Computer Vision Lab 1

Maria Diea - 12593125 Diana Epureanu - 12710199
Teodora Stoenescu - 12598291 Razvan Tomescu - 12277584

September 2019

1 Introduction

This assignment consists of an analysis of the photometric stereo algorithm and various image manipulation techniques such as color spaces, intrinsic image decomposition and color constancy. The photometric stereo algorithm is applied on different inputs, varying the number of used images, objects or even the color scheme. Different color spaces for image representation are studied, together with separating an image into its formation components and color-correction.

2 Photometric Stereo

Estimating Albedo and Surface Normal

To estimate the albedo and the surface map, it suffices to follow the given algorithm and compute the values for each point in the image array. The albedo is expected to present no shading, but when looking at the result of the `SphereGray5` folder presented in figure 1, a smooth shadow can be seen around the edges of the sphere. In principle, three is the minimum number of images to estimate albedo and the surface normal as shown in [1]. Even so, using a larger number of images can lead to more accurate results: when running the algorithm on `SphereGray25` it can be observed that the shadow area is much smaller. Trying all the 25 images at once is advantageous as it showcases a more substantial difference in results compared to the `SphereGray5` albedo.

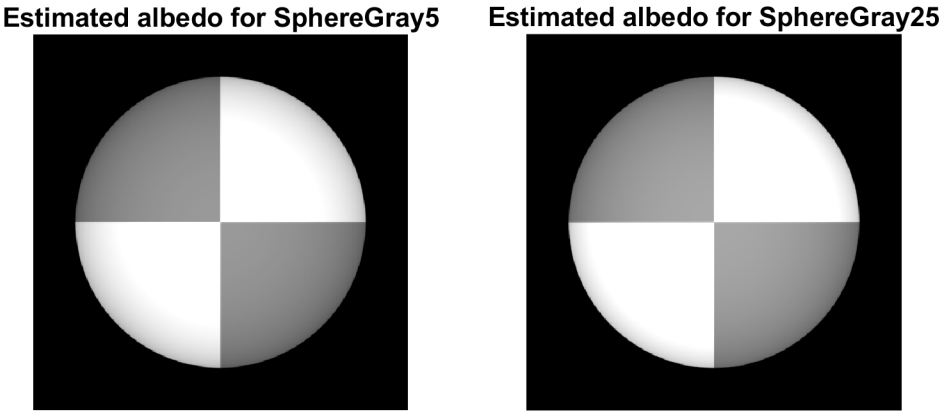


Figure 1: Two albedos of the same object showcasing the difference between using 5 images versus 25

Shadows have a meaningful impact in photometric stereo as any pixels that are shadowed in an image will be discarded and this can lead to a distorted final model. To deal with shadows, the book presents a trick where the contributions from the shadowed regions are zerod out. When removing the shadow trick and re-applying the algorithm to the grayscale sphere image, there is no difference between the new results and the ones in figure 1. This is due to the convex surface of the sphere as there are no deformities of the surface that would produce noisy shadows.

Test of Integrability

In order to compute $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$, we again followed the steps in Algorithm 5.1. For the computation of p, we divided the first dimension of the normal at the respective point by its third dimension, and for the computation of q we divided the second dimension of the normal at the respective point by its third dimension.
To compute the square error, we computed the value of $(\frac{\partial p}{\partial y} - \frac{\partial q}{\partial x})^2$ and we set the threshold to 0.1. Noise, or poorly light up pixels in the images, can lead to a larger error. Thus, the more the number of light sources increases, the less probability for noise. Therefore, the model naturally performs better when given 25 input images (1309 outliers) instead of 5 input images (1536 outliers).

Shape by Integration

To do integration in row-major order, you start at the top-left corner and integrate along the first row, then go down along each column. When running the algorithm on `SphereGray5` using column-major order, the second quadrant of the sphere appears to be skewed, going deeper towards the z-axis. When using row-major order, the same phenomena happens with the last quadrant. Constructing the height map based on the average of the two orders leads to an improved more uniform result in which the surface of the sphere does not present such drastic distortions even though they are still noticeable.

