Hao-en Sung (A53204772)
Tsung-han Lee (A53219632)
University of California, San Diego

October 17, 2016

1 Assignment 1

1.1 Perspective Projection [6 pts]

1. Calculate the coordinates of the endpoints of the projection of the ray onto the image plane. [2pt]

Sol. It is known that

$$Q = \begin{pmatrix} 4 \\ -7+2s \\ s \end{pmatrix}, \ s \in (-\infty, -1].$$

With the perspective projection, projected line can be written as

$$Q' = \begin{pmatrix} \frac{4}{s} \cdot f' \\ \frac{-7+2s}{s} \cdot f' \\ f' \end{pmatrix} = \begin{pmatrix} \frac{4f'}{s} \\ \frac{-7f'}{s} + 2f' \\ f' \end{pmatrix}, \ s \in (-\infty, -1].$$

Then, one of the closed endpoint at $s = -1$ is

$$\begin{pmatrix} -4f' \\ 9f' \\ f' \end{pmatrix}.$$

□

2. Find the equation of a ray $L \neq Q$ that is parallel to Q. How did you arrive at this equation? [2pt]

Sol. Since parallel rays in 3D space simply means they share the same $p2$ but may have different $p1$. So I can simply let

$$L = \begin{pmatrix} 4 \\ 2s \\ s \end{pmatrix}, \ s \in (-\infty, -1].$$

□

3. Can you find the point on the image plane , where the rays L and Q meet? If so, what is that point? [Hint: Think in terms of projective geometry] [2pt]

Sol. Similar to 1., I can write down projected L' as

$$L' = \begin{pmatrix} \frac{4f'}{s} \\ 2f' \\ f' \end{pmatrix}.$$

It is obvious that the line Q' and L' will never intersect with each other, since $\frac{-7f'}{s} + 2f' \neq 2s$. □

1.2 Geometry [6 pt]

1. Using Euclidean coordinates, find the equation of the line perpendicular to the family of lines $\tilde{y} = \tilde{x} + \lambda, \lambda \in (-\infty, \infty)$ and a distance D from origin. Express your answer in terms of the given parameters. [2pt]

Sol. I need to find those lines that perpendicular to the line family $\tilde{y} = \tilde{x} + \lambda, \lambda \in (-\infty, \infty)$. Thus, the line must be in the form $\tilde{x} + \tilde{y} + c = 0$.

To calculate the distance between a dot and a line, I have

$$D = \frac{|c - 0|}{\sqrt{1+1}}.$$

Thus two lines are

$$\tilde{x} + \tilde{y} \pm \sqrt{2}D = 0.$$

□

2. Prove the following two statements (using homogeneous coordinates) that follow from equation 1.

   (a) The cross product between two points gives us the line joining the two points. [2pt]

   Sol. Given two points $(x, y, 1)$ and $(x', y', 1)$, they can be regarded as two vectors in 3D space starting from $(0, 0, 0)$. It is known that the cross product between two vectors result in a normal vector v that is perpendicular to those two lines. Since judging whether a point lies in a line utilizes exactly the dot product, and the dot product between two lines and v are zero here. According to the definition of perpendicular, it can be regard as their intersection point. □

   (b) The cross product between two lines gives us their point of intersection. [2pt]

   Sol. Given two lines (a, b, c) and (a', b', c'), their cross product result in a vector v that is perpendicular to those two lines. Since the dot product between two lines and v are zero, according to the definition of perpendicular, it can be regard as their intersection point. □

3. What is the line, in homogeneous coordinates, joining the points $(1, 4)$ and $(4, 5)$. [2pt]

Sol. Regard $x_1 = (1, 4, 1)$ and $x_2 = (4, 5, 1)$, their cross product is $(-1, 3, -11)$. □

1.3 Image formation and rigid body transformations [12 points]

1. No rigid body transformation [3 pts]

Sol. Matrix E and I are described as follows, while generated figure and MATLAB code are included in Fig. 1 and Listing 1 in Appendix, respectively.

$$E = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \qquad K = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

□

2. Translation [3 pts]

Sol. Matrix E and I are described as follows, while generated figure and MATLAB code are included in Fig. 1 and Listing 1 in Appendix, respectively.

$$E = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \qquad K = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

□

3. Translation and rotation [3 pts]

Sol. Assume that the rotation degree around z-axis, y-axis and x-axis are α, β, γ, respectively. Rotation matrix can be written as

$$R = \begin{pmatrix} \cos\alpha \cdot \cos\beta & \cos\alpha \cdot \sin\beta \cdot \sin\gamma - \sin\alpha \cdot \cos\gamma & \cos\alpha \cdot \sin\beta \cdot \cos\gamma + \sin\alpha \cdot \sin\gamma \\ \sin\alpha \cdot \cos\beta & \sin\alpha \cdot \sin\beta \cdot \sin\gamma + \cos\alpha \cdot \cos\gamma & \sin\alpha \cdot \sin\beta \cdot \cos\gamma - \cos\alpha \cdot \sin\gamma \\ -\sin\beta & \cos\beta \cdot \sin\gamma & \cos\beta \cdot \cos\gamma \end{pmatrix}.$$

