**图形用户界面

低可信度描述已自动生成**

ME5405 Machine Vision

**Submitted by Teacher:** Prof. CHUI Chee Kong

Prof. SARTORETTI Guillaume

Zhang Tongtian A0284909M

Liu Siqi A0284955L

Li YiMing A0285133H

A report submitted

for the assignment of ME5405

in the

Department of Mechanical Engineering

National University of Singapore

Nov 18, 2023

**Abstract**

This report primarily presents the solutions and detailed analysis our team has provided for the image processing tasks given in the course, and concludes with the experiences gained and areas for improvement during the completion of the project.

The first chapter of the report introduces the background and topic of the project. The second chapter describes the principles of the algorithms used to handle each task, introduced in the order of the project tasks. We have proposed multiple solutions for some tasks where possible. For instance, for threshold division, we adopted two global threshold division methods and one local threshold division method; for image thinning, we used the Zhang-Suen algorithm and the Medial Axis Transform method. Particularly in the last two tasks of Image 2, we also boldly attempted several different unsupervised machine learning methods.

The third chapter mainly showcases the results obtained based on the algorithms used. The final chapter discusses the knowledge learned in this assignment, as well as the shortcomings in the completion of the project. The source code of the algorithms and flowcharts can be found in the appendix.

**Content**

目录

[**Abstract II**](#_Toc11976)

[**Content III**](#_Toc7575)

[**1. Introduction of algorithm 1**](#_Toc20075)

[1.1 Background 1](#_Toc22169)

[1.2 Given Tasks 1](#_Toc30398)

[**2. Introduction of algorithm 3**](#_Toc12161)

[2.1 Display the original image 3](#_Toc11200)

[2.2 Threshold the image and convert it into binary image 4](#_Toc21868)

[2.3 Determine an one-pixel thin image of the objects 6](#_Toc1441)

[2.4 Determine the outline 8](#_Toc5124)

[2.5 Label the different objects 9](#_Toc22468)

[2.6 Rotate the original image 11](#_Toc7161)

[2.7 Training Dataset 12](#_Toc19566)

[**3. Processing result 13**](#_Toc388)

[3.1 Display original image 13](#_Toc5585)

[3.2 Extract Sub-Image 14](#_Toc26984)

[3.3 Threshold and binary image 14](#_Toc31646)

[3.4 One-pixel thin image of object 15](#_Toc8894)

[3.5 Outline 16](#_Toc17924)

[3.6 Label 17](#_Toc15474)

[3.7 Rotate the Image 18](#_Toc31187)

[3.8 Training 19](#_Toc23390)

[**4. Conclusion 22**](#_Toc13987)

[**5. Reference 23**](#_Toc32344)

[**6. Appendix 23**](#_Toc20257)

1. **Introduction of algorithm**

## Background

Machine vision, a field that encompasses various approaches, methods, and algorithms, has evolved to enhance imaging-based analysis and inspection across diverse applications, particularly in the industrial sector. This technology spans both industrial and non-industrial realms, encompassing academic, educational, governmental, and military applications. Each of these applications presents unique constraints and requirements. The evolution of machine vision reflects the progress in both hardware, such as cameras and sensors, and software, including image processing algorithms and machine learning techniques. Over the years, advancements in machine vision have led to more sophisticated and efficient systems capable of complex tasks like pattern recognition, object detection, and automated decision-making. In academic settings, machine vision forms a crucial part of the curriculum, providing students with a comprehensive understanding of the underlying principles through practical projects and theoretical study. These projects enable students to apply basic image processing methods learned in lectures, deepening their grasp of the concepts and preparing them for real-world challenges in this technologically vital field.

## Given Tasks

The project was structured into two primary sections, each comprising various sub-parts along with specific images. The approach was designed to allow for the sequential construction of multiple distinct algorithms. The development of these algorithms and their details will be explained in Section 2, where the processed images will also be showcased and compared with MATLAB's built-in functions.

The tasks set for each part of the project are as follows:

**Part 1: Chromosomes**

Image 1 is a 64x64 32-level image. It is a coded array that contains an alphanumeric

character for each pixel. The range of these characters is 0-9 and A-V, which corresponds

to 32 levels of gray.



Figure 1 Image1

**Tasks:**

1. Display the original image on screen.

2. Threshold the image and convert it into binary image.

3. Determine an one-pixel thin image of the objects.

4. Determine the outline(s).

5. Label the different objects.

6. Rotate the original image by 30 degrees, 60 degrees and 90 degrees respectively.

**Part 2: Characters**

Image 2 is a JPEG color image, comprising three lines of characters.



Figure 2 Image2

**Tasks:**

1. Display the original image on screen.

2. Create an image which is a sub-image of the original image comprising the middle

line – HELLO, WORLD.

3. Create a binary image from Step 2 using thresholding.

4. Determine a one-pixel thin image of the characters.

5. Determine the outline(s) of characters of the image.

6. Segment the image to separate and label the different characters.

7. Using the training dataset to train the to recognize the different characters (“H”, “E”, “L”, “O”, “W”, “R”, “D”).

8. Experiment with pre-processing of the data. Discuss the sensitive to these changes.

1. **Introduction of algorithm**

In this section, we mainly introduced the analysis and understanding of various problems in the project is mainly introduced and provided some basic explanations for these algorithms according to the algorithms and their principles used for the target. Since tasks of Picture 1 and Picture 2 have many similar problems, some of the algorithms in this section are applicable to both pictures.

## Display the original image

In Task 1, we initially used the “fopen” function in MATLAB to open the provided text file. Subsequently, we defined end-of-line and carriage return characters to ensure accurate parsing of image data in the following steps. And then we employed the “fscanf” function to read characters from the file, ignoring end-of-line and carriage return characters, and stored the results in a 64x64 matrix.

As “fscanf” returns a column vector, we performed a transpose operation on the matrix. Afterwards, we converted the letters A-V and numbers 0-9 into corresponding 32-level grayscale values, and transform the image matrix into an 8-bit unsigned integer. In MATLAB, an 8-bit unsigned integer can represent 256 distinct values, ensuring that the image data can be correctly parsed and displayed in subsequent processing. Finally, we created a canvas, define a title, and displayed the image.

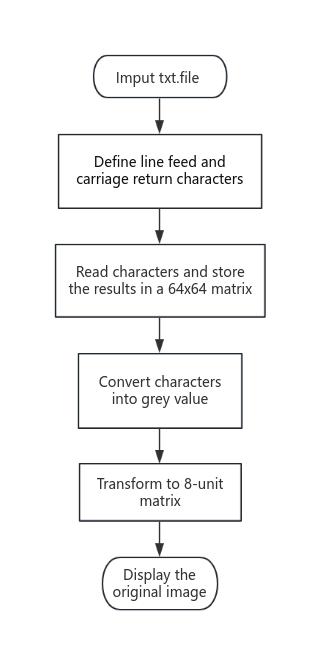


Figure 3 The Work Flowchart of Display Algorithm

## Threshold the image and convert it into binary image

To obtain a binary image from a grayscale image, the key issue is to correctly threshold the grayscale image. In this project, we have chosen three different methods. The detailed methods are as follows:

**2.2.1 Otsu algorithm**

Otsu’s method is a global thresholding technique. Therefore, before computation, it is necessary to obtain the grayscale histogram. The computed grayscale histogram is shown in the figure 4.

