Harbin Institute of Technology (Shenzhen)

Image Processing Project Report

Experiment Name: **Image enhancement & Morphological Algorithms**

Student Name: **许峻玮**

Student ID: **190110105**

Report Date: **2021-10-27**

Content

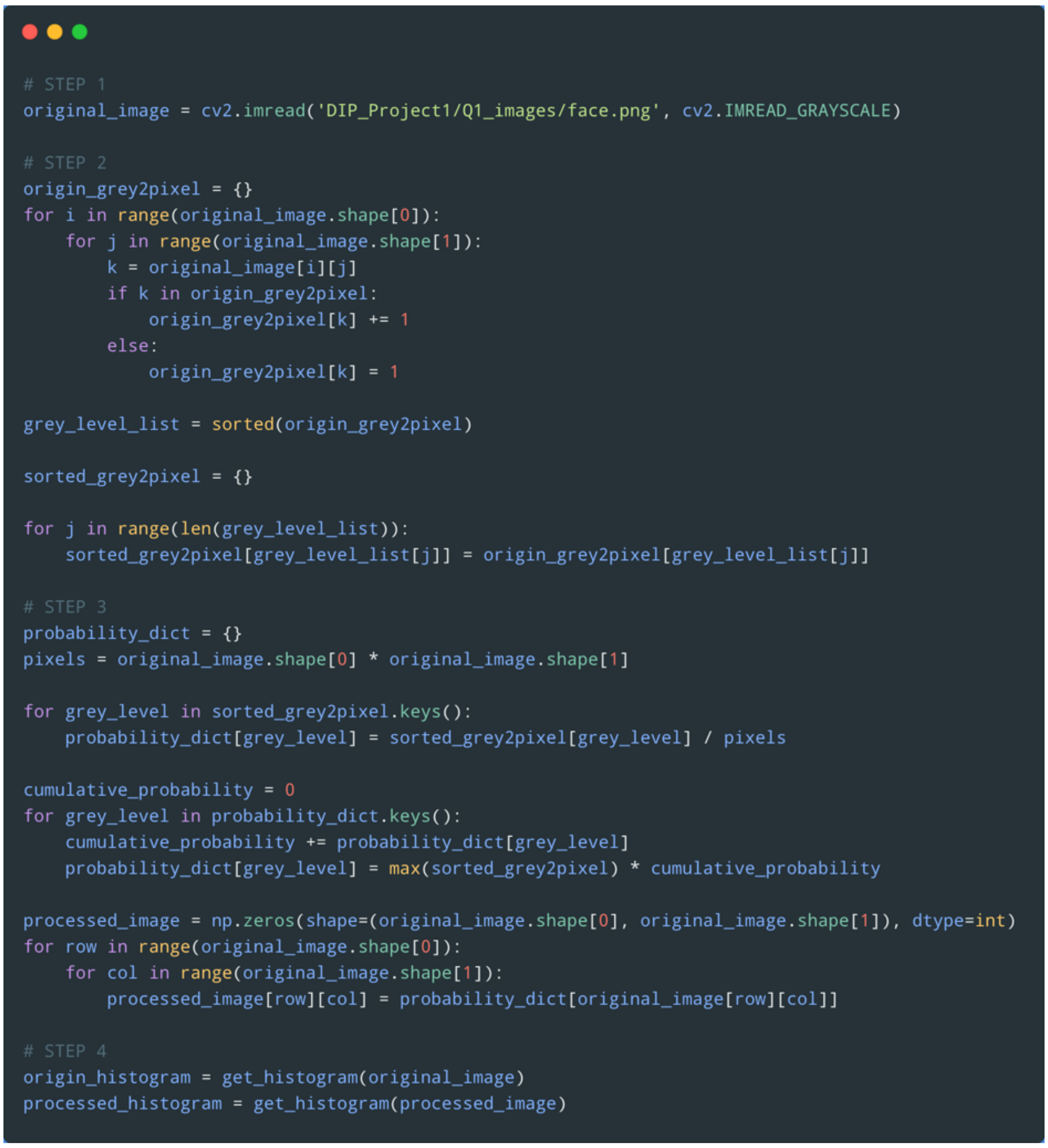
1 Project Content

1. Image enhancement
    a) In this experiment, we need to use certain methods to process the photos in order to make their pixel distribution more balanced.
    b) In this experiment, we need to reduce the salt noise in the given picture using medium filter denoising. While denoising, the clarity of the school emblem should be kept.
2. Morphological Algorithms in The Catcher in the Rye

    In this experiment, we need to fill the holes of all characters in the given images.

