Homework – 4

Name: Sridhar Mocherla

PART A:

Photometric stereo:

With the given set of 7 images under 7 different light sources(V), we can try to compute the surface normals using the equation in Figure 1.

$$i = Vg$$

$$g = V^{\Psi} i$$

$$N(x,y) = g(x,y)/|g|$$

Figure 1: Equation to compute surface normal

In MATLAB, we iterate through each image, pixel by pixel and for each one of them and compute the intensities. Then, using linear least squares method, we compute *g* at each of the pixels and correspondingly the 3x1 normal vector at that point. Below shows the MATLAB code snippet for the same.

```
%L - light direction vectors
%images - set of images
function [I,N] = compute_normals(images,L)
   numImages = size(images);
   I = zeros(400,400,7);
   for i=1:numImages
       img = rgb2gray(images{i});
       for j=1:400
           for k=1:400
               I(j,k,i) =img(j,k);
           end
       end
   end

    N = zeros(400,400,3);
   for h = 1:400
    for w = 1:400

            % Intensities
            i = reshape(I(h, w, :), [7, 1]);
            %Use linear least squares to obtain 3x1 normal vector
            n = (L.'*L)\(L.'*i);
                if n~=0
                     n=n/norm(n);
                 else
                     n = [0;0;0];
                 end
            N(h, w, :) = n;
```

```
    end
   end
end
```

We then compute the image with uniform albedo by finding the pixels for separate channels (R, G and B) using the equation in Figure 2.

$$J = V.N(x, y)$$
$$imgChannel(i, j) = I.J/(J.J)$$

Figure 2: Finding image with uniform albedo

We combine the pixel values obtained for each channel into a single image. The resultant image with uniform albedo is as shown in Figure 3.

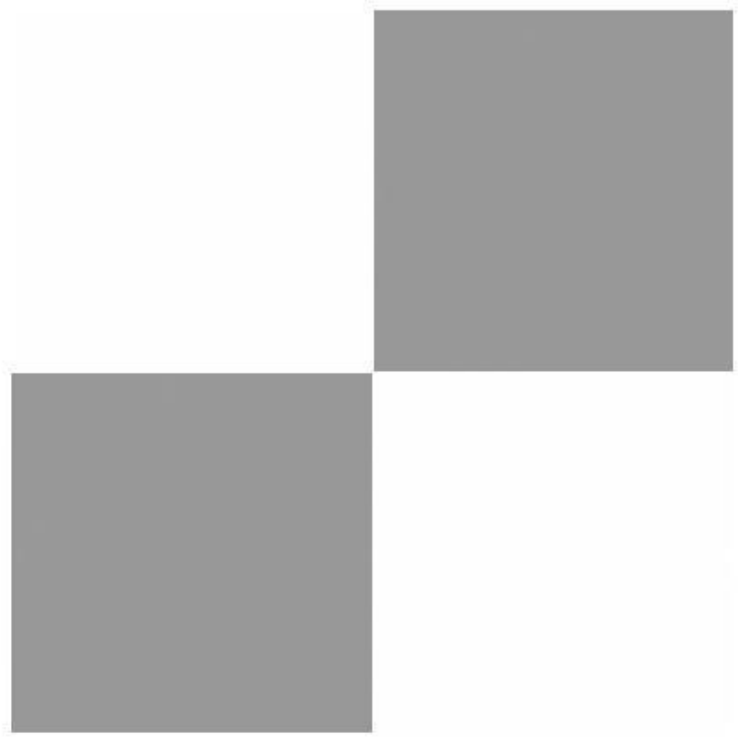

Figure 3: Image with uniform albedo recovered from input data – represented by |g|

Using the normals obtained in previous, we compute the partial derivatives at each pixel point and then integrate the partial derivatives to obtain z = f(x,y) by adding up steps of these derivative values. The MATLAB code snippet below illustrates this.

```
Fx = size(400,400);
Fy = size(400,400);
for i=1:400
```

```
    for j=1:400
        normal = reshape(normals(i,j,:),1,3);
        deriv = normal/normal(3);
        Fx(i,j) = deriv(1);
        Fy(i,j) = deriv(2);
    end
end

r=zeros(400,400);
for i=2:400
    r(i,1)=r(i-1,1)+Fy(i,1);
end

for i=2:400
    for j=2:400
        r(i,j)=r(i,j-1)+Fx(i,j);
    end
end
```

We use the MATLAB function surf() to display the depth map and it is shown in Figure 4.

