National University of Singapore
Facility of Engineering
Department of Mechanical Engineering

Machine Vision Computing Project

ME5405 Assignment AY20/21

Student Name: Sun Zhanhong A0225282J
Fu Yangqing A0225413R
Hu Yaxin A0225408J
Supervisor: Chui Chee Kong (A/Prof)
Lim Kian Meng (A/Prof)

Contents

List of Figures

1. Problem Statement and Solution

1.1. Problem Statement

In this project, we should process two images to meet some requirements. Here are the descriptions of the two images. Image 1 is a 64×64, 32 level images, which is shown a coded array that contains an alphanumeric character for each pixel in the image. The range of these characters is $0-9$ and $A-V$, which corresponds to 32 gray levels. Image 2 is a JPEG image of a label on a microchip whose characters are inverted. We should invert them first before proceed with the following tasks. There are some differences between the two images, so our process will be a little different. Image 1 and image 2 are shown in Figure 1(a) and Figure 1.(b) respectively.

(a) Image 1 (b) Image 2

Figure 1: Images used in this project.

The tasks we need to accomplish are as follows:

- Display the original image on screen.
- Create a binary image using thresholding.
- Segment the image to separate and identify the different characters.
- Rotate the characters in the image about their own respective centroids by 90 degrees clockwise.
- Rotate the characters in the image from Step 4 about their own respective centroids by 35 degrees counterclockwise.
- Determine the outline(s) of characters of the image.
- Determine a one-pixel thin image of the characters.
- Arrange the characters in one line with the sequence: A1B2C3 for Image 1 and 81344100ARHDFS for image 2.

We processed the images in the same order as the tasks, and the report are organized in the same order.

1.2. A Brief Description of the Solution

For task 2, we used Otsu and iteration method to realize image binarization, because Otsu and iterative method are better for binarization results for both image 1 and image 2. Although Kittler and Bernsen algorithms also have achieved good results for image 1, Kittler algorithm could not get result image for image 2 and Bernsen didn not have a good result. For task 3, we use a variety of algorithms for connectivity detection, quadtree, and region growth, among which connectivity detection uses two-pass and BFS algorithms. For task 4 and 5, we used reverse mapping, nearest neighbor interpolation and bilinear interpolation in our rotation algorithm. For task 6, class method and some other operators like Sobel, LoG, Canny are compared and implemented and only found that class method is best for binary images. For task 7, we tried Helditch and some other algorithms, finding that Helditch has the best performance and thus choose Helditch as our algorithm. For task 8, we define the divided parts as structures and rearrange them.

The final process of image 1 is as follows:

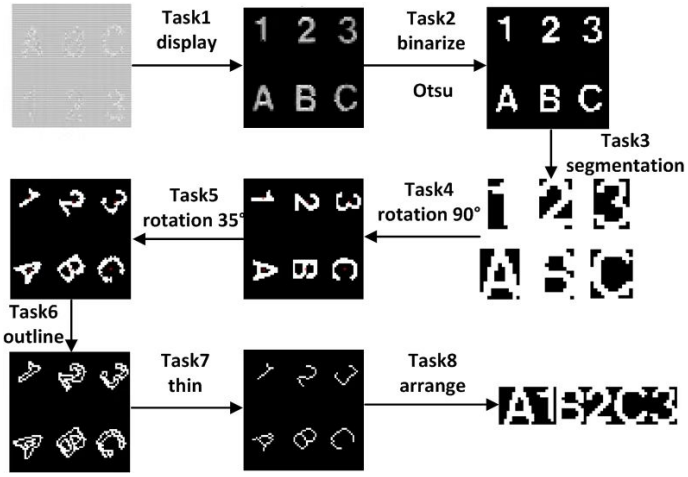


Figure 2: Process of image 1.

The final process of image 2 is belowed:

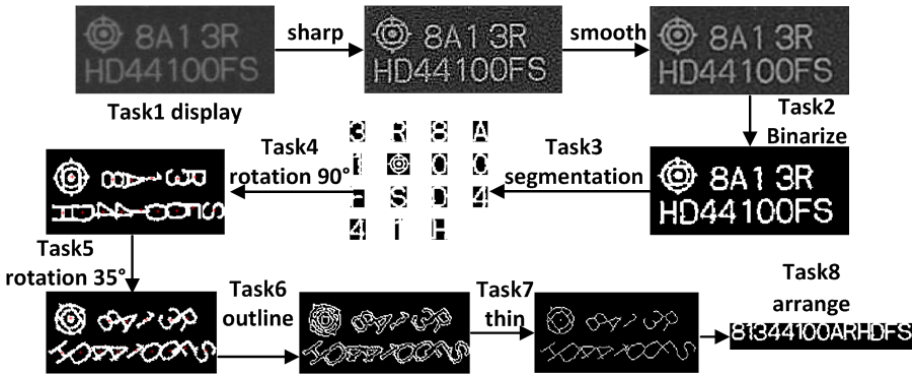


Figure 3: Process of image 2.

2. Binarizing Image Using Thresholding

Image binarization is a process of setting the gray value of pixels on the image to 0 or 255 to present the whole image with an obvious black and white effect. The binarization images are obtained from grayscale images with 256 brightness levels by selecting appropriate thresholds, which can still reflect the overall and local features of the image. Binary image is very important in digital image processing. First of all, image binarization is beneficial to the further processing of the image, making the image simpler, reducing the amount of data, and highlight the outline of the object of interest. Secondly, in order to process and analyze the binary image, the gray image should be binarized to obtain the binarized image. In this way, it is helpful for further processing of the image. The set property of the image is only related to the position of the point with pixel value of 0 or 255, and it no longer involves the multi-level value of the pixel, which makes the processing simple and the data processing and compression amount small. In order to obtain an ideal binary image, closed and connected boundaries are generally used to define non-overlapping regions. All pixels whose gray value is greater than or equal to the threshold value are judged to belong to a specific object, and their gray value is 255; otherwise, these pixels are excluded from the object area, and their gray value is 0, indicating the background or exceptional object area.

Image binarization is divided into global binarization and adaptive (local) binarization. There are many methods for image binarization at present, such as global mean threshold, Bernsen algorithm and OTSU algorithm. Image binarization is performed and the results are compared and analyzed

2.1. Global Mean Threshold

The global mean threshold method is a global processing of image pixels. By searching all pixel values in the image matrix, the mean value of all pixels is calculated, and the mean value is taken as the threshold value. Then, all pixels of the image are divided into two parts. The pixel value above the threshold value is 255, which is white. The pixel value below the threshold is 0 and is black. However, some object pixels or background pixels may be lost. The binarization results cannot reflect the source image information. So we did not use this method.

2.2. Iterative Algorithm

Iterative algorithm is a relatively simple threshold segmentation method, its main idea is to use the initial cycle iteration method of setting the threshold value to approach the optimal threshold gradually. Firstly, the image is grayed, and the initial value of the threshold is set as the average of the maximum R_{max} and minimum R_{min} image grays ($T = (R_{max} + R_{min})/2$). Based on the current threshold T and the gray value of each pixel of image, the image pixels are divided into foreground and background, and the grey value summation, calculate the average grey value foreground and background (u_1, u_2 respectively), and to determine whether a threshold value is equal to the average gray level of the target and the background and the average, if equal, in their average threshold, otherwise, the threshold is set to half of the average gray level and target and the background, to continue the iteration, until the calculated threshold.

