

計劃名稱：LED 頻率辨識與偵測

撰寫人：陳鎮華

最後更新：2019/10/12

目錄

系統簡介.....	3
系統環境.....	3
系統流程.....	3
系統架構.....	4
演算法分析.....	7
相關連結.....	9

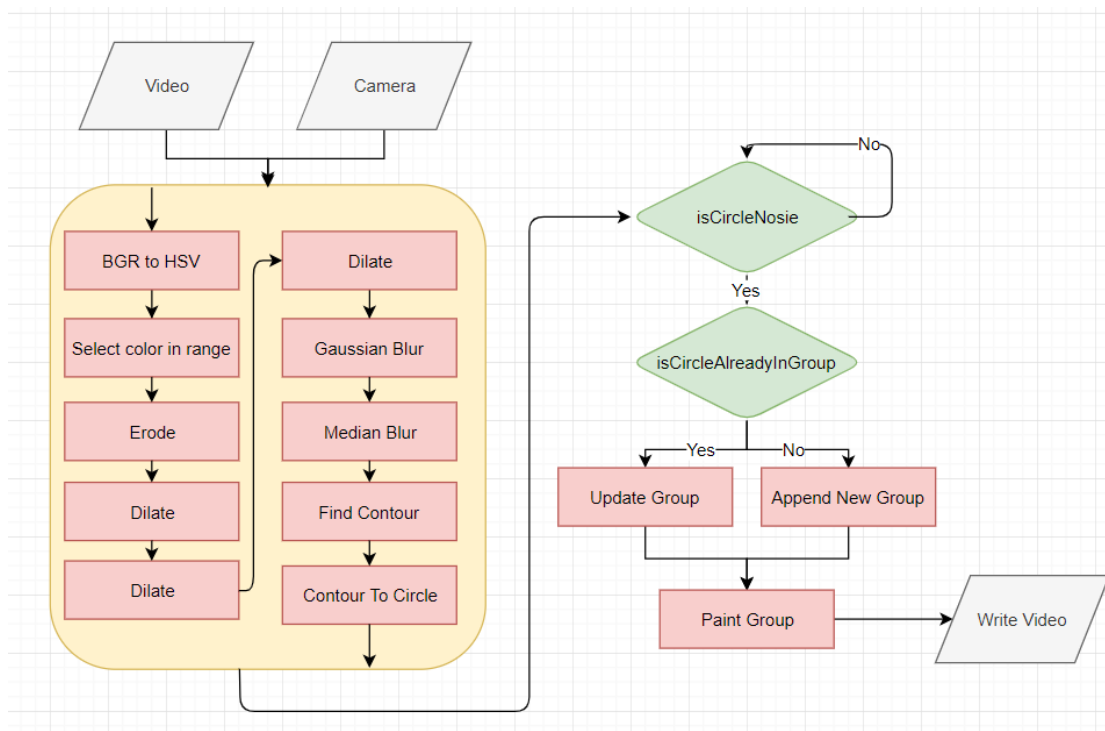
系統簡介

本程式的功能主要用於 LED 頻率之辨識，運用 OpenCV 影像辨識與處理技巧與 Python 程式語言進行撰寫。目的用於辨識魚體身上之 LED 燈光頻率，目前可辨識紅、藍、綠三種顏色之燈光及複數目標之辨識，並產生頻率資料之表格(.csv)及處理過後的影片資料(.mp4)。使用者可依需求透過使用者介面在程式執行時調整辨識之各參數(HSV 範圍、Group 範圍)，也可在程式碼頂端調整起始參數。目前演算法對於移動速度快、Group 互相交疊以及顏色範圍不夠明確(紅色)之魚體辨識能力較弱。

