

# **PS4-REPORT**

**Taking 1 late day**

## **INDEX**

- ❖ Algorithm
- ❖ Screenshot of the code
- ❖ Screenshot of readme.txt
- ❖ Output
- ❖ Points chosen: .json file

**Submitted by:**  
Akshay Antony  
PS-4  
CMU

## ALGORITHM:

1. Reading the 3 images: image-center, image-left, image-right
2. Creating an image called **result**, with size 3\*w and 3\*h to hold the stitched image
3. Select if you want to use the selected points from the .json file or select new images
  - a. If using the saved points load the points from the .json file
  - b. If selecting new points on the images call the function: *mousePick()*
  - c. After the 16 points have been selected (4 right, 4 center-right, 4-left, 4 center-left), the member function *combine()* is called.
4. The weights are saved to the .json file
5. The *combine()* function finds the transformation matrix  $M_r$  and  $M_l$ . Then uses those transformation matrices to transform the left image and right image.
6. Calculate the three masks( $m_r, m_l, m$ ) using the three images
7. Calculate the coefficients of the alpha blending matrix
8. Multiplying the **result** matrix with the alpha blending matrix
9. Displaying and saving the image

## CODE Screenshots:

```
example.py  wall_result.json  house_result.json  pittsburgh_result.json  door_result.json
1  # import the necessary packages
2  import cv2
3  import numpy as np
4  import sys
5  import json
6  import argparse
7
8  def savePick():
9      global pick, image_name
10     data = {}
11     data["pick"] = pick
12     filename = image_name + "_result.json"
13     with open(filename, 'w') as outfile:
14         json.dump(data, outfile)
15
16  def loadPick():
17      global pick, image_name
18      filename = image_name + "_result.json"
19      with open(filename) as file:
20         data = json.load(file)
21
22         pick = data["pick"]
23         print("The selected points for transformation\n",pick)
24         return pick
25
26  def combine():
27      global result, imageC, imageL, imageR, pick, temp, image_name
28      (h,w) = imageC.shape[:2]
29
30      cng = cv2.cvtColor(result, cv2.COLOR_BGR2GRAY)
31      th, mask_c = cv2.threshold(cng, 1, 255, cv2.THRESH_BINARY)
32      mask_c = mask_c / 255
33
34      # right
35      src_pts = np.empty([4,2], np.float32)
36      dst_pts = np.empty([4,2], np.float32)
37      source_points_l = np.empty([4,2], dtype=np.float32)
38      dst_points_l = np.empty([4,2], dtype=np.float32)
39
40      for i in range(4):
41         src_pts[i][0] = float(pick[0][i][0])
42         src_pts[i][1] = float(pick[0][i][1])
43         dst_pts[i][0] = float(pick[1][i][0]+w)
44         dst_pts[i][1] = float(pick[1][i][1]+h)
45         source_points_l[i,0] = np.float32(pick[2][i][0])
46         source_points_l[i,1] = np.float32(pick[2][i][1])
```

kite: Ready (unindexed), Line 12, Column 30

```

43     dst_pnts[i][0] = float(pick[1][i][0]+w)
44     dst_pnts[i][1] = float(pick[1][i][1]+h)
45     source_points_l[i,0] = np.float32(pick[2][i][0])
46     source_points_l[i,1] = np.float32(pick[2][i][1])
47     dst_pnts_l[i,0] = np.float32(pick[3][i][0]+w)
48     dst_pnts_l[i,1] = np.float32(pick[3][i][1]+h)
49
50     M = cv2.getPerspectiveTransform(src_pnts, dst_pnts)
51     rn = cv2.warpPerspective(imageR, M, (w*3,h*3))
52     rng = cv2.cvtColor(rn, cv2.COLOR_BGR2GRAY)
53     th, mask_r = cv2.threshold(rng, 1, 255, cv2.THRESH_BINARY)
54     #cv2.imwrite("mask_r.png", mask_r)
55     mask_r = mask_r / 255
56
57     # left image appears upper left corner, but it still works in blending.
58     M = cv2.getPerspectiveTransform(source_points_l, dst_points_l)
59     ln = cv2.warpPerspective(imageL, M, (w*3,h*3))
60     lng = cv2.cvtColor(ln, cv2.COLOR_BGR2GRAY)
61     th, mask_l = cv2.threshold(lng, 1, 255, cv2.THRESH_BINARY)
62     mask_l = mask_l / 255
63     #cv2.imwrite("mask_l.png", mask_l)
64     # alpha blending
65     # mask element: number of pictures at that coordinate
66     mask = np.array(mask_c + mask_l + mask_r, float)
67
68     # alpha blending weight
69     ag = np.full(mask.shape, 0.0, dtype=float)
70     # weight: 1.0 / (num of picture)
71     ag = 1.0 / np.maximum(1, mask) # avoid 0 division
72
73     # generate result image from 3 images + alpha weight
74     result[:, :, 0] = result[:, :, 0]*ag[:, :] + ln[:, :, 0]*ag[:, :] + rn[:, :, 0]*ag[:, :]
75     result[:, :, 1] = result[:, :, 1]*ag[:, :] + ln[:, :, 1]*ag[:, :] + rn[:, :, 1]*ag[:, :]
76     result[:, :, 2] = result[:, :, 2]*ag[:, :] + ln[:, :, 2]*ag[:, :] + rn[:, :, 2]*ag[:, :]
77     filename = image_name + "-stitched.jpg"
78     if cv2.imwrite(filename, result):
79         print("image written successfully")
80     cv2.imshow("result", result)
81
82     '''
83     pick 4 points from right image (red point)
84     '''
85     def right_click(event, x, y, flags, param):
86         if event == cv2.EVENT_LBUTTONDOWN:
87             mousePick(x, y, 0)
88

