A SKETCH-BASED EDUCATIONAL SYSTEM FOR LEARNING CHINESE HANDWRITING

A Thesis

by

TIANSHU CHU

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Tracy Hammond
Committee Members,	Yoonsuck Choe
	Noboru Matsuda
Head of Department,	Dilma Da Silva

December 2017

Major Subject: Computer Science

ABSTRACT

Learning Chinese as a Second Language (CSL) is a difficult task for students in English-speaking countries due to the large symbol set and complicated writing techniques. Traditional classroom methods of teaching Chinese handwriting have major drawbacks due to human experts' bias and the lack of assessment on writing techniques. In this work, we propose a sketch-based educational system to help CSL students learn Chinese handwriting faster and better in a novel way. Our system allows students to draw freehand symbols to answer questions, and uses sketch recognition and AI techniques to recognize, assess, and provide feedback in real time. Results have shown that the system reaches a recognition accuracy of 86% on novice learners' inputs, higher than 95% detection rate for mistakes in writing techniques, and 80.3% F-measure on the classification between expert and novice handwriting inputs.

DEDICATION

To my parents and grandparents who supported and encouraged me all the way through.

ACKNOWLEDGMENTS

I would like to express my gratitude to all the people who helped me throughout the way of finishing my thesis. I would like to give my special thanks to my advisor Dr. Tracy Hammond, who has guided and inspired me with her enormous knowledge and excellence in research. I would like to thank Dr. Yoonsuck Choe for offering his insights in Machine Learning, which has helped me greatly throughout the project. I would like to thank Dr. Noboru Matsuda, who is always willing to share his thoughts about language education.

My thanks go to the members of Sketch Recognition Lab. My special thanks go to Paul Taele for sharing his years long expertise in the field of Asian handwriting education. My thanks also go to Seth Polsley, Raniero Lara-Garduno, Vijay Rajanna, and Larry Powell, from whom I have gained valuable advice and feedback throughout this research.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis committee consisting of Professor Tracy Hammond [advisor] and Professor Yoonsuck Choe of the Department of Computer Science and Engineering, and Professor Noboru Matsuda of the Department of Teaching, Learning, and Culture.

All other work conducted for the thesis was completed by the student independently.

Funding Sources

The student has completed all work without receiving financial support.

NOMENCLATURE

AI	Artificial Intelligence
CSL	Chinese as a Second Language
DTW	Dynamic Time Warping
IRB	Institutional Review Board

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

Chinese has become a popular second language to learn in the Western world just for the past two decades [1]. It has been indicated by education researchers that learning handwriting is the most crucial as well as the most difficult part on the way to master the Chinese language [2]. While traditional classroom teaching methods are prone to human experts' bias and lack interactive feedback and instructions [3], we propose a sketch-based educational system that revolutionizes the way CSL students learn Chinese handwriting with the assistance of sketch recognition and Artificial Intelligence (AI) techniques.

1.1 The need for a new interactive educational system for learning Chinese handwriting

It can be an extremely difficult task for a Westerner to learn to recognize and even write Chinese characters because they have no resemblance with any Western language [4]. Multiple factors can be blocking the success for CSL learner to acquire Chinese handwriting in a short time. The Chinese language has a symbol set that numbers in thousands, while English has only 26 letters. The large number of symbols has not only made it almost impossible to to learn all the characters in a short time for CSL learners, but more importantly, since there is not a consistently obvious rule for writing techniques that can be applied to the entire symbol set, students need to gradually gain their perception and experience through a long-term practice. Writing techniques can be especially hard for students with fluency in English because each Chinese character is formed with a combination of several to tens of poly-line strokes in fixed ordering and directions, while each English letter has no more than 4 strokes. Moreover, there exist numerous visually similar characters that have completely different meanings, making it harder for students to recognize and memorize.

An efficient way of education can help CSL learner acquire the crucial skills in writing Chinese characters. Traditional classroom methods of teaching Chinese handwriting have major drawbacks. First of all, human teachers can have their bias on grading students' writing quality due to the long history and complexity of Chinese characters [5]. Moreover, the way human teachers assess students' writing ability is solely based on the visual forms of written characters, while the process of writing can be even more important for beginners. In addition, there are always too many students in classroom teaching methods, making it impossible for teachers to offer personalized feedback to each specific student. These problems can be solved by our proposed sketch-based educational system. By employing sketch recognition techniques, the system is able to accurately tell if a correct symbol has been written for a given question, and analyze both stroke and vision data to assess the student's performance. With the big amount of data we collected from both novice and expert users, we are also able to utilize Machine Learning techniques to find out important features to distinguish between well and badly written samples. With these combined, the system is able to give feedback in accordance with each student's performance in real time, and help them improve their writing skills.

1.2 Sketch recognition applied in educational systems

Sketch recognition is the automated recognition of hand-drawn diagrams by a computers [6, 7, 8]. It can be applied to classroom activities that involve writing and drawing. It revolutionizes the way teachers teach and students learn in knowledge domains where handdrawing is the most efficient way to express ideas and convey messages. The past decade has seen sketch recognition algorithms being applied in systems in different education domains that include civil engineering [9, 10, 11, 12], engineering design [13, 14], music theory [15], as well as East Asian language handwriting [16, 17, 18, 19]. One of the major improvements sketch recognition brings to these domains is that both gesture and vision information can be used to recognize and assess students' sketches, while human teachers usually only have access to the latter. While vision data presents the final form of the sketch, the lack of analysis and feedback on the process of drawing can prevent students from forming a good drawing habit. It has been shown that in some specific domains, it is the features that reflect how the user perform the drawing, rather than the final visual structure that distinguish experts from novices [20]. With sketch recognition techniques being applied in educational systems, students' drawing traces and speed can are kept track of, which provide valuable information to the instructors.

1.3 Benefits of sketch-based interactive learning in Chinese handwriting

Real-time feedback is crucial for students to realize their mistakes and make corrections accordingly [21]. Compared to traditional teaching methods, sketch-based educational systems can assess students' writing samples in real time and give immediate feedback to the students, which helps them form a good writing habit. In the early stages of learning Chinese handwriting, students can always be confused with the writing techniques in each character. Common mistakes novice learners tend to make include:

- **Broken and concatenated strokes.** The existence of multiple poly-line strokes in Chinese characters makes it unobvious to the students how they should separate the strokes. It has been observed from our user study that in most cases beginner level students would either write multiple single-line strokes in one concatenated stroke, or one poly-line stroke in multiple straight line segments.

- **Incorrect stroke ordering.** It is a general yet ambiguous rule to Western students that they should write strokes from top to bottom, and left to right. Because as the character contains more strokes, students can be confused whether they should write a vertical or horizontal line first, when such two strokes have start points close to each other. We find this especially important for those learners whose native languages are English because they often try to write the vertical stroke first in such cases, while usually it is the horizontal one that should go first.

- **Incorrect stroke directions.** Even after CSL students become used to writing the entire character as well as each stroke from top to bottom and left to right, there are cases where a stroke can be in diagonal orientation, where a general rule of stroke direction barely exists [22]. Novice students often write in wrong directions on these strokes.

The heavy existence of these three types of writing technique mistakes emphasizes the significance of real-time feedback. With the assistance of sketch recognition algorithms, we are able to first predict the students' intention on writing each stroke in one character by matching the strokes from the sample to those from the template. If the system fails to find a one-to-one match, we will be able to tell the existence of broken or concatenated strokes. If a one-on-one matching is found, the system detects if there exist wrong stroke orders or wrong stroke directions by comparing each sample stroke to the corresponding template stroke, and instructs the user on these mistakes.

After the students acquire the ability to follow the correct writing techniques of a given character, it is also important to learn to write it in a neat form. Since novice learners tend to focus on writing the characters correctly and similar to the templates, less attention has been paid to writing them with a good positioning of strokes. It has been stated that the assessment of Chinese handwriting depend on both local features (stroke level) and global features (character level) [5]. In this thesis work, we use Machine Learning techniques to find features that are important in distinguishing good from bad writing samples, and use these to automatically assess their visual quality.

In all, instead of having to wait for the human instructor's feedback on the students' entire handwriting set in one homework assignment or examination as in traditional classroom teaching methods, students can now learn a good way to improve their writing style and quickly apply them to their future learning.

1.4 Proposed system

In this thesis, we propose an educational system for learning Chinese handwriting that addresses some of the problems that traditional methods have not been able to deal with. We hope to reach higher recognition accuracy, and assess and provide feedback to students' handwriting samples in more aspects. Specifically, the goal of this work is to address the following problems:

1. **Symbol recognition:** Can we develop an efficient algorithm that reaches a reasonable accuracy on recognizing students' writing samples for our symbol set?

2. **Writing technique assessment and feedback:** Can we reach better detection rate for students' mistakes in writing techniques and provide richer feedback to them?

3. **Quality assessment:** Can we find important features that help distinguish good from bad writing quality?

The rest of this thesis reviews the literature, presents the methodology employed in our system, and analyzes evaluation results based on the data collected from user studies.

2. RELATED WORK

2.1 Sketch recognition

Sketch recognition algorithms can be classified into three categories. Gesture-based recognition methods rely on features that represent the the movement of points that form a sketch. Strokes are recorded and saved as a sequence of points that contain location and time information associated with them. Gesture-based systems focus on the process the sketch is drawn rather than what it looks like in a final form. The linear classifier proposed by Rubine et al. [23] uses 13 features to represent a sketch based on the x and y coordinates as well as the time of each point. Long et al. [24] further extended the work to include more features, for instance, density. Template matching algorithms are computationally cheaper since they need no training data. $1 [25] is an efficient algorithm for single stroke sketch recognition which is robust with size and rotation. To successfully recognize sketches drawn with multiple strokes, two algorithms built based upon $1 are proposed to encompass all possible stroke orders and directions. [26, 27]. $P [28], a template matching algorithm based on point clouds, was proposed to recognize gestures with less constraints to the sketches. Geometry-based recognition methods have less constraints to the drawing process but rather focus on the geometric features in sketches. Geometry-based sketch recognition has reached a big success in recognizing primitive shapes [29, 30, 31, 32, 33, 34, 6, 35, 36, 37]. Gladder [38], a novel sketch recognition method that combined gesture- and geometry-based techniques was proposed and outperformed either technique on its own. Vision-based recognition techniques solely rely on the visual structure of the sketches. Distances that are calculated based on the location of points are often used to measure similarities between sketches [39, 40]. These methods do not require users to draw in a pre-defined manner but recognize inputs from a vision level.

2.2 Sketch-based Intelligent Tutoring Systems

Sketch recognition have been applied and succeeded in practical applications that help students learn and teachers teach in many specific domains. Mechanix [9, 10, 11, 12] is an automated system to aid students learn introductory engineering. Maestoso [41, 15] was proposed to teach novice learners music theory through sketch practicing on quizzed music structures. In the domain of drawing, iCanDraw [42] is an efficient system to assist users in drawing human faces. EasySketch [43, 44, 45] and EasySketch2 [46] aim at developing children's self-regulating skills through sketching. Persketchtivity [13, 47, 48, 20] is a system proposed for engineering design, which is not only able to effectively recognize handdrawn inputs, but also assess their quality and provide real-time feedback to students. Sketchography [49] is a sketch-based educational system for teaching river drawing in Geography classes which used geometry-based recognition to recognize and extract features from sample drawings, and Machine Learning techniques to automatically grade students' drawings. Flow2Code [50] is an application that uses sketch recognition to help Computer Science students understand and express ideas of computer programs through drawing flowcharts. Sketch recognition and assessment have also been applied to the domains of Asian language education, such as Japanese Kanji [19], Mandarin phonetic symbols [16, 17, 51], and Chinese characters [52, 53, 54].

2.3 Computer-aided Chinese Language Education

Our proposed sketch-based educational system for Chinese handwriting composes three parts, namely, handwriting recognition, technique assessment, and visual quality assessment. Handwriting recognition has long been researched using deep neural networks [3]. The work by LeCun et al. has revolutionized handwriting recognition by back-propagation networks [55, 56] and can be extended to off-line Chinese character recognition [57, 58]. Online recognition rely on the trajectory of points, which is more relevant to our research. Gesture- and vision-based recognition methods [39, 28, 36] are independent from writing techniques so they can be very helpful in recognizing beginner level CSL learners' writing samples. They have also been applied to similar educational systems for Asian handwriting recognition and reached good results [19, 16, 17]. Observing that Chinese characters are typically formed with poly-line shapes on 8 directions [59], orientational features such as Gabor features can be very powerful in handwriting recognition [60, 61]. In addition, corner finding algorithms [30, 62, 63] are very useful for extracting features from writing samples. Machine Learning and fuzzy techniques have been employed extensively on assessing the quality of handwritings [64, 5], but they rely on pixel data that reflects the overall structure of symbols while little attention has been paid stroke-level features. Methods for detecting and correct students' writing techniques has been proposed based on stroke level features [19, 17], however, these methods have limited ability to instruct students when multiple types of technique mistakes coexist in one writing sample.

3. RECOGNITION OF HANDWRITTEN CHINESE CHARACTERS

The recognition accuracy is crucial to the system because it directly affects the efficiency of the entire education process and user experience. An efficient recognition algorithm for our system should be able to accurately tell which character the CSL learner tries to draw. This goal requires our recognition system 1) rely on features that distinguish among template characters and 2) have a reasonable tolerance for novice students' common mistakes. Previous work has shown template matching methods have reached reasonably good results on East Asian characters [17]. In this work, we propose a template matching algorithm for handwriting recognition based on the projections of writing samples on horizontal, vertical, left diagonal, and right diagonal orientation.

3.1 Preprocessing

CSL learners in the early stage of learning handwriting try to mimic the template images in their memory but pay less attention to correct techniques or a good visual structure. This brings obstacles for stroke-based template matchers to achieve consistent results for a character written with different quality issues. Preprocessing steps solved part of the problems.

