CV Assignment 1(Bhargav Muppalla)

Q1. Point the camera to a chessboard pattern or any known set of reference points that lie on the same plane. Capture a series of 10 images by changing the orientation of the camera in each iteration. Select any 1 image, and using the image formation pipeline equation, set up the linear equations in matrix form and solve for intrinsic and extrinsic parameters (extrinsic for that particular orientation). You will need to make measurements of the actual 3D world points, and mark pixel coordinates.

The above picture shows the object I have used to mark 6 world co-ordinates.

CV Assignment 1(Bhargav Muppalla)

Considering the corner of object as origin. So z will be 0 in real world co-ordinates.

Real world co-ordinates			Image coordinates	
x(mm)	y(mm)	z(mm)	u(pixels)	v(pixels)
0	0	0	416 ≅ 0	366 ≅ 0
			setting image co-ordinates to zero, since considering this as origin, all points will be relative to this point	
1) 25	25	0	498 ≅ 82.	426 366 60
2) 50	50	0	566 ≅ 150	496 ≅ 130
3) 75	75	0	630 ≅ 214	562 ≅ 196
4) 100	100	0	700 ≅ 284	626 ≅ 260
5) 125	125	0	766 ≅ 350	700 334
6) 150	150	0	834 ≅ 418	778 ≅ 412

CV Assignment 1(Bhargav Muppalla)

⇒ we have image coordinates and world coordinates. and we need to find projection matrix.

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ P_{31} & P_{32} & P_{33} & P_{34} \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

we need to find 12 unknown and we have collect 6 pair of points.

Re-arranging the linear equations

$$\begin{bmatrix}
X_w^{(1)} & Y_w^{(1)} & Z_w^{(1)} & 1 & 0 & 0 & 0 & 0 & -u_1 X_w^{(1)} & -u_1 Y_w^{(1)} & -u_1 Z_w^{(1)} & -u_1 \\
0 & 0 & 0 & 0 & X_w^{(1)} & Y_w^{(1)} & Z_w^{(1)} & 1 & -v_1 X_w^{(1)} & -v_1 Y_w^{(1)} & -v_1 Z_w^{(1)} & -v_1 \\
\vdots & & & & & & & & & & & \\
X_w^{(i)} & Y_w^{(i)} & Z_w^{(i)} & 1 & 0 & 0 & 0 & 0 & -u_i X_w^{(i)} & -u_i Y_w^{(i)} & -u_i Z_w^{(i)} & -u_i \\
0 & 0 & 0 & 0 & X_w^{(i)} & Y_w^{(i)} & Z_w^{(i)} & 1 & -v_i X_w^{(i)} & -v_i Y_w^{(i)} & -v_i X_w^{(i)} & -v_i \\
\vdots & & & & & & & & & & & \\
X_w^{(n)} & Y_w^{(n)} & Z_w^{(n)} & 1 & 0 & 0 & 0 & 0 & -u_n X_w^{(n)} & -u_n Y_w^{(n)} & -u_n Z_w^{(n)} & -u_n \\
0 & 0 & 0 & 0 & X_w^{(n)} & Y_w^{(n)} & Z_w^{(n)} & 1 & -v_n X_w^{(n)} & -v_n Y_w^{(n)} & -v_n Z_w^{(n)} & -v_n
\end{bmatrix}
\begin{bmatrix} P_{11} \\ P_{12} \\ P_{13} \\ P_{14} \\ P_{21} \\ P_{22} \\ P_{23} \\ P_{24} \\ P_{31} \\ P_{32} \\ P_{33} \\ P_{34} \end{bmatrix}
=
\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\boxed{A^T A p = \lambda p}$$

Eigen vector p with smallest eigen value λ of matrix $A^T A$ minimizes the loss function $L(h)$

Taking the help of matlab to solve the above matrices.

