CSE252A Computer Vision
Assignment 2

Yingyan Hua and Yanli Wang

Problem 1

Solution:

The equation relating irradiance to radiance is $\mathrm{E(x)} = \mathrm{L}(\mathrm{x}, \theta, \varphi)\cos\theta \mathrm{d}\omega$. Replacing $\mathrm{L}(\mathrm{x}, \theta, \varphi)$ with the given expression of L, we get $\mathrm{E(x)} = \mathrm{L}(\mathrm{x}^2 + y^2 + 3)\cos\theta\mathrm{d}\omega$. Since we also know that $\mathrm{d}\omega = \frac{d_{A_1} \times cos\theta}{r^2}$, the expression for E(x) becomes $\mathrm{E(x)} = \mathrm{L}(\mathrm{x}^2 + y^2 + 3)\cos\theta \frac{cos\theta}{r^2} \mathrm{d}A_1$. Using trigonometry (see Figure 1 below), we get $\cos^2\theta = \frac{1}{\mathrm{x}^2+\mathrm{y}^2+1}$. With this equation, we can get rid of the angular term $(\cos\theta\cos\theta)$ in the equation for E(x). It thus becomes $\mathrm{E(x)} = \mathrm{L}(\mathrm{x}^2 + y^2 + 3) \times \frac{1}{(\mathrm{x}^2+y^2+1)^2} \times \frac{1}{r^2} \mathrm{d}A_1$. This is the irradiance over a small surface area dA_1 and the total irradiance by the rectangular area is the integral of E(x) over x = -3 to x = 3 and y = -2 to y = 2. Therefore, the following integration gives us the answer to this problem.

$$\mathrm{E(total)} = \int_{-2}^{2}\int_{-3}^{3} \mathrm{L}(\mathrm{x}^2 + y^2 + 3) \times \frac{1}{(\mathrm{x}^2+y^2+1)^2} \times \frac{1}{r^2} \mathrm{d}A_1$$

$$= \int_{-2}^{2}\int_{-3}^{3} \mathrm{L}(\mathrm{x}^2 + y^2 + 3) \times \frac{1}{(\mathrm{x}^2+y^2+1)^2} \times \frac{1}{r^2} \mathrm{dxdy}$$

Using Matlab, we obtained the integration result for E(total) as 12.0669L.

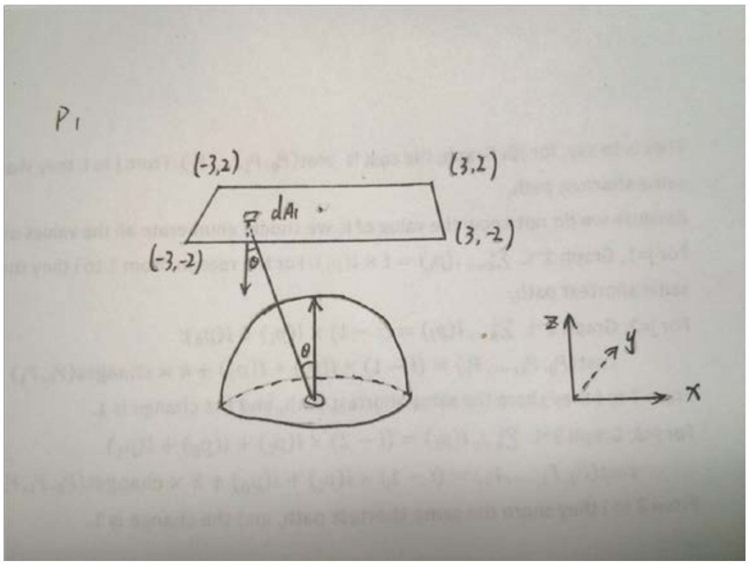


Figure 1

Matlab code:

```
function hw2_q1()
    function L = l(x,y)
        L = x.^2+y.^2+3
    end
    function FS = fs(x,y)
        FS = 1./(1+x.^2+y.^2);
    end
    function res = double(x,y)
        res = l(x,y).*fs(x,y).*fs(x,y);
    end

h = @double
res = quad2d(h, -3,3,-2,2)
end
```

Problem 2

Solution:

a. By similar reasoning as in Problem 1, we can get the irradiance from the top of the cylinder (see Figure 2 for trigonometry setup) as an integral of irradiance over $d\theta$ and $d\varphi$:

E_{top}(total) = $\int 2Lcos\theta d\omega$

$= \int_0^{2\pi}\int_0^{tan^{-1}\frac{r}{h}} 2Lcos\theta sin\theta d\theta d\varphi$

$= \pi L \int_0^{tan^{-1}\frac{r}{h}} sin2\theta d2\theta$

$= \pi L\left[1 - \cos\left(2tan^{-1}\frac{r}{h}\right)\right]$

Similarly, the total irradiance from the wall is:

E_{top}(total) = $\int Lcos\theta d\omega$

$= \int_0^{2\pi}\int_{tan^{-1}\frac{r}{h}}^{\frac{\pi}{2}} Lcos\theta sin\theta d\theta d\varphi$

$= \frac{1}{2}\pi L \int_{tan^{-1}\frac{r}{h}}^{\frac{\pi}{2}} sin2\theta d2\theta$

$= \frac{1}{2}\pi L\left[\cos\left(2tan^{-1}\frac{r}{h}\right) + 1\right]$

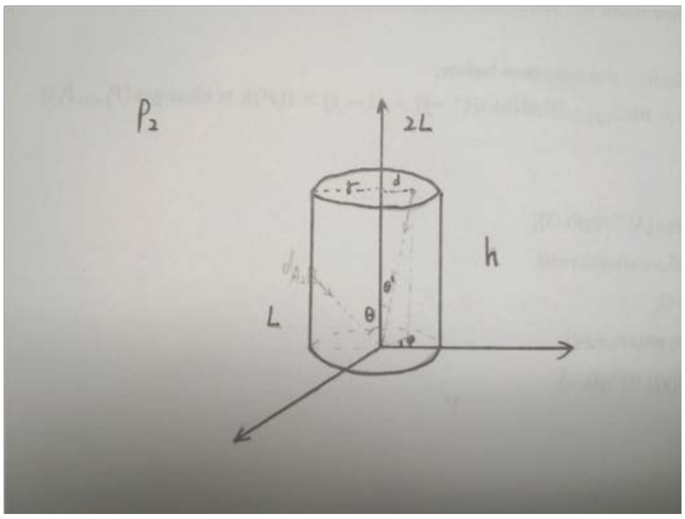


Figure 2

Therefore the total irradiance from the two surfaces is the sum of irradiance from the top and that from the wall:

E(total) = $\pi L\left[1 - \cos\left(2tan^{-1}\frac{r}{h}\right)\right] + \frac{1}{2}\pi L\left[\cos\left(2tan^{-1}\frac{r}{h}\right) + 1\right]$

$= \frac{1}{2}\pi L\left[3 - \cos\left(2tan^{-1}\frac{r}{h}\right)\right]$

b. If the radiance of the top is Ld^2 (where d is the distance to the center of the top, see Figure 2), the total irradiance received from the top becomes

E_{top}(total) = $\int_0^{2\pi}\int_0^{tan^{-1}\frac{r}{h}} Ld^2cos\theta sin\theta d\theta d\varphi$

$= 2\pi L \int_0^{tan^{-1}\frac{r}{h}} d^2 sin\theta cos\theta d\theta$

Since d can be related to the height (h) of the cylinder and angle (θ) by equation $\mathrm{d} = \mathrm{h}\tan\theta$, we derive a new expression for E_{top}(total) as follows.

E_{top}(total) = $2\pi L \int_0^{tan^{-1}\frac{r}{h}} d^2 sin\theta cos\theta d\theta$

$=2\pi L \int_0^{tan^{-1}\frac{r}{h}} h^2 tan^2\theta sin\theta cos\theta d\theta$

$=2\pi L h^2 \int_0^{tan^{-1}\frac{r}{h}} sin^2\theta tan\theta d\theta$

$= \frac{1}{2}\pi L h^2(\cos\left(2arctan\frac{r}{h}\right) - 4\log(\cos\left(arctan\frac{r}{h}\right) - 1))$

(PS: If there is irradiance from the wall, then the result should be

$$\frac{1}{2}\pi L h^2\left(\cos\left(2arctan\frac{r}{h}\right)-4\log\left(\cos\left(arctan\frac{r}{h}\right)-1\right)\right)+\frac{1}{2}\pi L\left[\cos\left(2tan^{-1}\frac{r}{h}\right)+1\right]\ )$$

Problem 3

Solution:

For an infinitesimal patch on the surface of a sphere (see Figure on the right), its solid angle is $d\omega=\frac{dA}{r^2}$. Since the area dA can also be represented as $dA \approx rd\theta \times rsin\theta d\varphi$, the solid angle becomes $d\omega=\frac{dA}{r^2}=sin\theta d\theta d\varphi$. The total number of steradians in a quarter sphere is thus an integration of solid angle over a quarter sphere where φ goes from 0 to π and θ goes from 0 to $\pi/2$. We have the following integration:

$$\int_0^{\frac{\pi}{2}}\int_0^{\pi} sin\theta d\varphi\, d\theta=\pi$$

Therefore, the answer to this problem is that the number of steradians in a quarter sphere is π.

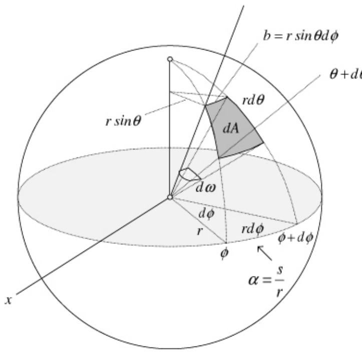


Figure 3 (Image borrowed from *www.et.**byu**.edu/~vps/ME340/TABLES/12.0.pdf.*)

Problem 4

Solution:

According to the argument of energy conservation in the textbook, the total energy leaving a surface must be less than or equal to the total energy arriving at the surface. If we represent the incoming and outgoing energy using integration of radiance and irradiance, we get the following relationship:

$$\int_{\Omega_i} L_i(x,\theta_i,\varphi_i)cos\theta_i d\omega_i \ge \int_{\Omega_o}[\int_{\Omega_i}\rho(\theta_i,\varphi_i;\theta_o,\varphi_o)L_i(x,\theta_i,\varphi_i)cos\theta_i d\omega_i]\ cos\theta_o d\omega_o.$$

Since the Lambertian surface given in this problem is one that reflects all light, the incoming and outgoing energy are equal to each other. From this information, we can get the following equation:

$$\int_{\Omega_i} L_i(x,\theta_i,\varphi_i)cos\theta_i d\omega_i = \int_{\Omega_o}[\int_{\Omega_i}\rho(\theta_i,\varphi_i;\theta_o,\varphi_o)L_i(x,\theta_i,\varphi_i)cos\theta_i d\omega_i]\ cos\theta_o d\omega_o.$$

Because the surface is an ideal Lambertian surface, $\rho(\theta_i,\varphi_i;\theta_o,\varphi_o)$ is considered to be a constant. If we let $\rho(\theta_i,\varphi_i;\theta_o,\varphi_o)$ be K, we can take it out from the integration on the right side of the equation:

$$\int_{\Omega_i} L_i(x,\theta_i,\varphi_i)cos\theta_i d\omega_i = K\int_{\Omega o}[\int_{\Omega_i} L_i(x,\theta_i,\varphi_i)cos\theta_i d\omega_i]\ cos\theta_o d\omega_o.$$

Since the solid angle ω_i of the incoming light and that of the outgoing light ω_o are two independent variables, the integration term $\int_{\Omega_i} L_i(x,\theta_i,\varphi_i)cos\theta_i d\omega_i$ is independent of the integration over ω_o. Again we can take this term out of the expression on the right side of the equation. If we let $\int_{\Omega_i} L_i(x,\theta_i,\varphi_i)cos\theta_i d\omega_i = J$, then we have: $J = JK\int_{\Omega_o} cos\theta_o d\omega_o$. Rearranging it, we get

$K = \frac{1}{\int_{\Omega_o} cos\theta_o d\omega_o}$. Replacing $d\omega_o$ by $sin\theta_o d\theta d\varphi$, we can integrate the denominator over a hemisphere, which is over φ from 0 to 2π and θ over 0 to π/2.

$$K = \frac{1}{\int_0^{\frac{\pi}{2}} \int_0^{2\pi} sin\theta_o cos\theta_o d\varphi d\theta} = \frac{1}{2\pi \int_0^{\frac{\pi}{2}} sin\theta_o cos\theta_o d\theta} = \frac{1}{\pi}$$

Therefore, the answer to the problem is that the BRDF $\rho(\theta_i, \varphi_i; \theta_o, \varphi_o)$ of such as a surface is $\frac{1}{\pi}$.

Problem5

Solution:

Part 1:

The surface normals and albedo were calculated based on the following equation: I(x, y) = kB(x) = g(x, y)·V_1, where g(x, y) = ρ(x, y)N(x, y), V_1 = kS_1, ρ(x, y) is the albedo, and N(x, y) is the surface normal. Given n light sources, for each of which V_i is known, we can obtain a matrix V, where

$$V = \begin{pmatrix} V_1^T \\ V_2^T \\ \dots \\ V_n^T \end{pmatrix} .$$

For each image point on the n images, we obtain a vector containing the image intensities:

I(x,y) = $\{I_1(x, y), I_2(x, y), \dots, I_n(x, y)\}$. Now we have I(x, y) = Vg(x, y) and g can be comuted by solving this linear equation per image point. Then we can extract the albedo from g because N is the unit normal, which means |g(x, y) | = ρ(x, y). The surface normal can also be extracted from g using the following equation, N(x, y) = $\frac{g(x,y)}{|g(x,y)|}$.

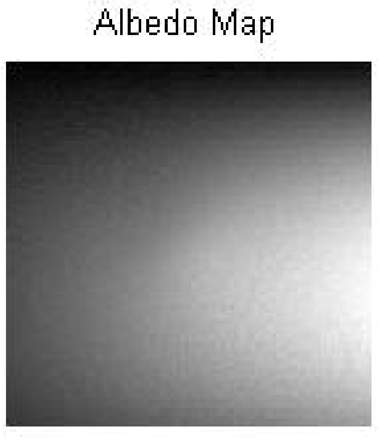
The height map was computed by the following procedure:

For each point in the image array, calculate p=N_1/N_3 and q=N_2/N_3. Then assign the height of the top left corner of height map to be zero. For each pixel in the left column of height map, assign its height value to be that of a previous point plus the corresponding q value. Then for each element of each row except for the leftmost, assign the height value to be that of a previous point plus the corresponding p value.

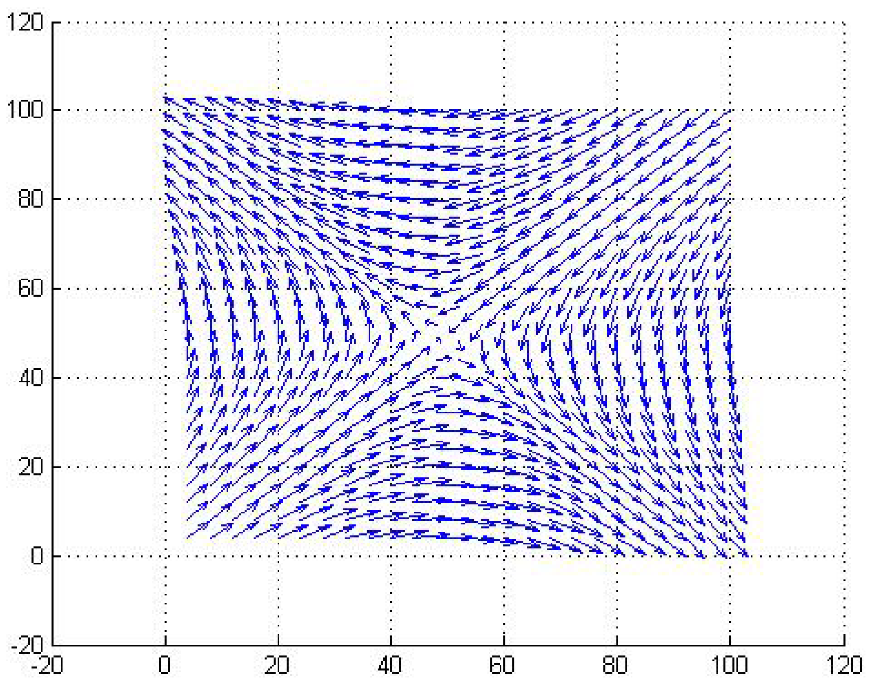
The detailed implementation in MATLAB is attached at the end of this report. The following are the results for albedo maps, surface normals, and height maps when 3 and 4 images were used in the computations, respectively.

