VIP: Assignment 2

Ulrik Stuhr Larsen, Frederik Voss, Casper Bresdahl
KU ID: tvr168, lvw655, whs715

December 16, 2019

Contents

1 A bit of modelling **1**
1.1 Write for a generic pixel, light source and normal, Lambert's law in its simplest form . . . 1
1.2 How is Lambert's law modified to deal with self shadows 1
1.3 What about cast shadows? Comment on the difference between the two 1
1.4 Comment on the modelling limits of Lambert's law 2
1.5 How can we obtain an estimate of albedo and normals in Woodham's approach to Photometric Stereo. Write the equation 2
1.6 What should be done if one uses RANSAC. Please describe 2

2 Implementation **3**

3 Beethoven Dataset **3**

4 mat_vase Dataset **4**

5 shiny_vase Dataset **5**

6 Buddha Dataset **7**

7 face Dataset **8**

1 A bit of modelling

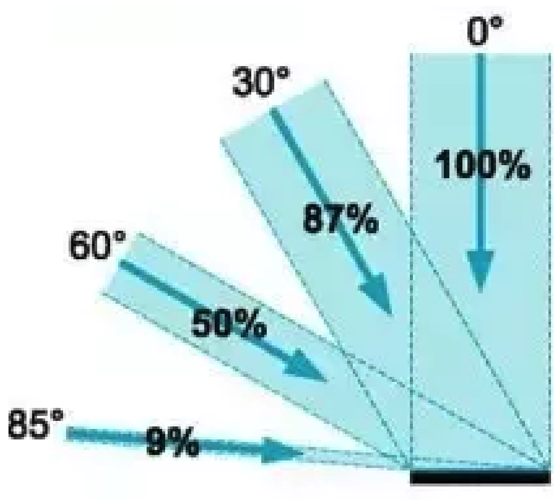
We have been asked to provide answers to the following questions.

1.1 Write for a generic pixel, light source and normal, Lambert's law in its simplest form

The linearised Lambertian model is given by:

$$I(p) = \rho(x)s(x) \cdot n(x) \tag{1}$$

and describes Lambert's cosine law [2]. This law states that the irradiance on a surface area is proportional to the cosine of the angle between the illuminating source and the surface norm. Because a Lambertian scatterer will then scatter the light according to the same law as an Lambertian emitter, the radiance on a surface will depend on the angle between the illuminating source and the surface norm, but it will not depend on the observation angle [3].



(a) Illustration taken from [5].

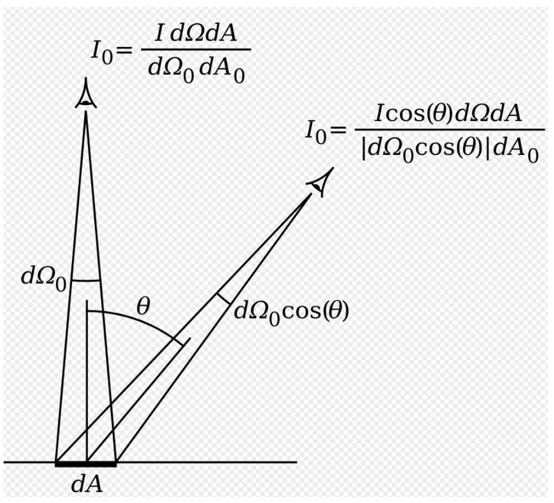

$I_0 = \frac{I\,d\Omega dA}{d\Omega_0\,dA_0}$
$I_0 = \frac{I\cos(\theta)d\Omega dA}{|d\Omega_0\cos(\theta)|dA_0}$
$d\Omega_0$
θ
$d\Omega_0\cos(\theta)$
dA
(b) Illustration taken from [3].

Figure 1: As seen in (a), the irradiance on the surface depends greatly on the angle between the illuminating source and the surface norm, but the radiance observed at different angles remains invariant as illustrated in (b).

Looking back on equation 1, $I(p)$ denotes the intensity at the pixel p, $\rho(x)$ the albedo at location x, $s(x)$ the vector from the location x to the light source and $n(x)$ the surface norm direction at location x.

