A Multi-fonts Kanji Character Recognition Method for Early-modern Japanese Printed Books with Ruby Characters

Taeka Awazu[1], Manami Fukuo[1], Masami Takata[2] and Kazuki Joe[2]
[1]*Graduate School of Humanities and Sciences, Nara Women's University, Nara, Japan*
[2]*Academic Group of Information and Computer Sciences, Nara Women's University, Nara, Japan*

Keywords: Character Recognition, Character Clipping, Genetic Programing, Early-modern Japanese Printed Books.

Abstract: The web site of National Diet Library in Japan provides a lot of early-modern (AD1868-1945) Japanese printed books to the public, but full-text search is essentially impossible. In order to perform advanced search for historical literatures, the automatic textualization of the images is required. However, the ruby system, which is peculiar to Japanese books, gives a serious obstacle against the textualization. When we apply existing OCRs to early-modern Japanese printed books, the recognition rate is extremely low. To solve this problem, we have already proposed a multi-font Kanji character recognition method using the PDC feature and an SVM. In this paper, we propose a ruby character removal method for early-modern Japanese printed books using genetic programming, and evaluate our multi-fonts Kanji character recognition method with 1,000 types of early-modern Japanese printed Kanji characters.

1 INTRODUCTION

The National Diet Library (NDL) in Japan keeps about 320,000 books dating from the Meiji era to the first half of Showa era (AD1868-1945). These books cover a broad range of books including philosophies, literatures, histories, technologies, natural sciences, etc. Many books are out of print although they are scientifically precious data. Therefore, in the NDL, library materials have been and will be handed down to the future generations as cultural assets by digitalizing the archives of the materials for wider and easier use. The electronic library service is provided as eDigital Library from the Meiji Eraf. At the NDL Web site, user can search the materials by setting up detailed items, such as title, author, publisher, and publication year. However, since the text of early-modern Japanese printed books is exhibited as image, full-text search is essentially impossible. In order to perform the full-text search, it is required extracting texts from images. Although there are considerably many early-modern Japanese printed books that are scientifically precious, the text extraction of hundreds of thousands of books is impossible in budget if it is performed by hand. There is no existing research on character recognition for early-modern Japanese printed books.

Based on the above backgrounds, we enlist cooperation from the NDL to have started the research project of automatic text extraction for eDigital Library from the Meiji Eraf. In extracting text data from image data of early-modern Japanese printed books, when existing OCRs are applied to the image data, the recognition rates are too low to be practical. We have reported that the recognition of the printing type segmented from the early-modern Japanese printed books is possible using the method of handwritten Kanji character recognition (C. Ishikawa and Joe, 2009). In early-modern Japanese printed books, it is naturally inferred that used for each publisher adopts a different typography. However, even if within the same publisher, it has also been reported that typography differs by age (M. Fukuo and Joe, 2012). For these reasons, we use the method of handwritten Kanji character recognition for text extraction of early-modern Japanese printed books.

In order to achieve automatic text extraction from image data of such old books, we have to automatically clip each character for its recognition. However, the erubyf system, which is peculiar to Japanese books, gives a serious obstacle against the clipping. The ruby system has been developed for low educated people to teach the way to read difficult Kanji characters. While there are six thousands of@Kanji characters used in Japan, all Japanese do not completely learn them. Low educated people can read only hundreds of Kanji characters. At the same

Awazu T., Fukuo M., Takata M. and Joe K..
A Multi-fonts Kanji Character Recognition Method for Early-modern Japanese Printed Books with Ruby Characters.
DOI: 10.5220/0004825306370645
In *Proceedings of the 3rd International Conference on Pattern Recognition Applications and Methods* (ICPRAM-2014), pages 637-645
ISBN: 978-989-758-018-5

time, Japanese characters include phonetic characters called Katakana and Hiragana distinct from Kanji characters, and all Japanese can read the phonetic characters. The ruby system gives the way of pronouncing difficult Kanji characters by locating small phonetic characters on the right side of each Kanji character.

