Chapter 4：進階影像辨識

本章重點

在樹莓派中實作進階影像辨識。

準備材料

可上網的電腦

樹莓派（含相機模組與外殼）

學習目標

1. OpenCV × 物體辨識。
2. MediaPipe 機器學習框架。
3. CVZone 電腦視覺套件。

4-1. OpenCV × 物體辨識

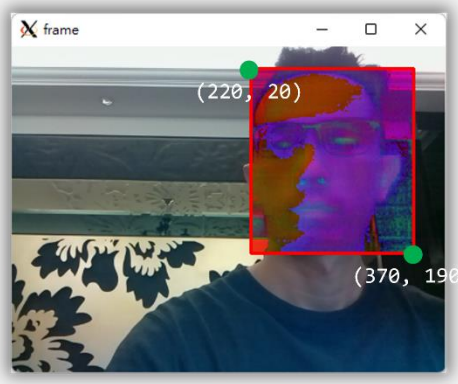
OpenCV 進行指定範圍的區域處理

ROI

只處理有興趣的區域 (Region of Interest, ROI)。例如：針對有人進出『大門』這個區域，進行影像辨識就好，其餘就不管。OpenCV 的影像資料為矩陣型態，所以透過矩陣計算就可輕易計算出 ROI。

4-1 指定一個區域處理。

```
import cv2

ESC = 27

rect = ((220, 20), (370, 190))    # 設定 ROI 的座標範圍
(left, top), (right, bottom) = rect

# 自訂一個取出子畫面的函數
def roiarea(frame):
    return frame[top:bottom, left:right]

# 自訂一個將 ROI 區域的資料貼回原本畫面的函數
def replaceroi(frame, roi):
    frame[top:bottom, left:right] = roi
    return frame

cap = cv2.VideoCapture(0)
ratio = cap.get(cv2.CAP_PROP_FRAME_WIDTH) /
cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
WIDTH = 400
HEIGHT = int(WIDTH / ratio)

while True:
    ret, frame = cap.read()
    frame = cv2.resize(frame, (WIDTH, HEIGHT))

    # 取出子畫面
    roi = roiarea(frame)
    roi = cv2.cvtColor(roi, cv2.COLOR_BGR2HSV)
```



```
    # 將處理完的子畫面貼回到原本畫面中
    frame = replaceroi(frame, roi)

    # 在 ROI 範圍處畫個框
    cv2.rectangle(frame, rect[0], rect[1], (0, 0, 255), 2)

    cv2.imshow('frame', frame)

    if cv2.waitKey(1) == ESC:
        break

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

物體移動追蹤

指定區域的物體移動追蹤

移動追蹤演算法：先設定好想要追蹤物體在畫面上的座標，然後當該物體移動到新地方時，OpenCV 會自動計算出移動後的座標。

- 比每一個畫面都用物件辨識演算法 (例如：類神經網路)，去找物件所在位置要快多了。
- OpenCV 提供了 8 種不同的演算法：Boosting、CSRT、GOTURN、KCF、 MedianFlow、MIL、MOSSE、TLD。

selectROI 函式：藉由這個函數在畫面上拖曳出一個矩形範圍，然後就可以得出這個矩形在畫面上的座標值了。

4-2 指定一個區域處理。

```
import cv2

ESC = 27

cap = cv2.VideoCapture('images/vtest.avi')
tracker = cv2.TrackerCSRT_create()
roi = None

while True:
    ret, frame = cap.read()
    if roi is None:
        roi = cv2.selectROI('frame', frame)
```

```
        if roi != (0, 0, 0, 0):
            tracker.init(frame, roi)
    success, rect = tracker.update(frame)

    if success:
        (x, y, w, h) = [int(i) for i in rect]
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

    cv2.imshow('frame', frame)