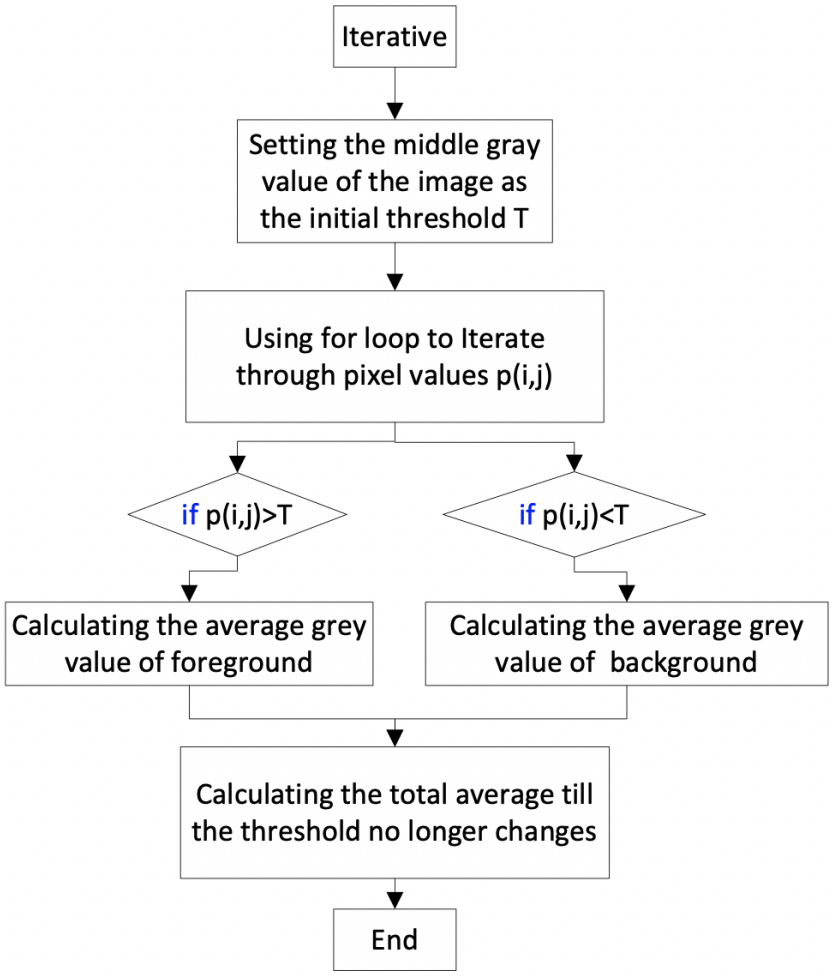


Figure 4: Flow chart of iterative algorithm.

2.3. OTSU Algorithm

OTSU algorithm was proposed by OTSU in 1979. The basic idea is to obtain the optimal threshold value, and then use the threshold value to divide the gray histogram into foreground and background, so as to achieve the maximum variance between two classifications and the minimum variance within a class, that is, the maximum separation between classes and the maximum similarity within a class. Therefore, OTSU method is also called the maximum variance between the largest classes. For the gray histogram of the image, set T to distinguish prospect of gray scale and the background gray binarization threshold, set w_0 to foreground pixels of the image in proportion to the total pixels, makeu_0 prospects for all pixels of the average gray level, set w_1 accounted for the proportion of total image pixels by background points, make u_1 to all the background pixels of the average gray level, the image of the average gray level of all pixels to $u = w_0u_0 + w_1u_1$. During the program operation, the value of T can be traversed successively from the minimum gray value of the image to the maximum gray value of the image. When T takes a certain value, the inter-class variance formula $u = w_0[(u_0 - u)]^2 + w_1[(u_1 - u)]^2$ can get the maximum, and then T is the optimal threshold of binarization.

2.4. Kittler Algorithm

Kittler algorithm is a threshold segmentation method based on histogram. The main idea is to assume that the grayscale image is composed of the target and the background, and the target and the background meet the mixed Gaussian distribution, calculate the mean value and variance of the target and the background, and obtain the minimum error objective function according to the idea of minimum classification error, and take the threshold of the minimum objective function as the optimal threshold. According to this threshold, the image is segmented into binary images.

2.5. Bernsen Algorithm

The principle of the Bernsen algorithm is to obtain the threshold value of the pixel according to the maximum and minimum values of the pixel in the local window where each pixel is located. It is assumed that the maximum value of the pixel gray value in the local window is $max(i, j)$, the minimum value is $min(i, j)$, then the local threshold of the window is $T(i, j) = (max(i, j) + min(i, j))/2$. Each pixel in the image is scanned in order, and T(i,j) is used to obtain the threshold value of the point, and then binarization processing is carried out for the point.

2.6. Comparison of Algorithms

The iterative method is simple and easy to understand, and is less affected by the target size and noise. The iterative method has a good segmentation effect on the foreground and background of the image, and can distinguish the main location of the foreground and background. However, it can't distinguish the fine points of the image sometimes, and the time complexity of the algorithm is relatively high.

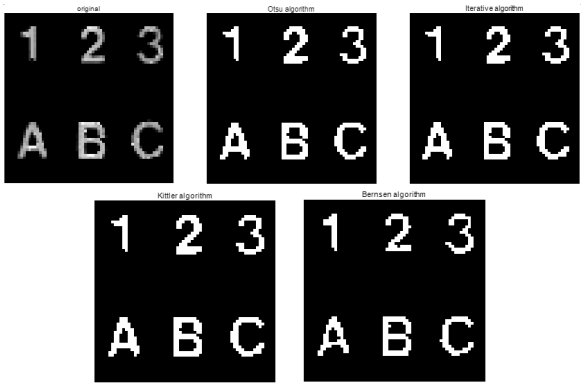


Figure 5: Result of binarization algorithms on image 1.

The inter-class variance of Otsu algorithm reaches the maximum value, which means the probability of misclassification of foreground and background pixels of the image is the lowest. The threshold value obtained by Otsu is relatively ideal. For images with different brightness conditions, as long as the illumination of the layout is uniform, the processing effect is relatively good, and the algorithm is relatively stable with low time complexity. However, the segmentation effect of Otsu on document images with large noise and images close to background is sometimes poor, and the Otsu method may be invalid when the image is unevenly illuminated.

Kittler algorithm has a good segmentation effect on the image with a roughly normal ratio of foreground and background size, and its operation is simple and fast. The Kittler algorithm mainly uses the grayscale and gradient information of the image, so it is susceptible to noise. In application, not only the size of the target should be estimated, but also the SNR should be considered.

Bernsen algorithm can process the image with uneven illumination, and it can obtain the threshold value of each pixel by the maximum and minimum values of pixels in the local window, which is relatively fast. However, the operation time is related to the window size, and the larger the local window is, the more time is spent. If the local window selection is too small, the accuracy of the calculation results will be affected, and then the effect of image binarization will be affected.

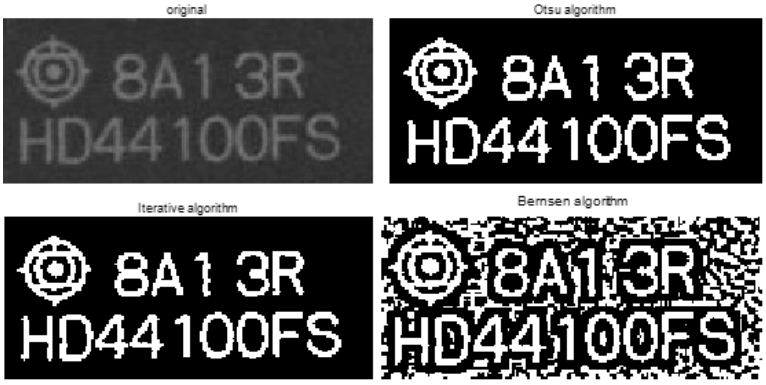


Figure 6: Result of binarization algorithms on image 2.

2.7. Algorithm Implementation

OTSU method (OTSU) is an algorithm to determine image binarization segmentation threshold, From the principle of Otsu method, this method is also called the maximum interclass variance method, because after image binarization segmentation according to the threshold obtained by Otsu method, the interclass variance of foreground and background images is the largest.

It is considered as the best algorithm for threshold selection in image segmentation. It is easy to calculate and is not affected by image brightness and contrast. Therefore, it has been widely used in digital image processing. It divides the image into background and foreground according to its gray scale. As variance is a measure of gray distribution uniformity, the greater the inter-class variance between background and foreground, the greater the difference between the two parts of the image. If part of the foreground is misclassified into background or part of the background is misclassified into foreground, the difference between the two parts will decrease. Therefore, the segmentation that maximizes the variance between classes means that the probability of misclassification is minimized.