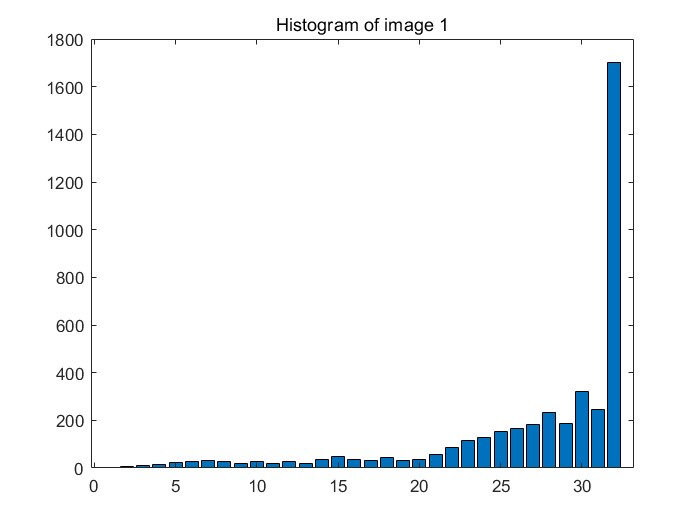


Figure 4 Histogram of Image 1

The principle of Otsu algorithm is to assume that there exists a threshold value ‘k’ that divides all pixels in the image into two categories: C1 (less than k) and C2 (greater than k). The mean values of these two categories of pixels are m1 and m2, respectively, and the global mean of the image is mG. At the same time, the probabilities of pixels being classified into categories C1 and C2 are P1 and P2, respectively. Therefore, the following formula can be obtained:

 (1)

 (2)

 (3)

 (4)

 (5)

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

 (6)

We simplify the above formula, substitute equation (6) into equation (4), and obtain:

 (7)

The ‘k’ that maximizes the value of the above formula is the threshold divided by the Otsu method. Afterwards, we use the threshold to process the original image, and we can obtain a binary image.

**2.2.2 Iterative threshold algorithm**

The detailed steps for obtaining the threshold by the iterative method are as follows:

1. Select an initial estimate T (128);
2. Use T to segment the image. This results in two sets of pixels: G1 consists of all pixels with a gray value greater than T, and G2 consists of all pixels with a gray value less than or equal to T.
3. Calculate the average gray value u1 and u2 for all pixels in G1 and G2.
4. Calculate a new threshold: T=1/2(u1 + u2).

Repeat steps (2) through (4) until the difference between the resulting T values is less than a predefined parameter T0.

**2.2.3 Adaptive threshold segmentation**

The detailed steps for adaptive threshold segmentation are as follows:

1. Divide the entire image into a series of sub-images that overlap by 50% with each other;
2. Make a histogram for each sub-image;
3. Detect whether the histogram of each sub-image is bimodal, if yes, use the graythresh function in the matlab toolbox to determine a threshold, otherwise do not process;
4. According to the threshold obtained from the sub-images with bimodal histograms, interpolate to obtain the threshold of all sub-images (pixels).

## Determine an one-pixel thin image of the objects

After the previous step, we have obtained a binary image. To obtain the one-pixel skeleton of the object, that is, the central axis of the image, a specific thinning algorithm is significant. Thinning is to peel off layer by layer, remove some pixels from the original image, but still maintain the original shape, until the skeleton of the image is obtained.

**2.3.1 Zhang-Suen algorithm**

The principle of Zhang-Suen algorithm is that whenever it runs the runs, it needs to traverse all non-zero pixels. When judging whether each pixel (P1) is deleted or retained, we need to pay attention to the values of its 8 neighboring pixels (P2, P3, P4, P5, P6, P7, P8). The order of neighboring pixels is specific, as shown in Table 1.1. The entire algorithm is divided into two steps to judge whether the point needs to be deleted.

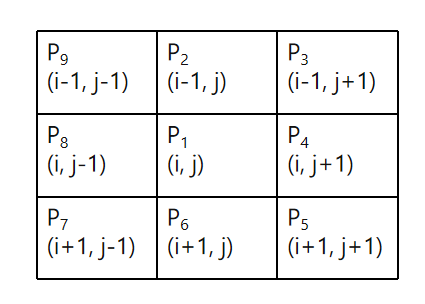


Table 1 Diagram of Zhang-Suen Algorithm [2]

In the first step, judge whether the point meets all the following conditions. If it does, delete it, otherwise keep it.

1. 2≤ B(P1) ≤ 6;
2. A(P1) = 1;
3. P2 \* P4 \* P6 = 0;
4. P4 \* P6 \* P8 = 0;

Where B refers to the sum of the number of target pixels (1 in binary) around the center pixel P1 between 2 and 6. And A refers to the number of times 0 - 1 occurs in two neighboring pixels in the 8-neighborhood pixels, in clockwise direction.

In the second step, determine whether the selected point satisfies all the following conditions, if so, delete it, otherwise keep it.

1. 2≤ B(P1) ≤ 6;
2. A(P1) = 1;
3. P2 \* P4 \* P8 = 0;
4. P2 \* P6 \* P8 = 0;

Comparing the judgment conditions of the two steps, the main differences are reflected in the third and fourth items. The third and fourth items of the first step are to remove points on the eastern and southern boundary lines, as well as the corner points on the northwest corner. The third and fourth items of the second step are to remove points on the western and northern boundary lines, and the corner points on the southeast corner.

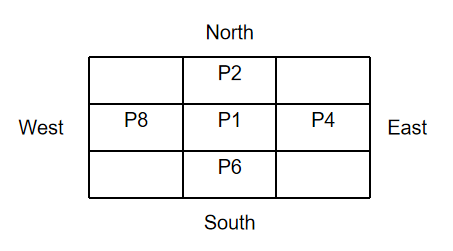


Figure 5 Points location [2]

The above two steps are iterated continuously until no new pixels are deleted at a certain stage, at which point the algorithm ends. During each iteration, the algorithm needs to maintain two storage spaces the size of the original image. One is used to save the input image, and the other saves the non-zero points that have been traversed and not deleted, that is, the new image.

**2.3.2 Medial axis transform**

We also used the “bwskel” function in the MATLAB Image Processing Toolbox for thinning of binary images. This function is based on the Medial Axis Transform method for thinning, which is similar to the Zhang-Suen algorithm. However, for 2D images, it only judges the pixel points in the four neighborhoods (up, down, left, and right) of the point to be tested, while Zhang-Suen uses an 8-neighborhood.