1.2 How is Lambert's law modified to deal with self shadows

A self shadow is a shadow cast on part of a object because that part is facing the opposite direction of the light source. Given the self shadowed part of the object is facing the opposite direction of the light, the surface norm will also point in the opposite direction. This means that if $s \cdot n \leq 0$ it is a self shadow. We can thus modify Lambert's cosine law to deal with self shadows by saying:

$$I(p) = \rho(x)max(s(x) \cdot n(x), 0)$$

1.3 What about cast shadows? Comment on the difference between the two

Cast shadows are created by one object, but might 'hit' another object in the scene. Because Lambert's cosine law is describes what happens in a single pixel (i.e. is local) we can not handle cast shadows as they may not be a product of the object they 'hit'. On the contrary, given self shadows are produced by the same object which casts them, they are local issues.

1.4 Comment on the modelling limits of Lambert's law

As discussed above, cast shadows can not be modelled by Lambert's cosine law. It is also assumed that the surface is diffuse (no specularities), and that the light emitted from the light source is parallel.

1.5 How can we obtain an estimate of albedo and normals in Woodham's approach to Photometric Stereo. Write the equation

From [6] we get the equation:

$$I = qNn$$

Where I denotes the image, q the reflectance factor (albedo), N the direction of incident illumination and n the unit surface normal. The reflectance factor is then given by:

$$q = |N^{-1}I|$$

And the normals can be obtained by:

$$n = \left(\frac{1}{q}\right) N^{-1}I$$

1.6 What should be done if one uses RANSAC. Please describe

RANSAC is a non-deterministic algorithm used to estimate model parameters where the data contains outliers which should not contribute to the solution. The algorithm is non-deterministic in the sense that it is only guaranteed to find a reasonable solution with a certain probability. This probability increases as the number of iterations allowed in the algorithm increases. The algorithm works by randomly sample n data points needed to estimate model parameters, and then evaluates this model by, for instance, count how many data points are within a thresholded distance to the estimated model [4].

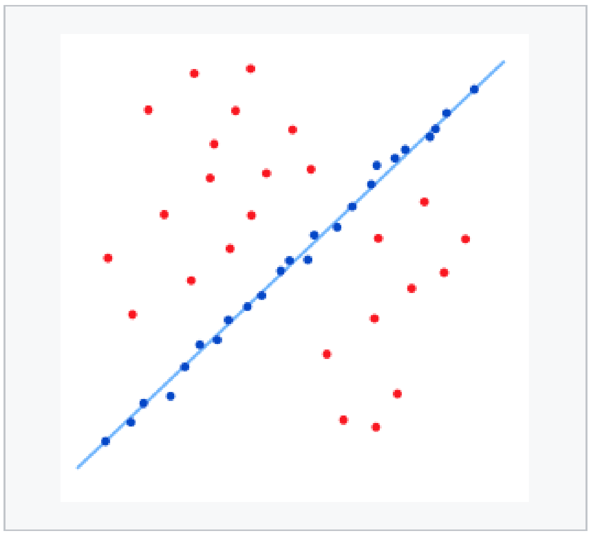


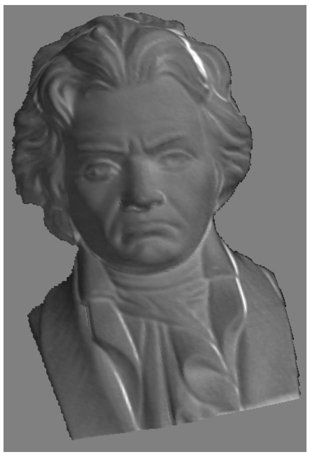
Figure 2: A fitted line with RANSAC goes through the inlier data, while the outlier data is ignored. Had a least square method been used, the line would have been skewed. Illustration from [4].

If one wishes to use RANSAC to estimate the normal field and albedo from a series of images, a matrix of light sources should be constructed together with a vector of pixel intensities from each image in the series, such that if there are z images with k pixels, one would end up with z vectors of dimension k. We can now use the supplied function *RANSAC_3dvector* which implements the RANSAC algorithm. The returned matrix can then be used to solve for the albedo and normal surface vector in that pixel, thus the RANSAC algorithm will have to be used on every pixel in the image.

