SarahBrown - CV - HW 3

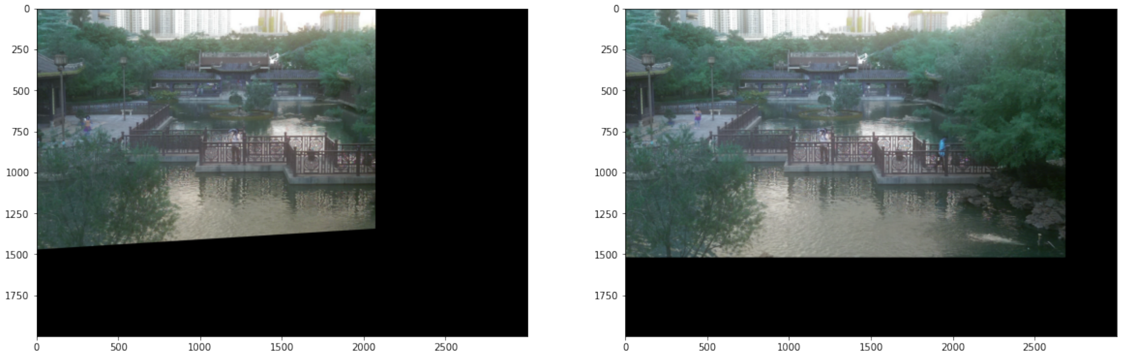
April 3, 2021

[5]:

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
from scipy.stats import norm
import scipy.io as sio
%matplotlib inline

window_name='stitch'

img1 = cv2.imread('IMAG4689.jpg')
img2 = cv2.imread('IMAG4688.jpg')

h = np.array([[ 1.55045419e-03,  3.02183837e-05, -9.78825638e-01],
       [ 2.78294686e-05,  1.46986982e-03, -2.04680411e-01],
       [ 8.37118544e-08,  1.54795992e-08,  1.31648165e-03]])
plt.figure(figsize=(20,20))
plt.subplot(1,2,1)
plt.imshow(cv2.warpPerspective(img1,h,(3000,2000)))
plt.subplot(1,2,2)
plt.imshow(cv2.warpPerspective(img2,np.eye(3),(3000,2000)))
```

[5]:

```
<matplotlib.image.AxesImage at 0x7fb86a14c490>
```



0.1 Q1a. (10 points) Create a stitching function that stitches img1 and img2 together by completing the following function

[6]:

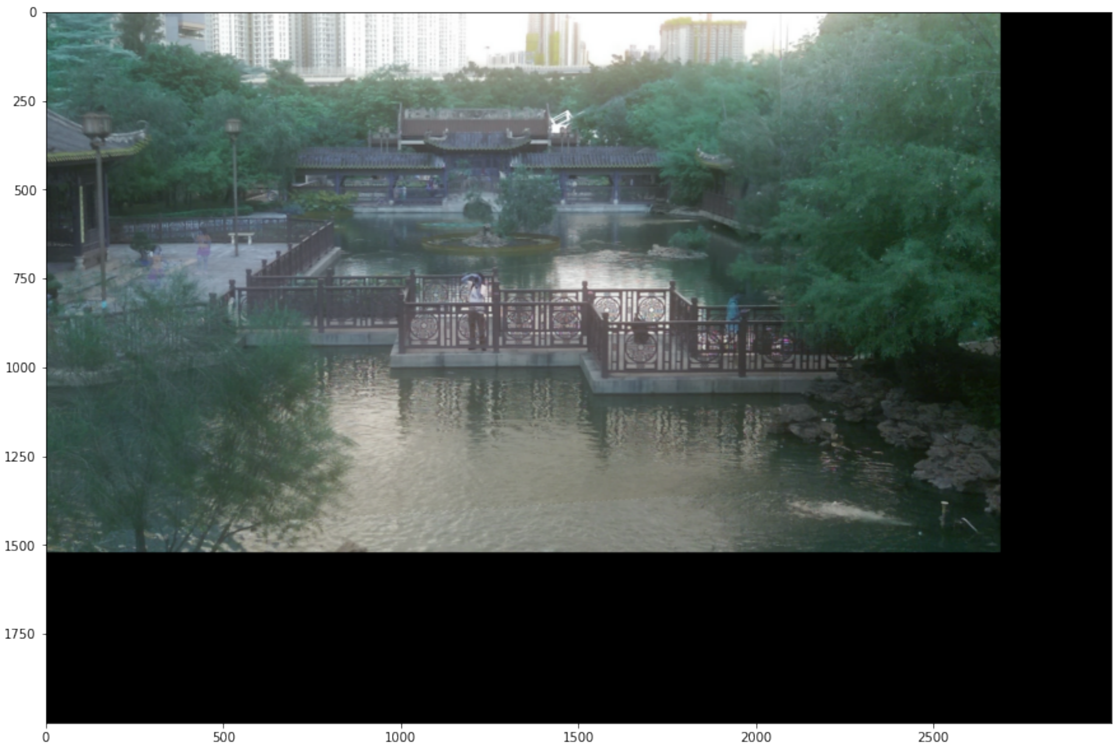
```
def stitch_images(img1,img2,h1,h2,fs):
# Input
#     img1: first image
#     img2: second image
#     h1: projective transform for img1
#     h2: projective transform for img2
#     fs: size of the output image
# Output
#     return img of size fs