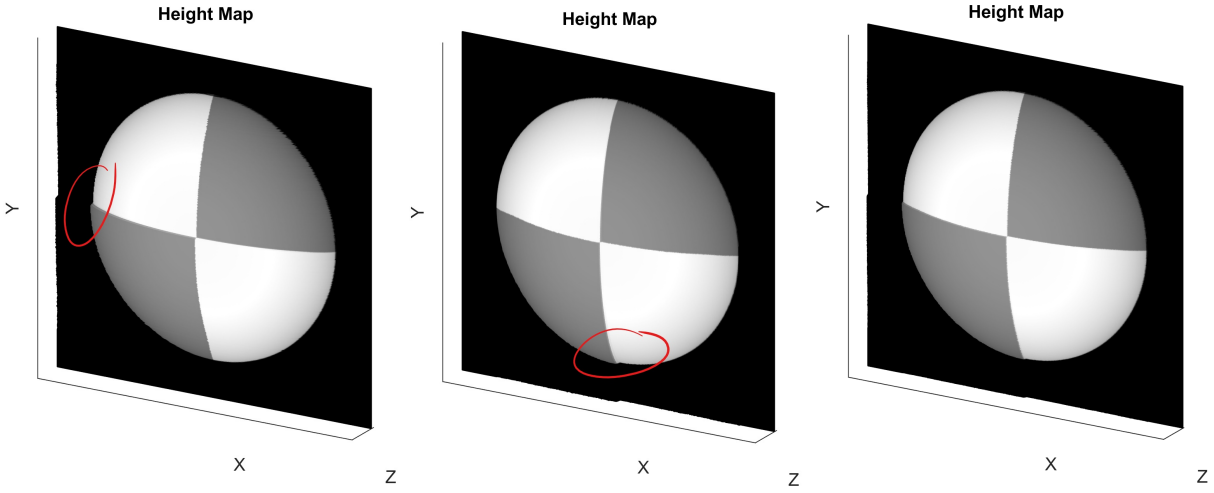


Figure 2: Column-major vs. row-major vs. average on `SphereGray5`

Due to the large number of images, there is no visible difference when using column or row-order for the `SphereGray25` set. The surface of the sphere appears smooth and convex.

Experiments with different objects

Using all 121 images, the albedo is very close to reality (see Figure 3), with 2457 outliers when the threshold is set to 0.1. If the light is chosen poorly then some white regions might appear always in shadow and thus be interpreted as black regions. For example, if only images with light sources coming from negative y positions are used, false dark regions will appear (see Figure 4), giving 5175 outliers. Another example is taking light positions with positive x coordinates, which causes a lot of shadow to occur on the left side of the images (see Figure 5) and 7212 outliers.