Application: It is the best method to obtain the global threshold of image. It is suitable for most situations where the global threshold of image is required. Advantages: simple and fast calculation, not affected by the image brightness and contrast.

Disadvantages: Sensitive to image noise; Can only be segmented for a single target; When the ratio of target and background size is large and the inter-class variance function may show double or multiple peaks, the effect is not good.

Algorithm 1: Otsu Algorithm

Input variable: A grey image

Output Variable: A binary image and its threshold

The first scan:
repeat for each pixel
  calculate the number of each gray value
until *all pixels are visited*;
The second scan:
repeat for each pixel
  calculate the proportion of each gray value in the total matrix and the grey mean of the image population
until *all pixels are visited*;
Initialize the threshold Th is 0 and the optimal threshold Th_best is 0
The third scan:
repeat for each pixel
  if *the grey value is less than Th* **then**
    calculate the proportion and grey mean less than Th
  end
  if *the grey value is greater than Th* **then**
    calculate the proportion and grey mean greater than Th
  end
  calculate the inter-class variance and take its maximum value as the optimal threshold.
until *all pixels are visited*;

The hypothesis of OTSU algorithm is that there is a threshold Th to divide all pixels of the image

into two categories C_1 (less than Th) and C_2 (greater than Th), then the respective mean values of these two categories of pixels are u_0 and u_1, and the global mean value of the image is u. At the same time, the probability that pixels are divided into C_1 and C_2 types is w_0 and w_1 respectively. Hence,

$$u = w_0 u_0 + w_1 u_1$$
$$w_0 + w_1 = 1$$

According to the concept of variance, the expression of inter-class variance is:

$$\sigma = w_0 \cdot (u_0 - u)^2 + w_1 \cdot (u_1 - u)^2$$

By simplifying the above equation and substituting equation, we can get:

$$\sigma = w_0 \cdot w_1 \cdot (u_0 - u_1)^2$$

During the program operation, the value of Th can be traversed successively from the minimum gray value of the image to the maximum gray value of the image. When T takes a certain value, the inter-class variance formula $\sigma = w_0 \cdot (u_0 - u)^2 + w_1 \cdot (u_1 - u)^2$ can get the maximum, and then Th is the optimal threshold of binarization.

$$Th_{best} = max(\sigma)$$

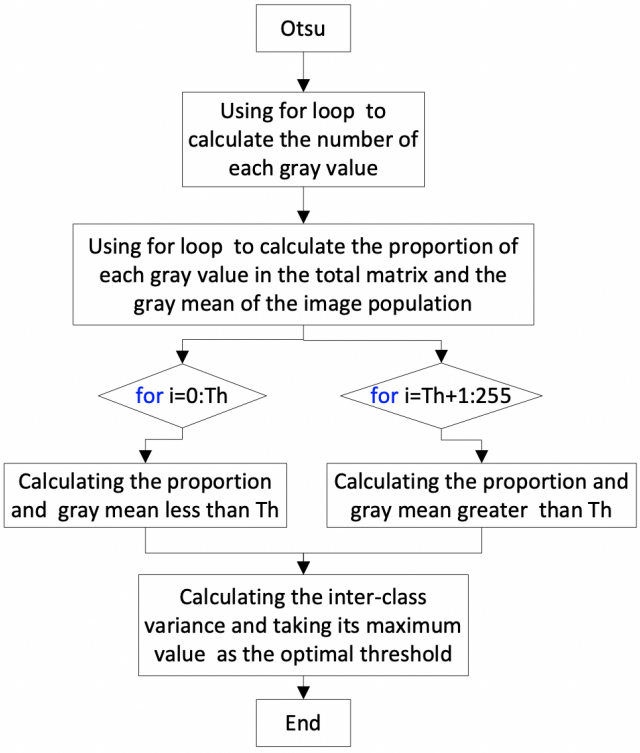


Figure 7: Flow chart of Ostu algorithm.

3. Traversal Connectivity Labeling

For image segmentation, we selected four commonly used methods for comparison:1.Two-pass scanning method 2.Region growing algorithm 3.Quadtree Split 4.BFS Connectivity Check

3.1. Two-Pass Scanning Method

The two-pass scanning method means that by scanning the image twice, all connected areas in the image can be found and marked.

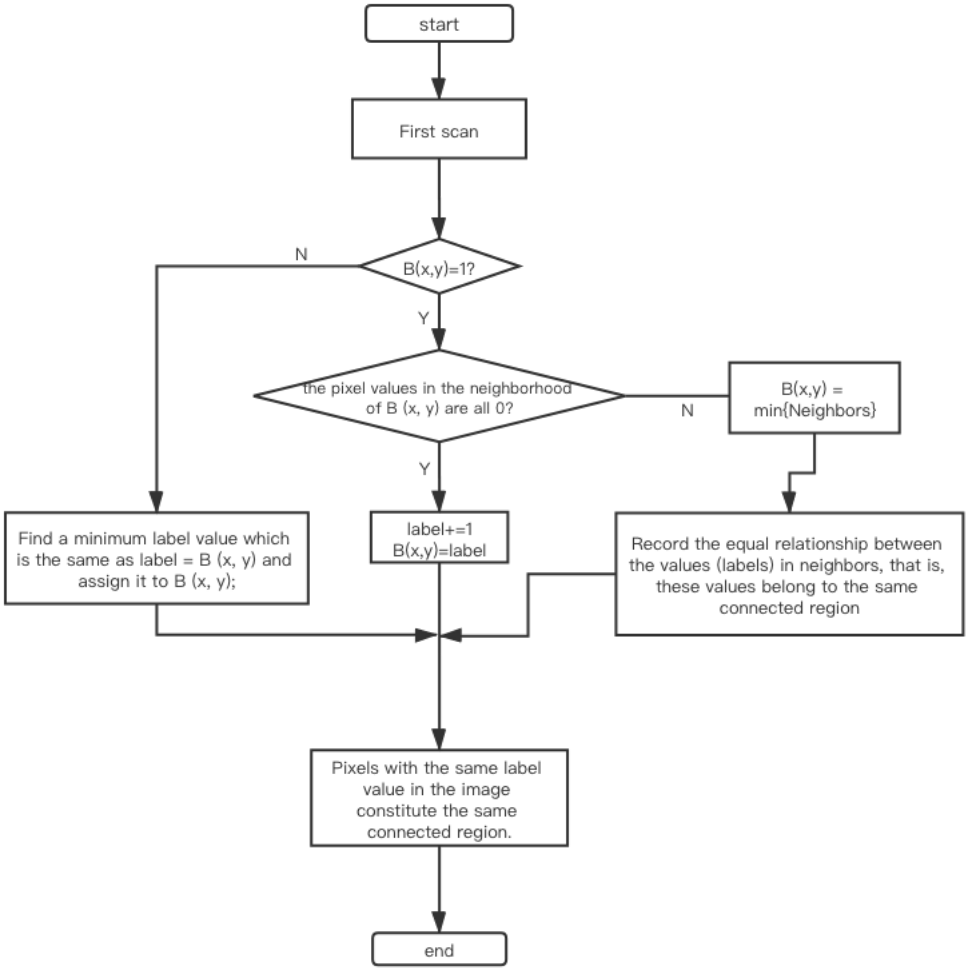


Figure 8: Two-pass scan flow chart.

Idea: In the first scan, a label is assigned to each pixel position. During the scanning process, the pixel sets in the same connected area may be assigned one or more different labels, so these need to belong to the same connected area but have different labels. The label of the value is merged, which is to record the connected relationship between them; the second scan is to classify the labeled pixels with the connected relationship into a connected area and assign the same label.

