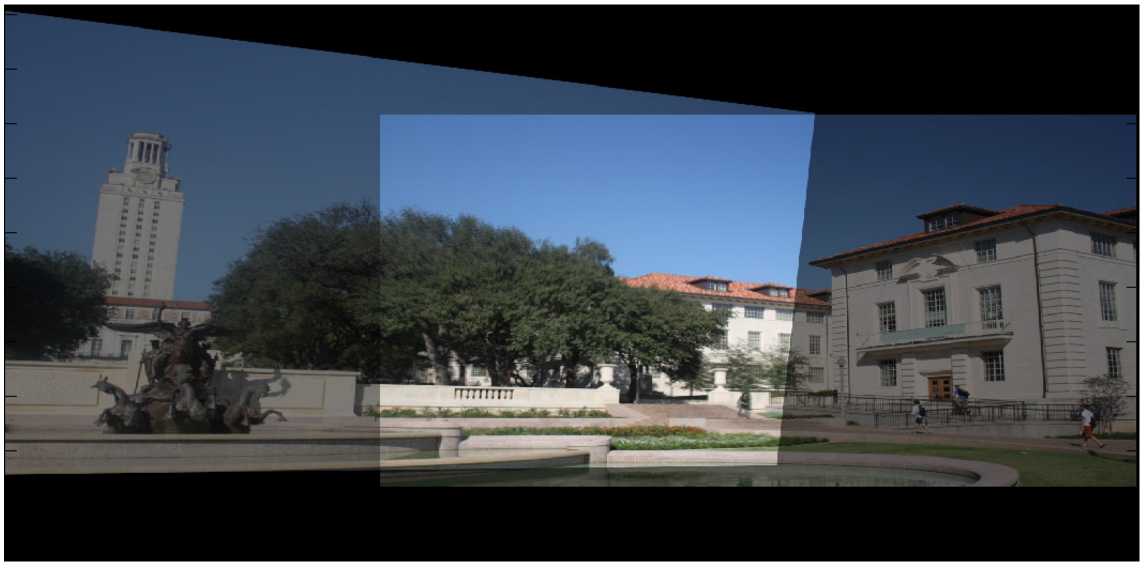
Lu Xia

ID: Lxia1

I. Programming problem: image mosaics [100 points]

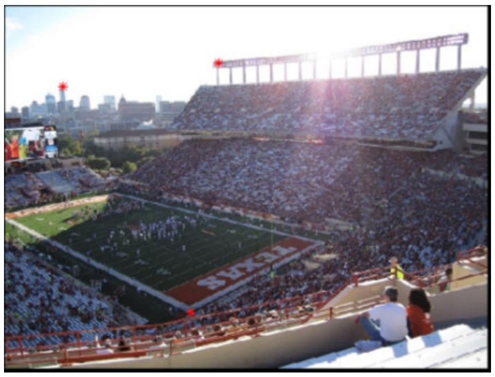
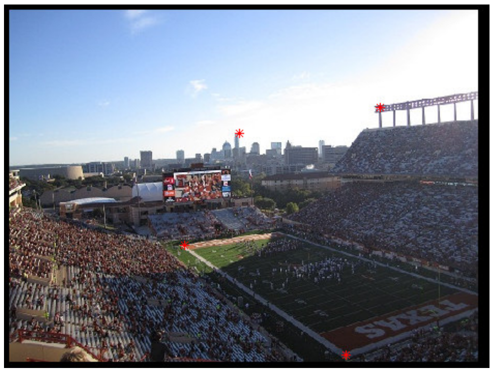
Apply your system to the provided pair of images, the output mosaic is as follow:



