Homework 3
Problem 2
Project Report
Photometric Stereo and Shape from Shading

Fnu Karan
NetID: kx361

April 9, 2017

Date Performed:	April 4, 2017
Instructor:	Professor Gerig

1 Objective

The objective is to come up with the 3D Geometry of a 3D object, with its 2D images taken by a camera with same light source at different positions.

2 Data Capture

2.1 Problem 2.1

The image samples are shown in Figure 1.

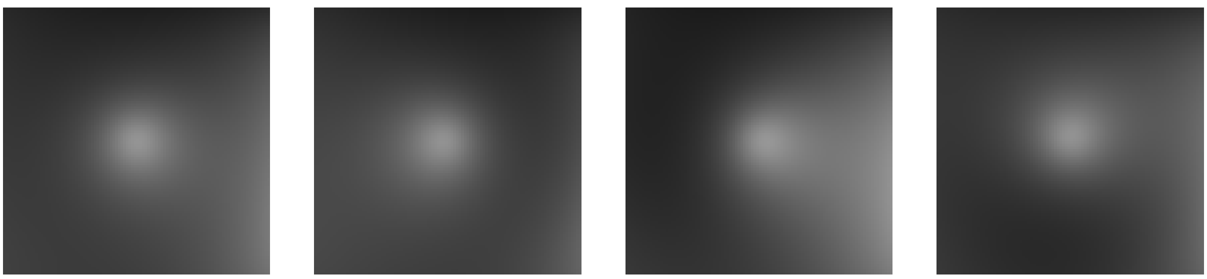


Figure 1: Input Synthetic Images

Along with these images, we are also provided with the light source vectors.

$$V_1 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$$
$$V_2 = \begin{bmatrix} -0.2 & 0 & 1 \end{bmatrix}^T$$
$$V_3 = \begin{bmatrix} 0.2 & 0 & 1 \end{bmatrix}^T$$
$$V_4 = \begin{bmatrix} 0 & -0.2 & 1 \end{bmatrix}^T$$

2.2 Problem 2.2

2.2.1 Sphere

The image samples are shown in Figure 2.

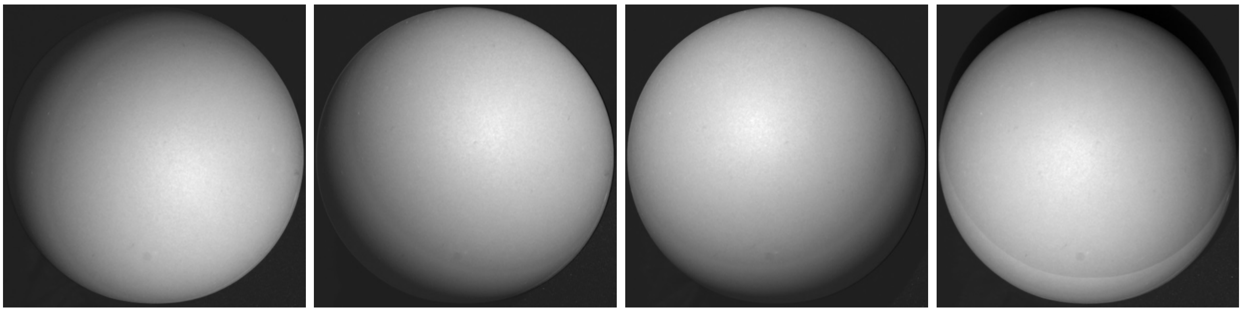


Figure 2: Input Sphere Images

Along with these images, we are also provided with the light source vectors.

$$V_1 = \begin{bmatrix}0.38359 & 0.236647 & 0.89266\end{bmatrix}^T$$
$$V_2 = \begin{bmatrix}0.372825 & -0.303914 & 0.87672\end{bmatrix}^T$$
$$V_3 = \begin{bmatrix}-0.250814 & -0.34752 & 0.903505\end{bmatrix}^T$$
$$V_4 = \begin{bmatrix}-0.203844 & 0.096308 & 0.974255\end{bmatrix}^T$$

2.2.2 Dog

The image samples are shown in Figure 3.

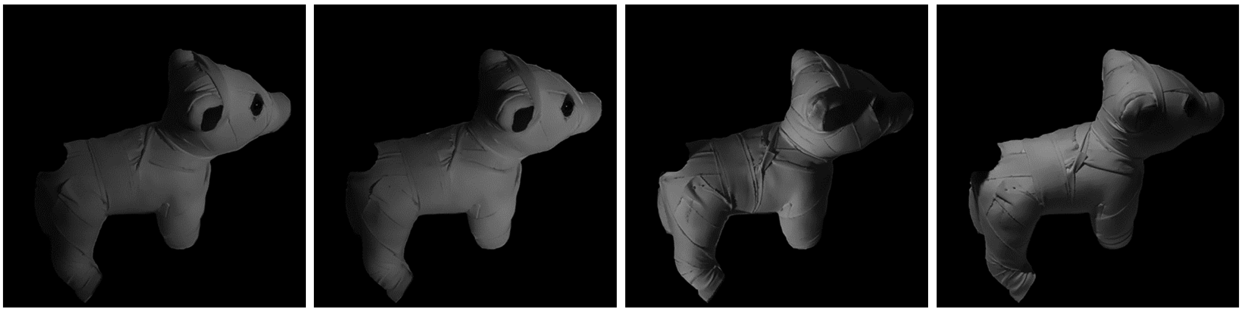


Figure 3: Input Dog Images

Along with these images, we are also provided with the light source vectors.

$$V_1 = \begin{bmatrix}16 & 19 & 30\end{bmatrix}^T$$
$$V_2 = \begin{bmatrix}13 & 16 & 30\end{bmatrix}^T$$
$$V_3 = \begin{bmatrix}-17 & 10.5 & 26.5\end{bmatrix}^T$$
$$V_4 = \begin{bmatrix}9 & -25 & 4\end{bmatrix}^T$$

3 Experimental Procedure

I followed this theory from the textbook to get to my results. We can represent each surface element as a vector:

$$S = \begin{bmatrix} x \\ y \\ z(x,y) \end{bmatrix} \tag{1}$$

Then, we can compute the partial derivatives of the surface as:

$$\frac{\partial S}{\partial x} = \begin{bmatrix} 1 \\ 0 \\ \frac{\partial z(x,y)}{\partial x} \end{bmatrix} \quad \frac{\partial S}{\partial y} = \begin{bmatrix} 0 \\ 1 \\ \frac{\partial z(x,y)}{\partial y} \end{bmatrix} \tag{2}$$

