Katakana to Romaji Translator using Tesseract OCR and OpenCV

Hans Christian Dureza, hans.christian.dureza@eee.upd.edu.ph
Sean Kristian Garibay, sean.kristian.garibay@eee.upd.edu.ph

I. Introduction

The Japanese writing systems use three sets of characters: katakana, hiragana, and kanji. While kanji may be used in the formal form of writing in japanese, there are thousands to choose from and each character may be read differently and may have a different meaning. Hiragana and katakana, having 46 and 48 characters respectively, are the scripts most commonly used by those who are trying to learn the language. They are a syllabary or syllabic scripts, meaning that each character represents a syllable in the spoken language. While hiragana and katakana are mostly similar audibly, hiragana are either used to form (native) japanese words not covered by kanji or for grammatical inflections. This means that while one might be able to read hiragana, one may not necessarily understand the meaning of the word. On the other hand, katakana is the set of characters generally used to transcribe foreign words into japanese syllabically, or it is used to spell out loan words. When it comes to katakana, reading it is exactly what it is.

II. Objectives

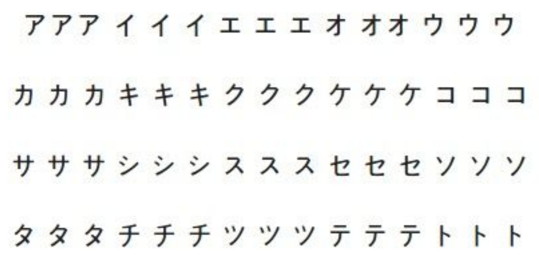
To create a program that makes learning the japanese characters - specifically katakana- easier, more efficient, and interactive. The program takes an image of a word written in katakana and outputs the word phonetically as it is spelled in the english alphabet or romaji

The program is implemented on a Windows PC using python with opencv and tesseract OCR.

III. Methodology

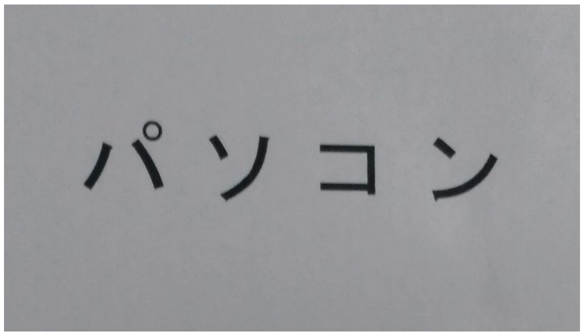
A. Training the Dataset

All 72 Katakana characters were typed into a Google Docs document. Each character were repeated at least two times, with characters from the same family placed in a single line, and a single space was inserted between characters. A screenshot of the document was taken and cropped to include only the katakana characters. A total of four images were produced in total and will be used as the training data set for training. Box files for each image was then generated using tesseract. The box files were then edited to include the correct character translation for each bounding box and to merge several bounding boxes that belong to a single character. Both the data set and the edited box files were then processed using tesseract to create the traineddata file.


ア ア ア イ イ イ エ エ エ オ オ オ ウ ウ ウ
カ カ カ キ キ キ ク ク ク ケ ケ ケ コ コ コ
サ サ サ シ シ シ ス ス ス セ セ セ ソ ソ ソ
タ タ タ チ チ チ ツ ツ ツ テ テ テ ト ト ト