3.2. Region Growing Algorithm

The design of the region growth algorithm mainly consists of the following three points:

- The determination of the growth seed point.
- The condition of the region growth
- The condition of the region growth stop

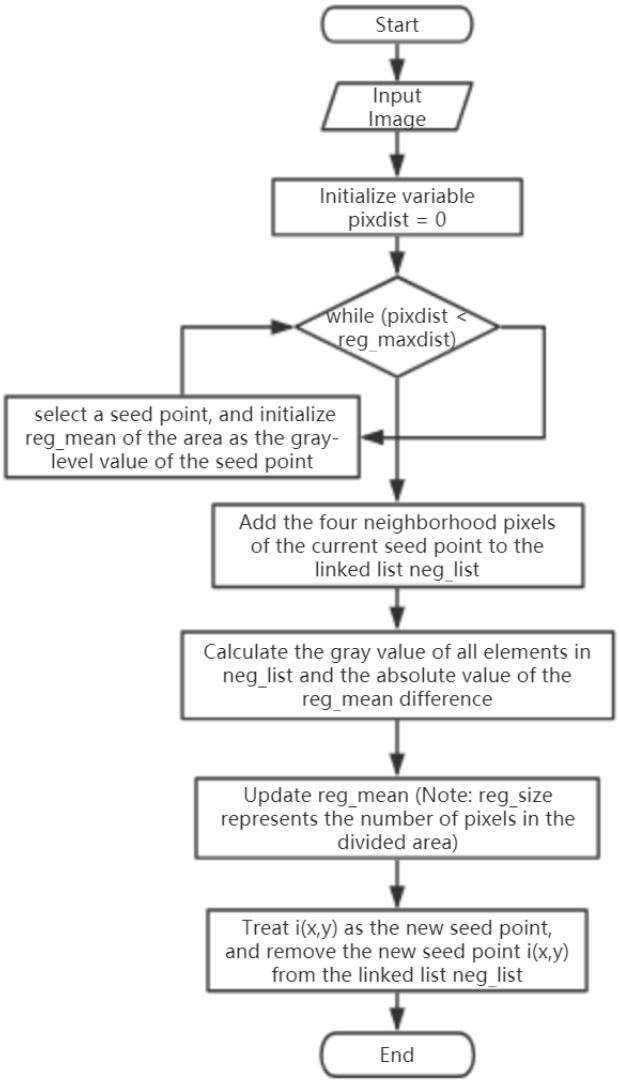


Figure 9: Region growing flow chart.

The number of seed points can be selected from one or more according to specific problems, and can be determined automatically or by human-computer interaction according to specific problems. The condition of region growth is actually some similarity criteria defined according to the continuity of

pixel gray levels, and the condition of stopping region growth defines a termination rule. Basically, when no pixel meets the conditions for joining a certain region , The area growth will stop. In the algorithm, define a variable, the maximum pixel gray value distance $reg_{maxdist}$. When the absolute value of the difference between the gray value of the pixel to be added and the average gray value of all pixels in the segmented area is less than or equal to $reg_{maxdist}$. The pixel is added to the already divided area. On the contrary, the area growth algorithm stops.

3.3. Quadtree Split

The basic idea of quadtree encoding is: First, an image or raster map (k>1, fill the network if insufficient) is equally divided into four first-level blocks, the order is upper left, upper right, lower left, and lower right; Then check all the grid attribute values (or gray values) block by block. If they are the same, the block will not be divided; if they are different, the subblock will be further divided into four secondary subblocks; such recursive division, Until the attributes or gray levels of each sub-block are equal.

3.4. BFS Connectivity Check

Breadth-first search uses breadth as the first keyword. When a fork is encountered, all nodes that can be reached directly by the fork are first visited, and then in the order in which these nodes are visited, continue to visit all the nodes that can be reached. Until all nodes have been visited or the termination conditions are met; The ideal way to implement the algorithm is to use a queue to continuously pop up the previous elements. During the search process, the next-level elements are continuously added to the back until the queue is empty or the conditions are met to complete the search.

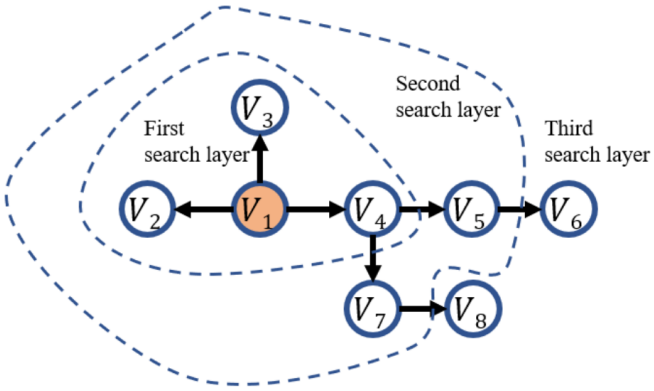


Figure 10: BFS connectivity check flow chart.

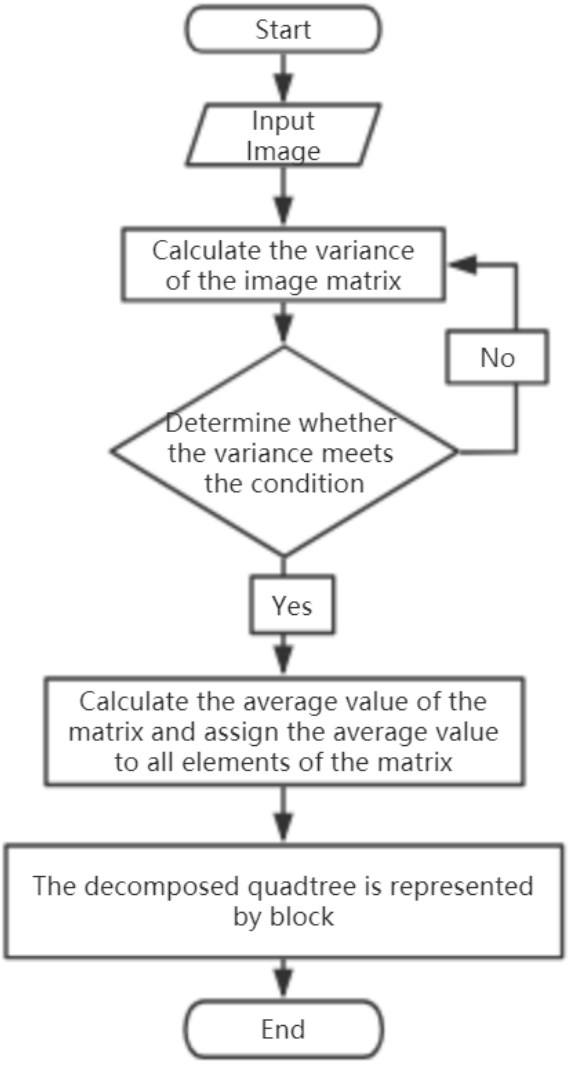


Figure 11: Quadtree split flow chart.

3.5. Result Compare

After using two-pass and BFS connectivity method, we use different labels to represent different features in the picture and display them in different colors. The first two methods are applied to the segmentation of binary images, and the latter two methods are applied to the segmentation of gray images. By using four algorithms to segment image 1 features separately, we can get the following results:

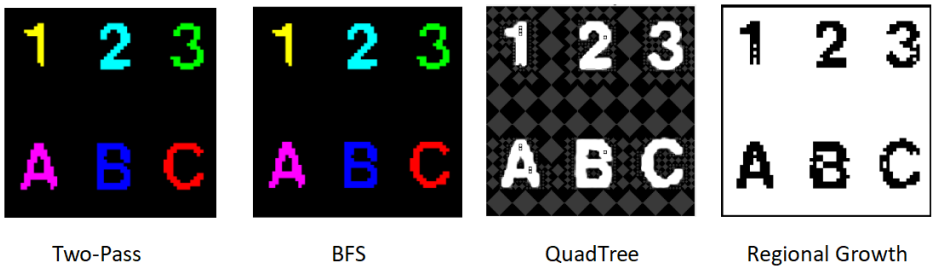


Figure 12: Segement result compare.

