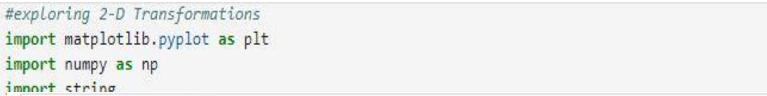
Department of Electronic and Telecommunication Engineering,

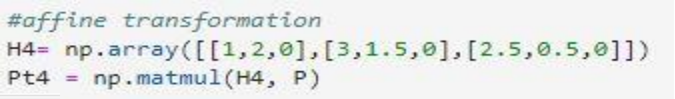
University of Moratuwa

R. Thiruparan

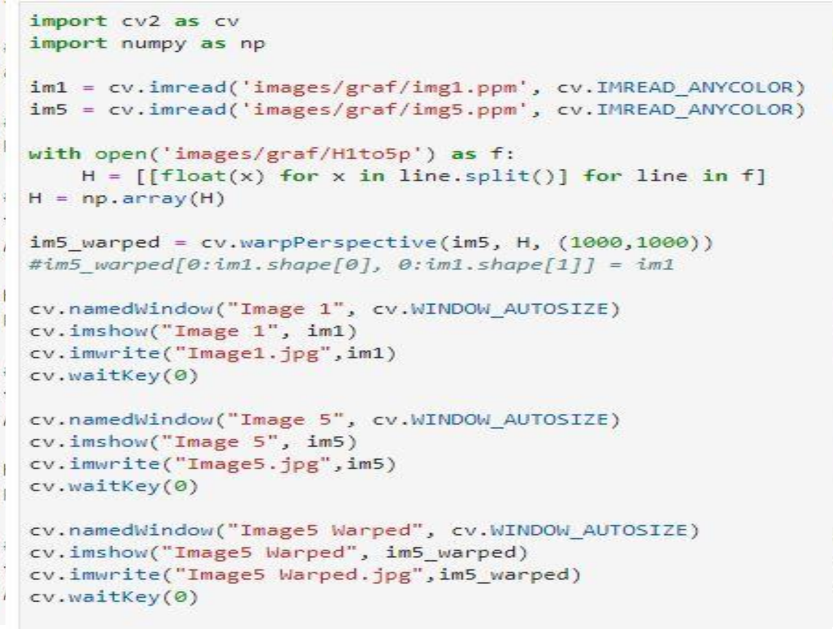
1.2D Transforms



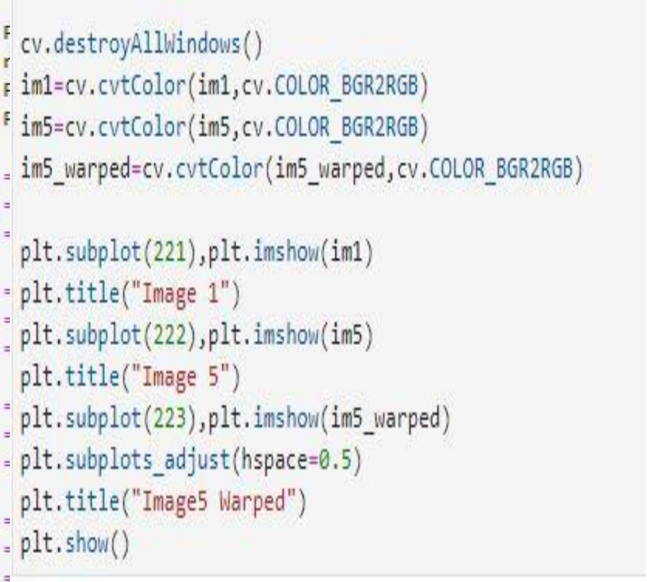
```
#exploring 2-D Transformations
import matplotlib.pyplot as plt
import numpy as np
import string
```



