Vision and image processing
Assignment 3: Photometric Stereo
University of Copenhagen
December 16, 2020

Mark Ambrodji
qpf823@alumni.ku.dk

Francesco Done'
qwg586@alumni.ku.dk
Exchange Student

Michelle Low
mdc577@alumni.ku.dk

Polina Olander
wdb694@alumni.ku.dk

1 A bit of modelling

- $I(x) = \rho(x)s(x) \cdot n(x)$, where $\rho(x)$ is the albedo at x and $\rho \in [0, 1]$, $s(x)$ is the light source direction and $n(x)$ is the surface normal direction.
- $I = \rho\ max(s \cdot n, 0)$: If the surface is in front of the light source, then the value of $max(s \cdot n, 0)$ becomes 0, but if the surface is behind the light source, then the intensity becomes $s \cdot n$ as $s \cdot n \leq 0$ when the light does not hit the pixel. At that point ρ scales the intensity based on the albedo of the object.
- Labert's Law does not cast shadows, as it is a local law. Self shadows reflect the intensity on the objects surface (local), where cast shadows is a reduction of intensity on a different surface dependent dependent of objects in front of it (global).
- Labert's Law only include the albedo of an object. Given this it can only handle matte object.
- The estimated Albedo p and Normals n are calculated with the formulas:

$$m = p \cdot n \qquad p = ||m|| \qquad n = \frac{m}{p}$$

in which m is known a priori since it is calculated with the formula:

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

- Using Ransac requires to estimate the 3D vector m for every pixel, so basically if the input image has xxy pixels, the algorithm is executed xxy times. While executing it, is needed to save the estimated m for the pixel i, inside a new matrix of size xxy in the i-th position.

2 Beethoven Dataset

Firstly we defined the minimum rectangle that includes the non-zero area in $mask$ array. After that, the $mask$ and the I arrays have been resized based on the interesting area (shp_i and shp_mask), in this way it has been possible to calculate J and S, so also S^{-1}. Also with these variables, the albedo modulated normal field has been obtained:

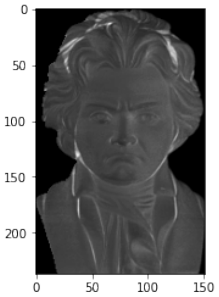


Figure 1: Obtained Albedo for Beethoven dataset

Since the size of M was (3, 35787), it was necessary to reshape it based on three dimensions, so the final size was (3, 237, 151): this happened because before calculating M, J has been reshaped in this way: (3, 237*151), so it was possible to multiply it by the inverse of S. After that p has been calculated as the norm of M and N as M/p, it represents the set of components to be used in the $unbiased\_integrate$ function, with the shaped mask shp_mask. Finally, from the provided ps_utils, the function $display\_surface\_matplotlib$ has been used for displaying the results, from different points of view as shown here below:

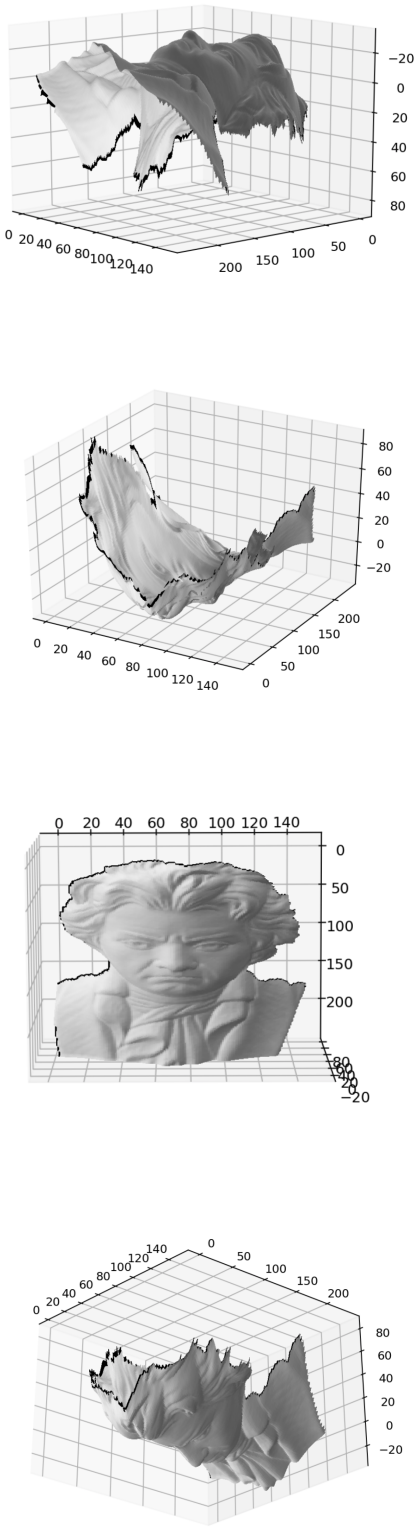


Figure 2: Results of Task 2, from different points of view

3 mat_vase Dataset

Since the *mat_vase* dataset is similar to the one used in *Task 2*, from the point of view that both contain three synthetic images, the program used was the same. The only difference is the size of interesting area, that is (3, 255, 144) instead of (3, 237, 151). The results are shown here below:

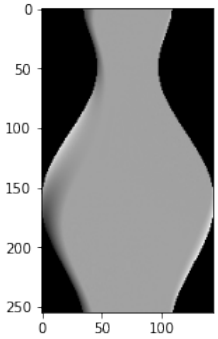
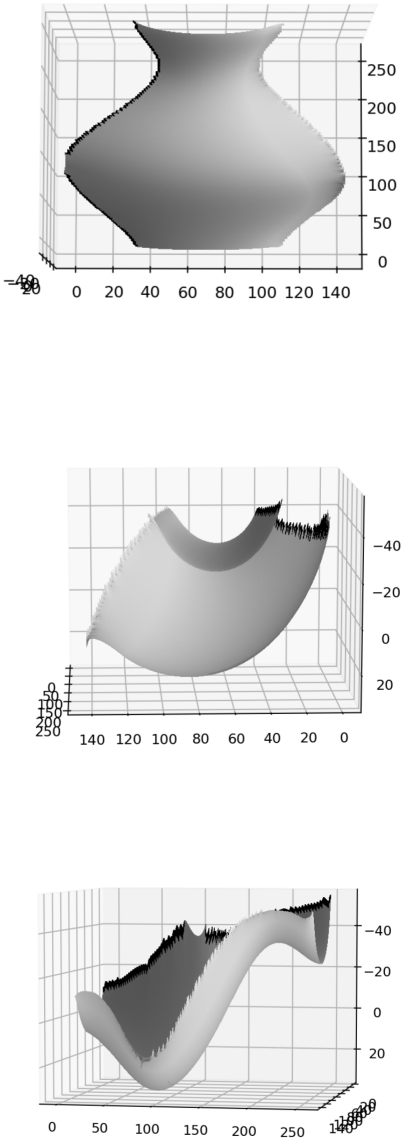


Figure 3: Obtained Albedo for mat_vase dataset



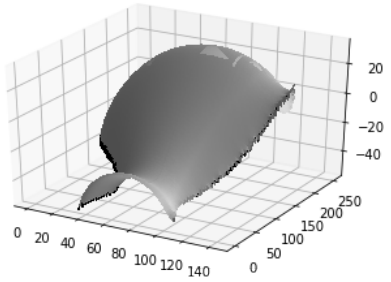


Figure 4: Results of Task 3, from different points of view

4 shiny_vase Dataset

For this task, a dataset with three synthetic images was used. Before calculating the Albedo, as explained in $task2$ and $task3$, Ransac algorithm was executed for estimating m: since the shape was (255, 144, 3), that is basically 3 images of size 255x144, it was needed to normalize it in a way that it was possible iterate over the pixels easily, so the new shape was (36720, 3). After this, the algorithm was executed pixel-by-pixel, saving the estimated m inside the matrix M_ransac, that has the same size. After that, n and p were calculated using M_ransac, the results are shown below:

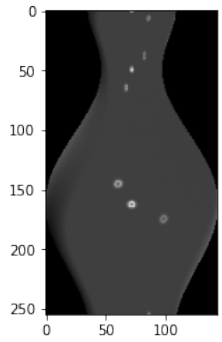


Figure 5: Obtained Albedo for shiny_vase dataset with Ransac estimation (same result without Ransac algorithm)

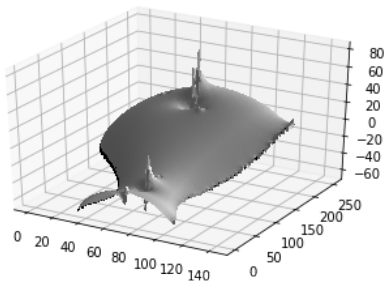


Figure 6: Results of Task 4, without applying Ransac algorithm

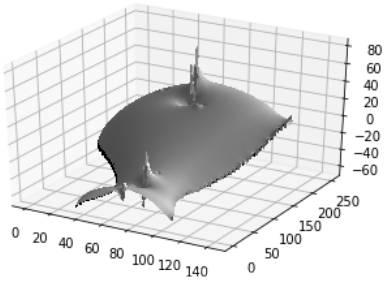


Figure 7: Results of Task 4, applying Ransac algorithm

Since the results obtained with ransac and without it are pretty the same, the $smooth\_normal\_field(...)$ algorithm has been used for improving the final result:

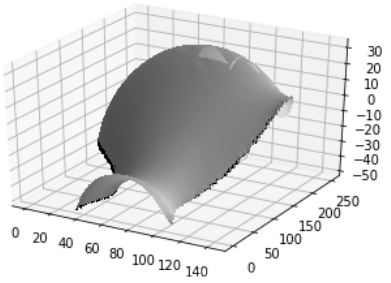


Figure 8: Results of Task 4, applying Ransac algorithm with smoothness

As is possible to see, the smooth version is definitely better than the others.

5 shiny_vase2 Dataset

The dataset used for this task includes 22 synthetic images, the process used is the same as $task4$. Differently from the previous task, the Albedo calculated with and without Ransac algorithm are completely different, moreover the formula used for obtaining M wasn't the one used for the previous tasks, but it was used the pseudo-inverse of S with the function $pinv$ provided in $numpy.linalg$ module. So the formula used is:

$$M = S^{\dagger} \cdot J$$

Below there are some figures representing the Albedos with and without Ransac algorithm, and the final results:

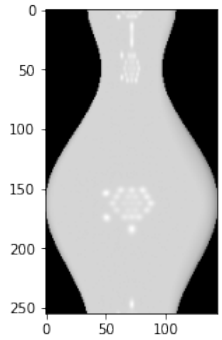


Figure 9: Obtained Albedo for shiny_vase2 dataset

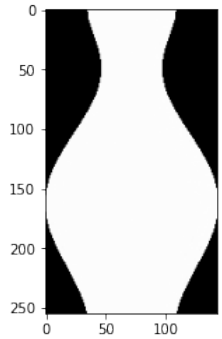


Figure 10: Obtained Albedo for shiny_vase2 dataset with Ransac estimation

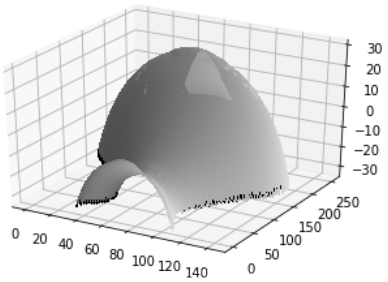


Figure 11: Results of Task 5, without applying Ransac algorithm

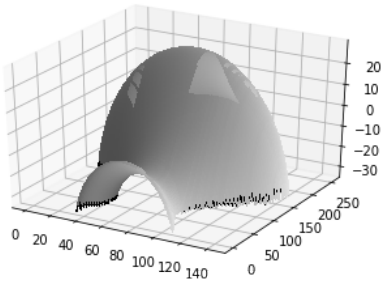


Figure 12: Results of Task 5, applying Ransac algorithm

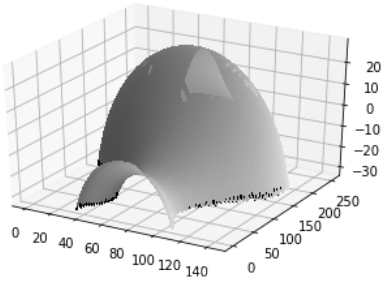


Figure 13: Results of Task 5, applying Ransac algorithm with smoothness

6 Buddha Dataset

A dataset containing 10 real images has been used. The particularity was that the $threshold$ parameter in the Ransac algorithm has been set to an high value (150.0 for these results) in order to make a good Albedo estimation: with low values the Albedo was too dark, so it was no possible to obtain 3D representations with that Albedo.