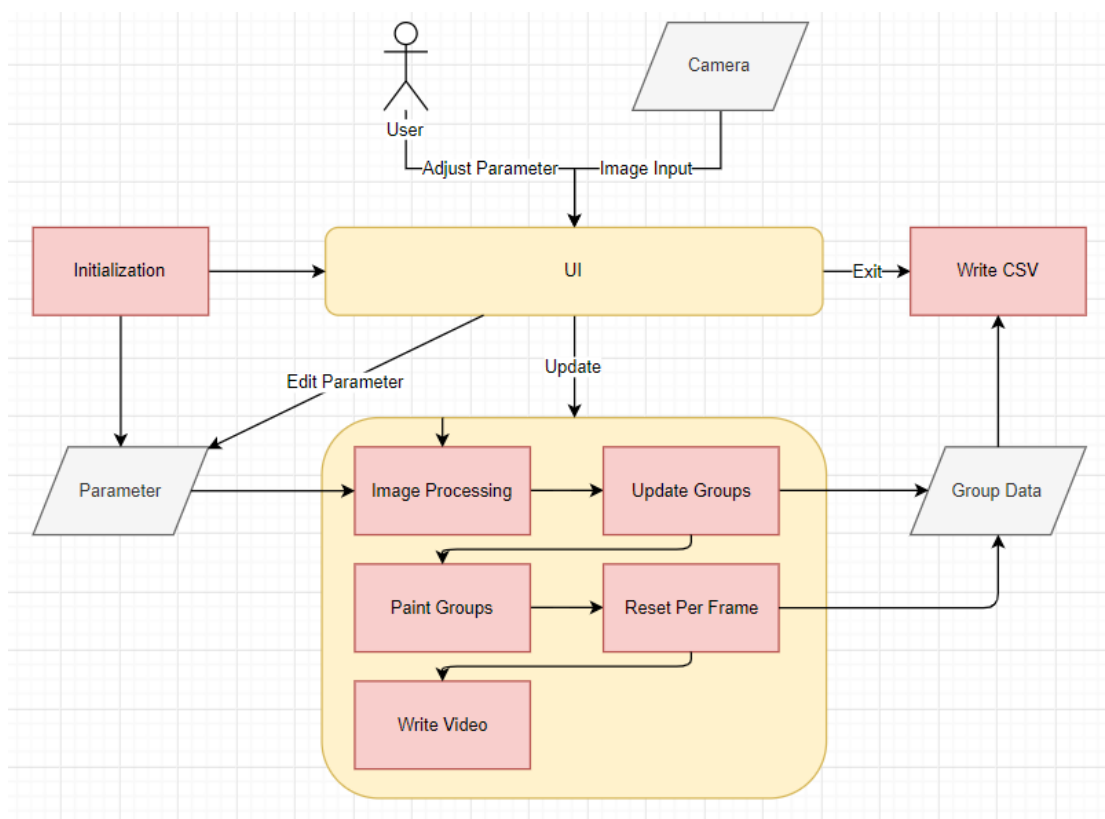
系統環境

- 語言：Python 3.6.7
- 函式庫：OpenCV 4.0.1
- 作業系統：Window 10

系統流程



系統架構



1. Initialization

初始化 UI 與影像處理參數

相關函式: `setTrackbar`、程式碼最上方的參數調整

```
LOWER = {'red':np.array([0, 180, 180]), 'green':np.array([30, 180, 180]), 'blue':np.array([90, 180, 180])}
UPPER = {'red':np.array([25, 255, 255]), 'green':np.array([80, 255, 255]), 'blue':np.array([130, 255, 255])}
COLOR = {'red':(0, 0, 255), 'green':(0, 255, 0), 'blue':(255, 0, 0)}
init_S_Lower = 180
init_S_Upper = 255
init_V_Lower = 180
init_V_Upper = 255
NOISE_RADIUS = 10
GROUP_RADIUS = 100
TIME_ERROR = 0.1
FPS = 20.0
inwidth = 640
inHeight = 480
```

LOWER 影像 HSV 之下限

UPPER 影像 HSV 之上限

COLOR 繪製影像之 BGR 顏色

init_S_Lower/Upper 初始 S 之下限/上限

init_V_Lower/Upper 初始 V 之下限/上限

NOISE_RADIUS 雜訊之半徑(小於此大小將會被判定為雜訊)

GROUP_RADIUS Group 之半徑(在此大小內將被視為同一 Group)

