Final Project of Digital Image Processing Handwritten Text Recognition

Jiaqi Han, Lihua An and Zui Chen

***Abstract*—Paragraph recognition for handwritten Chinese and English texts is a vital research area with diverse applications, including the digitization of ancient manuscript archives and automated grading systems. In this study, we present the development and experimentation of recognition algorithms for Chinese and English paragraphs on a single image for each language. For English recognition, we employed techniques such as skew correction, underline removal, line and word segmentation, and a two-stage recognition approach based on template matching. Multiple evaluation metrics were used to assess the performance of our algorithm. Our approach achieved excellent results with a character error rate (CER) of 0.133 for underlined text recognition and a CER of 0.252 for non-underlined text recognition. For Chinese recognition, we applied denoising techniques, line and word segmentation, and a one-stage recognition approach based on template matching. The accuracy of our algorithm was evaluated using character accuracy rates, with results of 82.1% achieved without manual fine-tuning and 100% accuracy achieved with manual fine-tuning. Through this project, we extensively practiced various techniques and algorithms involved in the entire process of paragraph recognition, gaining valuable insights into the application of digital image processing in the field of optical character recognition.**

***Index Terms*—English Handwriting Recognition, Chinese Handwriting Recognition, Template Matching, Handwritten Text Recognition**

I. Introduction

THE The digitization of written content is crucial in today's information-driven society, facilitating the effective organization, retrieval, and analysis of textual data. OCR technology has emerged as a powerful tool for automated document processing, enabling the conversion of printed text into editable and searchable electronic formats. While OCR systems have achieved remarkable accuracy for printed text in various languages, the recognition of handwritten characters (HTR) poses unique challenges due to the inherent variability and complexity associated with individual writing styles.

J. H. Author, is with Shanghaitech University, Shanghai, China. He is now postgraduate of SIST. (e-mail: hanjq2022@shanghaitech.edu.cn).

L. A. Author, is with Shanghaitech University, Shanghai, China. She is now postgraduate of SIST. (e-mail: anlh2022@shanghaitech.edu.cn).

Z. C. Author, is with Shanghaitech University, Shanghai, China. He is now postgraduate of SIST. (e-mail: chenzui2022@shanghaitech.edu.cn).

II. Task

The project consists of two sub-tasks, English text recognition and Chinese text recognition. For both tasks, symbols are not required to be recognized and deep learning methods are not allowed.

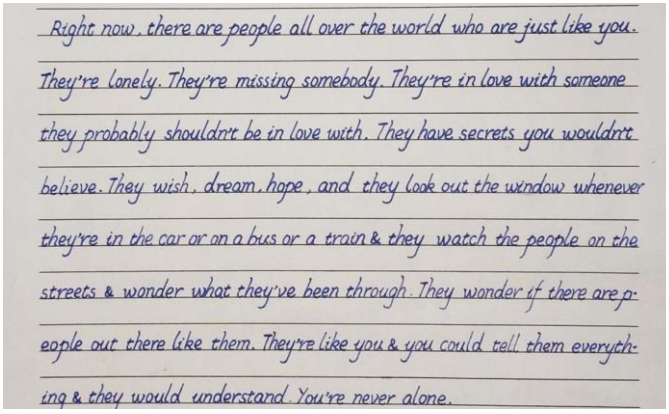

Fig. 1. The input image 1, paragraph of handwritten English text to be recognized.

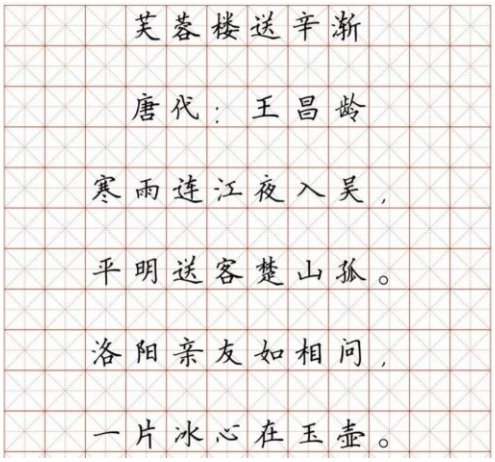

Fig. 2. The input image 2, paragraph of handwritten Chinese text to be recognized.

