CMSC 828D: Fundamentals of Computer Vision
Homework 10

Instructors: Larry Davis, Ramani Duraiswami,
Daniel DeMenthon, and Yiannis Aloimonos
Solution based on homework submitted by Haiying Liu

1. A sphere of Lambertian material of radius R lies on the ground. …

a. Write the function $z = f(x, y)$ that represents the half of the sphere that is visible from the camera.

Solution: The geometry of the sphere and camera is shown in the [Figure 1].

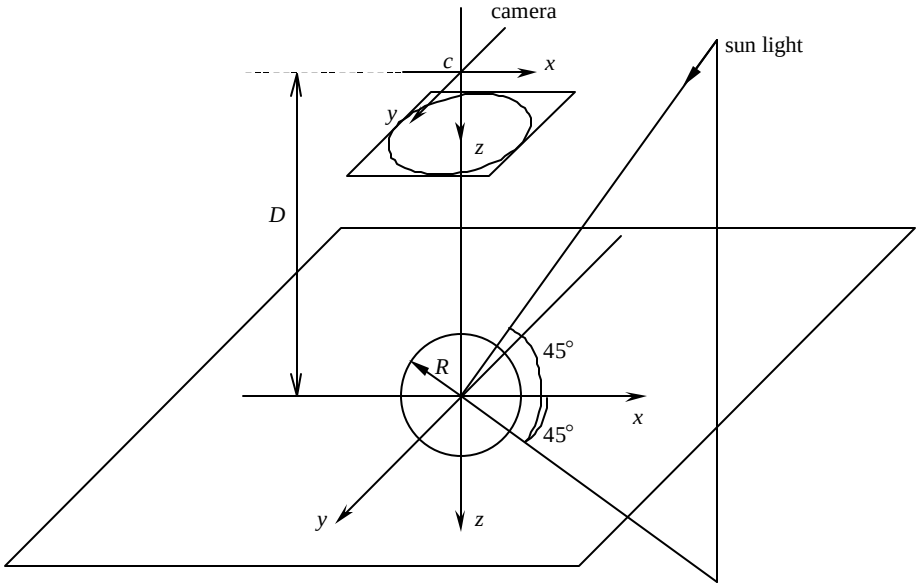


Figure 1: Geometry of the sphere and the camera.

In the coordinate originated at the center of the sphere, the coordinate (x', y', z') of a point in the surface satisfied $x'^2 + y'^2 + z'^2 = R^2$. For the upper half part of visible surface, we have negative z', i.e. $z' = f(x', y') = -\sqrt{R^2 - x'^2 - y'^2}$. The relationship between this coordinate the camera coordinate is a translation $(0,0, D)$. Thus the function $z = f(x, y) = D - \sqrt{R^2 - x^2 - y^2}$

b. Write the 3 coordinates of a unit normal to the sphere surface for a point to the sphere seen at a pixel (x, y).

Solution: Using the result from (a), we have

$$p = \frac{\partial f}{\partial x} = \frac{x}{\sqrt{R^2 - x^2 - y^2}}$$
$$q = \frac{\partial f}{\partial y} = \frac{y}{\sqrt{R^2 - x^2 - y^2}}$$

The normal is $(p, q, -1)$, and the unit normal is

$$\mathbf{n}(x, y) = \frac{(p, q, -1)}{\sqrt{p^2 + q^2 + 1}} = \frac{\left(x, y, -\sqrt{R^2 - x^2 - y^2}\right)}{R}$$

c. Write a Matlab function that gives the gray level due to …

Solution: Note the normal of light source is

$$\mathbf{n}_s = (\cos 45^\circ \cos 45^\circ, \cos 45^\circ \sin 45^\circ, -\sin 45^\circ) = (0.5, 0.5, -1/\sqrt{2}).$$

The brightest point is located at $(x_0, y_0) = (R \cdot \cos 45^\circ \cdot \cos 45^\circ, R \cdot \cos 45^\circ \cdot \sin 45^\circ) = \frac{1}{2}(R, R)$, where $R = \frac{256}{2} = 128$ pixels. According to the relationship between pixel brightness and scene brightness, we have:

$$I(x_0, y_0) = k \cos q \big|_{q=0} = 255 (\text{white}) \Rightarrow$$
$$k = \frac{255}{\cos 0} = 255$$

Then, every other point can be calculated by the known k:

$$I(x, y) = \begin{cases} k \dfrac{\mathbf{n}(x, y) \bullet \mathbf{n}_s(x, y)}{|\mathbf{n}(x, y)| \cdot |\mathbf{n}_s(x, y)|}, \text{if } \mathbf{n}(x, y) \bullet \mathbf{n}_s(x, y) \geq 0 \\ 0, \text{otherwise} \end{cases}.$$

The Matlab function is listed in appendix.

d. Write a Matlab function that generates the image of the sphere and a black background.

