



THE UNIVERSITY OF QUEENSLAND
AUSTRALIA

OFFLINE HANDWRITTEN MATHEMATICAL EXPRESSION RECOGNITION

by Yuting Sun

School of Information Technology and Electrical Engineering,
The University of Queensland.

Submitted for Master of Data Science Capstone 1

26th March 2019

Introduction

I would like to propose an application for converting the image of handwritten mathematical expressions (HMEs) into digital counterparts typeset in LaTeX. Over the years, researchers have made significant progress in traditional character recognition technologies, but recognising non-constrained HMEs remains difficult. In consideration of the problems appearing in related works, I will put extra efforts to strengthen the property of this application by dealing with these intractable cases. The final application will facilitate the workflow of digitalising HMEs not only to students and teachers of mathematical courses but also to engineers and researchers whose work involves plenty of mathematical equations. This project will take approximately nine months to implement the functionality based on the data science process which involves problem formulation, data preparation, modelling and optimisation, evaluation of results and application design.

Background

In this section, I will introduce the background for this project, including the necessity of developing HME recognition program and the categories of handwritten recognition approaches.

As electronic devices become part of the work routines of most professions, the traditional handwriting is becoming less prevalent. Keyboard typing is generally faster than handwriting for text entry, but it turns inefficient for mathematical expression input. LaTeX is known as the de facto standard for mathematical expression input within academia, as it has exceptional strengths in typesetting complex mathematical expressions (Knauff & Nejasmic, 2014). However, users need to spend substantial time typing the LaTeX code and searching in LaTeX documentation for uncommonly used symbols. Hence, a tool that can automatically generate the LaTeX representation from HMEs will prove useful.

For most handwritten recognition applications, two approaches are commonly utilised: online recognition which involves a digital medium, such as a built-in handwritten application on tablets, and offline recognition which uses a scanned image as its input data (Priya, Mishra, Raj, Mandal & Datta, 2016). Studies show that online handwriting recognition achieves higher accuracy compared to its offline counterpart because more information such as the order of strokes can be captured in online case (Dalbir, 2015). Nevertheless, the significance of offline HME recognition cannot be underestimated as mathematical work relies more on pen and paper in most cases. Therefore, I have chosen the offline approach in this project.

Related works

In this section, I will present the existing techniques and compare the current applications.

Researchers in recent years have developed plenty of techniques for HME recognition. For the segmentation of the math symbols, a novel approach outperformed in separating connected digits and obtain a success factor of 97.8% (Alhajj & Elnagar, 2003). For the recognition step, as LeNet (convolutional neural networks) architecture is considered as a successful application on recognising handwritten digits, it has been used in the offline recognition of math symbols and Greek letters and achieved 81.5% accuracy (Bluche, 2010). Also, other classifiers like random forest classifier, support vector machine and recurrent neural network (RNN) are utilised for both online and offline HME recognition (Peng & Zhang, n.d.). The Bidirectional Long Short-Term Memory (BLSTM) for RNN stands out in the existing classifiers by up to 97.8% accuracy for offline recognition and 96.8% accuracy for online recognition (Alvaro, Sanchez & Benedi, 2014). Apart from these, some work has also been done to improve the structure analysis accuracy for special symbols in HMEs. The procedure-oriented algorithm is proposed to group Σ with surrounding symbols based on the spatial relationships (H.J. Lee & M.C. Lee, 1994), and fuzzy logic enables the recognition of subscript and superscript by returning a rank list (Zhang, Blostein & Zanibbi, 2005).

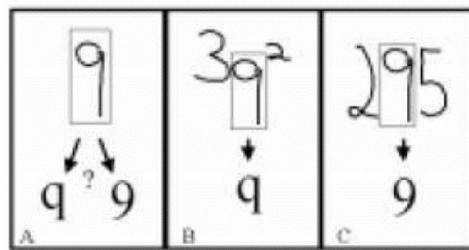
Based on these frontier techniques, some applications have been designed for practical use. For example, Myscript, an online application, allows users to convert HMEs into LaTeX in real time. Despite good performance on classifying simple and intermediate HMEs, it fails on more complex expressions (Schechter, Borus & Bakst, 2017). Also, MathpixOCR, an offline application, enables users to extract text and HMEs from images. It can handle complex printed mathematical expressions accurately, but not with HMEs because of the variations in handwriting (Appendix). Although current applications have already achieved satisfactory results in recognising simple HMEs, they can still improve by addressing equivocal symbols and complex expressions.

Problems posed by offline HME recognition

In this section, I will explain the challenges of recognising HMEs compared to plain text, which are due to the broader range of symbols and spatial relationship in the mathematical expressions. As image pre-processing is essential for offline recognition, I will also explain the problems caused by images with noise data.