    if cv2.waitKey(1) == ESC:
        break

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

即時影像的物體移動追蹤

加入 OpenCV 即時影像，同時進行物體移動追蹤。

4-3 加入即時影像與移動追蹤。

```
import cv2

ESC = 27

# 加入即時影像的設定
cap = cv2.VideoCapture(0)
ratio = cap.get(cv2.CAP_PROP_FRAME_WIDTH) /
cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
WIDTH = 400
HEIGHT = int(WIDTH / ratio)

tracker = cv2.TrackerCSRT_create()
roi = None

while True:
    ret, frame = cap.read()
    frame = cv2.resize(frame, (WIDTH, HEIGHT))
    if roi is None:
        roi = cv2.selectROI('frame', frame)
        if roi != (0, 0, 0, 0):
            tracker.init(frame, roi)
    success, rect = tracker.update(frame)
```

```
    if success:
        (x, y, w, h) = [int(i) for i in rect]
        cv2.rectangle(frame, (x,y), (x + w, y + h), (0, 255, 0), 2)

    cv2.imshow('frame', frame)

    if cv2.waitKey(1) == ESC:
        break

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

4-2. MediaPipe 機器學習框架

Google MediaPipe

MediaPipe 是 Google 公司在 2019 年發表的開放原始碼專案，此專案針對即時串流媒體和電腦視覺 (Computer Vision)，提供開放原始碼且跨平台的機器學習解決方案，這個解決方案就是機器學習管線 (ML Pipeline)。

基本上，Google MediaPipe 是一種圖表基礎系統 (Diagram Based System)，可以用來建構多模式影片、聲音和感測器等應用的機器學習管線，我們可以使用圖形方式來組織模組元件，例如：TensorFlow 或 TensorFlow Lite 推論模型和多媒體處理函數等，來建構出一個擁有感知功能的機器學習管線，能夠即時從媒體中辨識出人臉、手勢和姿勢等感知功能，其官方網址如下所示：

官網 https://mediapipe.dev/

在 Windows 安裝 MediaPipe

新增一個 Python 3.9 的虛擬環境 mediapipe，務必先安裝 OpenCV 後 (參見 Ch1)，繼續安裝 Google MediaPipe：

離開目前的虛擬環境，例如：opencvenv_39。

```
(opencvenv_39) >>> deactivate
```

MediaPipe 也用會到 OpenCV 相同的模組，所以從已建立的虛擬環境 opencv，複製出一個虛擬環境 mediapipe，再進行後續安裝：

```
>>> conda create -n mpenv_39 --clone opencvenv_39
```

載入虛擬環境 mpenv_39：

```
>>> activate mpenv_39
```

安裝 MediaPipe：

```
(mpenv_39) >>> pip install mediapipe
```

若發生有關 .DLL 的錯誤，安裝 Microsoft Visual C++ runtime DLLs：

```
(mpenv_39) >>> pip install msvc-runtime
```

在樹莓派安裝 MediaPipe

新增一個 Python 3 的虛擬環境 mediapipe，務必先安裝 OpenCV 後 (參見 Ch3)，繼續安裝 Google MediaPipe，因為 MediaPipe 版本的新舊並不相容，請安裝本次使用的 0.8.4.0 版，如下所示：

離開目前的虛擬環境，例如：opencv。

```
(opencv) ~$ deactivate
```

MediaPipe 也會到 OpenCV 相同的模組，所以從已建立的虛擬環境 opencv，複製出一個虛擬環境 mediapipe，再進行後續安裝：

```
~$ cpvirtualenv opencv mediapipe
```

在樹莓派 4 安裝 MediaPipe 的指令：

```
(mediapipe) ~$ pip install mediapipe-rpi4==0.8.4.0
```

若是在樹莓派 3 安裝 MediaPipe 的指令：

```
(mediapipe) ~$ pip install mediapipe-rpi3==0.8.4.0
```

目前最新版本 MediaPipe Python 套件會發生相依套件的版本問題。若匯入 mediapipe 模組有問題，需對 protobuf 進行降版處理。解決方法：移除自動安裝的 protobuf 套件後重新安裝 3.20.* 版本即可解決。

