Computer Vision 1
Assignment 1
Photometric Stereo & Color

Nils Hulzebosch (10749411) & Michelle Appel (10170359)

February 21, 2018

1 Introduction

In this report the findings of several photometric stereo, image recoloring and image decomposition experiments are discussed. The report is divided into sections that follow the same pattern as the exercise setup.

2 Photometric Stereo

In this experiment, a patch of surface will be reconstructed from a series of pictures of the surface, taken under different illuminants.

2.1 Estimating Albedo and Surface Normal

To reconstruct a patch of surface, the albedo and surface normal need to be estimated by analyzing several images of the surface, taken under different light source locations. The albedo $\rho(\mathbf{x})$ and surface normal $N(\mathbf{x})$ can be estimated using the given intensity $I(x, y)$ of a pixel located at (x, y), which is given by

$$\begin{aligned} I(x,y) &= \rho(x,y)\mathbf{N}(x,y) \cdot \mathbf{V}_1 \\ &= g(x,y) \cdot \mathbf{V}_1 \end{aligned}$$

where $\mathbf{V}_1 = k\mathbf{S}_1$, where k is a constant that describes the sensitivity of the camera and $\mathbf{S}_1$ is the light source vector. In this case the intensity I and $\mathbf{V}_1$ are given, so that $g(x, y)$ can be estimated by solving the LSE solution, using Matlab's *linsolve*, which solves $I(x, y) = g(x, y) \cdot \mathbf{V}_1$ for $g(x, y)$, for every pixel.

After estimating $g(x, y)$ for every pixel, the normal vectors and albedo values can be derived. The albedo $\rho(x, y) = |g(x, y)|$ and the normal vector $\mathbf{N}(x, y) = \frac{g(x,y)}{|g(x,y)|}$.

2.1.1 Albedo and normal field

For the SphereGray5 images it is expected for the albedo to contain two different values on the sphere, one for the two white surfaces and one for the two gray surfaces, and one other for the background (which is the area that is never illuminated, thus will be black). However, the result shows a slight gradient towards the visible edges of the shape as can be seen in figure 1 in the albedo image.

2.1.2 Number of images used for estimation

The minimum number of images needed to estimate the albedo and surface normal is 3, making it possible to solve with a squared solution. Naturally, the more images used for estimating, the better the result, as the contribution per individual image decreases as more images are used. This is demonstrated in figure 2, which shows the results of estimating the albedo and normal vectors using 25 images. Improvement can