According to plenty of related works, ambiguities in mathematical symbols and the sophisticated structure of mathematical formulae hinder the progress of machine recognition of HMEs (Zhang et al., 2017). Apart from Roman letters and digits, the mathematical expressions can also contain the Greek alphabet and plenty of math symbols. As a result, symbols with similar shape can hold different meanings for different contexts. For instance, the “q-9” problem (Miller & Viola, 1998), as we can see in Figure 1. Another good example is the symbol ‘<’, which is the operator ‘lower than’ for $a < b$ but can also present as the bracket in $\langle a, b \rangle$. Also, \prod is the product symbol for an equation

like $\prod_{k=0}^n x_k$, while π indicates the Greek letter pi. Thus, the machine cannot easily recognise these



characters with inherent ambiguities without context.

Figure 1. The q-9 ambiguities posed by different context.

In addition to the ambiguities caused by the original symbol shapes, the various handwriting styles across individuals especially joined and cursive writing style also cause some loss of precision through the segmentation and recognition process. The adjacent characters can connect in multiple ways when the handwriting imposes no constraints. These overlap and noisy strokes can fail the segmentation step (See Figure 2), which may further influence the accuracy of recognition (Jayarathna & Bandara, 2006).

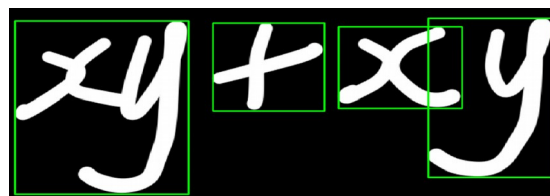


Figure 2. Failure of segmentation for the connected x-y.

Additionally, the mathematical expressions hold more complicated structure compared to text, which leads to the step of structure analysis in HMEs recognition. The structure analysis works by identifying the spatial relationship (i.e., in-line, superscript, or subscript) between symbols based on

their relative positions. However, this spatial relationship can only be identified globally when it involves superscript or subscript (Chan & Yeung, 2000), as shown in Figure 3.

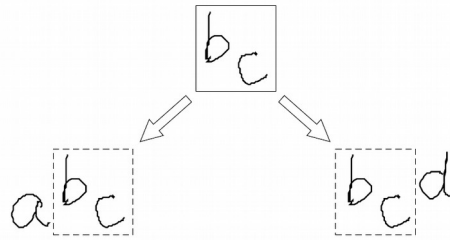


Figure 3. The subscript and superscript cannot be identified locally

Apart from the challenges in dealing with ambiguities and structure analysis, noise data in the image also reinforces the difficulty in offline recognition. Scanned handwritten drafts from different writing instruments and images taken by the camera can cause various problems, such as shadows, fold marks, overexposure, stains and skewed angle. Specific treatment for each of the problems is needed to correct the misleading information in the image. Without special treatment for this problem, the program can fail in the segmentation stage by misclassifying the stained area or shadow points as objects (see Figure 4).



Figure 4. Failure of segmentation because of shadows and fold marks.

Project objective

This project aims to develop an application based on machine learning techniques, and the overall objectives can be divided into three parts: high accuracy for recognising HMEs, short response time to get the output, and the interactivity with users.

Given the challenging nature of the problems, I decomposed the process of HME recognition into four parts: image pre-processing; expression segmentation; symbol recognition; structure analysis. To improve the accuracy of HME recognition, this project will apply different approaches to achieve the following functionalities in each stage:

- Strong noise handling for input images by pre-processing, including shadow removal, ruled lines removal, slant correction and noise removal
- Segmentation of connected symbols by [computer vision segmentation algorithms](#)
- Recognition of handwritten symbols by neural networks in deep learning
- Structure [analysis of complex expressions](#) by building a hierarchical tree indicating the relationship between symbols

For basic equations like arithmetic and algebra, I aim to achieve at least 90% success rate of the output, while for intermediate and advanced expressions including matrices, product and summation formulae, and calculus, I aim to achieve at least 80% accuracy.

As the length of the system response time is a significant performance indicator that greatly affects the users' experience, I also propose to achieve a fast recognition of different HMEs. This objective implies an efficient searching algorithm for the structure tree, limited backtracking and fewer recursions in the algorithm.

Although machine recognition can give satisfactory results, it is still evitable for users to check the results before applying. However, giving the rank list for recognised expressions can help reduce the further workload. This interactive property enables the users to choose a better conversion result when the top one choice is not satisfactory enough.

Proposed solution

Following the data science approach — including problem formulation, data preparation, modelling and optimisation, results evaluation and application design — will address the identified problems and achieve the objectives. The following parts will give the interpretation for each stage.

