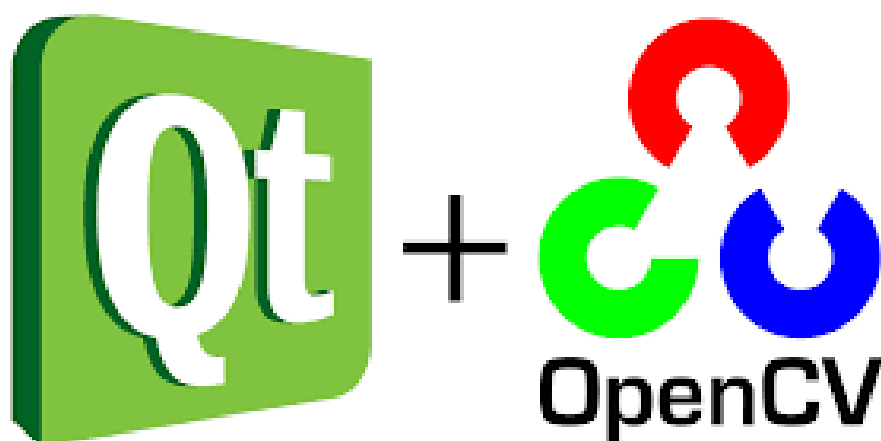


影像處理程式開發平台作業



使用 Qt + OpenCV 實作

資工三甲

4A8G0039

楊孟繁學

目錄

目錄.....	ii
HW3.....	1
Canny Edge Detection (Canny 邊緣檢測).....	1
Hough Line Transform (霍夫轉換)	2
HoughLinesP.....	3
HoughLines.....	3
Harris Corner Detection (Harris 角點檢測)	4
Feature Detection (特徵檢測).....	5
Feature Description (特徵匹配)	6
Finding contours (尋找輪廓).....	7
cv2.findContours	8
cv2.convexHull.....	8
cv2.boundingRect.....	9
cv2.minEnclosingCircle	9
cv2.minAreaRect.....	10
cv2.fitEllipse	10
Morphology Transformations (形態轉換)	11
cv2.MORPH_ERODE.....	12
cv2.MORPH_DILATE.....	12
cv2.MORPH_OPEN	13
cv2.MORPH_CLOSE.....	13
cv2. MORPH_GRADIENT	14
cv2.MORPH_TOPHAT	14
cv2.MORPH_BLACKHAT	15
Skeletonize	15
Perimeter.....	16
Final_HW	17
Hand Detection	17
Face Detection	18
Pose Detection	19

HW3

[GitHub 連結](#)

Canny Edge Detection (Canny 邊緣檢測)

使用 `cv2.Canny()` 抓取物件邊緣。

Threshold low value : 最低閾值

Ratio value : 高低閾值比

Ksize value : 模糊內核大小



Hough Line Transform (霍夫轉換)

使用 `cv2.Canny()` 抓取物件邊緣後再使用 `cv2.HoughLines()` 或 `cv2.HoughLinesP()` 找出圖中的直線。

Canny Edge Detection :

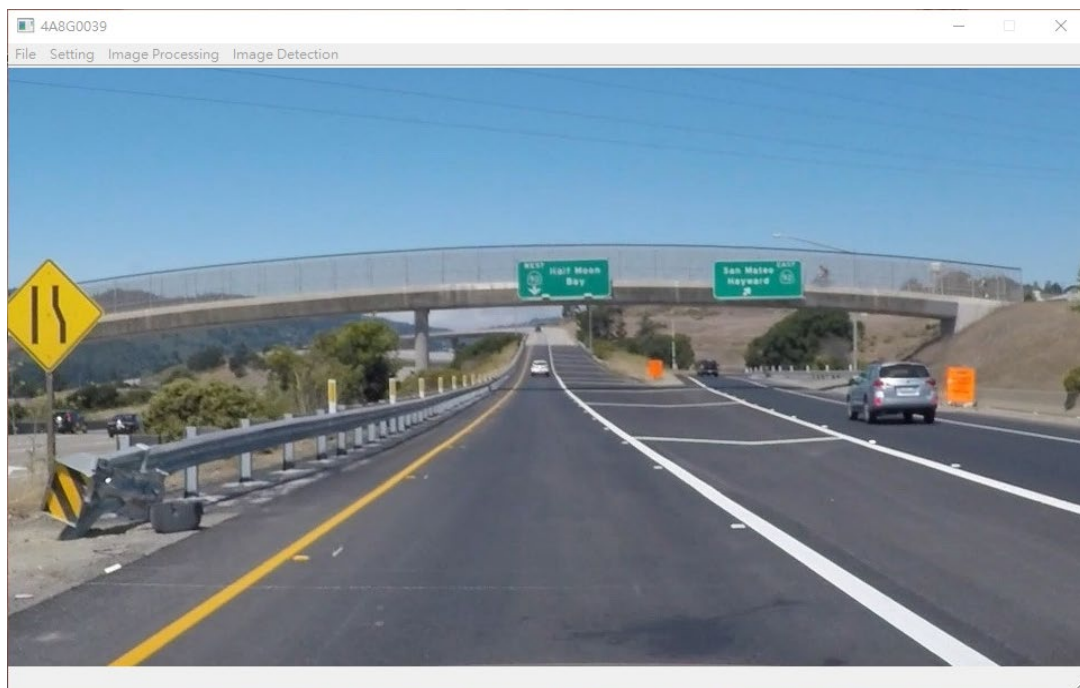
Threshold low value : 最低閾值

Ratio value : 高低閾值比

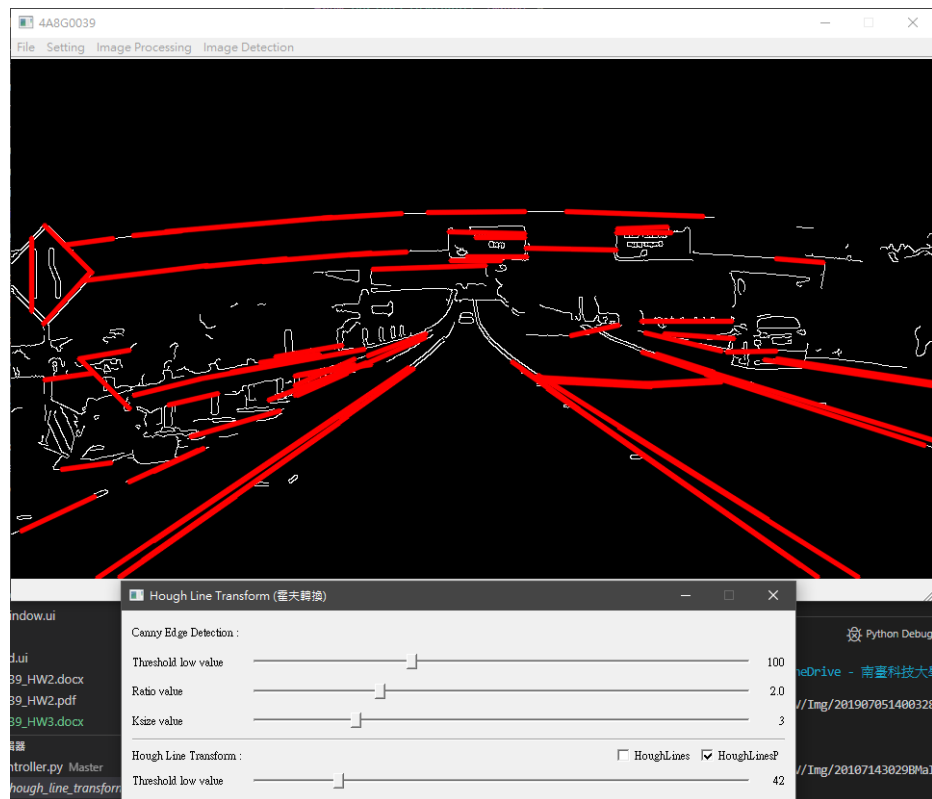
Ksize value : 模糊內核大小

Hough Line Transform :

