Assignment A2: Photometric Stereo

Clinton Fernandes

u1016390

02/02/2016

A2

CS 6320

1. Introduction

Photometric stereo is a technique for estimating the surface normals of objects by observation (using images) of that object under different lighting conditions. It is thus possible to reconstruct a patch of surface from a series of pictures of that surface. In this assignment we will develop functions to (i) create rendered images under different lighting conditions, (ii) perform the photometric stereo analysis and reconstruct the surface.

A questions that can be answered through this exercise is: What is the relationship between the accuracy of the reconstructed surface properties (rho, N, g, f) and:

a) Number of images used.
b) Distribution of light source locations.
c) Method of Integration i.e., starting from the left corner pixel or the center pixel.

2. Method

CS5320_ps_sphere: generates photometric stereo data for checker sphere

This function will generate photometric stereo data, in this case, for a checker sphere, where the variable size defines the diameter of the sphere. Data is saved in an array Q having dimensions of size x sixe x 7. Q contains x, y, z values of points on the sphere's surface, x, y, z components of the Normal to the surface at the points and the value of albedo.

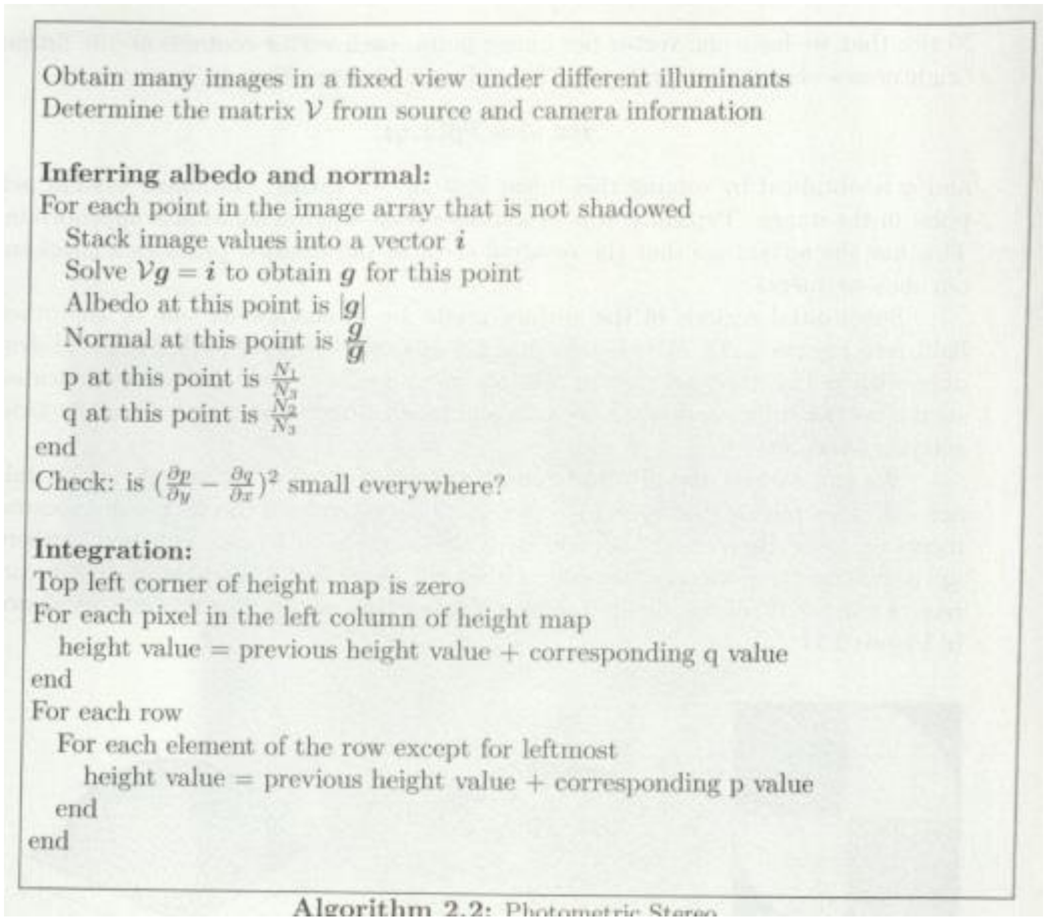
CS5320_ps_render: creates a gray level image of the scene

Given the input (photometric stereo data), this function computes the Intensity at different points on the sphere. Illumination at a point is given by formula. The output is a data representing a rendered image.

