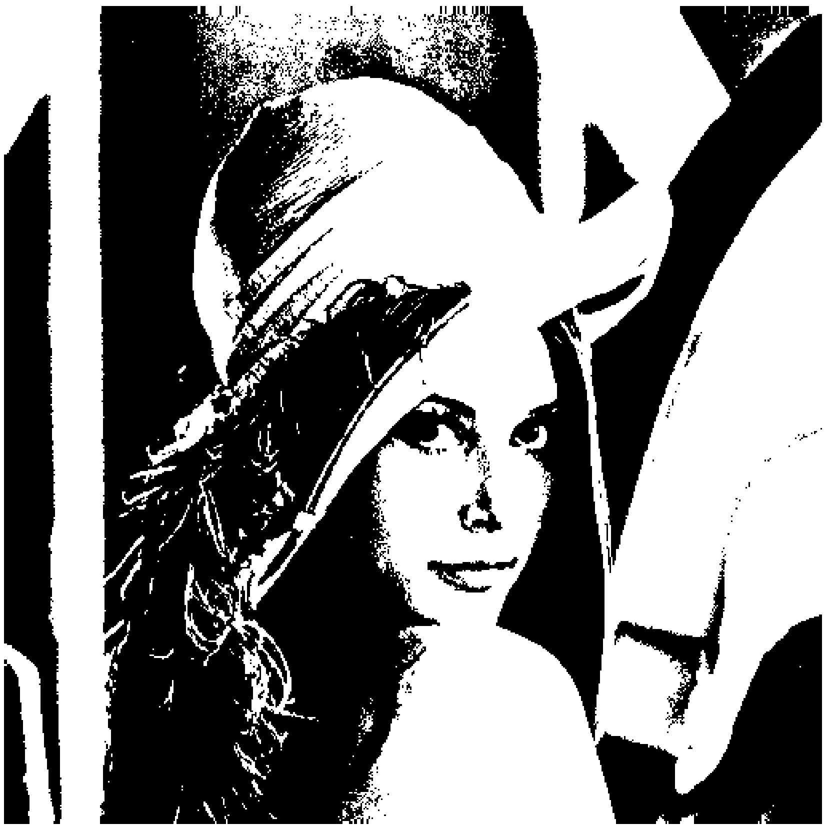
Computer Vision HW2

R08922a27 資工系 人工智慧碩士班 李吉昌

I use python 3.7 to implement all image processing requirements. Reading .bmp file by PIL, coloring and drawing by OpenCV(cv2), and then processing through NumPy array.

• (a) a binary image (threshold at 128)

1. Results



2. Code fragment

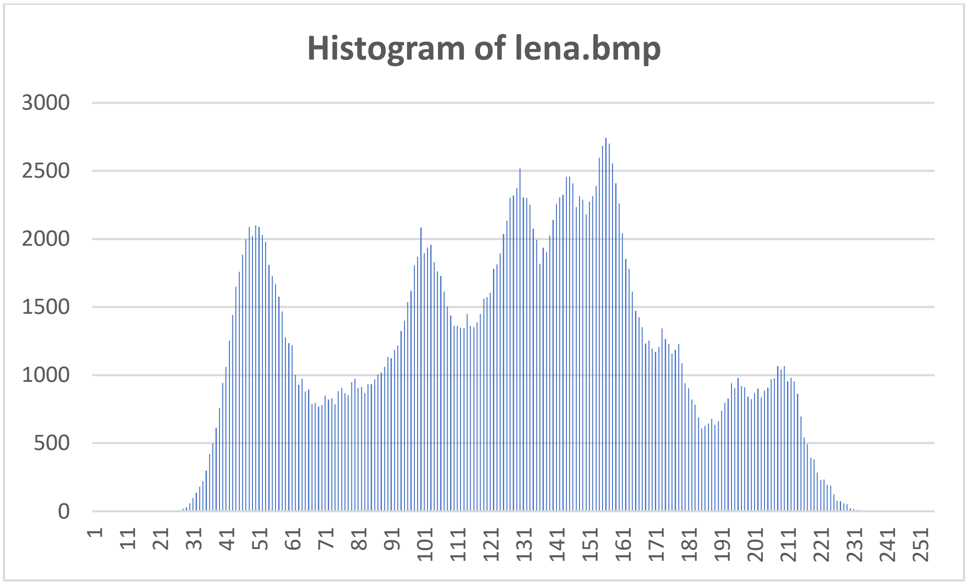
```
binarize = np.zeros(sample_arr.shape)
binarize[sample_arr > 127] = 255
PIL_image = Image.fromarray(binarize.astype('uint8'))
PIL_image.save('results/BinarizeAt128.bmp')
```

3. Brief description

Using Boolean operation to find out the indexes with which level values are higher/smaller than 128, assign 0/255 to those places.