It is well known that failure of character clipping due to existence of ruby characters reduces the character recognition rate for conventional OCRs. Since the early-modern Japanese printed books are not given standard typography such as current books, the character recognition rate of conventional OCRs for the old books is extremely low in general. As far as we know, any ruby removal methods for early-modern Japanese printed books have not been studied. As for existing methods to remove ruby characters from current books with standard typography, there are two main methods (N. Stamatopoulos and Gatos, 2009) (Fletcher and Kasturi, 1988): (1) Separating ruby characters linearly using density histogram and (2) separating ruby characters using circumscription rectangles. Both methods assume the standard typography for the target books. Otherwise, a lot of parts of a main Kanji character would be strongly connected with the corresponding ruby characters, and valleys of the histogram do not appear clearly. Therefore, good recognition results cannot be obtained in the case of (1). In the case of (2), strongly connected ruby characters to a main Kanji character would make it very difficult to construct a rectangle just for the main Kanji character, and ruby character removing becomes very difficult.

In this paper, we propose a ruby character removal method for early-modern Japanese printed books. We classify these books into several categories to generate ruby character removal filters for each category using the genetic programming (Koza, 1992). We use the following two methods for the classification. One is the classification using the information of publishers and publication ages added to the books. The other is the classification using the feature acquired from booksf images. In order to investigate the difference of these methods, we perform comparative experiments for ruby character removal.

Furthermore, in order to absorb various fonts of early-modern Japanese printed books, we perform recognition experiments with 1,000 kinds of Kanji characters from the old books using an off-line handwritten Kanji character recognition method with the PDC feature (N. Hagita and Masuda, 1983). By Japanese Industrial Standards (JIS), 2,965 Kanji characters that are frequently used are defined as the JIS level-1 Kanji character set that includes most of current usual documents. We use 1,000 kinds of Kanji characters from the JIS level-1 Kanji character set for our experiments.

The rest of the paper is organized as follows. Section 2 gives the description about fonts and the ruby system found in early-modern Japanese printed books. In section 3 we introduce existing methods for character clipping and their application to the ruby character removal. Our method of the ruby character removal using genetic programming and experiment results are presented in section 4. Section 5 gives the description about character clipping, off-line handwritten Kanji character recognition, and recognition experiments.

2 RUBY AND FONT OF EARLY-MODERN JAPANESE PRINTED BOOKS

Japanese letter consists of three kinds of characters: Hiragana, Katakana, and Kanji. The first two character sets are syllabaries and include about 70 phonetic characters while the last character set is an ideogram and include more than six thousands characters. Sentences in Japanese books are usually printed vertically using the above three character sets. Ruby is a system for low educated Japanese to support pronouncing difficult Kanji characters written in the books by locating small Hiragana or Katakana characters at the right side of difficult Kanji characters. We call the small phonetic characters and the (big) Kanji characters Ruby characters and parent characters, respectively. Most Japanese books adopt the ruby system, and currently most books are generated in a desktop publishing system, where the standard of fonts and ruby is defined by JIS. Early-modern Japanese printed books are published from the Meiji era (1868-1912) to the middle of the Showa era (1926-1980) with typographical printing. Since there are no standard typographies by the middle of the Showa era, various fonts and ruby systems are used in early-modern Japanese printed books. Figure 1 shows one of the current fonts and two fonts in early-modern expressing the same Kanji. When a ruby system is used in a

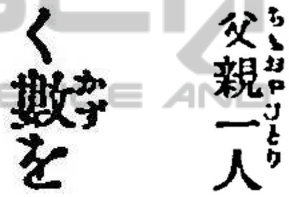
Figure 1: A current font and two old fonts expressing the same Kanji.

book, ruby characters are located at the right side of the corresponding parent Kanji character so that the

outer frames of the ruby characters contact with the outer frame of the parent Kanji character. Figure 2 shows an example of ruby character location.

Figure 2: An example of the positions of ruby.

The ruby characters found in early-modern Japanese printed books tend to be located to their parent Kanji characters with extremely close. In fact, ruby characters are often connected with their parent Kanji characters with the bleeding ink. In addition, there are many distorted ruby characters because of poor fonts. Figure 3 (a) shows an example that a ruby character connects with the parent Kanji character in an early-modern Japanese printed books, and (b) shows an example of distorted ruby characters.



(a) (b)

Figure 3: (a) An example of a ruby character connecting with the parent Kanji character, (b) An example of distorted ruby characters.

In the case of current books, it is possible to linearly separate ruby characters from parent Kanji characters by row because there is a constant distance between the ruby and parent characters defined by JIS. However, in the case of early-modern Japanese printed books, it is difficult to linearly separate ruby characters because of the above mentioned extremely narrow space and distorted ruby characters.