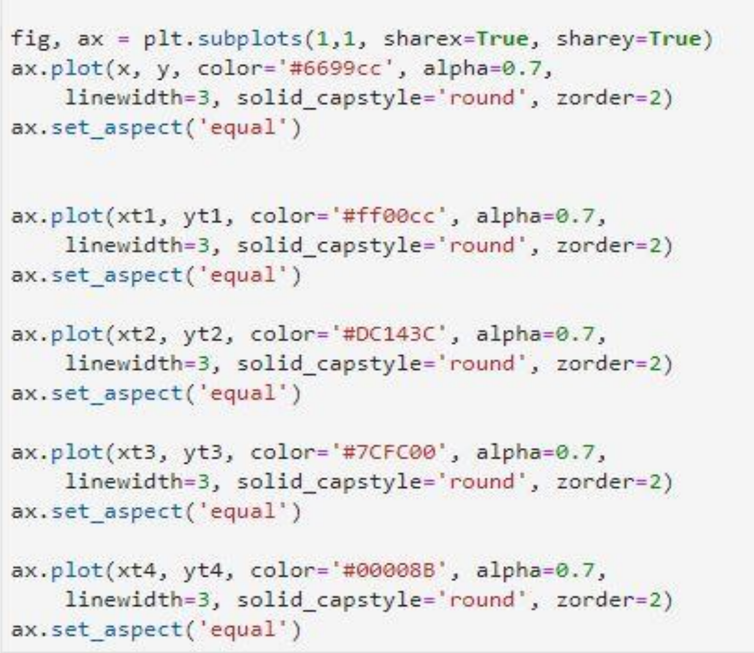
```
#affine transformation
H4= np.array([[1,2,0],[3,1.5,0],[2.5,0.5,0]])
Pt4 = np.matmul(H4, P)
```



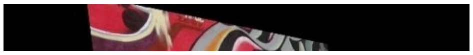
```
import cv2 as cv
import numpy as np

im1 = cv.imread('images/graf/img1.ppm', cv.IMREAD_ANYCOLOR)
im5 = cv.imread('images/graf/img5.ppm', cv.IMREAD_ANYCOLOR)

with open('images/graf/H1to5p') as f:
    H = [[float(x) for x in line.split()] for line in f]
H = np.array(H)

im5_warped = cv.warpPerspective(im5, H, (1000,1000))
#im5_warped[0:im1.shape[0], 0:im1.shape[1]] = im1

cv.namedWindow("Image 1", cv.WINDOW_AUTOSIZE)
cv.imshow("Image 1", im1)
cv.imwrite("Image1.jpg",im1)
cv.waitKey(0)

cv.namedWindow("Image 5", cv.WINDOW_AUTOSIZE)
cv.imshow("Image 5", im5)
cv.imwrite("Image5.jpg",im5)
cv.waitKey(0)

cv.namedWindow("Image5 Warped", cv.WINDOW_AUTOSIZE)
cv.imshow("Image5 Warped", im5_warped)
cv.imwrite("Image5 Warped.jpg",im5_warped)
cv.waitKey(0)
```

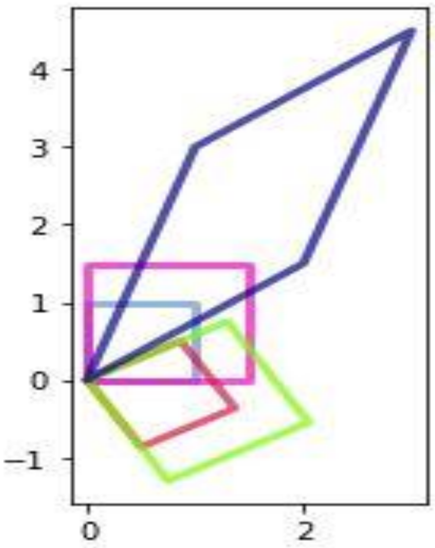


```
cv.destroyAllWindows()
im1=cv.cvtColor(im1,cv.COLOR_BGR2RGB)
im5=cv.cvtColor(im5,cv.COLOR_BGR2RGB)
im5_warped=cv.cvtColor(im5_warped,cv.COLOR_BGR2RGB)


plt.subplot(221),plt.imshow(im1)
plt.title("Image 1")
plt.subplot(222),plt.imshow(im5)
plt.title("Image 5")
plt.subplot(223),plt.imshow(im5_warped)
plt.subplots_adjust(hspace=0.5)
plt.title("Image5 Warped")
plt.show()
```



