Mingxuan Wang
University of California, San Diego

October 18, 2014

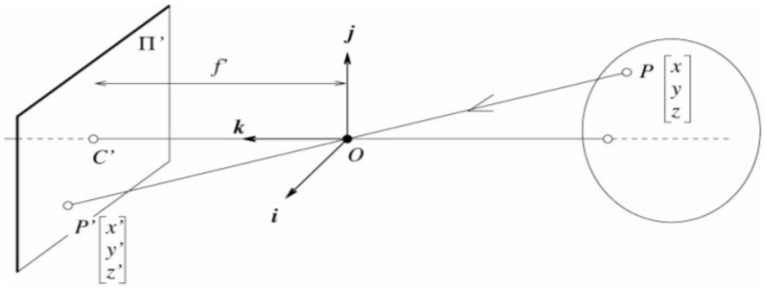
ASSIGNMENT 1

Problem 1. Perspective Projection [2 pts]

Consider a perspective projection where a point

$$P = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

is projected onto an image plane Π' represented $k = f' > 0$ by as shown in the following figure.



The first, second, and third coordinate axes are denoted by i, j, and k respectively. Consider the projection of a ray in the world coordinate system.

$$Q = \begin{bmatrix} 4 \\ 2 \\ 0 \end{bmatrix} + t \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$$

(a) What is the projection of the point when $t = -1$?

(b) What is the projected point's coordinates when $t = -\infty$? Note, to get full credit you cannot take the limit as t goes to $-\infty$.

Solutions:

(a) When $t = -1$, $Q = \begin{bmatrix} 4 \\ 1 \\ -1 \end{bmatrix}$. According to the Equation of Perspective Projection:
$x' = f'x/z$, $y' = f'y/z$ and $z' = f'$, we can calculate:
$x' = -4f'$, $y' = -f'$ and $z' = f'$.

(b) When $t = -\infty$, $z' = f'$ is always true.

And in homogeneous coordinates $Q = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$

The perspective projection transmation matrix is $H = \begin{bmatrix} f' & 0 & 0 & 0 \\ 0 & f' & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$

So $Q' = H * Q$ in homogeneous coordinates.Then transfer it to inhomogeneous coordinates, we can calculate:

$$Q' = \begin{bmatrix} 0 \\ f' \\ 1 \end{bmatrix}.$$

So the result is $x' = 0$, $y' = f'$ and $z' = f'$.

Problem 2. Thin Lens Equation [2 pts]

An illuminated arrow forms a real inverted image of itself at a distance w = 50cm, measured along the optic axis of a convex thin lens (see Figure 1). The image is just half the size of the object.

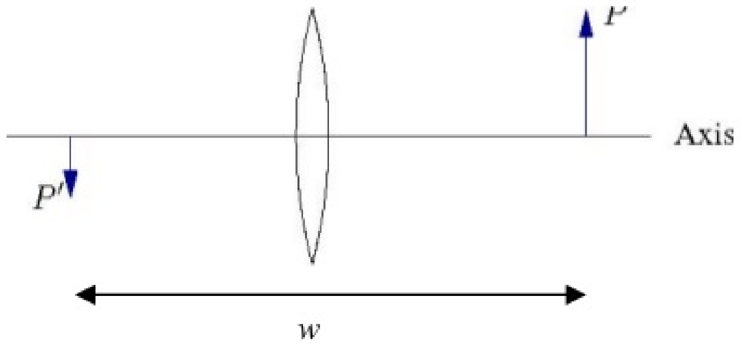


Figure 1: Problem 2 Setup

(a) How far from the object must the lens be placed?

(b) What is the focal length of the lens?

(c) Suppose the arrow moves x cm to the right while the lens and image plane remain fixed. This will result in an out of focus image; what is the radius of the corresponding blur circle formed from the tip of the arrow on the image plane assuming the diameter of the lens is d?

Solutions:

(a) According to Formula 1.5 in the textbook, we have:

$\frac{1}{z} - \frac{1}{Z} = \frac{1}{f}$, we also have:

$z + Z = 50$ and $2z = -Z$

then we can calculate:

$z = \frac{50}{3}, Z = -\frac{100}{3}$

So the lens must be placed $\frac{100}{3}$ cm from the object.