```
(mediapipe) ~$ pip uninstall protobuf
```

```
(mediapipe) ~$ pip install protobuf==3.20.*
```

MediaPipe 人臉偵測

MediaPipe 偵測臉部六個關鍵點

MediaPipe 人臉偵測 (Face Detection) 是使用 Blazeface 模型的一種超快速的人臉偵測，Blazeface 模型是 Google 開發的一種快速和輕量級的人臉辨識模型，可以在圖片中辨識出多張人臉和標示臉部 6 個關鍵點 (Key Points)，這是使用 Single Shot Detector 架構和客製化編碼器所建立的人臉辨識模型。

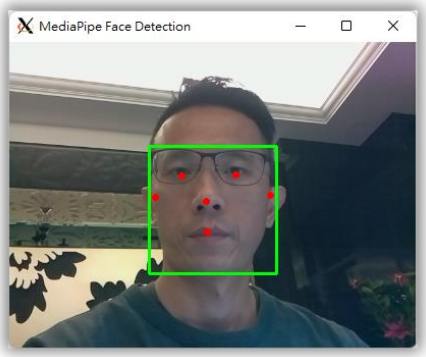
MediaPipe 人臉偵測所辨識出的臉部可以回傳臉部範圍的方框座標，再加上左眼、右眼、鼻尖、嘴巴、左耳和右耳共 6 個關鍵點座標。

4-4 MediaPipe 人臉偵測與臉部六個關鍵點。

```
import cv2
import mediapipe as mp

ESC = 27

mp_face_detection = mp.solutions.face_detection
mp_drawing = mp.solutions.drawing_utils

cap = cv2.VideoCapture(0)
ratio = cap.get(cv2.CAP_PROP_FRAME_WIDTH) /
cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
WIDTH = 400
HEIGHT = int(WIDTH / ratio)
face_detection =
mp_face_detection.FaceDetection(min_detection_confidence = 0.5)

while cap.isOpened():
    ret, frame = cap.read()
    frame = cv2.resize(frame, (WIDTH, HEIGHT))
    frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    frame.flags.writeable = False
    results = face_detection.process(frame)
    frame.flags.writeable = True
    frame = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)

    if results.detections:
        for detection in results.detections:
            mp_drawing.draw_detection(frame, detection)
```

```
    cv2.imshow('MediaPipe Face Detection', frame)

    if cv2.waitKey(1) == ESC:
        break

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

輸出結果：



MediaPipe 臉部網格

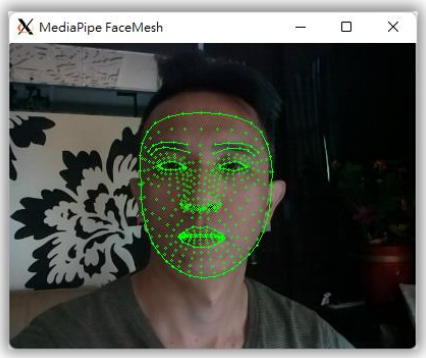
MediaPipe 臉部網格 (MediaPipe Face Mesh) 是使用 Blazeface 模型為基礎，可以預測出 468 個關鍵點，和使用網格來繪出 3D 臉部模型。

