HomeWork 3
Computer Vision and Scene Analysis

By:
Monil Shah
mds747

1) Theoretical Part

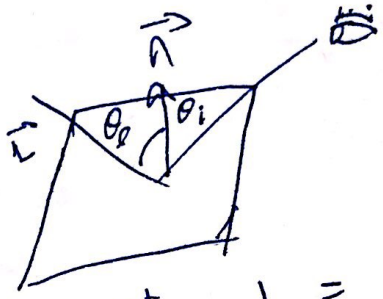
Assignment-3

1.1.1 Reflectance Map

⇒ The reflectance map in the camera can be given as

$R(p,q) = \rho(\hat{N} \cdot \hat{L})$

According to theory of Lambertian surface



we know that, $L = \rho \frac{\cos\theta_i}{\cos\theta_e}$

$\Rightarrow L = \rho \cos\theta_i$

In our case $\vec{L} = (0, 0, 1)$

Now,
A general surface can be expressed as a 2D height function, where the height is function of location on plan such as:

$$f(x,y) \to S = f(x,y) = z(x,y) \qquad \mathbb{R}^2 \to \mathbb{R}$$

∴ Surface Element vector

$$S = \begin{pmatrix} x \\ y \\ z(x,y) \end{pmatrix}$$

Computing, partial derivative of z, acc. to x & y.

$$\frac{\partial S}{\partial x} = \begin{pmatrix} 1 \\ 0 \\ \frac{\partial z(x,y)}{\partial x} \end{pmatrix} \qquad \frac{\partial S}{\partial y} = \begin{pmatrix} 0 \\ 1 \\ \frac{\partial z(x,y)}{\partial y} \end{pmatrix}$$

Defining, p & q,

$$p = \frac{\partial z(x,y)}{\partial x} \qquad q = \frac{\partial z(x,y)}{\partial y}$$

So, $\frac{\partial S}{\partial x} = \begin{pmatrix} 1 \\ 0 \\ p \end{pmatrix}$ $\frac{\partial S}{\partial y} = \begin{pmatrix} 0 \\ 1 \\ q \end{pmatrix}$

Normal is always orthogonal to surface

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

Normal vector in unit norm

$$\hat{n} = \frac{\vec{n}}{\sqrt{p^2+q^2+1}} \quad - (1)$$

So, normal vector looking down for our case, we get

$$\boxed{x^2+y^2+z^2=1} \quad \text{(sphere equation)}$$

$$\therefore z = \sqrt{1-x^2-y^2}$$

$$\Rightarrow \hat{n} = \sqrt{1-x^2-y^2}$$

So, we can know reflectance map

$$\boxed{R(p,q) = S\sqrt{1-x^2-y^2}}$$

Q-2 (1.1.2) General light source direction.

In our case, we have a one light source which has its own normal vector.

From above question

$$\hat{\vec{n}}_s = \frac{\vec{n}_s}{\sqrt{p_s^2 + q_s^2 + 1}} \quad \text{with } n_s = \begin{pmatrix} -p_s \\ -q_s \\ 1 \end{pmatrix}$$

$$\therefore L = S(\cos\theta_i)$$

$$= S\langle \vec{n}_s, \vec{n} \rangle$$

$$= S \frac{\vec{n}_s \vec{n}}{\sqrt{p_s^2 + q_s^2 + 1}\sqrt{p^2 + q^2 + 1}}$$

$$L = \frac{pp_s + qq_s + 1}{\sqrt{p_s^2 + q_s^2 + 1}\sqrt{p^2 + q^2 + 1}}$$

=> Modifying above equation we can get general equation for conic curve.

Hence, set of isophote curves associ to specific light source in our case are forming conic curve.

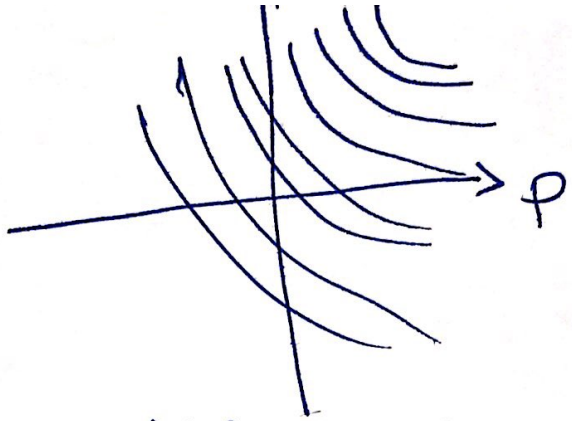


Fig. is ophote curves set associated to a specific light source.

Q-3 (1.1.3) Number of images for reconstruction of normals.

⇒ We now have a map of curves defined with our two unknowns p & q. But given a curve we can never get the associated values of p & q because there is a infinite set.

of possible solutions, as we deal with continuous curve. We need more information.

So, if we take a second picture we can get set of isophote curves associated to two specific light sources.

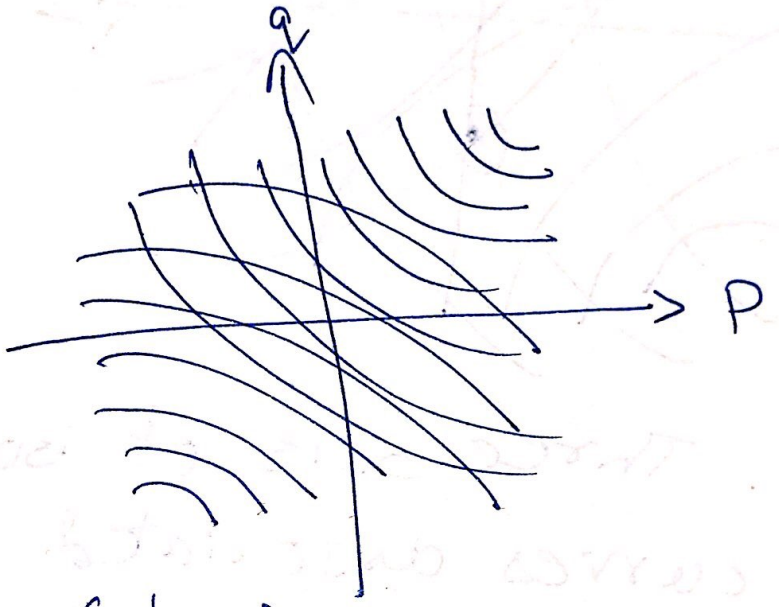


Fig. Set of isophote curves for two specific light sources.

This is interesting, but not enough as we reduce a large number of possible values to

a set of two possible paths
so, if we take a third image

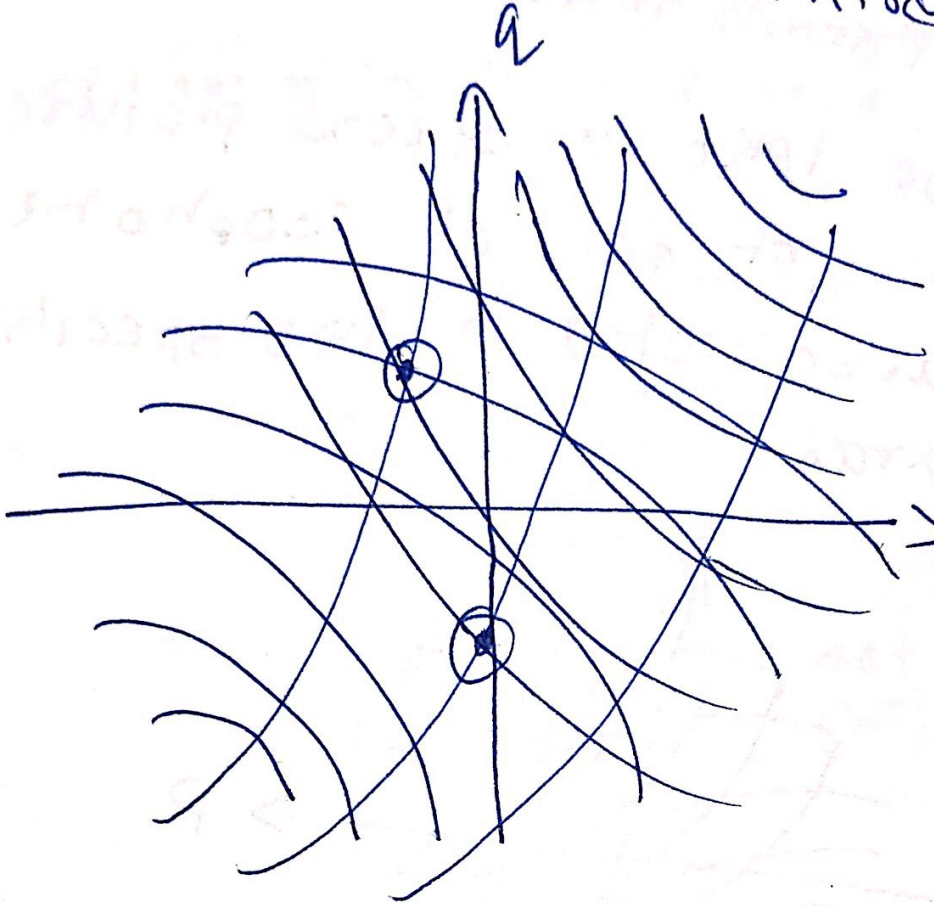


Fig. Three sets of isophot curves associated to three specific light sources. Hence.

$$\begin{bmatrix} I_1 \\ I_2 \\ I_3 \end{bmatrix} = \begin{bmatrix} S_1^T \\ S_2^T \\ S_3^T \end{bmatrix} \rho N$$

$$\Rightarrow \rho N = S^{-1} I$$

$\therefore$ For 3 images

$$\vec{N}(x,y) = \frac{\bar{g}(x,y)}{\rho(x,y)}$$

Hence, we need three images.