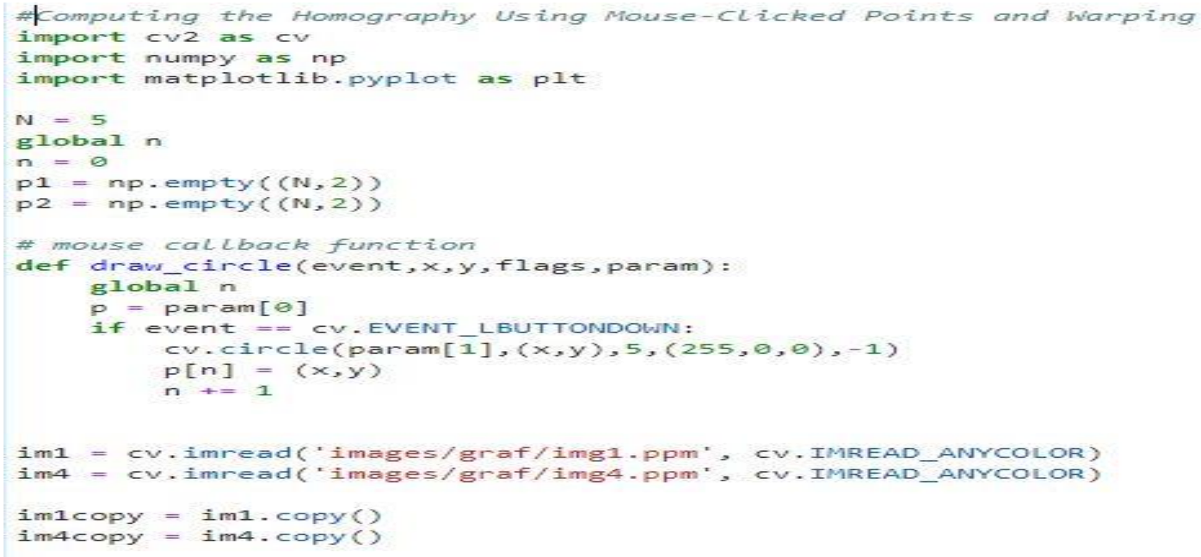
```
fig, ax = plt.subplots(1,1, sharex=True, sharey=True)
ax.plot(x, y, color='#6699cc', alpha=0.7,
    linewidth=3, solid_capstyle='round', zorder=2)
ax.set_aspect('equal')


ax.plot(xt1, yt1, color='#ff00cc', alpha=0.7,
    linewidth=3, solid_capstyle='round', zorder=2)
ax.set_aspect('equal')

ax.plot(xt2, yt2, color='#DC143C', alpha=0.7,
    linewidth=3, solid_capstyle='round', zorder=2)
ax.set_aspect('equal')

ax.plot(xt3, yt3, color='#7CFC00', alpha=0.7,
    linewidth=3, solid_capstyle='round', zorder=2)
ax.set_aspect('equal')

ax.plot(xt4, yt4, color='#00008B', alpha=0.7,
    linewidth=3, solid_capstyle='round', zorder=2)
ax.set_aspect('equal')
```



