IMAGE SEGMENTATION

October 27, 2019

Zijing Ou
16307050
Sun Yat-Sen University
School of Data and Computer Science

Contents

1 FOREGROUND AND BACKGROUND SEGMENTATION

1.1 OTSU

Otsu's method is an adaptive thresholding way for binarization in image processing. By going through all possible threshold values (from 0 to 255), it can find the optimal threshold value of input image.

Otsu's algorithm tries to find a threshold value (t) which minimizes the weighted within-class variance given by the relation:

$$\sigma_w^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t)$$

where

$$q_1(t) = \sum_{i=1}^{t} P(i) \& q_2(t) = \sum_{i=t+1}^{I} P(i)$$

$$\mu_1(t) = \sum_{i=1}^{t} \frac{iP(i)}{q_1(t)} \& \quad \mu_2(t) = \sum_{i \neq t+1}^{I} \frac{iP(i)}{q_2(t)}$$

$$\sigma_1^2(t) = \sum_{i=1}^{t} \left[i - \mu_1(t)\right]^2 \frac{P(i)}{q_1(t)} \& \sigma_2^2(t) = \sum_{i}^{I} \left[i - \mu_2(t)\right]^2 \frac{P(i)}{q_2(t)}$$

It actually finds a value of t which lies in between two peaks such that variances to both classes are minimal.

1.2 implement

```
void OSTU::getThreshold() {
   double variance = 0, P1, P2, m1, m2;
   double histogram[256];

  for (int i = 0; i < 256; i++) {
    histogram[i] = 0;
  }
  int pixelsNum = image._width * image._height;

  cimg_forXY(image, i, j) {
    ++histogram[int(image(i, j, 0))];
  }
```

```
  for (int i = 0; i < 256; i++) {
    P1 = 0; P2 = 0; m1 = 0; m2 = 0;

    for (int j = 0; j <= i; j++) {
      P1 += histogram[j];
      m1 += j * histogram[j];
    }
    if (P1 == 0) continue;
    m1 /= P1;
    P1 /= pixelsNum;

    for (int j = i + 1; j < 256; j++) {
      P2 += histogram[j];
      m2 += j * histogram[j];
    }
    if (P2 == 0) continue;
    m2 /= P2;
    P2 /= pixelsNum;

    double temp_variance = P1 * P2 * (m1 - m2) * (m1 - m2);
    if (variance < temp_variance) {
      variance = temp_variance;
      threshold = i;
    }
  }
}
```

1.3 Result

The result of OTSI is showed in Fig.1.

2 DIGITAL SEGMENTATION

2.1 Method

The idea comes down as follow:

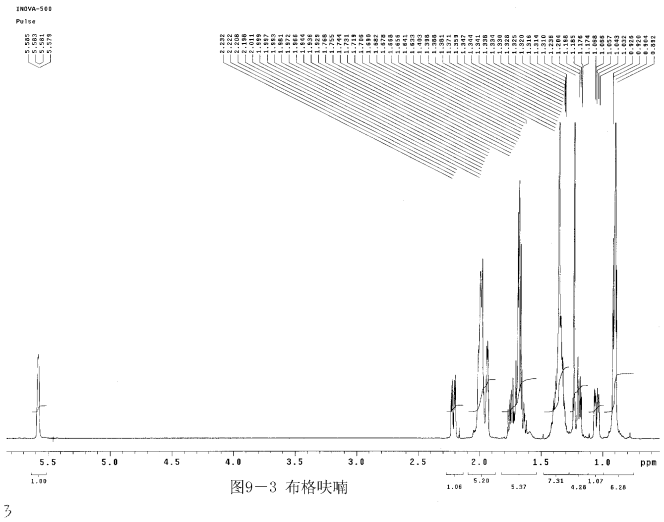

Figure 1: Blockchain Architecture Diagram.

- **Dilate the Original Image** If there are black dots in the eight fields of the pixel, the pixel will be changed to black.

```
    cimg_forXY(out, i, j) {
      bool flag = false;
      for (int x = -n / 2; x <= n / 2; x++) {
        if (flag) break;
        for (int y = -n / 2; y <= n / 2; y++) {
          int xx = i + x, yy = j + y;
          if (InImage(xx, yy) && !in(xx, yy)) {
            flag = true;
            break;
          }
        }
      }
      out(i, j, 0) = out(i, j, 1) = out(i, j, 2) = flag ? 0 :
         255;
    }
}
```

- **Searching for Connected Blocks** In the dilated image, searching for connected blocks, which could be easily realized by using depth-first searching. If the connected block is greater than 80 and less than 300, it is reserved, and vice versa.

