OPENCV ADVANCED

OPENCV 進階應用

秦語萱

ENVIRONMENT

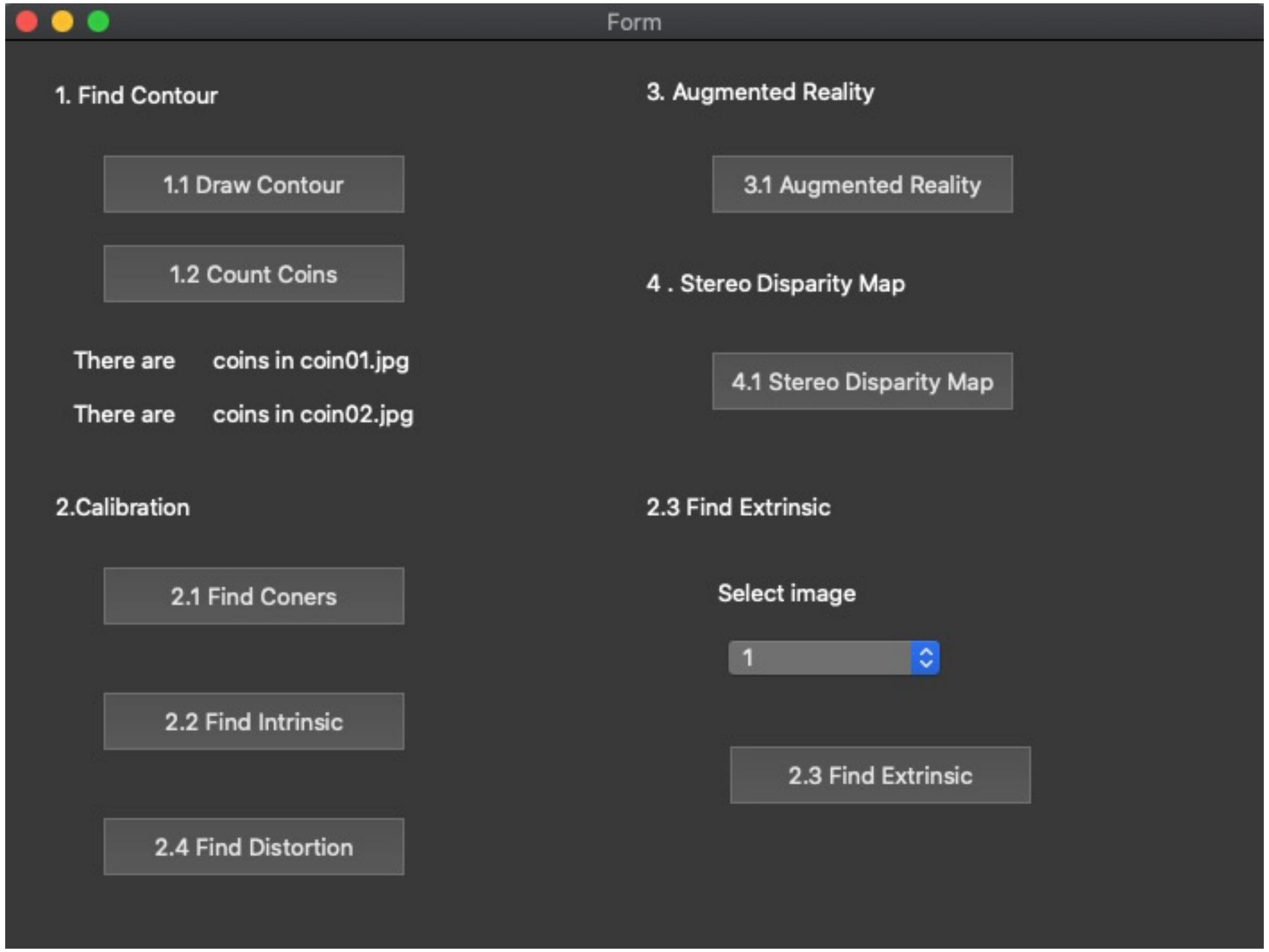
- Python
  - Python 3.7
  - opencv-contrib-python (3.4.2.17) cv2
  - Matplotlib 3.1.1, numpy
  - UI framework: pyqt5 (5.15.1)