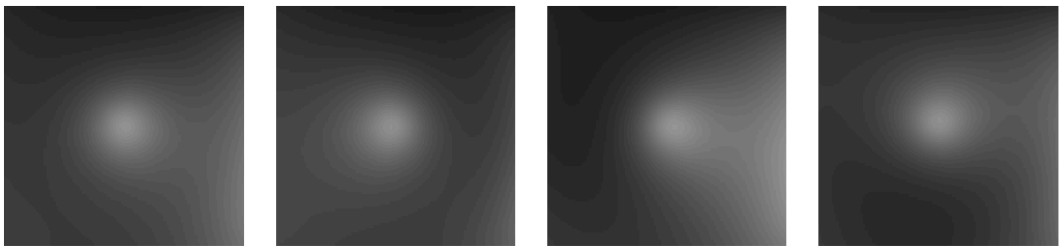
Practical Part:

1) Shape from Shading

For the first part of experiment , we take four synthetic images . The four images in the dataset are as below:

As we can see the images are taken with same camera position and different light source, we need to calculate the albedo map of the image first. As from the images it is visible we are dealing with the images that are bright in the center and darker in the surrounding area. The albedo is calculated of the light source coordinates which are provided to us.

DataSet 1:



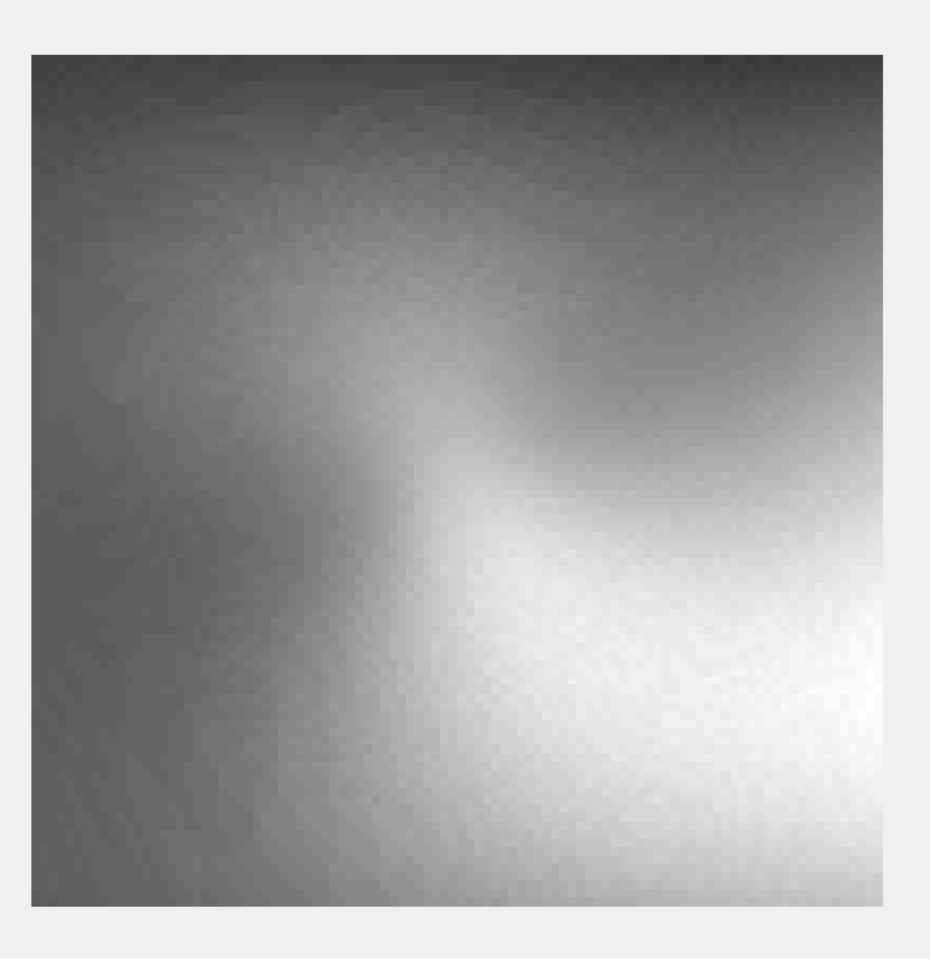
im1.png im2.png im3.png im4.png

L = [0,0,1;-0.2,0,1;0.2,0,1;0,-0.2,1]

Analyzing the albedo map we can determine that the albedo value of the part not in center is lower while that in the center is darker. As we further analyze we can see that which image is brighter than the other. As we can see from image below albedo map seems to be decent and the second step to get surface normals should be performed.

I) Albedo Map

The albedo is calculated using the formula where albedo = norm(inv(LigthSource' * LightSource) * (LightSource' * Image'))

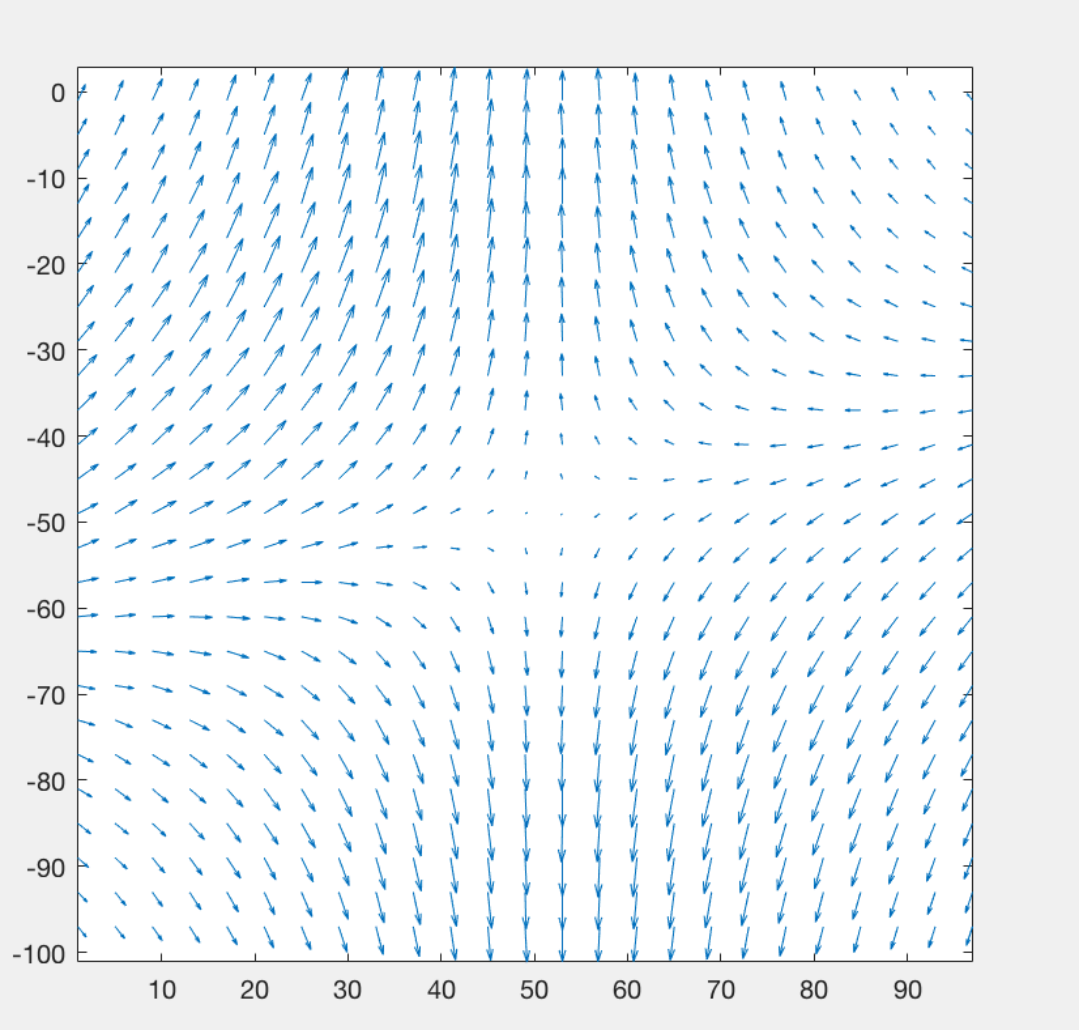


II) Surface Normal Vector Map

Below is the map of surface normals , the surface normals are represented in the 2D space here. As we analyze the results we can see that the center area of the images is brighter and corresponds towards the summit of the shape. The normals are clearly oriented towards the light source. Also we can infer that the z direction of the figure is largest compared to x and y components, so the 3D image reconstructed must have a larger z components. Lets see the results of the 3D reconstruction.

