Thesis

Detection of furigana text in images

Nikolaj Kjøller Bjerregaard

Supervisors: Stefan Heinrich & Veronika Cheplygina

IT UNIVERSITY OF COPENHAGEN

June 2022

Abstract—*Furigana* are pronunciation notes used in Japanese writing. Being able to detect these can help improve Optical Character Reading (OCR) performance or make more accurate digital copies of Japanese written media by correctly displaying *furigana*. This project focuses on detecting *furigana* in Japanese books and comics. While there has been research into the detection of Japanese text in general, there are currently no proposed methods for detecting *furigana*. I construct a new dataset containing Japanese written media and annotations of *furigana*. I propose an evaluation metric for such data which is similar to the evaluation protocols used in object detection except that it allows groups of objects to be labeled by one annotation. I propose a method for detection of *furigana* that is based on mathematical morphology and connected component analysis. I evaluate the detections of the dataset and compare different methods for text extraction. I also evaluate different types of images such as books and comics individually and discuss the challenges of each type of image. The proposed method reaches an F1-score of 76% on the dataset. The method performs well on regular books, but less so on comics, and books of irregular format. Finally, I show that the proposed method can improve the performance of OCR by 5% on the manga109 dataset.

Source code
https://github.com/nikolajkb/FuriganaDetection

1 Introduction

Furigana is a part of Japanese written language. Japanese uses both a phonetic (representing sounds, called *hiragana*) alphabet and a logographic (representing meaning, called *kanji*) alphabet. In written Japanese, the two are mixed to form sentences. For *kanji*, since the characters represent meaning, the reader may not always know how they are pronounced[1]. Therefore, writers may sometimes add notes next to kanji to indicate their pronunciation, these types of notes are called *furigana*. *Furigana* is typically written in the Hiragana alphabet. Fig. 1 shows *furigana* on a book page, Fig. 2 shows furigana on a comic book page.

Furigana can be problematic for systems that process text within images. *Furigana* does not change the meaning of the text and can thus be disregarded by computers for most purposes. For example, current OCR systems do not handle *furigana* well. The *furigana* is often mistaken as regular text and inserted into the output or characters are misclassified because of the *furigana* next to them. An example of this is described in Fig. 2.

Detecting the locations of *furigana* could also be used to make more accurate digital copies of scanned texts. Most data formats for displaying text such as pdf, epub, and html has support for displaying *furigana*. By detecting what text is *furigana*, *furigana* could be displayed correctly in these formats.

The proposed system aims to detect the location of *furigana*, so that other systems may use this information to better process Japanese text within images.

2 Task definition

This thesis focuses only on printed media. *Furigana* rarely appears "in the wild", such as on signs or advertisements. This is not to say that such cases do not exist, but they

1. Names is an example where the reading of *kanji* is particularly difficult, given that names written with the same characters can often be pronounced many different ways as described by Ogihara in [1].

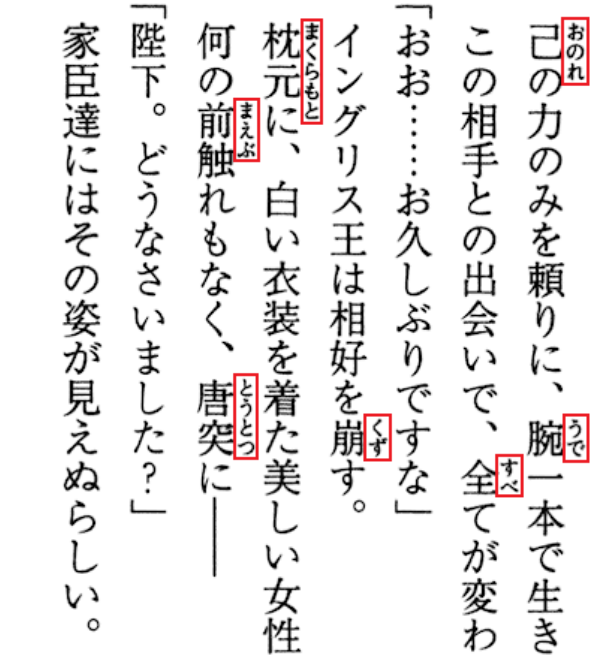


Fig. 1: *Furigana* marked by red boxes on an excerpt from a book. Source: "Eiyū-Ō, Bu o Kiwameru Tame Tensei-Su: Soshite, Sekai Saikyō no Minarai Kishi" by Hayaken [2].

Fig. 2: *Furigana* marked by red boxes on a comic book page. Using tesseract OCR on the top left speech bubble gives the following text:
こんとと 公才はお誠本の 本和導が バッサリだと/.
Removing the *furigana* from the image and running OCR again gives the following text:
今度はお旗本の 剣術指南役が バッサリだと/.
The version without *furigana* is a near-perfect transcription of the image (edit distance 2), while the version with *furigana* shares little similarity at all with the original text (edit distance 20). Source: "Akera Kanjinchō" by Yuki Kobayashi [3].

are rare, and there is currently no dataset containing such images. Additionally, *furigana* is generally not used in handwritten text. The images considered in this thesis can be divided into two categories: books and comics.

The input for the system is an image of a Japanese book or comic. The output is a list of axis-aligned bounding boxes (like in Fig. 1 and 2) specifying the location of the *furigana* in the image, if any is present.

See appendix A for a list of data used in this project.

3 Related work

There are currently no works directly treating the detection of *Furigana*. This means that there are no existing solutions to build on or compare with. Despite this, there is a wealth of papers that investigate similar problems. While these methods cannot be applied directly to the task proposed in this thesis, they can be used as a part of the solution, or elements of the methods described can be applied to the proposed task. In the first sub-section, I first describe papers that deal with analyzing text in images, specifically: OCR, scene text detection, and the use of mathematical morphology for detecting text in images. Next, I describe methods that are specifically designed for Japanese text. I describe methods for recognizing handwritten Japanese text and methods for recognizing text in Japanese comics.

3.1 Processing text in images

These works deal with processing and detecting text in general.

3.1.1 Optical character recognition

The problem of locating *furigana* is related to the more general problem of text localization. Text localization is a fundamental step in most systems that processes text in images. This includes OCR systems such as Tesseract [4], which uses connected component analysis to locate lines of text before splitting said lines into individual characters. In newer versions of tesseract, a Long short-term memory (LSTM) network based approach has also been introduced. OCRopus [5] starts by analyzing the physical layout of the page such as columns, text blobs, and lines to find the location of text. PaddlePaddleOCR(PP-OCR) [6] is another OCR system that is of particular interest for this project since it has placed a lot of focus on the processing of Chinese text, which is more comparable to Japanese than English. PP-OCR uses Differentiable Binarization as described by Liao et. al. [7] for text localization. This method involves using a probabilistic segmentation map of the text in the image to estimate the bounding boxes for the text.

3.1.2 Scene text

However, recently a more difficult challenge known as scene text detection has been widely studied. Historically, OCR systems mainly dealt with recognizing text in machine-written documents. Scene text deals with text that is captured "in the wild", that is, not in written media, but rather on images of street signs, billboards, storefronts, and more. Scene text is more difficult to process due to complex backgrounds, noise, warped perspective, and other challenges that do not appear in machine-written documents. Solutions to the scene text detection need to handle a problem that also shows up when processing text in comic books: How to find text in an image with many visual distractions?

Many datasets containing scene text have been created, such as COCO-Text [8] which is based on the MSCOCO [9] object detection dataset created by Microsoft, to which researchers from Cornell Tech have extracted images containing text and added annotations for the text. ICDAR [10], [11], [12] is created for the International Conference on Document Analysis and Recognition and has multiple versions containing scene text of different types. Total Text [13] by Ch'ng et. al. puts special focus on curved text. French Street Name Signs (FSNS) [14] contains more than one million images from Google Street View of french road signs. There is even a dataset containing Japanese text, downtown Osaka scene text [15] which was captured on a 360 degree camera and thus has many angles for each text instance.

The Robust Reading Competition is held every year and focuses on "written communication in unconstrained settings", which often includes scene text detection [16]. Current top text detection methods include: TextFuseNet [17], which uses Concurrent Neural Network (CNN) based networks in a multi-path fusion architecture to exploit character, word, and global level features. Corner-based Region Proposal Network [18], which uses detection of word-corners to predict bounding boxes. EAST [19], a popular text detection system that uses a Fully Convolutional Network to detect text locations.

3.1.3 Morphology for text detection

Several papers have been written that use morphology and connected component analysis for locating text. Such as Qi et al. [20] who present 12 features for text that can be found using connected component analysis and use Adaboost [21] to find scene text using these features. Wu et al. [22] use a similar approach of feature extraction and classification to detect text in scene text images. Dos Santos et al. [23] use morphology to extract features from pages containing handwritten text which are used to predict the locations of text lines.

3.2 Japanese text

These works deal exclusivly with Japanese text.