CV Assignment 1(Bhargav Muppalla)

```
a = [25,25,2,1,0,0,0,0,-2050,-2050,-164,-82;
    0,0,0,0,25,25,2,1,-1500,-1500,-120,-60;
    50,50,2,1,0,0,0,0,-7500,-7500,-300,-150;
    0,0,0,0,50,50,2,1,-6500,-6500,-260,-130;
    75,75,2,1,0,0,0,0,-16050,-16050,-428,-214;
    0,0,0,0,75,75,2,1,-14700,-14700,-392,-196;
    100,100,2,1,0,0,0,0,-28400,-28400,-568,-284;
    0,0,0,0,100,100,2,1,-26000,-26000,-520,-260;
    125,125,2,1,0,0,0,0,-43750,-43750,-700,-350;
    0,0,0,0,125,125,2,1,-41750,-41750,-668,-334;
    150,150,2,1,0,0,0,0,-62700,-62700,-836,-418;
    0,0,0,0,150,150,2,1,-61800,-61800,-824,-412];
```

```
b = transpose(a);
c = b*a;
p = min(c);
p=transpose(p)
```

Projection matrix p :

CV_HW1_Q1.mlx | p = 12×1

	1
1	-19343750
2	-19343750
3	-320900
4	-160450
5	-18553750
6	-18553750
7	-304500
8	-152250
9	-19343750
10	-19343750
11	-320900
12	-160450

We can now run QR factorization to get calibration matrix K and rotation matrix R

```
rk = [-19343750, -19343750,-320900;-18553750,-18553750,-304500;-19343750,-19343750,-320900];
[R,K] = qr(rk)
```

```
R = 3x3
     -0.5852    0.8107   -0.0169
     -0.5613   -0.4200   -0.7131
     -0.5852   -0.4078    0.7009

K = 3x3
10^7 ×
      3.3055    3.3055    0.0547
           0    0.0000   -0.0001
           0         0   -0.0002
```

CV Assignment 1(Bhargav Muppalla)

Calculating translation matrix using following equation

$$\mathbf{t} = K^{-1}\begin{bmatrix} p_{14} \\ p_{24} \\ p_{34} \end{bmatrix}$$

Here pr is column vector with values from last column of projection matrix

```
pr = [-160450;-152250;-160450];
t = ki*pr
```

```
t = 3×1
10^13 ×
    1.0959
   -1.0959
    0.0000
```

T is the translation vector.

Q2. Select any pair of images from the set-in problem 1 above. Compute the homography between those two images.

CV Assignment 1(Bhargav Muppalla)

	Real world coordinates of image 2		Image co-ordinates of image 2	
	X (mm)	Y (mm)	U (pixels)	V (pixels)
	0	0		
1)	25	25	75	50
2)	50	50	139	110
3)	75	75	190	172
4)	100	100	223	197
5)	125	125	291	250
6)	150	150	374	333

Representation of System of equations that is used to derive homography.

$$\begin{bmatrix} x_s^{(1)} & y_s^{(1)} & 1 & 0 & 0 & 0 & -x_d^{(1)} x_s^{(1)} & -x_d^{(1)} y_s^{(1)} & -x_d^{(1)} \\ 0 & 0 & 0 & x_s^{(1)} & y_s^{(1)} & 1 & -y_d^{(1)} x_s^{(1)} & -y_d^{(1)} y_s^{(1)} & -y_d^{(1)} \\ x_s^{(i)} & y_s^{(i)} & 1 & 0 & 0 & 0 & -x_d^{(i)} x_s^{(i)} & -x_d^{(i)} y_s^{(i)} & -x_d^{(i)} \\ 0 & 0 & 0 & x_s^{(i)} & y_s^{(i)} & 1 & -y_d^{(i)} x_s^{(i)} & -y_d^{(i)} y_s^{(i)} & -y_d^{(i)} \\ x_s^{(n)} & y_s^{(n)} & 1 & 0 & 0 & 0 & -x_d^{(n)} x_s^{(n)} & -x_d^{(n)} y_s^{(n)} & -x_d^{(n)} \\ 0 & 0 & 0 & x_s^{n} & y_s^{n} & 1 & -y_d^{(n)} x_s^{(n)} & -y_d^{(n)} y_s^{(n)} & -y_d^{(n)} \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

I have used 2 images . image 1 co-ordinates are same as the one used in question 1. The above picture has co-ordinates of second image.

