Homework 2

Meta information	
Name	Savan Kiran
Program	Masters in Computer Science
Questions skipped	PART D
Questions substituted	N/A
Extra credit questions	PART C

PART A

We start by normalizing the intensities at each pixel and further calculating the surface normal at each pixel. All the calculations are done independent of the albedo. Assuming there are 3 lights, {L1, L2, L3}, the intensity at a point on the surface because of the incident lights is given by,

$$I_1 = k_d N.L_1$$
$$I_2 = k_d N.L_2$$
$$I_3 = k_d N.L_3$$

We can re-write the equation in Matrix form as,

$$\begin{matrix} I_1 \\ I_2 \\ I_3 \end{matrix} = k_d \begin{matrix} L_1^T \\ L_2^T \\ L_3^T \end{matrix} .N$$

With more than 3 lights, the equation will be of the form,

$$I = LG$$
$$where, N = (1/k_d)G$$

It can be solved using least squares equation,

$$G = (L^T L)^{-1}(L^T I)$$

Below is the code snippet to normalize intensities at each point.

```
intensities=zeros(rows,cols,num_lights);
for k=1:num_lights
    %img=kth image
    for i=1:rows
        for j=1:cols
            r=img(i,j,1);
            g=img(i,j,2);
            b=img(i,j,3);
            intensities(i,j,k)=norm(double([r g b]));
        end
    end
end
```

Below is the code snippet to calculate surface normal at each point.

```
for i=1:rows
    for j=1:cols
        I=intensities(i,j,:);
        I=reshape(I,[num_lights,1]);
        normal=(lights'*lights)\(lights'*I);
        if norm(normal)~=0
            normal=normal/norm(normal);
        else
            normal=[0;0;0];
        end
        normals(i,j,:)=normal;
    end
end
```

Once we're done calculating the normal, we create an image of the surface with uniform albedo and light at (0,0,1) shown in Figure 1.

