NYU CS-GY 6643, Computer Vision
Assignment 3 (Practical Problems)

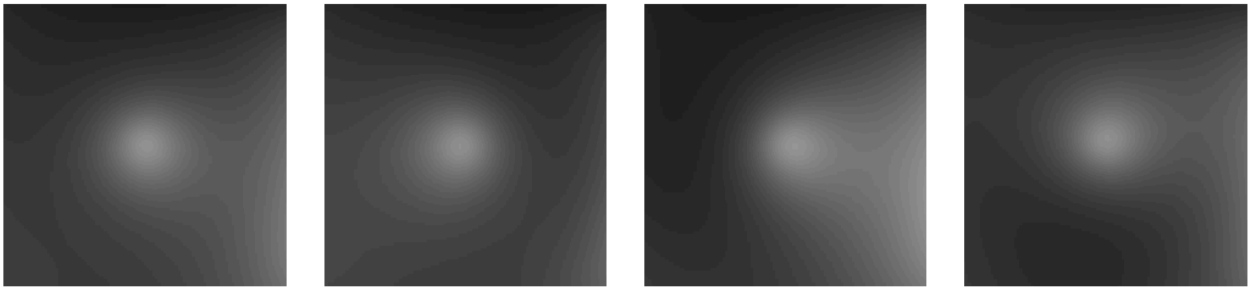
Name: Amitesh Kumar Sah
NYU ID: N19714360

2.1 Problem: Photometric Stereo: Synthetic Images

Objective:
The goal of this assignment is to implement an algorithm that reconstructs a surface using the concept of photometric stereo.

Assumption:
Here, we assumed a Lambertian reflectance function, but the albedo is unknown and non constant in the images.

Data set along with light direction



im1.png $\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$ im2.png $\begin{bmatrix} -0.2 \\ 0 \\ 1 \end{bmatrix}$ im3.png $\begin{bmatrix} 0.2 \\ 0 \\ 1 \end{bmatrix}$ im4.png $\begin{bmatrix} 0 \\ -0.2 \\ 1 \end{bmatrix}$

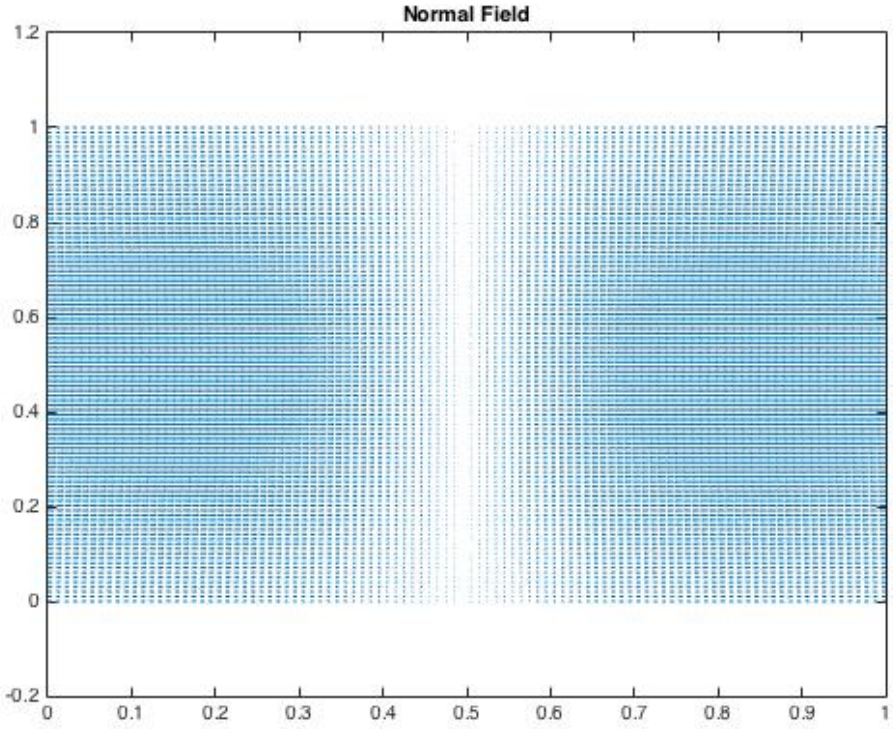
Procedure

1. Take 3 or 4 images of the same fixed object under different known light source direction.
2. Read the image in Matlab along with their light source direction respectively
3. The images have intensities in the range 0 to 255, so after reading the images divide them by 255 to get floating point values in the range of $[0 \cdots 1]$.
4. If the light source vector is not normalized, then normalize the light source vector
5. Here, I followed the F&P book pages 82 and 83 which explains the calculation of the normal and the albedo at each pixel.

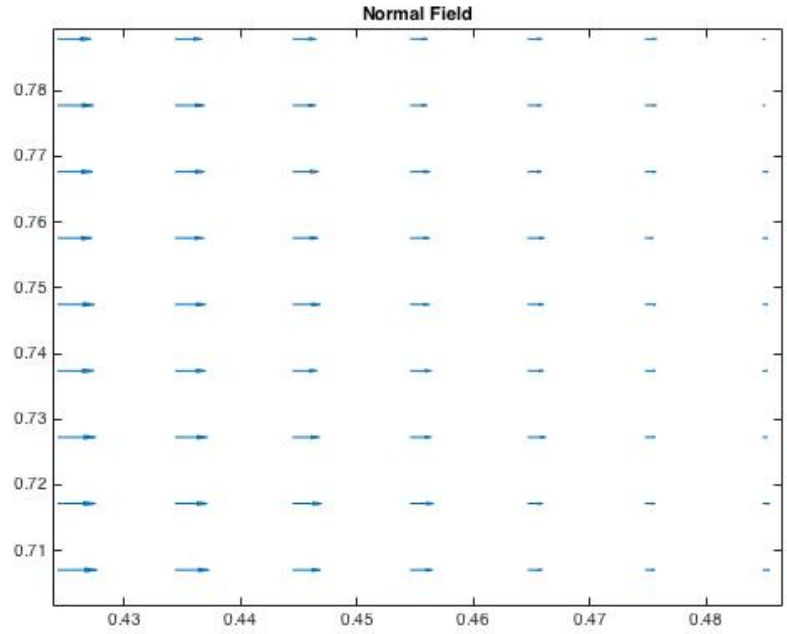
Matlab Code:
Matlab code is present in Appendix A

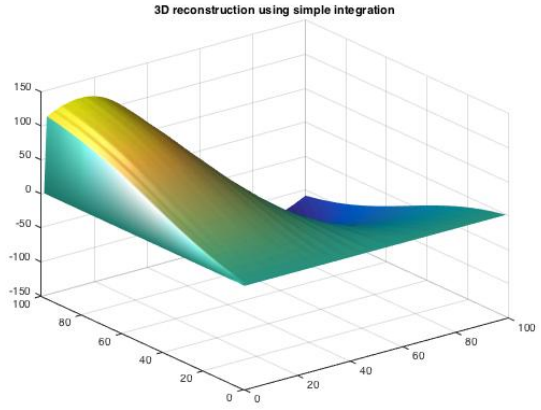
Output

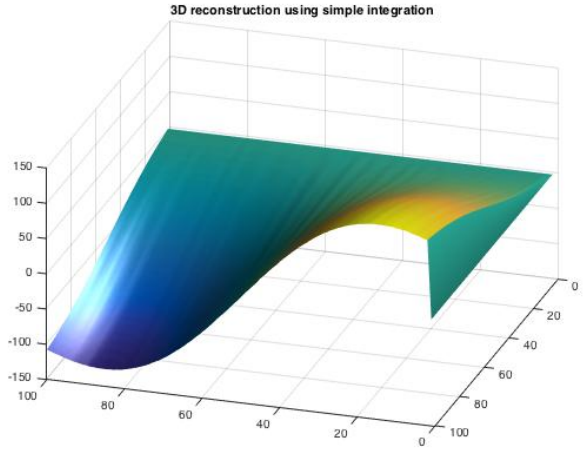
This is the albedo map of the image . Albedo was calculated by taking the magnitude of g(x,y) and it can be see that albedo is non constant in the images.