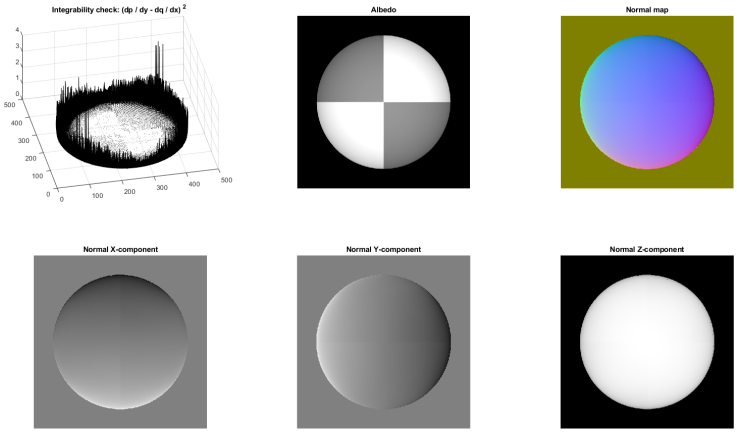


Figure 1: Integrability, albedo and normals of the SphereGray5 dataset

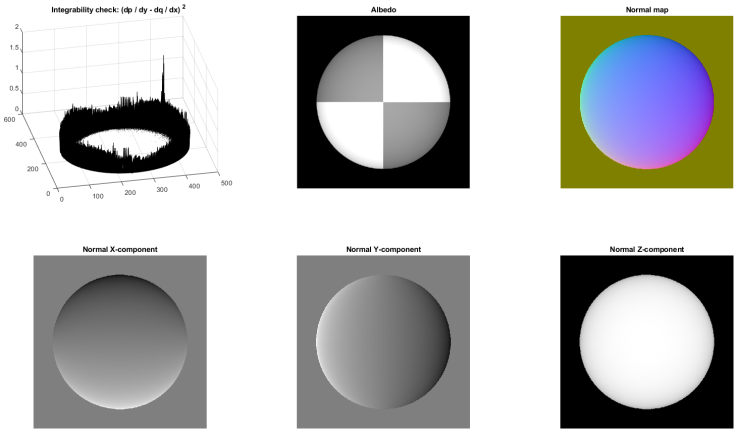


Figure 2: Integrability, albedo and normals of the SphereGray25 dataset

be observed with respect to the SphereGray5 dataset, as the former shows a more even albedo image. The latter estimates a larger errors near the edges of the visible surface as well as on the visible surface area. A higher intensity of a pixel $I(x, y)$ indicates a larger albedo value or the extent of the surface normal being in the direction of the light source. It seems that large albedo values in reality are mistaken for normal vectors being directed towards the camera, as the error within white areas of the sphere are larger than within the gray areas.

2.1.3 Shadow trick

The shadow trick eliminates the contribution of the pixels $I(x, y) = 0$, meaning that a pixel is completely shaded, for solving $g(x, y)$.

Disabling the shadow trick dramatically deteriorates performance of solving $g(x, y)$ using 5 images, as can be seen in figure 3, as clear traces of the light sources can be spotted, resulting in a rugged normal vector field where the vector field is expected to be smooth. This effect is a result of the $g(x, y)$ being over fitted to the low amount of images, making instabilities visible which appear at areas where transition between illumination and shadow occurs, because of the small values $I(x, y)$ takes on such edges. Using more images to make an estimation will fade out the contribution of these instabilities for solving $g(x, y)$, as can be seen

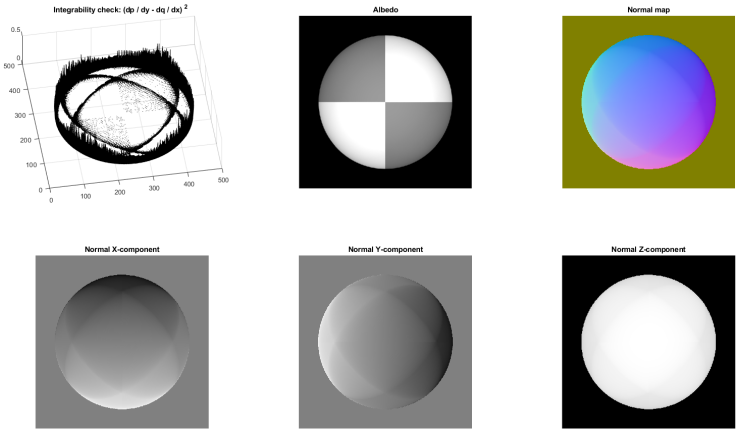


Figure 3: Integrability, albedo and normals of the SphereGray5 dataset with disabling of the shadow trick

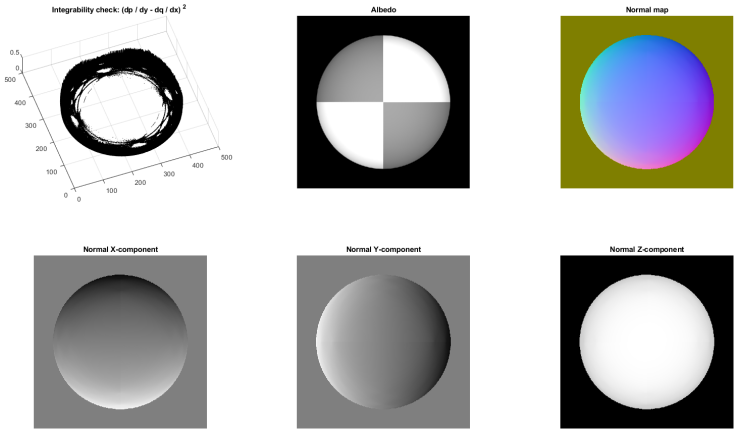


Figure 4: Integrability, albedo and normals of the SphereGray25 dataset with disabling of the shadow trick

in figure 3.

2.2 Test of Integrability

The test of integrability can be used to determine if the estimated derivatives will lead to a fluent surface. The derivatives over x and y are reconstructed from the estimated normal vectors. For a smooth surface it is expected that $\frac{\partial^2 f}{\partial x \partial y} = \frac{\partial^2 f}{\partial y \partial x}$ so a threshold of a difference < 0.01 between the two second order derivatives is set to be acceptable.