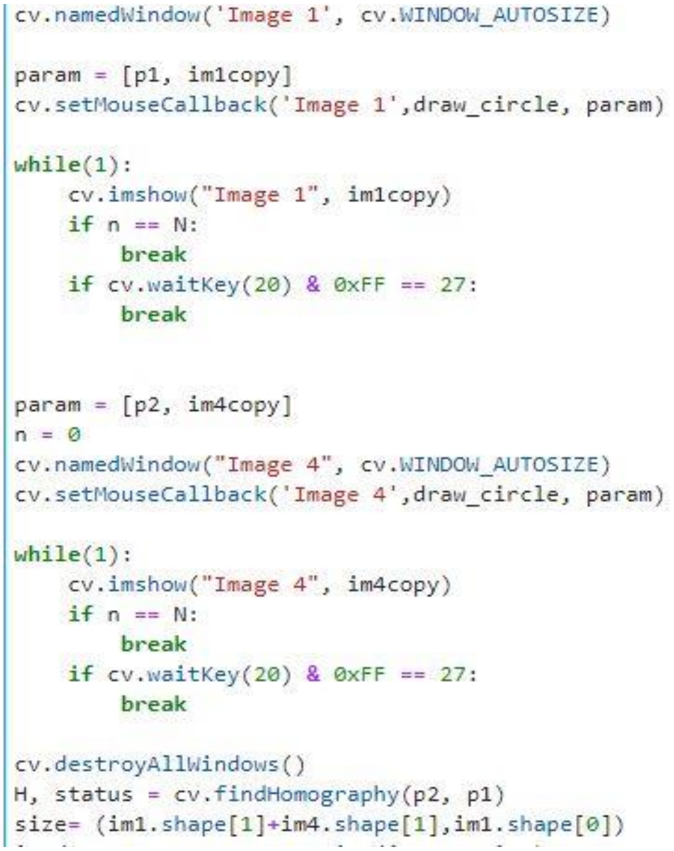


2.warping using a given homography

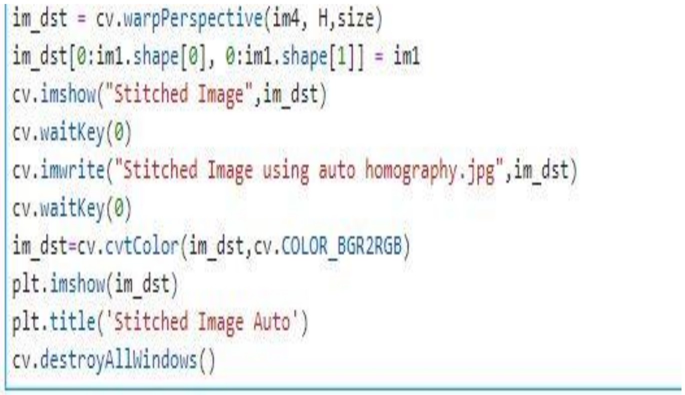
3. Computing the Homography Using Mouse-Clicked Points and Warping



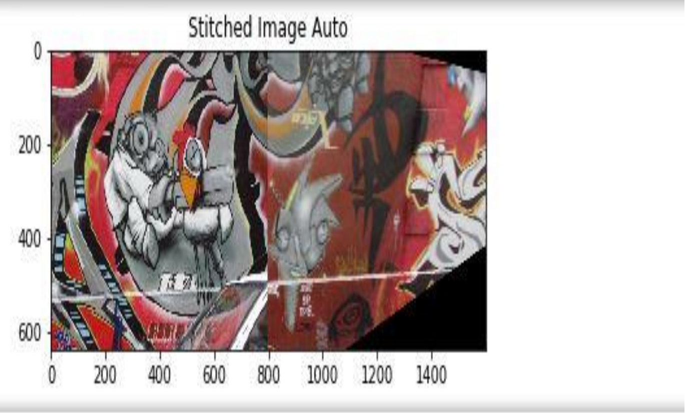
```
#Computing the Homography Using Mouse-Clicked Points and Warping
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt

N = 5
global n
n = 0
p1 = np.empty((N,2))
p2 = np.empty((N,2))

# mouse callback function
def draw_circle(event,x,y,flags,param):
    global n
    p = param[0]
    if event == cv.EVENT_LBUTTONDOWN:
        cv.circle(param[1],(x,y),5,(255,0,0),-1)
        p[n] = (x,y)
        n += 1


im1 = cv.imread('images/graf/img1.ppm', cv.IMREAD_ANYCOLOR)
im4 = cv.imread('images/graf/img4.ppm', cv.IMREAD_ANYCOLOR)

im1copy = im1.copy()
im4copy = im4.copy()
```



