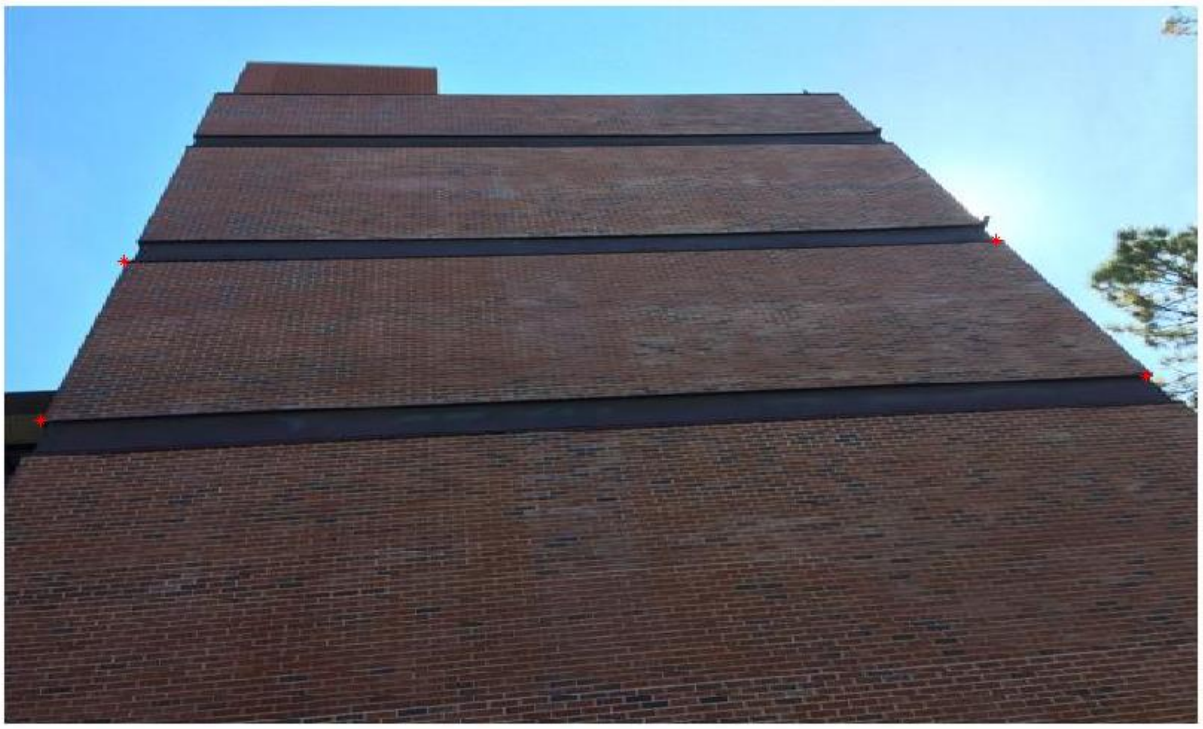
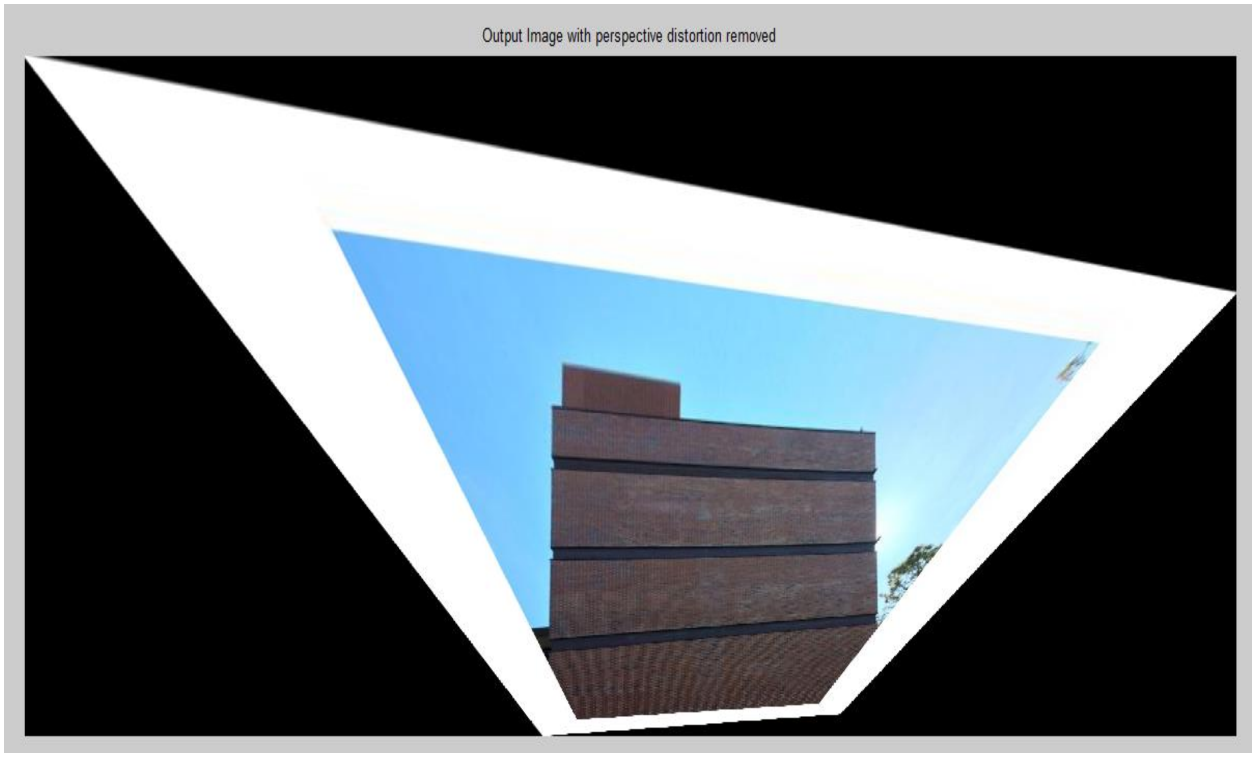