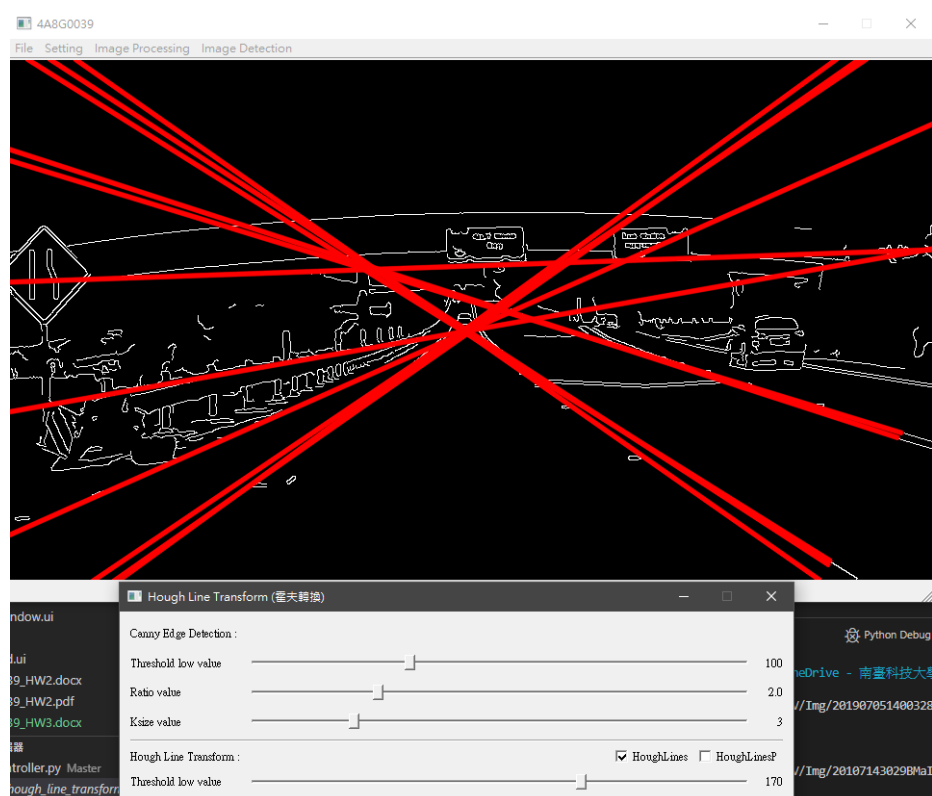
Threshold low value : 最低閾值



HoughLinesP



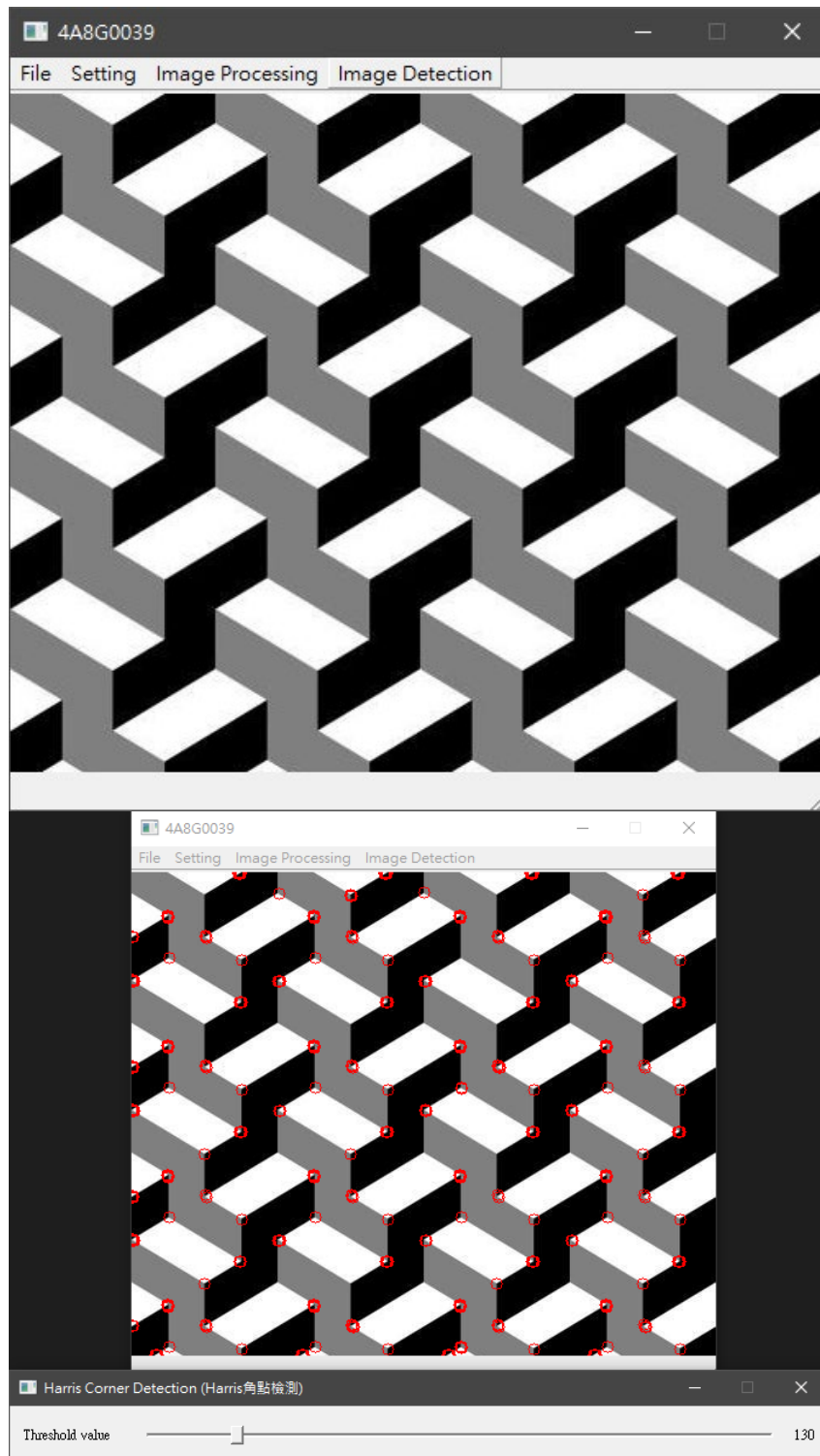
HoughLines



Harris Corner Detection (Harris 角點檢測)

使用 `cv2.cornerHarris()` 找出圖中的角並圈出來。

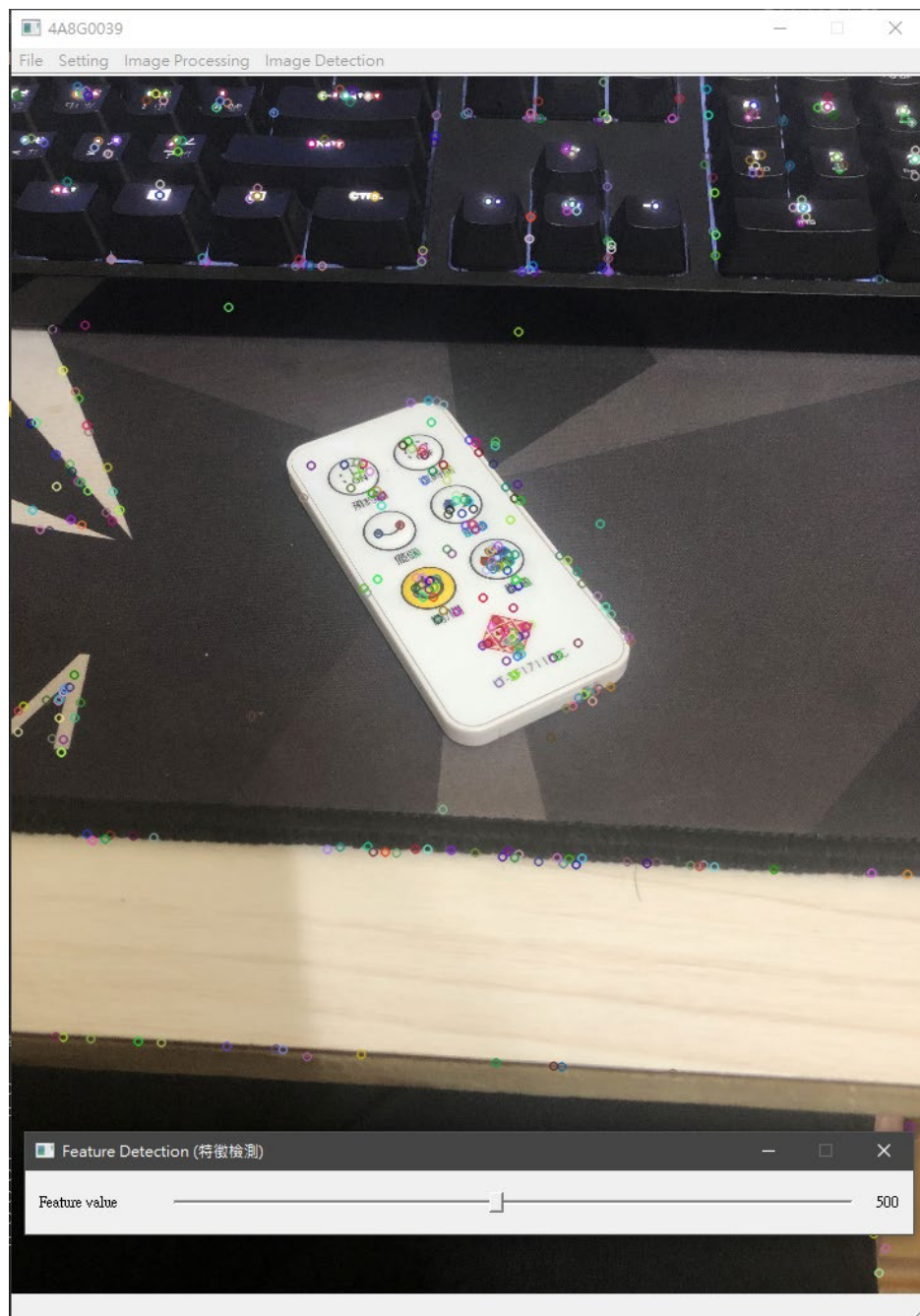
Threshold value : 閾值



Feature Detection (特徵檢測)