CV Assignment 1(Bhargav Muppalla)

Computing transpose of A matrix to get eigen vector by using the following relation.

$$\boxed{A^T A h = \lambda h}$$

Eigenvector h with smallest eigenvalue λ of matrix $A^T A$ minimizes the loss function L(h).

Using matlab for the calculation.

```
a = [82,60,1,0,0,0,-6150,-4500,-75;
    0,0,0,82,60,1,-4000,-3000,-50;
    150,130,1,0,0,0,-20850,-18070,-139;
    0,0,0,150,130,1,-16500,-14300,-110;
    214,196,1,0,0,0,-40660,-37240,-190;
    0,0,0,214,196,1,-36808,-33712,-172;
    284,260,1,0,0,0,-63332,-57980,-223;
    0,0,0,284,260,1,-55948,-51220,-197;
    350,334,1,0,0,0,-101850,-97194,-291;
    0,0,0,350,334,1,-87500,-83500,-250;
    418,412,1,0,0,0,-156332,-154088,-374;
    0,0,0,418,412,1,-139194,-137196,-333
    ];

b = transpose(a);

c = b*a
```

```
c = 9x9
10^10 ×
     0.0000    0.0000    0.0000         0         0         0   -0.0131   -0.0126   -0.0000
     0.0000    0.0000    0.0000         0         0         0   -0.0126   -0.0121   -0.0000
     0.0000    0.0000    0.0000         0         0         0   -0.0000   -0.0000   -0.0000
          0         0         0    0.0000    0.0000    0.0000   -0.0115   -0.0111   -0.0000
          0         0         0    0.0000    0.0000    0.0000   -0.0111   -0.0106   -0.0000
          0         0         0    0.0000    0.0000    0.0000   -0.0000   -0.0000   -0.0000
    -0.0131   -0.0126   -0.0000   -0.0115   -0.0111   -0.0000    7.2754    7.0336    0.0201
    -0.0126   -0.0121   -0.0000   -0.0111   -0.0106   -0.0000    7.0336    6.8054    0.0193
    -0.0000   -0.0000   -0.0000   -0.0000   -0.0000   -0.0000    0.0201    0.0193    0.0001
```

We get homogenous matrix by fetching minimum from each column

CV Assignment 1(Bhargav Muppalla)

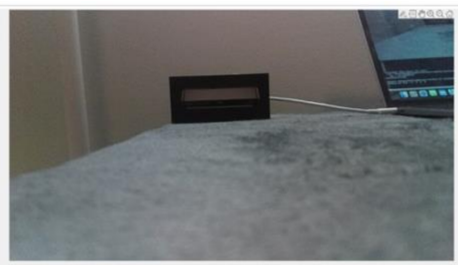
```
H = min(c);
H = transpose(H)
```

```
H = 9×1
    -131313604
    -125941864
       -389174
    -115377236
    -110718776
       -339950
    -131313604
    -125941864
       -389174
```

3. Write a MATLAB script to find the real world dimensions (e.g. diameter of a ball, side length of a cube) of an object using perspective projection equations. Validate using an experiment where you image an object using your camera from a specific distance (choose any distance but ensure you are able to measure it accurately) between the object and camera.

I am calculating the area of a rectangular box.

```
img = imread("/Users/bhargavmuppalla/Desktop/chessBoardImage2.
imshow(img)
z = 389;
fx = 1546.6;
fy = 1425.3;
[p1,p2] = ginput(3);
xw1 = z * (p1(1)/fx);
xw2 = z * (p1(2)/fx);
xw3 = z * (p1(3)/fx);
yw1 = z * (p2(1)/fy);
yw2 = z * (p2(2)/fy);
yw3 = z * (p2(3)/fy);

length = sqrt((yw3 - yw1)^2 + (xw3 - xw1)^2);
breadth = sqrt((yw2 - yw1)^2 + (xw2 - xw1)^2);

Area = length * breadth;

fprintf("Area of a rectangular box:"+ Area)
```