With given setting parameters, I have

$$\begin{aligned} R &= \begin{pmatrix} \cos\frac{\pi}{3} \cdot \cos\frac{\pi}{4} & \cos\frac{\pi}{3} \cdot \sin\frac{\pi}{4} \cdot \sin 0 - \sin\frac{\pi}{3} \cdot \cos 0 & \cos\frac{\pi}{3} \cdot \sin\frac{\pi}{4} \cdot \cos 0 + \sin\frac{\pi}{3} \cdot \sin 0 \\ \sin\frac{\pi}{3} \cdot \cos\frac{\pi}{4} & \sin\frac{\pi}{3} \cdot \sin\frac{\pi}{4} \cdot \sin 0 + \cos\frac{\pi}{3} \cdot \cos 0 & \sin\frac{\pi}{3} \cdot \sin\frac{\pi}{4} \cdot \cos 0 - \cos\frac{\pi}{3} \cdot \sin 0 \\ -\sin\frac{\pi}{4} & \cos\frac{\pi}{4} \cdot \sin 0 & \cos\frac{\pi}{4} \cdot \cos 0 \end{pmatrix} \\ &= \begin{pmatrix} \frac{1}{2} \cdot \frac{\sqrt{2}}{2} & \frac{1}{2} \cdot \frac{\sqrt{2}}{2} \cdot 0 - \frac{\sqrt{3}}{2} \cdot 1 & \frac{1}{2} \cdot \frac{\sqrt{2}}{2} \cdot 1 + \frac{\sqrt{3}}{2} \cdot 0 \\ \frac{\sqrt{3}}{2} \cdot \frac{\sqrt{2}}{2} & \frac{\sqrt{3}}{2} \cdot \frac{\sqrt{2}}{2} \cdot 0 + \frac{1}{2} \cdot 1 & \frac{\sqrt{3}}{2} \cdot \frac{\sqrt{2}}{2} \cdot 1 - \frac{1}{2} \cdot 0 \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \cdot 0 & \frac{\sqrt{2}}{2} \cdot 1 \end{pmatrix} \\ &= \begin{pmatrix} \frac{\sqrt{2}}{4} & -\frac{\sqrt{3}}{2} & \frac{\sqrt{2}}{4} \\ \frac{\sqrt{6}}{4} & \frac{1}{2} & \frac{\sqrt{6}}{4} \\ -\frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} \end{pmatrix} \end{aligned}$$

Matrix E and I are described as follows, while generated figure and MATLAB code are included in Fig. 1 and Listing 1 in Appendix, respectively.

$$E = \begin{pmatrix} \frac{\sqrt{2}}{4} & -\frac{\sqrt{3}}{2} & \frac{\sqrt{2}}{4} & 0 \\ \frac{\sqrt{6}}{4} & \frac{1}{2} & \frac{\sqrt{6}}{4} & 0 \\ -\frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \qquad K = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

□

4. Translation and rotation, long distance [3 pts]

Sol. Similar to 3., I can write down rotation matrix as

$$\begin{aligned} R &= \begin{pmatrix} \cos\frac{\pi}{3}\cdot\cos\frac{\pi}{2} & \cos\frac{\pi}{3}\cdot\sin\frac{\pi}{2}\cdot\sin 0 - \sin\frac{\pi}{3}\cdot\cos 0 & \cos\frac{\pi}{3}\cdot\sin\frac{\pi}{2}\cdot\cos 0 + \sin\frac{\pi}{3}\cdot\sin 0 \\ \sin\frac{\pi}{3}\cdot\cos\frac{\pi}{2} & \sin\frac{\pi}{3}\cdot\sin\frac{\pi}{2}\cdot\sin 0 + \cos\frac{\pi}{3}\cdot\cos 0 & \sin\frac{\pi}{3}\cdot\sin\frac{\pi}{2}\cdot\cos 0 - \cos\frac{\pi}{3}\cdot\sin 0 \\ -\sin\frac{\pi}{2} & \cos\frac{\pi}{2}\cdot\sin 0 & \cos\frac{\pi}{2}\cdot\cos 0 \end{pmatrix} \\ &= \begin{pmatrix} \frac{1}{2}\cdot 0 & \frac{1}{2}\cdot 1\cdot 0 - \frac{\sqrt{3}}{2}\cdot 1 & \frac{1}{2}\cdot 1\cdot 1 + \frac{\sqrt{3}}{2}\cdot 0 \\ \frac{\sqrt{3}}{2}\cdot 0 & \frac{\sqrt{3}}{2}\cdot 1\cdot 0 + \frac{1}{2}\cdot 1 & \frac{\sqrt{3}}{2}\cdot 1\cdot 1 - \frac{1}{2}\cdot 0 \\ -1 & 0\cdot 0 & 0\cdot 1 \end{pmatrix} \\ &= \begin{pmatrix} 0 & -\frac{\sqrt{3}}{2}\cdot 1 & \frac{1}{2} \\ 0 & \frac{1}{2} & \frac{\sqrt{3}}{2} \\ -1 & 0 & 0 \end{pmatrix} \end{aligned}$$

Matrix E and I are described as follows, while generated figure and MATLAB code are included in Fig. 1 and Listing 1 in Appendix, respectively.

