機器視覺作業報告

HW2

姓名:陳伸蓉
學號:00957119
日期:2023/5/6

方法

前置作業

```
import cv2
import cv2.aruco as aruco
import numpy as np
```

1. Calibrate the camera by the ChArUco board shown in CharUco_board.mp4 and show the parameters.(紅字為 bonus 更動之處,其餘地方沒改)

這邊在做偵測 marker 的前置作業，先指定要用的字典及棋盤大小等，若字典跟大小錯誤的話會無法偵測到。

```
cap = cv2.VideoCapture('CharUco_board.mp4')#讀取影片
#縮小畫面
frameWidth    = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))//2
frameHeight   = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))//2
#創建一個aruco標記檢測器參數對象。這邊使用默認參數
arucoParams   = aruco.DetectorParameters_create()
#設置aruco標記檢測器的角點優化方法為亞像素級別優化。
arucoParams.cornerRefinementMethod = aruco.CORNER_REFINE_SUBPIX
#獲取一個aruco字典，字典大小為6x6，標記數量為250
arucoDict     = aruco.Dictionary_get(aruco.DICT_6X6_250)

# 必須描述ChArUco board的尺寸規格
gridX         = 5 # 水平方向5格
gridY         = 7 # 垂直方向7格
squareSize    = 4 # 每格為4cmX4cm
# ArUco marker為2cmX2cm
#建立棋盤並指定字典
charucoBoard = aruco.CharucoBoard_create(gridX,gridY,squareSize,squareSize/2,arucoDict)

refinedStrategy = True#設置標記檢測器使用優化後的策略。#使用優化算法來提高檢測精度
criteria        = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 100, 0.00001)
#終止標準(criteria)，用於定義當檢測精度達到一定程度時停止迭代的條件。
#最大迭代次數 TERM_CRITERIA_MAX_ITER=100，最大誤差 cv2.TERM_CRITERIA_EPS=0.00001
#有一個條件被滿足，就會停止迭代
frameId         = 0
collectCorners = []
collectIds      = []
collectFrames   = []
```

cap = cv2.VideoCapture('my_board.mp4')

在 bonus 中所使用的字典也是 aruco.DICT_6X6_250

先偵測角點並把相機校正需要用的參數存起來。(collectCorners,collectIds, collectFrames)

```
while True:
    ret, frame = cap.read()
    if not ret:
        break
    frame = cv2.resize(frame,(frameWidth,frameHeight))
    #指定字典及參數進行偵測
    (corners, ids, rejected) = aruco.detectMarkers(frame, arucoDict, parameters=arucoParams)
    if refinedStrategy:
        corners, ids, _, _ = aruco.refineDetectedMarkers(frame,charucoBoard,corners,ids,rejected)
        #用來進一步精確化已經檢測到的標記角點的位置。它會針對檢測到的標記進行優化和校正，以改善標記的位置估計精度
            #(輸入圖像，charucoBoard:用來校正標記的Charuco板，corners:檢測到的標記的角點位置,ids是標記的ID號碼,
                # rejected是被拒絕的標記角點的位置)
            #此函數會返回優化後的標記角點位置，優化後的標記ID，以及被拒絕的標記角點的位置和ID。

# 如果當前幀是每100幀中的第50幀，並且檢測到了17個ArUco標記
    if frameId % 100 == 50 and len(ids)==17:  # 17 ArUco markers
        #如果偵測到了 17 個 ArUco marker，就把偵測到的角點位置和 ID 存入 collectCorners
        collectCorners.append(corners)# 將標記的角點坐標存起來
        collectIds.append(ids.ravel()) # 將標記的id存起來
        collectFrames.append(frame)# 將幀添加到列表中
    if len(corners) > 0:#有偵測到corner就畫
        aruco.drawDetectedMarkers(frame, corners, ids)
    cv2.imshow('Analysis of a CharUco board for camera calibration',frame)
    #顯示該影格，並等待 20 毫秒
    if cv2.waitKey(20) != -1:
        break
    frameId += 1
cv2.destroyAllWindows()
cap.release()
```