```

```
85 def right_click(event, x, y, flags, param):
86     if event == cv2.EVENT_LBUTTONUP:
87         mousePick(x, y, 0)
88
89     '''
90     pick 4 points from center (correspond to right, red point)
91     '''
92 def center_click_r(event, x, y, flags, param):
93     if event == cv2.EVENT_LBUTTONUP:
94         mousePick(x, y, 1)
95
96     '''
97     pick 4 points from left (blue point)
98     '''
99 def left_click(event, x, y, flags, param):
100     if event == cv2.EVENT_LBUTTONUP:
101         # add your code to select 4 points
102         mousePick(x, y, 2)
103
104     '''
105     pick 4 points from center (correspond to left, blue point)
106     '''
107 def center_click_l(event, x, y, flags, param):
108     if event == cv2.EVENT_LBUTTONUP:
109         # add your code to select 4 points
110         mousePick(x, y, 3)
111
112     '''
113     idea: handle mouse pick
114     idx
115     0: right
116     1: center (correspond to right)
117     2: left
118     3: center (correspond to left)
119
120     you can also create your own function for left + center selection
121     '''
122 def mousePick(x, y, idx):
123     global rn, cn, ln, imageR, imageC, imageL, pick, image_name
124     if idx == 0:
125         src = imageR
126         dst = rn
127         wn = "right"
128     elif idx == 1:
129         src = imageC
130         dst = cn
```

```
130     dst = cn
131     wn = "center"
132 elif idx == 2:
133     src = imageL
134     dst = ln
135     wn = "left"
136 elif idx == 3:
137     src = imageC
138     dst = cn
139     wn = "center"
140 # you need to add idx 2, 3 cases
141
142 #print(idx, x, y)
143 pick[idx].append((x,y))
144 dst = src.copy()
145 # red BGR color in OpenCV, you need to set to blue on left side
146 if idx == 3:
147     col = (255,0,0)
148 else:
149     col = (0, 0, 255)
150 # place circle on the picked point and text its serial (0-3)
151 for i in range(len(pick[idx])):
152     dst = cv2.circle(dst, pick[idx][i], 5, col, 2)
153     dst = cv2.putText(dst, str(i), (pick[idx][i][0]+10, pick[idx][i][1]-10),
154                       cv2.FONT_HERSHEY_SIMPLEX, 1, col, 1)
155 # please make sure when idx == 3, you need to show red color circle in dst
156 # this example erases red circle
157
158 cv2.imshow(wn, dst)
159 # to make sure image is updated
160 cv2.waitKey(1)
161 if len(pick[idx]) >= 4:
162     print('Is it OK? (y/n)')
163     i = input()
164     if i == 'y' or i == 'Y':
165         if idx >= 3:
166             savePick()
167             combine()
168         elif idx == 0:
169             print('center 4 points')
170             cv2.setMouseCallback("center", center_click_r)
171         elif idx == 1:
172             # only taking care of right and center, you need to replace 2 lines to start
173             # picking left and center correspondence
174             print("\nplease select 4 points on the left image")
175             cv2.setMouseCallback("left", left_click)
```

```
example.py wall_result.json house_result.json pittsburgh_result.json door_result.json
172         # only taking care of right and center, you need to replace 2 lines to start