2 Implementation

The results shown below are all created by the following steps. We first flatten all the images, and thus construct an array with dimensions k and size of the the image, where k is the number of images used. We then flatten the mask, and in a loop we check whether the pixel we are at, is true or not in the mask. If it is true, we copy the pixel value over in a new array. We call this array J. When using RANSAC, we then in a loop construct a temporary vector of size k. This vector and the light source matrix is then passed to *ransac_3dvector* which estimates the matrix m from which we extract the surface norm directions and the albedo as the length of the norm vector. When using the pseudo inverse method, we use *np.linalg.pinv* and pass it the light source matrix and the resulting matrix is then multiplied by J. We can then extract the surface norm directions and the albedo.
Afterwards we put back in the black pixels in the image by using a loop to check in the flattened mask, whether the value is true or false. If it is false the value in the array will be zero, else it is the value of the surface norm. We get the albedo by reshaping the array, and the normal field by using *unbiased_integrate*.

3 Beethoven Dataset


(a) Beethoven from the front

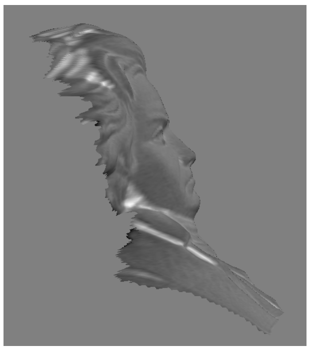

(b) Beethoven from the side

Figure 3: Beethoven in 3D

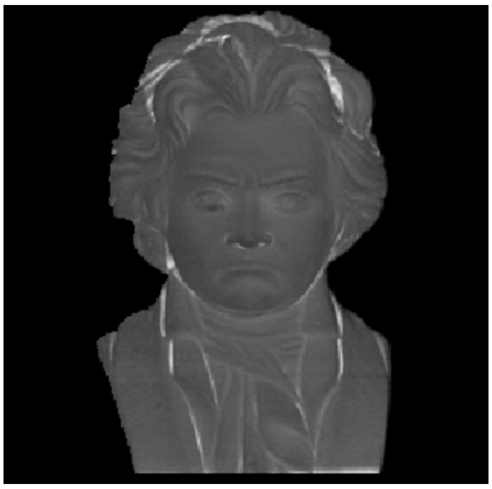
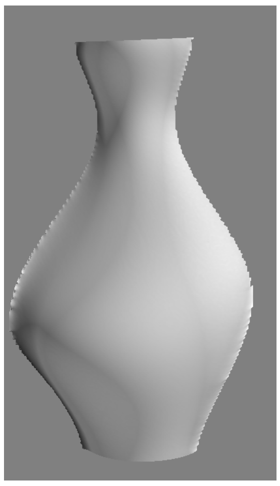