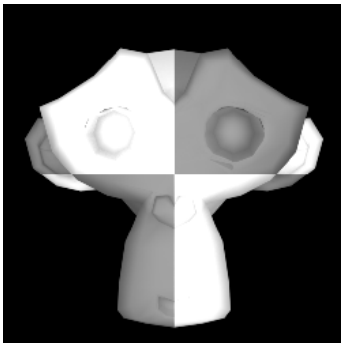


Figure 3: Albedo using all images, lights from all directions

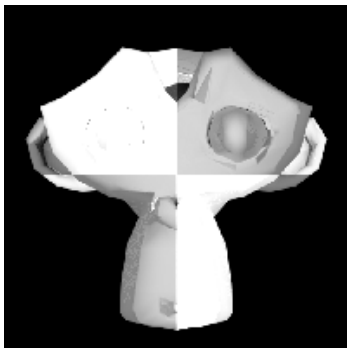


Figure 4: Albedo with y position of the light always negative

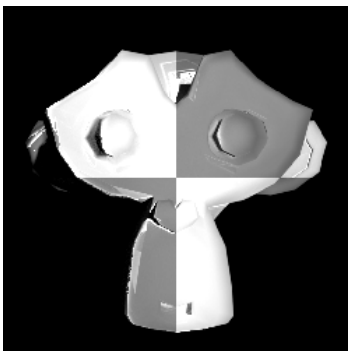


Figure 5: Albedo with x position of the light always positive

In order to for the implementation to work for 3-channel RGB inputs, instead of working only with one channel of the image_stack, all 3 channels were read and saved using the `load_syn_images` function. To compute the normal map, we used the 3 channels to compute the RGB image and used its gray scale for accurate results using the `estimate_alb_nrm` function. For the albedo, we constructed one separate albedo for each of the 3 channels and averaged the results.

Results

In case of the SphereColor which presents colors from all three channels, the results came immediately as seen in figures 6 and 7.

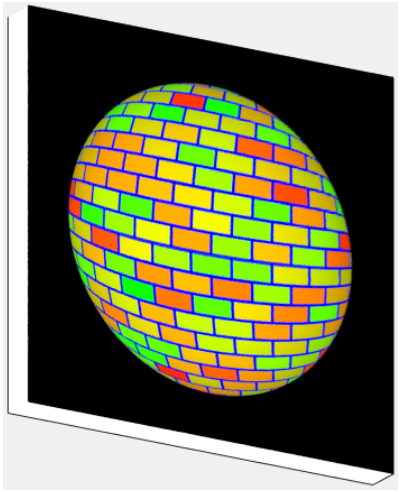


Figure 6: Height map

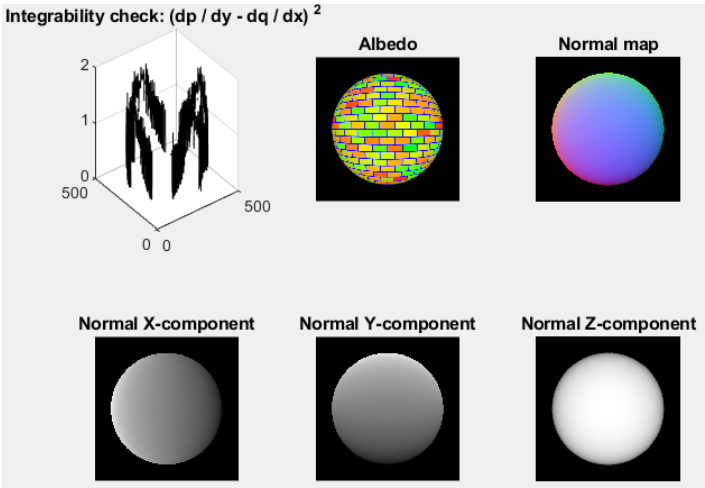


Figure 7: Summary

For the MonkeyColor dataset, the figures have all 0 pixels on the green channel. This presents a problem in the `load_syn_images` function. When conducting the normalization, the values in the image_stack are divided by the difference between the largest and lowest values of a pixel. This difference will obviously be 0 when all pixel values are 0. To avoid division by 0 and thus to fix the problem, we implemented a check that if all pixels of an image stack are 0, then the division is skipped and image stack is no longer modified. Now the algorithm will properly run on the MonkeyColor dataset as well, as seen in figures 8 and 9

