Gupta, Ajitesh
Mavalankar, Aditi Ashutosh
Tran, Derek
Department of Electrical Engineering and Computer Science
University of California, San Diego

October 19, 2016

HOMEWORK 1

Problem 1. **Perspective Projection** : We are first told to consider a perspective projection where a point $P = [x, y, z]^T$ is projected onto an image plane Π' represented by $k = f' > 0$. Then, given the coordinate frame $e = [i, j, k]^T$, consider the projection of a ray $Q = p_1 + sp_2$ in the world coordinate system where $p_1 = [4, -7, 0]^T$, $p_2 = [0, 2, 1]^T$, and $s \in (-\infty, -1]$.

1. Calculate the coordinates of the endpoints of the projection of the ray Q onto the image plane Π'.

   Solution To calculate the two endpoints of the ray Q onto the image plane, we will first plug the value f' as the k coordinate in order to move to the plane Π'. Next, following the equation of the ray Q, we will follow it's direction of $p_2 = [0, 2, 1]^T$. This will give us the point $E_1 = [4, -7 + 2s, f']$. The second endpoint is just the point that follows the ray's origin point down the k direction. This will give us the point $E_2 = [4, -7, f']$.

$$E_1 = \begin{bmatrix} 4 \\ -7 + 2s \\ f' \end{bmatrix} E_2 = \begin{bmatrix} 4 \\ -7 \\ f' \end{bmatrix} \tag{0-1}$$

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

   Solution Following the same method to find a parallel line in 2D, we can apply the same process to 3D rays. To so this, we just have to keep the "slope" or the direction of the ray the same. This will ensure that the ray's are parallel. Then, to make another ray that is different than the one we already have, we just have to change the origin point of the ray. Following these two steps, we can get...

$$L = \begin{bmatrix} 6 \\ 2 \\ 1 \end{bmatrix} + s \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix} \tag{0-2}$$

   Looking at our new ray L, we can see that it has the same "slope" or change of direction as Q which is $[0, 2, 1]^T$. We also can see that its origin differs from Q because it is $[6, 2, 1]^T$ instead of $[4, -7, 0]^T$.

3. Can you find the point on the image plane, where the rays L and Q meet? If so, what is that point?

   Solution In order to calculate the point where two parallel rays meet, we need to find their vanishing point. To do this we take the x and y coordinates of the slope vector and divide them by the z vector. Then taking the resulting vector $[\frac{x}{z}, \frac{y}{z}, 1]$ and multiplying it by the distance of the image plane from the origin coordinate frame.

$$V = f' \begin{bmatrix} \frac{0}{1} \\ \frac{2}{1} \\ 1 \end{bmatrix} = f' \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix} \tag{0-3}$$

Therefore, the point on the image plane where L and Q meet is $V = [0, 2, 1]$.

Problem 2. **Geometry** : We are given a line in the 2-D plane, whose equation is given by $a\tilde{x} + b\tilde{y} + c = 0$. We could also have written the equation of this line as $l^T x = 0$ where $l = (a, b, c)^T$ and $x = (x, y, 1)^T$. Both these are equivalent to each other. Here, x is the homogeneous representation of $\tilde{x} = (\tilde{x}, \tilde{y})^T$ and l is a homogeneous representation of $a\tilde{x} + b\tilde{y} + c = 0$. The line is defined to scale as $(a, b, c)^T$ and $\sigma(a, b, c)^T$ represent the same line if $\sigma \neq 0$. All points $(\tilde{x}, \tilde{y})$ that lie on the line a$\tilde{x} + b\tilde{y} + c = 0$ satisfy the equation $l^T x = 0$.

1. We have to find the equation of the line perpendicular to the given family of lines: $\tilde{y} = \tilde{x} + \lambda, \lambda \in (-\infty, \infty)$. The resultant line should be at a distance $\mathbf{D}$ from the origin. We are supposed to express our answer in terms of the parameters given to us.
   Solution
   The given equation of the line is $\tilde{y} = \tilde{x} + \lambda$.
   If we compare this with the general equation of line $\tilde{y} = m\tilde{x} + c$, where m is the slope and c is the intercept formed by the line on the Y-axis, we can see that here, $m = 1$ and $c = \lambda$.
   Let the equation of the desired line be $\tilde{y} = m'\tilde{x} + c'$.
   We know that the product of slopes of two perpendicular lines is -1. Since the slope of the given line is 1, the slope of the desired line will be -1.
   So, $m' = -1$.
   Now, the equation of the line becomes $\tilde{y} = -\tilde{x} + c'$ or $\tilde{x} + \tilde{y} - c' = 0$.
   To find the distance of a point(x_0, y_0) from any line $a\tilde{x} + b\tilde{y} + c = 0$, we use the following formula:
   $\mathbf{D} = \frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}}$
   Here, we substitute (x_0, y_0) with (0, 0) and $a\tilde{x} + b\tilde{y} + c = 0$ with our line equation $\tilde{x} + \tilde{y} - c' = 0$. So, we get
   $\mathbf{D} = \frac{|0+0-c'|}{\sqrt{1^2+1^2}} = \frac{|c'|}{\sqrt{2}}$
   Thus, $c' = \pm \mathbf{D}\sqrt{2}$.
   So, the equation of the line(s) is given by $\tilde{x} + \tilde{y} \pm \mathbf{D}\sqrt{2} = 0$.

2. (a) Prove that the cross product between two points gives us the line joining those two points.
   Solution
   Let the two points be $(x_1, y_1, 1)^T$ and $(x_2, y_2, 1)^T$
   To get the equation of the line joining two points, we use the two point formula:
   $\tilde{y} - y_1 = m(\tilde{x} - x_1)$
   The slope is given by $m = \frac{y_2 - y_1}{x_2 - x_1}$
   So, we get
   $\tilde{y} - y_1 = \frac{y_2 - y_1}{x_2 - x_1}(\tilde{x} - x_1)$
   Simplifying, we get
   $x_2\tilde{y} - x_1\tilde{y} - x_2y_1 + x_1y_1 = \tilde{x}y_2 - \tilde{x}y_1 - x_1y_2 + x_1y_1$
   Combining the terms containing x and y and the constants, we get
   $(y_1 - y_2)\tilde{x} + (x_2 - x_1)\tilde{y} + (x_1y_2 - x_2y_1) = 0$
   The cross product of the two points also results in
   $(y_1 - y_2)\tilde{x} + (x_2 - x_1)\tilde{y} + (x_1y_2 - x_2y_1)\tilde{1} = 0$
   Hence, proved.

(b) Prove that the cross product between two lines gives us their point of intersection.