3.2.1 Handwritten Japanese

Many of the papers that treat Japanese text detection only consider handwritten text. Zhu et. al. [24] describe a probabilistic model for recognizing text from the HANDS-kondate dataset of handwritten Japanese text [25]. Ly et. al. [26] made an end-to-end Deep Convolutional Recurrent Network trained on a dataset of synthetic handwritten text. Clanuwat et. el. [27] focus on recognizing ancient Japanese text using deep learning. Finally, Ly et. al. [28] focus on historical handwritten text in this paper, creating an attention-based row-column encoder-decoder model for recognizing such text. Despite dealing with handwritten text as opposed to machine-written text, these papers do highlight some of the general issues related to processing Japanese text in

images. Such as text being written in multiple orientations and the difficulty of segmenting Japanese characters.

3.2.2 Text detection in Japanese comics

There has also been research into processing text specifically in Japanese comics (also known as manga). A key dataset for much of the ongoing research is Manga109 [29], [30]. This dataset contains 109 different manga and around 20,000 pages in total. The dataset provides annotations which includes the locations of frames, characters, and text along with transcriptions of the text. Mantra Inc. has created a tool for automatic translation of manga, their paper discusses many of the key aspects in the processing of manga [31]. Although they mainly discuss methods for translation, the problem of localizing text is also mentioned. Their method involves an object detector for speech bubbles and one for text lines. Methods specifically for text localization in manga have also been proposed. Such as, [32], [33], [34], [35] and especially good results from [36]. These are discussed in more detail in section 6.2.

3.3 Conclusion - related work

As none of the papers described above are directly applicable to the task of detecting *furigana*, I propose a new method for the problem in this thesis. We might, however, pick out some useful parts of these methods and apply them to detecting *furigana*. Such as using OCR to recognize text, locating text in manga using one of the several proposed methods, and analyzing text features using connected component analysis.

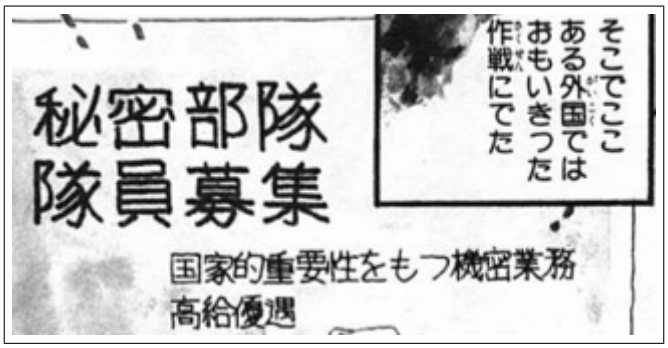
4 Problem analysis

This section analyses the problem of detecting *furigana* by detailing how *furigana* is used in Japanese and how it appears in written media. This information gives a general idea of what defines *furigana* as well as some pitfalls one may encounter when trying to locate it.

4.1 Characteristics of furigana

To understand how to solve the problem of locating *furigana*, we can begin by analyzing some characteristics of how *furigana* appears within Japanese text. The text shown in Fig. 1 and 2 is read from right to left and from top to bottom, that is to say, the lines of text are written vertically. This is the case for the majority of written media in Japan. However, Japanese text can be written both vertically and horizontally. For example, most digital text, such as on websites and in video games is written horizontally. In addition, short texts are often written horizontally, such as text on billboards or restaurant menus.

When the text is written vertically, the *furigana* will appear on the right side of the *kanji* that it is annotating. If the text is written horizontally, the *furigana* is usually written on top of the *kanji* or in a few cases on the bottom. If a word is made up of more than one kanji, the *furigana* may appear in one continuous string or with a space separating the *furigana* based on which *kanji* it is annotating. *Furigana* is always written in a smaller font than the main text.



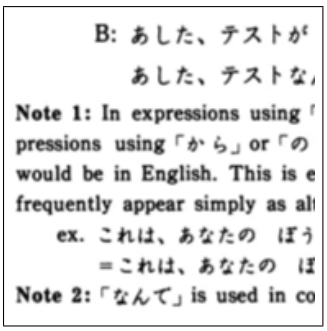


Fig. 3: Challenging texts. Top: here, different font sizes and orientations are mixed on a single page. Src: [37]. Bottom left: English and Japanese text mixed together in a textbook. Src: [38] Bottom right: text with special formatting. Src: [39]

Furigana is almost always written in the *hiragana* alphabet, but this is not always the case. Sometimes, the author may write *furigana* in the *katakana* alphabet, which is a second phonetic alphabet used in Japanese, mainly for foreign words. In some cases, an author might even write "*furigana*" using *kanji*. For example, an author might write ガール, a transliteration of the English word 'girl' and annotate it with 女子, the Japanese word for girl. This type of usage does not fit most definitions of *furigana*, but in this project we shall consider it part of the text that should be detected since it is still just a type of reading aid.

4.2 Challenges of Japanese written media

This list of challenges is the result of discoveries made throughout this project as well as my experience reading Japanese books and comics. Some examples are shown in Fig. 3

Any solution to the problem of detecting the location of *furigana* must consider these challenges imposed by the nature of the data in question. Failing to take these challenges into account will mean that the system does not generalize to a diverse set of data.

4.2.1 Images may contain more than just text

For regular books, pages almost exclusively consist of text, but that is not to say that other elements do not occur. Books may include images, graphs, or shapes used for formatting such as boxes, lines, circles, or arrows. As for comics, these contain a large number of visual elements, making it vital that a solution can deal with this complexity.

4.2.2 Images may be of different resolutions

The physical size of books differs, but even for books of the same size, the resolution of the scanned pages varies greatly. This means that relying on pixel-based sizes may not generalize well.

4.2.3 Font sizes may differ

Different books use different font sizes.

4.2.4 A single page may contain multiple font sizes

One cannot assume that a single page only has one font size. In books, headers are often larger than other text. And in comics, the font size might vary from one speech bubble to another.

4.2.5 Furigana is small

Furigana is usually written in very small font sizes. Depending on the quality of the scan, it might be difficult even for a human to tell what characters are written.

4.2.6 Images may be rotated

Sometimes an image will not have the correct orientation after being scanned.

4.2.7 Images may contain slanted, warped and curved text

Although most text in written media is written in straight lines, slanted, warped or curved text might appear. Particularly, in Japanese comics it is common for the author to occasionally include handwritten text, this text is often not written in a straight line. It should be noted that it is uncommon for such text to include *furigana*.

4.2.8 Text might not be black

Although rare, text might be written in a font that is not black. Particularly, text might be white on a black background.

4.2.9 Different text orientations on a single page

As mentioned previously, Japanese can be written horizontally and vertically. Additionally, there might be both horizontal and vertical text on the same page.

4.2.10 Pages may contain text that is not Japanese

Particularly, small bits of English text can sometimes be found in Japanese books. In language textbooks for Japanese learners, much of the text is often not Japanese.

5 Dataset

This section describes the dataset created for this project. The dataset was created to represent a diverse selection of books and comics. It includes all the cases described in the challenges section4.2, except for rotated pages. In this project, the dataset is only used for evaluating the performance of the system. During development, a separate development dataset was used, while the test dataset used to get the results in the experiments section 8 was only used after the system was implemented.

5.1 Data and sources

The data is split into two main categories: books and comics. The books used are mainly books targeted at younger audiences, particularly ”light novels” which is a genre of Japanese literature mainly targeted at young adults. The reason for this is that books written for adults often contain little to no *furigana*. The comics were chosen with no particular emphasis on any one type as most comics use *furigana*.

The included pages are the first pages from each of the works, that is, the first pages of the main content. Books often contain a few pages of illustrations, title page, table of contents, etc. Including these would leave little room for pages that are interesting for the task. Aside from this selection of the starting point, the pages are selected as an uninterrupted sequence, even if that means including pages with no *furigana*, full-page illustrations, etc.

The full list of works used can be seen in appendix A.

5.2 Annotation

The annotations are made using axis-aligned bounding boxes. This is a natural fit for the data since it is taken from scanned pages of printed media, and thus contains little to no warped text.

The dataset is annotated on character level. Most western text detection datasets are annotated on a word-level basis such as [8], [10], [13], while Yuan et al. annotate their Chinese scene text dataset on a character level [40]. *Furigana* is not regular text, so defining annotations based on words or other grammatical structures is not appropriate. Therefore, the least biased and most flexible style of annotation is to annotate on the character level. This also means that the definition of how the annotations should be done is unambiguous, and should thus provide consistent annotations.

I created the annotations for the dataset myself. Due to the simple and unambiguous definition of the annotations, the annotation work could easily be outsourced. To test this, I had 4 people with no knowledge of Japanese create annotations for a single page. They all completed this task with no errors in the annotation.

6 Method

This section details the proposed method developed during this project. At the end of the section, I describe why using current OCR systems to detect *furigana* is not viable. Fig. 4 shows a visualization of the proposed detection process.