This function calls the CS5320_ps_render function a number of times, each time giving a vector for the light source location. For each of the source location, a rendered image is formed. Output of this function is an array of different light source locations and another array of corresponding rendered images.

CS5320_pms: performs the photometric stereo analysis

This function performs the photometric stereo analysis according to the algorithm given below. Input to the function is the set of rendered images for different lighting source directions and the components of light source vector from the surface points. Output is the data of albedo, components of surface Normal, visibility factor and the depth map, at every point (x,y).



```
Obtain many images in a fixed view under different illuminants
Determine the matrix $\mathcal{V}$ from source and camera information

Inferring albedo and normal:
For each point in the image array that is not shadowed
    Stack image values into a vector $\boldsymbol{i}$
    Solve $\mathcal{V}\boldsymbol{g} = \boldsymbol{i}$ to obtain $\boldsymbol{g}$ for this point
    Albedo at this point is $|\boldsymbol{g}|$
    Normal at this point is $\frac{\boldsymbol{g}}{|\boldsymbol{g}|}$
    p at this point is $\frac{N_1}{N_3}$
    q at this point is $\frac{N_2}{N_3}$
end
Check: is $(\frac{\partial p}{\partial y} - \frac{\partial q}{\partial x})^2$ small everywhere?

Integration:
Top left corner of height map is zero
For each pixel in the left column of height map
    height value = previous height value + corresponding q value
end
For each row
    For each element of the row except for leftmost
        height value = previous height value + corresponding p value
    end
end
```

Algorithm 2.2: Photometric Stereo

CS5320_run_test: Run all the functions and also compute statistical date

This function in turn calls all the above functions, and computes the values of albedo, components of surface Normal, visibility factor and the depth map, at every point (x,y). This data of the reconstructed surface is then compared to the Original surface's date (contained in the array Q) and the error is computed. Also other statistical data is found.

3. Verification

CS5320_ps_sphere

Gives a Q(60x60x7) array if the size of the sphere is specified as 60.

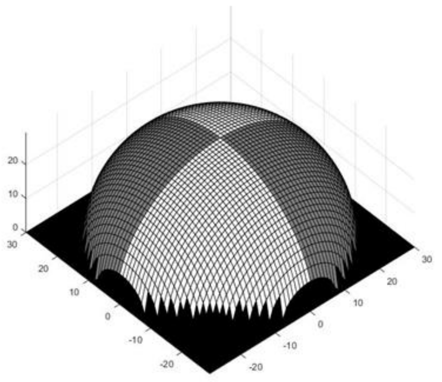


Figure 1: CS5320_ps_sphere

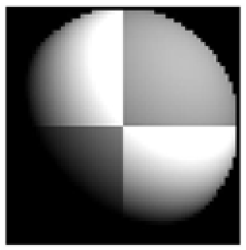


Figure 2: CS5320_ps_render

To verify CS5320_ps_render :

Consider the pixel at (25,25) in Q (60x60x7) array and the source of light be S= (0,0.7,1):

Here, Q (25, 25, :)

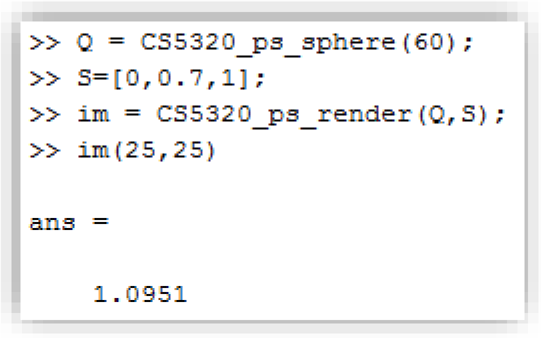
(x, y, z) = (-5.5; 5.5 28.4561) ; N = [-0.1864; 0.1864; 0.9646] ; albedo=1