Solution

Let the equations of the two lines be

$a\tilde{x} + b\tilde{y} + c\tilde{1} = 0$

and

$a'\tilde{x} + b'\tilde{y} + c'\tilde{1} = 0$

To find the point of intersection from the two line equations:

$y = -\frac{a\tilde{x}+c}{b}$

and

$y = -\frac{a\tilde{x}+c}{b}$

So, with these two equations, we get x:

$\frac{a\tilde{x}+c}{b} = \frac{a'\tilde{x}+c'}{b'}$

$\therefore ab'\tilde{x} + b'c = a'b\tilde{x} + bc'$

$\therefore x = \frac{bc'-b'c}{ab'-a'b}$

Substituting this value of x in the value of y obtained from the first line equation,

$y = -\frac{a\frac{bc'-b'c}{ab'-a'b}+c}{b}$

$\therefore y = -\frac{abc'-ab'c+ab'c-a'bc}{b(ab'-a'b)}$

$\therefore y = \frac{a'bc-abc'}{b(ab'-a'b)}$

$\therefore y = \frac{a'c-ac'}{ab'-a'b}$

Now, looking at the cross product of two lines, we get

$(bc' - b'c)\tilde{x} + (a'c - ac')\tilde{y} + (ab' - a'b)\tilde{1}$

Making the coefficient of the constant term 1, we get

$\frac{bc'-b'c}{ab'-a'b}\tilde{x} + \frac{a'c-ac'}{ab'-a'b}\tilde{y} + \tilde{1}$

Here, we can see that the coefficient of $\tilde{x}$ is $\frac{bc'-b'c}{ab'-a'b}$, and that of $\tilde{y}$ is $\frac{a'c-ac'}{ab'-a'b}$. These are the same as the coordinates of the point of intersection we calculated. We have normalized i.e. set the value of the coordinate of $\tilde{1}$ to 1 because the coordinates we have obtained are of the form $(\frac{bc'-b'c}{ab'-a'b}, \frac{a'c-ac'}{ab'-a'b}, 1)$.

Hence, proved.

3. Find the equation of the line joining the points $(1, 4)$ and $(4, 5)$ in homogeneous coordinates.

Solution

We know from 2.2(a) that the cross product of two points gives us the equation of the line joining them.

Let $(x_1, y_1) \equiv (1, 4)$ and $(x_2, y_2) \equiv (4, 5)$

Using the cross product calculated in 2.2(a),

$(y_1 - y_2)\tilde{x} + (x_2 - x_1)\tilde{y} + (x_1y_2 - x_2y_1)\tilde{1} = 0$

Substituting the values of the coordinates, we get

$(4 - 5)\tilde{x} + (4 - 1)\tilde{y} + (1 \times 5 - 4 \times 4)\tilde{1}$

$\therefore -\tilde{x} + 3\tilde{y} - 11\tilde{1} = 0$

or

$\tilde{x} - 3\tilde{y} + 11\tilde{1} = 0$

is the equation of the line joining $(1, 4)$ and $(4, 5)$.

Problem 3. **Image formation and rigid body transformations** : For this problem, we are given the following four points ${}^AP_1 = (-1, -0.5, 2)^T$, ${}^AP_2 = (1, -0.5, 2)^T$, ${}^AP_3 = (1, 0.5, 2)^T$, ${}^AP_4 = (-1, 0.5, 2)^T$ in world coordinates. We are supposed to do various rigid body transformations and image formations through the projective camera model and create 'photographs'

of the points. To perform these transformations and image formations, we use the following equations. ${}^BP = {}^B_AR\,{}^AP + {}^BO_A$ where ${}^BP$ is the point coordinate in the target (B) coordinate system, ${}^AP$ is the point coordinates in the source (A) coordinate system, ${}^B_AR$ is the rotation matrix from A to B, and ${}^BO_A$ is the origin of coordinate system A expressed in the B coordinates.And the equation $K[I|O]E^AP$. For each of the following, we had to calculate the extrinsic transformation matrix E, the intrinsic camera matrix K, and plot the four vertices using the provided plotsquare.m function.
Solution The solution code for this problem can be found in the Appendix in the listing that says "MATLAB source for problem 3".

1. No rigid body transformation : Focal length = 1. The optical axis of the camera is aligned with the z-axis.

   Solution For the transformation matrix E, since there are no rotations or transformations in this part, we just make it the identity matrix.

$$E = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{0-4}$$

   For the intrinsic camera matrix K, we make the first 2 diagonals the focal length and the last diagonal stays 1. The other values are 0 because we do not change the skew or the principle point of the camera.

$$K = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{0-5}$$

   The resulting image is shown below in Fig. **??** on the top-left plot labeled (3.1).

2. Translation : ${}^BO_A = (0, 0, 1)^T$. The optical axis of the camera is aligned with the z-axis.
   Solution For the transformation matrix E, since we have just a translation of ${}^BO_A = (0, 0, 1)^T$, we have that vector put in the top-right of the matrix while everything else looks like the identity matrix.

$$E = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{0-6}$$

   For the intrinsic camera matrix K, we make the first 2 diagonals the focal length and the last diagonal stays 1. The other values are 0 because we do not change the skew or the principle point of the camera.

$$K = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{0-7}$$

   The resulting image is shown below in Fig. **??** on the top-right plot labeled (3.2).

3. Translation and rotation : Focal length = 1. ${}^B_AR$ encodes a 60 degrees around the z-axis and then 45 degrees around the y-axis. ${}^BO_A = (0, 0, 1)^T$.

   Solution For the transformation matrix E, we now have rotation and translation. The top-left 3x3 part of E is for the rotation matrix where we will rotate about the z and then

y. To do this, we follow the matrices provided in the lecture for rotation about the z and y and multiply them together. Then for the translation, we put the vector ${}^{B}O_A = (0, 0, 1)^T$ in the top-right of the matrix.

$$E = \begin{bmatrix} 0.3536 & -0.6124 & 0.7071 & 0 \\ 0.8660 & 0.5000 & 0 & 0 \\ -0.3536 & 0.6124 & 0.7071 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{0-8}$$

For the intrinsic camera matrix K, we make the first 2 diagonals the focal length and the last diagonal stays 1. The other values are 0 because we do not change the skew or the principle point of the camera.

$$K = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{0-9}$$

The resulting image is shown below in Fig. **??** on the bottom-left plot labeled (3.3).

4. Translation and rotation, long distance : Focal length = 15. ${}^{B}_{A}R$ encodes a 60 degrees around the z-axis and then 90 degrees around the y-axis. ${}^{B}O_A = (0, 0, 13)^T$.

   Solution For the transformation matrix E, we now have rotation and translation. The top-left 3x3 part of E is for the rotation matrix where we will rotate about the z and then y. To do this, we follow the matrices provided in the lecture for rotation about the z and y and multiply them together. Then for the translation, we put the vector ${}^{B}O_A = (0, 0, 13)^T$ in the top-right of the matrix.

$$E = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0.8660 & 0.5000 & 0 & 0 \\ -0.5000 & 0.8660 & 0 & 13 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{0-10}$$

   For the intrinsic camera matrix K, we make the first 2 diagonals the focal length and the last diagonal stays 1. The other values are 0 because we do not change the skew or the principle point of the camera.

$$K = \begin{bmatrix} 15 & 0 & 0 \\ 0 & 15 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{0-11}$$

   The resulting image is shown below in Fig. **??** on the bottom-right plot labeled (3.4).