3.1.1 Resampling

The goal of resampling is to turn each handwriting sample and template into point clouds, so that stroke count, order, and directions will be irrelevant to the recognition result. Moreover, since the original stroke is produced by sampling the trace of students' drawing movements at a fixed rate that is determined by the hardware and software, strokes with similar shapes but different speeds can have completely different distributions of points, which make them not comparable. We resample each sample and template to the same amount of points, where the distances between neighboring points are always the same. The specific steps for resampling a multi-stroke sketch to point clouds can be found in [28]. Resampling is also conducted to the timestamps.

3.1.2 Translation

Most template matching algorithms calculates the similarity between a sample and a template based on Euclidean distances. The Euclidean distance between point (x_1, y_1) and (x_2, y_2) is calculated as the following:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}. \tag{3.1}$$

Due to the possible difference in the sizes of the writing frames of the template and sample, the Euclidean distance between two points that have similar locations relative to the bounding box the character can be large. In order to resolve this problem, we first set the centroids of each sample and template to be the original point $(0, 0)$, and then set the all the points relevant to the centroids.

3.1.3 Scaling

While students may try to draw a visually similar symbol to the template based on their memory, they often neglect the fact that a well-written character should have a good shape, which means the height to width ratio of the character plays an important role. A visually similar but badly shaped character are can be hard to recognize due to the stretching on either horizontal or vertical orientation. In order to resolve this, we rescale each point in the sample and template so that the sizes and ratios of the bounding boxes are the same.

3.2 Proposed template matching algorithm

3.2.1 The projection feature

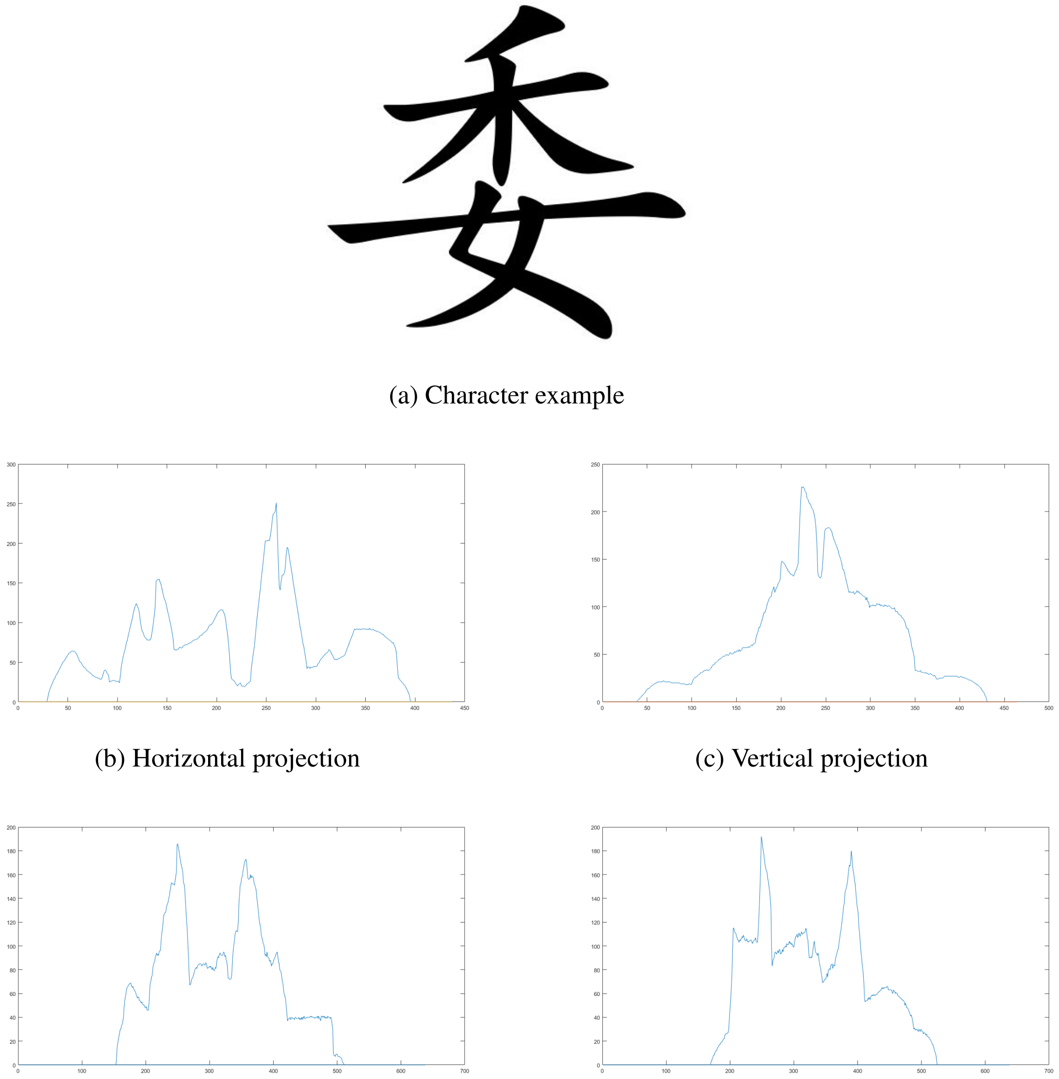
To accurately recognize novice students' writing samples with a good tolerance to both technique and visual mistakes, we propose an efficient template matching method motivated by the following observations:

1. Chinese characters can be seen as formed as multiple straight line segments, where each line segment can be roughly seen as positioned on one of the four orientations: horizontal, vertical, left diagonal, and right diagonal [59, 22].

2. CSL learners usually know which orientation each straight line segment should be positioned on, and know the relative sequence of each line segment on each orientation, but can place these segments with little consideration on how to balance them in the entire structure.

The first observation implies that projections on the four orientations provide useful information about the lengths and locations of stroke segments on each orientation. Fig. 3.1 shows a Chinese character with two horizontal stroke, one vertical stroke, and several diagonal strokes. The projection on one particular orientation contains peaks and valleys that indicate the locations and length of strokes on this orientation. These features can be especially helpful for recognizing novice users' handwriting samples because they typically write each character stroke by stroke so that the relative locations and orientations of each stroke are preserved in their written characters.

The second observation has inspired us to develop an efficient recognition algorithm that relies on the relative stroke locations on each orientation, and is invariant with the internal distribution of a character. Fig. 3.2 shows an example of a handwriting sample of the character "five" that has badly positioned strokes. In this example, the middle horizon-



(a) Character example

(b) Horizontal projection

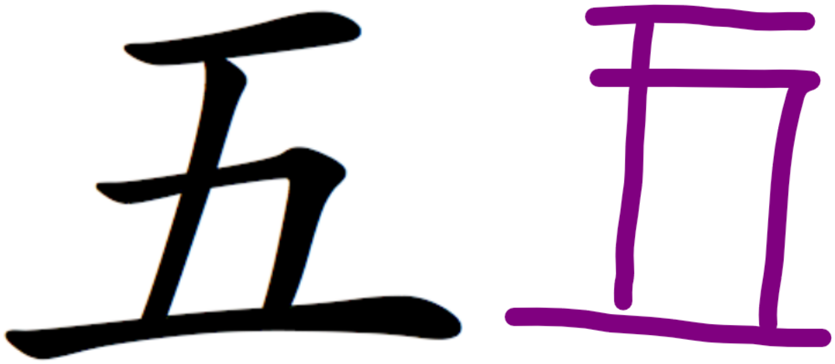
(c) Vertical projection

(d) Left diagonal projection

(e) Right diagonal projection

Figure 3.1: Projections on the four most important orientations of a Chinese character

tal stroke segment is too close to the top of the bounding box while the correct location should be around the middle. And the vertical stroke segment on the right is also too far from the middle. Euclidean distance based template matching algorithms [39, 28] will not work well on these handwriting samples because 1) The bad distribution of ink can introduce errors in the point matching step [28] and 2) Even if a correct stroke correspondence



(a) Template of the character "five"

(b) An example of a handwritten "five" with bad stroke positionings

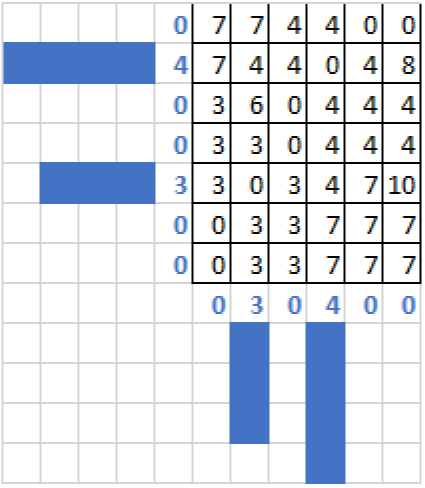
Figure 3.2: A comparison of the template and a sample with bad stroke positionings

is found, the distances calculated between points or point clouds can be very large and will lead to a low confidence in the recognition result [25, 26, 27].

3.2.2 Dynamic Time Warping

It can be observed from Fig. 3.1 and Fig. 3.2 that in a character where strokes are not well positioned, the relative ordering information of each peak in the projection arrays are still preserved, but with valleys of different sizes between them. Dynamic Time Warping (DTW) [65] is an efficient algorithm to match two sequences with similar patterns but different lengths or paces. The dynamic programming nature of this algorithm makes it possible to find a perfect match between such two sequences. As illustrated in Fig. 3.3, a perfect match can be found in two time series with different paces but contain similar patterns. Two sequences $0, 3, 0, 4, 0, 0$ and $0, 0, 3, 0, 0, 4, 0$ each have two peaks with heights 3 and 4 with different indexes and different intervals in between. However, a perfect matching can still be found using this dynamic programming algorithm.

The goal for DTW is to find a warping path between a sample sequence S and a



0	7	7	4	4	0	0
4	7	4	4	0	4	8
0	3	6	0	4	4	4
0	3	3	0	4	4	4
3	3	0	3	4	7	10
0	0	3	3	7	7	7
0	0	3	3	7	7	7
	0	3	0	4	0	0

Figure 3.3: An illustration of the DTW algorithm

template sequence T:

$$S = s_1, s_2, ..., s_i, ..., s_m \tag{3.2}$$

$$T = t_1, t_2, ..., t_j, ..., t_n \tag{3.3}$$

A $m * n$ matrix M can be formed where $M\left[i, j\right]$ denote the cost $\gamma\left(i, j\right)$ when the i_th element in S corresponds to the j_th element in T. DTW uses dynamic programming and local greedy matching to find a path that goes from $(0, 0)$ to $(m-1, n-1)$ that minimizes the cumulative cost. Applying the endpoint constraint [66], the cumulative cost is built using dynamic programming:

$$\gamma\left(i, j\right) = \delta\left(i, j\right) + \min\left(\gamma\left(i-1, j\right), \gamma\left(i, j\right), \gamma\left(i, j-1\right)\right) \tag{3.4}$$

where

$$\delta\left(i, j\right) = \mid s_i - t_j \mid . \tag{3.5}$$

3.3 Proposed recognition method

As stated previously, if a student tries to write a particular character, the information of relative positions of strokes on each orientation is preserved in four projection arrays. Using DTW to match projections from the sample to template, the following constraints are applied:

1. **Endpoint constraint:** The first and last point in the projection arrays must correspond to each other.

2. **Monotonicity constraint:** The elements corresponded to each other must be ordered with respect to the order of their original occurrences in the arrays.

3. **Warping window size constraint:** The warping window size is set to 3 for our algorithm.

To recognize a handwriting sample, we go through the following steps for each template to find the final template matching result:

1. Translate both the sample and template sketches with respect to the centroids of the sketches.

2. Transform both sketches to pixel arrays.

3. Scale both pixel arrays to $400 * 400$ arrays.

4. Calculate the projections of both pixel arrays on horizontal, vertical, left diagonal, and right diagonal orientations.

5. Calculate the DTW cumulative distance results on the projection of each orientation and sum them up.

6. The template matcher returns the template list sorted in ascending order by the summed DTW costs.

7. The first item in the list of templates returned by the template matcher will be the final recognition result.

4. TECHNIQUE ERROR DETECTION AND FEEDBACK GENERATION

While it is not a hard task for beginner level CSL learners to write characters that are visually similar to the templates, they can make a large amount of mistakes in writing techniques. In order to better instruct learners, it is important for our system to accurately detect their mistakes and offer informational feedback. This chapter covers the three types of technique mistakes we are aiming at, the methods we use to detect them, and the feedback we offer for each type.

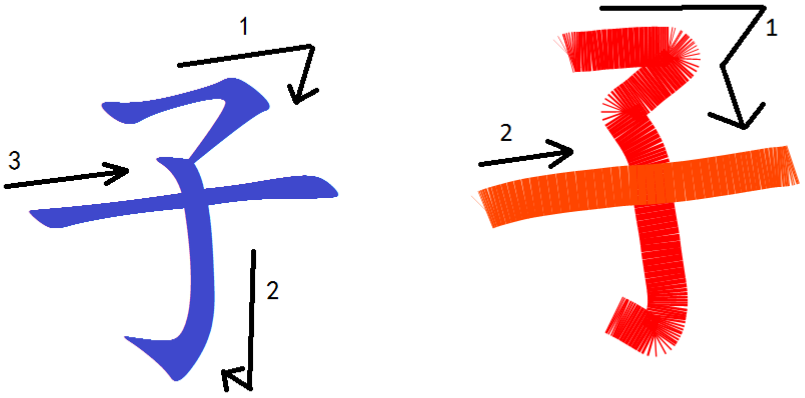
4.1 Stroke count

4.1.1 Overview

As stated previously, Chinese characters are formed with a set of poly-line strokes connecting or intersecting with each other. A complex combination of poly-line strokes can cause difficulties for beginners to tell how to separate these strokes. This results in the heavy existence of four types mistakes related to stroke count:

- **Concatenating strokes:** When two strokes are connecting end to end but should be written separately, students often write them in one stroke.
- **Broken strokes:** When one stroke is formed with multiple line segments, students often break it into several strokes at corners.
- **Missing strokes:** Students often forget to write some short strokes in a multi-stroke character.
- **Extra strokes:** Strokes in the sample that do not correspond to any stroke in the template sometimes exist in characters that are visually similar to other ones.