3 EXISTING METHODS APPLICABLE TO RUBY REMOVAL

As far as we know, there is no research to remove ruby characters for early-Modern Japanese printed books. In this section, we explain existing methods that are applicable to ruby character removal using character clipping for Japanese documents.

3.1 Black Pixel Projection Histogram

In black pixel projection histogram (N. Stamatopoulos and Gatos, 2009), characters are separated by cutting at turning points of the cross direction histogram. Figure 4 shows the black pixel projection histogram of a Kanji character.

Figure 4: Black pixel projection histogram.

Most Kanji characters are constructed with several parts. When a Kanji character has a space between the upper part and the lower part, the Kanji character may be divided as two Kanji characters. Figure 5 shows an example of such cases.

Figure 5: An example of a Kanji character separated as two Kanji characters.

When we use this method for ruby character removal, we linearly separate parent Kanji characters and ruby characters by cutting at turning points of the lengthwise direction histogram. However, it is difficult to separate them at a right position when a ruby character is strongly connected with the parent Kanji character.

3.2 Circumscription Rectangles

In this method, at first, a circumscription rectangle is generated by labeling strongly connected black pixels by eight direction neighborhoods. Then, the character divided with several small rectangles is segmented by the circumscription rectangle generated with duplicated small rectangles. Figure 6 (a) shows a Kanji character divided into several small rectangles, and (b) shows the character surrounded by the circumscription rectangle. Figure 7 shows a rectangle of a parent Kanji character including a ruby character.

(a) (b)

Figure 6: (a) A Kanji character divided with several rectangles, (b) A Kanji character surrounded by one.

Figure 7: A rectangle including a parent Kanji and a ruby character.

When we apply this method to ruby character removal, we cannot remove any ruby characters from

the parent character surrounded with the same circumscription rectangle.

4 RUBY CHARACTER REMOVAL FILTERS

Using genetic programming (GP), we propose a method to automatically generate approximate formulas expressing the boundary between the parent Kanji and the ruby characters.

4.1 Possible Methods

We surmise that the feature of ruby characters used in typography differs by publisher and age. The books with the same feature are classified, and ruby character removal filters are generated for each class. In the early-modern Japanese printed books, there are many strongly connected components with parent Kanji characters as well as frequent distortions of ruby characters by bled ink and poor quality of typography. It is difficult to separate the ruby characters just using histogram. We separate the ruby and the parent Kanji characters using machine learning. The commonly used machine learning includes SVM, neural network and genetic programming. In the case of SVM, when we separate a ruby and a parent Kanji characters, these two characters are classified using the feature vectors of these characters. For example, possible feature vectors are the gravity centers of the connected black pixels of the ruby and the parent Kanji characters. However, the premise of this method is that the ruby and the parent Kanji characters are separated. In early-modern Japanese printed books, ruby characters are often connected with their parent Kanji characters caused by bled ink and/or the poor quality of paper. For this reason, it is concluded that the method is not suitable for separating the ruby characters from the parent Kanji characters. Neural network also has the same problem. Therefore, applying to the unknown data in where ruby characters are connected with their parent Kanji characters, we believe that the best solution is to generate approximate formula presenting the boundary between ruby and the parent Kanji characters. In this paper, we generate such approximate formulas using genetic programming.

4.2 Procedure

Figure 8 shows the flow of the proposed method. The details are explained as follows.

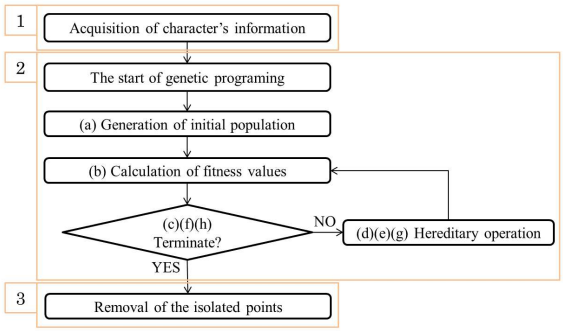


Figure 8: Flow of the proposed method.

1. Estimate the coordinate positions of the parent characters with ruby characters by row for original image and estimate the widths of the parent characters with ruby characters.
2. Generate formulas for ruby character removal with giving the training set to a GP.
   (a) Generate an initial population.
   (b) Calculate the fitness values using the positional information and the width estimated in (1) as terminal nodes.
   (c) Terminate the learning procedure when predefined condition is satisfied.
   (d) Cross a half of the population by the roulette selection.
   (e) Mutate individuals selected in random.
   (f) Calculate the fitness values.
   (g) Delete the individuals with low fitness values, and generate new individuals.
   (h) Go to (2c).
3. Apply the resultant formulas to separate parent Kanji and ruby characters, and remove the isolated pixels by a median filter.

