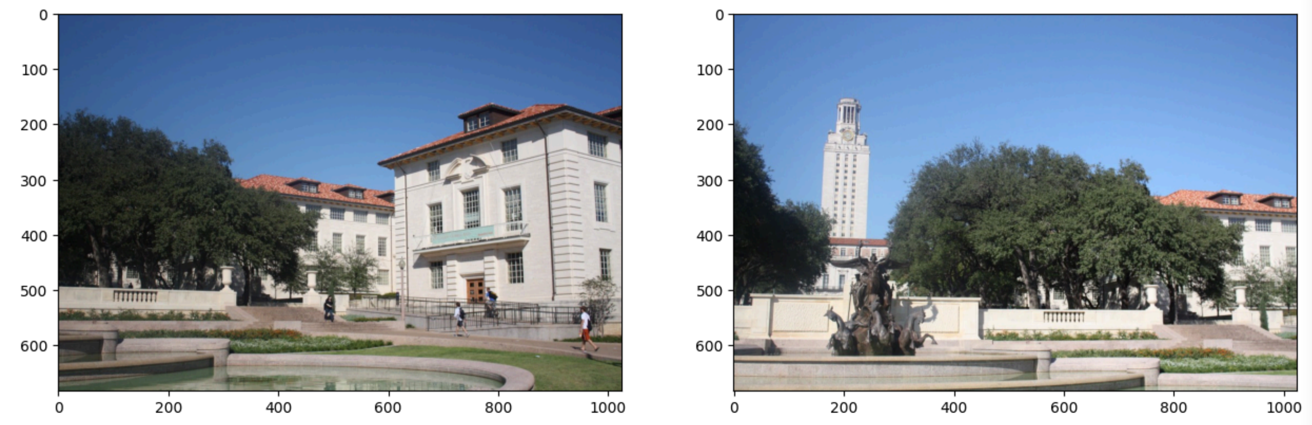
```
# do các bản opencv mới nhất không có SIFT (có bản quyền) nên ta cần downgrad
!pip uninstall opencv-python opencv-contrib-python
!pip install opencv-python opencv-contrib-python

```

```
WARNING: Skipping opencv-python as it is not installed.
WARNING: Skipping opencv-contrib-python as it is not installed.
Collecting opencv-python
  Downloading opencv_python-4.10.0.84-cp37-abi3-manylinux_2_17_x86_64.manylin
Collecting opencv-contrib-python
  Downloading opencv_contrib_python-4.10.0.84-cp37-abi3-manylinux_2_17_x86_64
Requirement already satisfied: numpy>=1.21.2 in /usr/local/lib/python3.10/dis
Downloading opencv_python-4.10.0.84-cp37-abi3-manylinux_2_17_x86_64.manylinux
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 62.5/62.5 MB 18.0 MB/s eta 0:00:0
Downloading opencv_contrib_python-4.10.0.84-cp37-abi3-manylinux_2_17_x86_64.m
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 68.7/68.7 MB 9.2 MB/s eta 0:00:00
Installing collected packages: opencv-python, opencv-contrib-python
Successfully installed opencv-contrib-python-4.10.0.84 opencv-python-4.10.0.8
```

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

```
def plot_img(img, size=(7,7), title=""):
    cmap = "gray" if len(img.shape) == 2 else None
    plt.figure(figsize=size)
    plt.imshow(img, cmap=cmap)
    plt.suptitle(title)
    plt.show()

def plot_imgs(imgs, cols=5, size=7, title=""):
    rows = len(imgs)//cols + 1
    fig = plt.figure(figsize=(cols*size, rows*size))
    for i, img in enumerate(imgs):
        cmap="gray" if len(img.shape) == 2 else None
        fig.add_subplot(rows, cols, i+1)
        plt.imshow(img, cmap=cmap)
    plt.suptitle(title)
    plt.show()


src_img = imageio.imread('http://www.ic.unicamp.br/~helio/imagens_registro/fot
tar_img = imageio.imread('http://www.ic.unicamp.br/~helio/imagens_registro/fot
src_gray = cv2.cvtColor(src_img, cv2.COLOR_RGB2GRAY)
tar_gray = cv2.cvtColor(tar_img, cv2.COLOR_RGB2GRAY)
plot_imgs([src_img, tar_img], size=8)
```

