ZENJI: HEURISTIC ALGORITHMS TO IDENTIFY AND CORRECT COMMON JAPANESE KANJI HANDWRITING MISTAKES

An Engineering Honors Thesis

by

COLE BROBERG

Submitted to the Undergraduate Research office at
Texas A&M University
in partial fulfillment of the requirements for the designation of

ENGINEERING HONORS IN COMPUTER SCIENCE AND ENGINEERING

Approved by
Faculty Research Advisor: Dr. Paul Taele

December 2024

Major: Computer Science

RESEARCH COMPLIANCE CERTIFICATION

Research activities involving the use of human subjects, vertebrate animals, and/or biohazards must be reviewed and approved by the appropriate Texas A&M University regulatory research committee (i.e., IRB, IACUC, IBC) before the activity can commence. This requirement applies to activities conducted at Texas A&M and to activities conducted at non-Texas A&M facilities or institutions. In both cases, students are responsible for working with the relevant Texas A&M research compliance program to ensure and document that all Texas A&M compliance obligations are met before the study begins.

I, Cole Broberg, certify that all research compliance requirements related to this Undergraduate Research Scholars thesis have been addressed with my Faculty Research Advisor prior to the collection of any data used in this final thesis submission.

This project did not require approval from the Texas A&M University Research Compliance & Biosafety office.

TABLE OF CONTENTS

ABSTRACT

Zenji: Heuristic Algorithms to Identify and Correct Common Japanese Kanji Handwriting Mistakes

Cole Broberg
Department of Computer Science and Engineering
Texas A&M University

Faculty Research Advisor: Dr. Paul Taele
Department of Computer Science and Engineering
Texas A&M University

The *kanji* writing system—tens of thousands of logographic Japanese characters originating from China—is by far the most complex of the three Japanese scripts. *Kanji* present a unique challenge for English-speaking Japanese learners, as they lack an equivalent in western tongues. Whether studying in a classroom or independently, Japanese language learners are often exposed to mnemonics or other techniques to aid in the memorization of these symbols. However, the classroom setting presently provides a unique opportunity for learners to hand-write the characters and receive feedback on their writing. The difficulty and lack of feedback on handwriting for independent learners discourages the practice of hand-writing *kanji*, which can stunt the learning process. This paper introduces *Zenji*, a system of heuristic algorithms to analyze student handwriting of *kanji*. *Zenji* provides a lightweight solution that may easily be run on students' personal computing devices and naturally integrated into existing mobile spaced-repetition solutions for *kanji* learning. The *Zenji* system provides users overall and per-stroke numeric, visual, and written feedback on various metrics, focusing on common mistakes. In validation testing, *Zenji*'s feedback proved to be highly accurate across the categories that it assesses, highlighting user errors and combining a classroom-like learning experience with the flexibility of independent study.

ACKNOWLEDGMENTS

Contributors

I would like to thank Dr. Paul Taele of Texas A&M University for valuable background and feedback on the writing of this paper.

Thanks also go to Samuel Torres, Miguel Garcia, Charlotte Harrington, Liam Benkel, and Naimur Rahman for their development work on the supporting interface for *Zenji*.

All other work conducted for the thesis was completed independently.

Funding Sources

This work did not receive any formal funding.

1. INTRODUCTION

For English-native learners of Japanese, literacy is a key part of the learning process, contributing to greater linguistic flexibility and a deeper understanding of Japan's culture. To achieve literacy, Japanese language learners must be familiar with the language's three scripts—two syllabic scripts, *hiragana* and *katakana*, and one logographic script, *kanji*.

The *kanji* writing system, ported over from China before the development of either of the syllabic scripts, has proven to be especially difficult for English speakers to learn. While there are a variety of reasons for this, the difficulty in learning the *kanji* largely stems from the sheer number of independent, similar, and complex characters that must be retained [1, 2].

To address this challenge, the most popular independent *kanji* learning strategies employ techniques such as mnemonic devices to aid in memorization [3]. However, the ability to hand-write *kanji* also has a significant impact on Japanese literacy [4]. While handwriting techniques are often a focus in a classroom setting, the practice of hand-writing characters for independent learners is de-emphasized, with popular commercial *kanji* learning platforms like *WaniKani* focusing on text-based recall for improved hardware compatibility [5]. This leads to reduced literacy relative to students with higher frequencies of *kanji* hand-writing practice [6].

To support the practice of hand-writing *kanji* in independent study, certain commercial tutoring systems exist which enable students to hand-write characters as a part of the usual spaced-repetition learning process [7]. However, such systems fail to provide the level of feedback that one would receive in the classroom, reducing student engagement with hand-writing *kanji* [8].