先用 calibrateCameraAruco 來進行相機校正

```
caliCorners=np.concatenate([np.array(x).reshape(-1,4,2) for x in collectCorners],axis=0)
#collectCorners 包含了攝像機捕捉到的所有標記的角點坐標，通過 np.array(x).reshape(-1,4,2)
#將每個標記的角點坐標變成一個 4 × 2 np array，利用np.concatenate 將所有標記的角點坐標形成一個 N × 4 × 2 的數組caliCorners。
#N是 ArUco 標記的總數。

counter=np.array([len(x) for x in collectIds])
#counter長度為ArUco標記的總數，數值為我每次找到的ID數量。

caliIds=np.array(collectIds).ravel()
#將所有標記的ID拉平成一維數組。

cameraMatrixInit = np.array([[ 1000.,    0., frameWidth/2.],[    0., 1000., frameHeight/2.],[    0.,    0.,           1.]])
#cameraMatrixInit 是一個 3 × 3 的矩陣，表示相機的內部參數矩陣。
#在這個例子中，將焦距設為 1000 像素，將主點設在圖像中心，沒有進行旋轉或者傾斜的校正。

distCoeffsInit   = np.zeros((5,1))#預設相機的畸變係數。

#aruco marker的相機矯正
ret, aruco_cameraMatrix, aruco_distCoeffs, aruco_rvects, aruco_tvects = aruco.calibrateCameraAruco(caliCorners,
                                    caliIds,counter,charucoBoard,(frameWidth,frameHeight),cameraMatrixInit,distCoeffsInit)
#在這個例子中，將所有畸變係數都初始化為 0。
#aruco.calibrateCameraAruco() 函數是 OpenCV 中用於相機標定的函數，用於計算相機的內部參數和畸變係數。
#caliCorners 是包含所有標記的角點坐標的數組。
#caliIds 是包含所有標記的 ID 的一維數組。
#counter 是包含每個標記 ID 數量

print("aruco_cameraMatrix\n",aruco_cameraMatrix)
print("aruco_distCoeffs\n",aruco_distCoeffs)
print("aruco_rvects\n",aruco_rvects)
print("aruco_tvects\n",aruco_tvects)
```

使用 calibrateCameraCharuco 做第二次校正來得到更精準的相機參數

```
caliCorners=[]
caliIds    =[]
for corners, ids, frame in zip(collectCorners,collectIds,collectFrames):
    ret, charucoCorners, charucoIds = aruco.interpolateCornersCharuco(corners,ids,frame,charucoBoard
                                                                      ,aruco_cameraMatrix,aruco_distCoeffs)
    #對棋盤格子的角落坐標進行內插，得到更準確的棋盤格子的角落坐標和其對應的ID。
    #使用Charuco棋盤格子的角點和編號來對圖像中的角點和編號進行內插，以便更準確地測量相機的參數。
    caliCorners.append(charucoCorners)
    caliIds.append(charucoIds)

ret, charuco_cameraMatrix, charuco_distCoeffs, charuco_rvects, charuco_tvects = aruco.calibrateCameraCharuco(caliCorners,
                                          caliIds,charucoBoard,(frameWidth,frameHeight), aruco_cameraMatrix,aruco_distCoeffs)


print("aruco_cameraMatrix\n",aruco_cameraMatrix)
print("aruco_distCoeffs\n",aruco_distCoeffs)
print("aruco_rvects\n",aruco_rvects)
print("aruco_tvects\n",aruco_tvects)
```

2. Detect the six ArUco markers in the video file arUco_marker.mp4. Show six different videos on those markers for augmented reality.

先讀取要撥放之 youtube 影片

```
import pafy
video = []
url = "https://www.youtube.com/watch?v=0QmZtYn3LcE"
ty_video = pafy.new(url, basic=False, gdata=False)
best = ty_video.getbest(preftype="mp4")  #读取最高品质的版本
videotemp = cv2.VideoCapture(best.url)
video.append(videotemp)
url      = "https://www.youtube.com/watch?v=7Glo9tYe1kU"
ty_video = pafy.new(url,basic=False,gdata=False)
best     = ty_video.getbest(preftype="mp4")#讀取最高品質的版本
vedio2    = cv2.VideoCapture(best.url)
video.append(vedio2)
url      = "https://www.youtube.com/watch?v=JsvG68OXwhk"
ty_video = pafy.new(url,basic=False,gdata=False)
best     = ty_video.getbest(preftype="mp4")#讀取最高品質的版本
vedio3    = cv2.VideoCapture(best.url)
video.append(vedio3)
url      = "https://www.youtube.com/watch?v=9fi41sVDw8c"
ty_video = pafy.new(url,basic=False,gdata=False)
best     = ty_video.getbest(preftype="mp4")#讀取最高品質的版本
vedio4    = cv2.VideoCapture(best.url)
video.append(vedio4)
url      = "https://www.youtube.com/watch?v=JbyXJO_yin4"
ty_video = pafy.new(url,basic=False,gdata=False)
best     = ty_video.getbest(preftype="mp4")#讀取最高品質的版本
vedio5    = cv2.VideoCapture(best.url)
video.append(vedio5)
url      = "https://www.youtube.com/watch?v=HU5Qq97cxvU"
ty_video = pafy.new(url,basic=False,gdata=False)
best     = ty_video.getbest(preftype="mp4")#讀取最高品質的版本
vedio6    = cv2.VideoCapture(best.url)
video.append(vedio6)
print(len(video))
```