We can then define two variables p and q such as:

$$p = \frac{\partial z}{\partial x} \quad q = \frac{\partial z}{\partial y} \tag{3}$$

So we obtain,

$$\frac{\partial S}{\partial x} = \begin{bmatrix} 1 \\ 0 \\ p \end{bmatrix} \quad \frac{\partial S}{\partial y} = \begin{bmatrix} 0 \\ 1 \\ q \end{bmatrix} \tag{4}$$

Then to compute the normal vector, we are going to rely on the fact that a normal is always orthogonal to the surface so its always orthogonal to the surface gradients. The best way to obtain an orthogonal vector from two vectors already orthogonal is to compute the cross (vector) product:

$$\vec{n} = \frac{\partial S}{\partial x} \times \frac{\partial S}{\partial y} = \begin{bmatrix} 1 \\ 0 \\ p \end{bmatrix} \times \begin{bmatrix} 0 \\ 1 \\ q \end{bmatrix} = \begin{bmatrix} -p \\ -q \\ 1 \end{bmatrix} \tag{5}$$

The last step to get a real normal vector is to give it a unit norm. So finally we have:

$$\hat{n} = \frac{\vec{n}}{\sqrt{p^2 + q^2 + 1}} \tag{6}$$

So we obtained the equation to estimate the normal vector associated to a surface element, which is function of the image intensity gradients regarding the 2D spatial variables of the image: x and y. The unknown to determine in this problem are then p and q. So lets discuss what they represent and how we can get them.
In our problem, we deal with a light source which also have its own normal vector and which could be expressed using the same method:

$$\hat{n_s} = \frac{\vec{n_s}}{\sqrt{p_s^2 + q_s^2 + 1}} \qquad \textit{with} \qquad n_s = \begin{bmatrix} -p_s \\ -q_s \\ 1 \end{bmatrix} \tag{7}$$

So we have the reflectance map as:

$$R(p,q) = \frac{1 + pp_s + qq_s}{\sqrt{1 + p^2 + q^2}\sqrt{1 + p_s^2 + q_s^2}} \tag{8}$$

Now, we have the equation,

$$I_j(x,y) = k\rho(x,y)\hat{N}(x,y)V_j \tag{9}$$

resulting in,

$$i(x,y) = \mathcal{V}g(x,y) \tag{10}$$

where i and $\mathcal{V}$ are:

$$i(x,y) = \begin{bmatrix} I_1(x,y) \\ I_2(x,y) \\ \ldots \\ I_n(x,y) \end{bmatrix} \quad \mathcal{V} = \begin{bmatrix} V_1^T \\ V_2^T \\ \ldots \\ V_n^T \end{bmatrix} \tag{11}$$

We can extract albedo from a measurement of g because N is the unit normal. This means $|g(x,y)| = \rho$(x,y). Because the albedo is in the range zero to one, any pixels where $|g|$ is greater than one are suspect - either the pixel is not working or $\mathcal{V}$ is incorrect.

We can extract the surface normal from g because the normal is unit vector.

$$N(x,y) = \frac{g(x,y)}{|g(x,y)|} \tag{12}$$

The surface is $(x, y, f(x, y))$, so the normal as a fuction of (x, y) is

$$N(x,y) = \frac{1}{\sqrt{1 + \frac{\partial f}{\partial x}^2 + \frac{\partial f}{\partial y}^2}} \begin{bmatrix} \frac{\partial f}{\partial x}^2 & \frac{\partial f}{\partial y}^2 & 1 \end{bmatrix}^T \tag{13}$$

To determine depth we need to determine $f(x, y)$ from measured values of the unit normal.

$$f(x,y) = \oint_C \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right) dl + c \tag{14}$$

In addition to this depth, the question asked us to calculate the depth by using one of the algorithms by $frankotchellappa$, whose results are also in the next section. Once we have depth we can plot the surface using MATLAB functions.

4 Results

4.1 Problem 2.1 - Synthetic Images

Here are the results obtained for synthetic images after coding the above explained algorithm in MATLAB.

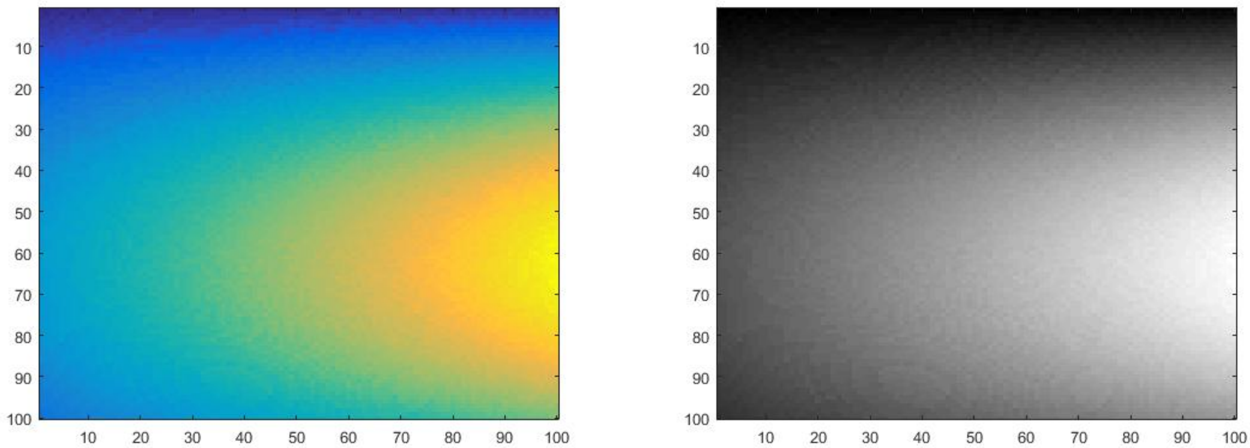
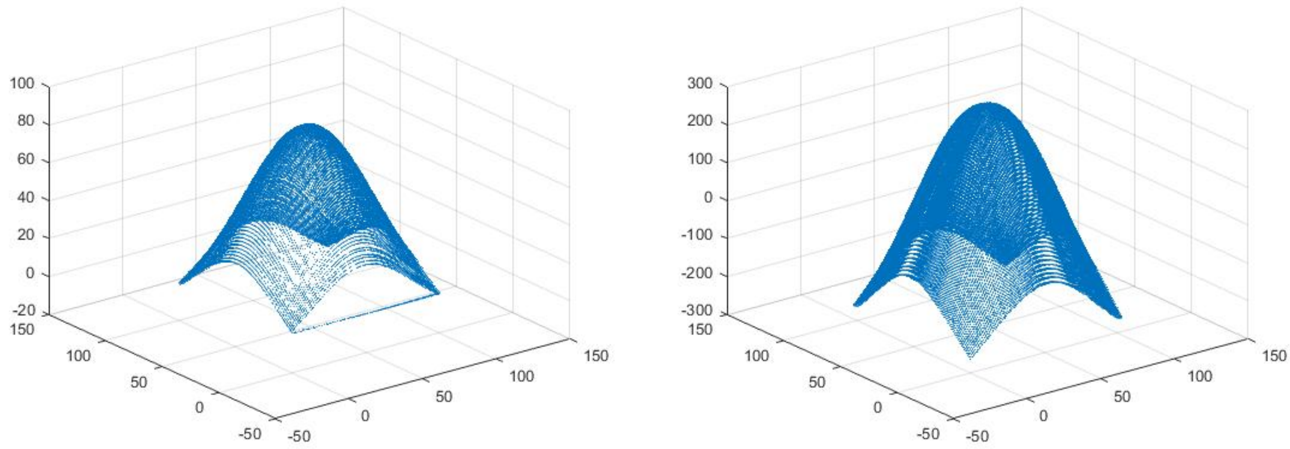


