

Machine Vision Report

Group 22

Chen Du | A0263219E

Tu Shuqi | A0263424J

Yao Shun | A0263111X

Table of Contents

[List of Figures and Tables 2](#_Toc119600741)

[Machine Vision Work 1 5](#_Toc119600742)

[Introduction 5](#_Toc119600743)

[Recover the Original Image 5](#_Toc119600744)

[Thresholding and Convert to Binary Image 6](#_Toc119600745)

[Convert Binary image into one-pixel image 7](#_Toc119600746)

[Finding Outlines 8](#_Toc119600747)

[Component Labeling 8](#_Toc119600748)

[Image Rotation 9](#_Toc119600749)

[Machine vision work 2 10](#_Toc119600750)

[Original Image 10](#_Toc119600751)

[Sub-image of Middle Line 11](#_Toc119600752)

[Histogram to Binary 11](#_Toc119600753)

[One-Pixel Image, Outline and Labeling Components 12](#_Toc119600754)

[KNN and SVM Classification 12](#_Toc119600755)

[K-Nearest Neighbors 13](#_Toc119600756)

[Support Vector Machine (SVM) 13](#_Toc119600757)

[Local Binary Patterns (LBP) 13](#_Toc119600758)

[Histogram of Oriented Gradients (HOG) 14](#_Toc119600759)

[Gray-Level Co-occurrence Matrices (GLCM) 16](#_Toc119600760)

[Hu moments 18](#_Toc119600761)

[Overall Accuracy of KNN and SVM 19](#_Toc119600762)

[Self-Organization Map (SOM) 20](#_Toc119600763)

[Classifier for the Pre-processed Image 23](#_Toc119600764)

[Test for the resized image 25](#_Toc119600765)

[Discussion and Reflection 27](#_Toc119600766)

# List of Figures and Tables

[Figure 1 Original image 5](#_Toc119600767)

[Figure 2 Grayscale histogram of original image 6](#_Toc119600768)

[Figure 3 Binary image 6](#_Toc119600769)

[Figure 4 One-pixel thin image 7](#_Toc119600770)

[Figure 5 Outline of objects 8](#_Toc119600771)

[Figure 6 Label of different objects 9](#_Toc119600772)

[Figure 7 Linear interpolation for P 9](#_Toc119600773)

[Figure 8 Rotation of 30, 60, 90 degrees respectively 10](#_Toc119600774)

[Figure 9 Original image 11](#_Toc119600775)

[Figure 10 Sub-image comprising the middle line 11](#_Toc119600776)

[Figure 11 Grayscale histogram of original image 11](#_Toc119600777)

[Figure 12 Binary image 12](#_Toc119600778)

[Figure 13 One-pixel thin image 12](#_Toc119600779)

[Figure 14 Outline of objects 12](#_Toc119600780)

[Figure 15 Label of different images 12](#_Toc119600781)

[Figure 16 Confusion Matrix of KNN using LBP 14](#_Toc119600782)

[Figure 17 Confusion Matrix of SVM using LBP 14](#_Toc119600783)

[Figure 18 Confusion Matrix of KNN using HOG 15](#_Toc119600784)

[Figure 19 Confusion Matrix of SVM using HOG 16](#_Toc119600785)

[Figure 20 Confusion Matrix of KNN using GLCM 17](#_Toc119600786)

[Figure 21 Confusion Matrix of SVM using GLCM 17](#_Toc119600787)

[Figure 22 Hu moments 18](#_Toc119600788)

[Figure 23 Confusion Matrix of KNN using HU 18](#_Toc119600789)

[Figure 24 Confusion Matrix of SVM using HU 19](#_Toc119600790)

[Figure 25 Overall Accuracy of KNN and SVM 19](#_Toc119600791)

[Figure 26 Distribution of Categories of SOM initial learning rate 0.5 21](#_Toc119600792)

[Figure 27 Confusion Matrix of SOM initial learning rate 0.5 21](#_Toc119600793)

[Figure 28 Distribution of Categories of SOM initial learning rate 0.1 22](#_Toc119600794)

[Figure 29 Confusion Matrix of SOM initial learning rate 0.1 23](#_Toc119600795)

[Figure 30 accuracy after padding for KNN and SVM 24](#_Toc119600796)

[Figure 31 accuracy after resizing for KNN and SVM 24](#_Toc119600797)

[Flowchart 1 Recover the Original Image 5](#_Toc119600798)

[Flowchart 2 Thresholding and Convert to Binary Image 6](#_Toc119600799)

[Flowchart 3 Convert Binary image to One-pixel Image 8](#_Toc119600800)

[Flowchart 4 Finding Outlines 8](#_Toc119600801)

[Flowchart 5 Rotation 10](#_Toc119600802)

[Table 1 8-neighorhood of P1 7](#_Toc119600803)

[Table 2 Overall Accuracy of KNN and SVM 20](#_Toc119600804)

[Table 3 accuracy after padding for KNN and SVM 24](#_Toc119600805)

[Table 4 accuracy after padding for KNN and SVM 24](#_Toc119600806)

[Table 5 Test Result for LBP, using KNN and SVM 25](#_Toc119600807)

[Table 6 Test Result for HOG, using KNN and SVM 25](#_Toc119600808)

[Table 7 Test Result for GLCM, using KNN and SVM 26](#_Toc119600809)

[Table 8 Test Result for Hu moment, using KNN and SVM 26](#_Toc119600810)

[Table 9 Test Result for SOM 27](#_Toc119600811)

# Machine Vision Work 1

## Introduction

In this section, we will use several traditional methods to process a coded image and analysis its property.