Fig. 4.1(a) shows a template character that contains three strokes. The first one goes horizontal from left to right, and then turns left diagonal, the second one goes vertical from top to bottom, then turn right diagonal, and the third one is a straight line that goes from left to right. The first and second strokes are both poly-line strokes and they connect with each other end to end. This type of stroke relationship is where students are likely to make mistakes of concatenating strokes. Fig. 4.1(b) shows a handwriting sample with this type of mistake, where the first and second strokes in the template are written in one concantenated stroke. Similarly, students can randomly break a poly-line stroke up at one of its corners, making mistakes of broken strokes.



(a) Template of the character "son"

(b) A handwritten "son" with concatenated strokes

Figure 4.1: An example of mistakenly concatenated strokes in a character written by one of our users

4.1.2 Finding stroke correspondences

In order to detect the specific mistakes in strokes from students' input and give interactive feedback, an efficient algorithm that matches strokes from template and sample is crucial. We propose algorithms to find stroke correspondences for the following three different conditions.

4.1.2.1 Sample has same amount of strokes as template

In this case, we believe student's handwriting has correct stroke count. So the task is to find the one-to-one correspondence from each sample stroke to each template stroke. This stroke correspondence finding problem can be modeled as an assignment problem that has been solved in graph theory [67]. This task can be translated into constructing a bipartite graph [68], where the strokes from the sample and template each form a vertex set with size n, and edges connect the two vertex sets so that their summed weight is minimized. The optimal solution can be found using the Hungarian algorithm [69]. The model has also been applied in finding the point correspondence in two different gestures, and it has been proved that a greedy approach can reach close to optimal results in finding point correspondence [28].

We assume that a greedy algorithm for finding matching strokes can reach close to optimal results due to the few number of strokes in each character. And we use Hausdorff distance to weight the edges. The algorithm works as follows: For each stroke in the sample s_i, find an unmatched stroke t_j in the template so that the Hausdorff distance between s_i and t_j is minimized. This algorithm outputs a one-to-one correspondence from sample strokes to template strokes.

4.1.2.2 Sample has fewer strokes than template

When the sample character contains less strokes than the template, it is indicated that either concatenated strokes exist in the sample or some template strokes are missing. We proposal a greedy algorithm that relies on Hausdorff distance and directed Hausdorff distance to find the stroke correspondence.

As introduced in [39], the Hausdorff distance between two point sets A and B is defined as:

$$H(A, B) = \max(h(A, B), h(B, A)) \tag{4.1}$$

where

$$h(A, B) = \max_{a \in A}(\min_{b \in B} ||a - b||) \tag{4.2}$$

is defined as the directed Hausdorff distance from A to B. The directed Hausdorff distance denotes the maximum distance from each point in A to its closest point in B. Note that in most cases $h(A, B) \neq h(B, A)$. It can be observed from Eq. 4.2 that directed Hausdorff distance measures how A is visually similar to a subset of B. The value $h(A, B)$ can have be very small when A approximately overlaps a subset of B, while in this case $h(B, A)$ can be large due to outliers. As a result, Hausdorff distance can be very sensitive to outliers. Observing these characteristics, we propose a greedy algorithm as described in Alg. **??** that finds a one to many stroke correspondence from sample to template by iterating through each sample stroke and finding the best set of template strokes that fit it.

4.1.2.3 Sample has more strokes than template

When sample has more strokes than template, there must be either broken stroke or extra strokes in the sample. For this condition, we can apply Alg. **??** to find a one-to-many correspondence from template strokes to sample strokes, which is equivalent to a many-to-one correspondence from sample strokes to template strokes.

4.1.3 Feedback

Apart from giving binary feedback to indicate whether the student has written correct stroke counts or not, the system also provides specific interactive instructions to users. Fig. 4.2 shows an example of our feedback to handwritings with mistakes on stroke count. We highlight broken strokes with the same color to indicate they should be written with one strokes, and we highlight concatenated strokes to indicate they should be written with

Algorithm 1 Algorithm for finding a one to many stroke correspondence from sample to template

```
function FindCorrespondence($Sample, Template$)
    $Result = \emptyset$
    for $StrokeS$ in $Sample$ do
        Sort($Template$) by DirectedHausdorffDistance($StrokeT, StrokeS$)
        $CurrentList = \emptyset$
        $CurrentDistance = \infty$
        for $StrokeT$ in $Template$ do
            $NewList = CurrentList + StrokeT$
            $NewDistance =$ HausdorffDistance($StrokeS, NewList$)
            if $NewDistance \leq CurrentDistance$ then
                $CurrentDistance = NewDistance$
                Add $StrokeT$ to $CurrentList$
                Remove $StrokeT$ from $Template$
            end if
        end for
        Add $CurrentList$ to $Result$
    end for
    return $result$
end function
```

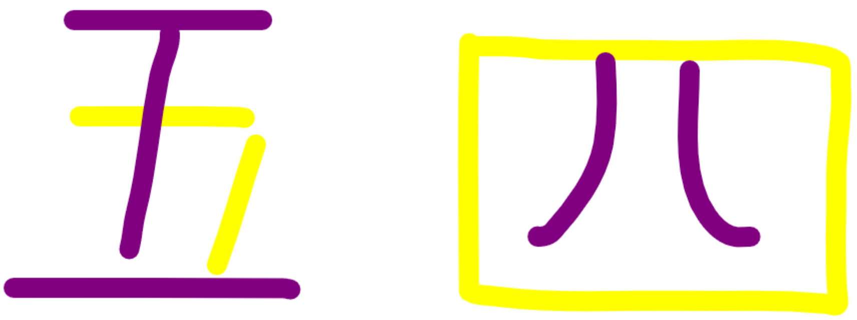
multiple strokes.

4.2 Stroke order

4.2.1 Stroke order judgment

For stroke order, we also give both binary judgment and specific feedback to the user's handwriting. Stroke order is judged as correct if each stroke in the sample is written in the same chronological order as in the template.

For characters that have correct stroke counts, the system look at the one to one stroke correspondence array and judge stroke order as correct if each element in the array is equal to its index, and incorrect otherwise. For handwriting samples that have more strokes than template, the system judges stroke order as correct only if both correctly written strokes



(a) Feedback for broken strokes

(b) Feedback for concatenated strokes

Figure 4.2: Feedback to broken and concatenated strokes. In the example on the left, the character "five" should be written with 4 strokes, the second being the middle stroke with one horizontal and one vertical stroke segment. In the example on the right, the rectangle that surrounds the character "four" should be written with 3 strokes.

and broken strokes are written in the correct chronological order. If a template stroke is detected as written in multiple broken strokes, we take each sample stroke s_i that it corresponds to, and find the matching part in the template by finding point P_{si} and P_{ei} on the template stroke, which has the minimum distance from the start and end points to that sample stroke, respectively. The broken strokes are in the correct order only if for each i, P_{si} has in chronologically earlier than $P_{s(i+1)}$ in the template. In addition, the system also requires that for each template stroke t_j, the each one of the of sample strokes that corresponds to must all appear earlier than the all sample strokes that corresponds to $t_{(j+1)}$ so as to be considered to have correct stroke order. The stroke order judgment for characters that have fewer strokes than template works similarly.

4.2.2 Feedback

To instruct users, we highlight the strokes in the sample that are not in the correct chronological order, so as to let users know which strokes are in the wrong order. Fig. **??** shows an example of the system's feedback on a handwriting sample with incorrect stroke order.

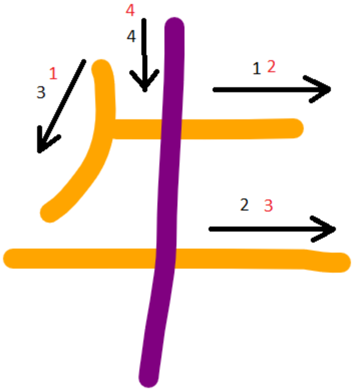


Figure 4.3: An example of the system's feedback on wrong stroke orders. The black number next to each stroke is the actual ordering of it, and the red numbers are the correct ordering.

4.3 Stroke direction

Stroke direction is defined as the chronological order of the start point and end point of each stroke. To judge if each stroke is written in the correct direction, the system calculates a vector $\vec{v_s}$ from the start to the end point, and calculates a vector $\vec{v_t}$ based on either a complete or a part of a template stroke that it corresponds to, and calculates $\cos(\vec{v_s}, \vec{v_t})$ as the cosine between these two vectors. A stroke is considered to have the correct direction only of the cosine value is positive. For each sample stroke that has a wrong direction, the system shows a green dot moving in the correct direction.

5. QUALITY ASSESSMENT

Assessing the overall quality of a handwritten Chinese character is a difficult task. The aim of this part of our work is to find out important features that reflect how a user masters Chinese handwriting. We look at both global and local features, that reflect not only how the sample is similar to the template as a whole, but also the internal balance of the character. We also take into account speed, which is significant to indicate how the user is familiar with handwriting, or how the user is serious in pursuing an accurate writing. Following is the list of features we extracted from each writing sample.

5.1 Bounding box ratio

Good handwritings always have proper shapes. A badly sized character can be either too thin or too flat. This feature, calculated as Eq. 5.1, measures the difference in height width ratio of a given character of a sample from that of the template, where w_s and h_s denote the width and height of the sample, and w_t and h_t denote the width and height of the template, respectively.

$$F_R = |w_s/h_s - w_t/h_t| \tag{5.1}$$

5.2 Centroid location (x-axis)

The location of the centroid in a handwriting sample reflect how ink is globally distributed within the character. It is commonly observed in badly written samples have unbalanced distribution, making the centroid located far away from the center of the bounding box [64]. Eq. 5.2 shows F_{CX}, the difference in the distance from centroid to center of a sample from that of the template.

$$F_{CX} = \left| \frac{x_{sampleCentroid} - x_{sampleCenter}}{w_s} - \frac{x_{templateCentroid} - x_{templateCenter}}{w_t} \right| \tag{5.2}$$

5.3 Centroid location (y-axis)

Similar to F_{CX}, F_{CY} reflects the difference of the vertical location of the centroid in the bounding box of a sample from that of the template.

$$F_{CY} = \left| \frac{y_{sampleCentroid} - y_{sampleCenter}}{h_s} - \frac{y_{templateCentroid} - y_{templateCenter}}{h_t} \right| \tag{5.3}$$

5.4 Hausdorff distance

Hausdorff distance is a metric that evaluates how the sample and the template overlap with each other. F_H is the Hausdorff distance between sample and template.

$$F_H = H(A, B) = \max(h(A, B), h(B, A)) \tag{5.4}$$

where

$$h(A, B) = \max_{a \in A}(\min_{b \in B} ||a - b||) \tag{5.5}$$

5.5 Tanimoto similarity coefficient

Tanimoto coefficient [39] measures the similarity between two binary images by calculating the overlap of their black and white pixels. Tanimoto coefficient between image A and image B is calculated as

$$T(A, B) = \frac{n_{ab}}{n_a + n_b - n_{ab}}, \tag{5.6}$$

where n_a and n_b are the numbers of black pixels in image A and image B, respectively, and n_{ab} is the number of overlapping black pixels. For images that the majority of pixels are white, the Tanimoto coefficient complement can be of more importance in measuring their similarity. The Tanimoto coefficient complement is calculated as

$$T^C(A, B) = \frac{n_{00}}{n_a + n_b - 2n_{ab} + n_{00}}, \tag{5.7}$$

where n_{00} is the number of overlapping white pixels. $T(A, B)$ and $T^C(A, B)$ are combined to form the Tanimoto similarity coefficient, calculated as

$$T_{SC}(A, B) = \alpha T(A, B) + (1 - \alpha)T^C(A, B). \tag{5.8}$$

α is dependent on the number of black pixels in each image and is calculated as

$$\alpha = 0.75 - 0.25 \left(\frac{n_a + n_b}{2n} \right), \tag{5.9}$$

where n is the total number of pixels in both images.

