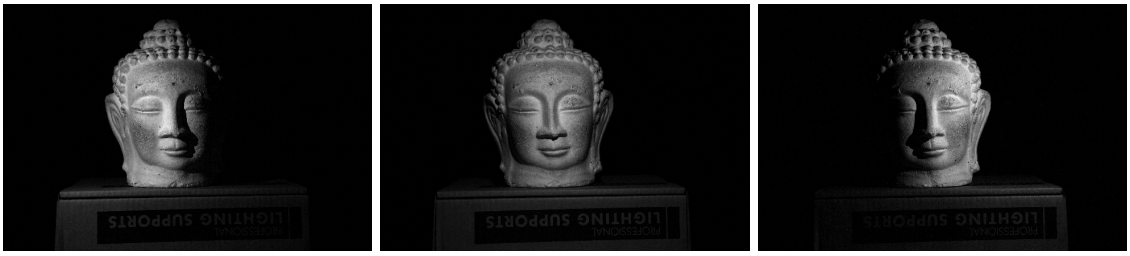

Figure 4: A selection of the Buddha Images (0, 4 and 8)

4.1 Albedo Map

The Albedo representation of the Buddha images has been extracted with the same approach as for Beethoven. The resulting map can be seen in Figure 5. It does seem to have been influenced slightly by the noise in the original images. Otherwise, the results if as we would expect, a roundish shape (the same shape as the mask used) without any highlights or shadows.

4.2 Depth Map

Figure 6 shows the resulting depth map for the Buddha images. The result is shown from several different angles. As expected, the result is not nearly as good as for the Beethoven image. It is a roundish, mostly flat shape, where some structures of the face can be detected, but only slightly. It seems as if the shape has been skewed diagonal to the z and x axis. In the center and the right image in the top row, the contours of the face can be sensed, especially the left eye, edge of nose and the mouth. The bottom row shows that the result is not entirely flat, although it is close to.

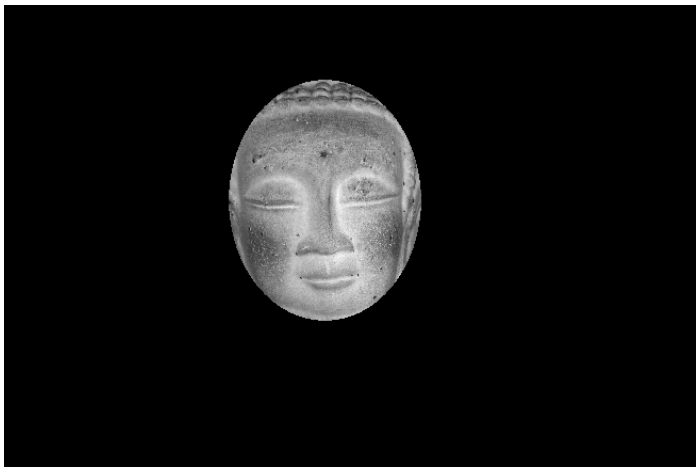

Figure 5: Buddha Albedo

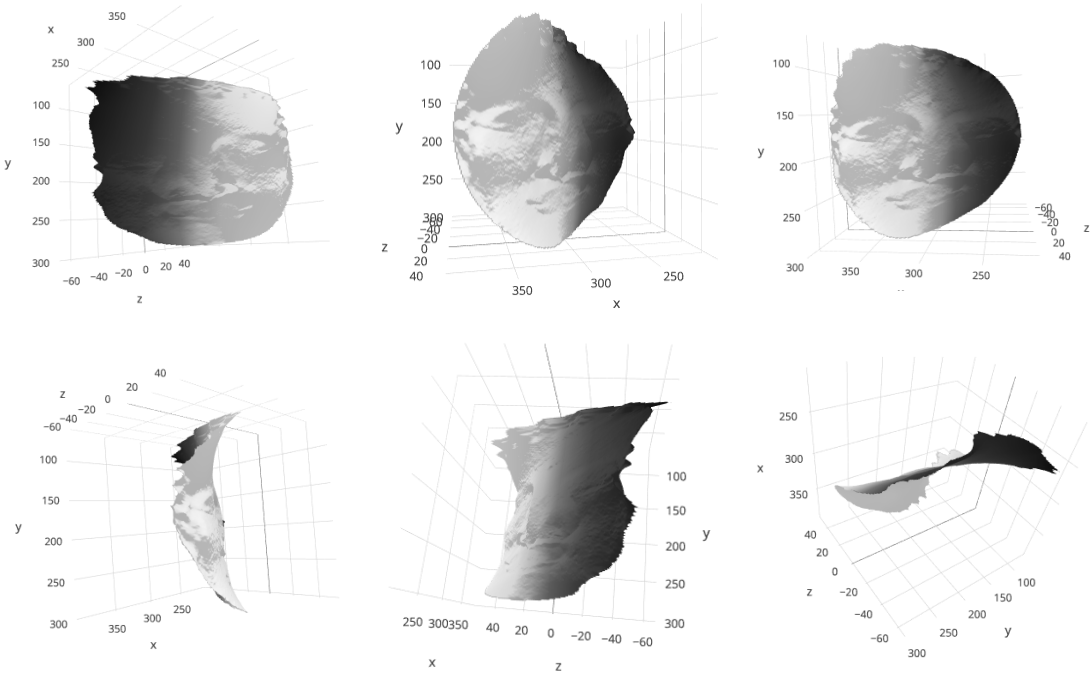

Figure 6: Buddha Depth Map

5 Reflections, Further Implementations

The results from the two datasets show as expected that our algorithm works better for the synthetic images than for the real images.

For the Buddha dataset one might consider utilizing a matrix inversion, instead of the pseudo inversion. As the matrix inversion requires the light matrix to be square in shape, considering that

the light vector for each image has 3 dimensions, exactly three images are required for the inversion. In order to optimize the result, we suggest to take the three images corresponding to the light vectors with the highest determinant. Matrix determinant is a measure of the difference of vectors' directions, and by choosing the combination of vectors with the highest determinant we assure that the light's directions on the chosen image is significantly different. This allows us to inverse the matrix and calculate M from it. An attempt of this approach can be seen in the script.

However, the pseudo inversion should work better if the dataset is very large (more than 1000 images). It would be interesting to see the results of a real dataset like Buddha, but with a significantly larger amount of images.

An additional approach to selecting the three best images is to ensure that the three images also have good measurements, i.e. that they're not too shadowy and not too bright. This can be done manually, or a function can be set up to calculate it automatically.

We have tried experimenting with taking only three images from the Buddha dataset. But unfortunately, because of the way the integration function is defined, even the images with the highest determinant returns NaN's.