In all cases appears the error to be the largest at areas where the surface seems to be approaching or is perpendicular to the camera. This is due to the fact that the estimated normal vector should have 0 for the z-component, but is initialized as one before being normalized, making the x or y-components approaching infinity. This causes unstable arithmetic operations, explaining the large errors. Also, for the SphereGray5 dataset, there are 4 areas that are perpendicular to the camera, that is only illuminated in one image, defining only derivatives in one of the x and y directions, but not in the other, explaining why it can be seen in figures 1 and 3 that the error is the largest at the visible edges in the direction of the light source.

2.3 Shape by integration

To get the depth of the surface using the calculated derivatives over x and y, given some starting point the integral over the pad towards that point can be taken which is given by

$$f(x,y) = \oint \{\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\} \cdot d\mathbf{l} + c$$

where c is the starting depth.

To reconstruct the surface of the patch, the derivatives over x and y derived from the estimated normals can be used. Given some starting point with some depth, going one step (pixel) in some direction and get the height of that point by adding the derivative at that point to the height of the last point. The surface can be constructed in the column direction and in the row direction. The average can be taken by simply adding those two height maps and dividing them by 2. Results are shown in figure 6.

2.4 Experiments with different objects

Estimating the albedo and normal field for the MonkeyGray dataset results in more errors than the sphere datasets as can be seen in figure 5, because the monkey figure contains more visible edges that are perpendicular to the camera, resulting in instable aritmatics, thus more error.

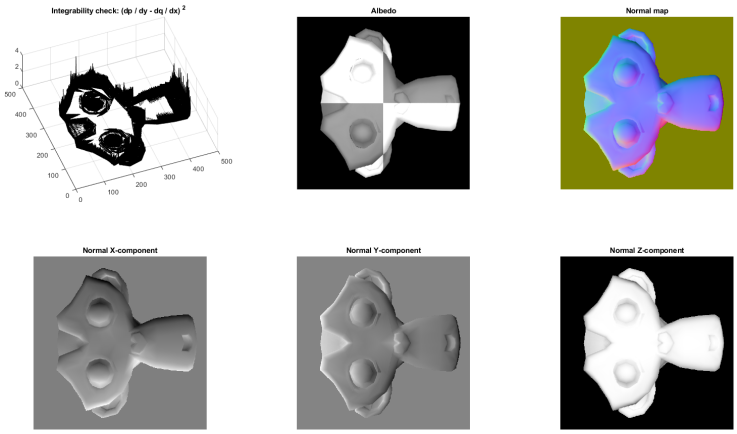


Figure 5: Integrability, albedo and normals of the MonkeyGray dataset

2.4.1 RGB

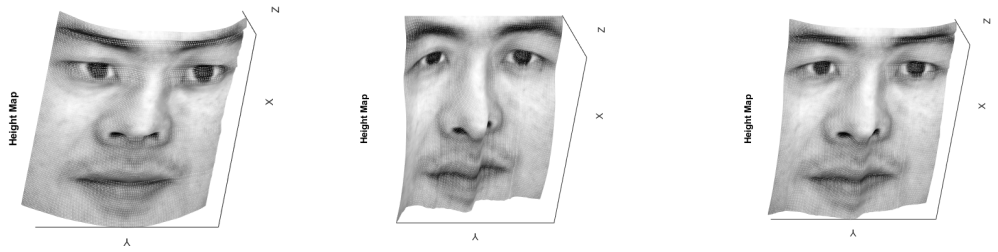
The model is extended to be used with the RGB channel by estimating $g(x, y)$ for all three channels separately. The resulting albedos, normals and errors will be combined by taking the average.

3 Color Spaces

RGB Color Model. As explained on Wikipedia[1], the RGB model is chosen because is has good primaries, which results in a large color triangle (and thus has a lot of possible colors).

As explained on the same Wikipedia page, digital cameras use an image sensor that works on the RGB model or a variation of the RGB model. For example, a Bayer filter has twice as many green detectors as red and blue to increase luminance.

