24-678: Computer Vision for Engineers
Ryan Wu
ID: weihuanw
PS4 Report
Due: Oct 6 2023

This file contains the following:
PS4-1 Image Mosaicing with Bi-linear Transformation

- pittsburgh-stitched.png
- wall-stitched.png
- house-stitched.png
- door-stitched.png
- readme.txt
- source code file(s) (attached to the end)

Findings and discussion:

We are tasked to stitch 3 given images into one combined image using mosaicking and bi-linear transformation. The functioning demo code was given but some minor edits were made to suit our project's needs.

In my program, the user is asked to select 4 points from the right image, 4 points from the left image, and 8 points from the center image to declare the image stitching criteria. The script generates a combined image using the user-selected points. The resulting output image may vary from user-defined points, but the overall rendering is sufficient for our use case.

PS4-1 Pittsburgh stitched image

Figure 1. The given Pittsburgh images (left, center, right).

Figure 2. The stitched Pittsburgh image.

Right picked points' coordinates: **[860, 513], [596, 446], [528, 700], [698, 835]**
Center-right picked points' coordinates: **[764, 146], [576, 148], [575, 332], [714, 392]**
Left picked points' coordinates: **[854, 332], [795, 673], [682, 528], [728, 268]**
Center-left picked points' coordinates: **[247, 145], [115, 443], [48, 286], [148, 56]**

PS4-1 Wall stitched image

Figure 3. The given wall images (left, center, right).

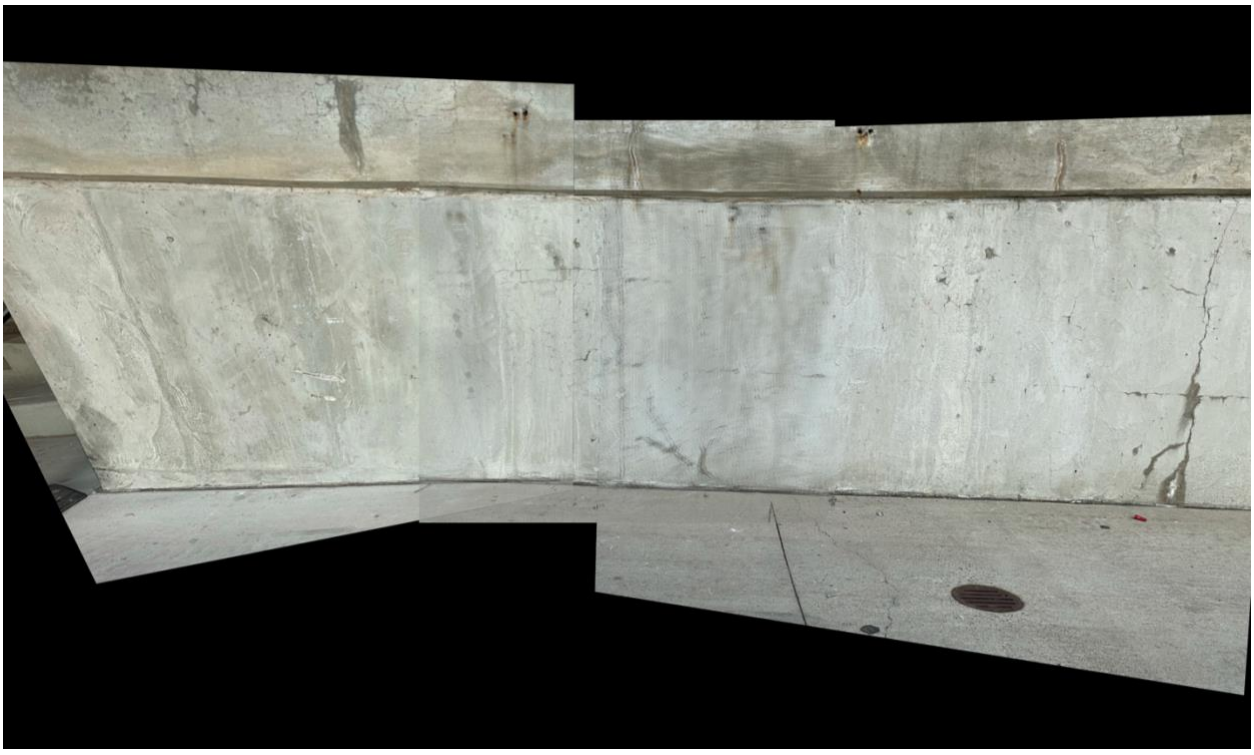

Figure 4. The stitched wall image.

Right image picked points' coordinates: **[514, 231], [559, 1105], [350, 1122], [394, 231]**
Center-right image picked points' coordinates: **[814, 199], [825, 867], [666, 862], [721, 199]**
Left image picked points' coordinates: **[899, 200], [984, 881], [899, 868], [821, 196]**
Center-left image picked points' coordinates: **[101, 180], [226, 843], [154, 846], [13, 177]**

PS4-1 House stitched image

Figure 5. The given house images (left, center, right).

Figure 6. The stitched house image.

