ImageMosaicing

February 17, 2019

1 Assignment 2

2 Image Mosaicing

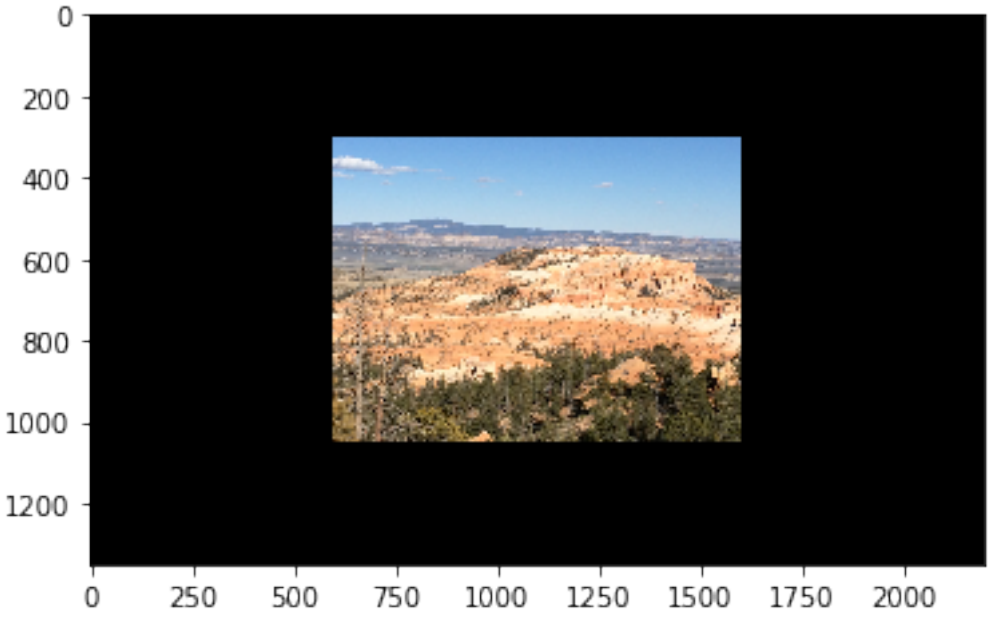
Parth Partani 20161034

```
In [1]: import cv2
        import sys
        import os.path
        import numpy as np
        import matplotlib.pyplot as plt
        import random
        from scipy import ndimage
        import scipy
```

2.1 Image Set 1

Inputing images and converting it into RGB format. Now adding border to the first image so that after stitching we get a complete image and not a cropped image.

```
In [216]: image1 = cv2.imread("test_images/img1_1.png")
          image1 = cv2.cvtColor(image1, cv2.COLOR_BGR2RGB)
          image2 = cv2.imread("test_images/img1_2.png")
          image2 = cv2.cvtColor(image2, cv2.COLOR_BGR2RGB)
          image1 = cv2.copyMakeBorder(image1,300,300,600,600, cv2.BORDER_CONSTANT)
          plt.imshow(image1)
          plt.show()
```



3 Part-1 : Finding Feature points and Descriptors

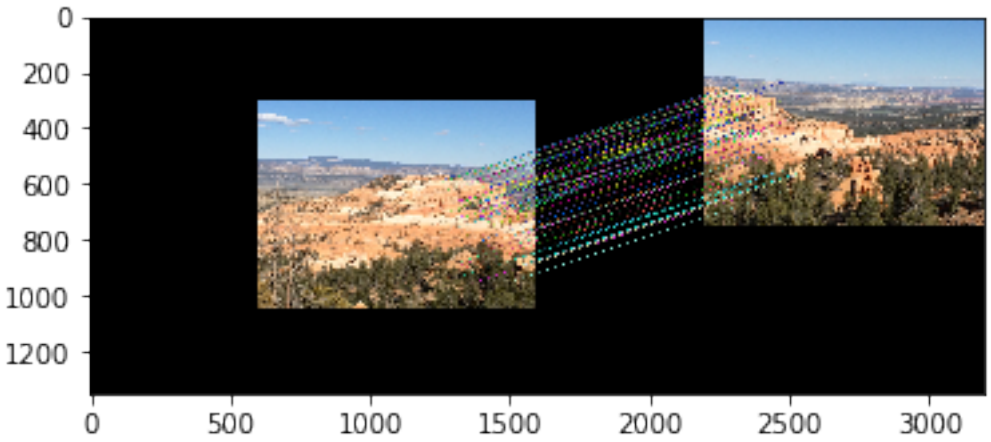
1. Firstly, we have to find out the features matching in both the images. These best matched features act as the basis for stitching. We extract the key points and sift descriptors for both the images.
2. kp1 and kp2 are keypoints, des1 and des2 are the descriptors of the respective images.
3. Now, the obtained descriptors in one image are to be recognized in the image too.
4. The BFMatcher() matches the features which are more similar. When we set parameter k=2, we are asking the knnMatcher to give out 2 best matches for each descriptor.
5. Often in images, there are tremendous chances where the features may be existing in many places of the image. So we filter out through all the matches to obtain the best ones.

```
In [99]: def Mapping(im1, im2, thresh):
             bf = cv2.BFMatcher()
             kp1, des1 = sift.detectAndCompute(im1,None)
             kp2, des2 = sift.detectAndCompute(im2,None)
             matches = bf.knnMatch(des1,des2, k=2)
             good = []
             List = []
             im1_points = []
             im2_points = []
             for m in matches:
                 if m[0].distance < thresh*m[1].distance:
                     good.append([m[0]])
                     (x1, y1) = kp1[m[0].queryIdx].pt
                     (x2, y2) = kp2[m[0].trainIdx].pt
```

```
                im1_points.append([x1,y1,1])
                im2_points.append([x2,y2,1])
                List.append([x1, y1, x2, y2])
        p1 = np.array(im1_points)
        p2 = np.array(im2_points)
        mapimg = cv2.drawMatchesKnn(im1,kp1,im2,kp2,good,None,flags=2)
        plt.imshow(mapimg)
        plt.show()
        return p1,p2,good
```