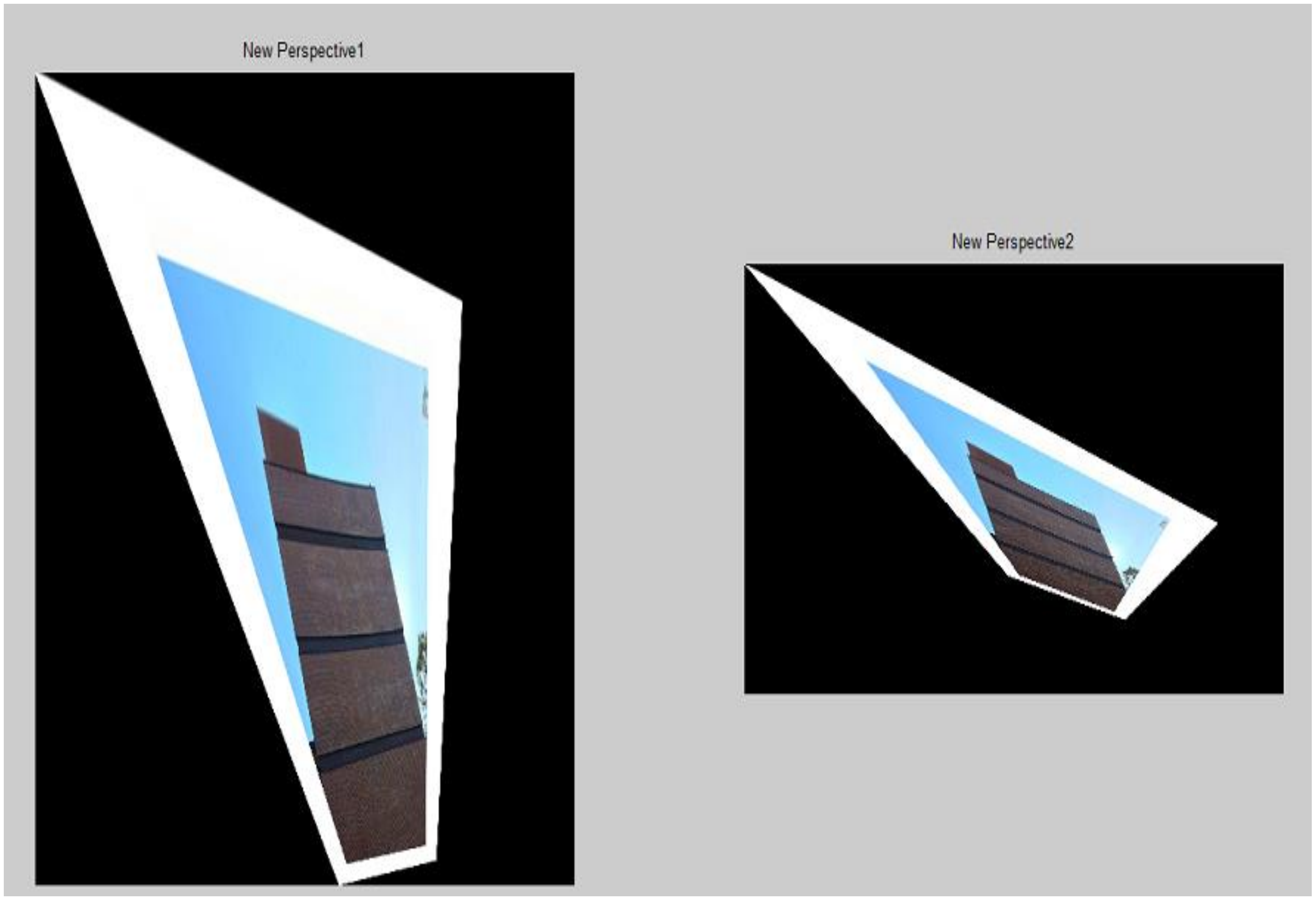
Ans-1: Input Image with pts of correspondences selected