By hand calculation

I=albedo * (N *S)

$$Intensity = 1 * [-0.1864 \quad 0.1864 \quad 0.9646] * [0 \;\; 0.7 \;\; 1]^T = 1.09508$$

By matlab calculation



```
>> Q = CS5320_ps_sphere(60);
>> S=[0,0.7,1];
>> im = CS5320_ps_render(Q,S);
>> im(25,25)

ans =

    1.0951
```

CS5320_createImages: Creates images from different lighting conditions.

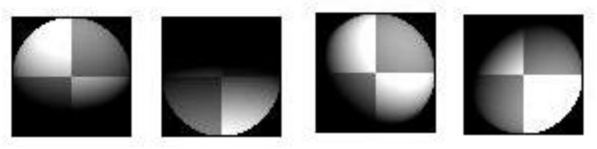
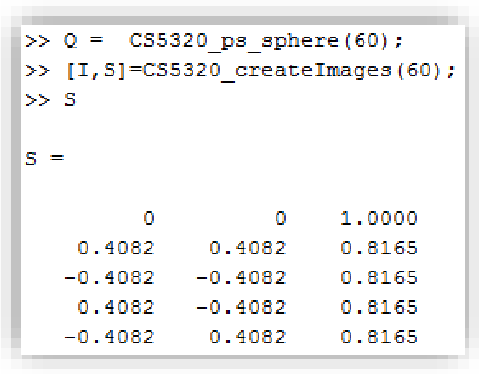


Figure 3: CS5320_createImages (for different light directions)

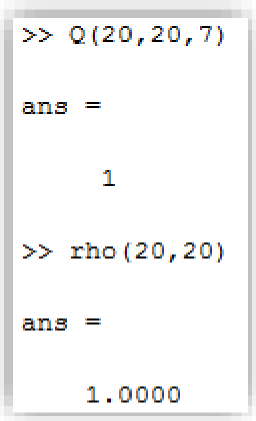
To verify CS5320_pms(I,S)

This can be done by comparing [rho, N, g, f] in the original Q to what pms produces.



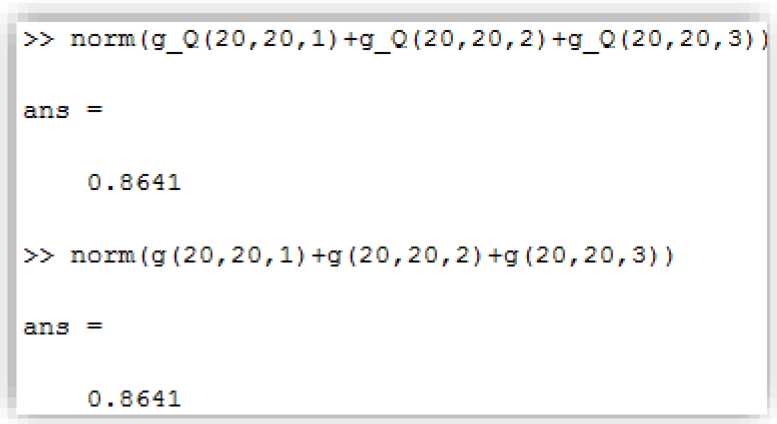
```
>> Q =  CS5320_ps_sphere(60);
>> [I,S]=CS5320_createImages(60);
>> S

S =

         0         0    1.0000
    0.4082    0.4082    0.8165
   -0.4082   -0.4082    0.8165
    0.4082   -0.4082    0.8165
   -0.4082    0.4082    0.8165
```

We can compare a pixel at (20,20)