In the announcement of the project, it is recommended to separate the recognition process into 5 steps: 1. Do denoising to eliminate the influence of unrelated factors; 2. Preprocess the image; 3. Segment the characters; 4. Perform text recognition; 5. Calculate the final result, which is the recognition accuracy.

The evaluation metric is accuracy, which is defined as the number of correctly recognized characters divided by the total number of characters. (However, for English task, this metric can be controversial, and we will provide detailed explanations of the evaluation criteria in the subsequent sections.)

III. English Recognition

We completed the recognition of English paragraphs first, as we believe this task is more challenging compared to the Chinese recognition task.

A. Observations

Observing the input, we found several features of the input.

Firstly, the handwritten text is written on a paper with lines, so the bottom of text is overlapped with the lines, we call these lines underlines.

Secondly, the text is in Italics, not Roman. This makes segmentation very hard to apply, thus making recognition harder.

Thirdly, the writing style of a person a paragraph typically has consistency, manifested in the basic similarity of each word's form.

B. Pipeline

Based on the observations mentioned above, a pipeline is designed.

1) Preprocessing

In preprocessing, skewed letters are corrected to make them in standardized form. Subsequently, underlines are removed, and binary thresholding is applied for denoising.

After the preprocessing, the words are in normal form and the underlines are removed, which makes the image easy to do segment.

2) Segment lines & words

In this stage, lines are segmented, followed by the segmentation of individual words. This process allows us to obtain subgraphs for each word, with special emphasis on preserving the spaces in the output.

3) Recognition

We employ a recognition algorithm based on two-stage template matching to recognize each subgraph. Subsequently, the recognition results of the subgraphs are merged according to their sequence to generate the final recognition result.

4) Evaluation

We conducted experiments to automatically compare the recognition results. Several metrics were calculated to evaluate the performance.

C. Preprocessing

1) Skew corrections

By utilizing the Hough transform, we can observe the orientation of all lines and curves present in the image. The angles corresponding to the most prominent orientations have the highest values on the Hough transform.

Among these angles, the most prominent orientation is horizontal since, up to this point, the underlines have not been removed, and the text in the image is arranged in rows. By masking the values in the Hough transform that represent horizontal and near-horizontal angles, we remove the influence of horizontal orientation. Then, we search for the maximum value point on the remaining Hough transform, which represents the orientation of the vertical lines that should be present in all English characters. We have identified the maximum value point at an angle of 65.97°.

Subsequently, we applied a horizontal shear using the angle of 65.97° and successfully correcting the image. The transformation matrix and the resulted image are displayed below.

$$\begin{bmatrix} 1 & tan(90° - 65.97°) & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

2) Removal of underlines

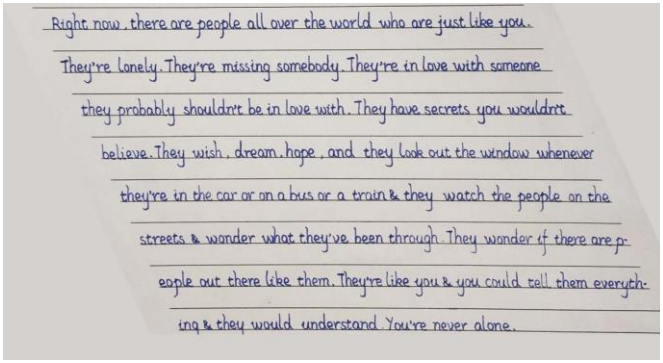

Fig. 3. The sheared image, where all characters are no longer skewed.

Using morphological methods, we create an appropriate structure element to represent horizontal lines, and use open operations to identify and extract all the lines. Afterwards, the horizontal lines are removed from the original binary image based on the position of the identified lines.

