Homework 1 - SOLUTIONS

Due Monday, January 21, 2013

Notes: Please email me your solutions for these problems (in order) as a single Word or PDF document. If you do a problem on paper by hand, please scan it in and paste it into the document (although I would prefer it typed!).

1. **(5 pts) I would like to get to know you and your interests so that I can provide the best possible educational experience for you in this course. Please describe yourself – your general background, education, interests, and goals. What specifically would you be interested in learning about computer vision? Are there any application areas that you are particularly interested in? Are you currently doing thesis research that might benefit from computer vision? Also if possible, please paste in a photo of yourself so I can start to learn names!**

2. **(10 pts) Show (by hand) that $(\mathbf{AB})^{\mathbf{T}} = \mathbf{B}^{\mathbf{T}}\mathbf{A}^{\mathbf{T}}$. Use as an example the matrices**

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$

Solution: Multiplying **AB** we get

$$\mathbf{AB} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}\begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} = \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{pmatrix}$$

So

$$(\mathbf{AB})^T = \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{21}b_{11} + a_{22}b_{21} \\ a_{11}b_{12} + a_{12}b_{22} & a_{21}b_{12} + a_{22}b_{22} \end{pmatrix}$$

The right hand side is

$$\mathbf{B}^T\mathbf{A}^T = \begin{pmatrix} b_{11} & b_{21} \\ b_{12} & b_{22} \end{pmatrix}\begin{pmatrix} a_{11} & a_{21} \\ a_{12} & a_{22} \end{pmatrix} = \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{21}b_{11} + a_{22}b_{21} \\ a_{11}b_{12} + a_{12}b_{22} & a_{21}b_{12} + a_{22}b_{22} \end{pmatrix}$$

So they are equal.

3. **(15 pts) Consider the symmetric matrix $\mathbf{C} = \begin{pmatrix} a & b \\ b & c \end{pmatrix}$. Calculate by hand**
   a. **The determinant of the matrix, |C|.**
   b. **The inverse of the matrix, $\mathbf{C}^{-1}$.**
   c. **The trace of the matrix.**
   d. **The eigenvalues of the matrix.**

Solution:

(a) $|\mathbf{C}| = ac - b^2$

(b) I learned the "cofactor" method to find an inverse: $\mathbf{A}^{-1} = \frac{1}{\det(\mathbf{A})}\begin{pmatrix} A_{11} & A_{21} & \cdots & A_{n1} \\ A_{12} & & & \vdots \\ \vdots & & & \vdots \\ A_{1n} & \cdots & \cdots & A_{nn} \end{pmatrix}$,

where A_{jk} is the cofactor of a_{jk}. For a 2x2 matrix this is

$$\mathbf{C}^{-1} = \frac{1}{\det(\mathbf{C})}\begin{pmatrix} c & -b \\ -b & a \end{pmatrix} = \frac{1}{ac-b^2}\begin{pmatrix} c & -b \\ -b & a \end{pmatrix}$$

You can (and should) verify that $\mathbf{C}^{-1}\mathbf{C} = \mathbf{I}$.

(c) The trace is just the sum of the diagonal elements; so it is just a+c.

(d) Eigenvalues and eigenvectors satisfy the equation $\mathbf{C}\,\mathbf{x} = v\,\mathbf{x}$, where $\mathbf{x}$ is a 2x1 eigenvector and v is the corresponding (scalar) eigenvalue. We can solve for the eigenvalues and eigenvectors by writing $(\mathbf{C} - v\,\mathbf{I})\mathbf{x} = 0$, and so we find v such that $\det(\mathbf{C} - v\,\mathbf{I}) = 0$.

$$\det\begin{pmatrix} a-v & b \\ b & c-v \end{pmatrix} = 0$$

or

$$(a-v)(c-v) - b^2 = 0$$

$$ac - av - cv + v^2 - b^2 = 0$$

$$v^2 + (-a-c)v + (ac-b^2) = 0$$

Solving the quadratic equation for v, we get

$$v = \frac{-(-a-c) \pm \sqrt{(-a-c)^2 - 4(ac-b^2)}}{2}$$

$$= \frac{(a+c) \pm \sqrt{a^2 + 2ac + c^2 - 4ac + 4b^2}}{2}$$

$$= \frac{(a+c) \pm \sqrt{(a-c)^2 + 4b^2}}{2}$$

Note that the quantity in the square root is always ≥ 0; thus the eigenvalues are real numbers (which is always true for a symmetric matrix).