Output Image with perspective distortion removed



Two new perspectives from the original perspective can be generated by applying different kinds of transformations to it like affine, non-uniform shear, rotation transforms etc. Here I applied the first two over the obtained image above and produced the following outputs:



Ans-2: For different sets of points of correspondences we obtained the following average projection error values(PEn) and camera calibration matrices(CamMatn) which has been included in the stats folder in the submission zip file.

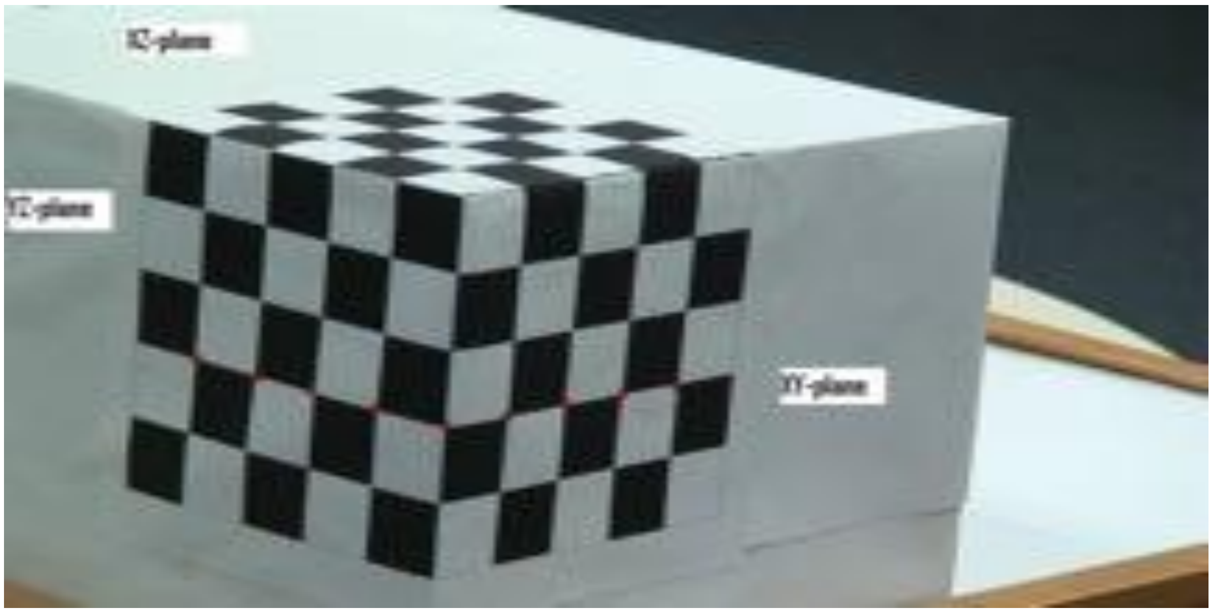
For n=6 we have camera calibration matrix as CamMatrix1 and PE1

For n=12 we have camera calibration matrix as CamMatrix2 and PE2

For n=18 we have camera calibration matrix as CamMatri3 and PE3

The **intrinsic and extrinsic parameter values** have also been calculated in the accuracy test code for camera calibration where predetermined nine points of the image have been selected as shown.

