Computer Vision Fall 2018

National Taiwan University

Assignment 3:

Projective Geometry

姓名：張邵瑀　　學號：r06921081

Part 1

---

solve_homography(u, v)

```
def solve_homography(u, v):
    N = u.shape[0]
    if v.shape[0] is not N:
        print(v.shape[0], N)
        print('u and v should have the same size')
        return None
    if N < 4:
        print('At least 4 points should be given')
    # --- version 1
    """
    A = []
    for u_i, v_i in zip(u, v):
        A.append([u_i[0], u_i[1], 1, 0, 0, 0, -u_i[0]*v_i[0], -u_i[1]*v_i[0]])
        A.append([0, 0, 0, u_i[0], u_i[1], 1, -u_i[0]*v_i[1], -u_i[1]*v_i[1]])
    A = np.array(A)
     # if you take solution 2:

    b = v.reshape(v.shape[0]*2,1)
    x = np.linalg.solve(A, b)
    x = np.concatenate((x, np.array([[1]])), axis = 0)

    # H = np.zeros((3, 3))
    H = x.reshape((3,3))
    # print(H)
    return H
    """
    # --- version 2
    A = []
    for u_i, v_i in zip(u, v):
        A.append([u_i[0], u_i[1], 1, 0, 0, 0, -u_i[0]*v_i[0], -u_i[1]*v_i[0], -v_i[0]])
        A.append([0, 0, 0, u_i[0], u_i[1], 1, -u_i[0]*v_i[1], -u_i[1]*v_i[1], -v_i[1]])
    A = np.array(A)
    ATA = np.dot(A.T, A)
    U, z, V = np.linalg.svd(ATA, full_matrices=True)
    H = U[:, -1]
    return H.reshape((3,3))/H[-1]
```

Output Image:

Part 2

QR code:

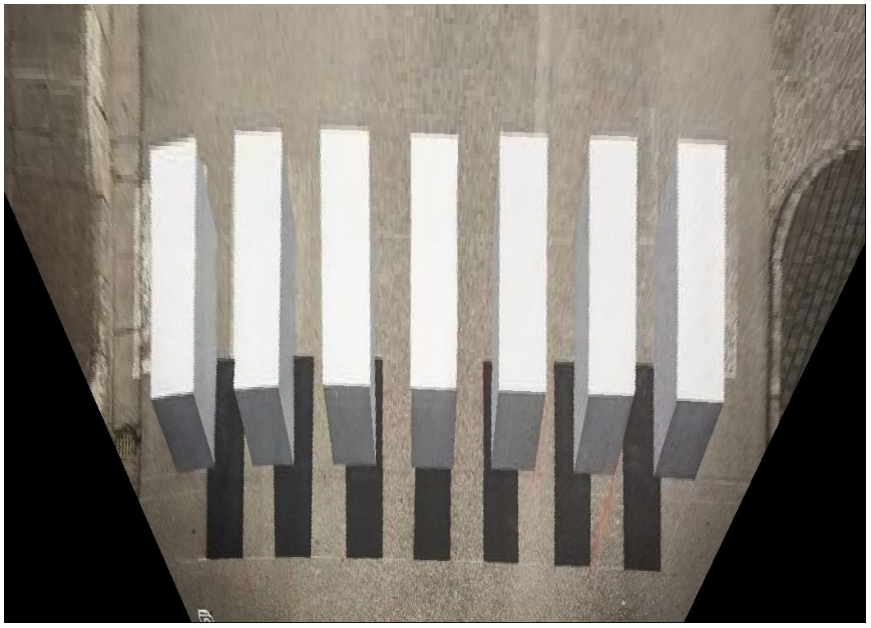
decoded link:

http://media.ee.ntu.edu.tw/courses/cv/18F/

Part 3

top view:

With one matrix



Can you get the parallel bars from the top view?

不能，第1種可能是因為路面本身就不平，所以在正面看時路濡的圖案會變型，除非投影到三維中才有可能全部攤平，在二維平面上沒有辦法拉平，第2種可能是因為相機本身的光學設計對於要捕捉的影像擷取後可能會有類似魚眼的形變可能要用棋盤校正法去回復影像。

bonus

我的bonus作法是先各自找出影片與Template中的QRcode位置，取出其四個頂點，再做Part 1的轉換，首先QRcode的找法是使用Opencv中的findContours找出護相包含階層數大於5的3個四邊型(QRcode的3個邊角)雖然一般QRcode通常在右下角會有一個階層數較小的定位點，以判斷QRcode的正面方向，但這次的QRcode的上半部被切掉不是封閉的，所以階層並無法當作判斷，所以我必須另外

找方法，所以我就將這3個四邊型的12個點丟到convexHull中找出點數最少的凸多邊形，其應該會是一個五邊型，但有時並沒有剛好5個頂點，這時就用approxPolyDP把接近同一條線上的近似到凸多邊形中，再由各點之間的距離求出最短的邊(藍色邊)在經由兩個邊的四個點求出兩線延伸的交點(bonus.py 中的cross function)，但其取得的點會有誤差(就算在Template中也如此)，所以我以該點為中心，抓取周圍100x100的影像(黃色區域)，再用findContours取出黑色像素圖形，再找出其中對於橘點最遠的頂點，這樣一來取出的點就非常準確了，接著只要把兩圖各自對到的點丟入PART 1的function中做貼上的動作，以下是結果連結，但因為我不想只貼照片，所以我是把一段影片嵌進去。

https://drive.google.com/file/d/1jUY2ZMLmVgPErIUlAjYyxXFzb-hrCOAA/view?usp=sharing