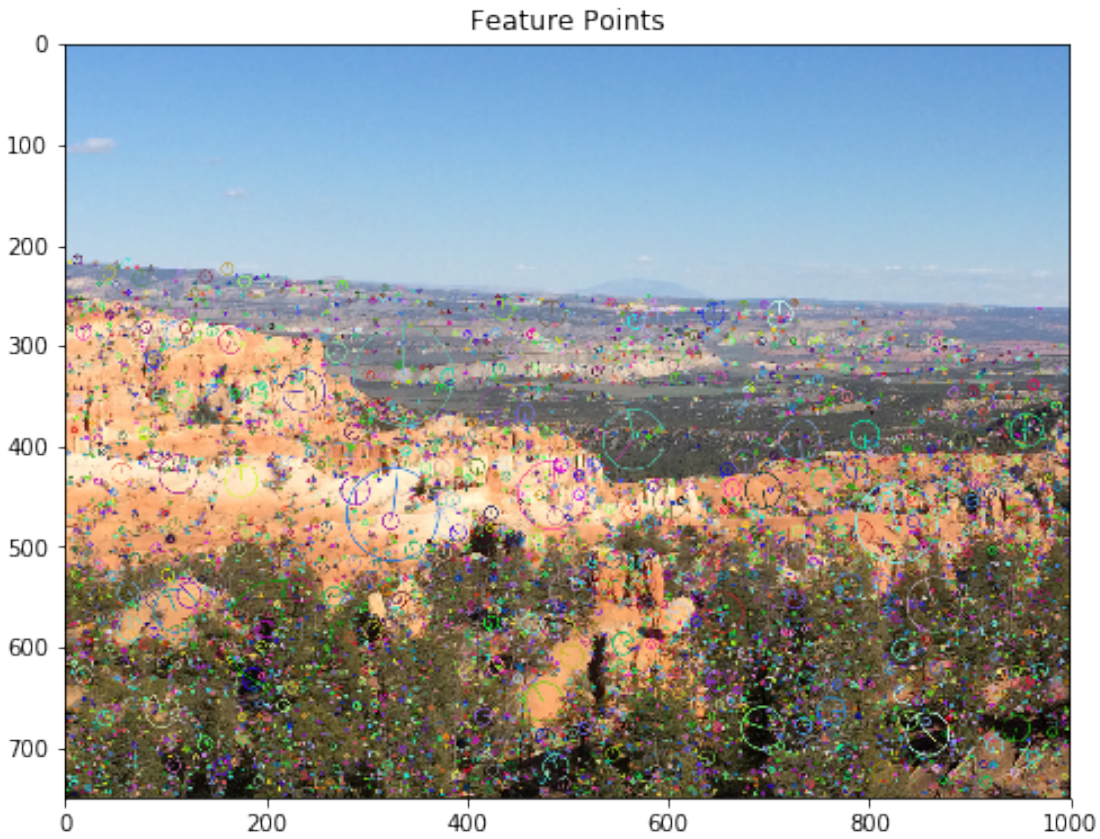
```
In [206]: def showFeaturePoints(im):
              kp, des = sift.detectAndCompute(im,None)
              res = cv2.drawKeypoints(im,kp,None,flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINT
              plt.figure(figsize=[8,8])
              plt.title("Feature Points")
              plt.imshow(res)
              plt.show()
```

```
In [217]: set1Mat = Mapping(image1, image2, 0.15)
```



3.1 Feature Points

```
In [218]: showFeaturePoints(image2)
```



4 Part -2 : Homographt Matrix

Finding Homography matrix using DLT method(also can be done using RANSAC).

```
In [6]: def homography(p1,p2):
            A =  np.zeros((2*p1.shape[0],9))

            for i in range(p1.shape[0]):
                A[2*i,:3] = -p2[i,2]*p1[i,:]
                A[2*i,6:9] = p2[i,0]*p1[i,:]

                A[2*i+1,3:6] = -p2[i,2]*p1[i,:]
                A[2*i+1,6:9] = p2[i,1]*p1[i,:]

            U, D, V = np.linalg.svd(A)
            projMat = np.reshape(V[8,:],(3,3))
            projMat = projMat/projMat[2,2]
            return projMat

In [219]: homo1 = homography(set1Mat[0],set1Mat[1])
```

```
print("Homography_matrix is")
print(homo1)
```

```
Homography_matrix is
[[  1.35017210e+00   2.32781847e-02  -1.68112302e+03]
 [  6.13206734e-02   1.31035246e+00  -4.98063300e+02]
 [  2.18636118e-04   3.17576004e-06   1.00000000e+00]]
```

It's time to align the images now. As you know that a homography matrix is needed to perform the transformation, and the homography matrix requires at least 4 matches.

```
In [101]: def cropImg(image):
              if len(image.shape) == 3:
                  flatImage = np.max(image, 2)
              else:
                  flatImage = image

              rows = np.where(np.max(flatImage, 0) > 0)[0]
              if rows.size:
                  cols = np.where(np.max(flatImage, 1) > 0)[0]
                  image = image[cols[0]: cols[-1] + 1, rows[0]: rows[-1] + 1]
              else:
                  image = image[0,0]
              return image
```

5 Part - 3 : Transforming one of the images to the others reference frame

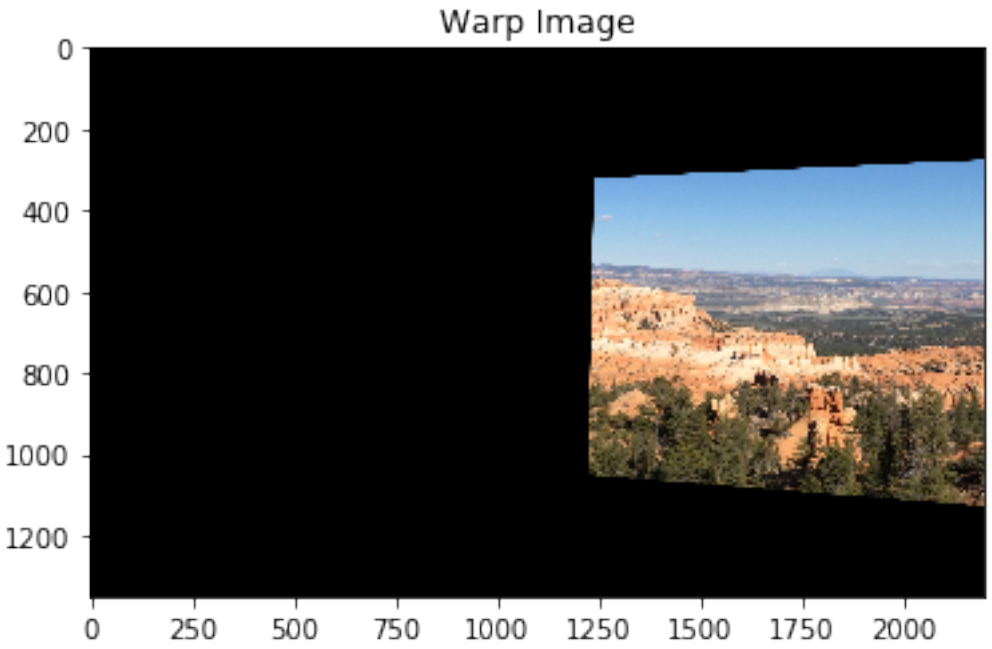
Now that we found the homography for transformation, we can now proceed to warp using function : cv2.warpPerspective(im2,scipy.linalg.inv(homo1), (im1.shape[1],im1.shape[0]))

