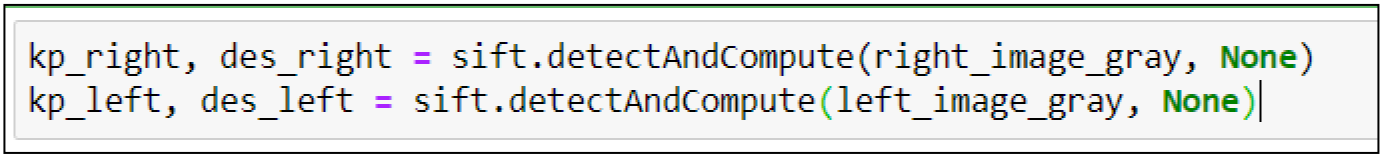
Image Stitching

Objective: The task is to stitch two given images, left.jpg and right.jpg to construct a panorama image.

Step-1- Find key points in the given images

I am using SIFT detector to extract keypoints and feature descriptors of the two images.



```
kp_right, des_right = sift.detectAndCompute(right_image_gray, None)
kp_left, des_left = sift.detectAndCompute(left_image_gray, None)
```

Step-2 – Match Keypoint:

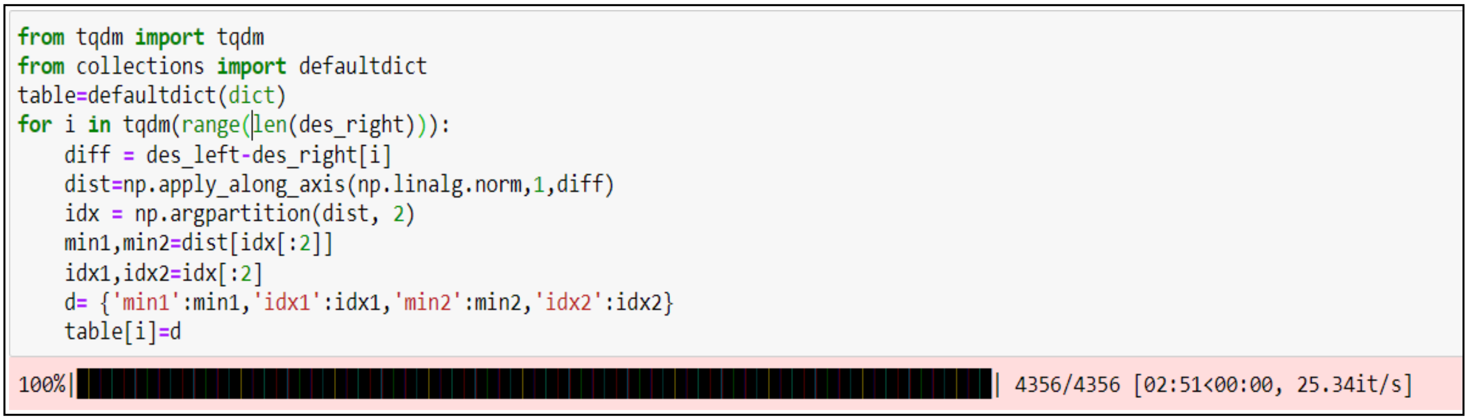
We need to find out the two nearest neighbor of a right image keypoint with respect to distance from the left image keypoints.

I used **np.linalg.norm** to calculate the **Euclidean distance**.

To calculate the two min distance and the corresponding index I used **np.argpartition** (https://numpy.org/doc/stable/reference/generated/numpy.argpartition.html)

I created a dictionary with the index of right image as key and value as:

{'min1’: min1,'idx1’: idx1,'min2’: min2,'idx2': idx2} -→ min1, min2 – two nearest neighbor, idx1,idx2 – index of two nearest neighbor in left image.

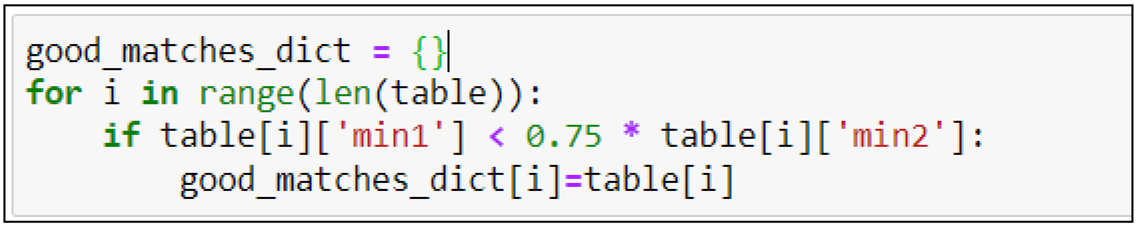


```
from tqdm import tqdm
from collections import defaultdict
table=defaultdict(dict)
for i in tqdm(range(len(des_right))):
    diff = des_left-des_right[i]
    dist=np.apply_along_axis(np.linalg.norm,1,diff)
    idx = np.argpartition(dist, 2)
    min1,min2=dist[idx[:2]]
    idx1,idx2=idx[:2]
    d= {'min1':min1,'idx1':idx1,'min2':min2,'idx2':idx2}
    table[i]=d
```

```
100%|██████████████████████████████████████████████████| 4356/4356 [02:51<00:00, 25.34it/s]
```