    normal = cv2.warpPerspective(np.ones_like(img1),h1,fs)  + cv2.
 ↪warpPerspective(np.ones_like(img2),h2,fs)
    normal[normal == 0] = 1

    stitchImg = (np.float32(cv2.warpPerspective(img1,h1,fs))  + np.float32(cv2.
 ↪warpPerspective(img2,h2,fs)))/np.float32(normal)
    stitchImg = np.uint8(stitchImg)

    return stitchImg
```

[7]:

```
plt.figure(figsize=(15,15))
h1=h
h2=np.eye(3)
plt.imshow(stitch_images(img1,img2,h1,h2,(3000,2000)))
```

[7]:

```
<matplotlib.image.AxesImage at 0x7fb869813760>
```



You should get something like the above running the code. Note that stitching is not apparent because the first image is shifted up and left and being cropped off.

0.2 Q2. (10 points) Find homography and test on your *own images*

[36]:
```python
# implement these function
def myFindHomography(match_xy):
# Input
#     match_xy:
#         first two columns: (x,y)-values in the original image
#         second two columns: (x,y)-values in the target image
# # Output
#     h: return homography of transforming from original to target frame
    sourcePoints = match_xy[:,0:2]
    destPoints = match_xy[:,2:4]

    #make Alist to store the A matrix
    Alist = []

    #make the two rows of the A matrix
    for i in range(len(match_xy[:,0])):
        x1 = sourcePoints[i,0]
        y1 = sourcePoints[i,1]
        x2 = destPoints[i,0]
        y2 = destPoints[i,1]
        aRow1 = np.array([-x1,-y1,-1,0,0,0,x1*x2,y1*x2,x2])
        aRow2 = np.array([0,0,0,-x1,-y1,-1,x1*y2,y1*y2,y2])

        Alist.append(aRow1)
        Alist.append(aRow2)

    A = np.array(Alist)

    u, s, vh = np.linalg.svd(A) # is singular values,

    #want the min sing value and its vector
    s = list(s)
    minS = min(s, key=abs)
    minSIndex = s.index(minS)
    h = vh[minSIndex]