Firstly, we need to recover the coded picture to an image. And then we will threshold the image and covert it into binary image to distinguish background and object. To have a straightforward view of the object, we will convert the binary image to a one-pixel thin image by deleting boundary pixels to get a thin image. Also, we will find the outline also by finding different in both row and column edge and combine row and column. Then we will find different objects by using connectivity. In the end we will rotate the original image.

## Recover the Original Image

To recover the original image from the text, first we need to read the txt file into an array. The coded image is a 32-level image with characters 0-9 and A-V. Since the characters are represented in ASCII code in MATLAB, we also need to convert these characters back to numbers. From the ASCII table, we need to minus 55 for each letter and minus 48 to each number. And then we can display the original picture by the numerical matrix.

Flowchart Recover the Original Image



Figure Original image

## Thresholding and Convert to Binary Image

Thresholding is a simple way to distinguish background and object. In our original image, we can tell that white pixels are represented as background, which means pixels with higher grayscales are backgrounds and lower grayscales are objects. To find a threshold value for the image, we need to build histogram for the image first, and then we need to manually select the histogram value.

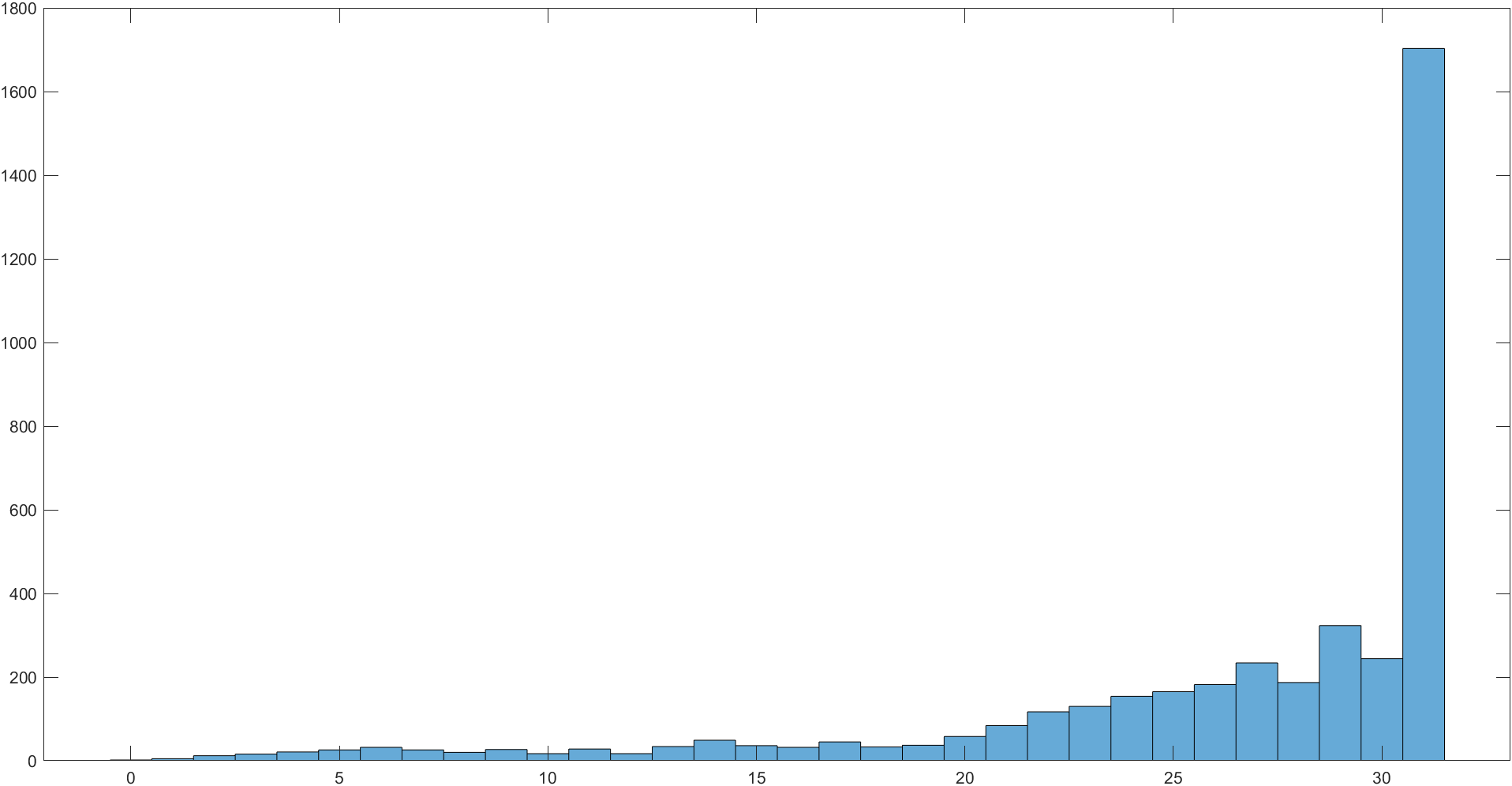


Figure Grayscale histogram of original image

Noted that grayscale is mainly concentrated in 31, which is the background. The object is mainly concentrated below 20. Thus, the threshold is selected as 21, which has a good effect on image binarization. After thresholding, value 0 represents object and value 1 represents background. The binary image is shown in the figure below.



Figure Binary image

Flowchart Thresholding and Convert to Binary Image

## Convert Binary image into one-pixel image

Based on binary image, T. Y. ZHANG and C. Y. SUEN proposed a fast parallel thinning algorithm to turn objects into one pixel thin[1]. The specific steps of this algorithm are as follows.

Step 1: For one-pixel P1, its 8-neighborhood is:

|  |  |  |
| --- | --- | --- |
| P9  (i-1,j-1) | P2  (i-1,j) | P3  (i-1,j+1) |