Algorithm 2: Two-pass scan algorithm

Input variable: A binary image

Output Variable: label-number and the image with label

The first scan:

repeat for each foreground pixel
  if *the left and upper neighbor pixels are invalid* **then**
    set a new label value v, label ++
  end
  if *one of the left or upper neighbor pixel is valid* **then**
    assign label value v to the pixel
  end
until *all pixels are visited*;

The second scan:

repeat for each foreground pixel
  if *the left and upper adjacent pixels are both valid* **then**
    assign smaller label value v_{min} to the pixel
  end
until *all pixels are visited*;

Through comparison, we can find that the effect of using two-pass scanning method and BFS connectivity method is similar. The region growth and QuadTree segmentation methods will show different effects due to different thresholds. Two-pass local table method is much faster and uses small memory space. The larger the image, the better the performance is when compared with the classical algorithm. Finally, we choose the Two-pass algorithm as the method to segment the image. When processing the second picture with the Two-pass algorithm, we found that the circle of the first feature is separated from the outside, so it is divided into another part, so we added a subordinate detection algorithm to make these two parts. The label is synthesized into the same:

- When the left and upper neighbor pixels of the pixel are invalid values, set a new label value for the pixel, label ++;
- When one of the left or upper neighbor pixels of the pixel is a valid value, the label of the valid value pixel is assigned to the label value of the pixel;
- When the left and upper adjacent pixels of the pixel are both valid values, the smaller label value is selected and assigned to the label value of the pixel.

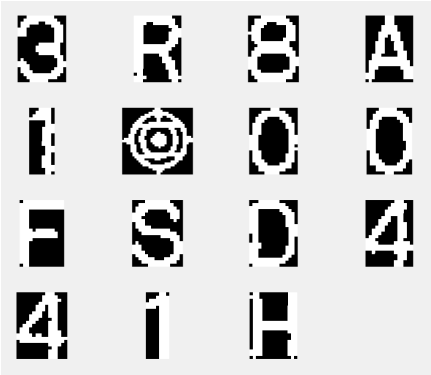


Figure 13: Task 2 segement.

4. Character Rotation

The calculation methods of rotation transformation can be divided into forward mapping and reverse mapping. The forward map-based approach scans each input pixel sequentially, then copies the values directly to the output image at the location determined by $T(w, z)$.The disadvantage of forward mapping is that multiple pixel points may be transformed into the same pixel point on the output image. Or maybe some pixels are not assigned at all. The reverse mapping process scans each output pixel in order, calculates the corresponding position on the input image with $T^{-1}(w, z)$, and interpolates it into the nearest input pixel to determine the value of the output pixel. Reverse mapping is easier to implement than forward mapping. So we use reverse mapping in our algorithm.

4.1. Centroid

The coordinates of centroid of a section of image can be calculated with formulas below:

$$\bar{x} = \sum_{i=1,j=1}^{i=n,j=n} x(i,j)$$
$$\bar{y} = \sum_{i=1,j=1}^{i=n,j=n} y(i,j)$$

4.2. Rotation Transform

The first step is to transform the coordinate of every pixel from original coordinate to rotation center coordinate. The equation is written as:

$$\begin{bmatrix} x_2 & y_2 & 1 \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ -0.5w & 0.5h & 1 \end{bmatrix}$$

where w and h are respectively the width and height of the image. And then we can perform the

central rotation according their own respective centroids.

$$\begin{bmatrix} \overline{x_2} & \overline{y_2} & 1 \end{bmatrix} = \begin{bmatrix} x_2 & y_2 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

From above equation we can derive the rotated coordination of P denoted as $(\overline{x_2}, \overline{y_2})$. Finally, we can do an inverse coordinate transform

$$\begin{bmatrix} \overline{x_1} & \overline{y_1} & 1 \end{bmatrix} = \begin{bmatrix} \overline{x_2} & \overline{y_2} & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0.5w & 0.5h & 1 \end{bmatrix}$$

to obtain the rotated coordination according to original coordinate $(\overline{x_1}, \overline{y_1})$ Therefore, the overall rotation transform is given as

$$\begin{bmatrix} \overline{x_1} & \overline{y_1} & 1 \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ -0.5w & 0.5h & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0.5w & 0.5h & 1 \end{bmatrix}$$

4.3. Nearest Neighbor Interpolation

The idea of nearest interpolation is simple. For a floating-point coordinate obtained by reverse transformation, simply round it to obtain an integer coordinate. The pixel value corresponding to this integer coordinate is the pixel value of the target pixel, that is, the floating-point coordinate is taken The pixel value corresponding to the nearest upper left corner point (upper right corner for DIB, because its scan line is stored in reverse order). It can be seen that the nearest neighbor interpolation is simple and intuitive, but the image quality obtained is not high. The image magnified by the nearest interpolation will have Mosaic phenomenon, while the image shrunk will also lose larger image information.

Algorithm:

$$\begin{aligned} f(x,y) &= g(x,y) \cdot h(x,y) \\ &= \sum_{l=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} g(l,k)h(x-l, y-k) \end{aligned}$$

First:

$$x' = [x];\ y' = [y]$$

Then, determine the distance between $f(x,y)$ and $g(x',y'), g(x'+1,y'), g(x',y'+1), g(x'+1,y'+1)$. Finally, assign $f(x,y)$ equals to the pixel with the smallest distance from $f(x,y)$.

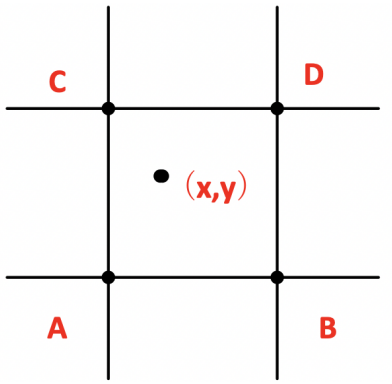


Figure 14: Nearest neighbour interpolation.

4.4. Bilinear Interpolation

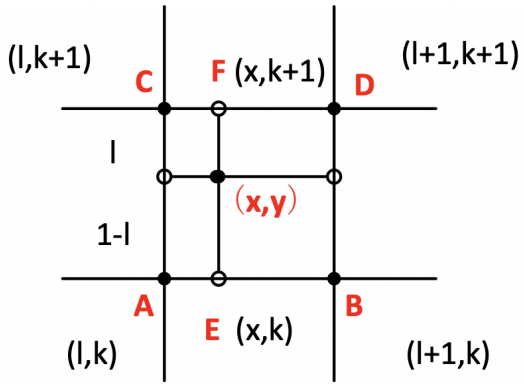


Figure 15: Bilinear interpolation.

In mathematics, bilinear interpolation is a linear extension of an interpolation function with two variables, and its core idea is to carry out linear interpolation in two directions respectively. Bilinear interpolation is the use of two linear interpolation to obtain the value of an unknown point. The bilinear interpolation method has a large amount of computation, but the image quality after scaling is high, so the pixel value will not be discontinuous. Because bilinear interpolation has the property of low pass filter, the high frequency component is damaged, so the image contour may be blurred to some extent.

(a) Seperate - 90 degree. (b) Combine.

(c) Seperate - 35 degree. (d) Combine.

Figure 16: Result of image rotation.

5. Edge Detection

Edge detection is to find the boundary between the object and the background. So far, the most common method of edge detection is to detect the discontinuity of gray values. This discontinuity is detected using first and second derivatives.

The first derivative gradient selected in image processing:

$$\nabla \mathbf{f} = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

The magnitude of this vector is:

$$\nabla f = mag(\nabla \mathbf{f}) = \left[g_x^2 + g_y^2\right]^{1/2} = \left[\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2\right]^{1/2}$$

For simplifying calculation, sometimes such value can be calculated by ignoring the square root calculation:

$$\nabla f \approx g_x^2 + g_y^2$$

Or by taking the absolute value to approximate:

$$\nabla f \approx |g_x| + |g_y|$$

One basic property of the gradient vector is that it points towards the maximum chaging rate of f

at coordinate (x, y). Such maximum changing happens at angle:

$$\alpha(x, y) = \arctan(g_y/g_x)$$

The second derivative in image processing is usually calculated using the Laplacian operator. The Laplacian of a two-dimensional function $f(x, y)$ consists of a second-order differential:

$$\nabla^2 f(x, y) = \frac{\partial^2 f(x, y)}{\partial x^2} + \frac{\partial^2 f(x, y)}{\partial y^2}$$

The Laplacian operator is rarely used directly in edge detection, because as a second derivative, it is extremely sensitive to noise. Its amplitude will produce double edges, and the direction of the edges cannot be detected. However, when used in combination with other edge detection techniques, the Laplacian is a powerful complementary method.