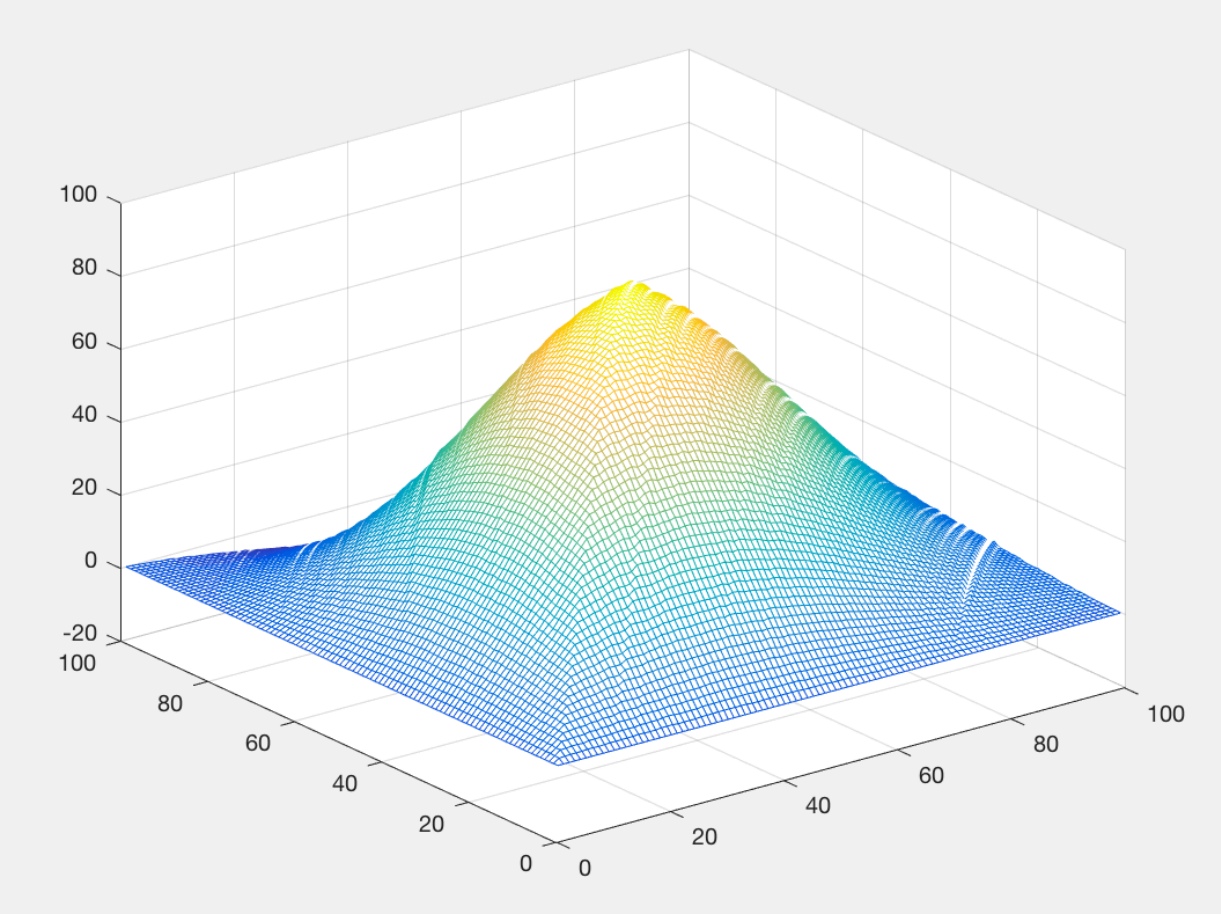
The surface normal is implemented by finding reflectance space p = normal1/normal3 and q= normal2/normal3.

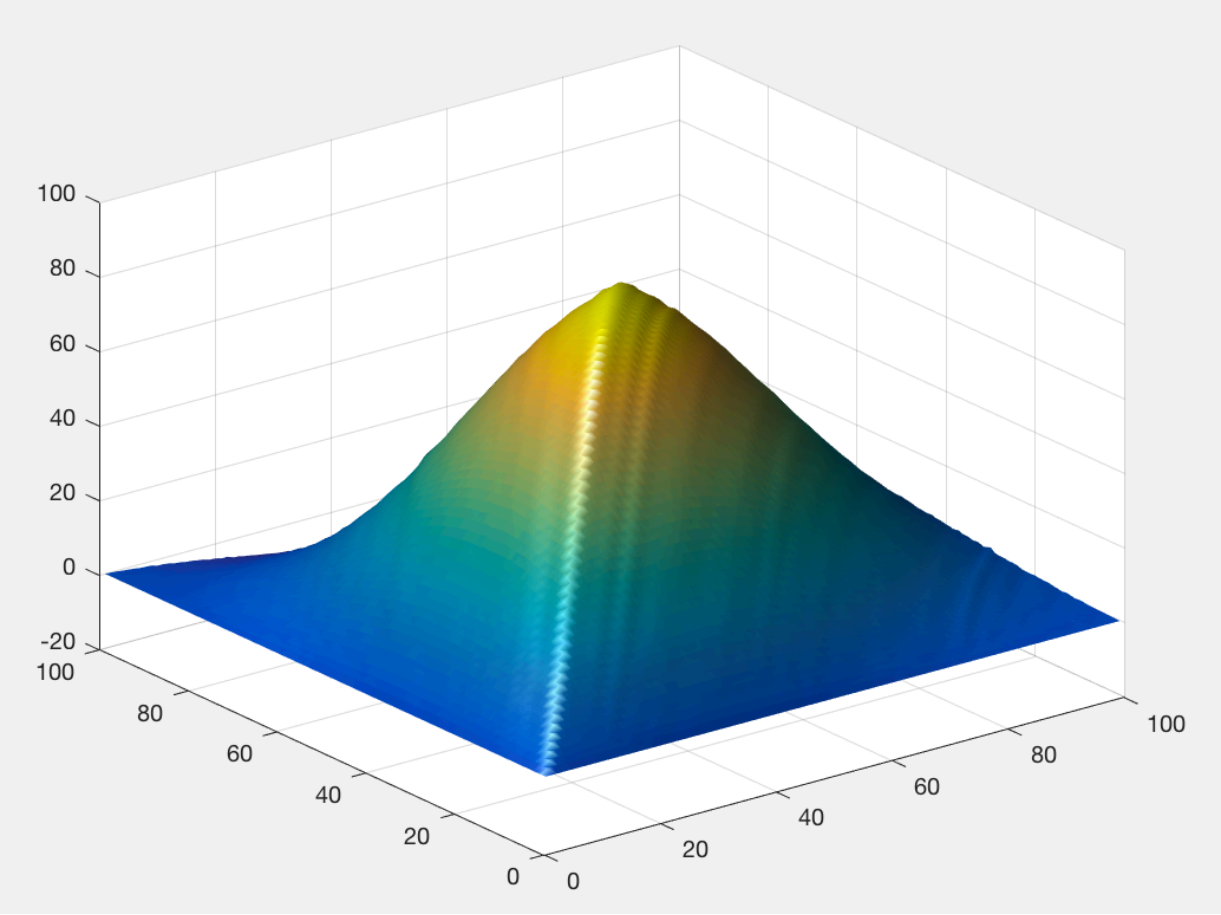
.



III) 3d Reconstruction

The 3D reconstruction results are as below, one is the reconstruction mesh on basis of the surface normals, while other is the final reconstructed shape after integration. So according to the depth discussion in the previous results we can say that the brightness increases in the center thus corresponding to the darker color on the summit of the 3D reconstructed image. Hence, we can decide the final image reconstructed.
The mesh and surf of the images are calculated using depth by formula depth = p + q which is simple integration.