[1] `https://en.wikipedia.org/wiki/RGB_color_model`

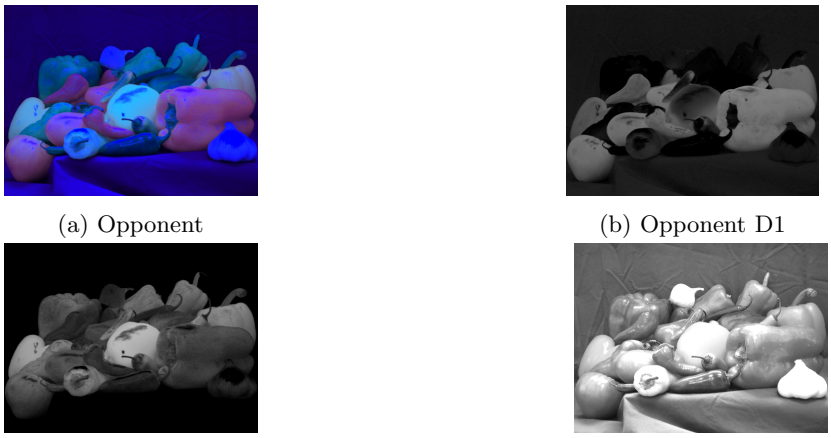


(a) Row directed (b) Column direction (c) Average

Figure 6: Surface reconstruction using a) row directed integration, b) column directed integration and c) taking the average of the row and column directed reconstruction

3.1 Color Space Conversion

The Color Space Conversion methods were implemented and applied onto the given image. The results of all methods are shown in Figure 7 (Opponent Color Space), Figure 8 (Normalized RGB (rgb) Color Space), Figure 9 (HSV Color Space), Figure 10 (YCbCr Color Space), and Figure 11 (Grayscale).



(a) Opponent (b) Opponent D1

(c) Opponent D2 (d) Opponent D3

Figure 7: The Opponent Color Space (a) and each of its dimensions (b, c, d) after conversion.

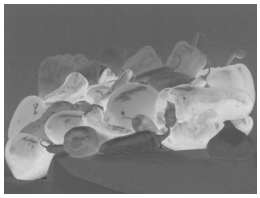
3.2 Color Space Properties

Opponent Color Space. The Opponent Color Space consists of three components. In this case, the first is the red-green channel, the second the blue-yellow channel, and the third the luminance channel.

Normalized RGB (rgb) Color Space. In the Normalized RGB (rgb) Color Space, each channel is normalised by dividing the value with the sum of the three channels. This can remove noise or shadows in the image. In the resulting image, (most) shadows are removed.

HSV Color Space. In the HSV Color Space, the RGB values are mapped to Hue, Saturation, and Value. The first is the color (in this case a point on a circle). The second is the amount of the color (lower = more white/gray, higher = more pure color). The third is the brightness, the lower the more black, the higher the more white. One advantage is that it is more intuitive for humans to see how the color is composed than

(a) Normalised RGB

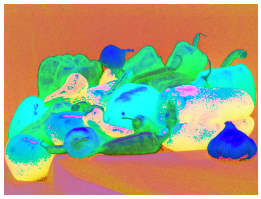


(b) Normalised RGB D1

(c) Normalised RGB D2

(d) Normalised RGB D3

Figure 8: The Normalised RGB Color Space (a) and each of its dimensions (b, c, d) after conversion.



(a) HSV

(b) Hue (D1)

(c) Saturation (D2)

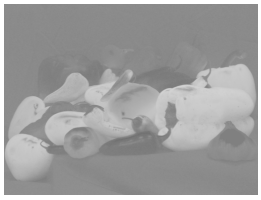
(d) Value (D3)

Figure 9: The HSV Color Space (a) and each of its dimensions: hue (b), saturation (c), and value (d) after conversion.

(a) YCbCr

(b) YCbCr D1

(c) YCbCr D2



(d) YCbCr D3

Figure 10: The YCbCr Color Space (a) and each of its dimensions (b, c, and d) after conversion.

(a) Grayscale (lightness)

(b) Grayscale (average)

(c) Grayscale (luminosity)

(d) Grayscale (rgb2gray)

Figure 11: The grayscale Color Spaces: lightness (a), average (b), luminosity (c), Matlab rgb2gray (d) after conversion.

seeing three RGB values. Therefore this model could be used for choosing a color during all kinds of human tasks, such as design tasks (which can later be mapped to RGB by a computer).

YCbCr Color Space. The YCbCr Color Space consists of a luminance component (Y), blue-difference component (CB), and red-difference component (CR).

