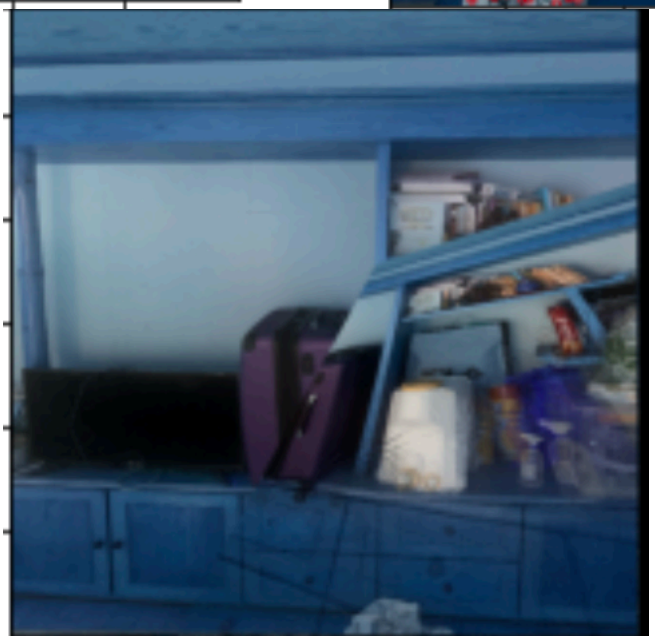
```
#%%
import numpy as np
import cv2
from matplotlib import pyplot as plt
import random
import tran

#%%
imageA = cv2.imread("3.jpg")

imageB = cv2.imread("4.jpg")
imageA = cv2.resize(imageA, (600,600))
imageB = cv2.resize(imageB, (600,600))

# %%
orb = cv2.xfeatures2d.SIFT_create()

# Find the key points and descriptors with ORB
keypoints1, descriptors1 = orb.detectAndCompute(imageA, None)
keypoints2, descriptors2 = orb.detectAndCompute(imageB, None)
imgA = cv2.drawKeypoints(imageA, keypoints1, None,
color=(255,0,0))
plt.imshow(imgA, cmap="gray")
plt.show()
imgB = cv2.drawKeypoints(imageB, keypoints2, None,
color=(255,0,0))
plt.imshow(imgB, cmap="gray")
plt.show()

bf = cv2.BFMatcher_create()

# Find matching points
matches = bf.knnMatch(descriptors1,descriptors2,k=2)
good = []
for m, n in matches:
    if(m.distance < 0.7 * n.distance):
        good.append(m)
        print(m.distance)
good.sort(key=lambda x: x.distance, reverse=False)
```

```
#%%
print(good[0].distance)
while(True):
    randomIndex = []
    for i in range(4):
        randomIndex.append(random.randint(0, len(good)-1))

    point1 = []
    point2 = []
    for i in randomIndex:

        x1 = keypoints1[good[i].queryIdx].pt[0]
        y1 = keypoints1[good[i].queryIdx].pt[1]
        x2 = keypoints2[good[i].trainIdx].pt[0]
        y2 = keypoints2[good[i].trainIdx].pt[1]
        point1.append([x1, y1])
        point2.append([x2, y2])
    # print(np.int32(point1))
    # H = tran.getPerspectiveTransform(point2, point1)
    H = cv2.getPerspectiveTransform(np.float32(point2),
np.float32(point1))
    # print("Perspective")
    error = good[0].distance * 5
    inliner = 0
    outliner = 0
    for i in good:
        x1 = keypoints1[i.queryIdx].pt[0]
        y1 = keypoints1[i.queryIdx].pt[1]
        res = H.dot([[x1],[y1],[1]])
        x2 = keypoints2[i.trainIdx].pt[0]#real ans
        y2 = keypoints2[i.trainIdx].pt[1]
        distance = ( (res[0] - x2)**2 + (res[1] - y2)** 2) ** 0.5

        if(distance < error):
            inliner += 1
        else:
            outliner += 1
```

```
    print(inliner, outliner)
    if(inliner/outliner > 0.98 and outliner != 0):
        break

#%%
#find good H
# dst = tran.warpPerspective(imageB, H , (800, 1200))
dst = cv2.warpPerspective(imageB, H, (800, 1200))
res = np.zeros((800,1200,3), dtype=np.uint8)

for i in range(imageA.shape[1]):
    for j in range(imageA.shape[0]):
        res[j][i] = imageA[j][i]

for i in range(imageB.shape[1]):
    for j in range(imageB.shape[0]):
        u, v , w = H.dot([[i], [j], [1]])
        x = int(u[0]/w[0])
        y = int(v[0]/w[0])
        if x >= 0 and x < imageB.shape[0]:#邊界處理
            if y >= 0 and y < imageB.shape[1]:
                res[x,y,:] = imageB[i,j,:]#destination scanning
        # if(x>= imageB.shape[1]):
        #     x = imageB.shape[1] - 1
        # if(x < 0 ):
        #     x = 0
        # if(y>= imageB.shape[0]):
        #     y = imageB.shape[0] - 1
        # if(y < 0):
        #     y = 0

        # res[y][x] = imageB[j][i]

plt.imshow(dst)
plt.imshow(res)

#pick 4 points
#per tranform
#m to n all keypoint
#cal point dis < errors
```

```
# take good point
#cal ratio tell good H
#repat until find good H
#project A to B
```



很難很麻煩