• (b) a histogram

1. Results



2. Code fragment

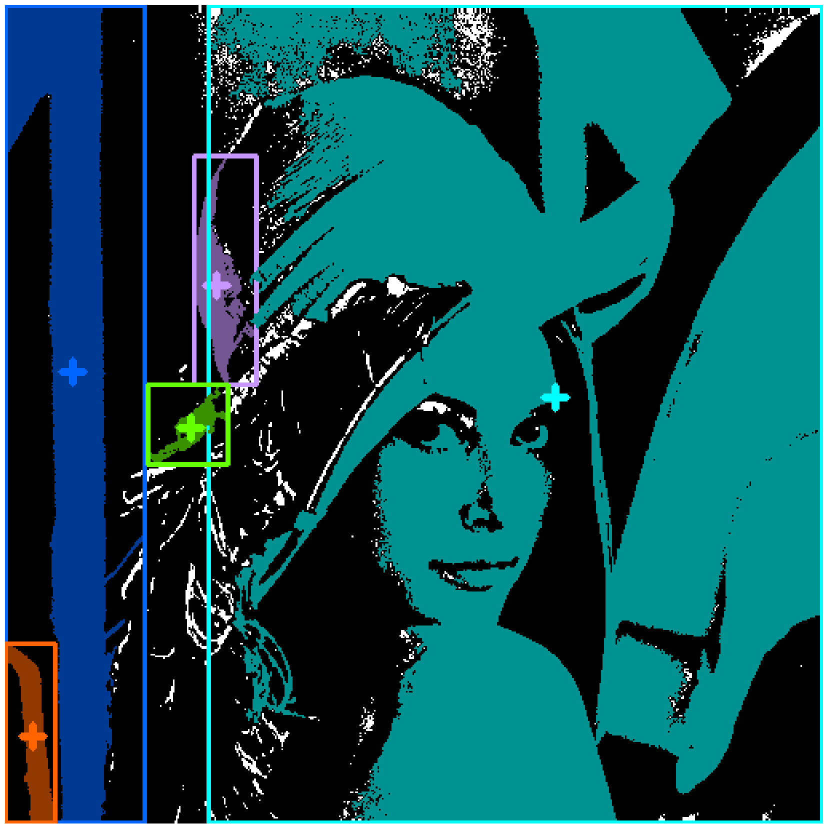
```
counter = np.zeros(255).astype(int)
for i in range(sample_arr.shape[0]):
    for j in range(sample_arr.shape[1]):
        counter[sample_arr[i, j]] += 1
np.savetxt("results/histogram.csv", counter, fmt='%d',delimiter=",")
```

3. Brief description

An integer array is used to count the numbers of each grey level; each level value corresponds to the index of each element.

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

1. Results



2. Code fragment

```
IsVisit = np.zeros(sample_arr.shape, dtype=bool)
IsVisit[sample_arr < 128] = True


def BFS(i, j, record, l):
    if IsVisit[i, j] == False:
        IsVisit[i, j] = True
        record.append((i, j))
        if i > 0 and IsVisit[i - 1, j] == False:
            BFS(i - 1, j, record, l+1)

        if i < IsVisit.shape[0] - 1 and IsVisit[i + 1, j] == False:
            BFS(i + 1, j, record, l+1)

        if j > 0 and IsVisit[i, j - 1] == False:
            BFS(i, j - 1, record, l+1)

        if j < IsVisit.shape[1] - 1 and IsVisit[i, j + 1] == False:
            BFS(i, j + 1, record, l+1)
    else:
        return


colors = [[255, 100, 0], [255, 150, 200], [
    255, 255, 0], [0, 255, 100], [0, 100, 255]]
color_idx = 0
draw = cv2.cvtColor(sample_arr[:], cv2.COLOR_GRAY2BGR)
draw[draw > 127] = 255
draw[draw < 128] = 0
for i in range(IsVisit.shape[0]):
    for j in range(IsVisit.shape[1]):
        record = []
        BFS(i, j, record, 0)
        if len(record) > 500:
            X_min, Y_min = sys.maxsize, sys.maxsize
            X_max, Y_max = 0, 0
            X, Y = [], []
            for (i, j) in record:
                Y_min = min(Y_min, i)
                Y_max = max(Y_max, i)
                X_min = min(X_min, j)
                X_max = max(X_max, j)
                Y.append(i)
                X.append(j)
                draw[i, j] = np.array(colors[color_idx]) // 1.75

            X, Y = int(np.mean(X)), int(np.mean(Y))
            cv2.line(draw, (X - 7, Y), (X + 7, Y), colors[color_idx], 3)
            cv2.line(draw, (X, Y - 7), (X, Y + 7), colors[color_idx], 3)
            cv2.rectangle(draw, (X_min, Y_min),
                          (X_max, Y_max), colors[color_idx], 2)
            color_idx += 1

cv2.imwrite('results/ConnectedComponents.bmp', draw)
```

3. Brief description

I select The BFS function is defined for collecting all the **4-connected neighborhoods** belonging to the same component candidate. Those pixels which have been visited or the level are lower than 127 would be marked as "visited" so that those pixels would never be revisited. The record list provides all the pixels' places in the same component candidate in each searching trial. The area of a candidate would then be used as a criterion to determine whether it is a connected component. The cv2 package is imported for drawing the "+" marks and the bounding boxes, one color for one component.