4-5 MediaPipe 臉部網格。

```
import cv2
import mediapipe as mp

ESC = 27

mp_face_mesh = mp.solutions.face_mesh
mp_drawing = mp.solutions.drawing_utils

cap = cv2.VideoCapture(0)

with mp_face_mesh.FaceMesh(min_detection_confidence = 0.5,
                           min_tracking_confidence = 0.5) as face_mesh:
    while cap.isOpened():
        ret, frame = cap.read()

        if not ret:
            break
```

```
        rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

        results = face_mesh.process(rgb_frame)

        if results.multi_face_landmarks:
            for face_landmarks in results.multi_face_landmarks:
                mp_drawing.draw_landmarks(image=frame,
                                          landmark_list=face_landmarks,
                                connections=mp_face_mesh.FACEMESH_TESSELATION)

        cv2.imshow('Face Mesh', frame)

        if cv2.waitKey(1) == ESC:
            break

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

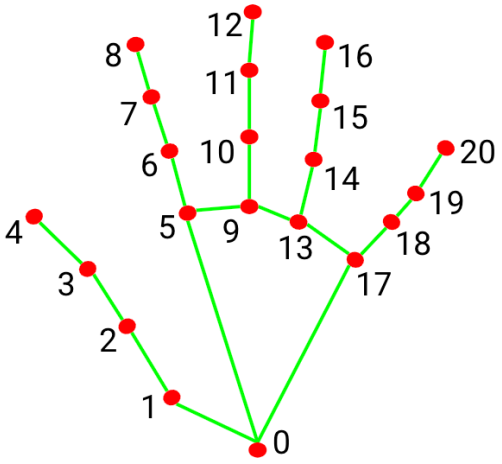
輸出結果：



- Python 程式在使用 OpenCV 讀取影像後，使用 MediaPipe 臉部網格進行人臉辨識和繪出 3D 臉部模型，程式執行的結果可以看到綠色線標示出的 3D 臉部網格。

MediaPipe 多手勢追蹤

MediaPipe 手勢 (MediaPipe Hands) 是使用手掌偵測模型 (Palm Detection Model) 進行多手勢追蹤，首先偵測出手掌和拳頭，然後使用手部地標模型 (Hand Landmark Model) 偵測出手部的 21 個關鍵點，如下圖所示：



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

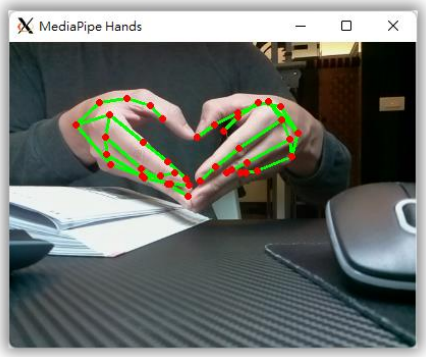
官網 https://google.github.io/mediapipe/solutions/hands.html

4-6 MediaPipe 多手勢追蹤。

```
import cv2
import mediapipe as mp

ESC = 27

mp_drawing = mp.solutions.drawing_utils
mp_hands = mp.solutions.hands

cap = cv2.VideoCapture(0)
ratio = cap.get(cv2.CAP_PROP_FRAME_WIDTH) /
cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
WIDTH = 400
HEIGHT = int(WIDTH / ratio)

hands = mp_hands.Hands(min_detection_confidence = 0.5,
                                        min_tracking_confidence = 0.5)

while cap.isOpened():
    ret, frame = cap.read()
    frame = cv2.resize(frame, (WIDTH, HEIGHT))
    frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    frame.flags.writeable = False
```

```
    results = hands.process(frame)
    frame.flags.writeable = True
    frame = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)

    if results.multi_hand_landmarks:
        for hand_landmarks in results.multi_hand_landmarks:
            mp_drawing.draw_landmarks(frame, hand_landmarks,
                                      mp_hands.HAND_CONNECTIONS)

    cv2.imshow('MediaPipe Hands', frame)

    if cv2.waitKey(1) == ESC:
        break

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

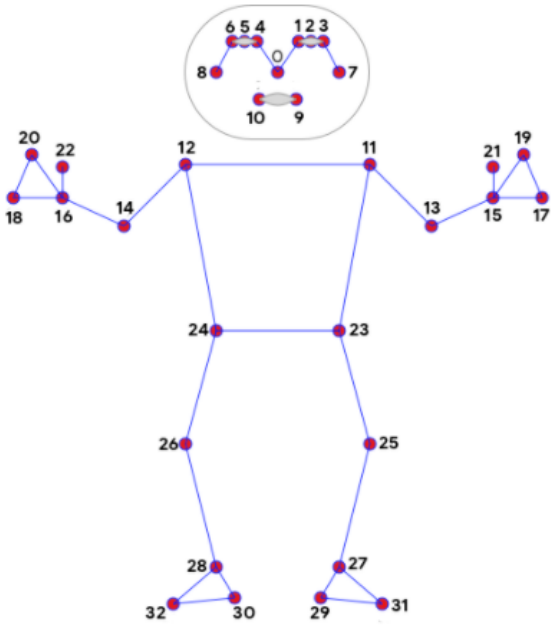
輸出結果：



- Python 程式在使用 OpenCV 讀取影像後，使用 MediaPipe 多手勢追蹤進行手勢辨識，程式執行的結果可以看到綠色線標示出手部地標的 21 個關鍵點 (紅色點)。

