Computer Vision HW2 R10921A10 電機碩一 廖彥朋

Write a program to generate:

(a) a binary image (threshold at 128)

(b) a histogram

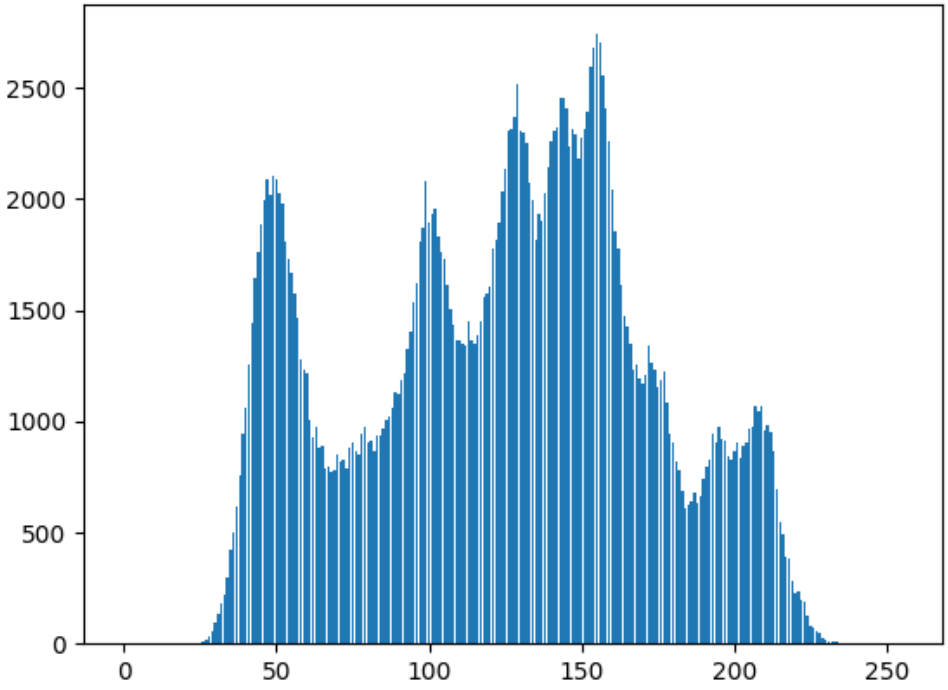
(c) connected components (regions with + at centroid, bounding box)

原始圖片:

實作結果:

(a) a binary image (threshold at 128)

(b) a histogram



(c) connected components (regions with + at centroid, bounding box)-(8-connectivity)

程式碼(Python):

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.patches as patches



def binary_image(img):
    img_binary = img.copy()
    for i in range(img_binary.shape[0]):
        for j in range(img_binary.shape[1]):
            for k in range(img_binary.shape[2]):
                if img[i, j, k] >= 128:
                    img_binary[i, j, k] = 255
                else:
                    img_binary[i, j, k] = 0
    return img_binary

def histogram(img):
    histogram = np.zeros(256, int)
    img_output = img.copy()
    for i in range(img_output.shape[0]):
        for j in range(img_output.shape[1]):
            histogram[img_output[i, j, 0]] += 1
    plt.bar(range(0,256), histogram)
    plt.savefig('histogram.png')
    plt.show()
    return histogram


# Define a stack so that we can use it to store of the location of each element.
class Stack:
    def __init__(self):
        self.list = []

    def push(self,item):
        self.list.append(item)

    def pop(self):
        return self.list.pop()

    def isEmpty(self):
        return len(self.list) == 0
def Connection_component(img):
    # Set lena.bmp as img.
    img = cv2.imread("lena.bmp")

    # Set the threshold value of the count of each label as 500.
    threshold_region = 500

    # Set the first label number value as 1.
    label_num = 1

    # Get width, height and rgb of lena.bmp.
    height, width, rgb = img.shape

    # This array records the label number of each element.
    label_list = np.zeros((height, width), int)

    # This array records whether we have visted the element or not.
    label_visited = np.zeros((height, width), int)

    # This array records the count of each label.
    label_count = np.zeros((height * width), int)

    # Build an array with the same size as lena.bmp.
    binary_img = img.copy()

    # Creation of binary image.
    for i in range(height):
        for j in range(width):
            for k in range(rgb):
                if img[i, j, k] >= 128:
                    binary_img[i, j, k] = 255
                else:
                    binary_img[i, j, k] = 0

    # Creation of processing image.
    img_process = np.zeros((height, width), int)
    for i in range(height):
        for j in range(width):
            for k in range(rgb):
                if binary_img[i, j, k] == 255:
                    img_process[i, j] = 1
                else:
                    img_process[i, j] = 0

```