Figure 4: Albedo for Beethoven

4 mat_vase Dataset


(a) mat_vase from the front

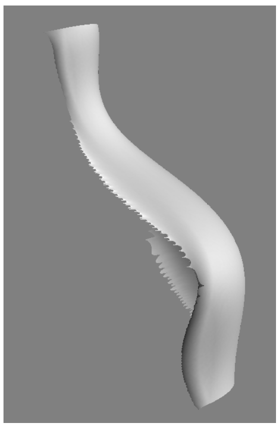

(b) mat_vase from the side

Figure 5: mat_vase in 3D

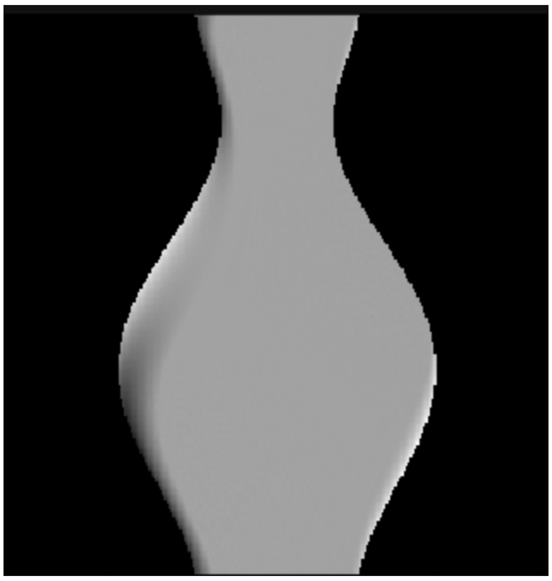


Figure 6: Albedo for math_vase

5 shiny_vase Dataset

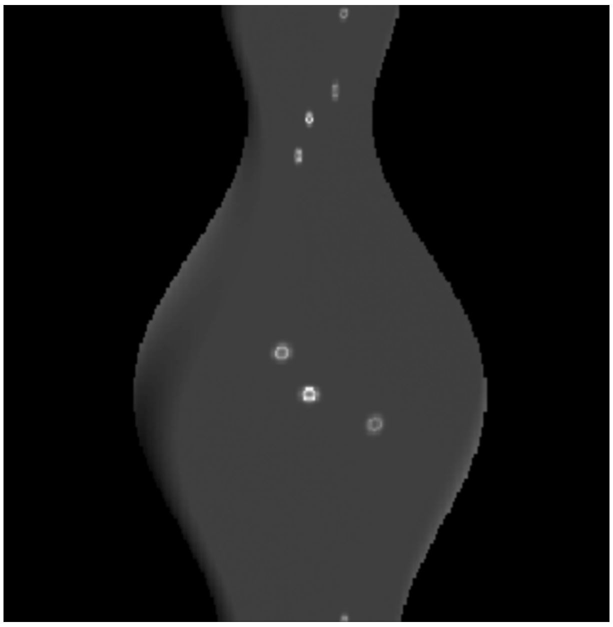
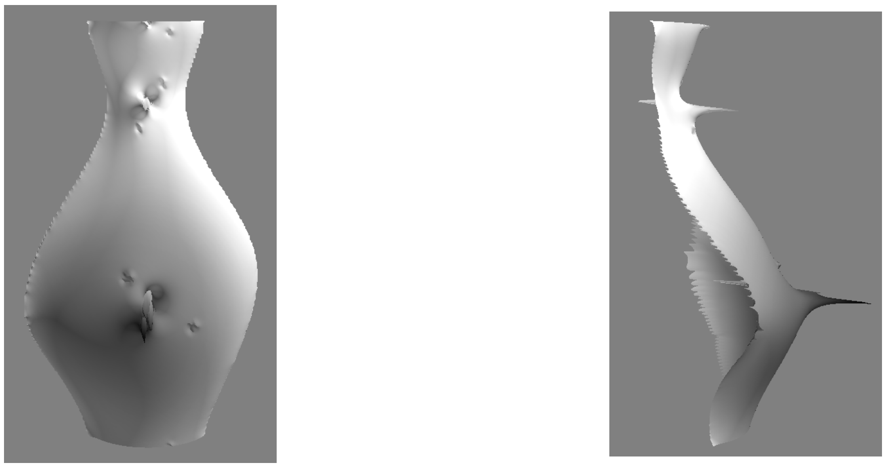


Figure 7: Albedo for shiny_vase



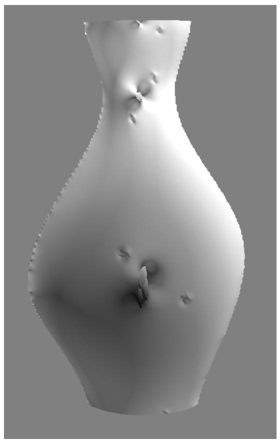
(a) shiny_vase from the front

(b) shiny_vase from the side

Figure 8: Shiny_vase in 3D

Due to the specularity on the vase in the original images, there is a large difference in albedo in this area. It appears that the algorithm interprets these with a laplacian, as this results in two protruding spikes at each of the places of the specularities - one negative and one positive (because the weights in the laplacian are both positive and negative).

Since RANSAC removes the outliers when finding a estimate for ρn, we expect it to perform the same or better than using the library function *np.linalg.pinv.*


(a) shiny_vase from the front with RANSAC

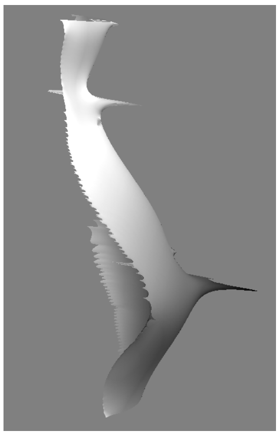
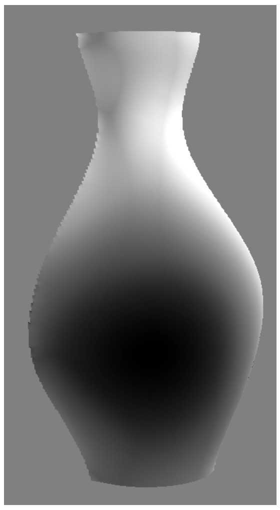

(b) shiny_vase from the side with RANSAC

Figure 9: shiny_vase with RANSAC in 3D

The resulting images and albedos from applying RANSAC appear to be identical to the results from when using the *np.linalg.pinv* function.