(b) According to the calculation above, we know that

$f = \frac{100}{9}$ cm

(c) assuming the new clear image is z'cm from the lens and the radius of the corresponding blur circle is r', we can calculate that

$z' = \frac{10000+300x}{600+27x}$

so that

$z' = \frac{z'}{z-z'} = d/2r'$

the result is:

$r' = \frac{3xd}{400+12x}$

Problem 3. Affine Projection [2pts]

Consider an affine camera and a line in 3D space. Consider three points (A, B, and C) on that line, and the image of those three points (a, b and c). Now consider the distance between a and b and the distance between a and c. Show that the ratio of the distance is independent of the direction of the line.

Solution:
The problem is we know that $B = \lambda A + (1-\lambda)C$, and we need to prove that $b = \lambda b + (1-\lambda)c.//$
According to affine camera model, we know that in homogeneous coordinates:

$$\begin{bmatrix} a \\ w \end{bmatrix} = \begin{bmatrix} M_{23} & t \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} A \\ 1 \end{bmatrix}$$

And in this case, $w = 1$; So there is:

$$a = M_{23}A + t, b = M_{23}B + t, c = M_{23}C + t$$

plug in these to $B = \lambda B + (1-\lambda)C$. We can get $b = \lambda a + (1-\lambda)c$.

Problem 4. Image formation and rigid body transformations [10 points]

This problem has written and coding components.
In this problem we will practice rigid body transformations and image formations through the projective and affine camera model. The goal will be to 'photograph' the following four points given by ${}^AP_1 = (-1,-0.5,2)^T$, ${}^AP_2 = (1,-0.5,2)^T$, ${}^AP_3 = (1,0.5,2)^T$, ${}^AP_4 = (-1,0.5,2)^T$ in world coordinates. To do this we will need two matrices. Recall, first, the following formula for rigid body transformation

$${}^BP = {}^B_A R\, {}^AP + {}^B O_A$$

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, ${}^BO_A$ is the origin of coordinate system A expressed in the B coordinates.The rotation and translation can be combined into a single 4*4 extrinsic parameter matrix, P_e, so that ${}^BP = P_e * {}^A P$. Once transformed, the points can be photographed using the intrinsic camera matrix, P_i which is a 3 * 4. Once these are found, the image of a point, ${}^AP$, can be calculated as $P_i * P_e * {}^A P$. We will consider four different settings of focal length, viewing angles and camera positions below.
For each part calculate or provide the following:

- the extrinsic transformation matrix.
- intrinsic camera matrix under the perspective camera assumption.
- intrinsic camera matrix under the affine camera assumption. In particular, around what point do you do the taylor series expansion?
- The actual points around which you did the taylor series expansion for the affine camera models.
- How did you arrive at these points?
- The affine projection is an approximation. Plot what happens when using a poor choice for the taylor series expansion, for example try (0; 5; 2).
- Calculate the image of the four vertices and plot using the supplied plotsquare.m/plotsquare.py function.

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

(b) Translation: ${}^{B}O_A = (0,0,1)^T$. The optical axis of the camera is aligned with the z-axis.

(c) Translation and rotation: Focal length = 1. ${}^{B}_{A}R$ encodes a 20 degrees around the z-axis and then 40 degrees around the y-axis. ${}^{B}O_A = (0,0,1)^T$.

(d) Translation and rotation, long distance: Focal length = 5. ${}^{B}_{A}R$ encodes a 20 degrees around the z-axis and then 40 degrees around the y-axis. ${}^{B}O_A = (0,0,10)^T$.

Solutions:

(a) The case is no rigid body transformation, so $P_e = \tilde{I}$.
Under the perspective camera assumption,

$$P_i(PP) = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

consider that f=1,

$$P_i(PP) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Under the affine camera assumption, I do the taylor series expansion around the point which is the center of the four points, we call it $C = (0,0,2)^T$. I choose this point because rigid body transformations preserve lengths and angles and this point is defined regardless of what the frame is.According to the affine projection, we know that:

$$P_i(AP) = \begin{bmatrix} f/z_0 & 0 & -fx_0/z_0^2 & fx_0/z_0 \\ 0 & f/z_0 & -fy_0/z_0^2 & fy_0/z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

consider that $f = 1, x_0 = 0, y_0 = 0, z_0 = 2$,

$$P_i(AP) = \begin{bmatrix} 1/2 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} (around C = (0,0,2)^T)$$

