Computer Vision 1: Photometric Stereo & Color Spaces

Bogdan Teleaga, 11413565
Mircea Mironenco, 11164182

February 20, 2017

1 Photometric Stereo

Recovering the surface of an object under different lighting conditions can be done using Photometric Stereo. In this experiment, the light source is simply chosen artificially, after which the albedo and normals are computed. Finally, the second order partial derivatives need to not differ significantly from each other for the experiment to be successful.

Initially we construct the light source matrix as unit vectors, with values chosen artificially:

$$V = \begin{bmatrix} 0 & 0 & -1 \\ 1 & -1 & -1 \\ -1 & -1 & -1 \\ 1 & 1 & -1 \\ -1 & 1 & -1 \end{bmatrix}$$

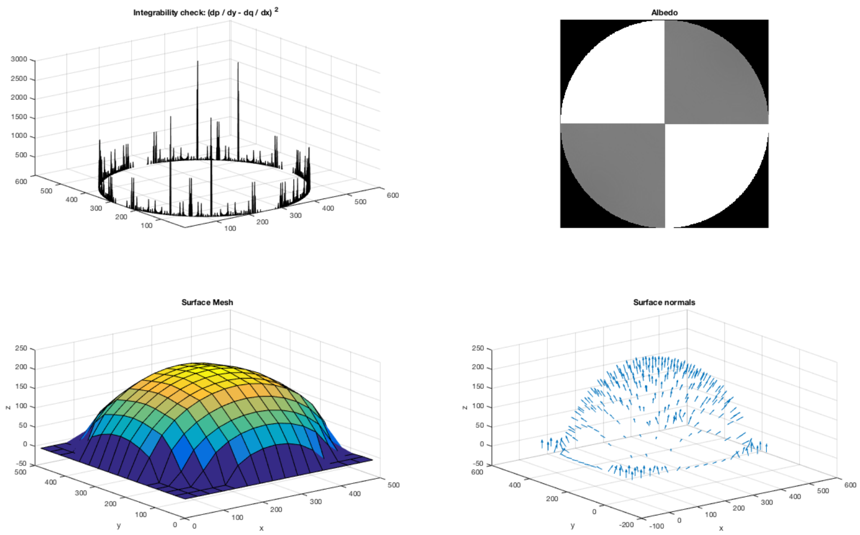
The light source is scaled and to obtain the albedo and surface normals we initially start by solving the system of equations $Ii = IV\vec{g}$ where i is the vector of intensities for a given pixel coordinate position in each image, and I is a matrix with i as it's diagonal, which has the role of accounting for shaded regions in the image.

Albedo is computed as the norm of $\vec{g}$. To obtain the surface normals N_1, N_2, N_3 we simply normalize $\vec{g}$.

To recover the surface one can simply do hill climbing after obtaining the surface derivatives as $p = \dfrac{\partial f}{\partial x} = \dfrac{N_3}{N_2}$ and $q = \dfrac{\partial f}{\partial y} = \dfrac{N_2}{N_3}$. The albedo, recovered surface and surface normals can be seen in Figure 1.

We perform a test of integrability to determine if there are outliers in the plotted surface. This can be done using either a neighborhood approximation or using MATLAB's *gradient* function to obtain the second order derivatives $\dfrac{\partial p}{\partial y}, \dfrac{\partial q}{\partial x}$. Results are also shown in Figure 1.

Figure 1: Photometric experiment results.



2 Color Spaces

The second part of the experiment involves computing several color spaces for a given image. The computed color spaces are:

- Opponent
- Normalized
- HSV
- YcBcR

Additionally the gray color space is computed using the averaging, lightness, luminosity methods, as well as MATLAB's built in *rgb2gray* function. The following figures show the visualization of each individual channel for the implemented color spaces, as well as the complete image.