As shown in the results, this method is sensitive to yellow (and white) and maps it to a high intensity. This could be beneficial for certain tasks where light (white / yellow) objects have a high priority to be recognised.

Greyscale methods. The following three algorithms are implemented from John Cook[2].

Lightness: take the average of the most prominent and least prominent colors.

Average: take the average of the three color channels.

Luminosity: take a weighted average, inspired by human perception, which is the most sensitive to green.

The fourth algorithm is implemented with the Matlab built-in function, which also takes a weighted average of the three channels, but with other values than those from John Cook.

As shown in the results, the first two methods transform the yellow pepper to a dark color, while the last two methods transform it to a lighter color (because yellow consists of green and red, which are both weighted higher than blue, resulting in a higher intensity). Also the green pepper appear more lightly in the image of the last two methods.

More on Color Spaces. The CMYK color space is, as opposed to the RGB model, a substractive model that is used in printing. The four colors that are used are cyan (C), magenta (M), yellow (Y), and key / black (K). As explained on Wikipedia[3], black ink could also be created by mixing cyan, magenta, and yellow, but it is beneficial to use black ink for reasons such as solidity, refinement (level of detail), speed, and costs.

The advantage of CMYK is that it works better than RGB on printed media, because mixing cyan, magenta, and yellow ink results in a larger spectrum of colors than mixing red, green, and blue ink.

4 Intrinsic Image Decomposition

This section describes the experiments with respect to intrinsic image decomposition. Two methods are shown: image formation and recoloring.

[2] `https://www.johndcook.com/blog/2009/08/24/algorithms-convert-color-grayscale/`

[3] `https://en.wikipedia.org/wiki/CMYK_color_model`

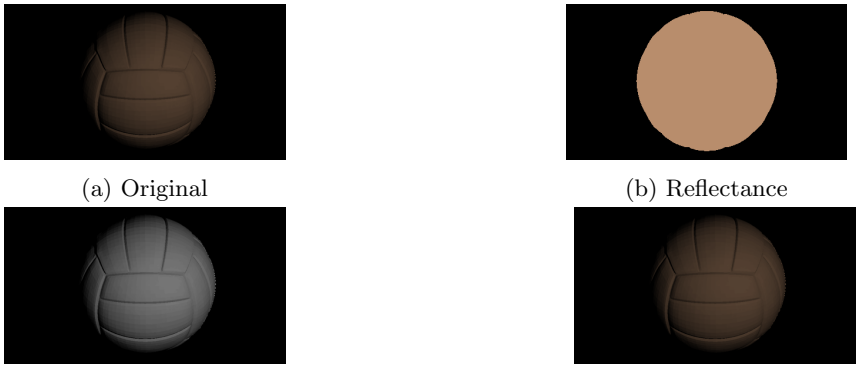
4.1 Image formation

Other Intrinsic Components. Two other components are structure and texture (Aujol & et al., 2006; Evangelopoulos & Maragos, 2008). The structure can be regarded as the large(st) objects, while the texture can be regarded as the details, "usually with some periodicity and oscillatory nature" (Aujol & et al., 2006). In our example, the structure of the ball is the large circle (globe) while the texture represents the material of the ball (for example, leather would have another texture than plastic).

Synthetic Images. Data-sets are often composed of synthetic images, because they can serve as a ground truth, since they are generated computationally. Therefore they can be used for validation. Natural images could also serve as ground truth for validation, but it is much harder to label them correctly, or takes much more time.

Furthermore, it can also be used to test and debug your algorithms, since it is known how the synthetic image is composed and thus how the algorithm should perform.

Image Formation. The image was reconstructed by multiplying the pixel intensity matrices of the reflectance and shading. Figure 13 shows the original image, the intrinsic images, and the reconstructed image.



(a) Original (b) Reflectance

(c) Shading (d) Reconstructed

Figure 12: The original image (a), reflectance (b), shading (c), and reconstructed image (d) that was created by multiplying reflectance and shading.

4.2 Recoloring

Recoloring. We simply select the pixel in the middle of the ball (since the color of the ball is uniform) and check the (normalised) R, G, and B values, which are 0.7216, 0.5529, and 0.4235 respectively.

To recolor the ball, the pixels that were colored are now replaced with a green color and a magenta color. The normalised R, G, and B values for green are 0, 1, and 0 respectively. The normalised R, G, and B values for magenta are: 1, 0, and 1 respectively.