In (1), the coordinate position and the average width of each row including Kanji and ruby characters are obtained from the training set, which consists of original images and target images. The original images include ruby characters, and the target images are generated by removing ruby characters by hand. Both images are binarized data.

In (2), we generate ruby removal formulas using GP learned with the values of (1). Figure 9 shows a variable x as a terminal node.
On the original image, we search the leftmost black pixel to get its coordinate position, and draw a straight line in the row direction from the pixel as the x-axis. The y-axis is a crosswise straight line on the basis of the upper-end of Kanji characters with ruby characters. The generated formulas are $y =$ (mathematical expression using terminal nodes), and applied to the

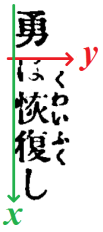


Figure 9: Variable x as a terminal node.

rows including ruby characters. In the GP, the composition of the individual is as follows. Tree nodes include four arithmetic operations as binary operators and absolute value and trigonometric (sin, cos) functions as unary operators. Terminal nodes include constant numbers of 1-9, the widths and the coordinate positions estimated in (1). The width of each character is the same value in the same row. In addition, variable x represents a lengthwise coordinate position of a Kanji character with ruby characters.

In (2a), an initial population is generated. An individual is a mathematical expression in a form of the tree structure consisting of terminal nodes and non-terminal nodes. A given number of individuals are generated, here we say N.

In (2b), fitness values are calculated. The fitness f is the agreement rate of pixel values between the target image and the output image which is deleted the ruby characters of the original image by generated formula. The calculation range of the fitness is not the whole image but the right half of the image because the ruby characters are in the right side. Figure 10 shows the whole range of the original image for fitness calculation. The right side of Kanji and ruby characters in Figure 10, which is surrounded by a red box, is the range for fitness calculation.

Figure 10: Range of the original image for fitness calculation.

In (2c), the predefined conditions are that the fitness value reaches 1 or the number of generations exceeds a given number.

In (2d), crossover is performed between selected parent individuals. The elite selection tends to converge in local optimum solutions by losing variety. The random selection has a problem that evolution of individuals is to be hard. The roulette selection maintains the variety of individuals because individuals are randomly chosen among some level of fitness values. Thus, we use the roulette selection. Probability p_i to choose individual i is defined as (1).

$$p_i = \frac{f_i}{\sum_{k=1}^{N} f_k} \quad (1)$$

A node for crossover is randomly selected among two parent individuals. The crossover is to exchange the partial tree structures below the selected node.

In (2e), mutation is performed between randomly selected individuals. A node of a tree structure is randomly selected and the information of the node is replaced with other information. At this point, the integrity of the node is maintained.

The fitness values of the next generation generated by genetic operation are calculated as same above in (2f), and a half of individuals with low fitness values is deleted in (2g). (2c)-(2g) are repeated until predefined conditions are satisfied.

In (3), the generated formulas are applied to separate parent Kanji and ruby characters, and it sometimes creates isolated pixels, which are easily removed using appropriate filters such as a median filter. Figure 11 shows an example image of isolated pixels.

Figure 11: Isolated pixels.

Usually such isolated pixels are enough small to be distinguished with parent Kanji characters. When such isolated pixel area has less than 10 pixels, it must be removed to prevent false recognition.

4.3 Classification of Early-modern Japanese Printed Books

Since there are a wide variety of fonts found in early-modern Japanese printed books, it is extremely difficult to generate universal ruby character removal formula applicable to all books. Therefore we try to divide these books into appropriate classes to generate a ruby character removal formula for each class.

4.3.1 Classification by Publisher and Year

It is obvious that fonts used in early-modern Japanese printed books are different for each publisher and year. We classify them by publisher and year. In this paper, we investigate three publishers:@Shunyodo, Hiyoshido, and Shinshindo@while we select three

periods of years: middle of the Meiji era (1883-1897), late of the Meiji era (1898-1912), and the Taisho era (1912-1925). In total, we have nine classes such as gShunyodo in the middle of Meijih. Training sets are prepared for each class with the total training set size of 10, 50, 100, 200, 300, 400, and 500 rows. When the training set size is more than 100 rows, no significant difference in fitness values are observed. Therefore, we use 100 rows (each 10 rows from 10 books) for each learning phase.