| P8  (i,j-1) | P1  (i,j) | P4  (i,j+1) |
| P7  (i+1,j-1) | P6  (i+1,j) | P5  (i+1,j+1) |

Table 8-neighorhood of P1

Step 2: Judge whether it satisfies the following conditions:

1. 2≤B≤6. Where B is the number of nonzero neighbors of P1.
2. A(P1)=1; A is the number of 01 patterns.
3. P2\*P4\*P6=0.
4. P4\*P6\*P8=0.

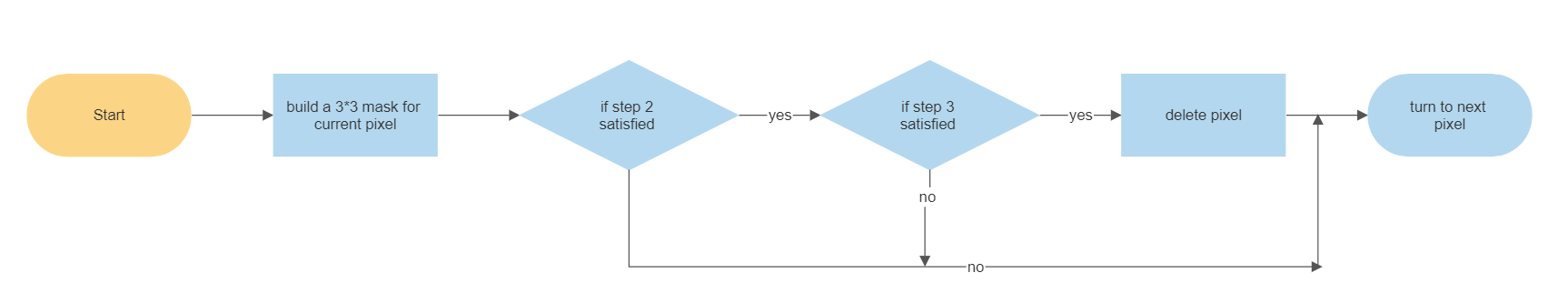
Step 3: If step 2 is not satisfied, P1 will not be deleted. If step 2 is satisfied, judge whether it satisfies the following conditions.

1. P2\*P4\*P8=0.
2. P2\*P6\*P8=0.

If step 3 is satisfied, P1 will be deleted. The result is as follows.



Figure One-pixel thin image



Flowchart Convert Binary image to One-pixel Image

## Finding Outlines

To find outlines of the image, the image is searched by row and by column. For each row, if a pixel is different from the one on the right, it is marked with a value of 1. Also, for each column, if a pixel is different from the one on the bottom, it is marked with a value of 1. Thus, two values are used to describe one pixel. By performing ‘OR’ operation on two values, the result obtained is the outline of the image.



Figure Outline of objects

Flowchart Finding Outlines

## Component Labeling

Breadth First Search is used to label different objects. For a pixel belonging to an object, it will be put into a queue. Its 8-neighborhood pixels which also belong to this object are also put into a queue, and so on. Then, from head to tail, use the same label to label the pixels in the queue. After that, queue is cleared, label adds one and become a new label to label the other objects.

After all the pixels are labeled, all pixels have values, and different objects have different labels. Thus, the whole labeled image can be displayed in grayscale, the same grayscale represents one object.

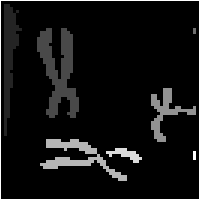


Figure Label of different objects

## Image Rotation

Rotation is used to rotate the matrix. The rotation matrix is:

Where is the angle of rotation. Each original point will have a corresponding point in target image through the rotation matrix. The coordinates of each point in the new image obtained are not integers, which means that rounding is necessary. However, rounding results in many unassigned points in the new image, and many black points in the new image. Thus, linear interpolation is used to solve this problem.