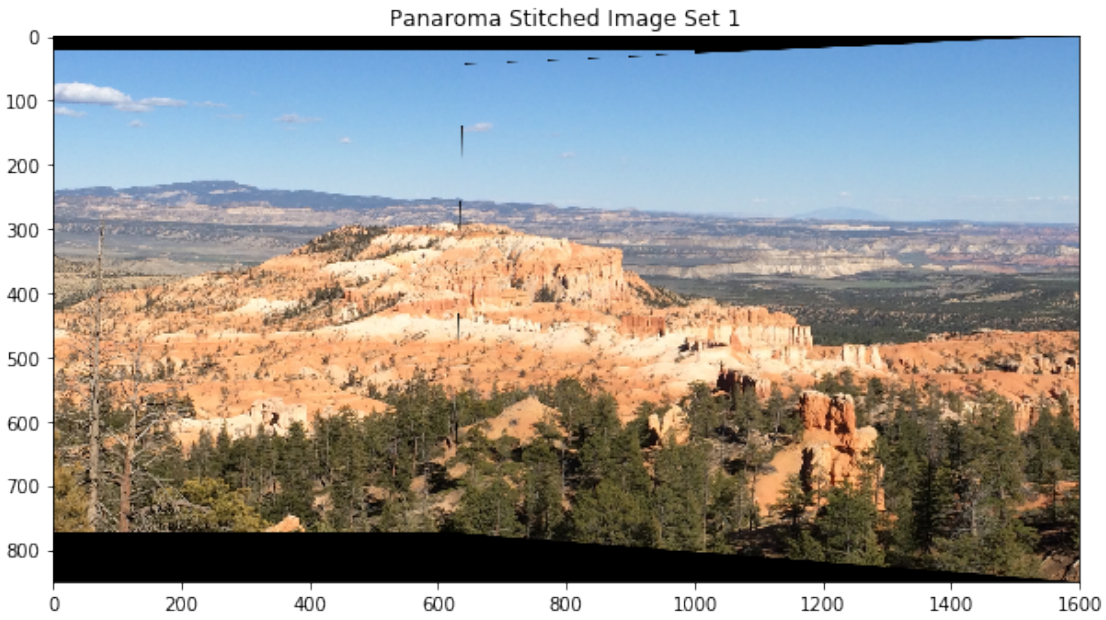
6 Part - 4 : Stitching Images

```
In [184]: def panaroma(im1, im2, homo1):
              out = cv2.warpPerspective(im2,scipy.linalg.inv(homo1), (im1.shape[1],im1.shape[0]))
              plt.title("Warp Image")
              plt.imshow(out)
              plt.show()
              output = out.copy()
              for i in range(im1.shape[0]):
                  for j in range(im1.shape[1]):
                      if(sum(out[i,j]) == 0):
                          output[i,j] = im1[i,j]
              return output
```

```
In [171]: pana1 = panaroma(image1,image2,homo1)
          pana1 = cropImg(pana1)
```

```
plt.figure(figsize=[10,10])
plt.imshow(pana1)
plt.title("Panaroma Stitched Image Set 1")
plt.show()
```





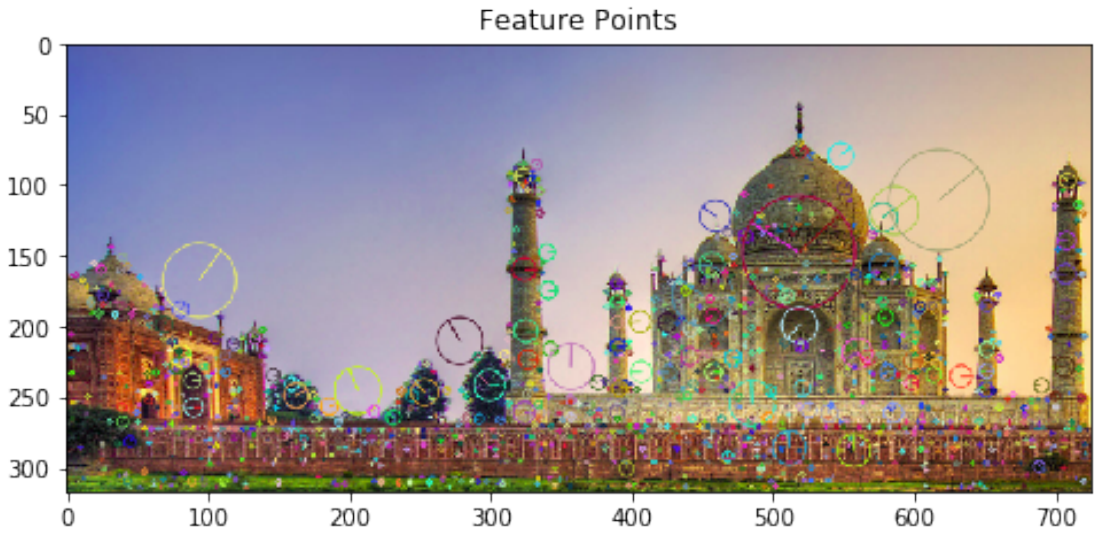
6.1 Image Set 2

As we did stitching for 2 images in previous Image set now we will do it for 6 images

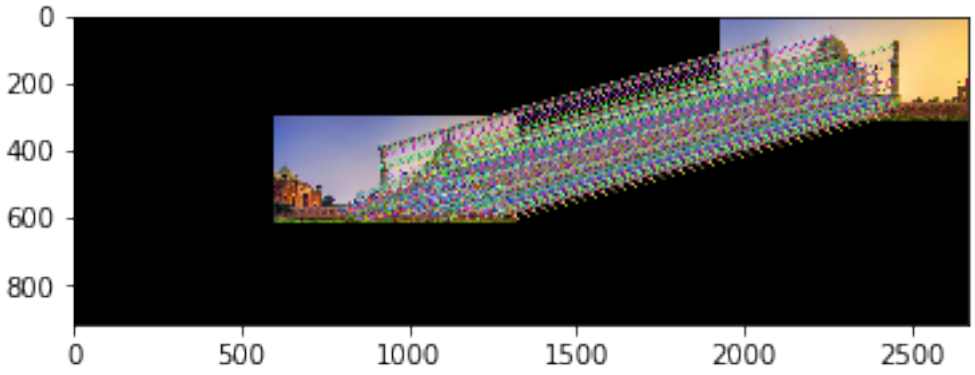
```
In [213]: n_images = 6
          images = []
          for i in range(n_images):
              x = cv2.imread("test_images/img2_" + str(i+1) + ".png")
              images.append(cv2.cvtColor(x, cv2.COLOR_BGR2RGB))
```