In this thesis, I propose a new method that is not directly based on any previous work since, as we saw in section 3, there are currently no proposed methods for detecting *Furigana*. However, it is possible to use some of the methods as part of the proposed system. According to preliminary experiments that I performed, scene text detection did not work for detecting text in Japanese media. The main reason for this is a lack of Japanese models. Most of these systems are trained for English text only. Despite not having a Japanese model, I tested TextFuseNet [17], the current top performer in many scene text tasks on data from this project. As expected, it did not detect any text in the images I tried. Using OCR did not work well either, this attempt is

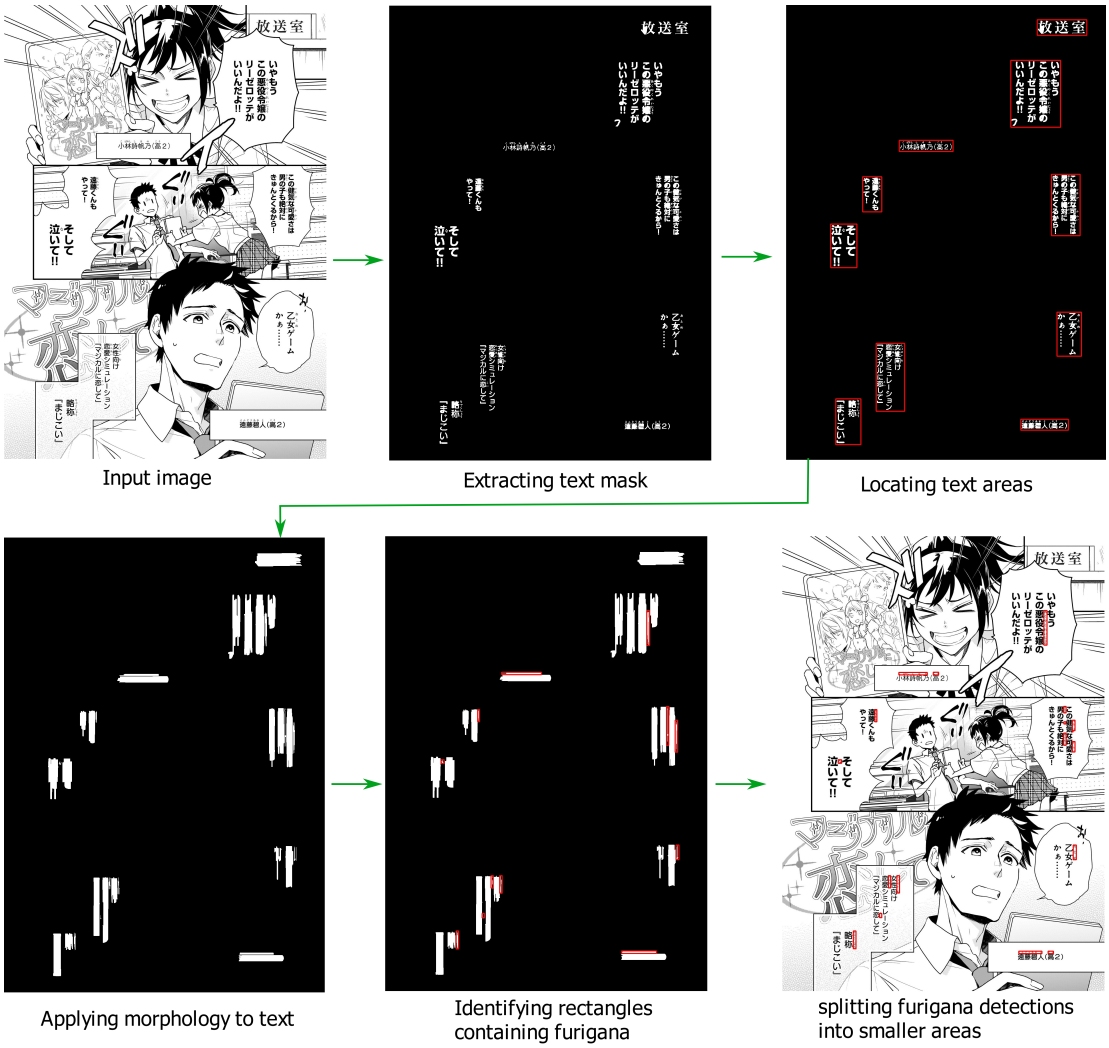


Fig. 4: Visualization of the detection process. Image source: ”Tsundere Akuyaku Reijō Rīzerotte to Jikkyō no Endō-kun to Kaisetsu no Kobayashi-san” by Suzu Enoshima [41].

described in section 8.4. On the other hand, the methods for extracting text from manga worked quite well for the data in this project. I saw good initial results using morphology[2] and connected component analysis to detect *furigana* in books. The text extraction methods allowed this approach to also work for comics. For these reasons, I chose to continue working on this approach which led to the method described in the following sections.

6.1 Outline of proposed method

The main steps involved in detecting *furigana* in the proposed system are as follows.

1) Text extraction
2) Finding text regions
3) Inferring text direction
4) Applying morphology to text lines
5) Estimating font size
6) Inferring which text is *furigana*
7) Splitting detections into smaller areas
8) Optional: verify *furigana* detections using OCR

In the following sections, the individual steps are described.

6.2 Extracting text from the image

For books, pages consist of mainly text, and the text can be extracted by simply thresholding the image. However, this is not the case for media like comics that contain visual elements in addition to text. To find *furigana* we first need to extract the text while not getting any of the background noise. In this project, one of the text detection models from Manga Image Translator by zyddnys and other contributors [3] is used for text extraction. The section below describes the reasoning for this choice.

2. Mathematical morphology involves defining a kernel (or structuring element) consisting of geometrical shape of pixels, e.g. a 2x2 square. This kernel ”slides” over a binary image, and depending on how the kernel intersects with pixels in the image, new pixels are created or existing ones are removed. For a good introduction, see https://towardsdatascience.com/understanding-morphological-image-processing-and-its-operations-7bcf1ed11756

3. https://github.com/zyddnys/manga-image-translator

6.2.1 Choice of text extractor

Text detection is a well-researched problem, Ye et al. [42] gives an overview of the topic, detailing the individual sub-problems of a text detection system and describing the different methods used to solve these problems, finally giving an overview of some of the proposed systems found in the literature.

There are several large datasets such as [8], [13] and [10] that contain images with annotated text and many text detection models have been proposed based on these datasets. There are a few problems with applying these models directly to the proposed system. Most importantly, these datasets do not contain Japanese text. Also, these datasets contain scene text, that is, text captured "in the wild" and not from written media like books and comics, this means that the regularity that these provide is not exploited. For these reasons, a more specific approach is needed.

A system capable of detecting text in comics should also be able to solve the relatively easier problem of detecting text in books. Arimaki et al. [32] propose a method for detecting text in manga (Japanese comic books) that considers geometrical features of connected components within the image to generate text region proposals. These regions are then classified using a deep learning model based on the ImageNet [43] classifier by Krizhevsky et al. [44] in order to filter away false positives. Their method generates square text regions. Chu et al. [33] propose a similar method of region proposal and classification. Their method uses a modified Faster R-CNN [45] to generate region proposals and classify said regions. Tolle et al. [34] propose a method whereby speech balloons in manga are detected using connected component analysis. This means that this method cannot detect text that is not inside a speech balloon. Piriyothinkul et al. [35] propose a method where letter candidates are generated using Stroke Width Transform. These candidates are then classified using an SVM trained on the HOG features of the letter proposals. The system developed by Mantra [31] also does text detection in manga but the paper does not go into much detail on how the text extraction is performed.

The results for these methods are of mixed quality, with [35] being the top performer with an F1-score of 0.5, although a direct comparison is not possible since different datasets are used for evaluation across these papers. None of the authors provide their source code, so I was unable to test the performance of these papers on the data used in this project.

Del Gobbo [36], [46] proposes a method, Manga Text Segmentation (MTS), which uses a deep network model with a U-net [47] architecture to perform end-to-end text detection. Besides good performance, the system outputs a pixel-level mask of the text. This is advantageous, especially in situations where text is placed on top of a background since this background noise will be filtered away. I tested the system developed by Del Gobbo on data from this project. Del Gobbo's system generally performed quite well on many pages. However, a common issue was an abundance of noise/false positives in the output. That is, there are a lot of shapes in the output that is not text. These performance issues might partially stem from the fact that relatively little training data was used, 450 images in total. Compared to, for example, over 60000 images in the COCO-Text dataset [8].

Besides methods proposed in the literature, there also exists a few hobby projects that aim to solve this problem. Two are of particular interest. First is Comic Text Detector (CTD) by dmMaze[4]. This project, like MTS [36], uses an end-to-end deep learning approach. This project however is trained on 13 thousand images of which 1/3 are from the manga109 dataset [30] also used by MTS, 1/3 are from western comics and 1/3 are synthetic data. The other project is Manga Image Translator (MIT). This project provides both its own text detection and one based on CTD. Their original text detector does not work for this project, as the mask is too fuzzy and dilated for the *furigana* to be separated from the main text. The implementation based on CTD on the other hand performs very well, even outperforming the original CTD due to having less noise in the mask. The reason for this difference is unclear since they supposedly use the same model and post-processing.