Zenji, a heuristic system of algorithms for the identification and correction of Japanese learners' common *kanji* handwriting mistakes, builds on former systems by integrating more advanced sketch analysis techniques into the environment of independent *kanji* study. The system provides Japanese as a Second Language (JSL) students a natural way to incorporate handwriting training into their preexisting routines for kanji memorization by generating numeric, visual,

and natural-language feedback on a per-stroke and whole-character basis. From testing, *Zenji* has proven reliable in its feedback across each of its key assessment metrics, ensuring that it gives JSL students the feedback they need to improve on their own. In this way, the *Zenji* system aims to supply a toolkit for Japanese learners and teachers, bridging the gap between the classroom and independent study and improving both student engagement and outcomes for *kanji* learning.

2. RELATED WORK

2.1 Intelligent Tutoring Systems

As an algorithmic system designed to provide personalized instruction to fit an individual student's needs, *Zenji* is classified as an intelligent tutoring system. This is a computational research field with an enormous amount of literature, and numerous approaches have been devised for different applications [9]. Work in this field includes BEETLE II for electricity and electronics tutoring [10], CRITS for programming tutoring [11], and ITALIC for English tutoring [12]. Many intelligent tutoring systems have methodological overlap with the *Zenji* system introduced in this paper, such as their frequent use of condition-action rule based reasoning and adaptations based on behavior in the learning path [9]. However, these applications lack direct parallel to this work due to their contributions being largely outside of the field of language learning and handwriting training.

2.2 Optical Character Recognition

Zenji is a system designed to recognize whether a character was written correctly and, if not correct, describe to the user how to amend their mistakes. Thus, at the most basic level, *Zenji* is a character recognition system. Character recognition has been a fruitful area of research since the mid-20th century, with famous algorithms such as Yann LeCun's LeNet-5 becoming staples of Optical Character Recognition (OCR) before the turn of the millennium [13]. Convolutional Neural Networks like LeNet-5 have also recently been applied with great effect to the recognition of Chinese characters (the basis for the *kanji* script) [14, 15, 16], with recent research focusing on developing zero-shot recognition models to handle the overwhelming requirement for training data in traditional approaches [17, 18, 19]. These models are highly capable when attempting to recognize handwritten Chinese characters, a task that generalizes to recognition of *kanji*. Unfortunately, while OCR models would be suitable for determining whether a student recalled the correct character in *kanji* learning, these models are ill-suited to generate feedback on how a user can improve

their writing.

2.3 Chinese/Japanese Handwriting Analysis

The *Zenji* system lies in the intersection of intelligent tutoring and handwriting analysis for East Asian scripts. This area of study has had significant recent developments, including the development of systems aimed at providing handwriting feedback for learners of the *kanji* script [1, 8, 20]. While these systems served as significant prototypes for *Zenji*, they largely focus on limited features such as stroke order and direction, while leaving important elements of a *kanji* character such as stroke interactions and structure unaccounted for.

The main inspiration for *Zenji* is Taele, Koh, and Hammond's 2020 *Kanji Workbook*, which looks at numerous features that will also be considered in *Zenji*. However, *Kanji Workbook*'s metrics were reconsidered for this paper based on classroom observations, and the system's implementation of star-based feedback and static background models limit its practical usefulness in an independent study setting [8]. *Zenji* aims to remedy these shortcomings by demonstrating effectiveness on a realistic kanji studying platform modeled after commercial solutions and implementing a more versatile feedback system that better emulates the classroom experience.

3. ZENJI SYSTEM

The *Zenji* assessment system takes user input and evaluates it on a set of proposed metrics. Prior systems have implemented similar paradigms, with the aforementioned *Kanji Workbook* evaluating student performance across ten metrics (stroke matches, validity, existence, order, direction, edits, length, closeness, speed, and symbol speed) [8]. The *Zenji* system represents an evolution of *Kanji Workbook*'s assessment criteria, using many similar metrics but introducing a number of unique metrics as well. Student performance is measured relative to model characters provided through the *KanjiVG* (*Kanji Vector Graphics*) project [21], the source for handwriting models in popular online Japanese dictionaries such as *JapanDict* and *Jisho.org* as well as popular commercial writing-based *kanji* studying apps [7, 22, 23].

3.1 Model Characters

The model characters were provided by the *KanjiVG* project. *KanjiVG* contains around 6,000 characters' vector graphics and notably all 2,136 current *Joyo Kanji* (the *kanji* designated by the government of Japan for official use), making it a suitable source for models in a *kanji* learning context [21].

KanjiVG SVG files also contain information about a *kanji*'s stroke groups, radicals, and variants, but *Zenji* only makes use of the SVGs' paths and stroke ordering [21]. The model SVGs are interpolated into a set of points and strokes to be stored in JSON format, similar to prior works in sketch analysis such as *LADDER* and *Kanji Workbook* [8, 24].