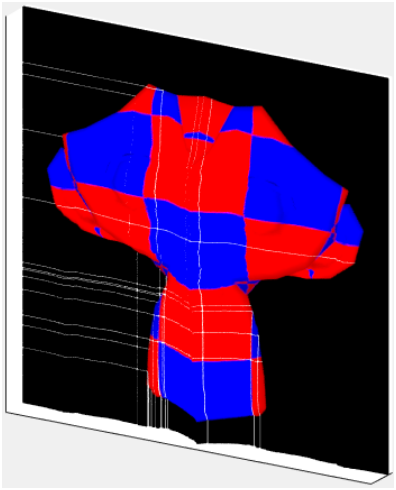


Figure 8: Height map

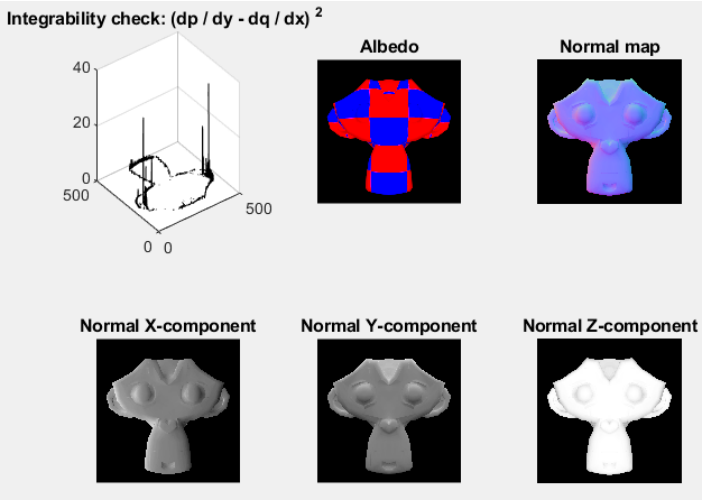


Figure 9: Summary

When running the algorithm on the Yale Face images, different integration paths result in fairly different results. In figure 10 it can be observed that column-major integration results in a flatter result, while the row-major integration distorts the face by exaggerating the depth of the nose. Averaging the results proves once more to be a more reliable strategy, as the final result closely resembles the actual shape of a face.

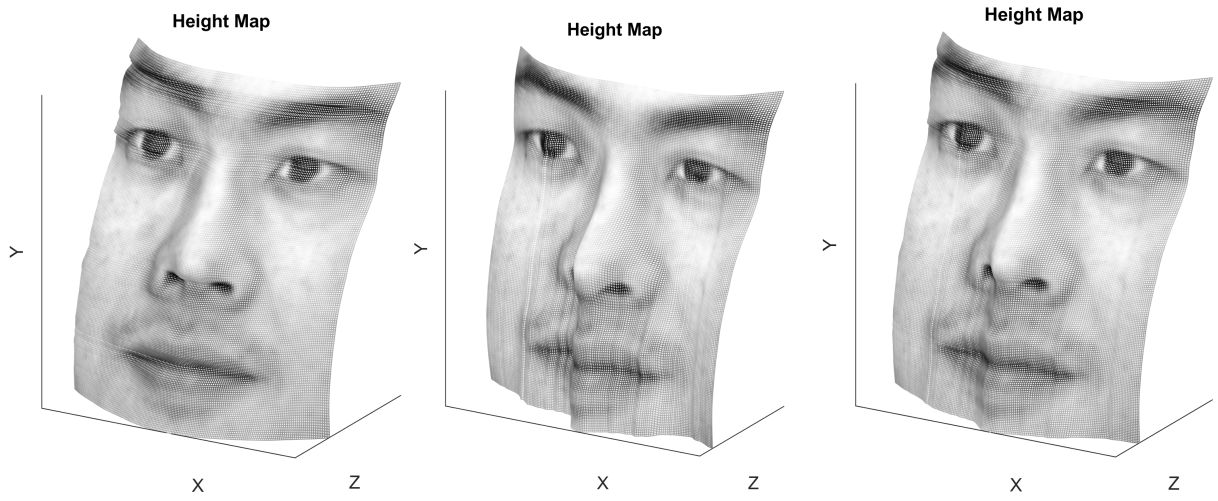


Figure 10: Column-major vs. row-major vs. average on a Yale Face image

The Yale Face images violate the Lambertian assumption of the shape-from-shading model as the surfaces are not entirely matte but rather have shiny spots, just as the ones in figure 11. Removing the images which violate the assumptions should lead to a more accurate normal vector and hence to a better height map.