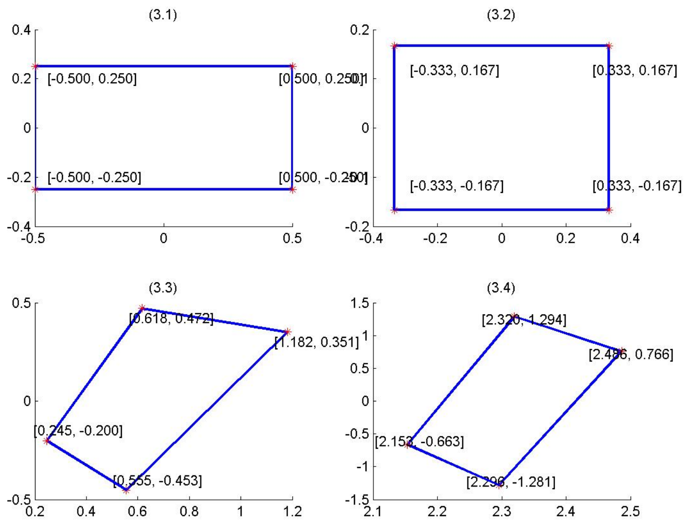


Figure 1: Shows all 4 resultant images for 3.1 to 3.4

Problem 4. **Rendering**

This problem deals with using two different point light sources to render a face using the Lambertian reflectance model. The data we are given consists of two albedo images, out of which one is uniform, and the other is more realistic, the face heightmap and the light sources. All these are given in the file facedata.mat

1. Plot both albedo maps in 2-D using imagesc.m

   Solution

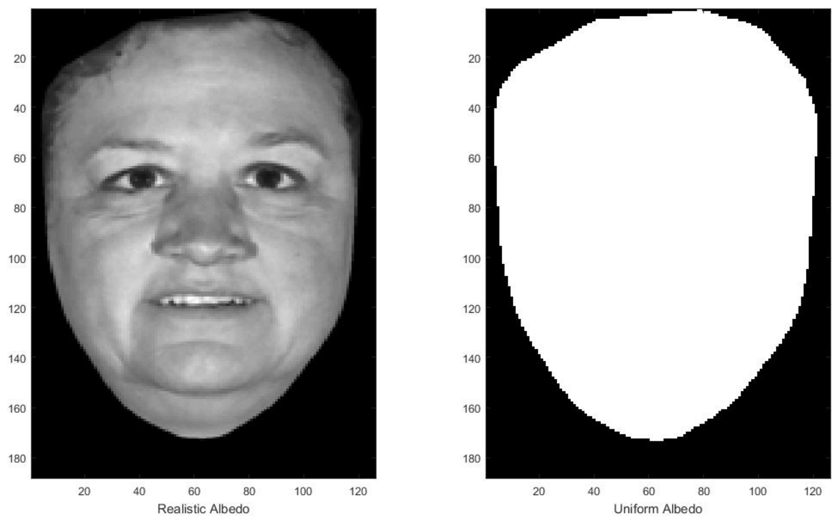


Figure 2: Image formed using the two albedos

On plotting the uniform and realistic albedos, we observe the following:

(a) The uniform albedo gives us a binary mask of the face and background.

(b) The realistic albedo gives us the real 2-D projection of the face.

(c) The shading and lighting conditions on the face are visible in the realistic albedo.

(d) The uniform albedo gives us no information about the object represented, except that it could be a face, just because it provides the outline. The realistic albedo, on the other hand, gives us the exact features and can be used for face recognition.

