Understanding Mathematical Expressions from Camera Image

[2]Chun-Yao Wang, [*1,2]Ying-Chin Lin, [1]Han Yuan Tan, [1]Jing-Yun Zeng
[1]Department of Applied Mathematics,
[2]Industrial Ph.D. Program of Internet of Things,
Feng Chia University, Taichung, Taiwan, R.O.C.
[*]yichlin@fcu.edu.tw

Abstract

Recognizing mathematical expressions from document or camera images helps us to understand and process mathematical expressions in scientific and technical documents, while it has been studied for over a decade. Based on previous works, we develop an automatic recognition tool, named EqnEye, which leverages the OpenCV library to perform image processing and the Tesseract tool to recognize mathematical symbols. We also apply correction methods before the recognition stage to improve the recognition accuracy. Experimental results exhibit the success of our recognition method with the correction design. In addition, porting the recognition tool to handy devices can produce more value-added applications.

1. Introduction

Scientific and technical documents are published at an incredible scale, and it's getting more and more time-consuming to retrieve relevant documents and locate targeted terms. Current search engines can rapidly find text-based keywords in plenty of documents, but retrieving relevant images is a challenge work. The *optical character recognition* (OCR) technology converts image-based content to searchable data, where the work of mathematical expression identification and recognition assists in extracting mathematical formula in the images. There are many applications followed by the recognition of expressions. For example, it can be translated with a markup language as LaTeX for the execution on computer algebra systems. It's also valuable in education since one can develop learning guide applications based on the recognition process. Porting the recognition tool to handy devices can produce more value-added applications. In this work, we develop an automatic recognition tool *EqnEye* of printed mathematical expression, while our tool leverages the OpenCV library [1] to perform image processing and Tesseract tool [2] to recognize mathematical symbols.

The rest of this paper is organized as follows. Related literature is given in Section 2. The processing flow and details of each component are described in Section 3. Section 4 presents the experimental results, while Section 5 gives concluding remarks of this work.

2. Related Works

The *mathematical expression recognition* (MER) problem is to recognize the mathematical expression (ME) in the electronic documents or images, while it usually consists of two stages: symbol recognition and structure analysis [3]. The former is to identify mathematical symbols and the latter tries to know the spatial relationships among symbols, where both are challenging. Mathematical notation comprises a great deal of symbols, fonts and typefaces, whereas the small signs, e.g., comma and dot, must be carefully processed to distinguish from noises. Moreover, the analysis of spatial relationships among symbols, particularly for handwritten notations, is more challenging because the ambiguous layout in an expression could confuse the order and presence of operators [4]. In contrast, the recognition of printed ME is much simple. Of special note are the symbols with disconnected signs (e.g., "i", "j", "=") and variable size (e.g., division sing), and the radical sign. Previous works to recognize printed mathematical symbols include PPC [5], kNN, SVM, HMM, etc [6]. After the stage of symbol recognition, the structure analysis decides the spatial relationships for recognized symbols to construct complete expression, while the commonly used methods are tree transformation [4] and geometric features [7].

In spite of many works to tackle MER before, there are studies proposed recently to raise the recognition accuracy. To combine wrongly split text lines, Lin et al. present a learning-based strategy to merge the text lines belonging to an independent formula [8]. Chu and Liu detect mathematical equations in an image document comprised of texts, figures and tables [9]. In addition, Kumar et al. develop a ternary tree based representation to orient mathematical symbols according to their spatial relationships [10]. These works are good reference to our development, and two stages are applied to tackle MER. In symbol recognition, we leverage Tesseract OCR [2] to recognize mathematical symbols; in structure analysis, the geometric features of bounding boxes are used to clarify the spatial relationships among recognized

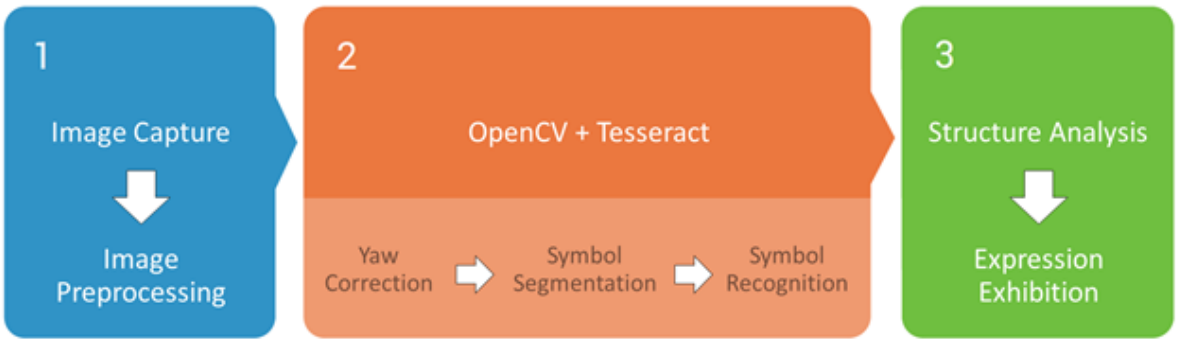

Figure 1. Flow chart of our MER tool

symbols and merge the symbols one by one to acquire the complete expression.