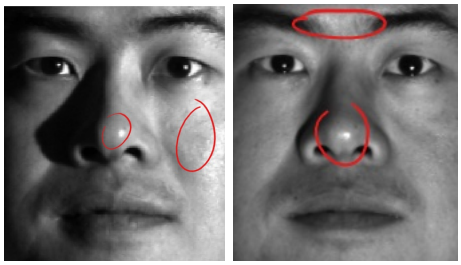


Figure 11: Shiny Yale Face images

3 Color spaces

RGB Color model(3)

We use the RGB model as a basis of our digital cameras and photography because it is an additive color system. It uses the projected light to define colors. It is therefore based on the primary colors of light, colors to which the human eye is very sensitive too, due to its three types of cones.
A standard digital camera captures the full RGB color image through the color filter array. Through interpolation, the color computed by the camera for a specific pixel is given by a combination of the color captured through its own filter and the two other colors captured by the pixels around it.

Color Space Conversion(10)

Figure 12 depicts the original image in RGB color space along with the 5 different color spaces it was converted to. The code implementation of each color space follows the instructions given in the assignment. The input image is separated into 3 separate channels, each channel is differently processed then they are concatenated on the third dimension in order to obtain the output image. In the grayscale case, the instructions from the linked paper were followed as well.

Color Space Properties(5)

- **HSV** stands for hue, saturation and value and it is a cylindrical color model. With hue, 0 degrees gives red, 120 degree gives green and 240 degrees gives blue. Saturation is the amount of color, where 100% is pure color and 0% is grayscale. Value is the brightness; 0% value means pitch black and 100% value means there is not black in the color. The RGB primary colors are mapped into dimensions that are easier for human eye to understand. It is used in feature detection and image segmentation.
- **YCbCr.** Y is the luma component, meaning the brightness of the color. Our eyes are more sensitive to it. Cb and Cr are the blue-difference and red-difference chroma components. Cb and Cr are less sensitive to our eyes. Therefore they do not need to be that accurate as long as Y is accurate. This is why YCbCr is used in image compression are extra details of the image which are not recognized by our eyes can be omitted.
- **Opponent color space** Our cone photoreceptors are linked together and form three opposing colour pairs which are blue and yellow, red and green, and black and white. When one part of the pair is activated, the other cannot be perceived anymore hence we cannot see colors like bluish yellow or reddish green. oRGB is used in color adjustment, color transformation. For example, the way colors are mapped on the axes is such that the warmer colors are on one side and the cooler colors are on the opposite side hence enabling us to compute shades of warm to cool in pictures.
- **Grayscale** images are composed of shades of gray, where each pixel carries the intensity information. Therefore the image does not have several color channels anymore, only one channel where each pixel value varies from 0 to 255. Although it only has one channel, the message of the image can stiil be conveyed, thus if we have storage limitations, we could also work with grayscale images instead of colored images.
- **Normalized RGB** simply means to divide each pixel's value by the sum of the pixel's value over all channels. What happens then is that the noise of the image caused by shadows or lights disappears. This is useful for example in object recognition since the shape of the objects will be clearly defined.

More on Color Spaces(2)

Another color space is CMYK. It stores the ink values cyan, magenta, yellow and black and it is a subtractive color model. CMYK is used in the printing process as it starts with a white board and it subtracts colors from that white in order to create images. Cyan is white light minus red, magenta is white light minus green and yellow is white light minus blue.

4 Intrinsic Image Decomposition

Other Intrinsic Components(4)

We can decompose the image into rectangles of uniform properties.[2]

Synthetic Images(2)

The reason why almost all intrinsic image decomposition datasets are composed of synthetic images is that the image must satisfy a few condition(which are hard reproduce usually with not-synthetic images such a photo a natural scene or something similar) such that intrinsic image decomposition works properly. For example, in the case we decompose using the albedo and the shading, we must not have indirect lighting on the picture and reflections between objects or surfaces, because this would make the intrinsic image decomposition not accurate.

Image Formation(4)

The result of the exercise can be seen in Figure 13. In order to implement it, we have looped through the channels of the albedo image and performed the pixel wise multiplication with the values from the shading image, storing the result in another variable.