Step 1: Problem formulation

The design of the program includes a sequence of steps and the form of data changes during the process (Bluche, 2010). Begin by scanning the handwritten image (text with math-zones, graphics and others) and extracting the mathematical expressions; the program will end up with the output of the LaTeX representation. The data representation is summed up in Figure 5 in chronological order.

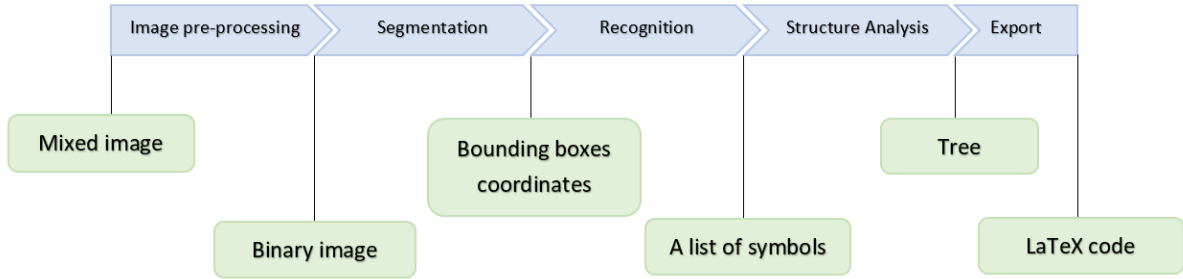


Figure 5. Changes of data representation along the process.

To begin with, image pre-processing can dramatically affect the quality of feature extraction and the results of the analysis in a positive way. For this project, the quality and the size of the image determine the accuracy of the HME segmentation and recognition. On the other hand, segmenting the math-zone from the mixed material can help the optical character recognition (OCR) system focus on recognising mathematical expressions. In order to identify the math-zones from graphics, tables and other interferential information, the morphological operation will play a role in extracting shape-based features and get the segmented chunks. However, research has proved that segmenting math-zones from text is a challenging problem (Chowdhury, Mandal, Das & Chanada, 2003). Therefore, for this project, I will apply the pre-processing techniques to the input images with only HMEs. The segmentation of math-zone will remain as a further improvement after the whole program is completed. The methods utilised for image pre-processing include grey scale processing, binarisation, median filter, and morphological operation (H. Wang, Y. Wang, Lu, Liu, Li & Zhang, 2016).

Greyscale processing aims to equalise the R, G, B colours in the image, which indicates the value of each pixel only carries intensity information. Based on the intensity value of the image, the Otsu's

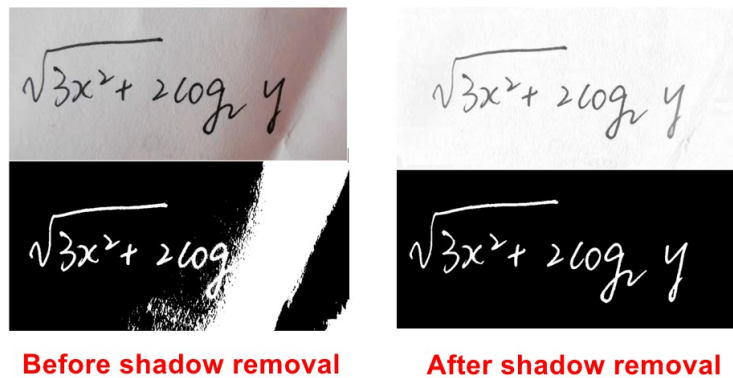


Figure 6. Binarisation results of image before and after shadow removal.

binarisation will be used to generate the binary image, in which the pixel value is either 0 or 255. Otsu method enables the computer to automatically select the threshold for replacing the pixel value according to the image statistical characteristics. However, for an image with a cast shadow, the binarisation will fail to separate the object from the background, as shown in Figure 6. To remove these shadows, the median filter is used for smoothing the image. It works by assigning the central pixel with the median value of pixels under the kernel window. Another common situation is writing with lined papers. This background can be removed by defining two kernels: one for detecting horizontal lines and the other for detecting vertical lines (Vyas, 2018). Moreover, two basic morphological operations named opening and closing are commonly used for noise removal (Open Source Computer Vision, n.d.). Opening means applying erosion followed by dilation, and it outperformed in removing noise dots on the image (see Figure 7). In contrast, closing is the reverse of opening, and it is commonly used for closing small black holes inside the objects in a binary image (See Figure 8).



Figure 7. Opening operation for removing noise dots.