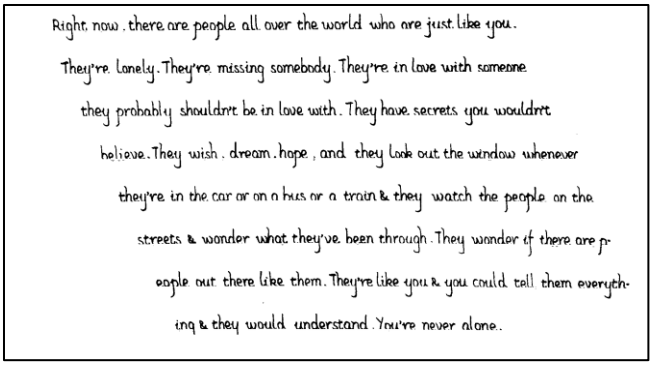

Fig. 4. The denoised image, where the image is binarized and the underlines are completely removed. Several repairs are done on the cut-off vertical lines.

However, directly deleting the lines at this point will cause text to break, which is detrimental to subsequent recognition work. Therefore, before removing lines, it is necessary to have an additonal operation. When finding the line, scan the binary image from left to right along the line. If the width of the line is less than 4 pixels, it is considered as a straight line, not a part of the text, and then fill it in white; on the contrary, in the case where a line is a part of the text, then no action is taken on it.

D. Segment lines & words

1) Text line segmentation

The text line segmentation process starts with the input of a grayscale image of a student's handwritten text. The primary goal of the method is to correctly identify and extract the lines of text in the composition.

Algorithm 1 Text Line Segmentation Process

Input: Grayscale image
Output: Beginning and ending indices of each valid text line
1. Slice the input grayscale image to keep central region
2. Binarize the sliced image with a threshold of 100
3. Identify all row indices of black pixels (value = 0)
4. Initiate lists for the beginning and ending indices of each line of text
5. for each pair of consecutive indices in the list of black pixel row indices do
 if the difference between them is greater than the minimum permissible gap then
 mark the pair as the end of the previous line and the start of the next line
 add these to the lists of line beginning and ending indices
 end if
 end for
6. Remove lines from the lists whose height (ending index - beginning index) is less than the minimum line height
7. return the beginning and ending indices of all remaining lines

The process begins by slicing the image to remove the extreme edges, focusing on the central region where most of the text is expected to be found. This sliced image is then subjected to a binarization process. The binarization threshold is set at 100, differentiating the text (black, pixel value <= 100) from the background (white, pixel value > 100).

The binarized image is then scanned for the presence of black pixels, representing text content. The pixel positions of these black pixels are then analyzed to deduce the structure of the handwritten text in terms of lines. A key aspect of this analysis is the identification of 'gaps' between lines of text. A gap is defined as a vertical distance in the image where no black pixels are found, exceeding a predefined minimum distance. Whenever such a gap is found, it signals the boundary between two lines of text.

After identifying the boundaries, we have a rough segmentation of the text into different lines. However, not all segments are necessarily valid lines of text. Some segments may simply be noise or artifacts. To filter out these invalid segments, we impose a minimum line height criterion. Any segment whose height is less than this value is discarded.

The final output of the process is the beginning and ending indices for each valid line of text in the composition, providing a clear and precise segmentation that can be used for further analysis or processing.

2) Word segmentation

After the text line segmentation process, the next step is to further segment these lines into individual words. This is achieved through a process akin to the previous line segmentation but specifically tailored to handle the characteristics of words in terms of spacing and size.

The grayscale image of a segmented line of text is first binarized to differentiate the handwritten text (black) from the background (white). With the binarized image at hand, we move towards locating the black pixels which signify the handwritten text. This time, the focus is on the column indices of these black pixels, as the objective is to carry out horizontal segmentation of words within a line.

Considering the impact of the underline, we applied the filtering process. Specifically, we disregard columns where the count of black pixels is less than a certain cutoff number. This value serves to counteract the potential influence of underlines, by effectively ignoring columns that primarily contain an underline but not significant text.