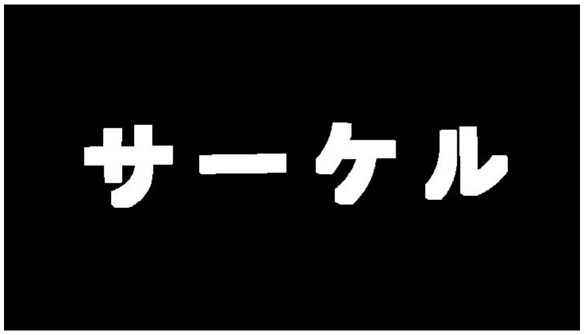
Example of the Data Set

B. Character Extraction



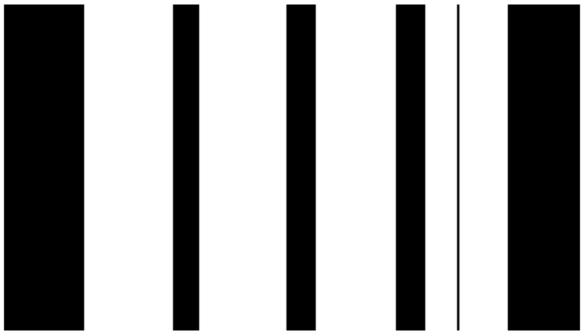
Example Raw Image

- Preprocessing - The image was thresholded and black and white inverted to get the foreground characters. A couple of morphological operations were performed like opening to remove the noise, and dilation to solidify each character.

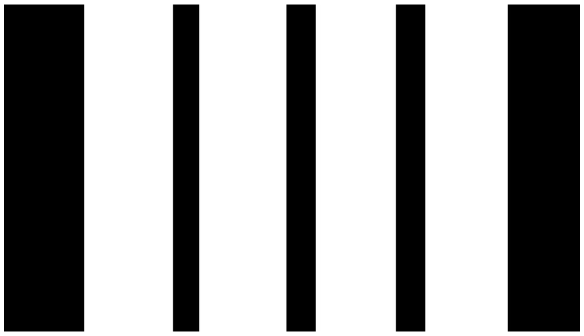


Example Preprocessed Image

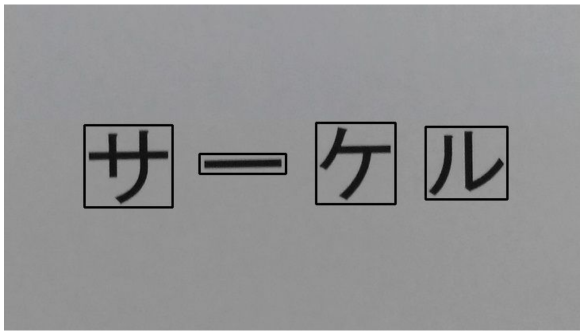
- A vertical projection of the characters wat taken and then projected back into the whole image to get the relative areas of each character.


Example Projection

Since some characters are not entirely connected, the spaces in between the characters was averaged to get the relative idea of the spacing of each character. The black spaces are then compared to this average spacing. Any two areas with black space less than that of 80% of the average will be fused together.


Corrected Character Regions

- Treating the characters inside each region as one whole blob, we took the connected components properties to get the bounding box for each character. These bounding boxes will each define the Region of Interest (ROI) in the image that will be fed into the tesseract OCR.