Solution: The Matlab function is listed in appendix. The image is shown below.

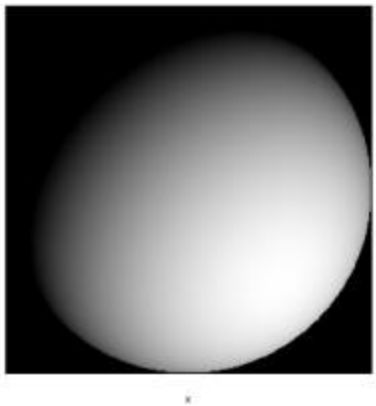


Figure 2: The image of the sphere with Lambertian surface.

2. Now a face made of plaster with Lambertian reflecting properties …

a. Transform the image into a gray level images. …

Solution: The face image is converted into gray level image and rescaled to [0, 1]. Note that the face is flipped to conform the same orientation in Problem 1, so that the equations derived in class can be used directly. The center of the cross is the tip of the nose.

b. Assume that the face is symmetrical with …

Solution: Since the face is symmetrical, the horizontal component of the normal on the curve C is zero, i.e. $p = 0$. According to the relationship between normal and pixel brightness (reflectance map), we have:

$$I = k\frac{p_s p + q_s q + 1}{\sqrt{p_s^{\,2} + q_s^{\,2} + 1}\sqrt{p^2 + q^2 + 1}}\bigg|_{p=0}$$

$$= k\frac{q_s q + 1}{\sqrt{p_s^{\,2} + q_s^{\,2} + 1}\sqrt{q^2 + 1}} \Rightarrow$$

$$I^2 = k^2\frac{(q_s q + 1)^2}{(p_s^{\,2} + q_s^{\,2} + 1)(q^2 + 1)} \Rightarrow$$

$$(q_s q + 1)^2 = \frac{I^2}{k^2}(p_s^{\,2} + q_s^{\,2} + 1)(q^2 + 1) \Rightarrow\big|_{c=\frac{I^2}{k^2}(p_s^{\,2}+q_s^{\,2}+1)}$$

$$0 = (q_s^2 - c)q^2 + 2q_s q + (1 - c) \Rightarrow$$

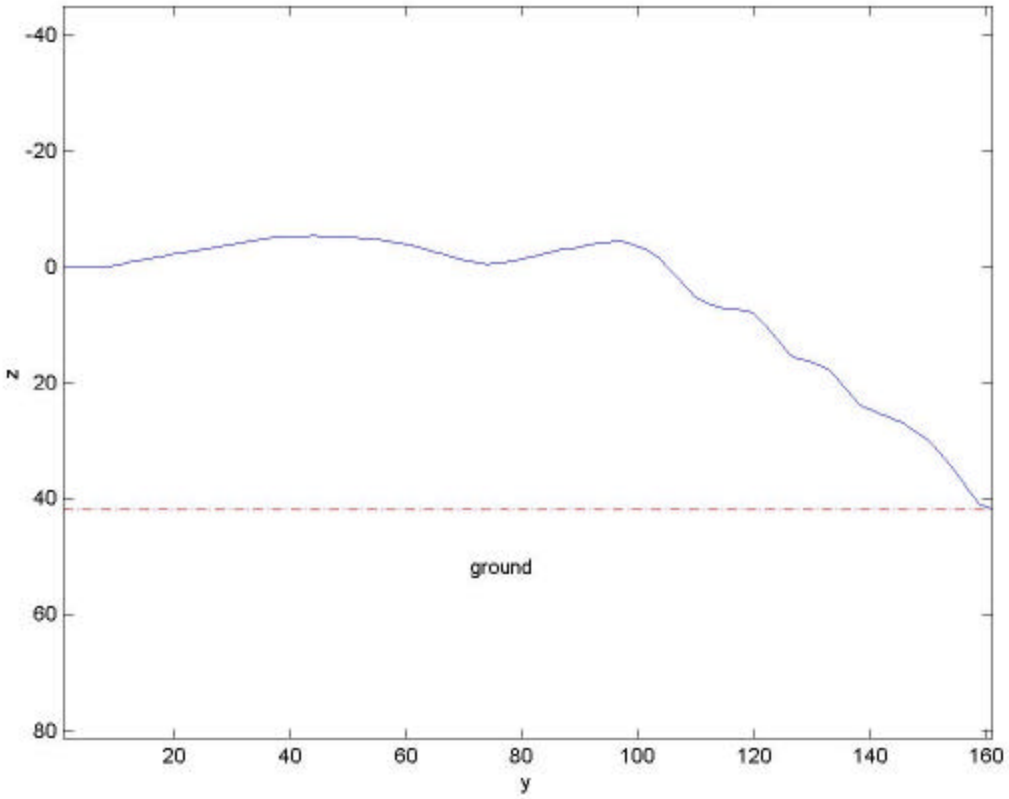
$$q_{1,2} = \frac{-2q_s \pm \sqrt{4q_s^2 - 4(q_s^2 - c)(1 - c)}}{4(q_s^2 - c)}$$

