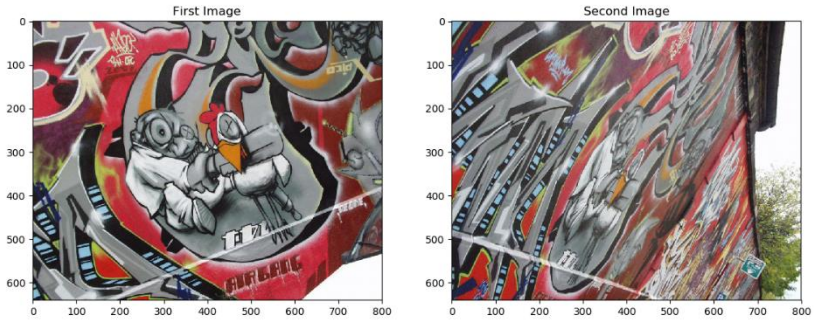
Q1. Homography Transformation Using Manually Given Points

```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
img2=cv.imread("C:/Users/Bumuthu Dilshan/Desktop/Assignment 2/img6.ppm")
img1=cv.imread("C:/Users/Bumuthu Dilshan/Desktop/Assignment 2/img2.ppm")
points=[[(450,240),(423,326)],[(352,567),(200,554)],[(583,157),(472,334)],[(100,133),(472,32)]]
A=np.zeros((8, 9))
d=0
f, axarr = plt.subplots(1, 2)
img1cvt=cv.cvtColor(img1,cv.COLOR_BGR2RGB)
img2cvt=cv.cvtColor(img2,cv.COLOR_BGR2RGB)
axarr[0].set_title("First Image")
axarr[0].imshow(img1cvt)
axarr[1].set_title('Second Image')
axarr[1].imshow(img2cvt)
plt.show()
for cor in points:
    A[d, :]=np.array([-cor[0][0], -
cor[0][1], -1, 0, 0, 0, cor[0][0] *
cor[1][0], cor[0][1] * cor[1][0],
cor[1][0]])
    A[d + 1, :] = np.array([0, 0, 0, -cor[0][0], -cor[0][1], -1, cor[0][0] * cor[1][1], cor[0][1] *
cor[1][1], cor[1][1]])
    d=d+2
u,s,v=np.linalg.svd(A)
s=0
H=np.reshape(v[8, :], (3, 3))
newImage= 255 * np.ones((1200, 1400, 3))
n = img1.shape[0]
m = img1.shape[1]
for x in range(n):
    for y in range(m):
        newImage[x + 200, y + 200, :]= img2[x, y, :]
        xi=np.array([y,x,1])
        xf=np.matmul(H, xi)
        newImage[int(xf[1] / xf[2]) + 200, int(xf[0] / xf[2]) + 200, :] = img1[x, y, :]
newImage=newImage.astype('uint8')
cv.imshow('Stitched Image', newImage)
cv.waitKey(0)
cv.destroyAllWindows()
```



To perform a plane projective transformation, there should be at least four points corresponding to another image.

Here four points are manually given and do the transformation. Using those points transformation matrix H is created. Taking pixel by pixel the transformation is done by multiplied by the transformation matrix.

Then two images are stitched in a same image.