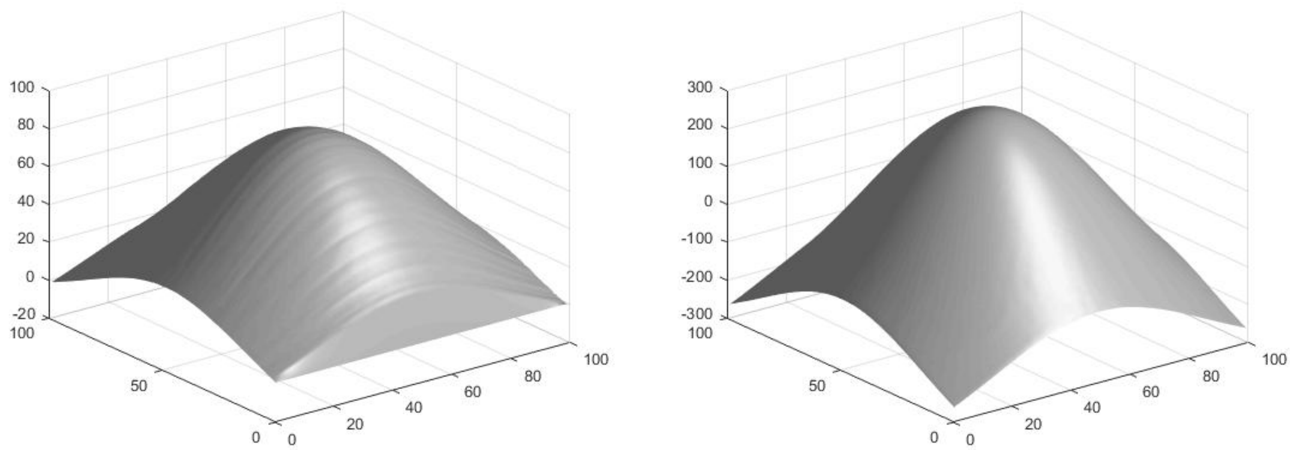
Figure 4: Albedo



(a) Normal vector from our calculation

(b) Frankotchellappa algorithm

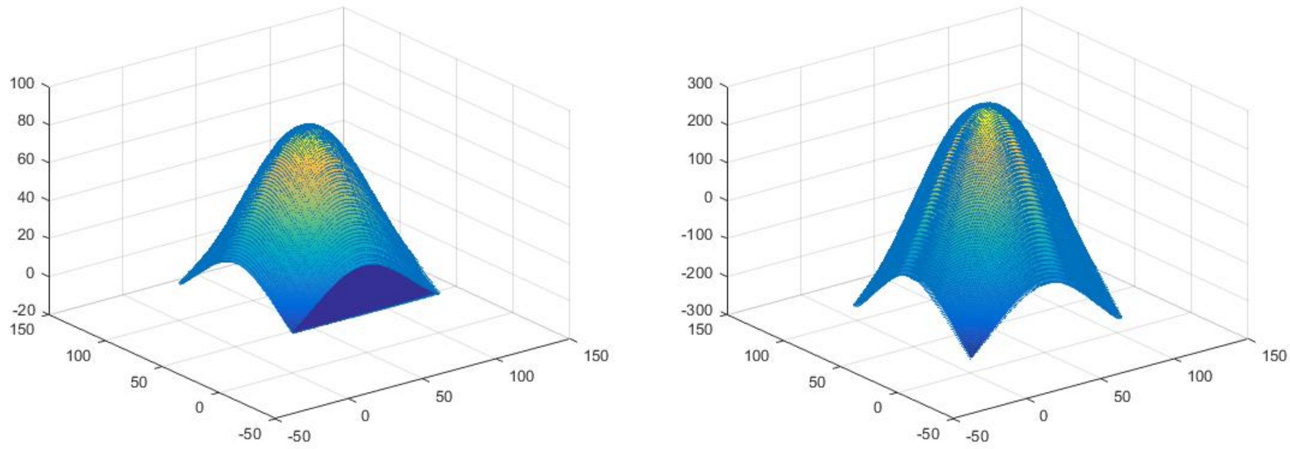
Figure 5: Normal Vector



(a) Graylevel depth image from our calculation

(b) Frankotchellappa Algorithm

Figure 6: Graylevel depth image



(a) Wireframe of a depth map from our calculation

(b) Frankotchellappa Algorithm

Figure 7: Wireframe of a depth map

4.2 Problem 2.2

4.2.1 Sphere

Here are the results obtained for sphere images after coding the above explained algorithm in MATLAB.