In addition, the image thresholding is a critical step to the image analysis, and it deeply affects whether the symbol recognition is correct in our tool. Previous works utilize the image features to develop thresholding method, including histogram shape, measurement space clustering, entropy, object attributes, spatial correlation, as well as local gray-level information [11]. Otsu method considers the minimum of intra-class variance, and is classical for the image thresholding to extract the internal characteristics of *region of interest* (ROI) [12].

3. Material and Method

Automatically recognizing ME is an important approach for extracting the mathematical meaning from electronic documents and images in scientific and engineering fields. Here, we develop the tool EqnEye for MER, which captures the image with a ME inside, recognizes each symbol as well as analyzes the spatial relations among these symbols. Figure 1 shows each stage of EqnEye and the detail is as follows.

3.1. Image capture

Easy-access devices for image capture, e.g., mobile phone, tablet and smart glasses, assist us in capturing the ME images. Figure 2 shows an image captured from the text book, and we will use it as a reference example throughout this paper.

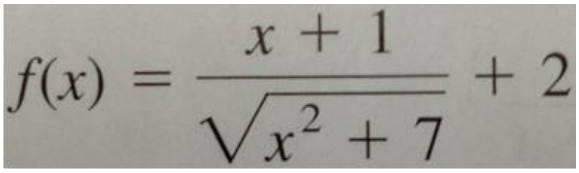

Figure 2. Image captured by camera

3.2. Image preprocessing

The captured image requires some preprocessing steps to mark the ROI, i.e., mathematical characters in the image, so that we can segment each individually. First, the source image is converted to greyscale, and next, we perform the *median filter* with 5*5 to reduce noises but preserve edges. Subsequently, we convert the grey-level image to black-and-white, i.e., *binarization*, which is a simple way of separating the ROI from the background. The threshold to separate the two classes of black and white pixels is automatically given by the Otsu method [12] according to the minimum of intra-class variance, while Figure 3 is the binarization result of Figure 2.

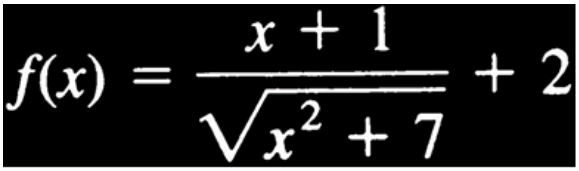

Figure 3. Binarization image

3.3. Yaw correction

When utilizing the camera to take a ME picture, the captured image may be rotated due to lack of stable support to user hands. The skew characters not only reduce the recognition accuracy but also make the estimation of spatial relationship among characters fail. Consequently, we need to correct the binary image before the segmentation stage. In the three-dimensional coordinate system, there are three kinds of rotations: *roll*, *pitch* and *yaw*. In this work, we consider the yaw rotation only because it's most common when using the camera on the handheld device. There are some works for the skew detection, including projection profiles analysis, nearest neighbors, *Hough transform*, etc [13]. Here, we use a heuristic to determine the rotation angle and rotate the binary image accordingly.

Since the text book or paper is static and user is motionless, it's reasonable to assume that the rotation angle is small, say smaller than 45 degrees. Each ME has the character of "=", thus it can be a clue of the rotation angle. We then create bounding box for each character of ME from left to right, which divides "=" into two different bounding boxes. If there is no skew, two bounding boxes for the equal sign should contain

only white pixels. Now we scan the bounding boxes one by one from the left, and once a bounding box BB satisfies the following two conditions, it contains a part of the symbol "=", i.e., a bar. (1). $length_x$(BB) > $length_y$(BB), where $length_x$(BB) and $length_y$(BB) are the lengths of BB in the x and y-direction, respectively; (2). There are two successive BBs of the same ratio N_w(BB)/N(BB), where N(BB) is the number of pixels in the BB and N_w(BB) is the number of white pixels the BB has. The condition (1) is set because we assume that the rotation angle is small enough, while the condition (2) helps to recognize the equal sign well. If no such BB, the image needs no yaw correction and the next stage is applied. Otherwise, we pick two end points (x_1, y_1), (x_2, y_2) of the bar in the BB and compute the rotation angle $\theta = \tan^{-1}((x_2-x_1)/(y_2-y_1))$. If $\theta < 0$, then the image is rotated θ degrees counterclockwise; otherwise, it is rotated θ degrees clockwise. The experiment in Section 4 will exhibit how the yaw correction affects the recognition accuracy.