使用 `cv2.SIFT_create()` 找出圖中的特徵點並圈出來。

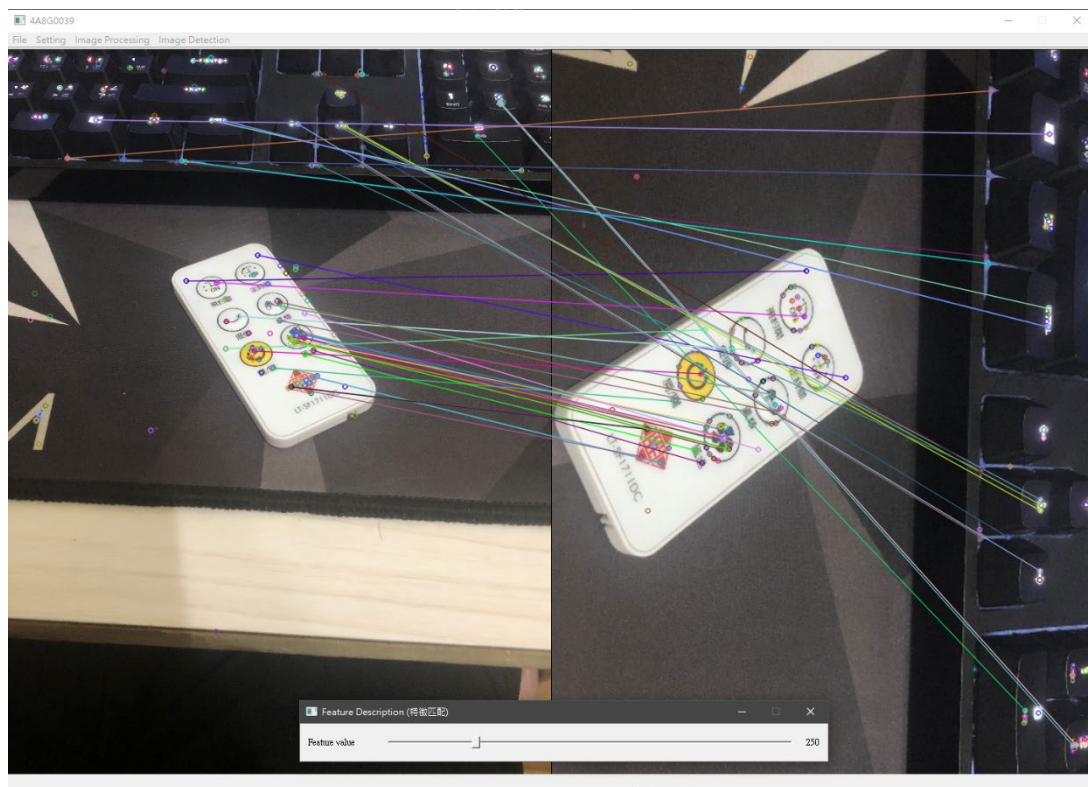
Feature value : 特徵點數量



Feature Description (特徵匹配)

使用 `cv2.SIFT_create()` 找出圖中的特徵點並圈出來，
再用 `cv2.BFMatcher()` 比對兩張圖的特徵點，
最後再把權重較高的對應點連起來。

Feature value：特徵點數量



Finding contours (尋找輪廓)