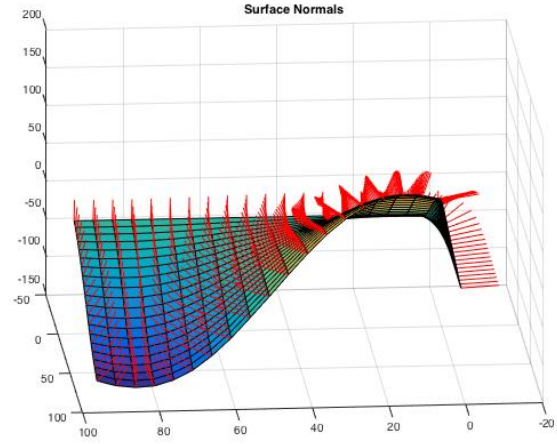


This is the normal field . This field is plotted as a vector of p and q where p is the derivative of z(x,y) with respect to x and q is the derivative of z(x,y) with respect to y. Here we can see that the normal field is directed towards center from both left and right side of the image. The vertical bright region indicate, it's the peak as the normal vector is perpendicular to the object.







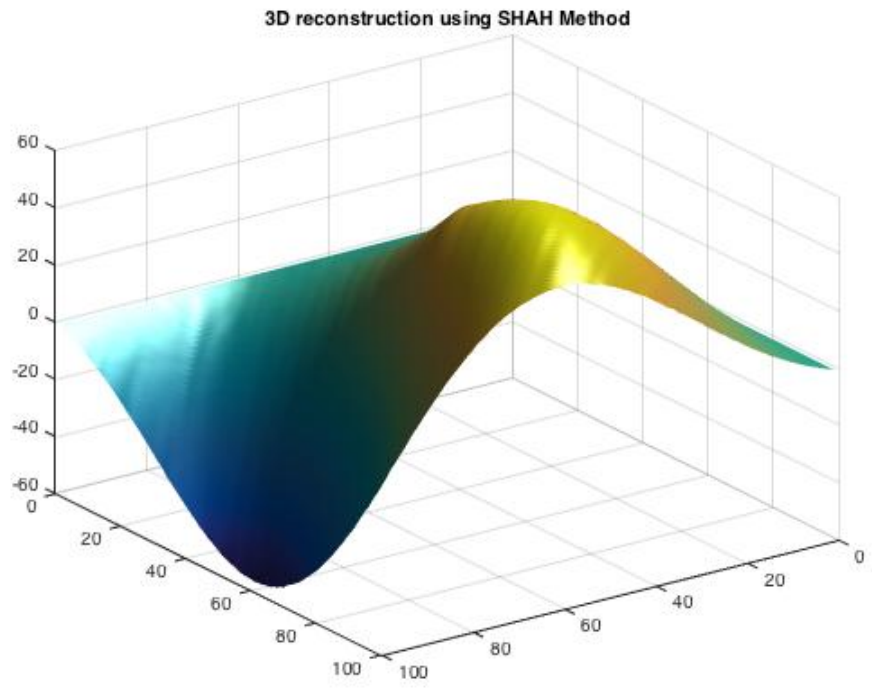


The reconstruction of 3d object is done by integration. Here , we performed the simple integration using the algorithm mentioned in the Forsyth and P. Book. I have also shown the surface normal to the reconstructed plot.
The algorithm is

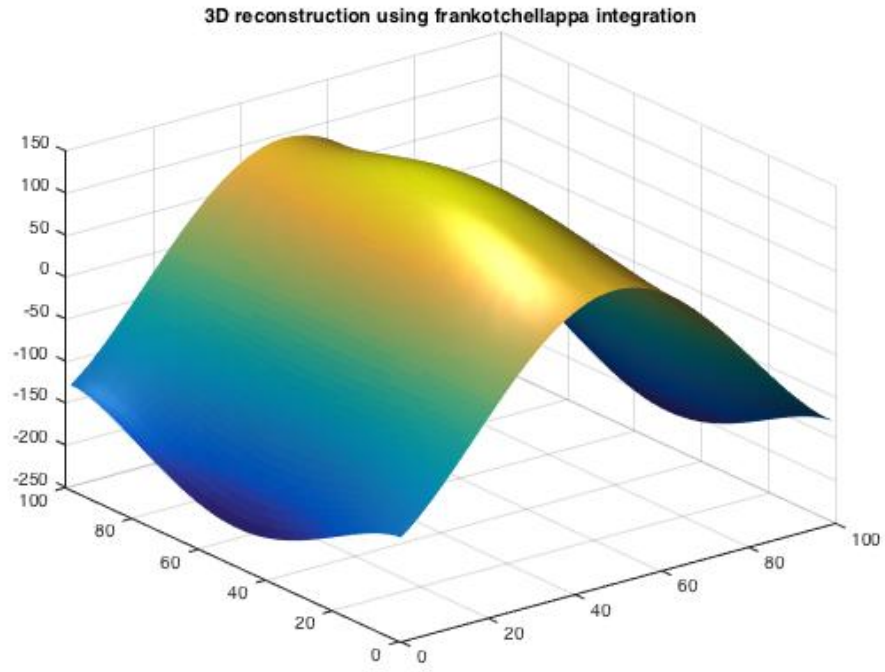
```
top left corner of height map is zero

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
```



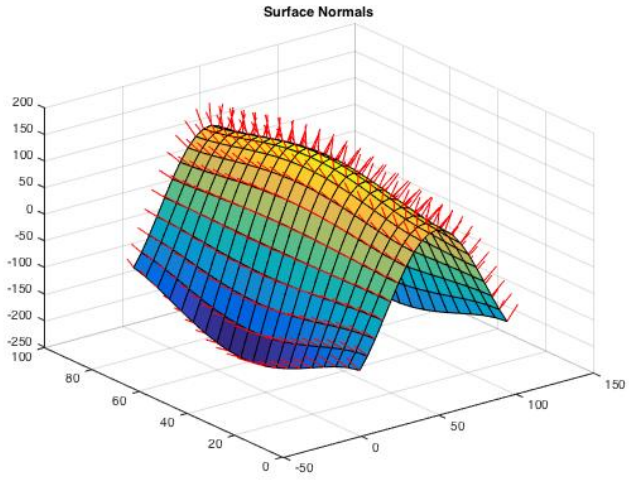
This depth map is created from frankotchellapa method

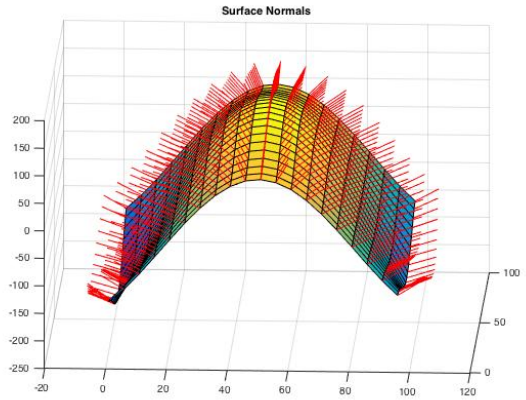


The integration is done by frankotchellappa algorithm. We see that we get a perfect smooth triangle canvas. The algorithm is