A) im1, im2, im4

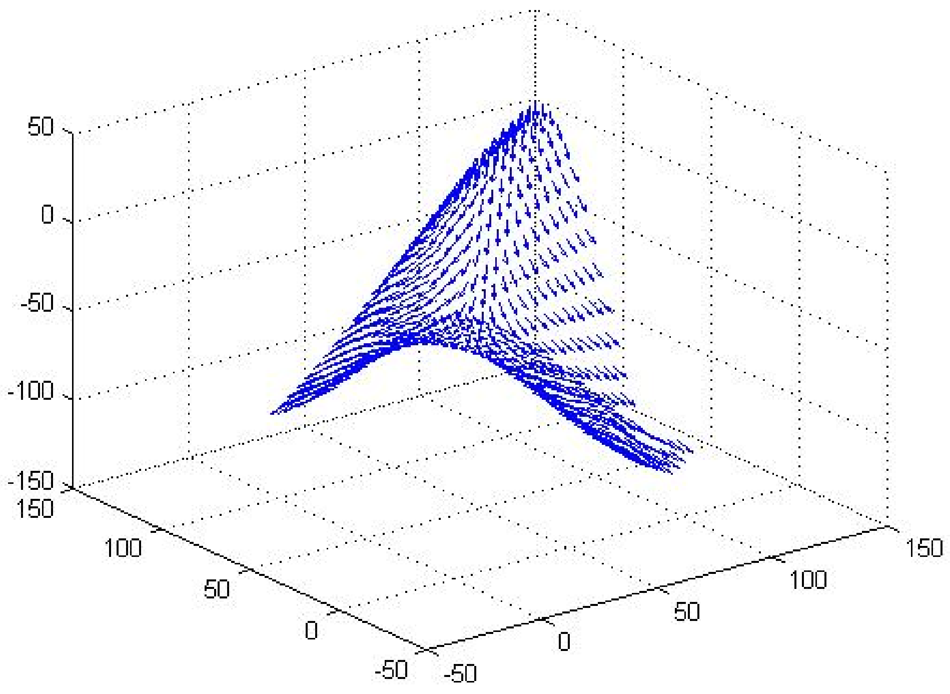
a) 3images_albedomap



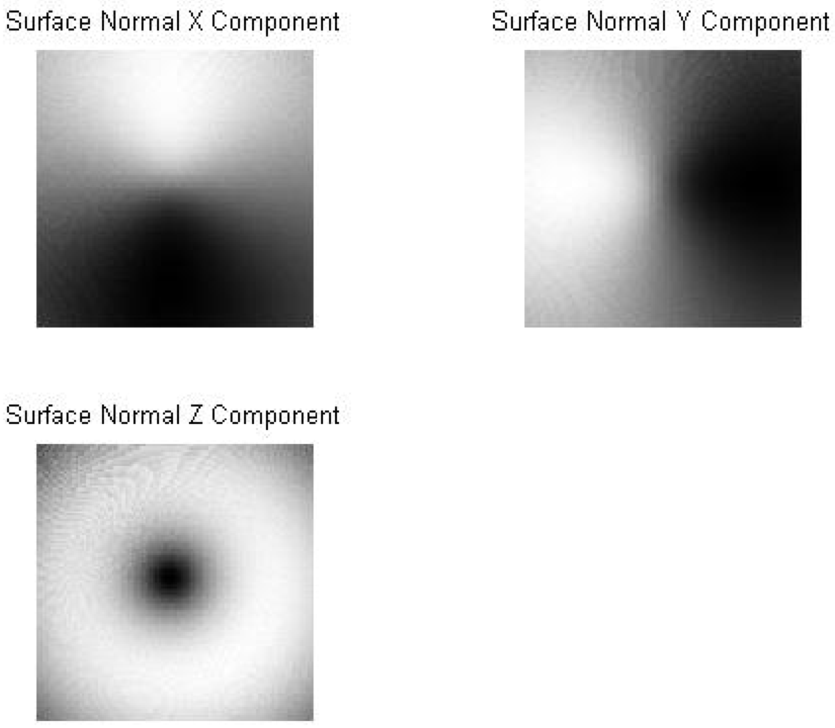
b) 3images_needlemap (2D view by quiver function)



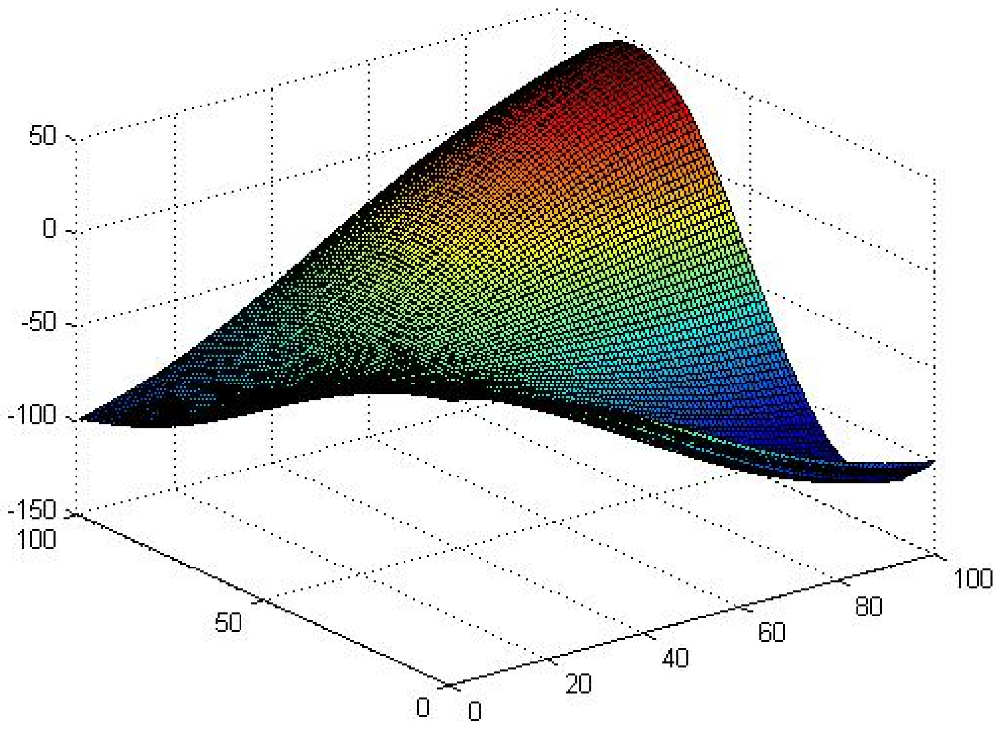
3images_needlemap (3D view by quiver3 function)



c) 3images_three_surfacenormals

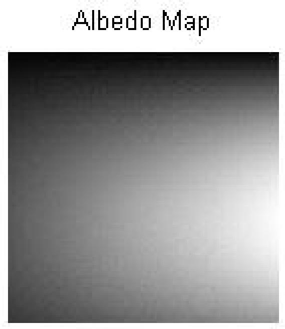


d) 3images_depthmap

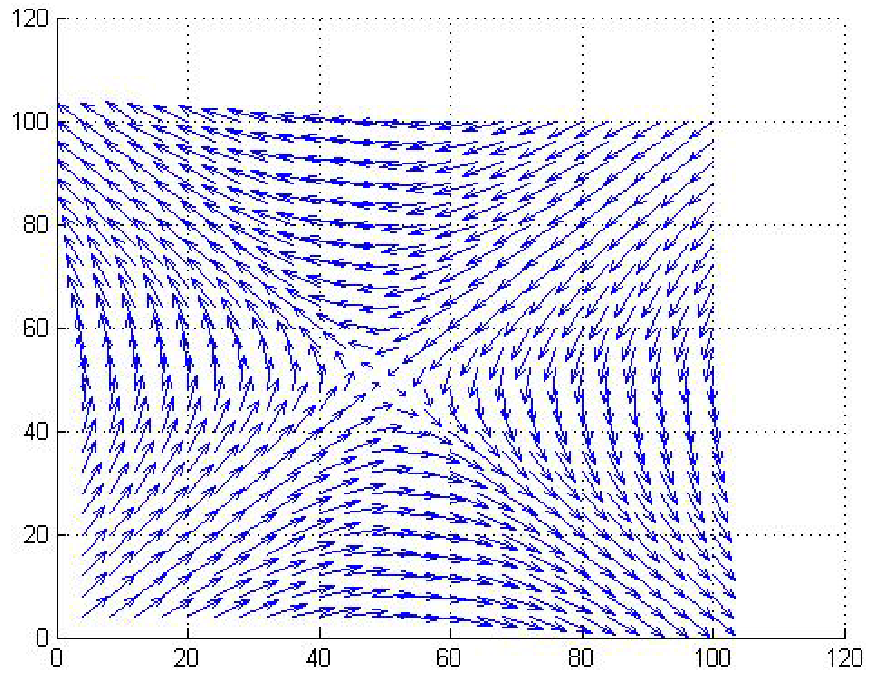


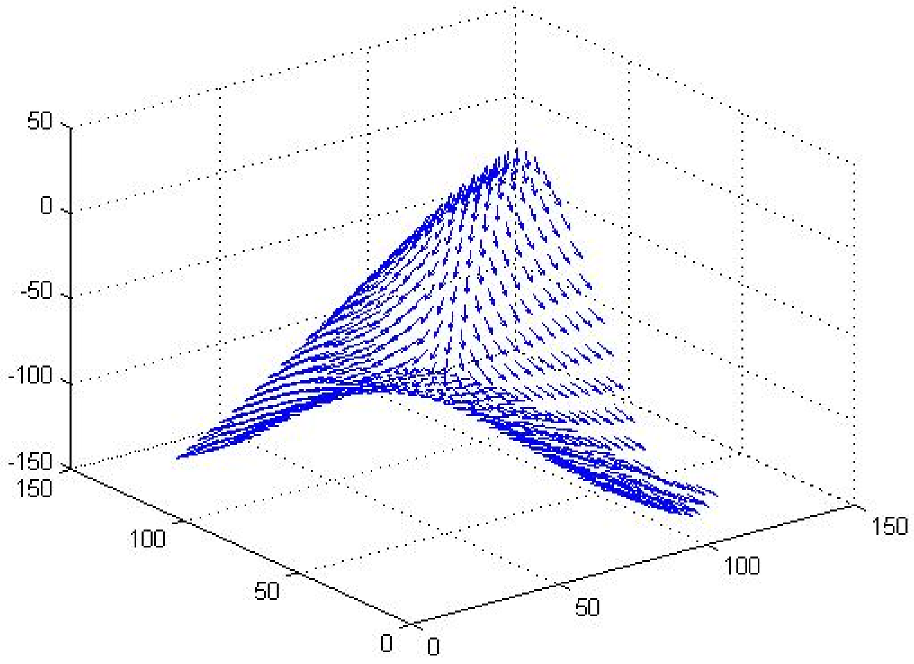
B) All four images

a) 4images_albedomap

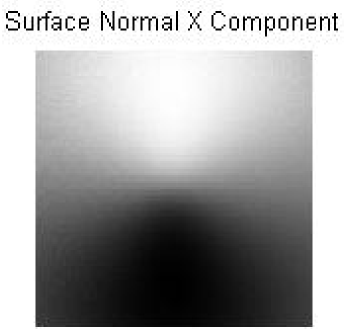


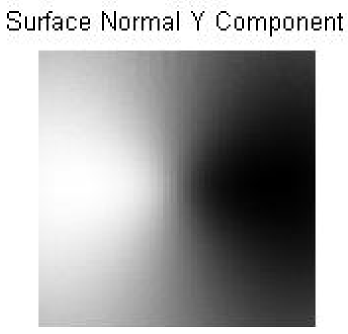
b) 4images_needlemap (2D view by quiver function)

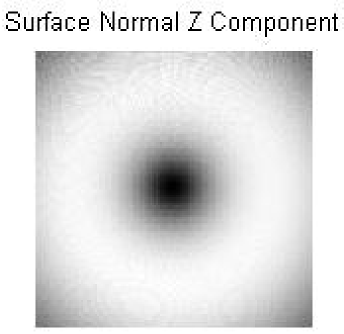




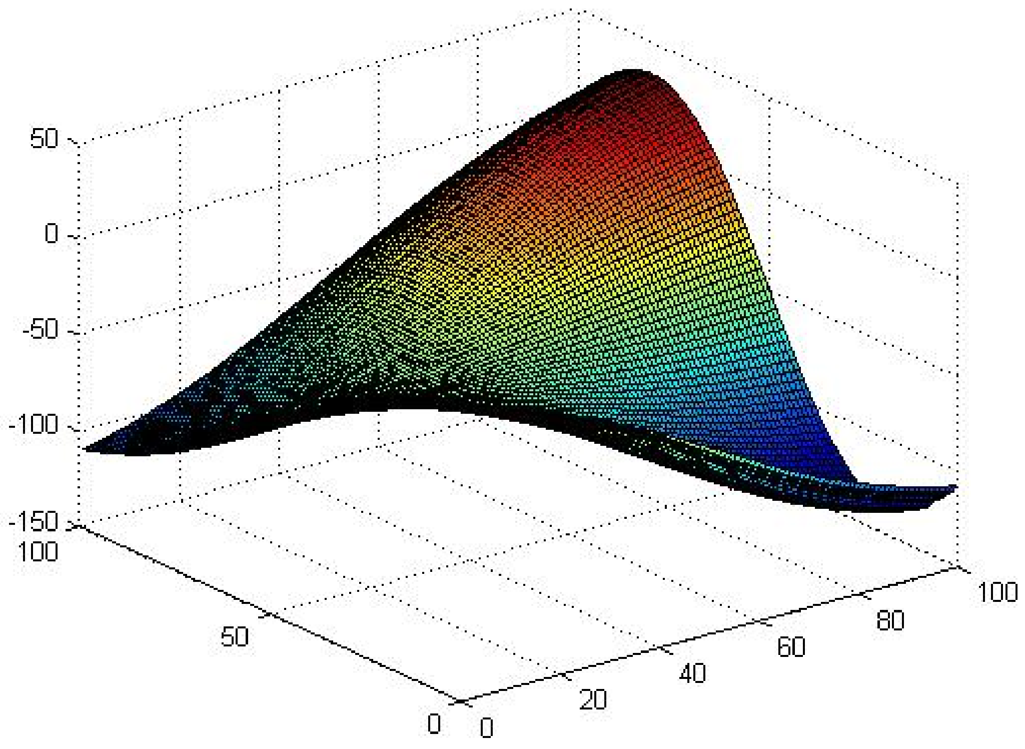
c) 4images_three_surfacenormal





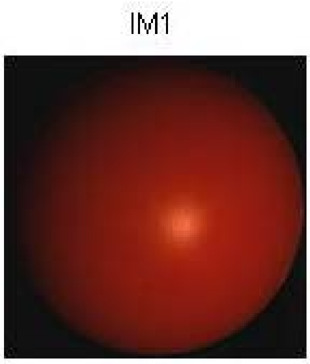


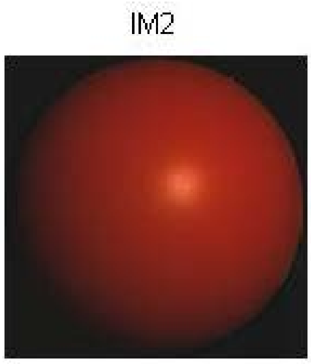
d) 4images_depthmap

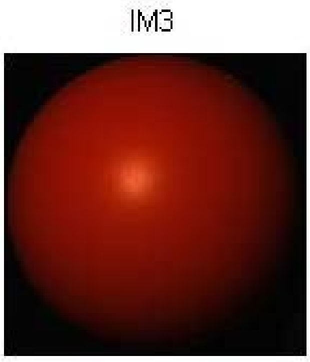


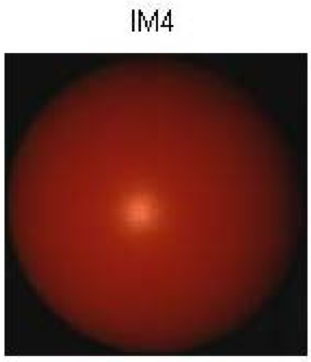
Part 2:

a) The original images (in RGB colorspace) were recovered using the function mat2gray() and then imshow(). The original sphere images are shown below.