Recoloring(5)

To find out the color of the ball we simply get a sample from the albedo. Since we saw the image, we observed it is centred, hence we can just sample from the center of the picture to get the color of the ball.
To recolor the ball with pure green we simply go through all point in the picture(pixels) and if they are not black, we change the color to pure green.
The color distributions over the object do not appear uniform, because of the shading of the picture and due to the fact that the ball is not a flat surface. Hence, light gives the impression that the color distributions over the object do not appear uniform.

5 Color Constancy

Grey-World(15)

The original image and the color corrected one can be seen in Figure 14. The algorithm produces an estimate of illumination by computing the mean of each channel of the image. In order to normalize channel **i**, each pixel from the channel is scaled by $\frac{mean}{mean_i}$, where mean is the illumination estimate, which will be 128, grey and $mean_i$ is the mean of the respective channel.

Give an example case for Grey-World Algorithm on where it might fail. Remember to include your reasoning.
The Grey-World Algorithm might fail when its assumption that under a white light source, the average color in a scene should be achromatic(i.e (grey, [128, 128, 128]). That can happen for example when the whole image has the same colour(say red for example), hence under a white light source the average color in the scene should be red in this case, the assumption of the Grey-World algorithm doesn't hold, so the the algorithm might fail.
Find out one more color constancy algorithms from the literature and explain it briefly.
Another color constancy algorithm is called **SCALE-BY-MAX**. It is very similar to the Grey-World algorithm, the only difference in the actual algorithm is that in the **SCALE-BY-MAX** algorithm we divide by the maximum value of red, green and blue instead of the average value of red, green and blue as in Grey-World algorithm. The assumption of the **SCALE-BY-MAX** algorithm is that the reflectance(the albedo) is uniform and there is only one white light source. Of course, this algorithm might fail when the assumption doesn't hold, for example when there are multiple surfaces with different surfaces or multiple light sources.

6 Conclusion

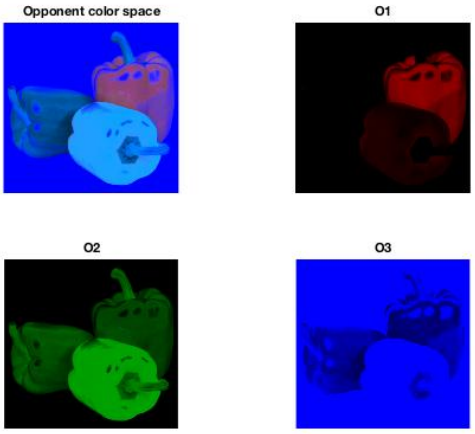
We have analysed interesting properties of images in each discussed section, compared some popular algorithms and investigated the most important applications that result from the image formation techniques. The code implementation from which we derived the results can be found in the submission attachment.

References

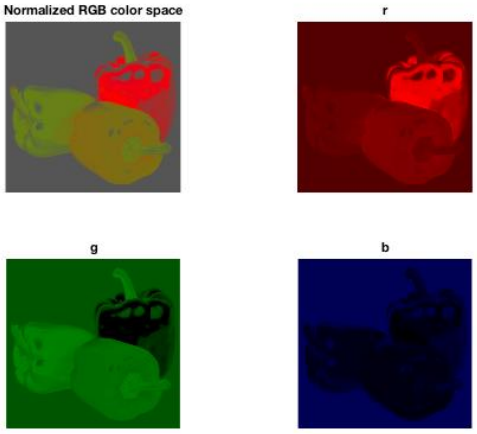
[1] R. J. Woodham, Photometric Method for Determining Surface Orientation from Multiple Images, Optical Engineering 19 (1) (1980) 139–144.

