機器視覺作業報告

影像拼接

姓名: 王心妤
學號: 00657127
日期: 2019/06/08

方法

程式碼: https://github.com/creek0810/2019_mv_hw4

在這次的作業中，我試圖將各個步驟分成不同的 function，主要分為 pretreat、calc_transform、stitch 及 show_image。這些 function 將在下面一一進行介紹。

1. pretreat

   該 function 主要的目的為，讀取影像、將該 frame 設置成適當的大小，最後利用 kpdetector 偵測該 frame 的 keypoint 及特徵描述。

```python
def pretreat(kpdetector, frame_num):
    cap.set(cv2.CAP_PROP_POS_FRAMES, frame_num)
    ret, cur_frame = cap.read()
    if ret == False:
        return False, False, False
    cur_frame = cv2.resize(cur_frame,(cur_frame.shape[1]//4,cur_frame.shape[0]//4))
    gray = cv2.cvtColor(cur_frame, cv2.COLOR_BGR2GRAY)
    kp = kpdetector.detect(gray,None)
    return cur_frame, kpdetector.compute(gray,kp)[1], kp
```

2. calc_transform

   該 function 利用 brute-force matcher 對傳入的特徵點進行匹配，利用匹配好的特徵點，建立 src 及 dst 兩個 point list。將 src 及 dst 兩個 list 傳入 findHomography，計算出將 src 貼到 dst 的變換矩陣。最後再將算出來的變換矩陣(A)，乘上 dst 到結果圖上的變換矩陣(T)，即可得到 src 貼到結果圖上所需要的變換矩陣。

```python
def calc_transfrom(bf, pre_dt, cur_dt, pre_kp, cur_kp, T):
    matches = bf.match(cur_dt, pre_dt)
    matches = sorted(matches, key = lambda x:x.distance)

    src = []
    dst = []
    for m in matches:
        src.append(cur_kp[m.queryIdx].pt + (1,))
        dst.append(pre_kp[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)
    T = T.dot(A)
    return T
```

3. stitch

該 function 透過 warpPerspective 計算出 cur_frame 經過 T 轉換矩陣貼到 result 上的透視轉換，並將此結果(warp_img)貼上 result。又由於 result 上某些地方可能被重複貼圖，所以建立 count，來記錄 pixel 上總共被貼了幾次，以便顯示時，對該 pixel 進行平均，顯示正確的圖。

```
def stitch(cur_frame, result, ones, count, T):
    warp_img = cv2.warpPerspective(cur_frame,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)
    result += warp_img
    count = t_count.astype(np.float) + count
    return count
```

4. show_image

該 function 將對 reuslt 進行平均處理(在 stitch 最後的解釋裡提到)，顯示，並儲存得到的結果(disp)。

```
def show_image(result, count, out):
    t_count = count.copy()
    t_count[t_count == 0] = 1

    disp = result.copy()
    disp[:,:,0] = result[:,:,0] / t_count
    disp[:,:,1] = result[:,:,1] / t_count
    disp[:,:,2] = result[:,:,2] / t_count

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

5. main

主要用來建立呼叫 function 時所需的變數，及定義貼圖的順序。

```
result = np.zeros((int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))//4,int(cap.get(cv2.CAP_PROP_FRAME_WID
count  = np.zeros((int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))//4,int(cap.get(cv2.CAP_PROP_FRAME_WID
ones = np.ones(((int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))//4,int(cap.get(cv2.CAP_PROP_FRAME_WIDTH
out = cv2.VideoWriter('./result/exercise1.avi',cv2.VideoWriter_fourcc(*'XVID'), 20.0,
                      ((int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))//4*3),(int(cap.get(cv2.CAP_PROP
# create kpdetector
kpdetector = cv2.xfeatures2d.SIFT_create()

# create BFMatcher object
bf = cv2.BFMatcher(cv2.NORM_L2, crossCheck=True)
```

主要需要宣告的變數

```
while middle_frame + frame_count < total_frame:
    if frame_count == 0:
        # init
        cur_frame, pre_dt_left, pre_kp_left = pretreat(kpdetector, middle_frame)
        pre_kp_right = pre_kp_left
        pre_dt_right = pre_dt_left

        if type(cur_frame) == type(True):
            break
        # calc loc
        T_left = np.eye(3)
        T_left[0,2] = (result.shape[1] - cur_frame.shape[1]) // 2 - 190
        T_left[1,2] = 0
        T_right = T_left
        # stitch
        count = stitch(cur_frame, result, ones, count, T_left)
        show_image(result, count, out)

    else:
        # stitch left
        cur_frame, dt2, kp2 = pretreat(kpdetector, middle_frame + frame_count)
        if type(cur_frame) == type(True):
            break
        T_right = calc_transfrom(bf, pre_dt_right, dt2, pre_kp_right, kp2, T_right)
        count = stitch(cur_frame, result, ones, count, T_right)
        pre_kp_right = kp2
        pre_dt_right = dt2