Before interpolation, each SVG is scaled to a canvas of 500×500 pixels. Once interpolated, The JSON files stored for each character contain the following information:

1. *Points:* Each point in the JSON file is a list of size 2, where the first index is a float x-value and the second is a float y-value. Points are each 10 pixels from each other starting at the beginning of the stroke and extending to the end. With the canvas size of 500×500 pixels, this produces approximately 50 points for a full-length stroke.
2. *Strokes:* Each stroke is a collection of points stored as a list. Points are ordered in the list, meaning the first point represents the start of the stroke and the last point represents the end.

3. *Geometry:* Each character contains a geometry attribute, which holds an ordered list of the character's strokes.

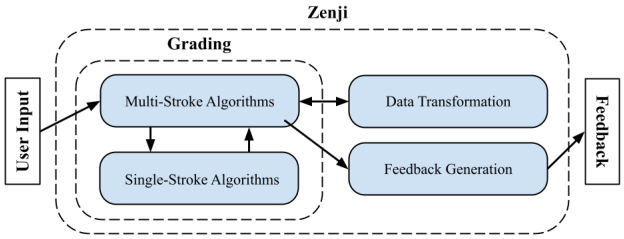

Figure 1: Zenji System Overview

3.2 Zenji System Overview

The Zenji system overview can be seen in Fig. 1. The general program flow consists of the following stages/layers:

3.2.1 User Input:

The user draws a character when prompted by the user interface. The character is recorded as an SVG and passed to the multi-stroke algorithms layer.

3.2.2 Multi-Stroke Algorithms:

The algorithms in this layer govern grading user input at the whole-character level, focusing on metrics like stroke order and number. Algorithms at this layer use the single-stroke algorithms as their base, altering input information and passing information back and forth to the data transformation and single-stroke layers as part of their optimization process. Once the multi-stroke algorithms complete, all metrics required for feedback generation have been evaluated. This information is passed to the feedback generation phase.

3.2.3 Data Transformation:

The data transformation layer interpolates and normalizes the user input, performing transformations to match the model character. Processed data and a metric based on the transformations

are passed down to the single stroke algorithms layer, and calculated metrics are passed up to the multi-stroke algorithms to be used for assessment and feedback generation.

3.2.4 Single-Stroke Algorithms:

The algorithms in this layer govern grading user input at the individual stroke level, focusing on metrics like stroke length, positioning, interactions, and angular matches. These algorithms serve as the base of the multi-stroke layer, so they may be called many times during the assessment process. Assessed metrics and certain key information are passed from this layer to the multi-stroke layer through the data transformation layer to be used for feedback generation.

3.2.5 Feedback Generation:

The feedback generation phase consumes the metrics and key information provided by the multi-stroke layer and largely calculated in the single-stroke layer. This phase determines and outputs the visual, numeric, and natural-language feedback to the user.

3.3 Multi-Stroke Algorithms

The multi-stroke algorithms layer of *Zenji*'s grading phase is responsible for assessing user input on metrics dealing with more than one input stroke. Key metrics in this category are stroke number (i.e., identifying missing or extra strokes) and stroke order (determining whether strokes were drawn in the temporal correct order, as indicated by the model characters' coordinates lists).

Unlike *Kanji Workbook* and many other prior works in this area, *Zenji* does not rely on users entering a character in a specific location on the screen [8]. Because of this, the point clouds representing the model character and the input do not have any inherent alignment with which to match points. This makes stroke matching a more complex problem that simple solutions like the Hausdorff distance or $P gesture classifier are ill-equipped to handle [25, 26]. To address this, *Zenji* employs a greedy breadth-first search overseer to explore stroke existences. The overseer has three modes of execution—one for extra strokes, one for missing strokes, and one for a correct number of strokes:

1. If a correct number of strokes (paths) are found:
    (a) Pass the path data to the data transformation layer for normalization and aspect ratio metric calculation.

(b) Use the $P gesture classification strategy to find a minimum-cost assignment of strokes, recording a metric for any discrepancies in stroke order [26].

(c) Pass the minimum-cost assignment to the single-stroke algorithms for grading.

(d) Pass the returned stroke grades and all metrics to the feedback generator.

2. Otherwise, if extra strokes are found:

   (a) Calculate each input path list with one path removed.

   (b) Use the process in step (1) to find the single input path removal with the highest number of passing strokes, breaking ties with average stroke grade.

   (c) Keep track of the best removed input path and use the best removal to continue with parts (b) and (c) until the correct number of strokes have been removed.

   (d) Pass the returned stroke grades and all metrics (including the removed input paths) to the feedback generator.

3. Otherwise, if missing strokes are found:

   (a) Calculate each model path list with one path removed.

   (b) Use the process in step (1) to find the single model path removal with the highest number of passing strokes, breaking ties with average stroke grade.

   (c) Keep track of the best removed model path and use the best removal to continue with parts (b) and (c) until the correct number of model strokes have been removed.

   (d) Pass the returned stroke grades and all metrics (including the removed model paths) to the feedback generator.