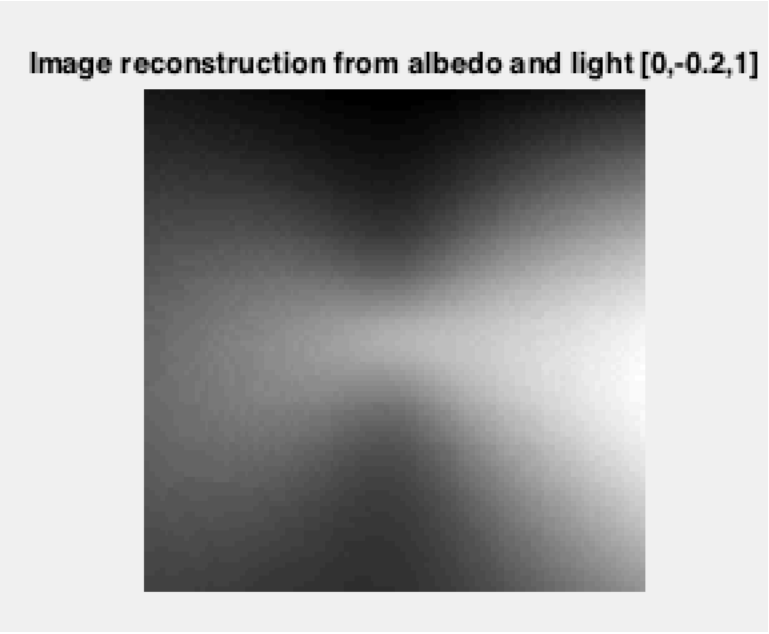
Set up matrices specifying frequencies in the x and y directions corresponding to the Fourier transforms of the gradient data. They range from -0.5 cycles/pixel to + 0.5 cycles/pixel. The fiddly bits in the line below give the appropriate result depending on whether there are an even or odd number of rows and columns

Integrate in the frequency domain by phase shifting by pi/2 and weighting the Fourier coefficients by their frequencies in x and y and then dividing by the squared frequency. eps is added to the denominator to avoid division by 0.





I have also shown the surface normal to the 3D plot after integration using frankotchellapa algorithm.



ISSUES THAT AROSE

As it was a synthetic image, I didn't face much difficulty in implementing it. When I used inv() or \ command in Matlab for matrix inversion , I was getting the error that Matrix is singular. So I tackled that problem by using pinv().

WAYS TO IMPROVE THE METHOD

By using a smoothing filter , one can get better smooth surface. Proper integration technique should be used which has less time complexity and memory efficient. One can also use the diagonal loading to avoid the inverse problem. To obtain the accuracy of the result , Choose several different integration paths, and build average height map

OBSERVATION and DISCUSSION

The result obtained using simple integration technique is not so effective as Noise and numerical (in)accuracy are added up and result in distorted surface.The plot is more leaned to one direction and the whole plot emerges from there. It is possible that there are many zeros near the boundary and the algorithm is not so quick.

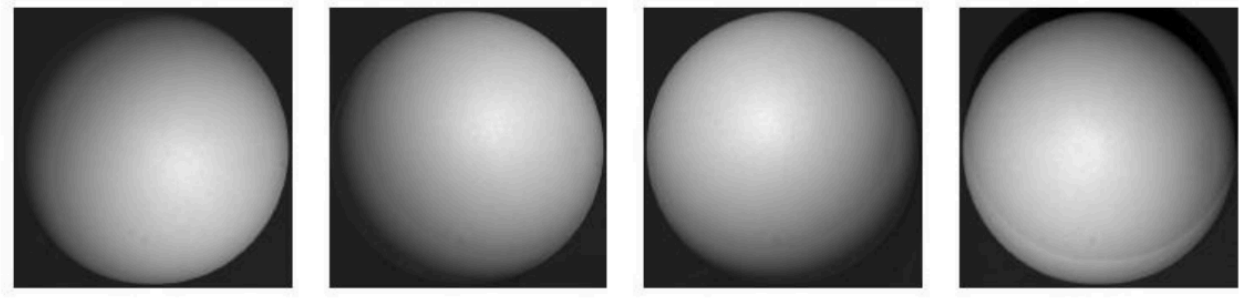
Whereas the result obtained from frankotchellapa gives us the perfect reconstruction. It takes into consideration of the albedo value when it is zero and it has to divide g(x,y) to obtain the normal vector.

A new shaded image is created by choosing a new light source position vector (0.-0.2,1) and calculating the dot product of surface normals and light source vector (both normalized) at each pixel, resulting in an image showing a light source not used for reconstruction

2.2 Shape from Shading from real Images

The goal of this assignment is to implement an algorithm that reconstructs a surface from real images using the concept of photometric stereo

Sphere Images:

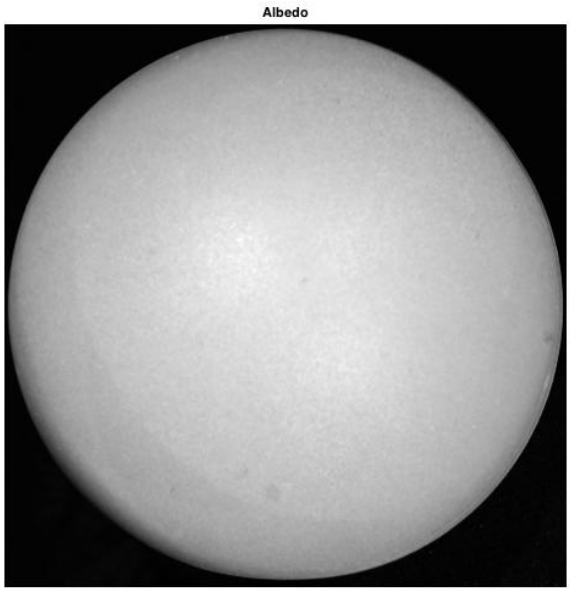


real1.bmp real2.bmp real3.bmp real4.bmp

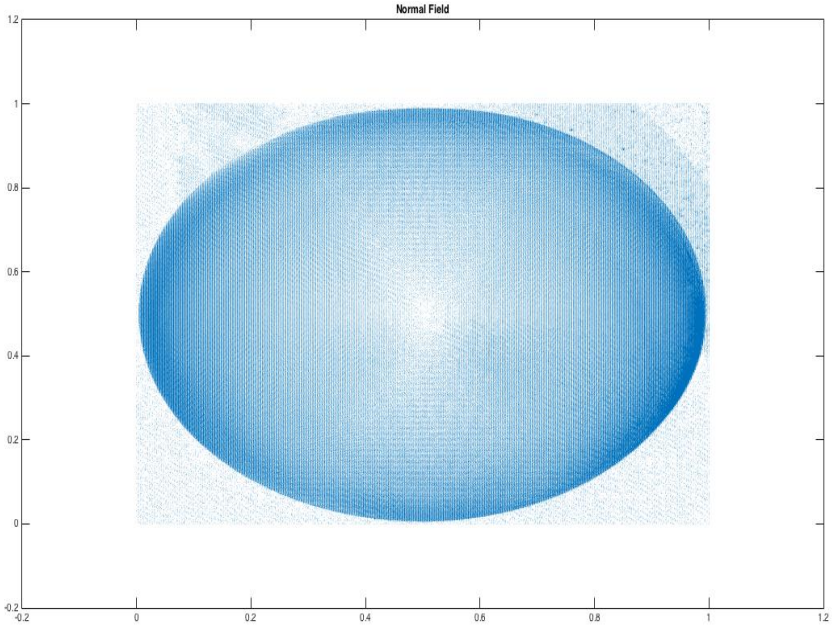
```
real1.bmp --> V1 = [0.38359 0.236647 0.892668]
real2.bmp --> V2 = [0.372825 -0.303914 0.87672]
real3.bmp --> V3 = [-0.250814 -0.34752 0.903505]
real4.bmp --> V4 = [-0.203844 0.096308 0.974255]
```

The procedure for this is same as above.