Right picked points' coordinates: **[740, 392], [701, 816], [568, 825], [597, 441]**
Center-right picked points' coordinates: **[1366, 199], [1358, 606], [1237, 612], [1215, 269]**
Left picked points' coordinates: **[1082, 505], [1057, 765], [983, 759], [991, 463]**
Center-left picked points' coordinates: **[201, 291], [160, 514], [84, 509], [115, 245]**

PS4-1 Door stitched image

Figure 7. The given door images (left, center, right)

Figure 8. The stitched door image.

Right picked points' coordinates: **[530, 243], [504, 1013], [103, 1069], [148, 138]**
Center-right picked points' coordinates: **[1361, 209], [1382, 985], [1036, 985], [1035, 176]**
Left picked points' coordinates: **[1751, 131], [1702, 1053], [1437, 1035], [1416, 169]**
Center-left picked points' coordinates: **[1031, 181], [975, 989], [758, 998], [767, 177]**

PS4-1 readme.txt

24-678: Computer Vision for Engineers
Ryan Wu
ID: weihuanw
PS4-1 Image Mosaicing with Bi-linear Transformation

Operating system: macOS Ventura 13.5.2
IDE you used to write and run your code: PyCharm 2023.1.4 (Community Edition)
The number of hours you spent to finish this problem: 8 hours.

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

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)

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)
```

```
40      rn = cv2.warpPerspective(imageR, M, (w*3,h*3))
41      rng = cv2.cvtColor(rn, cv2.COLOR_BGR2GRAY)
42      th, mask_r = cv2.threshold(rng, 1, 255, cv2.
    THRESH_BINARY)
43      #cv2.imwrite("mask_r.png", mask_r)
44      mask_r = mask_r / 255
45
46      # left image appears upper left corner, but it
    still works in blending.
47      for i in range(4):
48          src_pnts[i][0] = float(pick[2][i][0])
49          src_pnts[i][1] = float(pick[2][i][1])
50          dst_pnts[i][0] = float(pick[3][i][0] + w)
51          dst_pnts[i][1] = float(pick[3][i][1] + h)
52      M = cv2.getPerspectiveTransform(src_pnts,
    dst_pnts)
53
54      ln = cv2.warpPerspective(imageL, M, (w*3,h*3))
55      lng = cv2.cvtColor(ln, cv2.COLOR_BGR2GRAY)
56      th, mask_l = cv2.threshold(lng, 1, 255, cv2.
    THRESH_BINARY)
57      mask_l = mask_l / 255
58      #cv2.imwrite("mask_l.png", mask_l)
59
60      # alpha blending
61      # mask element: number of pictures at that
    coordinate
62      mask = np.array(mask_c + mask_l + mask_r, float
    )
63
64      # alpha blending weight
65      ag = np.full(mask.shape, 0.0, dtype=float)
66      # weight: 1.0 / (num of picture)
67      ag = 1.0 / np.maximum(1,mask) # avoid 0
    division
68
69      # generate result image from 3 images + alpha
    weight
70      result[:,:,0] = result[:,:,0]*ag[:,:] + ln[:,:,
    0]*ag[:,:] + rn[:,:,0]*ag[:,:]
71      result[:,:,1] = result[:,:,1]*ag[:,:] + ln[:,:,
```

```
 71 1]*ag[:,:] + rn[:,:,1]*ag[:,:]
 72     result[:,:,2] = result[:,:,2]*ag[:,:] + ln
    [:,:,2]*ag[:,:] + rn[:,:,2]*ag[:,:]
 73
 74     #cv2.imwrite("result.jpg", result)
 75     if dataset == 0:
 76         cv2.imwrite("wall-stitched.png", result)
 77     elif dataset == 1:
 78         cv2.imwrite("door-stitched.png", result)
 79     elif dataset == 2:
 80         cv2.imwrite("house-stitched.png", result)
 81     else:
 82         cv2.imwrite("pittsburgh-stitched.png",
    result)
 83     cv2.imshow("result", result)
 84
 85 '''
 86 pick 4 points from right image (red point)
 87 '''
 88 def right_click(event, x, y, flags, param):
 89     if event == cv2.EVENT_LBUTTONUP:
 90         mousePick(x, y, 0)
 91
 92 '''
 93 pick 4 points from center (correspond to right,
    red point)
 94 '''
 95 def center_click_r(event, x, y, flags, param):
 96     if event == cv2.EVENT_LBUTTONUP:
 97         mousePick(x, y, 1)
 98
 99 '''
100 pick 4 points from left (blue point)
101 '''
102 def left_click(event, x, y, flags, param):
103     if event == cv2.EVENT_LBUTTONUP:
104         # add your code to select 4 points
105         mousePick(x, y, 2)
106         # pass
107
108 '''
```