3.4 Data Transformation

The data transformation phase of *Zenji* is responsible for transforming user input so that it approximately matches the information in the model character's JSON interpolation data file. This layer consists of three main steps:

1. *Scaling:* The input SVG is scaled up or down using its bounding box and the bounding box of the model character, calculated from its point data. After this step, the height and width of the input and model SVGs will match. Our first metric is calculated here, based on the bounding box transformation. If our vertical scaling value is greater than our horizontal scaling value, the input was too wide for its height. If the opposite was true, the input was too tall for its width. We store the scaling ratio.

2. *Translation:* The scaled input SVG is translated such that its upper-left bounding box coordinate matches the upper-left coordinate if the model character. After this step, the input and model SVGs will be perfectly superimposed.

3. *Resampling:* Based on the number of points in the model character's interpolation data, we sample our paths to achieve the same number of total data points. If the user input is accurate, each input path will have the same number of points as the target. We store these values in nested lists that mirror the model's JSON structure.

Once the interpolation step is complete, we obtain properly transformed data which can be accurately used for assessment and feedback generation. We pass the coordinates object and our calculated scaling ratio metric back to the multi-stroke algorithms layer.

3.5 Single-Stroke Algorithms

The single-stroke algorithms layer of *Zenji*'s grading phase lies at the core of the assessment scheme. It is responsible for assessing user input on metrics for each individual stroke. Key metrics in this category are stroke length, angles, positioning, crosses, and intersections.

A numeric score for each stroke is calculated by finding a total error value—a linear combination which incorporates all of the error information returned by the five metrics—and subtracting it from a base score of 1. Since *Zenji*'s goal is to train handwriting without the use of a permanently-displayed model character, each metric must also have an acceptable dead zone where error does not detract from the stroke's score. These dead zones are largely arbitrary, but suitable values can quickly be found in practice by testing inputs. The individual metric information and numeric score for each stroke is then passed by this layer to the multi-stroke algorithms layer.

Certain algorithms used in this section, such as the positioning and length calculations, are similar to prior works. However, other algorithms (such as our angle calculation methodology) are novel contributions that support greater granularity in feedback generation. The metrics calculated in this layer are as follows:

3.5.1 Stroke Length:

This metric compares the rescaled SVG path lengths to the path lengths of the model SVGs, returning a ratio of input to model path length for each path.

3.5.2 Stroke Positioning:

This metric uses the interpolated point data to calculate a center of mass for each input and model stroke, returning the vector between the model and target centers. Since this stage would be

susceptible to one stroke significantly expanding the bounding box in some direction (and thereby shifting the center of every stroke), the center of mass of the entire model and input are transformed to match when these values are calculated. To account for a stroke being misplaced in a way that significantly alters the center of mass, if there exist any strokes with outlying positioning vectors, their data points are temporarily removed from both the input and model data and the remaining positioning vectors are recalculated with the new center of mass transformation. In this way, it is impossible for the stroke positioning metric to indicate that every stroke is poorly placed.

3.5.3 Stroke Angles:

This metric is based on the angular differences between the input stroke and the model stroke at each point along the model stroke's path. Since the input could have a different number of points than the model, these differences are calculated through two steps:

1. The angle between each point and its successor is calculated for both the input and the model.
2. For each angle in the model, a percentage representing its progress along the stroke is calculated. This value indicates which two input values to interpolate between, and a weighted average of those input values is found to estimate the input stroke angle at the correct position.

Once the angular values are found, a moving average is used to smooth both lists. This is done because calculating angular differences with the raw data is prone to very large discrepancies around the turning points in a model stroke, as users who cannot see the model are unlikely to make identically-placed turns in their input paths. After smoothing, the model angles and angular differences are returned.

3.5.4 Stroke Intersections:

This simple metric looks at the start and endpoint of each stroke, determining if they lie within a small margin of a point on any other path. This metric is calculated for both the model and user input and a comparison of the intersection matrices is used to return missing or extra intersections.

3.5.5 Stroke Crossings:

This metric checks sets of two consecutive points (not including the path endpoints) to see if they intersect with another set of two consecutive non-endpoints on another path. Similarly to

intersections, extra or missing paths are returned by computing the difference between the model and input metric matrices.

3.6 Feedback Generation

The feedback generation phase of the *Zenji* system transforms all of the previously-collected metrics and values into numeric, visual, and natural-language feedback for the user. Feedback generation based on the proposed metrics is performed as follows:

3.6.1 Numeric Feedback

The numeric feedback in the *Zenji* system primarily consists of an overall *kanji* score. This score is calculated based on the individual stroke grades and multi-stroke metrics and relies on a preset passing score. If a stroke grade is below the passing score, the overall *kanji* grade is immediately reduced by the distance between a full score and that passing score (i.e., for a passing score of 65%, a failing stroke reduces the overall grade by 35%). This logic also applies to strokes that were written in the wrong order, missing strokes, and extra strokes. The mean of the remaining stroke scores is then multiplied by the aspect ratio metric score and the maximum of the overall grade and 0 to obtain the final score.