```
    # Image processing of each pixel.
    for i in range(height):
        for j in range(width):
                    # If the pixel is 0, then mark it as visted.
                    if img_process[i, j] == 0:
                        label_visited[i, j] = 1
                    # If the pixel is 1 and it is not visted yet.
                    elif label_visited[i, j] == 0:
                        # Creation of a stack.
                        stack = Stack()
                        # Push the location of the pixel to the stack.
                        stack.push((i, j))
                        # While the stack is not empty.
                        while not stack.isEmpty():

                            row, col = stack.pop()

                            if label_visited[row, col] == 1:
                                continue
                            # Mark the pixel as visted.
                            label_visited[row, col] = 1
                            # Assign label number to the pixel.
                            label_list[row, col] = label_num
                            # Count of each label.
                            label_count[label_num] = label_count[label_num] + 1
                            # Check the 8 neighbors of each pixel.
                            for k in [row-1, row, row+1]:
                                for l in [col-1, col, col+1]:
                                    # if x and y is in the dimension of image.
                                    if (0 <= k < height) and (0 <= l < width):
                                        # Stack can be used to determined how many neighbors are matched.
                                        # If the value of the element is not 0 and not visited yet.
                                        if ((img_process[k, l] != 0) and (label_visited[k, l] == 0)):
                                            stack.push((k, l))
                        # Add 1 to label number.
                        label_num += 1

    # Creation of a stack.
    rectangles = Stack()
    # Check each label in label_count.
    for label_name, n in enumerate(label_count):
        # Pick up the labels which have at least 500 elements.
        if (n >= threshold_region):
            # print(label_name)
            # print(n)
            # The position of rectangle.
            rectRight = width
            rectLeft = 0
            rectTop = height
            rectBottom = 0
            # Sum of x and y value of each element.
            total_x = 0
            total_y = 0
            # Processing image of each pixel.
            for y in range(height):
                for x in range(width):
                    # Check whether the pixel is the label or not.
                    if (label_list[y, x] == label_name):
                        # Update of the sum of x and y value of each element.
                        total_x += x
                        total_y += y
                        # Update x and y value of each element if it has smaller value.
                        if (x > rectLeft):
                            rectLeft = x
                        if (x < rectRight):
                            rectRight = x
                        if (y < rectTop):
                            rectTop = y
                        if (y > rectBottom):
                            rectBottom = y
            # Calculation of the center of gravity
            middle_point_x = total_x // n
            middle_point_y = total_y // n
            # print(middle_point_x, middle_point_y)
            # Push the information of the rectangle to the stack.
            rectangles.push((rectLeft, rectRight, rectTop, rectBottom, middle_point_x, middle_point_y))

    # Draw bounding box and + on image.
    while not rectangles.isEmpty():
        # Assign the information of the rectangle, then remove it from the stack.
        rectLeft, rectRight, rectTop, rectBottom, middle_point_x, middle_point_y = rectangles.pop()
        # Draw the rectangle.
        cv2.rectangle(binary_img, (rectLeft, rectTop), (rectRight, rectBottom), (0, 0, 255), 2)
        # Draw the +.
        cv2.line(binary_img, (middle_point_x - 10, middle_point_y), (middle_point_x + 10, middle_point_y ),(0, 0, 255), 2 )
        cv2.line(binary_img, (middle_point_x, middle_point_y - 10), (middle_point_x, middle_point_y + 10 ),(0, 0, 255), 2 )
        # Save the image.
        cv2.imwrite('connectedImage.jpg', binary_img)

    return img

img = cv2.imread("lena.bmp")
img_binary = binary_image(img)
cv2.imwrite('binary_img.jpg', img_binary)
histogram(img)
Connection_component(img)
```

程式碼簡介:

一開始先引入 OpenCV、NumPy 和 matplotlib 的 Python 模組。OpenCV 主要是用於讀取以及寫入圖片檔(cv2.imread/cv2.imwrite)；NumPy 函式庫用以實現快速操作多維陣列的運算。其中，img.shape 指令用於得知圖片的三個維度，前兩個維度依序為圖片的高度與寬度，第三個維度則是圖片的 channel。Matplotlib 則是用於數值計算庫 NumPy 的繪圖。

a. a binary image (threshold at 128)

使用 3 層 for 迴圈將圖片上的每一個 pixel 進行處理，若是該 pixel 的像素值>=128，並將其像素值設成 255;若是該 pixel 的像素值並非>=128，則將此像素值設成 0。

b. a histogram

首先，建立一個大小為 256 的一維矩陣，使用 2 層 for 迴圈讀取圖片上每一個 pixel 的像素值，並統計每一個像素值的數量，最後再透過 Matplotlib 的函式繪製出數量對像素值的長條圖(x 軸:像素值、y 軸:數量)。

c. connected components (regions with + at centroid, bounding box)-(8-connectivity)

採用八連通以及 Seed Filling 進行實作，整體過程中需要特別注意的是透過 stack 進行八連通的操作。其中，針對面積 500 以外的數值進行過濾 conneted component，結果顯示有五個 bounding box 以及相同 label 的重心。具體的實作細節如 code 之註解所示。