2 Method Description

1. Image enhancement
    a) Picture below basically show how it works.



```python
# STEP 1
original_image = cv2.imread('DIP_Project1/Q1_images/face.png', cv2.IMREAD_GRAYSCALE)

# STEP 2
origin_grey2pixel = {}
for i in range(original_image.shape[0]):
    for j in range(original_image.shape[1]):
        k = original_image[i][j]
        if k in origin_grey2pixel:
            origin_grey2pixel[k] += 1
        else:
            origin_grey2pixel[k] = 1

grey_level_list = sorted(origin_grey2pixel)

sorted_grey2pixel = {}

for j in range(len(grey_level_list)):
    sorted_grey2pixel[grey_level_list[j]] = origin_grey2pixel[grey_level_list[j]]

# STEP 3
probability_dict = {}
pixels = original_image.shape[0] * original_image.shape[1]

for grey_level in sorted_grey2pixel.keys():
    probability_dict[grey_level] = sorted_grey2pixel[grey_level] / pixels

cumulative_probability = 0
for grey_level in probability_dict.keys():
    cumulative_probability += probability_dict[grey_level]
    probability_dict[grey_level] = max(sorted_grey2pixel) * cumulative_probability

processed_image = np.zeros(shape=(original_image.shape[0], original_image.shape[1]), dtype=int)
for row in range(original_image.shape[0]):
    for col in range(original_image.shape[1]):
        processed_image[row][col] = probability_dict[original_image[row][col]]

# STEP 4
origin_histogram = get_histogram(original_image)
processed_histogram = get_histogram(processed_image)
```

There are **four** steps when processing.

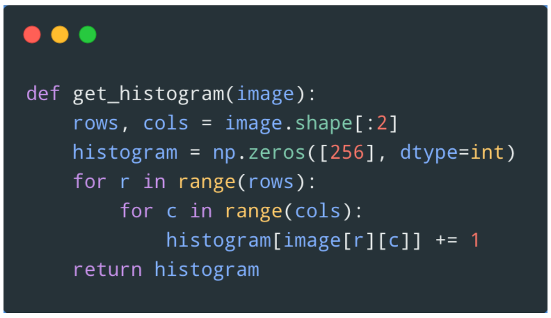
First, read the image and get the gray level matrix of the image.

Second, calculate the gray histogram of the original image. To achieve this, we count how many pixels does every grey level have. Then sort them by grey level from low to high, putting them in a dict. Then we get a dict, where the counts of every grey level appeared in the picture are stored.

Third, make histogram equalization. Specifically, a probability distribution mapping dict should be constructed to calculate the cumulative distribution function (a.k.a. CDF). According to the CDF, we can easily get the gray level matrix of the processed image.

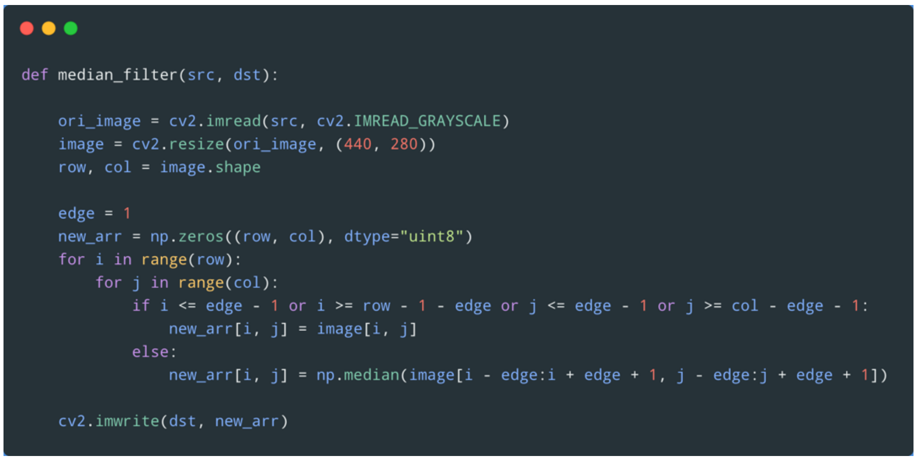
Fourth, get two grey level histograms and draw them.