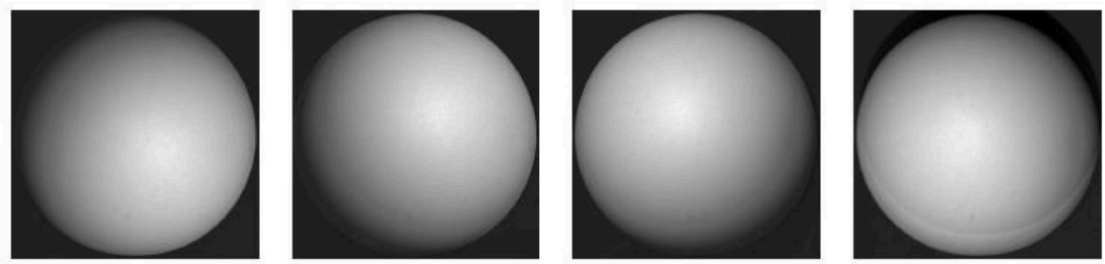

100
80
60
40
20
0
-20
100
80
60
40
20
0
0
20
40
60
80
100



2) Shape from shading from real images

We can perform our analysis on an other artificial data set to see how our algorithm is reconstructing the shape. This data set is a sphere so by knowing exactly what shape we should reconstruct we will be able to see the limitations of our algorithm.

DataSet 2:

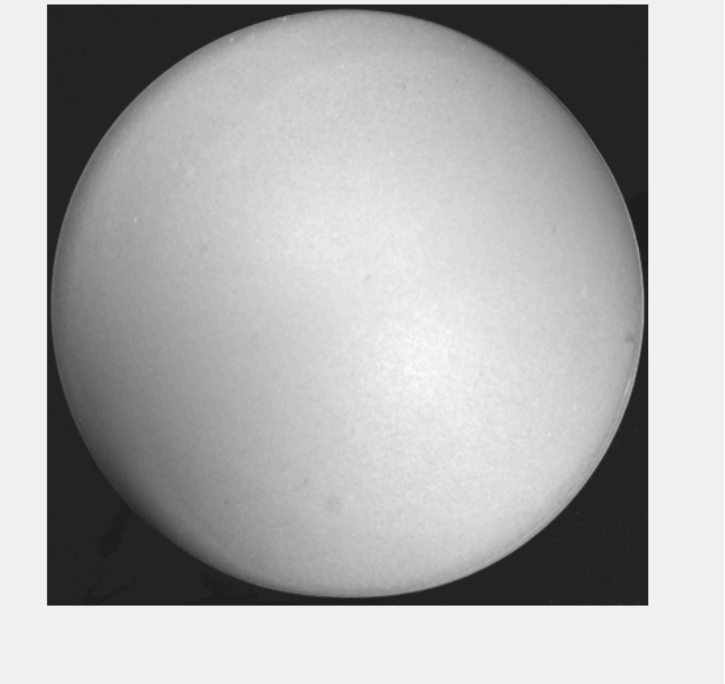


real1.bmp real2.bmp real3.bmp real4.bmp

```
L=[0.38359,0.236647,0.89266;0.372825,-0.303914,0.87672;
-0.250814,-0.34752,0.903505;-0.203844,0.096308,0.974255
]
```

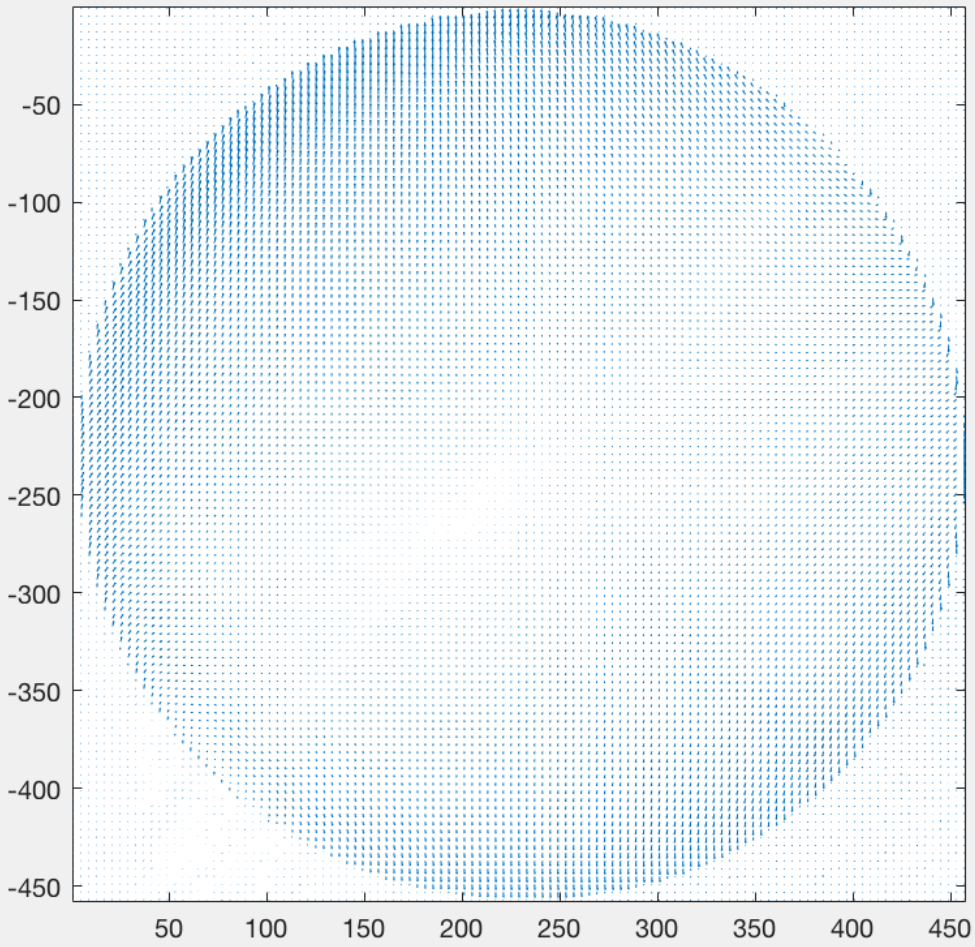
I) Albedo Map

The albedo map of the above dataset with the mentioned light sources is as given below. As mentioned earlier the same approach is used for albedo estimation. As we can see from the image albedo map is proper according to the images provided.