173         # picking left and center correspondence
174         print("\nplease select 4 points on the left image")
175         cv2.setMouseCallback("left", left_click)
176         # you need to add pick code
177         # print('left 4 points')
178         # cv2.setMouseCallback("left", left_click)
179     elif idx == 2:
180         print("\nplease select 4 points on the center image")
181         cv2.setMouseCallback("center", center_click_l)
182     else:
183         pick[idx] = []
184         dst = src.copy()
185         cv2.imshow(wn, dst)
186
187
188     parser = argparse.ArgumentParser(description='Combine 3 images')
189     parser.add_argument('-d', '--data', type=int, help='Dataset index', default=1)
190     args = parser.parse_args()
191     dataset = args.data
192     temp = dataset
193     if dataset == 0:
194         imageL = cv2.imread("ps4-images/wall-left.png")
195         imageC = cv2.imread("ps4-images/wall-center.png")
196         imageR = cv2.imread("ps4-images/wall-right.png")
197         image_name = "wall"
198
199     elif dataset == 1:
200         imageL = cv2.imread("ps4-images/door-left.jpg")
201         imageC = cv2.imread("ps4-images/door-center.jpg")
202         imageR = cv2.imread("ps4-images/door-right.jpg")
203         image_name = "door"
204
205     elif dataset == 2:
206         imageL = cv2.imread("ps4-images/house-left.jpg")
207         imageC = cv2.imread("ps4-images/house-center.jpg")
208         imageR = cv2.imread("ps4-images/house-right.jpg")
209         image_name = "house"
210
211     else:
212         imageL = cv2.imread("ps4-images/pittsburgh-left.jpg")
213         imageC = cv2.imread("ps4-images/pittsburgh-center.jpg")
214         imageR = cv2.imread("ps4-images/pittsburgh-right.jpg")
215         image_name = "pittsburgh"
216
217     result = cv2.copyMakeBorder(imageC, imageC.shape[0], imageC.shape[0], imageC.shape[1], imageC.shape[1],
```

```
example.py  wall_result.json  house_result.json  pittsburgh_result.json  door_result.json
217 result = cv2.copyMakeBorder(imageC,imageC.shape[0],imageC.shape[0],imageC.shape[1],imageC.shape[1],
218                               borderType=cv2.BORDER_CONSTANT,value=[0,0,0])
219
220 #print(imageL.shape,imageC.shape,imageR.shape, result.shape)
221
222 cv2.namedWindow("left",cv2.WINDOW_NORMAL)
223 cv2.namedWindow("center",cv2.WINDOW_NORMAL)
224 cv2.namedWindow("right",cv2.WINDOW_NORMAL)
225 cv2.namedWindow("result",cv2.WINDOW_NORMAL)
226
227 ln = imageL.copy()
228 cn = imageC.copy()
229 rn = imageR.copy()
230
231 cv2.imshow("left", ln)
232 cv2.imshow("center", cn)
233 cv2.imshow("right", rn)
234 cv2.imshow("result", result)
235
236 pick = []
237 pick.append([])
238 pick.append([])
239 pick.append([])
240 pick.append([])
241
242 print('use saved points? (y/n)')
243 i = input()
244 if i == 'y' or i == 'Y':
245     loadPick()
246     combine()
247 else:
248     print("right 4 points")
249     cv2.setMouseCallback("right", right_click)
250
251 cv2.waitKey(0)
252
253 # close all open windows
254 cv2.destroyAllWindows()
255
256
257
258
259
```