In mathematics, linear interpolation is a method using linear polynomials to construct new data points within the range of a discrete set of known data points. In this case, inverse rotation is done in the new image to obtain the corresponding grayscale. For a point P’ in the new image, through inverse rotation, P in the original image will be obtained. If P locates between four adjacent points, the grayscale of P can be calculated from the following formula.

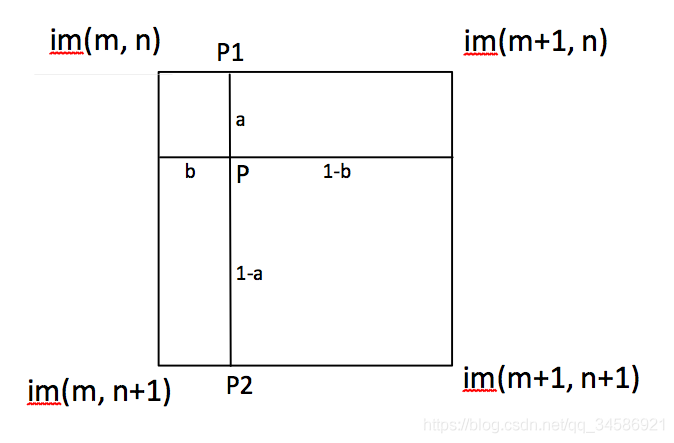


Figure Linear interpolation for P

Linear interpolation can effectively solve the problem of missing points, and make the whole image smoother. The results of a rotation angle of 30, 60 and 90 degrees respectively are shown as follows.



Figure Rotation of 30, 60, 90 degrees respectively

Flowchart Rotation

# Machine vision work 2

Introduction

In this section, we will process the picture of command window. We will plot the original image and extract the middle line from the image. Then the first following questions are similar as previous steps that we have already done: thresholding to a binary image, finding outline and label components. The processes are almost the same so the flowchart for these processes will be omitted. The last two question are new: we will use a dataset for an unsupervised classification to find out the characters from the image.

## Original Image

The original image is shown as follows.

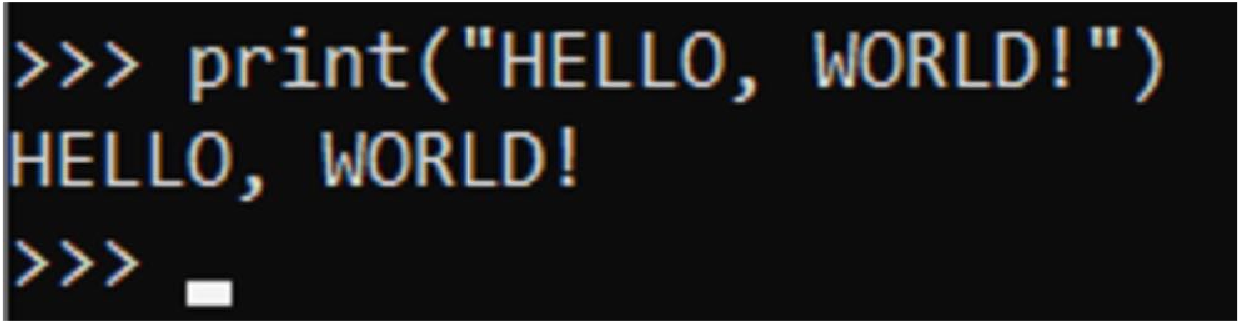


Figure Original image

## Sub-image of Middle Line

To create the sub-image of the original comprising the middle line, intercept from one third to two thirds of the height, and omit the part without letters behind.



Figure Sub-image comprising the middle line

## Histogram to Binary

The comprising grayscale histogram is shown in the figure.