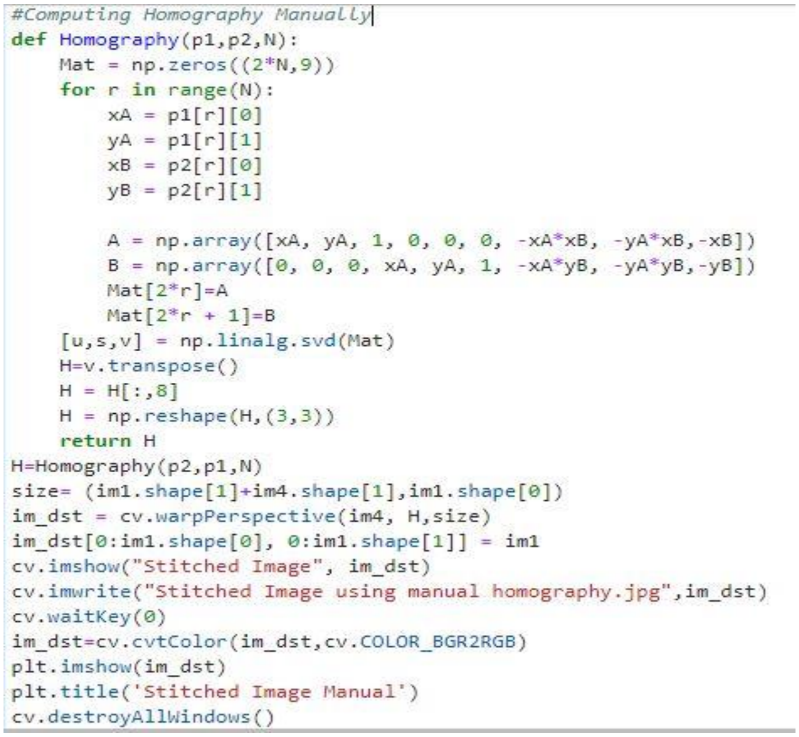
```
cv.namedWindow('Image 1', cv.WINDOW_AUTOSIZE)

param = [p1, im1copy]
cv.setMouseCallback('Image 1',draw_circle, param)

while(1):
    cv.imshow("Image 1", im1copy)
    if n == N:
        break
    if cv.waitKey(20) & 0xFF == 27:
        break


param = [p2, im4copy]
n = 0
cv.namedWindow("Image 4", cv.WINDOW_AUTOSIZE)
cv.setMouseCallback('Image 4',draw_circle, param)

while(1):
    cv.imshow("Image 4", im4copy)
    if n == N:
        break
    if cv.waitKey(20) & 0xFF == 27:
        break

cv.destroyAllWindows()
H, status = cv.findHomography(p2, p1)
size= (im1.shape[1]+im4.shape[1],im1.shape[0])
```



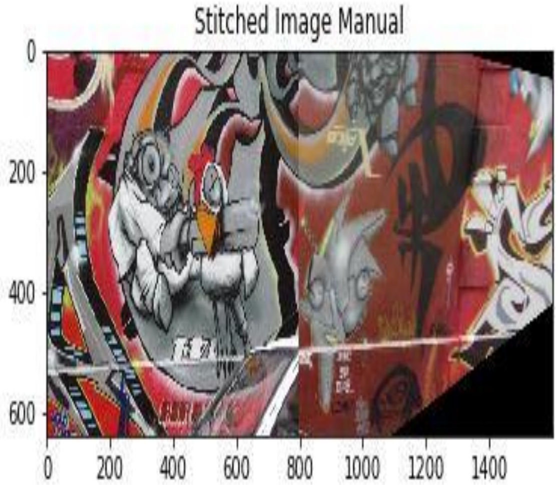
```
im_dst = cv.warpPerspective(im4, H,size)
im_dst[0:im1.shape[0], 0:im1.shape[1]] = im1
cv.imshow("Stitched Image",im_dst)
cv.waitKey(0)
cv.imwrite("Stitched Image using auto homography.jpg",im_dst)
cv.waitKey(0)
im_dst=cv.cvtColor(im_dst,cv.COLOR_BGR2RGB)
plt.imshow(im_dst)
plt.title('Stitched Image Auto')
cv.destroyAllWindows()
```