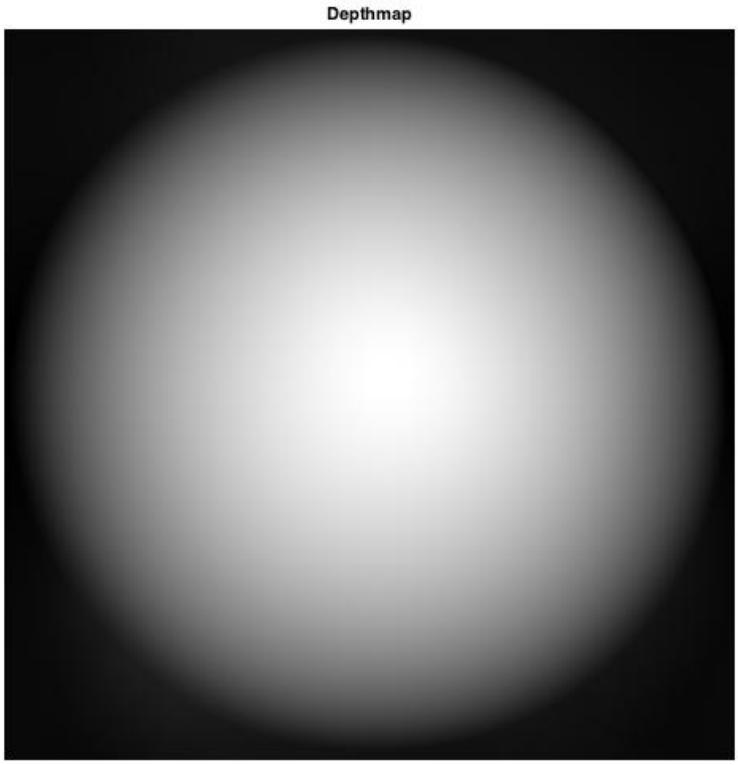
OUTPUT



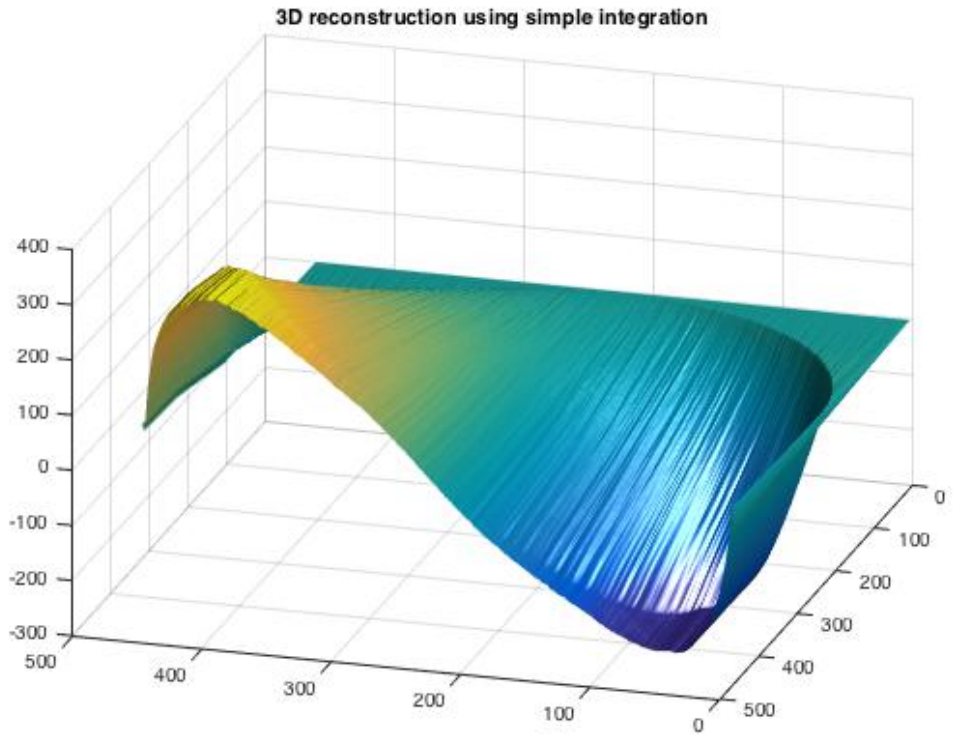
This is the albedo map of the image . Albedo was calculated by taking the magnitude of g(x,y) and it can be see that albedo is non constant in the images.



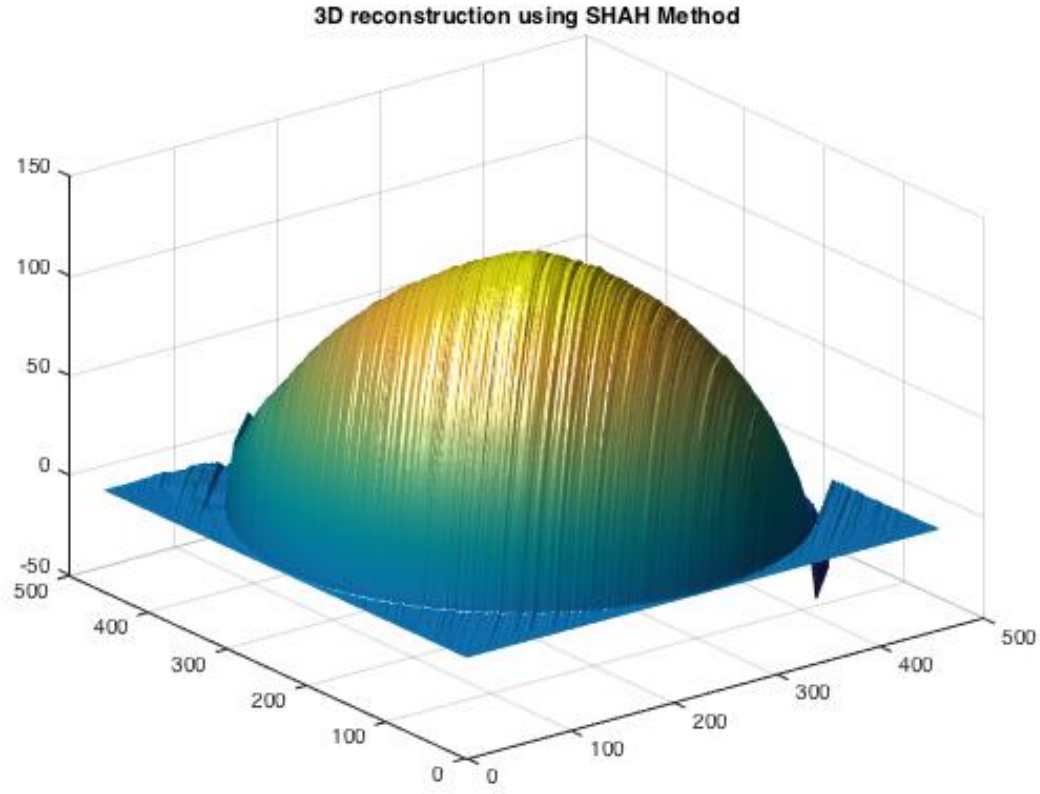
This is the normal field . This field is plotted as a vector of p and q where p is the derivative of z(x,y) with respect to x and q is the derivative of z(x,y) with respect to y. Here we can see that the normal field is away from center in all direction of the image. The center bright region indicate, it's the peak as the normal vector is perpendicular to the object.