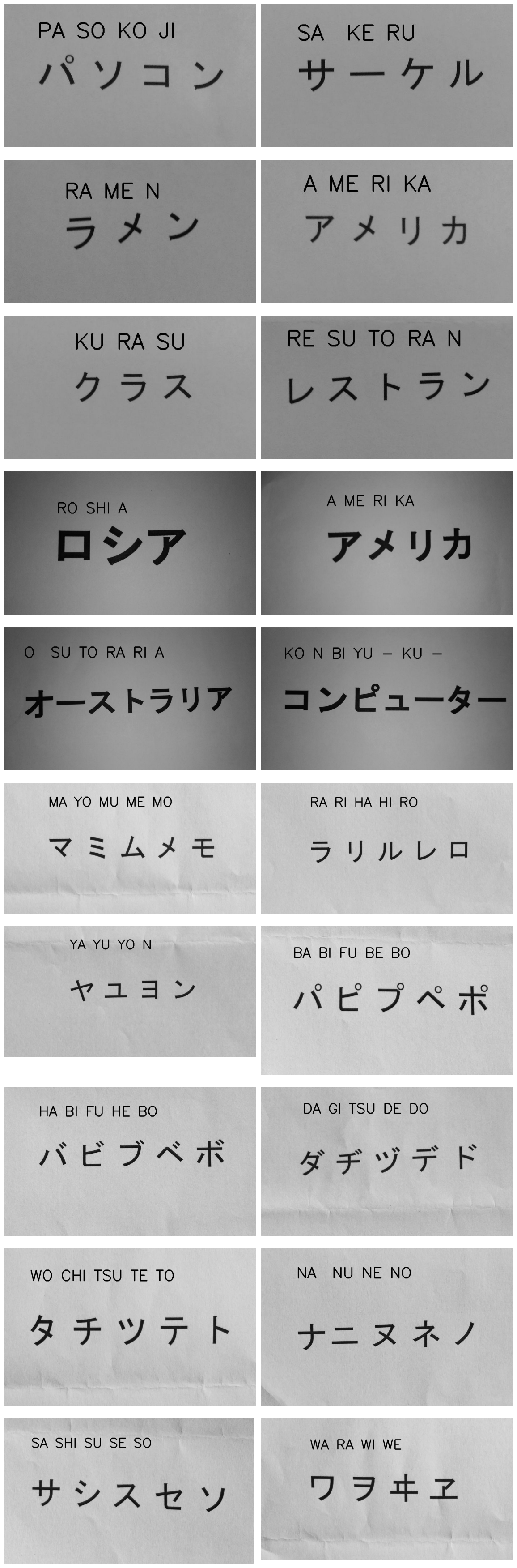


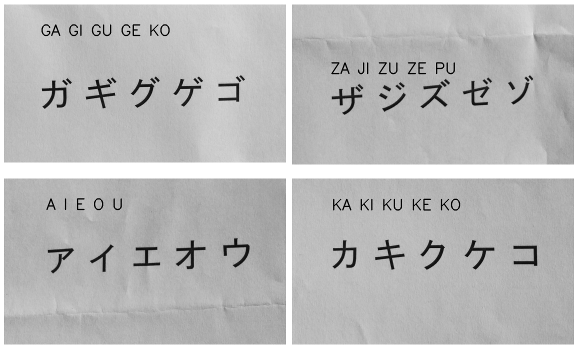
Visualized Bounding Box for Each Character

C. Testing the Program

All of the characters and several katakana words were printed in a single line into a piece of paper and photographed. The image taken were used as the testing data set for the program.

IV. Results and Analysis





Results of All Input Images with Additional Images Representing All Katakana Characters

# of Characters Correctly Identified	89
# of Characters Incorrectly Identified	21
# of Characters Undetected	3
# of Characters in Testing	113
Program Accuracy:	78.7%

V. Conclusion

The project proposed works enough with great accuracy. The being said the program has a handful of limitations to work properly;

- The input must feature a dark text with light background, and single line only
- The program is having issues distinguishing similar looking characters and those that have maru/ten-ten(PI -> BI, TA -> KA ,N -> JI)
- The program also has difficulty identifying horizontal characters such as the character NI and the dash character used for vowel elongation.

VI. Significant References

- Lõmps, J. (n.d.). *Recognising Hiragana Characters Using Optical Character Recognition*. University of Tartu.
- https://www.pyimagesearch.com/2018/09/17/opencv-ocr-and-text-recognition-with-tesseract
- https://www.pyimagesearch.com/2017/07/10/using-tesseract-ocr-python/
- http://pretius.com/how-to-prepare-training-files-for-tesseract-ocr-and-improve-characters-recognition/
- Smith, R. *An Overview of the Tesseract OCR Engine*. Google Inc..
- Vijayarani, S. and Sakila, A. (2015). PERFORMANCE COMPARISON OF OCR TOOLS. *International Journal of UbiComp*, [online] 6(3), pp.19-30. Available at: https://pdfs.semanticscholar.org/c45a/6d961d2cfb2c8ab81380323f169a9f082e32.pdf [Accessed 21 Nov. 2018].