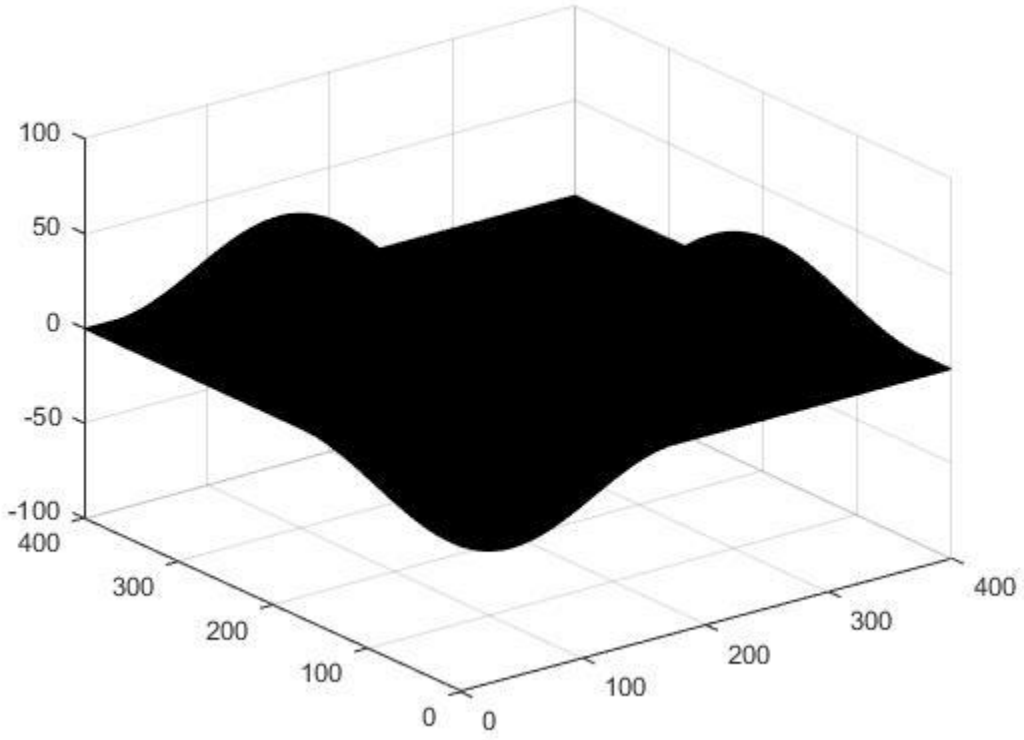

Figure 4: 3-D surface depth map obtained by integrating Fx and Fy

PART B:

Recovering surfaces for color photometric stereo:

With the given input data, we again use linear least squares method to obtain the surface normals, thereby the partial derivatives, which we integrate to obtain the surface depth map (as in partb.m).