5.6 Yule coefficient

Similar to Tanimoto coefficient, Yule coefficient measures the similarity between two binary images based on the overlapping of black and white pixels. Yule coefficient can be calculated as 5.10

$$Y(A, B) = \frac{n_{ab}n_{00} - (n_a - n_{ab})(n_b - n_{ab})}{n_{ab}n_{00} + (n_a - n_{ab})(n_b - n_{ab})} \tag{5.10}$$

5.7 Stroke length distribution

We observed from the user studies that novice CSL learners tend to draw strokes with extreme lengths, while a good handwriting requires a good ratio of stroke lengths. For a character with n strokes, its length distribution can be represented as a normalized sequence $Dist = (d_1, d_2, ..., d_n)$ where d_i is proportional to the length of the ith stroke to the total length of every stroke in the character. F_{LD} measures the similarity of length distribution of sample and template using Bhattacharyya distance [70].

$$F_{LD} = D_B(Dist_s, Dist_t) = -\ln(BC(Dist_s, Dist_t)) \tag{5.11}$$

where

$$BC(p, q) = \sum_{i=1}^{n} \sqrt{p_i q_i}. \tag{5.12}$$

5.8 Average stroke orientation similarity

The orientations of stroke play an important role in the visual perception of Chinese handwriting [71]. For each stroke, we use the angle from its start point to end point to approximate its orientation [23]. For each stroke from the sample, we measure its orientation similarity from the template by calculating the cosine of the angle it forms with its corresponding stroke from the template. F_{OA} is calculated as the average of these cosine values weighted by the lengths of strokes.

$$F_{OA} = \overline{\cos{(s_i, t_i)}} \tag{5.13}$$

5.9 Minimum stroke orientation similarity

This feature denotes the minimum orientation similarity of the sample stroke to its corresponding template stroke.

$$F_{OM} = \min\cos{(s_i, t_i)} \tag{5.14}$$

5.10 Average speed

The trade-off between accuracy and speed in sketching has long been discussed in the literature [72, 73]. We observed that expert users in our study needs a significantly longer time to provide good writings than novice. The average speed of a stroke is calculated as the length divided by the time spent on writing it. In order to make this feature invariant with users' pauses between strokes, F_{AS} is calculated as the average of speed of each stroke weighted by their lengths.

$$F_{AS} = \overline{\left(\frac{TotalPathLength}{TotalTime}\right)} \tag{5.15}$$

5.11 Speed fluidity

An experienced writer tend to write at a more steady pace. We use the ratio of minimum and maximum speeds in a handwriting to represent the speed fluidity of each stroke. F_{SF} is calculated as the average of speed fluidity of each stroke weighted by their lengths.

$$F_{SF} = \overline{\left(\frac{MinSpeed}{MaxSpeed}\right)} \tag{5.16}$$

5.12 Horizontal projection difference

The projection feature has been extensively used for assessing the visual quality of Chinese handwritings [5, 64]. The difference between two projection arrays P and Q are defined as $D_P(P, Q) = (\sum_{x=1}^{n} |P(x) - Q(x)|)/(\sum_{x=1}^{n} |P(x) + Q(x)|)$. F_{HP} measures the difference in the horizontal projections between sample and template.

$$F_{HP} = D_P(SampleHorizProjection, TemplateHorizProjection) \tag{5.17}$$

5.13 Vertical projection difference

Similar to F_{HP}, F_{VP} measures the difference in the vertical projections between sample and template.

$$F_{VP} = D_P(SampleVertProjection, TemplateVertProjection) \tag{5.18}$$

6. RESULTS AND EVALUATION

6.1 User study

To evaluate the performance of our system, we collected data from 3 groups of students from Texas A&M University. At the beginning of each user study, we presented the information sheet which has been approved by the institutional review board (IRB) of Texas A&M University and started the experiment only after getting the user's consent. The first group of users contains 11 expert students whose native languages are Chinese and they all have been using Chinese as their primary language since childhood. These students were asked to write each of the 27 characters 3 times with the best of their abilities, and these data are used to evaluate our symbol recognition algorithm, as well as labeled data for training and testing our machine learning algorithm that classifies between good and bad handwriting samples. The second group contains 9 expert students who have more than 10 years of experience writing Chinese characters, and they were asked to provide casual style handwriting samples. The essential step of the writing technique assessment part of our system is the accuracy for finding stroke correspondences from sample to template. We hope to see robustness of our stroke matching algorithm through testing on casual data provided by Chinese users which are typically written in a cursive way and thus contain many concatenated strokes. So we use data from group 2 to evaluate our stroke matching sysmte and compare the results with similar East Asian language educational systems [17]. The third group contains 10 novice students who have none or little experience in Chinese handwriting. These users were asked to draw each character 3 times to mimic the template presented to them, and use the feedback provided by our system to improve their writing techniques. Data from group 3 will be used to evaluate our recognition algorithms on symbols as well as writing techniques, and further used as labeled data for our machine learning algorithm that classifies between good and bad handwritings. After collecting data from novice users, we also asked them to share their thoughts and give feedback on our system through a survey. Table 6.1, table 6.2 table 6.3 show the demographic information of the three groups of users respectively.

We evaluate the performance of our system from four aspects. Firstly, we test the proposed handwriting Chinese character recognition algorithm against state-of-the-art template matching algorithms as well as previous East Asian language educational systems [28, 17]. This test is done using data from group 1 and group 3. Secondly, we test our algorithms for assessing students' writing techniques. In this part, we evaluate the stroke matching algorithm on data from both group 2 and group 3, and then evaluate the writing technique judgments on only data from group 3. Thirdly, we evaluate the significance of features for classifying between good and bad handwritings using data from group 1 and group 3. Lastly, we evaluate the usability of our system based on the technique mistakes novice users make overtime and their feedback on our system using data from group 3.

ID	Fluency in Chinese	Native language	Gender	Number of characters
1	Expert	Chinese	Male	85
2	Expert	Chinese	Female	94
3	Expert	Chinese	Male	77
4	Expert	Chinese	Male	84
5	Expert	Chinese	Male	97
6	Expert	Chinese	Male	76
7	Expert	Chinese	Female	85
8	Expert	Chinese	Male	83
9	Expert	Chinese	Female	79
10	Expert	Chinese	Female	71
11	Expert	Chinese	Female	90

Table 6.1: Information of users of group 1

ID	Fluency in Chinese	Native language	Gender	Number of characters
12	Expert	Chinese	Female	55
13	Expert	Chinese	Male	55
14	Expert	Chinese	Male	56
15	Expert	Chinese	Female	175
16	Expert	Chinese	Male	56
17	Expert	Chinese	Female	56
18	Expert	Chinese	Female	52
19	Expert	Chinese	Female	66
20	Expert	Chinese	Male	161

Table 6.2: Information of users of group 2

ID	Fluency in Chinese	Native language	Gender	Number of characters
21	Novice	English	Female	94
22	Novice	English	Male	86
23	Noivce	Japanese	Male	70
24	Novice	Korean	Female	94
25	Novice	English	Male	114
26	Novice	English	Male	105
27	Novice	English	Male	86
28	Novice	English	Male	93
29	Novice	Hindi	Male	105
30	Novice	English	Female	114

Table 6.3: Information of users of group 3

6.2 Handwriting recognition

In this section, we compare the recognition rate of our proposed system with the following methods/systems:

1. **$P:** [28]: A state-of-the-art template matching algorithm for multi-stroke sketches based on point clouds. In order to minimize the recognition errors caused by various writing techniques such as stroke count, order, and directions, the $P algorithm treats each sketch as a cloud of points. To recognize a given handdrawn input sample, both

the template and the sample are normalized to have the same amount of points and to be bounded by a square of the same size. A greedy algorithm is applied to match each point from the sample to the template. The final distance between the sample and the template is calculated as the summ of distances of each point pair weighted by the significance of the point.

2. **BopoNoto** [17]: An ITS used for teaching Chinese Zhuyin symbols that have reached reasonably high recognition rate. It is important to note that though the domain Zhuyin have different symbols than Chinese characters, Zhuyin symbols have been invented using strokes that are used in Chinese characters and the writing of Zhuyin symbols follow the same manner as Chinese characters. The BopoNoto system uses a two-part template matching algorithm for recognizing Zhuyin symbols. In the first stage, the system calculates the Hausdorff similarity [15] between the sample and each template, and returns a sorted list of symbols. In the second stage, the system calculates the point coverage ratio [17] of templates that rank top 10% in the list and take the template with the highest point coverage ratio as the recognition result. The second step has been proved crucial for reducing recognition error caused by samples that are visually similar to a subset of points of a particular template. This algorithm has successfully recognized the entire Zhuyin symbol set [17].

We compare the recognition rates, and rankings of correct answer in the lists returned by each template matcher. All three algorithms are run on the datasets from both expert users (group 1) and novice users (group 3).

6.2.1 Results

6.2.1.1 Recognition rate

Recognition rate is defined as the number of times that the correct prediction ranks first in the list returned by the template matcher divided by the total number of writing samples. As can be observed from Fig. 6.1, our overall recognition rate reaches 91%, which is significantly higher than BopoNoto's 87% and BopoNoto's 81%. Fig. 6.2 shows that our proposed method reached 98% and 85% recognition rates for expert and novice users, respectively, which is better than BopoNoto's 94% and 79%, and $P's 89% and 73%.

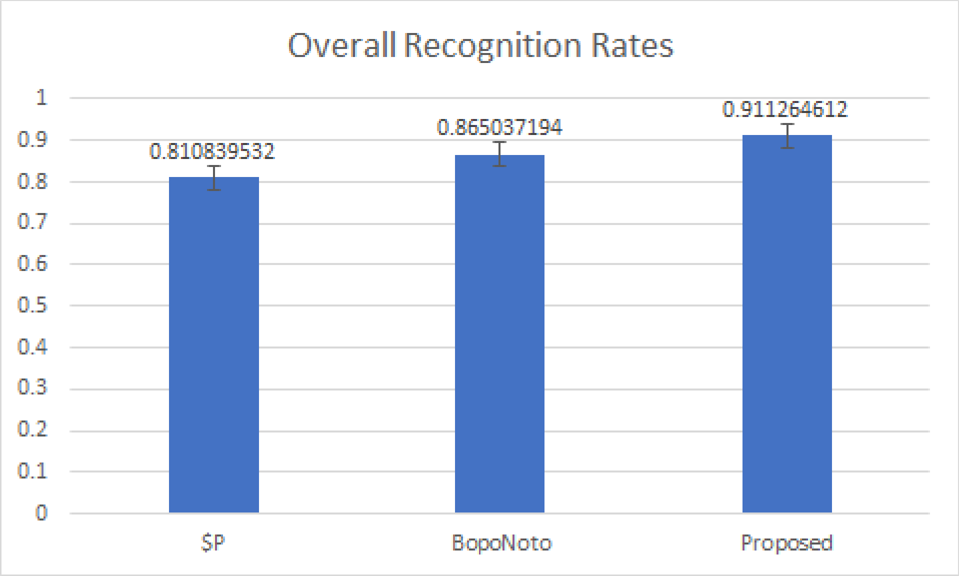


Figure 6.1: Overall recognition rates of the three algorithms on the entire data set

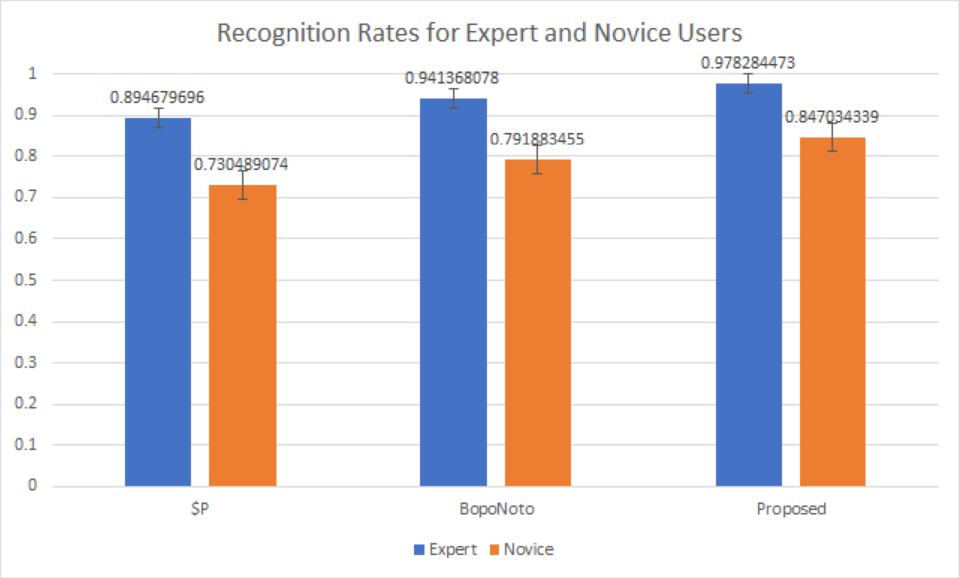


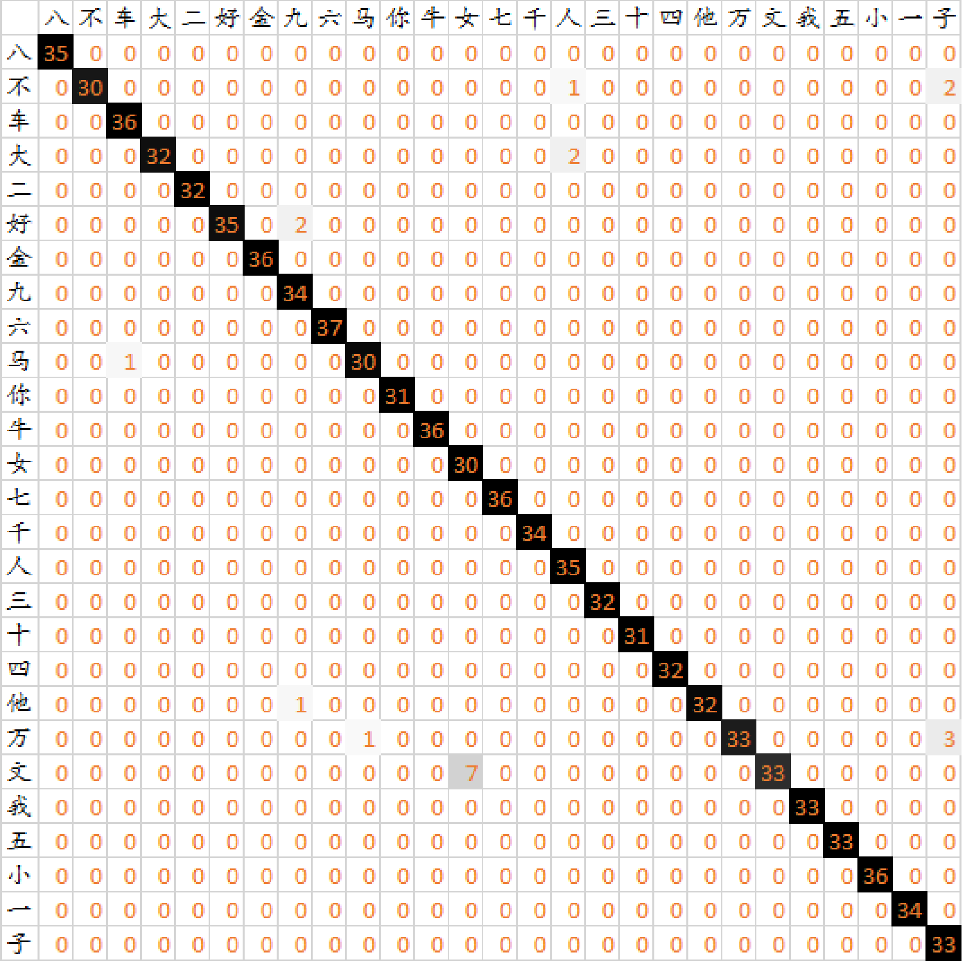
Figure 6.2: Recognition rates of the three algorithms on expert and novice datasets

6.2.1.2 *Confusion matrices*

Fig. 6.3 and Fig. 6.4 show the confusion matrices of our proposed classifier on the 27 symbols. It can be observed that expert data are almost always accurately classified. The most miss classified character is "text", which is often classified as a very similar character "female". However our algorithm often gets confused on characters with more complex strokes like "he" and "you", whose projections have less obvious patterns.