2. Plot the faces in 3-D using the heightmap and albedo (both albedos - uniform and realistic).

   Solution

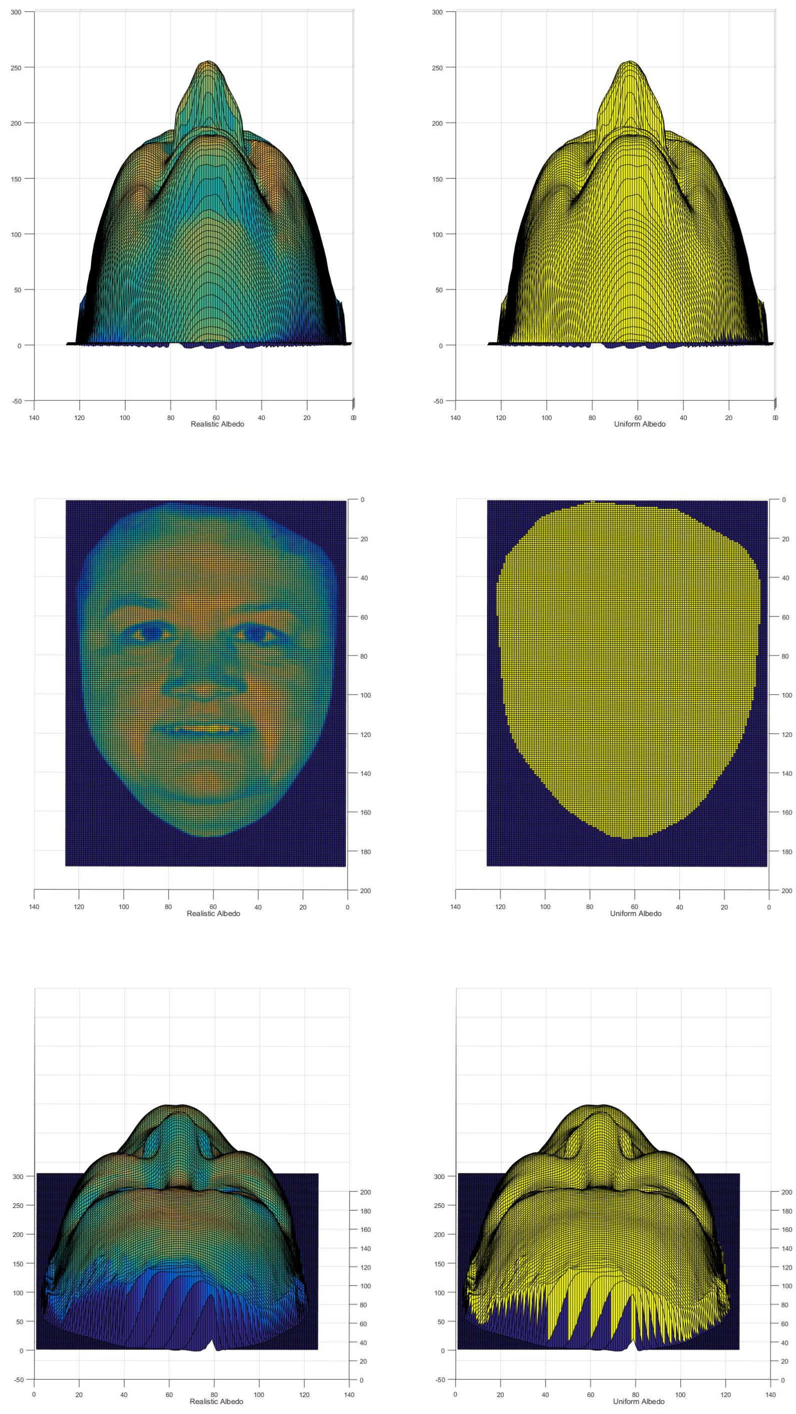


Figure 3: Image formed using the two albedos along with the heightmap

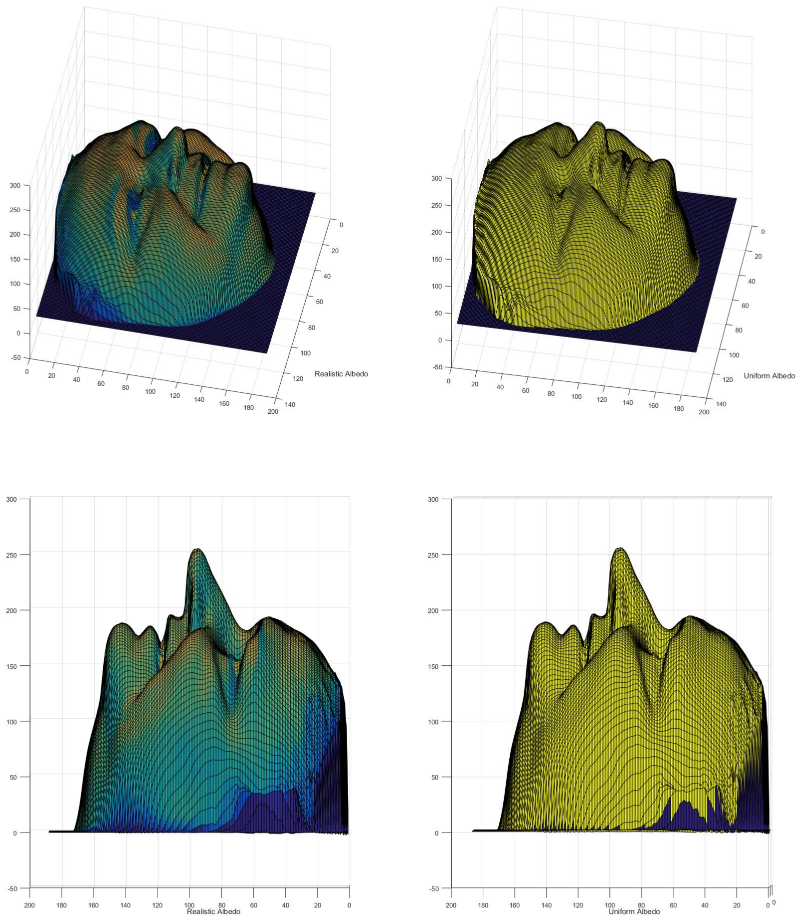


Figure 4: Image formed using the two albedos along with the heightmap

When we plot the face using both the heightmap and albedo, we observe the following:

(a) For different views of the face, different information is revealed in greater detail, which could not have been extracted from the 2-D view alone.

(b) If we take different views:

  i. Frontal view
    - The uniform albedo does not show the face or any feature. It only gives the mask.
    - The realistic albedo gives the same features as the ones observed in the 2-D projection.

  ii. Upper side views
    - Most of the dominant features are visible in both the uniform albedo as well as the realistic albedo.

- A few other details such as eyes, teeth are more clearly visible in the realistic albedo as compared to the uniform albedo.

iii. Top and bottom views

- These views reveal that the nose is longer and bigger than it appears in the 2-D projection.

(c) Here also, the realistic albedo gives a better overall idea of the shading and texture than the uniform albedo.

(d) The face seems to be unrealistically elongated towards the boundaries. This may be due to an error in the estimation of the normals at those points.

3. Use quiver3.m to find the surface normals and display them as a quiver plot. The surface normals are given by
$[-\frac{\delta f}{\delta x}, -\frac{\delta f}{\delta y}, 1]$
Solution
Each normal vector should be normalized i.e. it should be of unit length and to make the quiver plot look better, you should subsample.

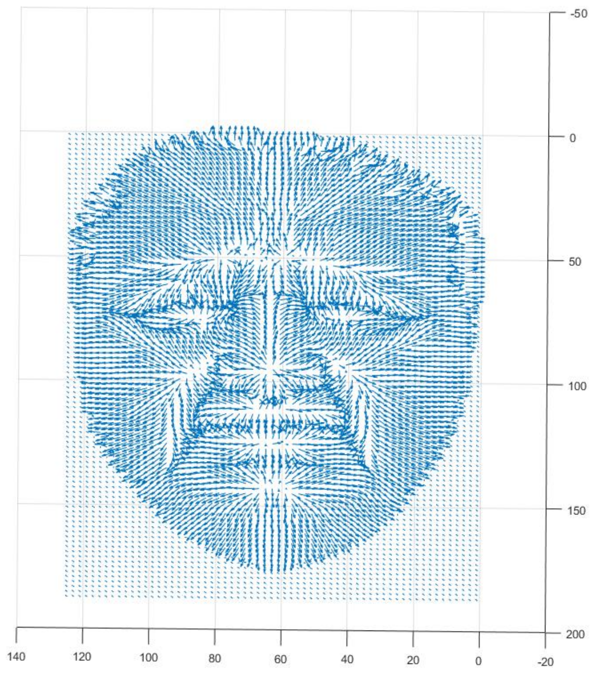


Figure 5: Quiver plot of the surface normals - Front view

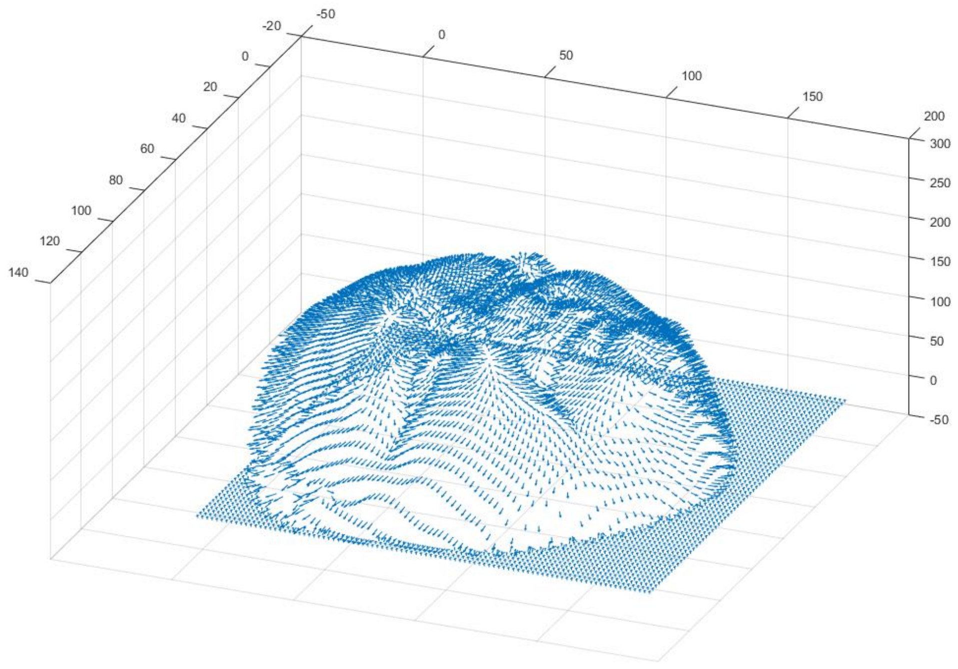


Figure 6: Quiver plot of the surface normals - Side View

4. For each albedo, generate 3 images: one using light source 1, one using light source 2 and one using both light sources. The general image formation equation is given by
   $I = a(x,y)\hat{n}^T(x,y)\hat{s}(x,y)\frac{s_0}{d^2(x,y)}$
   Here,
   $a(x,y)$: albedo for pixel (x,y)
   $\hat{n}(x,y)$: surface normal
   $\hat{s}(x,y)$: light source direction
   s_0 : light source intensity
   $d(x,y)$: distance to the light source
   $\langle .,. \rangle$: dot product
   Let the light source intensity be 1. Display the images in a 2 × 3 subplot figure with titles. Use imagesc.m to display these images. Do NOT make the distant light source assumption.
   Solution