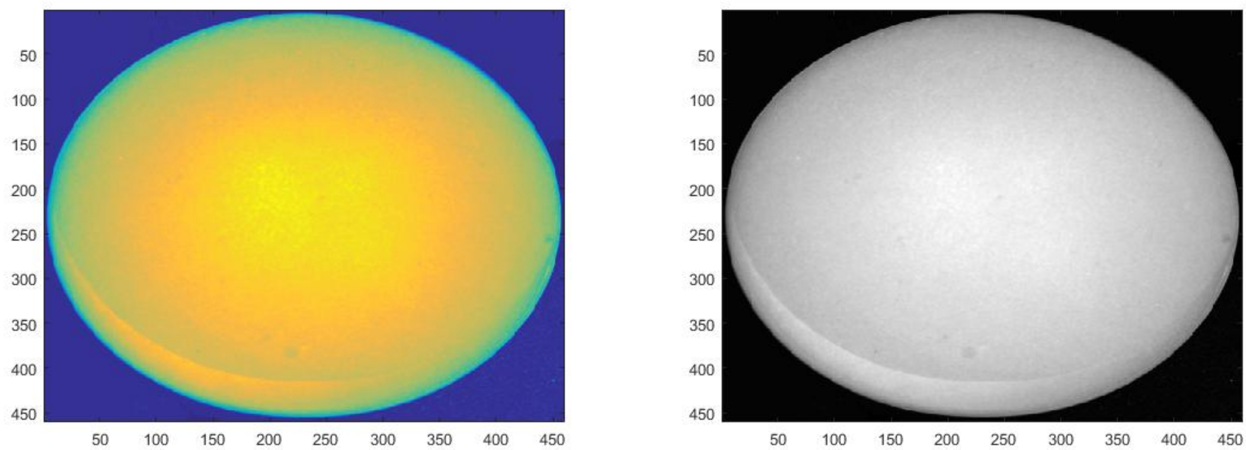
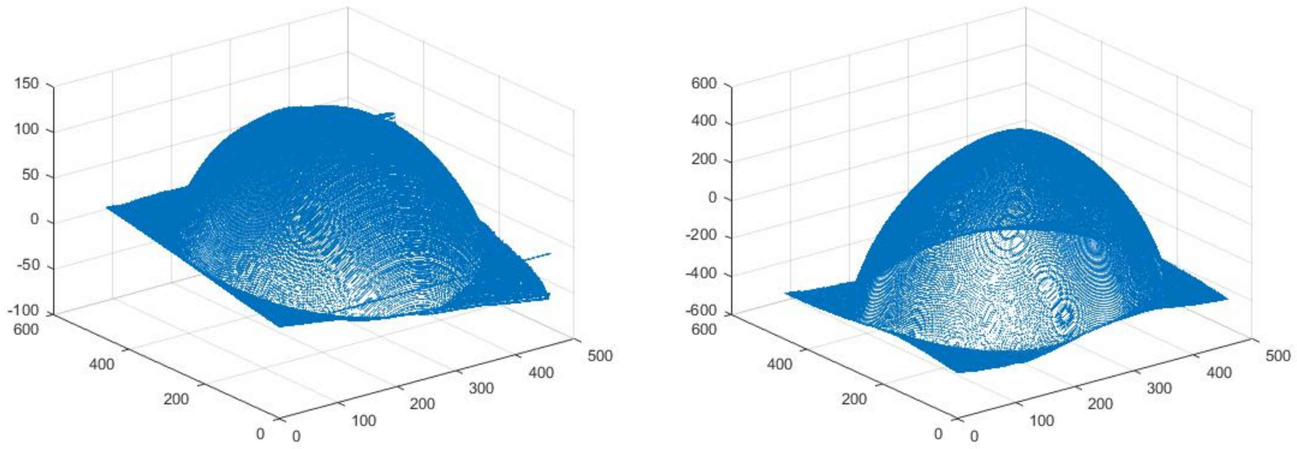