## Determine the outline

The objective is to extract the outline from a binary image. The algorithm behind this process is based on the comparison of adjacent pixel values to identify transitions from an object (typically represented by 1s in a binary image) to the background (represented by 0s). Here's an integrated explanation of the algorithm's principle:

Initialization: The algorithm begins by initializing “outline\_img” with the same dimensions as the binary image (binary\_img). This image will hold the outline of the objects found in the binary image.

Row-wise Operation: The code first performs a row-wise scan of the image. For each pixel, it compares the pixel value with its immediate right and left neighbors. If a pixel has a higher value than its neighbor (indicating a transition from object to background), the algorithm marks that pixel or its neighbor as part of the outline in “outline\_img”.

Column-wise Operation: Similarly, the algorithm scans the image column-wise. It compares each pixel with its top and bottom neighbors, marking the transitions in “outline\_img”. This process ensures that vertical transitions are also captured in the outline image.

Combining Row and Column Operations: By combining these two scanning operations, the algorithm effectively captures the boundaries of objects in both horizontal and vertical directions. This is crucial for accurately outlining the shape of objects in the binary image.

## Label the different objects

We chose a two-step connected component labeling algorithm for the categorization of different objects. The specific method is as follows:

1. Scan each pixel of the image in the first step. Following a top-to-bottom, left-to-right order, examine the four or eight neighboring pixels around the current pixel. If there are pixels with the same value within these neighbors, they are assigned the same label; if not, they are assigned different labels.
2. Create a lookup table for assigning final labels to pixels. In the second scan step, use the lookup table to update the component labels.

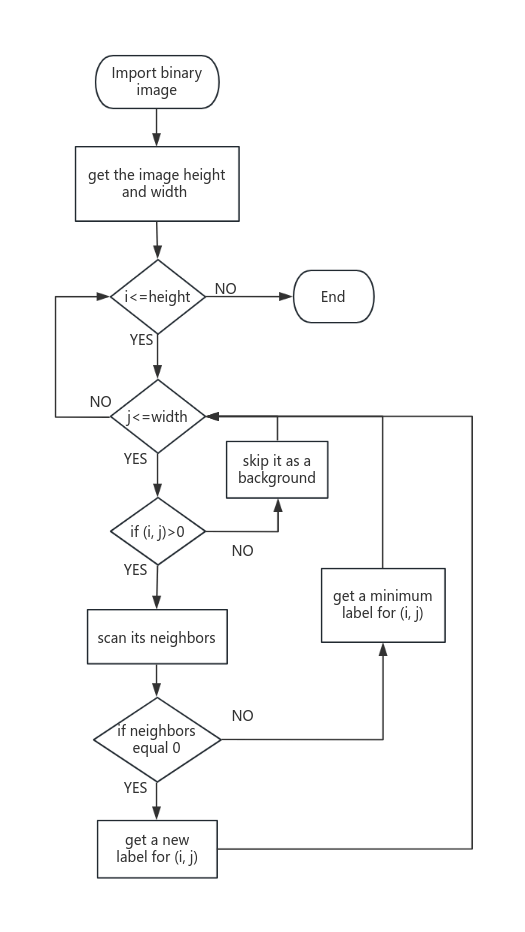


Figure 6 First scan step flow chart

As for Image2’s processing, from the previous steps, we have obtained the appropriate binary image, so we process the target binary image. The general step is to find connected components (individual characters) directly in the picture, but because the two characters "WO" in the target picture are close to each other, they are stuck together. If they are judged as one character, it will cause a large deviation in the judgment of the subsequent classifier, so we have carried out some special processing. We first performed the bwconncomp (judging individual characters) operation on the binary image, and then also searched for the centroid of each character. Since each character needs to be segmented later, we introduced ''BoundingBox'' as the segmentation box, which is a rectangular box drawn based on the position of the above-mentioned centroid. Taking into account the particularity of this example, we define it as follows: cut the BoundingBox with a width-to-height ratio greater than 1 into two parts with uniform width and assign independent counts to each. At the same time, to make subsequent results easy to find, we also numbered each individual character corresponding to the above-mentioned counting number.

Bwconncomp is a function used to find all connected regions (or connected components) in a binary image. A connected area refers to a set of pixels in an image in which adjacent pixels have the same value (usually 1, indicating white) and are connected to each other through a certain connection method. Finally it returns a structure, usually containing the following fields.

Its work is generally divided into the following steps:

1. Scan the image:

At the beginning of the function, the entire binary image is scanned, searching for points with a pixel value of 1.

1. Determine connectivity:

bwconncomp determines which pixels are connected to each other based on the specified connectivity. Connectivity can be 4-connected or 8-connected:

4-connected: A pixel has four neighbors above, below, left and right.

8-connected: A pixel has a total of eight neighbors on the top, bottom, left, right and four corners.

1. Mark components:

The function assigns a unique label to each connected region through a technique called label passing. When a new connected region is found, the function assigns a new label to all pixels in this region.

1. Record information:

For each connected region, bwconncomp logs the following information:

Pixel index list: A list of pixel indexes contained in each connected region.

Number of regions: The total number of connected regions in the image.

Region size: The number of pixels contained in each connected region.

## Rotate the original image

The rotation of an image can be seen as the rotation of all pixel points. As shown in Figure 2.x, it shows the coordinate transformation process that a certain point A goes through to rotate to point A’, where D is the distance from the rotation center.

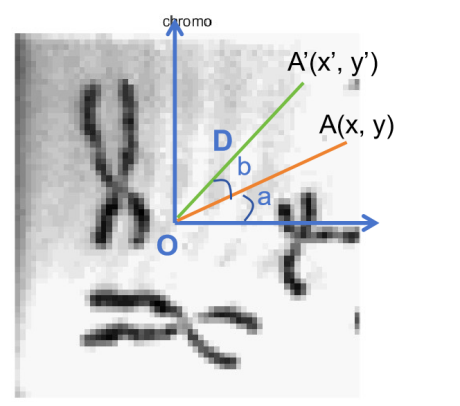


Figure 7 Schematic of Image Rotate

Assuming D is the distance from point A to the rotation center, it can be obtained through trigonometric functions:



The above formula can be transformed into matrix operations：



The general steps of the algorithm is as follows:

1. Construct a new matrix to store the image obtained by rotation.
2. Choose the rotation center.
3. Calculate the coordinates obtained by rotating each pixel point of the original image.
4. Map the pixel point values of the original image to the new image.