$$E = \begin{pmatrix} 0 & -\frac{\sqrt{3}}{2} & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & \frac{\sqrt{3}}{2} & 0 \\ -1 & 0 & 0 & 13 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \qquad K = \begin{pmatrix} 15 & 0 & 0 \\ 0 & 15 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

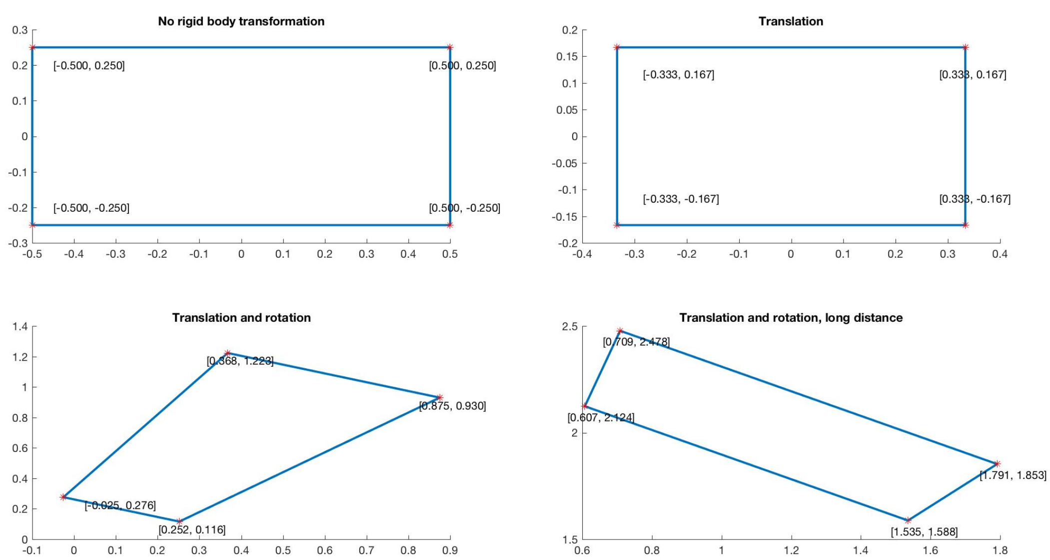

Figure 1: Image Transformation

□

1.4 Rendering [14 points]

1. Plot the face in 2-D [2 pts]

Sol. I simply read in *facedata.mat* file and render it with *imagesc.m* function. 2D images for albedo and uniform albedo are shown in Fig 2.

Since in the uniform albedo graph, all the small piece of surface reflect the light in the same magnitude, there is no difference between the small piece of the graph. Therefore, it forms to be in the same color.

□

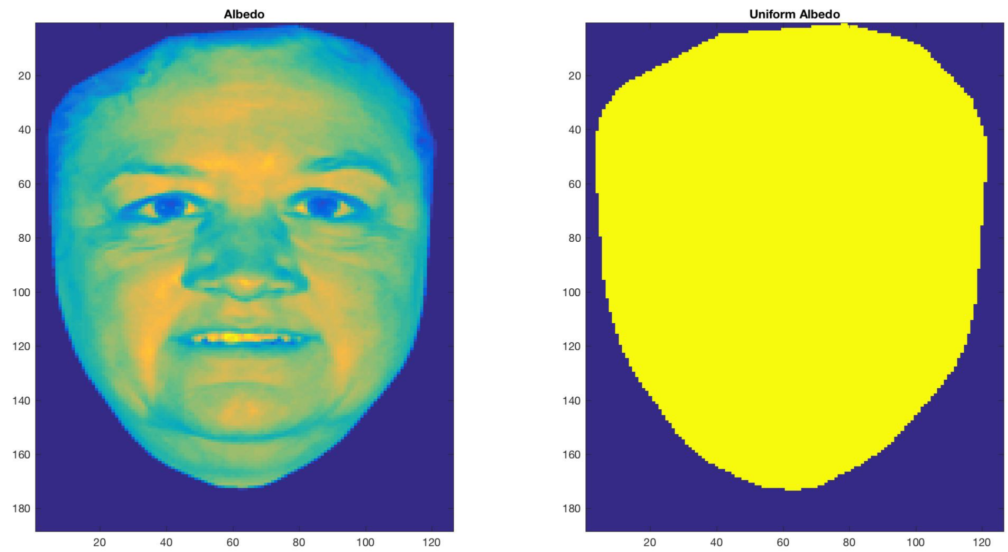

Figure 2: 2D Face

2. Plot the face in 3-D [2 pts]

Sol. Similar to 1., I simply read in *facedata.mat* file and render it with *surf.m* function. 3D images for albedo and uniform albedo are shown in Fig 3.

Same explanation as written in 1.4.1.