In this method, the parameters for genetic programming include the number of individuals, the number of possible generations, the crossover rate and the mutation rate. When the number of individuals is varied in 1,000 from 1,000 to 5,000, the fitness values of more than 3,000 individuals are almost converged. Therefore we use 3,000 individuals. As for other parameters, the upper limit of the number of generation is 200, the crossover rate is 0.8 and the mutation rate is 0.2 for the empirical reason.

We perform experiments 10 times for each class. Table 1 shows the best agreement rate of pixel values for each class between the training image and the images that ruby characters have been removed from.

Table 1: The best agreement rate for each classi%j.

	Middle Meiji Era	Late Meiji Era	Taisho
Shunyodo	98.8	98.9	98.8
Hiyosido	98.0	98.5	97.5
Shinshindo	98.5	98.6	98.5

In all classes, the agreement rates are around 98%. Furthermore, we compare the ruby character removal ratios by the proposal method with the linearly separating method using black pixel projection histograms. The cutting positions on the black pixel projection histogram are decided using a discriminant analysis method. The ruby removal ratio of the average of 9 classes is about 98.5% by the proposal method and is about 82.9% by the linearly separating method, respectively. Therefore, the result means that the proposal method is superior to the linearly separating method using black pixel projection histograms. Formulas (2) are an example generated with gShunyodo in the middle of the Meiji erah.

$$\begin{aligned} y \;=\; & ((8/3)+((\text{width average}-(\cos((2*\pi \\ & *x/(((4-(\cos((2*\pi*x/((\sin((2*\pi*x/ \\ & (((5+3)/2))-\pi))/2))-\pi/2))/1))/2)) \\ & -\pi/2))/(8/3)))-(\cos((2*\pi*x/ \\ & (((\text{width average}+4)/2))-\pi/2))/(7/5)))) \end{aligned} \quad (2)$$

Figures 12 show the curves representing formula (2) with the images that ruby characters have been removed from.

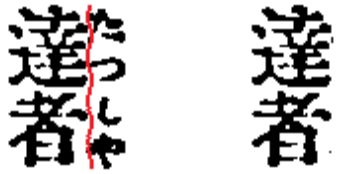


Figure 12: The curve by formula (2) and the ruby character removal result.

We have investigated publishers for early-modern Japanese printed books and found that the number of publishers exceeds ten thousands. Most of them are so small that some printing offices seem to be shared for use by many small publishers. It means that publisher based classification is not efficient when it is applied to all early-modern Japanese printed books since a huge number of publishers would make the classification extremely difficult.

4.3.2 Classification by Row Characteristic

Not using the information of publisher and year added to the books, we make use of characteristic of rows calculated from the books. We take notice the ratio between width and height of Kanji characters. It is calculated with the averages of widths and heights of Kanji characters in each row. Let f represent width/height, and we classify the books using f. Training sets are 900 rows in total as same as 4.2.1. Values of f are approximately between 1.4 and 1.8, where most values are between 1.5 and 1.7. We divide the range f values into eleven intervals with overlapping: [-:1.4], [1.35:1.45], [1.4:1.5], [1.45:1.55], [1.5:1.6], [1.55:1.65], [1.6:1.7], [1.65:1.75], [1.7:1.8], [1.75:1.85], and [1.8:-]. Although the classifications for [-:1.4], [1.35:1.45], [1.7:1.8], [1.75:1.85], and [1.8:-] do not have 100 rows of training set, they are at least over 50 rows and we judge they do not affect the experiment so much. Table 2 shows the best agreement rate between target images and images after ruby character removal for each class.
The agreement rates are not lower than 99% for all classes. By scrutinizing all the generated formulas closely, it turns out that the generated formulas for the classes of [-:1.4], [1.35:1.45], [1.4:1.5], [1.45:1.55], [1.5:1.6], [1.55:1.65] and [1.6:1.7] are the same. Formula (3) is the mathematical expression after the scrutiny.

$$y \;=\; \text{width average}+6 \quad (3)$$

By contrast, the formulas generated for classes of [1.65:1.75], [1.7:1.8], [1.75:1.85], and [1.8:-] are

Table 2: The best agreement rates for each class (%).