設置參數

```
cap = cv2.VideoCapture('arUco_marker.mp4')
markerSize  = 6 #6cm

#縮小視窗。
frameWidth    = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))//2
frameHeight   = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))//2
#創建ArUco標記的參數對象
arucoParams = aruco.DetectorParameters_create()
#提高角點檢測的精度
arucoParams.cornerRefinementMethod = aruco.CORNER_REFINE_SUBPIX
#arucoParams.adaptiveThreshConstant
arucoDict    = aruco.Dictionary_get(aruco.DICT_7X7_50)
```

cap = cv2.VideoCapture('my_marker.mp4')

arucoDict = aruco.Dictionary_get(aruco.DICT_6X6_250)

先針對讀取進來的影片每一幀做二值化處理

```
while True:
    ret, frame = cap.read()
    if not ret:
        break
    frame = cv2.resize(frame,(frameWidth,frameHeight))
    tempframe=cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)#轉成灰階進行二值化
    rettemp, th1 = cv2.threshold(tempframe, 115, 255, cv2.THRESH_BINARY)
    (corners, ids, rejected) = aruco.detectMarkers(th1, arucoDict, parameters=arucoParams
                                                   ,cameraMatrix=charuco_cameraMatrix, distCoeff=charuco_distCoeffs)
```

在 bnous 的影片中，角點均有偵測到，所以並未使用二值化是直接對 frame 進行偵測

重要程式碼片段說明:

在這邊因為會有角點偵測不到的問題，所以在老師提供的影片中我進行了二值化處理，最後找到最適合的數值是 115。而我也有查到 arucoParams 中有一個參數叫 adaptiveThreshConstant 可以用來調整自適應閾值二值化的閾值，但因為在調整參數時找不到最佳數值，所以最後還是使用對影片二值化的方式。

在用 detectMarkers 時有加上相機校正的參數 cameraMatrix=charuco_cameraMatrix, distCoeff=charuco_distCoeffs，提高偵測的準確率。

進行二維投影，並畫出角點及姿態軸

img_corners =np.array([[[vframe.shape[1],vframe.shape[0]],
[0,vframe.shape[0]],[0,0],[vframe.shape[1],0]]], dtype=np.float32)

```
    if len(corners)>0:#如果有偵測到
        #if len(corners)!=6:
         #   print(ids)
        for i in range(len(ids)):#歷遍所有偵測到的角點
            if ids[i] in range(1,7):
                ret1, vframe = video[int(ids[i]-1)].read()#第幾號corners的id是多少
                now_corner=corners[i]
                if ret1==False:
                    break;
                img_corners = np.array([[[vframe.shape[1],vframe.shape[0]],[0,vframe.shape[0]],[0,0],
                                        [vframe.shape[1],0]]], dtype=np.float32)
                    #要對到角點的影片像素點位置
                M = cv2.getPerspectiveTransform(img_corners,now_corner)
                    #算出從影片到要映射的區域到標記角點的四邊形之間的透視變換矩陣M
                warped_frame = cv2.warpPerspective(vframe,M, (frameWidth,frameHeight))
                    #將影片中要映射的區域映射到標記角點的四邊形中
                cv2.fillPoly(frame, now_corner.astype(int), 0, 255)
                    #將marker位置塗黑
                frame = cv2.add(frame, warped_frame)
                    #直接把要放進marker的影片跟原始影片相加即可
        aruco.drawDetectedMarkers(frame, corners)

        rvects, tvects, _ = aruco.estimatePoseSingleMarkers(corners, markerSize, charuco_cameraMatrix, charuco_distCoeffs)
        for rvec,tvec in zip(rvects,tvects):
            aruco.drawAxis(frame, charuco_cameraMatrix, charuco_distCoeffs, rvec, tvec, markerSize/2)
               #aruco.drawAxis()函數在標記上繪製姿態軸。最後，使用cv2.drawMarker()函數在2D圖像上繪製投影點的標記

    cv2.imshow('arUco marker',frame)
    if cv2.waitKey(20) != -1:
        break

cv2.destroyAllWindows()
cap.release()
```

video[int(ids[now_ids])].read()