4. (15 pts) Consider a rotation about the X axis of 1.1 radians, followed by a rotation about the Y axis of -0.5 radians, followed by a rotation about the Z axis by 0.1 radians (this order of rotations is called the "XYZ fixed angles" convention).

a. Give the 3x3 rotation matrix corresponding to the rotations above.
b. Since the rotation matrix is "orthonormal" matrix (i.e., a square matrix whose rows and columns are orthogonal unit vectors), its inverse is equal to its transpose. Show this.
c. Give the 3x3 rotation matrix where the same rotations described in part (a) are done in the opposite order; i.e., first a rotation about the Z axis of by 0.1 radians, followed by a rotation about the Y axis of -0.5 radians, followed by a rotation about the X axis by 1.1 radians (this convention is called "ZYX fixed angles"). The matrix should be different.

Solution:

(a) The rotation matrix for a rotation about the X axis is $\mathbf{R}_X(\theta_X) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_X & -\sin\theta_X \\ 0 & \sin\theta_X & \cos\theta_X \end{pmatrix}$.

Similarly, $\mathbf{R}_Y(\theta_Y) = \begin{pmatrix} \cos\theta_Y & 0 & \sin\theta_Y \\ 0 & 1 & 0 \\ -\sin\theta_Y & 0 & \cos\theta_Y \end{pmatrix}$ and $\mathbf{R}_Z(\theta_Z) = \begin{pmatrix} \cos\theta_Z & -\sin\theta_Z & 0 \\ \sin\theta_Z & \cos\theta_Z & 0 \\ 0 & 0 & 1 \end{pmatrix}$.

The combined rotation matrix is $R = R_Z(0.1)\, R_Y(-0.5)\, R_X(1.1)$.
We can do this in Matlab:

```
cx = cos(1.1);    sx = sin(1.1);
Rx = [1 0 0; 0 cx -sx; 0 sx cx];
cy = cos(-0.5);    sy = sin(-0.5);
Ry = [cy 0 sy; 0 1 0; -sy 0 cy];
cz = cos(0.1);    sz = sin(0.1);
Rz = [cz -sz 0; sz cz 0; 0 0 1];
R = Rz*Ry*Rx

R =
    0.8732   -0.4704   -0.1274
    0.0876    0.4087   -0.9085
    0.4794    0.7821    0.3981
```

(b) Using the matrix above,

```
>> R*R'

ans =
    1.0000    0.0000         0
    0.0000    1.0000    0.0000
         0    0.0000    1.0000
```

The product yields the identity matrix, so $\mathbf{R}^T$ must be the inverse of **R**. In general: Let $\mathbf{q}_i$ be the i[th] column of matrix **Q**. If **Q** is orthonormal, then the dot product of any pair of columns is zero (the dot product of a column with itself is one); i.e., $\mathbf{q}_i^T \mathbf{q}_j = \delta_{ij}$. Note that $\mathbf{q}_i^T$ is the i[th] row of matrix $\mathbf{Q}^T$. So the ij[th] entry of $\mathbf{Q}^T\,\mathbf{Q}$ is $\mathbf{q}_i^T \mathbf{q}_j = \delta_{ij}$. Thus $\mathbf{Q}^T\,\mathbf{Q} = \mathbf{I}$, meaning that $\mathbf{Q}^T = \mathbf{Q}^{-1}$.

(c) Doing the rotations in the opposite order yields a different rotation matrix:

```
>> R = Rx*Ry*Rz

R =
       0.3981   -0.7821   -0.4794
       0.8650    0.4940   -0.0876
       0.3054   -0.3798    0.8732
```

5. (15 pts) Consider the "cameraman.tif" image in Matlab. Assume that the camera that took this image can be modeled by a pinhole camera, with square pixels, and the optical center is at the center of the image. Assume that the tall building in the distance is 40 meters wide, and the distance to the building is 2 km. What is the focal length of the camera, and the field of view?