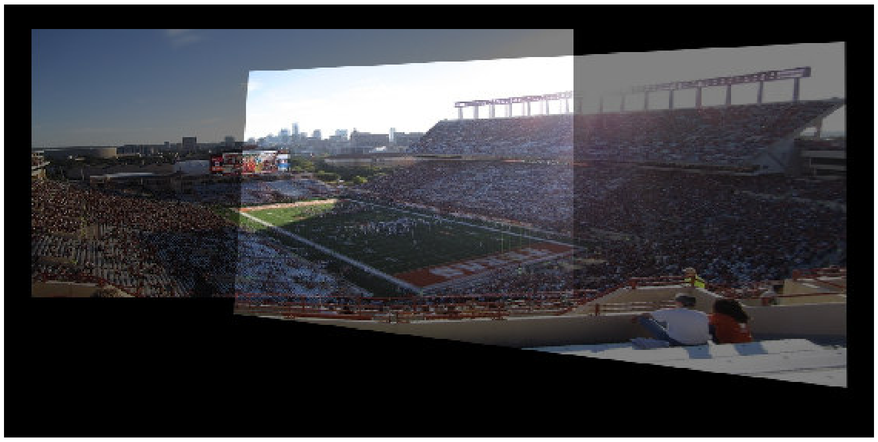
2. Additional example:

Warp image2 to image1:

Corresponding points

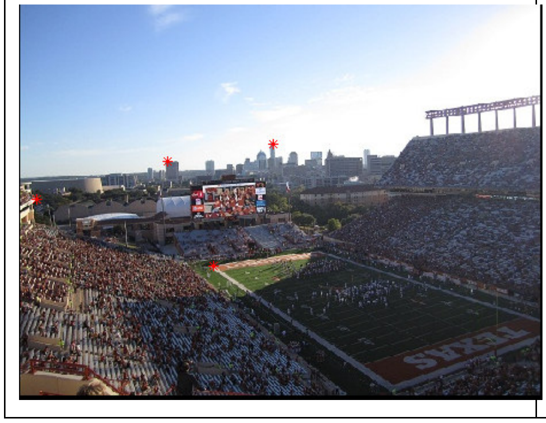
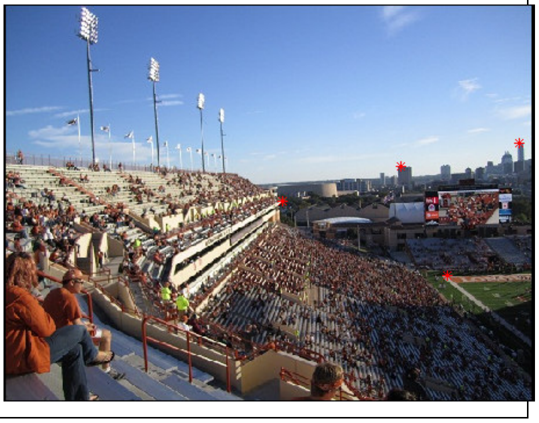
Image1	Image2
	

Mosaic (image1&image2)