使用 `cv2.Canny()` 抓取物件邊緣後再使用 `cv2.findContours()` 找出物件的輪廓。

找出輪廓後就可：

- 使用 `cv2.convexHull()` 找出物件的凸包(最外框)。

- 使用 `cv2.boundingRect()` 找出物件的最小矩形框。

- 使用 `cv2.minEnclosingCircle()` 找出物件的最小圓形框。

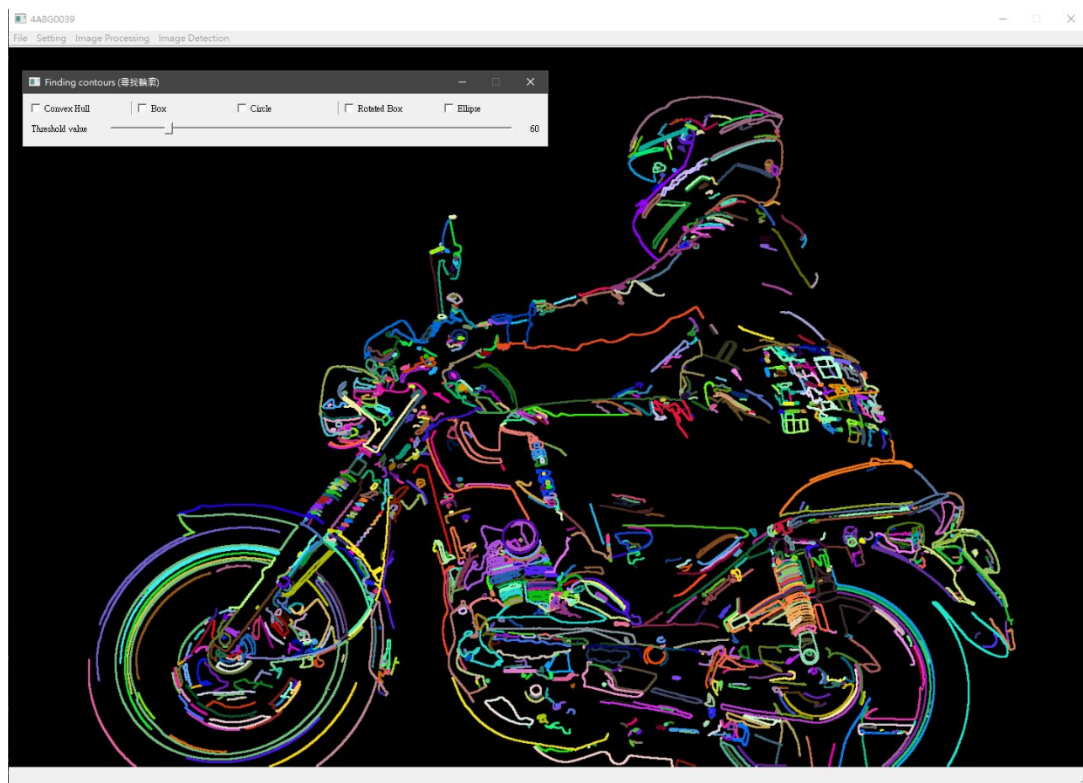
- 使用 `cv2.minAreaRect()` 找出物件的最小旋轉矩形框。

- 使用 `cv2.fitEllipse()` 找出物件的最小橢圓形框。

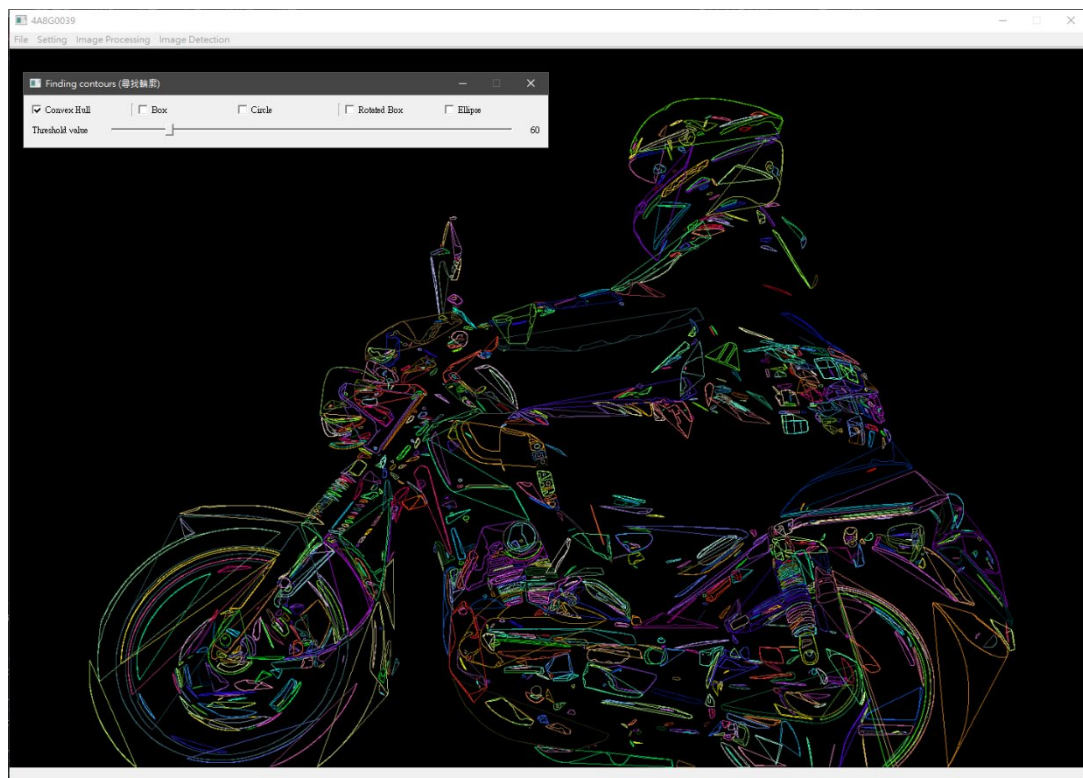
Threshold value : Canny 閾值



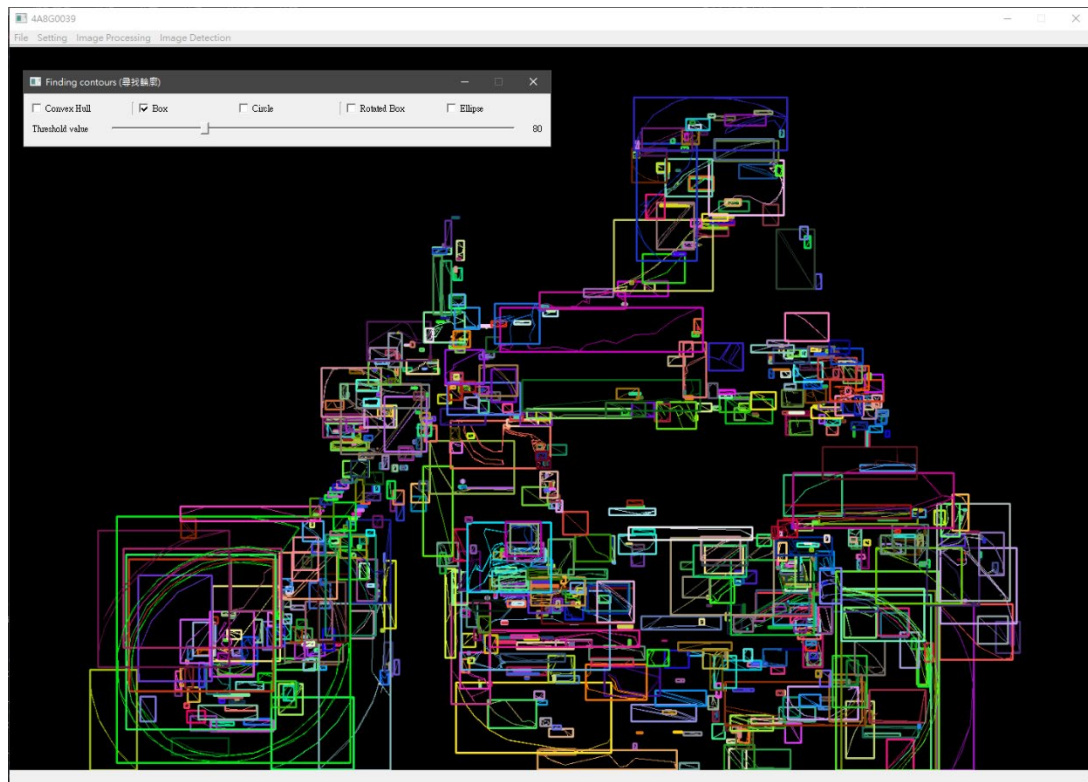
cv2.findContours



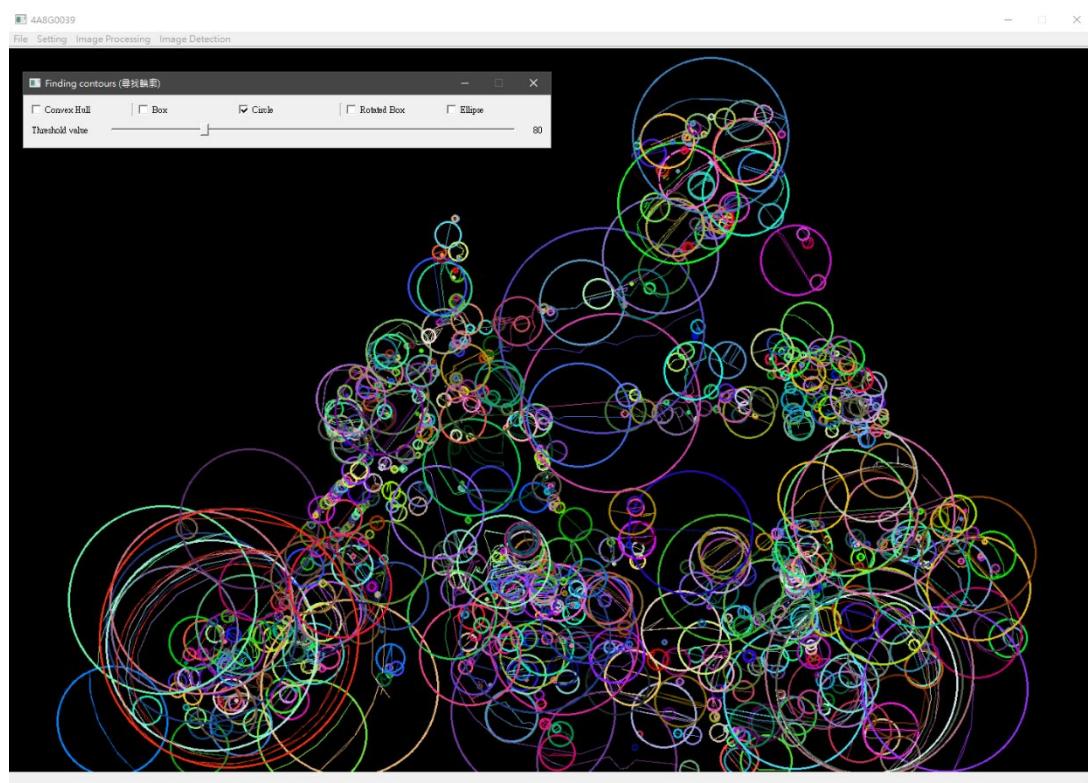