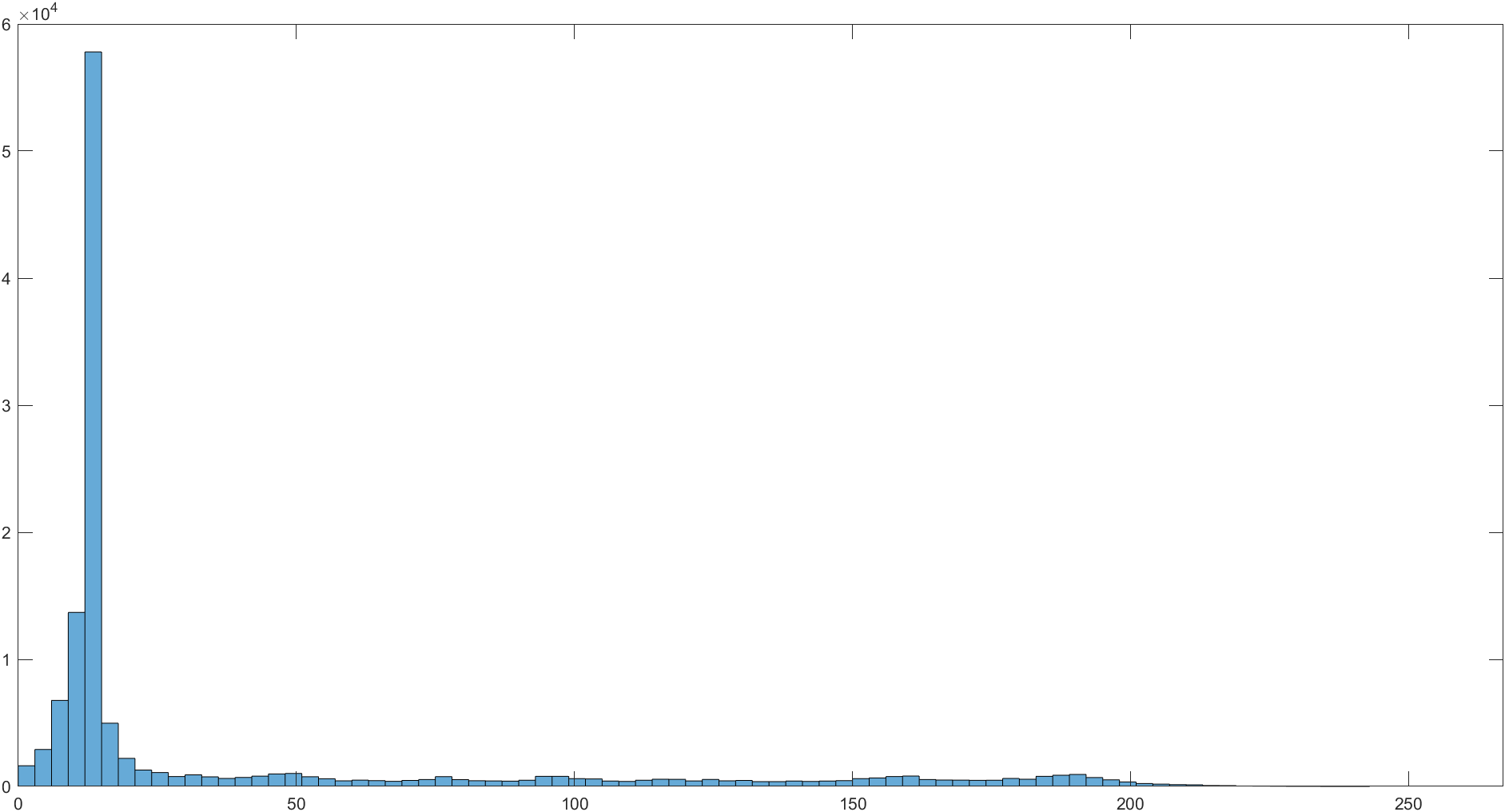


Figure Grayscale histogram of original image

Noted that grayscale is mainly concentrated between 3 and 21, which represents the background. After some experiments, the grayscale of letters is mainly higher than 100. Thus, thresholding is selected as 100. The result of binarization is shown in Figure 12.

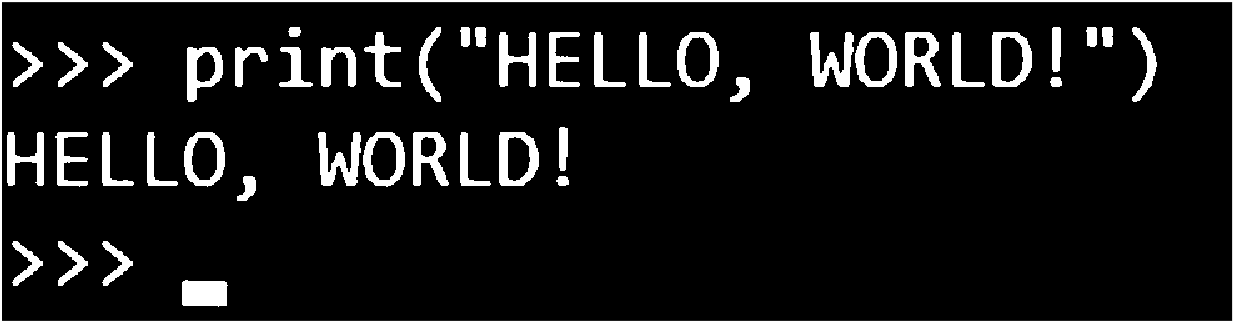


Figure Binary image

## One-Pixel Image, Outline and Labeling Components

The fast parallel thinning algorithm and the method to find outline also have a good performance in this case. It is worth noting that, contrary to the previous example, 0 represents the background and 1 represents the object. Therefore, when using the above method, you should first reverse the result, and then reverse the result again. The one-pixel thin image and outline are shown as follows.