While unconventional, this grading scheme was chosen so that users must accurately reproduce each stroke of the *kanji* in order to progress. As referenced in the introduction, many *kanji* are extremely similar, and an error in a single stroke could potentially produce an entirely different meaning. This grading scheme ensures that users learn the proper technique for each stroke while still providing the opportunity to improve after mastering the basic form.

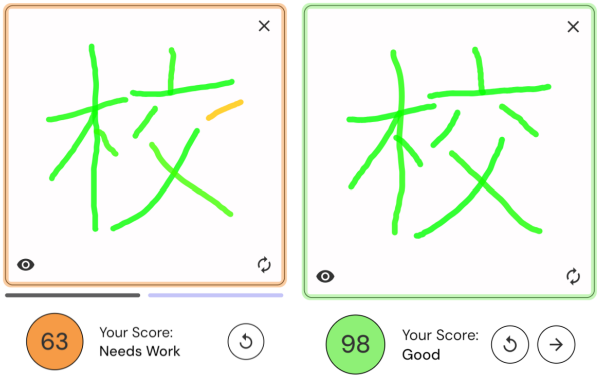

Figure 2: Numeric and visual feedback generated by Zenji

In Fig. 2, we see that this drawing of the character 校, when done correctly on the right, achieves an overall score of 98 (small errors and inconsistencies contribute to two lost points on average per stroke), whereas the incorrect direction of stroke 8 on the left drops the overall score by 35 points.

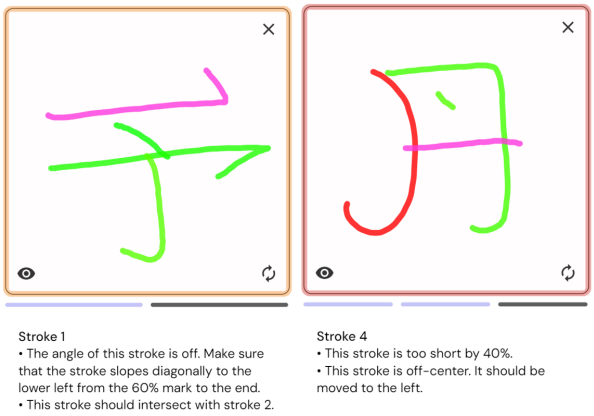

Figure 3: Per-stroke natural-language and visual feedback

3.6.2 Visual Feedback

The visual feedback in the *Zenji* system is primarily color-coding of strokes and overall scores. Strokes are color-coded from bright green (good) to bright red (bad) based on their individ-

ual scores. In Fig. 2, we see that the stroke of interest (stroke 8) is orange, since its angle does not align with the model. As is implied by the two dots above the overall score, our implementation of *Zenji* allows us to swipe to the right to see more details about why a stroke may have failed. When we swipe, the stroke of interest is highlighted in a unique purple color to let us know that the feedback we are to see is related to it (Fig. 3).

3.6.3 Written Natural-Language Feedback

The *Zenji* system generates written feedback on both a full-character and individual stroke basis. For individual strokes, generated feedback is based on the stroke's metrics, namely length, angles, positioning, intersections, and crosses. For entire characters, generated feedback is based on the multi-stroke algorithms' and data transformer's metrics for missing and extra strokes, stroke order, and aspect ratio.

In Fig. 3, we see four of the five possible categories for single-stroke feedback generation. On the left, the character 予 has been written with the right part of the first stroke in the wrong direction. In this instance, the greater granularity offered by calculating angles across the entire model stroke comes into play. Using the angles of the model stroke and the differences calculated in the single-stroke layer, we find that the discrepancy in stroke angle occurs around 60% of the way through and that the stroke should turn towards the lower left. Since we calculated our angle heuristic in this way, we are able to detect such an error and generate accurate and meaningful feedback for the user. The feedback also tells us that the stroke should intersect with stroke 2 (directly below it), a feature of the character that would have existed had we finished stroke 1 with the proper angle.

On the right, we see two more of the single-stroke metrics at play. While two strokes in this depiction of 丹 are incorrect, our user has swiped right twice to view information on the second of the two. This stroke should ideally be a flat horizontal line extending significantly beyond the body of the character in both directions. However, the user has drawn it somewhat small and squished to the right. Our feedback generation system recognizes that the character is clearly too short and that its center of mass is shifted to the right, providing that valuable insight to the user.