Image for Testing Accuracy of Calibration Matrix: (Marked red points used for testing in the order specified that is from Right to Left)



Codes: **Version of MATLAB© Code for n=6:*'assignment2_2w6pts.m'***

```
clc;
close all;
clear all;
%%

input=imread('Image.JPG');
inter3=Get2DPoints(input,18);
wcs3=[1 2 1 4 0 0 5 4 2 3 0 0 2 1 4 3 0 0;
      1 2 0 0 2 4 5 4 0 0 1 2 1 2 0 0 4 5;
      0 0 1 2 1 4 0 0 2 3 1 2 0 0 1 1 3 5];
%the selected 18 world coordinates in specified order
are:(1,1,0),(2,2,0),(1,0,1),(4,0,2),(0,2,1),(0,4,4)%
%-------------------------------------------------------
:(5,5,0),(4,4,0),(2,0,2),(3,0,3),(0,1,1),(0,2,2)%
```

```
%-------------------------------------------------------
:(2,1,0),(1,2,0),(4,0,1),(3,0,1),(0,4,3),(0,5,5)%
CamMat3=getCameraMatrix(inter3,wcs3);
save CamMat3;
```

Some abstractions that have been used in all these versions including the code for testing the accuracy of the camera calibration matrices obtained.

```
function P = getCameraMatrix(x, X)

 A = getA(x, X);
 [U, S, V] = svd(A);
 P = V(:, size(V,2));
 P = reshape(P, 4, 3)';

 return
```

```
function A = getA(x, X)
 n = size(x, 2);

 h = 1;
 for k =1:n
A(h, :) = [X(1, k) X(2, k) X(3, k) 1 0 0 0 0 -x(1, k)*X(1, k) -x(1, k)*X(2,
k) -x(1, k)*X(3, k) -x(1, k)];
A(h + 1, :) = [0 0 0 0 X(1, k) X(2, k) X(3, k) 1 -x(2, k)*X(1, k) -x(2,
k)*X(2, k) -x(2, k)*X(3, k) -x(2, k)];

 h = h + 2;


 end
 return;

 %DLT is being implemented here for a camera calibration matrix%
```

Version of MATLAB© Code for n=12:*'assignment2_2w12pts.m'*

```
clc;
close all;
clear all;
%%

input=imread('Image.JPG');
inter2=Get2DPoints(input,12);
wcs2=[1 2 1 4 0 0 5 4 2 3 0 0;
      1 2 0 0 2 4 5 4 0 0 1 2;
      0 0 1 2 1 4 0 0 2 3 1 2];
%the selected 12 world coordinates in specified order
are:(1,1,0),(2,2,0),(1,0,1),(4,0,2),(0,2,1),(0,4,4)%
%-------------------------------------------------------
:(5,5,0),(4,4,0),(2,0,2),(3,0,3),(0,1,1),(0,2,2)%
```

```
CamMat2=getCameraMatrix(inter2,wcs2);
save CamMat2;
```

Version of MATLAB© Code for n=18:*'assignment2_2w18pts.m'*

```
clc;
close all;
clear all;
%%

input=imread('Image.JPG');
inter3=Get2DPoints(input,18);
wcs3=[1 2 1 4 0 0 5 4 2 3 0 0 2 1 4 3 0 0;
      1 2 0 0 2 4 5 4 0 0 1 2 1 2 0 0 4 5;
      0 0 1 2 1 4 0 0 2 3 1 2 0 0 1 1 3 5];
%the selected 18 world coordinates in specified order
are:(1,1,0),(2,2,0),(1,0,1),(4,0,2),(0,2,1),(0,4,4)%
%-------------------------------------------------------
:(5,5,0),(4,4,0),(2,0,2),(3,0,3),(0,1,1),(0,2,2)%
%-------------------------------------------------------
:(2,1,0),(1,2,0),(4,0,1),(3,0,1),(0,4,3),(0,5,5)%
CamMat3=getCameraMatrix(inter3,wcs3);
save CamMat3;
```