ENVIRONMENT

- In Main.py but can’t execute
- 詳細project 連同 gui檔案（需要import與安裝上一頁環境參數）
- 在 https://github.com/chinyu1229/opencv_stereo

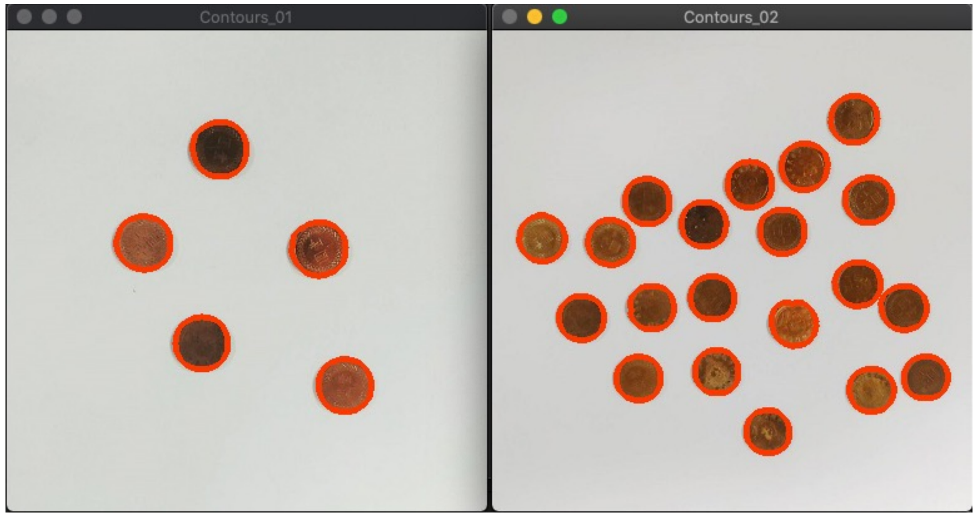
GUI & FUNCTION

1. Find Contour
    - 1.1 Draw Contour
    - 1.2 Count Coins
2. Camera Calibration
    - 2.1 Corner detection
    - 2.2 Find the intrinsic matrix
    - 2.3 Find the extrinsic matrix
    - 2.4 Find the distortion matrix
3. Augmented Reality
4. Stereo Disparity Map
    - 4.1 Compute disparity map
    - 4.2 Calculate the depth



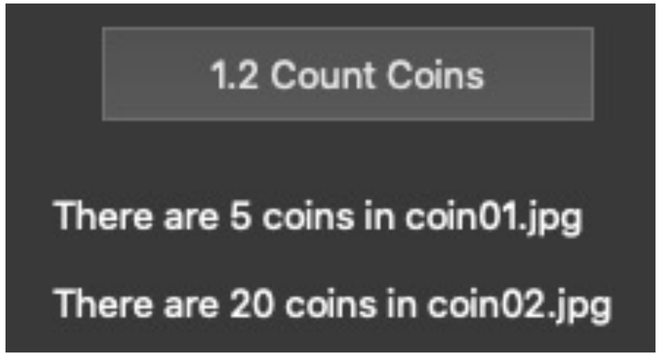
1.1 Find Contour – Draw Contour

- Given: two color images
- **Draw Contour**: Using OpenCV functions to find the contours of coins in two images.
- step1: RGB -> Grayscale -> binary
- step2: use Gaussian blur to remove the noise
- step3: using edge detection functions(cv2.Canny()) to get better results
- step4: using cv2.findContours() & cv2.drawContours then show the results



1.2 **Find Contour – Count Coins**

- Given: two color images
- **Draw Contour**: Using OpenCV functions to find how many coins in two images
- step1: RGB -> Grayscale -> binary
- step2: use Gaussian blur to remove the noise
- step3: using edge detection functions(cv2.Canny()) to get better results
- step4: using cv2.findContours() which return value is num of coins