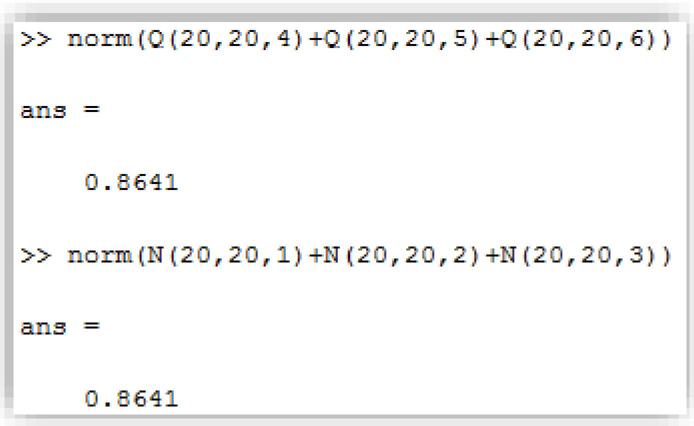
```
>> Q(20,20,7)

ans =

     1

>> rho(20,20)

ans =

    1.0000
```

Albedo



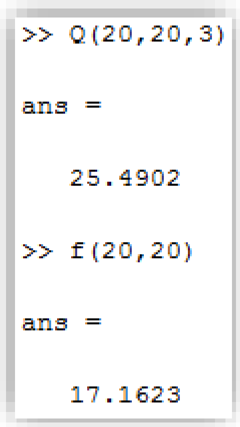
```
>> norm(g_Q(20,20,1)+g_Q(20,20,2)+g_Q(20,20,3))

ans =

    0.8641

>> norm(g(20,20,1)+g(20,20,2)+g(20,20,3))

ans =

    0.8641
```

g (surface property) ; g is the value obtained from pms



```
>> norm(Q(20,20,4)+Q(20,20,5)+Q(20,20,6))

ans =

    0.8641

>> norm(N(20,20,1)+N(20,20,2)+N(20,20,3))

ans =

    0.8641
```

Normal



```
>> Q(20,20,3)

ans =

   25.4902

>> f(20,20)

ans =

   17.1623
```