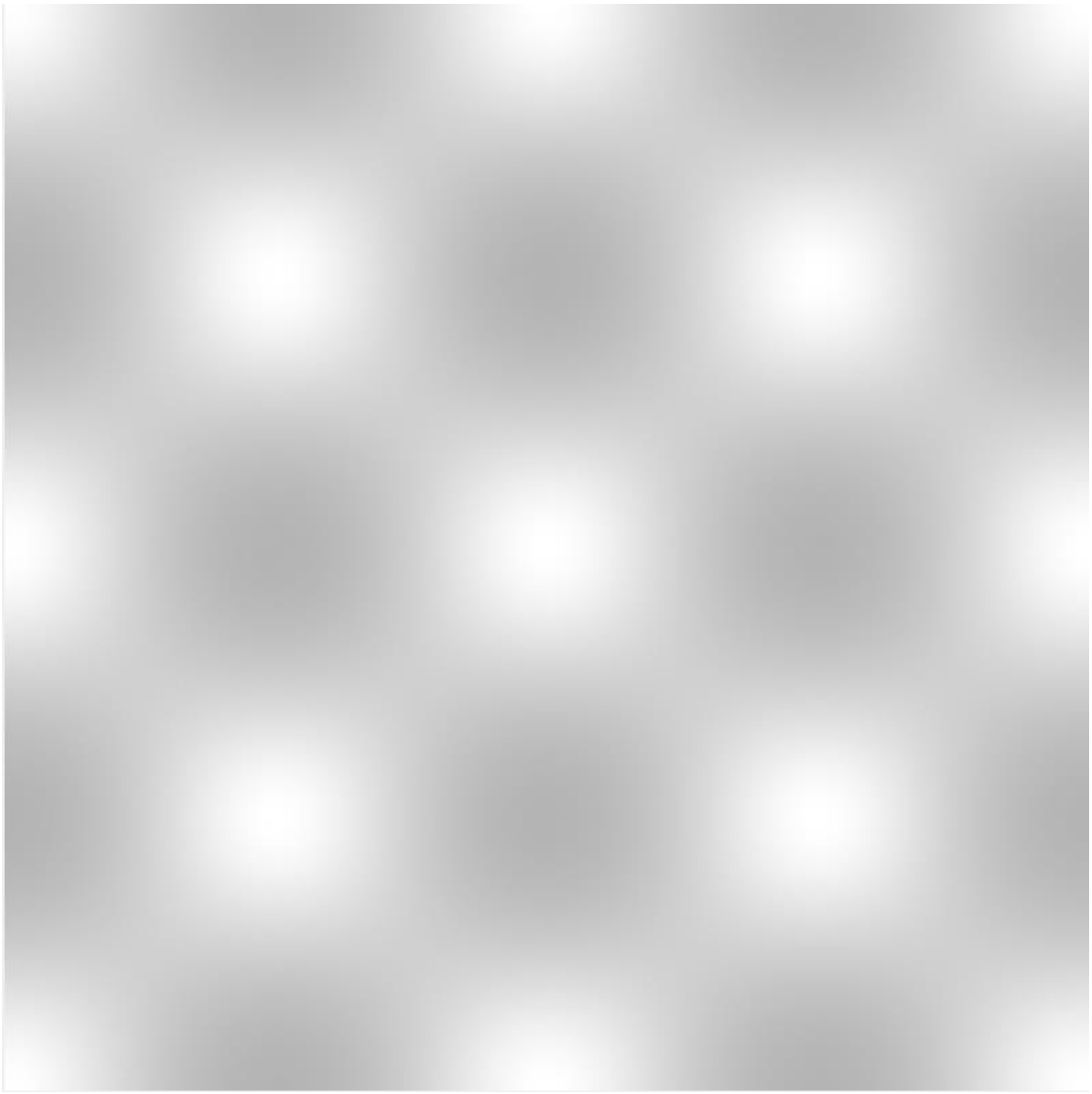

Figure 1. Image created with the calculated normal and light at (0,0,1)

Below is the code snippet to create the image,

```
newImage=zeros(rows,cols,channels);
s=[0 0 1];
for i=1:rows
    for j=1:cols
        nx=normals(i,j,1);
        ny=normals(i,j,2);
        nz=normals(i,j,3);
        shading=dot([nx,ny,nz],s);
        c=double(shading);
        if c<0
            c=0;
        end
        newImage(i,j,:)=[c c c];
    end
end
```

We now compute the depth map of the surface and make a 3D plot of the surface z=f(x,y). We start at pixel (0,0) as z=0 and calculate further depth information from the start value using partial derivatives on the way. We start by finding two vectors, $V_1 \; and \; V_2$ that are orthogonal to the normal which are given by,

$$V_1 = (x+1, y, z_{x+1,y}) - (x, y, z_{x,y})$$
$$V_1 = (1, 0, z_{x+1,y} - z_{x,y})$$
$$N.V_1 = 0$$
$$n_x + n_z(z_{x+1,y} - z_{x,y}) = 0$$

$$V_2 = (x, y+1, z_{x,y+1}) - (x, y, z_{x,y})$$
$$V_2 = (0, 1, z_{x,y+1} - z_{x,y})$$
$$N.V_2 = 0$$
$$n_x + n_z(z_{x,y+1} - z_{x,y}) = 0$$

Below is the code snippet to calculate the depth along the path.

```
Z=zeros(rows,cols);
for i=1:rows
    for j=1:cols
        nx=normals(i,j,1);
        ny=normals(i,j,2);
        nz=normals(i,j,3);
        if i==1 && j==1
            Z(i,j)=0;
        elseif j==1
            Z(i,j)=((-ny+nz*Z(i-1,j))/nz);
        else
            Z(i,j)=((-nx+nz*Z(i,j-1))/nz);
        end
    end
end

[X,Y]=meshgrid(1:rows,1:cols);
figure('Name','Part A - Surface Map');
surf(X,Y,Z);
```

The 3D plot of the surface is shown in Figure 2. We can clearly see the bulge and depth alternating in the four quadrants just like we show 4 alternating quadrants in the 2D image.