6.2.1.3 *Rank of matched template in the result list*

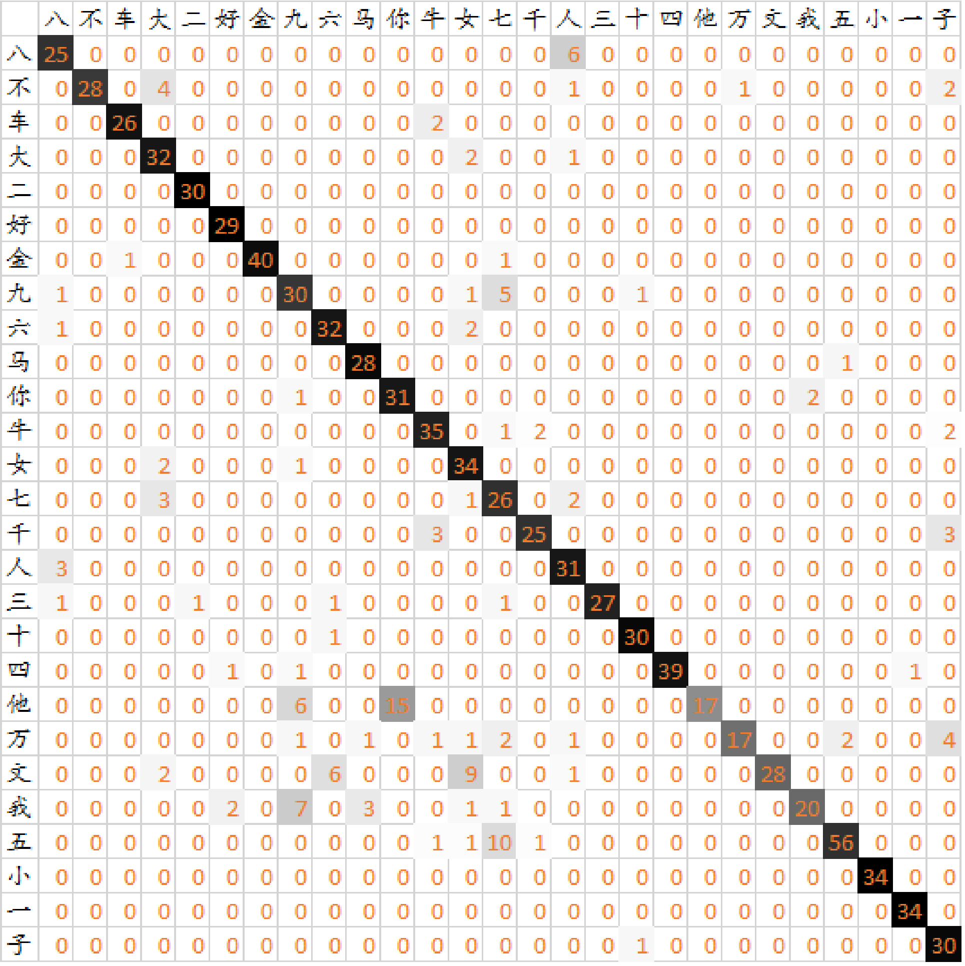
As pointed out in the previous subsection, each template matcher returns a list of labels their similarity to the sample. So it is a good indication when the correct templates show in the top several indexes of the results list. In the implementation of our system, we judge each handwriting as correct as long as the answer ranks top 3 in the recognition results list. Fig. 6.5 shows our proposed template matching algorithm reaches 98% in ranking the correct templates top 3, beating BopoNoto's 89% and $P's 93%. Fig. 6.6 shows our



	八	不	车	大	二	好	金	九	六	马	你	牛	女	七	千	人	三	十	四	他	万	文	我	五	小	一	子
八	35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
不	0	30	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	2
车	0	0	36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
大	0	0	0	32	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0
二	0	0	0	0	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
好	0	0	0	0	0	35	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
金	0	0	0	0	0	0	36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
九	0	0	0	0	0	0	0	34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
六	0	0	0	0	0	0	0	0	37	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
马	0	0	1	0	0	0	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
你	0	0	0	0	0	0	0	0	0	0	31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
牛	0	0	0	0	0	0	0	0	0	0	0	36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
女	0	0	0	0	0	0	0	0	0	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0
七	0	0	0	0	0	0	0	0	0	0	0	0	0	36	0	0	0	0	0	0	0	0	0	0	0	0	0
千	0	0	0	0	0	0	0	0	0	0	0	0	0	0	34	0	0	0	0	0	0	0	0	0	0	0	0
人	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	35	0	0	0	0	0	0	0	0	0	0	0
三	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	32	0	0	0	0	0	0	0	0	0	0
十	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	31	0	0	0	0	0	0	0	0	0
四	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	32	0	0	0	0	0	0	0	0
他	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	32	0	0	0	0	0	0	0
万	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	33	0	0	0	0	0	3
文	0	0	0	0	0	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	0	33	0	0	0	0	0
我	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	33	0	0	0	0
五	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	33	0	0	0
小	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	36	0	0
一	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	34	0
子	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	33

Figure 6.3: The confusion matrix of our proposed classifier on expert data

system has reached 100% and 95% in this metric for expert and novice data, respectively, beating BopoNoto's 96% and 83%, and $P's 97% and 90%. Fig. 6.7 shows the correct answers are almost always ranked top by our proposed template matching algorithm, with an average index of 0.023 for experts, and 0.352 for novice, which is significantly better than BopoNoto's 0.35 and 1.287, and $P's 0.318 and 1.065. It can also be indicated that our proposed method is more robust to the level of students' handwriting ability, while the average ranking of BopoNoto varies largely with user groups. Fig. 6.8 shows the number of occurrences that the correct classification result ranks at each index int the result lists



	八	不	车	大	二	好	金	九	六	马	你	牛	女	七	千	人	三	十	四	他	万	文	我	五	小	一	子
八	25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0
不	0	28	0	4	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	2
车	0	0	26	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
大	0	0	0	32	0	0	0	0	0	0	0	0	2	0	0	1	0	0	0	0	0	0	0	0	0	0	0
二	0	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
好	0	0	0	0	0	29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
金	0	0	1	0	0	0	40	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
九	1	0	0	0	0	0	0	30	0	0	0	0	1	5	0	0	0	1	0	0	0	0	0	0	0	0	0
六	1	0	0	0	0	0	0	0	32	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
马	0	0	0	0	0	0	0	0	0	28	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
你	0	0	0	0	0	0	0	1	0	0	31	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0
牛	0	0	0	0	0	0	0	0	0	0	0	35	0	1	2	0	0	0	0	0	0	0	0	0	0	0	2
女	0	0	0	2	0	0	0	1	0	0	0	0	34	0	0	0	0	0	0	0	0	0	0	0	0	0	0
七	0	0	0	3	0	0	0	0	0	0	0	0	1	26	0	2	0	0	0	0	0	0	0	0	0	0	0
千	0	0	0	0	0	0	0	0	0	0	0	3	0	0	25	0	0	0	0	0	0	0	0	0	0	0	3
人	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	31	0	0	0	0	0	0	0	0	0	0	0
三	1	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	27	0	0	0	0	0	0	0	0	0	0
十	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	30	0	0	0	0	0	0	0	0	0
四	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	39	0	0	0	0	0	0	1	0
他	0	0	0	0	0	0	0	6	0	0	15	0	0	0	0	0	0	0	0	17	0	0	0	0	0	0	0
万	0	0	0	0	0	0	0	1	0	1	0	1	1	2	0	1	0	0	0	0	17	0	0	2	0	0	4
文	0	0	0	2	0	0	0	0	6	0	0	0	9	0	0	1	0	0	0	0	0	28	0	0	0	0	0
我	0	0	0	0	0	2	0	7	0	3	0	0	1	1	0	0	0	0	0	0	0	0	20	0	0	0	0
五	0	0	0	0	0	0	0	0	0	0	0	1	1	10	1	0	0	0	0	0	0	0	0	56	0	0	0
小	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	34	0	0
一	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	34	0
子	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	30

Figure 6.4: The confusion matrix of our proposed classifier on novice data

returned by the three template matchers for novice users. It clearly shows that correct predictions are heavily distributed on the first several indexes in our system. And the correct predictions are always in the top 13, while BopoNoto and $P's results both have long tails.

6.2.2 Analyses and discussions

The recognition results have shown that our proposed system have better recognition rates than the BopoNoto system as well as the $P template matching algorithm. It can

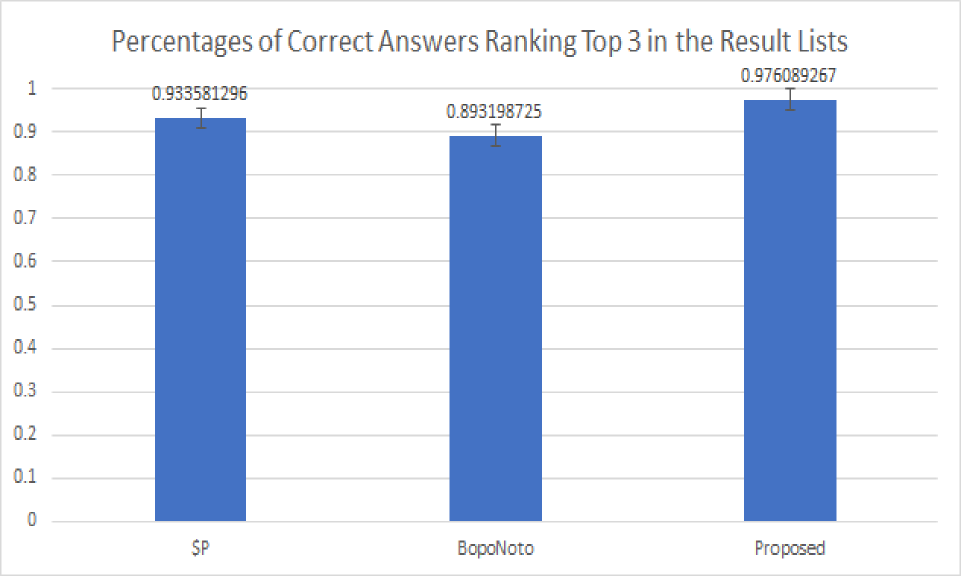


Figure 6.5: The percentages of writing samples whose corresponding templates rank top 3 in the result lists

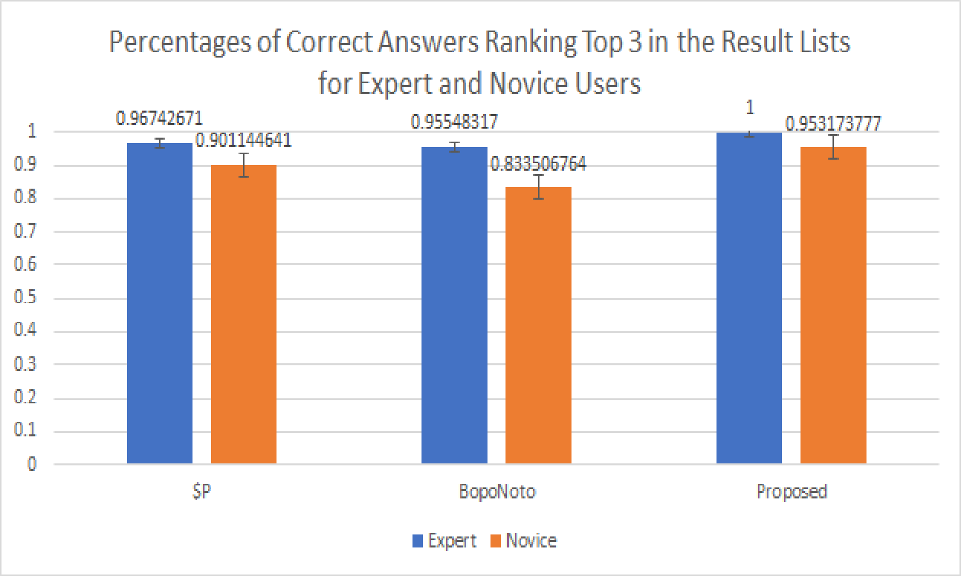
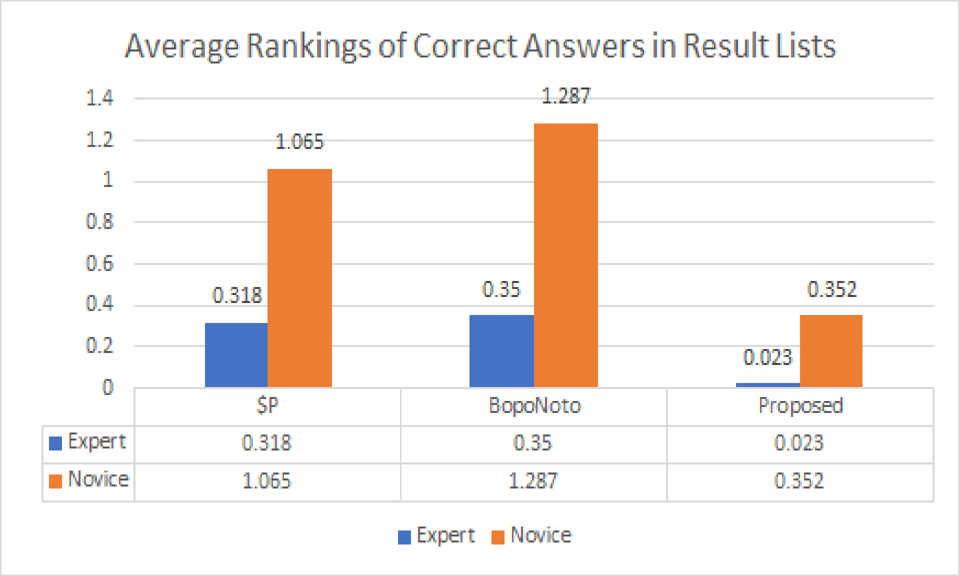