## README.TXT

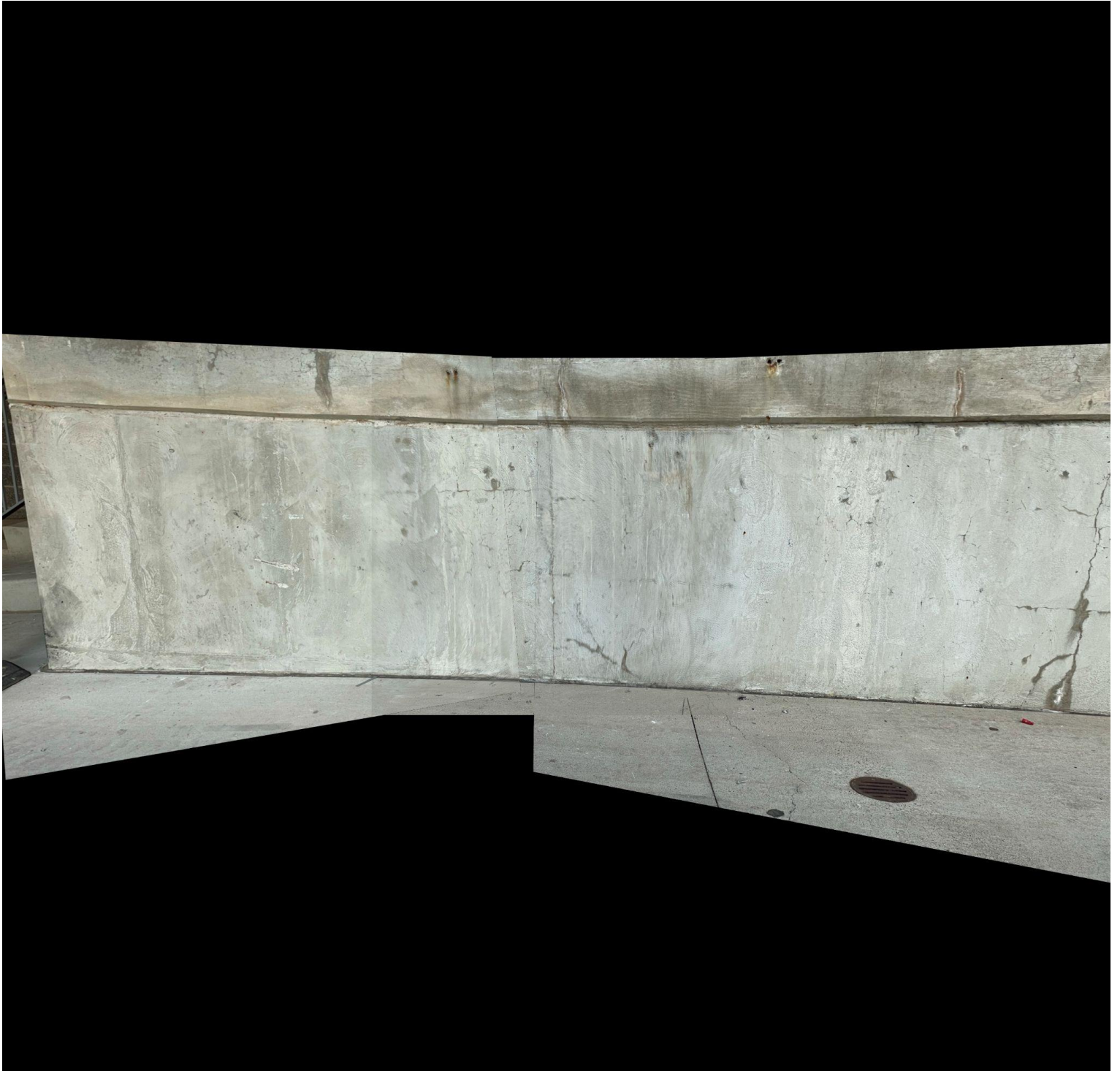
```
1 Python version: Python 3.9.6
2 OpenCV Version: 4.5.2
3 Operating System: Linux 20.02
4 IDE: Sublime text, run via terminal
5 Almost spend 6 hours for this problem
```

**OUTPUTS**

- 1. pittsburgh-stitched.png**



2. wall-stitched.png





3. house-stitched.jpg



4. door-stitched.jpg



# POINTS CHOSEN: .json FILE

## 1. pittsburgh.jpg

```
File Edit Selection Find View Goto Tools Project Preferences Help
example.py wall_result.json x house_result.json x pittsburgh_result.json door_result.json
1 {"pick": [[[764, 338], [730, 1010], [320, 951], [239, 226]], [[669, 46], [766, 505], [479, 540], [295, 64]],
2 [[842, 251], [876, 793], [1116, 724], [1168, 186]], [[255, 66], [162, 570], [396, 563], [568, 81]]]}
```

## 2. wall.jpg

```
example.py wall_result.json house_result.json x pittsburgh_result.json x door_result.json
1 {"pick": [[[701, 397], [730, 1114], [75, 1149], [57, 477]], [[971, 328], [970, 897], [479, 855], [474, 374]],
2 [[879, 242], [767, 841], [1118, 899], [1123, 243]], [[79, 223], [14, 837], [359, 842], [310, 222]]]}
```

## 3. house.jpg

```
File Edit Selection Find View Goto Tools Project Preferences Help
example.py wall_result.json x house_result.json pittsburgh_result.json x door_result.json
1 {"pick": [[[306, 569], [380, 790], [579, 842], [658, 375]], [[973, 406], [1060, 574], [1242, 626], [1261, 209]],
2 [[1554, 775], [1584, 512], [1193, 506], [1057, 761]], [[555, 520], [590, 330], [301, 298], [158, 512]]]}
```

## 4. door.jpg

```
File Edit Selection Find View Goto Tools Project Preferences Help
example.py wall_result.json x house_result.json x pittsburgh_result.json door_result.json
1 {"pick": [[[235, 374], [199, 924], [793, 1051], [812, 370]], [[1099, 376], [1103, 866], [1724, 1074], [1690, 302]],
2 [[939, 327], [939, 812], [1580, 910], [1559, 362]], [[245, 267], [216, 825], [887, 877], [884, 371]]]}
```