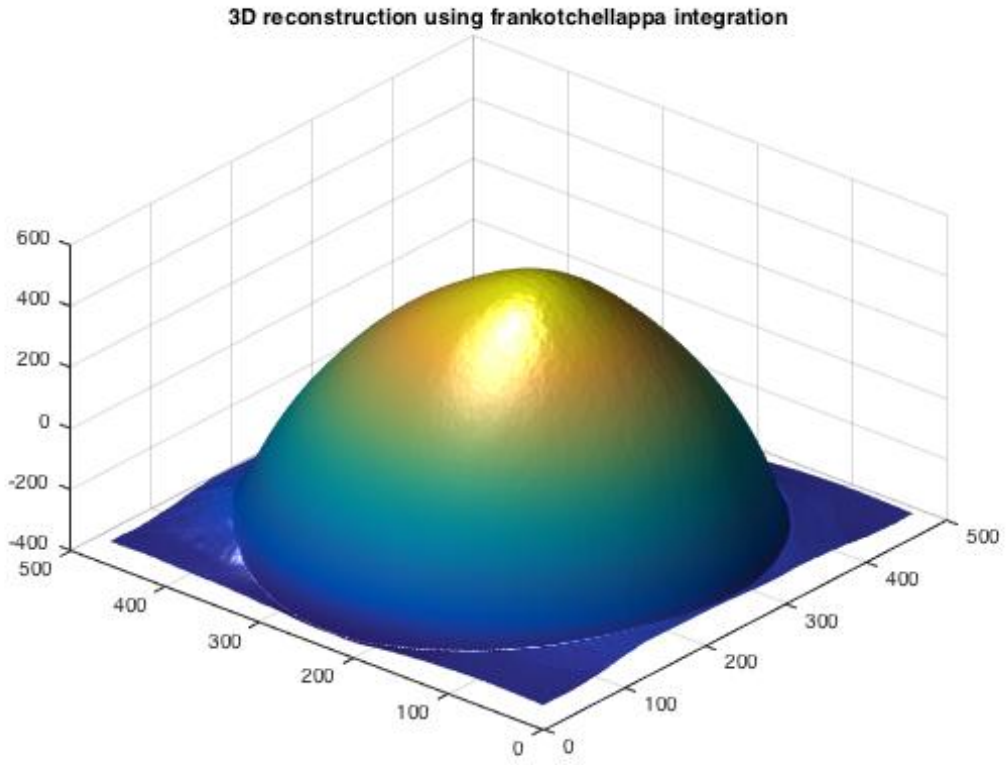


This is the depthmap. Bright indicates that the image is at front and gray dark represent the image is at back.

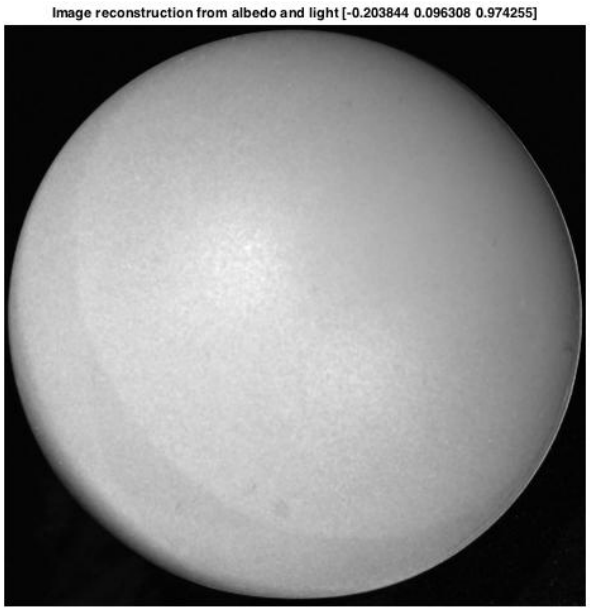


The reconstruction of 3d object is done by integration. Here , we performed the simple integration using the algorithm mentioned in the Forsyth and P. Book. Its not the result what we wanted , hence this technique doen't work. The algorithm is given above.





The integration is done by frankotchellappa algorithm. We see that we get a perfect smooth sphere canvas.



SHAPE FROM SHADING FOR THE DOG

Dog Images:

dog1.tif

$$\begin{bmatrix} 16 \\ 19 \\ 30 \end{bmatrix}$$

dog2.tif

$$\begin{bmatrix} 13 \\ 16 \\ 30 \end{bmatrix}$$

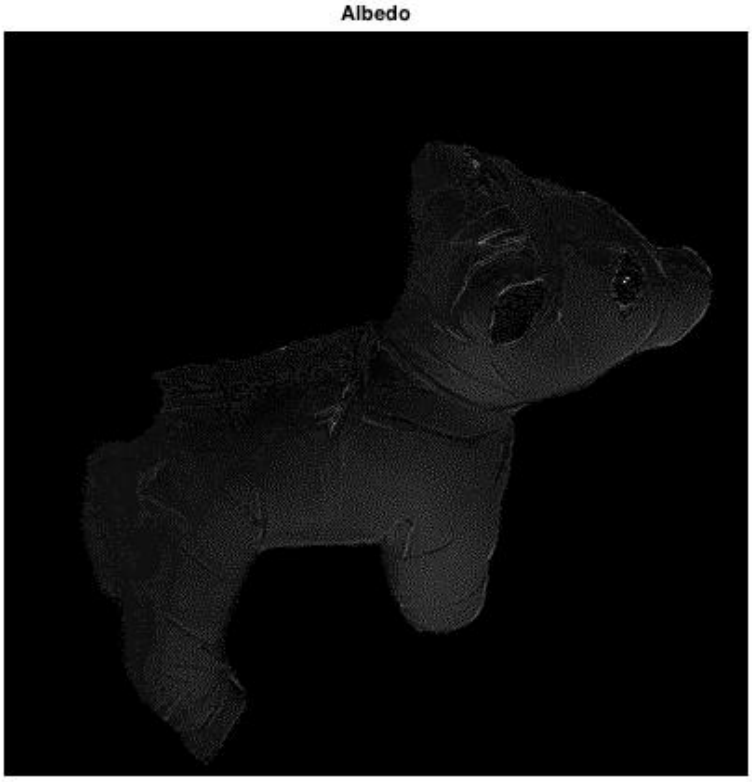
dog3.tif

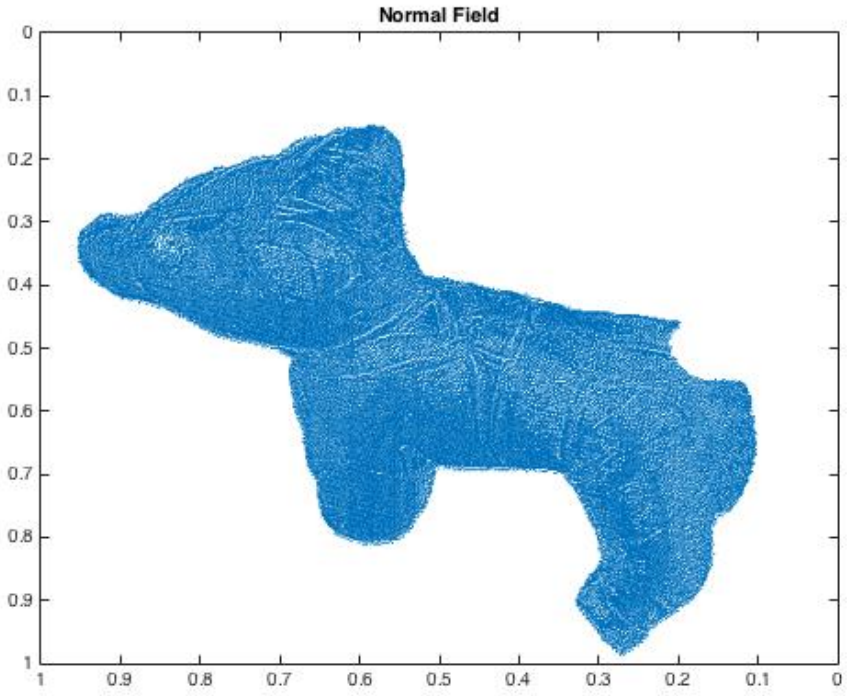
$$\begin{bmatrix} -17 \\ 10.5 \\ 26.5 \end{bmatrix}$$

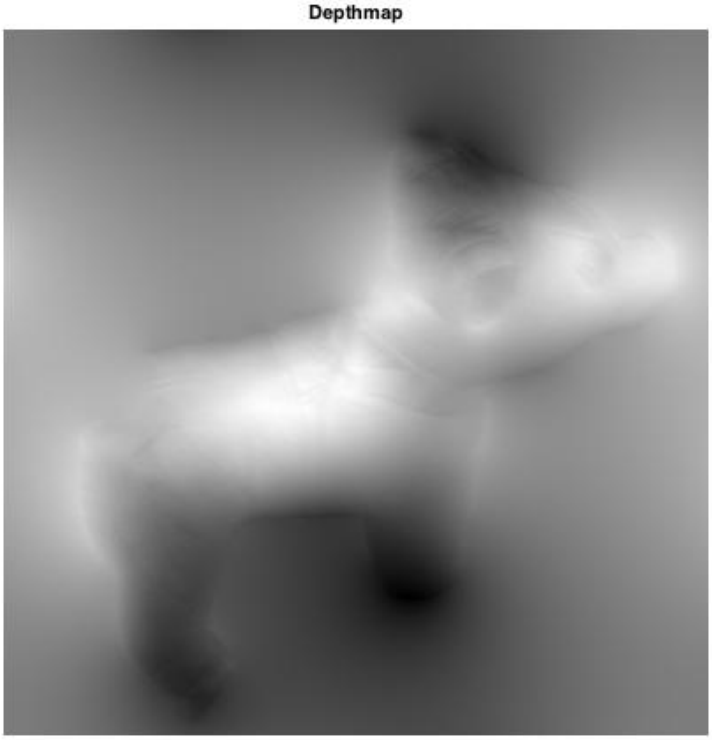
dog4.tif

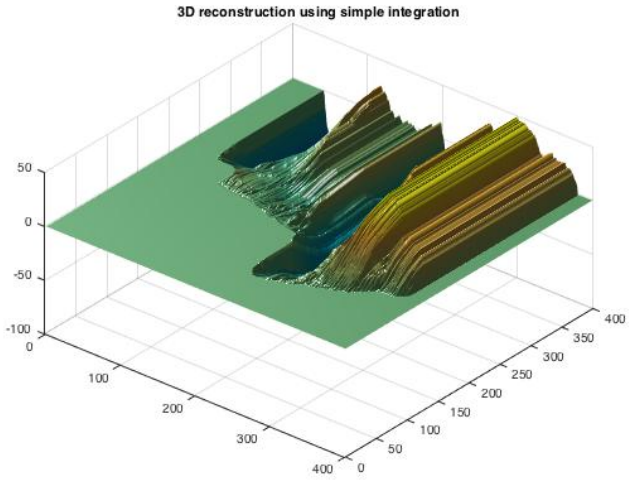
$$\begin{bmatrix} 9 \\ -25 \\ 4 \end{bmatrix}$$