Additionally, following picture shows how we get the grey level histograms.



```python
def get_histogram(image):
    rows, cols = image.shape[:2]
    histogram = np.zeros([256], dtype=int)
    for r in range(rows):
        for c in range(cols):
            histogram[image[r][c]] += 1
    return histogram
```

b) Picture below basically show how it works.

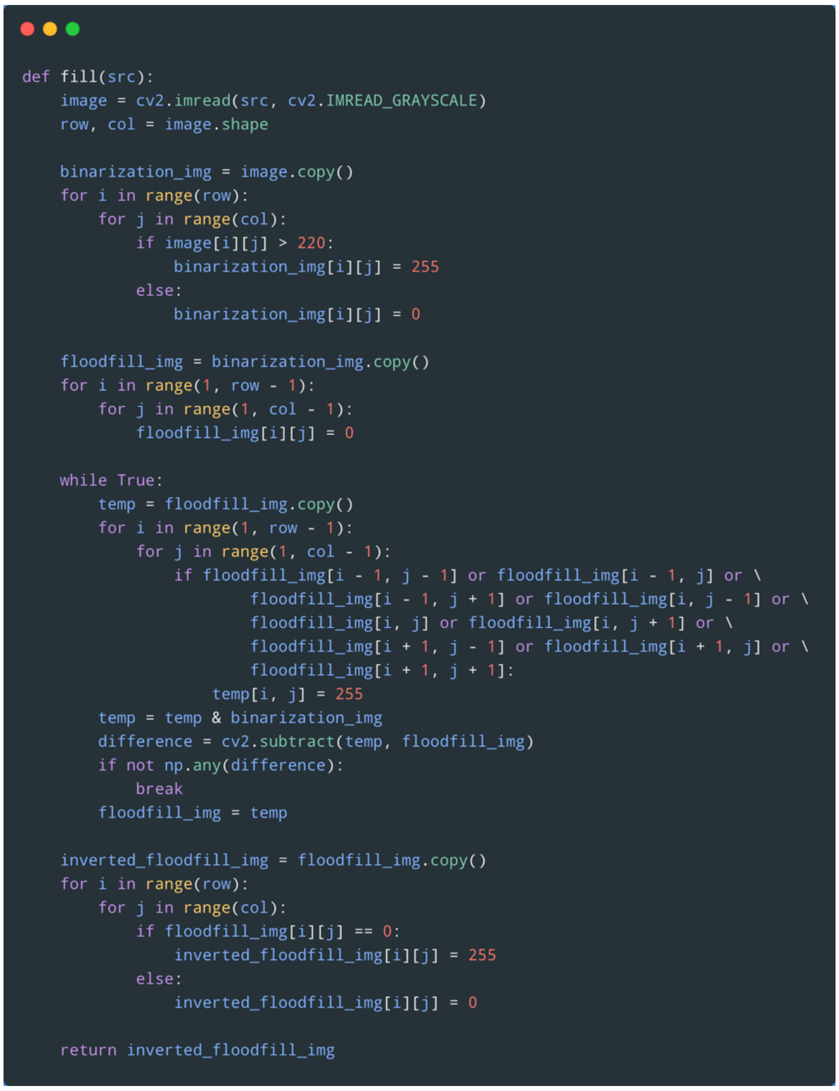


```python
def median_filter(src, dst):

    ori_image = cv2.imread(src, cv2.IMREAD_GRAYSCALE)
    image = cv2.resize(ori_image, (440, 280))
    row, col = image.shape

    edge = 1
    new_arr = np.zeros((row, col), dtype="uint8")
    for i in range(row):
        for j in range(col):
            if i <= edge - 1 or i >= row - 1 - edge or j <= edge - 1 or j >= col - edge - 1:
                new_arr[i, j] = image[i, j]
            else:
                new_arr[i, j] = np.median(image[i - edge:i + edge + 1, j - edge:j + edge + 1])

    cv2.imwrite(dst, new_arr)
```

Getting this done isn't harder than former problem. We still need to read the image and get the gray level matrix of the image. According to the prompt, the image should be resized to 440*280. Then we traverse every single pixel —— ignore it if it's on the boundary, otherwise, pick all eight pixels near it and take the median grey level of the nine pixels as the center pixel's grey level. All we have to do is just above.