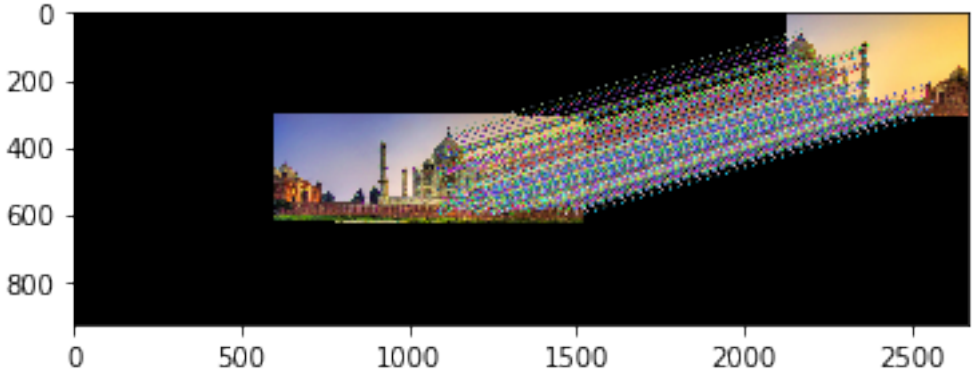
6.1.1 Feature Points

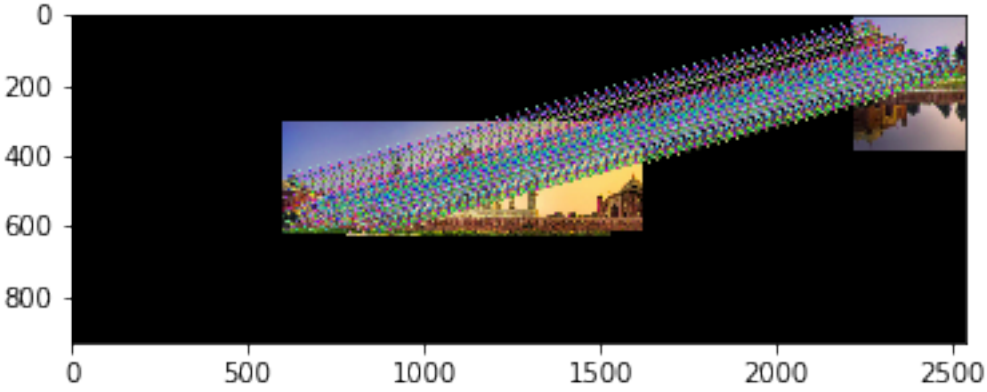
```
In [214]: showFeaturePoints(images[0])
```

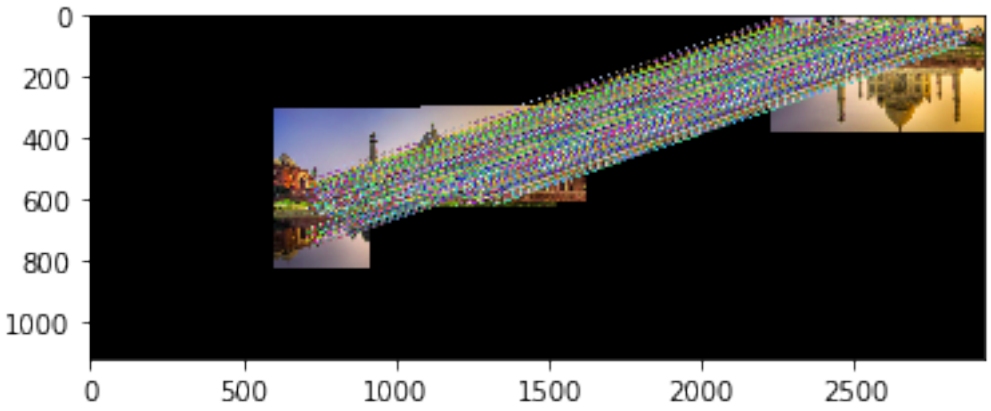


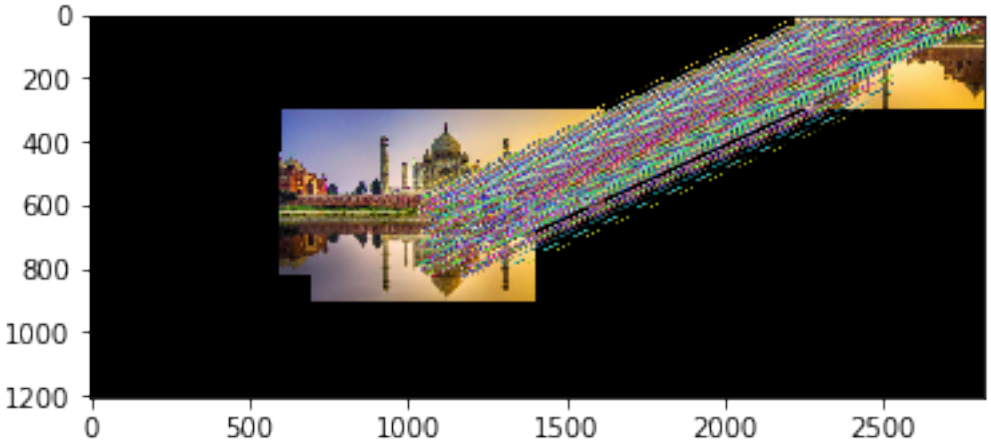

```
In [215]: res = images[0]
          for i in range(n_images-1):
              res = cv2.copyMakeBorder(res,300,300,600,600, cv2.BORDER_CONSTANT)
              set2Mat = Mapping(res, images[i+1],0.2)
              homo2 = homography(set2Mat[0],set2Mat[1])
              if i==4:
                  print("Homography_matrix is")
                  print(homo2)
              pana2 = panaroma(res,images[1+i],homo2)
              res = cropImg(pana2)
          plt.figure(figsize=[12,12])
          plt.imshow(res)
          plt.title("Panaroma Stitched Image for Set 2")
          plt.show()
```