Figure 8. Closing operation for closing inside dots

After pre-processing the image, the output binary image should only contain the clean and clear handwritten trace (see Figure 9). Afterwards, segmentation algorithms can extract the pixels of

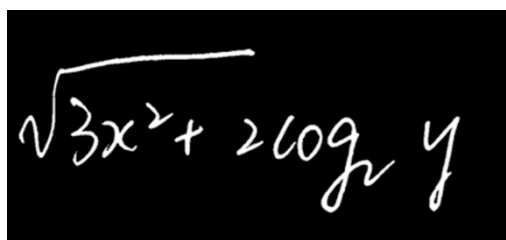


Figure 9. Binary image with clear handwritten trace.

every symbol in the binary image. I have implemented two methods for this segmentation stage: watershed algorithm and finding contours. The watershed algorithm is a stable segmentation process which interprets the image as a topographic landscape, and the watershed boundaries separate the catchment basins (Mathivanan, Ganesamoorthy & Maran, 2014). Before conducting this algorithm, the sure foreground which is a certain part of the objects and the sure background which is a certain part of the black background in the binary image. Then, the uncertain part of the image will undergo the watershed algorithm to find the corresponding boundaries (See Figure 10 & 11). However, in order to find the foreground,

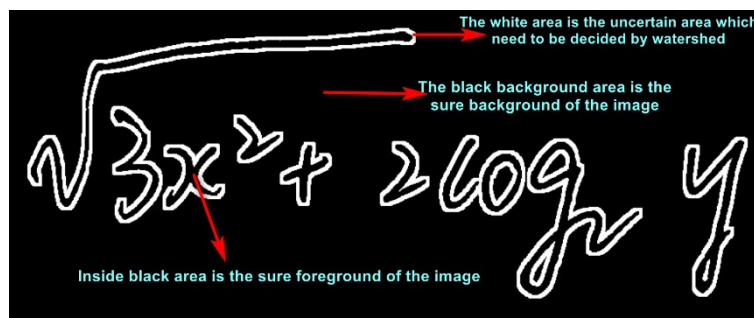


Figure 10. Find the sure foreground and background before implementing watershed algorithm.

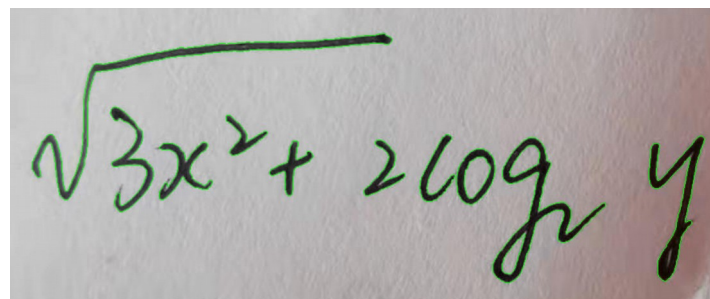
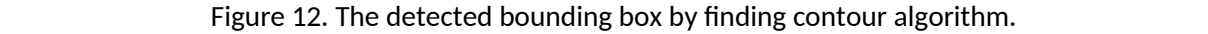


Figure 11. The result of the watershed segmentation.

the distance transformation is involved in calculating the distance between pixel, which requires a manually set threshold for the determination of foreground parts. Hence, the watershed segmentation is highly dependent on the stroke size of the image, and it makes the object boundaries hard to be correctly detected with same parameters as the input images may have different stroke size. Another algorithm is called finding contour. This algorithm can automatically



160

Figure 13. The dendrogram of segmented components.

Step 2: Data preparation

I chose "HASYv2" dataset (Thoma, 2017), and the "Handwritten math symbols" dataset (Nano, 2017) from Kaggle to build the training model for the recognition of symbols.

The HASYv2 dataset has 369 classes including the Latin characters with uppercase (A-Z) and lowercase (a-z), Arabic numerals (0-9), Greek letters, arrows, brackets and so on (See Figure 14). The whole dataset contains 168,233 binary images of size 32x32.

