機器視覺作業報告

Homework#4

姓名: 黃楚祐
學號: 00557043
日期: 2019/6/7

一、橋

方法

從中間向兩邊縫合

1. 在迴圈內由中間，用兩個 index 分別向左向右讀圖片

```
while forwardFrameNum < totalFrame and backwardFrameNum > -1:
    cv2.setTrackbarPos('frame no.','matchingForward',forwardFrameNum)
    cv2.setTrackbarPos('frame no.','matchingBackward',backwardFrameNum)
    frame2Forward = cv2.imread('./dataset1T/dataset_1_%d.jpg' % forwardFrameNum) #向左讀取照片
    frame2Forward = cv2.resize(frame2Forward,(frame2Forward.shape[1]//4,frame2Forward.shape[0]//4))
    frame2Backward = cv2.imread('./dataset1T/dataset_1_%d.jpg' % backwardFrameNum)#向右讀取照片
    frame2Backward = cv2.resize(frame2Backward,(frame2Backward.shape[1]//4,frame2Backward.shape[0]//4))
```

2. 向左向右都要對圖片生出 key point

```
    gray = cv2.cvtColor(frame2Forward, cv2.COLOR_BGR2GRAY)
    kp2Forward = kpdetector.detect(gray,None)
    dt2Forward = kpdetector.compute(gray,kp2Forward)[1] #對向右一張的圖片產生key point和detection

    gray = cv2.cvtColor(frame2Backward, cv2.COLOR_BGR2GRAY)
    kp2Backward = kpdetector.detect(gray,None)
    dt2Backward = kpdetector.compute(gray,kp2Backward)[1]#對向右=左一張的圖片產生key point和detection
```

3. 回圈內的第一張圖片(中間)直接擺上去

```
   if forwardFrameNum == totalFrame // 2 - 25: #第一張圖片直接擺上結果
       TForward = np.eye(3)
       TForward[0,2] = result.shape[1] / 2 - frame2Forward.shape[1] * 0.6 #第一張圖片擺的橫向位置
       TForward[1,2] = 0 #第一張圖片擺縱橫向位置

       TBackward = np.eye(3)
       TBackward[0,2] = result.shape[1] / 2 - frame2Forward.shape[1] * 0.6
       TBackward[1,2] = 0


       result = cv2.warpPerspective(frame2Forward,TForward,(result.shape[1],result.shape[0])).astype(np.float)
       tCount= cv2.warpPerspective(ones,TForward,(result.shape[1],result.shape[0])).astype(np.float)
       count += tCount.astype(np.float)
       disp = result.copy()
       cv2.imshow('stitched image',disp.astype(np.uint8))


       frame1Forward = frame2Forward
       kp1Forward = kp2Forward
       dt1Forward = dt2Forward

       frame1Backward = frame2Backward
       kp1Backward = kp2Backward
       dt1Backward = dt2Backward
```

4. 第一張圖片以外，分別對下一張圖片找出變換矩陣後做透視轉換

```
else:
    AForward, matchesForward = matchesAndHomography(dt1Forward, dt2Forward, kp1Forward, kp2Forward, forwardFrameNum)
    # 向右找出A和matches
    TForward = TForward.dot(AForward)
    warp_img = cv2.warpPerspective(frame2Forward,TForward,(result.shape[1],result.shape[0])).astype(np.float)
    tCount  = cv2.warpPerspective(ones,TForward,(result.shape[1],result.shape[0])).astype(np.float)
    result+= warp_img
    count += tCount.astype(np.float)

    ABackward, matchesBackward = matchesAndHomography(dt1Backward, dt2Backward, kp1Backward, kp2Backward, backwardFrameNum)
    # 向左找出A和matches
    TBackward = TBackward.dot(ABackward)
    warp_img = cv2.warpPerspective(frame2Backward,TBackward,(result.shape[1],result.shape[0])).astype(np.float)
    tCount  = cv2.warpPerspective(ones,TBackward,(result.shape[1],result.shape[0])).astype(np.float)
    result+= warp_img
    count += tCount.astype(np.float)

    tCount= count.copy()
    tCount[tCount == 0] = 1
    disp = result.copy()

    disp[:,:,0] = result[:,:,0] / tCount
    disp[:,:,1] = result[:,:,1] / tCount
    disp[:,:,2] = result[:,:,2] / tCount

    cv2.imshow('stitched image',disp.astype(np.uint8))
```