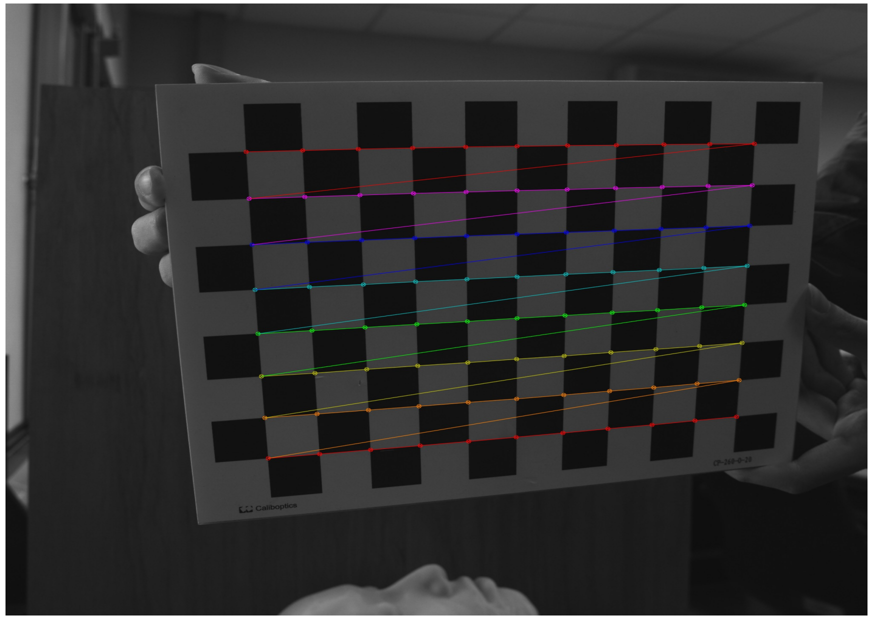


```
There are 5 coins in coin01.jpg

There are 20 coins in coin02.jpg
```

2.1 CORNER DETECTION

- Given : 15 images, 1.bmp ~ 15.bmp
- Find and draw the corners on the chessboard for each image
- step1 : using cv2.findChessboardCorners to find chessboard corners and return 'ret' value
- step2 : using cv2.drawChessboardConers to show 1~15 images



2.2 FIND THE INTRINSIC MATRIX

```
ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(objpoints, imgpoints, gray.
shape[::-1], None, None)
```

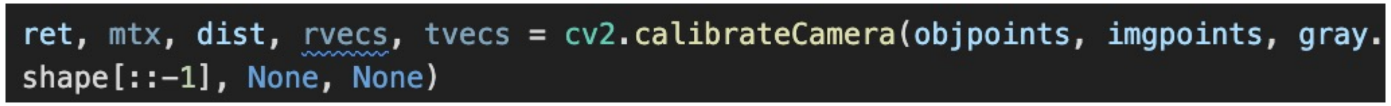
- Given : 15 images, 1.bmp ~ 15.bmp
- Find the intrinsic matrix
- cv2.calibrateCamera to print intrinsic matrix (return value is mtx)

Camera Calibration (相機校正) 是利用多個已知的世界座標跟其對映的影像座標的點，來求該 Camera 的內部參數矩陣(Intrinsic Matrix) 及外部參數矩陣(Extrinsic Matrix)
理論基礎為：世界座標(3D)經過相機外部參數矩陣的作用轉換成相機座標(3D)，而相機座標(3D) 再經過相機內部參數矩陣的投影作用轉換成影像座標(2D)。
根據針孔相機成像成理，假設 Q 點的世界座標為 (X,Y, Z)，而影像座標的座標為 (u, v)，可以得到以下相機矩陣關係：

$$s\underbrace{\begin{bmatrix} u \\ v \\ 1 \end{bmatrix}}_{\text{成像平面座標}} = \underbrace{\begin{bmatrix} \alpha_x & s & u_o \\ 0 & \alpha_y & v_o \\ 0 & 0 & 1 \end{bmatrix}}_{\text{內部參數矩陣}} \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}}_{\text{外部參數矩陣}} \underbrace{\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}}_{\text{世界座標}} = K\left[R \,\middle|\, t\right]Q.$$

2.3 FIND THE EXTRINSIC MATRIX

- Given: intrinsic parameters, distortion coefficients, and the list of 15 images
- Find the extrinsic matrix of the chessboard for each of the 15 images
- step1 : cv2.calibrateCamera to find given parameters
- step2 : use rotation matrix(return value is rvecs) & transformation matrix (return value is tvecs) to calculate the extrinsic matrix

