Computer Vision 1 - Assignment 1
Photometric Stereo & Color Spaces

Selene Baez Santamaria (10985417) - Andrea Jemmett (11162929)

February 17, 2016

1 Photometric Stereo

The goal of this assignment is to use Matlab to implement the photometric stereo algorithm, which aims to recover a patch of surface from multiple pictures under different light sources.

For this implementation we assume five light sources are involved, all of them distant. They are positioned facing the front, left-above, right-above, left-below and right-below corners of the surface showed in the images.

We created a function that is called without arguments and produces three figures, one for the surface albedo, one for the surface normals and one for the reconstructed shape.

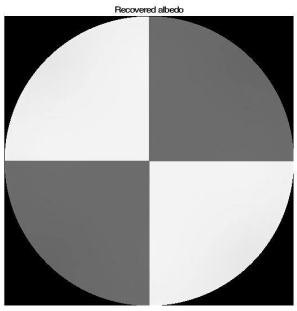

Figure 1: Reconstructed albedo

Here follows a descriptions of the steps executed inside the function:

1. Read the given images for a sphere under different light sources and store them together in a three dimensional matrix. For this we use

the Matlab *imread* and *im2double* functions to get vectors of desired precisions.

2. Represent the light sources with vectors assuming a coordinate system with origin in the center of the image

3. Determine matrix V from light sources. Here we introduce the scalar k which controls the camera response to the input radiance

4. Create structures to store albedo, normal, p and q per pixel

5. Repeat for each pixel:

   (a) Retrieve the pixel values for all images and store them as i

   (b) Construct diagonal matrix I with i values along the main diagonal

   (c) Solve linear system of equations for g using Matlab's *pinv* function as follows:

$$A = I * V$$
$$b = I * i$$
$$g = pinv(A) * b$$

   (d) Calculate albedo, normal, p and q as given by the formulas in page 85 of the book. To avoid NaN values we check for albedo = 0 and set the normal, p and q values to 0.

   (e) Ensure smoothness of heigh map by checking p and q. For this purpose we calculate the second derivative and filter out big noisy values.

6. Show recovered albedo (See Fig. 1);

7. Show surface normals using Matlab's *quiver*3 function (See Fig. 2);

8. apply algorithm to reconstruct the surface height map and show it (See Fig 3).

Regarding the scalar k used at step 3, we found after experimentation that a value of 100 gives the best results. For the derivative check at step 5e we used a threshold of 30 and for those pixels that do not pass the check we set p and q to zero.

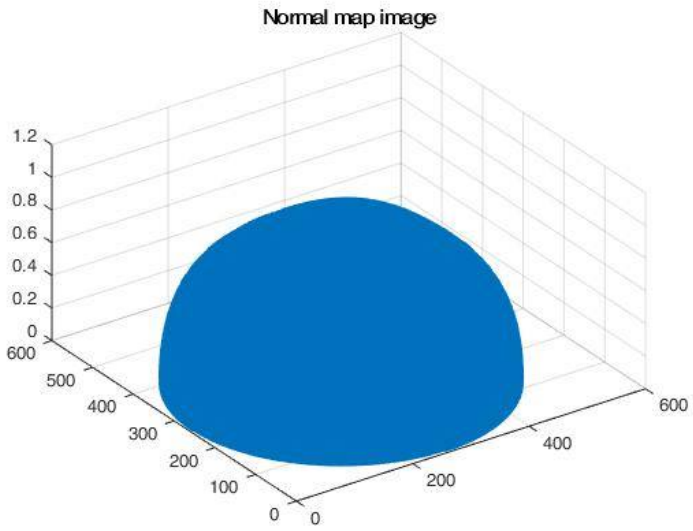

Figure 2: Surface normals

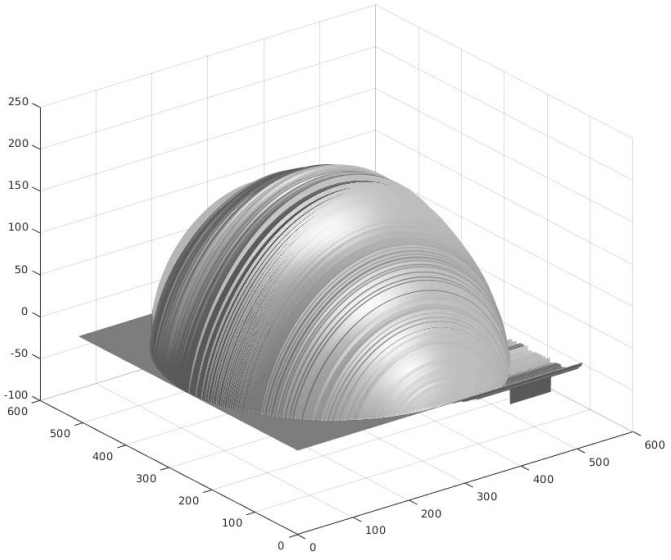

Figure 3: Reconstructed height map

2 Color Spaces

The goal of this assignment is to use Matlab to convert images between RGB and other color spaces.

We created a function that is called with two arguments: the name of the image to be converted, and the color space to be converted to. Again we used the function *imread* to read the image. To compute the converted image we had to cast the 8 bits integer image (with values in the range 0 - 255) to doubles. The converted image is then displayed as three gray-scale images (one for each channel). We experimented with the given image "bricks.jpg" as well as other colorful images.

2.1 Opponent Color Space

For the RGB to *Opponent* color space conversion, we used the following formula:

$$\begin{pmatrix} O_1 \\ O_2 \\ O_3 \end{pmatrix} = \begin{pmatrix} \frac{R-G}{\sqrt{2}} \\ \frac{R+G-2B}{\sqrt{6}} \\ \frac{R+G+B}{\sqrt{3}} \end{pmatrix}$$

yielding the following image:

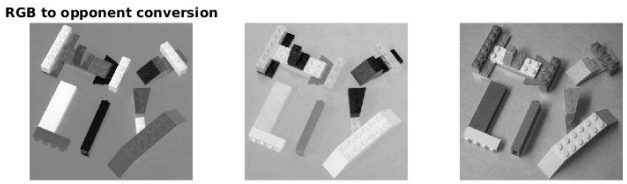

Figure 4: RGB to Opponent Color Space

2.2 rgb Color Space

For the RGB to *rgb* color space conversion, we used the following formula:

$$\begin{pmatrix} r \\ g \\ b \end{pmatrix} = \begin{pmatrix} \frac{R}{R+G+B} \\ \frac{G}{R+G+B} \\ \frac{B}{R+G+B} \end{pmatrix}$$

yielding the following image:

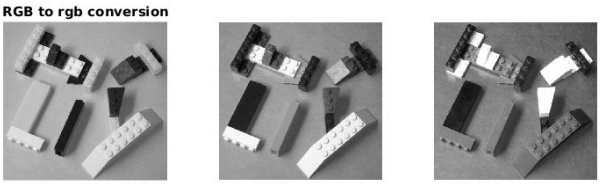

Figure 5: RGB to rgb Color Space

2.3 HSV Color Space

For the RGB to *HSV* color space conversion, we used the Matlab function *rgb2hsv*, which yield the following image:

Figure 6: RGB to HSV Color Space