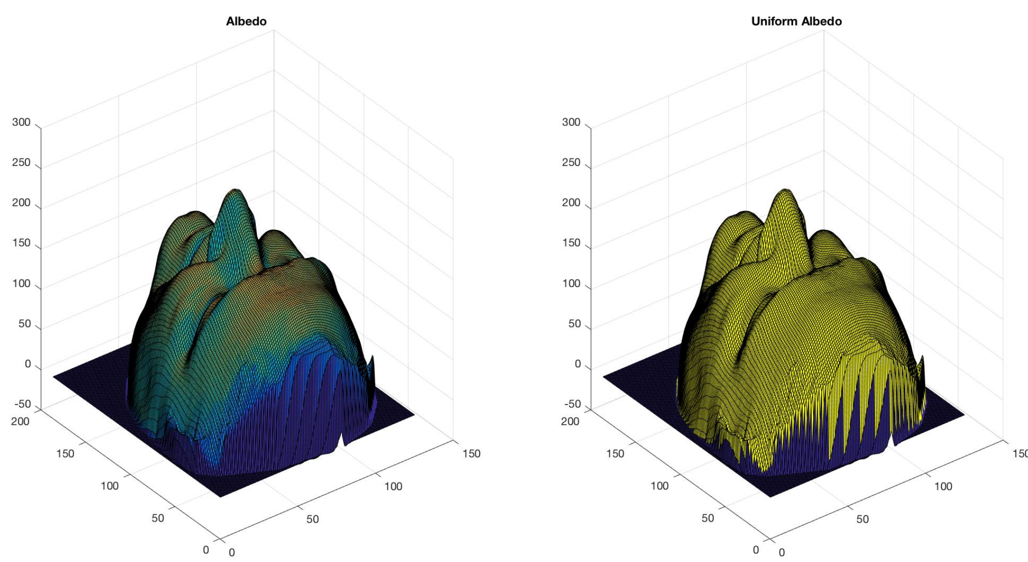

Figure 3: 3D Face

□

3. Surface normals [5 pts]

Sol. Surface normals can be derived from *heightmap.mat* using the height integration result. To have a easily understanding figure, only 1% of surface vectors are subsampled, and they are shown in Fig 4.

□

4. Render images [5 pts]

Sol. For two single-light scenarios, each pixel of image can be calculated with given formula. For the two-light scenarios, I just calculated the average effect caused by two single lights. Note that since the pixel only contains non-negative values, one needs to use truncate it to 0 once it has negative value. Figures with three kinds of light direction for both albedo and uniform albedo are shown in Fig 5.