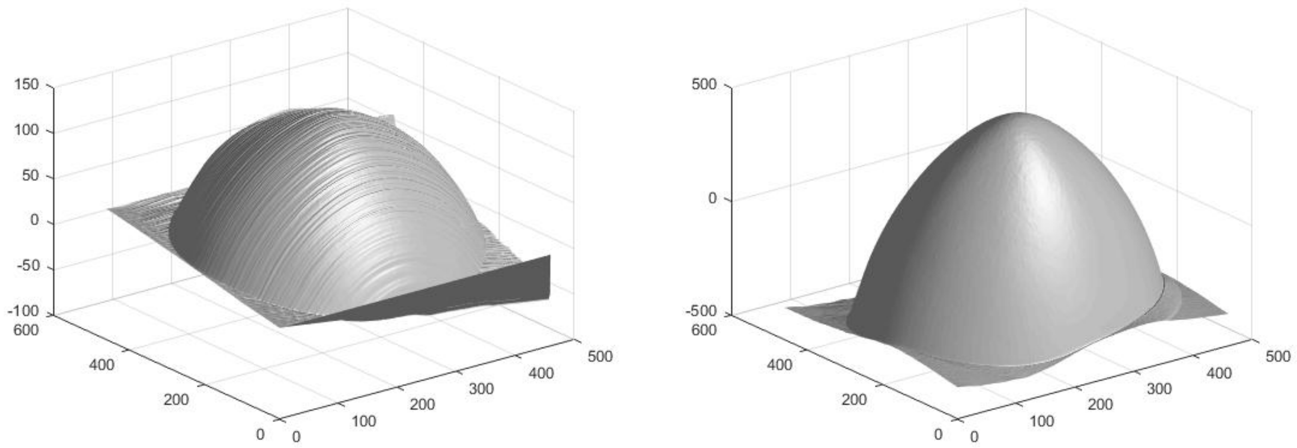
Figure 8: Albedo



(a) Normal vector from our calculation

(b) Frankotchellappa algorithm

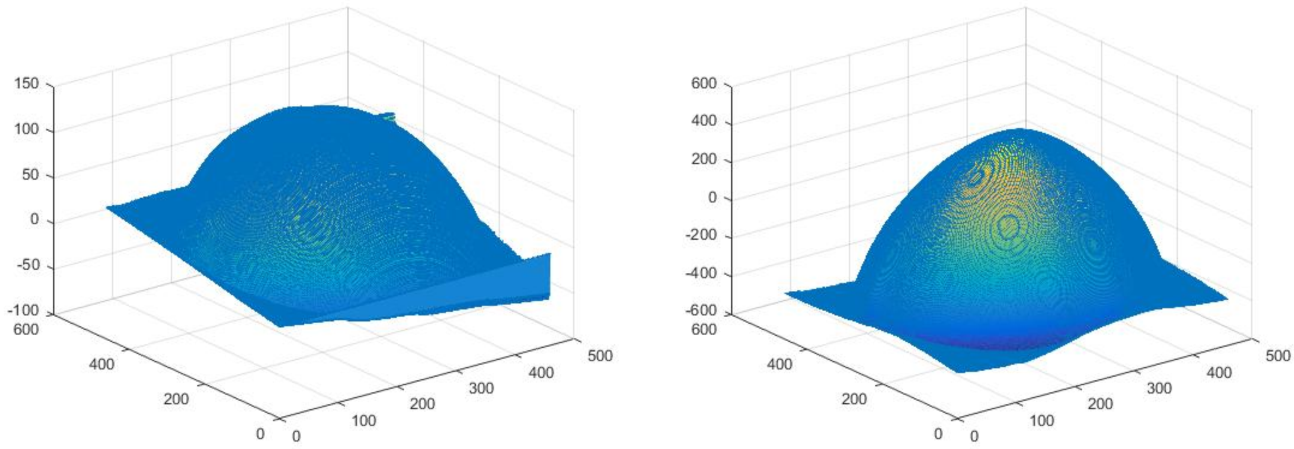
Figure 9: Normal Vector



(a) Graylevel depth image from our calculation

(b) Frankotchellappa Algorithm

Figure 10: Graylevel depth image



(a) Wireframe of a depth map from our calculation

(b) Frankotchellappa Algorithm

Figure 11: Wireframe of a depth map

4.2.2 Dog

Here are the results obtained for sphere images after coding the above explained algorithm in MATLAB.

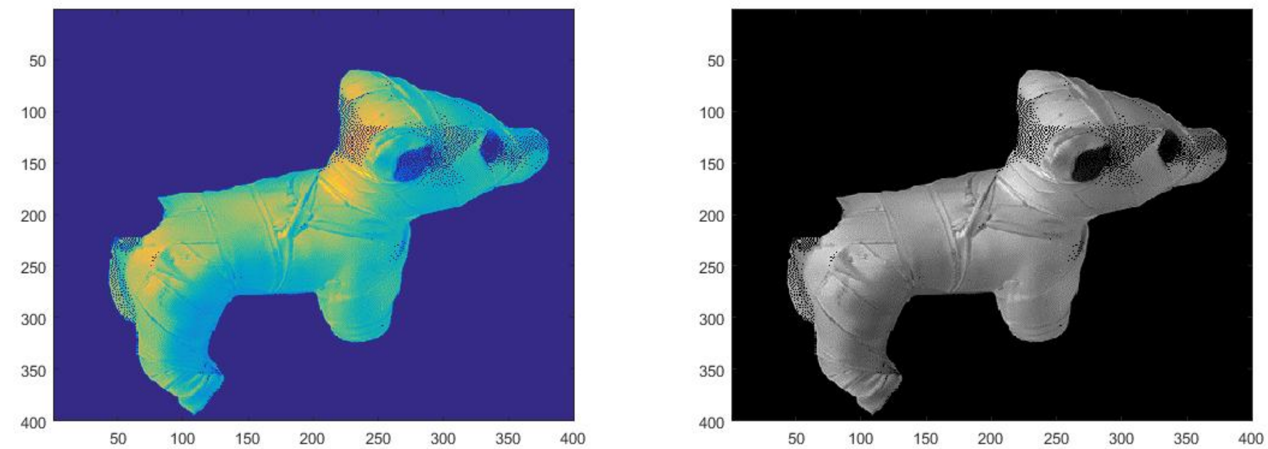
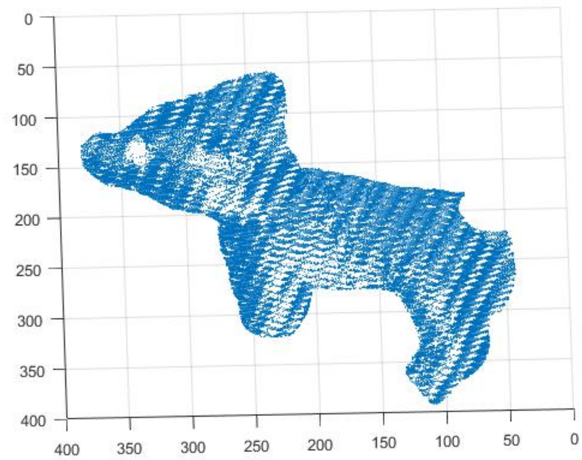
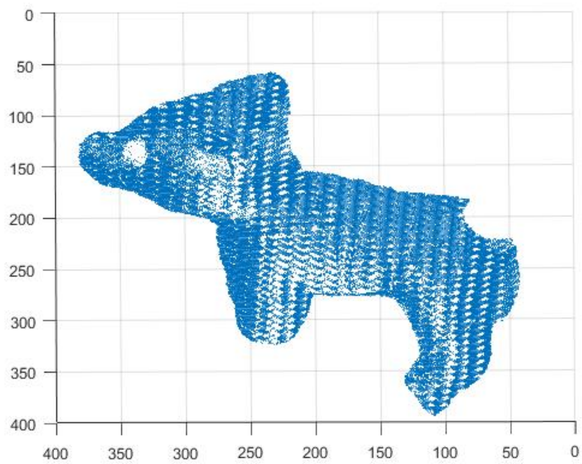