Figure 6.6: The percentage of writing samples whose corresponding templates rank top 3 in the results list for expert and novice users



	$P	BopoNoto	Proposed
Expert	0.318	0.35	0.023
Novice	1.065	1.287	0.352

Figure 6.7: Average rankings of correct templates in the result lists

be observed from our experiments that BopoNoto's two-part recognition algorithm has performed significantly better than using Hausdorff distance as the only similarity metric. This proves that the point coverage ratio has successfully reduced the errors caused by writing inputs that are visually similar to a subset of points of a particular template. Though BopoNoto reaches pretty decent recognition rate on expert data, the result has validated our observation that the proposed template matching algorithm is more robust with the user's ability in Chinese handwriting. The results have also proved our hypothesis that projections is an important feature for classification due to the nature of strokes in Chinese characters, and that dynamic programming can efficiently avoid the possible errors in recognition introduced by the distribution of strokes in students' handwritings. From the confusion matrices, it can be observed that like many other vision-based template matching algorithms, our system can get confused on visually similar symbols. However, from Fig. 6.8 we can see that the correct answer are almost always ranked top in the result list by our template matcher, and that our system has less variance in classifying novice

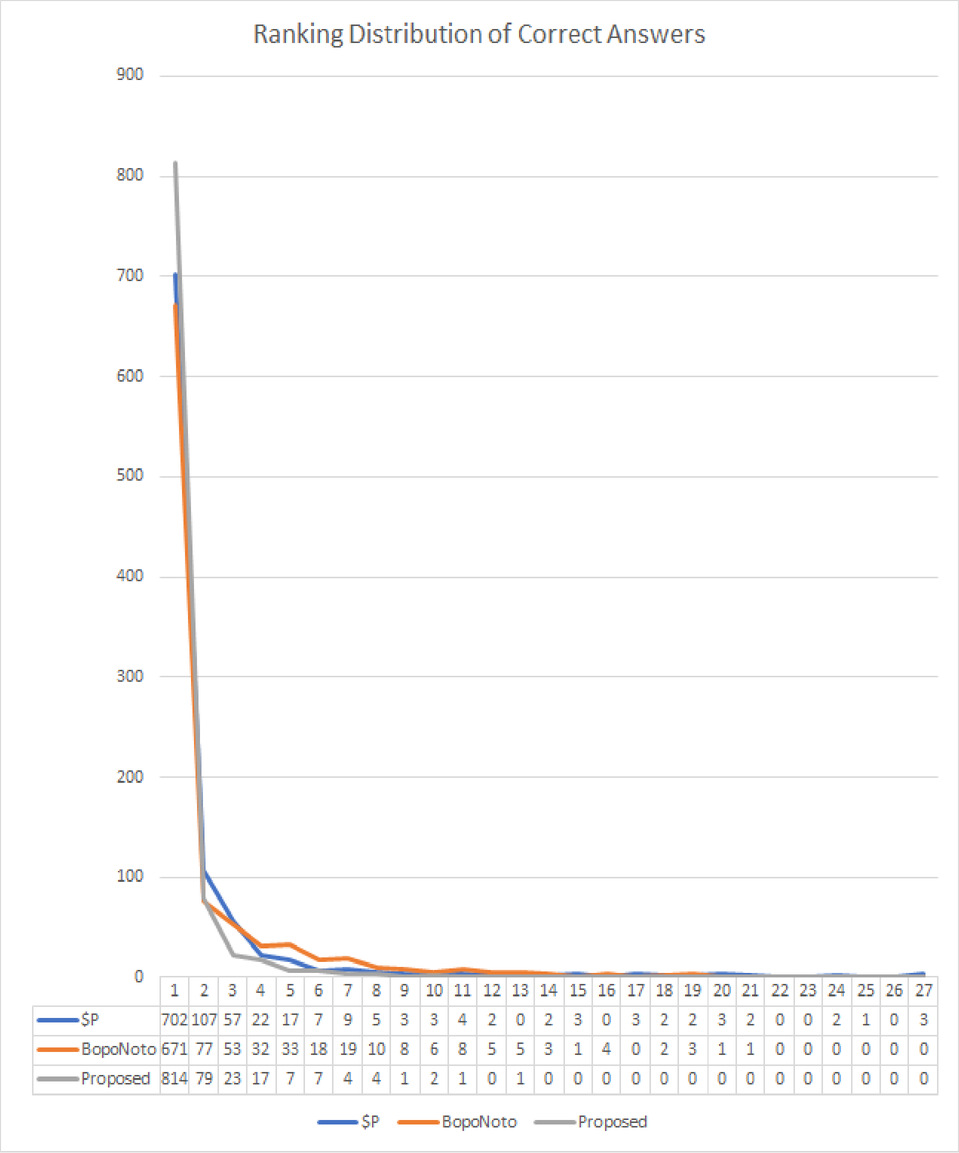


Figure 6.8: Number of predictions at each index in the result lists returned by the template matchers on the novice dataset

users' inputs than BopoNoto and $P.

6.3 Handwriting technique assessment

In this section, we evaluate our system's ability to assess students' writing techniques and the accuracy in detecting their mistakes.

6.3.1 Results

6.3.1.1 Stroke matching

The algorithm for finding stroke correspondence from sample to template is essential for assessing the techniques of each handwriting sample since it determines the correctness of our stroke order and direction assessments. In order to prove the robustness our stroke matching algorithm, we tested it on both novice data (group 3) and cursive writing samples from 9 expert users (group 2). For samples that have correct stroke counts, we compare our method with BopoNoto [17], which uses a similar greedy algorithm to find stroke correspondences. Table 6.4 shows 1455 out of 1651 handwritings are written with correct amounts of strokes, 136 with less strokes, and 36 with more strokes than template. And our system is able to find correct stroke correspondences with an accuracy of 98.5%, which is significantly better than BopoNoto's 92.4%. While BopoNoto is not able to deal with characters with concatenated and broken strokes, our system is able to correctly find stroke correspondences for samples that have less strokes than template with an accuracy of 85.6%, and for those that have more strokes with an accuracy of 97.3%.

Type	**Correct stroke count**	**Less strokes**	**More strokes**
Total number	1455	159	37
Successful match (proposed)	1433	136	36
Successful match (BopoNoto)	1344	N/A	N/A

Table 6.4: Stroke matching result

6.3.1.2 Stroke order judgment

In our proposed system, we provide binary to the students regarding the correctness of the order of strokes in each character they write. BopoNoto [17] has proposed an efficient way to judge stroke order for characters that have correct stroke counts, but breaks when stroke count is incorrect. Observing that more information is yet to be retrieved and utilized in broken strokes and concatenated strokes, our proposed method aims at providing richer feedback, by assessing and instructing on all characters regardless of stroke count correctness.

1. We first evaluated the accuracy of our stroke order judgment method on characters with correct stroke counts and compared it against BopoNoto [17]. This result is largely dependent on the one to one stroke matching accuracy. Table 6.5 and Table 6.6 show our method reaches an accuracy of 98.6% and an F-measure of 99.1%, performing slightly better than BopoNoto's 96.2% and 95.4%, which can be observed from Table 6.7 and Table 6.8.

	Predicted: correct	**Predicted: incorrect**
Actual: correct	1143	17
Actual: incorrect	4	287

Table 6.5: Confusion matrix of proposed stroke order judgment when stroke count is correct

Accuracy	**Precision**	**Recall**	**F-measure**
98.6%	99.7%	98.5%	99.1%

Table 6.6: Results of proposed stroke order judgment when stroke count is correct

2. We then evaluated our system's ability to judge stroke order on characters with incorrect stroke counts. Table 6.9 and Table 6.10 show our method is able to accurately judge stroke order when stroke counts are incorrect with an accuracy of 87.8% and an F-measure of 89%.

	Predicted: correct	**Predicted: incorrect**
Actual: correct	1079	83
Actual: incorrect	12	277

Table 6.7: Confusion matrix of BopoNoto's stroke order judgment when stroke count is correct

Accuracy	**Precision**	**Recall**	**F-measure**
96.2%	98.9%	92.9%	95.4%

Table 6.8: Results of BopoNoto's stroke order judgment when stroke count is correct

	Predicted: correct	**Predicted: incorrect**
Actual: correct	102	25
Actual: incorrect	0	49

Table 6.9: Confusion matrix of proposed stroke order judgment when stroke count is incorrect

Accuracy	**Precision**	**Recall**	**F-measure**
85.8%	100%	80.3%	89.0%

Table 6.10: Results of proposed stroke order judgment when stroke count is incorrect

6.3.1.3 Stroke direction judgment

We evaluate the binary classification result on the correctness of stroke directions. We evaluate this judgment on characters with correct and incorrect stroke counts separately. We first test our algorithm on handwriting samples that have correct stroke counts. Table 6.11 and Table 6.12 show our method accurately classifies between characters with and without direction errors with an accuracy of 98.3% and an F-measure of 99.1%, which is significantly better than BopoNoto's 80.4% and 88.1%. Table 6.15 and Table 6.16 show our method is able to detect wrong directions when stroke count is incorrect as well, where our classification reaches an accuracy of 87.2% and an F-measure of 91.9%.

	Predicted: correct	**Predicted: incorrect**
Actual: correct	1317	20
Actual: incorrect	5	106

Table 6.11: Confusion matrix of proposed stroke order judgment when stroke count is correct

Accuracy	**Precision**	**Recall**	**F-measure**
98.3%	99.6%	98.5%	99.1%

Table 6.12: Results of proposed stroke order judgment when stroke count is correct

	Predicted: correct	**Predicted: incorrect**
Actual: correct	1055	282
Actual: incorrect	2	109

Table 6.13: Confusion matrix of BopoNoto's stroke order judgment when stroke count is correct

Accuracy	**Precision**	**Recall**	**F-measure**
80.4%	99.8%	78.9%	88.1%

Table 6.14: Results of BopoNoto's stroke order judgment when stroke count is correct

	Predicted: correct	**Predicted: incorrect**
Actual: correct	130	23
Actual: incorrect	0	27

Table 6.15: Confusion matrix of proposed stroke direction judgment when stroke count is incorrect

Accuracy	**Precision**	**Recall**	**F-measure**
87.2%	100%	85.0%	91.9%

Table 6.16: Results of proposed stroke direction judgment when stroke count is correct

6.3.2 Analyses and discussions

The results in this section have indicated that our proposed system is able to detect students' writing technique mistakes in a more well-rounded way so as to provide richer feedback to them. The proposed system is able to accurately find matching strokes from sample to templates, for writing samples with both correct and incorrect stroke counts. The successful stroke matching enables our system to fully utilize information from every stroke in the sample and give feedback to users. The proposed system reaches high accuracy on detecting mistakes in stroke count, order, and directions. While BopoNoto [17] tends to assess the correctness of each writing sample in a sequential way and will not give feedback on one aspect if there are mistakes on the previous one, our proposed system assesses the three aspects of technique independently, providing richer feedback to users.

6.4 Handwriting quality assessment

6.4.1 Experiment

In this section, we evaluate the importance of features on distinguishing between good and bad handwritings. We use data from user group 1 and group 3 as training data, automatically labeled as "good" and "bad" handwritings, respectively. In this section, we first conducted statistical analyses to these two labeled datasets and the feature values associated with them. We then performed a Best First Search algorithm [74] to select the subset of features that has the best separability between the two labeled datasets. We then built a classifier to distinguish between the two labeled datasets using the selected features.

6.4.2 Results

6.4.2.1 Statistical analysis

We conducted Welch's t-test [75] on our data set to see how well these features distinguish expert and novice data. T-values show how the average values are different in these two data sets, and features with p-values lower than 0.05 are considered to be significant. Table 6.17 shows the t-test results of the features we extracted for handwriting quality assessment.

Feature	Description	t-value	p-value
f1	Ratio of bounding box	-7.99319	$3.0384 * 10^{-15}$
f2	Centroid location (x)	-9.9457	$1.3064 * 10^{-22}$
f3	Centroid location (y)	-9.87485	$2.5608 * 10^{-22}$
f4	Hausdorff distance	-18.4768	$4.0003 * 10^{-69}$
f5	Tanimoto coefficient	15.1258	$2.2817 * 10^{-48}$
f6	Yule coefficient	5.188746	$2.4140 * 10^{-7}$
f7	Stroke length distribution difference	-9.28028	$9.6376 * 10^{-20}$
f8	Stroke orientation similarity (average)	7.477118926	$1.5143 * 10^{-13}$
f9	Stroke orientation similarity (min)	6.024062129	$2.2875 * 10^{-9}$
f10	Average speed	-11.96446616	$4.7238 * 10^{-31}$
f11	Speed fluidity	-12.39988266	$2.8252 * 10^{-33}$
f12	Horizontal projection difference	-12.44523023	$4.9875 * 10^{-34}$
f13	Vertical projection difference	-7.517797442	$9.0902 * 10^{-14}$

Table 6.17: T-test results for features

6.4.2.2 Feature selection

Although the t-test indicated all 13 features have very low p-values, it does not guarantee that the entire feature set is optimal for our classification task. Subset selection is a method for selecting a subset of features that represent the data well [76]. We used Weka [77, 78], a statistical analysis and machine learning software based on Java to run subset selection using the Best First Search algorithm [74]. The feature subset selection result in Table 6.18 shows 10 out of the 13 features were selected, excluding f6, f9, and f13.