MediaPipe 人體姿態估計

MediaPipe 姿勢 (MediaPipe Pose) 是使用 BlazePose 偵測模型來進行人體姿態估計 (Human Pose Estimation)，首先偵測出人體後，使用人體地標模型 (Pose Landmark Model, BlazePose GHUM 3D) 偵測出人體的 33 個關鍵點，如下圖所示：



0. nose
1. left_eye_inner
2. left_eye
3. left_eye_outer
4. right_eye_inner
5. right_eye
6. right_eye_outer
7. left_ear
8. right_ear
9. mouth_left
10. mouth_right
11. left_shoulder
12. right_shoulder
13. left_elbow
14. right_elbow
15. left_wrist
16. right_wrist
17. left_pinky
18. right_pinky
19. left_index
20. right_index
21. left_thumb
22. right_thumb
23. left_hip
24. right_hip
25. left_knee
26. right_knee
27. left_ankle
28. right_ankle
29. left_heel
30. right_heel
31. left_foot_index
32. right_foot_index

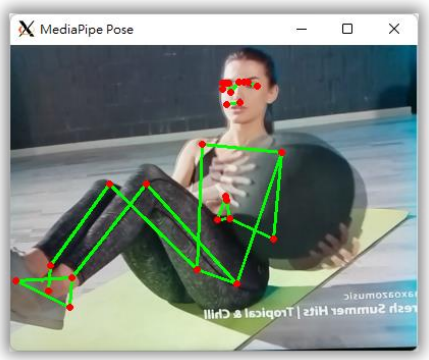
官網 https://google.github.io/mediapipe/solutions/pose.html

4-7 MediaPipe 人體姿態估計。

```
import cv2
import mediapipe as mp

ESC = 27

mp_drawing = mp.solutions.drawing_utils
mp_pose = mp.solutions.pose

cap = cv2.VideoCapture(0)
ratio = cap.get(cv2.CAP_PROP_FRAME_WIDTH) /
cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
WIDTH = 400
HEIGHT = int(WIDTH / ratio)

pose = mp_pose.Pose(min_detection_confidence = 0.5,
                                        min_tracking_confidence = 0.5)

while cap.isOpened():
    ret, frame = cap.read()
```

```
        frame = cv2.resize(frame, (WIDTH, HEIGHT))
        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        frame.flags.writeable = False
        results = pose.process(frame)
        frame.flags.writeable = True
        frame = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)
        mp_drawing.draw_landmarks(frame, results.pose_landmarks,
                                                  mp_pose.POSE_CONNECTIONS)
        cv2.imshow('MediaPipe Pose', frame)

        if cv2.waitKey(1) == ESC:
            break

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

輸出結果：



- Python 程式在使用 OpenCV 讀取影像後，使用 MediaPipe 人體姿態估計進行姿勢的辨識，程式執行的結果可以看到綠色線標示出人體地標的 33 個關鍵點（紅色）。

4-3. CVZone 電腦視覺套件

CVZone 套件

CVZone 是基於 OpenCV 和 MediaPipe 的 Python 套件，我們可以使用更少的程式碼，和更容易的方式來輕鬆進行人臉辨識、3D 臉部網格、多手勢追蹤和人體姿態估計等電腦視覺應用。

官網 https://github.com/cvzone/cvzone

在 Windows 安裝 CVZone

CVZone 需要搭配 OpenCV 與 mediapipe，請先複製之前的 mediapipe 虛擬環境：

離開目前的虛擬環境，例如：mpenv_39。

```
(mpenv_39) >>> deactivate
```

複製之前的 mediapipe 虛擬環境：

```
>>> conda create -n czenv_39 --clone mpenv_39
```

載入虛擬環境 czenv_39：

```
>>> activate czenv_39
```

安裝 CVZone 的指令：

```
(mpenv_39) >>> pip install cvzone
```

在樹莓派安裝 CVZone

CVZone 需要搭配 OpenCV 與 mediapipe，請先複製之前的 mediapipe 虛擬環境：

離開目前的虛擬環境，例如：mediapipe。