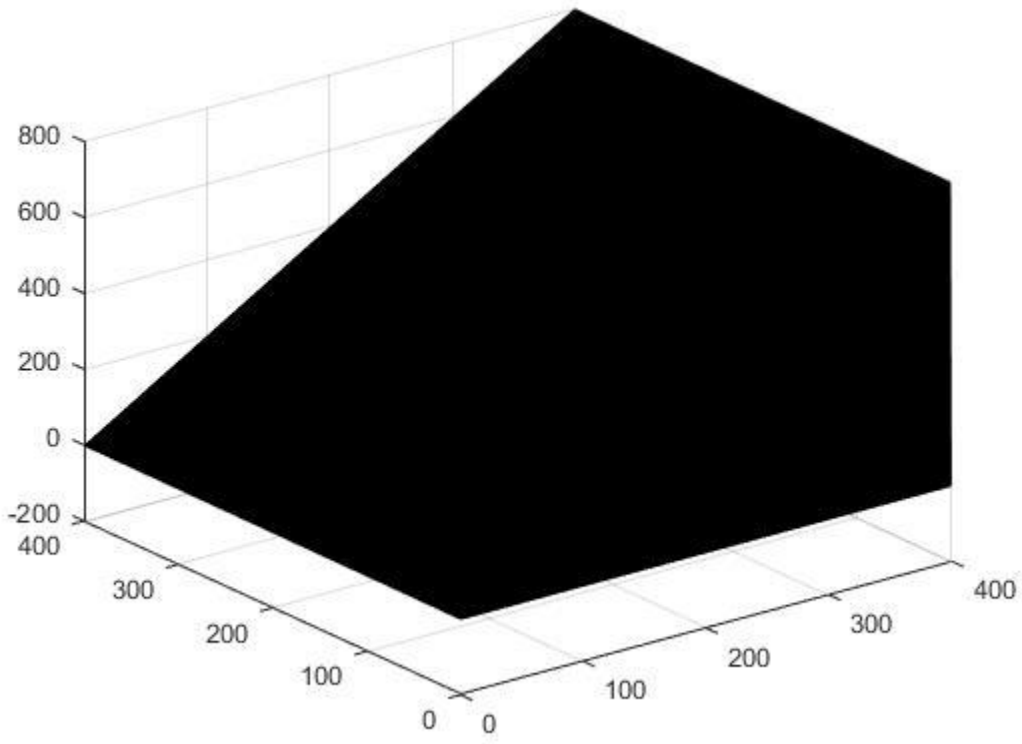

Figure 5: Surface depth map of the image color_photometric_stereo_1.tiff

PART C:

1. The estimate of the illuminant color is (R,G,B) = (110,141,250) after scaling it to 250. MATLAB code snippet to compute this is given below.

```
r=[];g=[];b=[];
for i=x(1):1:x(2)
    for j=y(1):1:y(2)
        i = uint8(i);
        j = uint8(j);
        r = [r,I2(i,j,1)];
        g = [g,I2(i,j,2)];
        b = [b,I2(i,j,3)];
    end
end
meanR2 = mean(r);
```

```
meanG2 = mean(g);
meanB2 = mean(b);
scaleFactor = 4.6669;
meanR2 = uint8(scaleFactor*meanR2);
meanG2 = uint8(scaleFactor*meanG2);
meanB2 = uint8(scaleFactor*meanB2);
```

2. The estimate of the illuminant color for solux-4100 light is (59,100,250). This is obtained similarly using the above MATLAB code.

3. The angular error is 11.2831 degrees.

4. Mapping image to canonical light:

Figure 6: Bluish image in solix color

Figure 7: Image corrected using diagonal model

Figure 8: Image formed by canonical light

5. The RMS (r,g) errors are
    a. Canonical vs standard solux image = 0.0787
    b. Canonical vs Corrected image = 0.0612

6. Max RGB algorithm:
   Illumination colors (as obtained in partc.m):
    a. Apples2 – (182,228,250)
    b. Ball – (112,143,250)
    c. Blocks – (250,242,209)

   Angular errors:
    a. Apples2 – 24.9263 degrees
    b. Ball – 11.7077 degrees
    c. Blocks – 35.3980 degress

7. Image correction using Max RGB algorithm:

Figure 9: Corrected version of apples2.tif using solux light and Max RGB algorithm