[-:1.4]	99.0
[1.35:1.45]	99.0
[1.4:1.5]	99.0
[1.45:1.55]	99.0
[1.5:1.6]	99.0
[1.55:1.65]	99.0
[1.6:1.7]	99.0
[1.65:1.75]	99.2
[1.7:1.8]	99.2
[1.75:1.85]	99.0
[1.8:-]	99.0

different. For example, the formula generated for [1.7:1.8] is shown below.

$$\begin{aligned} y = {} & (8-((\text{width average}/8)-(\text{width average}- \\ & (5/(\cos((2*\pi*x/(((8*5)/2))-\pi/2))- \\ & (\text{width average}-((4/(6*\cos((2*\pi*x/ \\ & ((\cos((2*\pi*x/(((8*5)/2))-\pi/2))/2)) \\ & -\pi/2))))/\text{width average})))))))) \end{aligned}$$

As the result, early-modern Japanese printed books can be classified into four classes of [-:1.65], [1.65:1.75], [1.7:1.8], and [1.8:-]. Therefore, we conclude that the width/height based classification is superior to the publisher and year based classification.

5 KANJI CHARACTER RECOGNITION AND EXPERIMENTS

When we apply existing OCRs to early-modern Japanese printed books without removing ruby characters, the recognition rate does not reach even 50%. Applying to 4,500 rows used in the previous section, the recognition rate is about 80%. Since existing OCRs have been developed under the assumption that the target fonts are officially defined, the recognition rate for early-modern Japanese printed books is extremely low. Furthermore, when the function of ruby character removal is not implemented on the existing OCRs, the recognition rate is worse. Therefore, in this paper, the ruby character removal is applied with the method of previous section, and we perform Kanji character recognition using off-line handwritten Kanji character recognition with the PDC feature and SVM.

5.1 Character Clipping

A Japanese character consists of several strongly connected components in many cases. Figure 13 shows a Hiragana character which consists of two components.

Figure 13: A Hiragana with two components.

Because of separated components in multiple, it is difficult to clip a character just using the circumscription rectangle method described in subsection 3.2. However, we can clip such a character by integrating the components. The conditions for the integration are as follows.

- The distance between two vertically located components is 0.3 times less than the vertical distance between the corresponding two adjacent characters.
- The distance between two horizontally located components is 0.2 times less than the average width of the characters in the row.

The average width is explained in subsection 4.1. In this experiment, we use 10,526 characters from 4,500 rows. The number of characters that are not correctly clipped is 2,182, and the clipping rate is about 98.0%. The main reason for the failure is integration errors for vertically located components. Especially, the integration errors are found in Kanji numeral characters of gtwoh and gthreeh. Figure 14 shows the Kanji numerals of gtwoh and gthreeh.

Figure 14: Kanji numerals of 2 and 3.

5.2 Multi-Fonts Kanji Character Recognition

In this subsection, we give the overview of our multi-fonts Kanji character recognition method presented in (C. Ishikawa and Joe, 2009). The flow of our recognition method is as follows. First, images containing clipped characters are prepared for training data, and several pre-processes such as binarization, normalization and noise reduction are applied. From each pre-processed image, the PDC (Peripheral Direction Contributivity) (N. Hagita and Masuda, 1983) feature is extracted to generate feature vectors. Each feature vector is labelled with the category according to the kind of the clipped Kanji character. The feature vectors as training data are given to an SVM to generate a dictionary that is used for the recognition by the SVM. The SVM learns separated hyper-planes in the PDC feature vector space with the training data. The trained SVM can classify test data according to the separated hyper-planes.

5.2.1 Experiments

To validate that our ruby character removal filter improves the multi-fonts Kanji character recognition considerably, we perform some experiments. We use 450 titles from 3 publishers to get 1,000 Kanji characters clipped by the method described in 5.1, and the number of Kanji character types is more than three thousands. The number of character images for each Kanji type is less than ten. Since it is necessary to divide the Kanji character data into training data and test data for each Kanji type, we use 1,000 types of Kanji characters (total 8,448 characters) that have more than five images for each type. The ratio of the number of training data to the total data is varied from 10% to 60% for each Kanji character type. We verify the recognition rate difference by the ratio of training data. Among the 1,000 types of Kanji characters, each of 796, 181 and 23 types has less than 10 (average 8.98), between 10 and 50 (average 31.75), and more than 50 (average 88.18) images, respectively. Table 3 shows the experimental results.