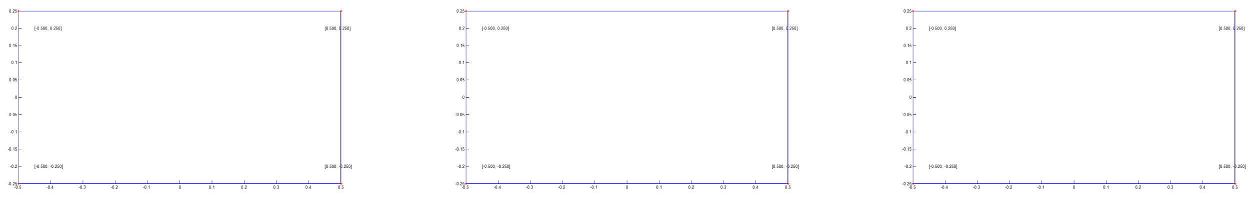
Similarly, the affine projection matrix around $(0,5,2)^T$ is:

$$P_i(AP) = \begin{bmatrix} 1/2 & 0 & 0 & 0 \\ 0 & 1/2 & -5/4 & 5/2 \\ 0 & 0 & 0 & 1 \end{bmatrix} (around(0,5,2)^T)$$

we can plot this condition in matlab as Figure 2.

(b) The case is Translation with ${}^{B}O_A = (0,0,1)^T$.
We can calculate P_e, $P_i(PP)$, $P_i(AP)$(two different points around which do the taylor series expansion) by matlab. The code is appendix.
The result is:

$$P_e = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



a b c

Figure 2: (a) proj origin, (b) affine origin around(0,0,2) (c) affine origin around(0,5,2).

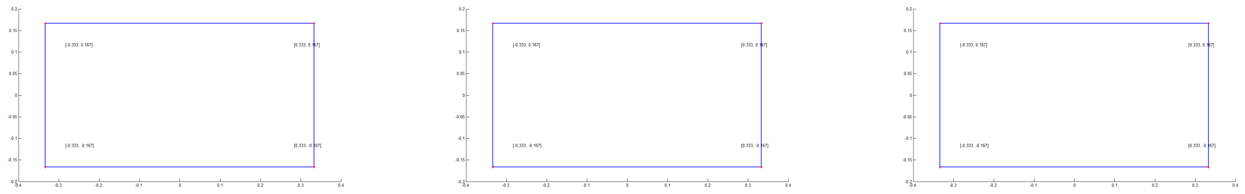
$$P_i(PP) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$P_i(AP) = \begin{bmatrix} 0.3333 & 0 & 0 & 0 \\ 0 & 0.3333 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} (around - translation - of(0,0,2)^T)$$

$$P_i(AP) = \begin{bmatrix} 0.3333 & 0 & 0 & 0 \\ 0 & 0.3333 & -0.5556 & 1.6667 \\ 0 & 0 & 0 & 1 \end{bmatrix} (around - translation - of(0,5,2)^T) \quad (1)$$

The process of calculation and plotting according to the result are both done by matlab. And the result can read from the workspace of matlab after running the code.
Note: Parameters in the code can be altered according to different conditions.(con,f,etc.)

we can plot this condition in matlab as Figure 3.



a b c

Figure 3: (a) proj origin, (b) affine origin around(0,0,2) (c) affine origin around(0,5,2).

(c) The case is Translation and rotation: Focal length = 1. ${}^B_AR$ encodes a 20 degrees around the z-axis and then 40 degrees around the y-axis. ${}^BO_A = (0,0,1)^T$.
We can calculate P_e, $P_i(PP)$, $P_i(AP)$(two different points around which do the taylor series expansion) by matlab. The code is appendix.
The result is:

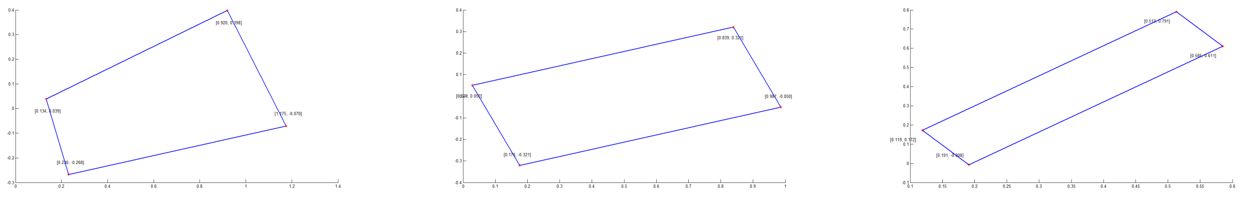
$$P_e = \begin{bmatrix} 0.7198 & -0.2620 & 0.6428 & 0 \\ 0.3420 & 0.9397 & 0 & 0 \\ -0.6040 & 0.2198 & 0.7660 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$P_i(PP) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$P_i(AP) = \begin{bmatrix} 0.3949 & 0 & -0.2005 & 0.5077 \\ 0 & 0.3949 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} (around - translationrotation - of(0,0,2)^T)$$