Figure 2: RGB Normalized

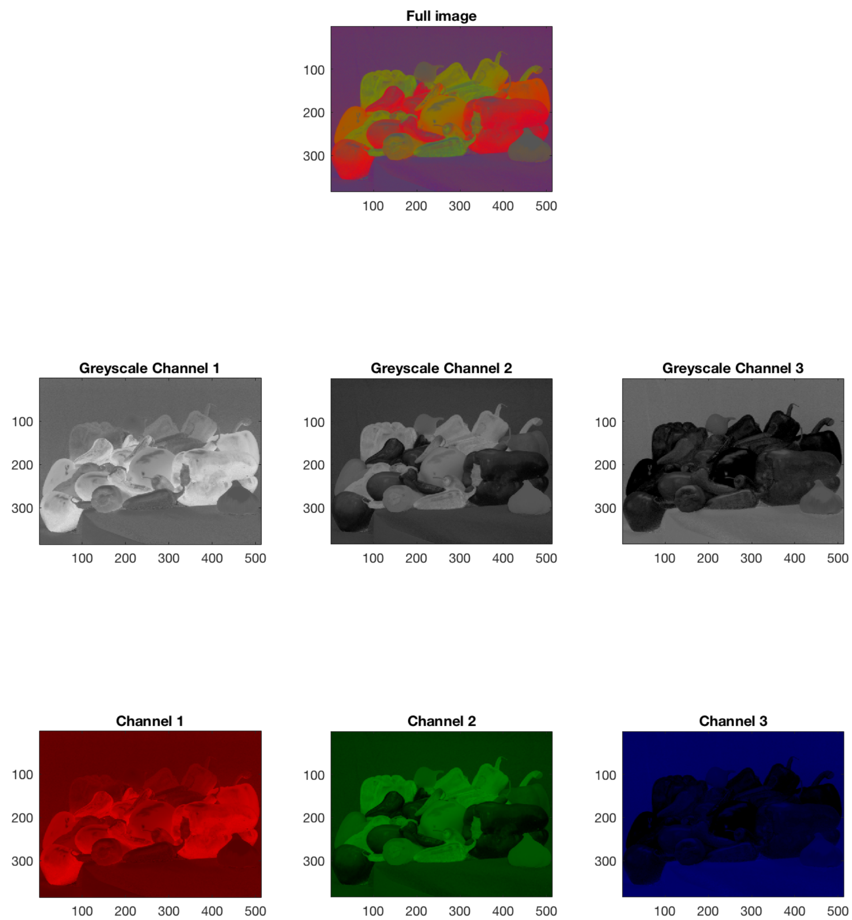


Figure 3: Opponent

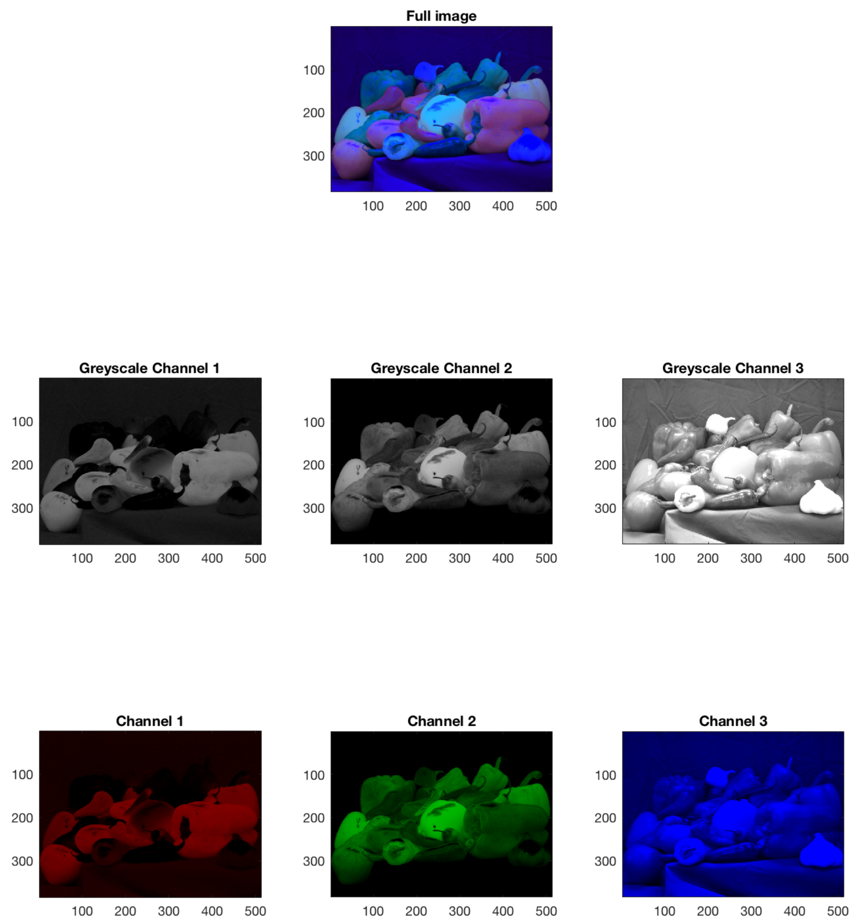


Figure 4: Grayscale

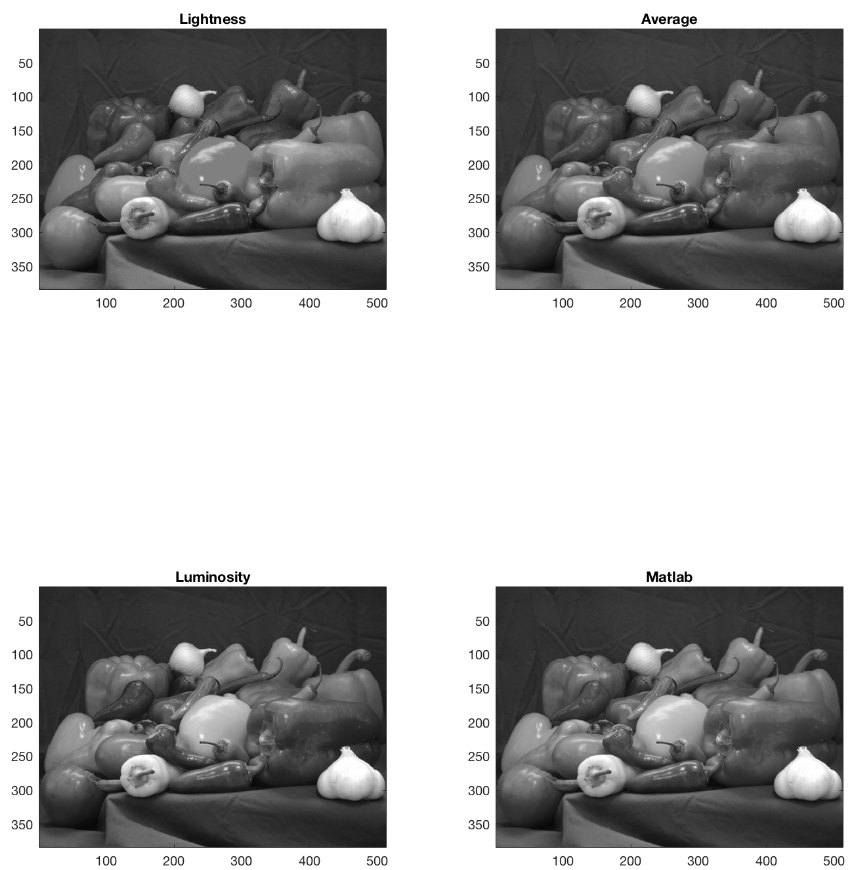


Figure 5: YcBcR