We then interpret the remaining column indices as a series, where a significant jump between consecutive indices indicates the presence of a space between words. By defining a minimum gap distance, we can accurately delineate the boundaries of individual words. The identified segments are further scrutinized for their width. Segments with a width smaller than a pre-determined minimum width, are dismissed as invalid word segments, as they are more likely to be artifacts or isolated punctuation marks.

In conclusion, the word segmentation process returns the beginning and ending indices of each valid word within the provided line of text. This method allows for reliable and accurate word segmentation, which is crucial for subsequent text recognition and analysis tasks.

Algorithm 2 Word Segmentation Process

Input: Grayscale image
Output: Beginning and ending indices of each valid word
1. Binarize the input grayscale image with a threshold of 100
2. Identify all column indices of black pixels
3. Trim the identified indices where the black pixel count is less than a cutoff number
4. Initiate lists for the beginning and ending indices of each word
5. for each pair of consecutive indices in the trimmed list do
 if the difference between them is greater than the minimum permissible gap then
 mark the pair as the end of the previous word and the start of the next word
 add these to the lists of word beginning and ending indices
 end if
 end for
6. Remove words from the lists whose width (ending index - beginning index) is less than the minimum word width
7. return the beginning and ending indices of all remaining words.

We achieve 100% accuracy on text line segmentation and 100% accuracy on word segmentation. However, character level segmentation is too hard without intensive parameter adjustment, so we ignore it.

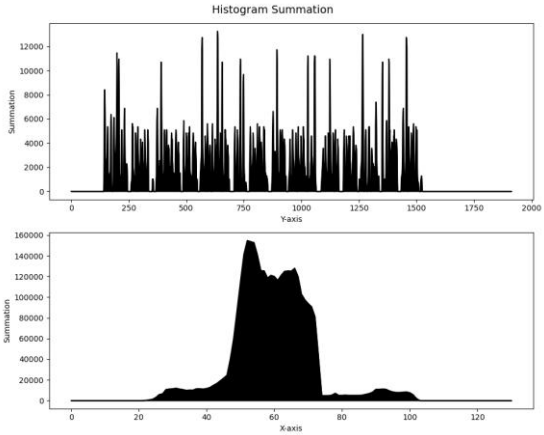

Fig. 5. The y-axis, x-axis histograms of the first line. Each nearly-zero minimum in y-axis histogram is part of a white space or left & right border.

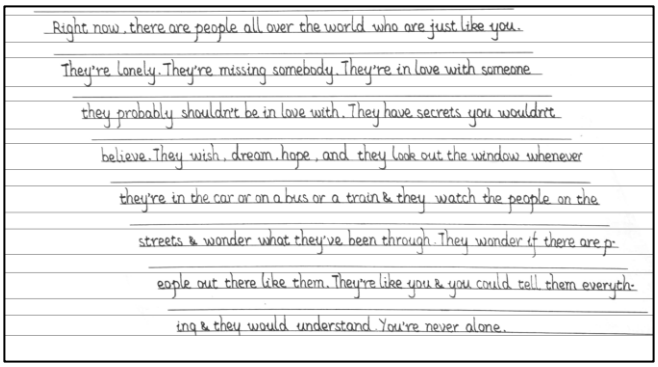

Fig. 6. The visualization of the recognized lines, where black lines are drawn for the upper and lower boundaries of each line. In actual segmentation, the upper and lower boundaries have some relaxation (20 pixels each boundary).