- **Digital Segmentation** Obtaining the subscripts of the upper left corner and the lower right corner of each connected block, and then draw the rectangle.

2.2 Result

Dilation

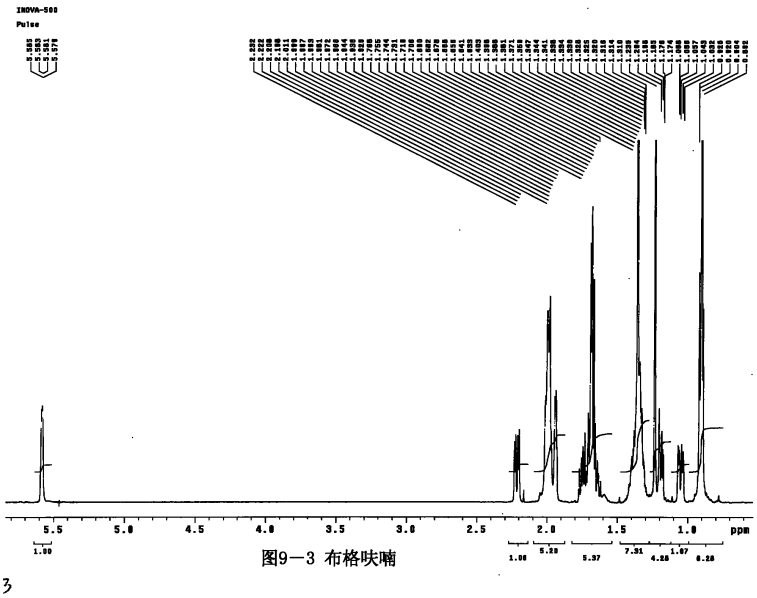

图9—3 布格呋喃

Digital Segmentation

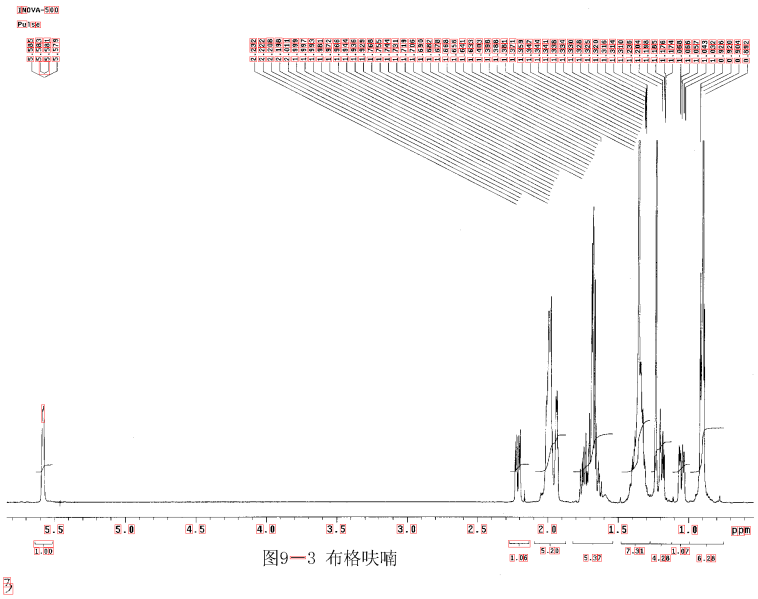

图9—3 布格呋喃

3 OCR

3.1 Axis Recognition

Traverse all connected blocks, if the aspect ratio is greater than 200, it is considered as the coordinate axis. Then boxing it out. The result is showed as follow:

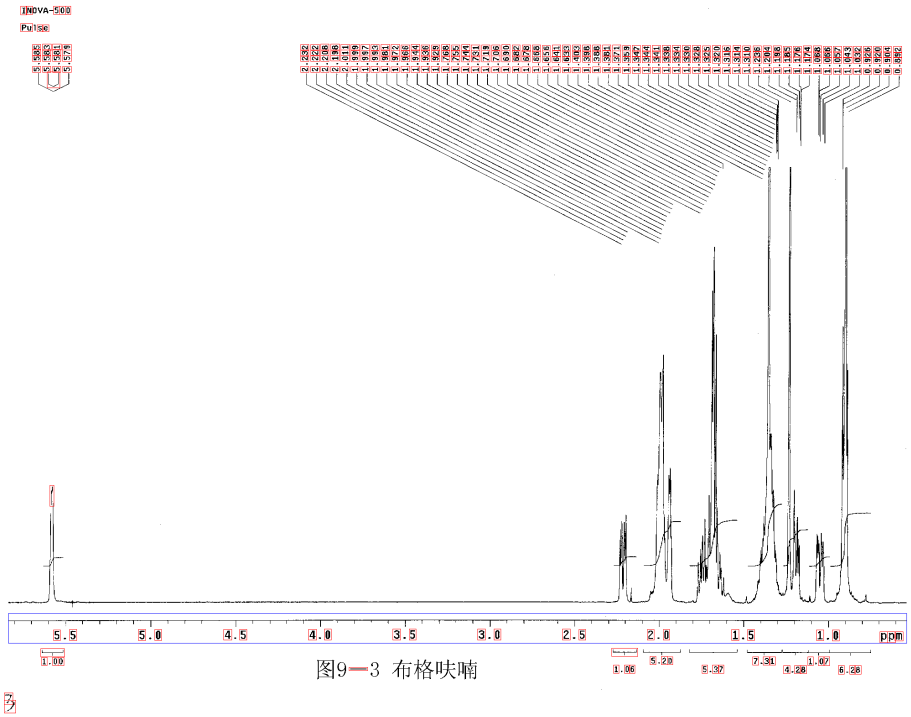

图9—3 布格呋喃

3.2 Digital Segmentation

In this section, I donnot use opencv. Either, I presave 10 pictures arranged 0-9 (as follow).

(a) 0 (b) 1 (c) 2 (d) 3 (e) 4 (f) 5 (g) 6 (h) 7 (i) 8 (j) 9

I resize them in the same size, then calculate the L2 distance, the picture with the smallest distance as the recognized number. The code is available at https://github.com/J-zin/computer-vision/tree/master/04/HW4