6.4.2.3 Classification

Based on the features selected, we applied a Random Forest [79] on our dataset using selected features to classify between expert and novice data. We first used 10-fold cross-

Feature	**Description**	**Selection result**
f1	Ratio of bounding box	Selected
f2	Centroid location (x)	Selected
f3	Centroid location (y)	Selected
f4	Hausdorff distance	Selected
f5	Tanimoto coefficient	Selected
f6	Yule coefficient	Not selected
f7	Stroke length distribution difference	Selected
f8	Stroke orientation similarity (average)	Selected
f9	Stroke orientation similarity (min)	Not selected
f10	Average speed	Selected
f11	Speed fluidity	Selected
f12	Horizontal projection difference	Selected
f13	Vertical projection difference	Not selected

Table 6.18: Feature selection result

validation [80] and Table 6.19 and Table 6.20 shows our classification result reaches an

	Predicted: expert	**Predicted: novice**
Actual: expert	653	150
Actual: novice	176	674

Table 6.19: Confusion matrix of classification with 10-fold cross-validation

Class	**Precision**	**Recall**	**F-measure**
Expert	0.788	0.813	0.800
Novice	0.818	0.793	0.805
Overall	0.803	0.803	0.803

Table 6.20: Detailed accuracy by class with 10-fold cross-validation

F-measure of 0.803. We also used leave-one-out cross-validation [81] on the dataset. The results in Table 6.21 and Table 6.22 show that we reach an F-measure of 0.799. In order to prove we are not overfitting the dataset, we randomly selected two thirds of the dataset as training data, and tested on the remaining one third. An F-measure of 0.771 is reached

	Predicted: expert	**Predicted: novice**
Actual: expert	648	155
Actual: novice	177	673

Table 6.21: Confusion matrix of classification with leave-one-out cross-validation

Class	**Precision**	**Recall**	**F-measure**
Expert	0.785	0.807	0.796
Novice	0.813	0.792	0.802
Overall	0.800	0.799	0.799

Table 6.22: Detailed accuracy by class with leave-one-out cross-validation

as indicated in Table 6.23 and Table 6.24. We also performed the classification by using data from the first 13 users (7 expert users and 6 novice users) as training data, and the rest as test data, which reached an F-measure of 0.674, as can be seen in Table 6.25 and Table 6.26. Last but not least, we performed a leave-one-user-out cross-validation on all of our user data and repeated the experiment 21 times. We reached a weighted F-measure of 0.804, which can be seen from Table 6.27

	Predicted: expert	**Predicted: novice**
Actual: expert	201	50
Actual: novice	75	219

Table 6.23: Confusion matrix of classification with randomly selected 2/3 of dataset used as training data and 1/3 as test data

Class	**Precision**	**Recall**	**F-measure**
Expert	0.728	0.801	0.763
Novice	0.814	0.745	0.778
Overall	0.775	0.771	0.771

Table 6.24: Detailed accuracy by class with randomly selected 2/3 of dataset used as training data and 1/3 as test data

	Predicted: expert	**Predicted: novice**
Actual: expert	187	87
Actual: novice	109	218

Table 6.25: Confusion matrix of classification with first 13 users' data as training data and the rest as test data

Class	**Precision**	**Recall**	**F-measure**
Expert	0.632	0.682	0.656
Novice	0.715	0.667	0.690
Overall	0.677	0.674	0.674

Table 6.26: Detailed accuracy by class with first 13 users' data as training data and the rest as test data

Class	**Precision**	**Recall**	**F-measure**
Expert	1	0.701	0.798
Novice	1	0.698	0.811
Overall	1	0.699	0.804

Table 6.27: Detailed accuracy by class with leave-one-user-out cross-validation

6.4.3 Analyses and discussions

We are using a dataset composing of 803 samples written by experts, and 850 samples written by novice learners to find significant features to distinguish between well- and badly-written characters. The statistical analysis has shown that similarity metrics such as Hausdorff distance and Tanimoto coefficient, are most important to tell good from bad samples. We also observed one of the common characteristic of novice data can have a bad shape overall, which is reflected by the ratio of bounding box. We also found features that reflect the internal balance of characters, such as stroke length distribution, projections, and location of centroids, have good separability on handwritings from experts and novices. Another finding is that novice students have more difficulty mastering the horizontal balance of characters than vertical, and the horizontal projection feature is much more important than vertical projection in distinguishing between the two user groups. Using the selected features, our system is able to classify between expert and novice handwritings with an F-measure of 0.803 using 10-fold cross-validation, 0.799 using leave-one-out cross-validation, 0.771 using two thirds of randomly selected data as training data and the rest as test data, and 0.674 using the first 13 users' data as training data, and the rest as test data.

6.5 Usability

6.5.1 Experiment

In this study, we focus on evaluating the usability of our user interface for CSL students by looking at how our feedback has helped improve learners' handwriting techniques. Users from group 3 were asked to handwrite each character mimicking the template provided, and use the feedback provided by our system as debug information to improve their future tries. We count the mistakes they make in writing techniques in each try, and observe if our system is able to help them write better. After the data collection, we asked these novice users to provide feedback for our system by giving scores from 1-5 to indicate how each part of the feedback had helped them improve.

6.5.2 Results

We first analyzed data on the number of technical mistakes the students made on each try. Fig. 6.9 shows the number of mistakes in students' handwriting decrease significantly with the help of our instructional feedback. In the survey after collecting data from students, we asked the students to give scores from 1 to 5 to our feedback and animations on each writing technique, and let us know their thoughts and comments. Fig. 6.10 shows the scores each students give to our technique feedback. From Table 6.28 we can observe that students are very satisfied on the system's feedback on stroke count and stroke direction, but sometimes get confused on the feedback for stroke order. We hope to figure

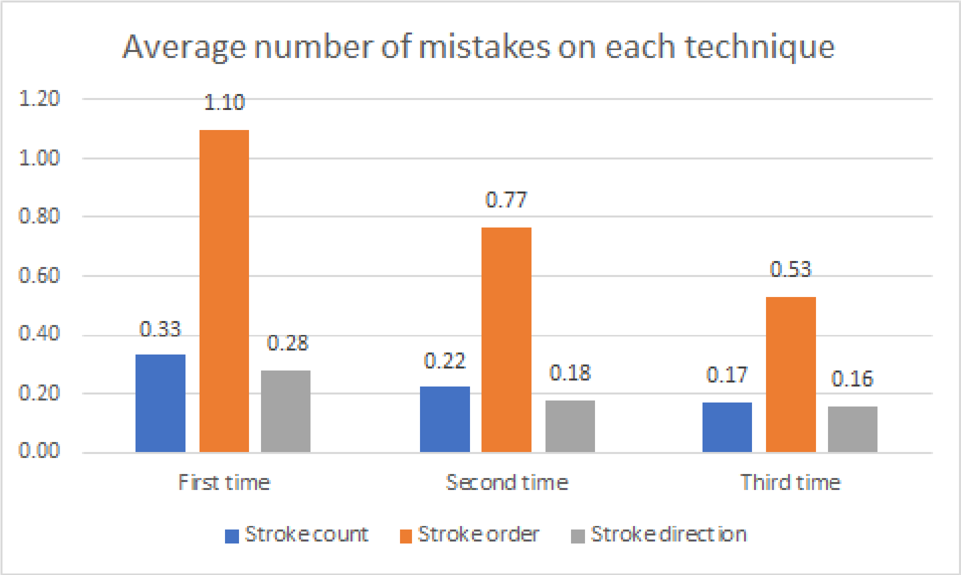


Figure 6.9: Average number of mistakes made on each try in each technique

out a better way to instruct on stroke order in the near future. Students mention that the way we highlight broken and concatenated strokes are very helpful to tell them how to separate strokes. One of the students also mention the trace we present by animations on strokes with wrong directions helps her clearly remember which stroke should go in which direction. Some students also mention that it is not easy to clearly understand how they should fix their stroke order mistakes based on our feedback. These good feedbacks are very helpful for us to improve our system.

Stroke count	Stroke order	Stroke direction
4.22	3.33	4.11

Table 6.28: Average ratings on each technique feedback

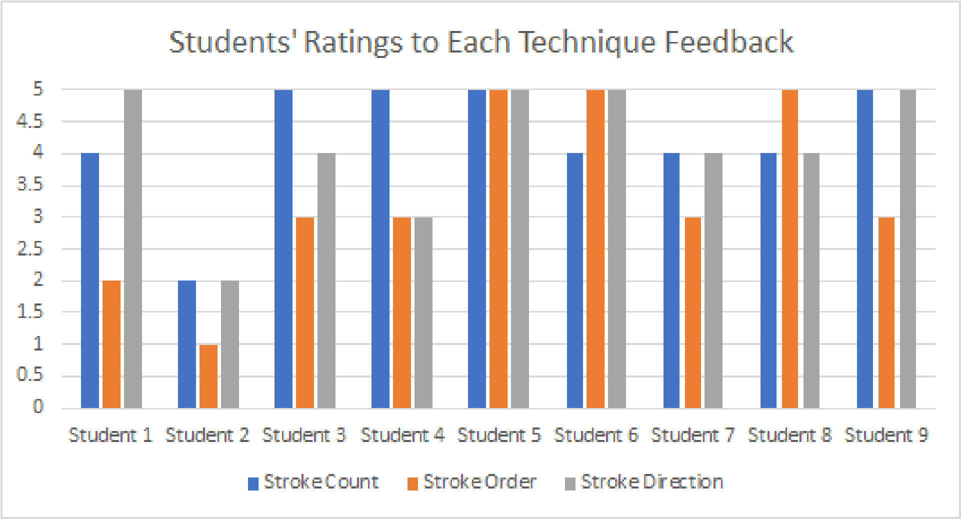


Figure 6.10: Scores given by students on system's feedback on each technique

6.5.3 Analyses and discussions

The results have shown that our system gives clear feedback and instructions to users and is able to help them improve their writing techniques overtime. We noticed that stroke count and stroke directions are not big obstacles for CSL learners, since it is often straightforward to tell how many strokes are their in the character, and it is often clear that stroke directions are fundamentally from left to right, and from up to bottom. Students tend to make way more mistakes on stroke order, which is hard to summarize giving multiple intersection strokes in each character. We also observed from students inputs that students whose native languages are English tend to draw vertical strokes first, while in most case it is the horizontal strokes that should go first. As can be observed from Fig. 6.9 that novice students have an average of more than 1 mistakes on stroke order the first time they write, and even at the third time they still make an average of 0.53 mistakes. This indicates the great need for better stroke order instructions. From the survey result, we can observe that our feedback on stroke count mistakes are very helpful to users. However, the way we give feedback on stroke order mistakes needs to be more clear.

7. FUTURE WORK

Pressure is an important feature for Chinese handwriting and it is highly related to speed and is significant to distinguish from novice and experienced writers. We plan to use pressure-sensitive devices with our system in future developments and collection, assess students' writing techniques related to pressure, and find out how pressure affects the visual quality of handwritings.

We also plan to integrate visual quality assessment into our system by developing an automatic grading algorithm based on the features that we proved to be significant, so that students not only learn to write each character correctly and with good techniques, but also learn to reach a good visual quality. In addition, we hope to deploy our system to an entry level Chinese language class in an American college, and see if our system can do better jobs in teaching handwriting than human teachers.

8. CONCLUSIONS

In this thesis, we presented a sketch-based educational system for Chinese handwriting. Compared to tradition classroom education, our system is able to analyze students' digital writing data and give feedback in real time. We developed a new method for handwriting recognition that reaches an accuracy of 86% on novice learners' data. Our technique assessment system is able to accurately detect students' mistakes on stroke count, stroke order, and stroke direction, and give feedback in real time that helps students improve their future writing style. We also collected handwriting samples from both experts and novice students, and found 10 features that are significant for handwriting quality assessment. We were also able to use these features to classify expert data and novice data using machine learning techniques, which reached an F-measure of around 80%. We evaluated the usability of our system on novice CSL learners, and observed the users had overall positive feedback on our educational system.

REFERENCES

[1] L. Tsung and K. Cruickshank, *Teaching and learning Chinese in global contexts: CFL worldwide*. Bloomsbury Publishing, 2010.

[2] J. Z. Xing, *Teaching and learning Chinese as a foreign language: A pedagogical grammar*, vol. 1. Hong Kong University Press, 2006.

[3] P. Taele, "Freehand sketch recognition for computer-assisted language learning of written east asian languages," MS Master's Thesis, Texas A&M University (TAMU), College Station, TX, USA, December 2010. Advisor: Tracy Hammond. 96 pages.

[4] M. E. Everson, "Literacy development in chinese as a foreign language," *Teaching Chinese as a foreign language: Theories and applications*, pp. 97–111, 2009.

[5] C.-C. Han, C.-H. Chou, and C.-S. Wu, "An interactive grading and learning system for chinese calligraphy," *Machine Vision and Applications*, vol. 19, no. 1, pp. 43–55, 2008.

[6] T. Hammond and R. Davis, "Ladder, a sketching language for user interface developers," *Computers & Graphics*, vol. 29, no. 4, pp. 518–532, 2005.

[7] T. Hammond, D. Logsdon, J. Peschel, J. Johnston, P. Taele, A. Wolin, and B. Paulson, "A sketch recognition interface that recognizes hundreds of shapes in course-of-action diagrams," in *CHI'10 Extended Abstracts on Human Factors in Computing Systems*, pp. 4213–4218, ACM, 2010.

[8] J. Jorge and F. Samavati, *Sketch-based interfaces and modeling*. Springer Science & Business Media, 2010.

[9] S. Valentine, F. Vides, G. Lucchese, D. Turner, H.-h. Kim, W. Li, J. Linsey, and T. Hammond, "Mechanix: a sketch-based tutoring and grading system for free-body diagrams," *AI Magazine*, vol. 34, no. 1, p. 55, 2012.

[10] O. Atilola, C. Osterman, T. Hammond, and J. Linsey, "Mechanix: the development of a sketch recognition truss tutoring system," *Proceedings of the American*, 2012.

[11] S. Valentine, R. Lara-Garduno, J. Linsey, and T. Hammond, "Mechanix: A sketch-based tutoring system that automatically corrects hand-sketched statics homework," in *The impact of pen and touch technology on education*, pp. 91–103, Springer, 2015.