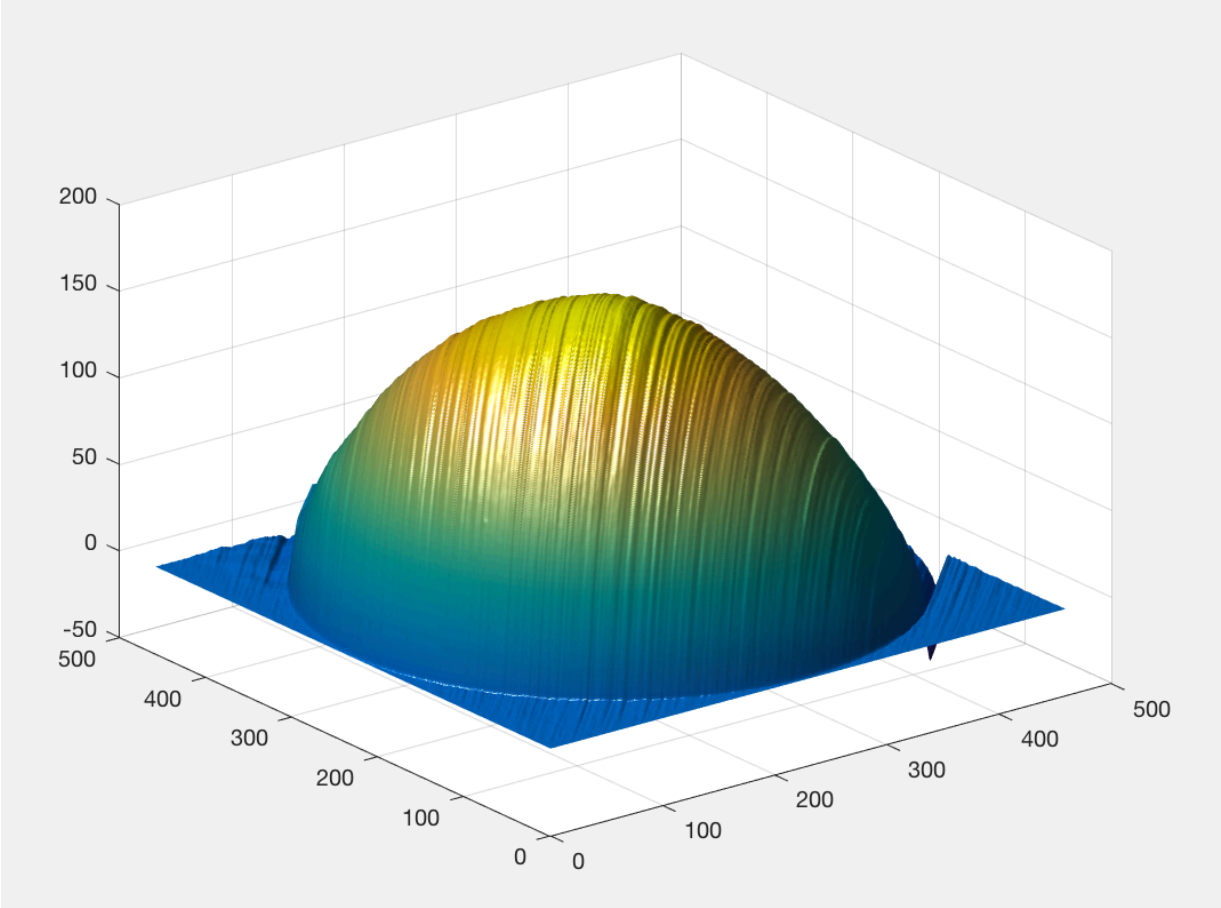
II) Normal Vector Map

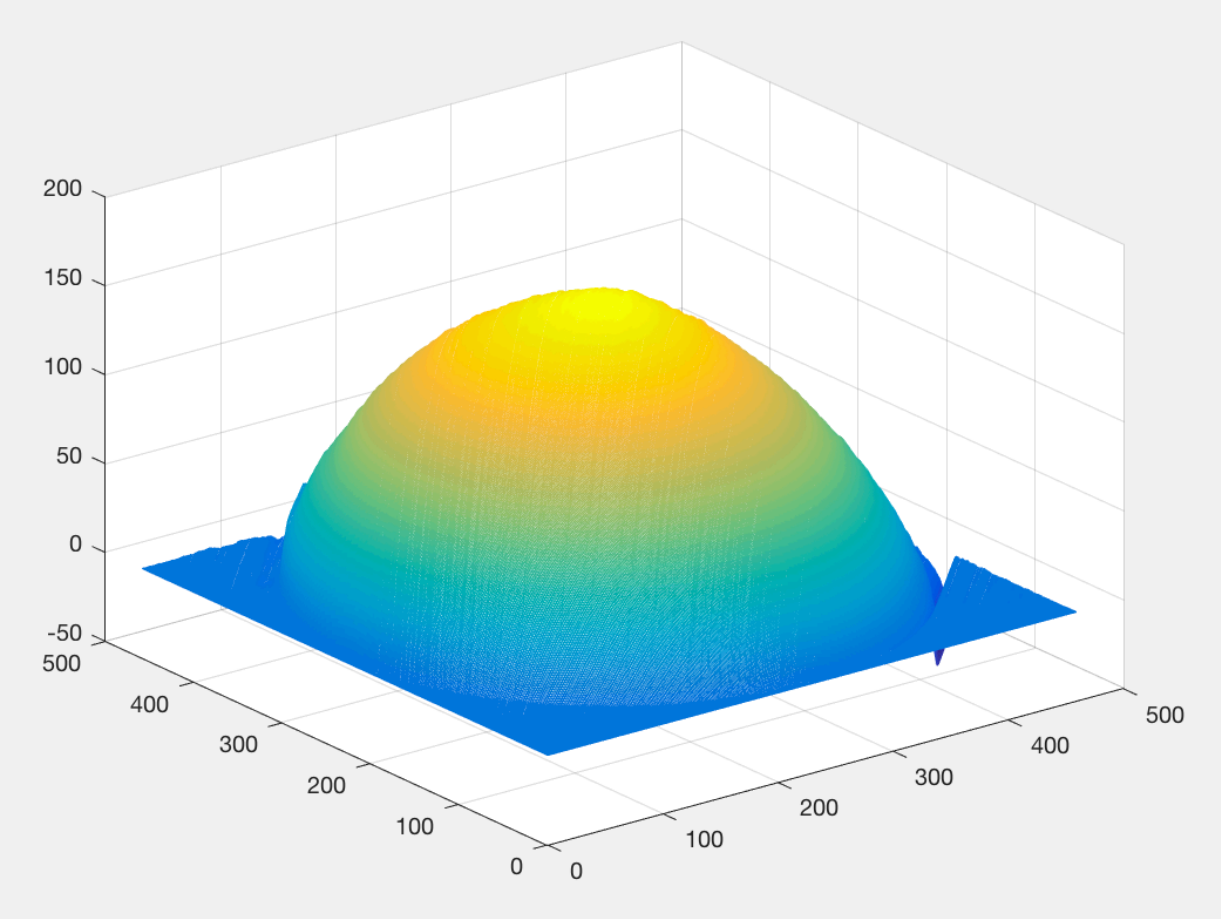
As we can see here, the shape appears to be quite good, but some issues also appears with the normal vector estimation. Also the implementation Approach used is same as for above dataset.



III) 3D Reconstruction from real images

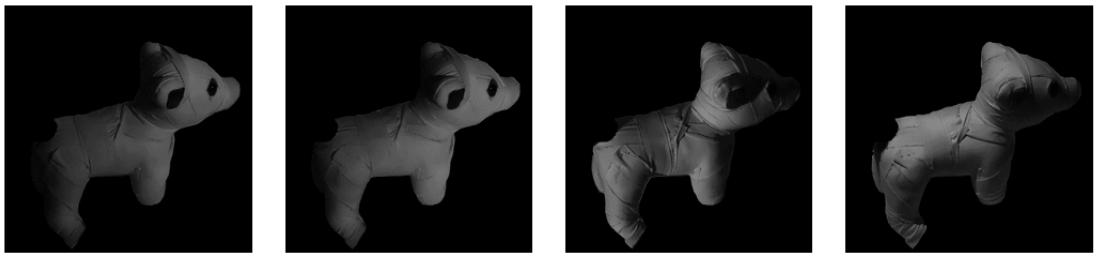
As we can see the image reconstructed is not upto the mark, the some of the normal vector estimation has gone wrong forming irregularity in the shape. Overall a good solution of reconstructed shape is possible with the help of simple integration method.


200
150
100
50
0
-50
500
400
300
200
100
0
0
100
200
300
400
500



3) Shape from shading from real images

```
L=[16,19,30;13,16,30;-17,10.5,26.5;9,-25,4]
DataSet 3:
```



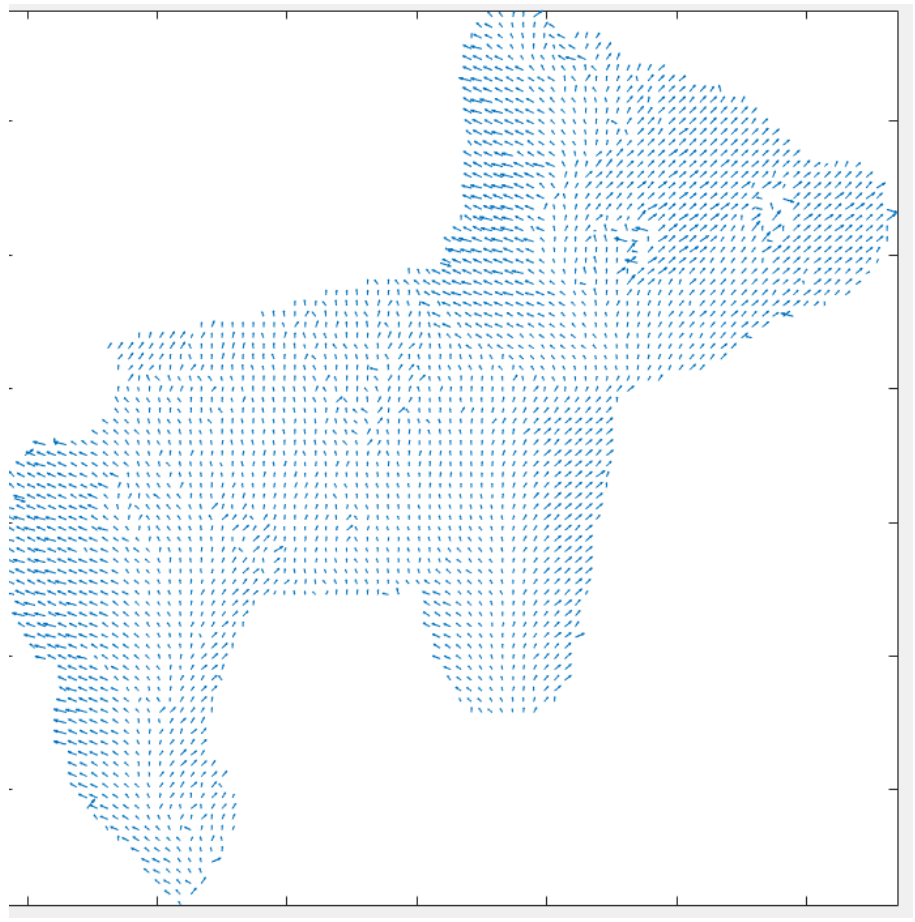
dog1.tif dog2.tif dog3.tif dog4.tif

I) Albedo Map

The albedo map of the provided dataset is as below. The implementation method is same as mentioned above.