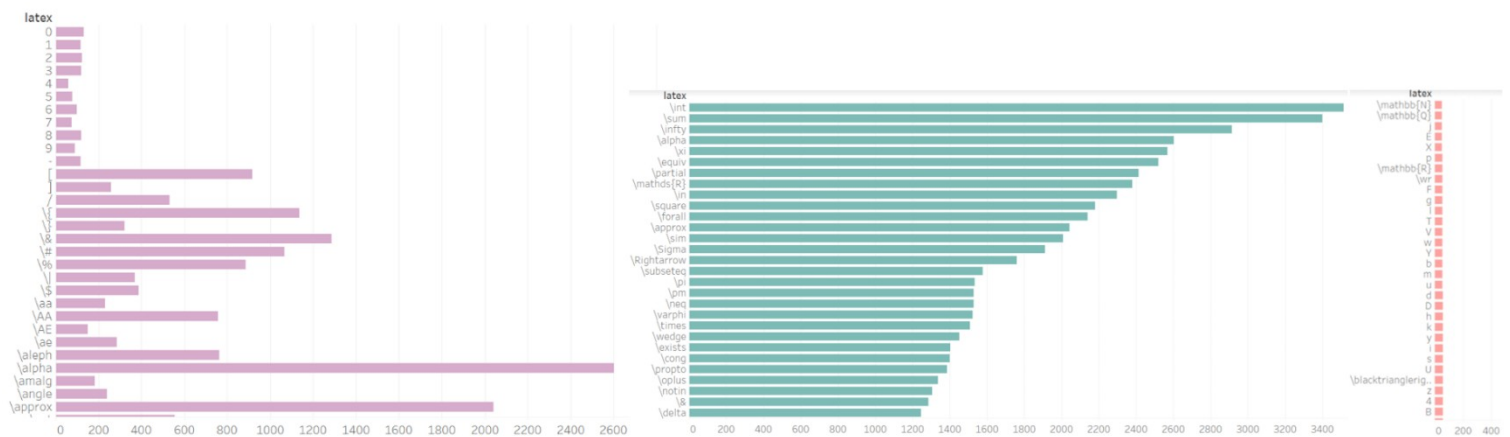


Figure 14. Part of the histogram of the count of LaTeX symbols in the dataset.

However, From the histogram above, it is obvious that the symbols are not evenly distributed. It is essential to balance the dataset before training to avoid the biased prior for categories caused by the neural network (Peng & Zhang, n.d.). The neural network will result in an incorrect statistical conclusion if some categories have higher occurring rates. Therefore, I need to obtain more training samples for the less frequent categories by using random elastic transformations and resample 800 images from the categories with more than 1000 samples. Another problem of this dataset is the missing some common symbols like the equal sign and all the circular function symbols, so I need to complete the dataset by combining with another dataset.

The other dataset for filling the vacancies is the "Handwritten math symbols" dataset, however this dataset has more than 80% repeated samples. For example, the symbol X has a repetitive rate of

21261/26594. After removing the duplicate samples, the final size of the dataset shrunk to 80,000. To combine only the missing samples of HASyV2 dataset, the data augmentation is required as well to balance the sample the dataset. To get better results, I will collect these handwritten symbols from different volunteers to maintain the diversity of handwriting styles. As the image size of the dataset is 45x45, I also need to enlarge the image size of HASyV2 dataset to make sure all the images are of same size without losing information in the images. After data resampling and data augmentation, the final dataset should contain around $(369 + 7) * 800 = 300,800$ jpg images with size 45x45.

Step 3: Modelling and optimization

at the symbol recognition stage, I will compare the accuracy of four different recognition algorithms —Neural network (NN), convolutional neural network (CNN), recurrent neural network (RNN) and random forest classifier (Schechter, Borus & Bakst, 2017).

Firstly, I will use the sigmoid activation for the neural network to transform the feature vector into $x \in [0,1]$ for the single hidden layer. The prediction is the class with the maximum probability of $p(y=i|x; \theta)$, where θ indicates the parameters of the model and i represents one of the 376 classes in the dataset.

Secondly, I will also implement CNN with two (feature-extracting) convolutional layers, two pooling layers and three fully-connected layers. The architecture of CNN is shown in Figure 15. The activation function for CNN is ReLU (rectified linear unit), and the prediction is the LaTeX symbol with the maximum score. As CNN is the most popular model for OCR, this project will mostly focus on this model as the start of the work.

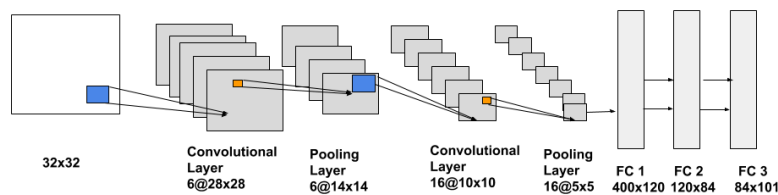


Figure 15. CNN architecture

Thirdly, as mentioned in the section about related works, the Bidirectional Long Short-Term Memory (BLSTM) for RNN is proved to be the best techniques for offline recognition. For this project, it will be an optimisation option for the recognition step. Finally, the random forest classifier will be used as an explorative attempt for OCR, as it is an efficient classifier for face recognition with high accuracy (97.17%) (Kremic & Subasi, 2015).

Step 4: Results evaluation

Based on the project objectives, the evaluation tasks before the application design will be decomposed into five parts: overall accuracy evaluation, confusion matrix for test and training dataset, performance of segmentation, the result of image processing and measurement of response time.

For the recognition accuracy, the testing cases (around 1000) will be collected from the research participants who volunteer to write various given math expressions. The design of the test expressions will be based on a different level of complexity and will also involve the easily mistaken symbols and structures to test the robustness of the program. The accuracy will be estimated by the percentage of correctly converted LaTeX representation.