Version of MATLAB© Code for testing the accuracy of camera calibration matrix obtained, computing projection error values and computing the intrinsic & extrinsic parameters:***'AccuracytestCalibMat.m'***

```
clc;
close all;
clear all;
%%
input=imread('Image.JPG');
inter4=Get2DPoints(input,9);
%the marked test points are stated below%
wcs4=[4 3 2 1 0 0 0 0 0;
      3 3 3 3 3 3 3 3 3;
      0 0 0 0 0 1 2 3 4
      1 1 1 1 1 1 1 1 1];
load CamMat3;%Camera Matrix obtained from 18 points of corresspondences%
load CamMat2;%Camera Matrix obtained from 12 points of corresspondences%
load CamMat1;%Camera Matrix obtained from 6 points of corresspondences%
hprime3=CamMat3*wcs4;
hprime2=CamMat2*wcs4;
hprime1=CamMat1*wcs4;
%Part taken for 18 points of corresspondences%
%%
cs4=hprime3(1:2,:);
h3(1,:)=hprime3(3,:);
h3(2,:)=hprime3(3,:);
cs4=cs4./h3;
nx1=cs4(1,:);ny1=cs4(2,:);
ox1=inter4(1,:);oy1=inter4(2,:);
PE3=sqrt((ox1-nx1).^2+(oy1-ny1).^2);
PE3=mean2(PE3);%this is projection error value for 18 points of
corresspondences%
save PE3;
%%
```

```
cs4=hprime2(1:2,:);
h3(1,:)=hprime2(3,:);
h3(2,:)=hprime2(3,:);
cs4=cs4./h3;
nx1=cs4(1,:);ny1=cs4(2,:);
ox1=inter4(1,:);oy1=inter4(2,:);
PE2=sqrt((ox1-nx1).^2+(oy1-ny1).^2);
PE2=mean2(PE2);%this is projection error value for 12 points of
corresspondences%
save PE2;
%%
cs4=hprime1(1:2,:);
h3(1,:)=hprime1(3,:);
h3(2,:)=hprime1(3,:);
cs4=cs4./h3;
nx1=cs4(1,:);ny1=cs4(2,:);
ox1=inter4(1,:);oy1=inter4(2,:);
PE1=sqrt((ox1-nx1).^2+(oy1-ny1).^2);
PE1=mean2(PE1);%this is projection error value for 6 points of
corresspondences%
save PE1;
%%
%Obtained decreasing values i.e PE1>PE2>PE3 since 1.3804>1.0752>0.7410%
%Precise Order to execute code: first execute codes with 6,12 and 18 pts and
then execute this code%
[Q1,R1]=qr(CamMat3(:,1:3));
R1=R1./R1(3,3);
%Intrinsic Camera Parameters obtained by the upper triangular matrix after rq
decomposition user defined function%
alphax=R1(1,1);alphay=R1(2,2);s=R1(1,2);tx=R1(1,3);ty=R1(2,3);
%Extrinsic Parameters namely the rotation matrix and the translation vector
we have in matrix q1%
RotationMatrix=Q1(1:2,1:2);
TranslationVector=Q1(1:2,3);
```

Answers:

a) Intrinsic Parameters: aplhax= -493.6328 alphay= -795.7858 s= 116.7136 tx= 416.8931 ty= 196.2177