## Training Dataset

In the project, a range of sophisticated image processing and machine learning techniques are employed. The K-Nearest Neighbors (KNN) algorithm is utilized as a key non-parametric method for classification and regression, involving steps like selecting a distance metric, identifying nearest neighbors, and employing a voting mechanism for classification. Various feature extraction methods such as RawPixels, Histogram, Edge Detection, Gabor Features, and the Hough Transform are explored in conjunction with KNN, each offering unique advantages and posing specific challenges. Additionally, the Self-Organizing Map (SOM) Classifier, known for its pattern recognition and data visualization capabilities, is applied, emphasizing its self-organizing nature and the importance of parameters like grid size and neighborhood function. The training and testing of classifiers involve using unsupervised learning methods on a dataset, with a focus on accuracy and the classifier's ability to correctly identify individual characters. This comprehensive approach demonstrates a deep dive into machine vision techniques, showcasing their application in character recognition and classification tasks.

**2.7.1 K-Nearest Neighbors (KNN):**

KNN is a non-parametric method used for classification and regression. It involves choosing a distance metric to calculate distances between test samples and training samples, finding the nearest neighbors, and employing a voting mechanism for classification. The algorithm is straightforward and adaptable but is computationally intensive and sensitive to the curse of dimensionality.

Feature Extraction Methods with KNN:

**RawPixels**: Uses direct pixel values from images, beneficial for its simplicity and directness but leads to high dimensionality and sensitivity to image noise and lighting changes.

Histogram: Utilizes grayscale histograms to describe color distribution and intensity distribution in images, requiring normalization and useful for image comparison and classification.

**Edges**: Employs edge detection to identify object boundaries in images, highlighting significant brightness changes.

**Gabor Features**: Involves Gabor filters sensitive to spatial frequencies and orientations, offering directivity and frequency selectivity but computationally intensive.

**Hough Transform**: Used for detecting geometric shapes in images, efficient but computationally expensive, especially for complex shapes or high-resolution images.

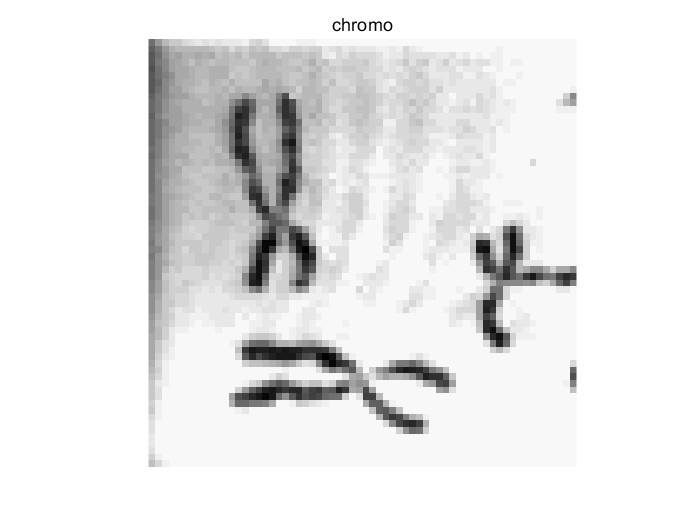
**Self-Organizing Map (SOM)**: SOM, also known as a Kohonen network, is suitable for pattern recognition, data visualization, and data compression. It is a self-organizing classifier that organizes and classifies input data through the network's own learning capabilities, maintaining the topology of the input data. SOM can handle large amounts of data and is suitable for exploratory data analysis and pattern discovery but may be challenging to determine optimal network parameters.

**Support Vector Machine (SVM)**: SVM is a supervised learning model used for classification and regression tasks. It works by finding the hyperplane that best separates different classes in the feature space. The SVM algorithm aims to maximize the margin between the data points of different classes, making the classifier robust to new samples. SVM is effective in high-dimensional spaces and is versatile, as it can be used with various kernel functions to handle non-linear classification. It's particularly useful in applications where the distinction between different classes is not immediately obvious in the raw feature space. However, the choice of the kernel and its parameters can significantly affect the performance of the SVM, and it requires careful tuning.

1. **Processing result**

## Display original image

We display an image using functions like “imread” and “imshow”, the process involves two key steps. First, “imread” reads the image data from a file and converts it into an array format that MATLAB can work with. For RGB images, this results in a 3-dimensional array, representing the image's width, height, and color channels. Then, when “imshow” is used, MATLAB maps the values in this array to the corresponding pixel values on your screen.



文本

描述已自动生成

Figure 8 Display Image

## Extract Sub-Image

The objective is to extract the middle section containing the text "HELLO, WORLD". The algorithm's primary focus is on dividing the image into three equal vertical segments and then selecting the middle segment for further processing. This is achieved by first determining the image's dimensions, then calculating the height of each third segment. The central segment is identified by defining a rectangular cropping area that starts from one-third of the image's height and extends to two-thirds. This cropping area is of the same width as the original image but only a third of its height. The “imcrop” function is then used to extract this specified middle section. Finally, this cropped sub-image is displayed, showcasing the targeted text area.



Figure 9 Sub-Image

## Threshold and binary image

Figure 10 shows the binary images obtained from the first image, chromo, using three different threshold segmentation methods. We can find that the binary image results vary greatly depending on the method chosen. Both the Otsu and iterative methods belong to global threshold segmentation, and the larger the threshold chosen, the more information is retained. On the other hand, Figure (c) uses a local threshold segmentation method, which retains a moderate amount of information. Figure 11 shows the binary images processed by Otsu threshold algorithm.

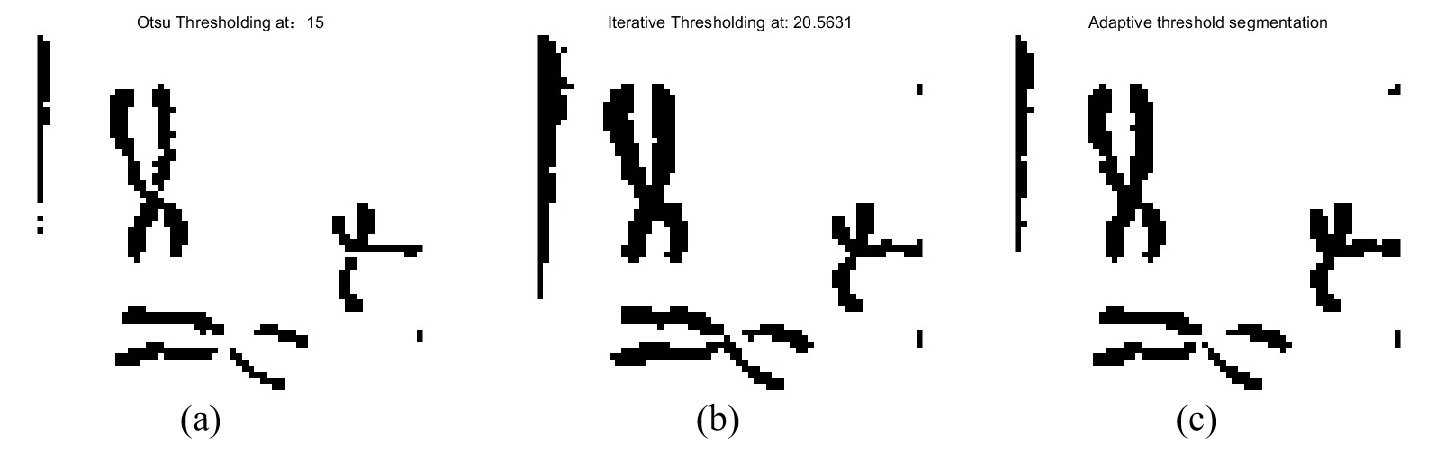


Figure 10 (a) Otsu threshold result; (b) Iterative threshold result; (c) adaptive thresholds segmentation