0
200
400
600
800
0
500
1000
1500
2000
2500

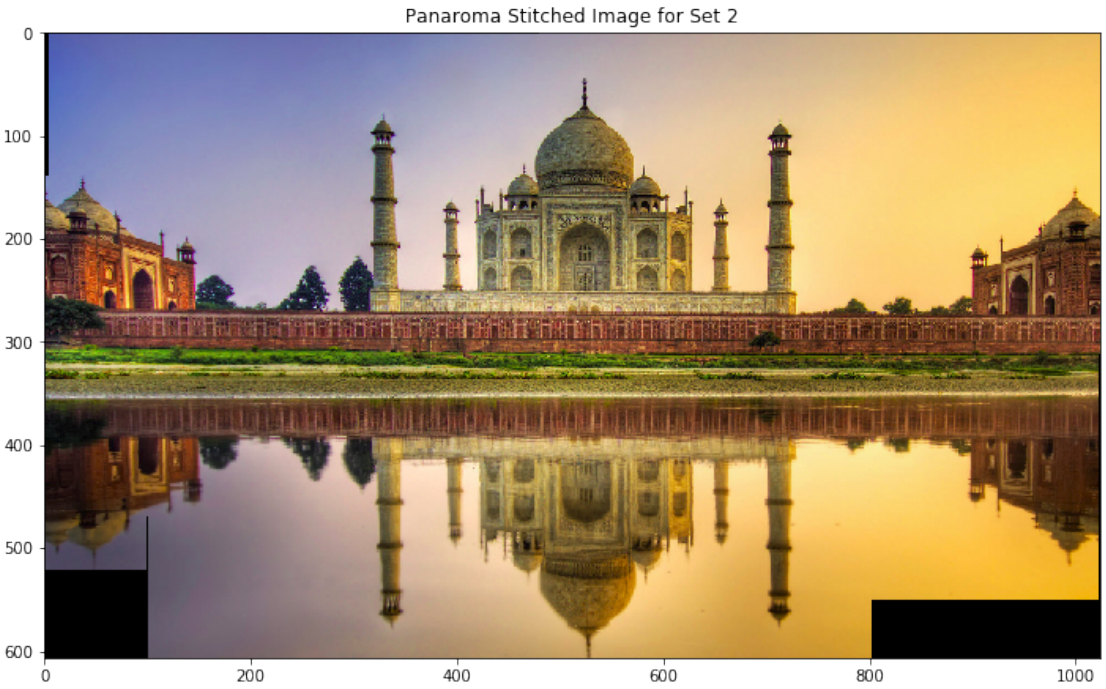

0
200
400
600
800
0
500
1000
1500
2000
2500


0
200
400
600
800
0
500
1000
1500
2000
2500






```
Homography_matrix is
[[  1.00008852e+00   9.40530998e-06  -1.02609805e+03]
 [ -4.99386469e-07   1.00008246e+00  -5.45044123e+02]
 [  3.13457653e-08   6.57396274e-08   1.00000000e+00]]
```