重要程式碼片段說明:

這邊有一個需要注意的小地方是不同字典的 ids 的 base 不同，aruco.DICT_7X7_50 是 1-base，而 aruco.DICT_6X6_250 是 0-base，所以一個需要-1 另一個則不用。

在算透視變換矩陣 M 時，因為 marker 角點的順序是以 marker 的左上、右上、右下、左下的方式來排列，所以需要依據影片中顯示出來的位置來調整 img_corners 的點順序。但 img_corners 的順序一定會是順時針或逆時針的，如果排序錯誤則會導致影像出來結果是扭曲的。

getPerspectiveTransform 會得到一張跟影片大小一樣且想放入之影片的圖片已在要放入之位置上且其餘位置為黑色的圖片。而得到要放上去的圖片後再用塗黑的方式將 frame 中要放進去的那個 marker 的像素值直接變成 0，就可以直接透過 add 的方式放入 frame 中而不受影響。(下方為大致上的示意圖)

- 使用三維投影來讓圖片立傾斜(這邊只放置有差異之程式碼)

跟二維的差異處:

需先宣告一個 3D 角點座標 markerCorners3D，點座標為四個角點的(x,y,z)，x 軸控制左右，y 軸控制上下，z 軸則控制看起來是否會浮起來。每軸的座標都是以 0 為中心，所以如果想要用 3D 角點座標來實做二維投影時，點座標會是[[3,3,0],[-3,3,0],[-3,-3,0],[3,-3,0]]，這是因為我的 marker 大小為 6*6，所以 X 跟 Y 的範圍都是-3~3(總長為 6)，而 Z 軸的部分均為 0(二維要在平面上，沒有第三軸)，不同的數值會造成不同的效果，而這邊座標點的順序為左下角、右下角，右上角，左上角。

原本是歷遍所有 corner 的座標來實作，而這邊則改成歷遍所有的 zip(rvects,tvects)的方式。也從直接拿 corner 來計算透視變換矩陣，改成用 proj_pt_with_dist 來計算。

```
markerCorners3D = np.array([[5,4,1],[-1,4,6],[-1,-2,6],[5,-2,1]],dtype=float)
    #(X軸偏移量(橫的)，Y偏移量(直的)，Z軸偏移量(看起來會浮起來))
while True:
    ret, frame = cap.read()
    if not ret:
        break
```

```
    if len(corners) > 0:
      #  aruco.drawDetectedMarkers(frame, corners, ids)
        #if len(corners)!=6:
        #    print(ids)
        rvects, tvects, _ = aruco.estimatePoseSingleMarkers(corners, markerSize, charuco_cameraMatrix, charuco_distCoeffs)
        #於每個檢測到的標記，它使用 aruco.estimatePoseSingleMarkers() 函數來估算標記的姿態。
        #這個函數返回旋轉向量 (rvects) 和平移向量 (tvects)，以及用於估算姿態的相機矩陣和失真係數。這些向量可以用來在標記上繪製姿態軸。
       # print(ids)
        now_ids=0
        for rvec,tvec in zip(rvects,tvects):
            ret1, vframe = video[int(ids[now_ids]-1)].read()
            if ret1==False:
                break
            proj_pt_with_dist,_ = cv2.projectPoints(markerCorners3D, rvec, tvec, charuco_cameraMatrix, charuco_distCoeffs)
           #cv2.projectPoints()函數將3D標記角點投影到2D圖像平面
            #aruco.drawAxis(frame, charuco_cameraMatrix, charuco_distCoeffs, rvec, tvec, markerSize/2)
            #aruco.drawAxis()函數在標記上繪製姿態軸。最後，使用cv2.drawMarker()函數在2D圖像上繪製投影點的標記
            img_corners = np.array([ [0,vframe.shape[0]],[vframe.shape[1],vframe.shape[0]],[vframe.shape[1],0],[0,0]],
                                   dtype=np.float32)
            proj_pt_with_dist_temp=proj_pt_with_dist
            proj_pt_with_dist_temp=proj_pt_with_dist.reshape(4, 2)
            proj_pt_with_dist_temp=proj_pt_with_dist_temp.astype(np.float32)
            M = cv2.getPerspectiveTransform(img_corners,proj_pt_with_dist_temp)
            vframe_tran = vframe
            warped_frame = cv2.warpPerspective(vframe_tran,M, (frameWidth,frameHeight))
           # print(proj_pt_with_dist.shape)
            cv2.fillPoly(frame, [proj_pt_with_dist_temp.astype(int)], 0, 16)
            frame = cv2.add(frame, warped_frame)
            now_ids+=1
```

video[int(ids[now_ids])].read()