Figure 11 Otsu Threshold Result

## One-pixel thin image of object

Figure 13 shows the binary image obtained from Image 1 using the Otsu threshold segmentation algorithm and the results of one-pixel refinement. Figures (a) and (b) are the results obtained using the Zhang-Suen algorithm and the Medial Axis Transform algorithm, respectively. Comparing the two images, we can see that the image obtained from the Zhang-Suen algorithm has better connectivity and retains more pixels of the object, but it has some burrs. The image obtained from the Medial Axis Transform algorithm is simpler but has poorer connectivity.

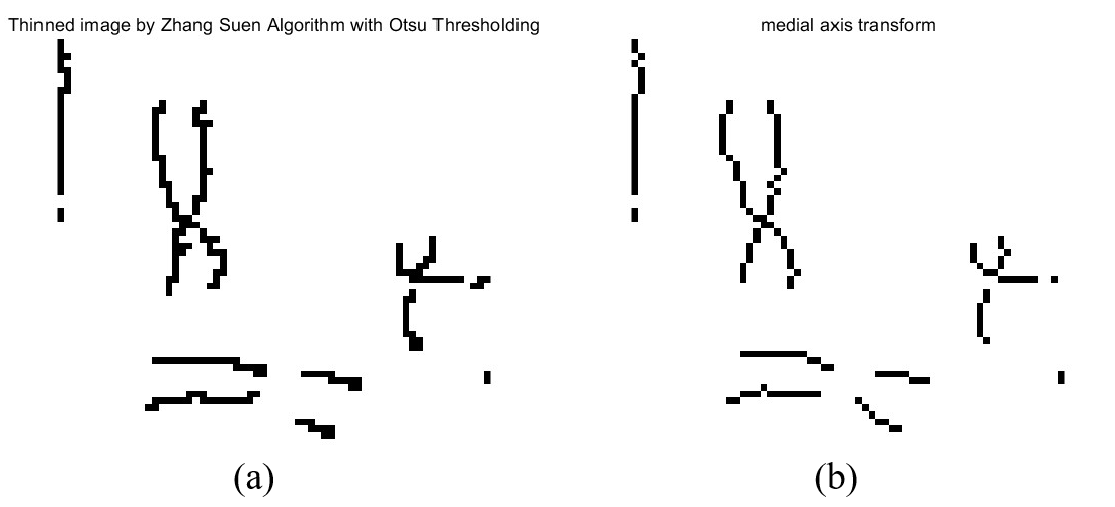


Figure 12 (a) Thinned image by Zhang-suen algorithm; (b) Thinned image by medial axis transform

As for Image2, we encountered unexpected errors when processing image 2 with the similar algorithm. The following is an analysis of the relevant reasons.

Original algorithm selection: The thinning method based on Zhang-Suen algorithm is selected, which is a typical eight-neighborhood based image thinning technology. The algorithm achieves refinement by iteratively removing pixels that meet certain conditions.

Problem identification: During the implementation, it was observed that thin lines like "twigs" appeared in the refined image, which may be caused by the instability of the noise amplification or thinning algorithm in the original image. The “randperm” function is used in the algorithm to randomize the order of processing pixels, which may lead to discontinuity and instability in the thinning process.

Tuning and optimization: Randomization is removed, and pixels are processed sequentially to enhance the stability and continuity of the algorithm. This paper discusses whether to adopt the four-neighborhood method, which may be more effective in text image processing. However, since the Zhang-Suen algorithm is essentially based on eight neighborhoods, this change requires a completely different algorithm design.

图形用户界面, 文本, 应用程序

描述已自动生成

Figure 13 Thinned image2 by Zhang-suen algorithm



Figure 14 Thinned Image2 by Using bwmorph tool

* 1. **Outline**

To extract and visualize the outline of objects within a binary image:

Initialization: The process begins by creating an image (outline\_img) with the same dimensions as the input binary image (binary\_img). This new image is designated to hold the detected outlines of objects.

Combining Scans: By integrating the results of both the row-wise and column-wise scans, the algorithm effectively delineates the complete outline of each object in the binary image. This includes both horizontal and vertical edges of the objects.

Output: The final result is an image where the outlines of objects are clearly marked, which can be useful in various applications like object detection, shape analysis, and image segmentation tasks in machine vision and image processing.

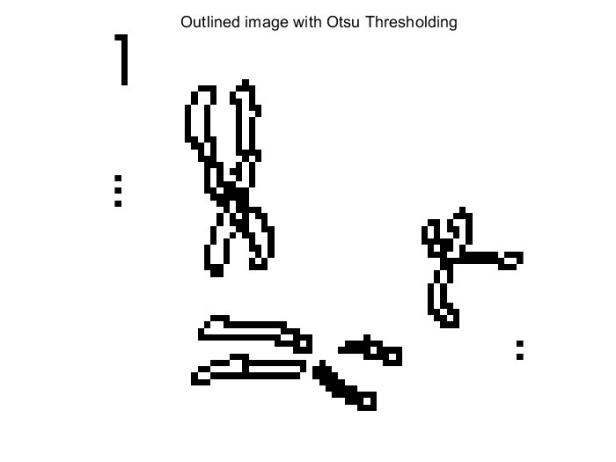


Figure 15 Outlined image of image 1



Figure 16 Outlined image of image 2

* 1. **Label**

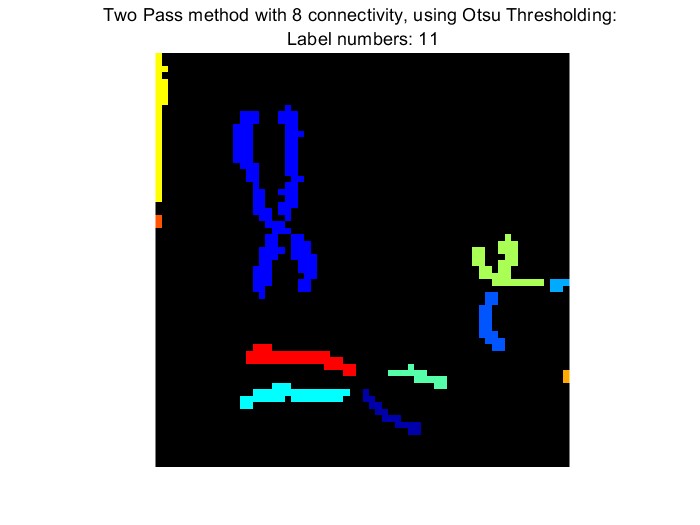


Figure 17 Label result by two pass method with 8 connectivity

In image2, Figure11 shows the result by using Otsu thresholding method. We observed that character ‘W’ and ‘O’ connects, which may bring troubles to label processing. Hence, we increase the threshold to eliminate the connected part, as is showed in Figure 18.

