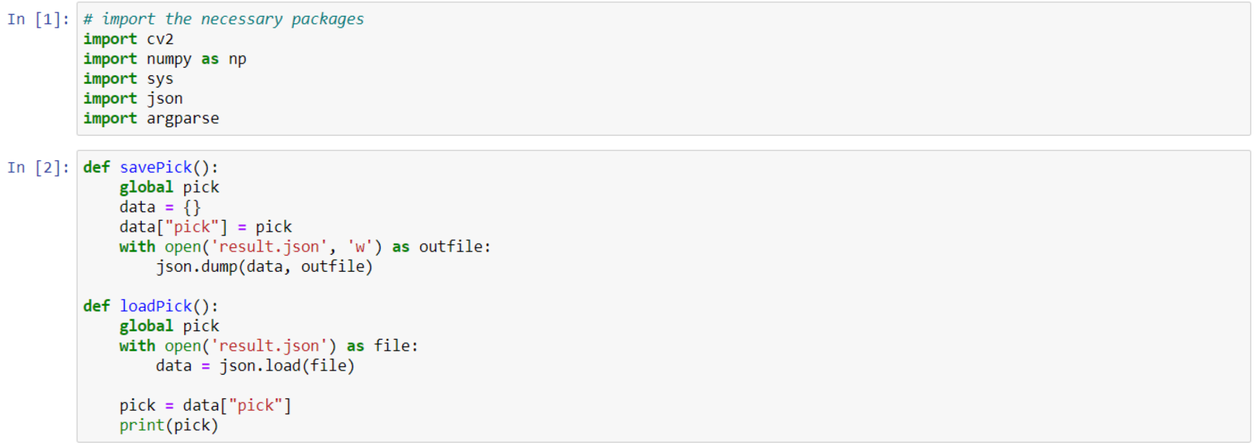
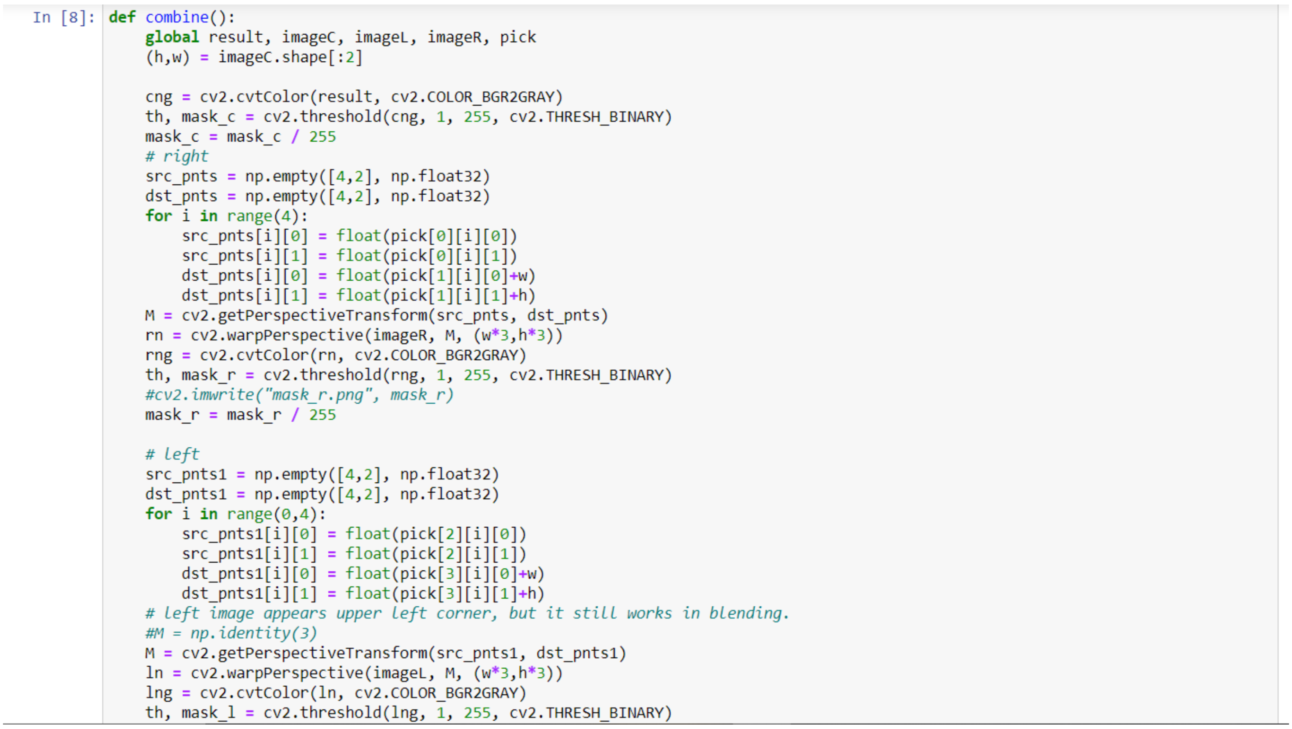
# Image Mosaicing with Bi-Linear Transformation