depth map

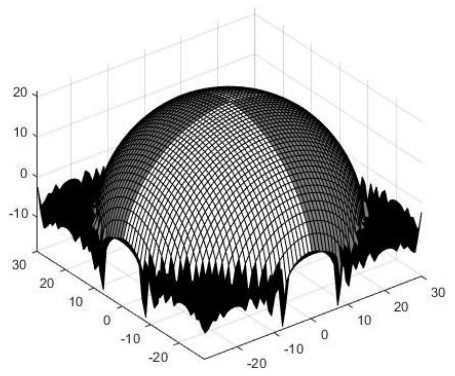
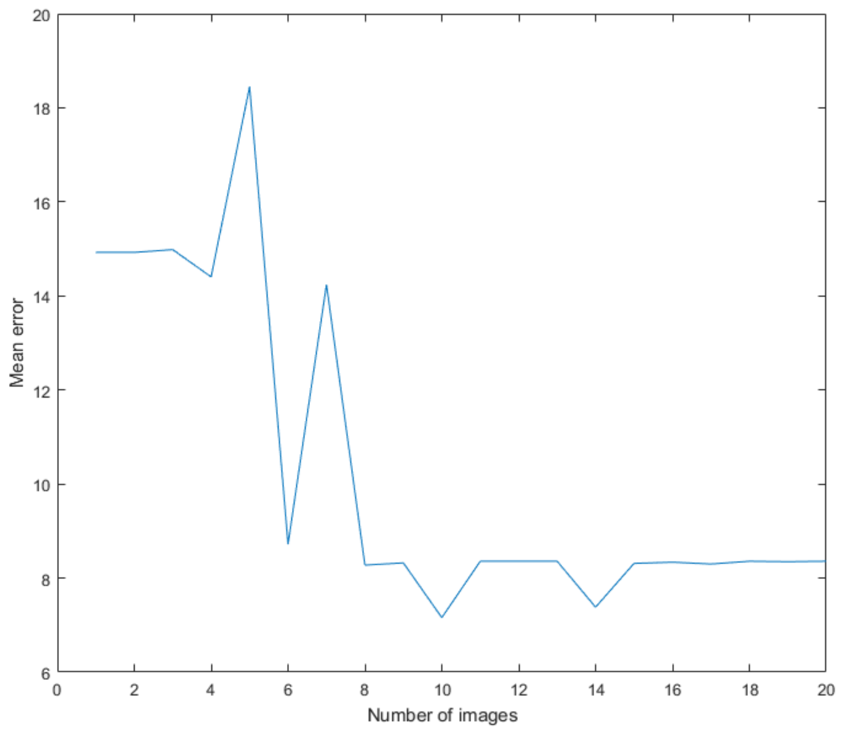
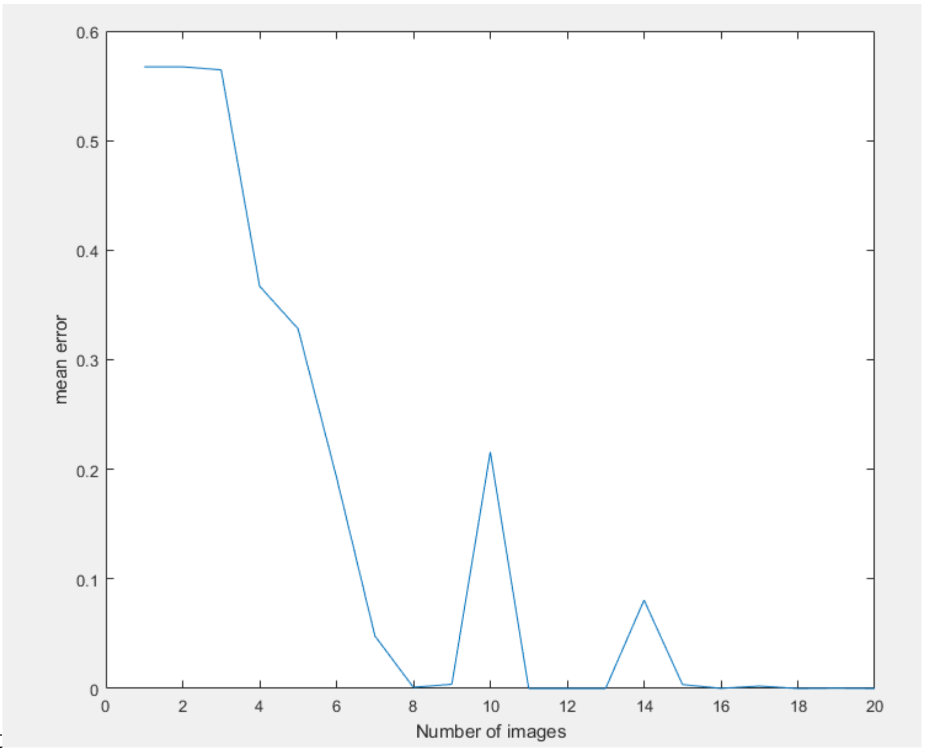