```
109 pick 4 points from center (correspond to left,
    blue point)
110 '''
111 def center_click_l(event, x, y, flags, param):
112     if event == cv2.EVENT_LBUTTONUP:
113         # add your code to select 4 points
114         mousePick(x, y, 3)
115         # pass
116 
117 '''
118 idea: handle mouse pick
119 idx
120 0: right
121 1: center (correspond to right)
122 2: left
123 3: center (correspond to left)
124 
125 you can also create your own function for left +
    center selection
126 '''
127 def mousePick(x, y, idx):
128     global rn, cn, ln, imageR, imageC, imageL,
    pick
129     if idx == 0:
130         src = imageR
131         dst = rn
132         wn = "right"
133         col = (0, 0, 255)  # right side red
134     elif idx == 1:
135         src = imageC
136         dst = cn
137         wn = "center"
138         col = (0, 0, 255)  # right side red
139     # you need to add idx 2, 3 cases
140     elif idx == 2:
141         src = imageL
142         dst = ln
143         wn = "left"
144         col = (255, 0, 0)  # left side blue
145     elif idx == 3:
146         src = imageC
```

```
147         dst = cn
148         wn = "center"
149         col = (255, 0, 0)  # left side blue
150     else:
151         return
152 
153     print(idx, x, y)
154     pick[idx].append((x,y))
155     dst = src.copy()
156     # red BGR color in OpenCV, you need to set to
    blue on left side
157     # place circle on the picked point and text
    its serial (0-3)
158 
159     if idx > 1:
160         color = (255, 0, 0)
161     else:
162         color = col
163 
164     for i in range(len(pick[idx])):
165         dst = cv2.circle(dst, pick[idx][i], 5,
    color, 2)
166         dst = cv2.putText(dst, str(i), (pick[idx][
    i][0]+10, pick[idx][i][1]-10),
167                           cv2.FONT_HERSHEY_SIMPLEX
    ,1, color, 1)
168 
169     # please make sure when idx == 3, you need to
    show red color circle in dst
170     # this example erases red circle
171 
172     cv2.imshow(wn, dst)
173     # to make sure image is updated
174     cv2.waitKey(1)
175     if len(pick[idx]) >= 4:
176         print('Is it OK? (y/n)')
177         i = input()
178         if i == 'y' or i == 'Y':
179             if idx >= 3:
180                 savePick()
181                 combine()
```

```
182             elif idx == 0:
183                 print('center 4 points')
184                 cv2.setMouseCallback("center",
    center_click_r)
185             elif idx == 1:
186                 # only taking care of right and
    center, you need to replace 2 lines to start
187                 # picking left and center
    correspondence
188                 print('left 4 points')
189                 cv2.setMouseCallback("left",
    left_click)
190             elif idx == 2:
191                 print('center 4 points')
192                 cv2.setMouseCallback("center",
    center_click_l)
193                 # you need to add pick code
194         else:
195             pick[idx] = []
196             dst = src.copy()
197             cv2.imshow(wn, dst)
198
199
200 parser = argparse.ArgumentParser(description='
    Combine 3 images')
201 parser.add_argument('-d', '--data', type=int, help
    ='Dataset index', default=1)
202 args = parser.parse_args()
203 dataset = args.data
204
205 if dataset == 0:
206     imageL = cv2.imread("wall-left.png")
207     imageC = cv2.imread("wall-center.png")
208     imageR = cv2.imread("wall-right.png")
209 elif dataset == 1:
210     imageL = cv2.imread("door-left.jpg")
211     imageC = cv2.imread("door-center.jpg")
212     imageR = cv2.imread("door-right.jpg")
213 elif dataset == 2:
214     imageL = cv2.imread("house-left.jpg")
215     imageC = cv2.imread("house-center.jpg")
```

```
216     imageR = cv2.imread("house-right.jpg")
217 else:
218     imageL = cv2.imread("pittsburgh-left.jpg")
219     imageC = cv2.imread("pittsburgh-center.jpg")
220     imageR = cv2.imread("pittsburgh-right.jpg")
221 
222 result = cv2.copyMakeBorder(imageC,imageC.shape[0
    ],imageC.shape[0],imageC.shape[1],imageC.shape[1],
223                                 borderType=cv2.
    BORDER_CONSTANT,value=[0,0,0])
224 
225 print(imageL.shape,imageC.shape,imageR.shape,
    result.shape)
226 
227 cv2.namedWindow("left",cv2.WINDOW_NORMAL)
228 cv2.namedWindow("center",cv2.WINDOW_NORMAL)
229 cv2.namedWindow("right",cv2.WINDOW_NORMAL)
230 cv2.namedWindow("result",cv2.WINDOW_NORMAL)
231 
232 ln = imageL.copy()
233 cn = imageC.copy()
234 rn = imageR.copy()
235 
236 cv2.imshow("left", ln)
237 cv2.imshow("center", cn)
238 cv2.imshow("right", rn)
239 cv2.imshow("result", result)
240 
241 pick = []
242 pick.append([])
243 pick.append([])
244 pick.append([])
245 pick.append([])
246 
247 print('use saved points? (y/n)')
248 i = input()
249 if i == 'y' or i == 'Y':
250     loadPick()
251     combine()
252 else:
253     print("right 4 points")
```

```
254     cv2.setMouseCallback("right", right_click)
255 
256 cv2.waitKey()
257 
258 # close all open windows
259 cv2.destroyAllWindows()
260 
```