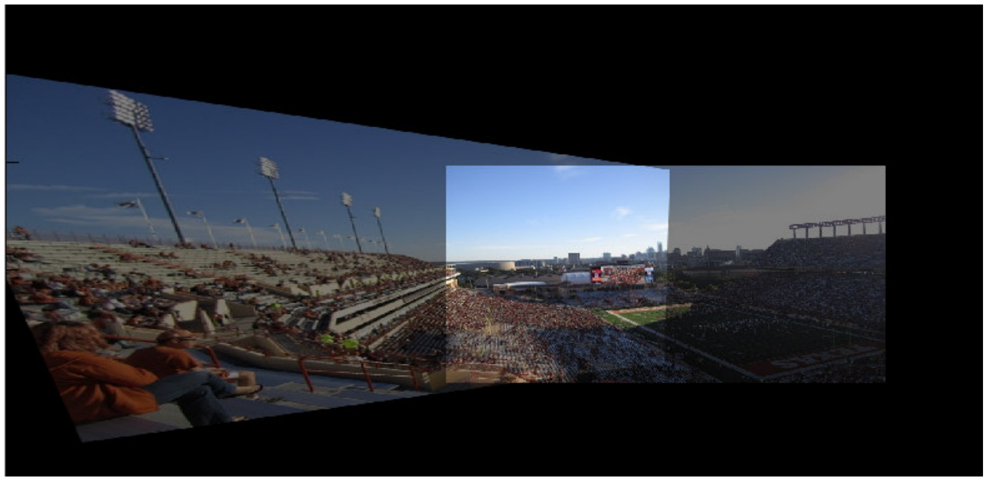


Warp image3 to image2

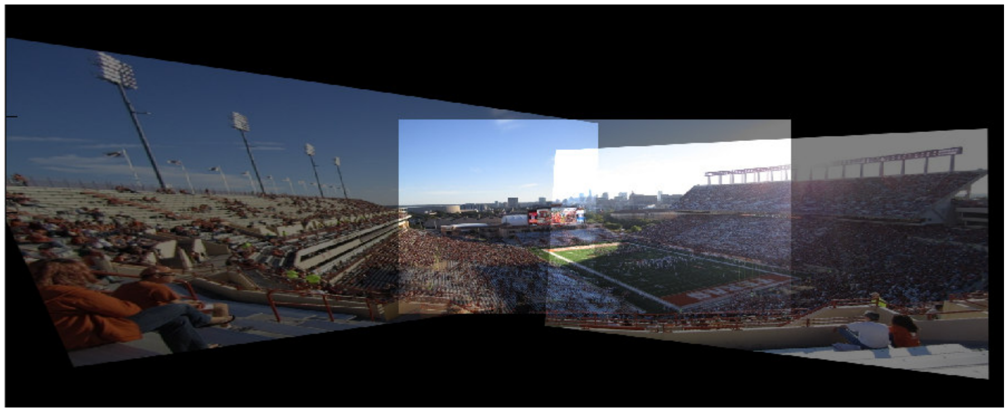
Corresponding points

Image2	Image3
	

Mosaic (image2&image3)

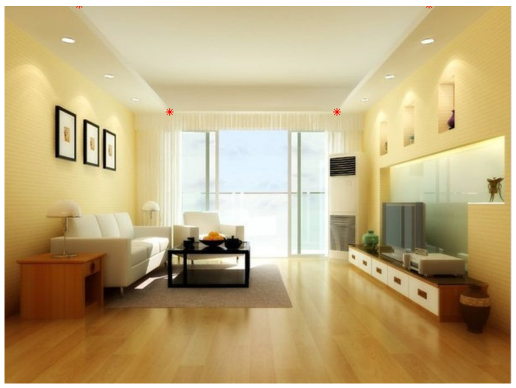


Mosaic(image1&2&3)



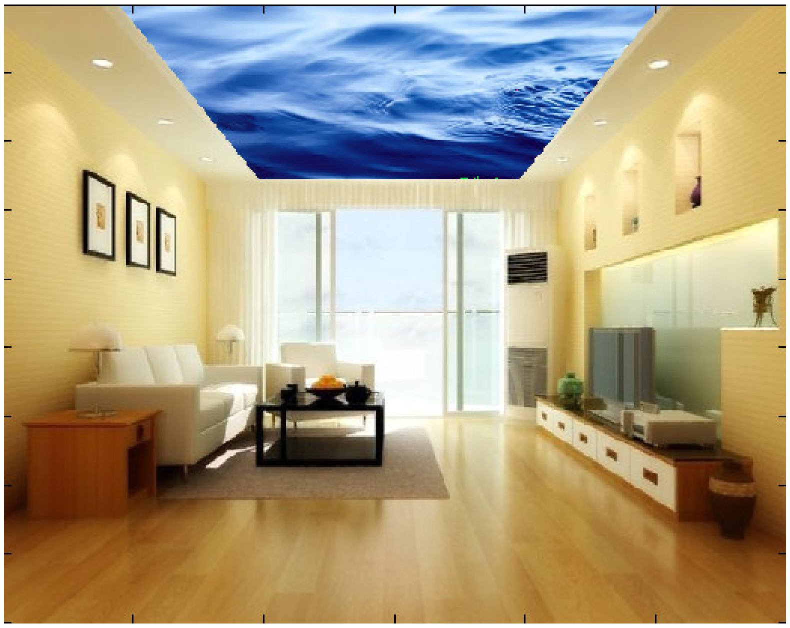
3. Warp one image into a “frame” region in the second image.

