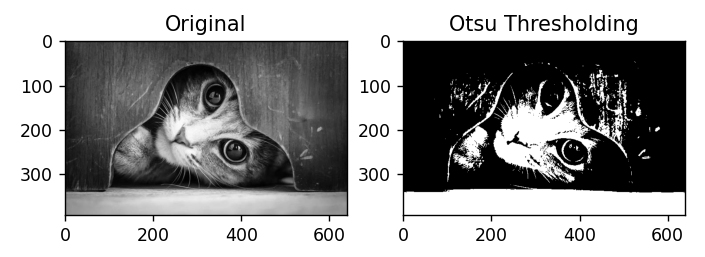
**Comment**

透過實作 Otsu’s thresholding，理解其利用灰階直方圖統計資訊，嘗試每一個閾值將圖像區分為前景與背景，並選取能使類間變異最大化的閾值。該方法不需人工設定參數，適用於目標與背景灰度分布差異明顯的場景，能有效提升二值化的自動化與穩定性。

**Photo**



**Program**

import cv2

import numpy as np

import matplotlib.pyplot as plt

def otsu\_threshold(image):

    pixel\_counts = np.bincount(image.ravel(), minlength=256)

    total\_pixels = image.size

    sum\_total = np.dot(np.arange(256), pixel\_counts)

    sumB = 0

    wB = 0

    maximum = 0

    threshold = 0

    for i in range(256):

        wB += pixel\_counts[i]

        if wB == 0:

            continue

        wF = total\_pixels - wB

        if wF == 0:

            break

        sumB += i \* pixel\_counts[i]

        mB = sumB / wB

        mF = (sum\_total - sumB) / wF

        # 類間變異

        var\_between = wB \* wF \* (mB - mF) \*\* 2

        if var\_between > maximum:

            maximum = var\_between

            threshold = i

    return threshold

# 讀取灰階圖像

img = cv2.imread('cat.jpg', cv2.IMREAD\_GRAYSCALE)

# 計算 Otsu 閾值

thresh\_val = otsu\_threshold(img)

# 應用閾值二值化

\_, thresh\_img = cv2.threshold(img, thresh\_val, 255, cv2.THRESH\_BINARY)

# 顯示結果

print(f"Otsu Threshold Value: {thresh\_val}")

cv2.imwrite('otsu\_result.jpg', thresh\_img)

plt.subplot(1,2,1)

plt.title("Original")

plt.imshow(img, cmap='gray')

plt.subplot(1,2,2)

plt.title("Otsu Thresholding")

plt.imshow(thresh\_img, cmap='gray')

plt.show()