In conclusion, the best text extractor currently available according to the requirements of this project is MIT (using their implementation based on CTD), and thus this is used for text detection in the proposed method. It should be noted however that each system has their pros and cons. Particularly, there is a trade off between the amount of false positives and false negatives, some systems mark too many shapes as text, and some mark too little. MIT erroneously filters away some of the furigana, whereas CTD and MTS generally preserve more of it. But the higher amount of noise in the latter two means that MIT outperforms them. The performance of the different methods are compared in section 8.1.

6.3 Finding text areas and Inferring text direction

A single page may contain text of different orientations and of differing font sizes. Therefore, multiple text areas are located on each page. In a comic, a text area might equate to one speech bubble and on a book page, there might be a text area for the main text and one for the header.

First, we run a morphological closing operation to join characters that are close to each other.

These initial areas are then split into horizontal and vertical groups based on the orientation of the text. This is done by considering the height/width ratio of the text area. If the area is taller than it is wide it is considered vertical, if not it is considered horizontal. This simple yet effective approach has some limitations, which is discussed in section 8.5.

Next, areas that are close to each other are merged. This is done for horizontal and vertical groups individually to avoid merging horizontal and vertical text into one area. Doing this improves the performance of the system slightly, as can be seen in section 8.5.

The result is a list of rectangles that define text areas as well as information about whether a given text area is horizontal or vertical.

4. https://github.com/dmMaze/comic-text-detector

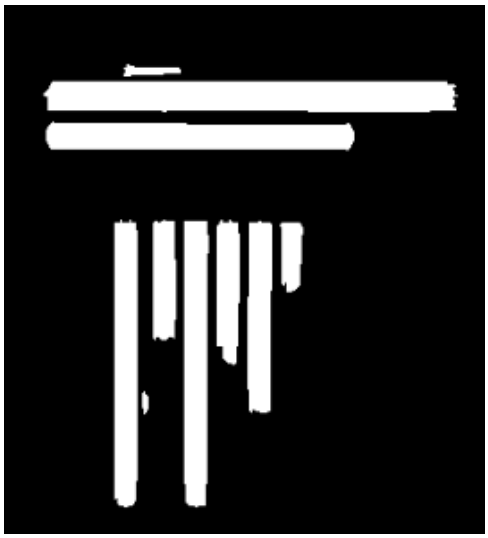


Fig. 5: Morphology applied to horizontal and vertical text. Closing operation with a long thin kernel. The lines of text are clearly visible. The small blobs next to two of the lines are *furigana*.

6.4 Applying morphology to text lines

In this step, the characters in each line of text are merged to make the entire line appear as a long rectangle. To achieve this, a very long and thin kernel, e.g. a 1/40 ratio, is used while doing a closing operation. The kernel is horizontal for horizontal text and vertical for vertical text. The result is that the text is merged in the direction of the text but not to the sides where the *furigana* is located. An example of this can be seen in Fig. 5. The bounding boxes of the connected components are returned as input to the next step.

Before the operation described above is performed, erosion is performed with a small kernel. This is to avoid having the *furigana* touch the main text and accidentally being detected as part of the main text. This is an important step, as not being able to separate *furigana* from the main text is a big source of errors. The difference in performance can be seen in section 8.5.

6.5 Estimating font size

We now have a list of rectangles that may be either *furigana* or regular text. We can differentiate *furigana* because it is smaller then the regular text. However, in order to generalize this for a diverse set of images, we need an approach that does not assume any absolute sizes. The way this is done is to estimate the font size of the main text.

We assume that the main text has the largest total area of all the shapes. We first get the thickness of all rectangles: the width, if the text is vertical and the height, if the text is horizontal. Then, we find the minimum and maximum thickness. We then divide this interval into bins and distribute the rectangles into them according to their thickness. The bin with the largest total area is assumed to be the bin containing the main text. We then calculate the average thickness of the rectangles within this bin, this is the font size. This algorithm is show in algorithm 1.

Algorithm 1 Algorithm for finding font size

Input: a list of rectangles R, a bin size b
$x \leftarrow$ **min**(thickness of rectangle in R)
$y \leftarrow$ **max**(thickness of rectangle in R)
if $x = y$ **then**
  return x
end if
$font\_size \leftarrow 0$
$max\_area \leftarrow 0$
for $i = x$ to y **do**
  $area \leftarrow$ **sum**(area of rectangles in R with thickness between i and i+b)
  if $area \geq max\_area$ **then**
    $max\_area \leftarrow area$
    $font\_size \leftarrow$ **mean**(thickness of rectangles in bin)
  end if
end for
return $font\_size$

6.6 Inferring which text is *furigana*

Once we have the font size, inferring which text is *furigana* is quite simple, we define *furigana* as all text rectangles that are less than half the size of the main text, plus a small buffer and a minimum value to avoid marking noise, and return the rectangles that fit this requirement.

6.7 Splitting detections into smaller areas

Before returning the final detections, they are split into smaller clusters of characters to avoid marking empty space. Failing to do this often results in evaluation case 9 from Fig.6, meaning that the detection is marked as a false positive. Section 8.5 shows the decrease in performance when not splitting the clusters. The splitting is done by running a morphological closing operation on the text within the detection to merge characters that are close to each other but not separated clusters of characters.

6.8 (Optional) Verifying *furigana* detections using OCR

We can use OCR to verify the detected *furigana* locations. A weakness of the method described above is that it does not have any concept of language, it is based purely on visual processing. This means that the system does not make any guarantees that the areas selected even contain text. Noise, figures, parts of illustrations, etc. can be selected as *furigana* if it has the right shape and location.

Tesseract is used for OCR operations. While tesseract has its flaws when it comes to Japanese text, it was by far the best OCR solution out of the ones tested. Paddlepaddle failed to detect any text at all in most cases and OCRopus did not have a compatible Japanese model.

At the end of the previous step, a list of rectangles possibly containing *furigana* is provided. These are used to crop the original image to get a new image containing only the area of interest. Before passing this image to the OCR engine, two prepossessing steps are applied. The image is resized to double size and a white margin is added to the image. This helps improve the performance of tesseract [5].

5. These pre-processing steps are taken from tesseracts documentation https://tesseract-ocr.github.io/tessdoc/ImproveQuality.html

The following configurations are applied to tesseract. Tesseract has two models for Japanese text, one for horizontal text and one for vertical, the correct one is selected based on the orientation of the text. Tesseract allows the user to specify the Page Segmentation Mode (PSM). PSM allows one to specify what kind of text the input image contains. If the text is horizontal, the PSM is set to "Single Line", for vertical text it is set to "Single Block Vertical Text" and for square text, assumed to be one character, it is set to "Single Character". Finally, using the tessedit_char_whitelist variable, the set of accepted characters is limited to *hiragana* and *katakana* characters. If no restriction on the set of valid characters is done, there will very often be some character that happens to have high confidence. Also, invalid characters would not be disregarded. There is only one case where this approach fails and that is in detecting *furigana* that is written in *kanji*. However this is very rare, and as described in section 4.1 it is debatable whether this kind of notation can even be considered *furigana*.

In order to determine if the image provided is valid, the confidences of the words returned by tesseract are inspected. If the mean confidence is above a threshold, the image is considered valid. Additionally, if the confidence of any word is above another, higher, threshold, the image is also accepted. If an image contains no characters, or characters not of a correct type, the confidences will be low or there will simply be no detected words. The use of the term "word" is somewhat misleading, I found that in most cases the "words" returned by tesseract corresponded to single characters or groups of two or three characters. This behaviour is understandable given that there is no logical way to split *furigana* into words.

For filtering away invalid detections, this approach works very well as can be seen in the experiments section 8.1 from the increase in precision. The main issue with it is that sometimes, word confidences are very low even for valid text. This means that when using this verification step, a lot of correct detections will be filtered away.

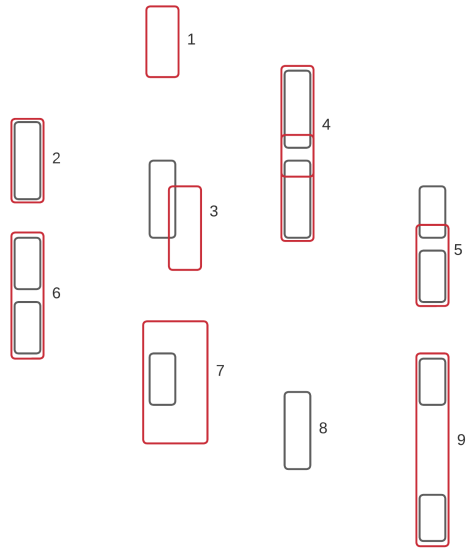


Fig. 6: Different cases for evaluation. The red boxes should be seen as the predictions made by the system and the black boxes are the ground truth. 1: FP, detection not on ground truth. 2: TP, detection fits ground truth. 3: FP, detection does not sufficiently cover ground truth. 4: TPx2, detections overlap, but sufficiently cover ground truth. 5: TP+FN, detection overlaps one ground truth annotation but does not sufficiently overlap the second one. 6: TPx2, detection overlaps two ground truth annotations. 7: FP, detection area too large. 8: FN, no detection. 9: FP, detection area too large.