Extrinsic Parameters: Rotation Matrix= -0.991942755706745 -0.126680730422914

0.126681300564209 -0.991943394410882

Translation Matrix= -0.00124977615072609

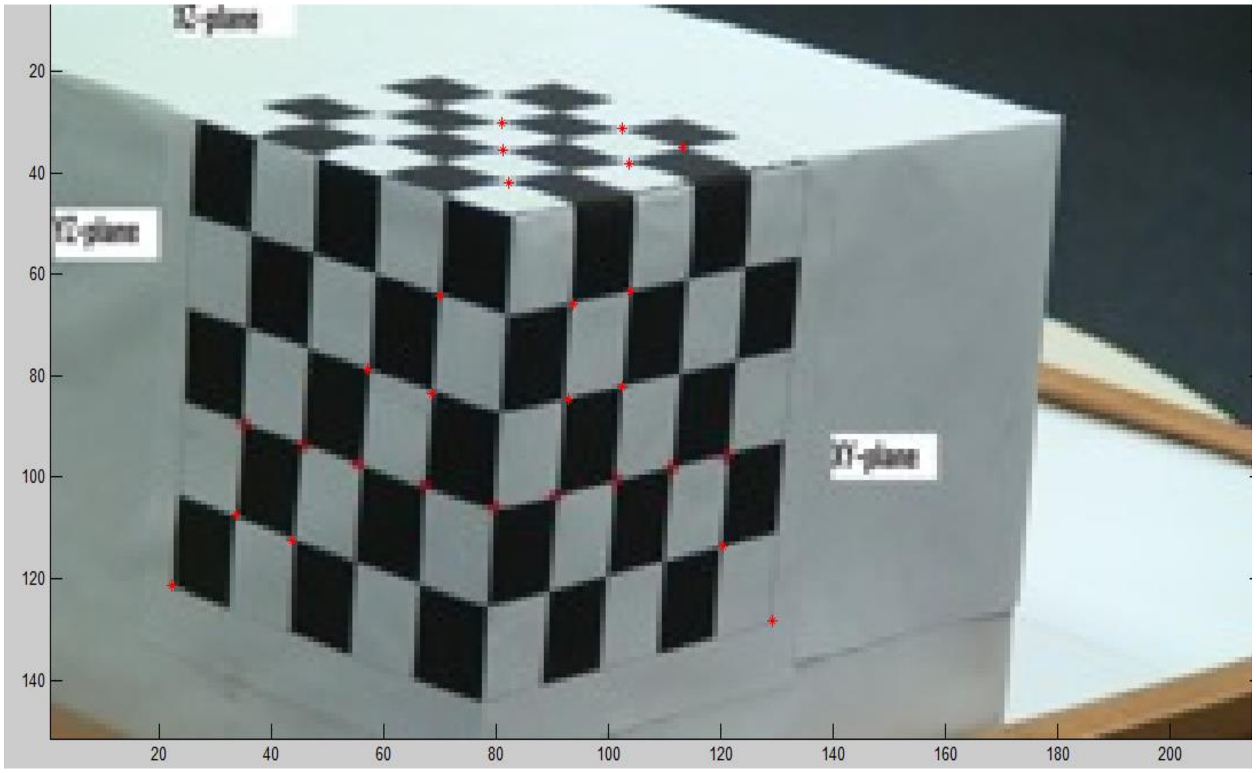
-0.000387778259903065

b) Average Projection Error values obtained in decreasing form i.e PE1(for 6 pts)>PE2(for 12 pts)>PE3(for 18 pts).