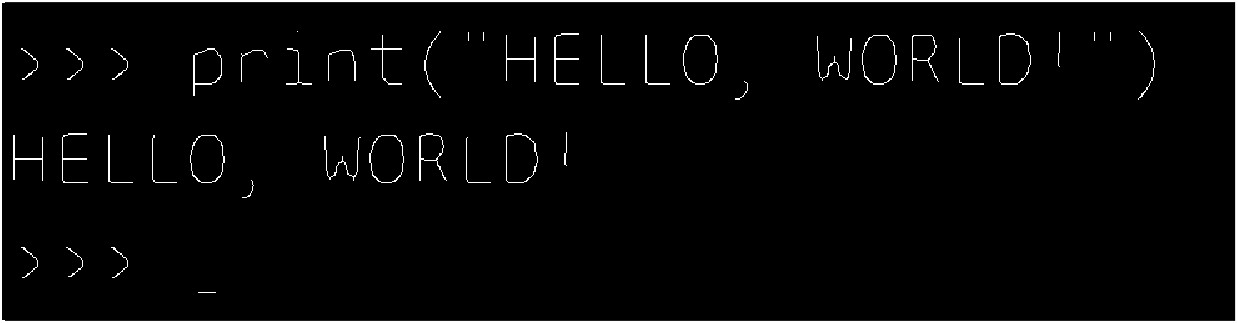


Figure One-pixel thin image

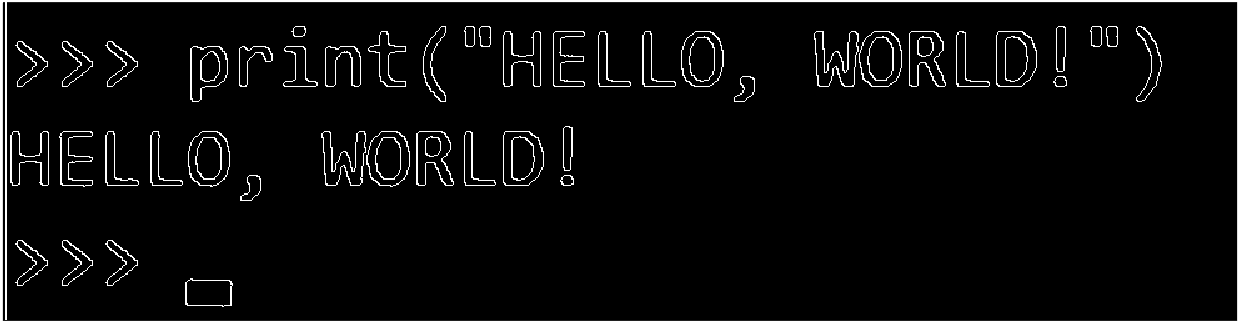


Figure Outline of objects

Similarly, Breadth First Search is used to label different objects. The results of label are shown in Figure 15.



Figure Label of different images

## KNN and SVM Classification

In this section, we use both SVM and KNN methods to classify the characters in the image ‘hello world’. The following image is the confusion matrix of training set and testing set. In these figures, the horizontal axis refers to sample’s label, and the vertical axis refers to prediction’s label. The gray boxes are TP, TN, FP, FN and other relevant unbalanced distribution sample evaluation index parameters. For the KNN and SVM classification, we use four different image process: LBP, HOG, GLCM and Hu.

### K-Nearest Neighbors

* + Label all the reference images and generate a 𝑁-dimensional feature space.
  + Represent all reference images as data points within the feature space.
  + Represent the target image in the feature space, and calculate its distance to all reference images (in feature space). Arrange the reference images in ascending order of the distances.
  + Identify the 𝑘 nearest neighbors around the target image (better if k is odd, here k = 3).
  + The target image is classified as the label of the majority class among these k-nearest neighbors.

### Support Vector Machine (SVM)

Key idea: separate classes with a hyperplane, including convex optimization problem with constraints:

### Local Binary Patterns (LBP)

The LBP feature vector is created in the following manner:

* Divide the examined window into cells.
* For each pixel in a cell, compare the pixel to each of its 8 neighbors. Follow the pixels along a circle, clockwise or counter-clockwise.
* Where the center pixel's value is greater than the neighbor's value, write “0”. Otherwise, write “1”.
* Compute the histogram, over the cell, of the frequency of each "number" occurring. This histogram can be seen as a 256-dimensional feature vector.
* Concatenate (normalized) histograms of all cells. This gives a feature vector for the entire window.
* The feature vector can now be processed using the Support Vector Machine and K Nearest Neighbors.



Figure Confusion Matrix of KNN using LBP



Figure Confusion Matrix of SVM using LBP

### Histogram of Oriented Gradients (HOG)

HOG process the figure as the following:

* Crop out a region of interest.
* For each pixel (x, y), calculate the magnitude 𝜇 and angle 𝛳 gradient, 𝐺𝑋 and 𝐺𝑌, where 𝐼 is the intensity of the pixel.
* The matrices of magnitude and angle are divided into 8x8 cells to form a block. For each block, a 9-point histogram is calculated with 9 bins with an angle range 𝛥𝜃 of 20 degrees.
* Clubbing is done to reduce the size of a block, by overlapping 4 blocks into a single block with only 4 cells. With that, the number of blocks should be reduced by 1 in both directions.
* To reduce the effect of lighting on different parts of the image, normalize the histograms.
* Combine the histogram of the blocks to generate a HOG feature vector.