(a) shiny_vase from the front with RANSAC and smoothing

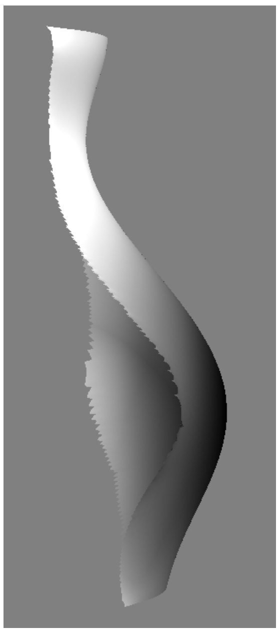
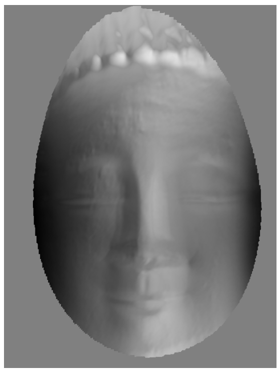

(b) shiny_vase from the side with RANSAC and smoothing

Figure 10: shiny_vase with RANSAC and smoothing in 3D

We see that since the specularities are relatively small and local, they can be smoothed out quite nicely. We observe that the resulting images do not display any of the previously observed spikes.

6 Buddha Dataset


(a) Buddha from the front with pseudo inverse

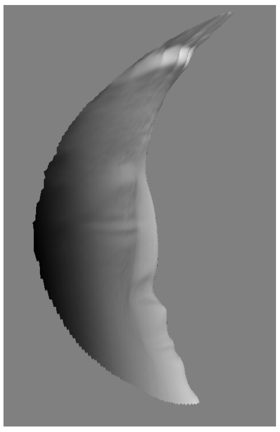
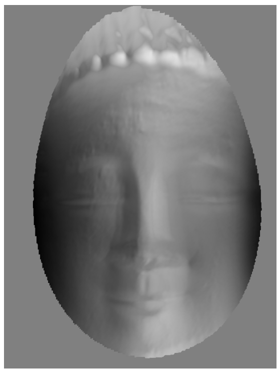

(b) Buddha from the side with pseudo inverse

Figure 11: Buddha with pseudo inverse in 3D


(a) Buddha from the front with pseudo inverse and smoothing

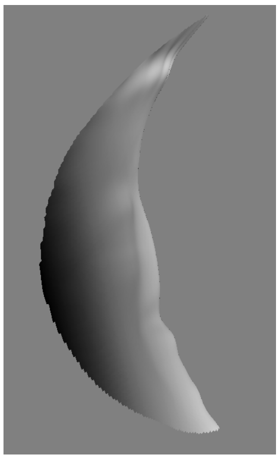
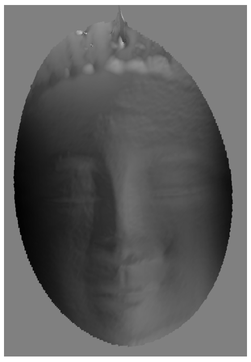

(b) Buddha from the side with pseudo inverse and smoothing

Figure 12: Buddha with pseudo inverse and smoothing in 3D


(a) Buddha from the front with RANSAC

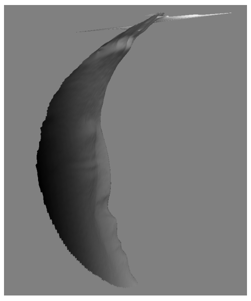

(b) Buddha from the side with RANSAC

Figure 13: Buddha with RANSAC in 3D

In Figure 13b and Figure 12b we see that the estimation using RANSAC is visibly worse than using the *np.linalg.pinv* function. The top of the buddha face has protrusions similar to the ones observed on the non-smoothed versions (both RANSAC and pseudo-inverse) of the shiny_vase pictures. The image representation appears significantly darker due to the depth difference imposed by the large positive spike.