```
(mediapipe) ~$ deactivate
```

複製之前的 mediapipe 虛擬環境：

```
~$ cpvirtualenv mediapipe cvzone
```

在樹莓派 4 安裝 1.5.2 版的 CVZone：

```
(cvzone) ~$ pip install cvzone==1.5.2
```

CVZone 人臉偵測

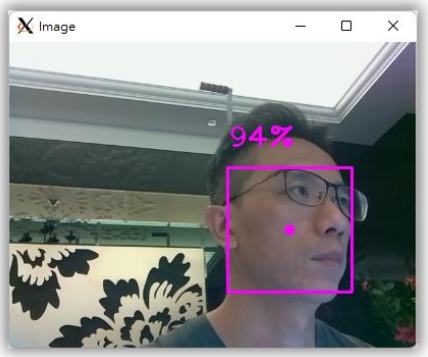
CVZone 是使用 FaceDetector 物件進行人臉偵測，然後呼叫 findFaces() 方法來找出可能的人臉。

4-8 CVZone 人臉偵測。

```
from cvzone.FaceDetectionModule import FaceDetector
import cv2

ESC = 27

cap = cv2.VideoCapture(0)
ratio = cap.get(cv2.CAP_PROP_FRAME_WIDTH) /
cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
WIDTH = 400
HEIGHT = int(WIDTH / ratio)

detector = FaceDetector()

while cap.isOpened():
    success, img = cap.read()
    img = cv2.resize(img, (WIDTH, HEIGHT))

    img, bboxs = detector.findFaces(img)

    if bboxs:
        # bboxInfo - 'id', 'bbox', 'score', 'center'
        center = bboxs[0]['center']
        cv2.circle(img, center, 5, (255, 0, 255), cv2.FILLED)

    cv2.imshow('Image', img)
    if cv2.waitKey(1) == ESC:
        break
```

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

輸出結果：



- Python 程式在使用 OpenCV 讀取影像後，使用 CVZone 進行人臉辨識，程式執行的結果可以看到框出的人臉和上方顯示百分比指出是人臉可能性，和標示出方框的中心點。

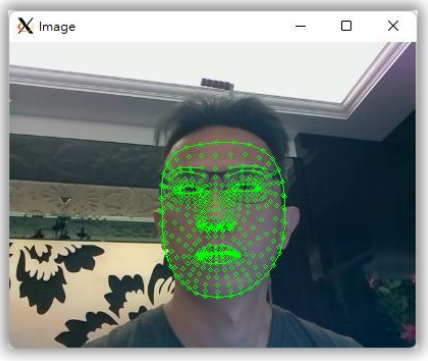
CVZone 臉部網格

CVZone 是使用 FaceMeshDetector 物件進行人臉偵測，然後呼叫 findFaceMesh() 方法來偵測和繪出臉部網格。

4-9 CVZone 臉部網格。

```
from cvzone.FaceMeshModule import FaceMeshDetector
import cv2

ESC = 27

cap = cv2.VideoCapture(0)
ratio = cap.get(cv2.CAP_PROP_FRAME_WIDTH) /
cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
WIDTH = 400
HEIGHT = int(WIDTH / ratio)

detector = FaceMeshDetector(maxFaces = 2)

while cap.isOpened():
    success, img = cap.read()
    img = cv2.resize(img, (WIDTH, HEIGHT))
    img, faces = detector.findFaceMesh(img)
    if faces:
        print(faces[0])
    cv2.imshow('Image', img)