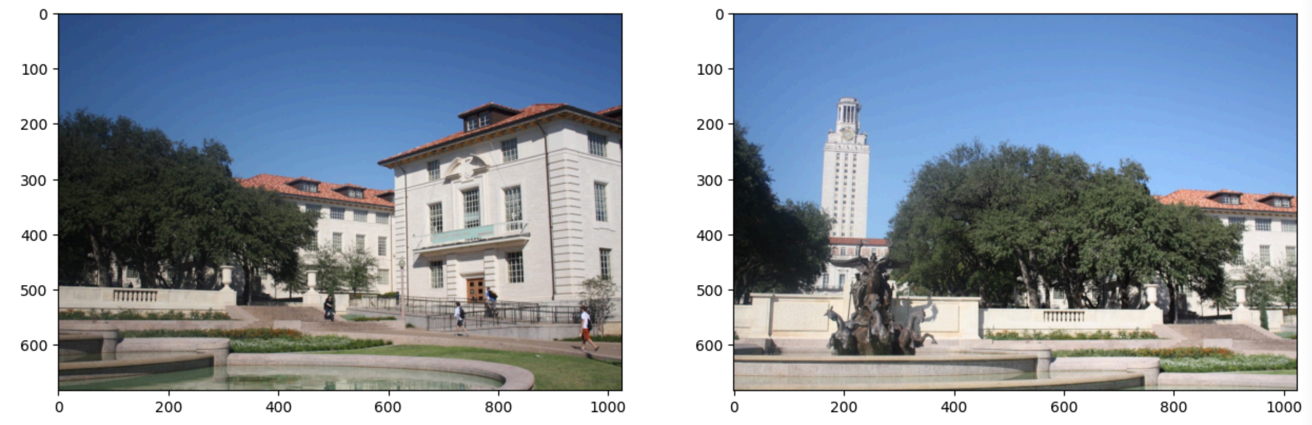
```
<ipython-input-6-a3971d96f4d1>:19: DeprecationWarning: Starting with ImageIO 
  src_img = imageio.imread('http://www.ic.unicamp.br/~helio/imagens_registro/
<ipython-input-6-a3971d96f4d1>:20: DeprecationWarning: Starting with ImageIO 
  tar_img = imageio.imread('http://www.ic.unicamp.br/~helio/imagens_registro/
```



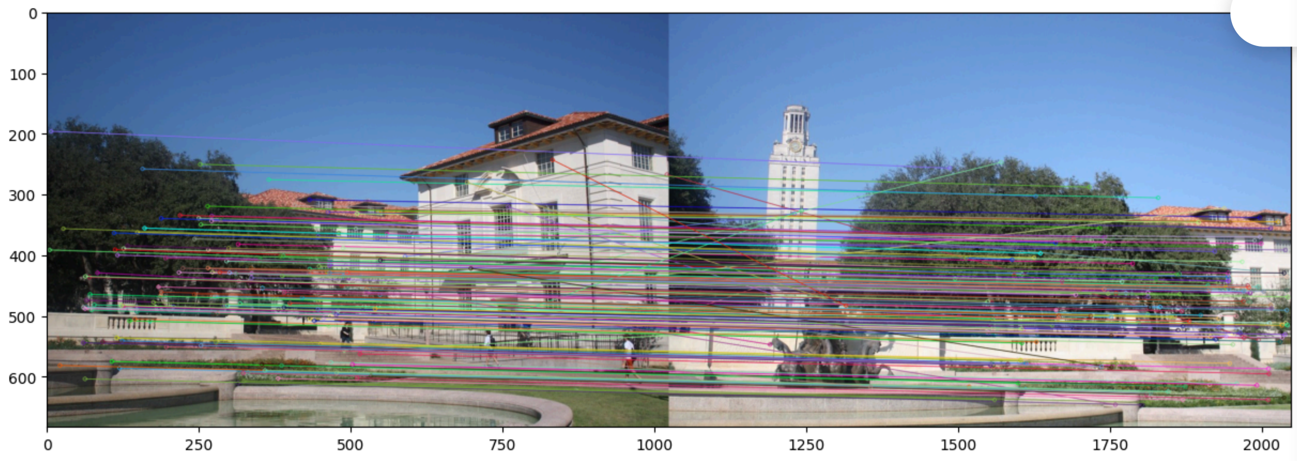
```
def plot_imgs(imgs, cols=5, size=7, title=""):
    rows = len(imgs)//cols + 1
    fig = plt.figure(figsize=(cols*size, rows*size))
    for i, img in enumerate(imgs):
        cmap="gray" if len(img.shape) == 2 else None
        fig.add_subplot(rows, cols, i+1)
        plt.imshow(img, cmap=cmap)
    plt.suptitle(title)
    plt.show()