[2] P. Chitale, T. Huntsberger, A decomposition technique for image compression `https://ieeexplore.ieee.org/document/138573?fbclid=IwAR0rVu5i8pWG0rfdTvZyTPJGEA3CDJ2_DJABn_7SK41H0gr4vRYoc9oi16U`

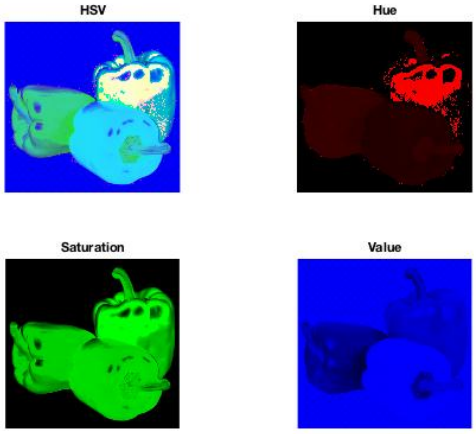
(a) Original image



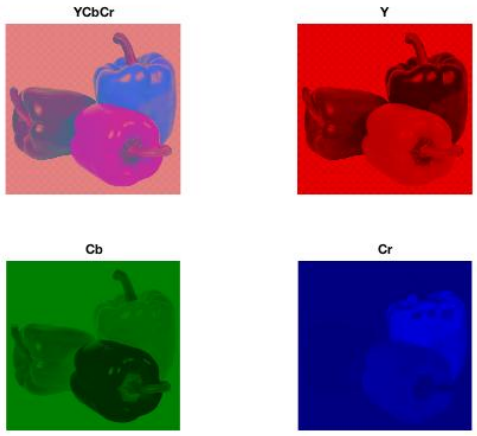
(b) Opponent color space



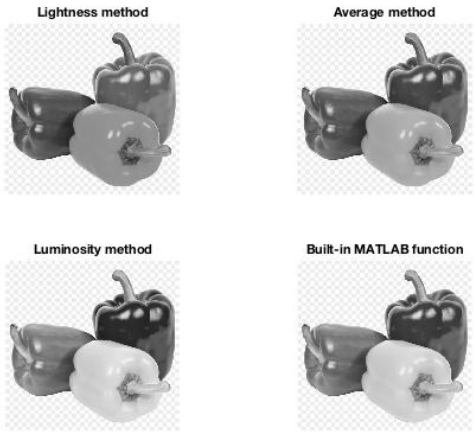
(c) Normalized RGB



(d) HSV



(e) YCbCr



(f) Grayscale

Figure 12: Different color spaces

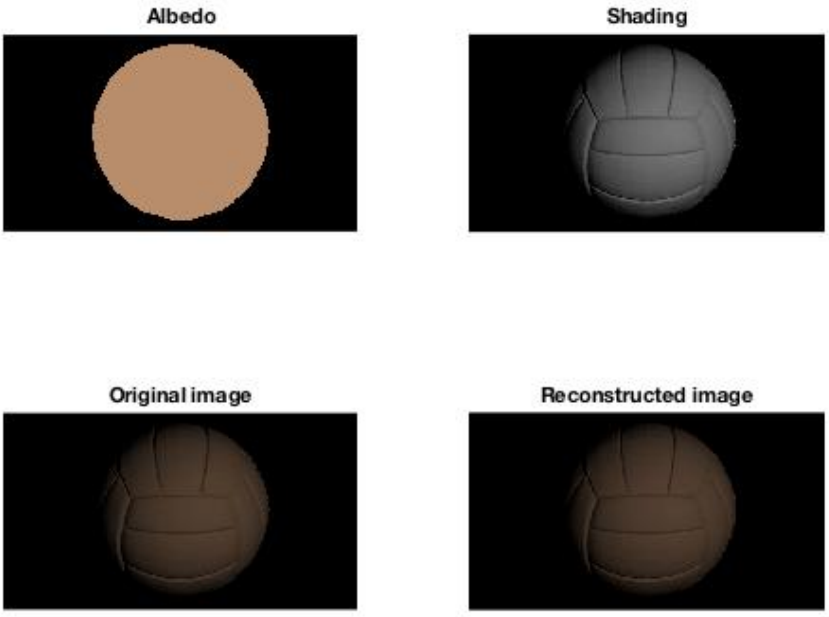


Figure 13: Intrinsic image decomposition
We can decompose the image into rectangles of uniform properties.

Figure 14: Color constancy