3.4. Symbol segmentation

In this phase, each symbol is within a bounding box shown by Figure 4, which fully encloses the symbol and is obtained by the projection analysis [12]. For the clarity, the Canny edge detector [12] is applying to the binary image in Figure 4.

Figure 4. Bounding box for each symbol with Canny edge detector

At first glance, each bounding box fully encloses the mathematical symbol; in fact, it contains individual glyph only and hence needs further correction. There are two types required to be modified here. One is that a mathematical symbol is wrongly divided into more bounding box, i.e., "i", "j", "=", "÷". By the geometric feature and position, we can find out related bounding boxes and outline the real bounding boxes for these symbols. The other is that a bounding box could contain nothing, for example, there are two bounding boxes to the number 0 and one box is within the other one. The inside box contains nothing and should be removed. After this phase, we have the one to one mapping between the bounding box and mathematical symbol.

3.5. Symbol recognition

We leverage the Tesseract tool [2] to recognize mathematical symbols. Tesseract is an open source OCR engine and its development has been sponsored by Google since 2006. We integrated it into our system to identify the symbol within every bounding box.

3.5. Structure analysis

After the last phase, we obtain a sequence of symbols, and this phase is to decide the spatial relationship among these symbols enclosed by bounding boxes (BBs). Initially, BBs are sorted in ascending order based on the 2-dimensional coordinate of the left-up corner of the box, i.e., (a, b) in Figure 5. We follow the idea in [4] but detect three compounds of radical sign, fractional sign and exponentiation relation here. The radical sign is easy to be detected since its BB contains more BBs. The detection of fractional sign is also simple, because its BB is a bar and the BBs behind it lie to the either up or low of it. To identify the exponentiation relation is slightly complicated, and in our approach, it must satisfies that $b + t\mathrm{H} > b' + t\mathrm{H}'$ with $t = 0.5$, where b, b', H, H' are shown by Figure 5. The condition considers the y-coordinates of centers of the base and exponent. Though Zanibbi *et al.* considered more complicated cases and a range of t in [0, 0.5] for handwritten mathematical notations [4], here we empirically set $t = 0.5$ for detecting printed symbols.

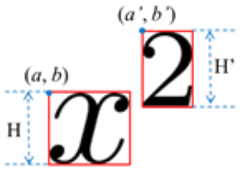

Figure 5. An example of 2D coordinate of bounding boxes

Following the recognition of the structure and compound symbols, an ME parse tree can be constructed. Figure 6 is an example tree for the image in Figure 2.

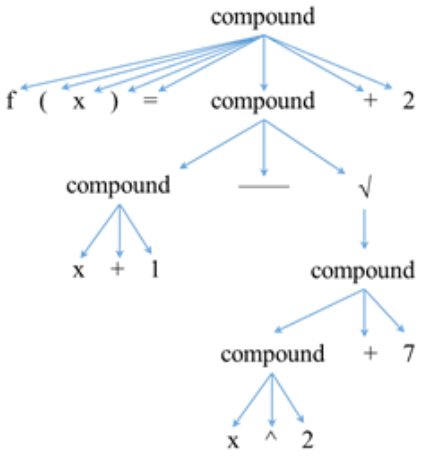

Figure 6. An example of an ME parse tree

3.5. Expression exhibition

To get the ME, one needs to merge each node of the ME parse tree from button to up, where we add parentheses to each compound node to avoid ambiguity. The output can be a string or the LaTeX form, which is convenient for further process.