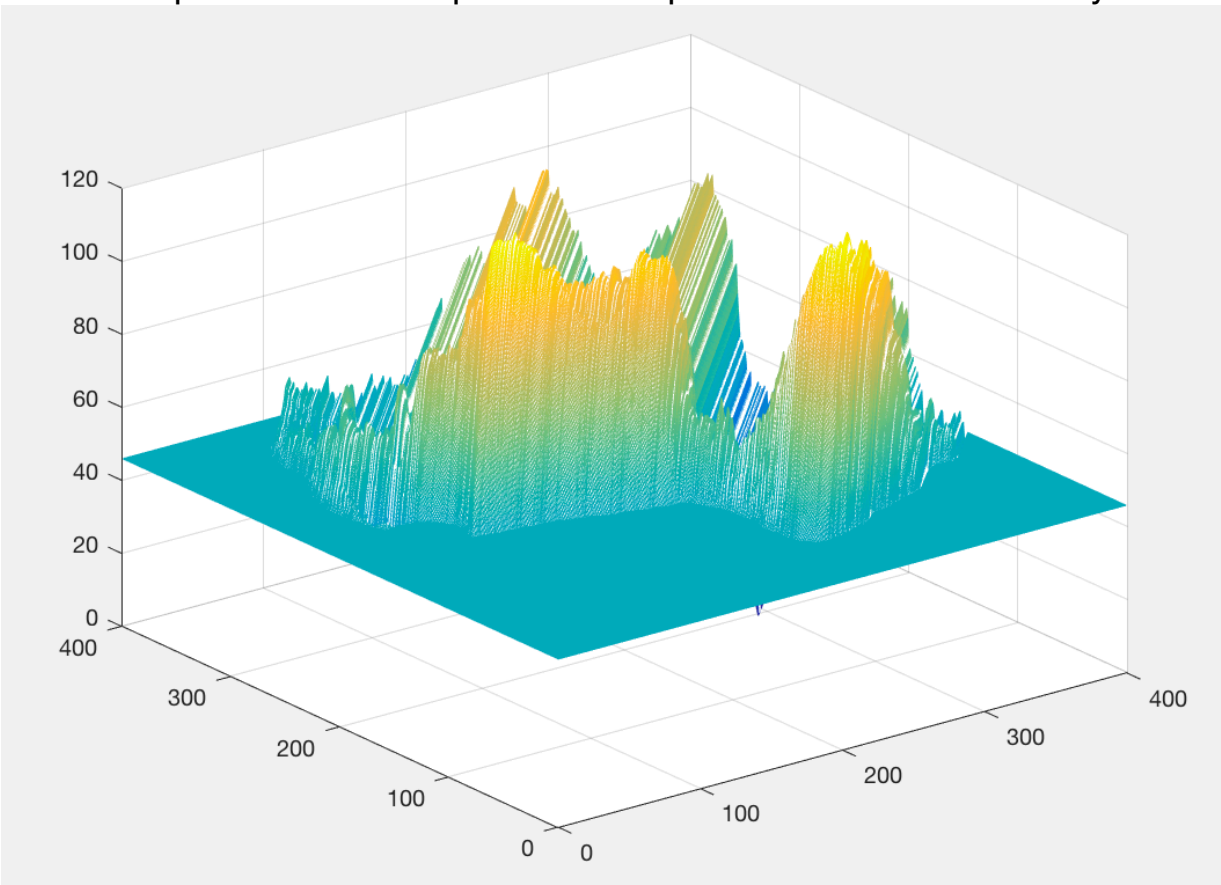
II) Normal Vector Map

As shown in figure below the normal vector representation is not too bad considering the method implemented. Also this normal vector representation can be made better with better implemented algorithm.

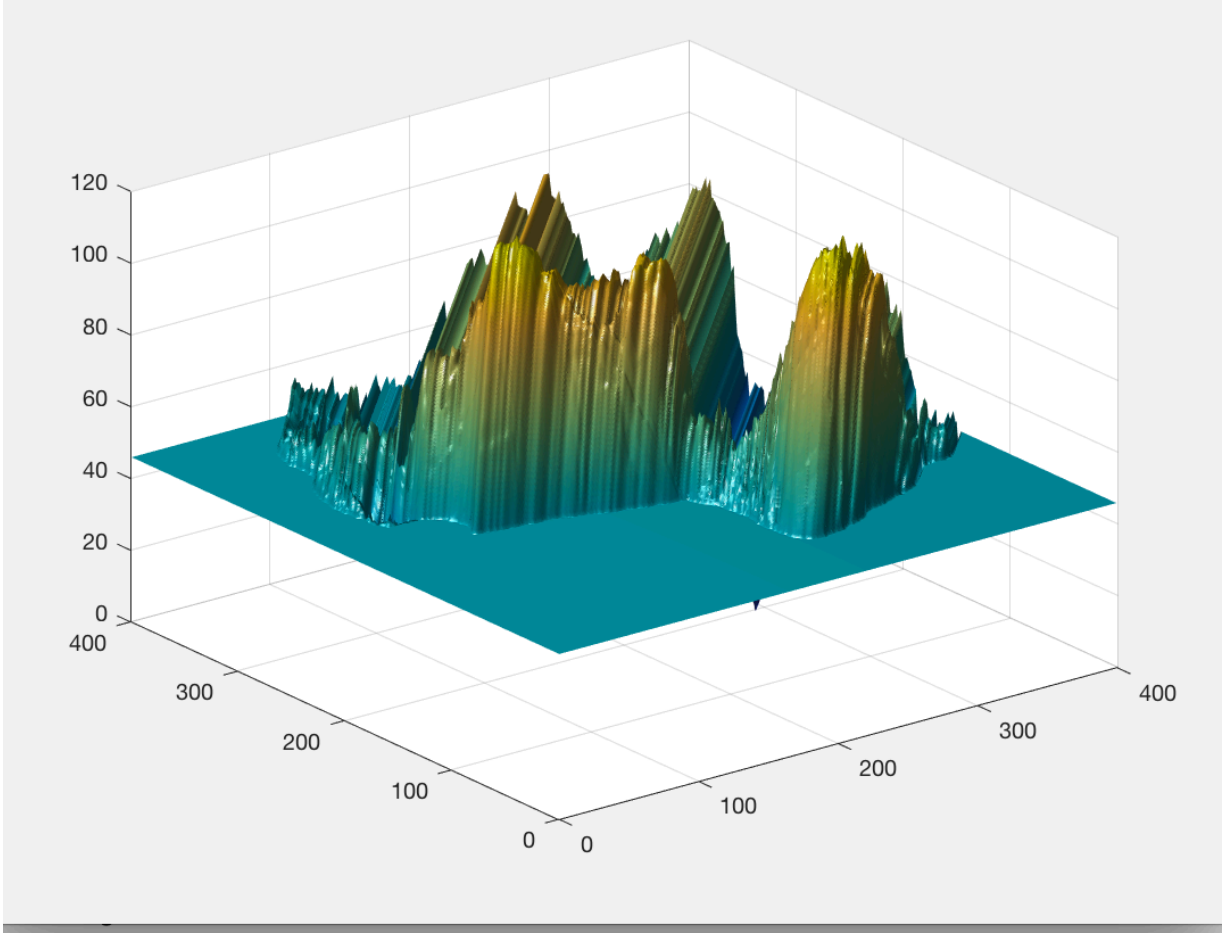


III) 3D reconstruction

With this data set we can clearly see the limitations of our implementation which fail in reconstructing a correct solution for the shape of the object. We obtain peaks and some pieces of shape but the results are really bad.