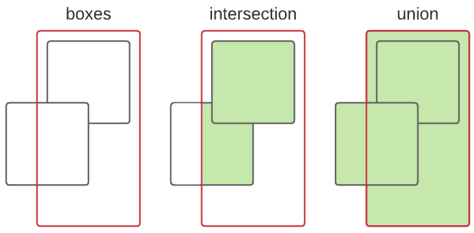


Fig. 7: Example of intersection and union in n-IOU

7 Evaluation Metric

For evaluating the performance of the system, I use an approach similar to that used in most object detection challenges. Each bounding box detected by the system is compared to the bounding boxes of the annotation, if the detection is similar enough to one of the annotations, the detection is evaluated as a true positive (TP). Otherwise, it is evaluated as a false positive (FP). These counts of TP and FP can then be used to calculate a recall, precision, and F1-score. This approach to evaluation is illustrated in Fig. 6.

This protocol is very similar to the Intersection Over Union (IOU) metric used in most object detection evaluation algorithms such as [48] by Padilla et al. in which different approaches to object detection evaluation are described and the evaluation used by Microsoft COCO [9]. In the IOU metric, whether a detection sufficiently overlaps a ground truth label is decided by looking at the IOU between the detection and the label. The protocol described in Fig. 6 differentiates itself only in case number 6. Here the detection covers two ground truths. In usual object detection systems, this is an error since the two ground truths should be detected as two distinct objects. However, in the case of detecting *furigana*, there is only one class of object. Additionally, there is no need to separate distinct objects. Since the goal is simply to detect all furigana, it should not matter how/if the characters are grouped together as long as there is not excessive space between grouped characters.

To facilitate this difference, I introduce the concept of IOU between multiple objects, henceforth referenced to as n-IOU. Usually, IOU is only defined between two objects, a good description of this can be found in Rezatofighi et al.'s paper on generalized intersection over union [49], however this does not fit the method of evaluation defined above. In Fig. 7 the n-IOU between the red box and the two black boxes is illustrated. The intersection is the area where the red box intersects with either of the black boxes, the union is the union of all three boxes. The concept can be extrapolated to any amount of black boxes. Mathematically, n-IOU is defined as

$$\frac{R \cap (B_1 \cup B_2 \cup \cdots \cup B_n)}{R \cup B_1 \cup B_2 \cup \cdots \cup B_n}$$

TABLE 1: Main results. R: Recall(%). P: Precision(%). F1: F1-score(%). Scores marked in bold are the top scores for R/P/F1 within each category. MIT: Manga Image Translator[6]. CTD: Comic Text Detector[7]. MTS: Manga Text Segmentation [46].

Method	All			Books			Books (no textbooks)			Books (only textbooks)			Comics		
	R	P	F1	R	P	F1	R	P	F1	R	P	F1	R	P	F1
Final method (MIT)	**75**	83	**76**	88	84	85	**91**	92	91	80	60	67	**59**	82	**67**
With OCR validation (MIT)	66	**91**	73	83	**92**	**86**	85	**95**	89	77	**84**	**79**	45	**90**	57
Alternative text detection: thresholding	47	47	46	82	79	79	**91**	94	**92**	56	38	44	0	0	0
Alternative text detection: CTD	73	78	73	**89**	82	83	**91**	92	91	80	54	62	55	72	61
Alternative text detection: MTS	74	77	73	88	81	82	87	84	84	**91**	74	78	57	71	61

This definition can be extrapolated to work with any amount of R sets as well which would make the definition $\frac{(R_1 \cup R_2 \cup \cdots \cup R_n) \cap (B_1 \cup B_2 \cup \cdots \cup B_n)}{R_1 \cup R_2 \cup \cdots \cup R_n \cup B_1 \cup B_2 \cup \cdots \cup B_n}$ The latter is not used in this project.

In case 6 in fig 6, the IOU is not large enough between either of the two black boxes and the red box for them to count as a true positive, however the n-IOU between the two black boxes and the red box is large enough. This is scored as two true positives, one for each ground truth. Using n-IOU has the benefit of abstracting away the exact format of the annotations and predictions, meaning that the clustering of characters or lack thereof does not impact the evaluation.

7.1 implementation of the evaluation function

The evaluation protocol is implemented as follows. For each prediction made by the system, the Intersection Over Area (IOA) between the prediction and each of the ground truth labels is calculated. The IOA can be thought of as how much of the label is inside the prediction. Labels that have a sufficient overlap are then selected. The threshold is decided by the user and should be set to the value that gives the best results, as it is only used to make it easier for the system to find correct detections. If there is one or more overlapping labels, the IOU or n-IOU between the predictions and the label(s) is calculated. If this is above the threshold defined when running the evaluation, the number of true positives is increased by the amount of matching labels. The standard threshold of 0.5 commonly used in object detection is also the default here. The matching labels are added to a list of used labels and cannot be used again for other predictions. From these counts of true and false positives, the recall, precision, and f1-score are calculated. If true positives + false positives equals 0, precision is undefined and the image is skipped. If true positives + false negatives equals 0, recall is undefined and the image is skipped.

8 Experiments

In this section, several experiments are described. In section 8.1, the performance of the method according to evaluation protocol is listed and these results are discussed. In section 8.2, the types of errors present in the current method are enumerated and visual examples are given. Section 8.3, shows how the system can be used to improve the performance of OCR. Section 8.4 details the experiment I did to see if it would be possible to detect *furigana* using an OCR engine. Finally, section 8.5 showcases some additional experiments relating to various configurations of the system.

8.1 Main experiment: performance of system according to evaluation protocol

This section details the performance of the system according to the evaluation protocol defined in section 7. Table 1 lists the experimental results for the test data.

I have added two columns splitting books into two sub-categories: regular books and textbooks. Textbooks are language books that are used to teach Japanese. I choose to include pages from two such textbooks in the dataset. We can see that both the recall and especially the precision is quite low for textbooks. Whereas Japanese books generally follow a predictable layout, these textbooks do not. The layout of the pages is often quite complex and often includes English text as well as Japanese, making these books a more difficult problem. It should be noted that the number of textbook pages is quite small, so these results are not as reliable as the other categories.

The first row shows the results from the final method which is using MIT as text detection. On the whole dataset, an F1-score of 76 is achieved. MIT is especially good for comics where it beats the other text detection methods. This leads to it beating the other methods in overall F1-score. But as we can see, it does not hold top scores in all areas.

The second row shows the scores with the optional step of validating detections using OCR enabled. As expected, this increases precision at the cost of recall. The type of data that sees the biggest improvement when using validation is textbooks. This is due to the high level of noise and false detections for this data type. This improvement in score for textbooks means that the F1 score across all books is higher when using validation, although the score for only regular books is slightly lower.

The third row shows results for the 'thresholding' text detection method. This is simply thresholding the entire image without using any model for extracting text. For regular books, we can see that this method works quite well, reaching an F1-score of 92. Another advantage of the thresholding approach is that it is much faster than the other text detection methods. With the thresholding method, the entire dataset is processed in about 10 seconds on the computer used for testing whereas MIT and CTD take about 150 seconds to process the dataset. However, the thresholding method struggles with textbooks, and is completely useless for comics, getting an F1-score of 0.

6. https://github.com/zyddnys/manga-image-translator
7. https://github.com/dmMaze/comic-text-detector

TABLE 2: Error types

Error nr.	Error description	Pages	Total
1	Errors in text mask	33	260
2	Disconnected bits of a character detected as *furigana*	17	26
3	*Furigana* is so close to main text that it is not correctly separated from it	12	173
4	Special characters (exclamation mark, underline etc.) is detected as *furigana*	8	44
5	Wrong text orientation detected	6	112
6	Text with non-standard formatting (slanted, hand-written etc.)	6	49
7	*Furigana* is too small	3	3
8	English text	2	2
9	Incorrect text area estimation	1	16

The fourth row shows the results from CTD text detection. CTD's main strength is regular books. But for comics, CTD does not perform as well as MIT due to having more noise in the mask.

The Fifth row shows the results from MTS. Overall, MTS also sees weaker results compared to MIT. However, MTS greatly outperforms the other text detection methods in textbooks. There are two main reasons for this. One, MTS generally includes more text in its mask. Whereas the other text detection methods often fail to include some text, especially *furigana*, this is not as common for MTS. Two, MIT and CTD include some noise present in the scanned pages. MTS does not include this noise.