Table 3: The recognition rates with varying the rate of training data and the number of images for each Kanji character set (%).

The ratio of training data	Total	The number of images for each Kanji type		
		less than 10	between 10 and 50	more than 50
10%	89.65	75.06	93.28	99.33
20%	94.66	87.89	96.69	99.89
30%	96.19	91.37	99.43	99.82
40%	96.50	92.38	99.08	99.89
50%	97.14	94.07	99.54	99.89
60%	97.18	94.09	99.77	99.96

The row of total shows that recognition rate goes up as the ratio of training data increases. The row of gless than 10h shows that it gives much influence on the total recognition rates. In this case, when the ratio of training data is low, the number of training data is extremely small. Therefore, it is difficult to absorb the difference in the features of many fonts, and it leads the low recognition of 75%. However, the recognition rate reaches 94% when the ratio of training data is more than 50%. In the case of gbetween 10 and 50h, the recognition rate of 99% is achieved with 30% of training data. Moreover, in the case of gmore than 50h, just 10% of training data is enough to get the recognition rate of 99%. Figure 15 shows an example where several different fonts are correctly recognized. The images failed in recognition tend to be crushed by blot of ink or be blurred. Figure 16 shows a misrecognition example.

Figure 15: An example of the correct recognition of different fonts.

Figure 16: A misrecognition example.

The experimental results show that the recognition rate of over 99% is achieved with at least nine images as training data, and the difference in the features among various fonts can be absorbed by our method.

6 CONCLUSIONS

In this paper, we proposed a ruby character removal method for early-modern Japanese printed books using genetic programming, and we performed recognition experiments of 1,000 types of Kanji characters. We confirmed that the proposed method removes more than 99% of ruby characters in early-modern Japanese printed books. Using the Kanji characters clipped without ruby characters, we performed Kanji character recognition experiments for multi-fonts of early-modern Japanese printed books with 1,000 types of Kanji characters from 2,965 types of the JIS Level-1 Kanji character set. The experimental results show that our Kanji character recognition method for early-modern Japanese printed books, which has been originally developed for handwritten Kanji character recognition, achieves 97% of recognition rate that is rich in very practical. The proposed ruby character removal method is to classify rows of early-modern books into several groups and generate mathematical formula by genetic programming to remove ruby character for each group. We make a comparison between the groups by the information of publisher and year as metadata and by the feature values obtained from the row images. As the result, the classification by the feature values brought better ruby character removal with less classes. In the Kanji character recognition experiments with 1,000 Kanji character types, the average recognition rate is 94% using 50% of training data when the number of training data images of each Kanji character type is less than 10. When the number of images exceeds 10, the average recognition rate achieves 99%. It turned out that the recognition rate of over 99% is achieved with at least nine images as training data, and the difference

in the features among various fonts can be absorbed by our method. Our future work is to increase the number of Kanji character types up to the JIS Level-2 (about six thousands). Moreover, the layout analysis for early-modern Japanese printed books is another important and difficult study we will try. As a challenging research topic, we have an open problem of bleed through, which is not removed by simple filtering or image processing, in early-modern Japanese printed books.

REFERENCES

C. Ishikawa, N. Ashida, Y. E. M. T. T. K. and Joe, K. (2009). Recognition of Multi-Fonts Character in Early-Modern Printed Books. *Proceedings of The 2009 International Conference on Parallel and Distributed Processing Technologies and Applications (PDPTA' 2009)*, 2:728–734.

Fletcher, L. A. and Kasturi, R. (1988). A Robust Algorithm for Text String Separation from Mixed Text/Graphics Images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 10(6):910–918.

Koza, J. (1992). *Genetic Programing : On the Programming of Computers by Means of Natural Selection*. The MIT Press.

M. Fukuo, M. T. and Joe, K. (2012). The Kanji character recognition evalution for the modern book of the same publisher (in Japanese). *The Information Processing Society of Japan. Mathematical Modeling and Problem Solving(MPS)*, 26:1–6.

N. Hagita, S. N. and Masuda, I. (1983). Handprinted Chinese Characters Recognition by Peripheral Direction Contributivity Feature. *IEICE*, J66-D(10):1185–1192.

N. Stamatopoulos, G. L. and Gatos, B. (2009). A Comprehensive Evaluation Methodology for Noisy Historical Document Recognition Techniques. *AND 2009 Proceedings of The Third Workshop on Analytics for Noisy Unstructured Text Data*, pages 47–54.