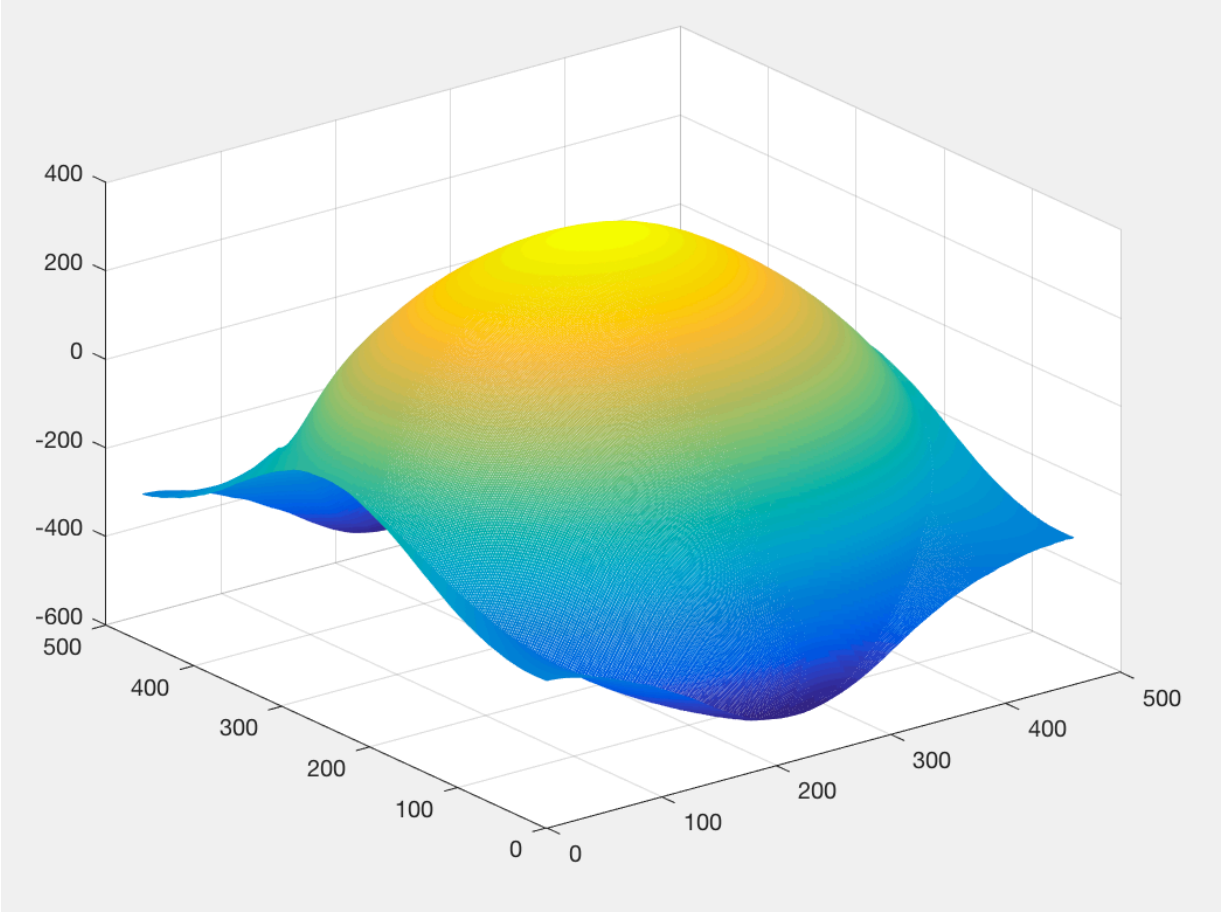
.

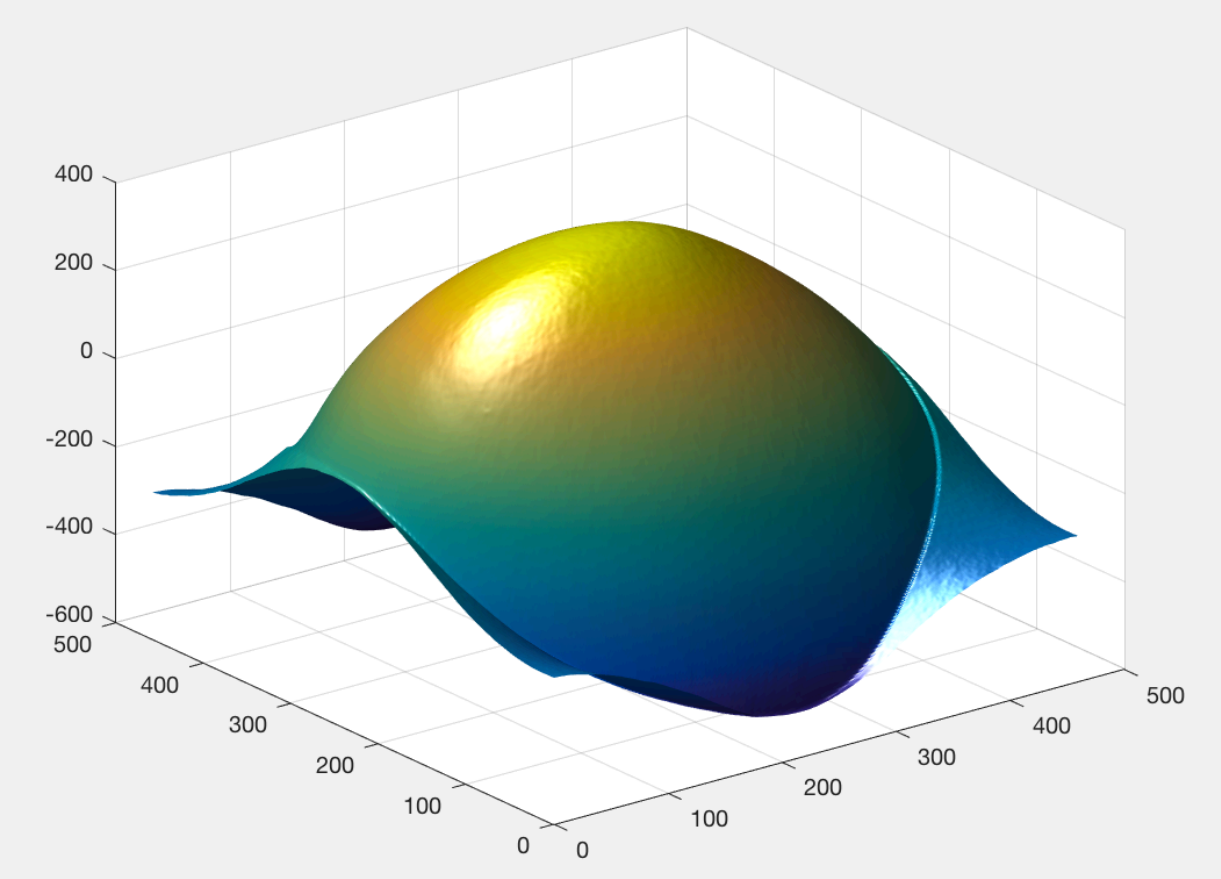


4) Bonus Homework Chapella Method:

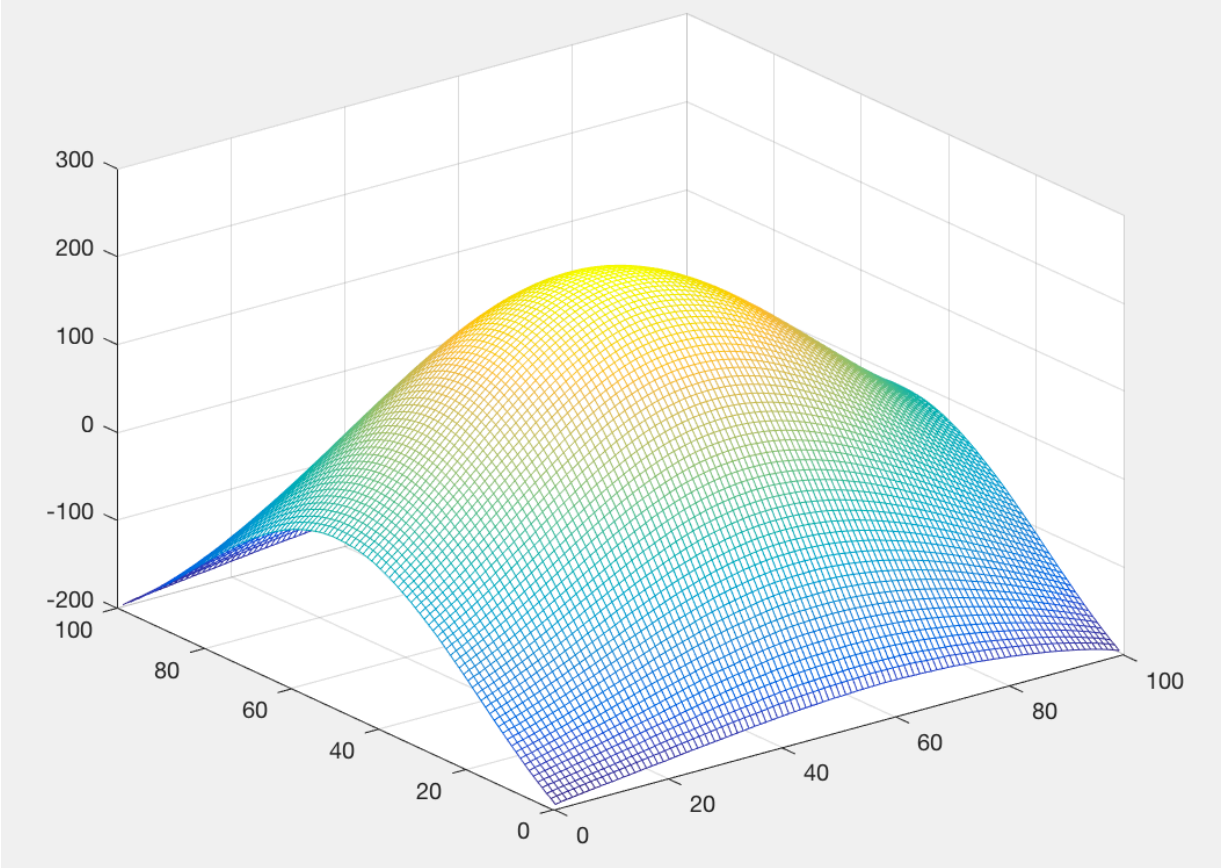
Reference : http://www.peterkovesi.com/matlabfns/
FrankotChapella:
In this section we are implementing the Frankot Chapella algorithm to reconstruct the results from our data set. In this method we solve are solving the system of equations to reconstruct the images.