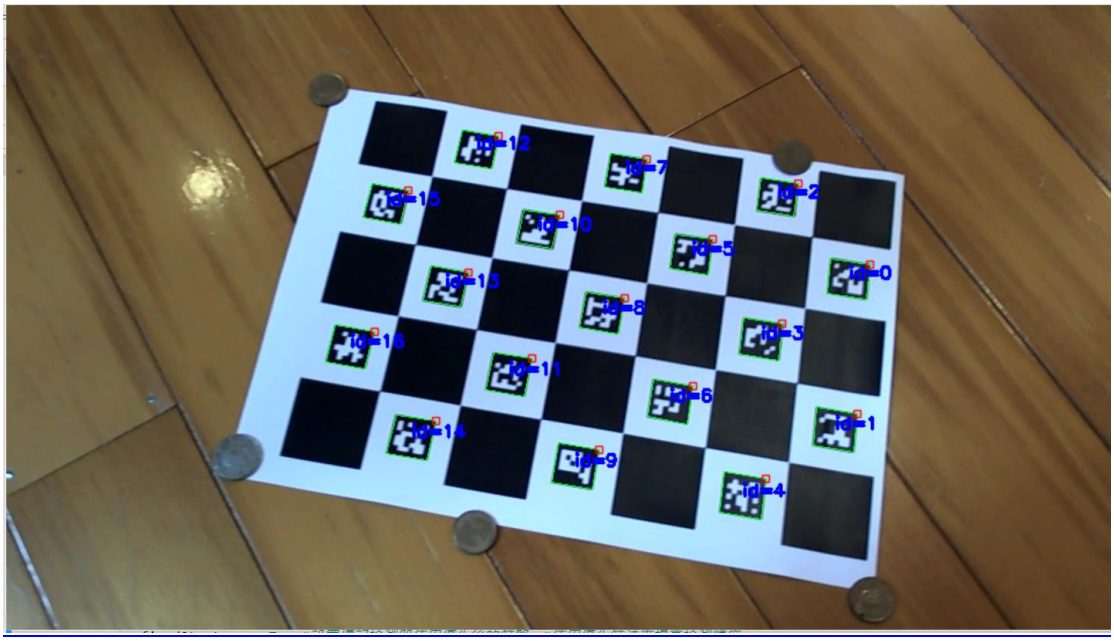
重要程式碼片段說明:

在這邊因為我想要呈現往左及往下偏移並且傾斜，所以我將 X 的範圍+2(往左偏)，Y 的範圍+1，左邊角點的 Z 值為 1、右邊角點 Z 值為 6(在影片中大部分的情況下看起來會是右高左低)來達成(以二維的影片中的姿態軸來調整)。

在使用 proj_pt_with_dist 來算轉換矩陣時有一個需要注意的地方是它的 shape 會是(4,1,2)，所以需要透過 reshape 的方式將其改成(4,2)，而 cv2.projectPoints 中將會使用到 markerCorners3D 來透過改變數值得到不同效果。

結果

(1)原始影片的棋盤:



相機校正得到的參數:

aruco.calibrateCameraAruco 所得到之結果

```
aruco_cameraMatrix
 [[913.81070779   0.         479.5586767 ]
 [  0.         921.0914589  293.00945991]
 [  0.           0.           1.        ]]
aruco_distCoeffs
 [[ 0.06739156]
 [-0.11176159]
 [-0.00446193]
 [-0.00266387]
 [-0.01974274]]
aruco_rvects
 (array([[ 1.85560181],
       [ 2.13559441],
       [-0.43460999]]), array([[ 2.02379095],
       [ 2.16232636],
       [-0.37108907]]), array([[ 1.95358902],
       [ 2.13934218],
       [-0.43973432]]), array([[ 2.07453931],
       [ 2.14265824],
       [-0.2240849 ]]), array([[ 1.84966146],
       [ 2.40870897],
       [-0.12752218]]), array([[ 2.00997394],
       [ 2.25545981],
       [-0.31581337]]), array([[ 2.03659511],
       [ 2.15591552],
       [-0.52993239]]), array([[ 2.12746156],
       [ 2.19254096],
       [-0.09196582]]), array([[ 2.22641063],
       [ 2.13735615],
       [-0.11921934]]), array([[ 2.23525175],
       [ 2.03372319],
       [-0.02535581]]), array([[ 2.23620076],
       [ 2.09170461],
       [-0.1405798 ]]))
aruco_tvects
 (array([[-12.14589974],
       [-10.64405907],
       [ 56.98317687]]), array([[-13.38017857],
       [-18.22560279],
       [ 59.1431909 ]]), array([[-10.80232673],
       [ -8.92702527],
       [ 59.25512522]]), array([[-18.97385837],
       [-17.05133588],
       [ 60.35444519]]), array([[-11.66868767],
       [-17.99299082],
       [ 58.60553023]]), array([[-14.54365133],
       [-14.41334339],
       [ 54.91649119]]), array([[-9.25622858],
       [-9.73031501],
       [52.63833396]]), array([[-20.43461122],
       [-15.20900616],
       [ 53.34331024]]), array([[-17.37367361],
       [-11.22448361],
       [ 52.50109744]]), array([[-23.96197376],
       [-10.68196653],
       [ 48.76065871]]), array([[-17.63697996],
       [ -8.88850214],
       [ 51.3847817 ]]))
```

aruco.calibrateCameraCharuco 所得到之結果

```
aruco_cameraMatrix
 [[888.25457479   0.         482.34262359]
 [  0.         896.25270996 305.57029901]
 [  0.           0.           1.        ]]
aruco_distCoeffs
 [[ 0.08518861]
 [-0.41241448]
 [-0.00235481]
 [-0.00228344]
 [ 1.0810133 ]]
aruco_rvects
 (array([[ 1.85560181],
       [ 2.13559441],
       [-0.43460999]]), array([[ 2.02379095],
       [ 2.16232636],
       [-0.37108907]]), array([[ 1.95358902],
       [ 2.13934218],
       [-0.43973432]]), array([[ 2.07453931],
       [ 2.14265824],
       [-0.2240849 ]]), array([[ 1.84966146],
       [ 2.40870897],
       [-0.12752218]]), array([[ 2.00997394],
       [ 2.25545981],
       [-0.31581337]]), array([[ 2.03659511],
       [ 2.15591552],
       [-0.52993239]]), array([[ 2.12746156],
       [ 2.19254096],
       [-0.09196582]]), array([[ 2.22641063],
       [ 2.13735615],
       [-0.11921934]]), array([[ 2.23525175],
       [ 2.03372319],
       [-0.02535581]]), array([[ 2.23620076],
       [ 2.09170461],
       [-0.1405798 ]]))
```

```
aruco_tvects
 (array([[-12.14589974],
       [-10.64405907],
       [ 56.98317687]]), array([[-13.38017857],
       [-18.22560279],
       [ 59.1431909 ]]), array([[-10.80232673],
       [ -8.92702527],
       [ 59.25512522]]), array([[-18.97385837],
       [-17.05133588],
       [ 60.35444519]]), array([[-11.66868767],
       [-17.99299082],
       [ 58.60553023]]), array([[-14.54365133],
       [-14.41334339],
       [ 54.91649119]]), array([[-9.25622858],
       [-9.73031501],
       [52.63833396]]), array([[-20.43461122],
       [-15.20900616],
       [ 53.34331024]]), array([[-17.37367361],
       [-11.22448361],
       [ 52.50109744]]), array([[-23.96197376],
       [-10.68196653],
       [ 48.76065871]]), array([[-17.63697996],
       [ -8.88850214],
       [ 51.3847817 ]]))
```

機器視覺 hw2:

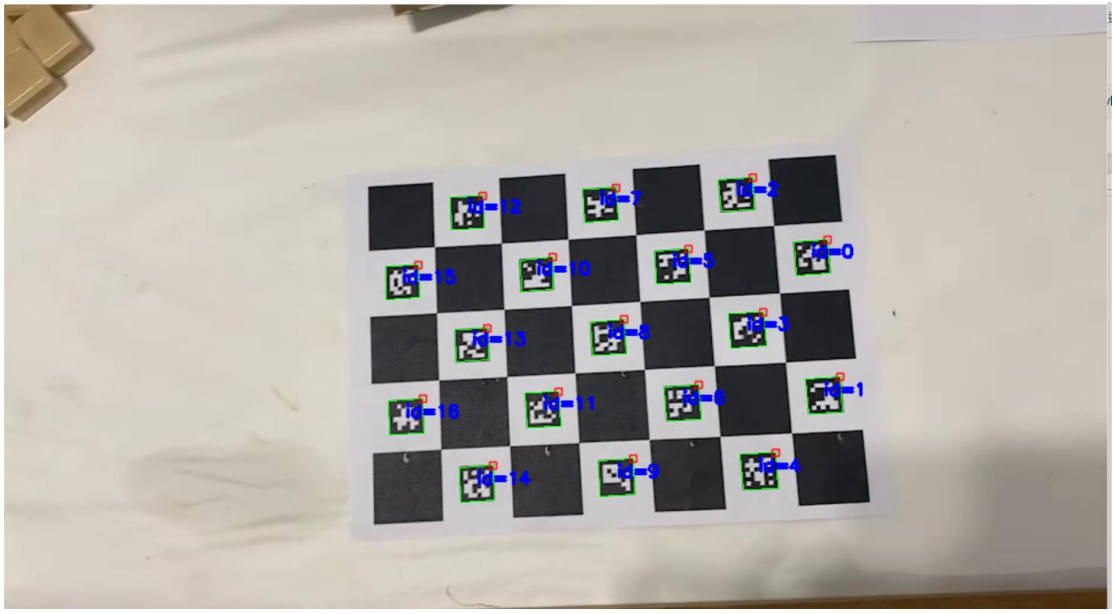
https://youtu.be/SZD4xLsPad8

機器視覺 hw2 _三維投影:

https://youtu.be/JeHse1dBltU

加分題:

(1) 自製棋盤



相機校正得到的參數:

aruco.calibrateCameraAruco 所得到之結果

```
aruco_cameraMatrix
 [[1.02611762e+03 0.00000000e+00 4.50486683e+02]
 [0.00000000e+00 1.02712086e+03 2.50462719e+02]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]
aruco_distCoeffs
 [[ 0.31418055]
 [-2.19776115]
 [-0.00846452]
 [-0.01169103]
 [ 5.56200806]]
aruco_rvects
 (array([[ 2.20769754],
       [ 2.09697985],
       [-0.28837733]]), array([[ 2.14352443],
       [ 2.13472685],
       [-0.20981446]]), array([[ 2.1744785 ],
       [ 2.12359673],
       [-0.1346619 ]]), array([[ 2.17723921],
       [ 2.11362488],
       [-0.14872246]]), array([[ 2.14159608],
       [ 2.11638477],
       [-0.17999863]]), array([[ 2.15664621],
       [ 2.13756111],
       [-0.18296628]]), array([[ 2.16476145],
       [ 2.12310708],
       [-0.14962539]]), array([[ 2.13446906],
       [ 2.13115148],
       [-0.18356212]]), array([[ 2.11285054],
       [ 2.11359504],
       [-0.19885244]]), array([[ 2.25662244],
       [ 2.00250762],
       [-0.22367729]]))
aruco_tvects
 (array([[-9.2437495 ],
       [-6.49439566],
       [71.72418778]]), array([[-13.91594345],
       [ -7.53325354],
       [ 98.08455889]]), array([[-14.53415188],
       [ -8.91781241],
       [ 48.04121761]]), array([[-14.19578761],
       [ -8.04949804],
       [ 47.96436813]]), array([[-14.88437011],
       [ -8.94642087],
       [ 49.76306423]]), array([[-18.90970408],
       [ -9.14556548],
       [ 49.68834575]]), array([[-16.06213656],
       [ -9.83847993],
       [ 49.34890016]]), array([[-15.10470265],
       [-10.59566506],
       [ 50.74065718]]), array([[-13.14879155],
       [ -8.79781816],
       [ 50.98714409]]), array([[-13.29926604],
       [ -7.57276976],
       [ 51.72310165]]))
```