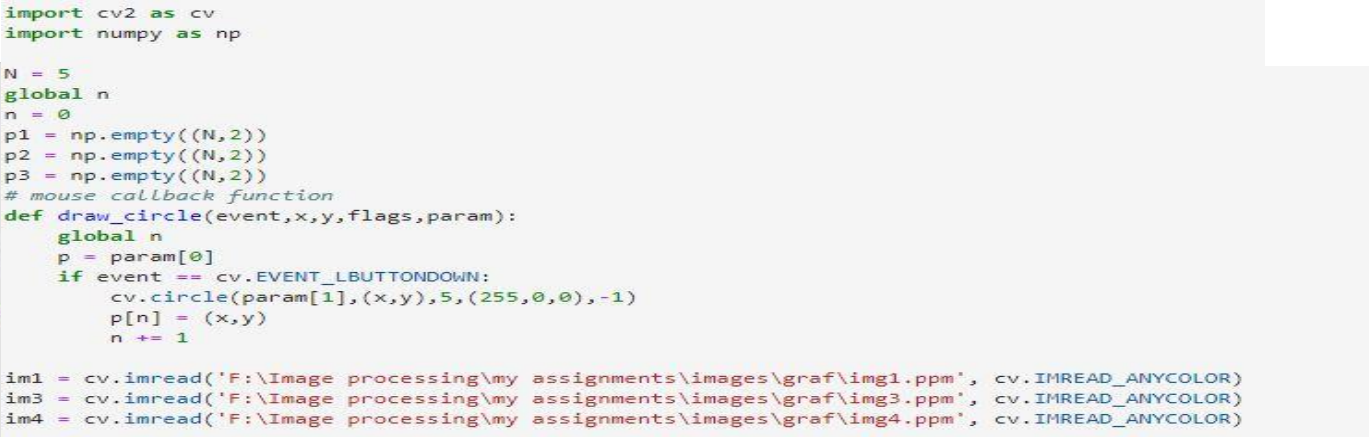
4. computing Homography manually



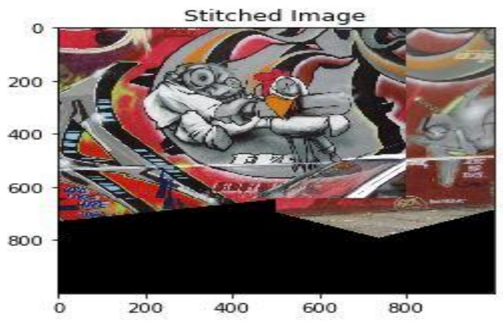
```
#Computing Homography Manually
def Homography(p1,p2,N):
    Mat = np.zeros((2*N,9))
    for r in range(N):
        xA = p1[r][0]
        yA = p1[r][1]
        xB = p2[r][0]
        yB = p2[r][1]

        A = np.array([xA, yA, 1, 0, 0, 0, -xA*xB, -yA*xB,-xB])
        B = np.array([0, 0, 0, xA, yA, 1, -xA*yB, -yA*yB,-yB])
        Mat[2*r]=A
        Mat[2*r + 1]=B
    [u,s,v] = np.linalg.svd(Mat)
    H=v.transpose()
    H = H[:,8]
    H = np.reshape(H,(3,3))
    return H
H=Homography(p2,p1,N)
size= (im1.shape[1]+im4.shape[1],im1.shape[0])
im_dst = cv.warpPerspective(im4, H,size)
im_dst[0:im1.shape[0], 0:im1.shape[1]] = im1
cv.imshow("Stitched Image", im_dst)
cv.imwrite("Stitched Image using manual homography.jpg",im_dst)
cv.waitKey(0)
im_dst=cv.cvtColor(im_dst,cv.COLOR_BGR2RGB)
plt.imshow(im_dst)
plt.title('Stitched Image Manual')
cv.destroyAllWindows()
```