TIME_ERROR	時間容忍之誤差範圍
FPS	影片幀數
inWidth/inHeight	輸入影像之長寬

2. Input

Camera 影像輸入可透過參數調整

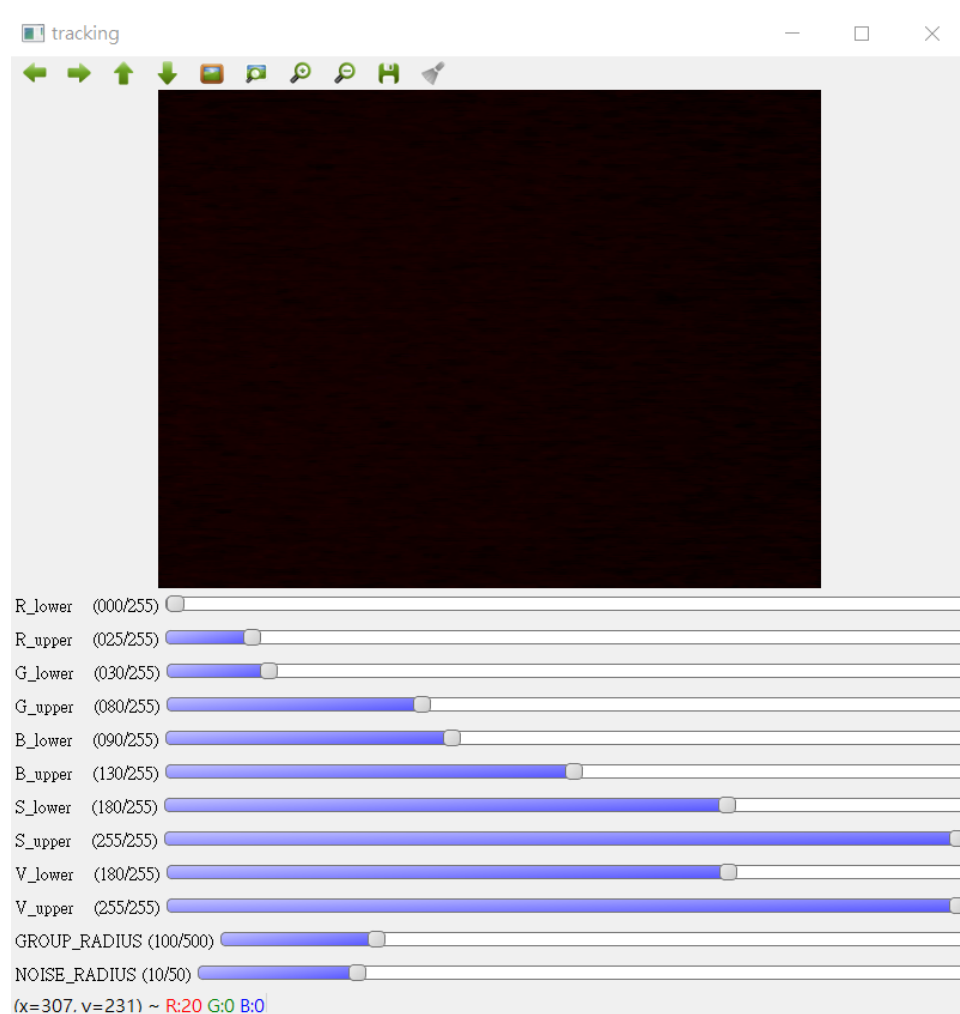
相關函式：main 部分的 cap

```

219  #-----main-----#
220  setTracker()
221  groups = []
222  cap = cv2.VideoCapture(0)
223  start_time_point = time.time()
224  now = time.time()
225
226  fourcc = cv2.VideoWriter_fourcc('m', 'p', '4', 'v')
227  output_video = cv2.VideoWriter('output.mp4', fourcc, FPS, (inWidth, inHeight))
228