Figure Confusion Matrix of KNN using HOG



Figure Confusion Matrix of SVM using HOG

### Gray-Level Co-occurrence Matrices (GLCM)

Given a grey-level image , co-occurrence matrix computes how often pair of pixels with a specific value and offset occur in the image.

The offset is a position operator that can be applied to any pixel in the image (ignoring edge effects)

An image with p different pixel values will produce a p\*p co-occurrence matrix, for the given offset.

The value of the co-occurrence matrix gives the number of times in the image that the and pixel values occur in the relation given by the offset.

For an image with p different pixel values, the p\*p co-occurrence matrix C is defined over an n\*m image , parameterized by an offset as:



Figure Confusion Matrix of KNN using GLCM



Figure Confusion Matrix of SVM using GLCM

### Hu moments

Hu Moments (or rather Hu moment invariants) are a set of 7 numbers calculated using central moments that are invariant to image transformations. The first 6 moments have been proved to be invariant to translation, scale, and rotation, and reflection. While the 7th moment’s sign changes for image reflection.

The 7 moments are calculated using the following formulae:

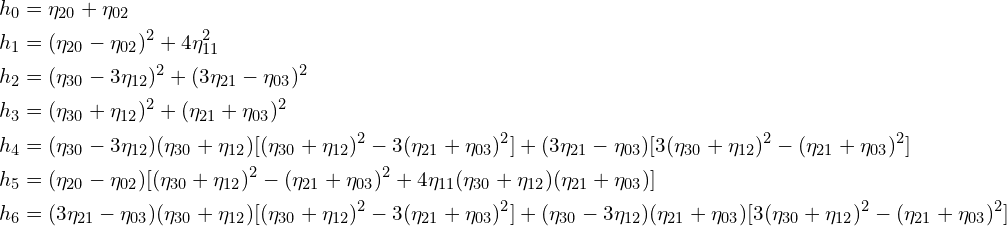


Figure Hu moments



Figure Confusion Matrix of KNN using HU



Figure Confusion Matrix of SVM using HU

### Overall Accuracy of KNN and SVM

图形用户界面, 应用程序, 表格, Excel

描述已自动生成

Figure Overall Accuracy of KNN and SVM

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 'accuracyTable' | 'LBP' | 'HOG' | 'GLCM' | 'Hu' |
| 'KNN' | 0.910011248593926 | 0.929133858267717 | 0.515748031496063 | 0.622047244094488 |
| 'SVM' | 0.913948256467942 | 0.936445444319460 | 0.367266591676041 | 0.556242969628796 |

Table Overall Accuracy of KNN and SVM

### Self-Organization Map (SOM)

SOM method are the following steps:

* Initialize the weights
* At iteration n, randomly select vector from dataset
* Under Euclidean distance, find closest weight 𝑤𝑗 as Best Matching Unit (BMU) to
* Update towards ,
  + - 𝜇(𝑛): learning rate, decreases with time
    - : topological distance to BMU
* Repeat 2-4 until max number of iterations, or until converges
* Make contextual map: for each , give the label of closest 𝑥𝑖 in all data set to
* For a new data vector 𝑦𝑖 (new image): label is that of closest

In our process, we tried two different initial learning rate. The first trial’s parameters are given below:

* Self-organizing map network size: 10\*10
* The total number of neuron nodes is 100
* Iterations: 1000
* Initial learning rate: 0.5



Figure Distribution of Categories of SOM initial learning rate 0.5

We find the total accuracy = 0.3633



Figure Confusion Matrix of SOM initial learning rate 0.5

And the second trial’s parameters are given:

* Self-organizing map network size: 10\*10
* The total number of neuron nodes is 100
* Iterations: 1000
* Initial learning rate: 0.1



Figure Distribution of Categories of SOM initial learning rate 0.1



Figure Confusion Matrix of SOM initial learning rate 0.1

The total accuracy is 0.3318.

### Classifier for the Pre-processed Image

To measure the classifiers’ sensitive to pre-processed image, we also used the classifiers above to test the padded and resized image. The results are shown below:

#### Padding

应用程序, 表格, Excel

描述已自动生成

Figure accuracy after padding for KNN and SVM

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 'accuracyTable' | 'LBP' | 'HOG' | 'GLCM' | 'Hu' |
| 'KNN' | 0.300000000000000 | 0.100000000000000 | 0.300000000000000 | 0.200000000000000 |
| 'SVM' | 0.300000000000000 | 0.600000000000000 | 0.300000000000000 | 0.200000000000000 |

Table accuracy after padding for KNN and SVM

For SOM, both two initial training rates give the accuracy of 0.3.