[12] T. Nelligan, S. Polsley, J. Ray, M. Helms, J. Linsey, and T. Hammond, "Mechanix: a sketch-based educational interface," in *Proceedings of the 20th International Conference on Intelligent User Interfaces Companion*, pp. 53–56, ACM, 2015.

[13] B. Williford, P. Taele, T. Nelligan, W. Li, J. Linsey, and T. Hammond, "Persketchtivity: an intelligent pen-based educational application for design sketching instruction," in *Revolutionizing Education with Digital Ink*, pp. 115–127, Springer, 2016.

[14] W. Li, E. Hilton, T. Hammond, and J. Linsey, "Persketchtivity: an intelligent pen-based online education platform for sketching instruction," in *Proceedings of the conference on Electronic Visualisation and the Arts*, pp. 133–141, British Computer Society, 2016.

[15] P. Taele, L. Barreto, and T. A. Hammond, "Maestoso: An intelligent educational sketching tool for learning music theory.," in *AAAI*, pp. 3999–4005, 2015.

[16] P. Taele and T. Hammond, "Lamps: A sketch recognition-based teaching tool for mandarin phonetic symbols i," *Journal of Visual Languages & Computing*, vol. 21, no. 2, pp. 109–120, 2010.

[17] P. Taele and T. Hammond, “Boponoto: An intelligent sketch education application for learning zhuyin phonetic script.,” in *DMS*, pp. 101–107, 2015.

[18] P. Taele and T. Hammond, “An intelligent sketch-based educational interface for learning complex written east asian phonetic symbols,” in *Revolutionizing Education with Digital Ink*, pp. 129–140, Springer, 2016.

[19] P. Taele and T. Hammond, “Hashigo: A next-generation sketch interactive system for japanese kanji.,” in *IAAI*, vol. 9, pp. 153–158, 2009.

[20] S. P. Ashok Kumar, “Evaluation of conceptual sketches on stylus-based devices,” MS Master’s Thesis, Texas A&M University (TAMU), College Station, TX, USA, May 2016. Advisor: Tracy Hammond.

[21] H. Chen, R. T. Wigand, and M. S. Nilan, “Optimal experience of web activities,” *Computers in human behavior*, vol. 15, no. 5, pp. 585–608, 1999.

[22] Y.-M. Su and J.-F. Wang, “A novel stroke extraction method for chinese characters using gabor filters,” *Pattern Recognition*, vol. 36, no. 3, pp. 635–647, 2003.

[23] D. Rubine, *Specifying gestures by example*, vol. 25. ACM, 1991.

[24] A. C. Long Jr, J. A. Landay, L. A. Rowe, and J. Michiels, “Visual similarity of pen gestures,” in *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pp. 360–367, ACM, 2000.

[25] J. O. Wobbrock, A. D. Wilson, and Y. Li, “Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes,” in *Proceedings of the 20th annual ACM symposium on User interface software and technology*, pp. 159–168, ACM, 2007.

[26] L. Anthony and J. O. Wobbrock, “A lightweight multistroke recognizer for user interface prototypes,” in *Proceedings of Graphics Interface 2010*, pp. 245–252, Canadian Information Processing Society, 2010.

[27] L. Anthony and J. O. Wobbrock, “$ n-protractor: A fast and accurate multistroke recognizer,” in *Proceedings of Graphics Interface 2012*, pp. 117–120, Canadian Information Processing Society, 2012.

[28] R.-D. Vatavu, L. Anthony, and J. O. Wobbrock, “Gestures as point clouds: a $ p recognizer for user interface prototypes,” in *Proceedings of the 14th ACM international conference on Multimodal interaction*, pp. 273–280, ACM, 2012.

[29] B. Paulson and T. Hammond, “Paleosketch: accurate primitive sketch recognition and beautification,” in *Proceedings of the 13th international conference on Intelligent user interfaces*, pp. 1–10, ACM, 2008.

[30] T. M. Sezgin, T. Stahovich, and R. Davis, “Sketch based interfaces: Early processing for sketch understanding,” in *ACM SIGGRAPH 2006 Courses*, p. 22, ACM, 2006.

[31] B. Yu and S. Cai, “A domain-independent system for sketch recognition,” in *Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, pp. 141–146, ACM, 2003.

[32] T. Hammond and R. Davis, “Tahuti: A geometrical sketch recognition system for uml class diagrams,” in *ACM SIGGRAPH 2006 Courses*, p. 25, ACM, 2006.

[33] T. Hammond, “Automatically generating sketch interfaces from shape descriptions,” in *Proceedings of the 4th Annual MIT Student Oxygen Workshop*, 2004.

[34] T. Hammond and R. Davis, “Ladder: A sketch recognition language,” in *IN MIT COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE LABORATORY ANNUAL RESEARCH ABSTRACT. MIT CSAIL*, 2004.

[35] T. Hammond and R. Davis, “Ladder: A language to describe drawing, display, and editing in sketch recognition,” in *ACM SIGGRAPH 2006 Courses*, p. 27, ACM, 2006.

[36] T. A. Hammond, *Ladder: A perceptually-based language to simplify sketch recognition user interface development*. PhD thesis, Massachusetts Institute of Technology, 2007.

[37] T. Hammond and R. Davis, “Creating the perception-based ladder sketch recognition language,” in *Proceedings of the 8th ACM Conference on Designing Interactive Systems*, pp. 141–150, ACM, 2010.

[38] P. Corey and T. Hammond, “Gladder: Combining gesture and geometric sketch recognition.,” in *AAAI*, vol. 8, pp. 1788–1789, 2008.

[39] L. B. Kara and T. F. Stahovich, “An image-based, trainable symbol recognizer for hand-drawn sketches,” *Computers & Graphics*, vol. 29, no. 4, pp. 501–517, 2005.

[40] S. Belongie, J. Malik, and J. Puzicha, “Shape context: A new descriptor for shape matching and object recognition,” in *Advances in neural information processing systems*, pp. 831–837, 2001.

[41] L. Barreto, P. Taele, and T. Hammond, “A stylus-driven intelligent tutoring system for music education instruction,” in *Revolutionizing Education with Digital Ink*, pp. 141–161, Springer, 2016.

[42] D. Dixon, M. Prasad, and T. Hammond, “icandraw: using sketch recognition and corrective feedback to assist a user in drawing human faces,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 897–906, ACM, 2010.

[43] H.-h. Kim, S. Valentine, P. Taele, and T. Hammond, “Easysketch: A sketch-based educational interface to support childrenâĂŹs self-regulation and school readiness,”

in *The Impact of Pen and Touch Technology on Education*, pp. 35–46, Springer, 2015.

[44] H.-h. Kim, P. Taele, S. Valentine, J. Liew, and T. Hammond, “Developing intelligent sketch-based applications to support childrenâĂŹs self-regulation and school readiness,”

[45] H.-H. Kim, *A Fine Motor Skill Classifying Framework to Support Children’s Self-Regulation Skills and School Readiness*. PhD thesis, 2016.

[46] H. Kim, P. Taele, J. Seo, J. Liew, and T. Hammond, “Easysketch2: A novel sketch-based interface for improving childrenâĂŹs fine motor skills and school readiness,” in *Proceedings of the Joint Symposium on Computational Aesthetics and Sketch Based Interfaces and Modeling and Non-Photorealistic Animation and Rendering*, pp. 69–78, Eurographics Association, 2016.

[47] B. Williford, “Sketchtivity: Improving creativity by learning sketching with an intelligent tutoring system,” in *Proceedings of the 2017 ACM SIGCHI Conference on Creativity and Cognition*, pp. 477–483, ACM, 2017.

[48] S. Keshavabhotla, B. Williford, S. Kumar, E. Hilton, P. Taele, W. Li, J. Linsey, and T. Hammond, “Conquering the cube: learning to sketch primitives in perspective with an intelligent tutoring system,” in *Proceedings of the Symposium on Sketch-Based Interfaces and Modeling*, p. 2, ACM, 2017.

[49] A. N. Bhat, *Sketchography-Automatic Grading of Map Sketches for Geography Education*. PhD thesis, 2017.

[50] J.-I. Herrera-Camara and T. Hammond, “Flow2code: from hand-drawn flowcharts to code execution,” in *Proceedings of the Symposium on Sketch-Based Interfaces and Modeling*, p. 3, ACM, 2017.

[51] P. Taele and T. A. Hammond, “A geometric-based sketch recognition approach for handwritten mandarin phonetic symbols i.,” in *DMS*, pp. 270–275, 2008.

[52] P. Taele and T. Hammond, “Using a geometric-based sketch recognition approach to sketch chinese radicals.,” in *AAAI*, vol. 8, pp. 1832–1833, 2008.

[53] P. Taele and T. Hammond, “Chinese characters as sketch diagrams using a geometric-based approach,” in *Proceedings of the 2008 IEEE Symposium on Visual Languages and Human-Centric Computing Workshop on Sketch Tools for Diagramming*, pp. 74–82, 2008.

[54] Z. Yin, “Chinese calligraphist: A sketch based learning tool for learning written chinese,” MS Master’s Thesis, Texas A&M University (TAMU), College Station, TX, USA, August 2014. Advisor: Tracy Hammond.

[55] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, “Backpropagation applied to handwritten zip code recognition,” *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[56] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, “Handwritten digit recognition with a back-propagation network,” in *Advances in neural information processing systems*, pp. 396–404, 1990.

[57] R. Romero, R. Berger, R. Thibadeau, and D. Touretzky, “Neural network classifiers for optical chinese character recognition,” in *Proceedings of the Fourth Annual Symposium on Document Analysis and Information Retrieval*, pp. 385–98, 1995.

[58] R. D. Romero, D. S. Touretzky, and R. H. Thibadeau, “Optical chinese character recognition using probabilistic neural networks,” *Pattern recognition*, vol. 30, no. 8, pp. 1279–1292, 1997.

[59] Z.-L. Bai and Q. Huo, "A study on the use of 8-directional features for online handwritten chinese character recognition," in *Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on*, pp. 262–266, IEEE, 2005.

[60] Q. Huo, Y. Ge, and Z.-D. Feng, "High performance chinese ocr based on gabor features, discriminative feature extraction and model training," in *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, vol. 3, pp. 1517–1520, IEEE, 2001.

[61] K. Ding, Z. Liu, L. Jin, and X. Zhu, "A comparative study of gabor feature and gradient feature for handwritten chinese character recognition," in *Wavelet Analysis and Pattern Recognition, 2007. ICWAPR'07. International Conference on*, vol. 3, pp. 1182–1186, IEEE, 2007.

[62] A. Wolin, B. Eoff, and T. Hammond, "Shortstraw: A simple and effective corner finder for polylines.," in *SBM*, pp. 33–40, 2008.

[63] Y. Xiong and J. J. LaViola Jr, "Revisiting shortstraw: improving corner finding in sketch-based interfaces," in *Proceedings of the 6th Eurographics Symposium on Sketch-Based Interfaces and Modeling*, pp. 101–108, ACM, 2009.

[64] R. Sun, Z. Lian, Y. Tang, and J. Xiao, "Aesthetic visual quality evaluation of chinese handwritings.," in *IJCAI*, pp. 2510–2516, 2015.

[65] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series.," in *KDD workshop*, vol. 10, pp. 359–370, Seattle, WA, 1994.

[66] W. Li and T. A. Hammond, "Recognizing text through sound alone.," in *AAAI*, 2011.

[67] J. Edmonds, "Paths, trees, and flowers," *Canadian Journal of mathematics*, vol. 17, no. 3, pp. 449–467, 1965.

[68] T. H. Cormen, *Introduction to algorithms*. MIT press, 2009.

[69] C. H. Papadimitriou and K. Steiglitz, *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.

[70] T. Kailath, "The divergence and bhattacharyya distance measures in signal selection," *IEEE transactions on communication technology*, vol. 15, no. 1, pp. 52–60, 1967.

[71] C.-L. Liu, I.-J. Kim, and J. H. Kim, "Model-based stroke extraction and matching for handwritten chinese character recognition," *Pattern Recognition*, vol. 34, no. 12, pp. 2339–2352, 2001.

[72] A. Welford, A. Norris, and N. Shock, "Speed and accuracy of movement and their changes with age," *Acta psychologica*, vol. 30, pp. 3–15, 1969.

[73] D. E. Meyer, J. Keith-Smith, S. Kornblum, R. A. Abrams, and C. E. Wright, "Speed-accuracy tradeoffs in aimed movements: Toward a theory of rapid voluntary action.," 1990.

[74] L. Xu, P. Yan, and T. Chang, "Best first strategy for feature selection," in *Pattern Recognition, 1988., 9th International Conference on*, pp. 706–708, IEEE, 1988.

[75] G. D. Ruxton, "The unequal variance t-test is an underused alternative to student's t-test and the mann–whitney u test," *Behavioral Ecology*, vol. 17, no. 4, pp. 688–690, 2006.

[76] M. E. Broadbent, M. Brown, K. Penner, I. Ipsen, and R. Rehman, "Subset selection algorithms: Randomized vs. deterministic," *SIAM Undergraduate Research Online*, vol. 3, pp. 50–71, 2010.

[77] G. Holmes, A. Donkin, and I. H. Witten, "Weka: A machine learning workbench," in *Intelligent Information Systems, 1994. Proceedings of the 1994 Second Australian and New Zealand Conference on*, pp. 357–361, IEEE, 1994.

[78] S. R. Garner *et al.*, “Weka: The waikato environment for knowledge analysis,” in *Proceedings of the New Zealand computer science research students conference*, pp. 57–64, 1995.

[79] A. Liaw, M. Wiener, *et al.*, “Classification and regression by randomforest,” *R news*, vol. 2, no. 3, pp. 18–22, 2002.

[80] R. Kohavi *et al.*, “A study of cross-validation and bootstrap for accuracy estimation and model selection,” in *Ijcai*, vol. 14, pp. 1137–1145, Stanford, CA, 1995.

[81] P. A. Lachenbruch and M. R. Mickey, “Estimation of error rates in discriminant analysis,” *Technometrics*, vol. 10, no. 1, pp. 1–11, 1968.