cv2.convexHull



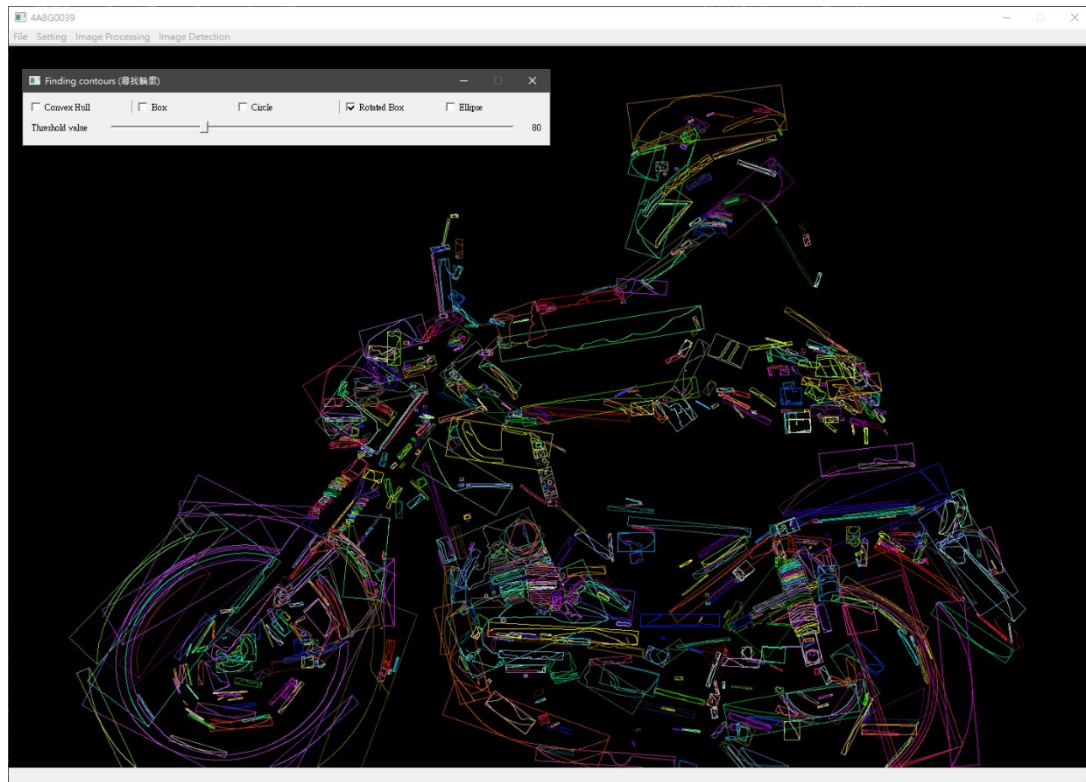
cv2.boundingRect



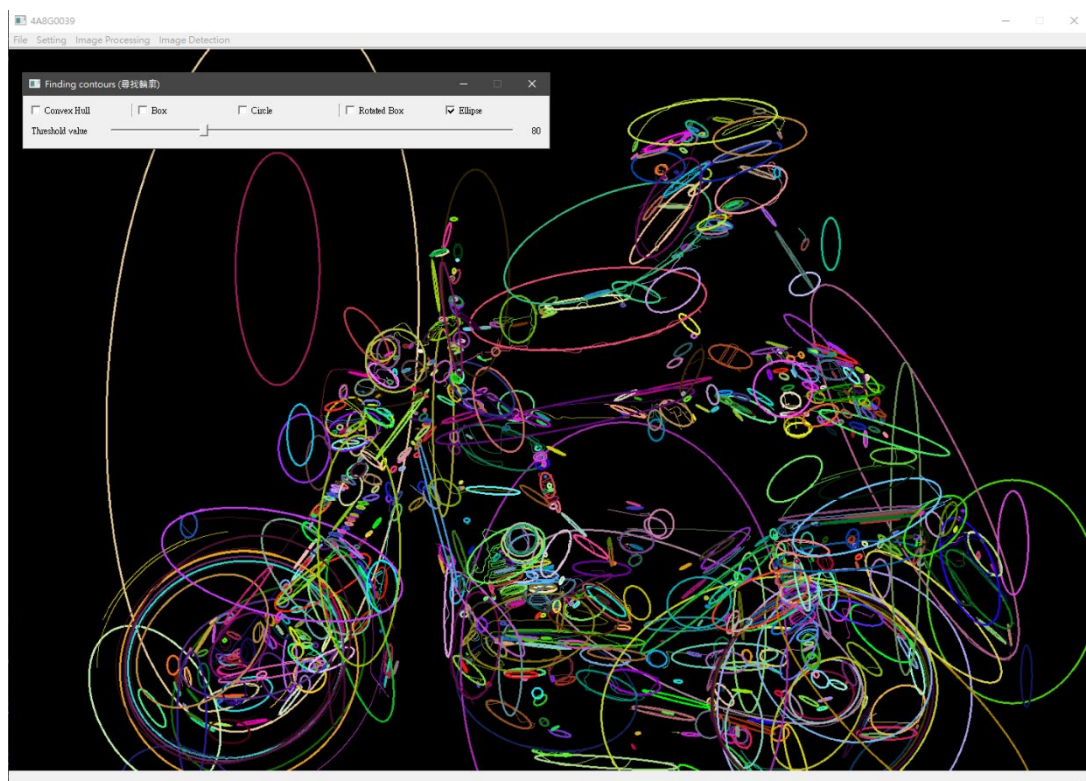
cv2.minEnclosingCircle



cv2.minAreaRect



cv2.fitEllipse



Morphology Transformations (形態轉換)

使用 `cv2.MORPH_ERODE` 實作腐蝕圖像。

使用 `cv2.MORPH_DILATE` 實作擴大圖像。

使用 `cv2.MORPH_OPEN` 實作腐蝕圖像後擴大圖像。

使用 `cv2.MORPH_CLOSE` 實作擴大圖像後腐蝕圖像。

使用 `cv2.MORPH_GRADIENT` 實作擴大圖像 - 腐蝕圖像。

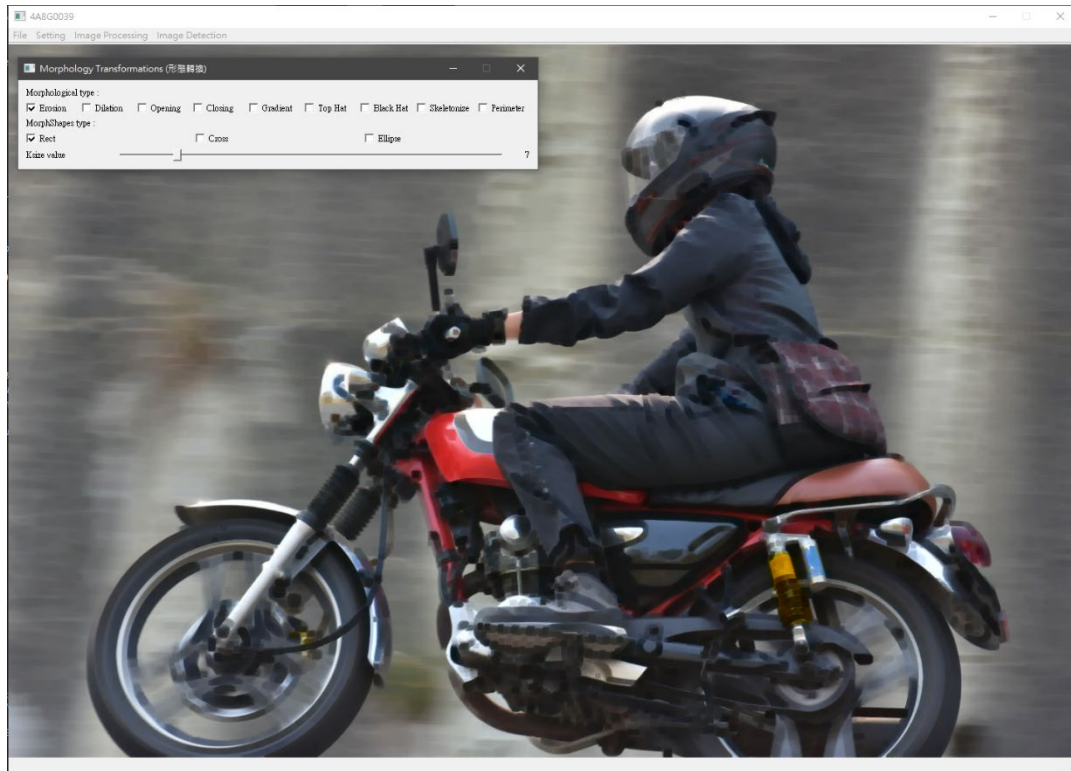
使用 `cv2.MORPH_TOPHAT` 實作原圖像 - `cv2.MORPH_OPEN`。