Apart from this overall accuracy evaluation, I will also use the basic measures derived from the confusion matrix to evaluate the performance of classifiers in the recognition step. These measures include accuracy, precision, sensitivity, specificity and F-score. The performance of segmentation can be evaluated by the success ratio of segmenting components from the test cases. Additionally, the evaluation of image processing will be conducted by measuring the ratio of successfully generated binary images from the test cases.

For the response time, the tolerable processing time is 4s for each expression. The processing time is evaluated from finishing uploading the image to getting the output of corresponding LaTeX representation. The test results will be measured by the average response time and the time difference between the minimum and maximum response time.

Step 5: Application design

After the program testing, an interactive application will be designed for practical use. The preliminary design conception is a web application that allows users to upload an image and gets the LaTeX representation from the image. It will also give a rank list of the optional LaTeX representation, and the interactive GUI will enable the users to modify the LaTeX code if the machine fails to recognise some of the symbols.

Resources

In this section, I will present the language, the tools and the open source library that will be utilised in this project.

Python will be used through the process of program development. HTML, CSS and JavaScript will be used for application design.

OpenCV (Open Source Computer Vision Library) will be used for image pre-processing and segmentation.

TensorFlow, an open source library for large-scale machine learning, will be used for the modelling of recognition step.

As the dataset is more than 300,000 images, the model training process will rely on Hadoop.

WordPress, an UI design tool, will provide a decent interface design of the website.

Proposed schedule

The proposed working schedule is shown as the Gantt chart.

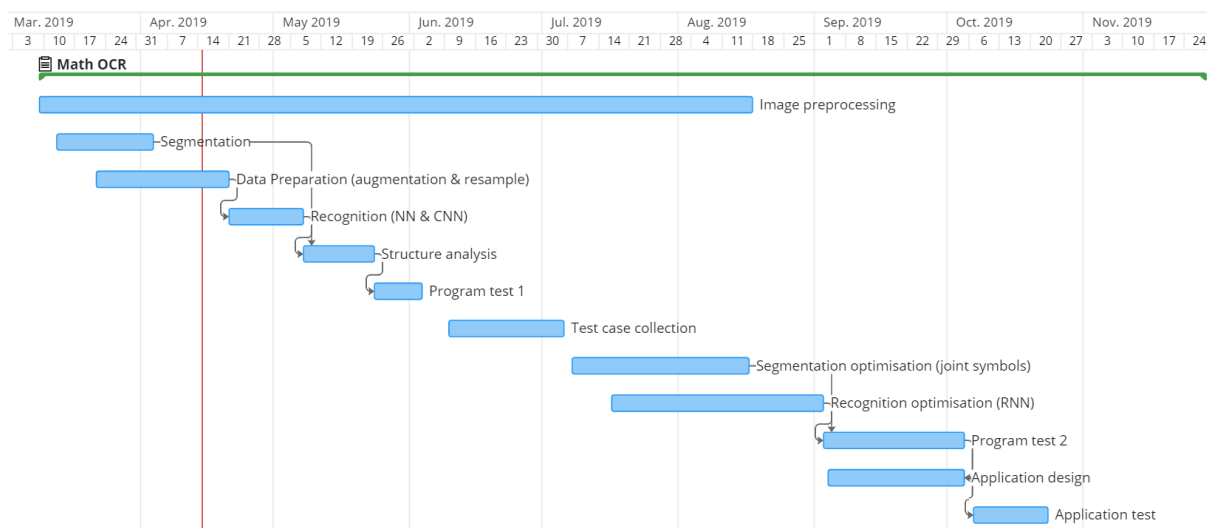


Figure 16. Gantt Chart of the project schedule

As the image pre-processing is independent of all the tasks, it will be managed individually along the process to handle different kinds of image problems appeared during the project.

The program test 1 will be aimed at testing the completeness of the program, that is, to detect the errors and problems that cannot be tolerated, like the failure of recognising most of the common symbols or errors of output LaTeX code structure.

The test cases will be collected during the winter vacation and will be used for program test 2.

Researcher experience and qualifications

As a data science student in UQ, I have fundamental data analysis skills and have in-depth knowledge about machine learning. Apart from this, I am proficient in different computer languages, like Java, R and Python. Also, I have learnt data structure and algorithms in Java, which may help with the tree-structured search algorithm in the structure analysis step of this project. Also, I have experience in web design, which may enhance the stability and performance of the web application.

Costs

This project does not involve any financial costs, the only consumption for this project is time.

The time costs can be divided into five categories: design thinking time, implement time, optimisation time, computational time, test time and backtrack time.

The design thinking time is at the start of the project, which took one week for doing research and making a project plan.