Figure 3: CS5320_pms

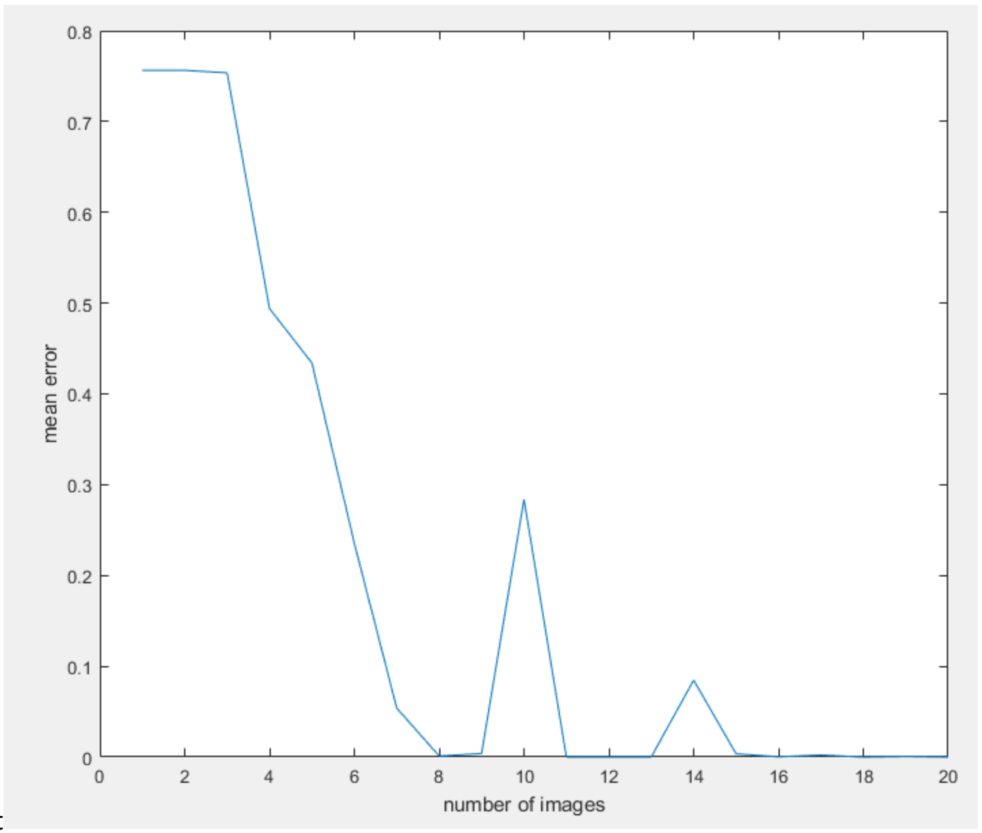
4. Data



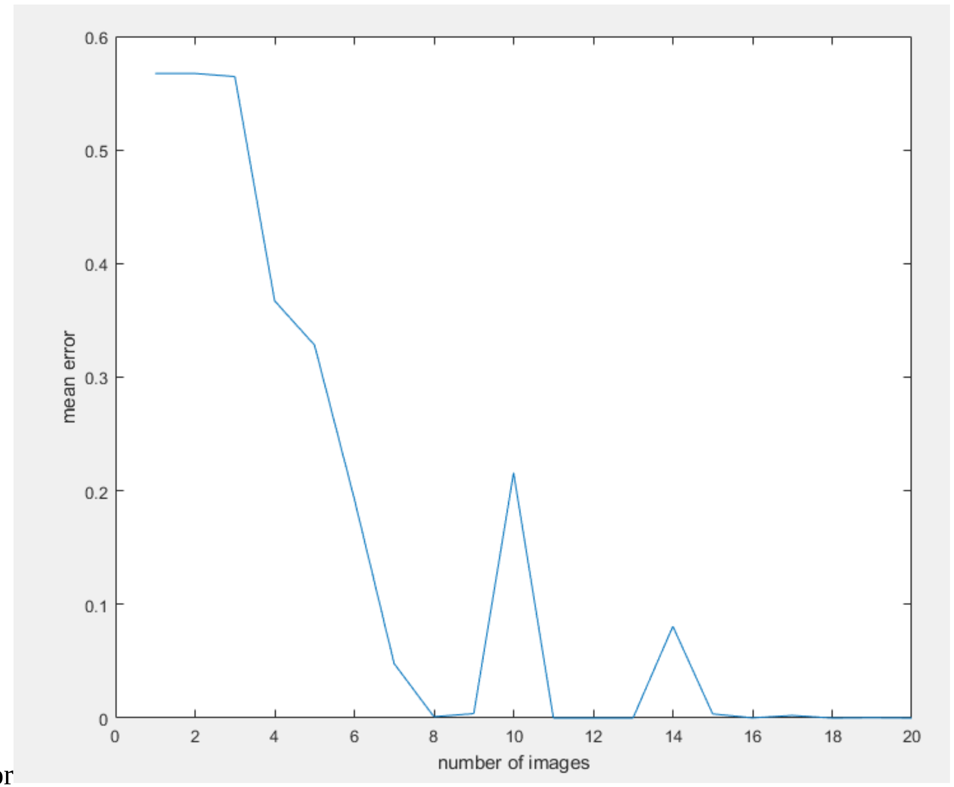
Depthmap error plot



Albedo error plot



Normal error plot



Visibility factor

5. Analysis

For 5 sources :

S1 = [0,00.6,0.8];

S2 = [0.4,0.4,0.3];

S3 = [-0.4,-0.4,0.5];

S4 = [0.4,-0.4,0.7];

S5 = [-0.4,0.4,0.1];

The error data was found to be:

parameter	mean	variance	Confidence	intervals
f	8.056128	5.566086	7.990856	8.121401
rho	0.044571	0.034193	0.039455	0.049687
N	0.061372	0.057621	0.054731	0.068013
g	0.044571	0.034193	0.039455	0.049687

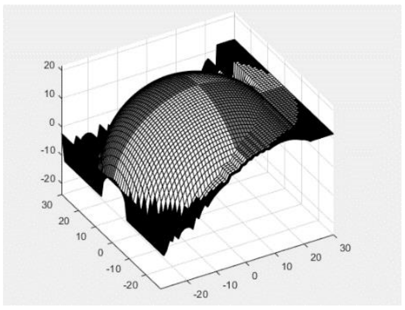
6. Interpretation

It was found that in any given case, the error in the depth map was comparatively larger than in any other parameters. This is because the depth map function may not have had been defined properly.

It was however observed that, as the increase number of images was increased, the mean errors in the parameters would reduce and would almost remain constant after a certain number of images.

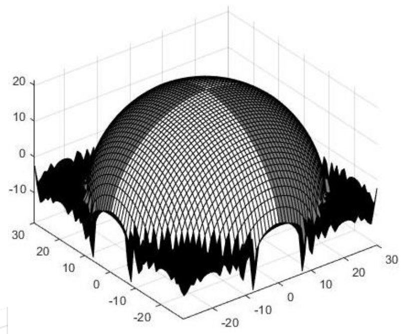
Also, it can be seen that for a particular number of images, say 5, the accuracy of the rendered surface improves if the light sources are distributed equally around the object. i.e., if light falls on the object in an uniform distributed manner.