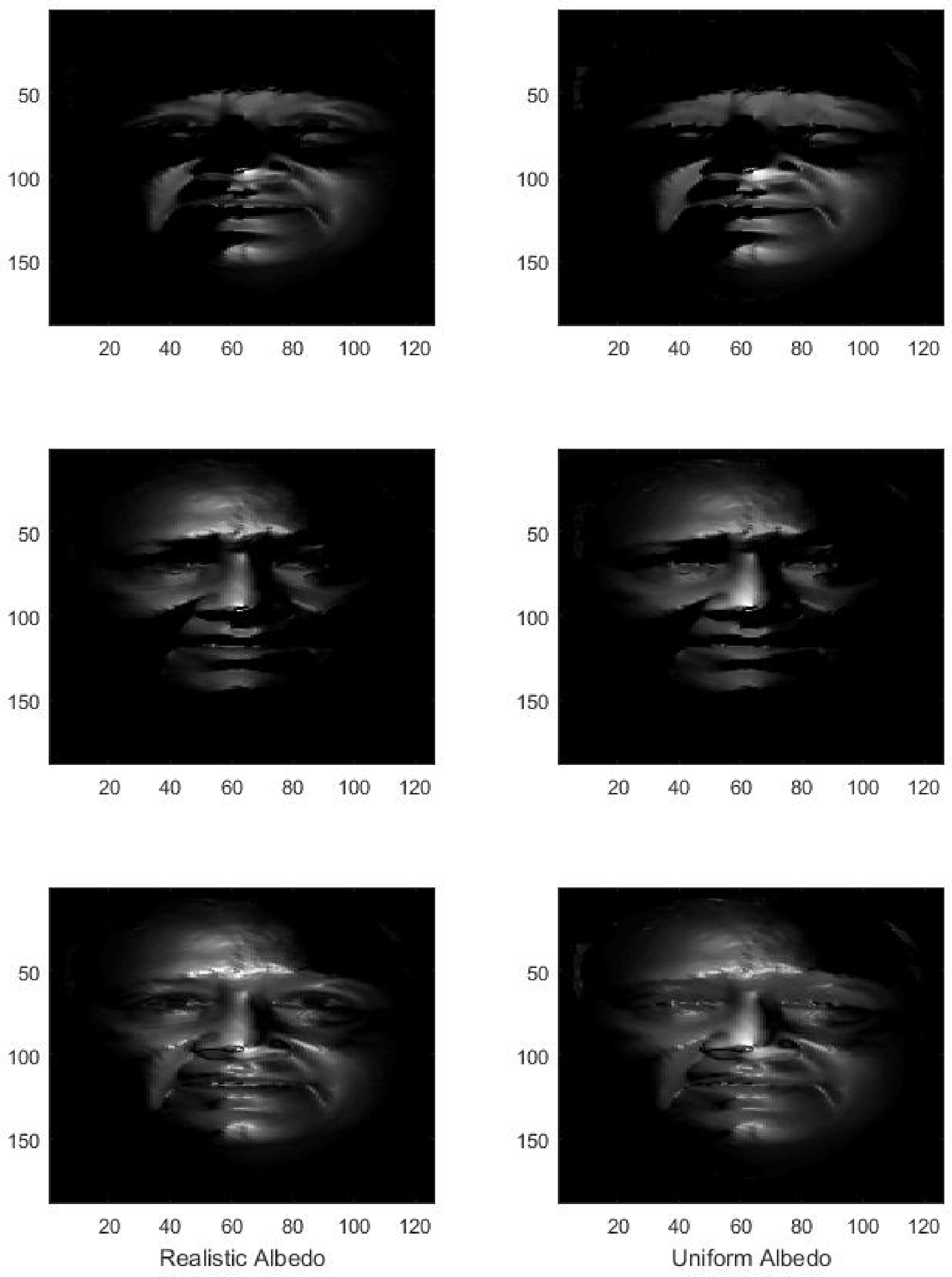


Figure 7: Rendered face images in the three lighting conditions.

The light sources are near the bottom right corner in the first case and a little away from the top left corner in the first case which is clearly visible by the shading in the images rendered in those cases. The third image is just a combination of the other two images.

Problem 5. **Photometric Stereo**

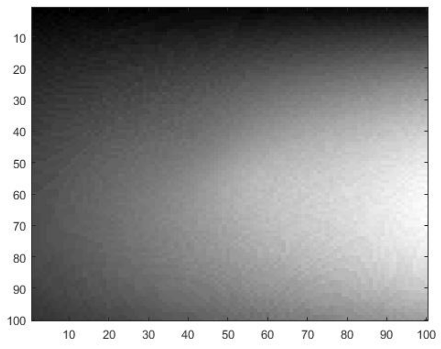
This question deals with formulating an algorithm to reconstruct a surface using the concept of photometric stereo. The input given to the algorithm will be a set of multiple images, each with their light source. The program should be composed of two parts:

(a) Read the images and the light sources. Estimate the albedo map and surface normals.

(b) Reconstruct the depth map from the normals. You could start from the scanline based shape by the integration method. If this is unsuccessful in generating the desired output, you can use the Horn integration technique in Matlab.

Try this on the synthetic dataset (synthetic_data.mat) using a subset of three images, and all four images. Include

(a) Estimated albedo map
Solution



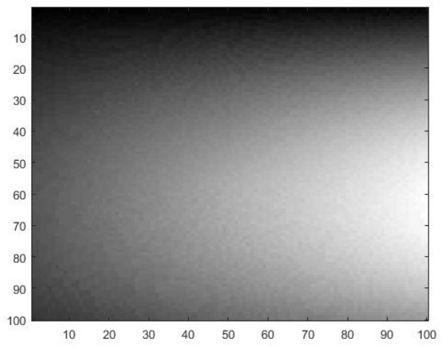


Figure 8: Shows the albedo map for both albedos

(b) Estimated surface normals by showing either needlemap (using functions meshgrid and quiver3) or three images showing three components of surface normal.

The information obtained from both 3 images and 4 images is nearly equal with 4 being a little different than 3. The gradient image in z axis shows that surface is only flat near the center.