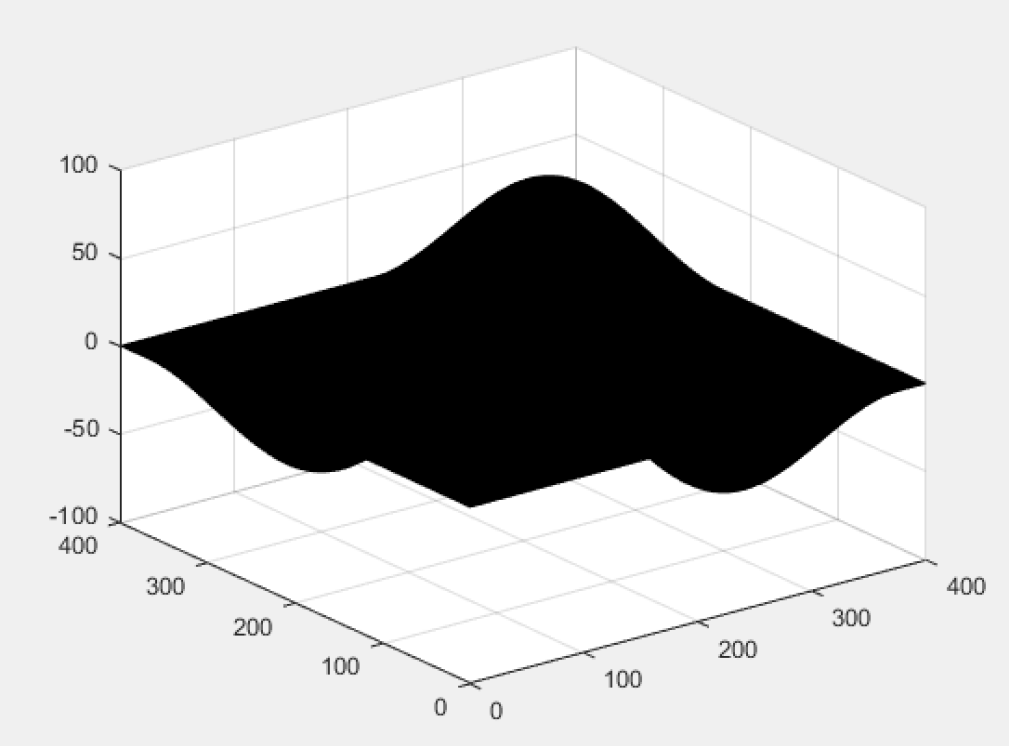

Figure 2. 3D plot of the surface showing depth information

PART B

Here, we have just one image of the surface which is illuminated by 5 colored lights at various positions with all the lights being turned on at the same time. To simplify the problem, we aggregate 5 colored lights to be represented exactly as 3 lights (All red, All green, All blue). We would need to calculate the positions of the new lights. Let us consider that the lights are placed on 3 axes (x, y, z).

We start with normalizing the position of each light and summing up the contribution of each light w.r.t the normalized position along each of the 3 channels (R, G, B). It is shown in the code snippet below.

```
lr=0;
lg=0;
lb=0;
for k=1:num_lights
    x=light_pos(k,1);
    y=light_pos(k,2);
    z=light_pos(k,3);
    r=light_colors(k,1);
    g=light_colors(k,2);
    b=light_colors(k,3);
    p=double(norm([x y z]));
    lr=double(lr+p*r);
    lg=double(lg+p*g);
    lb=double(lb+p*b);
end
```

We now calculate the intensity of each point just like before. Along the way, we also calculate the position of the new lights that replaced given 5 lights for each point. The code snippet below explains this.

```
for i=1:rows
    for j=1:cols
        r=image(i,j,1);
        g=image(i,j,2);
        b=image(i,j,3);
        intensities(i,j,1)=norm(double(r));
        intensities(i,j,2)=norm(double(g));
        intensities(i,j,3)=norm(double(b));
        lights3(i,j,1)=double(r)/lr;
        lights3(i,j,2)=double(g)/lg;
        lights3(i,j,3)=double(b)/lb;
    end
end
```

Now that we know the new position of the lights and the intensity of each point on the image, we can calculate the normal in the same way as Part A as the problem reduces to conventional photometric stereo problem.

We now calculate the 3D depth at each point starting at (0,0) as z=0 depth like what we did in part A. We continue calculating the depth at further points using partial derivatives. Figure 3 and Figure 4 shows the 3D surface depth for '`color_photometric_stereo_1.tiff`' and '`color_photometric_stereo_2.tiff`' respectively.