If the light sources were kept along one side, then there would be shadow region on the other side and hence, the surface was not rendered properly.



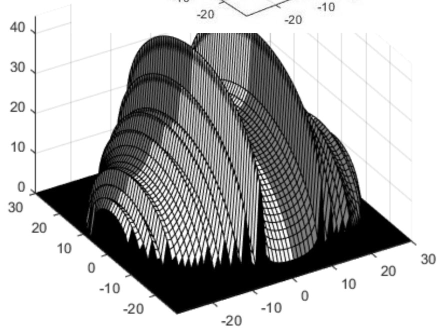
7. Critique

The depth map function can be defined in some other form. In its present form, a pixel in the center is selected, and then traversed in the top and bottom direction along the middle column with an increment of q (df/dy) and then incremented along left and right of each row by p (df/dx). The height is taken as maximum value of p or q.



In a different way, we can start from the left corner pixel. Set this pixel value to 0. Make each succeeding row of the first column equal to previous value + q. Now for each row, starting from second column, set the value of succeeding column pixel equal to previous value + p.

However, a step like surface is obtained.



In both the above methods, the maximum height of depth map was found to be about 21, even though original sphere had a height of 30.

The depth function can also be defined by taking the **f= previous value + sqrt(p+q)** as we go to a new pixel

8. Log

Time spent:

Coding/De-bugging = 25 hrs

Report = 3 hrs