Figure 12: Albedo

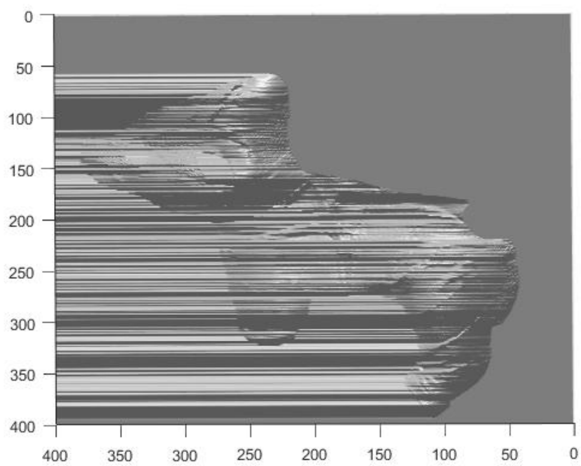


(a) Normal vector from our calculation

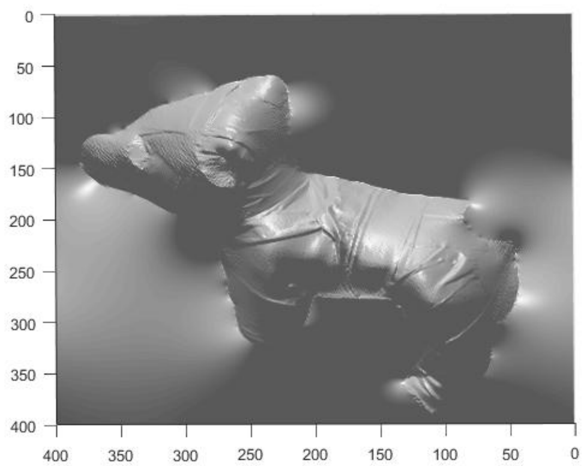


(b) Frankotchellappa algorithm

Figure 13: Normal Vector

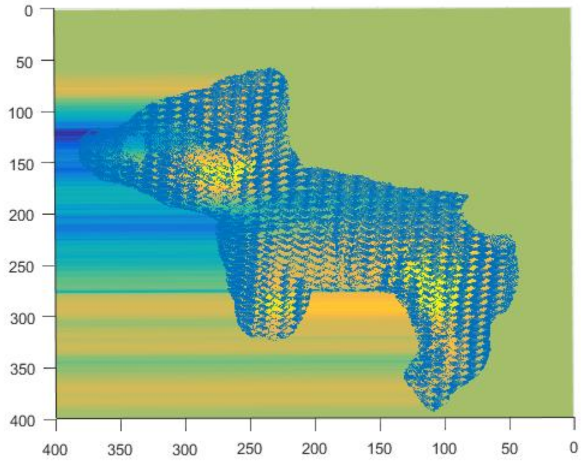


(a) Graylevel depth image from our calculation

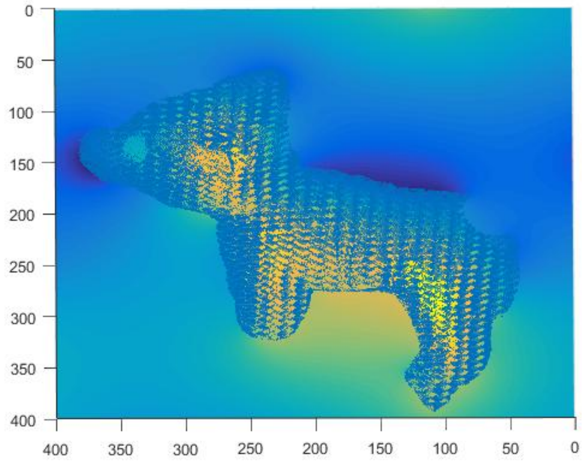


(b) Frankotchellappa Algorithm

Figure 14: Graylevel depth image



(a) Wireframe of a depth map from our calculation



(b) Frankotchellappa Algorithm

Figure 15: Wireframe of a depth map

5 Discussion

5.1 Problem 2.1 - Synthetic Images

For the results of synthetic images, all four images were used. According to these images, we see that we are dealing with a bright area in the center and darker in the surrounding areas. Lets analyse this result (Figure 4) because its quite surprising at first sight regarding the images we used. If we look at the second and the third image where the light source is symmetrical. We have (0.2, 0, 1) and (-0.2, 0, 1) for the second and the third image respectively. If we compare them we can see that the third image is brighter on the right side of the third image. We can also add that this albedo map make some sense with regard to the dark areas which appear to have a low albedo value and the center of the image has an albedo that seems to agree with the images.

If we analyse the normal's result, then the brightest area on the pictures is the center of the image which clearly correspond to the summit of this shape and the normals are clearly oriented to the light source. (Figure 5)

Finally, we plot the final surface reconstruction we obtained for this set of images. In this resoncstruction, we used all the four images given, which also corresponds to the normal vectors as discussed above. (Figure 6 and 7.)

One interesting thing to note is that the results are pretty close to the algorithm by frankotchellappa. The normals, and depth images are almost the same. The difference comes near the edges, so I guess that the while integrating there have been some problems which I was not able to resolve.

5.2 Problem 2.2

5.2.1 Sphere

For this dataset, again all the given four images were used. We can see that the albedo seems fair, since the most bright region is towards the center and decreasing brightness as we go away from it. This is demonstrated in Figure 8.

Coming over to Normal vectors, they are pretty close as to what the images are. The shape appears to be pretty good, but there are some issues which appear as shown in figure 9. Let's have a look at the shape reconstruction. As we can see in figure 10 and 11 that again near the edges the algorithm does not seem to work, which was also the problem in the synthetic images as well, but this is not the problem in the algorithm by frankotchellappa.

5.2.2 Dog

The albedo seems fair, since in the images we can see that the there is area which is the brightest, and all the darker regions are indicates by blue (Figure 12)

Coming on to the normal vectors, they also seem fair comparable to the algorithm by frankotchellappa.

The problem arises when we try and reconstruct the images. Again the reconstruction seems fine in the start but as we approach towards the end, the fall in depth is not treated properly and that results in this bad recontruction. On the other hand the image reconstruction using the algorithm by frankotchellappa, we get a very good reconstruction (Figure 14 and 15).

This demonstrates that in our algorithm, depth calculation is the only issue here. Rest the algorithm seems to work pretty fine.

6 MATLAB Code

The code is self explanatory and can be understood with the comments written in the code.