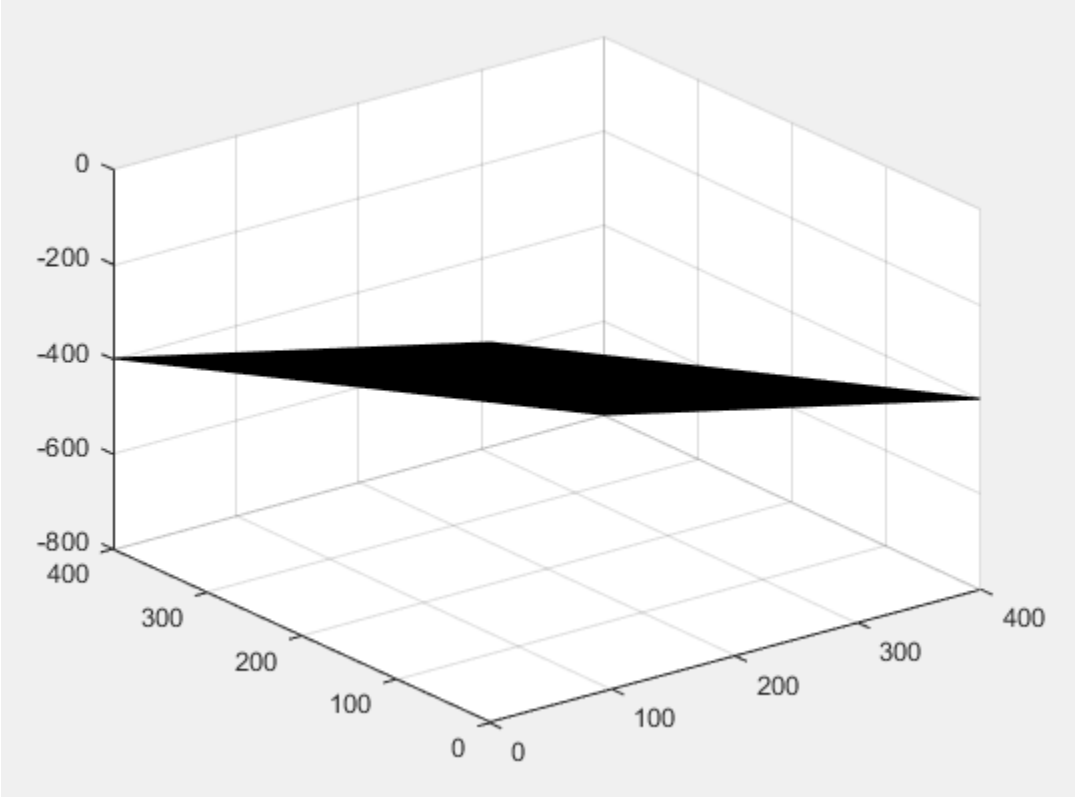

Figure 3. 3D plot of the surface '`color_photometric_stereo_1.tiff`' showing depth