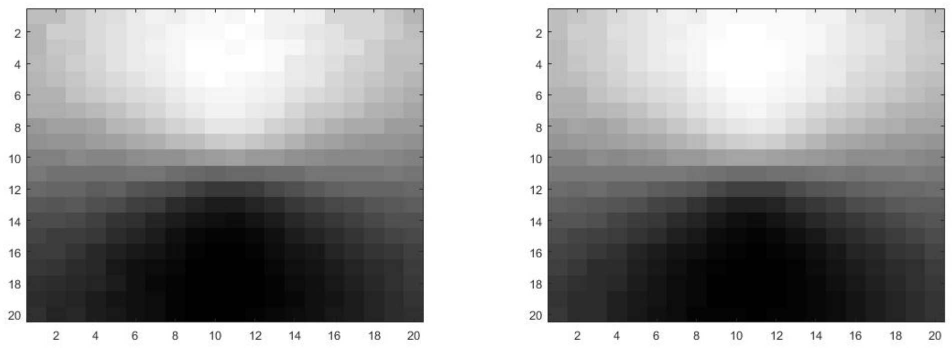


Figure 9: Shows the x component of surface normal for both albedos

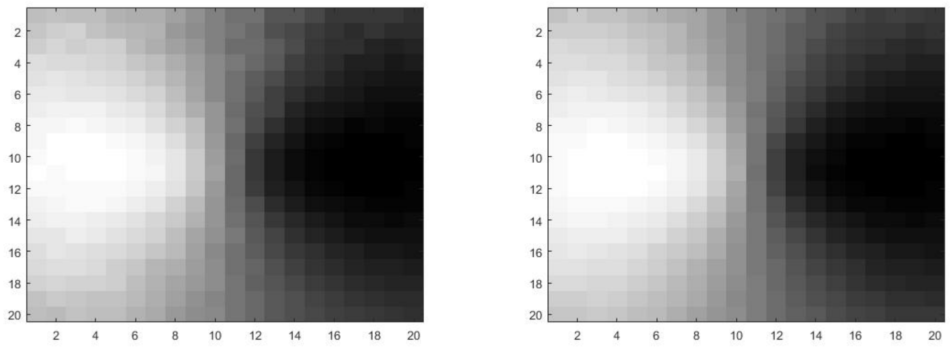


Figure 10: Shows the y component of surface normal for both albedos

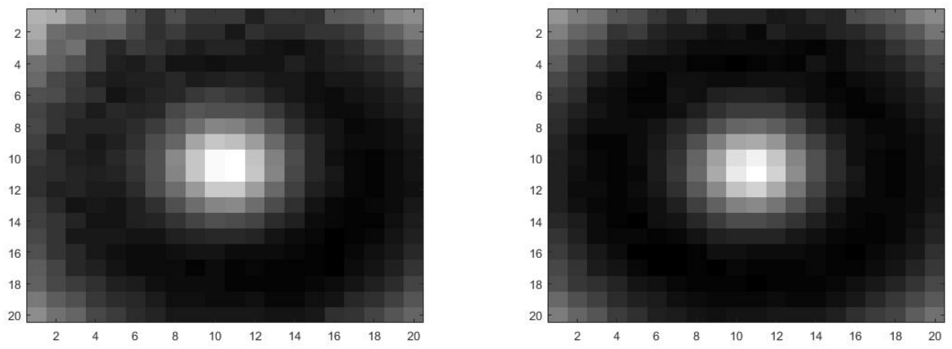


Figure 11: Shows the z component of surface normal for both albedos

(c) A wireframe of a depth map (using surf function).

Horn's method provides a better estimate of the surface as we see that in case of naive method there is some unwanted curvature near the edges which is not present in the smooth surface produced by the horn's method.