Step-3: Ratio test

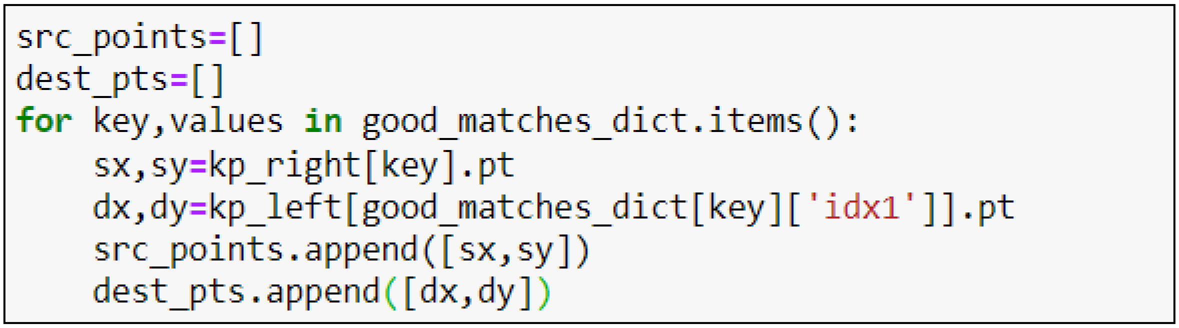
I used ratio test to find good matches using **η=0.75**



```
good_matches_dict = {}
for i in range(len(table)):
    if table[i]['min1'] < 0.75 * table[i]['min2']:
        good_matches_dict[i]=table[i]
```

I extracted coordinates of the good matching points from the keypoints given by SIFT.

I created src_pts(right_image) and dest_pts(left_image) list which stores coordinates values which can be further used for calculating homography matrix.



```
src_points=[]
dest_pts=[]
for key,values in good_matches_dict.items():
    sx,sy=kp_right[key].pt
    dx,dy=kp_left[good_matches_dict[key]['idx1']].pt
    src_points.append([sx,sy])
    dest_pts.append([dx,dy])
```

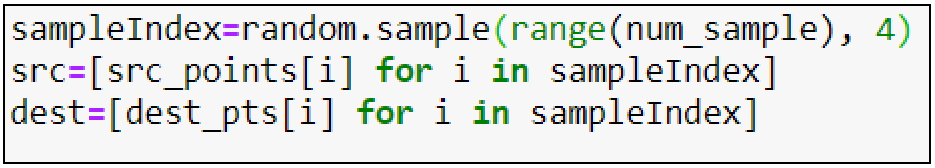
Step-4: Implementing Ransac Algorithm

Number of iteration: i=1000

Smallest number of points required: n = 4

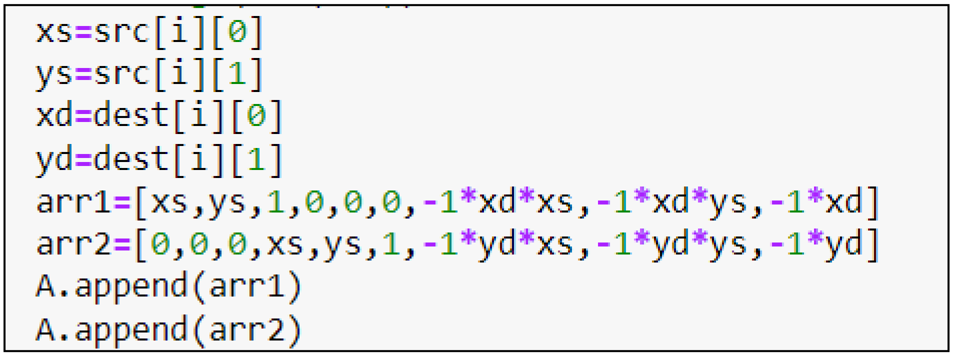
Threshold = 5

1. Randomly sample 4 points from src and dest list.



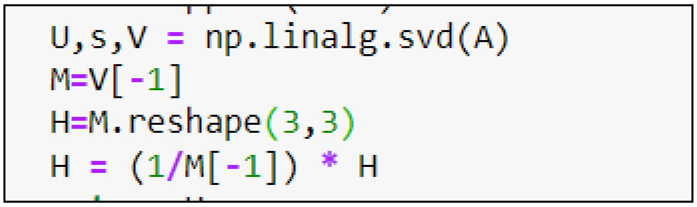
```
sampleIndex=random.sample(range(num_sample), 4)
src=[src_points[i] for i in sampleIndex]
dest=[dest_pts[i] for i in sampleIndex]
```