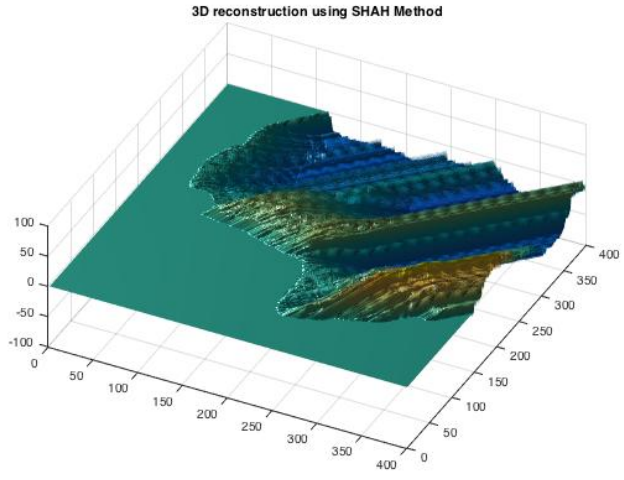
OUTPUT Obtained

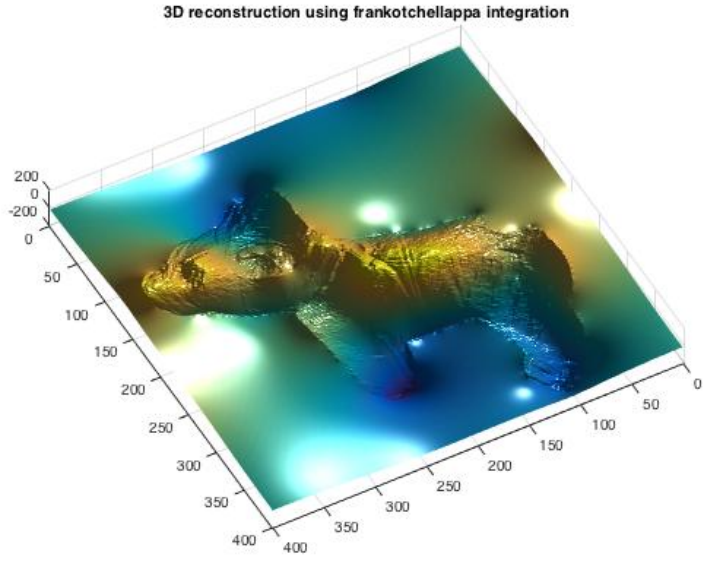


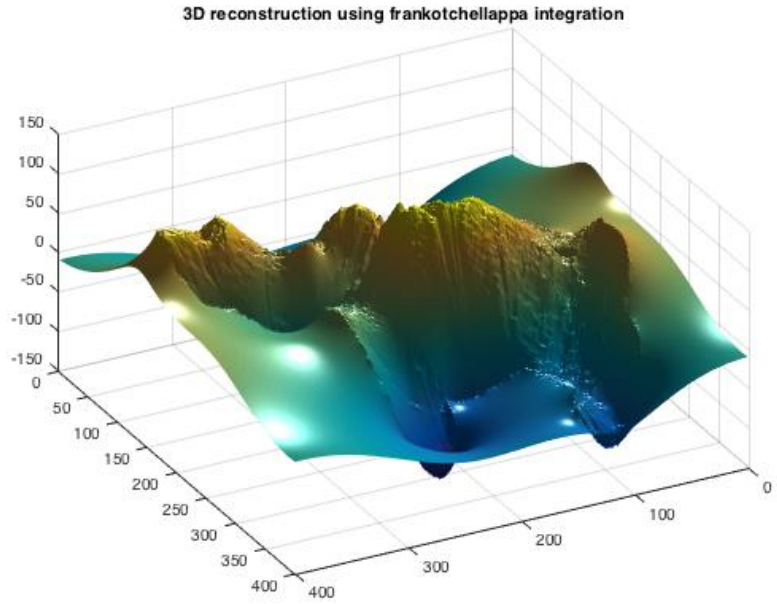

Normal Field
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0

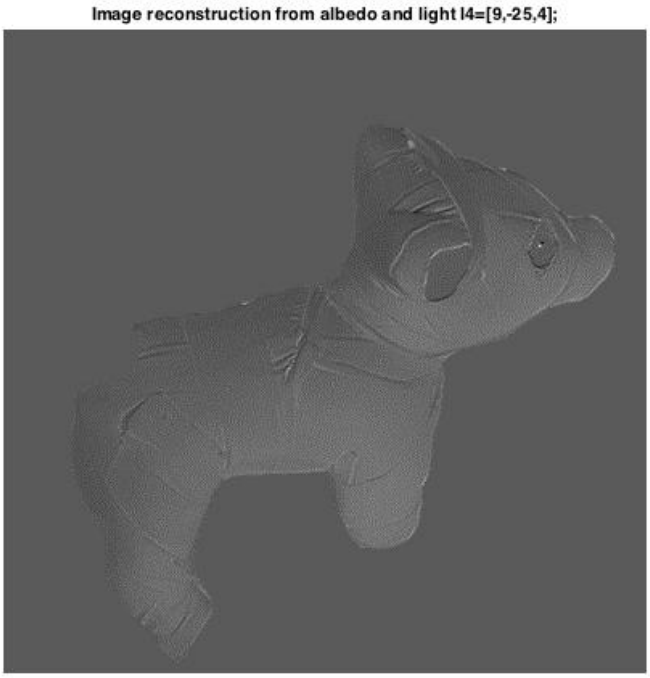

Depthmap


3D reconstruction using simple integration


3D reconstruction using SHAH Method


3D reconstruction using frankotchellappa integration


3D reconstruction using frankotchellappa integration



ISSUES THAT AROSE

As it was a real image, I faced little difficulty in implementing it. When I used inv() or \ command in Matlab for matrix inversion , I was getting the error that Matrix is singular. So I tackled that problem by using pinv().

WAYS TO IMPROVE THE METHOD

By using a smoothing filter , one can get better smooth surface. Proper integration technique should be used which has less time complexity and memory efficient. One can also use the diagonal loading to avoid the inverse problem. To obtain the accuracy of the result , Choose several different integration paths, and build average height map

OBSERVATION and DISCUSSION

The result obtained using simple integration technique is not so effective as Noise and numerical (in)accuracy are added up and result in distorted surface. The plot is more leaned to one direction and the whole plot emerges from there. It is possible that there are many zeros near the boundary and the algorithm is not so quick.

For Shah's method, in case of sphere we got very good result, but for dog, we are not getting the good result.

Whereas the result obtained from frankotchellapa gives us the perfect reconstruction. It takes into consideration of the albedo value when it is zero and it has to divide g(x,y) to obtain the normal vector.

A new shaded image is created by choosing a new light source position vector [-0.203844 0.096308 0.974255] for the sphere and [9,-25,4] for the dog and calculating the dot product of surface normals and light source vector (both normalized) at each pixel, resulting in an image showing a light source not used for reconstruction. The image obtained is similar to the image 4 given in the data.

CONCLUSION

In this assignment, shape from shading technique is used, which allow us to reconstruct the shape of an object using multiple images where the light source position is changing and according to the object reflecting properties modify the shading. We noticed that with the integration technique we used that a bunch of issues occurred. Theoretically we only need 3 images to solve our equation system. But most of the time there are dark areas in the image which prevent us from getting complete information about the shape. This resulted in either an unsolvable numerical system due to the lack of information or to results that were completely wrong due to partial information about the shape that were used for computation. We illustrated this aspect by showing how using only three images could affect the reconstruction of the shape of the object. The integration method is pretty simple and is based on normal integration. But this method does not really take in account some properties of the shape as differentiability. Indeed an interesting property of shapes is, in most cases, to be differentiable and continuous. So we need to enforce properties on the second derivative, which is not done in this implementation. So an interesting point of improvement would be to use Horn method based on Fourier transform to enforce this property. Our attempt to do so was not very successful but with more investment we would succeed. Noise as always with digital images, and processing approximation can be a great source of errors so using multiple images and a Least Square Estimate appears here to be a better solution.

References:

1. Lecture slides taught by professor Guido Gerig.
2. Textbook Computer Vision : A modern Approach by Forsyth and Ponce

Appendix A

```
clc;clear all;close all

img1=imread('im1.png');
img1=im2double(img1);
l1=[0 0 1];

img2=imread('im2.png');
img2=im2double(img2);
l2=[0.2 0 1];

img3=imread('im3.png');
img3=im2double(img3);
l3=[-0.2 0 1];

 img4=imread('im4.png');
% img4=im2double(img4);
% l4=[0 0.2 1];

S=[l1;l2;l3];
I=zeros(3,1);
[h,w]=size(img1);
g=zeros(h,w,3);
for i=1:h
   for j=1:w
      I(:,1)=[img1(i,j);img2(i,j);img3(i,j)];
      Is=[img1(i,j),0,0;0,img2(i,j),0;0,0,img3(i,j)];
      Im=Is*I;
      g(i,j,:)=pinv(Is*S)*Im;
% g(i,j,:)=S\I;
   end
end
 albedo=sqrt(g(:,:,1).^2+ g(:,:,2).^2+g(:,:,3).^2);
 figure,imshow(albedo,[]);title('Albedo')
 N=zeros(h,w,3);
N(:,:,1)=g(:,:,1)./albedo;
N(:,:,2)=g(:,:,2)./albedo;
N(:,:,3)=g(:,:,3)./albedo;
p=N(:,:,1)./N(:,:,3);
q=N(:,:,2)./N(:,:,3);
x1=linspace(0,1,100);
y1=linspace(0,1,100);
```

```
[x,y] = meshgrid(x1,y1);
figure
quiver(x,y,p,q),title('Normal Field');
fx=zeros(h,w);
depth=zeros(h,w);
a=0;
%%
% Integration using simple algorith used in forsyth
for i=2:h
   depth(i,1)=depth(i-1,1)+q(i,1);
end
for i=2:h
   for j=2:w
   depth(i,j)=depth(i-1,j)+p(i,j);
   end
end
depthc=zeros(h,w);

%%

% Integration using frankotchellappa
 d2=frankotchellappa(p,q);
 DepthMap=d2;

[ X, Y ] = meshgrid( 1:w, 1:h );
figure;
surf( X, Y, depth, 'EdgeColor', 'none' ); title('3D reconstruction using simple
integration');
camlight left;
lighting phong

figure;
surf( X, Y, DepthMap, 'EdgeColor', 'none' ); title('3D reconstruction using simple
integration');
camlight left;
lighting phong

figure
surfnorm(X(1:5:100,1:5:100),Y(1:5:100,1:5:100),d2(1:5:100,1:5:100)),title('Surface
Normals');

% %%
% l4=[0,-0.2,1];
%
```

```
% I4=zeros(h,w);
% nor=zeros(1,3);
% for i=1:h
%     for j=1:w
%         nor(1,:)=[p(i,j),q(i,j),1];
% I4(i,j)=albedo(i,j)*(dot(nor(1,:),l4(1,:),2));
%     end
% end
% figure,
% subplot(1,2,1),imshow(img4,[]);title('Image 4');
% subplot(1,2,2),imshow(I4,[]);title('Image reconstruction from albedo and light [0,-
0.2,1]');
```