5. 找出變換矩陣方法和張老師 sample code 一樣

```
def matchesAndHomography(dt1, dt2, kp1, kp2, frameNum):  #計算向左或向右的matches
    # Match descriptors.
    matches = bf.match(dt2,dt1)

    print('FrameNum: {}, # of matches:{}'.format(frameNum,len(matches)))


    # Sort in ascending order of distance.
    matches = sorted(matches, key = lambda x:x.distance)

    src = []
    dst = []
    for m in matches:
        src.append(kp2[m.queryIdx].pt + (1,))
        dst.append(kp1[m.trainIdx].pt + (1,))

    src = np.array(src,dtype=np.float)
    dst = np.array(dst,dtype=np.float)
    # find a homography to map src to dst
    A, mask = cv2.findHomography(src, dst, cv2.RANSAC)
    return A, matches
```

結果

過程影片：https://youtu.be/gVbtc9uiJIk

結論

從中間向兩旁縫圖片的方法比單方向的方法要好。

不過圖片的左上和右上發現是被剪掉的，需要使用更大的陣列才能完整顯示。

二、 Pizza

方法

由不同的起始圖片 ，用 greedy 的方式各自找到最佳的縫合順序

```
images = loadAllImage()
minStitch = 10e10

while frameNum < totalFrame:


    tmpMin, disp = stitchingMatches(frameNum, frame2.shape[0], frame2.shape[1])
    #以不同的開頭圖片找最適合的縫合順序，回傳每張圖片縫合時的matches distance加總，回傳逢合的圖片

    if tmpMin < minStitch: #找出matches distance最小的縫合方法
        minStitch = tmpMin
        minDisp = disp

    cv2.imshow('best stitched image',minDisp.astype(np.uint8))
    #顯示最好的縫合照片

    key = cv2.waitKey(1000) & 0xFF
    if key == 27:
        break
    frameNum += 1
cv2.waitKey()
cv2.destroyAllWindows()
```

1. 先把 12 張圖片都讀到記憶體

```
def loadAllImage(): #把全部朝片讀近來
    images = np.zeros((totalFrame, int(frame2.shape[0])//imgShrink,int(frame2.shape[1])//imgShrink,3)) #12張照片都放在這裡
    for x in range(frameNum, totalFrame):
        frame = cv2.imread('./dataset2/DSC_%d.JPG' % (frameNameStart + (x+1) % totalFrame))
        frame = cv2.resize(frame,(frame.shape[1]//imgShrink,frame.shape[0]//imgShrink))
        images[x] = frame
    images = np.array(images,dtype=np.uint8)
    return images
```

2. 在迴圈內以不同的照片為開頭，比較縫合後照片的 matches distance 總和，選出最佳的縫合後照片

```
while frameNum < totalFrame:


    tmpMin, disp = stitchingMatches(frameNum, frame2.shape[0], frame2.shape[1])
    #以不同的開頭圖片找最適合的縫合順序，回傳每張圖片縫合時的matches distance加總，回傳逢合的圖片

    if tmpMin < minStitch: #找出matches distance最小的縫合方法
        minStitch = tmpMin
        minDisp = disp

    cv2.imshow('best stitched image',minDisp.astype(np.uint8))
    #顯示最好的縫合照片
```