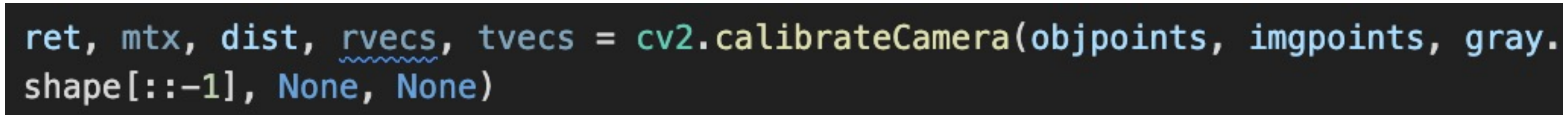


```
ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(objpoints, imgpoints, gray.
shape[::-1], None, None)
```

$$s \underset{\text{成像平面座標}}{\begin{bmatrix} u \\ v \\ 1 \end{bmatrix}} = \underset{\text{內部參數矩陣}}{\begin{bmatrix} \alpha_x & s & u_o \\ 0 & \alpha_y & v_o \\ 0 & 0 & 1 \end{bmatrix}} \underset{\text{外部參數矩陣}}{\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}} \underset{\text{世界座標}}{\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}} = K \begin{bmatrix} R \mid t \end{bmatrix} Q.$$

2.4 FIND THE DISTORTION MATRIX

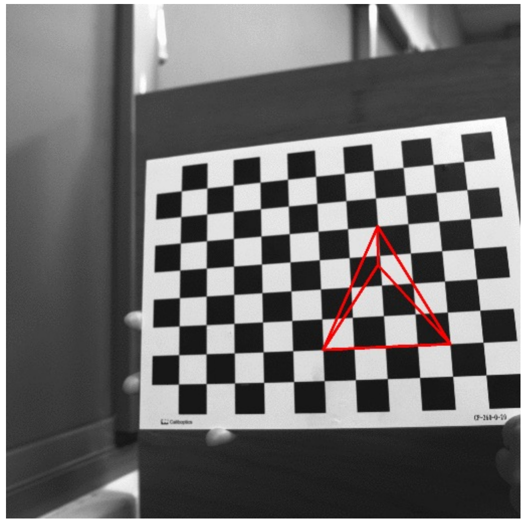
- Given: 15 images
- Find the distortion matrix $[k_1, k_2, p_1, p_2, k_3]$
- cv2.calibrateCamera to print distortion matrix (return value is dist)



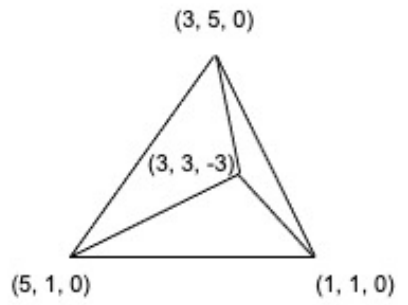
```
ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(objpoints, imgpoints, gray.
shape[::-1], None, None)
```

3. AUGMENTED REALITY

- Given: 15 images
  - 1) Calibrate 5 images to get intrinsic, distortion and extrinsic parameters
  - 2) Draw a "pyramid" on the chessboards images(1.bmp to 5.bmp)
  - 3) Click the button to show the tetrahedron on the picture. Show each picture 0.5 seconds (total 5 images)
- Set pyramid vetex then using cv2.drawContours() & cv2.line() to draw pyramids



- 3D Object coordinates:
  Vertex (3, 3, -3)
  Corners(1, 1, 0)(3, 5, 0)(5, 1, 0)