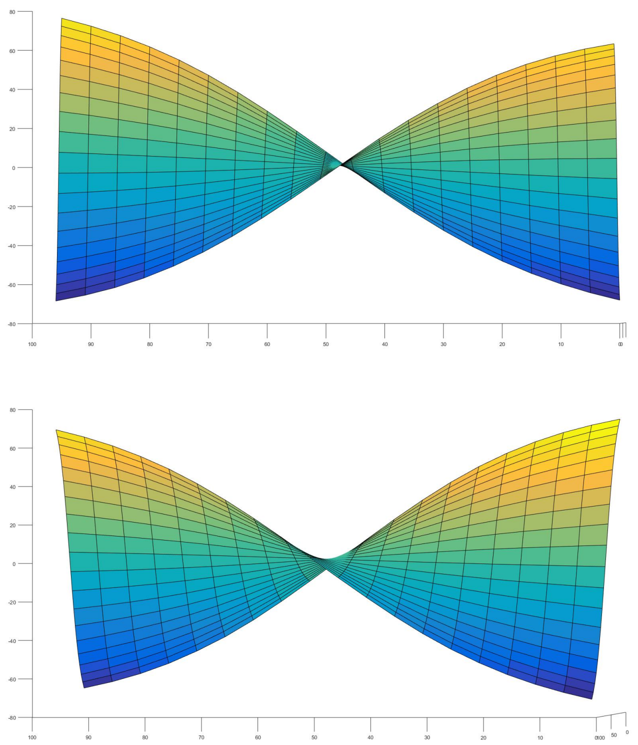


Figure 12: Naive scanline based shape for both albedos

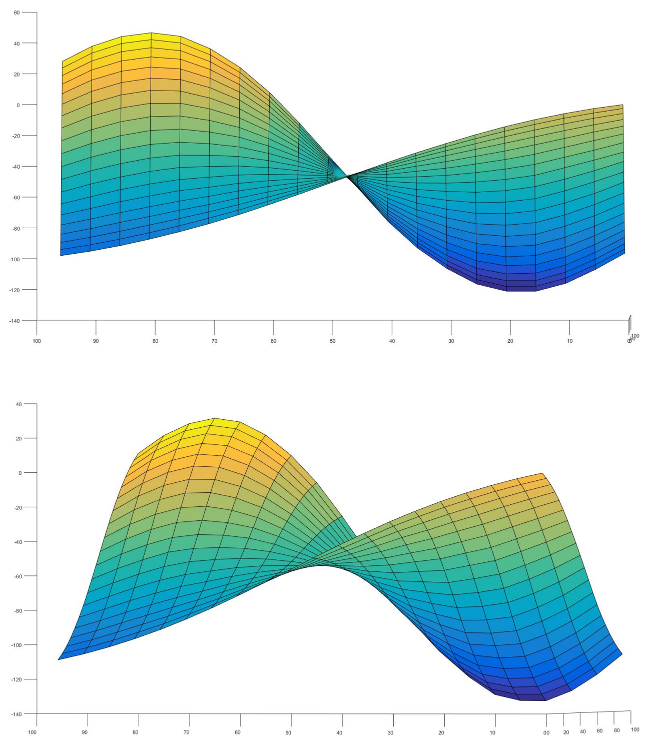


Figure 13: Horn's integration method based shape for both albedos

Appendix

```
% This script creates all the necessary matrices for problem 3.
% Then calculates the new points based of off the given points and passes
% them into the given function, plotsquare.m

%3.1: No rigid body transformation%
K = eye(3);
E = eye(4);
IO = eye(3,4);

AP1 = K * IO * E * [-1; -0.5; 2; 1];
AP2 = K * IO * E * [ 1; -0.5; 2; 1];
AP3 = K * IO * E * [ 1;  0.5; 2; 1];
AP4 = K * IO * E * [-1;  0.5; 2; 1];

disp(E);
disp(K);

save_fig = figure;
subplot(2,2,1);
title('(3.1)');
plotsquare([AP1 AP2 AP3 AP4]);
```

```matlab

%3.2: Translation%
K = eye(3);
E = [1, 0, 0, 0;
     0, 1, 0, 0;
     0, 0, 1, 1;
     0, 0, 0, 1];
IO = eye(3,4);

AP1 = K * IO * E * [-1; -0.5; 2; 1];
AP2 = K * IO * E * [ 1; -0.5; 2; 1];
AP3 = K * IO * E * [ 1;  0.5; 2; 1];
AP4 = K * IO * E * [-1;  0.5; 2; 1];

disp(E);
disp(K);

subplot(2,2,2)
title('(3.2)');
plotsquare([AP1 AP2 AP3 AP4])

%3.3: Translation and rotation%
K = eye(3);
Z = [cosd(60), -sind(60), 0;
     sind(60),  cosd(60), 0;
            0,         0, 1];
Y = [ cosd(45), 0, sind(45);
             0, 1,        0;
     -sind(45), 0, cosd(45)];
R = Y*Z;
T = [0;0;1];
E = [R T; 0,0,0,1];
IO = eye(3,4);

AP1 = K * IO * E * [-1; -0.5; 2; 1];
AP2 = K * IO * E * [ 1; -0.5; 2; 1];
AP3 = K * IO * E * [ 1;  0.5; 2; 1];
AP4 = K * IO * E * [-1;  0.5; 2; 1];

disp(E);
disp(K);

subplot(2,2,3);
title('(3.3)');
plotsquare([AP1 AP2 AP3 AP4]);

%3.4: Translation and rotation, long distance%
K = [15,  0, 0;
      0, 15, 0;
      0,  0, 1];
Z = [cosd(60), -sind(60), 0;
     sind(60),  cosd(60), 0;
            0,         0, 1];
Y = [ cosd(90), 0, sind(90);
             0, 1,        0;
     -sind(90), 0, cosd(90)];
R = Y*Z;
T = [0;0;13];
E = [R T; 0,0,0,1];
IO = eye(3,4);

AP1 = K * IO * E * [-1; -0.5; 2; 1];
```

```matlab
AP2 = K * IO * E * [ 1;  -0.5;  2;  1];
AP3 = K * IO * E * [ 1;   0.5;  2;  1];
AP4 = K * IO * E * [-1;   0.5;  2;  1];

disp(E);
disp(K);

subplot(2,2,4);
title('(3.4)');
plotsquare([AP1 AP2 AP3 AP4]);
saveas(save_fig, '3-1ImageProjs.jpg');
```

Listing 1: MATLAB source for problem 3

```matlab
load('facedata.mat');
figure
subplot(1,2,1);
imagesc(albedo);
colormap(gray);

subplot(1,2,2);
imagesc(uniform_albedo);
colormap(gray);
```

Listing 2: MATLAB source for problem 4.1

```matlab
load('facedata.mat');
figure
subplot(1,2,1);
surf(heightmap,albedo);

subplot(1,2,2);
surf(heightmap,uniform_albedo);
```

Listing 3: MATLAB source for problem 4.2

```matlab
load('facedata.mat');
[ht, wd] = size(heightmap);
u = zeros([ht - 1, wd - 1]);
v = zeros([ht - 1, wd - 1]);
w = zeros([ht - 1, wd - 1]);
X = zeros([ht - 1, wd - 1]);
Y = zeros([ht - 1, wd - 1]);
Z = zeros([ht - 1, wd - 1]);
for i=2:ht - 1
    for j = 2:wd - 1
        dx = heightmap(i, j + 1) - heightmap(i, j - 1);
        dy = heightmap(i + 1, j) - heightmap(i - 1, j);
        mag = norm([dx, dy, 1]);
        u(i, j) = -dx / mag;
        v(i, j) = -dy / mag;
        w(i, j) = 1 / mag;
        X(i, j) = j;
        Y(i, j) = i;
        Z(i, j) = heightmap(i, j);
    end
end

for j = 2:wd - 1
    dx = heightmap(1, j + 1) - heightmap(1, j - 1);
    dy = heightmap(2, j) - heightmap(1, j);
    mag = norm([dx, dy, 1]);
    u(1, j) = -dx / mag;
    v(1, j) = -dy / mag;
    w(1, j) = 1 / mag;
    X(1, j) = j;
    Y(1, j) = 1;
    Z(1, j) = heightmap(1, j);
    dx = heightmap(ht, j + 1) - heightmap(ht, j - 1);
    dy = heightmap(ht, j) - heightmap(ht - 1, j);
    mag = norm([dx, dy, 1]);
    u(ht, j) = -dx / mag;
    v(ht, j) = -dy / mag;
    w(ht, j) = 1 / mag;
    X(ht, j) = j;
    Y(ht, j) = ht;
    Z(ht, j) = heightmap(ht, j);
end

for j = 2:ht - 1
     dy = heightmap(j + 1, 1) - heightmap(j - 1, 1);
     dx = heightmap(j, 2) - heightmap(j, 1);
     mag = norm([dx, dy, 1]);
     u(j, 1) = -dx / mag;
     v(j, 1) = -dy / mag;
     w(j, 1) = 1 / mag;
     X(j, 1) = 1;
     Y(j, 1) = j;
     Z(j, 1) = heightmap(j, 1);
     dy = heightmap(j + 1, wd) - heightmap(j - 1, wd);
     dx = heightmap(j, wd) - heightmap(j, wd - 1);
     mag = norm([dx, dy, 1]);
     u(j, wd) = -dx / mag;
     v(j, wd) = -dy / mag;
     w(j, wd) = 1 / mag;
     X(j, wd) = wd;
     Y(j, wd) = j;
     Z(j, wd) = heightmap(j,wd);
 end

dy = heightmap(2, 1) - heightmap(1,1);
dx = heightmap(1,2) - heightmap(1,1);
mag = norm([dx, dy, 1]);
u(1,1) = -dx / mag;
v(1,1) = -dy / mag;
w(1,1) = 1 / mag;
X(1,1) = 1;
Y(1,1) = 1;
Z(1,1) = heightmap(1,1);

dy = heightmap(1,wd) - heightmap(1, wd-1);
dx = heightmap(2, wd) - heightmap(1, wd);
mag = norm([dx, dy, 1]);
u(1,wd) = -dx / mag;
v(1,wd) = -dy / mag;
w(1,wd) = 1 / mag;
X(1,wd) = wd;
Y(1,wd) = 1;
Z(1,wd) = heightmap(1,wd);

dy = heightmap(ht, 1) - heightmap(ht-1, 1);
dx = heightmap(ht, 2) - heightmap(ht, 1);
mag = norm([dx, dy, 1]);
u(ht, 1) = -dx / mag;
```

```matlab
v(ht, 1) = -dy / mag;
w(ht, 1) = 1 / mag;
X(ht, 1) = 1;
Y(ht, 1) = ht;
Z(ht, 1) = heightmap(ht,1);

dy = heightmap(ht, wd) - heightmap(ht - 1, wd);
dx = heightmap(ht, wd) - heightmap(ht, wd - 1);
mag = norm([dx, dy, 1]);
u(ht, wd) = -dx / mag;
v(ht, wd) = -dy / mag;
w(ht, wd) = 1 / mag;
X(ht, wd) = wd;
Y(ht, wd) = ht;
Z(ht, wd) = heightmap(ht,wd);

save('normals.mat','u','v','w');

%% Downsampling

sampling_rate = 2;
X = downsample(X,sampling_rate);
X = downsample(X',sampling_rate)';
Y = downsample(Y,sampling_rate);
Y = downsample(Y',sampling_rate)';
Z = downsample(Z,sampling_rate);
Z = downsample(Z',sampling_rate)';
heightmap = downsample(heightmap,sampling_rate);
heightmap = downsample(heightmap',sampling_rate)';
u = downsample(u,sampling_rate);
u = downsample(u',sampling_rate)';
v = downsample(v,sampling_rate);
v = downsample(v',sampling_rate)';
w = downsample(w,sampling_rate);
w = downsample(w',sampling_rate)';

figure,
quiver3(X, Y, Z, u, v, w,1);
```