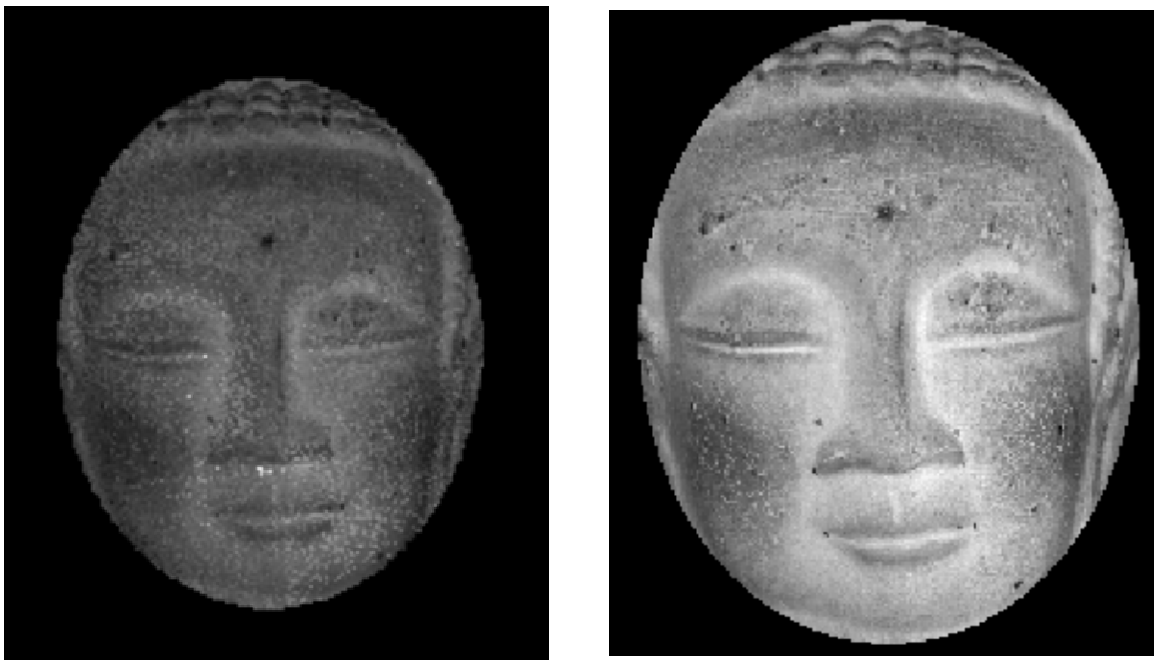

(a) Albedo for Buddha with RANSAC (b) Albedo for Buddha with pseudo inverse

Figure 14: Buddha albedos

7 face Dataset

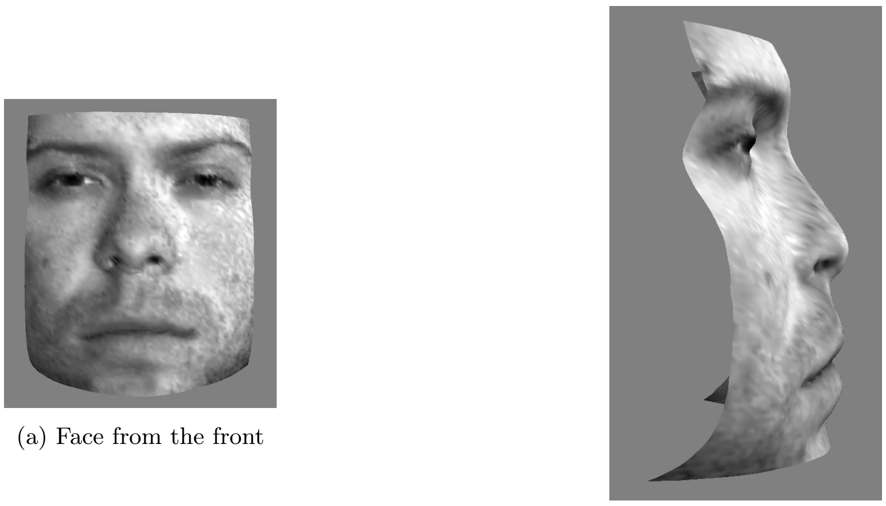
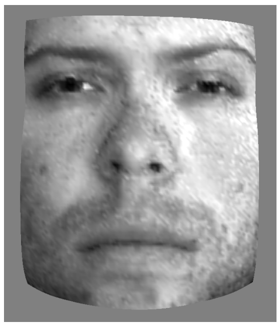
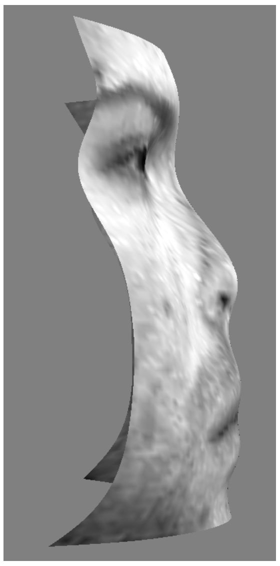