Example 1

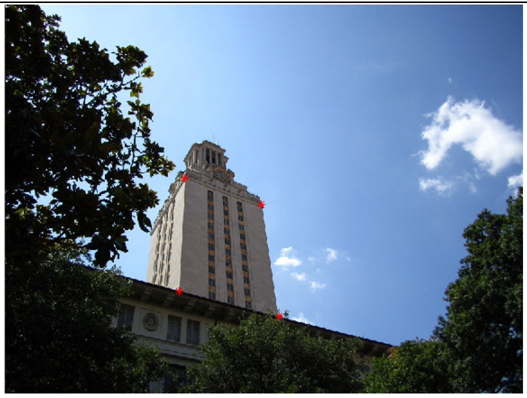
Image1	Image2
	

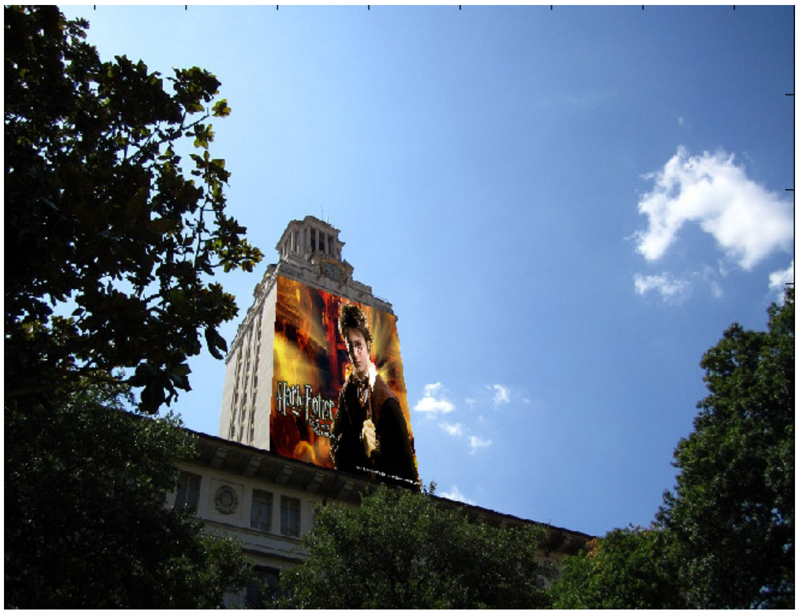
Result:

A room underwater.