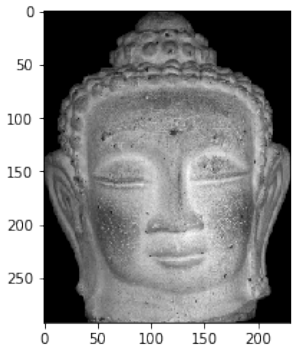


Figure 14: Obtained Albedo for Task 6 without Ransac estimation

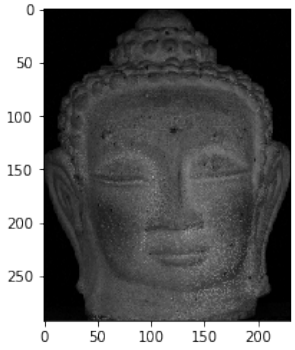


Figure 15: Obtained Albedo for Task 6 with Ransac estimation

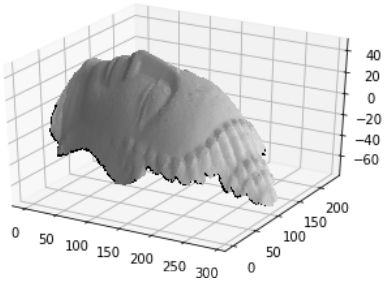


Figure 16: Results of Task 6, without applying Ransac algorithm

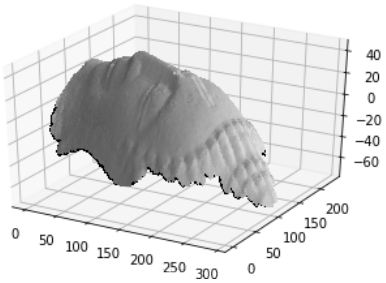


Figure 17: Results of Task 6, applying Ransac algorithm

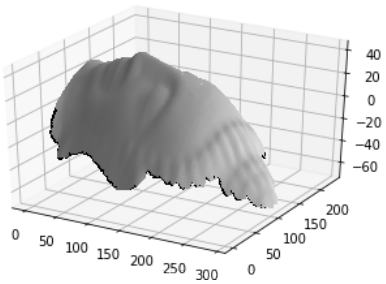


Figure 18: Results of Task 6, applying Ransac algorithm with smoothness

7 face Dataset

The $face$ dataset has been used, which includes 27 real images. The program used is the same as on the previous tasks (the one with pseudo-inverse of S). Below it is possible to see the Albedo obtained via Ransac (with $threshold = 10.0$), and the 3D representations with different smoothness.

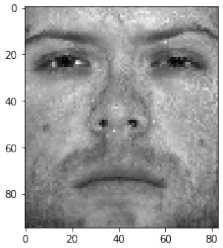


Figure 19: Obtained Albedo for Task 7 with Ransac estimation

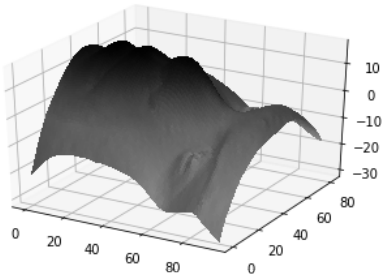


Figure 20: Results of Task 7, without applying smoothness

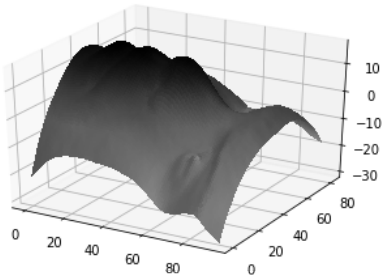


Figure 21: Results of Task 7, applying Ransac algorithm with smoothness, $iters = 5$

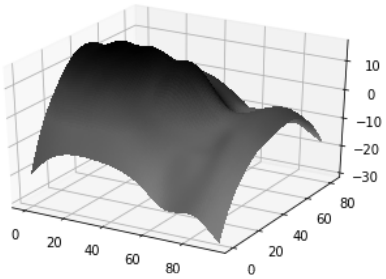


Figure 22: Results of Task 7, applying Ransac algorithm with smoothness, $iters = 100$

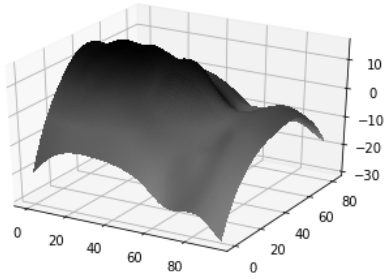


Figure 23: Results of Task 7, applying Ransac algorithm with smoothness, $iters = 1000$

It is easy to see that the smooth representations are the same for a high value of $iters$ parameter, while for lower values (like 5), it is possible to see a real face less sharp than the one without smoothness.

8 Using the program

With this report is possible to find also a folder containing:

- *datasets*: Beethoven.mat, Buddha.mat, face.mat, mat_vase.mat, shiny_vase.mat and siny_vase2.mat;
- *python files*: assignment3.py, ps_utils.py, task2.py, task3.py, task4.py, task5.py, task6.py, task7.py;

Running the command on the terminal

```
python assignment3.py
```

it is possible to run the main program. Afterwards, it is possible to launch the tasks individually by pressing the buttons:

```
2 - Task 2, Beethoven Dataset
3 - Task 3, mat_vase Dataset
4 - Task 4, shiny_vase Dataset
5 - Task 5, shiny_vase2 Dataset
6 - Task 6, Buddha Dataset
7 - Task 7, face Dataset
```