PE1=1.3804 PE2=1.0752 PE3=0.7410

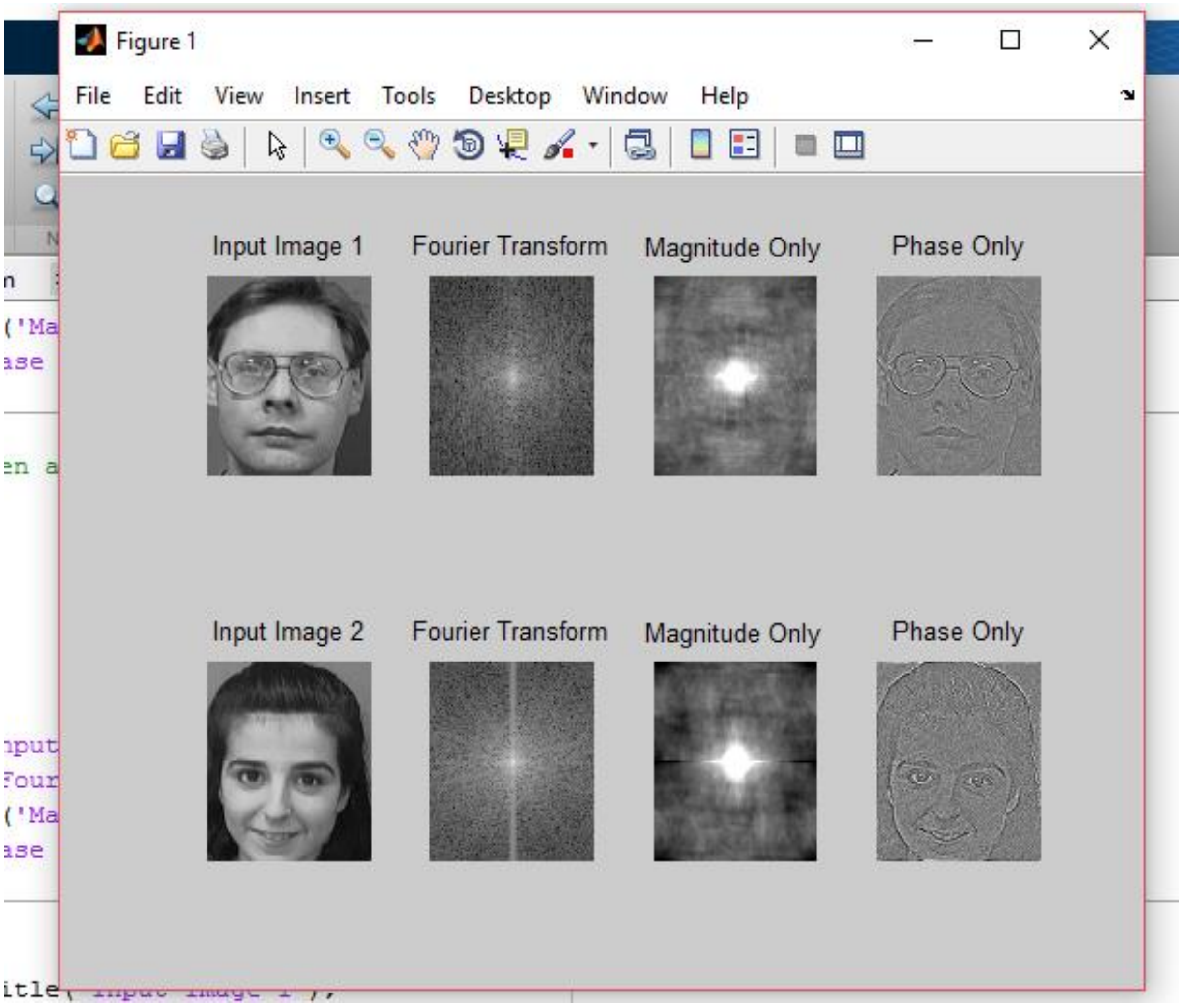
Note: the values for projection error may approximately change when the code is run again.

All other values are approximate values and may differ under different circumstances.



Input Image of Question 2 with 18 input points selected to find out the camera calibration matrix. These includes the points from the 6 point and 12point version.

Ans-3: These are the following outputs shown by the code named '*assignment2_3.m*'.



OBSERVATIONS: -

- The input image reconstructed from the **magnitude** spectrum only. It can be inferred that that the intensity values of **low** frequency pixels are comparatively more than **high** frequency pixels.
- The input image reconstructed from **phase** spectrum only. It is clearly observed that the intensity values of **high** frequency (edges & lines) pixels are comparatively more than **low** frequency pixels.
- In other words, the magnitude spectrum reflects strength of the harmonics in an image and the phase spectrum tells as to where the harmonic lies in space.

Facts:

- The phase determines the shift in the sinusoid components of the image. With zero phase, all the sinusoids are centered at the same location and we get a symmetric image whose structure has no real correlation with the original image at all. This makes it centered at the same location which defines that the sinusoids are a maximum for that given location because of which there is a big **white patch** in the middle of the magnitude-only reconstruction.

- The phase-only reconstruction preserves feature because of the principle of **phase congruency**. At the locations of edges and lines, most of the sinusoid components have the **same phase**. This property is efficiently used to detect lines and edges. **Hence this proves that phase information is relatively more important with respect to magnitude.**
- Changing the magnitude of the various component sinusoids changes the shape of the feature. When you do a phase-only reconstruction, you set all the magnitudes to one, which changes the shape of the features, but not their location. In many images the low frequency components have a magnitude higher than the high frequency components, so phase-only reconstruction does look like a high-pass filter. Hence, we can conclude that phase consists of information associated with locations of features.
- We **cannot** just *add* the phase-only and magnitude-only images to get the original. In order to do so we have to multiply them in the Fourier domain and **inverse Fourier transform** them back to get the original.
- Recovering image with just phase and we can get **something recognizable**. But recovering the same with solely the magnitude produces **unrecognizable** images.
- Re-construction using just the magnitude (and an assumed phase of zero) will work for symmetric images. Phase is needed to make the top looks different from the bottom and left look is disparate from the right side of the image, etc. The reason is all the cosine waveform DFT basis vectors are exactly symmetric around the center of an FFT window. Thus, when not shifted, we can only reconstruct a symmetric result.

