Computer Vision 2019 Fall

Homework #2

B06902059 資工三 謝宜儒

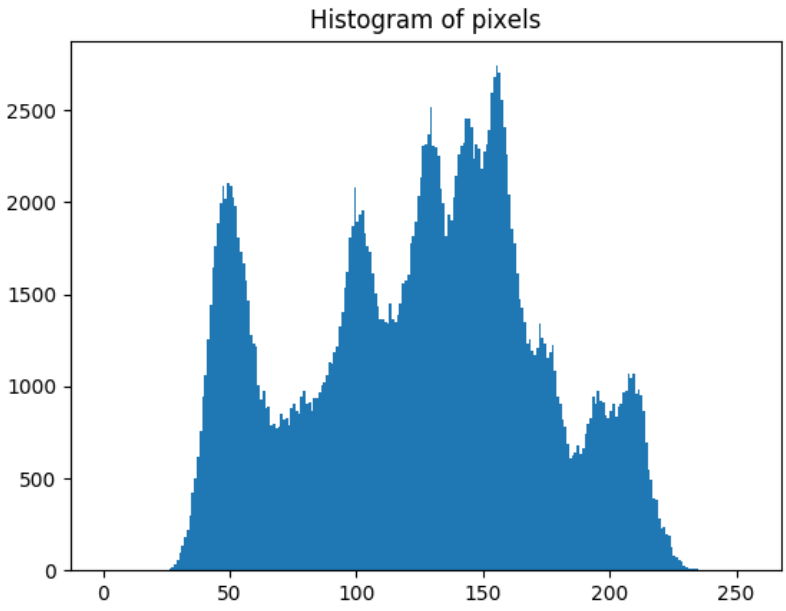
Description

This homework focuses on some basic image manipulations, which are done pixel-wise.

Results

(a) a binary image (threshold at 128)

(b) a histogram



(c) connected components (regions with circles at centroids and bounding boxes)

The connected components are 4-connected.

The centroids are marked by blue circles and the bounding boxes are marked by green rectangles.

Algorithm: Use iterative BFS to find all the connected components and record the coordinates of the borders during the BFS process.

Source Code (fragment)

```
# (a) Binary image
def plot1(img):
  binary_128_image = img.copy()
  threshold = 128
  for i in range(length):
    for j in range(width):
      if(img[i][j][0] < threshold):
        binary_128_image[i][j] = [0, 0, 0]
```

```
      else:
        binary_128_image[i][j] = [255, 255, 255]

  cv2.imwrite('01_binary_128.bmp', binary_128_image)
  cv2.imshow('Binary at 128 image', binary_128_image)
  cv2.waitKey(0)
  cv2.destroyAllWindows()

# (b) Histogram
def plot2(img):
  pixels = []
  for i in range(length):
    for j in range(width):
      pixels.append(img[i][j][0])

  plt.hist(pixels, bins = range(0, 256))
  plt.title('Histogram of pixels')
  plt.savefig('02_histogram.png')
  plt.show()

# (c) Connected components
def plot3(img):
  binary_128_image = img.copy()
  labels = np.zeros((length, width))
  threshold = 128
  for i in range(length):
    for j in range(width):
      if(img[i][j][0] < threshold):
        binary_128_image[i][j] = [0, 0, 0]
      else:
        binary_128_image[i][j] = [255, 255, 255]
        labels[i][j] = 1

  visited = np.zeros((length, width))
  for i in range(length):
    for j in range(width):
      if(labels[i][j] == 1 and visited[i][j] == 0):
        pixel_count = 0
        x_sum, y_sum = i, j
        stack = []
        stack.append([i, j])
        visited[i][j] = 1
        left, right, upper, lower = 0, width - 1, i, length - 1
        while(len(stack) != 0):
          v = stack.pop()
          vi, vj = v[0], v[1]
          x_sum += vi
          y_sum += vj
          pixel_count += 1
          left, right = max(vj, left), min(vj, right)
          upper, lower = max(vi, upper), min(vi, lower)
```

```
          if(vj > 0 and labels[vi][vj - 1] == 1 and visited[vi][vj - 1] !=
1):
            stack.append([vi, vj - 1])
            visited[vi][vj - 1] = 1
          if(vj < width - 1 and labels[vi][vj + 1] == 1 and visited[vi][vj +
1] != 1):
            stack.append([vi, vj + 1])
            visited[vi][vj + 1] = 1
          if(vi > 0 and labels[vi - 1][vj] == 1 and visited[vi - 1][vj] !=
1):
            stack.append([vi - 1, vj])
            visited[vi - 1][vj] = 1
          if(vi < length - 1 and labels[vi + 1][vj] == 1 and visited[vi + 1]
[vj] != 1):
            stack.append([vi + 1, vj])
            visited[vi + 1][vj] = 1


        if(pixel_count >= 500):
          cv2.rectangle(binary_128_image, (left, upper), (right, lower),
            (0, 255, 0), 3)
          centroid_x, centroid_y = int(x_sum / pixel_count), int(y_sum /
pixel_count)
          cv2.circle(binary_128_image, (centroid_y, centroid_x), 10, (255, 0,
0), 3)



  cv2.imwrite('03_connected_component.bmp', binary_128_image)
  cv2.imshow('Connected component with borders', binary_128_image)
  cv2.waitKey(0)
  cv2.destroyAllWindows()
```

To run the source code, type the following line in a terminal:

```
python3 hw2.py [input image] [problem number]
```

Reference

https://blog.gtwang.org/programming/opencv-drawing-functions-tutorial/