□

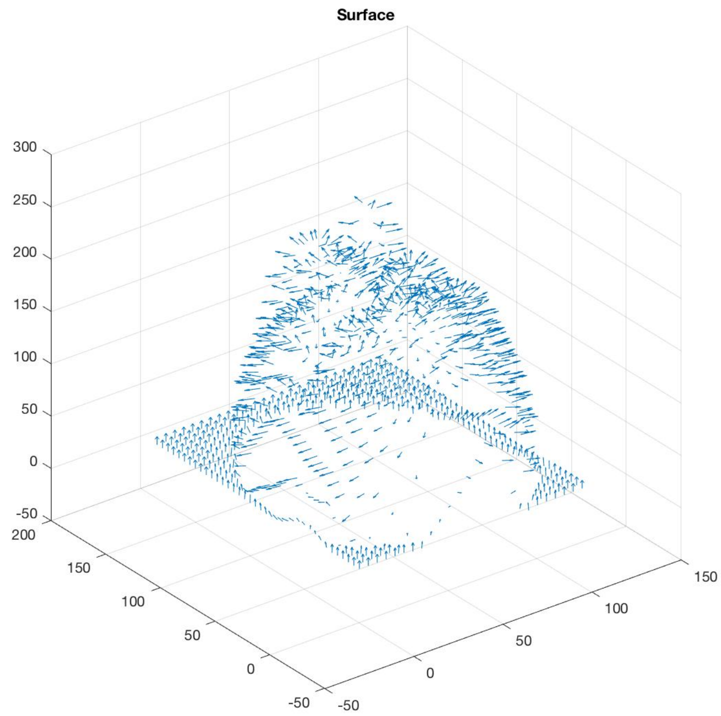

Figure 4: Surface Normals

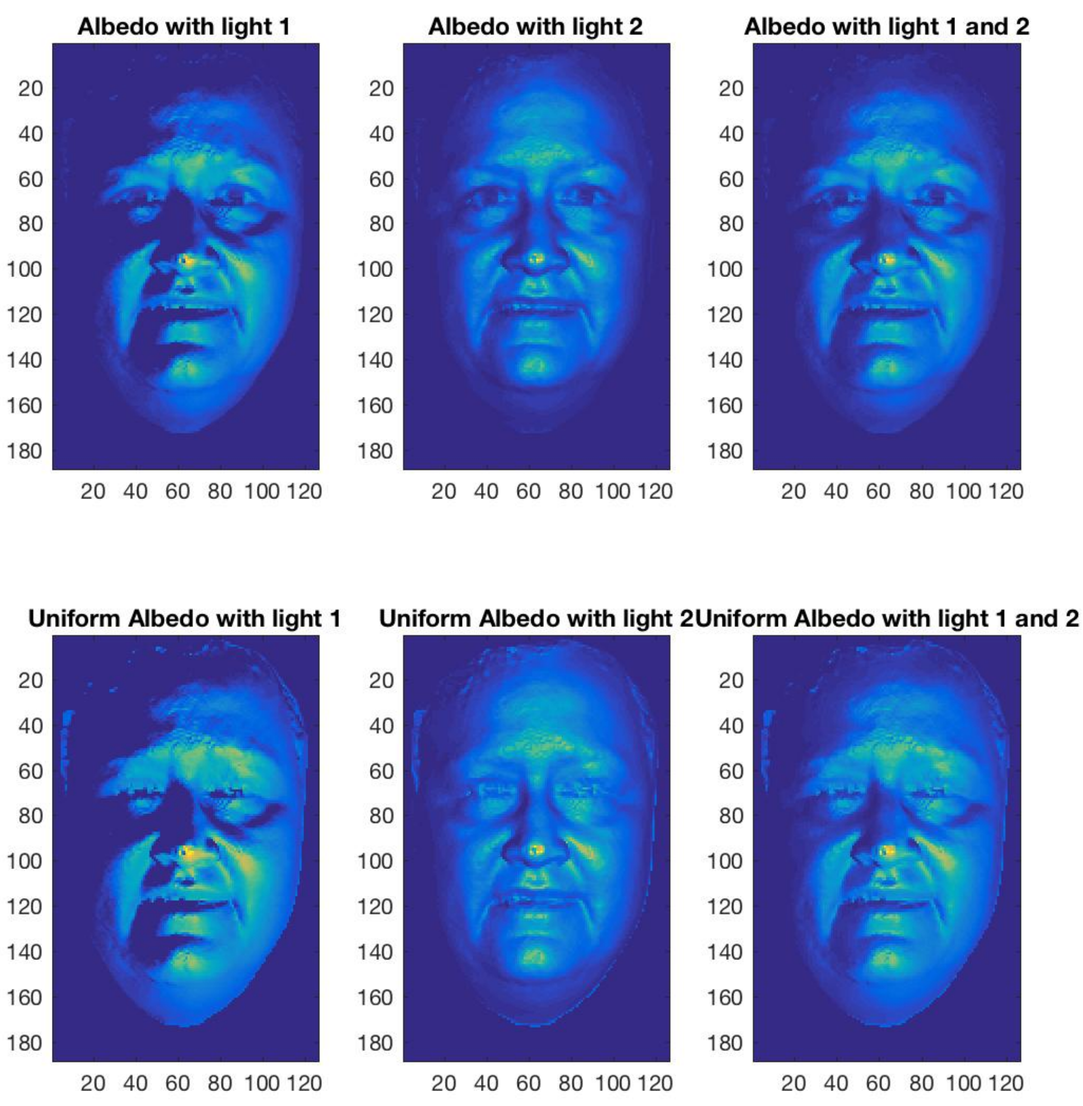

Figure 5: Render Images

1.5 Photometric Stereo [18 points]

1. Estimated albedo map [3 pts $\times 2 = 6$ pts]

Sol. Firstly, I loaded in four images and four corresponding light vectors, and normalize the image matrix to $[0, 1]$ by dividing all pixels by 256.

Later, I applied anonymous functions to each element, or said pixel, of image to calculate $G = Kd \times N = L\ I$. It is noticeable that using *arrayfun* not only increases the code readability but enhances the code efficiency.

After above-said preprocessing, I can render albedo figure easily by calculating $Kd = |G|$ as Fig. 6.

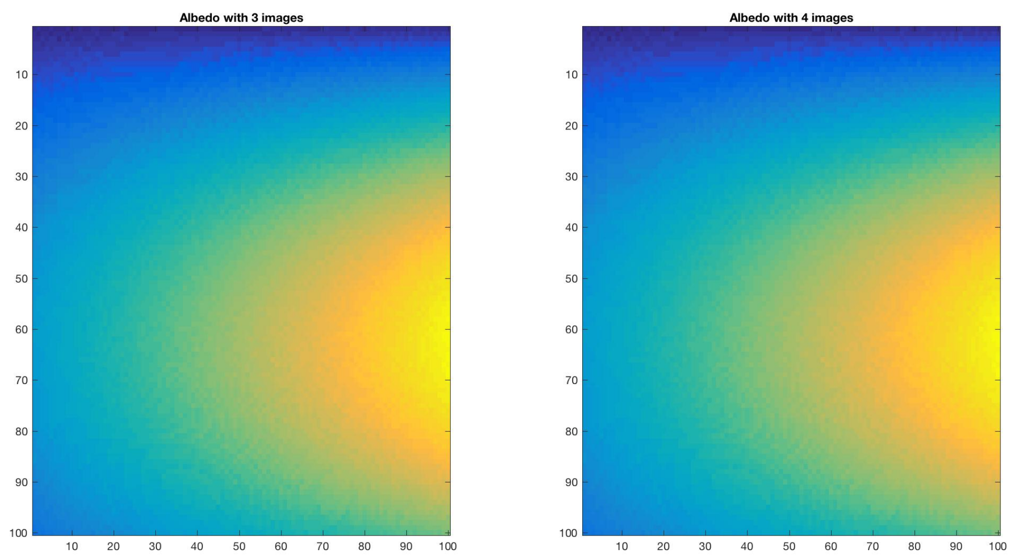

Figure 6: Stereo Albedo

□

2. Estimated surface normals [3 pts $\times 2 = 6$ pts]

Sol. Following 1., surface normals can be obtained by normalizing G with Kd pixel by pixel. I also need to recover depth map here, so that I can add height information into *quiver3.m* to create correct needle figure. The result of depth map is calculated by integration of $\frac{N_x}{N_z}$ and $\frac{N_y}{N_z}$.