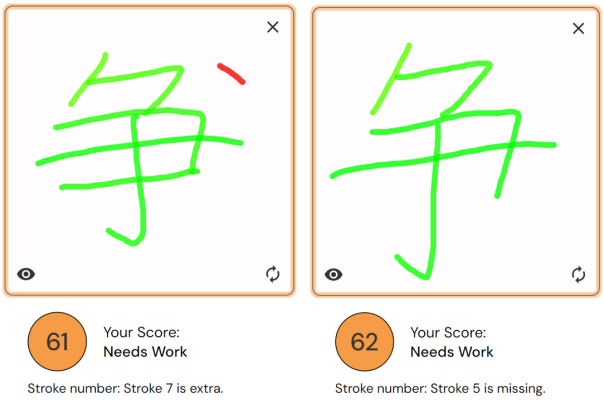

Figure 4: Missing and extra stroke feedback

In Fig. 4, we see a sample of the feedback generated by *Zenji* for missing or extra strokes. Since these are related to the stroke as a whole, they are included on the main page with the overall score. Even though the user is not drawing on a model background with a known location, Zenji is able to correctly identify the missing and extra strokes and provide that information to the user without impacting the other strokes' individual scores.

Figure 5: Stroke order feedback and scale/placement flexibility

Fig. 5 displays *Zenji*'s feedback generation for stroke order mistakes. In the figure, a user has flipped two strokes of 共. While this error is not visible to an observer, *Zenji* uses the

stroke matching information returned by the $P gesture matching to recognize this error, generate a message, and highlight the flipped strokes for the user [26]. In these images, we also see that Zenji's data transformation techniques allow for accurate assessment of user handwriting regardless of the location or scale of the user's input.

4. VALIDATION TESTING

Validation testing on the *Zenji* system used a random sample of 50 *Joyo Kanji* with a mean of 7.53 strokes and standard deviation of 2.76 strokes. Each kanji was tested twice for each criterion for a total of 100 data points per criterion. *Correct Input* testing occurred once with centralized input location and typical input size (the ideal case) and once with randomized input location and size (the adversarial case). To test mistakes, one stroke of the input kanji was randomly selected to be drawn with the specified error. For the second trial of the *kanji*, the same stroke was used, but a different organic erroneous input was produced. For the *Stroke Angle* criterion, the first input flipped one stroke (i.e., drew one stroke starting from the wrong end) and the second randomly rotated the stroke into an incorrect direction. For *Stroke Order*, tests randomly selected from two-stroke swaps, three-stroke cycles, and four-strokes double-pair swaps (when applicable).

In the following results, a *Correct* designation denotes the case where the mistake or lack thereof was detected and the expected feedback was generated. *Partial* denotes the case where the mistake was detected, but the input caused erroneous detection of some other mistake. For the *Intersections and Crosses* criterion, this category includes not detecting all mistakes for the designated stroke (e.g., a stroke loses a required intersection and gains an erroneous cross, but feedback only mentions the cross). An *Incorrect* designation denotes the case where a mistake was not detected or some other mistake was detected instead, generating incorrect feedback.

Table 1: Validation Testing Results

Input	**Correct**	**Partial**	**Incorrect**
Correct Input	96	0	4
Stroke Angle	95	4	1
Stroke Length	85	7	8
Stroke Position	96	2	2
Intersections and Crosses	84	6	10
Stroke Order	100	0	0
Stroke Number	96	0	4
Total	652 (93.14%)	19 (2.71%)	29 (4.14%)

4.1 Correct Input

Correct input was evaluated successfully in 96% of test cases. The system was more likely to make a mistake when grading small strokes with no clear point of reference on a larger structure, such as the first two (left) strokes of 冷. A human grader would recognize the higher difficulty of precisely placing these floating strokes and be relatively lenient, whereas *Zenji* currently has no such ability. Other issues were noticed with especially small inputs, where the technology of the drawing pad becomes a limiting factor and interpolation of the SVGs can be inconsistent. While not significant for most of our metrics, small differences in the path endpoint placements is important for the intersections and crossings algorithms, which are a main cause of failure on correct inputs.

4.2 Stroke Angle Metric

The stroke angle metric was evaluated correctly in 95% of test cases. The partially correct evaluations were most common when shifting the angle of a stroke significantly altered the input's bounding box, which could cause other strokes to be seen as incorrectly placed. Angle detection was highly accurate for flipping strokes, which is likely the most common mistake in this category

for beginner and intermediate *kanji* learners.

4.3 Stroke Length Metric

The stroke length metric was evaluated correctly in 85% of test cases. As seen in Table 1, both partially and fully incorrect feedback is relatively common in this category. In testing, it was observed that length alterations that significantly expanded or shrunk the *kanji*'s bounding box can cause stroke issues to be misinterpreted, especially leading to other strokes being read as too short or long (a *Partial* designation). Length issues that did not alter the bounding box or caused only slight changes were interpreted with very high accuracy.

4.4 Stroke Position Metric

The stroke position metric was evaluated correctly in 96% of test cases. Errors in this metric occurred when very long strokes were positioned in a way that significantly altered the *kanji*'s bounding box. Alterations of this type can cause large shifts in the *kanji*'s center of mass, leading to erroneous conclusions from the position algorithm. The quarantining system that recalculates the center of mass when an outlier is found creates a phenomenon where large mistakes in positioning are usually less error-prone than smaller ones. This metric was interpreted very accurately for small strokes regardless of whether they were positioned to alter the bounding box. This is a promising result, as learners are more likely to misplace small strokes that do not contribute as heavily to the overall *kanji* structure.