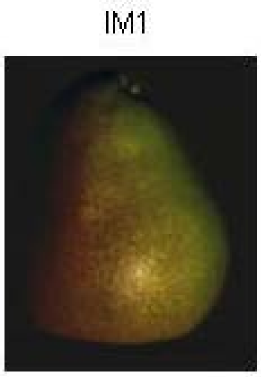


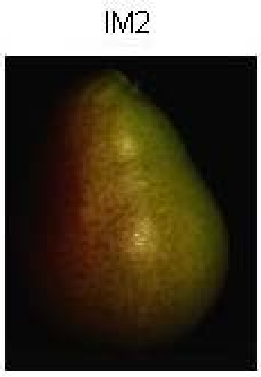


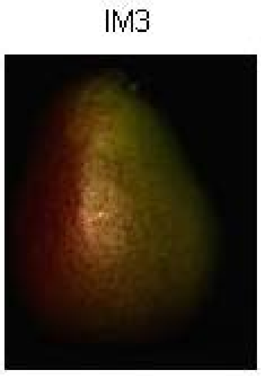


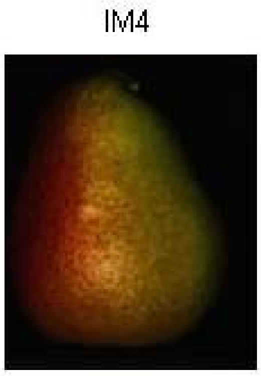


The following are the original pear images.







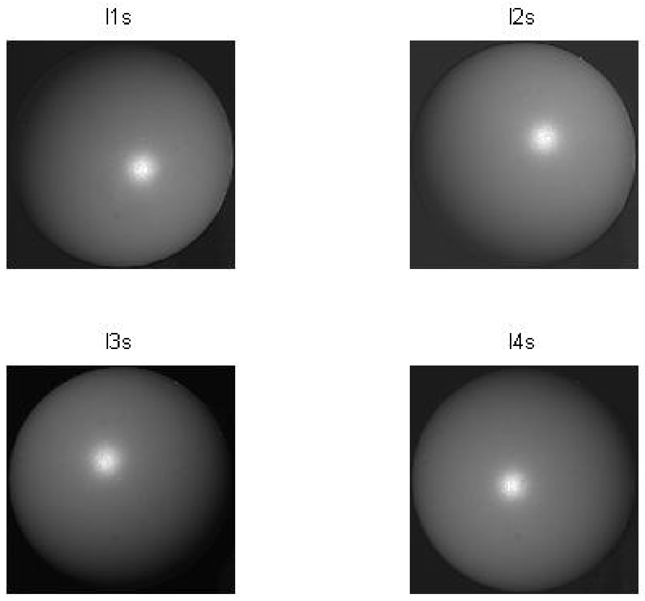


b) To convert the images from RGB space to SUV space, we first calculated the rotation matrices using the fact that the RGB coordinates rotate by the azimuthal angle (φ_s) and by elevation angle (θ_s) subsequently to align with the direction of the source color S. The new SUV image is then calculated by the equation $I_{SUV} = [R]I$, where $[R] \in SO(3)$ that satisfies $[R]S = [1, 0, 0]$. The two rotation matrices we got are:

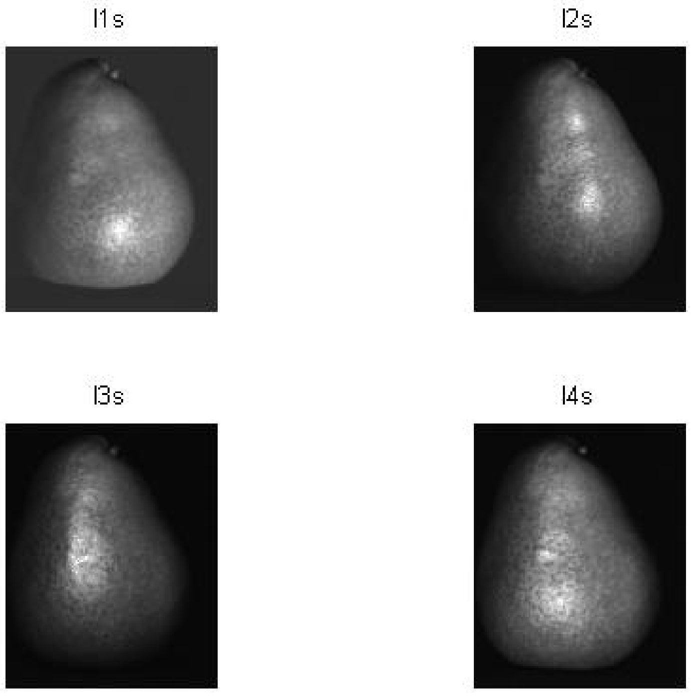
$$R_y = \begin{bmatrix} 0.7601 & 0 & 0.6498 \\ 0 & 1.0000 & 0 \\ -0.6498 & 0 & 0.7601 \end{bmatrix} \qquad R_z = \begin{bmatrix} 0.8232 & 0.5678 & 0 \\ -0.5678 & 0.8232 & 0 \\ 0 & 0 & 1.0000 \end{bmatrix}$$

The recovered S channel of the given images are shown below.

S channel of sphere:

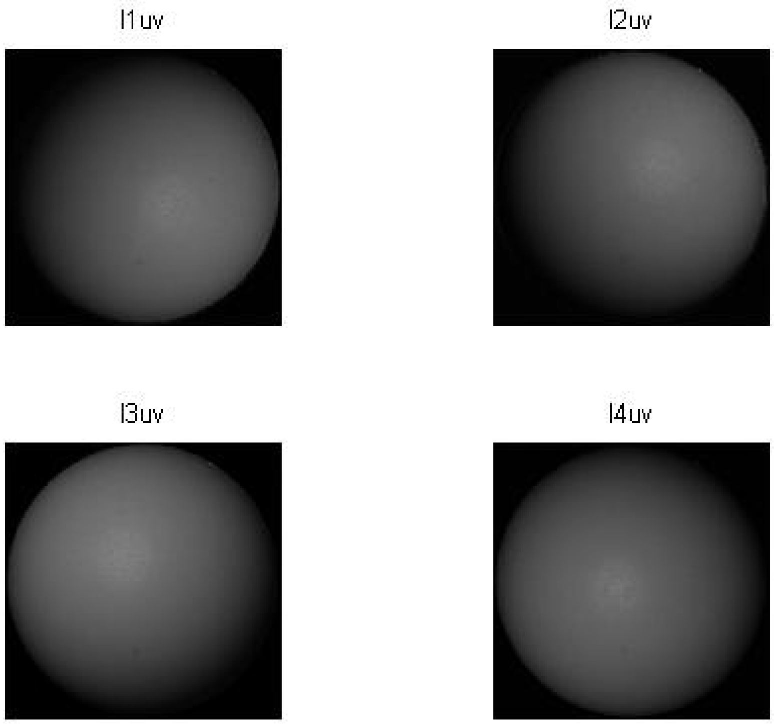


S channel of pear

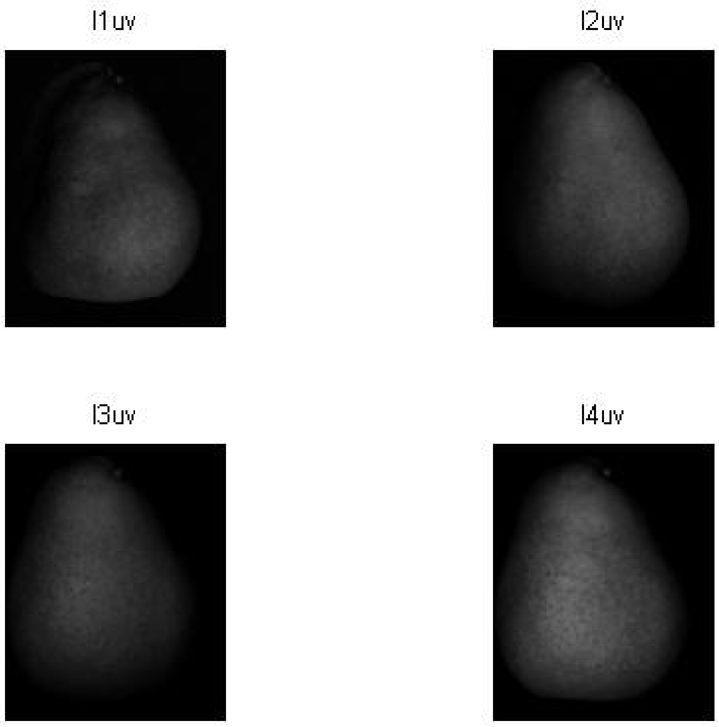


c) The diffuse part of the images was computed by the equation G = $\sqrt{U^2 + V^2}$. The recovered diffuse part of the images is shown as follows.

Diffuse part of sphere



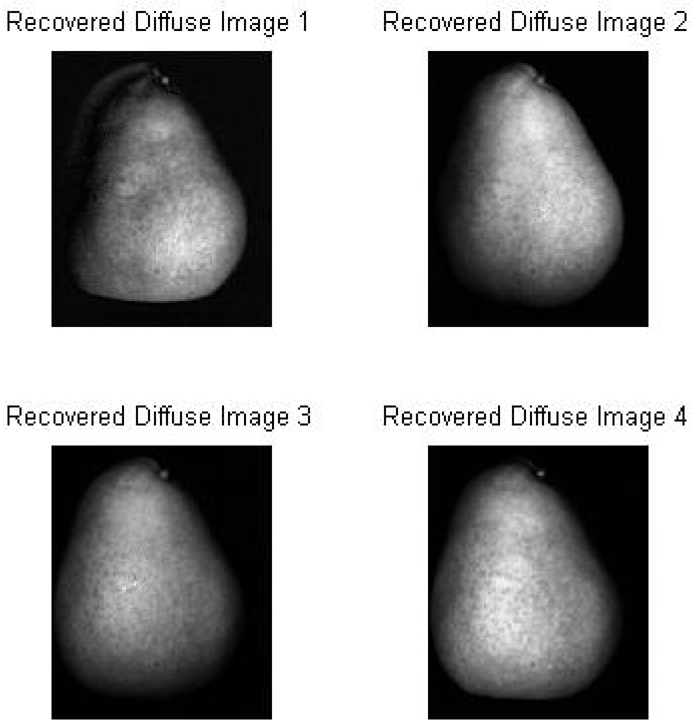
Diffuse part of pear



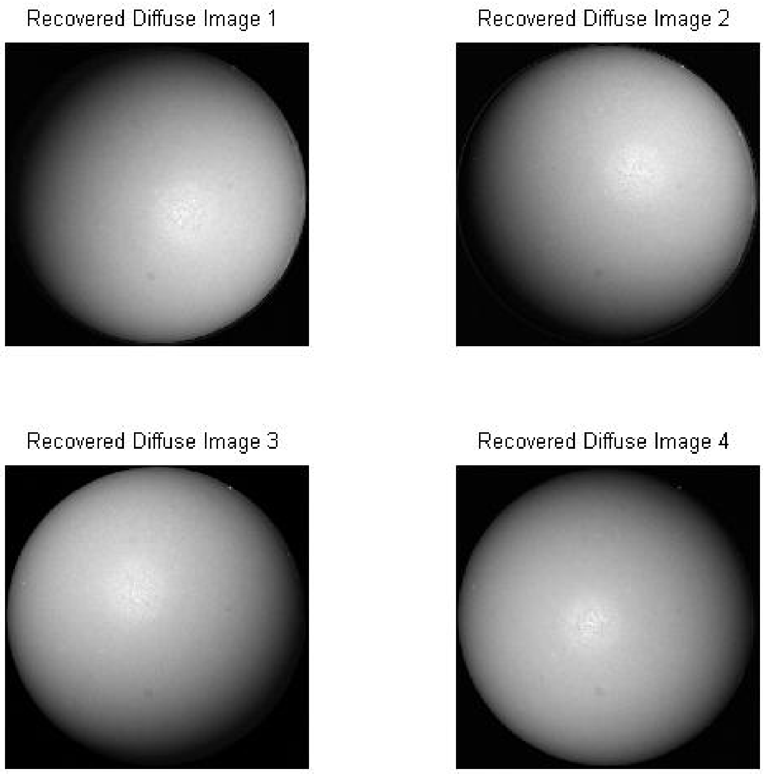
Part 3

a) Same calculations as in part2 were done here to find the diffuse images. The recovered diffuse images are shown below.

   1) Pear diffuse images:


Recovered Diffuse Image 1
Recovered Diffuse Image 2
Recovered Diffuse Image 3
Recovered Diffuse Image 4

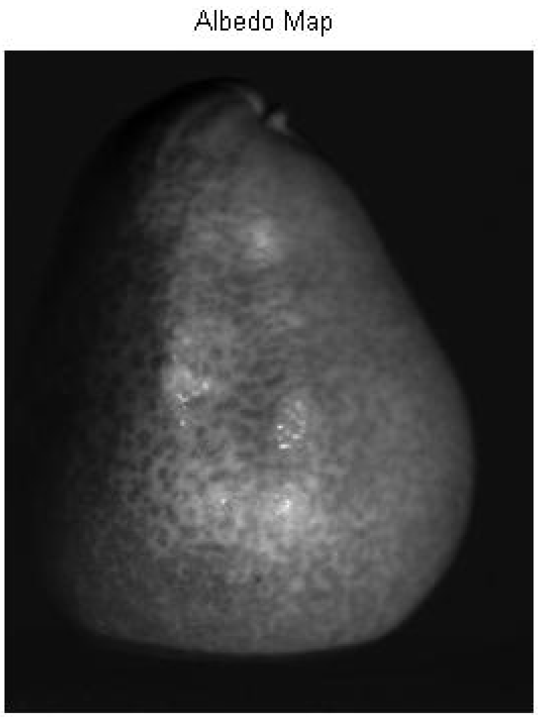
   2) Sphere diffuse images:


Recovered Diffuse Image 1
Recovered Diffuse Image 2
Recovered Diffuse Image 3
Recovered Diffuse Image 4

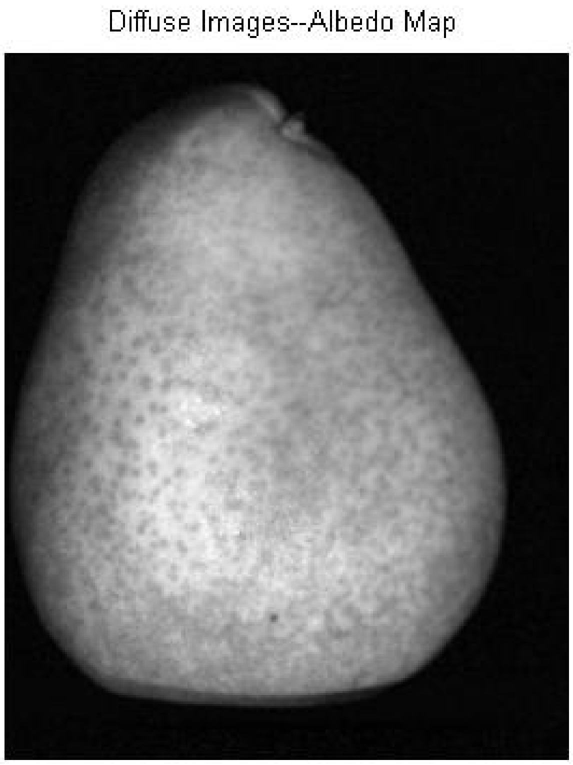
b) The albedo maps and height maps were estimated using the method described in part 1. The following images are the results for pear images and sphere images respectively.