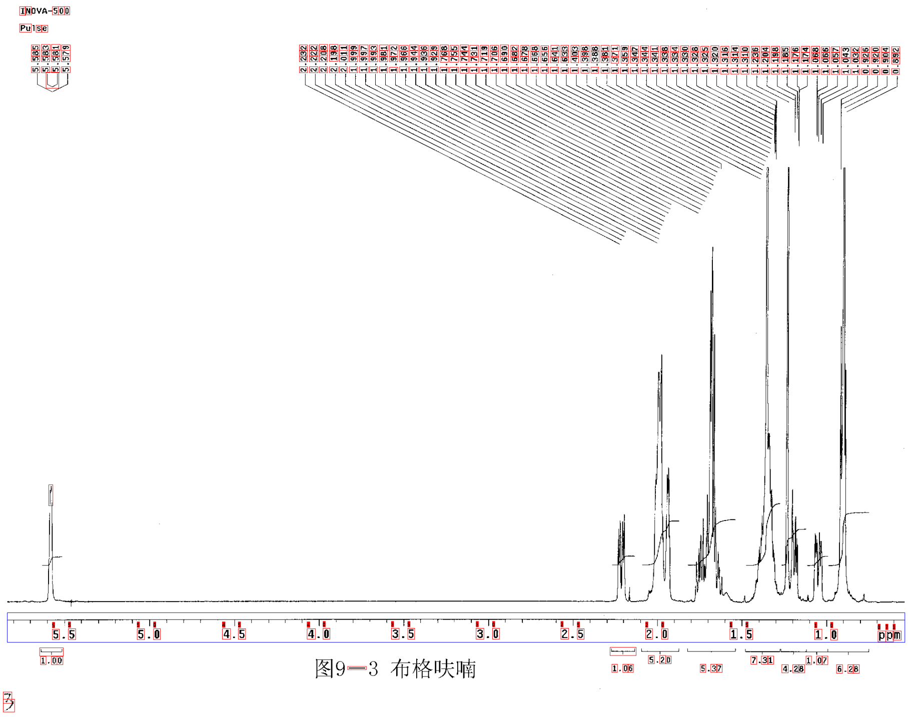

图9—3 布格呋喃

1 ASSIGNMENT

作业 4

输入图像：

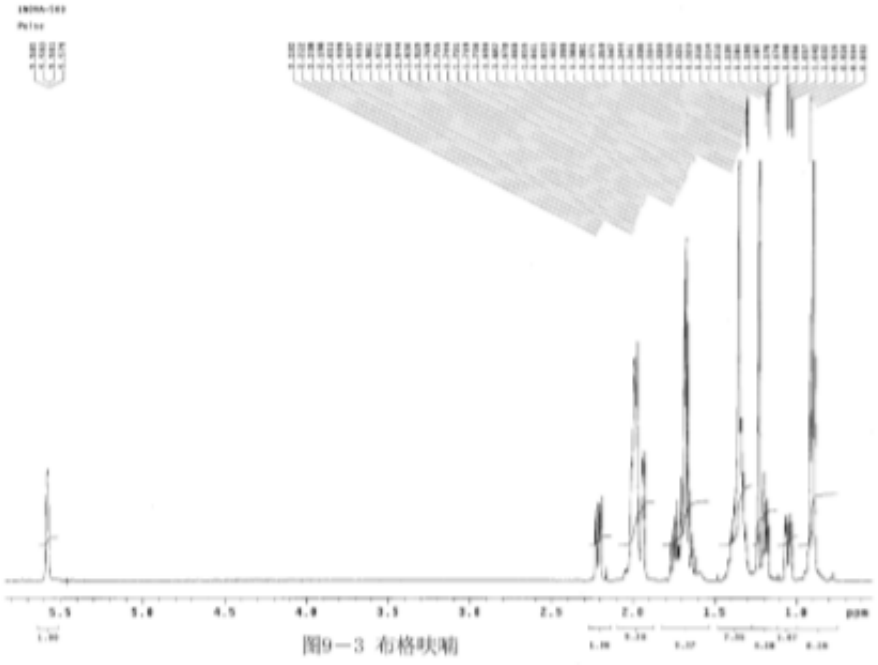

图9－3 布格呋喃

- 任务一： 把图像的前后景分割.
- 任务二： 把图像中的数字切割出来，基本思路如下：
  1. 先做图像的 Delate(膨胀操作).
  2. 求出图像中的连通块.
  3. 去除黑色像素大于 T(T=100)的联通块.
  4. 在原图上把联通块(黑色像素<=T)用红色框标记(即用红色框圈住联通块，如下图的红色框)

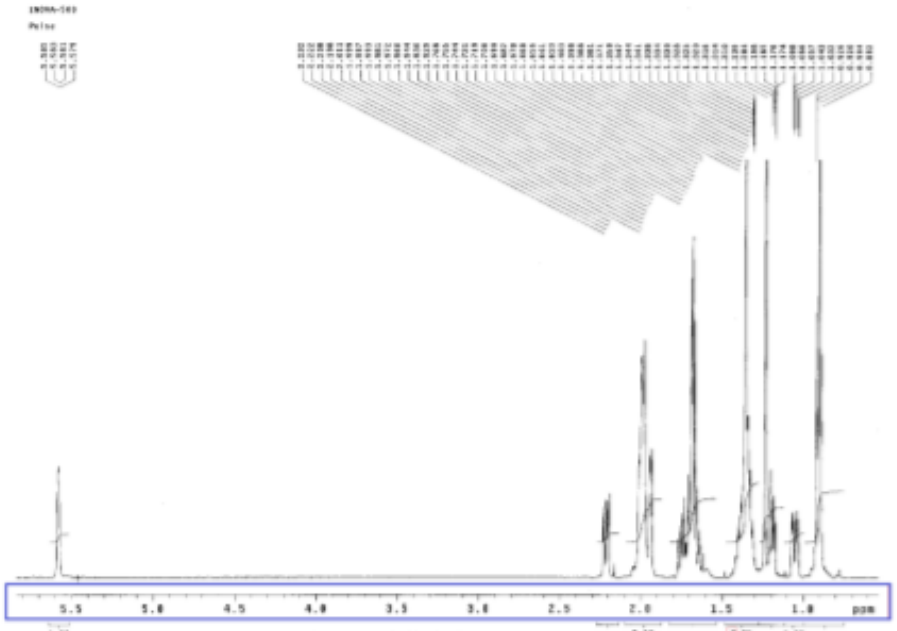

图9－3 布格呋喃

- 任务三： 把图像中标尺OCR，基本思路：

  1. 计算标尺对应的位置和区域(即上图的蓝色框区域).

  2. 识别标尺图像中的刻度数字，可以调用 OpenCV.