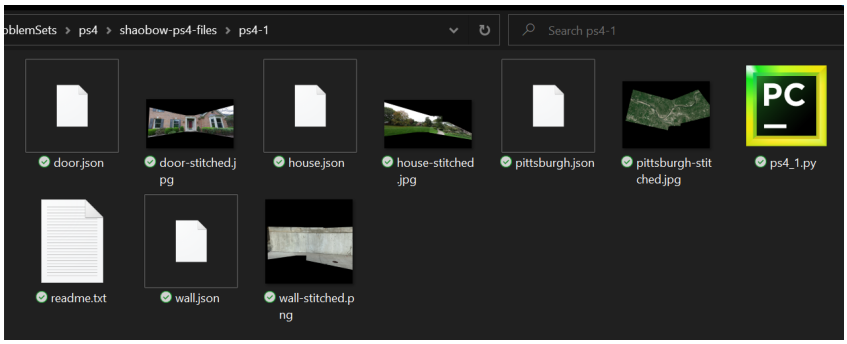
PS4

Shaobo Wang

PS4-1 Image Mosaicing with Bi-linear Transformation

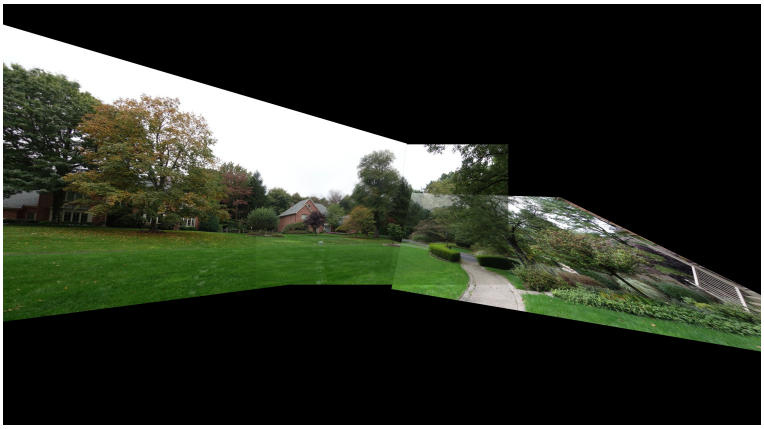
Screenshots of all the files in the “ps4-1” folder:

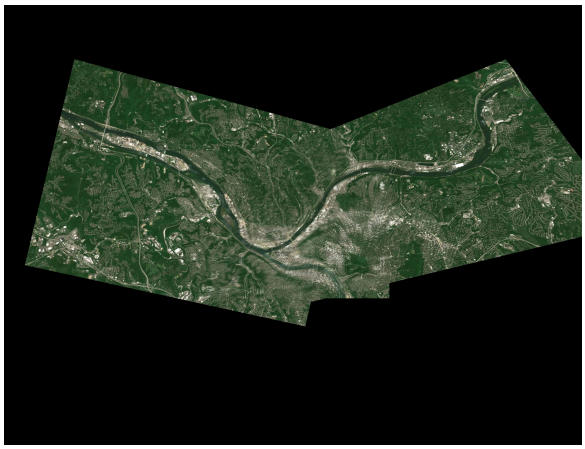


Terminal commands:

```
>> python ps4_1.py --d 0
>> python ps4_1.py --d 1
>> python ps4_1.py --d 2
>> python ps4_1.py --d 3
```

Output images:





Source code:

Operating system: Windows 10
IDE: PyCharm

```
# import the necessary packages
import cv2
import numpy as np
import sys
import json
import argparse


# insert string before
def insert_name(name, str2add):
    dot_idx = name.find(".")
    new_name = name[:dot_idx] + str2add + name[dot_idx:]
    return new_name


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


def loadPick():
    global pick, pt_file
    with open(pt_file) as file:
        data = json.load(file)
```

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


def combine():
    global result, imageC, imageL, imageR, pick, img_name
    (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_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[2][i][0])
        src_pnts[i][1] = float(pick[2][i][1])
        dst_pnts[i][0] = float(pick[3][i][0] + w)
        dst_pnts[i][1] = float(pick[3][i][1] + h)
    M = cv2.getPerspectiveTransform(src_pnts, dst_pnts)
    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[:, :]

    img_output_name = insert_name(img_name, "-stitched")
    cv2.imwrite(img_output_name, result)
    cv2.imshow("result", result)


'''
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)


'''
idea: handle mouse pick
idx
0: right
1: center (correspond to right)
2: left
3: center (correspond to left)
```

```
you can also create your own function for left + center selection
'''



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"

    # 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)
    # place circle on the picked point and text its serial (0-3)
    for i in range(len(pick[idx])):
        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)
    # 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()
            elif idx == 0:
                print('center 4 points')
                cv2.setMouseCallback("center", center_click_r)
            elif idx == 1:
                print('left 4 points')
                cv2.setMouseCallback("left", left_click)
            elif idx == 2:
                print('center 4 points')
                cv2.setMouseCallback("center", center_click_l)
                # 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

        else:
            pick[idx] = []
            dst = src.copy()
            cv2.imshow(wn, dst)


parser = argparse.ArgumentParser(description='Combine 3 images')
parser.add_argument('-d', '--data', type=int, help='Dataset index', default=3)
args = parser.parse_args()
dataset = args.data
global img_name, pt_file
if dataset == 0:
    imageL = cv2.imread("wall-left.png")
    imageC = cv2.imread("wall-center.png")
    imageR = cv2.imread("wall-right.png")
    img_name = "wall.png"
    pt_file = "wall.json"
elif dataset == 1:
    imageL = cv2.imread("door-left.jpg")
    imageC = cv2.imread("door-center.jpg")
    imageR = cv2.imread("door-right.jpg")
    img_name = "door.jpg"
    pt_file = "door.json"
elif dataset == 2:
    imageL = cv2.imread("house-left.jpg")
    imageC = cv2.imread("house-center.jpg")
    imageR = cv2.imread("house-right.jpg")
    img_name = "house.jpg"
    pt_file = "house.json"
else:
    imageL = cv2.imread("pittsburgh-left.jpg")
    imageC = cv2.imread("pittsburgh-center.jpg")
    imageR = cv2.imread("pittsburgh-right.jpg")
    img_name = "pittsburgh.jpg"
    pt_file = "pittsburgh.json"

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)

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()
```