Albedo maps for pear images

1) Original pear image:

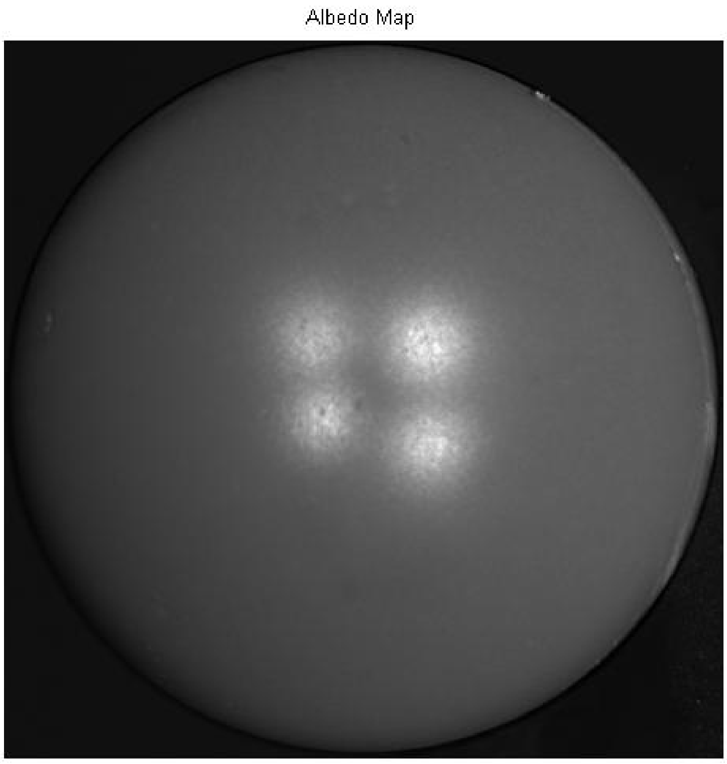


2) Diffuse pear image:

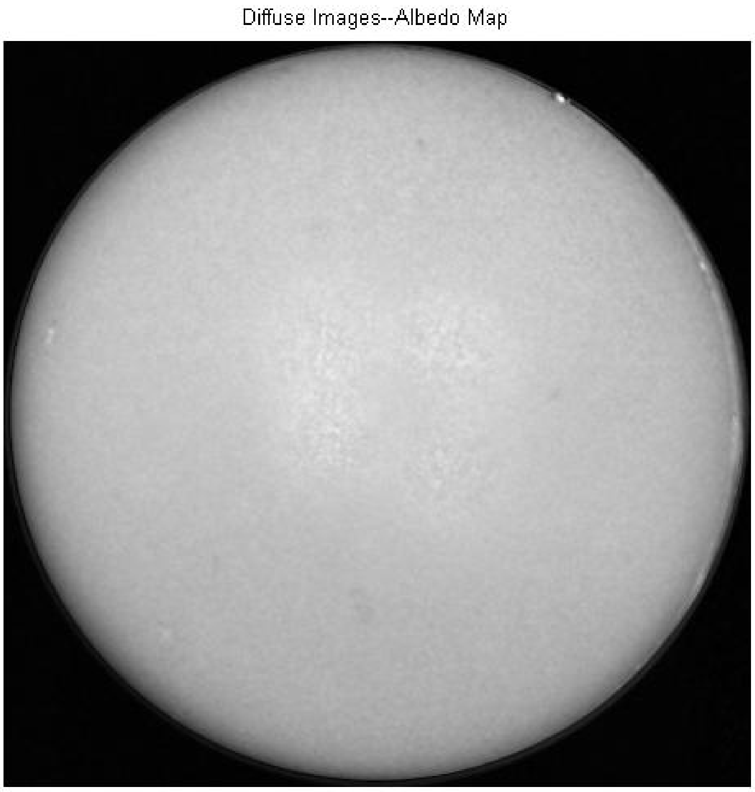


Albedo maps for sphere images

1) Original sphere image:



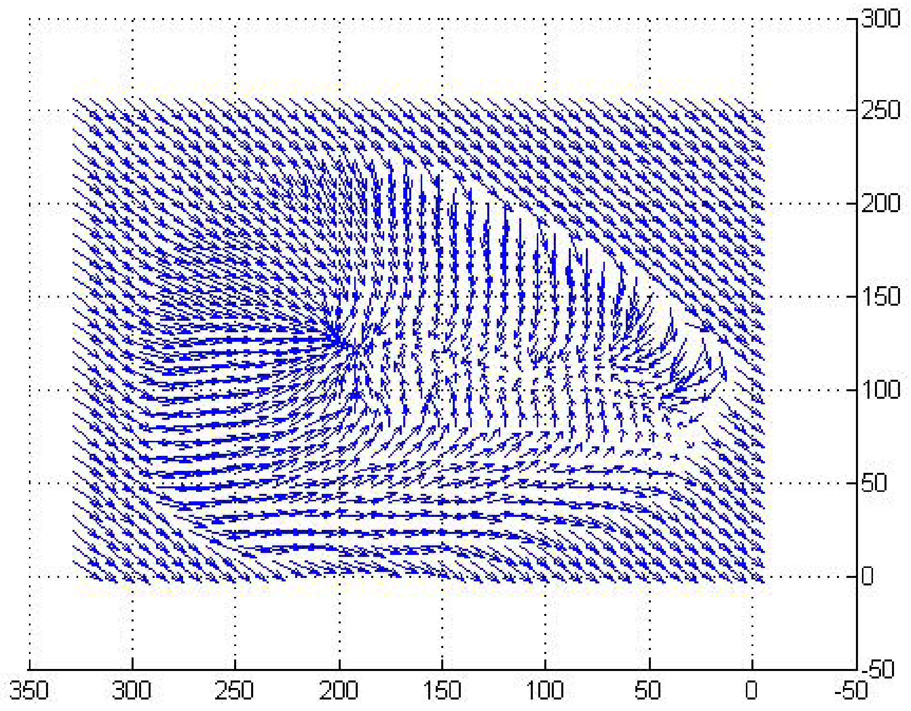
2) Diffuse sphere image:

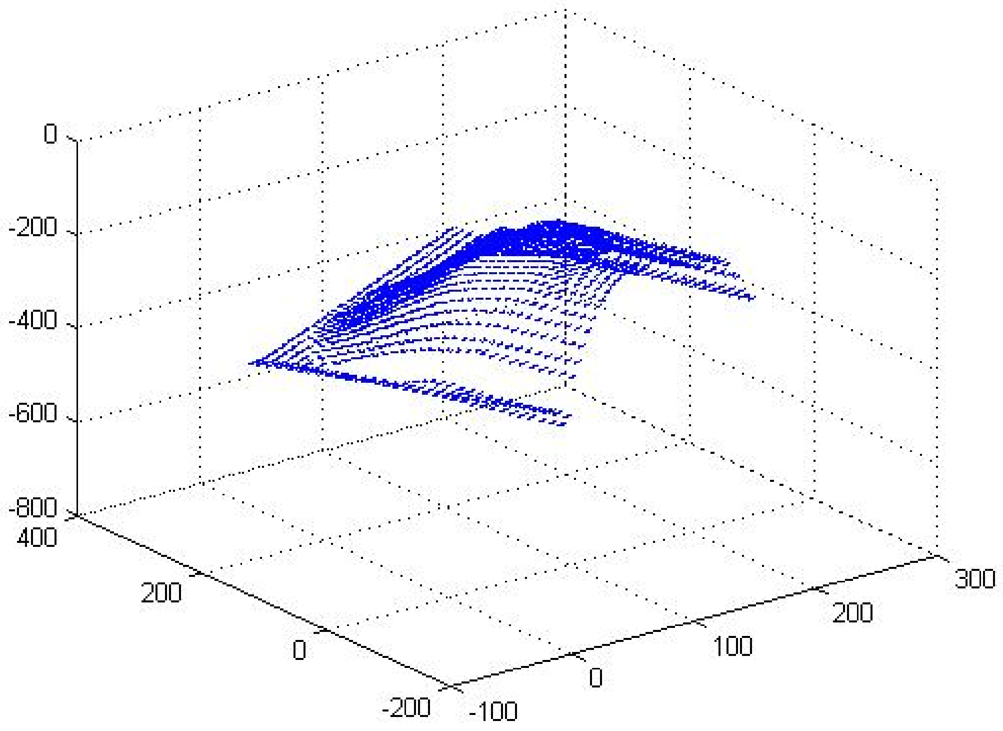


c) Needle maps and surface normals

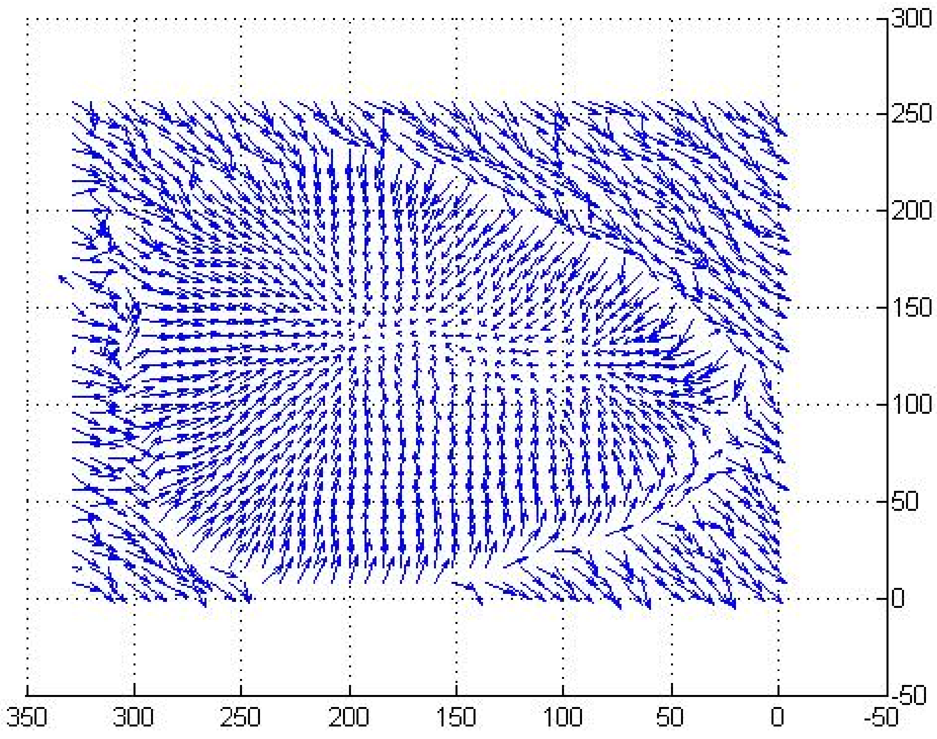
Neddle maps for pear images:

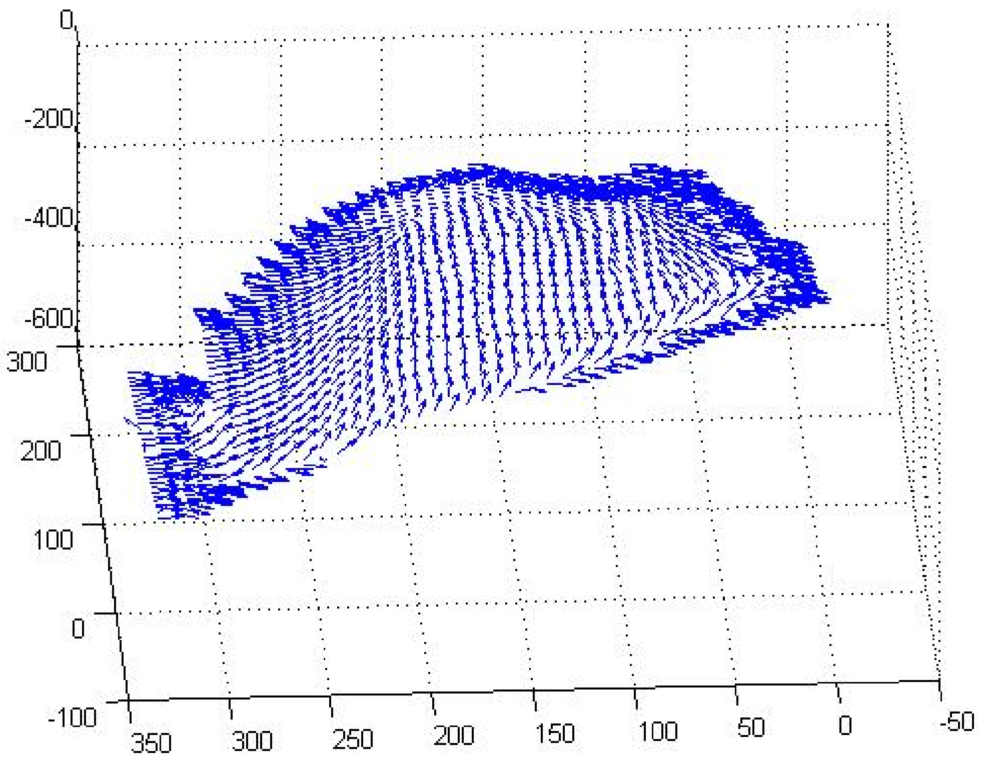
1) Original pear image needle maps (2D and 3D views)





2) Diffuse pear image (2D and 3D views)

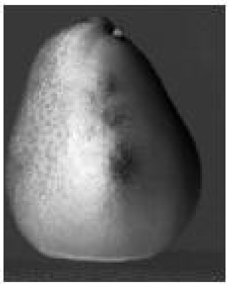




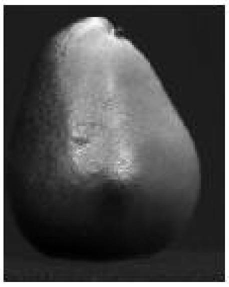
Surface normals for pear images

1) Original pear image:

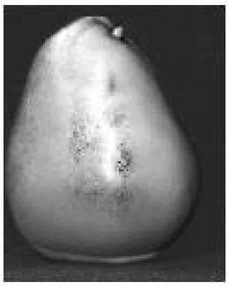
Surface Normal X Component



Surface Normal Y Component

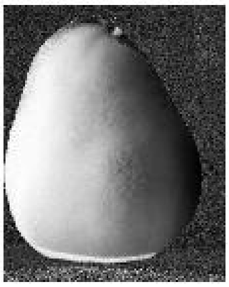


Surface Normal Z Component

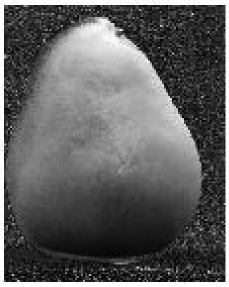


2) Diffuse pear image

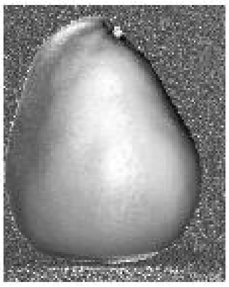
Diffuse Images--Nx Component



Diffuse Images--Ny Component

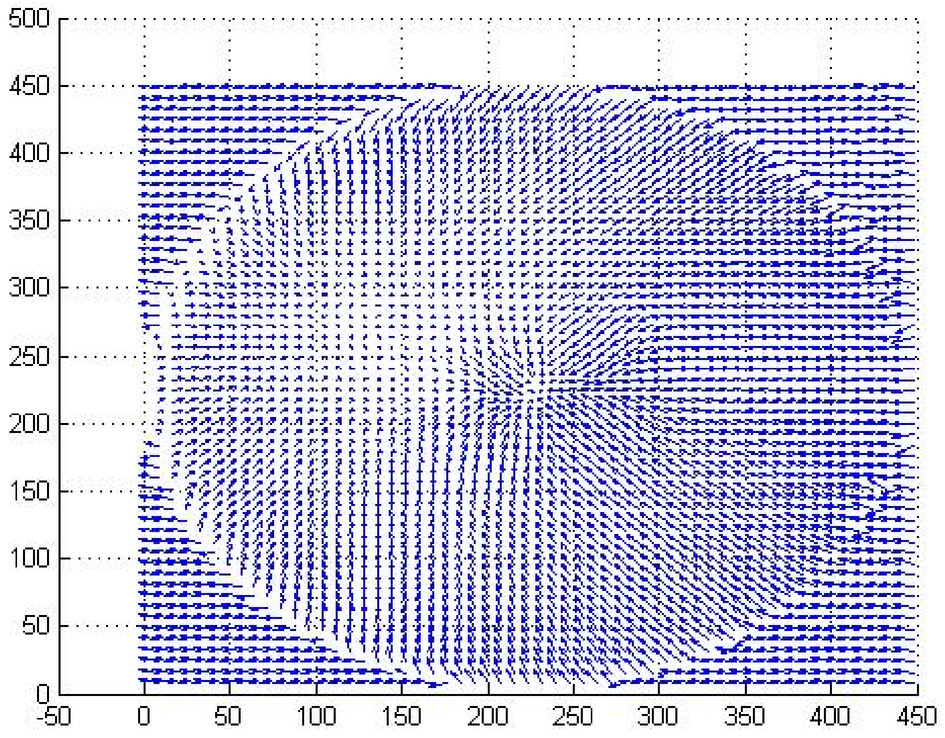


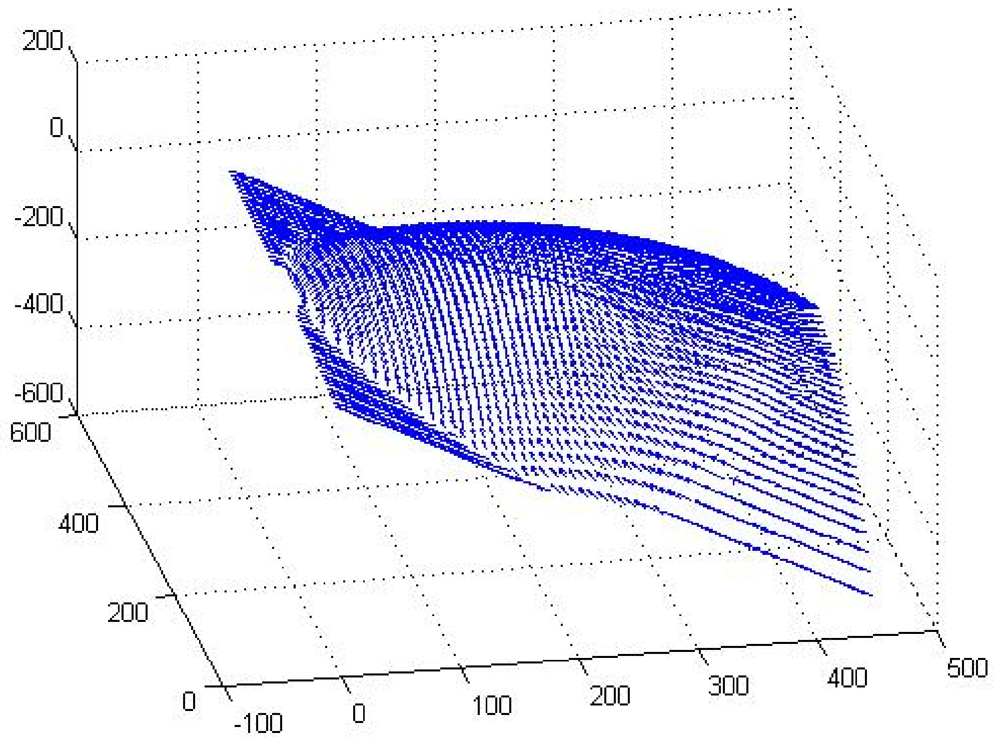
Diffuse Images--Nz Component



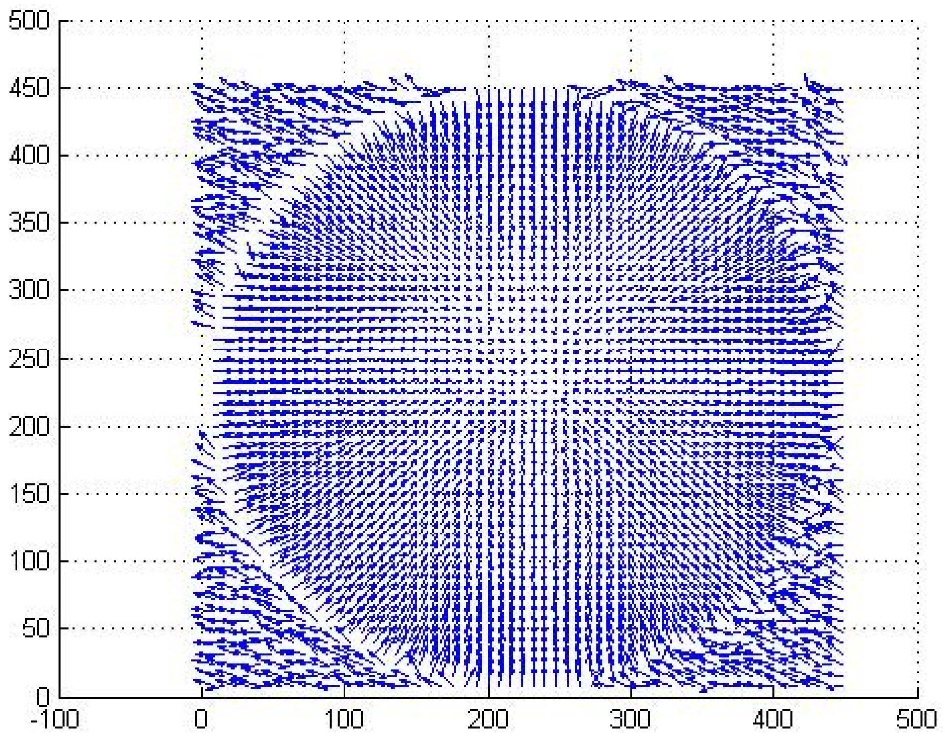
Needle maps for sphere images

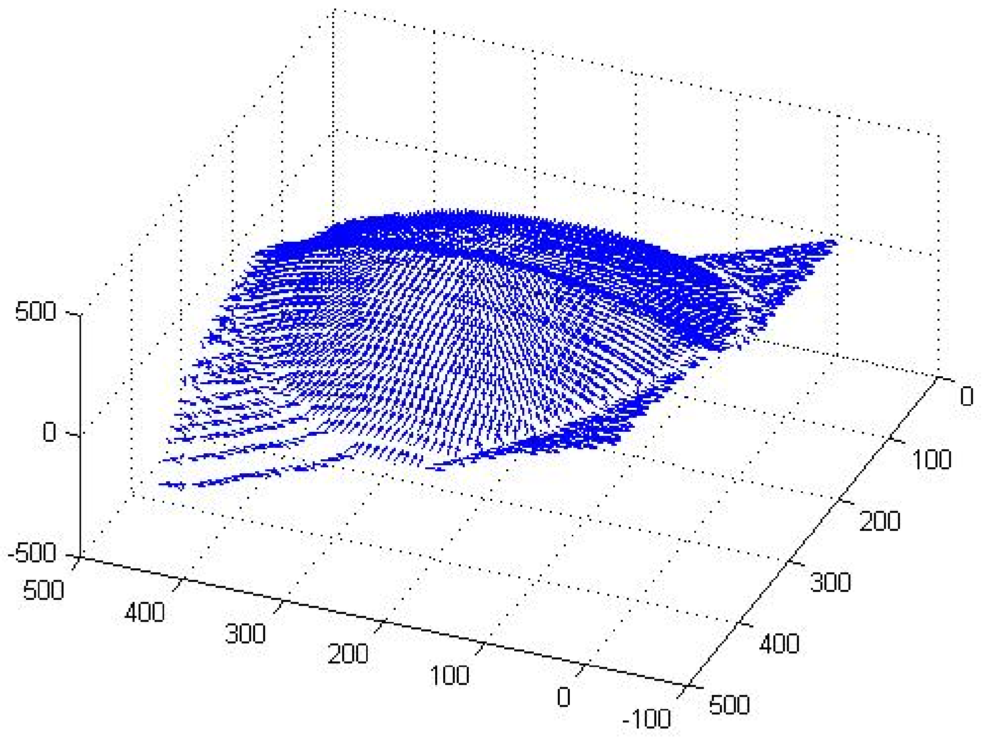
1) Original sphere image (2D and 3D views)





2) Diffuse sphere image (2D and 3D views)

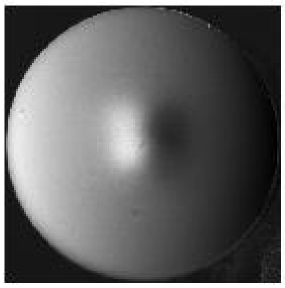




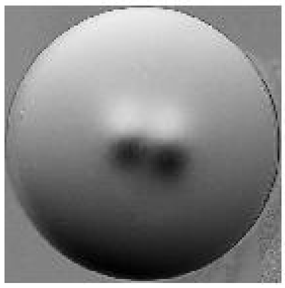
Surface normals for sphere images

1) Original sphere image

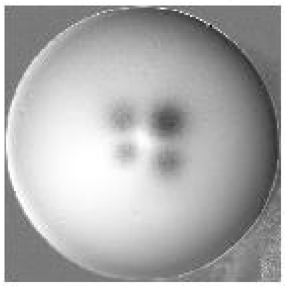
Surface Normal X Component



Surface Normal Y Component

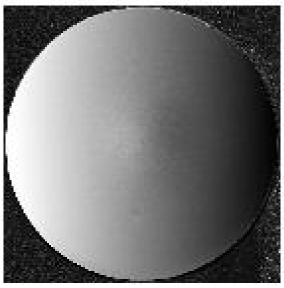


Surface Normal Z Component

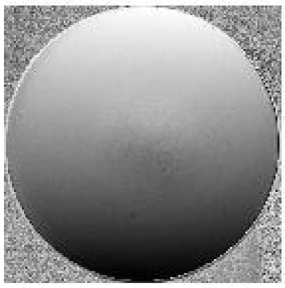


2) Diffuse sphere image:

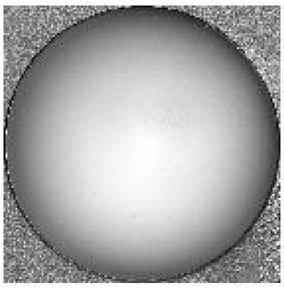
Diffuse Images--Nx Component



Diffuse Images--Ny Component

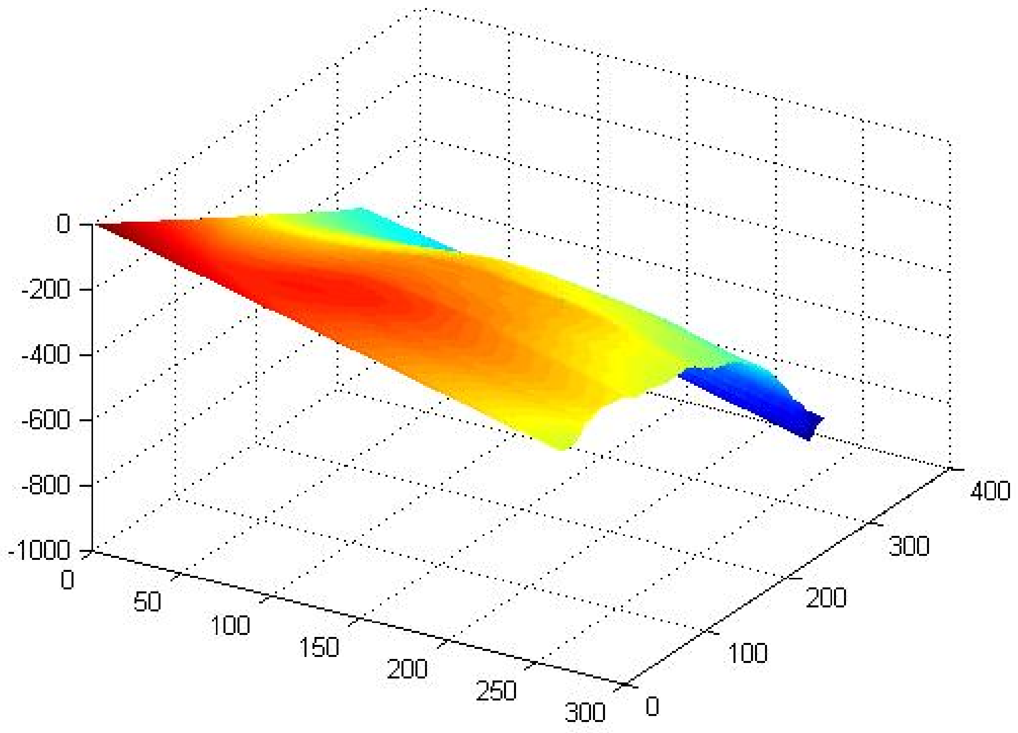


Diffuse Images--Nz Component

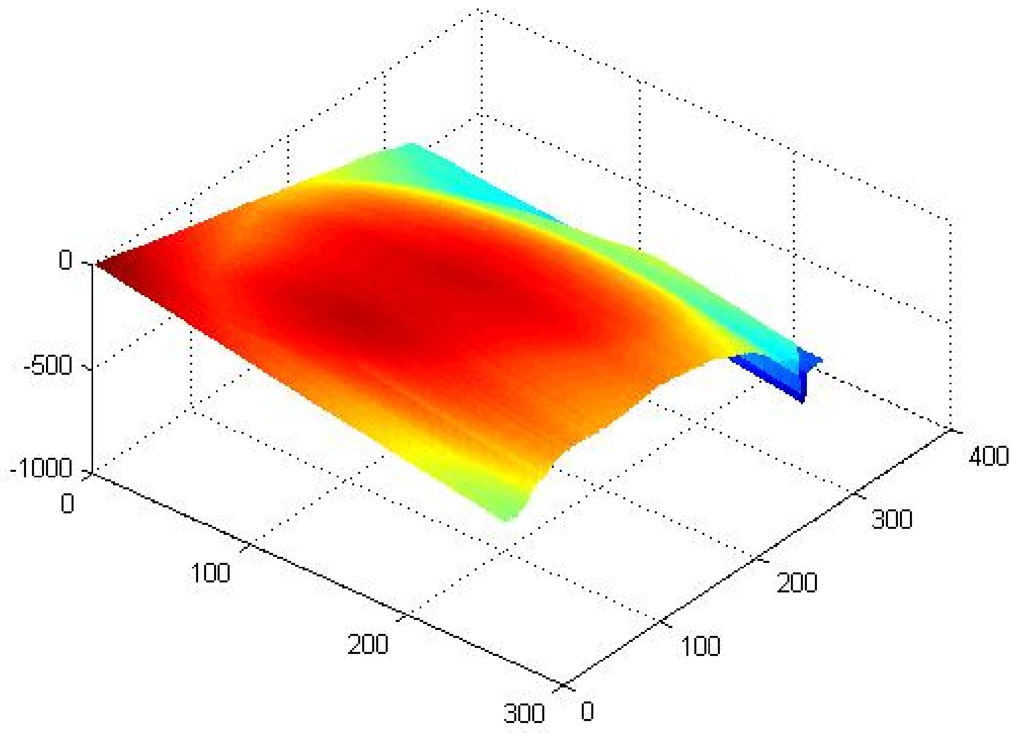


d) Depth map for pear images

1) Original pear image

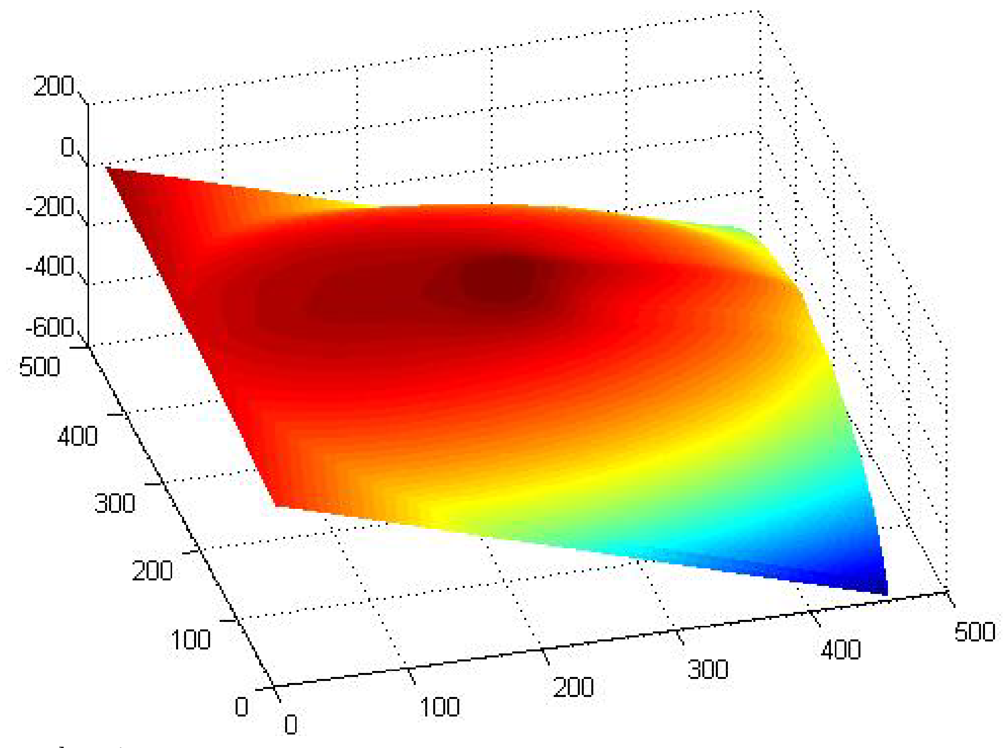


Diffuse pear image

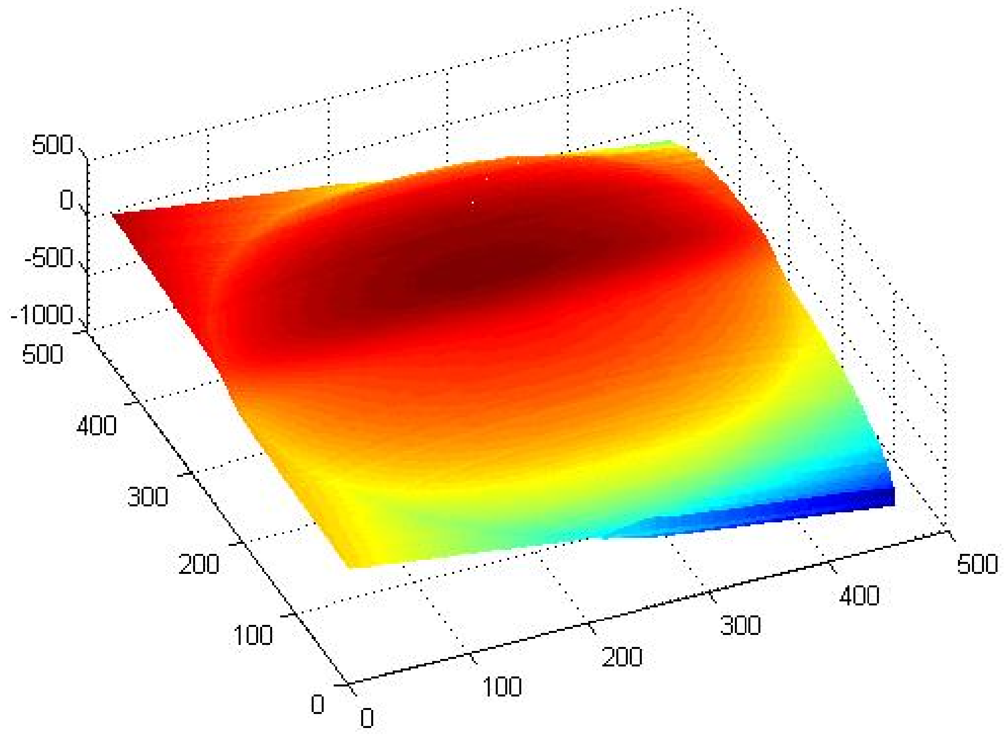


Depth map for sphere images

1) Original sphere image:



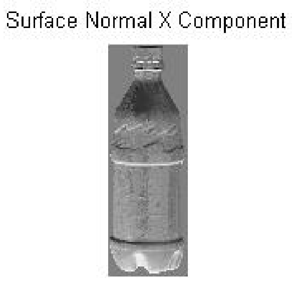
Diffuse sphere image:

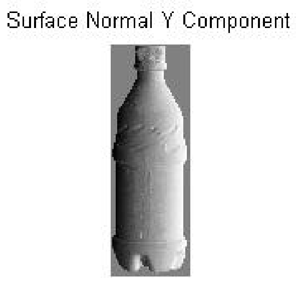


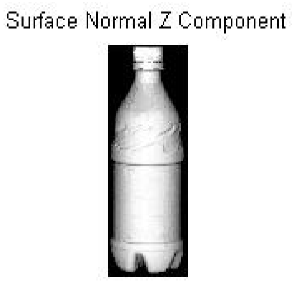
Part 4

To find the surface normals of the bottle, we first constructed a mathematical sphere using the provided sphere mask image. We then used the createns() and knnsearch() functions in MATLAB to find the corresponding image points on the spheres. The surface normals on these image points were used as those of the bottle image. The height map was computed by the naïve integration method. Image results are shown below.

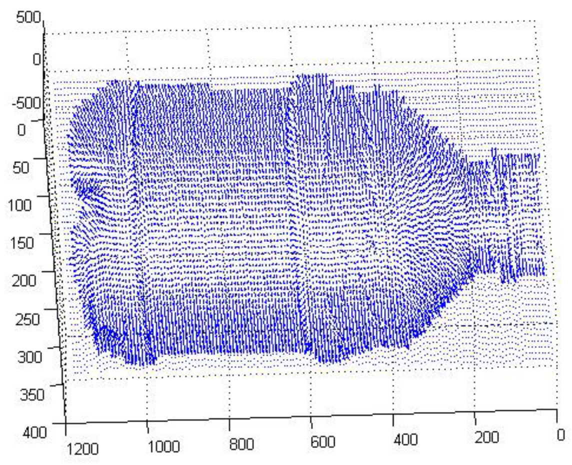
a) Surface normals of the bottle

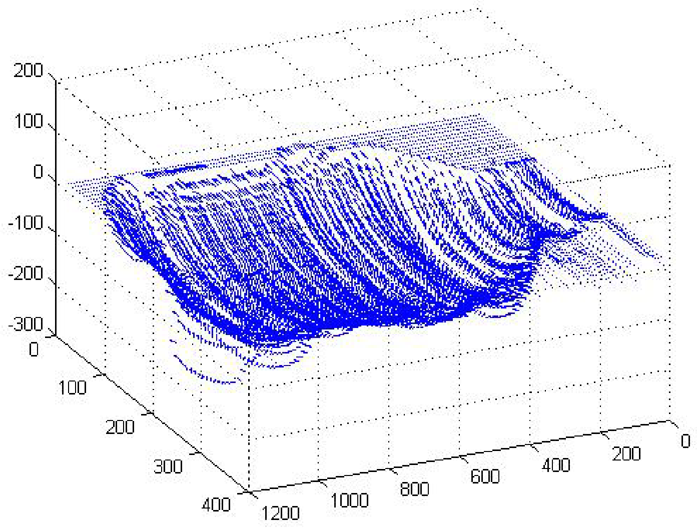




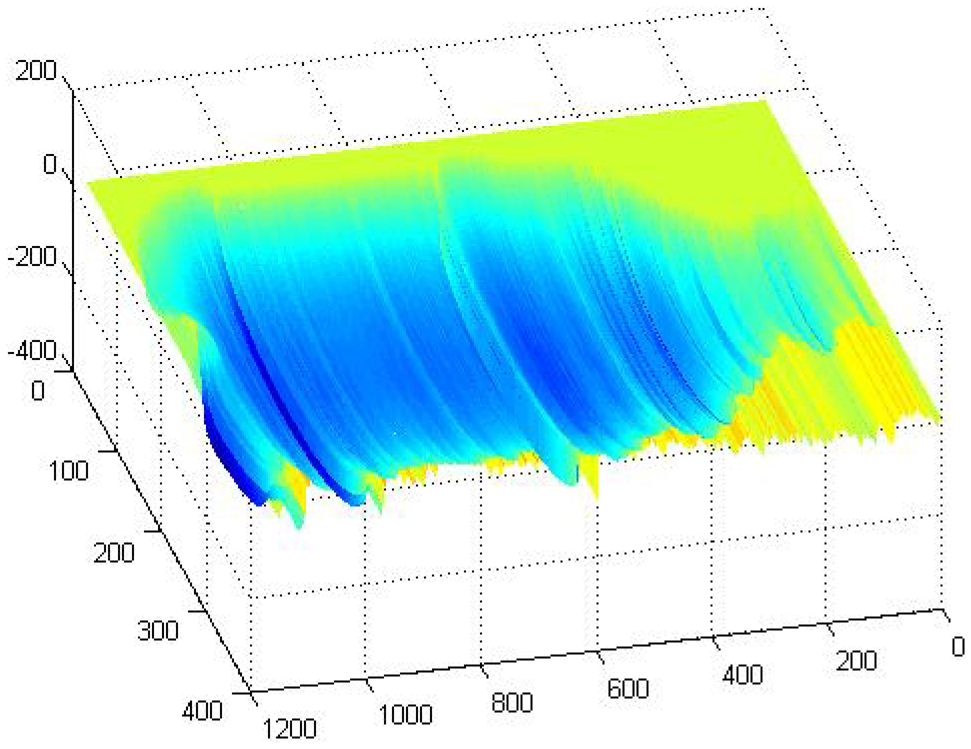


b) Needle map (2D and 3D views)





c) Depth map of the bottle



d) What are the differences in assumptions made (with respect to light source, BRDF, etc.) for this algorithm compared to the algorithm from Part 3?

In part 3, the assumption is that the color of the illuminant is constant over the surface. The color space rotation method applies to non-diffuse materials and does not require an explicit BRDF model or reference object. In this method, the light directions are known. While in part 4, the light sources under which the images are taken are unknown to us. We assume that any two surface points with the same normal have the same color by orientation-consistency cue. To satisfy orientation-consistency, we also assume that both points have the same BRDF, the light sources are directional, the camera is orthographic, and there are no shadows, interreflections, transparency effects, or other non-local effects that do not depend purely on the BRDF of a single point.

Attached code:

Part 1:

findNormal.m

```
function [normal, alb] = FindNormal(I, L)
    I = I';
    g = L\I;
    alb = norm(g);
    normal = g/alb;
```

hw2_Q5_p1_3images.m

```
load synthetic_data.mat
ims(:,:,1) = im1;
ims(:,:,2) = im2;
ims(:,:,3) = im4;
n = size(ims);
for i = 1:n(1)
for j = 1:n(2)
sNormal(i,j,1) = 0.0;
sNormal(i,j,2) = 0.0;
sNormal(i,j,3) = 0.0;
albedo(i,j) = 0.0;
end
end
L = [l1; l2; l4];
for i = 1:n(1)
for j = 1:n(2)
for im = 1:3
I(im) = double(ims(i,j,im));
end
[normal, alb] = FindNormal(I, L);
sNormal(i,j,1) = normal(1);
sNormal(i,j,2) = normal(2);
sNormal(i,j,3) = normal(3);
albedo(i,j) = alb;
end
end

sn1 = sNormal(:,:,1);
sn2 = sNormal(:,:,2);
sn3 = sNormal(:,:,3);
```

```
for i = 1:n(1)
for j = 1:n(2)
p(i,j) = sNormal(i,j,1)./sNormal(i,j,3);
q(i,j) = sNormal(i,j,2)./sNormal(i,j,3);
end
end

Height(1,1) = 0;
for i = 2:n(1)
Height(i,1) = Height(i-1,1) + q(i-1,1);
end

for i = 1:n(1)
for j = 2:n(2)
Height(i,j) = Height(i,j-1) + p(i,j-1);
end
end

[X Y ] = meshgrid(1:n(2), 1:n(1));

for i = 1:uint8(n(1)/4)
for j = 1:uint8(n(2)/4)
x(i,j) = X(i*4, j*4);
y(i,j) = Y(i*4, j*4);
ps(i,j) = p(i*4, j*4);
qs(i,j) = q(i*4, j*4);
height(i,j) = Height(i*4, j*4);
sn1_s(i,j) = sn1(i*4, j*4);
sn2_s(i,j) = sn2(i*4, j*4);
sn3_s(i,j) = sn3(i*4, j*4);
end
end


figure, imshow(albedo, [min(albedo(:)) max(albedo(:))])
hold on
title('Albedo Map');
hold off

figure;
subplot(2,2,1), imshow(sn1, [min(sn1(:)) max(sn1(:))])
hold on
title('Surface Normal X Component');
hold off
```

```
subplot(2,2,2), imshow(sn2, [min(sn2(:)) max(sn2(:))])
hold on
title('Surface Normal Y Component');
hold off

subplot(2,2,3), imshow(sn3, [min(sn3(:)) max(sn3(:))])
hold on
title('Surface Normal Z Component');
hold off

figure, quiver3(x,y,height, sn1_s, sn2_s, sn3_s)
figure, surf(Height)

hw2_Q5_p1_4images.m
load synthetic_data.mat
ims(:,:,1) = im1;
ims(:,:,2) = im2;
ims(:,:,3) = im4;
n = size(ims);
for i = 1:n(1)
for j = 1:n(2)
sNormal(i,j,1) = 0.0;
sNormal(i,j,2) = 0.0;
sNormal(i,j,3) = 0.0;
albedo(i,j) = 0.0;
end
end
L = [l1; l2; l4];
for i = 1:n(1)
for j = 1:n(2)
for im = 1:3
I(im) = double(ims(i,j,im));
end
[normal, alb] = FindNormal(I, L);
sNormal(i,j,1) = normal(1);
sNormal(i,j,2) = normal(2);
sNormal(i,j,3) = normal(3);
albedo(i,j) = alb;
end
end

sn1 = sNormal(:,:,1);
sn2 = sNormal(:,:,2);
```

```
sn3 = sNormal(:,:,3);
for i = 1:n(1)
for j = 1:n(2)
p(i,j) = sNormal(i,j,1)./sNormal(i,j,3);
q(i,j) = sNormal(i,j,2)./sNormal(i,j,3);
end
end

Height(1,1) = 0;
for i = 2:n(1)
Height(i,1) = Height(i-1,1) + q(i-1,1);
end
for i = 1:n(1)
for j = 2:n(2)
Height(i,j) = Height(i,j-1) + p(i,j-1);
end
end

[X Y ] = meshgrid(1:n(2), 1:n(1));

for i = 1:uint8(n(1)/4)
for j = 1:uint8(n(2)/4)
x(i,j) = X(i*4, j*4);
y(i,j) = Y(i*4, j*4);
ps(i,j) = p(i*4, j*4);
qs(i,j) = q(i*4, j*4);
height(i,j) = Height(i*4, j*4);
sn1_s(i,j) = sn1(i*4, j*4);
sn2_s(i,j) = sn2(i*4, j*4);
sn3_s(i,j) = sn3(i*4, j*4);
end
end


figure, imshow(albedo, [min(albedo(:)) max(albedo(:))])
hold on
title('Albedo Map');
hold off

figure;
subplot(2,2,1), imshow(sn1, [min(sn1(:)) max(sn1(:))])
hold on
title('Surface Normal X Component');
hold off
```

```
subplot(2,2,2), imshow(sn2, [min(sn2(:)) max(sn2(:))])
hold on
title('Surface Normal Y Component');
hold off

subplot(2,2,3), imshow(sn3, [min(sn3(:)) max(sn3(:))])
hold on
title('Surface Normal Z Component');
hold off

figure, quiver3(x,y,height, sn1_s, sn2_s, sn3_s)
figure, surf(Height)
```