Figure 10: apples2.tif obtained using solux light and without correction

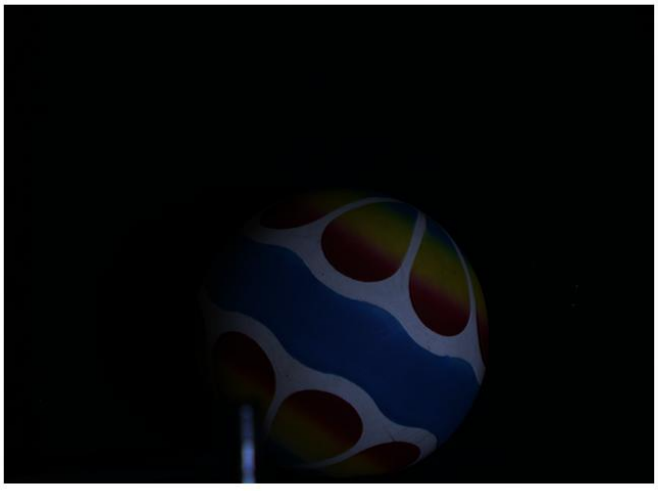

Figure 11: ball.tif obtained using solux light and without correction

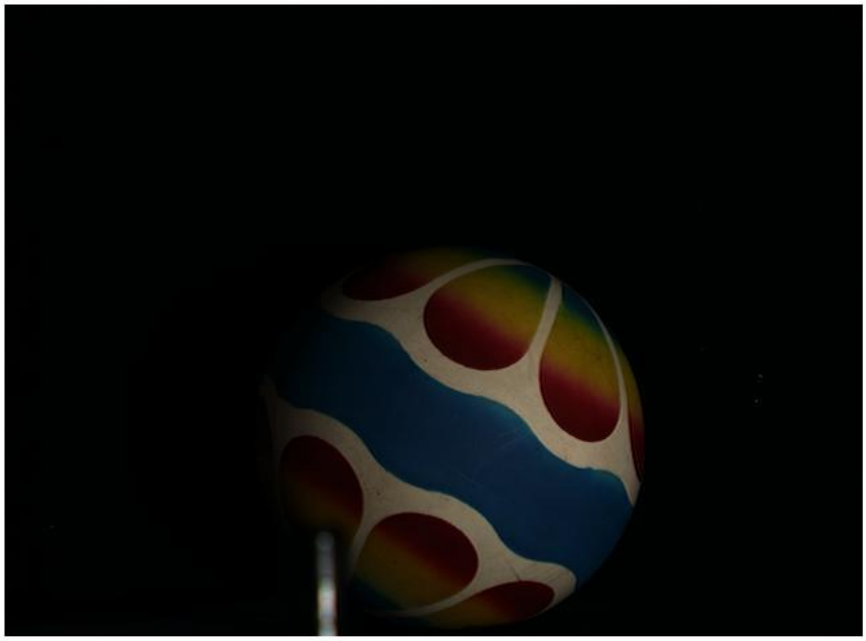

Figure 12: ball.tif using solux light and Max RGB correction

Figure 13: blocks.tif obtained using solux light and without correction

Figure 14: blocks.tif obtained using solux light and Max RGB correction

Below MATLAB code snippet shows application of Max RGB correction

```
function [output] = display_max_rgb_image(I)
m = size(I,1);
n = size(I,2);
temp_R = double(I(:,:,1));
temp_G = double(I(:,:,2));
temp_B = double(I(:,:,3));
output = zeros(m,n,3);
    Rmax     = max(temp_R(:));
    Gmax      = max(temp_G(:));
Bmax      = max(temp_B(:));
            Max         = max([Rmax Gmax Bmax]);
            Kr          = Max/Rmax;
            Kg          = Max/Gmax;
            Kb          = Max/Bmax;
            output(:,:,1) = Kr*double(I(:,:,1));
            output(:,:,2) = Kg*double(I(:,:,2));
            output(:,:,3) = Kb*double(I(:,:,3));
            output = uint8(output);
            figure('Name','Displaying image using MaxRGB algorithm');
            imshow(output);
```

The (r,g) RMS errors for the pair of images using MaxRGB are:

1. Apples2 – 0.0279

2. Ball – 0.0937
3. Blocks – 0.0167

8. Image correction using Grey world algorithm:

First, we determine the light colors for each image as in partc.m.

a. Apples2 – (49,78,250)
b. Ball – (129,166,250)
c. Blocks - (250,242,209)

Angular errors:

a. Apples2 – 4.65 degrees
b. Ball – 15.73
c. Blocks – 37.68

We now show the corrected version of the images using the Greyworld algorithm below.