## CODE



```
In [1]: # import the necessary packages
        import cv2
        import numpy as np
        import sys
        import json
        import argparse
```

```
In [2]: def savePick():
            global pick
            data = {}
            data["pick"] = pick
            with open('result.json', 'w') as outfile:
                json.dump(data, outfile)

        def loadPick():
            global pick
            with open('result.json') as file:
                data = json.load(file)

            pick = data["pick"]
            print(pick)
```



```
In [8]: def combine():
            global result, imageC, imageL, imageR, pick
            (h,w) = imageC.shape[:2]

            cng = cv2.cvtColor(result, cv2.COLOR_BGR2GRAY)
            th, mask_c = cv2.threshold(cng, 1, 255, cv2.THRESH_BINARY)
            mask_c = mask_c / 255
            # right
            src_pnts = np.empty([4,2], np.float32)
            dst_pnts = np.empty([4,2], np.float32)
            for i in range(4):
                src_pnts[i][0] = float(pick[0][i][0])
                src_pnts[i][1] = float(pick[0][i][1])
                dst_pnts[i][0] = float(pick[1][i][0]+w)
                dst_pnts[i][1] = float(pick[1][i][1]+h)
            M = cv2.getPerspectiveTransform(src_pnts, dst_pnts)
            rn = cv2.warpPerspective(imageR, M, (w*3,h*3))
            rng = cv2.cvtColor(rn, cv2.COLOR_BGR2GRAY)
            th, mask_r = cv2.threshold(rng, 1, 255, cv2.THRESH_BINARY)
            #cv2.imwrite("mask_r.png", mask_r)
            mask_r = mask_r / 255

            # left
            src_pnts1 = np.empty([4,2], np.float32)
            dst_pnts1 = np.empty([4,2], np.float32)
            for i in range(0,4):
                src_pnts1[i][0] = float(pick[2][i][0])
                src_pnts1[i][1] = float(pick[2][i][1])
                dst_pnts1[i][0] = float(pick[3][i][0]+w)
                dst_pnts1[i][1] = float(pick[3][i][1]+h)
            # left image appears upper left corner, but it still works in blending.
            #M = np.identity(3)
            M = cv2.getPerspectiveTransform(src_pnts1, dst_pnts1)
            ln = cv2.warpPerspective(imageL, M, (w*3,h*3))
            lng = cv2.cvtColor(ln, cv2.COLOR_BGR2GRAY)
            th, mask_l = cv2.threshold(lng, 1, 255, cv2.THRESH_BINARY)
```

```
    mask_l = mask_l / 255
    #cv2.imwrite("mask_l.png", mask_l)

    # alpha blending
    # mask element: number of pictures at that coordinate
    mask = np.array(mask_c + mask_l + mask_r, float)

    # alpha blending weight
    ag = np.full(mask.shape, 0.0, dtype=float)
    # weight: 1.0 / (num of picture)
    ag = 1.0 / np.maximum(1,mask) # avoid 0 division

    # generate result image from 3 images + alpha weight
    result[:,:,0] = result[:,:,0]*ag[:,:] + ln[:,:,0]*ag[:,:] + rn[:,:,0]*ag[:,:]
    result[:,:,1] = result[:,:,1]*ag[:,:] + ln[:,:,1]*ag[:,:] + rn[:,:,1]*ag[:,:]
    result[:,:,2] = result[:,:,2]*ag[:,:] + ln[:,:,2]*ag[:,:] + rn[:,:,2]*ag[:,:]

    cv2.imwrite("result_wall.jpg", result)
    cv2.imshow("result", result)
```

In [9]:

```
'''
pick 4 points from right image (red point)
'''
def right_click(event, x, y, flags, param):
    if event == cv2.EVENT_LBUTTONUP:
        mousePick(x, y, 0)

'''
pick 4 points from center (correspond to right, red point)
'''
def center_click_r(event, x, y, flags, param):
    if event == cv2.EVENT_LBUTTONUP:
        mousePick(x, y, 1)

'''
pick 4 points from left (blue point)
'''
def left_click(event, x, y, flags, param):
    if event == cv2.EVENT_LBUTTONUP:
        # add your code to select 4 points
        mousePick(x, y, 2)

'''
pick 4 points from center (correspond to left, blue point)
'''
def center_click_l(event, x, y, flags, param):
    if event == cv2.EVENT_LBUTTONUP:
        # add your code to select 4 points
        mousePick(x, y, 3)
```