        # stith right
        cur_frame, dt2, kp2 = pretreat(kpdetector, middle_frame - frame_count)
        if type(cur_frame) == type(True):
            break
        T_left = calc_transfrom(bf, pre_dt_left, dt2, pre_kp_left, kp2, T_left)
        count = stitch(cur_frame, result, ones, count, T_left)
        pre_kp_left = kp2
        pre_dt_left = dt2
```

HW 1 的貼圖順序，當 frame_count(當前 frame 到 middle_frame 的差值)非 0 時，便往左貼，再往右貼

```python
for frame_num in range(58, 69):
    if frame_num == 58:
        # init
        cur_frame, pre_dt, pre_kp = pretreat(kpdetector, path + str(frame_num) + ".JPG")

        if type(cur_frame) == type(True):
            break
        # calc loc
        T = np.eye(3)
        T[0,2] = (result.shape[1] - cur_frame.shape[1]) // 2
        T[1,2] = (result.shape[0] - cur_frame.shape[0]) // 2
        # stitch
        count = stitch(cur_frame, result, ones, count, T)
        show_image(result, count)

    else:
        # stitch
        cur_frame, dt2, kp2 = pretreat(kpdetector, path + str(frame_num) + ".JPG")
        if type(cur_frame) == type(True):
            break
        T = calc_transfrom(bf, pre_dt, dt2, pre_kp, kp2, T)
        count = stitch(cur_frame, result, ones, count, T)
        pre_kp = kp2
        pre_dt = dt2
        show_image(result, count)
```

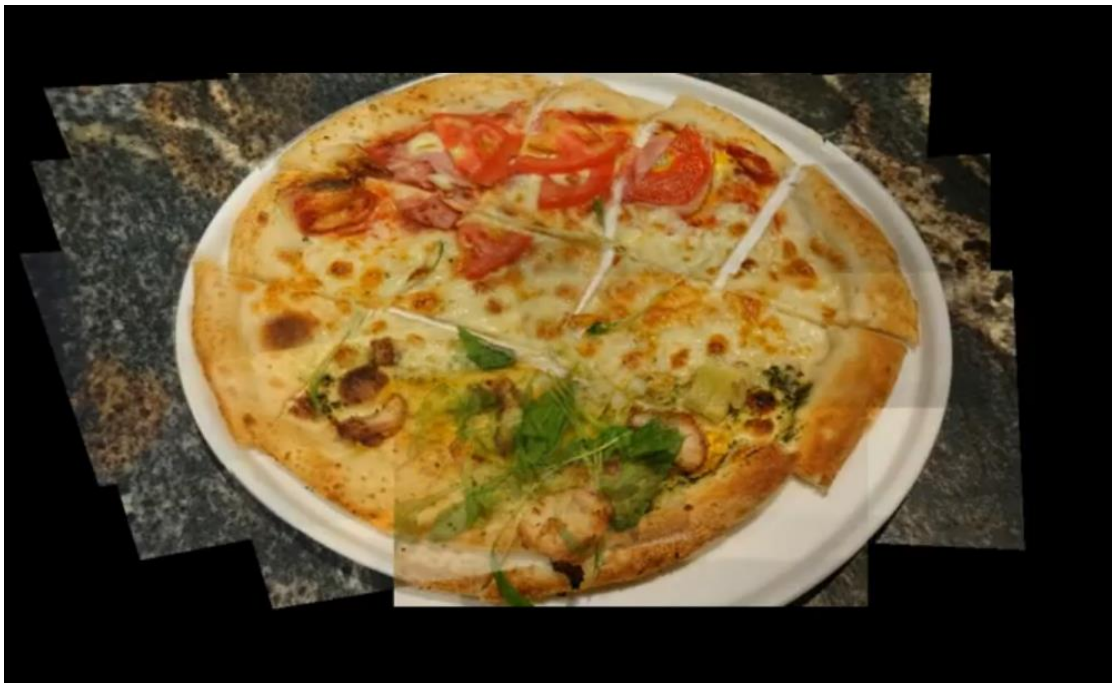
HW 2 的貼圖順序，從編號 58 貼到 68

結果

HW 1: https://www.youtube.com/watch?v=UcUl6aber7Y&feature=youtu.be

利用影片讀取，從中間開始往兩側進行影像縫合。雖說失真程度已經比從左邊開始或從右邊開始好，但明顯的可以看出成果圖右側較左側模糊，推測是因為在拍攝右側影像時，速度較快所致。

HW 2: https://www.youtube.com/watch?v=UPU53mazLw0&feature=youtu.be
將切成數塊的 pizza 圖片進行影像縫合，最後一塊的位置明顯的跟先前拚好的圖重疊在一起，導致效果沒那麼好。



結論

這次的作業中，我學習到了如何使用特整點 detector 來做特徵偵測，並利用這些特徵點來進行影像縫合。
在拼貼 pizza 的部分，我曾嘗試在拼接時，將當前圖片與當前拼接好的進行特徵比對來縫合，試圖解決最後一張圖片會與先前拼接好的重疊的問題，但是效果不理想(特徵比對錯誤率偏高，導致整個圖拼錯位置)，所以最後還是以傳統的方式進行拼接。

參考文獻

無