With surface normals N_x, N_y, N_z and depth map D given, results created by *quiver3* are shown as Fig. 7.

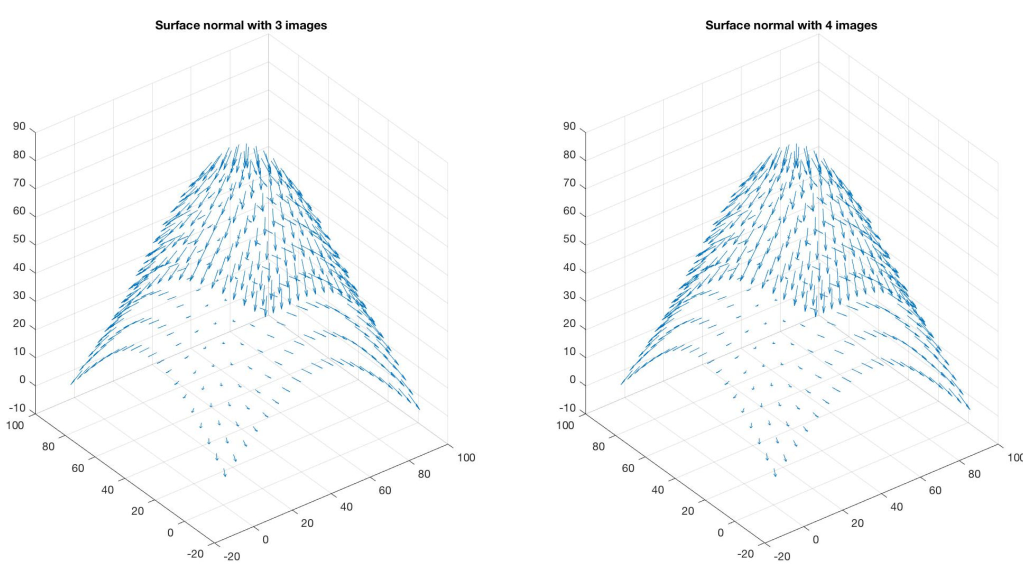

Figure 7: Stereo Surface Normal

□

3. A wireframe of a depth map [3 pts $\times 2 = 6$ pts]

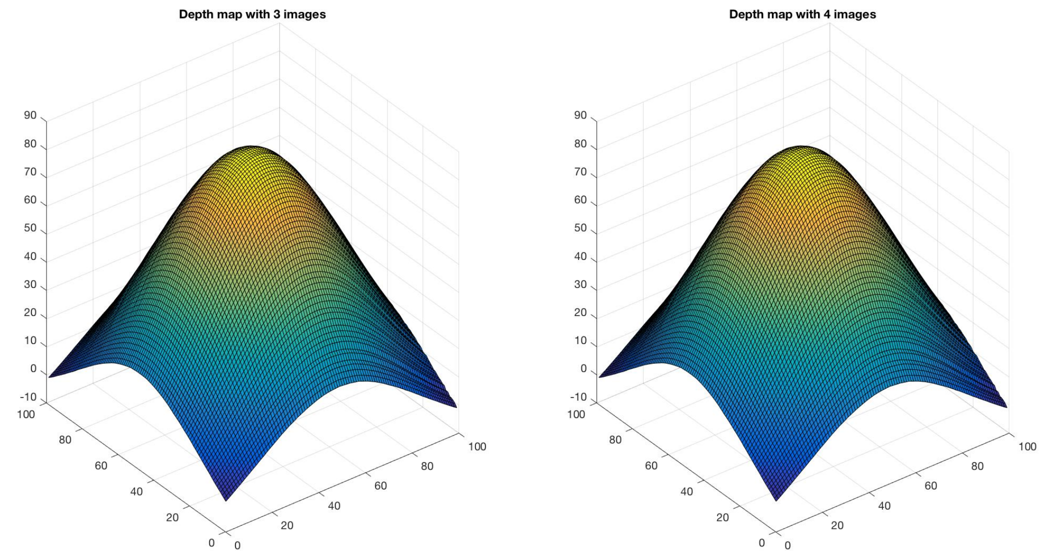

Figure 8: Stereo Depth Map

Sol. Directly utilize the depth matrix obtained in 2., depth map is shown as Fig. 8.