i.e. for each gray level along curve C, there are generally two possible unit normals $(0, q_1, -1)$ and $(0, q_2, -1)$.

The Matlab function is listed in appendix.

c. Find a location along curve C where the surface patch faces the camera and …

Solution: In class, we have derived that $dz = pdx + qdy$. In our case, since $p = 0$ along the curve C, $dz = qdy$. For a reference point at the tip of nose, we can calculate the relative position (dz) of other points along the curve C.



The Matlab function is listed in appendix. The profile of the face is rearranged as required and shown above.

Appendix

- **hw10_1.m**

```
function hw10_1
% Syntax: hw10_1
%
% Description: CMSC828D HW10_1
%
% Author: Haiying Liu
%   Date: Nov. 13, 2000
%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

dbstop if error

msg = nargchk(0, 0, nargin);
if (~isempty(msg))
  error(strcat('ERROR:', msg));
end

clear msg;

%=========================================================================

I = ComputeBrightness;

figure;
imshow(I, []);
xlabel('x');
ylabel('y');

print -djpeg hw10_1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function I = ComputeBrightness
% Syntax: I = ComputeBrightness
%
% Description: Compute the brightness of a lambertian sphere surface.
%
% Author: Haiying Liu
%   Date: Nov. 13, 2000
%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

dbstop if error

msg = nargchk(0, 0, nargin);
if (~isempty(msg))
  error(strcat('ERROR:', msg));
end

clear msg;
```

```
%=========================================================================

% Initialization
imSz  = 256;
white = 255;
global R;
R     = imSz / 2;
theta = 45 * pi / 180;
ns    = [cos(theta) * cos(theta), cos(theta) * sin(theta), -sin(theta)];

% Compute k
x0    = R * cos(theta) * cos(theta);
y0    = R * cos(theta) * sin(theta);

n_xy  = ComputeNorm(x0, y0);
k     = white / (n_xy * ns');

I     = ones(imSz) * (white + 1);

% Compute brightness
for row = 1:imSz
  for col = 1:imSz

    y = row - R;
    x = col - R;

    if x * x + y * y < R * R
      n_xy = ComputeNorm(x, y);
      I(row, col) = k * (n_xy * ns');
      if I(row, col) < 0
        I(row, col) = 0;
      end
    end

  end
end

% Rescale the brightness to minI~255;
dark = min(I(:))
I    = (I == white + 1) .* dark + (I < white + 1) .* I;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function n_xy = ComputeNorm(x, y)
% Syntax: n_xy = ComputeNorm(x, y)
%
% Description: Compute the norm of the point on the sphere surface.
%
% Author: Haiying Liu
%   Date: Nov. 13, 2000
%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

dbstop if error

msg = nargchk(2, 2, nargin);
```

```
if (~isempty(msg))
  error(strcat('ERROR:', msg));
end

clear msg;

%=========================================================================

global R;

temp = sqrt(R * R - x * x - y * y);
p = x / temp;
q = y / temp;
n_xy = [p, q, -1];
n_xy = n_xy ./ norm(n_xy);
```