使用 `cv2.MORPH_BLACKHAT` 實作
`cv2.MORPH_CLOSE` - 原圖像。

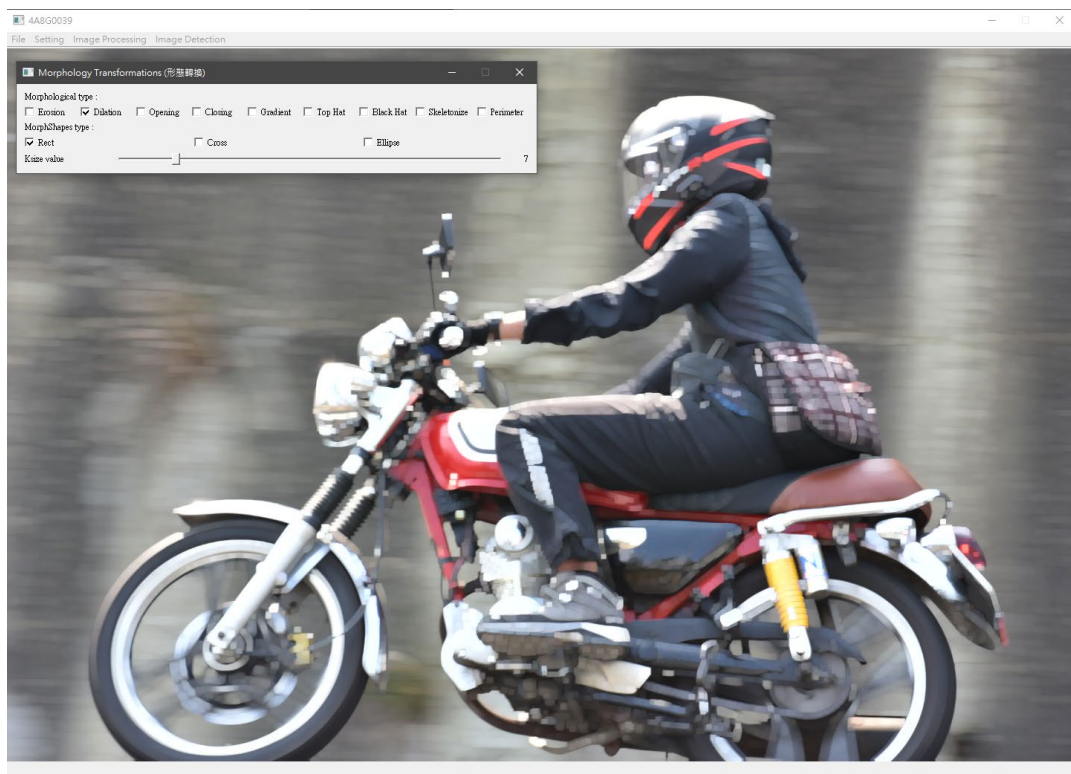
Ksize value : 模糊內核大小



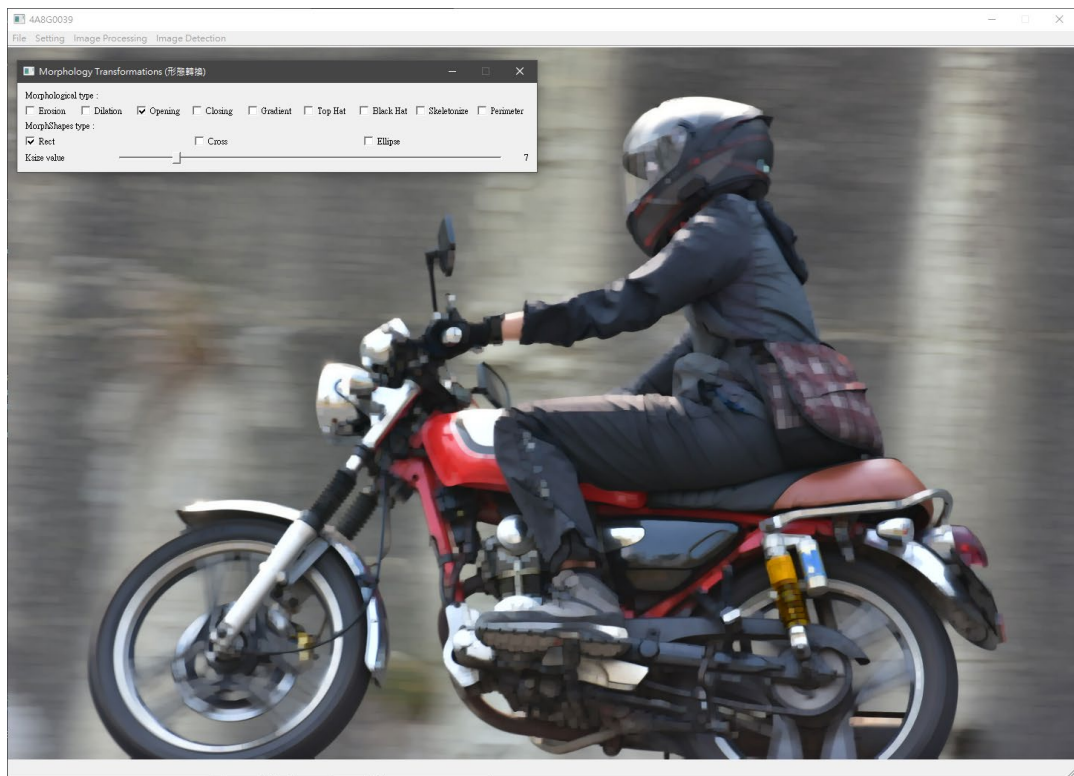
cv2.MORPH_ERODE



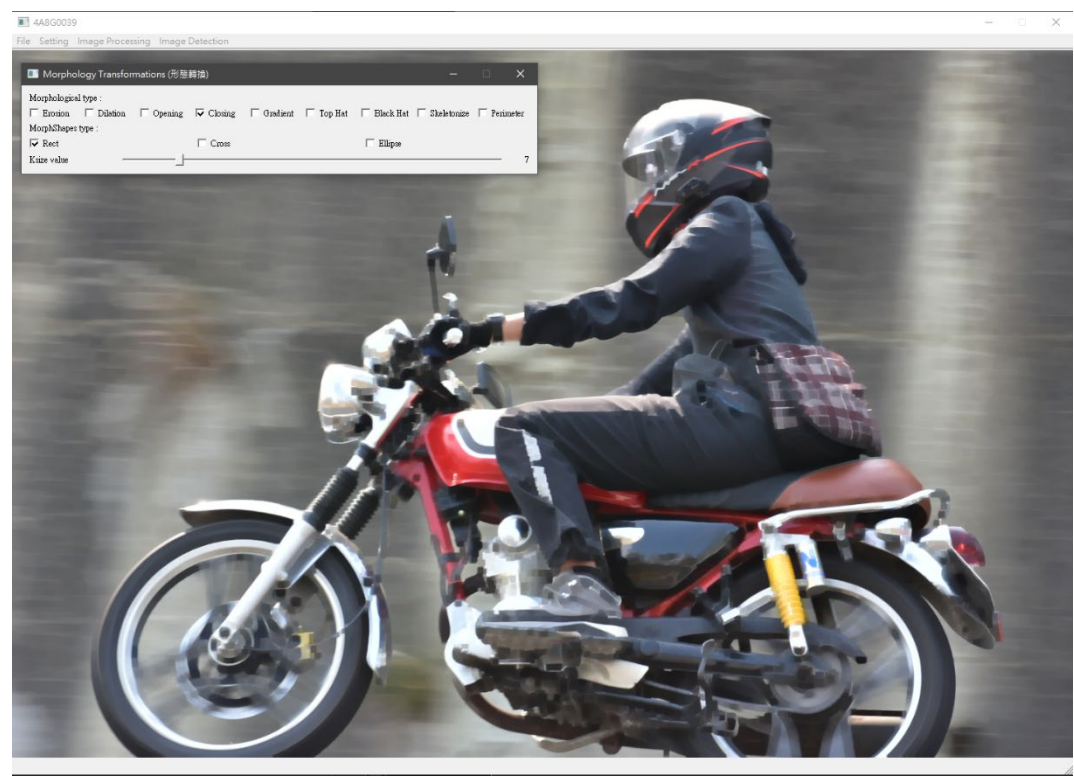
cv2.MORPH_DILATE



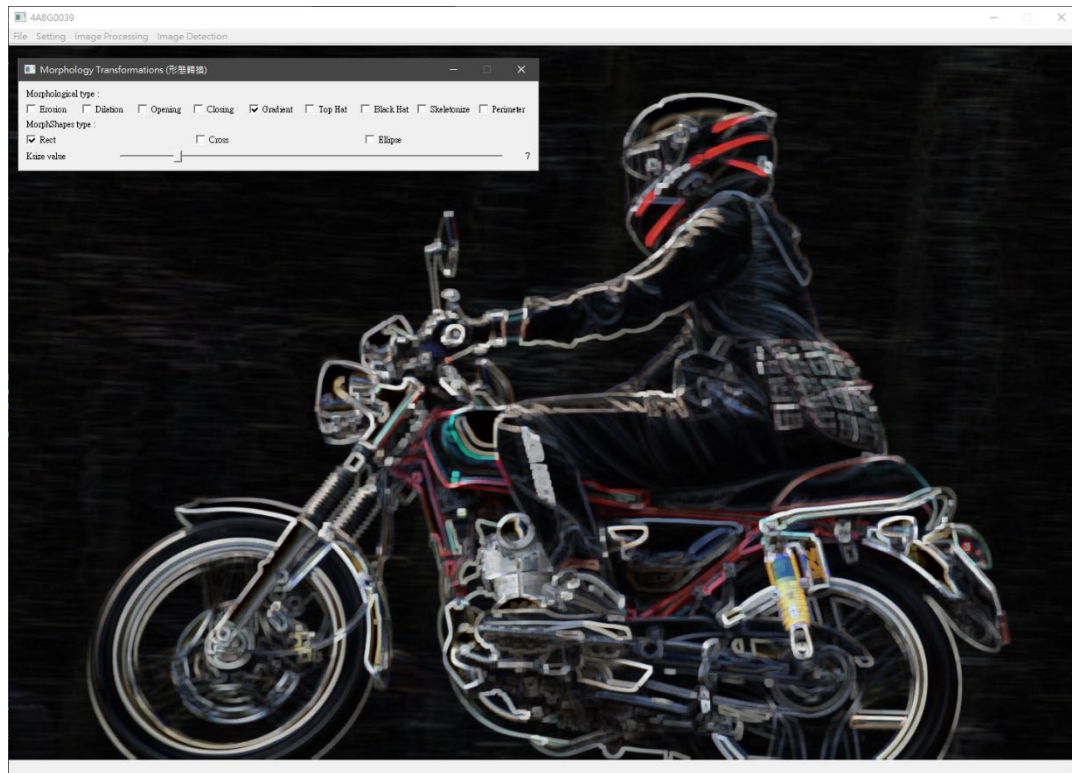
cv2.MORPH_OPEN



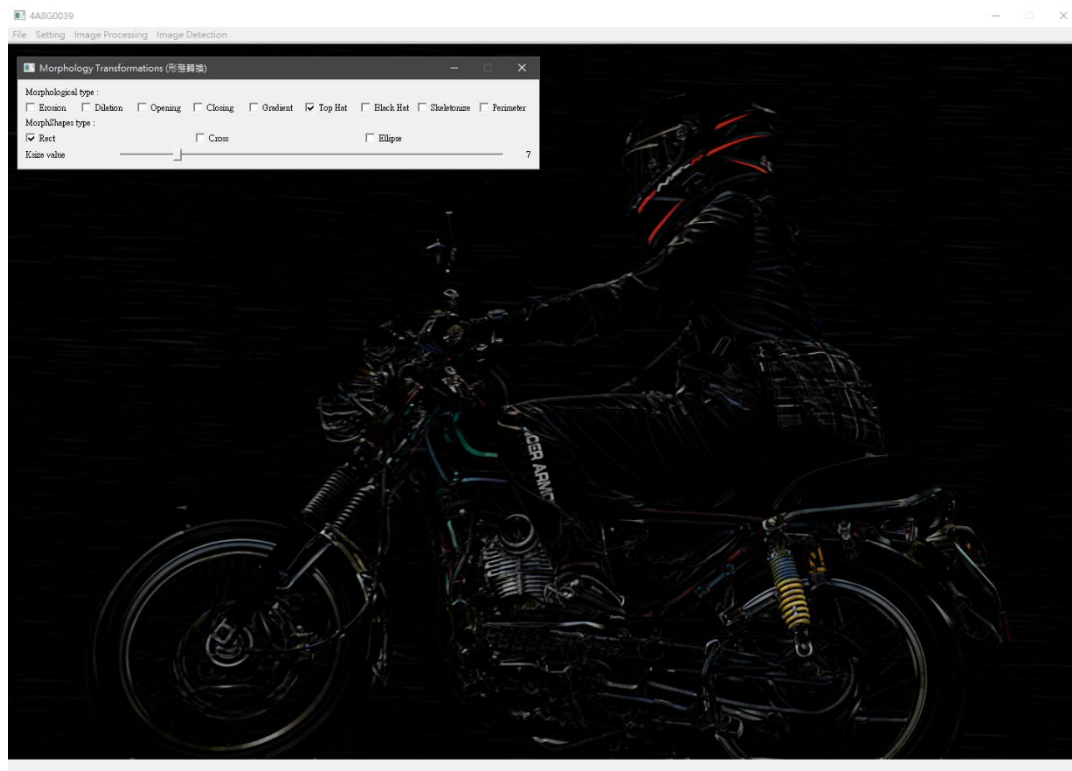
cv2.MORPH_CLOSE



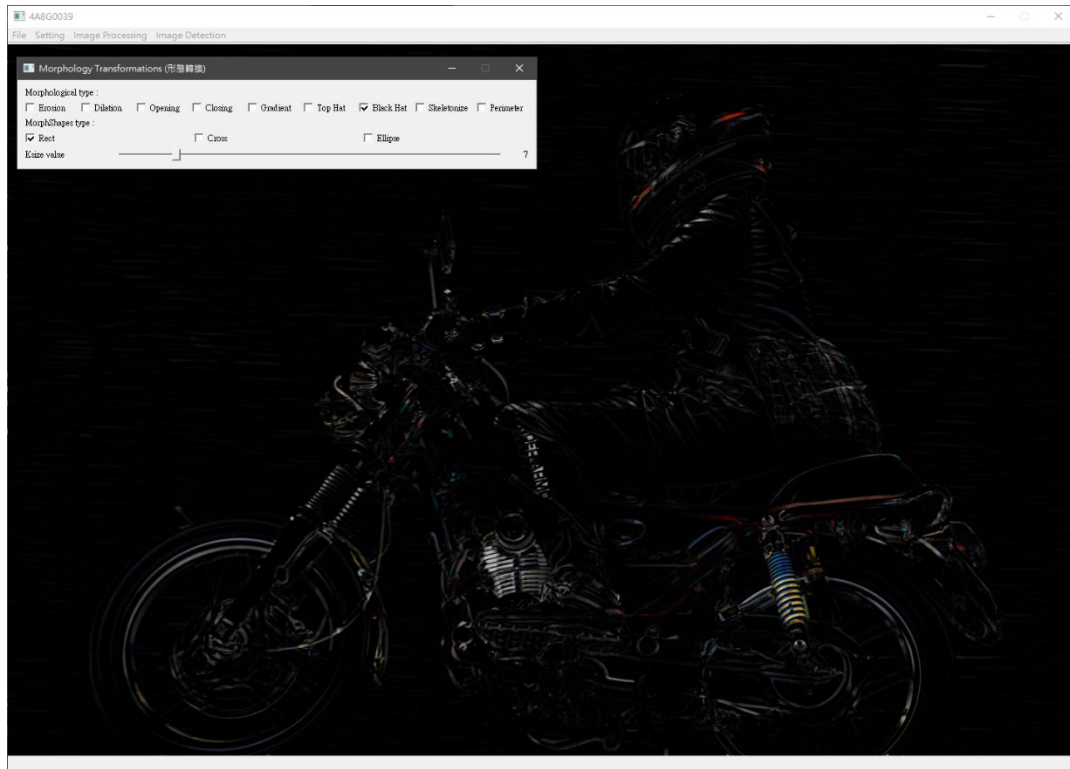
cv2.MORPH_GRADIENT



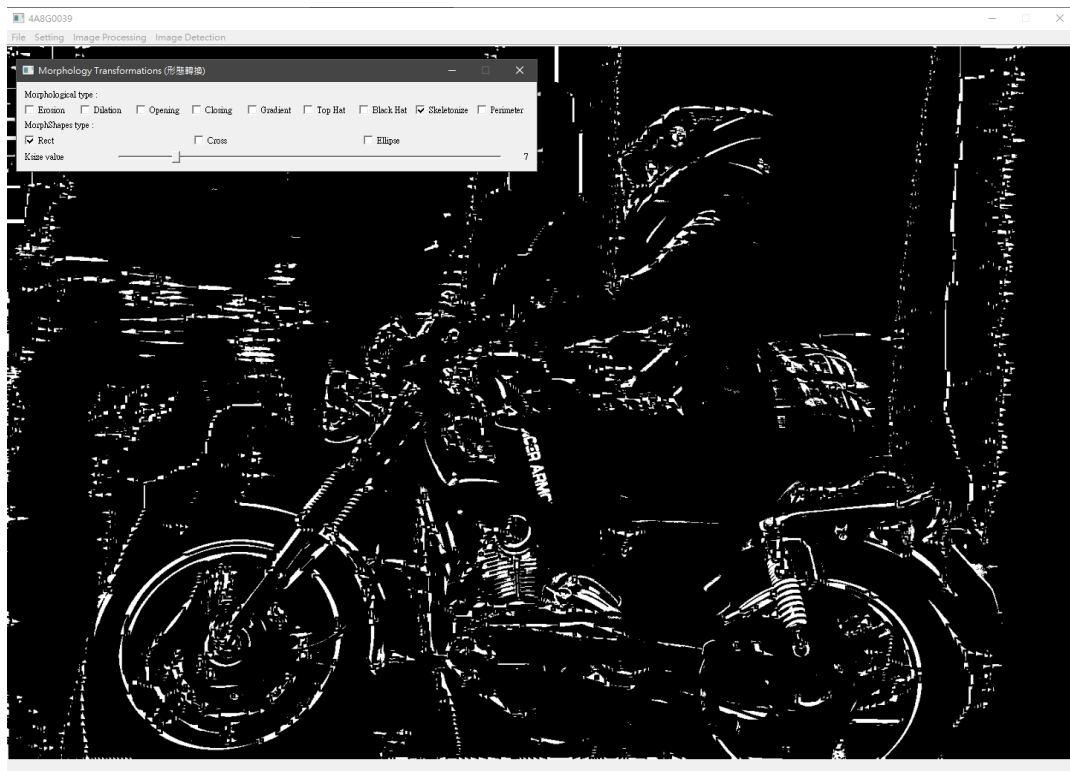
cv2.MORPH_TOPHAT



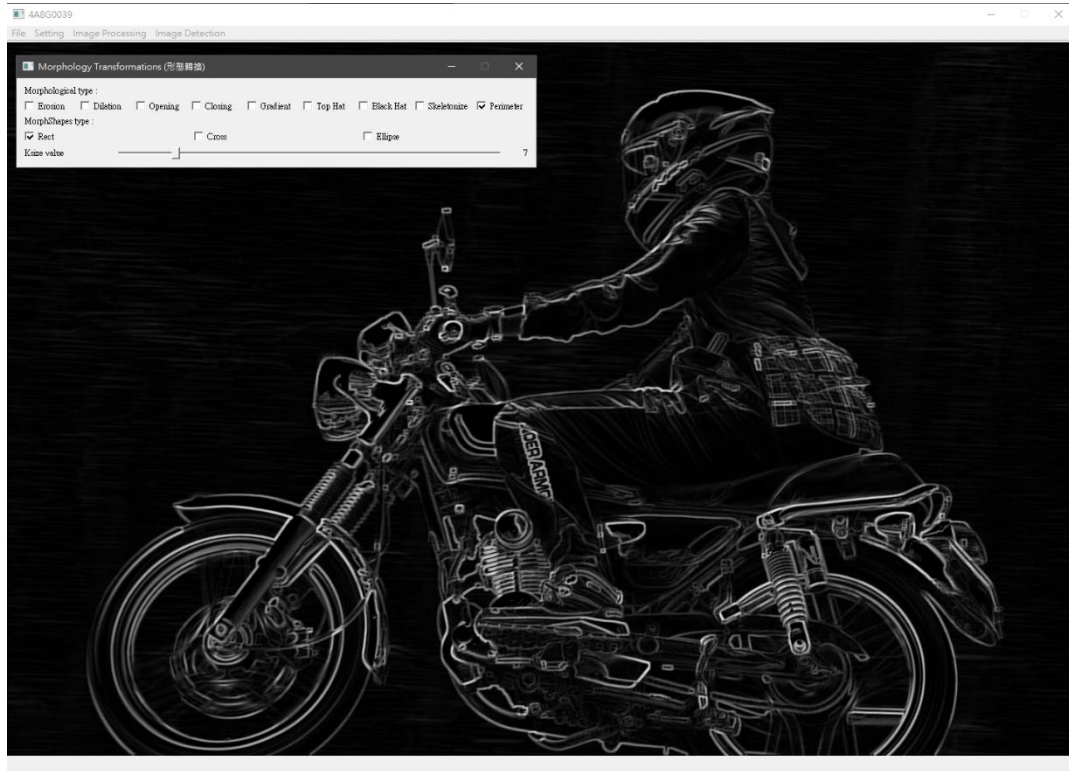
cv2.MORPH_BLACKHAT



Skeletonize



Perimeter



Final_HW

[GitHub 連結](#)

Hand Detection

```
self.mp_hands = mp.solutions.hands
self.mp_Hands = self.mp_hands.Hands(min_detection_confidence=0.5,
min_tracking_confidence=0.5) #設定手部偵測數值
self.mp_drawing = mp.solutions.drawing_utils