After replacing all the non-zero (non-black) pixels with these values, the ball is reconstructed with green and magenta, as shown in Figure 13.

(a) Original (b) Green reflectance (c) Magenta reflectance

Figure 13: The original ball (a), the reconstructed ball with pure green reflectance (b), and the reconstructed ball with magenta reflectance (c).

The displayed color do not seem pure and uniform, due to the shading, which itself is not uniform. There is no shading in the middle of the ball (were the color seems the closest to the real color) and maximum shading at the edges (were the color seems the furthest from the real color, and closest to black). This is also visible when reconstructing the original image: the standard reflectance itself it very light (it would be labelled as beige), while the reconstructed image is pretty dark (it would be labelled as brown).

5 Color consistency

In this part color consistency was performed using the Gray-World Algorithm. This works by decreasing color channels that have a higher mean than the mean of RGB, and increasing color channels that have a lower mean than the mean of RGB. This way, all colors are scaled towards the mean, especially colors that are very strong.

Figure 14 shows an image that was originally reddish, but was recovered using the Gray-World Algorithm to appear more natural (white).

(a) Original (b) Gray-World

Figure 14: Provided image of building before (left) and after (right) applying Gray-World Algorithm. Notice the decrease in red.

The Gray-World Algorithm assumes that the average color in each channel is gray (128, 128, 128). Therefore, the algorithm may not work well when this assumption does not hold (when there is a skewed distribution of colors). In Figure 15 and Figure 16, two examples are shown were this is indeed the case. The first shows an image of a blue sky[4], which has clearly a lot of blue in it, and not so much red and green. Applying the Gray-World Algorithm results in increased values for red and green, which seems less natural than the original image. The second example shows a field of grass with some buildings and sky[5]. Clearly it has a lot of green and less red and blue. Applying the Gray-World Algorithm results in increased values for red and blue, creating a purple glace, which also seems less natural than the original image. Therefore, it is important to check whether the assumption holds, because the algorithm does not perform well with extreme differences in colors.

(a) Original (b) Gray-World

Figure 15: Image of blue sky with clouds before (left) and after (right) applying Gray-World Algorithm. Notice the increase in red and green and decrease in blue colors.

Other Color Consistency algorithms. One algorithm is White Patch (Retinex) algorithm were the intensities of all (non-maximum) pixels are increased. The maximum of each color channel is divided by the mean, which results in increasing intensities for channels that have a low(er) mean than maximum, making the colors more intense (brighter).

[4] `https://www.pexels.com/photo/nature-sky-clouds-blue-53594/`
[5] `https://www.pexels.com/photo/grass-meadow-estate-7174/`

(a) Original

(b) Gray-World

Figure 16: Image of grass field with building and sky before (left) and after (right) applying Gray-World Algorithm. Notice the increase in red and blue and decrease in green colors.

Another algorithm is the Scale By Max algorithm, which is very similar to the Gray-World Algorithm but uses maxima instead of means (Agarwal et al., 2006).

6 Conclusion

6.1 Photometric Stereo

Photometric Stereo can be used to reconstruct the surface of a 3D object, using 2D images under different illuminants by estimating the albedo and surface normals and then integrating the shape.

6.2 Color Spaces

Several Color Spaces were implemented with each different properties, which could be useful in specific tasks.

6.3 Intrinsic Image Decomposition

An image of a ball was reconstructed using reflectance and shading (these sources are sufficient), and different colors for reflectance were used to recolor the ball. Other useful properties are structure and texture.

6.4 Color Constancy

In these experiments, the true color of objects was reconstructed by correcting the images for the light source.

7 Bibliography

Agarwal, V., Abidi, B. R., Koschan, A., & Abidi, M. A. (2006). An Overview of Color Constancy Algorithms. Journal of Pattern Recognition Research 1 (2006), 42-54.

Aujol, J. F., Gilboa, G., Chan, T., & Osher, S. (2006). Structure-texture image decomposition—modeling, algorithms, and parameter selection. International journal of computer vision, 67(1), 111-136.

Evangelopoulos, G. & Maragos, P. (2008). Image Decomposition into Structure and Texture Subcomponents with Multifrequency Modulation Constraints. Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR '08), Anchorage, AK, 23-28 Jun. 2008.

Link to the github code: `https://github.com/MichelleAppel/CV1_ass1`