```

3. UI



可透過 UI 調整影像處理之參數以及處理過後之影像即時繪製

相關函式：modifyParameter

4. Update

A. Image Processing

透過 OpenCV 進行影像處理，詳細演算法在演算法分析部分說明
相關函式：updateGroups, selectColor

B. Update Groups

處理完的影像資料與 Groups 進行比較並更新以及辨識出部分影像繪製
相關函式：updateGroups, contour2Group, matchGroup, contourInGroup, updateColorInGroup

C. Paint Groups

將每個 Group 繪製上影像
相關函式：paintGroups, isGroupLEDLightOn, calCenter

D. Reset Per Frame

確認 Group 內的 LED 是否持續亮著並調整重製
相關函式：resetGroups

E. Write Video

將處理完的影像寫成 mp4 檔
相關函式：writeVideo

5. Write CSV

將 Group 內的資料寫成 csv 表格
相關函式：writeFile

演算法分析

A. selectColor

```
86 def selectColor(img, color):
87     img = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
88     mask = np.ones((5, 5), np.uint8)
89     thresh = cv2.inRange(img, LOWER[color], UPPER[color])
90     thresh = cv2.erode(thresh, mask)
91     thresh = cv2.dilate(thresh, mask)
92     thresh = cv2.dilate(thresh, mask)
93     thresh = cv2.dilate(thresh, mask)
94     thresh = cv2.GaussianBlur(thresh, (5, 5), 0)
95     thresh = cv2.medianBlur(thresh, 5)
96
97     return thresh
```

將影像按照以下順序進行處理

cv2.cvtColor	從 BGR 色彩空間轉為 HSV 色彩空間
cv2.inRange	將顏色參數之範圍內的影像濾出
cv2.erode	影像處理之侵蝕運算(消除雜點)
cv2.dilate	影像處理之膨脹運算(將影像破洞補起)
cv2.GaussianBlur	影像處理之高斯模糊化(使影像較接近真實影像)
cv2.medianBlur	影像處理之中位數模糊化(使影像偏差降低)

B. updateGroups

```
def updateGroups(frame, groups):
    for color, value in UPPER.items():
        img = selectColor(frame.copy(), color)
        contours, hierarchy = cv2.findContours(img, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

        for contour in contours:
            center, radius = contour2Group(contour)
            if radius > NOISE_RADIUS:
                matchGroup(center, color, groups)
                cv2.circle(frame, (int(center[0]), int(center[1])), int(radius), COLOR[color], 3)
                cv2.rectangle(frame, \
                    (int(center[0] - radius), int(center[1] - radius)), \
                    (int(center[0] + radius), int(center[1] + radius)), \
                    (255, 255, 255), 5)

    return groups
```

cv2.findContours 影像處理找出畫面中的凸包

for 迴圈將所有 contour 與 group 進行比較並更新與繪製

C. contour2Group

```
98
99     def contour2Group(contour):
100         center, radius = cv2.minEnclosingCircle(contour)
101
102         return center, radius
103
```

cv2.minEnclosingCircle 影像處理透過凸包找出近似之圓形

D. matchGroup

```
130     def matchGroup(center, color, groups):
131         groupExist = False
132
133         for group in groups:
134             if contourInGroup(center, group):
135                 groupExist = True
136                 group = updateColorInGroup(center, color, group)
137                 break
138
139         if groupExist == False:
140             groups.append({color:LED(True, center, now)})
```

將所有各個 center 與 group 交叉比較、分類並決定更新或新增 group

E. contourInGroup

```
112
113     def contourInGroup(center, group):
114         if cv2.norm(center, calCenter(group)) < GROUP_RADIUS:
115             return True
116         else:
117             return False
118
```

cv2.norm 凸包與 group 計算距離決定是否歸類為同一 group

相關連結

Github: https://github.com/shizsun0609tw/LED_Recognizaion

OpenCV: <https://reurl.cc/e5WnVx>

凸包: <https://zh.wikipedia.org/wiki/%E5%87%B8%E5%8C%85>

侵蝕與膨脹: <https://reurl.cc/72jnbb>

高斯模糊: <https://reurl.cc/K6Xq0g>

中值模糊: https://en.wikipedia.org/wiki/Median_filter

Draw.io: <https://www.draw.io/>