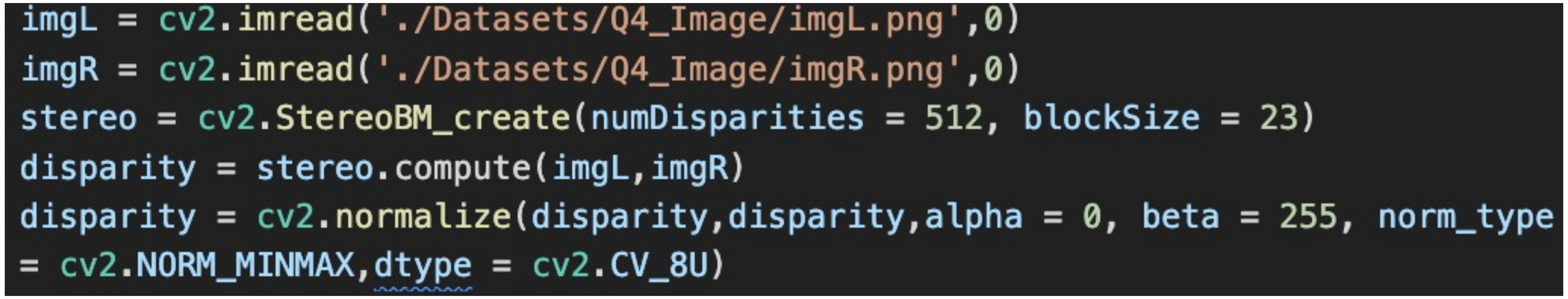
4. STEREO DISPARITY MAP

- Given: a pair of images L.png & R.png
  - 4.1 compute the disparity map/image based on Left and Right stereo images
  - 4.2 Select a point on disparity map from 4.1, calculate the depth and show both the disparity value and the depth on the image

    當採取兩個同一水平線上的攝像頭進行拍攝的時候，同一物體將在兩個攝像機內被拍攝到，在兩個攝像機內部，這個物體相對於攝像機中心點位置有不同的座標。left是該物體在左攝像機內相對位置，right是該物體在右攝像機內相對位置。
    兩個攝像機相距S，焦距為f，物體P距離攝像機z，z也就是景深。當我們將兩幅影象重疊在一起的時候，左攝像機上P的投影和右攝像機上P的投影位置有一個距離|Xleft| |Xright|，這個距離稱為Disparity。
    根據相似三角形可以得到z=sf/d，計算d的值的過程中需要對兩幅影象進行匹配，尋找到物體P在兩幅影象中的相對位置。在對影象進行匹配的過程中，需要用到cost computation，即通過尋找同一水平線上兩幅影象上的點的最小誤差來確定這兩個點是否是同一個物體所成的像。

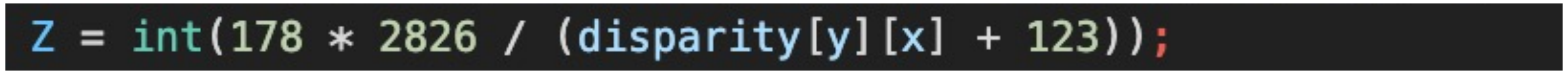
4. STEREO DISPARITY MAP(CONT.)

- basic algorithm: calculate disparity



```
imgL = cv2.imread('./Datasets/Q4_Image/imgL.png',0)
imgR = cv2.imread('./Datasets/Q4_Image/imgR.png',0)
stereo = cv2.StereoBM_create(numDisparities = 512, blockSize = 23)
disparity = stereo.compute(imgL,imgR)
disparity = cv2.normalize(disparity,disparity,alpha = 0, beta = 255, norm_type
= cv2.NORM_MINMAX,dtype = cv2.CV_8U)
```

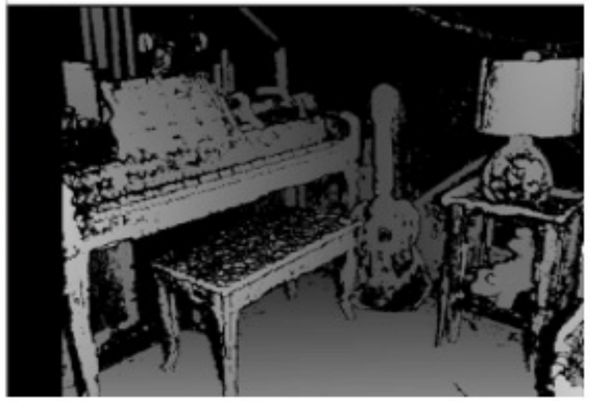
- calculate depth using z = sf / d, s is given 178, f is given 2826



```
Z = int(178 * 2826 / (disparity[y][x] + 123));
```

imgL.png

imgR.png


Result Video

THE END

THANK YOU!