2. Created matrix A using the 4 sampled points



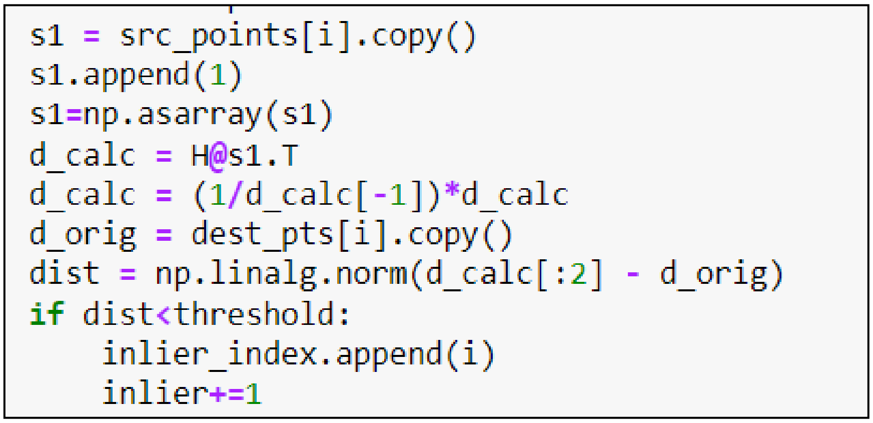
```
 xs=src[i][0]
 ys=src[i][1]
 xd=dest[i][0]
 yd=dest[i][1]
 arr1=[xs,ys,1,0,0,0,-1*xd*xs,-1*xd*ys,-1*xd]
 arr2=[0,0,0,xs,ys,1,-1*yd*xs,-1*yd*ys,-1*yd]
 A.append(arr1)
 A.append(arr2)
```

3. Apply SVD on matrix A and get the last element of V.T and created homography matrix of size 3x3. To maintain the degree of freedom as 8 we need to divide all the elements of homography matrix with the last element of the matrix.



```
U,s,V = np.linalg.svd(A)
M=V[-1]
H=M.reshape(3,3)
H = (1/M[-1]) * H
```

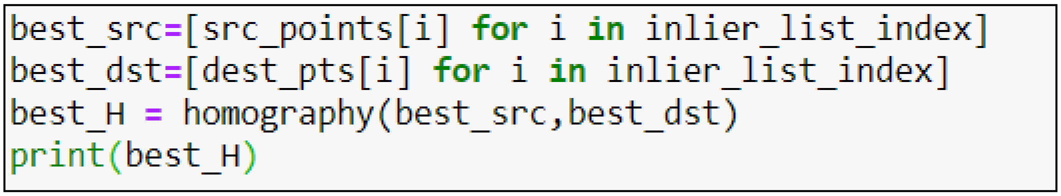
4. Using the Homography matrix and src points I calculated the dest points and compared it with the original dest points taking threshold value as 5.



```
s1 = src_points[i].copy()
s1.append(1)
s1=np.asarray(s1)
d_calc = H@s1.T
d_calc = (1/d_calc[-1])*d_calc
d_orig = dest_pts[i].copy()
dist = np.linalg.norm(d_calc[:2] - d_orig)
if dist<threshold:
    inlier_index.append(i)
    inlier+=1
```

If the Euclidean distance between dest_original and dest_calculated is less than threshold, then inliercount will increase.

5. For the iteration where I get maximum number of inliers, I store the inlier points.
6. Calculate best homography matrix using the inlier points stored in the last step.



```
best_src=[src_points[i] for i in inlier_list_index]
best_dst=[dest_pts[i] for i in inlier_list_index]
best_H = homography(best_src,best_dst)
print(best_H)
```

Step-5: Stitching Image using best homography matrix

1. Find the corners of left and the right image.
2. Perform the perspective transform of the corners of the right image
3. Concatenate the left image and transformed right image
4. Get the maximum and minimum coordinates of the concatenated image
5. Create a translation matrix to perform affine transformation using the minimum x and y coordinates
6. Using warp perspective library of the opencv stitch the image. It takes homography matrix and canvas size.