APPENDIX B

```
clc;clear all;close all

img1=imread('sphere-images/real1.bmp');
img1=im2double(img1);
l1=[0.38359 0.236647 0.892668];

img2=imread('sphere-images/real2.bmp');
img2=im2double(img2);
l2=[0.372825 -0.303914 0.87672];

img3=imread('sphere-images/real3.bmp');
img3=im2double(img3);
l3=[-0.250814 -0.34752 0.903505];

% img4=imread('im4.png');
% img4=im2double(img4);
% l4=[-0.203844 0.096308 0.974255];

S=[l1;l2;l3];
I=zeros(3,1);
[h,w]=size(img1);
g=zeros(h,w,3);
for i=1:h
    for j=1:w
        I(:,1)=[img1(i,j);img2(i,j);img3(i,j)];
        Is=[img1(i,j),0,0;0,img2(i,j),0;0,0,img3(i,j)];
        Im=Is*I;
        g(i,j,:)=pinv(Is*S)*Im;
    end
```

```
end
 albedo=sqrt(g(:,:,1).^2+ g(:,:,2).^2+g(:,:,3).^2);
 figure,imshow(albedo,[]);title('Albedo')
 N=zeros(h,w,3);
N(:,:,1)=g(:,:,1)./albedo;
N(:,:,2)=g(:,:,2)./albedo;
N(:,:,3)=g(:,:,3)./albedo;
p=N(:,:,1)./N(:,:,3);
q=N(:,:,2)./N(:,:,3);
x1=linspace(0,1,h);
y1=linspace(0,1,w);
[x,y] = meshgrid(x1,y1);
figure
quiver(x,y,p,q),title('Normal Field');
fx=zeros(h,w);
depth=zeros(h,w);
a=0;

% Integration using simple algorith used in forsyth
for i=2:h
    depth(i,1)=depth(i-1,1)+q(i,1);
end
for i=2:h
    for j=2:w
    depth(i,j)=depth(i-1,j)+p(i,j);
    end
end
depthc=zeros(h,w);

%%
%compute depth using SHAH'S Method
depths=zeros(h,w);
for i = 2:h
for j = 2:w
depths(i,j) = depths(i-1,j-1)+q(i,j)+p(i,j);
end
end
%%
% Integration using frankotchellappa
 d2=frankotchellappa(p,q);
 DepthMap=-d2;
figure,imshow(DepthMap,[]);title('Depthmap');
[ X, Y ] = meshgrid( 1:w, 1:h );

figure;
surf( X, Y, depth, 'EdgeColor', 'none' ); title('3D
```

```
reconstruction using simple integration');
camlight left;
lighting phong


figure;
surf( X, Y, depths, 'EdgeColor', 'none' ); title('3D
reconstruction using SHAH Method');
camlight left;
lighting phong


figure;
surf( X, Y, DepthMap, 'EdgeColor', 'none' ); title('3D
reconstruction using frankotchellappa integration');
camlight left;
lighting phong

figure
surfnorm(X(1:5:100,1:5:100),Y(1:5:100,1:5:100),DepthMap(1:5:10
0,1:5:100)),title('Surface Normals');

%%
l4=[-0.203844 0.096308 0.974255];

I4=zeros(h,w);
nor=zeros(1,3);
for i=1:h
    for j=1:w
        nor(1,:)=[p(i,j),q(i,j),1];
I4(i,j)=albedo(i,j)*(dot(nor(1,:),l4(1,:),2));
    end
end
figure,

imshow(I4,[]);title('Image reconstruction from albedo and
light [-0.203844 0.096308 0.974255]');
```

APPENDIX C

```
clc;
clear all;
close all;

img1=imread('dog1.png');
img1=rgb2gray(img1);
```

```
img1=im2double(img1);
l1=[16 19 30];
l1=l1/norm(l1);

img2=imread('dog2.png');
img2=rgb2gray(img2);
img2=im2double(img2);
l2=[13 16 30];
l2=l2/norm(l2);

img3=imread('dog3.png');
img3=rgb2gray(img3);
img3=im2double(img3);
l3=[-17 10.5 26.5];
l3=l3/norm(l3);

S= [l1;l2;l3];
[h,w]=size(img1);

b=ones(h,w,3);
b=double(b);
g=zeros(h,w,3);

p=ones(h,w);
p=double(p);
q=p;

for i=1:h
    for j=1:w
        E=[img1(i,j);img2(i,j);img3(i,j)];
        E=double(E);
        tb=(inv(S'*S))*S'*E;
        g(i,j,:)=tb;
        nbm=norm(tb);
        if(nbm==0)
            b(i,j,:)=0;
        else
            b(i,j,:)=tb/nbm;
        end
        tM=[b(i,j,1) b(i,j,2) b(i,j,3)];
        nbm=norm(tM);
        if(nbm==0)
            tM=[0 0 0];
        else
            tM=tM/nbm;
        end
```

```
        p(i,j)=tM(1,1);
        q(i,j)=tM(1,2);
    end
end

albedo=sqrt(g(:,:,1).^2+ g(:,:,2).^2+g(:,:,3).^2);
figure,imshow(albedo,[]);title('Albedo')

x1=linspace(0,1,h);
y1=linspace(0,1,w);
[x,y] = meshgrid(x1,y1);
figure
quiver(x,y,p,q),title('Normal Field');
%%
% Integration using simple algorith used in forsyth
depth=zeros(h,w);
for i=2:h
    depth(i,1)=depth(i-1,1)+q(i,1);
end
for i=2:h
    for j=2:w
    depth(i,j)=depth(i-1,j)+p(i,j);
    end
end
depthc=zeros(h,w);

%%
%compute depth using SHAH'S Method
depths=zeros(h,w);
for i = 2:h
for j = 2:w
depths(i,j) = depths(i-1,j-1)+q(i,j)+p(i,j);
end
end
%%
% Depthmap reconstruction using Frankotchellapa
d2=frankotchellappa(p,q);
 DepthMap=d2;
figure,imshow(DepthMap,[]);title('Depthmap');
[ X, Y ] = meshgrid( 1:w, 1:h );

figure;
surf( X, Y, depth, 'EdgeColor', 'none' ); title('3D
reconstruction using simple integration');
camlight left;
lighting phong
```

```
figure;
surf( X, Y, depths, 'EdgeColor', 'none' ); title('3D
reconstruction using SHAH Method');
camlight left;
lighting phong


figure;
surf( X, Y, DepthMap, 'EdgeColor', 'none' ); title('3D
reconstruction using frankotchellappa integration');
camlight left;
lighting phong

figure
surfnorm(X(1:5:100,1:5:100),Y(1:5:100,1:5:100),DepthMap(1:5:10
0,1:5:100)),title('Surface Normals');

%%
l4=[9,-25,4];
l4=l4/norm(l4);

I4=zeros(h,w);
nor=zeros(1,3);
for i=1:h
    for j=1:w
        nor(1,:)=[p(i,j),q(i,j),1];
I4(i,j)=albedo(i,j)*(dot(nor(1,:),l4(1,:),2));
    end
end
figure,
imshow(I4,[]);title('Image reconstruction from albedo and
light l4=[9,-25,4];');
```