$$P_i(AP) = \begin{bmatrix} 0.2754 & 0 & 0.0019 & -0.0067 \\ 0 & 0.2754 & -0.3563 & 1.2939 \\ 0 & 0 & 0 & 1 \end{bmatrix} (around - translationrotation - of(0,5,2)^T)$$

we can plot this condition in matlab as Figure 4.



a b c

Figure 4: (a) proj origin, (b) affine origin around(0,0,2) (c) affine origin around(0,5,2).

(d) The case is Translation and rotation: Focal length = 5. ${}^B_AR$ encodes a 20 degrees around the z-axis and then 40 degrees around the y-axis. ${}^BO_A = (0,0,10)^T$.
We can calculate P_e, $P_i(PP)$, $P_i(AP)$(two different points around which do the taylor series expansion) by matlab. The code is appendix.
The result is:

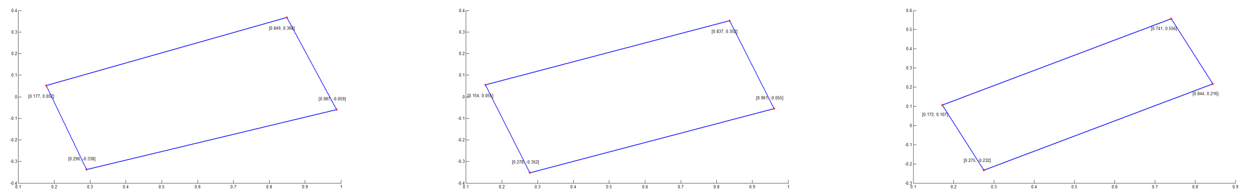
$$P_e = \begin{bmatrix} 0.7198 & -0.2620 & 0.6428 & 0 \\ 0.3420 & 0.9397 & 0 & 0 \\ -0.6040 & 0.2198 & 0.7660 & 10 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$P_i(PP) = \begin{bmatrix} 5 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$P_i(AP) = \begin{bmatrix} 0.4336 & 0 & -0.0438 & 0.5574 \\ 0 & 0.4336 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} (around - translationrotation - of(0,0,2)^T)$$

$$P_i(AP) = \begin{bmatrix} 0.3958 & 0 & 7.6584e-04 & -0.0097 \\ 0 & 0.3958 & -0.1472 & 1.8598 \\ 0 & 0 & 0 & 1 \end{bmatrix} (around - translationrotation - of(0,5,2)^T)$$

we can plot this condition in matlab as Figure 5.



a b c

Figure 5: (a) proj origin, (b) affine origin around(0,0,2) (c) affine origin around(0,5,2).

Listing 1: Codes of hw 1 pb 4

```
clear;
Rz=[cosd(20) -sind(20) 0;sind(20) cosd(20) 0;0 0 1];
Ry=[cosd(40) 0 sind(40);0 1 0;-sind(40) 0 cosd(40)];   %degree can be altered
R=Ry*Rz;
O=[0;0;10];                                             %O can be altered
bottom=[0 0 0 1];
Pe=[R O;bottom];
f=5;                                                    %f can be altered
Pipp=[f 0 0 0;0 f 0 0;0 0 1 0];
con=2;                                                  %condition can be altered

taylorw1=[0;0;2;1];taylorw2=[0;5;2;1];
taylorc1=Pe*taylorw1;
taylorc2=Pe*taylorw2;
if con==1
    x0=taylorc1(1,:);y0=taylorc1(2,:);z0=taylorc1(3,:);
end
if con==2
    x0=taylorc2(1,:);y0=taylorc2(2,:);z0=taylorc2(3,:);
end

Piap1=[f/z0 0 -f*x0/z0^2 f*x0/z0;0 f/z0 -f*y0/z0^2 f*y0/z0;0 0 0 1];

X=[-1 1 1 -1;-0.5 -0.5 0.5 0.5;2 2 2 2;1 1 1 1];
PP=Pipp*Pe*X;
AP=Piap1*Pe*X;
```

```

I=AP;

colors = 'rrrr';
I = I ./repmat(I(3, :), 3, 1);
hold on
for i = 1 : 4
line([I(1, i) I(1,mod(i, 4) + 1)], [I(2, i)I(2,mod(i, 4) + 1)],'LineWidth', 2);
plot(I(1, i), I(2, i), strcat(colors(i),'*'));
text(I(1,i)-sign(I(1,i)) * .05, I(2,i) - sign(I(2,i)) *.05,
sprintf('[%.3f, %.3f]', I(1,i), I(2,i)));
end
```