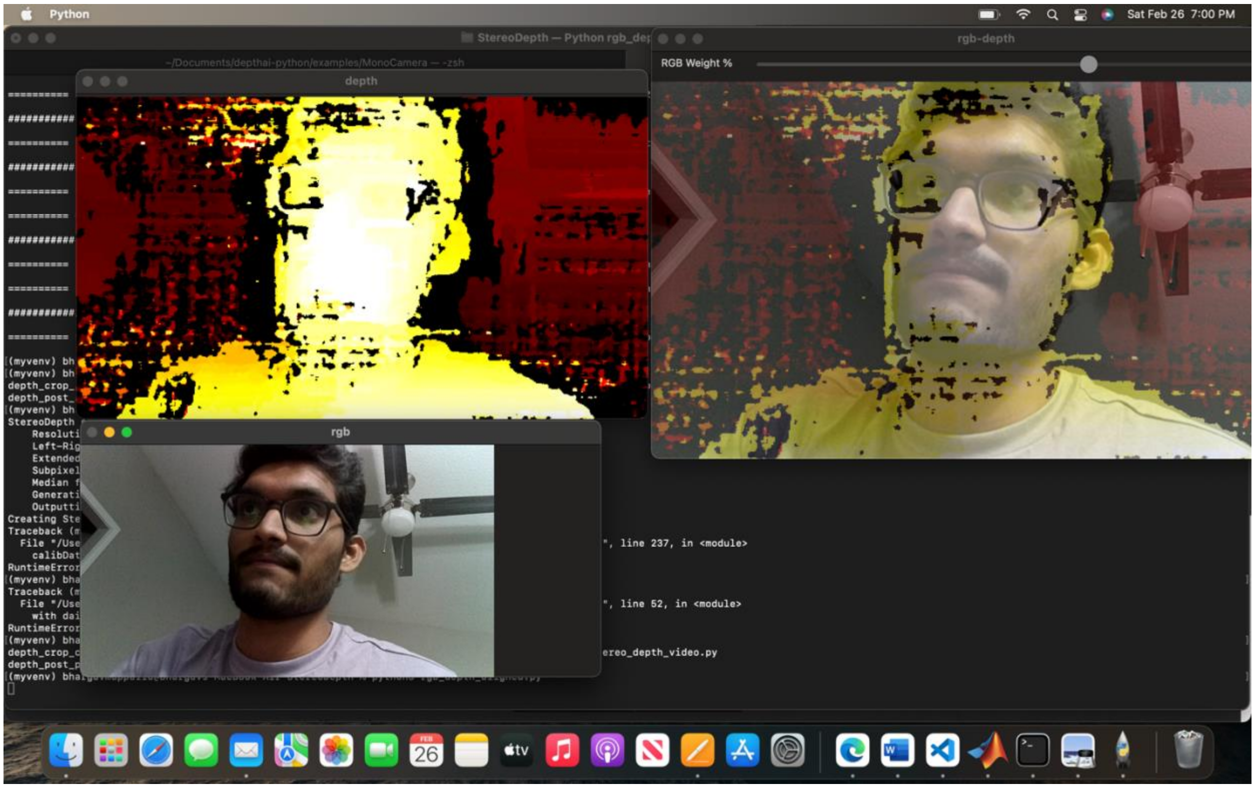


```
p1 = 3×1
         690
         702
        1116

p2 = 3×1
    292.0000
    484.0000
    280.0000

Area of a rectangular box:5626.6219
```

5. Setup your application to show a RGB stream from the mono camera and a depth map stream from the stereo camera simultaneously. Is it feasible? What is the maximum frame rate and resolution achievable?

It is feasible to run RGB stream from mono camera and a depth map stream from stereo camera. Maximum framerate is 60fps. And resolution is 1920 X 1080.

CV Assignment 1(Bhargav Muppalla)



6. Run the camera calibration tutorial. Compare the output with answers from Part A and Matlab calibration exercise.

[bhargavmuppalla/CV: Assignment files of computer vision (github.com)](https://github.com/bhargavmuppalla/CV)

uploaded all MatLab scripts in the above link.