The implement time will be spent on realising the functions based on the project schedule. For the image pre-processing, I have spent time one week to deal with the fundamental problem like grey scale processing and binarisation, and another one week for noise removal. However, the advanced problem may emerge during the process and need further attention as the image problems are more dependent on the practical scenarios. So, the estimated time for image pre-processing is one month in total. For the segmentation part, I have spent three weeks to find an appropriate algorithm for this project and another week to build the hierarchical tree of the segmented components.

Afterwards, the recognition step may need two weeks to implement. The last but not least, as structure analysis involves more complicated algorithms; it will take three weeks to implement. To sum up, the total implement time may take around four months.

The optimisation time is similar to implement time, which involves the implementation of advanced algorithms. The overall optimisation time may take around 1.5 months.

The computational time is mainly about the training time for the classifiers. Studies show that the computational time can be reduced by distributed computing (Basit et al., 2016). The result is shown in Figure 17. As the '4 Practical' indicates the distributed computing results, I can estimate that the training time on Hadoop with 300,000 images is around 6000s, which is less than 2 hours.



Figure 17. Training time vs numbers of images.

The test time involves the time of collecting test cases (3 weeks), the time of testing programs and applications, and the time of summarising the test results. The overall consumption of test time is two months.

The backtrack time is hard to estimate. The backtrack may happen when detecting problems in further steps which indicates the wrong implement in the previous steps. In this case, I need to get back to the previous step and modify the problems. It may lead to the chain reaction and the time can only be estimated based on the practical situation.

Conclusion

To summarise, offline HME recognition is still a technology in their infancy. Based on the existing challenges and the inevitable problems of offline HME recognition, the data science approach guides the whole process of program development. With the help of this step-by-step solution, the decreased incidence of error in preliminary work before recognition and the best approach selected for each stage can help maximise the accuracy of HME recognition. Although this project requires large amounts of time and effort due to the complexity of recognition systems caused by the diversity of handwriting, the successful implementation of this project will significantly benefit the potential users in scientific and engineering disciplines. Besides, this attempt to realise the function of recognising HME images will give some enlightenment for future researchers. Therefore, the value of its role is immeasurable.

References

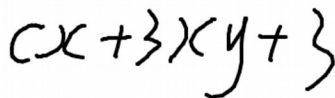
- Alhajj, R., & Elnagar, A. (2003). A novel approach to separate handwritten connected digits. *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings* (pp. 1198–1202). IEEE. <https://doi.org/10.1109/ICDAR.2003.1227847>
- Alvaro, F., Sanchez, J.A., & Benedi, J.M. (2014). Offline features for classifying handwritten math symbols with recurrent neural networks. In F. Alvaro, J.A. Sanchez, & J.M. Benedi (Eds.), *2014 22nd International Conference on Pattern Recognition* (pp.2944-2949). IEEE Computer Society. <https://doi.org/10.1109/ICPR.2014.507>
- Basit, N., Zhang, Y., Wu, H., Liu, H., Bin, j., He, Y., & Hendawi, A.M. (2016). MapReduce-based deep learning with handwritten digit recognition case study. In N. Basit, Y. Zhang, H. Wu, H. Liu, J. Bin, Y. He & A.M. Hendawi (Eds), *2016 IEEE International Conference on Big Data (Big Data)* (pp. 1690–1699). IEEE. <https://doi.org/10.1109/BigData.2016.7840783>
- Bluche, T. (2010). Mathematical Formula Recognition using Machine Learning Techniques (Doctoral dissertation, University of Oxford).
- Chan, K. F., & Yeung, D. Y. (2000). Mathematical expression recognition: A survey. *International Journal on Document Analysis and Recognition*, 3(1), 3–15. <https://doi.org/10.1007/PL00013549>
- Chowdhury, S.P., Mandal, S., Das, A.K., & Chanda, B. (2003). Automated segmentation of math-zones from document images. In S.P. Chowdhury, S. Mandal, A.K. Das, & B. Chanda (Eds.), *Proceedings of the Seventh International Conference on Document Analysis and Recognition* (pp. 755–759). IEEE Computer Society. <https://doi.org/10.1109/ICDAR.2003.1227763>
- Dalbiri, S. K. S. (2015). Review of online & offline character recognition. *International Journal of Engineering and Computer Science*, 4(5), 11729-11732. <http://www.ijecs.in/index.php/ijecs/article/view/3507>