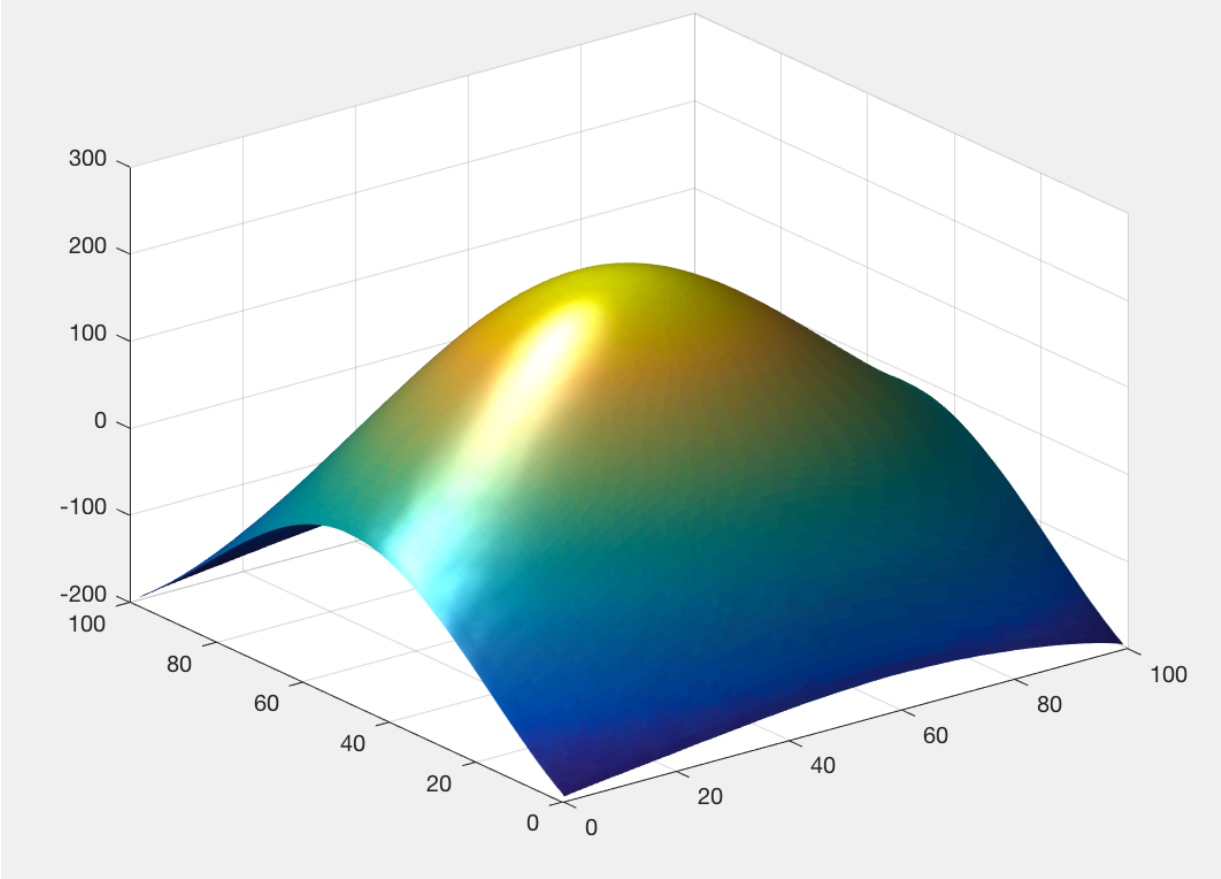
For the dataset of Sphere images, results are below :


400
200
0
-200
-400
-600
500
400
300
200
100
0
0
100
200
300
400
500



For the dataset of Synthetic dataset images , the results are as below:


300
200
100
0
-100
-200
100
80
60
40
20
0
0
20
40
60
80
100



Implementation:

```
I1=imread('im1.png');
I2=imread('im2.png');
I3=imread('im3.png');
I4=imread('im4.png');
images = zeros(size(I1,1),size(I1,2));
for i = 1:size(I1,1)
for j = 1:size(I1,2)
images(i,j,1) = I1(i,j);
images(i,j,2) = I2(i,j);
images(i,j,3) = I3(i,j);
images(i,j,4) = I4(i,j);
end
end
% Initializations
albedo=zeros(size(I1,1),size(I1,2));
p = zeros(size(I1,1),size(I1,2));
```

```
q = zeros(size(I1,1),size(I1,2));
normal_vector = zeros(size(I1,1), size(I1,2),1);
for i = 1:size(I1,1)
for j = 1:size(I1,1)
normal_vector(i,j,1) = 0;
normal_vector(i,j,2) = 0;
normal_vector(i,j,3) = 0;
end
end
L=[0,0,1;0.2,0,1;-0.2,0,1;0,0.2,1]; %position of light source
%L=[0,0,1;-0.2,0,1;0.2,0,1;0,-0.2,1]
%L=[-16,-19,30;-13,-16,30;17,-10.5,26.5;-9,25,4];
%L=[-0.38359,-0.236647,0.89266;-0.372825,0.303914,0.87672;0.250814,0
.34752,0.903505;0.203844,-0.096308,0.974255]
%L=[-16,-19,30;-13,-16,30;17,-10.5,26.5;-9,25,4];
normal=[0;0;0];
% processing
for i = 1:size(I1,1)
for j = 1:size(I1,2)
I(1) = double(images(i,j,1));
I(2) = double(images(i,j,1));
I(3) = double(images(i,j,3));
I(4) = double(images(i,j,4));
A = L'*L;
b = L'*I';
g = inv(A)*b;
albedo(i,j) = norm(g);
normal = g/albedo(i,j);
normal_vector(i,j,1) = normal(1);
normal_vector(i,j,2) = normal(2);
normal_vector(i,j,3) = normal(3);
p(i,j) = normal(1)/normal(3);
q(i,j) = normal(2)/normal(3);
end
end


maxalbedo = max(max(albedo) );
```

```
if( maxalbedo > 0)
albedo = albedo/maxalbedo;
end

depth=zeros(size(I1,1));
for i = 2:size(I1,1)
for j = 2:size(I1,2)
depth(i,j) = depth(i-1,j-1)+q(i,j)+p(i,j);
end
end
%[offset,indice] = min(depth(:));
%depth = depth-offset;
%depth = frankotchepIla(p,q);
%[offset,indice] = min(depth(:));
%depth = depth-offset;


[ X, Y ] = meshgrid( 1:size(I1,1), 1:size(I1,2) );
figure(2);
surf( X, Y, depth,'EdgeColor','none');
camlight left;
lighting phong
figure(3);
mesh( X, Y, depth);
figure(4);
spacing = 4;
[x,y] = meshgrid(1:spacing:size(I1,1), 1:spacing:size(I1,2));
quiver(x,-y,p(1:spacing:end, 1:spacing:end),q(1:spacing:end,
1:spacing:end));
axis tight;
axis square;
figure(1);
imshow(albedo);
%display estimated surface
%figure(1);
%surfl(depth);
%colormap(gray);
%grid off;
```

```
%shading interp
```