Then we use “bwlabel” function to label each connected component in the image, returning a labeled image and the total number of components. The regionprops function is then employed to obtain the bounding box of each component. An array is created to store the cropped images of each character. The code iterates through the connected components, using “imcrop” to extract each character based on its bounding box. Finally, the separated characters are displayed in a figure window, with each character shown in its subplot and labeled accordingly.



Figure 18 Binary Image by Modified Otsu Threshold Algorithm

Another method to solve this problem is adding a special condition processing. Considering the particularity of this example, we define it as follows: cut the BoundingBox with a width-to-height ratio greater than 1 into two parts with uniform width and assign independent counts to each. At the same time, to make subsequent results easy to find, we also numbered each individual character corresponding to the above-mentioned counting number.

文本

中度可信度描述已自动生成

Figure 19 Label and Segment Characters of Image2(Separating ’W’ and ‘O’ by Increasing the Threshold Value)

文本

描述已自动生成

Figure 20 Label and Segment Characters of Image2(Separating ’W’ and ‘O’ by Special Processing)

* 1. **Rotate the Image**

Figure 21 shows the results of rotating Image 1 clockwise by 30 degrees, 60 degrees, and 90 degrees, respectively. During the rotation, the output canvas was adjusted to make the output image large enough to contain the entire rotated image. Due to the use of bilinear interpolation, the rotated image does not have noise points, which proves that all pixel points have basically been successfully rotated to the target angle. The final processed image has excellent effects.

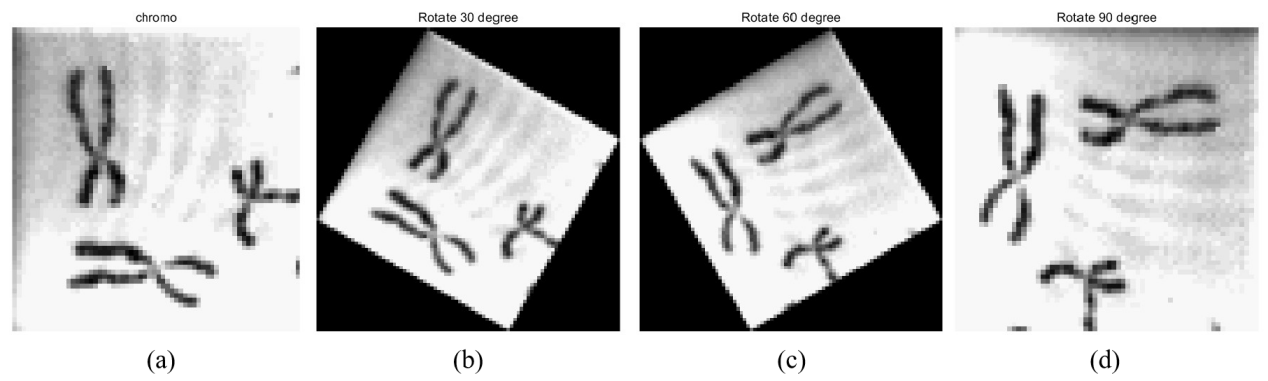


Figure 21 Different degree of Rotating of Image1

* 1. **Training**

In this project, three unsupervised learning methods - K-Nearest Neighbors (KNN), Self-Organizing Maps (SOM), and Support Vector Machines (SVM) - are used to train models with various feature extraction techniques.

Different approaches like RawPixels, Histogram, Edge Detection, Gabor Features, and Hough Transform are explored. RawPixels uses direct pixel values from images, providing simplicity but potentially leading to high dimensionality. Histograms analyze the color and intensity distribution, while Edge Detection focuses on identifying object boundaries. Gabor Features are sensitive to spatial frequencies and orientations, and the Hough Transform is used to detect geometric shapes.

The SOM classifier is a self-organizing network suitable for pattern recognition and data visualization, organizing input data based on its learning capabilities. This classifier does not require pre-labeled data and is effective in exploratory data analysis.

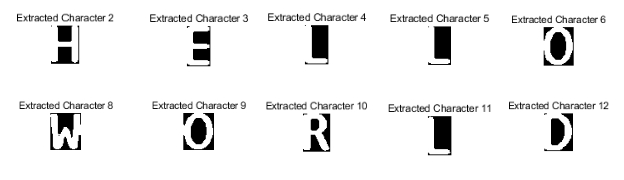
Each method is evaluated for its effectiveness in recognizing different characters, with particular attention given to the impact of feature extraction techniques on the performance of each classifier. The project demonstrates the application of these techniques in machine learning, highlighting their strengths, limitations, and suitability for various tasks in image processing and pattern recognition.

**KNN- RawPixels**

The first feature extraction method we adopted was the pixel value method. We introduced the data set, set labels according to the sub-folder titles, and integrated them into a total data set, shuffled it, and randomly called 75% of it. As the training set, the remaining 25% is used as the test set. In preliminary experiments we set the k value to 3. When evaluating the classifier, we divide the number of correct labels produced by the predict function by the total number of ratios. Since the prediction file needs to be converted into a binary image, benefiting from the conclusions obtained in Part 6, we directly extract the images corresponding to each separated character and display them independently, and at the same time add them to a total unit. in the lattice array extractedCharacters. It is worth noting that since the data set has black characters on a white background and the target image has white characters on a black background, each image needs to be color inverted. Afterwards, the elements in the array are extracted one by one, then adjusted to match the pixel size of the training data map, and the same (RawPixels) feature extraction method is used to extract features and predict them.

Accuracy under this setting is approximately 0.94376

The valid character numbers representing letters are 2, 3, 4, 5, 6, 8, 9, 10, 11, 12



The matches are respectively:

Classification results for character 2: SampleE

Classification results for character 3: SampleE

Classification results for character 4: SampleE

Classification result for character 5: SampleE

Classification result for character 6: SampleE

Classification results for character 8: SampleE

Classification results for character 9: SampleE

Classification result for character 10: SampleE

Classification results for character 11: SampleE

Classification results for character 12: SampleE

**KNN- Histogram**

Since the training process is basically the same and the only change is the feature extraction method, there will be no repetitive description here.

In preliminary experiments we set the k value to 10.

The Accuracy under this setting is about 0.28965

The matches are respectively:

Classification results for character 2: SampleD

Classification results for character 3: SampleD

Classification results for character 4: SampleR

Classification result for character 5: SampleR

Classification results for character 6: SampleD

Classification results for character 8: SampleD

Classification result for character 9: SampleD

Classification results for character 10: SampleD

Classification results for character 11: SampleD

Classification results for character 12: SampleD

**KNN-Edges**

In preliminary experiments we set the k value to 3.