Example 2

Image1	Image2
	

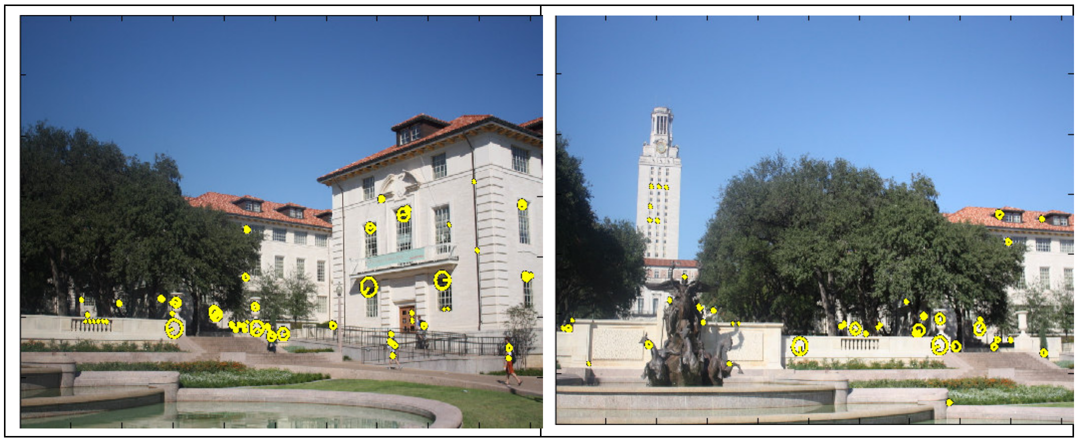


II. [OPTIONAL] Extra credit [up to 10 pts each, max 30 pts]

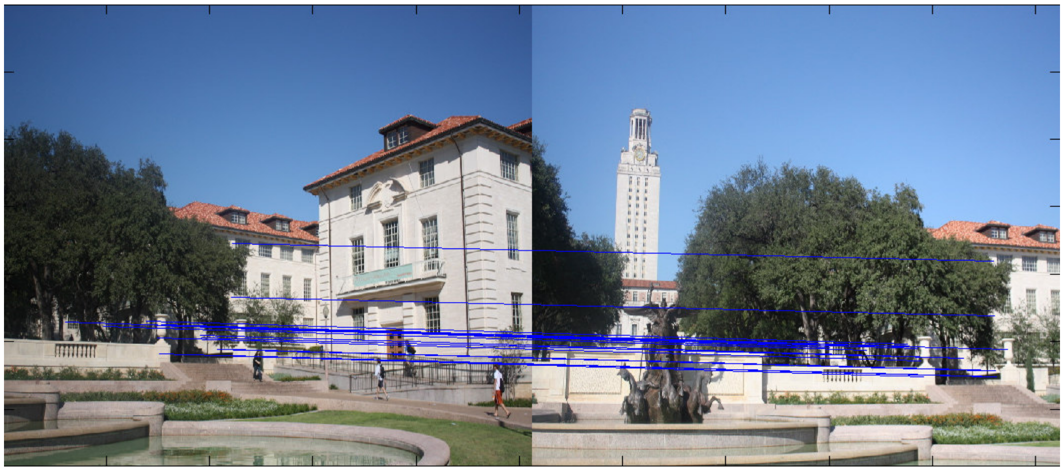
1. Replace the manual correspondence stage with automatic interest point detection and local feature matching.

Interest points:

uttower1	Uttower2


Note: 50 interesting points are randomly selected and showed.

Corresponding points by feature matching:


We can see that the corresponding points created by the algorithm are quite accurate.

2. Implement RANSAC for robustly estimating the homography matrix from noisy correspondences. Show with an example where it successfully gives good results even when there are outlier (bad) correspondences given as input. Compare the robust output to the original (non-RANSAC) implementation where all correspondences are used.

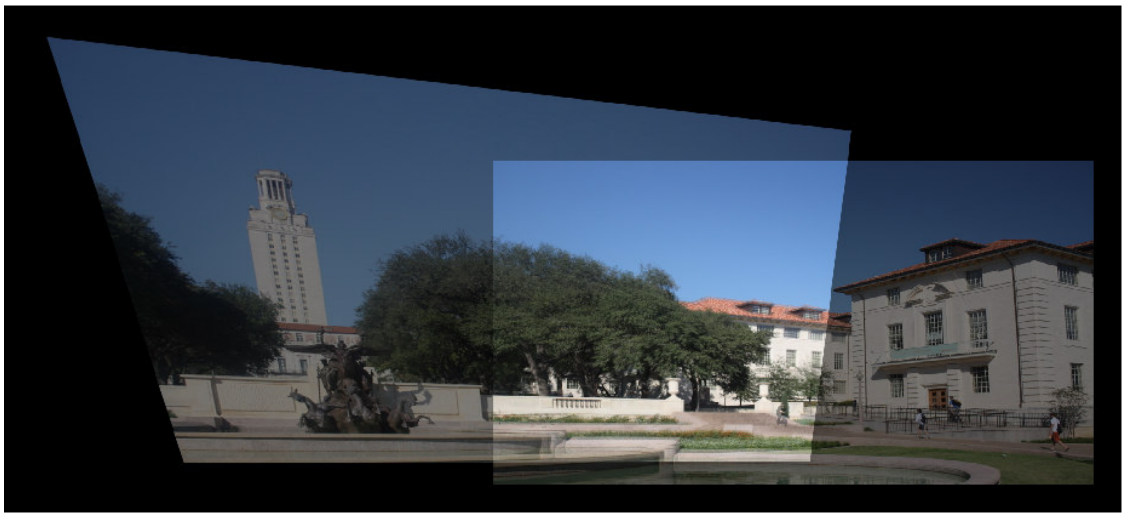
Note: The result from the SIFT Feature matching is too good that there are no outliners in the corresponding points created by the algorithm. In order to test the function of RANSAC, I intentionally add 80% outliers to the corresponding points. So that the inlier rate is 20% (very low).