aruco.calibrateCameraCharuco 所得到之結果

```
aruco_cameraMatrix
 [[1.02611762e+03 0.00000000e+00 4.50486683e+02]
 [0.00000000e+00 1.02712086e+03 2.50462719e+02]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]
aruco_distCoeffs
 [[ 0.31418055]
 [-2.19776115]
 [-0.00846452]
 [-0.01169103]
 [ 5.56200806]]
aruco_rvects
 (array([[ 2.20769754],
       [ 2.09697985],
       [-0.28837733]]), array([[ 2.14352443],
       [ 2.13472685],
       [-0.20981446]]), array([[ 2.1744785 ],
       [ 2.12359673],
       [-0.1346619 ]]), array([[ 2.17723921],
       [ 2.11362488],
       [-0.14872246]]), array([[ 2.14159608],
       [ 2.11638477],
       [-0.17999863]]), array([[ 2.15664621],
       [ 2.13756111],
       [-0.18296628]]), array([[ 2.16476145],
       [ 2.12310708],
       [-0.14962539]]), array([[ 2.13446906],
       [ 2.13115148],
       [-0.18356212]]), array([[ 2.11285054],
       [ 2.11359504],
       [-0.19885244]]), array([[ 2.25662244],
       [ 2.00250762],
       [-0.22367729]]))
aruco_tvects
 (array([[-9.2437495 ],
       [-6.49439566],
       [71.72418778]]), array([[-13.91594345],
       [ -7.53325354],
       [ 98.08455889]]), array([[-14.53415188],
       [ -8.91781241],
       [ 48.04121761]]), array([[-14.19578761],
       [ -8.04949804],
       [ 47.96436813]]), array([[-14.88437011],
       [ -8.94642087],
       [ 49.76306423]]), array([[-18.90970408],
       [ -9.14556548],
       [ 49.68834575]]), array([[-16.06213656],
       [ -9.83847993],
       [ 49.34890016]]), array([[-15.10470265],
       [-10.59566506],
       [ 50.74065718]]), array([[-13.14879155],
       [ -8.79781816],
       [ 50.98714409]]), array([[-13.29926604],
       [ -7.57276976],
       [ 51.72310165]]))
```

機器視覺 hw2_bnous:

https://youtu.be/V8S8HGHhWSE

機器視覺 hw2_三維投影_bnous: (這部影片有點久，可能是電腦效能的問題)

https://youtu.be/wH3FdWlBvm4

結果評述:

在老師給的影片中跟我的影片中做法的差異主要只在於有無二值化的部分，而我認為我的影片不需要二值化是因為他的採光還不錯，沒有陰影的部分，所以不需要進行二值化操作。而兩部影片的拍攝手法是差不多的，都是上下左右跟斜角晃動的方式。

而在三維投影時影片有時候會跟出現跟原本預想的(左低右高)不一樣的情況，這是因為偵測到他的姿態軸有晃動的原因。

結論

在實作的過程中發現自己還是對 aruco marker 很不熟悉，在一開始時，有查到資料說偵測到的 marker 點會是以左上、右上、右下、左下的方式排，但一開始誤以為是以座標為基準順時針排而非以 marker 為基準，導致自己一直不懂為何出來的結果是錯的(有過圖片正確但上下顛倒或是扭曲之類的)，而在那時我是以 flip 的方式將每一幀的圖片翻轉，但其實我所要做的事情只是交換所找到的點的順序即可。而這個過程也加深了我對 aruco marker 的印象。在實作中也發現如果找到的點的順序如果放置錯誤則會發生變形，所以在實作時需要確定自己所找到的點是如何排列，而若是單純的影片顯示角度不對則只是因為 marker 的左上角所在的位置未必是我們期望影片左上角所在之位置。

一開始的時候我誤以為 proj_pt_with_dist 跟 corner 的作用差不多，卻忘了其實最根本的原因是 markerCorners3D 的數值，後來跟同學討論後才想到一開始會一樣只是因為我的矩陣將 Z 軸數值都設為 0 的關係，而在調整數值時一開始也有一些小混亂，忘記調整角度要看的是姿態軸而非一般所理解的 xyz 座標。

而這次的作業也讓我更了解如何透過 ArUco 標記來擴增實境以及這些函式的使用方式。

在錄製的時候也因為秒數太久而重錄了好多次(秒數好像跟我的電腦當前校能有關)，但最後一部影片的秒數因為不管麼樣都壓不下來所以只好直接上傳了。

參考文獻

【OpenCV】25 - 綜合運用 4，用 OpenCV 來把圖片 p 到各種奇怪的地方吧！ 透視投影 cv2.warpPerspective, merge two images

https://www.wongwonggoods.com/all-posts/python/python_opencv/opencv-warpperspective/