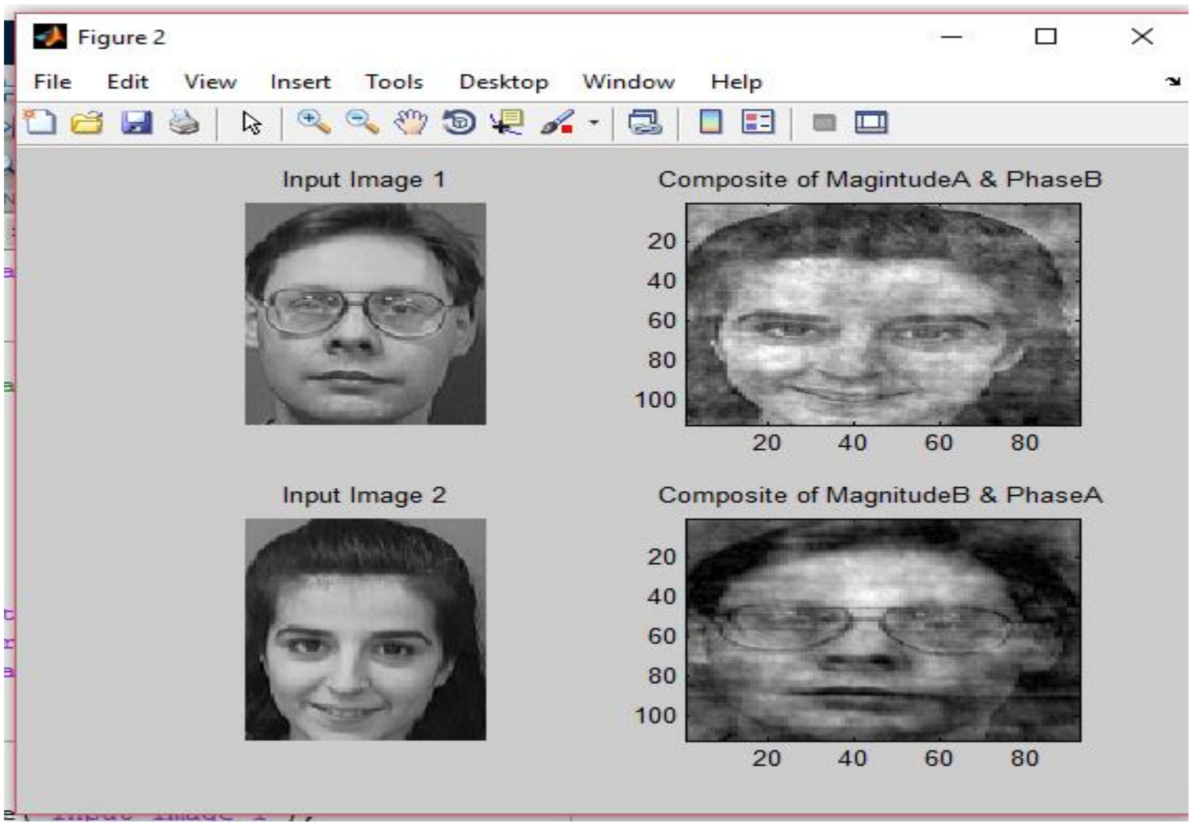


Figure 2
File Edit View Insert Tools Desktop Window Help
Input Image 1
Composite of MagintudeA & PhaseB
Input Image 2
Composite of MagnitudeB & PhaseA
20 40 60 80 100
20 40 60 80

- Since we know based on the observations made earlier that: **Phase carries most of the information in an image and is dominant over the magnitude information.** This is integral to the deductions we establish.
- As a result, **It’s understood why the composition of images are similar to images whose respective phases have been included during composition calculation because of which the top right image looks like input image 2 and the bottom right image looks like input image 1.**