Listing 4: MATLAB source for problem 4.3

```matlab
load('normals.mat');
load('facedata.mat');
[ht, wd] = size(heightmap);
d1 = zeros([ht,wd]);
d2 = zeros([ht, wd]);
s1 = cell([ht, wd]);
s2 = cell([ht, wd]);
img1 = zeros([ht, wd]);
img2 = zeros([ht, wd]);
img3 = zeros([ht, wd]);
img4 = zeros([ht, wd]);
img5 = zeros([ht, wd]);
img6 = zeros([ht, wd]);
for i = 1:ht
    for j = 1:wd
        d1(i, j) = pdist([j, i, heightmap(i, j); lightsource(1, :)],'euclidean');
        d2(i, j) = pdist([j, i, heightmap(i, j); lightsource(2, :)],'euclidean');
        s1{i, j} = lightsource(1, :) - [j, i, heightmap(i, j)];
        mag = norm(s1{i, j});
        s1{i, j} = s1{i, j} ./ mag;
        s2{i, j} = lightsource(2, :) - [j, i, heightmap(i, j)];
        mag = norm(s2{i, j});
        s2{i, j} = s2{i, j} ./ mag;

        %% Realistic Albedo
        bs = albedo(i, j) * dot([u(i, j), v(i, j), w(i, j)], s1{i, j});
        bs = max(bs, 0);
        img1(i, j) = bs / (d1(i, j) * d1(i, j));

        bs = albedo(i, j) * dot([u(i, j), v(i, j), w(i, j)], s2{i, j});
        bs = max(bs, 0);
        img2(i, j) = bs / (d2(i, j) * d2(i, j));

        img3(i, j) = (img1(i, j) + img2(i, j)) ;%/ 2;

        %% Uniform Albedo

        bs_u = uniform_albedo(i, j) * dot([u(i, j), v(i, j), w(i, j)], s1{i, j});
        bs_u = max(bs_u, 0);
        img4(i, j) = bs_u / (d1(i, j) * d1(i, j));

        bs_u = uniform_albedo(i, j) * dot([u(i, j), v(i, j), w(i, j)], s2{i, j});
        bs_u = max(bs_u, 0);
        img5(i, j) = bs_u / (d2(i, j) * d2(i, j));

        img6(i, j) = (img4(i, j) + img5(i, j)) ;%/ 2;
    end
end
figure,
subplot(3,2,1)
imagesc(img1);
colormap(gray);
subplot(3,2,3)
imagesc(img2);
colormap(gray);
subplot(3,2,5)
imagesc(img3);
colormap(gray);
subplot(3,2,2)
imagesc(img4);
colormap(gray);
subplot(3,2,4)
imagesc(img5);
colormap(gray);
subplot(3,2,6)
imagesc(img6);
colormap(gray);
```

Listing 5: MATLAB source for problem 4.4

```
close all;
clc;
clear;
load('synthetic_data.mat');
b = cell([100,100]);
a = zeros([100,100]);
n = cell([100,100]);
p = zeros([100,100]);
q = zeros([100,100]);
s = [l1;l2;l4];
psuedo_inv = inv((s'*s))*s';
im1 = im2double(im1);
im2 = im2double(im2);
im3 = im2double(im3);
im4 = im2double(im4);
```

```
for i=1:100
    for j=1:100
        e = [im1(i,j);im2(i,j);im4(i,j)];
        b{i,j} = psuedo_inv*e;
        a(i,j) = norm(b{i,j});
        n{i,j} = b{i,j}./a(i,j);
        p(i,j) = n{i,j}(1)/n{i,j}(3);
        q(i,j) = n{i,j}(2)/n{i,j}(3);
    end
end
figure,
imagesc(a),colormap(gray);

heightmap = zeros([100,100]);
for i=2:100
    heightmap(i,1) = heightmap(i-1,1)+q(i,1);
end

X = zeros([100,100]);
Y = zeros([100,100]);
u = zeros([100,100]);
v = zeros([100,100]);
w = zeros([100,100]);

%% Naive scanline integration

for i=1:100
    for j=1:100
        if(j~=1)
            heightmap(i,j) = heightmap(i,j-1)+p(i,j);
        end
        X(i,j) = j;
        Y(i,j) = i;
        u(i,j) = n{i,j}(1);
        v(i,j) = n{i,j}(2);
        w(i,j) = -n{i,j}(3);
    end
end

%% Horn's method

heightmap2 = integrate_horn2(p,q,ones([100,100]),100000,0);

%% Downsampling

sampling_rate = 5;
X = downsample(X,sampling_rate);
X = downsample(X',sampling_rate)';
Y = downsample(Y,sampling_rate);
Y = downsample(Y',sampling_rate)';
heightmap = downsample(heightmap,sampling_rate);
heightmap = downsample(heightmap',sampling_rate)';
heightmap2 = downsample(heightmap2,sampling_rate);
heightmap2 = downsample(heightmap2',sampling_rate)';
u = downsample(u,sampling_rate);
u = downsample(u',sampling_rate)';
v = downsample(v,sampling_rate);
v = downsample(v',sampling_rate)';
w = downsample(w,sampling_rate);
w = downsample(w',sampling_rate)';

figure,
imagesc(u);
colormap(gray);

figure,
imagesc(v);
colormap(gray);

figure,
imagesc(w);
colormap(gray);

figure,
quiver3(X, Y, heightmap, u, v, w);
hold on,
surf(X, Y, heightmap);
view(-35,45)

figure,
quiver3(X, Y, heightmap2, u, v, w);
hold on,
surf(X, Y, heightmap2);
view(-35,45)
```

Listing 6: MATLAB source for problem 5

Submitted by Gupta, Ajitesh
Mavalankar, Aditi Ashutosh
Tran, Derek on October 19, 2016.