    if cv2.waitKey(1) == ESC:
        break

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

輸出結果：



- Python 程式在使用 OpenCV 讀取影像後，使用 CVZone 進行人臉辨識，和繪出臉部 3D 網格。

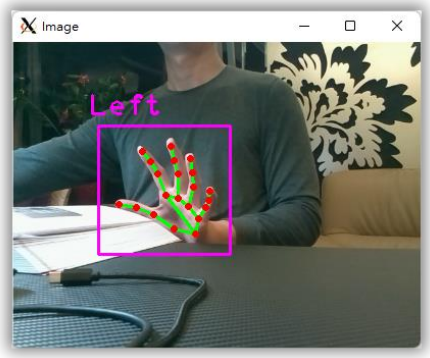
CVZone 多手勢追蹤

CVZone 的 HandDetector 物件可以偵測手勢、計算伸出幾個手指，和測量 2 個手指之間的距離。

4-10 CVZone 多手勢追蹤。

```
import cv2
from cvzone.HandTrackingModule import HandDetector

ESC = 27

detector = HandDetector(maxHands = 2)

cap = cv2.VideoCapture(0)

while True:
    success, img = cap.read()
    if not success:
        break

    hands, img = detector.findHands(img)

    if hands:
        for hand in hands:
            x, y, w, h = hand['bbox']
            cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 3)

            for point in hand['lmList']:
              cv2.circle(img, (point[0], point[1]), 5, (255, 0, 0),
                                                              cv2.FILLED)

    cv2.imshow('Hand Gesture Tracking', img)

    if cv2.waitKey(1) == ESC:
        break

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

輸出結果：



- Python 程式在使用 OpenCV 讀取影像後，使用 CVZone 進行多手勢追蹤，和標示偵測出的是右手或左手。

CVZone 偵測手指數

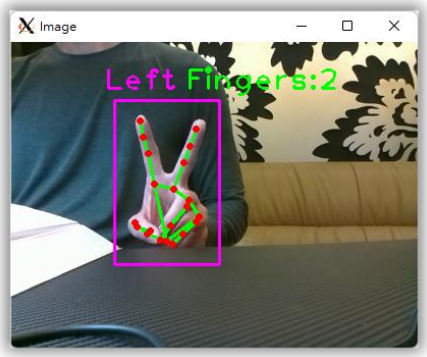
Python 程式在使用 OpenCV 讀取影像後，使用 CVZone 進行多手勢追蹤，可以偵測出手勢共伸出幾個手指。

4-11 CVZone 偵測手指數。

```
from cvzone.HandTrackingModule import HandDetector
import cv2

ESC = 27

cap = cv2.VideoCapture(0)
ratio = cap.get(cv2.CAP_PROP_FRAME_WIDTH) /
cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
WIDTH = 400
HEIGHT = int(WIDTH / ratio)

detector = HandDetector(detectionCon = 0.5, maxHands = 1)

while cap.isOpened():
    success, img = cap.read()
    img = cv2.resize(img, (WIDTH, HEIGHT))
    hands, img = detector.findHands(img)
    if hands:
        hand = hands[0]
        bbox = hand['bbox']
        fingers = detector.fingersUp(hand)
        totalFingers = fingers.count(1)
        cv2.putText(img, f'Fingers:{totalFingers}', (bbox[0] + 50,
                bbox[1] - 30), cv2.FONT_HERSHEY_PLAIN, 2, (0, 255, 0), 2)
```

```
    cv2.imshow('Image', img)

    if cv2.waitKey(1) == ESC:
        break

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

輸出結果：



- 例如：Left Fingers:2，就是左手 2 個手指。

CVZone 範例：辨識剪刀、石頭和布的手勢

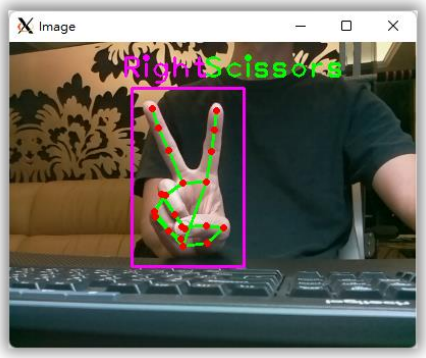
使用 CVZone 依據手掌伸出的手指數來辨識出手勢是剪刀、石頭或布。在 Python 程式可以使用 HandDetector 物件的 fingersUp() 方法來偵測手掌伸出哪些手指，進而編寫程式來判斷剪刀、石頭和布。