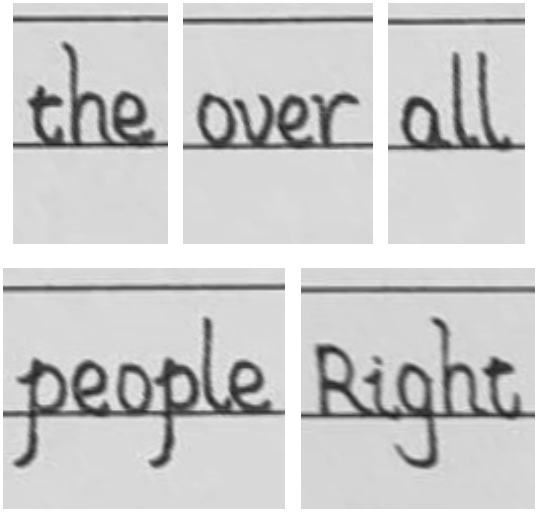

Fig. 7. Several segmented sub-images of words.

E. Recognition

1) Basic idea

The basic idea of the recognition algorithm is to use template matching.

Template matching can be seen as a very basic form of object detection. To find the template image (T) in the source image (I), we slide the template from left to right and from top to bottom in the source image. At each (x, y) position, a correlation coefficient is calculated to represent the matching situation. The choice of similarity calculation algorithm has a significant impact on the recognition performance.

We experimented with various template matching algorithms and ultimately found that the normalized correlation coefficients algorithm yielded the best results. We use this algorithm to measure the similarity between two image blocks:

$$R(x,y) = \frac{\sum_{x',y'} (T'(x',y') \cdot I'(x+x',y+y'))}{\sqrt{\sum_{x',y'} T'(x',y')^2 \cdot \sum_{x',y'} I'(x+x',y+y')^2}}$$

For each position of template T on source image I, take the image blocks of the overlapping parts of the two, calculate the similarity results, and store them in our result matrix R. The closer the calculated coefficient value is to 1, the higher the matching degree.

Although template matching is very simple and computationally efficient, there are many limitations. If there are any changes in scale, rotation, or perspective of the target, template matching may fail. During the pre-processing, we have corrected the original image and filtered the background, making template matching easier to achieve higher performance. However, for template matching, the decisive factor remains the quality of the template itself. The quality of the template directly determines the success of the matching process. Obtaining templates that are most suitable for matching the target image is crucial.

2) Two-stage recognition algorithm

Based on the assumption of handwritten paragraph recognition: an individual maintains consistent writing style within the same paragraph. Combined with observations of the target image, we believe that the shape of each identical character in the image is essentially consistent. Therefore, as long as we can find a suitable template for each character, we can effectively recognize all the characters.

However, we are not allowed to directly extract templates from the original image. Therefore, we have designed a two-stage recognition method to indirectly obtain templates from the original image, aiming to achieve optimal recognition results.

In the first stage, we utilize characters (only those that appear in the image) from templates obtained from other sources. The sizes of the templates are modified manually. These templates undergo size adjustments and some fine-tuning to create initial matching templates. Using these templates, we perform template matching on the original image. For each character,

we identify the position on the original image that yields the highest response. Based on these positions, we extract regions of the same size as the initial matching templates to serve as templates for the second round of matching.

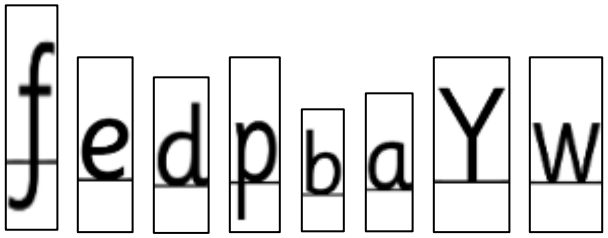

Fig. 8. Several templates of the first stage.

In this stage, the majority of characters can be found with the correct new templates on the original image (where the new templates correspond to the correct characters rather than different characters). However, to maximize the recognition performance, we conducted manual guided parameter adjustments in this step. This ensured that all characters could be matched with their correct counterparts on the original image.

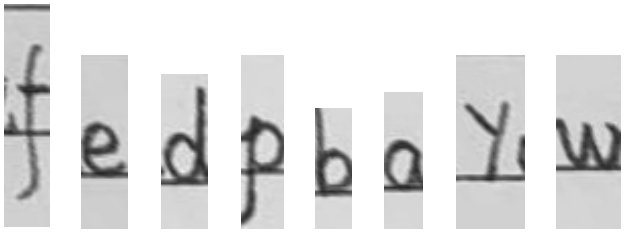

Fig. 9. Several templates of the second stage.