    H = np.reshape(h,(3,3))
    return H
```

[46]:

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
from scipy.stats import norm
import scipy.io as sio
%matplotlib inline

# PLEASE REPLACE WITH YOUR OWN IMAGES HERE
img1 = cv2.imread('File_000.jpg')
img2 = cv2.imread('File_001.jpg')

desc = cv2.xfeatures2d.SIFT_create()

ratio_thresh = 0.6

kps1, descs1 = desc.detectAndCompute(cv2.cvtColor(img1,cv2.COLOR_BGR2GRAY),␣
 ↪None)
kps2, descs2 = desc.detectAndCompute(cv2.cvtColor(img2,cv2.COLOR_BGR2GRAY),␣
 ↪None)
matcher = cv2.DescriptorMatcher_create(cv2.DescriptorMatcher_FLANNBASED)
knn_matches = matcher.knnMatch(descs1, descs2, 2)

good_matches = []
for m,n in knn_matches:
    if m.distance < ratio_thresh * n.distance:
        good_matches.append(m)
```

```
print(len(good_matches))
match_xy=np.array([[*(kps1[q.queryIdx].pt),*(kps2[q.trainIdx].pt)] for q in␣
 ↪good_matches])
# sio.savemat('match_xy',{'match_xy':match_xy}) # save index to matlab format

# match_xy is a matrix with each row equals x1,y1,x2,y2,
# where (x1,y1) and (x2,y2) are matched coordinates in img1 and img2,␣
 ↪respectively
```

```
872
```

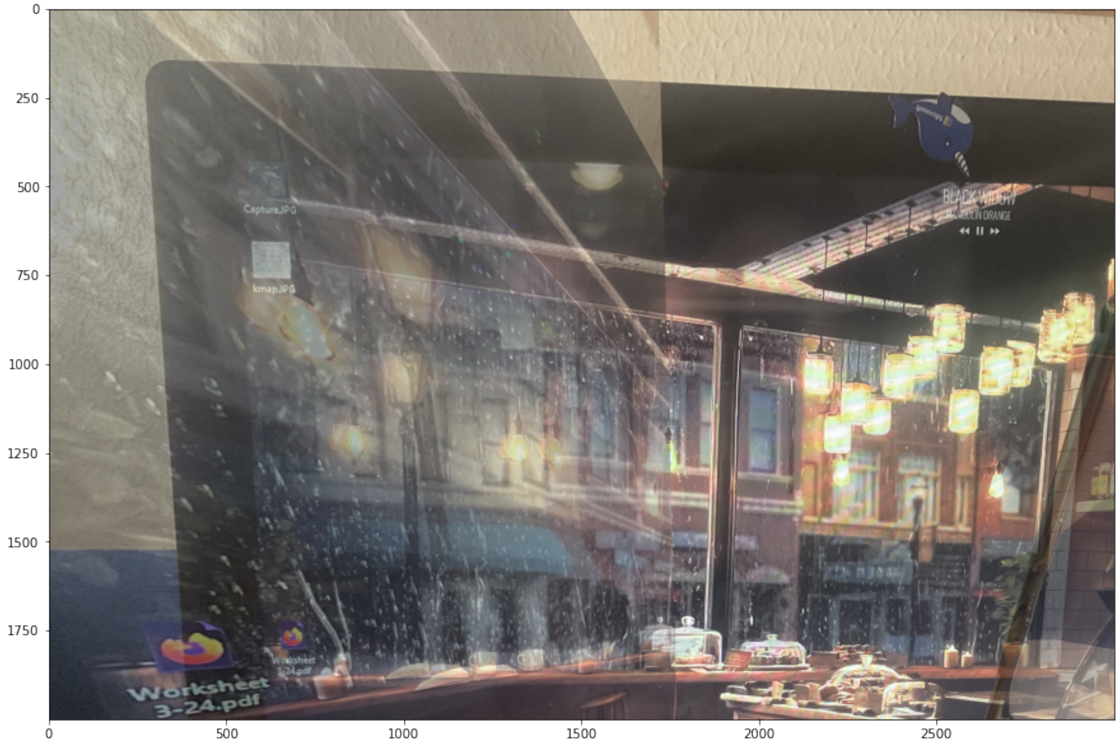
[47]:

```
# Test Q2a here
h1 = myFindHomography(match_xy)
h2 = np.eye(3)
plt.figure(figsize=(15,15))
stitched = stitch_images(img1,img2,h1,h2,(3000,2000))
rgbStitch = cv2.cvtColor(stitched, cv2.COLOR_BGR2RGB)
plt.imshow(rgbStitch)
```

```
0.8588756300263782
```

[47]:

```
<matplotlib.image.AxesImage at 0x7fb868cef370>
```


Capture.JPG
kmap.JPG
Worksheet 3-24.pdf
BLACK WIDOW