4-12 CVZone 辨識剪刀、石頭和布的手勢。

```
from cvzone.HandTrackingModule import HandDetector
import cv2

detector = HandDetector(detectionCon = 0.5, maxHands = 1)
img = cv2.imread('images/Scissors.png', cv2.IMREAD_COLOR)

hands, img = detector.findHands(img)

if hands:
    hand = hands[0]
    fingers = detector.fingersUp(hand)
    print(fingers)
    totalFingers = fingers.count(1)
    if totalFingers == 5:
        print('Paper')
    if totalFingers == 0:
```

```
        print('Rock')
    if totalFingers == 2:
        if fingers[1] == 1 and fingers[2] == 1:
            print('Scissors')

cv2.imshow('Image', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

輸出結果：



上述 `detector.fingersUp()` 方法的回傳值是 5 根手指（從姆指開始至小指）的清單。例如：`[0, 1, 1, 0, 0]`，`1` 代表伸出該手指；`0` 則沒有。此例是伸出食指和中指，我們可以使用此清單內容來辨識出手勢是剪刀、石頭或布。

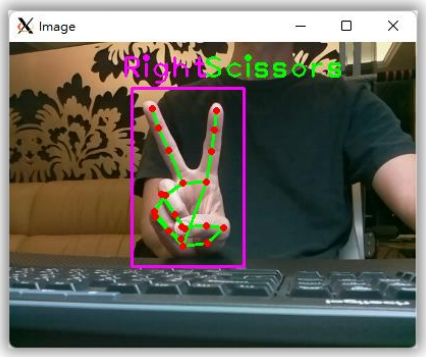
接下來，進行即時影像的辯勢：

4-13 CVZone 辨識剪刀、石頭和布的手勢。

```
from cvzone.HandTrackingModule import HandDetector
import cv2

ESC = 27

cap = cv2.VideoCapture(0)
ratio = cap.get(cv2.CAP_PROP_FRAME_WIDTH) /
cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
WIDTH = 400
HEIGHT = int(WIDTH / ratio)

detector = HandDetector(detectionCon = 0.5, maxHands = 1)

while cap.isOpened():
    success, img = cap.read()
    img = cv2.resize(img, (WIDTH, HEIGHT))
```

```
    hands, img = detector.findHands(img)
    if hands:
        hand = hands[0]
        bbox = hand['bbox']
        fingers = detector.fingersUp(hand)
        totalFingers = fingers.count(1)
        print(totalFingers)
        msg = 'None'
        if totalFingers == 5:
            msg = 'Paper'
        if totalFingers == 0:
            msg = 'Rock'
        if totalFingers == 2:
            if fingers[1] == 1 and fingers[2] == 1:
                msg = 'Scissors'

        cv2.putText(img, msg, (bbox[0] + 50, bbox[1] - 30),
                    cv2.FONT_HERSHEY_PLAIN, 2, (0, 255, 0), 2)

    cv2.imshow('Image', img)

    if cv2.waitKey(1) == ESC:
        break

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

輸出結果：



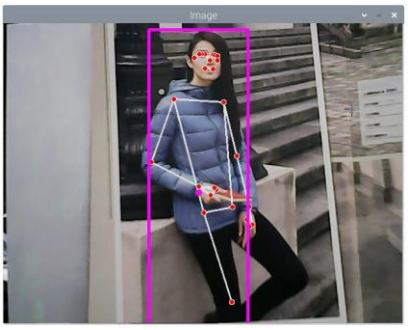
CVZone 人體姿態估計

CVZone 是使用 PoseDetector 物件進行人體姿態估計，我們可以呼叫 findPose() 和 findPosition() 方法來偵測出人體和找出關鍵點。

4-14 CVZone 人體姿態估計。

```
from cvzone.PoseModule import PoseDetector
import cv2

cap = cv2.VideoCapture(0)
ratio = cap.get(cv2.CAP_PROP_FRAME_WIDTH) /
cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
WIDTH = 400
HEIGHT = int(WIDTH / ratio)
detector = PoseDetector()

while cap.isOpened():
    success, img = cap.read()
    img = cv2.resize(img, (WIDTH, HEIGHT))
    img = detector.findPose(img)
    lmList, bboxInfo = detector.findPosition(img, bboxWithHands = False)
    if bboxInfo:
        center = bboxInfo['center']
        cv2.circle(img, center, 5, (255, 0, 255), cv2.FILLED)
    cv2.imshow('Image', img)
    if cv2.waitKey(1) == 27:
        break

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

輸出結果：



- Python 程式在使用 OpenCV 讀取影像後，使用 CVZone 進行人體姿態估計。