Based on previous discussion,the basic idea of edge detection is to use one of the following two principles to find the fast-changing position in an image:

- Find the position where the magnitude of the first derivative of the gray level is greater than a specified threshold;
- Find the position where the second derivative of gray level has zero crossing.

5.1. Sobel Detector

The first derivative can be approximated numerically as a difference. Sobel edge detector uses the discrete difference of rows and columns of a 3×3 area to calculate the gradient. The center pixel of every row or column is averaged by 2 to provide a kind of smoothing effect.

$$\Delta f = \left[g_x^2 + g_y^2\right]^{1/2} = \left\{[(z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)]^2 + [(z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)]^2\right\}^{1/2},$$

where z means the gray level of such pixel. Therefore, if at (x, y), $\nabla f \geq T$ (T is the threshold valus), then this pixel is an edge pixel.

5.2. Laplacian of a Gaussian

Consider a Gaussian function:

$$G(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

where σ is the standard deviation. The Laplacian of this function is:

$$\nabla^2 G(x, y) = \frac{\partial^2 G(x, x)}{\partial x^2} + \frac{\partial^2 G(x, x)}{\partial y^2} = \left[\frac{x^2 + y^2 - 2\sigma^2}{\sigma^4}\right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$

We convolve the image with $\nabla^2 G(x, y)$ knowing that it has two effects: It smooths the image, thus reducing noise; and it computes the Laplacian, which yeilds a double-edge image. Locating edges then consists of finding the zero crossings between the double edges.

5.3. Canny Detector

1. The image is smoothed using a Gaussian filter with a specified standard deviation, σ, to reduce noise.

2. The local gradient, $[g_x^2 + g_y^2]^{1/2}$, and edge direction, $\tan^{-1}(g_x/g_y)$, are computed at each point. An edge point is defined to be a point whose strength is locally maximum in the direction of the gradient.

3. The edge points determined in 2. give rise to ridges in the gradient magnitude image. The algorithm then tracks along the top of these ridges and sets to zero all pixels that are not actually on the ridge top so as to give a thin line in the output, a process known as nonmaximal suppression.The ridge pixels are then thresholded by so-called hysteresis thresholding,which is based on using two thresholds, T_1 and T_2,with $T_1 < T_2$. Ridge pixels with values greater than T_2 are said to be 'strong' edge pixels. Ridge pixels with values between T_1 and T_2 are said to be 'weak' edge pixels.

4. Finally, the algorithm performs edge linking by incorporating the weak pixels that are 8-connected to the strong pixels.

5.4. Algorithm Implementation

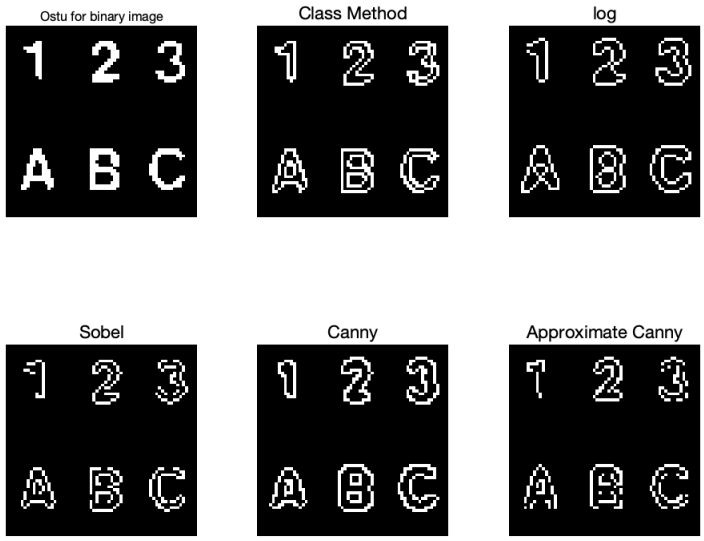


Figure 17: Result of edging algorithms on image 1.

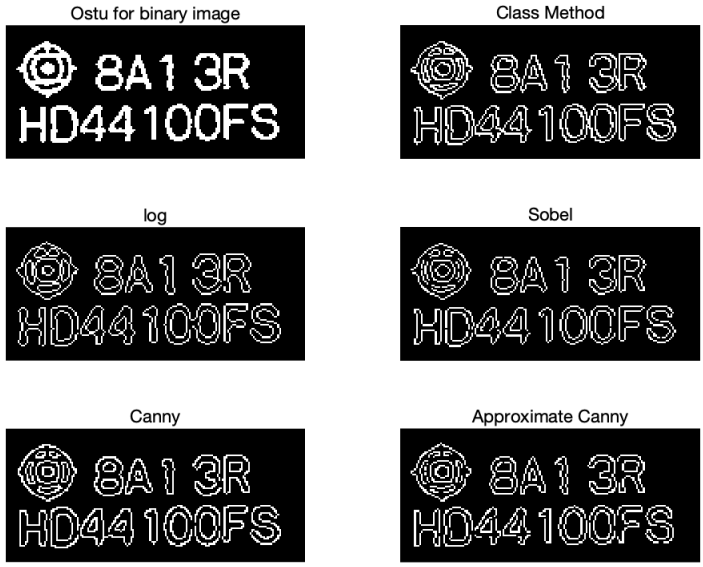


Figure 18: Result of Edging Algorithms on image 2.

Based on the analysis of characteristic, we finally choose the class method to identify the edge of each characteristic. The detail is shown below.

The principle of edge detection is very simply: If the gray-level of an object pixel is different from one or more of its neighbour pixels, this pixel is an edge pixel. To be more specific, if an object pixel has one or more pixels whose gray-level is zero, then we define this pixel as an edge pixel. Inversely, instead of detecting the regions, we can delete all the inner pixels.

Algorithm 3: Edge detection algorithm

Input variable: A binary image

Output Variable: An image showing the edge of each character

repeat for each bright region
  if $g(x-1,y-1)\cdot g(x-1,y+1)\cdot g(x+1,y-1)\cdot g(x+1,y+1)=1$ **then**
    Let $g(x,y)=0$
  end
until *all characters are visited*;

6. Image Thinning

An important approach for representing the structural shape of a planar region is to reduce it to a graph. This reduction may be accomplished by obtaining the skeleton of the region via a thinning (also called skeletonizing) algorithm. The skeleton of a region may be defined via the medial axis transformation(MAT).The MAT of a region R with border b is as follows. For each point p in R, we

find its closest neighbor in b. If p has more than one such neighbor, it is said to belong to the medial axis (skeleton) of R. Although the MAT of a region is an intuitive concept,direct implementation of this definition is expensive computationally, as it involves calculating the distance from every interior point to every point on the boundary of a region. Numerous algorithms have been propsed for improving computational efficiency while at the same time attempting to approximate the medial axis representation of a region.

Any refinement algorithm generally must meet the following 4 requirements:

- The skeleton image must maintain the connectivity of the original image target edge;
- The skeleton image should be as close as possible to the center line of the target edge of the original image;
- Thinning results should strive to obtain a line image of one pixel width as much as possible;
- Thinning speed must be fast.

6.1. Helditch Algorithm

The Helditch thinning algorithm is a serial processing method, and the final result is 8 neighboring connecting lines (that is, each pixel of the thinned thin line is considered to be connected to the 8 neighboring thin line pixels around it.