4.5 Intersections and Crossings Metric

The intersections and crossings metric was evaluated correctly in 84% of test cases, the lowest of any tested metric. The detection of stroke intersections is somewhat inconsistent between the target and model data due to path interpolation, which does not guarantee the inclusion of the closest point on one path to an endpoint of another. To remedy this, an intersection threshold must be carefully selected to weigh the trade-off between missing intersections and including close stroke endpoints that are not actually overlapped with another path.

Zenji was designed with use on learners' mobile devices in mind, which limits the precision with which one can expect *kanji* to be drawn. This further exacerbates the problem of finding a

hard definition of a stroke intersection. In *Zenji*'s grading scheme, stroke intersection errors were weighted very lightly in acknowledgment of this challenge. However, learners' knowledge gaps in this category are more likely to be crossing a stroke that should not be crossed or vice-versa, an evaluation that *Zenji* performs much more consistently.

4.6 Stroke Order Metric

The stroke order metric was evaluated correctly in 100% of the test cases. With input that would otherwise be evaluated correctly (i.e., avoiding the scenarios described in Section 4.1), *Zenji*'s implementation of the $P gesture classifier is extremely effective in evaluating the correct stroke order [26]. Future combined-mistake testing or user studies would reveal more about the accuracy of stroke order determinations in real-world use cases.

4.7 Stroke Number Metric

The stroke number metric was evaluated correctly in 96% of test cases. This is another metric that relies on the $P classifier [26]. Errors were most common when extra strokes had significant overlap with good input strokes or multiple extra strokes caused shifts in the bounding box. The latter of these conditions presents a weakness with our overseeing algorithm, which can only remove one of these erroneous strokes at a time and thus may not be able to find an effective match on the first iteration. However, evaluations were highly accurate for single-stroke errors and inputs without overlapping extra strokes, which likely cover the most common learning mistakes.

5. DISCUSSION AND FUTURE WORK

The *Zenji* system is able to effectively generate feedback for a variety of user inputs for each metric that it assesses. In validation, *Zenji* achieved an overall score of 93.14%, with a minimum metric score of 84% and maximum of 100%. Zenji innovates by avoiding reliance on fixed-template matching and introducing novel features such as granular angle and stroke interaction assessment. This enables the system to be used in a recall setting where seeing a background of the key character would run counter to the learner's goal.

However, *Zenji* is by no means a perfect system. In validation testing, limitations of the bounding box transforms led to reduced accuracy on the generation of angle, length, position, and stroke number feedback. The limitations of our interpolation technique and intended use case also significantly reduced the usability of intersection data. Since *Zenji* lacks knowledge of the overall structure of any *kanji*, the system lacks the ability to discern the relative importance of positional relationships between individual strokes, a drawback that even ideally set position error parameters cannot address. Finally, while fast, our choice for a stroke number overseer is vulnerable to miscalculations when multiple mistakes are present in a single input, a likely scenario for real user data.

Future work on alleviating some of these limitations could make *Zenji*'s successors far more versatile and accurate to real classroom feedback. *Zenji* is also a perfect use case for the integration of language models to generate less repetitive and more descriptive, natural sounding feedback for the user. A language model may, for example, be better able to giving the user feedback on correcting stroke order mistakes than our structured algorithmic approach. Thorough testing would be required to ensure that hallucinations do not significantly affect the accuracy of generated feedback.

In the near future, multiple-mistake feedback generation testing could provide more detail on the power of integrated algorithms for gesture classification and the resilience of our single-

mistake validation results to adversarial conditions. A large-scale longitudinal study on the outcomes of English-speaking students of Japanese is also a natural next step in demonstrating the effectiveness of the system.

6. CONCLUSION

In this paper, we introduced *Zenji*, a system of heuristic algorithms for assessing the *kanji* handwriting of students of the Japanese language and generating feedback on commonly-seen mistakes. The system builds on prior work by implementing an algorithmic design that could effectively be incorporated into a recall-based *kanji* studying application through new multi-stroke analysis techniques with no location or scale assumptions. The system assesses novel metrics for user input such as granular smoothed angle comparison and stroke interactions, enabling it to provide more structured and specific feedback than previously possible on a wide range of *kanji* features. In validation testing, *Zenji* achieved an accuracy score of 93.14%, demonstrating its ability to reliably respond to varied user inputs on each of its key metrics. *Zenji* innovates by building on its unprecedented level of metric collection to generate natural-language feedback and specific handwriting suggestions for users, an experience that has historically only been available in the classroom setting.

REFERENCES

[1] P. Taele and T. Hammond, "Hashigo: A next-generation sketch interactive system for japanese kanji," in *Twenty-First IAAI Conference*, 2009.