8.2 Error types

Table 2 shows the categories of errors within the detections of the test data. The errors are enumerated both by the number of pages that a given error type appears on and by how many errors in total are caused by this error type. For example, error type 2 appears on 17 pages but only causes 26 errors in total, while error type 5 causes over 112 errors on only 6 pages. E.g. when the wrong text orientation is detected, all the *furigana* within that block of text is not detected and creates many false negatives.

Figure 8 shows examples of these errors.

8.3 Improvement in the performance of OCR

One of the goals of detecting *furigana* was to improve the performance of OCR of Japanese text. To test this with the proposed method, I used the manga109 dataset. Manga109 includes annotations of manga. For each page there is an annotation for each text block along with the location of the text block.

The experiment was performed as follows. For each page, the *furigana* is detected and removed by replacing the *furigana* area with white pixels. Then, for each text block, OCR (tesseract) is performed on the text block in the original image and the version with the *furigana* removed. For each of these, the Levenshtein edit-distance [54], and based on this the Character Error Rate ($distance/n$), is calculated. We can then compare the two to see if there has been an improvement in the CER. The results of this are shown in table 3.

TABLE 3: Performance of OCR with and without *furigana* removal. Performance is given in CER (lower is better).

Data/method	text blocks	CER with *furigana*	CER *furigana* removed	improvement
first 15 books of manga109	20893	0.496	0.472	+5%
first 15 books of manga109, books with frequent *furigana* usage	15345	0.510	0.476	+7%
first 15 books of manga109, books with infrequent *furigana* usage	5548	0.459	0.461	-0.3%
Akkera Kanjinchou, *furigana* removed with proposed method	1340	0.532	0.473	+11%
Akkera Kanjinchou, *furigana* removed manually	1340	0.532	0.450	+15%

Removing *furigana* using the proposed method improves the performance of OCR slightly. The amount of improvement varies based on the book. The largest improvement was observed in Belmondo with an improvement of 12% and the smallest improvement was seen in Burari Tessen Torimonocho, where the CER decreased by 1.6%.

One thing to note about these results is that not all the manga in manga109 include *furigana*. In cases where *furigana* is not used, trying to remove *furigana* can only negatively affect the performance, since everything that is removed is a false positive. The second and third row show this difference in performance.

It is also of interest to compare removing *furigana* using the proposed method to removing *furigana* manually as a "gold standard" *furigana* removal. To test this, I manually removed the *furigana* from the Akkera Kanjinchou manga. Removing *furigana* using the proposed method sees an improvement of 11%, while removing *furigana* manually improves the performance by 15%. In other words, the proposed method reaches 72% of the maximum possible improvement in performance that can be achieved by removing *furigana*.

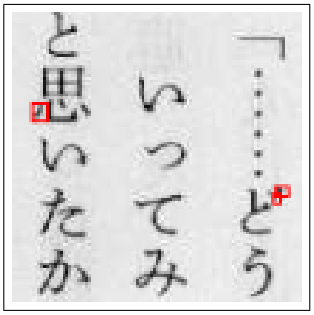
8.4 Detecting *furigana* using OCR

An interesting alternative to the method described in sections 6 would be to use OCR to detect what text is *furigana*. The first step of most OCR algorithms is to detect the location of text. In theory, we could cut out many of the error-prone steps in the proposed method by using these detections as the starting point for the detection of *furigana*. If we had the bounding boxes for the main text and the *furigana*, it would be simple to compare the sizes and determine which is *furigana*.

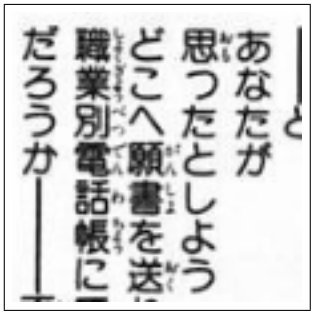
In practice, this approach has some major problems. The assumption that we can get accurate text locations with main text and *furigana* does not hold. To test this approach, I used tesseract to get text locations. Tesseract often fails to detect main text and *furigana* as separate lines, which creates inaccurate text lines as well as making it impossible to use this information to separate *furigana* and main text.

Fig. 8: Examples of error types. The red boxes show *furigana* detections, except for (9) where it shows the detected text area. The images with black background show the text mask.

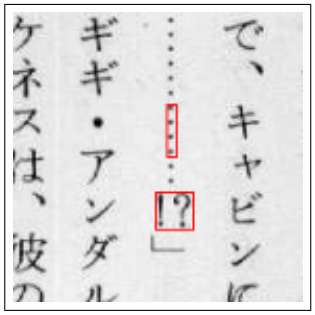
(1) Error in text mask, there are two lines of text but among the characters is a lot of noise, most noticeably the white blob starting from the middle of the two lines and extending to the right line. Src: [50]



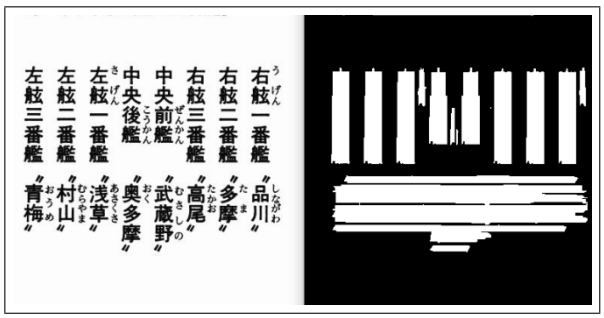
(2) Disconnected bits of a characters detected as *furigana*. Src: [51]



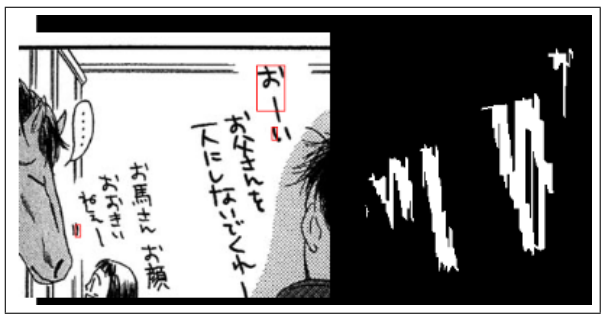
(3) *Furigana* close to main text, which results in the *furigana* not being separated from the main text and thus not being detected. Src: [37]



(4) Special characters detected as *furigana*. Src: [51]

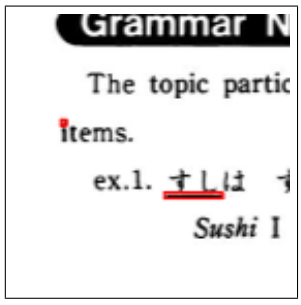


(5) Wrong orientation detected. The original text is shown on the left and the text mask with morphology applied to it on the right. The text on the top is correctly detected as being vertical, but the text on the bottom is detected as being horizontal which results in the text lines being ”stretched” in the wrong direction. Src: [52]

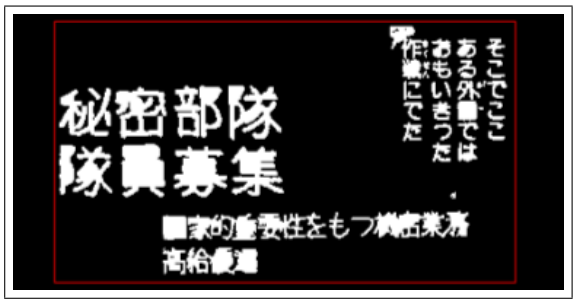


(6) text with non-standard formatting. On the right is the original image. On the left is the image with morphology applied, here we can see what goes wrong. Some of the lines are merged together into one large blob, smaller blobs next to this will be detected as *furigana*. Src: [39]

(7) *Furigana* too small. The first two *furigana* are correctly detected, but the last one is so thin that it disappears when processing the image. Src: [53]



(8) English text causing problems. The dot above the i in ”items” is detected as *furigana*. Src: [38]



(9) Wrong text area estimation. Here we see three text areas, however since they are very close the system detects them as one area. The red box shows the detected area. Src: [37]

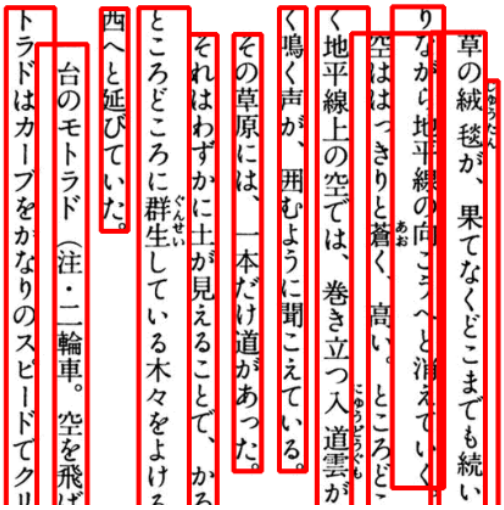


Fig. 9: Text lines in a book as detected by tesseract. Src: [55]

These issues with text lines can be seen in figure 9. The *furigana* is correctly separated from the main text in the first line, but not so in any of the following lines. This was the case for all the images I tested. The same issues are present at the word level bounding boxes, where words often contain both *furigana* and main text (tesseract is not made to give accurate character level annotations, so these are not accurate regardless).