The main step of the algorithm is to traverse the image pixels. In a traversal, for each pixel, if the following 6 conditions are met at the same time, it will be marked as to be removed. After a traversal, if there are pixels to be removed, all the pixels to be removed will be traversed this time. In addition to the pixel value change, and then restart the traversal until there is no pixel to be removed at the end of a certain traversal, indicating that the refinement is completed, and the algorithm ends.

- The current pixel value is 1, which means it is not a background.
- At least one of P's axial adjacent neighbors (x1, x3, x5, x7) has a value of 0 (if all are 1, obviously P cannot be removed).
- At least two of P's 8 neighbors have a value of 0 (if only one has a value of 0, it is a thin line endpoint, and if there is no value of 0, it is an isolated point, and none of it can be removed).
- At least one white neighbor (value 1) of P's 8 neighbors has not been marked for removal (this condition has basically no effect on the result, so many people directly removed it when introducing it, and keep it consistent with the paper).
- Calculate the crossing number of P: Nc(p), Nc(p)=1.

$$N_c^8(p) = \sum_{i=1}^{4} (\bar{x}_{2i-1} - \bar{x}_{2i-1}\bar{x}_{2i}\bar{x}_{2i+1})$$

- If x3 and x5 have been marked to be removed, use the values of x3=0 and x5=0 to recalculate the number of connections and be equal to 1. This condition is because when traversing to point P, points x2, x3, x4, and x5 have been judged to end (starting from the upper left corner and traversing from top to bottom from left to right), but their values have not changed at this time It is 0 (the pixel value will be changed uniformly after the end of one traversal), so we

need to use their new value to calculate the number of connections. Based on the 8-connectivity scheme we used, deleting the diagonal neighbors (x2, x4) will not change P The number of points connected, so just consider the axial x3, x5.

6.2. Deutch Algorithm

The processing method adopted by Deutch refinement algorithm is parallel processing, and its algorithm also uses two layers of sub-loops. In the initial sub-loop, the conditions for deleting the target pixel P are:

- $\sum P_k = 0$ $(0 \leq k \leq 7)$
- N_c=1
- $P_0 + P_2 + P_4 + P_6 \leq 3$. If the conditions are met, the point P is marked as a deleteable pixel. In the next sub-cycle, the entire image is scanned and all such pixels are deleted. The skeleton obtained by this algorithm is an incomplete 8-connection, which can be regarded as an 8-connection graph with deleteable points.

6.3. Pavlidis Algorithm

Pavlidis refinement algorithm is also a kind of parallel algorithm. It strips off boundary pixels layer by layer by matching two templates, and the resulting skeleton shape is 8-connected.

Pavlidis asynchronous thinning algorithm is achieved by mixing parallel and serial algorithms. Using bit operations to match specific patterns, the resulting skeleton shape is 8-connected.

6.4. Zhang Fast Parallel Calculation

For Zhang fast parallel calculation, when judging whether a point should be deleted or not, it should be judged based on its eight adjacent points. To sum up, there are the following criteria:

- Internal points cannot be deleted;
- Outliers cannot be deleted;
- The end of a straight line cannot be deleted;
- If P is a boundary point, after removing P, if the connected component does not increase, then P can be deleted.

6.5. Comparison of Thining Algorithms

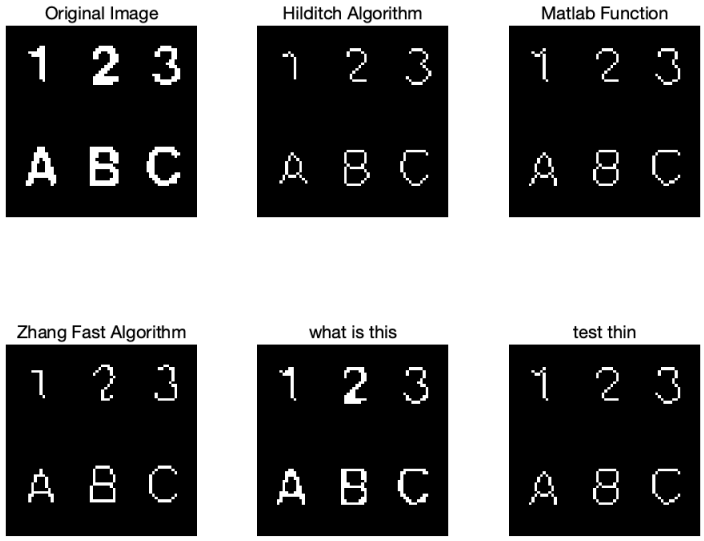


Figure 19: Result of thinning algorithms on image 1.

Figure 20: Result of thinning algorithms on image 2.

6.6. Implementation of Hilditch Algorithm

The core idea of Hilditch algorithm is to traverse the image pixels in a loop and determine whether the pixels are deleted according to the rules, so as to achieve the purpose of thinning the image. Consider the following 8-neighbourhood of a pixel P_1:

P9	P2	P3
P8	P1	P4
P7	P6	P5

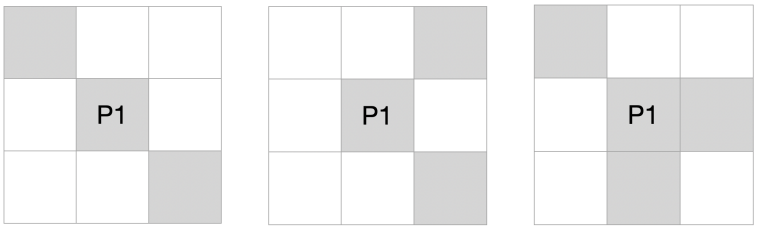
Figure 21: 8-neighbourhood of a pixel P_1.

We want to decide whether to peel off p_1 or keep it as part of the resulting skeleton. For this purpose we arrange the 8 neighbours of p_1 in a clock-wise order and we define the two functions:

$$B(P_1) = \text{number of non-zero neighbour of } P_1$$
$$A(P_1) = \text{number of 0,1 patterns in the sequence } P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9, P_2$$

If we summarize the conditions of Hilditch's algorithm, it should be the folloeing four conditions:

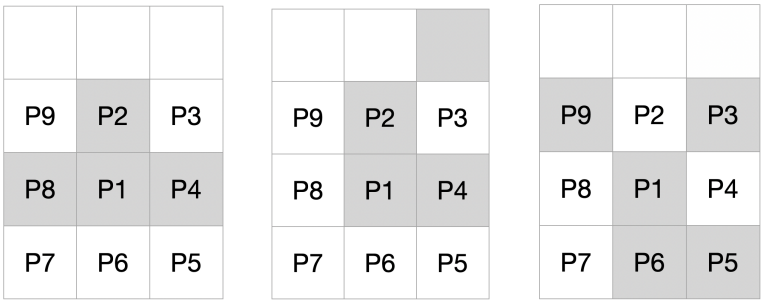
- $2 \leq B(P_1) \leq 6$
  This condition combines two sub-conditions, first that the number of non-zero neighbors of P_1 is greater than or equal to 2 and second that it be less than or equal to 6. The first condition ensures that no end-point pixeland no isolated one be deleted (any pixel with 1 black neighbor is an end-point pixel), the second condition ensures that the pixel is a boundary pixel.
- $A(P_1) = 1$
  This is a connectivity test. In fact, if you consider the below pictures where $A(P_1) > 1$, you can see that by changing P_1 to 0 the pattern will become disconnected.



(a) $A(P_1) = 2$ (b) $A(P_1) = 2$ (c) $A(P_1) = 3$

Figure 22: Conditions of $A(P_1) > 1$

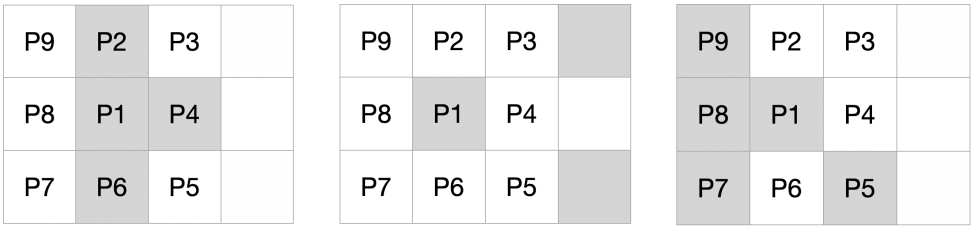
- $P_2 \cdot P_4 \cdot P_8 = 0$ or $A(P_2) \neq 1$
  This condition ensures that 2-pixel wide vertical lines do not get completely eroded by the algorithm.