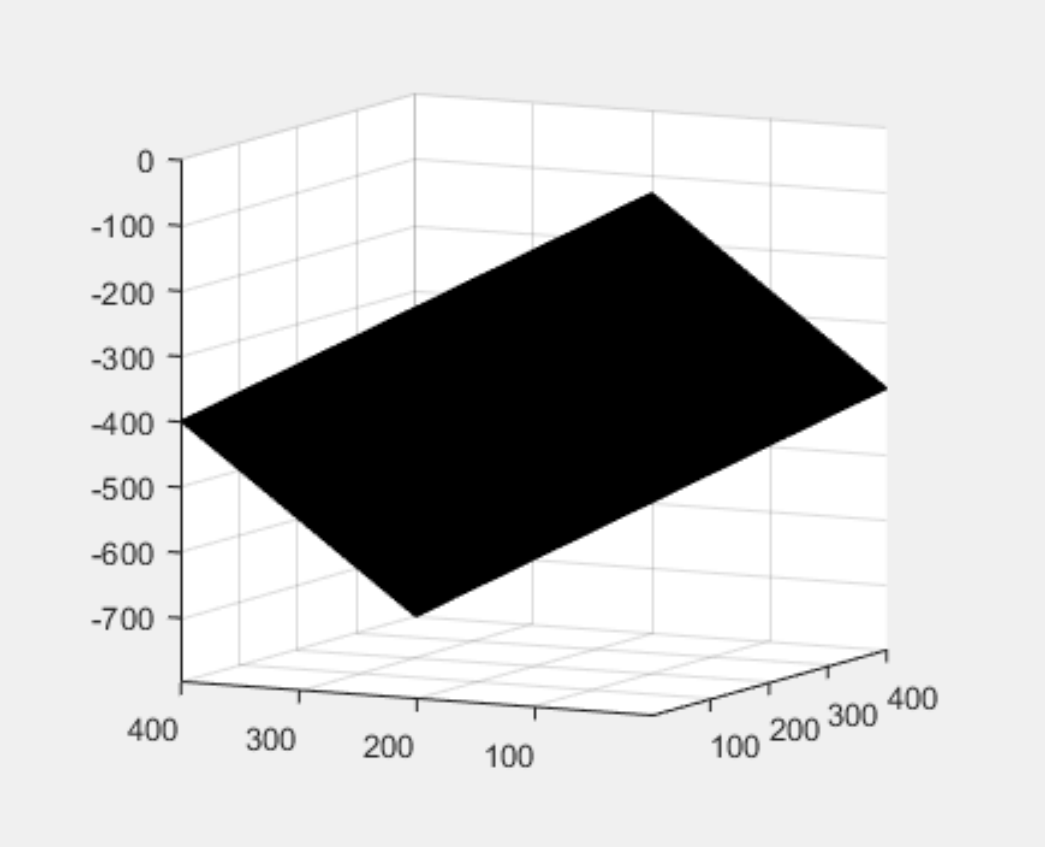

Figure 4. 3D plot of the surface '`color_photometric_stereo_2.tiff`' showing depth

PART C

Consulting Figure 5, we will derive an expression for g as a function of $\varphi_i - \varphi_0$ and vice-versa.

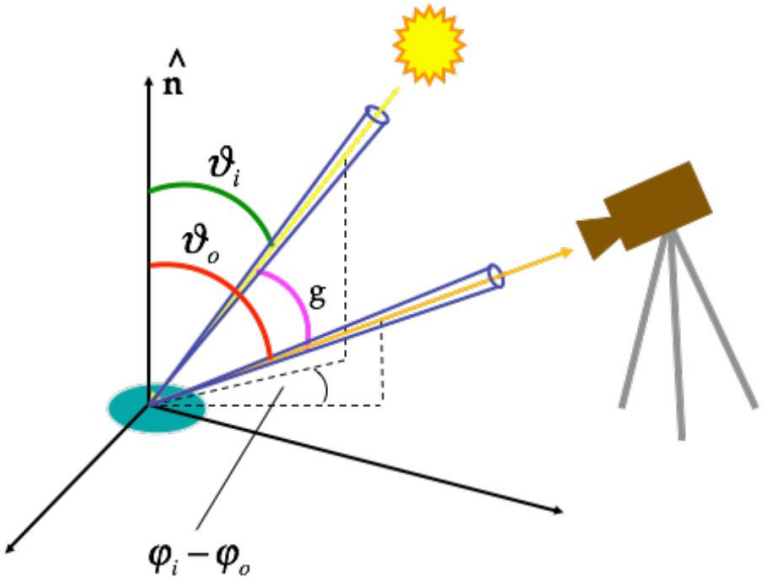

Figure 5. The BRDF figure from notes

If we consider the incident light vector as A and the reflected light vector toward the camera as B, the dot product between them would give us,

$$\cos g = A.B$$

The angle between the incident and reflected light vector, g, is the same as the difference in projections of these vectors on ZX plane.

$$g = \varphi_i - \varphi_0$$

If we assume that the camera is placed along the Z axis, then $\varphi_0 = 0$. Therefore, we get,

$$\cos \varphi_i = A.B$$

REFERENCES

Apart from the lecture slides and class notes, I referred to the following links for more information.

1. http://pages.cs.wisc.edu/~csverma/CS766_09/Stereo/stereo.html#Normal_Map_Generation_:
2. http://pages.cs.wisc.edu/~lizhang/courses/cs766-2008f/syllabus/10-09-shading/shading.pdf
3. http://kobus.ca/research/projects/depth_exam/depth_exam.html