6.1 Problem 2.1 - Synthetic Images

```
clc
clear

%Read the images
i1 = imread('C:\Users\K-Chak\Google Drive\NYU\Spring 2017\Computer Vision\Assignment
    \3\p2\synth-images\im1.png');
i2 = imread('C:\Users\K-Chak\Google Drive\NYU\Spring 2017\Computer Vision\Assignment
    \3\p2\synth-images\im2.png');
i3 = imread('C:\Users\K-Chak\Google Drive\NYU\Spring 2017\Computer Vision\Assignment
    \3\p2\synth-images\im3.png');
i4 = imread('C:\Users\K-Chak\Google Drive\NYU\Spring 2017\Computer Vision\Assignment
    \3\p2\synth-images\im4.png');

%Scaling the images
i1 = double(i1)/255;
i2 = double(i2)/255;
i3 = double(i3)/255;
i4 = double(i4)/255;

%Initializing the light vectors and normalizing them
v1 = [0 0 1]';
v1=v1/norm(v1,2);
v2 = [-0.2 0 1]';
v2=v2/norm(v2,2);
v3 = [0.2 0 1]';
v3=v3/norm(v3,2);
v4 = [0 -0.2 1]';
v4=v4/norm(v4,2);

%Create a compiled light source vector
V = [v1'; v2'; v3'; v4'];

rows = size(i1,1);
cols = size(i1,2);

%Initialize g, albedo, p, q, normal, depth
g = zeros(rows,cols,3);
albedo = zeros(rows,cols);
p =  zeros(rows,cols);
q =  zeros(rows,cols);
normal = zeros(rows,cols, 3);
depth=zeros(rows,cols);

%Running the algorithm in the textbook
for x = 1:rows;
    for y = 1:cols;
```

```

        %creating the i matrix
        i = [i1(x,y); i2(x,y); i3(x,y); i4(x,y)];

        %creating the diagonal matrix from i
        I = diag(i);

        %preparing the parameters to get g
        A = I * i;
        B = I * V;

        if (rank(B)< 3)
                continue;
        end

        %solving for g
        temp_g = B\A;

        %Storing the g for this x,y in the matrix
        g(x,y,:) = temp_g;
        %calculating albedo
        albedo(x,y) = norm(temp_g);
        %calculating the normal
        normal(x,y,:) = temp_g/norm(temp_g);

        %calculating the gradient in x and y
        p(x,y) = normal(x,y,1)/normal(x,y,3);
        q(x,y) = normal(x,y,2)/normal(x,y,3);
    end
end

%normalizing the albedo
maxalbedo = max(max(albedo));
if( maxalbedo > 0)
    albedo = albedo/maxalbedo;
end

%calculating depth across first column
for i = 2:rows
    depth(i,1) = depth(i-1,1) + q(i,1);
end

%calculating depth across all rows using calculated depth of first column
for i = 2:rows
    for j = 2:cols
        depth(i,j) = depth(i,j-1)+p(i,j);
    end
end

%Albedo Map Colored
figure;
imagesc(albedo);

%Albedo Map Grayscale
figure;
imagesc(albedo);
colormap(gray);

%Normal Vectors
figure
```

```
spacing = 1;
[temp_g ,Y] = meshgrid(1:spacing:rows, 1:spacing:cols);
quiver3(temp_g,Y,-depth, normal(:,:,1),normal(:,:,2),normal(:,:,3))

%Graylevel depth image
figure
surfl(-depth);
colormap(gray);
grid on;
shading interp

%Wireframe of a depth map
figure
spacing = 1;
[temp_g ,Y] = meshgrid(1:spacing:rows, 1:spacing:cols);
quiver3(temp_g,Y,-depth, normal(:,:,1),normal(:,:,2),normal(:,:,3))
hold on
surf( temp_g, Y, -depth, 'EdgeColor', 'none' );
hold off

%Calculating depth using frankotchellappa algorithm
fkDepth = frankotchellappa(p,q);

%Normal Vectors
figure
spacing = 1;
[temp_g ,Y] = meshgrid(1:spacing:rows, 1:spacing:cols);
quiver3(temp_g,Y,-fkDepth, normal(:,:,1),normal(:,:,2),normal(:,:,3))

%Graylevel depth image
figure
surfl(-fkDepth);
colormap(gray);
grid on;
shading interp

%Wireframe of a depth map
figure
spacing = 1;
[temp_g ,Y] = meshgrid(1:spacing:rows, 1:spacing:cols);
quiver3(temp_g,Y,-fkDepth, normal(:,:,1),normal(:,:,2),normal(:,:,3))
hold on
surf( temp_g, Y, -fkDepth, 'EdgeColor', 'none' );
hold off
```

6.2 Problem 2.2

6.2.1 Sphere

```
clc
clear

%Read the images
i1 = imread('C:\Users\K-Chak\Google Drive\NYU\Spring 2017\Computer Vision\Assignment
    \3\p2\sphere-images\real1.bmp');
i2 = imread('C:\Users\K-Chak\Google Drive\NYU\Spring 2017\Computer Vision\Assignment
    \3\p2\sphere-images\real2.bmp');
i3 = imread('C:\Users\K-Chak\Google Drive\NYU\Spring 2017\Computer Vision\Assignment
    \3\p2\sphere-images\real3.bmp');
```

```
i4 = imread('C:\Users\K-Chak\Google Drive\NYU\Spring 2017\Computer Vision\Assignment
    \3\p2\sphere-images\real4.bmp');

%Scaling the images
i1 = double(i1)/255;
i2 = double(i2)/255;
i3 = double(i3)/255;
i4 = double(i4)/255;

%Initializing the light vectors and normalizing them
v1 = [0.38359 0.236647 0.89266]';
v1=v1/norm(v1,2);
v2 = [0.372825 -0.303914 0.87672]';
v2=v2/norm(v2,2);
v3 = [-0.250814 -0.34752 0.903505]';
v3=v3/norm(v3,2);
v4 = [-0.203844 0.096308 0.974255]';
v4=v4/norm(v4,2);

%Create a compiled light source vector
V = [v1'; v2'; v3'; v4'];

rows = size(i1,1);
cols = size(i1,2);

%Initialize g, albedo, p, q, normal, depth
g = zeros(rows,cols,3);
albedo = zeros(rows,cols);
p =  zeros(rows,cols);
q =  zeros(rows,cols);
normal = zeros(rows,cols, 3);
depth=zeros(rows,cols);

%Running the algorithm in the textbook
for x = 1:rows;
    for y = 1:cols;

        %creating the i matrix
        i = [i1(x,y); i2(x,y); i3(x,y); i4(x,y)];

        %creating the diagonal matrix from i
        I = diag(i);

        %preparing the parameters to get g
        A = I * i;
        B = I * V;

        if (rank(B)< 3)
                continue;
        end

        %solving for g
        temp_g = B\A;

        %Storing the g for this x,y in the matrix
        g(x,y,:) = temp_g;
        %calculating albedo
        albedo(x,y) = norm(temp_g);
        %calculating the normal
        normal(x,y,:) = temp_g/norm(temp_g);
```

```
        %calculating the gradient in x and y
        p(x,y) = normal(x,y,1)/normal(x,y,3);
        q(x,y) = normal(x,y,2)/normal(x,y,3);
    end
end

%normalizing the albedo
maxalbedo = max(max(albedo));
if( maxalbedo > 0)
    albedo = albedo/maxalbedo;
end

%calculating depth across first column
for i = 2:rows
    depth(i,1) = depth(i-1,1) + q(i,1);
end

%calculating depth across all rows using calculated depth of first column
for i = 2:rows
    for j = 2:cols
        depth(i,j) = depth(i,j-1)+p(i,j);
    end
end

%Albedo Map Colored
figure;
imagesc(albedo);

%Albedo Map Grayscale
figure;
imagesc(albedo);
colormap(gray);

%Normal Vectors
figure
spacing = 1;
[temp_g ,Y] = meshgrid(1:spacing:rows, 1:spacing:cols);
quiver3(temp_g,Y,-depth, normal(:,:,1),normal(:,:,2),normal(:,:,3))

%Graylevel depth image
figure
surfl(-depth);
colormap(gray);
grid on;
shading interp

%Wireframe of a depth map
figure
spacing = 1;
[temp_g ,Y] = meshgrid(1:spacing:rows, 1:spacing:cols);
quiver3(temp_g,Y,-depth, normal(:,:,1),normal(:,:,2),normal(:,:,3))
hold on
surf( temp_g, Y, -depth, 'EdgeColor', 'none' );
hold off

%Calculating depth using frankotchellappa algorithm
fkDepth = frankotchellappa(p,q);

%Normal Vectors
```

```
figure
spacing = 1;
[temp_g ,Y] = meshgrid(1:spacing:rows, 1:spacing:cols);
quiver3(temp_g,Y,-fkDepth, normal(:,:,1),normal(:,:,2),normal(:,:,3))

%Graylevel depth image
figure
surfl(-fkDepth);
colormap(gray);
grid on;
shading interp

%Wireframe of a depth map
figure
spacing = 1;
[temp_g ,Y] = meshgrid(1:spacing:rows, 1:spacing:cols);
quiver3(temp_g,Y,-fkDepth, normal(:,:,1),normal(:,:,2),normal(:,:,3))
hold on
surf( temp_g, Y, -fkDepth, 'EdgeColor', 'none' );
hold off
```