The Accuracy under this setting is approximately 0.14286

The matches are respectively:

Classification results for character 2: SampleD

Classification results for character 3: SampleD

Classification results for character 4: SampleD

Classification results for character 5: SampleD

Classification results for character 6: SampleD

Classification results for character 8: SampleD

Classification results for character 9: SampleD

Classification results for character 10: SampleD

Classification results for character 11: SampleD

Classification results for character 12: SampleD

KNN-Gabor

In preliminary experiments we set the k value to 3.

However, due to unknown reasons, it may be that the combination of the classifier model and algorithm is not suitable for the given training set, causing the classifier training to be extremely slow and difficult to produce results.

KNN-HoughTransform

In preliminary experiments we set the k value to 3.

However, due to unknown reasons, it may be that the combination of the classifier model and algorithm is not suitable for the given training set, causing the classifier training to be extremely slow and difficult to produce results.

SOM- RawPixels

The Accuracy under this setting is about 0.96325

The matches are respectively:

Classification results for character 2: SampleH

Classification results for character 3: SampleE

Classification results for character 4: SampleL

Classification result for character 5: SampleL

Classification results for character 6: SampleO

Classification results for character 8: SampleW

Classification result for character 9: SampleO

Classification results for character 10: SampleR

Classification results for character 11: SampleL

Classification results for character 12: SampleD

7.2.2SOM- Histogram

The Accuracy under this setting is about

The matches are respectively:0.26531

Classification results for character 2: SampleH

Classification results for character 3: SampleE

Classification results for character 4: SampleH

Classification result for character 5: SampleH

Classification results for character 6: SampleO

Classification results for character 8: SampleW

Classification result for character 9: SampleO

Classification results for character 10: SampleO

Classification results for character 11: SampleH

Classification results for character 12: SampleO

7.2.3SOM- Edges

The Accuracy under this setting is about

The matches are respectively:0.68252

Classification results for character 2: SampleH

Classification results for character 3: SampleE

Classification results for character 4: SampleL

Classification result for character 5: SampleL

Classification results for character 6: SampleO

Classification results for character 8: SampleR

Classification result for character 9: SampleO

Classification results for character 10: SampleR

Classification results for character 11: SampleL

Classification results for character 12: SampleR

7.2.4SOM-Gabor

In preliminary experiments，however, due to unknown reasons, it may be that the combination of the classifier model and algorithm is not suitable for the given training set, causing the classifier training to be extremely slow and difficult to produce results.

7.2.5SOM-HoughTransform

In preliminary experiments，however, due to unknown reasons, it may be that the combination of the classifier model and algorithm is not suitable for the given training set, causing the classifier training to be extremely slow and difficult to produce results.

7.3SVM

Support vector machine (SVM) is a powerful supervised learning model used for classification and regression analysis. Proposed by Vladimir Vapnik and Alexey Chervonenkis in 1963, SVM is particularly suitable for classification problems of small and medium-sized complex data sets. SVM looks for the largest boundary between data categories to achieve the best classification. By using kernel functions, SVM is able to process linearly inseparable data and map it to a higher-dimensional space for classification.

Like KNN, it has strong generalization ability. By controlling the model complexity through regularization parameters, SVM avoids the problem of over-fitting. For linearly separable SVM, it looks for a hyperplane in the feature space to maximize the boundaries between data points of different categories. When the data is linearly inseparable, SVM will use the kernel technique to map the data to a high-dimensional space using the kernel function and find the dividing hyperplane in this space. By introducing slack variables, a certain degree of classification error is allowed to improve the generalization ability of the model. Commonly used kernel functions include: linear kernel, polynomial kernel and radial basis function (RBF) kernel, etc. Good at handling binary classification problems (such as spam detection, image recognition), multi-category classification problems and regression problems.

SVM performs well for small and medium-sized data sets, especially nonlinear data; it has good theoretical support and strong generalization ability. But it is computationally intensive for large-scale data sets; requires appropriate kernel functions and regularization parameters, which may require expertise; and is sensitive to missing data and noise. Overall, SVM is a powerful and flexible tool, especially when dealing with complex classification problems where the sample size is not particularly large. However, its effectiveness depends heavily on the correct choice of kernel function and regularization parameters. This experiment uses the default kernel function.

图表, 散点图

描述已自动生成

**SVM- RawPixels**

The Accuracy under this setting is about 0.97919

The matches are respectively:

Classification results for character 2: SampleH

Classification results for character 3: SampleE

Classification results for character 4: SampleL

Classification result for character 5: SampleL

Classification results for character 6: SampleO

Classification results for character 8: SampleW

Classification result for character 9: SampleO

Classification results for character 10: SampleR

Classification results for character 11: SampleL

Classification results for character 12: SampleD

**SVM- Histogram**

The Accuracy under this setting is about 0.32396

The matches are respectively:

Classification results for character 2: SampleH

Classification results for character 3: SampleE

Classification results for character 4: SampleL

Classification result for character 5: SampleL

Classification results for character 6: SampleO

Classification results for character 8: SampleR

Classification result for character 9: SampleO

Classification results for character 10: SampleR

Classification results for character 11: SampleL

Classification results for character 12: SampleO

**SVM- Edges**

The Accuracy under this setting is about

The matches are respectively: 0.94376

Classification results for character 2: SampleH

Classification results for character 3: SampleE

Classification results for character 4: SampleL

Classification result for character 5: SampleL

Classification results for character 6: SampleO

Classification results for character 8: SampleW

Classification result for character 9: SampleO

Classification results for character 10: SampleR

Classification results for character 11: SampleL

Classification results for character 12: SampleD

**SVM-Gabor**

In preliminary experiments，however, due to unknown reasons, it may be that the combination of the classifier model and algorithm is not suitable for the given training set, causing the classifier training to be extremely slow and difficult to produce results.

**SVM-HoughTransform**

In preliminary experiments，however, due to unknown reasons, it may be that the combination of the classifier model and algorithm is not suitable for the given training set, causing the classifier training to be extremely slow and difficult to produce results.