This result also gives some insight into why *furigana* worsens the output of OCR systems. The *furigana* makes the system detect text lines incorrectly, which in the worst case means that the main text will not be detected correctly since the *furigana* next to the characters might cause misclassifications.

8.5 Additional experiments

Table 4 shows the result of changing various details of the method. The first row is the main result from section 8.1. The second row is related to section 6.3. The third row is related to section 6.4. The fourth row is related to section 6.7.

Only the fifth row has not been discussed previously. Inferring the direction of the text is a non-trivial problem. The approach described in section 6.3 fails if, for example, we have a long row of short vertical lines in an area. Then the area is longer than it is tall, but the lines are still vertical. A better approach is to look at the height/width ratio of the individual lines. This can be done by using an OCR engine (here tesseract) to find the lines of text within the text area. The number of vertical and horizontal lines is counted, if there are more vertical lines, the area is considered vertical and vice versa. However, this method is much more computationally expensive. And as we can see, it ends up with worse performance than the simpler method.

9 Discussion

In section 8.1, we saw that the system performs better on books than on comics. I found that most of the errors related to comics are from older comics. the dataset is a mix of newer and older comics. In older comics, the *furigana* is written very close to the main text, add to this the often poor resolution of older comics and it becomes hard to separate the main text from the *furigana*. This is the main reason for the low recall seen on comics as it means that much of the *furigana* is not detected. For newer comics, the scores are more comparable to books.

TABLE 4: Experiments using different configurations of the method.

Experiment	Recall(%)	Precision(%)	F1(%)
Baseline	75	83	76
Not merging text areas	74	87	77
Not using erosion during morphology step	63	86	68
No splitting of *furigana* areas	46	80	54
Using OCR to detect text orientation	75	82	76

Section 8.3 showed that it is possible to improve the performance of OCR by removing *furigana*. One thing that stands out is that the error rate is quite high, even with the *furigana* removed manually. This indicates that *furigana* is not the main problem with tesseract's OCR. Other OCR services show better results than this. For example, using Google's OCR solution seems to make much fewer errors when transcribing Japanese text, despite the fact that it is based on tesseract. This shows that there should be ample room for improvement in the general performance of the OCR. If the error rate was lower overall, it is possible that the relative improvement seen from removing *furigana* would be higher than the improvements seen in section 8.3. A big contributor to the high error rate is that when the detected text for a single text area is longer than the expected text, the error rate for this text area will be very high. As an example, in one text area the expected text is "!!?", but tesseract outputs "|、』,條ん ル/^nq トリ自生", in this case, the character error rate is 5. These kinds of detections significantly heighten the error rate.

Section 8.5 showed the results of the OCR-based method of inferring text orientation. The method generally works well, but text lines are not always detected correctly. Adding some heuristics for when the OCR-based method should be used, I was able to achieve a performance that was slightly better than the main results. However, given that this method significantly slows down the system, the lack of strong improvement, and the general inconsistency of the method, I decided to not use this method of inferring text orientation.

10 Future Work

The performance of the method depends in large part on the performance of the auxiliary problem of extracting Japanese text from book and comic pages into a text mask. The best text detection system for the proposed method is Manga Image Translator. This problem could use more attention, as the current methods are still far from perfect.

One issue with the proposed method is that it does not start by considering the text itself, a human reads a text one character at a time and can use its position, size and meaning relative to other characters to determine its role in the text, while the proposed method sees only connected components of pixels. A method that is integrated into an OCR system would be able to use this textual information, but would require an OCR engine capable of fine-grained processing on the character level.

Another possibility would be to modify the text line detection of an OCR engine such that it would recognize *furigana* as distinct lines. This would make it possible to implement a simple *furigana* detection mechanism based on the location and size of the detected lines.

11 Conclusion

Detection of *furigana* in Japanese written media is a problem that has not seen much research. In this thesis, I formally define the problem, discuss the challenges related to it and propose a method for detecting the locations of *furigana* in Japanese books and comics based on morphology and connected component analysis. For evaluating this method, I created a new dataset of images with annotations of *furigana*. The evaluations were done using a protocol very similar to the one commonly used for evaluating object detection algorithms, but with some minor tweaks to make it more suitable for the task of detecting *furigana*. Additionally, I have shown that using this method can improve the performance of OCR in texts where *furigana* is used frequently. The method performs well on pages of regular layout but struggles with unusual or complex text layout. While comics and textbooks might require more fundamental changes to the method, it's likely that with some fine-tuning this method can achieve near-perfect results on regular Japanese books.

Appendix A
Dataset

Table 5 lists the books and comics included in the test dataset. The images and annotations, which also includes the development set with 57 images, can be provided on request by contacting nikolajnkb@gmail.com.

TABLE 5: Dataset

Title	Romanized title	Author	Publisher	Nr. of pages	Pages used
		Books			
ようこそ実力至上主義の教室	Yōkoso Jitsuryoku Shijō Shugi no Kyōshitsu e	Shōgo Kinugasa	Media Factory	5	book 3, page 10-14
境界線上のホライゾン	Kyōkaisen-jō no Horaizon	Minoru Kawakami	ASCII Media Works	5	book 1, page 22-26
新機動戦記ガンダムW Frozen Teardrop	Shin Kidō Senki Gandamu Wingu Furozen Tiadoroppu	Katsuyuki Sumisawa	Kadokawa Shoten	5	book 1, page 8-12
魔法科高校の劣等生	Mahōka Kōkō no Rettōsei	Tsutomu Satō	Shōsetsuka ni Narō	5	book 1, page 12-16
ニートだけどハロワにいったら異世界につれてかれた	Nītodakedo Harowa ni Ittara Isekai ni Tsuretekareta	Katsura Kasuga	Media Factory	5	book 1, page 10-14
ガーリー・エアフォース	Gārī Ea Fōsu	Kōji Natsumi	ASCII Media Works	5	book 1, page 12-17
上級へのとびら	Jōkyu e no Tobira	Mayumi Oka et al.	Kurosio Publishers	5	page 166-170
Japanese for everyone	Japanese for Everyone	Susumu Nagara	Gakken	5	page 86-90
		Comics			
ボン・クラージュ！乙女	Bon Kurāju! Otome	Akisato Wakuni	Shogakukan	5	book 1, page 7-11
ダービークイーン	Dābīkuīn	Ashihara Hinako	Shogakukan	5	book 1, page 5-10
カミヨメ	Kami Yome	Tomiyaki Kagisora	Square Enix	5	book 1, page 5-9
ミステリと言う勿れ	Misuteri to Iu Nakare	Yumi Tamura	Shogakukan	5	book 1, page 6-10
Spy × Family	Spy × Family	Tatsuya Endo	Shueisha	5	book 7, page 5-10
屋根裏部屋の公爵夫人	Yaneura Beya no Koshaku Fujin	Fumino Mori	Kadokawa Game Linkage	5	book 1, page 5-9
ペパミント・スパイ	Pepaminto supai	Noriko Sasaki	Hakusensha	5	book 1, page 6-10

REFERENCES

[1] Y. Ogihara, "I know the name well, but cannot read it correctly: difficulties in reading recent japanese names," *Humanities and Social Sciences Communications*, vol. 8, no. 1, pp. 1–7, 2021.

[2] Hayaken, *Eiyū-Ō, Bu o Kiwameru Tame Tensei-Su: Soshite, Sekai Saikyō no Minarai Kishi*. Hobby Japan, 2019.

[3] Y. Kobayashi, *Akkera Kanjinchō*. Shueisha, 2001.

[4] R. Smith, "An overview of the tesseract ocr engine," in *Ninth International Conference on Document Analysis and Recognition*, vol. 2. IEEE, 2007, pp. 629–633.

[5] T. M. Breuel, "The ocropus open source ocr system," in *Document Recognition and Retrieval XV*, vol. 6815. International Society for Optics and Photonics, 2008, p. 68150F.

[6] Y. Du, C. Li, R. Guo, X. Yin, W. Liu, J. Zhou, Y. Bai, Z. Yu, Y. Yang, Q. Dang *et al.*, "Pp-ocr: A practical ultra lightweight ocr system," *arXiv preprint arXiv:2009.09941*, 2020.

[7] M. Liao, Z. Wan, C. Yao, K. Chen, and X. Bai, "Real-time scene text detection with differentiable binarization," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, 2020, pp. 11 474–11 481.