- **hw10_2.m**

```
function hw10_2
% Syntax: hw10_2
%
% Description: CMSC828D HW10_2
%
% Author: Haiying Liu
%   Date: Nov. 14, 2000
%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

dbstop if error

msg = nargchk(0, 0, nargin);
if (~isempty(msg))
  error(strcat('ERROR:', msg));
end

clear msg;

%=========================================================================
%= Part a.

% Read the image.
face  = imread('FACE.jpeg');
face  = rgb2gray(face);
face  = double(face);

% Smooth the image by the wiener filter to reduce the noise
% introduced by JPEG compression.
face  = wiener2(face, [5, 5]);

% Rescale it to dark~white;
white = 1;
dark  = 0;
maxI  = max(face(:));
minI  = min(face(:));
face  = (face - minI) * (white - dark) / (maxI - minI) + dark;
```

```
% Flip the face so that it has the same orientation as Problem 1,
% so that the equations can be directly applied.
face  = fliplr(face);
face  = flipud(face);

nRow  = size(face, 1);
nCol  = size(face, 2);

% Show the face.
figure;
imshow(face, []);

%========================================================================
%= Part b.

% See function Compute_n below.
%Compute_n(0.6) % e.g.

%========================================================================
%= Part c.

% Find the nose tip faces the camera.
curveC_x        = nCol - 68 + 1; % plane P

noseTipRow      = 0;         % Initialization
noseTip_n       = [0, 0, 0]; % Initialization
idealNoseTip_n  = [0, 0, -1];
noseTipRowRange = [67, 77];

minDiff         = Inf;

for row = noseTipRowRange(1):noseTipRowRange(2)
  intensity = face(row, curveC_x);
  q         = Compute_q(intensity);
  for index = 1:2
    n       = [0, q(index), -1];
    diff    = norm(n / norm(n) - idealNoseTip_n);
    if diff < minDiff
      minDiff    = diff;
      noseTipRow = row;
      noseTip_n  = n;
    end
  end
end

% Mark out the nose tip.
hold on;
plot([0, nCol], [noseTipRow, noseTipRow], 'w');
plot([curveC_x, curveC_x], [0, nRow], 'w');

print -djpeg hw10_2a;

% Compute profile.
faceRowRange = [11, 171];
z            = zeros(nRow, 1);
last_n       = noseTip_n;
```

```
% From nose tip to forehead. (note the face is upside down)
for row = noseTipRow + 1:faceRowRange(2)
  intensity = face(row, curveC_x);
  q   = Compute_q(intensity);

  n1  = [0, q(1), -1];
  n2  = [0, q(2), -1];

  unit_last_n = last_n / norm(last_n);

  if norm(n1 / norm(n1) - unit_last_n) < ...
      norm(n2 / norm(n2) - unit_last_n)
    dz     = q(1);
    last_n = n1;
  else
    dz     = q(2);
    last_n = n2;
  end

  z(row) = z(row - 1) + dz;
end

% From nose tip to chin. (note the face is upside down)
for row = noseTipRow - 1:-1:faceRowRange(1)
  intensity = face(row, curveC_x);
  q   = Compute_q(intensity);

  n1  = [0, q(1), -1];
  n2  = [0, q(2), -1];

  unit_last_n = last_n / norm(last_n);

  if norm(n1 / norm(n1) - unit_last_n) < ...
      norm(n2 / norm(n2) - unit_last_n)
    dz     = -q(1);
    last_n = n1;
  else
    dz     = -q(2);
    last_n = n2;
  end

  z(row) = z(row + 1) + dz;
end

% Rearrange the profile, so that the forehead is at
% origin of (y, z) corrdinate system.
profile = z(faceRowRange(2):-1:faceRowRange(1));
profile = profile - profile(1); % take forehead as origin of (y, z)

figure;
nPoint = size(profile, 1);
plot([1:nPoint], profile);
axis ij;
axis equal;
xlabel('y');
ylabel('z');
```

```
print -djpeg hw10_2b;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function q = Compute_q(I)
% Syntax: q = Compute_q(I)
%
% Description: Compute the two q's accordint to the intensity.
%
% Author: Haiying Liu
%   Date: Nov. 14, 2000
%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

dbstop if error

msg = nargchk(1, 1, nargin);
if (~isempty(msg))
  error(strcat('ERROR:', msg));
end

clear msg;

global white;

%=========================================================================

% Compute unit normal for light source ns = [ps, qs, rs].
theta  = 45 * pi / 180;
ps     = cos(theta) * cos(theta);
qs     = cos(theta) * sin(theta);
rs     = -sin(theta);

% Compute q.
% The two norm will be in the form of [0, q, -1].
c(1)   = qs * qs - I * I;
c(2)   = -2 * qs * rs;
c(3)   = rs * rs - I * I;
q      = roots(c);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function n_unit = Compute_n(I)
% Syntax: n_unit = Compute_n(I)
%
% Description: Compute the unit normal from the intensity.
%
% Author: Haiying Liu
%   Date: Nov. 14, 2000
%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

dbstop if error
```

```
msg = nargchk(1, 1, nargin);
if (~isempty(msg))
  error(strcat('ERROR:', msg));
end

clear msg;

global white;

%=========================================================================

% Compute unit normal for light source ns = [ps, qs, rs].
q = Compute_q(I);
nRow = size(q, 1);
n = [zeros(nRow, 1), q, -ones(nRow, 1)];
for index = 1:nRow
  n_unit(index, :) = n(index, :) / norm(n(index, :));
end
```