2. Morphological Algorithms in The Catcher in the Rye

Picture below basically show how it works.



```python
def fill(src):
    image = cv2.imread(src, cv2.IMREAD_GRAYSCALE)
    row, col = image.shape

    binarization_img = image.copy()
    for i in range(row):
        for j in range(col):
            if image[i][j] > 220:
                binarization_img[i][j] = 255
            else:
                binarization_img[i][j] = 0

    floodfill_img = binarization_img.copy()
    for i in range(1, row - 1):
        for j in range(1, col - 1):
            floodfill_img[i][j] = 0

    while True:
        temp = floodfill_img.copy()
        for i in range(1, row - 1):
            for j in range(1, col - 1):
                if floodfill_img[i - 1, j - 1] or floodfill_img[i - 1, j] or \
                        floodfill_img[i - 1, j + 1] or floodfill_img[i, j - 1] or \
                        floodfill_img[i, j] or floodfill_img[i, j + 1] or \
                        floodfill_img[i + 1, j - 1] or floodfill_img[i + 1, j] or \
                        floodfill_img[i + 1, j + 1]:
                    temp[i, j] = 255
        temp = temp & binarization_img
        difference = cv2.subtract(temp, floodfill_img)
        if not np.any(difference):
            break
        floodfill_img = temp

    inverted_floodfill_img = floodfill_img.copy()
    for i in range(row):
        for j in range(col):
            if floodfill_img[i][j] == 0:
                inverted_floodfill_img[i][j] = 255
            else:
                inverted_floodfill_img[i][j] = 0

    return inverted_floodfill_img
```

We need **five** steps to process.

First, read the image and get the gray level matrix of the image as usual.

Second, binarize the image. To achieve this, we need to set values equal to or above 220 to 255, while setting values below 220 to 0.

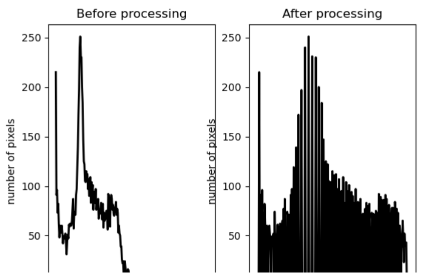
Third, construct a mask matrix. While the boundary is the same as the binarized image, its inner is completely black.

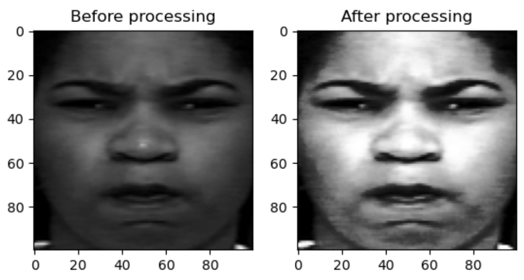
Fourth, make a floodfill! This process won't finish until all pixels cannot floodfill to its eight directions.

Fifth, invert floodfilled image as we do in step 2, which is the simplest step.

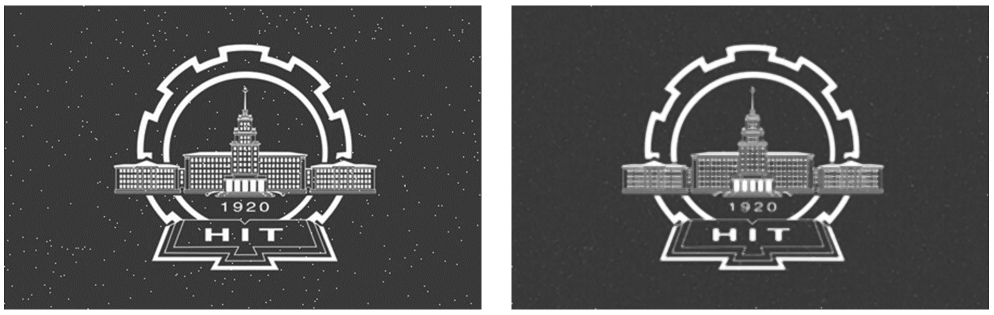
3 Experiment Results and Analysis

1. Image enhancement
   a) Apart from the processed image, we've also got two grey level histograms of the images before and after processing.
   Comparing to the two histograms, we clearly see the effect of our code. With the two images,