[8] A. Veit, T. Matera, L. Neumann, J. Matas, and S. Belongie, "Coco-text: Dataset and benchmark for text detection and recognition in natural images," *arXiv preprint arXiv:1601.07140*, 2016.

[9] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European Conference on Computer Vision*. Springer, 2014, pp. 740–755.

[10] D. Karatzas, L. Gomez-Bigorda, A. Nicolaou, S. Ghosh, A. Bagdanov, M. Iwamura, J. Matas, L. Neumann, V. R. Chandrasekhar, S. Lu *et al.*, "Icdar 2015 competition on robust reading," in *13th International Conference on Document Analysis and Recognition*. IEEE, 2015, pp. 1156–1160.

[11] D. Karatzas, F. Shafait, S. Uchida, M. Iwamura, L. G. i Bigorda, S. R. Mestre, J. Mas, D. F. Mota, J. A. Almazan, and L. P. De Las Heras, "Icdar 2013 robust reading competition," in *2013 12th International Conference on Document Analysis and Recognition*. IEEE, 2013, pp. 1484–1493.

[12] A. Shahab, F. Shafait, and A. Dengel, "Icdar 2011 robust reading competition challenge 2: Reading text in scene images," in *International Conference on Document Analysis and Recognition*. IEEE, 2011, pp. 1491–1496.

[13] C. K. Ch'ng and C. S. Chan, "Total-text: A comprehensive dataset for scene text detection and recognition," in *2017 14th IAPR International Conference on Document Analysis and Recognition*, vol. 1. IEEE, 2017, pp. 935–942.

[14] R. Smith, C. Gu, D.-S. Lee, H. Hu, R. Unnikrishnan, J. Ibarz, S. Arnoud, and S. Lin, "End-to-end interpretation of the french street name signs dataset," in *European Conference on Computer Vision*. Springer, 2016, pp. 411–426.

[15] M. Iwamura, T. Matsuda, N. Morimoto, H. Sato, Y. Ikeda, and K. Kise, "Downtown osaka scene text dataset," in *European Conference on Computer Vision*. Springer, 2016, pp. 440–455.

[16] D. Karatzas, L. Gómez, A. Nicolaou, and M. Rusiñol, "The robust reading competition annotation and evaluation platform," in *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*. IEEE, 2018, pp. 61–66.

[17] J. Ye, Z. Chen, J. Liu, and B. Du, "Textfusenet: Scene text detection with richer fused features." in *International Joint Conference on Artificial Intelligence*, 2020, pp. 516–522.

[18] L. Deng, Y. Gong, Y. Lin, J. Shuai, X. Tu, Y. Zhang, Z. Ma, and M. Xie, "Detecting multi-oriented text with corner-based region proposals," *Neurocomputing*, vol. 334, pp. 134–142, 2019.

[19] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang, "East: an efficient and accurate scene text detector," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 5551–5560.

[20] Z. Qi, M. Kimachi, Y. Wu, and T. Aziwa, "Using adaboost to detect and segment characters from natural scenes," in *Proceedings of Workshop on Camera-Based Document Analysis and Recognition, International Conference on Document Analysis and Recognition*, 2005.

[21] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[22] J.-C. Wu, J.-W. Hsieh, and Y.-S. Chen, "Morphology-based text line extraction," *Machine Vision and Applications*, vol. 19, no. 3, pp. 195–207, 2008.

[23] R. P. dos Santos, G. S. Clemente, T. I. Ren, and G. D. Cavalcanti, "Text line segmentation based on morphology and histogram projection," in *2009 10th International Conference on Document Analysis and Recognition*. IEEE, 2009, pp. 651–655.

[24] B. Zhu, X.-D. Zhou, C.-L. Liu, and M. Nakagawa, "A robust model for on-line handwritten japanese text recognition," *International Journal on Document Analysis and Recognition*, vol. 13, no. 2, pp. 121–131, 2010.

[25] T. Matsushita and M. Nakagawa, "A database of on-line handwritten mixed objects named "kondate"," in *2014 14th International Conference on Frontiers in Handwriting Recognition*, 2014, pp. 369–374.

[26] N. T. Ly, C. T. Nguyen, and M. Nakagawa, "Training an end-to-end model for offline handwritten japanese text recognition by generated synthetic patterns," in *2018 16th International Conference on Frontiers in Handwriting Recognition*. IEEE, 2018, pp. 74–79.

[27] T. Clanuwat, M. Bober-Irizar, A. Kitamoto, A. Lamb, K. Yamamoto, and D. Ha, "Deep learning for classical japanese literature," *arXiv preprint arXiv:1812.01718*, 2018.

[28] N. T. Ly, C. T. Nguyen, and M. Nakagawa, "An attention-based row-column encoder-decoder model for text recognition in japanese historical documents," *Pattern Recognition Letters*, vol. 136, pp. 134–141, 2020.

[29] A. Fujimoto, T. Ogawa, K. Yamamoto, Y. Matsui, T. Yamasaki, and K. Aizawa, "Manga109 dataset and creation of metadata," in *Proceedings of the 1st International Workshop on Comics Analysis, Processing and Understanding*, 2016, pp. 1–5.

[30] K. Aizawa, A. Fujimoto, A. Otsubo, T. Ogawa, Y. Matsui, K. Tsubota, and H. Ikuta, "Building a manga dataset "manga109" with annotations for multimedia applications," *IEEE MultiMedia*, vol. 27, no. 2, pp. 8–18, 2020.

[31] R. Hinami, S. Ishiwatari, K. Yasuda, and Y. Matsui, "Towards fully automated manga translation," *arXiv preprint arXiv:2012.14271*, 2020.

[32] Y. Aramaki, Y. Matsui, T. Yamasaki, and K. Aizawa, "Text detection in manga by combining connected-component-based and region-based classifications," in *2016 IEEE International Conference on Image Processing*. IEEE, 2016, pp. 2901–2905.

[33] W.-T. Chu and C.-C. Yu, "Text detection in manga by deep region proposal, classification, and regression," in *2018 IEEE Visual Communications and Image Processing*. IEEE, 2018, pp. 1–4.

[34] H. Tolle and K. Arai, "Method for real time text extraction of digital manga comic," *Int. J. Image Process*, pp. 669–676, 2011.

[35] B. Piriyothinkul, K. Pasupa, and M. Sugimoto, "Detecting text in manga using stroke width transform," in *2019 11th International Conference on Knowledge and Smart Technology (KST)*. IEEE, 2019, pp. 142–147.

[36] J. Del Gobbo and R. M. Herrera, "Unconstrained text detection in manga," *arXiv preprint arXiv:2010.03997*, 2020.

[37] S. Noriko, *Peppermint Spy*. Hakusensha, 1985.

[38] S. Nagara, *Japanese for Everyone: A Functional Approach to Daily Communication*. Japan Publications Trading, 1990.

[39] H. Ashihara, *Derby Queen*. Shogakukan, 1999.

[40] T.-L. Yuan, Z. Zhu, K. Xu, C.-J. Li, T.-J. Mu, and S.-M. Hu, "A large chinese text dataset in the wild," *Journal of Computer Science and Technology*, vol. 34, no. 3, pp. 509–521, 2019.

[41] S. Enoshima, *Tsundere Akuyaku Reijō Rīzerotte to Jikkyō no Endō-kun to Kaisetsu no Kobayashi-san*. Enterbrain, 2018.

[42] Q. Ye and D. Doermann, "Text detection and recognition in imagery: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 7, pp. 1480–1500, 2014.

[43] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2009.

[44] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 25, 2012.

[45] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[46] J. D. Gobbo and R. Matuk Herrera, "Unconstrained text detection in manga: A new dataset and baseline," in *European Conference on Computer Vision*. Springer, 2020, pp. 629–646.

[47] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International*

Conference on Medical Image Computing and Computer-assisted Intervention. Springer, 2015, pp. 234–241.
[48] R. Padilla, S. L. Netto, and E. A. da Silva, "A survey on performance metrics for object-detection algorithms," in *2020 International Conference on Systems, Signals and Image Processing*. IEEE, 2020, pp. 237–242.
[49] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, "Generalized intersection over union: A metric and a loss for bounding box regression," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 658–666.
[50] T. Endo, *Spy x Family*. Shueisha, 2019.
[51] K. Sumisawa, *Shin Kidō Senki Gandamu Wingu Furozen Tiadoroppu*. Kadokawa Shoten, 2016.
[52] M. Kawakami, *Kyōkaisen-jō no Horaizon*. Kadokawa Shoten, 2008.
[53] K. Kasuga, *Niitodakedo Harowa ni Ittara Isekai ni Tsuretekareta*. Media Factory, 2020.
[54] V. I. Levenshtein *et al.*, "Binary codes capable of correcting deletions, insertions, and reversals," in *Soviet Physics Doklady*, vol. 10, no. 8, 1966, pp. 707–710.
[55] K. Sigsawa, *Kino no Tabi*. ASCII Media Works, 2000.