The RANSAC algorithm converges as follow:

```
Minimal sample set dimension = 4
Squared noise threshold = 73.563766, (assuming Gaussian noise, for sigma = 1.000000)
Iteration =        1/ 95511315. Inliers =        4/     115 (rank is r = -2732.03151473)
Iteration =        3/ 95511315. Inliers =        4/     115 (rank is r = -2260.59250120)
Iteration =        5/ 95511315. Inliers =        4/     115 (rank is r = -2183.69200254)
Iteration =        8/ 95511315. Inliers =        4/     115 (rank is r = -2052.10956993)
Iteration =        9/ 95511315. Inliers =        4/     115 (rank is r = -2043.30156386)
Iteration =       11/ 95511315. Inliers =        4/     115 (rank is r = -1725.15093934)
Iteration =       20/  2728888. Inliers =        7/     115 (rank is r = -1548.02645323)
Iteration =      266/    69965. Inliers =       15/     115 (rank is r = -1540.21098857)
Iteration =      389/    24635. Inliers =       19/     115 (rank is r = -1507.86933672)
Iteration =      561/    10780. Inliers =       23/     115 (rank is r = -1470.07823104)
Iteration =     5336/    10780. Inliers =       23/     115 (rank is r = -1469.19593106)
Iteration =    10403/    10780. Inliers =       23/     115 (rank is r = -1464.77379393)
Iteration =    10674/    10780. Inliers =       23/     115 (rank is r = -1462.48859330)
Restimating the parameter vector... Done
Final number of inliers = 23/115
Converged in 10781 iterations (64.771879 seconds)
```

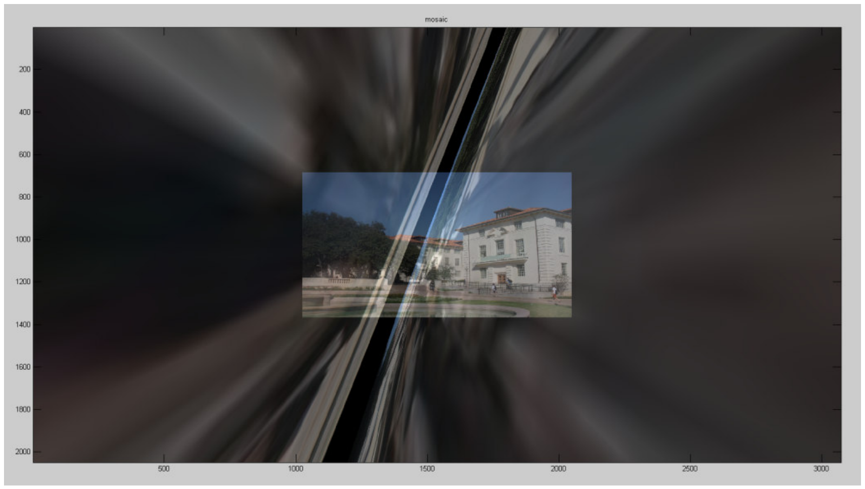
We can see that the algorithm correctly find the 23 inliers.

Mosaic generated from the inliers:

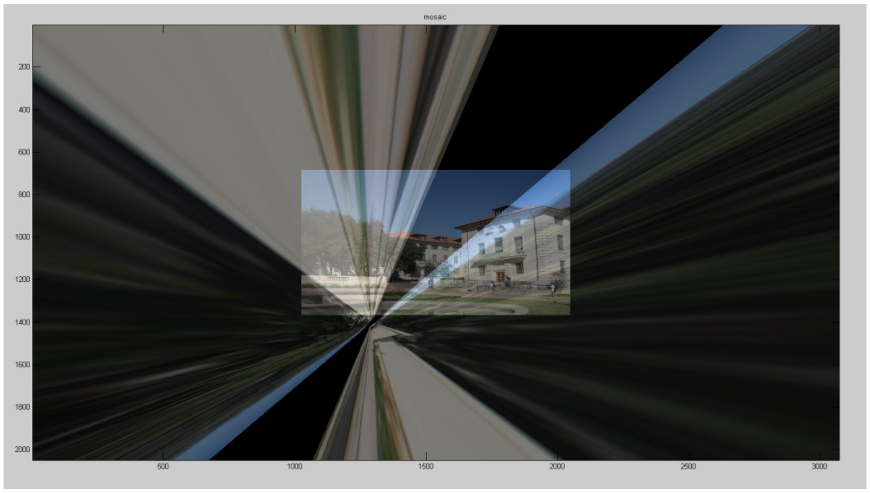


If all the correspondence are used:

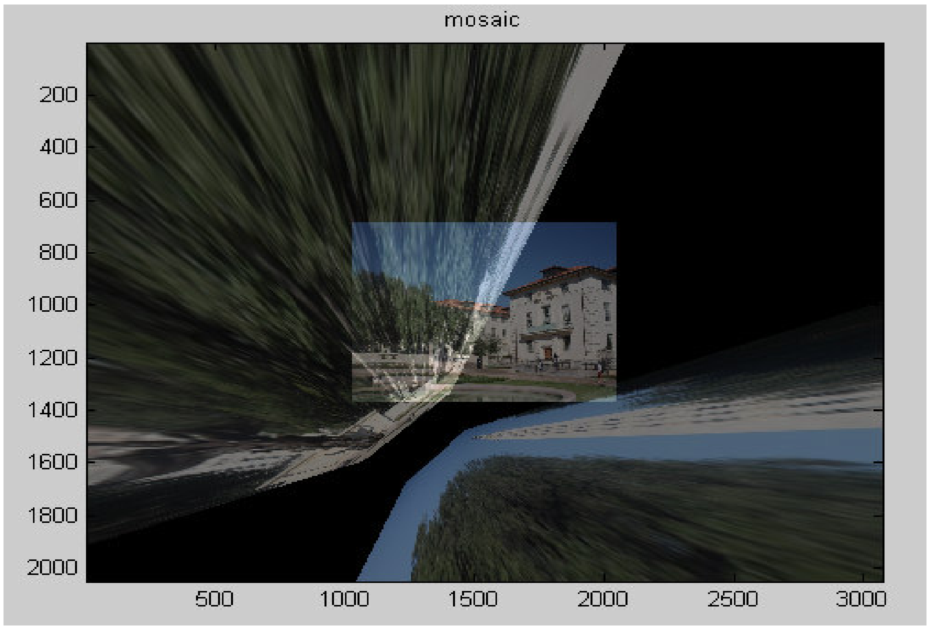
inlier rate P=0.25


mosaic
200
400
600
800
1000
1200
1400
1600
1800
2000
500
1000
1500
2000
2500
3000

P=0.75

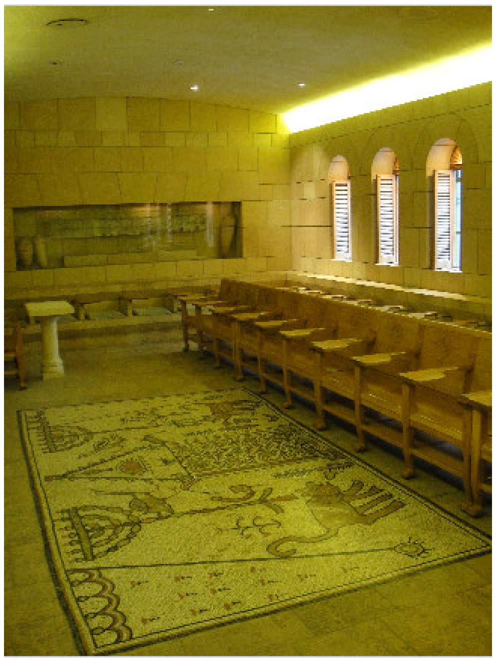
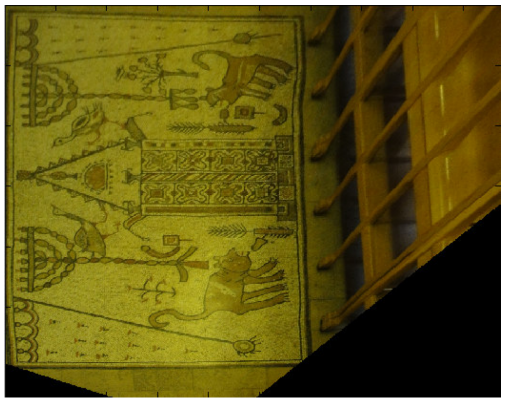