(a) Face from the front (b) Face from the side

Figure 15: Face in 3D

The result from using RANSAC with 27 picture is a fairly accurate representation of the face. There are no odd spikes as seen in some of the previous images when using RANSAC (such as Figure 8), and it seems like the geometry of the face is captured pretty well.



(a) Face from the front with smoothing



(b) Face from the side with smoothing

Figure 16: Face with smoothing in 3D

When applying smoothing on the face, it looks like the face has been smashed from the front. We see that the natural protrusion of the nose and lips are reduced such that they almost blend into each other, giving the appearance of a very flat face. The folds at the cheek also nearly disappear as a result of this smoothing. This negative effect is not as present when smoothing some of the previous images (Figure 8). For example, the shiny_vase did not have any detailed features to preserve, opposite to the face. It is also worth noting that smoothing does not really change anything if the result is seen from the front. This is expected since it is impossible to see depth in 2D (although depth can be inferred from other knowledge).

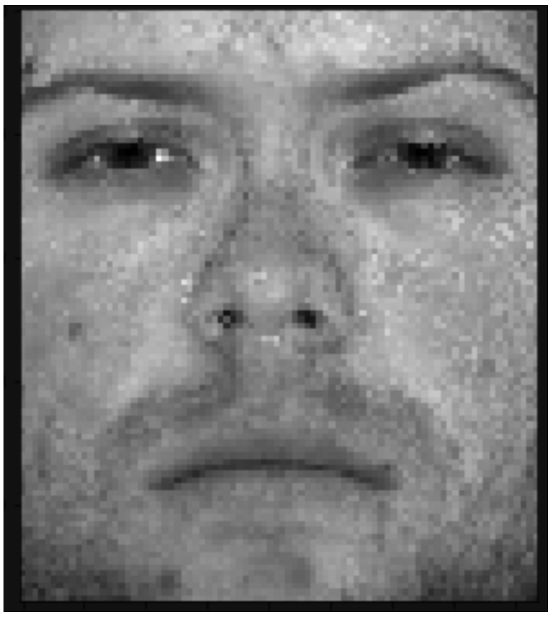


Figure 17: Albedo for face

References

[1] Forsyth, David A., Ponce, Jean. 2012. 'Computer Vision a Modern Approach'. Pearson 2nd. ed.

[2] Absalon.ku.dk. 'PS.pdf'. `https://absalon.ku.dk/courses/36154/files/folder/Slides?preview=3307133`. Visited: December 7th 2019.

[3] en.wikipedia.org. 'Lambert's cosine law'. `https://en.wikipedia.org/wiki/Lambert%27s_cosine_law`. Visited: December 7th 2019.

[4] en.wikipedia.org. 'Random sample consensus'. `https://en.wikipedia.org/wiki/Random_sample_consensus`. Visited: December 7th 2019.

[5] quora.com. 'What is Lamberts Cosine Law?'. `https://www.quora.com/What-is-Lamberts-Cosine-Law`. Visited: December 7th 2019.

[6] Robert J. Woodham. 1980. 'Photometric method for determining surface orientation from multiple images'. Department of Computer Science University of British Columbia.