6.2.2 Dog

```
clc
clear

%Read the images
i1 = imread('C:\Users\K-Chak\Google Drive\NYU\Spring 2017\Computer Vision\Assignment
    \3\p2\dog-png\dog1.png');
i2 = imread('C:\Users\K-Chak\Google Drive\NYU\Spring 2017\Computer Vision\Assignment
    \3\p2\dog-png\dog2.png');
i3 = imread('C:\Users\K-Chak\Google Drive\NYU\Spring 2017\Computer Vision\Assignment
    \3\p2\dog-png\dog3.png');
i4 = imread('C:\Users\K-Chak\Google Drive\NYU\Spring 2017\Computer Vision\Assignment
    \3\p2\dog-png\dog4.png');

%Scaling the images
i1 = double(i1)/255;
i2 = double(i2)/255;
i3 = double(i3)/255;
i4 = double(i4)/255;

%Initializing the light vectors and normalizing them
v1 = [16 19 30]';
v1=v1/norm(v1,2);
v2 = [13 16 30]';
v2=v2/norm(v2,2);
v3 = [-17 10.5 26.5]';
v3=v3/norm(v3,2);
v4 = [9 -25 4]';
v4=v4/norm(v4,2);

%Create a compiled light source vector
V = [v2'; v3'; v4'];

rows = size(i1,1);
cols = size(i1,2);

%Initialize g, albedo, p, q, normal, depth
```

```
g = zeros(rows,cols,3);
albedo = zeros(rows,cols);
p =  zeros(rows,cols);
q =  zeros(rows,cols);
normal = zeros(rows,cols, 3);
depth=zeros(rows,cols);

%Running the algorithm in the textbook
for x = 1:rows;
    for y = 1:cols;

        %creating the i matrix
        i = [i2(x,y); i3(x,y); i4(x,y)];

        %creating the diagonal matrix from i
        I = diag(i);

        %preparing the parameters to get g
        A = I * i;
        B = I * V;

        if (rank(B)< 3)
                continue;
        end

        %solving for g
        temp_g = B\A;

        %Storing the g for this x,y in the matrix
        g(x,y,:) = temp_g;
        %calculating albedo
        albedo(x,y) = norm(temp_g);
        %calculating the normal
        normal(x,y,:) = temp_g/norm(temp_g);

        %calculating the gradient in x and y
        p(x,y) = normal(x,y,1)/normal(x,y,3);
        q(x,y) = normal(x,y,2)/normal(x,y,3);
    end
end

%normalizing the albedo
maxalbedo = max(max(albedo));
if( maxalbedo > 0)
    albedo = albedo/maxalbedo;
end

%calculating depth across first column
for i = 2:rows
    depth(i,1) = depth(i-1,1) + q(i,1);
end

%calculating depth across all rows using calculated depth of first column
for i = 2:rows
    for j = 2:cols
        depth(i,j) = depth(i,j-1)+p(i,j);
    end
end

%Albedo Map Colored
```

```
figure;
imagesc(albedo);

%Albedo Map Grayscale
figure;
imagesc(albedo);
colormap(gray);

%Normal Vectors
figure
spacing = 1;
[temp_g ,Y] = meshgrid(1:spacing:rows, 1:spacing:cols);
quiver3(temp_g,Y,-depth, normal(:,:,1),normal(:,:,2),normal(:,:,3))

%Graylevel depth image
figure
surfl(-depth);
colormap(gray);
grid on;
shading interp

%Wireframe of a depth map
figure
spacing = 1;
[temp_g ,Y] = meshgrid(1:spacing:rows, 1:spacing:cols);
quiver3(temp_g,Y,-depth, normal(:,:,1),normal(:,:,2),normal(:,:,3))
hold on
surf( temp_g, Y, -depth, 'EdgeColor', 'none' );
hold off

%Calculating depth using frankotchellappa algorithm
fkDepth = frankotchellappa(p,q);

%Normal Vectors
figure
spacing = 1;
[temp_g ,Y] = meshgrid(1:spacing:rows, 1:spacing:cols);
quiver3(temp_g,Y,-fkDepth, normal(:,:,1),normal(:,:,2),normal(:,:,3))

%Graylevel depth image
figure
surfl(-fkDepth);
colormap(gray);
grid on;
shading interp

%Wireframe of a depth map
figure
spacing = 1;
[temp_g ,Y] = meshgrid(1:spacing:rows, 1:spacing:cols);
quiver3(temp_g,Y,-fkDepth, normal(:,:,1),normal(:,:,2),normal(:,:,3))
hold on
surf( temp_g, Y, -fkDepth, 'EdgeColor', 'none' );
hold off
```