□

1.6 Appendix

(a) Code for Section 3: Image formation and rigid body transformations

Code to manipulate given *plotsquare.m* are included as follows.

```
res = figure('visible','off');
set(res, 'PaperPosition', [0 0 40 20]);

% Four 3D points
obj = [-1 -0.5 2; 1 -0.5 2; 1 0.5 2; -1 0.5 2];
obj(:, 4) = [1 1 1 1];

% Four scenarios
E = cell(4);
K = cell(4);
S = cell(4);

K{1} = [1 0 0; 0 1 0; 0 0 1];
E{1} = [1 0 0 0; 0 1 0 0; 0 0 1 0; 0 0 0 1];
S{1} = 'No rigid body transformation';

K{2} = [1 0 0; 0 1 0; 0 0 1];
E{2} = [1 0 0 0; 0 1 0 0; 0 0 1 1; 0 0 0 1];
S{2} = 'Translation';

K{3} = [1 0 0; 0 1 0; 0 0 1];
E{3} = [sqrt(2) / 4 -sqrt(3) / 2 sqrt(2)/4 0;
        sqrt(6)/4 1/2 sqrt(6)/4 0;
        -sqrt(2)/2 0 sqrt(2)/2 1; 0 0 0 1];
S{3} = 'Translation and rotation';

K{4} = [15 0 0; 0 15 0; 0 0 1];
E{4} = [0 -sqrt(3)/2 1/2 0;
        0 1/2 sqrt(3)/2 0;
        -1 0 0 13; 0 0 0 1];
S{4} = 'Translation and rotation, long distance';

% Draw subplots accordingly
for k = 1:4
    subplot(2,2,k);
    dat = K{k} * [eye(3) zeros(3, 1)] * E{k} * obj';
    plotsquare(dat);
    title(S{k});
end

% Save generated figure
saveas(res, '../res/image_transformation.jpg');
```

Listing 1: Code for Section 3

(b) Code for Section 4-1: Plot the face in 2-D

Code for plot 2D-face images are included as follows.

```
load('../dat/facedata.mat');

res = figure('visible','off');
set(res, 'PaperPosition', [0 0 40 20]);

subplot(1,2,1);
imagesc(albedo);
title('Albedo');

subplot(1,2,2);
imagesc(uniform_albedo);
title('Uniform Albedo');

saveas(res, '../res/face_2D.jpg');
```

Listing 2: Code for Section 4-1

(c) Code for Section 4-2: Plot the face in 3-D

Code for plot 3D-face images are included as follows.

```
load('../dat/facedata.mat');

res = figure('visible','off');
set(res, 'PaperPosition', [0 0 40 20]);

subplot(1,2,1);
surf(heightmap, albedo);
title('Albedo');

subplot(1,2,2);
surf(heightmap, uniform_albedo);
title('Uniform Albedo');

saveas(res, '../res/face_3D.jpg');
```

Listing 3: Code for Section 4-2

(d) Code for Section 4-3: Surface normals

Code for plot surface normals are included as follows.

```
load('../dat/facedata.mat');

%% Draw Surface Normal

res = figure('visible','off');
set(res, 'PaperPosition', [0 0 20 20]);

[n, m] = size(heightmap);
nx = zeros(n, m);
ny = zeros(n, m);
nz = zeros(n, m);

for i = 1:n
    for j = 1:m
        if i < n
            b = heightmap(i,j) - heightmap(i+1,j);
        else
            b = heightmap(i,j);
        end
        if j < m
            a = heightmap(i,j) - heightmap(i,j+1);
        else
            a = heightmap(i,j);
        end
        nz(i,j) = sqrt(1/(a*a+b*b+1));
        nx(i,j) = a * nz(i,j);
        ny(i,j) = b * nz(i,j);
    end
end

x = 1:5:130;
y = 1:5:190;
[X,Y] = meshgrid(x, y);

ss_h = imresize(heightmap, 0.2);
ss_nx = imresize(nx, 0.2);
```

```
ss_ny = imresize(ny, 0.2);
ss_nz = imresize(nz, 0.2);

quiver3(X, Y, ss_h, ss_nx, ss_ny, ss_nz);
title('Surface');

saveas(res, '../res/surface_normal.jpg');
```

Listing 4: Code for Section 4-3

(e) Code for Section 4-4: Render images

Code for rendering images are included as follows.

```
load('../dat/facedata.mat');

res = figure('visible','off');
set(res, 'PaperPosition', [0 0 20 20]);

[n, m] = size(heightmap);
img_a_1 = zeros(n, m);
img_a_2 = zeros(n, m);
img_a_12 = zeros(n, m);
img_ua_1 = zeros(n, m);
img_ua_2 = zeros(n, m);
img_ua_12 = zeros(n, m);

for i = 1:n
    for j = 1:m
        if i < n
            b = heightmap(i,j) - heightmap(i+1,j);
        else
            b = heightmap(i,j);
        end
        if j < m
            a = heightmap(i,j) - heightmap(i,j+1);
        else
            a = heightmap(i,j);
        end
        nz = sqrt(1/(a*a+b*b+1));
        nx = a * nz;
        ny = b * nz;

        dist_1 = norm(lightsource(1,:) - [i, j, heightmap(i,j)]);
        img_a_1(i,j) = max(0, albedo(i,j) * [nx ny nz] ...
            * lightsource(1,:)' * 1.0 / (dist_1^2));
        img_ua_1(i,j) = max(0, uniform_albedo(i,j) * [nx ny nz] ...
            * lightsource(1,:)' * 1.0 / (dist_1^2));

        dist_2 = norm(lightsource(2,:) - [i, j, heightmap(i,j)]);
        img_a_2(i,j) = max(0, albedo(i,j) * [nx ny nz] ...
            * lightsource(2,:)' * 1.0 / (dist_1^2));
        img_ua_2(i,j) = max(0, uniform_albedo(i,j) * [nx ny nz] ...
            * lightsource(2,:)' * 1.0 / (dist_1^2));

        img_a_12(i,j) = img_a_1(i,j) + img_a_2(i,j);
        img_ua_12(i,j) = img_ua_1(i,j) + img_ua_2(i,j);
    end
end

subplot(2,3,1);
imagesc(img_a_1);
title('Albedo with light 1');

subplot(2,3,2);
imagesc(img_a_2);
title('Albedo with light 2');

subplot(2,3,3);
imagesc(img_a_12);
title('Albedo with light 1 and 2');

subplot(2,3,4);
imagesc(img_ua_1);
title('Uniform Albedo with light 1');

subplot(2,3,5);
imagesc(img_ua_2);
title('Uniform Albedo with light 2');

subplot(2,3,6);
imagesc(img_ua_12);
title('Uniform Albedo with light 1 and 2');

saveas(res, '../res/render.jpg');
```