Q2. SIFT features with RANSAC-based Homography computation

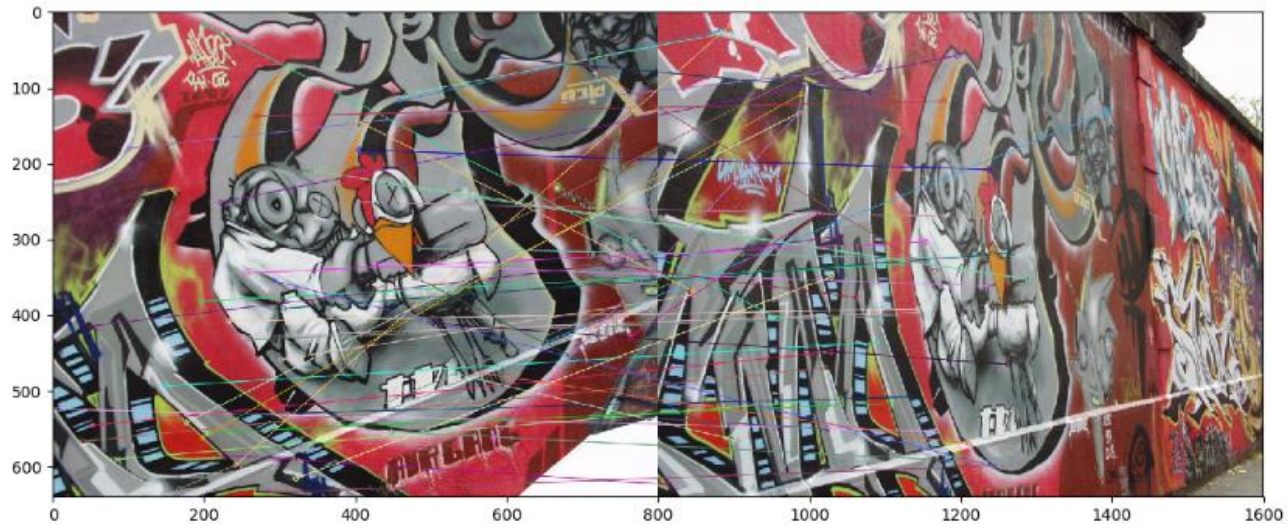
```
import matplotlib.pyplot as plt
import cv2
import numpy as np

img1 = cv2.cvtColor(cv2.imread('C:/Users/Bumuthu Dilshan/Desktop/Assignment 2/img2.ppm',
cv2.IMREAD_COLOR), cv2.COLOR_BGR2RGB)
img2 = cv2.cvtColor(cv2.imread('C:/Users/Bumuthu Dilshan/Desktop/Assignment 2/img5.ppm',
cv2.IMREAD_COLOR), cv2.COLOR_BGR2RGB)

sift = cv2.xfeatures2d.SIFT_create()
kp1, des1 = sift.detectAndCompute(img1, None)
kp2, des2 = sift.detectAndCompute(img2, None)
bf = cv2.BFMatcher()
matches = bf.knnMatch(des1, des2, k=2)
better = []
for m, n in matches:
    if m.distance < 0.7 * n.distance:
        better.append(m)
src_pts = np.float32([kp1[m.queryIdx].pt for m in better]).reshape(-1, 1, 2)
dst_pts = np.float32([kp2[m.trainIdx].pt for m in better]).reshape(-1, 1, 2)
M, mask = cv2.findHomography(src_pts, dst_pts, cv2.RANSAC, 5.0)
resultImg = 255 * np.ones((1000, 800, 3))
for x in range(img1.shape[0]):
    for y in range(img1.shape[1]):
        resultImg[x + 150, y, :] = img2[x, y, :]
        xi = np.array([y, x, 1])
        xf = np.matmul(M, xi)
        resultImg[int(xf[1] / xf[2]) + 150, int(xf[0] / xf[2]), :] = img1[x, y, :]
resultImg = resultImg.astype(np.uint8)
plt.imshow(resultImg)
plt.show()
img3 = cv2.drawMatches(img1, kp1, img2, kp2, better, None, flags=2)
plt.imshow(img3)
plt.show()
```

SIFT functions are used to detect features which are points and descriptor in the images. Brute force matcher is used to compare the key points and get the best matching points. RANSAC method is used to construct the homography transformation function which is used to transform two images.

Here the result has been better than the transformation with manually given points.



Brute force matching and Lowe's ratio test are given the matchings shown in the image.

Q4. Stitching more images

```
import matplotlib.pyplot as plt
import cv2
import numpy as np
def getPanorama(img1, img2, h, l, dx, dy):
    sift = cv2.xfeatures2d.SIFT_create()
    kp1, des1 = sift.detectAndCompute(img1, None)
    kp2, des2 = sift.detectAndCompute(img2, None)
    bf = cv2.BFMatcher()
    matches = bf.knnMatch(des1, des2, k=2)
    good = []
    for m, n in matches:
        if m.distance < 0.7 * n.distance:
            good.append(m)
    src_pts = np.float32([kp1[m.queryIdx].pt for m in good]).reshape(-1, 1, 2)
    dst_pts = np.float32([kp2[m.trainIdx].pt for m in good]).reshape(-1, 1, 2)
    M, mask = cv2.findHomography(src_pts, dst_pts, cv2.RANSAC, 5.0)
    resultImg = 255 * np.ones((h, l, 3))
    for x in range(img1.shape[0]):
        for y in range(img1.shape[1]):
            resultImg[x + dx, y + dy, :] = img2[x, y, :]
            xi = np.array([y, x, 1])
            xf = np.matmul(M, xi)
            resultImg[int(xf[1] / xf[2]) + dx, int(xf[0] / xf[2] + dy), :] = img1[x, y, :]
    resultImg = resultImg.astype('uint8')
    return resultImg
img1 = cv2.cvtColor(cv2.imread('C:/Users/Bumuthu Dilshan/Desktop/Assignment 2/img2.ppm',
cv2.IMREAD_COLOR), cv2.COLOR_BGR2RGB)
img2 = cv2.cvtColor(cv2.imread('C:/Users/Bumuthu Dilshan/Desktop/Assignment 2/img5.ppm',
cv2.IMREAD_COLOR), cv2.COLOR_BGR2RGB)
img3 = cv2.cvtColor(cv2.imread('C:/Users/Bumuthu Dilshan/Desktop/Assignment 2/img3.ppm',
cv2.IMREAD_COLOR), cv2.COLOR_BGR2RGB)
imidiateImg = getPanorama(img1, img2, 940, 800, 160, 0)
finalImg = getPanorama(img3, imidiateImg, 1050, 800, 20, 0)
plt.imshow(finalImg)
plt.show()
```

In getPanorama function, two images are stitched using RANSEC method. Many images can be stitched by taking two images at once. Here I have used img2 and img3 to stitch with img5.