* 1. **Discussion on approaches**

pre-processing of the data & hyperparameter tuning

pre-processing of the data

The preprocessing of training set images can be roughly divided into the following aspects:

Adjustment of image (pixel) size (such as bilinear interpolation, cubic interpolation)

Use padding to maintain the image’s original aspect ratio

Different image enhancement techniques (such as contrast adjustment, histogram equalization)

Data enhancement techniques (such as rotation, scaling, shearing)

In this project, the image is complete and does not require filling operations. The data enhancement technology is to improve the model's generalization ability and has little impact on the sensitivity of the specific data set. Therefore, we focus on scaling and scaling the training image. image enhancement technology

Adjustment of image (pixel) size

This technique is generally implemented through the imresize function. By adjusting its size, we find that when increasing the pixel size of the image within a certain range, it helps to improve the performance of the classifier because it contains more details and information. But it will seriously slow down the speed of classifier learning. When it is larger than a certain value, the traditional machine learning methods we use (such as KNN or SVM) will suffer performance degradation due to the curse of dimensionality, and there will also be a risk of overfitting. In practical applications and when the added detail also contains noisy or irrelevant information, this can have a negative impact on certain types of classifiers. When we adjusted the pixel size from 32\*32 to 128\*128, after comparing the three classifiers horizontally, we came to the following conclusions: Sensitivity: KNN>SOM>SVM. This may be because the kernel trick of SVM helps it overcome the pixel The problem of size and KNN is affected by the "curse of dimensionality". After horizontally comparing three classifiers (that can produce results), the following conclusions are drawn: Sensitivity: Histogram > Raw Pixels > Edges). This may be because the histogram feature itself is based on the pixel intensity distribution, and the histogram equalization directly changes it. This distribution is obtained, which has a significant impact on histogram-based feature extraction. When the original pixels are used as features, they directly depend on the pixel intensity of the image. Histogram equalization can significantly affect raw pixel-based features by changing the pixel intensity distribution. For edge detection, it usually relies on local changes in pixel intensity. Histogram equalization may enhance these local changes, thereby affecting the results of edge detection.

Image enhancement technology

Increasing the contrast of an image can help highlight details in the image, especially if the image is dark or has low contrast.

But over-enhancement of contrast can lead to information loss, especially in extremely high or low areas of image brightness. Equalization improves the overall brightness distribution of an image, making different areas of the image more clearly distinguishable. This technique is particularly effective for enhancing low-contrast images, which can help classifiers better identify features. The image enhancement technique we use in this example is histogram equalization. After introducing histogram equalization and comparing the three classifiers horizontally, we came to the following conclusion: Sensitivity: KNN>SOM>SVM. This may be because the KNN classifier classifies images based on distance in high-dimensional space. Very sensitive to contrast and brightness distribution. After horizontally comparing three classifiers (that can produce results), the following conclusions are drawn: Sensitivity: Raw Pixels > Edges > Histogram). This may be because when using raw pixels as features, changes in image size directly affect the feature vector. dimensions.

hyperparameter tuning

The adjustment of hyperparameters is mainly divided into two parts. One is the feature extraction method, and the other is the K value of the classifier. For the feature extraction part, we have already performed the permutation and combination of commonly used learning models with KNN, SOM, and SVM in the seventh step. See Part 7 for specific values and 8.4 for analysis results.

For the adjustment of the k value, taking the KNN-Raw Pixels case as an example, we tried multiple values (values = 1:2:20;), and finally obtained the local optimum at k=3.

**Overall sensitivity analysis**

Judging from the debugging results, the sensitivity of the classifier is relatively slow in the behavior of preprocessing images. Although it does have a certain impact on accuracy, it seems to mainly affect the recognition of single target letters. The main purpose of adjusting the input data set is to ensure that the classifier can run normally (such as uniform input size). Of course, we improved the training speed of the model by reducing the pixel size, but this also means a loss of information, and the result is also reflected in the subsequent decline in accuracy. In addition, images are sometimes preprocessed to match a specific feature extraction method. For example, edge detection and normalized pixel values are processed on the dataset before Gabor mode processing, which can help improve its accuracy. However, through searching for information, we found that blindly improving the accuracy of images may not always have positive benefits. For example, when the number of data sets is insufficient, overfitting may occur.

Through adjustment, the adjustment of hyperparameters will have a greater impact on both the accuracy value and the accuracy rate. Among them, the impact of feature extraction methods is particularly serious. Under a specific data set and classifier model, choosing a suitable feature extraction method can not only significantly speed up the processing, but also improve accuracy and target text recognition capabilities. In general, the k value is used to adjust the accuracy under each classifier. Through our attempts, the size of the data set generally has nothing to do with the choice of k. A smaller k can improve accuracy, but will lead to overfitting.; Larger k will lead to underfitting. The feature extraction method seems to change the core of the classifier, allowing the training model to better adapt to different data environments.

Conclusion and recommendations (optimal combination)

After comprehensive observation, we believe that for the current data set and task, the ability of the classifier model is ranked as: SVM>SOM>KNN

For KNN, the Raw Pixels (k=3) feature extraction method is local optimal;

For SOM, the Raw Pixels feature extraction method is local optimal;

For SVM, the Raw Pixels feature extraction method is local optimal.

1. **Conclusion**

In this project, we successfully completed the tasks and displayed the results in the figure window. To obtain a binary image, we used three different methods for threshold division. The subsequent image processing is mainly based on the binary image obtained from the Otsu method division. At the same time, in the rotation image, we also encountered the problem of incomplete image display. After consulting the Matlab documentation, we finally successfully solved it.

Through constant attempts, we realized that although the front-end image processing process seems cumbersome, it is a very necessary pre-processing, because it can greatly optimize the subsequent classifier training and final results. We also learned about many different unsupervised learning models, analyzed the logic behind their operation, and understood their different characteristics and modes. Not only that, but there are also various options for feature extraction, and various combinations and preprocessing allow us to deal with complex and diverse real-life situations. Due to the size of the project and the breadth of knowledge, we also realized the importance of teamwork during the learning process. During the process of debugging the program, although errors were reported repeatedly and sometimes the results were not satisfactory, and due to time constraints, we have not yet been able to complete the ideal design, but looking back on our efforts along the way still makes us deeply touched.

In exploring specific algorithms, we discuss the effect of algorithm selection, parameter tuning on the result, and how to choose the appropriate method for the specific application scenario.

Overall, our projects cover a range of key concepts and techniques for image processing and machine learning, as well as the considerations and challenges of these techniques in practical applications.

At the end of our report, we would like to express our profound gratitude for the incredible learning experience it has provided. The platform to delve deeply into the intricacies of machine vision and its related technologies has been immensely valuable for all of us. We are particularly thankful to our instructor, whose expertise and guidance have been instrumental in demystifying complex concepts and fostering a practical understanding of the subject. The collaborative efforts and insights from our team members have also been pivotal in enhancing the learning experience, allowing for diverse perspectives and innovative problem-solving approaches. This course has not only equipped us with theoretical knowledge but has also honed our practical skills, preparing us for real-world applications in the field of machine vision. The discussions, projects, and hands-on exercises have been challenging yet rewarding, pushing the boundaries of our understanding and igniting a passion for further exploration in this dynamic field. Our appreciation extends to everyone involved in making this course a rich and enlightening journey.

1. **Reference**
2. Otsu, Nobuyuki. "A threshold selection method from gray-level histograms." IEEE transactions on systems, man, and cybernetics 9.1 (1979): 62-66.
3. Zhang, Tongjie Y., and Ching Y. Suen. "A fast parallel algorithm for thinning digital patterns." Communications of the ACM 27.3 (1984): 236-239.
4. **Appendix**