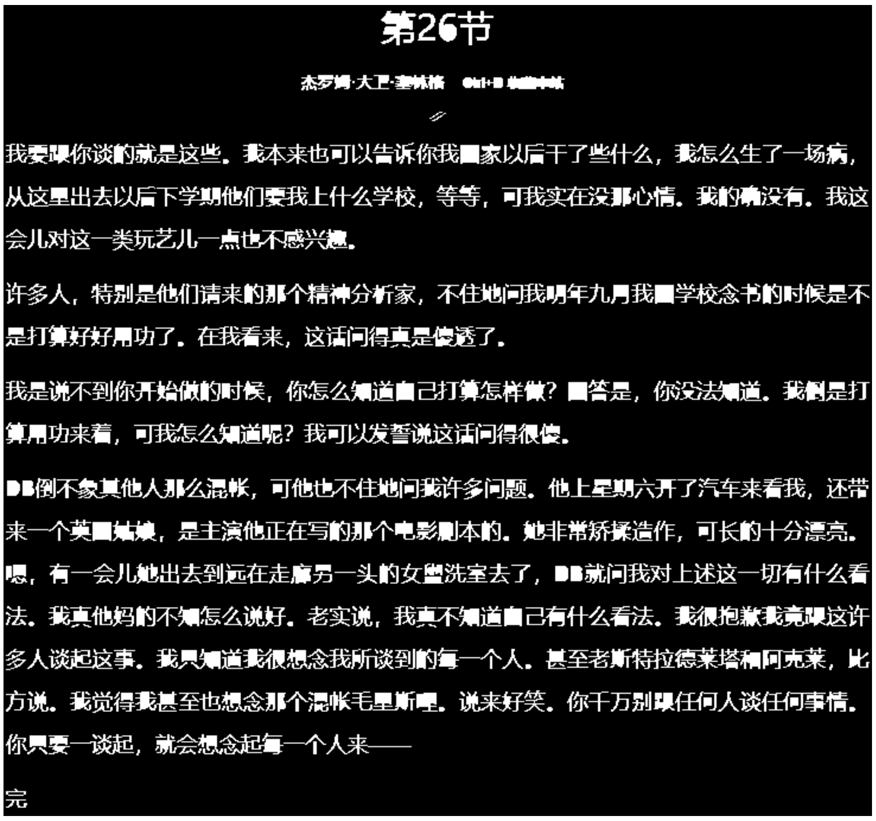


   the result is more convincing. I can proudly conclude that this experiment is much more successful than I thought.
   b) Comparing to the original image, most salt noise were reduced. Although there are still some evident traces. The definition of the image drops a little but not too much, which is acceptable. Speaking of what aspect can we improve to process better, I think the adaptive median filter should be used. Then an image with less salt noise and more definition will replace the current one.



2. Morphological Algorithms in The Catcher in the Rye

We successfully fill the holes in characters and get two black pictures. Basically, I'm satisfied with the display effect.



第26节

杰罗姆·大卫·塞林格 [illegible]