Figure 15: apples2.tif using solux light and greyworld correction

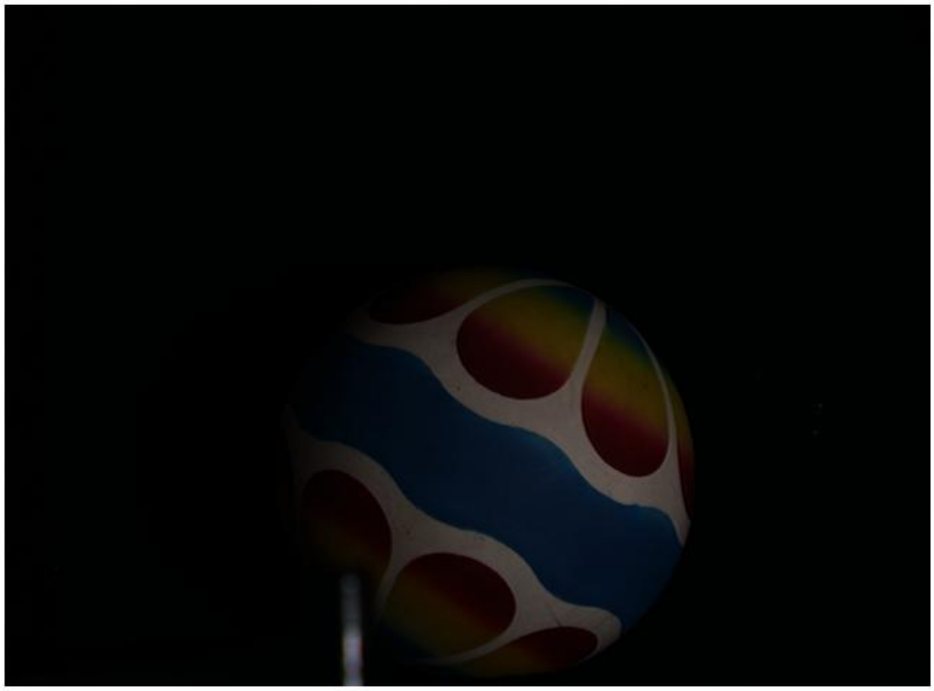

Figure 16: ball.tif with greyworld correction using solux light

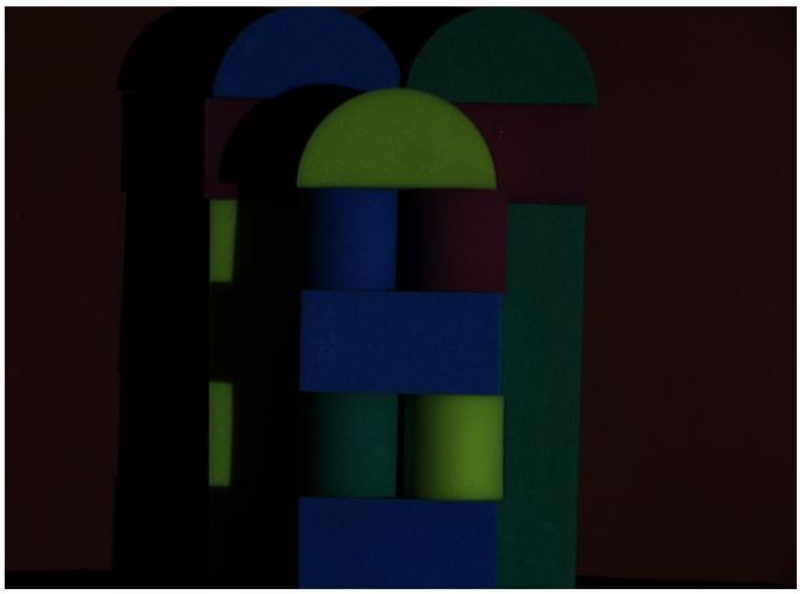

Figure 17: blocks.tif with solux light and greyworld correction

RMS errors using greyworld correction:

1. Apples2 – 0.1598
2. Ball – 0.0645
3. Blocks – 0.0739

Overall, Max RGB seems to have lesser RMS error compared to Grey World correction.