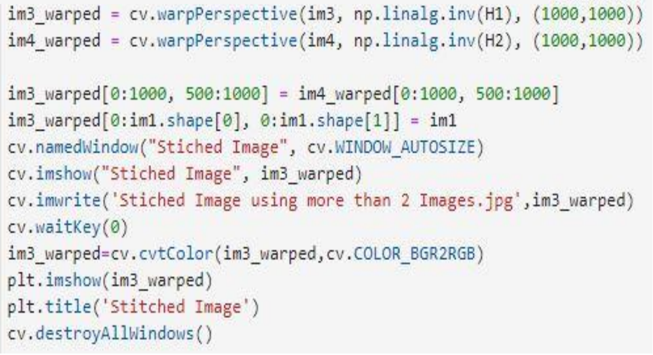


5.stich more than two images using mouse clicked points



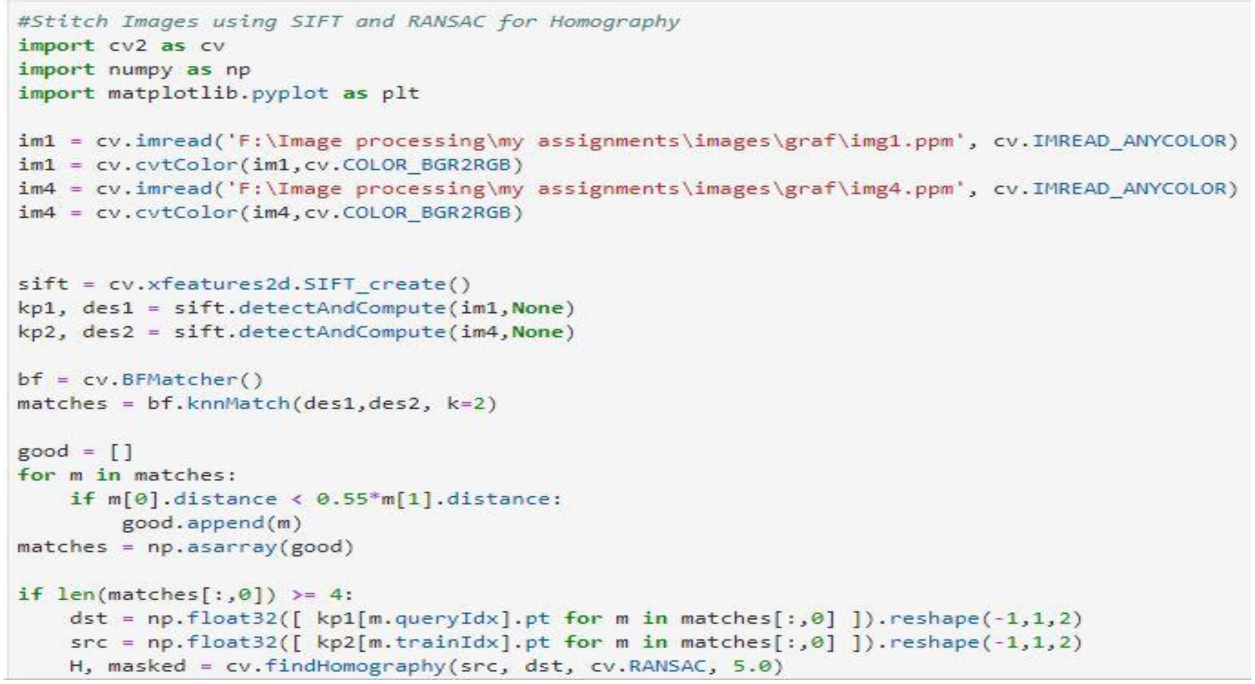
```
import cv2 as cv
import numpy as np

N = 5
global n
n = 0
p1 = np.empty((N,2))
p2 = np.empty((N,2))
p3 = np.empty((N,2))
# mouse callback function
def draw_circle(event,x,y,flags,param):
    global n
    p = param[0]
    if event == cv.EVENT_LBUTTONDOWN:
        cv.circle(param[1],(x,y),5,(255,0,0),-1)
        p[n] = (x,y)
        n += 1

im1 = cv.imread('F:\Image processing\my assignments\images\graf\img1.ppm', cv.IMREAD_ANYCOLOR)
im3 = cv.imread('F:\Image processing\my assignments\images\graf\img3.ppm', cv.IMREAD_ANYCOLOR)
im4 = cv.imread('F:\Image processing\my assignments\images\graf\img4.ppm', cv.IMREAD_ANYCOLOR)
```