Solution:
The building is about Δx = 12 pixels wide in the image. Using similar triangles:

$$\frac{\Delta x}{f} = \frac{\Delta X}{Z}$$

f = (12 pixels)*(2000 m)/(40 m) = 600 pixels

We find the field of view knowing that the height of the image is H = 256 pixels:

$$\tan(\theta/2) = (H/2)/f$$

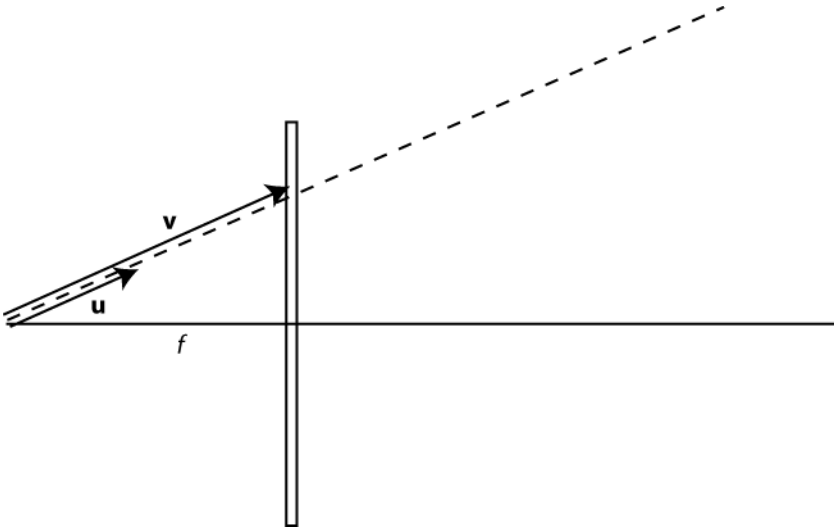
This yields θ = 0.4204 radians = 24.085 degrees. (The vertical field of view is the same because the image width is also 256 pixels.)

6. (15 pts) Using the parameters found in the previous problem, draw a "full moon" as a white circle in the "cameraman.tif" image[1]. The unit vector direction to the moon is (ux,uy,uz) = (0.0984, -0.1476, 0.9841) in camera coordinates, where we use the usual convention that +X is to the right, +Y is down, and +Z is forward. Give the code you used, and the image.

Solution:

[1] You will have to look up the angular size of the moon as seen from earth.

To find the location of the moon in the image, we take the unit vector **u** as given above, and scale it so that its "z" component is equal to the focal length *f*. This places the tip of the vector **v** on the image plane, as shown in the figure.



Since *f* is in units of pixels, **v** is also in units of pixels. So the first two elements of **v** give the pixel location in the image. We still need to offset by the center of the image.

Matlab code:

```
u = [0.0984;  -0.1476;  0.9841];
f = 600;            % Focal length in pixels
v = u*(f/u(3));     % Scale u such that v(3) = f
x0 = v(1) + 128     % Offset to center of image
y0 = v(2) + 128
```

The center of the moon is at (x,y) location (187.99, 38.01). To draw the moon as a white circle, we need to calculate the diameter in the image.
The full moon as viewed from earth has an apparent diameter of about $\theta = 0.5$ degrees (30 arc minutes). If placed at the center of the image, $\tan(\theta/2) = r/f$, where r is the radius of the moon in pixels. This comes out to be r = 2.618 pixels.

Matlab code to draw the moon:

```
% Size of moon
da = 0.5 * pi/180;      % angular diameter of moon
w2 = f*tan(da/2)       % Half width in pixels

for x=1:size(I,2)
    for y=1:size(I,1)
        if (x-x0)^2 + (y-y0)^2 <= w2^2
```

```
            I(y,x) = 255;
        end
    end
end
```

7. (25 pts) A camera observes the following 7 points, defined in WORLD coordinates (meters):

6.8158	7.8493	9.9579	8.8219	9.5890	13.2690	10.8082
-35.1954	-36.1723	-25.2799	-38.3767	-28.8402	-58.0988	-48.8146
43.0640	43.7815	40.1151	46.6153	42.2858	59.1422	56.1475

The pose of the camera with respect to the world is given by the following:

- **Translation of camera origin with respect to the world is (10,-25,40) in meters.**
- **Orientation of the camera with respect to the world is given by the angles provided in problem #4.**

(a) Compute the homogeneous transformation matrix, ${}^{C}_{W}H$, assuming that the convention being used is "XYZ fixed angles".
(b) Transform the 7 points from WORLD coordinates to CAMERA coordinates.
(c) Project the 7 points in CAMERA coordinates onto an image. Use the following parameters for the camera. Size of the image is 256 columns (width) by 170 rows (height). Center of projection is at the image center. Effective focal length is 400 pixels. Show the resulting image (hint: it should be a familiar object).