In the second stage, we utilize templates obtained from the original image for matching, on each sub-image of single word and obtain candidate positions (represented as bounding boxes, considering that each candidate position corresponds to a template with a certain size). The selection of candidate positions is modulated by the character frequency in English to prevent excessive recognition of low-frequency characters. Taking the character 'e' as a reference, which is the most common character in English, the number of candidate bounding boxes is limited to a range of 1000 to 2000. An adaptive threshold algorithm is employed to ensure that the number of candidate bounding boxes falls within this range. For any given character x, the upper & lower bound of candidate bounding boxes is determined by $bound_x = bound_e \times \frac{freq_x}{freq_e}$, where $bound_e$ represents the bound of character ‘e’, $freq_e$ represents the character frequency of character ‘e’, $freq_x$ represents the character frequency of x.

After performing the second-stage matching for each character, we gather all the bounding boxes together and apply non-maximum suppression (NMS) based on their confidence scores. We set the Intersection over Union (IoU) threshold to 0.15 to minimize the recognition of overlapping characters.

By performing the aforementioned steps, we are able to obtain a recognition result for each word. Subsequently, we concatenate the words together using spaces based on their order to obtain the final recognition result, which is a string:

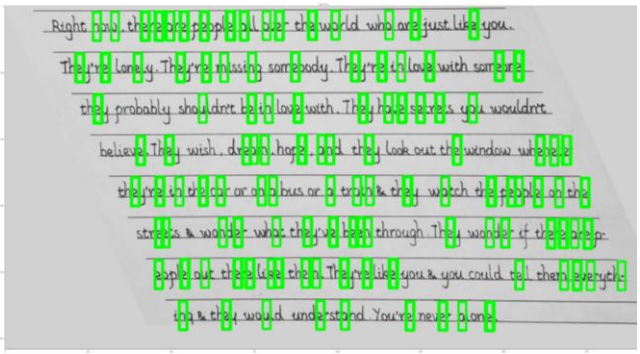

Fig.10. The candidate bounding boxes of character 'e'

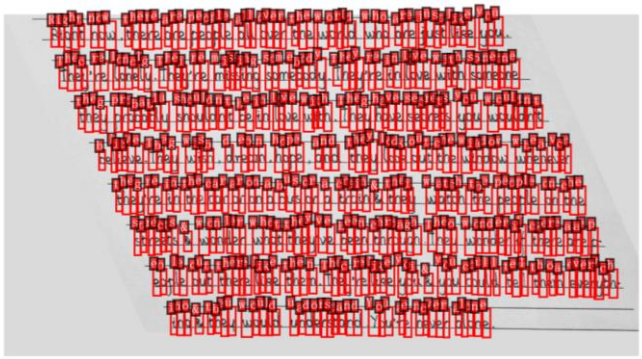

Fig. 11. The NMS result of the whole image. Note that this step is not necessary for recognition. The actual template matching and NMS are done on sub-images of each word.

"Right now there are peoole all over the world who are just like you Thecre lonclg Theipre missing somebody Theyre in love with soneone theg probablu shouldnt be in love with Theg have secrets you wouldnt blicve Tbeg wisb dbbeofn hope and they lodk out che wiadow wbeacvcr thegre in the car ar on a bus or a crai tnei watch tbe oeople on che strect wonler whot theyve been chroagh Theo wonder if ckere aae pe eople out there like then Thecpre like yoi you coull tel chcia eocrychs in tbeu would uadorstand Youjae ncver alone"

F. Evaluation

1) Evaluation metrics

In the Task section we mentioned the accuracy is defined as the number of correctly recognized characters divided by the total number of characters.

However, due to the nature of paragraph OCR, the concept of the number of correctly recognized characters is undefined without alignment.