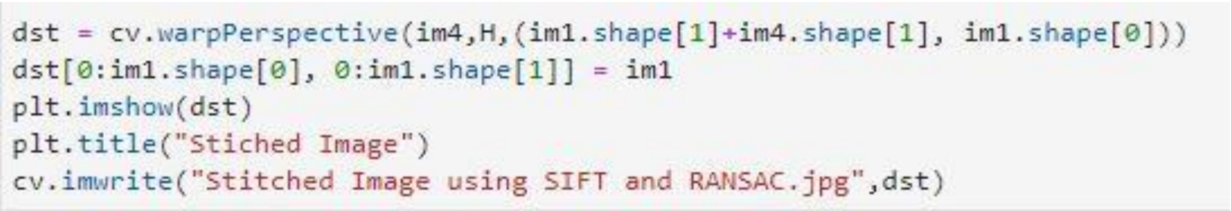


```
im3_warped = cv.warpPerspective(im3, np.linalg.inv(H1), (1000,1000))
im4_warped = cv.warpPerspective(im4, np.linalg.inv(H2), (1000,1000))

im3_warped[0:1000, 500:1000] = im4_warped[0:1000, 500:1000]
im3_warped[0:im1.shape[0], 0:im1.shape[1]] = im1
cv.namedWindow("Stiched Image", cv.WINDOW_AUTOSIZE)
cv.imshow("Stiched Image", im3_warped)
cv.imwrite('Stiched Image using more than 2 Images.jpg',im3_warped)
cv.waitKey(0)
im3_warped=cv.cvtColor(im3_warped,cv.COLOR_BGR2RGB)
plt.imshow(im3_warped)
plt.title('Stitched Image')
cv.destroyAllWindows()
```

6. Stitch Images using SIFT and RANSAC for Homography

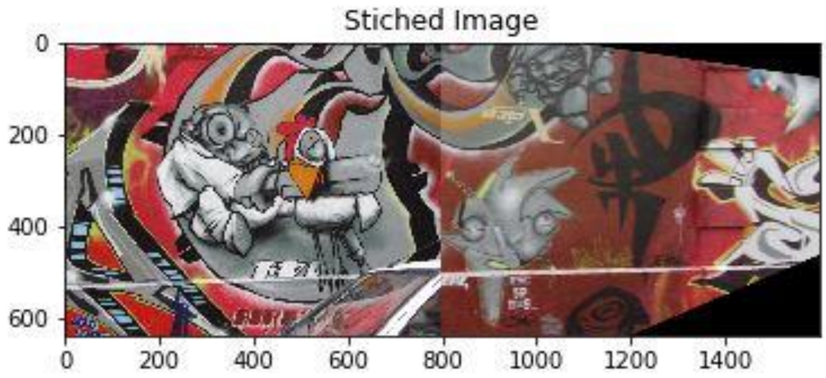


```python
#Stitch Images using SIFT and RANSAC for Homography
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt

im1 = cv.imread('F:\Image processing\my assignments\images\graf\img1.ppm', cv.IMREAD_ANYCOLOR)
im1 = cv.cvtColor(im1,cv.COLOR_BGR2RGB)
im4 = cv.imread('F:\Image processing\my assignments\images\graf\img4.ppm', cv.IMREAD_ANYCOLOR)
im4 = cv.cvtColor(im4,cv.COLOR_BGR2RGB)


sift = cv.xfeatures2d.SIFT_create()
kp1, des1 = sift.detectAndCompute(im1,None)
kp2, des2 = sift.detectAndCompute(im4,None)

bf = cv.BFMatcher()
matches = bf.knnMatch(des1,des2, k=2)

good = []
for m in matches:
    if m[0].distance < 0.55*m[1].distance:
        good.append(m)
matches = np.asarray(good)

if len(matches[:,0]) >= 4:
    dst = np.float32([ kp1[m.queryIdx].pt for m in matches[:,0] ]).reshape(-1,1,2)
    src = np.float32([ kp2[m.trainIdx].pt for m in matches[:,0] ]).reshape(-1,1,2)
    H, masked = cv.findHomography(src, dst, cv.RANSAC, 5.0)
```



```python
dst = cv.warpPerspective(im4,H,(im1.shape[1]+im4.shape[1], im1.shape[0]))
dst[0:im1.shape[0], 0:im1.shape[1]] = im1
plt.imshow(dst)
plt.title("Stiched Image")
cv.imwrite("Stitched Image using SIFT and RANSAC.jpg",dst)
```

```
True
```


Stiched Image