Problem 5. Image warping and merging [10 pts]

Write files computeH.m and warp.m that can be used in the following skeleton code. warp takes as inputs the original image, corners of an ad in the image, and finally, H, the homography. Note that the homography should map points from the gallery image to the sign image, that way you will avoid problems with aliasing and sub-sampling effects.

Listing 2: hw 1 pb5 skeleton code

```
    I1 = imread('F:/stadium.jpg');
    % get points from the image
    figure(10)
    imshow(I1)
    % select points on the image, preferably the corners of an ad.
    points = ginput(4);
    figure(1)
    subplot(1,2,1);
    imshow(I1);
    new_points = [1 1;1 201;101 201;101 1]; % choose your own set of points to warp yo
   H = computeH(points, new_points);
    % warp will return just the ad rectified. This will require cropping the ad out of
    % stadium image.
    warped_img = warp(I1, points, H);
    subplot(1,2,2);
    imshow(warped_img);
```

Output:

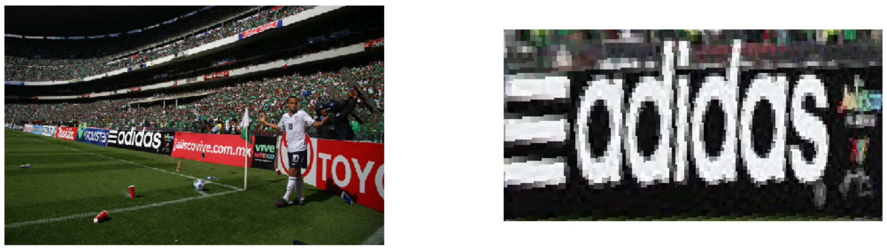


Figure 6: Problem 5 Output

Codes:

Listing 3: hw 1 pb5 computeH.m

```
function [H] = computeH(points,new_points)
    temp=points(:,1);
    points(:,1)=points(:,2);
    points(:,2)=temp;
    x1=[points(1,:),1]';x2=[points(2,:),1]';
    x3=[points(3,:),1]';x4=[points(4,:),1]';
    x=[x1,x2,x3];
    l=inv(x)*x4;
    l1=l(1,:);l2=l(2,:);l3=l(3,:);
    H1inv=[l1*x1,l2*x2,l3*x3];
    H1=inv(H1inv);

    x1p=[new_points(1,:),1]';
    x2p=[new_points(2,:),1]';
    x3p=[new_points(3,:),1]';
    x4p=[new_points(4,:),1]';
    xp=[x1p,x2p,x3p];
    lp=inv(xp)*x4p;
    l1p=lp(1,:);l2p=lp(2,:);l3p=lp(3,:);
    H2inv=[l1p*x1p,l2p*x2p,l3p*x3p];

    H=H2inv*H1;
end
```

Listing 4: hw 1 pb5 warp.m

```
function [ warped_img ] = warp( I1,points,H )
Hinv=inv(H);
for COLOR=1:1:3
    for X_my=1:1:101
        for Y_my=1:1:201
                a=Hinv*[X_my;Y_my;1];
                xh=a(1,:);
                yh=a(2,:);
                zh=a(3,:);
                xih=floor(xh/zh);yih=floor(yh/zh);
                xih_f=floor(xih);yih_f=floor(yih);
                xih_c=ceil(xih);yih_c=ceil(yih);
                u=xih_c-xih;v=yih_c-yih;
                i1=u*v*I1(xih_f,yih_f,COLOR);
                i2=u*(1-v)*I1(xih_f,yih_c,COLOR);
                i3=v*(1-u)*I1(xih_c,yih_f,COLOR);
                i4=(1-v)*(1-u)*I1(xih_c,yih_c,COLOR);
                i=i1+i2+i3+i4;
                warped_img(X_my,Y_my,COLOR)=i;

        end
    end
end
```

Submitted by Mingxuan Wang on October 18, 2014.