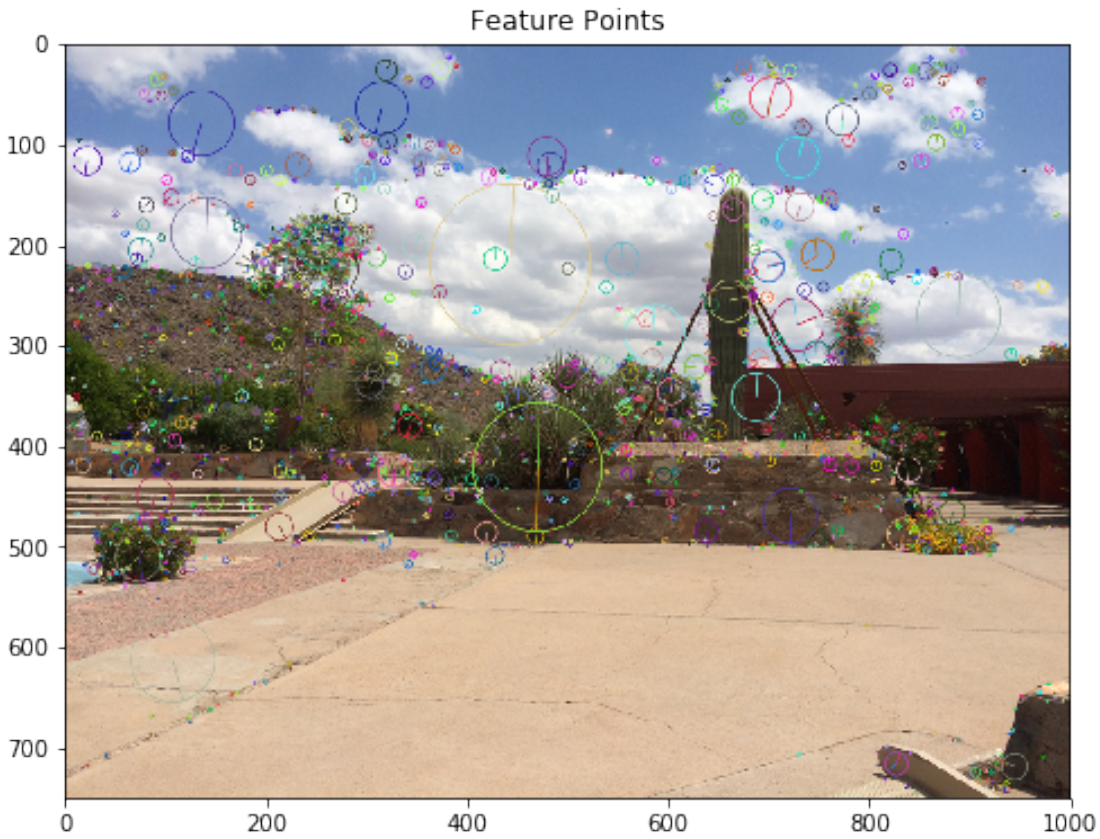


6.2 Image Set 3

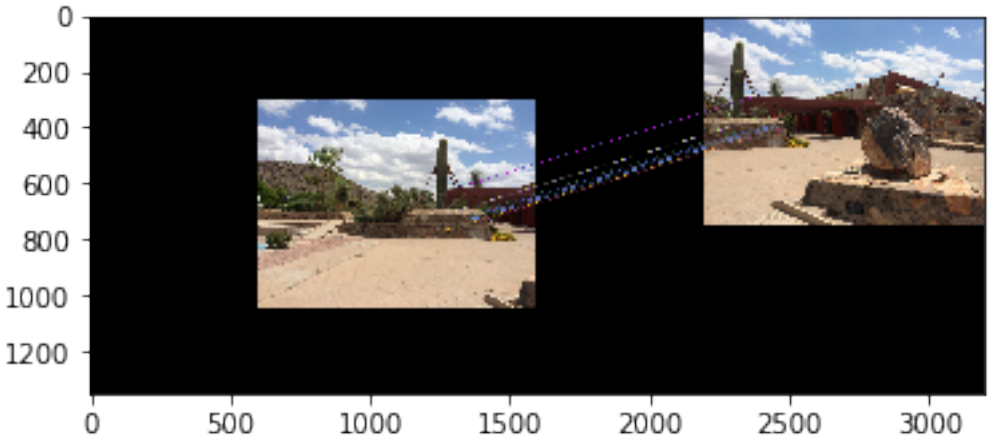
```
In [210]: n_images = 2
          images = []
          for i in range(n_images):
              x = cv2.imread("test_images/img3_" + str(i+1) + ".png")
              images.append(cv2.cvtColor(x, cv2.COLOR_BGR2RGB))
```

6.2.1 Feature Points

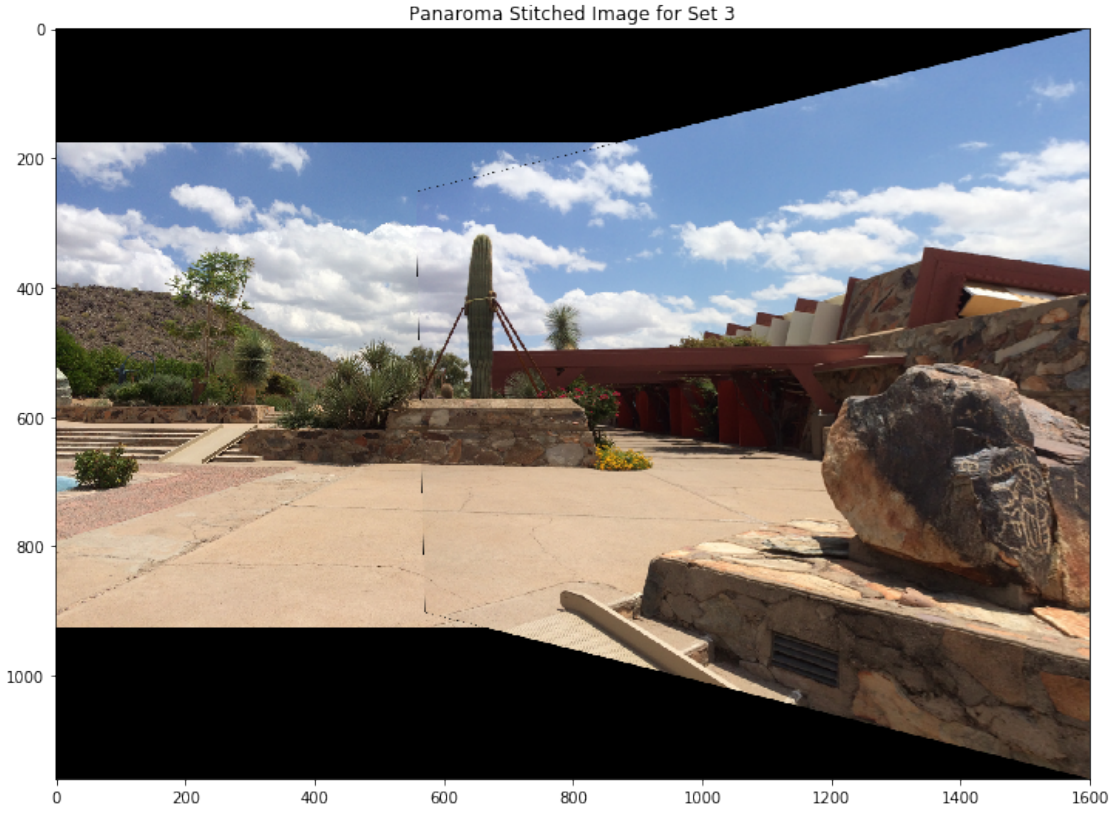
```
In [211]: showFeaturePoints(images[0])
```




```
In [212]: res = images[0]
          for i in range(n_images-1):
              res = cv2.copyMakeBorder(res,300,300,600,600, cv2.BORDER_CONSTANT)
              set2Mat = Mapping(res, images[i+1],0.2)
              homo2 = homography(set2Mat[0],set2Mat[1])
              if i==0:
                  print("Homography_matrix is")
                  print(homo2)
              pana2 = panaroma(res,images[1+i],homo2)
              res = cropImg(pana2)
          plt.figure(figsize=[12,12])
          plt.imshow(res)
          plt.title("Panaroma Stitched Image for Set 3")
          plt.show()
```




```
Homography_matrix is
[[  1.17548686e+01  -2.46315897e-01  -1.35192618e+04]
 [  2.46716921e+00   1.01974833e+01  -6.68752704e+03]
 [  6.67692820e-03   5.84444863e-05   1.00000000e+00]]
```