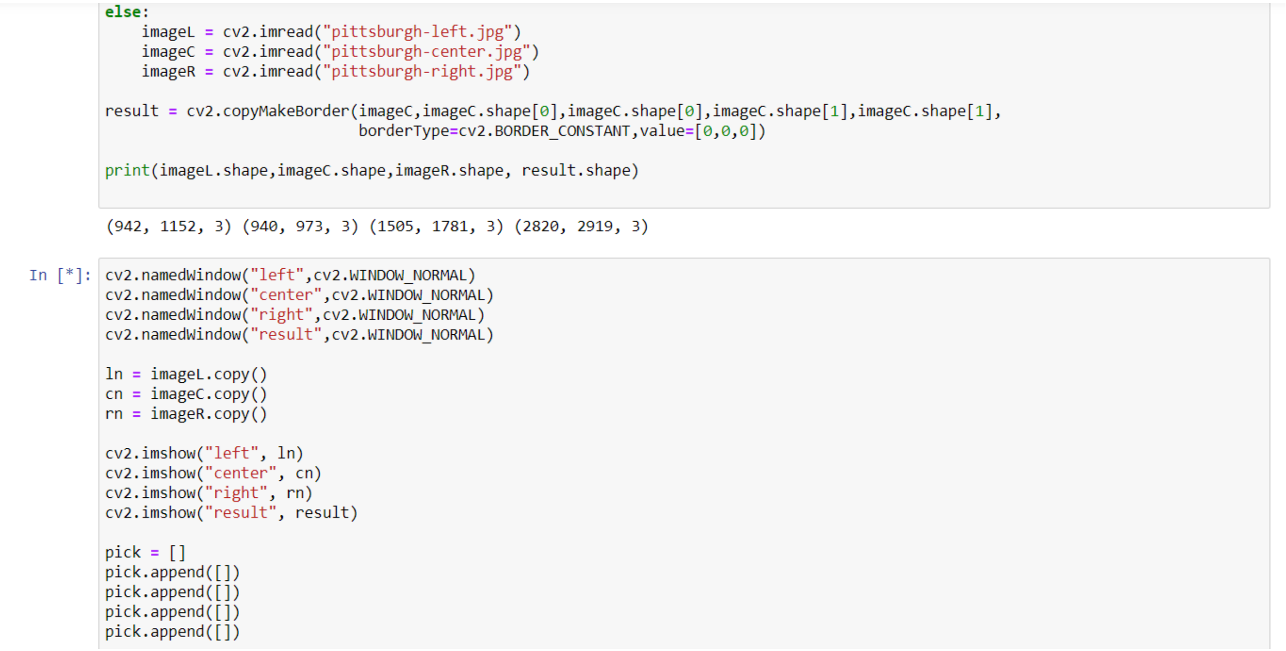
In [10]:

```
def mousePick(x, y, idx):
    global rn, cn, ln, imageR, imageC, imageL, pick
    if idx == 0:
        src = imageR
        dst = rn
        wn = "right"
    elif idx == 1:
        src = imageC
        dst = cn
        wn = "center"
    elif idx == 2:
        src = imageL
        dst = ln
        wn = "left"
    elif idx == 3:
        src = imageC
        dst = cn
        wn = "center"
```

```
# you need to add idx 2, 3 cases

#print(idx, x, y)
pick[idx].append((x,y))
dst = src.copy()
# red BGR color in OpenCV, you need to set to blue on left side
col = (0, 0, 255)
col1 = (255, 0, 0)
# place circle on the picked point and text its serial (0-3)
for i in range(len(pick[idx])):
    if (idx == 0 or idx ==1):
        dst = cv2.circle(dst, pick[idx][i], 5, col, 2)
        dst = cv2.putText(dst, str(i), (pick[idx][i][0]+10, pick[idx][i][1]-10),
                          cv2.FONT_HERSHEY_SIMPLEX,1, col, 1)
    elif (idx == 2 or idx ==3):
        dst = cv2.circle(dst, pick[idx][i], 5, col1, 2)
        dst = cv2.putText(dst, str(i), (pick[idx][i][0]+10, pick[idx][i][1]-10),
                          cv2.FONT_HERSHEY_SIMPLEX,1, col1, 1)
# please make sure when idx == 3, you need to show red color circle in dst
# this example erases red circle

cv2.imshow(wn, dst)
# to make sure image is updated
cv2.waitKey(1)
if len(pick[idx]) >= 4:
    print('Is it OK? (y/n)')
    i = input()
    if i == 'y' or i == 'Y':
        if idx >= 3:
            savePick()
            combine()
            #print('center 4 points')
            #cv2.setMouseCallback("center", center_click_l)
        elif idx == 0:
            print('center 4 points')
            cv2.setMouseCallback("center", center_click_r)
        elif idx == 1:
            # only taking care of right and center, you need to replace 2 lines to start
            # picking left and center correspondence
            # you need to add pick code
            print('left 4 points')
            cv2.setMouseCallback("left", left_click)
        elif idx == 2:
            print('center 4 points')
            cv2.setMouseCallback("center", center_click_l)
    else:
        pick[idx] = []
        dst = src.copy()
        cv2.imshow(wn, dst)
```

In [11]:

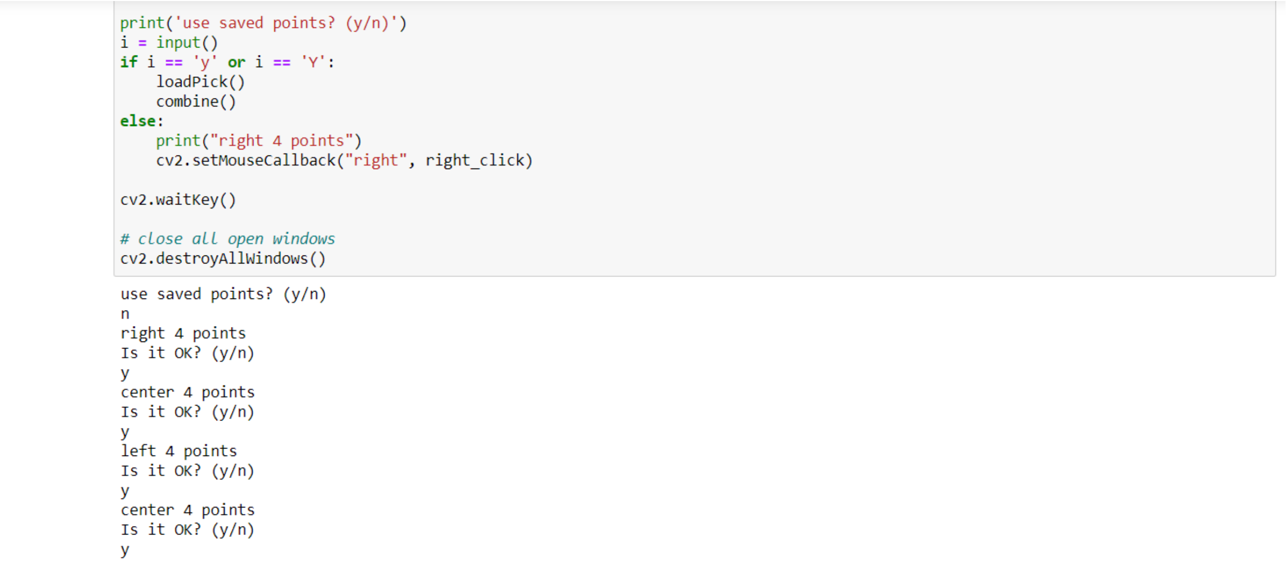
```
parser = argparse.ArgumentParser(description='Combine 3 images')
parser.add_argument('-d', '--data', type=int, help='Dataset index', default=0)
args, unknown = parser.parse_known_args()
dataset = args.data

if dataset == 0:
    imageL = cv2.imread("wall-left.png")
    imageC = cv2.imread("wall-center.png")
    imageR = cv2.imread("wall-right.png")
elif dataset == 1:
    imageL = cv2.imread("door-left.jpg")
    imageC = cv2.imread("door-center.jpg")
    imageR = cv2.imread("door-right.jpg")
elif dataset == 2:
    imageL = cv2.imread("house-left.jpg")
    imageC = cv2.imread("house-center.jpg")
    imageR = cv2.imread("house-right.jpg")
```



```
else:
    imageL = cv2.imread("pittsburgh-left.jpg")
    imageC = cv2.imread("pittsburgh-center.jpg")
    imageR = cv2.imread("pittsburgh-right.jpg")

result = cv2.copyMakeBorder(imageC,imageC.shape[0],imageC.shape[0],imageC.shape[1],imageC.shape[1],
                            borderType=cv2.BORDER_CONSTANT,value=[0,0,0])

print(imageL.shape,imageC.shape,imageR.shape, result.shape)
```

```
(942, 1152, 3) (940, 973, 3) (1505, 1781, 3) (2820, 2919, 3)
```

```
In [*]: cv2.namedWindow("left",cv2.WINDOW_NORMAL)
        cv2.namedWindow("center",cv2.WINDOW_NORMAL)
        cv2.namedWindow("right",cv2.WINDOW_NORMAL)
        cv2.namedWindow("result",cv2.WINDOW_NORMAL)

        ln = imageL.copy()
        cn = imageC.copy()
        rn = imageR.copy()

        cv2.imshow("left", ln)
        cv2.imshow("center", cn)
        cv2.imshow("right", rn)
        cv2.imshow("result", result)

        pick = []
        pick.append([])
        pick.append([])
        pick.append([])
        pick.append([])
```



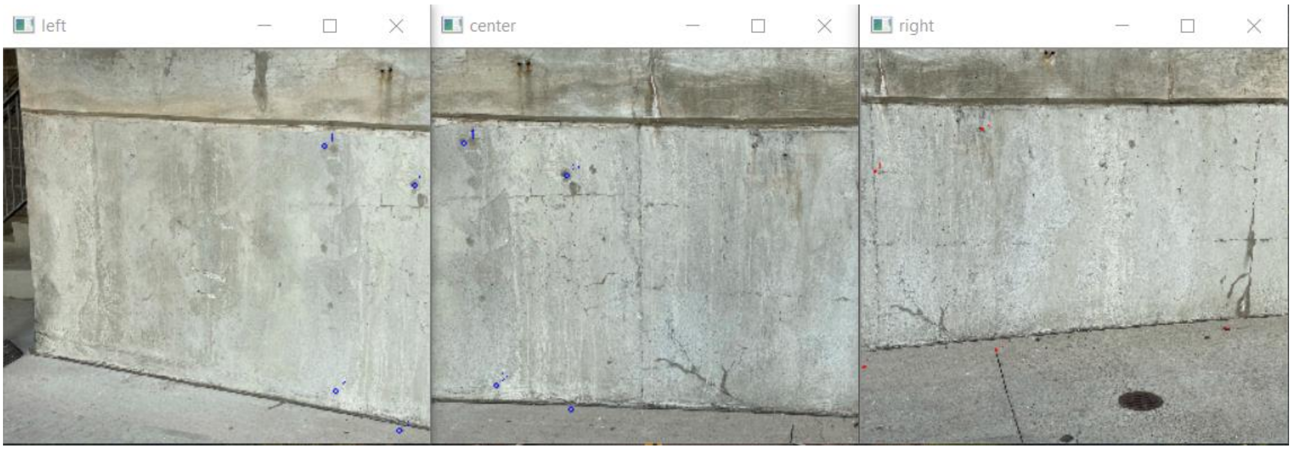
```
print('use saved points? (y/n)')
i = input()
if i == 'y' or i == 'Y':
    loadPick()
    combine()
else:
    print("right 4 points")
    cv2.setMouseCallback("right", right_click)

cv2.waitKey()

# close all open windows
cv2.destroyAllWindows()
```

```
use saved points? (y/n)
n
right 4 points
Is it OK? (y/n)
y
center 4 points
Is it OK? (y/n)
y
left 4 points
Is it OK? (y/n)
y
center 4 points
Is it OK? (y/n)
y
```