mosaic
200
400
600
800
1000
1200
1400
1600
1800
2000
500
1000
1500
2000
2500
3000

P=0.9

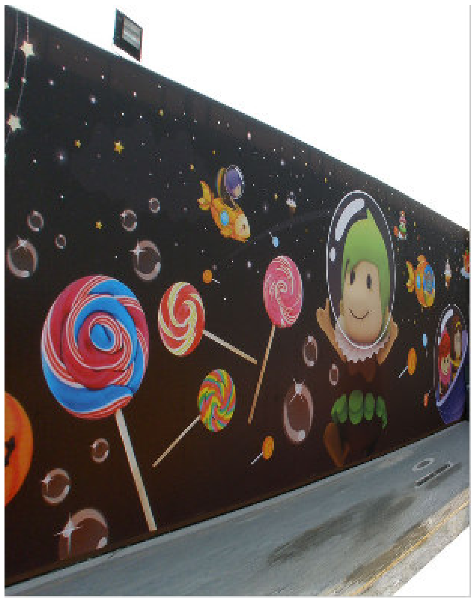


See the result is severely damaged by the outliers. The noise make the result totally meaningless.

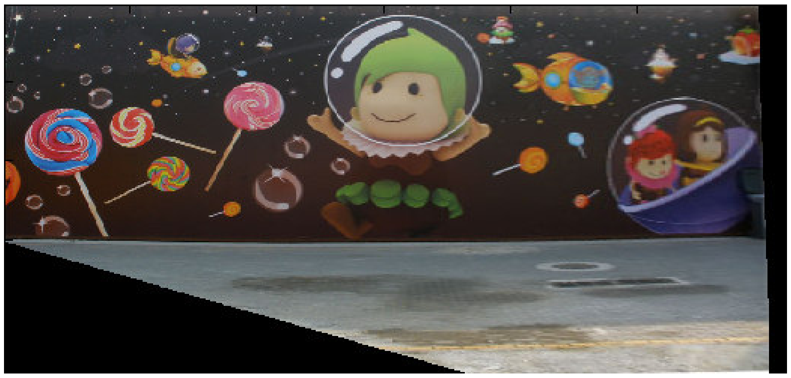
3. Rectify an image with some known planar surface (say, a square floor tile, or the rectangular face of a building façade) and show the virtual fronto-parallel view.

Original image	Parallel view.
	

Original image



Parallel view:



Codes:

```
%% image mosaics
% warp one image into another
clear all
clc
close all
%% read images
im1=imread('uttower1.JPG');%
im2=imread('uttower2.JPG');%uttower2.JPG
%% enlarge the image
[N,M,D]=size(im1);
largeIm1=uint8(zeros(3*N,3*M,D));
largeIm1(N+1:2*N,M+1:2*M,:)=im1;
[N,M,D]=size(im2);
largeIm2=uint8(zeros(3*N,3*M,D));
largeIm2(N+1:2*N,M+1:2*M,:)=im2;
% % manually select the corresponding points
num=4;
figure(1), imshow(largeIm1);hold on
% [X1,Y1] = ginput(num);
X1=[1126;1337;1502;1496]; %tower points
Y1=[1186;1299;990;1152]; %tower points
figure(2), imshow(largeIm2);hold on
% [X2,Y2] = ginput(num);
X2=[1589;1800;1951;1958]; %tower points
Y2=[1224;1334;1012;1180]; %tower points
% calcualte the transform matrix H
H=trasformMatrix(X1, X2, Y1, Y2);
%display the corresponding points
for k=1:length(X1)
   P=H*[X1(k);Y1(k);1];
```