- Jayarathna, U.K.S., & Bandara, G.E.M.D.C. (2006). New segmentation algorithm for offline handwritten connected character segmentation. *First International Conference on Industrial and Information Systems*. 540-546. <https://doi.org/10.1109/ICIIS.2006.365787>
- Knauff, M., & Nejasmic, J. (2014). An efficiency comparison of document preparation systems used in academic research and development. *PLoS ONE*, 9(12): e115069, 1-12. <https://doi.org/10.1371/journal.pone.0115069>
- Kremic, E., & Subasi, A. (2015, January). Performance of Random Forest and SVM in Face Recognition. Retrieved from https://www.researchgate.net/publication/274638232_Performance_of_Random_Forest_and_SVM_in_Face_Recognition
- Lee, H.J., & Lee, M.C. (1994). Understanding mathematical expressions using procedure-oriented transformation. *Pattern Recognition*, 27(3), 447-457. [https://doi.org/10.1016/0031-3203\(94\)90121-X](https://doi.org/10.1016/0031-3203(94)90121-X)
- Mathivanan, P., Ganesamoorthy, B., & Maran, P. (2014). Watershed algorithm based segmentation for handwritten text identification. *ICTACT Journal on Image and Video Processing*, 4(3), 767-772. <https://doi.org/10.21917/ijivp.2014.0111>
- Miller, E. G., Viola, P. A. (1998). Ambiguity and constraint in mathematical expression recognition. *AAAI-98 proceedings* (pp. 784-791). Madison, WI: American Association for Artificial Intelligence.
- Nano, X. (2017). Handwritten math symbols dataset. Retrieved from <https://www.kaggle.com/xainano/handwrittenmathsymbols>
- Open Source Computer Vision. (n.d.). Morphological transformations. Retrieved from https://docs.opencv.org/trunk/d9/d61/tutorial_py_morphological_ops.html
- Peng, Y., & Zhang, Y. (n.d.). Offline Mathematical Expression Recognition. Retrieved from <https://pdfs.semanticscholar.org/e4f7/92e155a5310815716435f5ed7af1360dda8b.pdf>
- Priya, A., Mishra, S., Raj, S., Mandal, S., & Datta, S. (2016). Online and offline character recognition: A survey. *2016 International Conference on Communication and Signal Processing (ICCSP)* (pp. 0967-0970). IEEE. <https://doi.org/10.1109/ICCSP.2016.7754291>
- Schechter, A., Borus, N., & Bakst, W. (2017). Converting handwritten mathematical expressions into LATEX (CS229 Final project paper). Retrieved from <http://cs229.stanford.edu/proj2017/final-reports/5241761.pdf>
- Tatman, R. (2017). Handwritten Mathematical Expressions. Retrieved from <https://www.kaggle.com/rtatman/handwritten-mathematical-expressions>

- Thoma, M. (2017). HASyV2 Dataset (Friend Of MNIST). Retrieved from <https://www.kaggle.com/martinthoma/hasyv2-dataset-friend-of-mnist>
- Vyas, K. (2018, July 22). A box detection algorithm for any image containing boxes. Retrieved from <https://medium.com/coinmonks/a-box-detection-algorithm-for-any-image-containing-boxes-756c15d7ed26>
- Wang, H., Wang, Y., Lu, L., Liu, J., Li, S., & Zhang, Y. (2016). An improved algorithm for symbol segmentation of mathematical formula images. In H. Wang, Y. Wang, L. Lu, J. Liu, S. Li, & Y. Zhang (Eds.), *2016 16th International Symposium on Communications and Information Technologies (ISCIT)* (pp. 461–464). IEEE. <https://doi.org/10.1109/ISCIT.2016.7751674>
- Zhang, J., Du, J., Zhang, S., Liu, D., Hu, Y., Hu, J., ... Dai, L. (2017). Watch, attend and parse: An end-to-end neural network based approach to handwritten mathematical expression recognition. *Pattern Recognition*, 71, 196–206. <https://doi.org/10.1016/j.patcog.2017.06.017>
- Zhang, L., Blostein, D., & Zanibbi, R. (2005). Using fuzzy logic to analyze superscript and subscript relations in handwritten mathematical expressions. In L. Zhang, D. Blostein, & R. Zanibbi (Eds.), *ICDAR '05: Proceedings of the Eighth International Conference on Document Analysis and Recognition* (pp. 972–976), DC, USA: IEEE Computer Society.

APPENDIX

Failure cases for MathpixOCR


$$cx + 3xy + 3$$

1. Input image:

$$(x + 3)(y + 3)$$

Output result:

Correct: $cx + 3xy + 3$

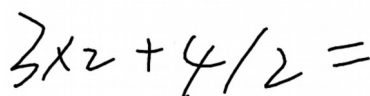

$$x = \frac{-b \pm \sqrt{b^2 - kac}}{2a}$$

2. Input image:

$$x = \frac{-b \pm \sqrt{b^2 - kac}}{2a}$$

Output result:

Correct: $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$


$$3x^2 + 4/2 =$$

3. Input image:

$$3x_2 + 4/2 =$$

Output result:

Correct: $3 \times 2 + \frac{4}{2} = 8$