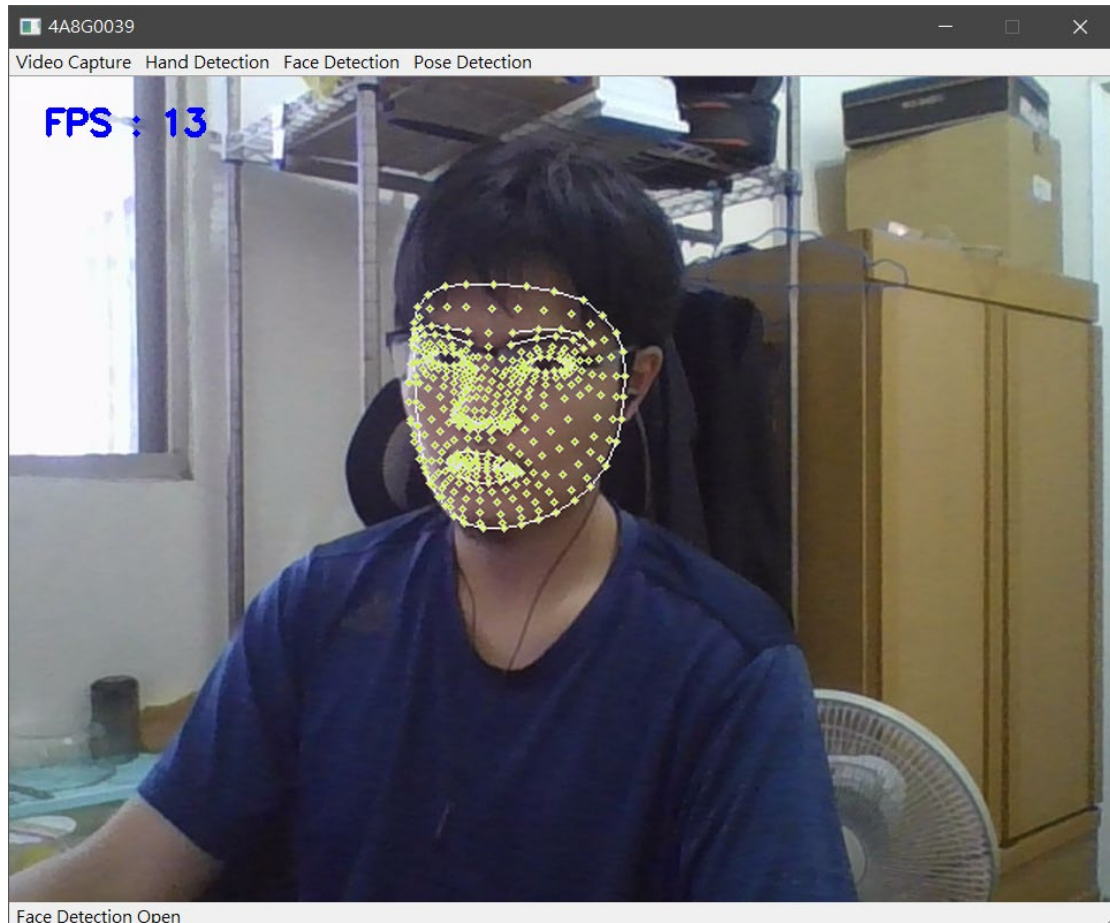
hands_result = self.mp_Hands.process(imgRGB) #偵測手部
if hands_result.multi_hand_landmarks:
    for handLms in hands_result.multi_hand_landmarks: #讀取手部標記點
        self.mp_drawing.draw_landmarks(frame, handLms,
self.mp_hands.HAND_CONNECTIONS,
self.mp_drawing.DrawingSpec(color=(255,0,255), thickness=2,
circle_radius=3), self.mp_drawing.DrawingSpec(color=(0,0,255),
thickness=2)) #畫出手部標記點
```



Face Detection

```
self.mp_face_mesh = mp.solutions.face_mesh
self.mp_FaceMesh =
self.mp_face_mesh.FaceMesh(min_detection_confidence=0.5,
min_tracking_confidence=0.5) #設定臉部偵測數值
self.mp_drawing = mp.solutions.drawing_utils

face_mesh_result = self.mp_FaceMesh.process(imgRGB) #偵測臉部
if face_mesh_result.multi_face_landmarks:
    for face_mesh_Lms in face_mesh_result.multi_face_landmarks: #讀取臉
        部標記點
            self.mp_drawing.draw_landmarks(frame, face_mesh_Lms, self.mp_face
            _mesh.FACEMESH_CONTOURS, self.mp_drawing.DrawingSpec(color=(0,255,20
            0), thickness=1, circle_radius=1),
            self.mp_drawing.DrawingSpec(color=(255,255,255), thickness=1,
            circle_radius=1)) #畫出臉部標記點
```



Pose Detection

```
self.mp_pose = mp.solutions.pose
self.mp_Pose = self.mp_pose.Pose(min_detection_confidence=0.5,
min_tracking_confidence=0.5) #設定軀幹偵測數值
self.mp_drawing = mp.solutions.drawing_utils

Pose_result = self.mp_Pose.process(imgRGB) #偵測軀幹
self.mp_drawing.draw_landmarks(frame, Pose_result.pose_landmarks,
self.mp_pose.POSE_CONNECTIONS,
self.mp_drawing.DrawingSpec(color=(255,0,0), thickness=2,
circle_radius=3), self.mp_drawing.DrawingSpec(color=(0,255,0),
thickness=2)) #畫出軀幹標記點
```