Solution:

(a) H_c_w is

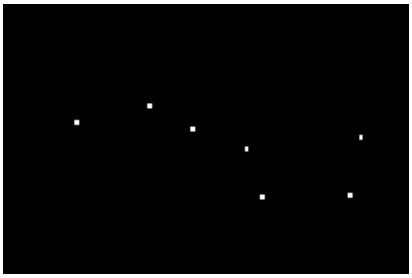
```
    0.8732   -0.4704   -0.1274   10.0000
    0.0876    0.4087   -0.9085  -25.0000
    0.4794    0.7821    0.3981   40.0000
         0         0         0    1.0000

H_w_c
    0.8732    0.0876    0.4794  -25.7187
   -0.4704    0.4087    0.7821  -16.3633
   -0.1274   -0.9085    0.3981  -37.3603
         0         0         0    1.0000
```

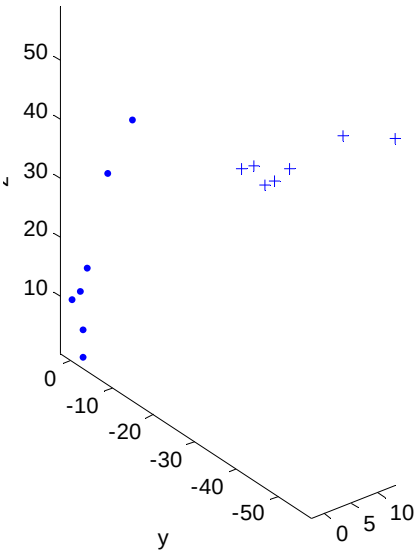
(b) Points in camera coordinates:

```
P_c
   -2.2047   -1.0439   -0.0061    0.9709    0.4005    9.1319    6.3608
   -0.2723   -0.5966   -0.0046    0.2614    0.4117   -0.0932    2.5165
   10.8875   11.9290    0.3055   14.9357    4.4510   37.2725   27.9596
    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000
```

(c) Image:



3D plot:



Matlab code:

```
clear all       % good idea to do these at the beginning of each program
close all

T = [10; -25; 40];  % Translation of camera origin in world
thetaXYZ = [1.1; -0.5; 0.1];    % XYZ angles in radians (camera to world)

% Construct rotation matrix (camera to world)
cx = cos(thetaXYZ(1));    sx = sin(thetaXYZ(1));
cy = cos(thetaXYZ(2));    sy = sin(thetaXYZ(2));
cz = cos(thetaXYZ(3));    sz = sin(thetaXYZ(3));

Rx = [1   0  0;  0  cx -sx;  0  sx cx];
Ry = [cy  0 sy;  0   1   0; -sy 0  cy];
Rz = [cz -sz 0;  sz  cz  0;  0  0   1];
R = Rz*Ry*Rx;

% Construct homogeneous transformation matrix, camera to world
H_c_w = [ R  T; 0 0 0 1 ];
disp('H_c_w'), disp(H_c_w);

% Compute the homogeneous transformation matrix, world to camera
H_w_c = inv(H_c_w);
disp('H_w_c'), disp(H_w_c);

% Here are the points in world coordinates (we append a 1 in the fourth
% element so as to make them homogeneous coordinates)
P_w = [
    6.8158    7.8493    9.9579    8.8219    9.5890   13.2690   10.8082;
  -35.1954  -36.1723  -25.2799  -38.3767  -28.8402  -58.0988  -48.8146;
   43.0640   43.7815   40.1151   46.6153   42.2858   59.1422   56.1475;
    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000];

% Transform points from world to camera coordinates
P_c = H_w_c * P_w;
disp('P_c'), disp(P_c);

% Here are the given parameters of the camera:
H = 170;        % height of image in pixels
W = 256;        % width of image in pixels
f = 400;        % focal length in pixels
cx = W/2;       % optical center
cy = H/2;

% Project the points onto the image using the perspective projection
% equations.
P_img = zeros(2,7);
for i=1:7
    P_img(1,i) = f*P_c(1,i)/P_c(3,i) + cx;
    P_img(2,i) = f*P_c(2,i)/P_c(3,i) + cy;
end

disp('P_img'), disp(P_img);

% Create a blank image and put dots at those locations
I = zeros(H,W);
for i=1:7
```

```
    r = round(P_img(2,i));     % row is y
    c = round(P_img(1,i));     % col is x
    I(r-1:r+1,c-1:c+1) = 255;
end

figure, imshow(I, []);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% We can also display in 3D.
figure, plot3(P_c(1,:), P_c(2,:), P_c(3,:), '.');
hold on
plot3(P_w(1,:), P_w(2,:), P_w(3,:), '+');
xlabel('x'), ylabel('y'), zlabel('z');
axis equal
axis vis3d
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```