Part2:

hw2_Q5_p2_sphere

```
load specular-sphere.mat

theta=asin(c(2,1));
fi=atan(c(3,1)/c(1,1));
Ky=[cos(fi), 0, sin(fi); 0, 1, 0; -sin(fi), 0, cos(fi)];
Kz=[cos(-theta), -sin(-theta), 0; sin(-theta), cos(-theta), 0; 0, 0 ,1];

IM1=mat2gray(im1);
IM2=mat2gray(im2);
IM3=mat2gray(im3);
IM4=mat2gray(im4);

for i=1:455
for j=1:455
I1suv(i,j,:)=Kz*Ky*[IM1(i,j,1);IM1(i,j,2);IM1(i,j,3)];
end
end

I1s=I1suv(:,:,1);
imshow(I1s);
imshow(I1s,[0.0704,0.9310]);

for i=1:455
for j=1:455
I1uv(i,j)=sqrt(I1suv(i,j,2)*I1suv(i,j,2)+I1suv(i,j,3)*I1suv(i,j,3));
end
end
imshow(I1uv);
```

```
for i=1:455
for j=1:455
I2suv(i,j,:)=Kz*Ky*[IM2(i,j,1);IM2(i,j,2);IM2(i,j,3)];
end
end

I2s=I2suv(:,:,1);
imshow(I2s);
imshow(I2s,[0.0704,0.9310]);

for i=1:455
for j=1:455
I2uv(i,j)=sqrt(I2suv(i,j,2)*I2suv(i,j,2)+I2suv(i,j,3)*I2suv(i,j,3));
end
end
imshow(I2uv);

for i=1:455
for j=1:455
I3suv(i,j,:)=Kz*Ky*[IM3(i,j,1);IM3(i,j,2);IM3(i,j,3)];
end
end

I3s=I3suv(:,:,1);
imshow(I3s);
imshow(I3s,[0.0704,0.9310]);

for i=1:455
for j=1:455
I3uv(i,j)=sqrt(I3suv(i,j,2)*I3suv(i,j,2)+I3suv(i,j,3)*I3suv(i,j,3));
end
end
imshow(I3uv);

for i=1:455
for j=1:455
I4suv(i,j,:)=Kz*Ky*[IM4(i,j,1);IM4(i,j,2);IM4(i,j,3)];
end
end

I4s=I4suv(:,:,1);
imshow(I4s);
imshow(I4s,[0.0704,0.9310]);
```

```
for i=1:455
for j=1:455
I4uv(i,j)=sqrt(I4suv(i,j,2)*I4suv(i,j,2)+I4suv(i,j,3)*I4suv(i,j,3));
end
end
imshow(I4uv);


figure
subplot(2,2,1);
imshow(I1uv);
title('I1uv');
subplot(2,2,2);
imshow(I2uv);
title('I2uv');
subplot(2,2,3);
imshow(I3uv);
title('I3uv');
subplot(2,2,4);
imshow(I4uv);
title('I4uv');

figure
subplot(2,2,1);
imshow(I1s);
title('I1s');
subplot(2,2,2);
imshow(I2s);
title('I2s');
subplot(2,2,3);
imshow(I3s);
title('I3s');
subplot(2,2,4);
imshow(I4s);
title('I4s');

figure
subplot(2,2,1);
imshow(IM1);
title('IM1');
subplot(2,2,2);
imshow(IM2);
title('IM2');
subplot(2,2,3);
```

```
imshow(IM3);
title('IM3');
subplot(2,2,4);
imshow(IM4);
title('IM4');

hw2_Q5_p2_spear
load specular-pear.mat
theta=asin(c(2,1));
fi=atan(c(3,1)/c(1,1));
Ky=[cos(fi), 0, sin(fi); 0, 1, 0; -sin(fi), 0, cos(fi)];
Kz=[cos(-theta), -sin(-theta), 0; sin(-theta), cos(-theta), 0; 0, 0 ,1];

IM1=mat2gray(im1);
IM2=mat2gray(im2);
IM3=mat2gray(im3);
IM4=mat2gray(im4);

for i=1:328
for j=1:262
I1suv(i,j,:)=Kz*Ky*[IM1(i,j,1);IM1(i,j,2);IM1(i,j,3)];
end
end

I1s=I1suv(:,:,1);
imshow(I1s);
imshow(I1s,[0.0704,0.9310]);

for i=1:328
for j=1:262
I1uv(i,j)=sqrt(I1suv(i,j,2)*I1suv(i,j,2)+I1suv(i,j,3)*I1suv(i,j,3));
end
end
imshow(I1uv);

for i=1:328
for j=1:262
I2suv(i,j,:)=Kz*Ky*[IM2(i,j,1);IM2(i,j,2);IM2(i,j,3)];
end
end

I2s=I2suv(:,:,1);
imshow(I2s);
imshow(I2s,[0.0704,0.9310]);
```

```
for i=1:328
for j=1:262
I2uv(i,j)=sqrt(I2suv(i,j,2)*I2suv(i,j,2)+I2suv(i,j,3)*I2suv(i,j,3));
end
end
imshow(I2uv);

for i=1:328
for j=1:262
I3suv(i,j,:)=Kz*Ky*[IM3(i,j,1);IM3(i,j,2);IM3(i,j,3)];
end
end

I3s=I3suv(:,:,1);
imshow(I3s);
imshow(I3s,[0.0704,0.9310]);

for i=1:328
for j=1:262
I3uv(i,j)=sqrt(I3suv(i,j,2)*I3suv(i,j,2)+I3suv(i,j,3)*I3suv(i,j,3));
end
end
imshow(I3uv);

for i=1:328
for j=1:262
I4suv(i,j,:)=Kz*Ky*[IM4(i,j,1);IM4(i,j,2);IM4(i,j,3)];
end
end

I4s=I4suv(:,:,1);
imshow(I4s);
imshow(I4s,[0.0704,0.9310]);

for i=1:328
for j=1:262
I4uv(i,j)=sqrt(I4suv(i,j,2)*I4suv(i,j,2)+I4suv(i,j,3)*I4suv(i,j,3));
end
end
imshow(I4uv);

figure
subplot(2,2,1);
```

```
imshow(I1uv);
title('I1uv');
subplot(2,2,2);
imshow(I2uv);
title('I2uv');
subplot(2,2,3);
imshow(I3uv);
title('I3uv');
subplot(2,2,4);
imshow(I4uv);
title('I4uv');

figure
subplot(2,2,1);
imshow(I1s);
title('I1s');
subplot(2,2,2);
imshow(I2s);
title('I2s');
subplot(2,2,3);
imshow(I3s);
title('I3s');
subplot(2,2,4);
imshow(I4s);
title('I4s');

figure
subplot(2,2,1);
imshow(IM1);
title('IM1');
subplot(2,2,2);
imshow(IM2);
title('IM2');
subplot(2,2,3);
imshow(IM3);
title('IM3');
subplot(2,2,4);
imshow(IM4);
title('IM4');
```

Part3:

findNormal.m

```
function [normal, alb] = FindNormal(I, L)
```

```
    I = I';
    g = L\I;
    alb = norm(g);
    normal    = g/alb;

hw2_Q5_p3_pear
load specular-pear.mat
theta=asin(c(2,1));
fi=atan(c(3,1)/c(1,1));
Ky=[cos(fi), 0, sin(fi); 0, 1, 0; -sin(fi), 0, cos(fi)];
Kz=[cos(-theta), -sin(-theta), 0; sin(-theta), cos(-theta), 0; 0, 0 ,1];

IM1=mat2gray(im1);
IM2=mat2gray(im2);
IM3=mat2gray(im3);
IM4=mat2gray(im4);

for i=1:328
for j=1:262
I1suv(i,j,:)=Kz*Ky*[IM1(i,j,1);IM1(i,j,2);IM1(i,j,3)];
end
end

I1s=I1suv(:,:,1);
imshow(I1s,[0.0704,0.9310]);

for i=1:328
for j=1:262
I1uv(i,j)=sqrt(I1suv(i,j,2)*I1suv(i,j,2)+I1suv(i,j,3)*I1suv(i,j,3));
end
end

for i=1:328
for j=1:262
I2suv(i,j,:)=Kz*Ky*[IM2(i,j,1);IM2(i,j,2);IM2(i,j,3)];
end
end

I2s=I2suv(:,:,1);
imshow(I2s,[0.0704,0.9310]);

for i=1:328
for j=1:262
I2uv(i,j)=sqrt(I2suv(i,j,2)*I2suv(i,j,2)+I2suv(i,j,3)*I2suv(i,j,3));
```

```
end
end

for i=1:328
for j=1:262
I3suv(i,j,:)=Kz*Ky*[IM3(i,j,1);IM3(i,j,2);IM3(i,j,3)];
end
end

I3s=I3suv(:,:,1);
imshow(I3s,[0.0704,0.9310]);

for i=1:328
for j=1:262
I3uv(i,j)=sqrt(I3suv(i,j,2)*I3suv(i,j,2)+I3suv(i,j,3)*I3suv(i,j,3));
end
end

for i=1:328
for j=1:262
I4suv(i,j,:)=Kz*Ky*[IM4(i,j,1);IM4(i,j,2);IM4(i,j,3)];
end
end

I4s=I4suv(:,:,1);
imshow(I4s,[0.0704,0.9310]);

for i=1:328
for j=1:262
I4uv(i,j)=sqrt(I4suv(i,j,2)*I4suv(i,j,2)+I4suv(i,j,3)*I4suv(i,j,3));
end
end

subplot(2,2,1), imshow(I1uv, [min(I1uv(:)) max(I1uv(:))]);
hold on
title('Recovered Diffuse Image 1');
hold off
subplot(2,2,2), imshow(I2uv, [min(I2uv(:)) max(I2uv(:))]);
hold on
title('Recovered Diffuse Image 2');
hold off
subplot(2,2,3), imshow(I3uv, [min(I3uv(:)) max(I3uv(:))]);
hold on
title('Recovered Diffuse Image 3');
```

```
hold off
subplot(2,2,4), imshow(I4uv, [min(I4uv(:)) max(I4uv(:))]);
hold on
title('Recovered Diffuse Image 4');
hold off



ims(:,:,:,1) = IM1;
ims(:,:,:,2) = IM2;
ims(:,:,:,3) = IM3;
ims(:,:,:,4) = IM4;
n = size(ims);
for i = 1:n(1)
for j = 1:n(2)
sNormal(i,j,1) = 0.0;
sNormal(i,j,2) = 0.0;
sNormal(i,j,3) = 0.0;
albedo(i,j) = 0.0;
end
end

for im = 1:4
grayims(:,:,im) = rgb2gray(ims(:,:,:,im));
end

L = [l1; l2; l3; l4];

for i = 1:n(1)
for j = 1:n(2)
for im = 1:4
I(im) = double(grayims(i,j,im));
end
[normal, alb] = FindNormal(I, L);
sNormal(i,j,1) = normal(1);
sNormal(i,j,2) = normal(2);
sNormal(i,j,3) = normal(3);
albedo(i,j) = alb;
end
end

sn1 = sNormal(:,:,1);
sn2 = sNormal(:,:,2);
sn3 = sNormal(:,:,3);
```

```
for i = 1:n(1)
for j = 1:n(2)
p(i,j) = sNormal(i,j,1)./sNormal(i,j,3);
q(i,j) = sNormal(i,j,2)./sNormal(i,j,3);
end
end

Height(1,1) = 0;

for i = 2:n(1)
Height(i,1) = Height(i-1,1) + q(i-1,1);
end

for i = 1:n(1)
for j = 2:n(2)
Height(i,j) = Height(i,j-1) + p(i,j-1);
end
end

[X Y ] = meshgrid(1:n(2), 1:n(1));
for i = 1:floor(n(1)/8)
for j = 1:floor(n(2)/8)
x(i,j) = X(i*8, j*8);
y(i,j) = Y(i*8, j*8);
ps(i,j) = p(i*8, j*8);
qs(i,j) = q(i*8, j*8);
height(i,j) = Height(i*8, j*8);
sn1_s(i,j) = sn1(i*8, j*8);
sn2_s(i,j) = sn2(i*8, j*8);
sn3_s(i,j) = sn3(i*8, j*8);
end
end

figure
imshow(albedo, [min(albedo(:)) max(albedo(:))])
hold on
title('Albedo Map');
hold off

figure
subplot(2,2,1), imshow(sn1, [min(sn1(:)) max(sn1(:))])
hold on
title('Surface Normal X Component');
```

```
hold off

subplot(2,2,2), imshow(sn2, [min(sn2(:)) max(sn2(:))])
hold on
title('Surface Normal Y Component');
hold off

subplot(2,2,3), imshow(sn3, [min(sn3(:)) max(sn3(:))])
hold on
title('Surface Normal Z Component');
hold off

figure, quiver3(x,y,height, sn1_s, sn2_s, sn3_s)
figure, h=surf(Height)
set(h,'edgecolor','none');


for i = 1:328
for j = 1:262
sNormaluv(i,j,1) = 0.0;
sNormaluv(i,j,2) = 0.0;
sNormaluv(i,j,3) = 0.0;
albedo(i,j) = 0.0;
end
end


imsuv(:,:,1) = I1uv;
imsuv(:,:,2) = I2uv;
imsuv(:,:,3) = I3uv;
imsuv(:,:,4) = I4uv;

for i = 1:328
for j = 1:262
sNuv(i,j,1) = 0.0;
sNuv(i,j,2) = 0.0;
sNuv(i,j,3) = 0.0;
albedouv(i,j) = 0;
end
end

for i = 1:328
for j = 1:262
for im = 1:4
```

```
I(im) = double(imsuv(i,j,im));
end
[normaluv, albuv] = FindNormal(I, L);
sNuv(i,j,1) = normaluv(1);
sNuv(i,j,2) = normaluv(2);
sNuv(i,j,3) = normaluv(3);
albedouv(i,j) = albuv;
end
end

snuv1 = sNuv(:,:,1);
snuv2 = sNuv(:,:,2);
snuv3 = sNuv(:,:,3);

for i = 1:328
for j = 1:262
puv(i,j) = sNuv(i,j,1)/sNuv(i,j,3);
quv(i,j) = sNuv(i,j,2)/sNuv(i,j,3);
end
end

Heightuv(1,1) = 0;

for i = 2:328
Heightuv(i,1) = Heightuv(i-1,1) + quv(i-1,1);
end
for i = 1:328
for j = 2:262
Heightuv(i,j) = Heightuv(i,j-1) + puv(i,j-1);
end
end
[Xuv Yuv ] = meshgrid(1:262, 1:328);
for i = 1:floor(328/8)
for j = 1:floor(262/8)
xuv(i,j) = Xuv(i*8, j*8);
yuv(i,j) = Yuv(i*8, j*8);
psuv(i,j) = puv(i*8, j*8);
qsuv(i,j) = quv(i*8, j*8);
heightuv(i,j) = Heightuv(i*8, j*8);
snuv1_s(i,j) = snuv1(i*8, j*8);
snuv2_s(i,j) = snuv2(i*8, j*8);
snuv3_s(i,j) = snuv3(i*8, j*8);
end
end
```

```
figure
imshow(albedouv, [min(albedouv(:)) max(albedouv(:))])
hold on
title('Diffuse Images--Albedo Map');
hold off

figure
subplot(2,2,1), imshow(snuv1, [min(snuv1(:)) max(snuv1(:))])
hold on
title('Diffuse Images--Nx Component');
hold off

subplot(2,2,2), imshow(snuv2, [min(snuv2(:)) max(snuv2(:))])
hold on
title('Diffuse Images--Ny Component');
hold off

subplot(2,2,3), imshow(snuv3, [min(snuv3(:)) max(snuv3(:))])
hold on
title('Diffuse Images--Nz Component');
hold off

figure, quiver3(xuv,yuv,heightuv, snuv1_s, snuv2_s, snuv3_s)
figure, h=surf(Heightuv)
set(h,'edgecolor','none');
```