If we adopt a character-by-character comparison starting from the first position of the ground truth and check the correctness of each character, the accuracy could be very strange if there is any misalignment in the predicted sequence, for example, ‘bast’ recognized as ‘loast’ results in 0% accuracy. Yet, this accuracy does not match human perception, and the problem becomes even more severe when considering the entire text as a single string.

One possible approach is to calculate the number of incorrect characters for each word, sum up the total number of errors across all words, and then calculate the accuracy. However, this method of calculating accuracy is not a universally recognized approach and may appear to be an invented metric.

Instead, the standard measurements of OCR are based on Minimum Edit Distance, also known as Levenshtein Distance (LD), which is the minimum steps of single-character edit (replace, insert or delete) from one string to another.

There are two widely recognized metrics in the field of handwritten paragraph recognition: CER and WER.

Both of them are calculated by $\frac{\text{LD}}{COUNT_{GT}}$, where $COUNT_{GT}$ is the count of the ground truth (character count for CER and word count for WER), and LD is the Levenshtein Distance between predicted and ground truth (character level for CER and word level for WER).

If we insist on calculating a so-called character-level accuracy, we can approximate it by defining the predicted string as $\frac{COUNT_{GT}-\text{LD}}{COUNT_{GT}}$, where LD is the character level Levenshtein Distance. This equals to 1 minus the Character Error Rate (1-CER). Note that this is not a recognized metric, and tend to be a bit better than human perception.

Also, if the words can be correctly recognized, it is also possible to compute the word-level accuracy.

2) Experiments & Results

For English recognition, we conducted two experiments. The difference is the templates and the images. One experiment uses the sheared image (with underlines) and templates with underlines. The other experiment uses the preprocessed image (without underlines, denoised and binarized) and templates without underlines. The result is shown below.

Templates & Image	CER ↓	WER ↓	Word Accuracy ↑	Character Accuracy ↑
With underline	0.133	0.443	55.67%	87.63%
Without underline	0.252	0.701	22.68%	75.81%

Table. 1. English recognition result

We speculate that image & templates with underlines have better recognition performance due to the preservation of additional information. Although underlines are considered as noise, they do not have a significant impact. However, in the case of removing underlines and performing image denoising, the removal of underlines and binary denoising may result in a substantial loss of information compared to the benefits of denoising. Also, the template matching algorithm might not perform well on binary images, compared to grayscale images.

IV. Chinese Recognition

The pipeline and methods of Chinese recognition largely inherit those of English recognition, with only a few minor differences.

A. Observations

Observing the input, we found several features of the input.

Firstly, the handwritten Chinese characters are all written on a pale red grid paper with a grid pattern, and they are arranged in a very organized manner.

Secondly, Chinese recognition operates at the character level, rather than at the word level as in English recognition. Furthermore, almost every Chinese character appears only once.

Thirdly, Chinese characters, despite not being in standard form, do not have a uniform slant angle.

B. Pipeline

1) Preprocessing

In preprocessing, the background grid pattern will be removed, leaving only the characters.

2) Segment lines & characters

In this stage, lines are segmented, followed by the segmentation of individual words. This process allows us to obtain subgraphs for each word.

3) Recognition

We employ a recognition algorithm based on one-stage template matching to recognize each subgraph. Subsequently, the recognition results of the subgraphs are merged according to their sequence to generate the final recognition result.

4) Evaluation

We conducted experiments to automatically compare the recognition results with the ground truth.

C. Preprocessing

For the background of Chinese recognition, it is easy to filter by color due to the large difference of color between font and background. We use the HSI color space to extract and remove the red part by selecting an appropriate threshold, and then use the dilation operation to connect the breakpoints of the text to obtain good background removal results. The resulting text is very clear.