(a) Unsatisfied (b) $A(P_2) \neq 1$ (c) $P_2 \cdot P_4 \cdot P_8 = 0$

Figure 23: Some examples

- $P_2 \cdot P_4 \cdot P_6 = 0$ or $A(P_4) \neq 1$
  This condition ensures that 2-pixel wide horizontal lines do not get completely eroded by the algorithm.



(a) Unsatisfied (b) $A(P_4) \neq 1$ (c) $P_2 \cdot P_4 \cdot P_6 = 0$

Figure 24: Some examples

- Stop when no more pixels can be removed.

Hilditch's algorithm is a parallel-sequential algorithm. It is parallel because at one pass all pixels are checked at the same time and decisions are made whether to remove each of the checked pixels. It is sequential because this step just mentioned is repeated several times (until no more changes are done).

However, Hilditch's algorithm turned out to be not the perfect algorithm for skeletonization because it does not work on all patterns. In fact, there are patterns that are completely erased by the algorithm.

7. Arrange the characters

In order to rearrange the sequence of all the separate character, we use the algorithm as follows. Firstly, we calculate the width of each character and figure out the maximum of the width. Then we fill the matrix of other character with zero rows to satisfy the requirement of all the matrix being the same dimension. Finally, we rearrange the characters as the sequence of A1B2C3 and 81344100ARHDFS with a zero column added between two characters.

Figure 25: Arrangement result.

8. App Design

8.1. User Interface

In order to facilitate function realization and method comparison, we carried out APP design of all programs. The APP can complete all image processing tasks, from loading images to re-arrangement. It can run three binarization methods and four image segmentation methods, but the second picture only applies to the first two image segmentation methods.

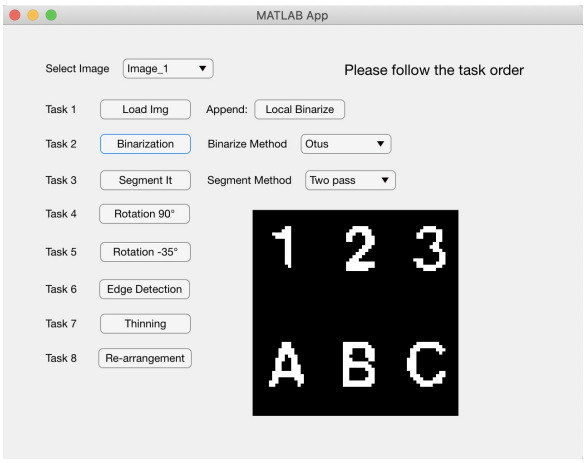


Figure 26: GUI image 1 task 2.

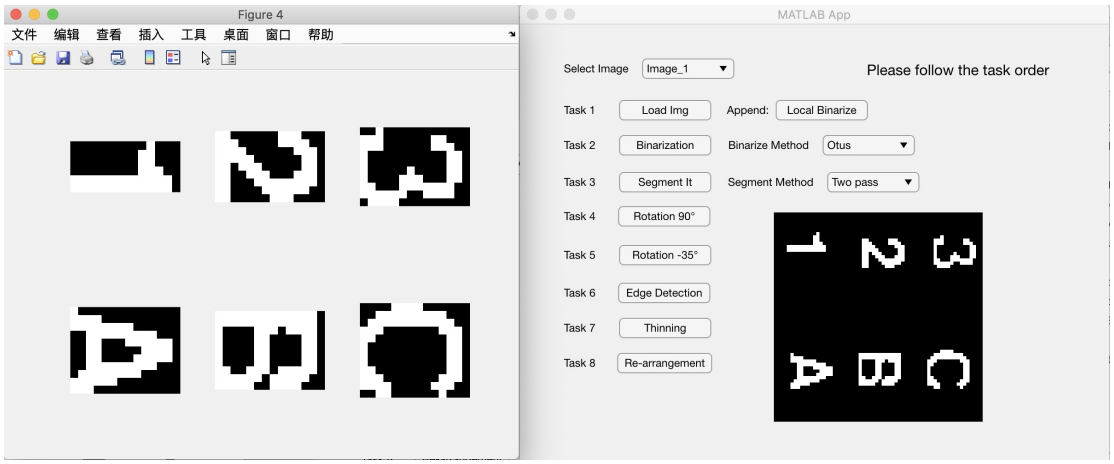


Figure 27: GUI image 1 task 4.

Figure 28: GUI image 2 task 2.

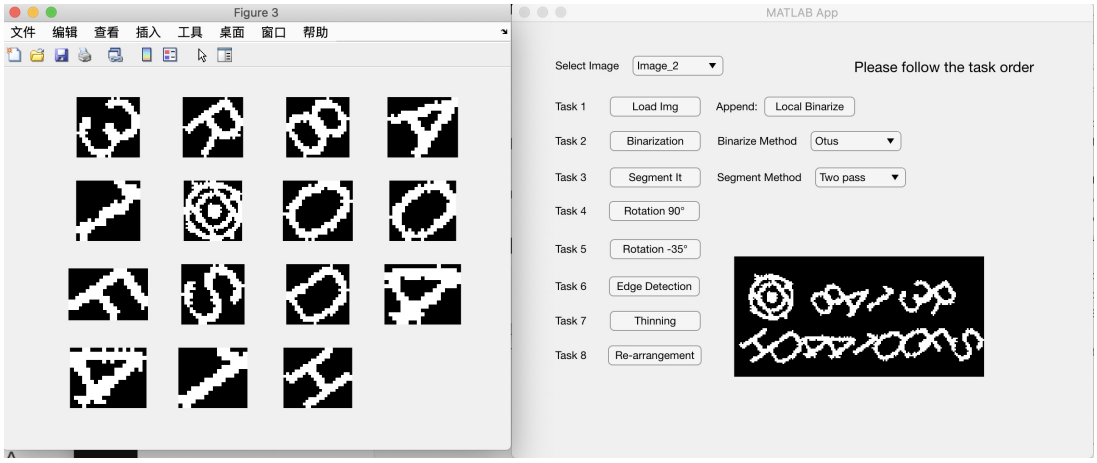


Figure 29: GUI image 2 task 5.

The use of this APP should run the buttons in the order of operation, skipping may cause operating errors. The left side indicates which task each function solves, and each corresponding button will call back the function to solve the task. We have defined global variables for the APP so that each sub-function can be called reasonably, and added a picture display inside the APP to compare pictures with other methods.

9. Conclusion and Potential Improvements

For image 1, the input picture is composed of numbers and letters, and each letter and number corresponds to a gray value. So we thought of using ASCII code table to map the relationship between character and gray value, and display the new picture in the gray range of [0,31]. Being different from image 1, image 2 has a lot of noise and is not clear ,so we preprocess the picture before binarizing. we apply sharpening and gaussian filter procedure to increase the contract between useful characters and noise, and then we use thresholding to derive binary image. After segment operation, both two images can be used in the same rotation,outline and thin algorithms.

For now, the requirement of this project has already been satisfied, but some potential improvements can still be done in the future.

- **Algorithm Efficiency:** For now our algorithms are still in individual .m files as functions. This is to make our main code simple and readable. However, this is not the most efficient way to organise a big project where function wrapping may be done; Also, our code now cannot save and use various variables efficiently.
- **Cross Processing:** Our code now can only deal with the image based on the order required as tasks. In this situation, we cannot process the original image in the order different from task required.
- **Binary Limitation**: This project mainly deals with binary image. However, maybe in most real situations we should deal with gray-level image or even color image. This is also a potential improvement we can make.