hw2_Q5_p3_sphere.m

```
load specular-sphere.mat
theta=asin(c(2,1));
fi=atan(c(3,1)/c(1,1));
Ky=[cos(fi), 0, sin(fi); 0, 1, 0; -sin(fi), 0, cos(fi)];
Kz=[cos(-theta), -sin(-theta), 0; sin(-theta), cos(-theta), 0; 0, 0 ,1];

IM1=mat2gray(im1);
IM2=mat2gray(im2);
IM3=mat2gray(im3);
IM4=mat2gray(im4);
sz = size(IM1);

for i=1:sz(1)
for j=1:sz(2)
I1suv(i,j,:)=Kz*Ky*[IM1(i,j,1);IM1(i,j,2);IM1(i,j,3)];
```

```
end
end

I1s=I1suv(:,:,1);
imshow(I1s,[0.0704,0.9310]);

for i=1:sz(1)
for j=1:sz(2)
I1uv(i,j)=sqrt(I1suv(i,j,2)*I1suv(i,j,2)+I1suv(i,j,3)*I1suv(i,j,3));
end
end

for i=1:sz(1)
for j=1:sz(2)
I2suv(i,j,:)=Kz*Ky*[IM2(i,j,1);IM2(i,j,2);IM2(i,j,3)];
end
end

I2s=I2suv(:,:,1);
imshow(I2s,[0.0704,0.9310]);

for i=1:sz(1)
for j=1:sz(2)
I2uv(i,j)=sqrt(I2suv(i,j,2)*I2suv(i,j,2)+I2suv(i,j,3)*I2suv(i,j,3));
end
end

for i=1:sz(1)
for j=1:sz(2)
I3suv(i,j,:)=Kz*Ky*[IM3(i,j,1);IM3(i,j,2);IM3(i,j,3)];
end
end

I3s=I3suv(:,:,1);
imshow(I3s,[0.0704,0.9310]);

for i=1:sz(1)
for j=1:sz(2)
I3uv(i,j)=sqrt(I3suv(i,j,2)*I3suv(i,j,2)+I3suv(i,j,3)*I3suv(i,j,3));
end
end

for i=1:sz(1)
for j=1:sz(2)
```

```
I4suv(i,j,:)=Kz*Ky*[IM4(i,j,1);IM4(i,j,2);IM4(i,j,3)];
end
end

I4s=I4suv(:,:,1);
imshow(I4s,[0.0704,0.9310]);

for i=1:sz(1)
for j=1:sz(2)
I4uv(i,j)=sqrt(I4suv(i,j,2)*I4suv(i,j,2)+I4suv(i,j,3)*I4suv(i,j,3));
end
end

subplot(2,2,1), imshow(I1uv, [min(I1uv(:)) max(I1uv(:))]);
hold on
title('Recovered Diffuse Image 1');
hold off
subplot(2,2,2), imshow(I2uv, [min(I2uv(:)) max(I2uv(:))]);
hold on
title('Recovered Diffuse Image 2');
hold off
subplot(2,2,3), imshow(I3uv, [min(I3uv(:)) max(I3uv(:))]);
hold on
title('Recovered Diffuse Image 3');
hold off
subplot(2,2,4), imshow(I4uv, [min(I4uv(:)) max(I4uv(:))]);
hold on
title('Recovered Diffuse Image 4');
hold off



ims(:,:,:,1) = IM1;
ims(:,:,:,2) = IM2;
ims(:,:,:,3) = IM3;
ims(:,:,:,4) = IM4;
n = size(ims);
for i = 1:n(1)
for j = 1:n(2)
sNormal(i,j,1) = 0.0;
sNormal(i,j,2) = 0.0;
sNormal(i,j,3) = 0.0;
albedo(i,j) = 0.0;
end
```

```
end

for im = 1:4
grayims(:,:,im) = rgb2gray(ims(:,:,:,im));
end

L = [l1; l2; l3; l4];

for i = 1:n(1)
for j = 1:n(2)
for im = 1:4
I(im) = double(grayims(i,j,im));
end
[normal, alb] = FindNormal(I, L);
sNormal(i,j,1) = normal(1);
sNormal(i,j,2) = normal(2);
sNormal(i,j,3) = normal(3);
albedo(i,j) = alb;
end
end

sn1 = sNormal(:,:,1);
sn2 = sNormal(:,:,2);
sn3 = sNormal(:,:,3);

for i = 1:n(1)
for j = 1:n(2)
p(i,j) = sNormal(i,j,1)./sNormal(i,j,3);
q(i,j) = sNormal(i,j,2)./sNormal(i,j,3);
end
end

Height(1,1) = 0;

for i = 2:n(1)
Height(i,1) = Height(i-1,1) + q(i-1,1);
end

for i = 1:n(1)
for j = 2:n(2)
Height(i,j) = Height(i,j-1) + p(i,j-1);
end
end
```

```
[X Y ] = meshgrid(1:n(2), 1:n(1));
for i = 1:floor(n(1)/8)
for j = 1:floor(n(2)/8)
x(i,j) = X(i*8, j*8);
y(i,j) = Y(i*8, j*8);
ps(i,j) = p(i*8, j*8);
qs(i,j) = q(i*8, j*8);
height(i,j) = Height(i*8, j*8);
sn1_s(i,j) = sn1(i*8, j*8);
sn2_s(i,j) = sn2(i*8, j*8);
sn3_s(i,j) = sn3(i*8, j*8);
end
end

figure
imshow(albedo, [min(albedo(:)) max(albedo(:))])
hold on
title('Albedo Map');
hold off

figure
subplot(2,2,1), imshow(sn1, [min(sn1(:)) max(sn1(:))])
hold on
title('Surface Normal X Component');
hold off

subplot(2,2,2), imshow(sn2, [min(sn2(:)) max(sn2(:))])
hold on
title('Surface Normal Y Component');
hold off

subplot(2,2,3), imshow(sn3, [min(sn3(:)) max(sn3(:))])
hold on
title('Surface Normal Z Component');
hold off

figure, quiver3(x,y,height, sn1_s, sn2_s, sn3_s)
figure, h=surf(Height)
set(h,'edgecolor','none');


for i = 1:sz(1)
for j = 1:sz(2)
sNormaluv(i,j,1) = 0.0;
sNormaluv(i,j,2) = 0.0;
sNormaluv(i,j,3) = 0.0;
albedo(i,j) = 0.0;
end
end


imsuv(:,:,1) = I1uv;
imsuv(:,:,2) = I2uv;
imsuv(:,:,3) = I3uv;
imsuv(:,:,4) = I4uv;

for i = 1:sz(1)
for j = 1:sz(2)
sNuv(i,j,1) = 0.0;
sNuv(i,j,2) = 0.0;
sNuv(i,j,3) = 0.0;
albedouv(i,j) = 0;
end
end

for i = 1:sz(1)
for j = 1:sz(2)
for im = 1:4
I(im) = double(imsuv(i,j,im));
end
[normaluv, albuv] = FindNormal(I, L);
sNuv(i,j,1) = normaluv(1);
sNuv(i,j,2) = normaluv(2);
sNuv(i,j,3) = normaluv(3);
albedouv(i,j) = albuv;
end
end

snuv1 = sNuv(:,:,1);
snuv2 = sNuv(:,:,2);
snuv3 = sNuv(:,:,3);

for i = 1:sz(1)
for j = 1:sz(2)
puv(i,j) = sNuv(i,j,1)/sNuv(i,j,3);
quv(i,j) = sNuv(i,j,2)/sNuv(i,j,3);
end
end
```

```
Heightuv(1,1) = 0;

for i = 2:sz(1)
Heightuv(i,1) = Heightuv(i-1,1) + quv(i-1,1);
end
for i = 1:sz(1)
for j = 2:sz(2)
Heightuv(i,j) = Heightuv(i,j-1) + puv(i,j-1);
end
end
[Xuv Yuv ] = meshgrid(1:sz(2), 1:sz(1));
for i = 1:floor(sz(1)/8)
for j = 1:floor(sz(2)/8)
xuv(i,j) = Xuv(i*8, j*8);
yuv(i,j) = Yuv(i*8, j*8);
psuv(i,j) = puv(i*8, j*8);
qsuv(i,j) = quv(i*8, j*8);
heightuv(i,j) = Heightuv(i*8, j*8);
snuv1_s(i,j) = snuv1(i*8, j*8);
snuv2_s(i,j) = snuv2(i*8, j*8);
snuv3_s(i,j) = snuv3(i*8, j*8);
end
end

figure
imshow(albedouv, [min(albedouv(:)) max(albedouv(:))])
hold on
title('Diffuse Images--Albedo Map');
hold off

figure
subplot(2,2,1), imshow(snuv1, [min(snuv1(:)) max(snuv1(:))])
hold on
title('Diffuse Images--Nx Component');
hold off

subplot(2,2,2), imshow(snuv2, [min(snuv2(:)) max(snuv2(:))])
hold on
title('Diffuse Images--Ny Component');
hold off

subplot(2,2,3), imshow(snuv3, [min(snuv3(:)) max(snuv3(:))])
hold on
```

```
title('Diffuse Images--Nz Component');
hold off

figure, quiver3(xuv,yuv,heightuv, snuv1_s, snuv2_s, snuv3_s)
figure, h=surf(Heightuv)
set(h,'edgecolor','none');


Part 4:
smask = imread('spheremask.png');
bmask = imread('bottlemask.png');

MAX = [0,0];

  for i = 1:332
          k = 0;
  for j = 1:328;
          if (smask(i, j) == 255)
                k = k+1;
          end
  end
            if (k>MAX(1,1))
                  MAX(1,1) = k;
                  MAX(1,2) = i;
          end
  end

  CenterY = [0,0];

  for i = 1:328
          k = 0;
  for j = 1:332;
          if (smask(j, i) == 255)
                k = k+1;
          end
  end
            if (k>CenterY(1,1))
                  CenterY(1,1) = k;
                  CenterY(1,2) = i;
          end
  end

maskZ = zeros(332,328);
for i = 1:332
```

```
        for j = 1:328
                if((143^2 - (i-170)^2 - (j-162)^2)>0)
                maskZ(i,j) = sqrt(143^2 - (i-170)^2 - (j-162)^2);
                elseif((143^2 - (i-170)^2 - (j-162)^2)==0)
                maskZ(i,j) = 0;
                end
        end
end

for i = 1:332
        for j = 1:328
                maskN(i,j,1) = (i-170)/143;
                maskN(i,j,2) = (j-162)/143;
                maskN(i,j,3) = (maskZ(i,j)-0)/143;
        end
end

s1 = imread('sphere1.png');
s2 = imread('sphere2.png');
s3 = imread('sphere3.png');
s4 = imread('sphere4.png');
s5 = imread('sphere5.png');
s6 = imread('sphere6.png');
s7 = imread('sphere7.png');
s8 = imread('sphere8.png');
b1 = imread('bottle1.png');
b2 = imread('bottle2.png');
b3 = imread('bottle3.png');
b4 = imread('bottle4.png');
b5 = imread('bottle5.png');
b6 = imread('bottle6.png');
b7 = imread('bottle7.png');
b8 = imread('bottle8.png');

[row col] = find(smask);
a(:,1) = row(:);
a(:,2) = col(:);

maskNx = maskN(:,:,1);
maskNy = maskN(:,:,2);
maskNz = maskN(:,:,3);

for i = 1:65233
a(i,3) = maskNx(a(i,1),a(i,2));
```

```
a(i,4) = maskNy(a(i,1),a(i,2));
a(i,5) = maskNz(a(i,1),a(i,2));
end

ricut = zeros(65233,24);
for i = 1:65233
ricut(i,1) = s1(a(i,1),a(i,2),1);
ricut(i,2) = s1(a(i,1),a(i,2),2);
ricut(i,3) = s1(a(i,1),a(i,2),3);
ricut(i,4) = s2(a(i,1),a(i,2),1);
ricut(i,5) = s2(a(i,1),a(i,2),2);
ricut(i,6) = s2(a(i,1),a(i,2),3);
ricut(i,7) = s3(a(i,1),a(i,2),1);
ricut(i,8) = s3(a(i,1),a(i,2),2);
ricut(i,9) = s3(a(i,1),a(i,2),3);
ricut(i,10) = s4(a(i,1),a(i,2),1);
ricut(i,11) = s4(a(i,1),a(i,2),2);
ricut(i,12) = s4(a(i,1),a(i,2),3);
ricut(i,13) = s5(a(i,1),a(i,2),1);
ricut(i,14) = s5(a(i,1),a(i,2),2);
ricut(i,15) = s5(a(i,1),a(i,2),3);
ricut(i,16) = s6(a(i,1),a(i,2),1);
ricut(i,17) = s6(a(i,1),a(i,2),2);
ricut(i,18) = s6(a(i,1),a(i,2),3);
ricut(i,19) = s7(a(i,1),a(i,2),1);
ricut(i,20) = s7(a(i,1),a(i,2),2);
ricut(i,21) = s7(a(i,1),a(i,2),3);
ricut(i,22) = s8(a(i,1),a(i,2),1);
ricut(i,23) = s8(a(i,1),a(i,2),2);
ricut(i,24) = s8(a(i,1),a(i,2),3);
end
[row col] = find(bmask);
ab(:,1) = row(:);
ab(:,2) = col(:);
ticut = zeros(337894,24);
for i = 1:337894
ticut(i,1) = b1(ab(i,1),ab(i,2),1);
ticut(i,2) = b1(ab(i,1),ab(i,2),2);
ticut(i,3) = b1(ab(i,1),ab(i,2),3);
ticut(i,4) = b2(ab(i,1),ab(i,2),1);
ticut(i,5) = b2(ab(i,1),ab(i,2),2);
ticut(i,6) = b2(ab(i,1),ab(i,2),3);
ticut(i,7) = b3(ab(i,1),ab(i,2),1);
ticut(i,8) = b3(ab(i,1),ab(i,2),2);
ticut(i,9) = b3(ab(i,1),ab(i,2),3);
ticut(i,10) = b4(ab(i,1),ab(i,2),1);
ticut(i,11) = b4(ab(i,1),ab(i,2),2);
ticut(i,12) = b4(ab(i,1),ab(i,2),3);
ticut(i,13) = b5(ab(i,1),ab(i,2),1);
ticut(i,14) = b5(ab(i,1),ab(i,2),2);
ticut(i,15) = b5(ab(i,1),ab(i,2),3);
ticut(i,16) = b6(ab(i,1),ab(i,2),1);
ticut(i,17) = b6(ab(i,1),ab(i,2),2);
ticut(i,18) = b6(ab(i,1),ab(i,2),3);
ticut(i,19) = b7(ab(i,1),ab(i,2),1);
ticut(i,20) = b7(ab(i,1),ab(i,2),2);
ticut(i,21) = b7(ab(i,1),ab(i,2),3);
ticut(i,22) = b8(ab(i,1),ab(i,2),1);
ticut(i,23) = b8(ab(i,1),ab(i,2),2);
ticut(i,24) = b8(ab(i,1),ab(i,2),3);
end

  ns = CREATENS(double(ricut),'NSMethod','kdtree');
idxx = knnsearch(ns,double(ticut));

n=size(bmask);
bottleN=zeros(1176,398,3);
for i=1:337894
        bottleN(ab(i,1),ab(i,2),1)=a(idxx(i,1),3);
        bottleN(ab(i,1),ab(i,2),2)=a(idxx(i,1),4);
        bottleN(ab(i,1),ab(i,2),3)=a(idxx(i,1),5);
end

sn1 = bottleN(:,:,2);
sn2 = bottleN(:,:,1);
sn3 = bottleN(:,:,3);
for i = 1:n(1)
for j = 1:n(2)
        if (bottleN(i,j,3)==0)
                p(i,j)=0;
                q(i,j)=0;
        else
        p(i,j) = bottleN(i,j,2)./bottleN(i,j,3);
        q(i,j) = bottleN(i,j,1)./bottleN(i,j,3);
        end
end
end
```

```
Height(1,1) = 0;
for i = 2:n(1)
Height(i,1) = Height(i-1,1) + q(i-1,1);
end

for i = 1:n(1)
for j = 2:n(2)
Height(i,j) = Height(i,j-1) + p(i,j-1);
end
end
figure ,H=surf(Height);
set(H, 'edgecolor', 'none');


%bbmask=double(bmask);
%g = integrate_horn2(q,p,bbmask,50000,1);
%figure ,H=surf(g);
%set(H, 'edgecolor', 'none');

[X Y ] = meshgrid(1:n(2), 1:n(1));

for i = 1:floor(n(1)/8)
for j = 1:floor(n(2)/8)
x(i,j) = X(i*8, j*8);
y(i,j) = Y(i*8, j*8);
height(i,j) = Height(i*8, j*8);
sn1_s(i,j) = sn1(i*8, j*8);
sn2_s(i,j) = sn2(i*8, j*8);
sn3_s(i,j) = sn3(i*8, j*8);
end
end

subplot(2,2,1), imshow(sn1, [min(sn1(:)) max(sn1(:))])
hold on
title('Surface Normal X Component');
hold off

subplot(2,2,2), imshow(sn2, [min(sn2(:)) max(sn2(:))])
hold on
title('Surface Normal Y Component');
hold off

subplot(2,2,3), imshow(sn3, [min(sn3(:)) max(sn3(:))])
hold on
```

```
title('Surface Normal Z Component');
hold off

figure, quiver3(x,y,height, sn1_s, sn2_s, sn3_s)
```