[2] H. Rose and L. Harbon, "Self-regulation in second language learning: An investigation of the kanji-learning task," *Foreign Language Annals*, vol. 46, no. 1, pp. 96–107, 2013. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/flan.12011

[3] J. W. Heisig, *Remembering the kanji*. University of Hawai'i Press, 2011.

[4] S. Otsuka and T. Murai, "The unique contribution of handwriting accuracy to literacy skills in japanese adolescents," *Reading and Writing*, vol. 37, no. 5, pp. 1183–1208, 2024.

[5] Tofugu Team, "Wanikani: A kanji learning application by tofugu," 2024, accessed: 26 November 2024. [Online]. Available: https://www.wanikani.com/

[6] Y. Butler, "Kanji acquisition among language minority students in japan: A comparative study of japanese-as-a-second-language students born in japan," *Working Papers in Educational Linguistics*, 4 2011.

[7] Luli Languages LLC, "Learn japanese! - kanji [mobile application]," 2016, published: February 2016; accessed: 26 November 2024. [Online]. Available: https://apps.apple.com/us/app/learn-japanese-kanji/id1078107994

[8] P. Taele, J. I. Koh, and T. Hammond, "Kanji workbook: a writing-based intelligent tutoring system for learning proper japanese kanji writing technique with instructor-emulated assessment," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 08, 2020, pp. 13 382–13 389.

[9] E. Mousavinasab, N. Zarifsanaiey, S. R. Niakan Kalhori, M. Rakhshan, L. Keikha, and M. Ghazi Saeedi, "Intelligent tutoring systems: a systematic review of characteristics, applications, and evaluation methods," *Interactive Learning Environments*, vol. 29, no. 1, pp. 142–163, 2021.

[10] M. Dzikovska, N. Steinhauser, E. Farrow, J. Moore, and G. Campbell, "Beetle ii: Deep natural language understanding and automatic feedback generation for intelligent tutoring in

basic electricity and electronics," *International Journal of Artificial Intelligence in Education*, vol. 24, pp. 284–332, 2014.

[11] P. Mohammed and P. Mohan, "Dynamic cultural contextualisation of educational content in intelligent learning environments using icon," *International Journal of Artificial Intelligence in Education*, vol. 25, pp. 249–270, 2015.

[12] D. P. Vinchurkar and M. Sasikumar, "Intelligent tutoring system for voice conversion in english," in *2015 IEEE 15th International Conference on Advanced Learning Technologies*. IEEE, 2015, pp. 314–316.

[13] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[14] P. Melnyk, Z. You, and K. Li, "A high-performance cnn method for offline handwritten chinese character recognition and visualization," *soft computing*, vol. 24, no. 11, pp. 7977–7987, 2020.

[15] X. Liu, B. Hu, Q. Chen, X. Wu, and J. You, "Stroke sequence-dependent deep convolutional neural network for online handwritten chinese character recognition," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 11, pp. 4637–4648, 2020.

[16] Y. Li and Y. Li, "Design and implementation of handwritten chinese character recognition method based on cnn and tensorflow," in *2021 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*. IEEE, 2021, pp. 878–882.

[17] Z. Cao, J. Lu, S. Cui, and C. Zhang, "Zero-shot handwritten chinese character recognition with hierarchical decomposition embedding," *Pattern Recognition*, vol. 107, p. 107488, 2020.

[18] J. Chen, B. Li, and X. Xue, "Zero-shot chinese character recognition with stroke-level decomposition," *arXiv preprint arXiv:2106.11613*, 2021.

[19] D. Gui, K. Chen, H. Ding, and Q. Huo, "Zero-shot generation of training data with denoising diffusion probabilistic model for handwritten chinese character recognition," in *International Conference on Document Analysis and Recognition*. Springer, 2023, pp. 348–365.

[20] T. Chu, P. Taele, and T. Hammond, "Supporting chinese character educational interfaces with richer assessment feedback through sketch recognition." in *Graphics Interface*, 2018, pp. 50–57.

[21] U. Apel, "Kanjivg: Kanji vector graphics project," 2024, accessed: 26 November 2024. [Online]. Available: https://kanjivg.tagaini.net/

[22] Jisho Team, "Jisho.org: Japanese dictionary," 2024, accessed: 26 November 2024. [Online]. Available: https://jisho.org/

[23] JapanDict Team, "Japandict: Japanese dictionary," 2024, accessed: 26 November 2024. [Online]. Available: https://www.japandict.com/

[24] T. Hammond and R. Davis, "Ladder, a sketching language for user interface developers," *Computers & Graphics*, vol. 29, no. 4, pp. 518–532, 2005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0097849305000865

[25] W. Rucklidge, *Efficient visual recognition using the Hausdorff distance*. Springer, 1996.

[26] 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*, 2012, pp. 273–280.