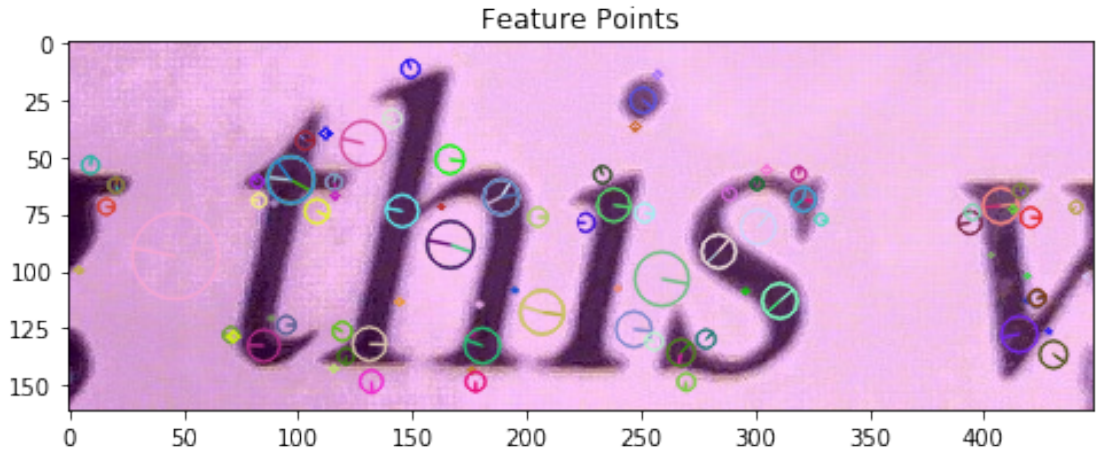


6.3 Image Set 4

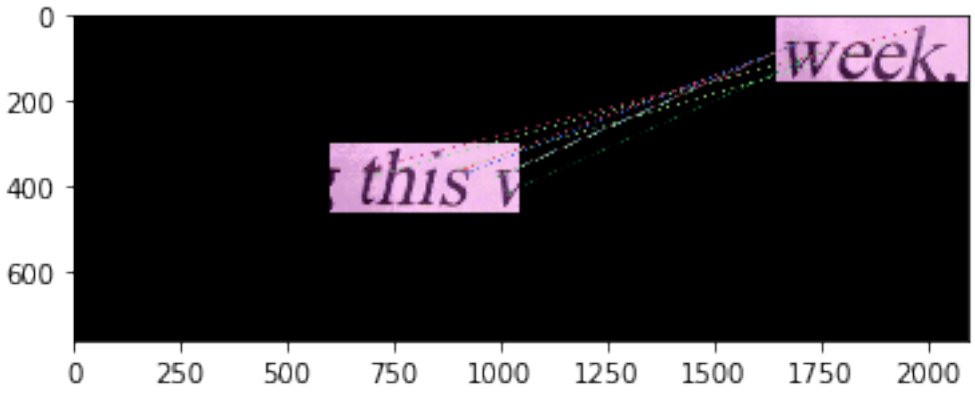
```
In [208]: n_images = 2
          images = []
          for i in range(n_images):
              x = cv2.imread("test_images/img4_" + str(i+1) + ".jpg")
              images.append(cv2.cvtColor(x, cv2.COLOR_BGR2RGB))
```

6.3.1 Feature Points

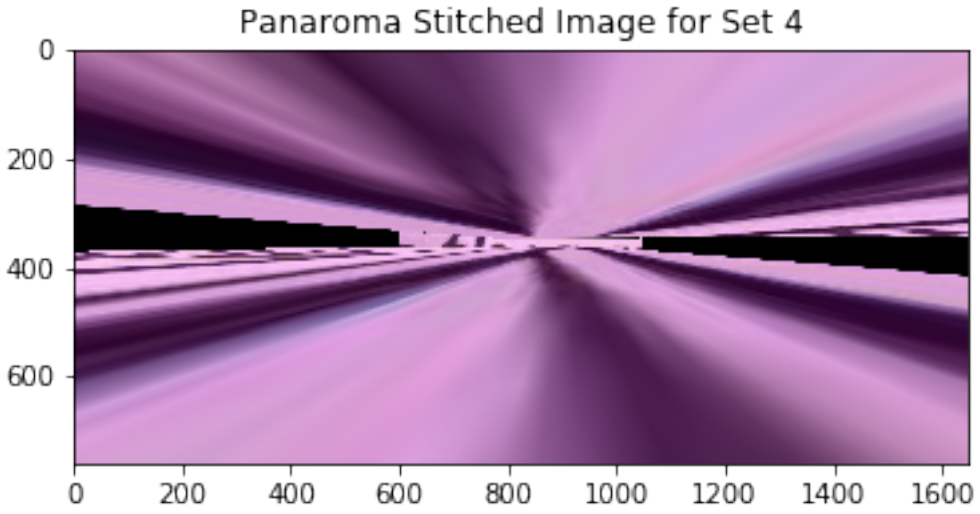
```
In [209]: showFeaturePoints(images[0])
```




```
In [191]: res = images[0]
          for i in range(n_images-1):
              res = cv2.copyMakeBorder(res,300,300,600,600, cv2.BORDER_CONSTANT)
              set2Mat = Mapping(res, images[i+1],0.65)
              homo2 = homography(set2Mat[0],set2Mat[1])
              if i==0:
                  print("Homography_matrix is")
                  print(homo2)
              pana2 = panaroma(res,images[1+i],homo2)
              res = cropImg(pana2)
          plt.imshow(res)
          plt.title("Panaroma Stitched Image for Set 4")
          plt.show()
```




```
Homography_matrix is
[[  1.36781413e-02  -1.71219806e-01   4.86552037e+01]
 [ -8.33400861e-04  -3.25374181e-01   1.17957058e+02]
 [ -6.47357418e-06  -2.78869937e-03   1.00000000e+00]]
```



- Here as we are not able to map considerable amount of feature points, therefore this will result in a bad image.

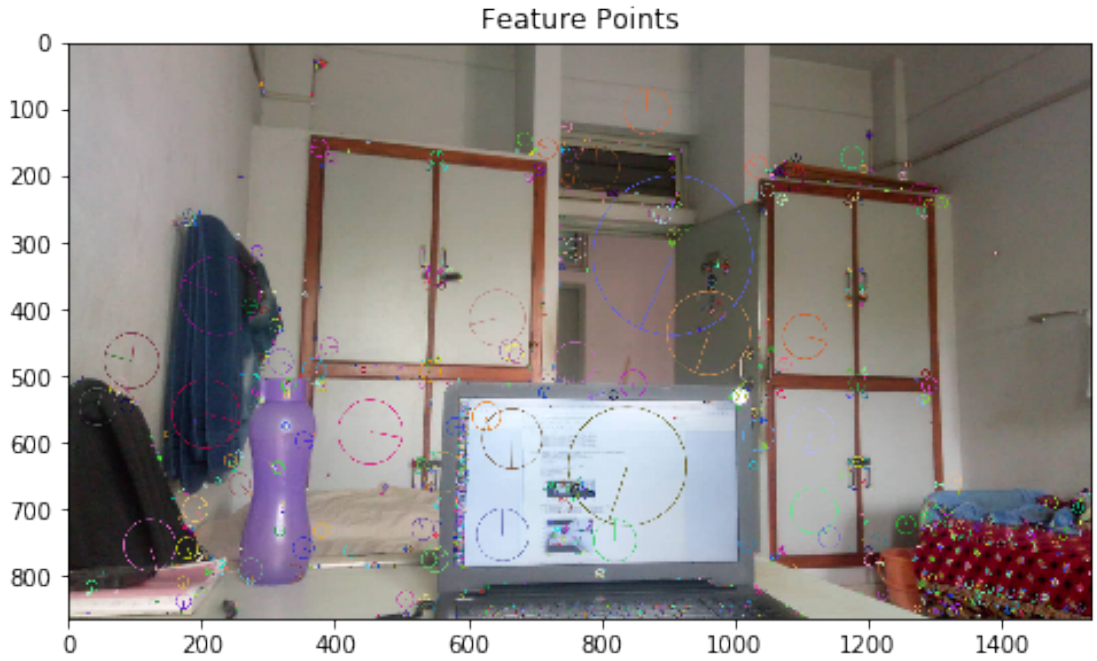
6.4 Images with my own camera

```
In [204]: n_images = 3
          images = []
          for i in range(n_images):
```