```

```
src_img = imageio.imread('http://www.ic.unicamp.br/~helio/imagens_registro/fot
tar_img = imageio.imread('http://www.ic.unicamp.br/~helio/imagens_registro/fot
src_gray = cv2.cvtColor(src_img, cv2.COLOR_RGB2GRAY)
tar_gray = cv2.cvtColor(tar_img, cv2.COLOR_RGB2GRAY)
plot_imgs([src_img, tar_img], size=8)
```

```
<ipython-input-8-06ac79e0b437>:1: DeprecationWarning: Starting with ImageIO v
  src_img = imageio.imread('http://www.ic.unicamp.br/~helio/imagens_registro/
<ipython-input-8-06ac79e0b437>:2: DeprecationWarning: Starting with ImageIO v
  tar_img = imageio.imread('http://www.ic.unicamp.br/~helio/imagens_registro/
```



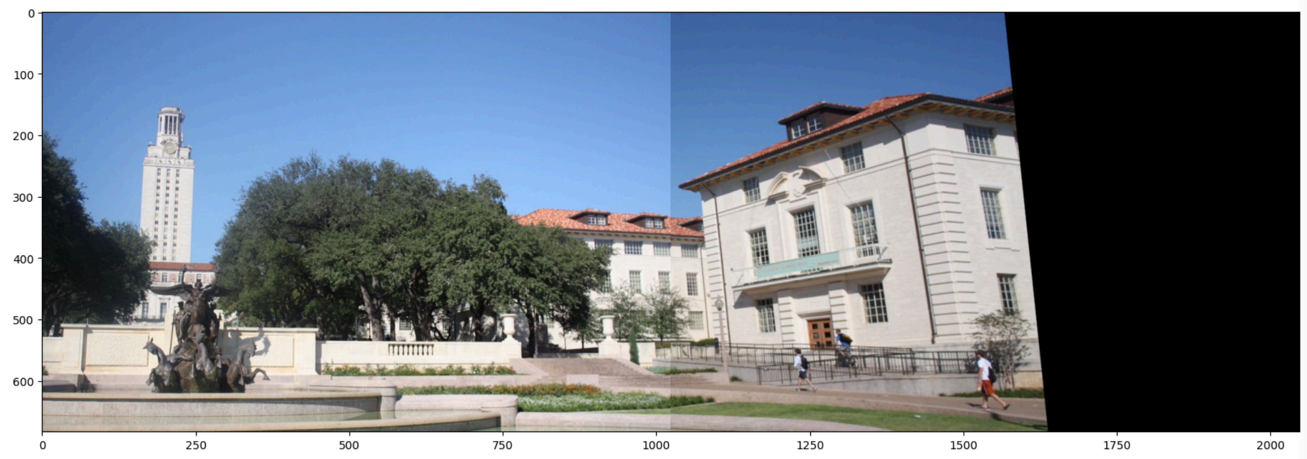
```
SIFT_detector = cv2.xfeatures2d.SIFT_create()
kp1, des1 = SIFT_detector.detectAndCompute(src_gray, None)
kp2, des2 = SIFT_detector.detectAndCompute(tar_gray, None)

## Match keypoint
bf = cv2.BFMatcher(cv2.NORM_L2, crossCheck=False)

## Bruce Force KNN trả về list k ứng viên cho mỗi keypoint.
rawMatches = bf.knnMatch(des1, des2, 2)
matches = []
ratio = 0.75

for m,n in rawMatches:
    # giữ lại các cặp keypoint sao cho với kp1, khoảng cách giữa kp1 với ứng v
    if m.distance < n.distance * 0.75:
        matches.append(m)

# do có cả nghìn match keypoint, ta chỉ lấy tầm 100 -> 200 cặp tốt nhất để tốc
matches = sorted(matches, key=lambda x: x.distance, reverse=True)
matches = matches[:200]

img3 = cv2.drawMatches(src_img, kp1, tar_img, kp2, matches, None,flags=cv2.Dra
plot_img(img3, size=(15, 10))

## Nhìn vào hình dưới đây, ta thấy các cặp Keypoint giữa 2 ảnh đã được match k

```



```
kp1 = np.float32([kp.pt for kp in kp1])
kp2 = np.float32([kp.pt for kp in kp2])
pts1 = np.float32([kp1[m.queryIdx] for m in matches])
pts2 = np.float32([kp2[m.trainIdx] for m in matches])

# estimate the homography between the sets of points
(H, status) = cv2.findHomography(pts1, pts2, cv2.RANSAC)
print(H)



```

```
[[ 7.71511477e-01  2.56342115e-02  4.48360482e+02]
 [-1.30549192e-01  9.03445838e-01  7.72905034e+01]
 [-2.01517330e-04 -4.37391165e-05  1.00000000e+00]]
```

```
h1, w1 = src_img.shape[:2]
h2, w2 = tar_img.shape[:2]
result = cv2.warpPerspective(src_img, H, (w1+w2, h1))
result[0:h2, 0:w2] = tar_img
plot_img(result, size=(20,10))

```



```
Start coding or generate with AI.
```