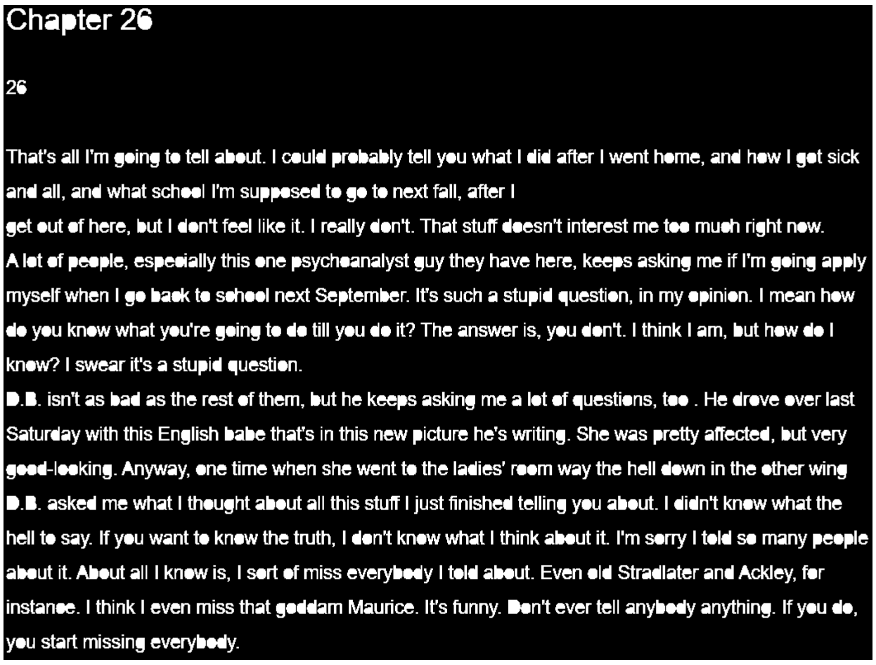
我要跟你谈的就是这些。我本来也可以告诉你我回家以后干了些什么，我怎么生了一场病，从这里出去以后下学期他们要我上什么学校，等等，可我实在没那心情。我的确没有。我这会儿对这一类玩艺儿一点也不感兴趣。

许多人，特别是他们请来的那个精神分析家，不住地问我明年九月我回学校念书的时候是不是打算好好用功了。在我看来，这话问得真是傻透了。

我是说不到你开始做的时候，你怎么知道自己打算怎样做？回答是，你没法知道。我倒是打算用功来着，可我怎么知道呢？我可以发誓说这话问得很傻。

DB倒不象其他人那么混帐，可他也不住地问我许多问题。他上星期六开了汽车来看我，还带来一个英国姑娘，是主演他正在写的那个电影剧本的。她非常矫揉造作，可长的十分漂亮。嗯，有一会儿她出去到远在走廊另一头的女盥洗室去了，DB就问我对上述这一切有什么看法。我真他妈的不知怎么说好。老实说，我真不知道自己有什么看法。我很抱歉我竟跟这许多人谈起这事。我只知道我很想念我所谈到的每一个人。甚至老斯特拉德莱塔和阿克莱，比方说。我觉得我甚至也想念那个混帐毛里斯哩。说来好笑。你千万别跟任何人谈任何事情。你只要一谈起，就会想念起每一个人来——

完



Chapter 26

26

That's all I'm going to tell about. I could probably tell you what I did after I went home, and how I got sick and all, and what school I'm supposed to go to next fall, after I
get out of here, but I don't feel like it. I really don't. That stuff doesn't interest me too much right now.
A lot of people, especially this one psychoanalyst guy they have here, keeps asking me if I'm going apply myself when I go back to school next September. It's such a stupid question, in my opinion. I mean how do you know what you're going to do till you do it? The answer is, you don't. I think I am, but how do I know? I swear it's a stupid question.
D.B. isn't as bad as the rest of them, but he keeps asking me a lot of questions, too . He drove over last Saturday with this English babe that's in this new picture he's writing. She was pretty affected, but very good-looking. Anyway, one time when she went to the ladies' room way the hell down in the other wing D.B. asked me what I thought about all this stuff I just finished telling you about. I didn't know what the hell to say. If you want to know the truth, I don't know what I think about it. I'm sorry I told so many people about it. About all I know is, I sort of miss everybody I told about. Even old Stradlater and Ackley, for instance. I think I even miss that goddam Maurice. It's funny. Don't ever tell anybody anything. If you do, you start missing everybody.

4 Summary

1. What difficulties I encountered?

   There are few instructions guiding me to achieve every step, which suffered me a lot debugging the code. Comparing to other experiment such as *Operating System* and *Database System*, we've got a concrete instruction and a satisfying experimental environment. Many problems are listed in the guidebook ahead of time in case we need it. Additionally, there are teachers and TAs helping us finish the experiment on class, while we lamentedly don't have any class hours.
2. What did I learn from this experiment?

   I'm more proficient in DIP of course, especially using *Python* to achieve image enhancement and morphology algorithms. Last but not least, my ability to code and debug enhanced a lot while facing and fighting countless bugs, English writing is absolutely more fluent, too :)