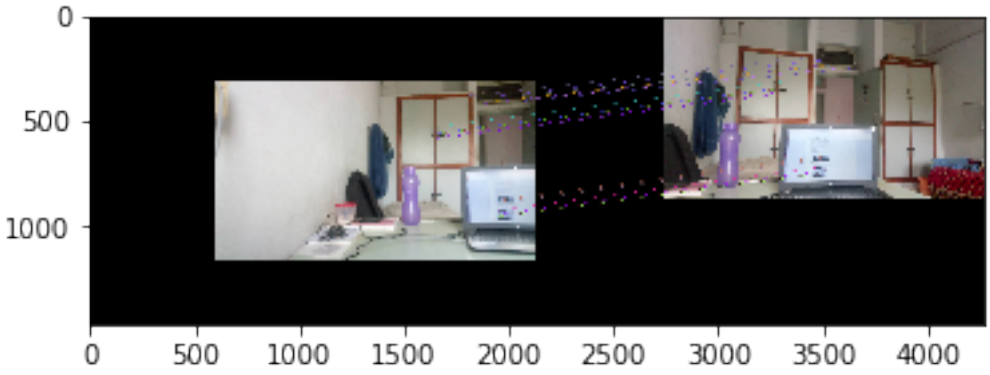
```
        x = cv2.imread("test_images/img5_" + str(i+1) + ".jpg")
        images.append(cv2.cvtColor(x, cv2.COLOR_BGR2RGB))
#       plt.imshow(images[i])
#       plt.show()
```

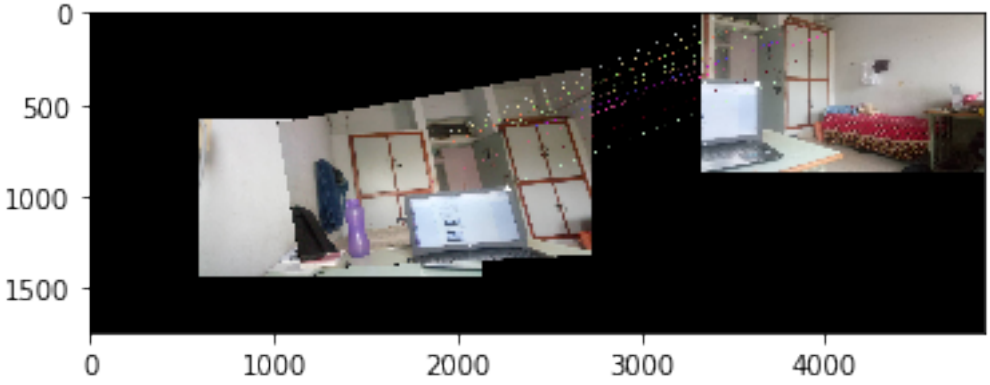
6.4.1 Feature Points

```
In [207]: showFeaturePoints(images[1])
```

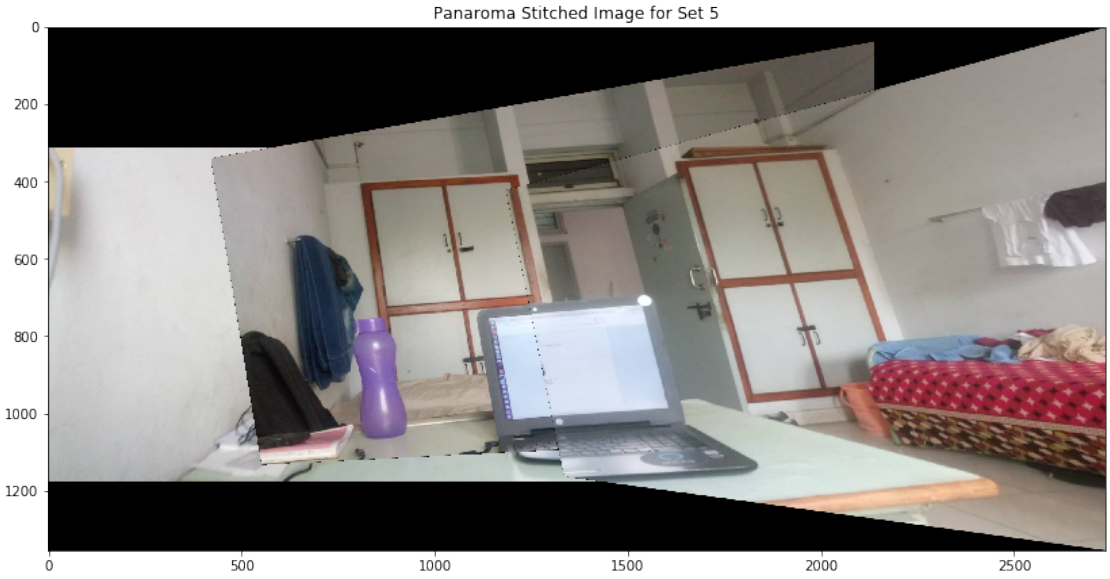



```
In [205]: res = images[0]
          for i in range(n_images-1):
              res = cv2.copyMakeBorder(res,300,300,600,600, cv2.BORDER_CONSTANT)
              set2Mat = Mapping(res, images[i+1],0.4)
              homo2 = homography(set2Mat[0],set2Mat[1])
              if i==1:
                  print("Homography_matrix is")
                  print(homo2)
              pana2 = panaroma(res,images[1+i],homo2)
              res = cropImg(pana2)
          plt.figure(figsize=[14,14])
          plt.imshow(res)
          plt.title("Panaroma Stitched Image for Set 5")
          plt.show()
```






```
Homography_matrix is
[[ -2.27796718e+01   4.63352541e+00   3.74839297e+04]
 [ -6.25007103e+00  -2.30409061e+01   2.69370578e+04]
 [ -1.07881714e-02  -6.83638391e-04   1.00000000e+00]]
```


Panaroma Stitched Image for Set 5
0
200
400
600
800
1000
1200
0
500
1000
1500
2000
2500