3. stitchingMatches 函式(步驟 3~5): 開頭圖片直接擺進 result

```
def stitchingMatches(frameIndex, height, width): ##每次進入這個function前挑一個起始圖片

    result = np.zeros((int(height)//imgShrink*frameEnlarge,int(width)//imgShrink*frameEnlarge,3))
    count  = np.zeros((int(height)//imgShrink*frameEnlarge,int(width)//imgShrink*frameEnlarge))
    ones = np.ones(((int(height)//imgShrink,int(width)//imgShrink)))

    sumMatchesDst = 0
    gray = cv2.cvtColor(images[frameIndex], cv2.COLOR_BGR2GRAY)
    kp2 = kpdetector.detect(gray,None)
    dt2 = kpdetector.compute(gray,kp2)[1]
    T      = np.eye(3)
    T[0,2] = images[frameIndex].shape[1]
    T[1,2] = images[frameIndex].shape[0]
    result = cv2.warpPerspective(images[frameIndex],T,(result.shape[1],result.shape[0])).astype(np.float)
    t_count= cv2.warpPerspective(ones,T,(result.shape[1],result.shape[0])).astype(np.float)
    count += t_count.astype(np.float)
    disp = result.copy()

    cv2.imshow('stitched image',disp.astype(np.uint8))
    key = cv2.waitKey(20) & 0xFF
```

4. 複製原本的圖片群陣列，並把開頭圖片排除，免得下次又被挑為最 match 的圖片

```
    frame1 = images[frameIndex] #起始圖片
    kp1 = kp2
    dt1 = dt2

    tmpImages = images.copy()

    end = totalFrame - 1

    tmpImages[frameIndex] = tmpImages[end]

    end -=1
    #把使用的初始圖片從圖片群中排除
```

5. 跑 11 次迴圈，在這 11 次裡找出最好的照片順序並把 matches distance 相加，以便在步驟 2 的時候比較優劣，在迴圈的最後都會把該次挑的照片從照片群裡移除

```
88      while end > -1:
89          i, matches, dt2, kp2, frame2, matchDst = phaseMatches(tmpImages, dt1, end)
90          #找出最適合的圖片
91
92
93          sumMatchesDst += matchDst
```

6. phaseMatches 函式: 在照片群裡剩下的圖片挑一張最好的照片做為下一張縫合的圖片 ，判定圖片的好壞是用 matchesDstRate = [sum(matches.distance) / len(matches)] 最後回傳最小的 matchesDstRate，給步驟 5 相加

```
28  def phaseMatches(images, dt1, frameRemain): #選擇最好的下一張圖片
29      matchesDstMin = 1000000
30      i = 0
31      while i <= frameRemain:
32          gray = cv2.cvtColor(images[i], cv2.COLOR_BGR2GRAY)
33          kp2 = kpdetector.detect(gray,None)
34          dt2 = kpdetector.compute(gray,kp2)[1]
35          matches = bf.match(dt2,dt1)
36
37          matchesDstRate = sum(m.distance for m in matches) // len(matches)
38          #使用(distance總和除以matches的數量)來判斷優劣，越小越好
39
40          if(matchesDstRate < matchesDstMin) :
41              matchesDstMin = matchesDstRate
42              imgIndex = i
43              maxMatches = matches
44              dt2Max = dt2
45              kp2Max = kp2
46              imgMax = images[i].copy()
47          i+=1
48      maxMatches = sorted(maxMatches, key = lambda x:x.distance)
49      return imgIndex, maxMatches, dt2Max, kp2Max, imgMax, matchesDstMin
50      #回傳imgIndex以避免重複使用，回傳matchesDstMin表示最好的matchesDstRate以便之後加總判斷整個順序是否為最好的結果
```

結果

演算法認為最好的圖片:

我認為最好的圖片:

過程影片 : https://youtu.be/273WQiQwmaA

結論

問題總是出在最後一張，最後一張在貼上去的時候總會跟另外一邊較早貼好的照片對不上，我還不知道該怎麼解決。