#### resizing

截图里有图片

描述已自动生成

Figure accuracy after resizing for KNN and SVM

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 'accuracyTable' | 'LBP' | 'HOG' | 'GLCM' | 'Hu' |
| 'KNN' | 0.800000000000000 | 0.700000000000000 | 0.300000000000000 | 0.300000000000000 |
| 'SVM' | 0.400000000000000 | 0.700000000000000 | 0.400000000000000 | 0.100000000000000 |

Table accuracy after padding for KNN and SVM

For SOM, both two initial training rates give the accuracy of 0.1.

## Test for the resized image

The size of the original image is different from our training data, so we need to use resized image for our final test of the image ‘HELLO WORLD’. The results are given below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| LBP | | | | |
| No. | character | label | KNN | SVM |
| 1 | D | 1 | 1 | 4 |
| 2 | E | 2 | 2 | 3 |
| 3 | H | 3 | 3 | 3 |
| 4 | L | 4 | 4 | 4 |
| 5 | L | 4 | 4 | 4 |
| 6 | L | 4 | 4 | 4 |
| 7 | O | 5 | 3 | 4 |
| 8 | O | 5 | 3 | 4 |
| 9 | R | 6 | 6 | 3 |
| 10 | W | 7 | 7 | 3 |

Table Test Result for LBP, using KNN and SVM

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| HOG | | | | |
| No. | character | label | KNN | SVM |
| 1 | D | 1 | 2 | 2 |
| 2 | E | 2 | 2 | 2 |
| 3 | H | 3 | 3 | 3 |
| 4 | L | 4 | 4 | 4 |
| 5 | L | 4 | 4 | 4 |
| 6 | L | 4 | 4 | 4 |
| 7 | O | 5 | 4 | 2 |
| 8 | O | 5 | 5 | 2 |
| 9 | R | 6 | 3 | 6 |
| 10 | W | 7 | 7 | 7 |

Table Test Result for HOG, using KNN and SVM

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| GLCM | | | | |
| No. | character | label | KNN | SVM |
| 1 | D | 1 | 2 | 2 |
| 2 | E | 2 | 2 | 2 |
| 3 | H | 3 | 2 | 1 |
| 4 | L | 4 | 4 | 4 |
| 5 | L | 4 | 4 | 4 |
| 6 | L | 4 | 7 | 4 |
| 7 | O | 5 | 2 | 2 |
| 8 | O | 5 | 2 | 2 |
| 9 | R | 6 | 2 | 2 |
| 10 | W | 7 | 1 | 2 |

Table Test Result for GLCM, using KNN and SVM

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Hu | | | | |
| No. | character | label | KNN | SVM |
| 1 | D | 1 | 1 | 4 |
| 2 | E | 2 | 6 | 2 |
| 3 | H | 3 | 6 | 2 |
| 4 | L | 4 | 4 | 2 |
| 5 | L | 4 | 4 | 2 |
| 6 | L | 4 | 1 | 2 |
| 7 | O | 5 | 2 | 1 |
| 8 | O | 5 | 6 | 6 |
| 9 | R | 6 | 4 | 1 |
| 10 | W | 7 | 4 | 4 |

Table Test Result for Hu moment, using KNN and SVM

|  |  |  |  |
| --- | --- | --- | --- |
| No. | character | label | SOM |
| 1 | H | 3 | 6 |
| 2 | E | 2 | 7 |
| 3 | L | 4 | 5 |
| 4 | L | 4 | 4 |
| 5 | O | 5 | 7 |
| 6 | W | 7 | 4 |
| 7 | O | 5 | 7 |
| 8 | R | 6 | 4 |
| 9 | L | 4 | 7 |
| 10 | D | 1 | 4 |

Table Test Result for SOM

From these tables above, the test results are satisfied with our previous test result for Resized image.

# Discussion and Reflection

In the above two sub-projects, we use several different ways of image processing. In the traditional methods, thresholding the image to binary image is very important to the following processes including converting to one-pixel thin image, outline detection and labeling different components since all the processes above requires distinguish between background and objects. Also, thresholding enables us to eliminate noise in the figure. Therefore, a proper thresholding value need to choose carefully such that the following steps can have better result. Converting the image to one-pixel image allows the computer to observe the object’s shape in a simpler and more accessible way. While finding outline is a better way to observe the object for human being. During the process of rotation, we need to use linear polynomials to find grayscale for each pixel, to avoid the mismatch for non-integer pixel during the rotation.

In the process of classification, we use four different image processing methods for KNN and SVM, among which LBP and HOG both show high accuracy for KNN and SVM, whose accuracy rate are above 90%, while GLCM and Hu moments are less accurate. SOM is also used in the classification; however, the accuracy rate is relatively low compared with KNN and SVM.

We also test our classifier for the preprocessed image including padding and resizing. In our test, all the three classifiers are sensitive to padding, only SVM with HOG remains accuracy of 0.6. While in the resized image, KNN with LBP and HOG, SVM with HOG remains high accuracy. All other methods fail.

Considering all the classification and preprocessing tests, SVM with HOG shows the best result in both accuracy in classification and preprocessing resistance.

# Reference

1. Zhang T Y, Suen C Y. A fast parallel algorithm for thinning digital patterns[J]. Communications of the ACM, 1984, 27(3): 236-239.