Listing 5: Code for Section 4-4

(f) Code for Section 5: Photometric Stereo

Code for estimated albedo map, surface normals and depth map from three or four images are included as follows.

```
load('../dat/synthetic_data.mat');

%% Preparation

% Prepare image matrix
img = zeros(4, 10000);
img(1,:) = im1(:);
img(2,:) = im2(:);
img(3,:) = im3(:);
img(4,:) = im4(:);
img = arrayfun(@(x) double(x) / 256, img);
imgc = num2cell(img, 1);

% Prepare small image matrix
s_img = img([1 2:4],:);
s_imgc = num2cell(s_img, 1);

% Prepare light direction sources
lgt = zeros(4, 3);
lgt(1,:) = l1(:) / norm(l1(:));
lgt(2,:) = l2(:) / norm(l2(:));
lgt(3,:) = l3(:) / norm(l3(:));
lgt(4,:) = l4(:) / norm(l4(:));

% Calculate pesudo inverse of light matrix
plgt = pinv(lgt);

% Calculate G = Kd * N = L \ I
G = arrayfun(@(x) plgt * x{:}, imgc, 'UniformOutput', false);
s_G = arrayfun(@(x) plgt * x{:}, s_imgc, 'UniformOutput', false);

G = reshape(G, [100, 100]);
s_G = reshape(s_G, [100, 100]);



%% (1) Draw Albedo

% Calculate Kd = |G|
Kd = arrayfun(@(x) norm(x{:}), G, 'UniformOutput', false);
s_Kd = arrayfun(@(x) norm(x{:}), s_G, 'UniformOutput', false);

res = figure('visible','off');
set(res, 'PaperPosition', [0 0 40 20]);

subplot(1,2,1);
imagesc(cell2mat(s_Kd));
title('Albedo with 3 images');

subplot(1,2,2);
imagesc(cell2mat(Kd));
title('Albedo with 4 images');

saveas(res, '../res/stereo_albedo.jpg');



%% (2) Draw Surface Normal

% Calculate N
N = cellfun(@(x,y) x/y, G, Kd, 'UniformOutput', false);
N = reshape(cell2mat(N), [3, 100, 100]);
Nx = squeeze(N(1,:,:));
Ny = squeeze(N(2,:,:));
Nz = squeeze(N(3,:,:));

s_N = cellfun(@(x,y) x/y, s_G, s_Kd, 'UniformOutput', false);
s_N = reshape(cell2mat(s_N), [3, 100, 100]);
s_Nx = squeeze(s_N(1,:,:));
s_Ny = squeeze(s_N(2,:,:));
s_Nz = squeeze(s_N(3,:,:));

% Calculate D
D = zeros(100, 100);
s_D = zeros(100, 100);
for i = 2:100
    D(i,1) = D(i-1,1) - Nx(i,1)/Nz(i,1);
    s_D(i,1) = s_D(i-1,1) - s_Nx(i,1)/s_Nz(i,1);
end
for i = 1:100
    for j = 2:100
        D(i,j) = D(i,j-1) - Ny(i,j)/Nz(i,j);
        s_D(i,j) = s_D(i,j-1) - s_Ny(i,j)/s_Nz(i,j);
    end
```

```
end

% Subsample graph
x = 1:5:100;
y = 1:5:100;
[X,Y] = meshgrid(x, y);
ss_Nx = imresize(Nx, 0.2);
ss_Ny = imresize(Ny, 0.2);
ss_Nz = imresize(Nz, 0.2);
ss_D = imresize(D, 0.2);

res = figure('visible','off');
set(res, 'PaperPosition', [0 0 40 20]);

subplot(1,2,1);
quiver3(X, Y, ss_D, ss_Nx, ss_Ny, ss_Nz);
title('Surface normal with 3 images');

subplot(1,2,2);
quiver3(X, Y, ss_D, ss_Nx, ss_Ny, ss_Nz);
title('Surface normal with 4 images');

saveas(res, '../res/stereo_surface_normal.jpg');



%% (3) Draw depth map

res = figure('visible','off');
set(res, 'PaperPosition', [0 0 40 20]);

subplot(1,2,1);
x = 1:1:100;
y = 1:1:100;
[X,Y] = meshgrid(x, y);
surf(X, Y, D);
title('Depth map with 3 images');

subplot(1,2,2);
surf(s_D);
title('Depth map with 4 images');

saveas(res, '../res/stereo_depth_map.jpg');
```

Listing 6: Code for Section 5

*Submitted by Hao-en Sung (A53204772)
Tsung-han Lee (A53219632) on October 17, 2016.*