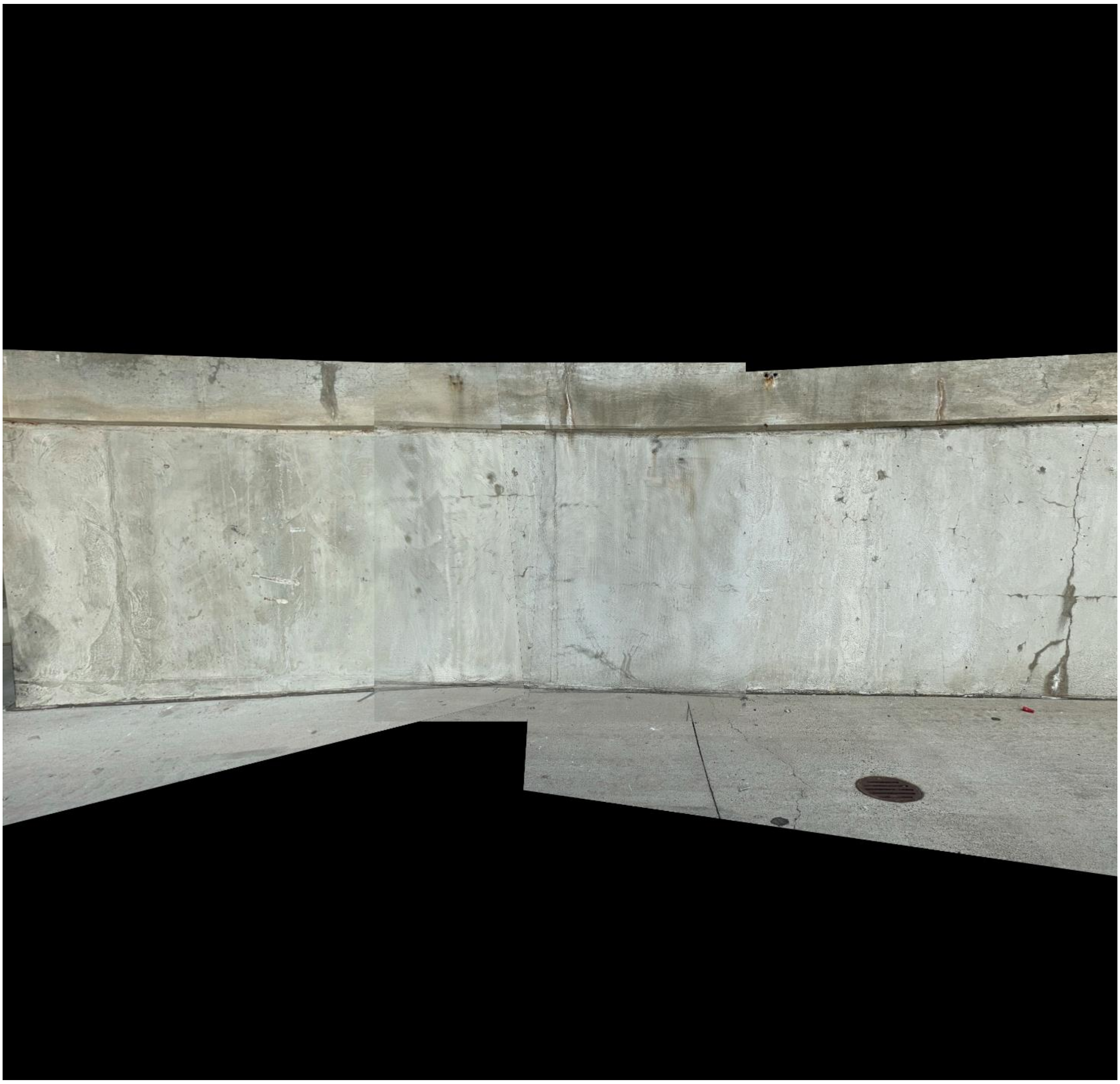
---

OUTPUT:

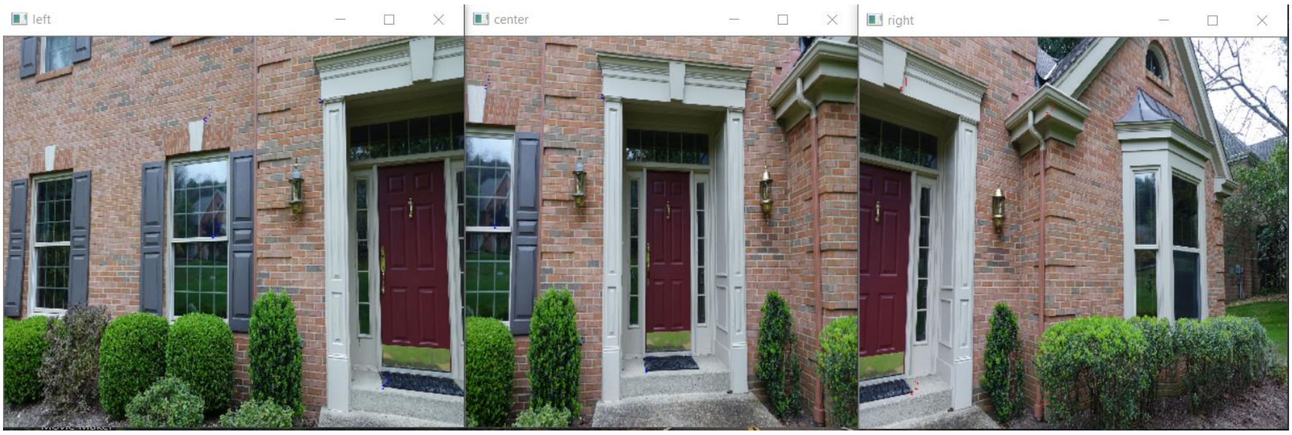
WALL Point Selection:



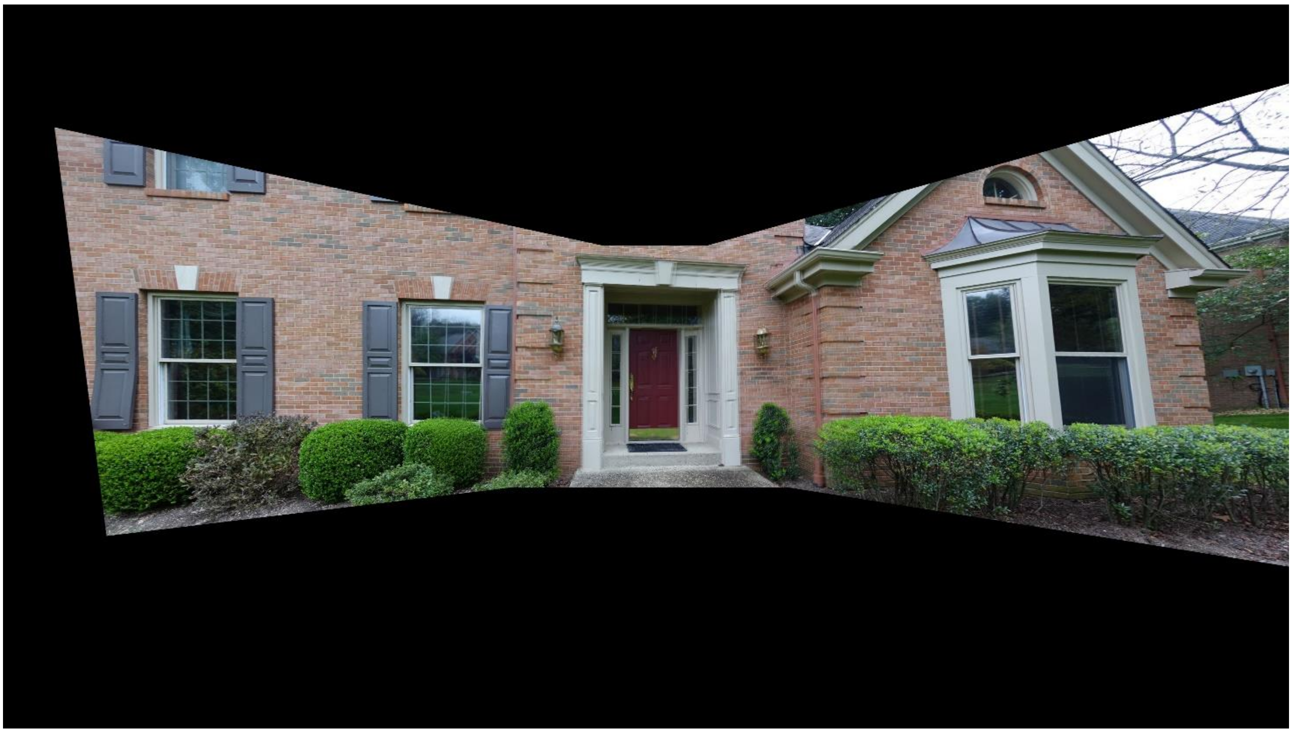
Wall Output Image:



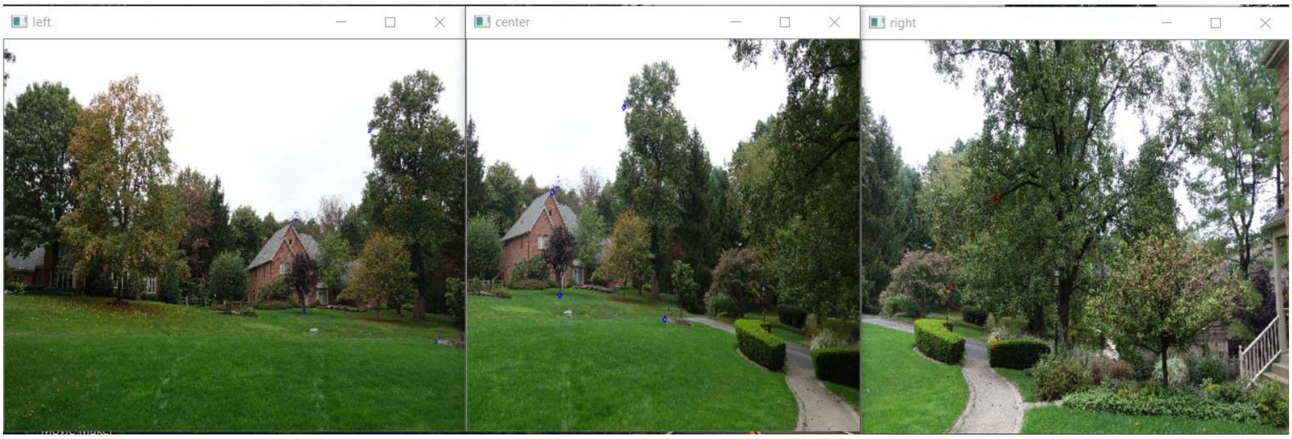
## DOOR Point Selection:



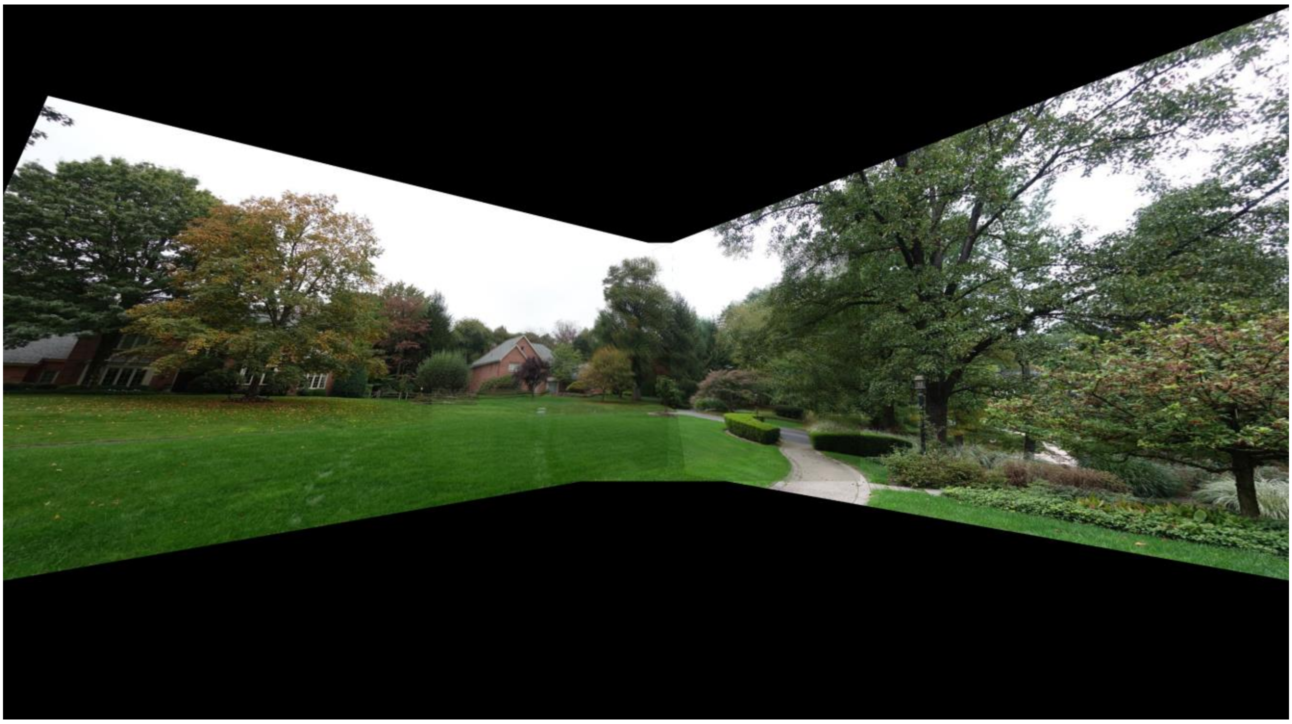
## Door Output:



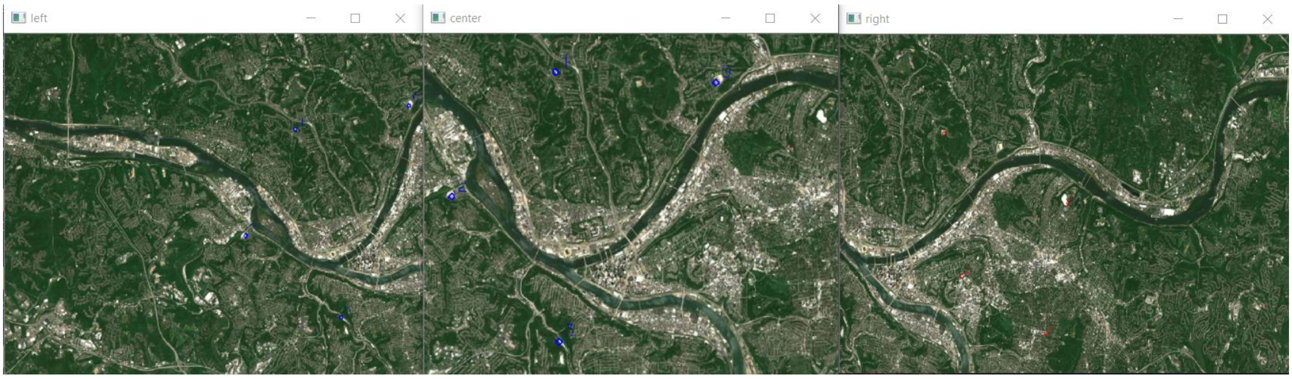
HOUSE Points Selection:



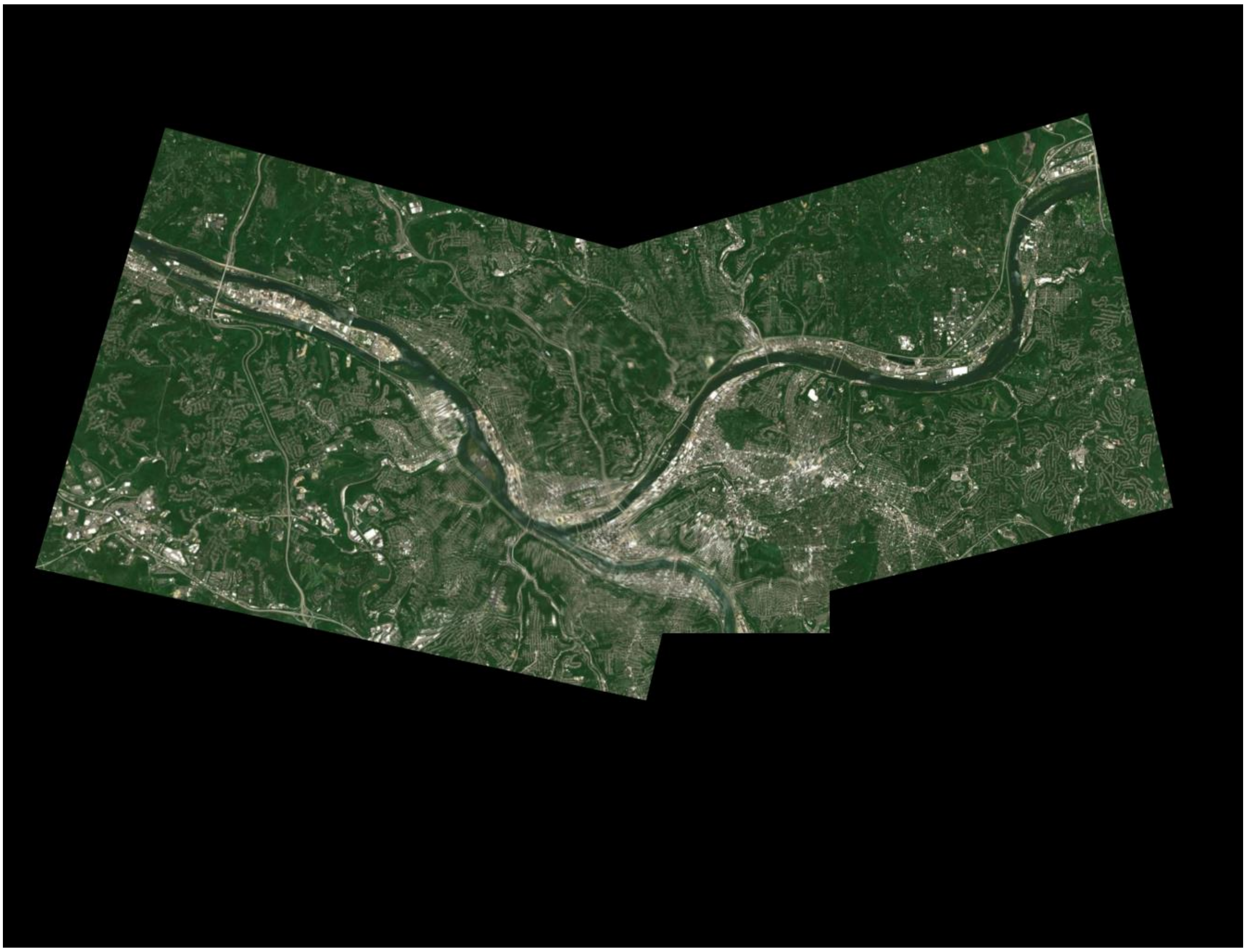
House OUTPUT:



PITTSBURGH Points Selection:



PITTSBURGH Output:



OPERATING SYSTEM: Windows 10

IDE Used: Jupyter Notebook

Number of Hours Spent: 5 Hours