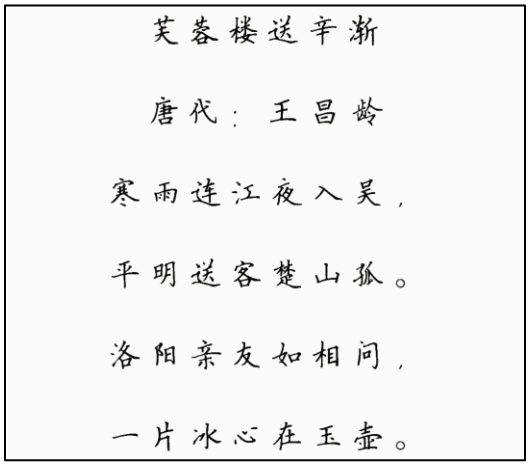

Fig. 12. The preprocessed image

D. Segment lines & characters

The segmentation process is generally same to English segmentation, but do not require thresholding.

E. Recognition

Since almost all Chinese characters appear only once in the image, two-stage template matching becomes meaningless. Therefore, we proceed with single-stage template matching.

Images of Chinese characters are extracted from a standard font library and resized accordingly, then used as templates to perform template matching on the preprocessed image.

For each sub-image got from segmentation, the character with maximum response is chosen. Subsequently, we concatenate the words together to obtain the final recognition result.

F. Evaluation

Segmentation of Chinese characters easily gets 100% correct, so we can calculate the character-level accuracy. We use accuracy as the only evaluation metric.

We compared the results with two sets of templates, one is only adjusted with global size, and one is has 9 characters fine-tuned at stroke level.

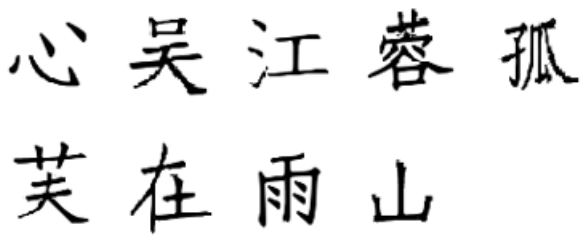

Fig. 13. The 9 stroke-level fine-tuned characters

Templates	Accuracy
Regular script	82.1%
Modified script	100%

Table. 2. Chinese recognition result

Due to the presence of numerous visually similar characters in Chinese and their uniform size, the differentiation between characters is much smaller compared to English when scaled to the same size. As a result, the recognition performance is not ideal. Only through targeted adjustments to the stroke size and position using Microsoft Paint tool, can we achieve 100% recognition. However, it is important to note that such targeted optimizations are subject to controversy.

V. Discussions

A. Fit the single image

The task of this project is a bit strange, since it gives only one image for English and Chinese recognition respectively. So what we actually do is to fit the test case.

Many algorithms we designed aim to achieve the highest correctness we could achieve, but not generalization capability, such as the two-stage template matching algorithm, which may perform badly for characters not appear in the image. The first stage of this algorithm also requires severe manual parameter adjustment to achieve best performance.

The lack of generalization makes our implementation have little use in more general scenarios. The significance of undertaking this project has been severely diminished.

A wiser design of the task is to provide a series of samples, and conduct test on a previously unseen test case for comparison.

VI. Conclusion

In this project, we solved the two recognition tasks via various recognition approaches. This project is a beneficial practice to use non-learning approach to do OCR. The outcome shows the effectiveness of simple DIP algorithms, though still inferior to deep-learning based algorithms.

VII. WorkLoad Distribution

A. ***Jiaqi Han***

1) *Recognition algorithms*
2) *Code integration*
3) *Presentation*
4) *1/2 part of the report*

B. ***Lihua An***

1) *Preprocessing algorithms*
2) *Realize cv2 built in algorithms*
3) *1/4 part of the report*

C. ***Zui Chen***

1) *Segmentation algorithms*
2) *1/4 part of the report*

References

[1] Yoo, Jae-Chern, and Tae Hee Han. "Fast normalized cross-correlation." Circuits, systems and signal processing 28 (2009): 819-843.

[2] Li, Minghao, et al. "Trocr: Transformer-based optical character recognition with pre-trained models." arXiv preprint arXiv:2109.10282 (2021).