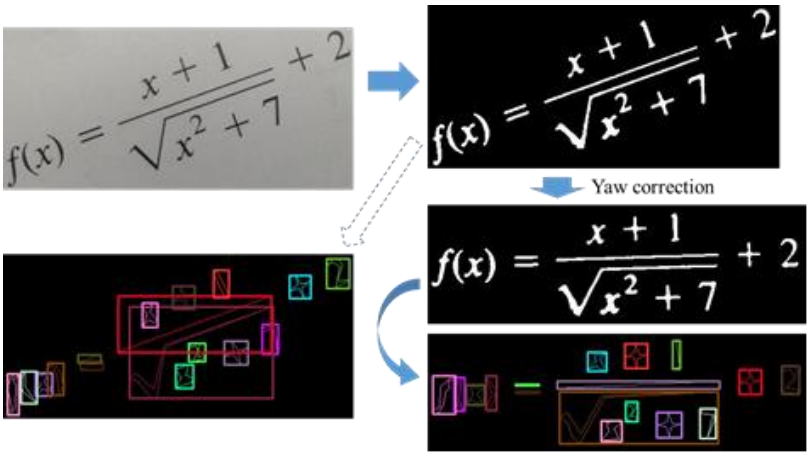

Figure 8. An example of a skew expression and the yaw correction

4. Experiments

All experiments are conducted on NVIDIA GeForce GTX760 with the global memory 4GB, which is mounted with the Intel Core i7 3.6 GHz of PC machine. Take example in Figure 2 as the input, our MER tool identifies all symbols and returns the string "(f (x) = ((x + 1) / (((x ^ 2) + 7) ^ (1 / 2))) + 2)", which are the correct expression of the source. Now, we take a skew picture as shown by Figure 8. Without the correction, the bounding boxes can be marked normally but many bugs could appear. First, Tesseract incorrectly recognizes almost all symbols, except the symbols "f", "(", ")", "x" as well as "2". Next, the spatial relationship among symbols is in disorder. It hardly performs segmentation correction and structure analysis in Sections 3.4 and 3.6, respectively. With the yaw correction introduced in Section 3.3, our program automatically rotates the binary image of the skew expression by 19 degrees clockwise, and the processing steps can be normally applied. In the example of Figure 8, our tool accurately recognizes all symbols and their spatial relationship after performing the yaw correction.

Besides, we test more equations from textbook under different considerations: normal, skew and shadowy images (Figure 7). In the experiments, the rotation angle is less than 10 degree, and we shadow an equation by at least fifty percentage. We have totally 10 groups of equations, while each group contains 3 images corresponding to three different considerations. For an equation under a specific consideration, we test one time, and call it success if it can recognize the equation accurately; otherwise, it is fail. For failed recognitions, we also examine the major reasons in the processing flow. Table 1 is the experimental results.

Table 1 exhibits the high success rate, say 90%, when the equation image is captured in a normal situation. The only one miss in the normal consideration results from the fail in the structure

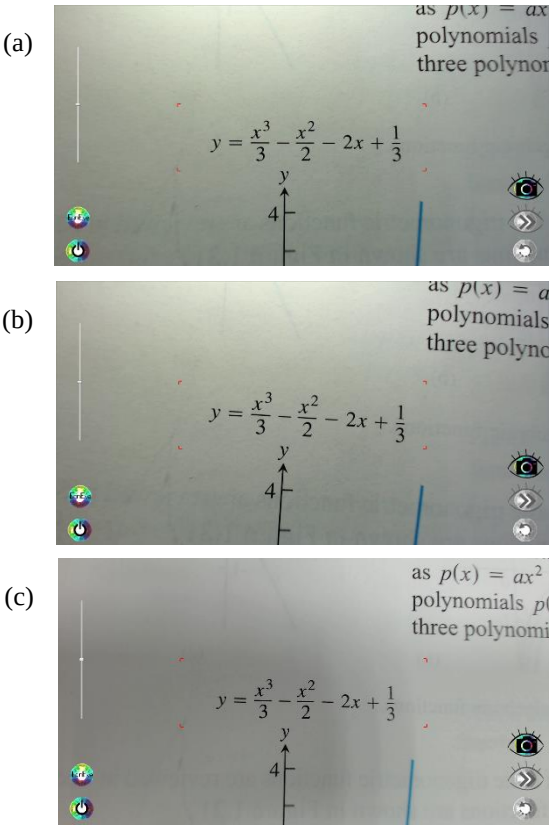

Figure 7. (a) Normal, (b) skew and (c) shadowy images from camera

analysis. Moreover, when the image is skew, we also receive good performance. Two fails come from the false identification of the equal sign in the stage of yaw correction. Finally, we get the worst result if the equation is covered by shadow, where the recognition almost crashes in the image processing. Since the binarization is done by the Otsu method, it hardly discriminates the characters of equation from the background affected by the shadow.

There are some reasons such that our tool acquires the high rate of success. The denoising and

binarization stages could destroy much of the symbol, which is more serious to the image of low resolution. Our tool copes with the high resolution image of 1920*1080 to mitigate the incomplete symbols. Once the symbol is accurately marked as the input, the recognition of Tesseract engine has good quality. The printed expression also reduces the complexity of the structure analysis. On the other hand, though the resolution of captured image is high, the execution time of our tool is within 400 milliseconds, where half the time is spent on the phase of image preprocessing.

		Three Considerations		
		Normal	**Skew**	**Shadowy**
Success		9 (90%)	8 (80%)	5 (50%)
Fail	Image processing	0	0	4
	Yaw correction	0	2	0
	Segmentation correction	0	0	0
	Character recognition	0	0	1
	Structure analysis	1	0	0

Table 1. Experimental results

5. Conclusion

As large amounts of scientific and technical documents are published, it's getting more and more time-consuming to retrieve relevant documents and locate targeted terms. Current search engines can look for text-based keywords in the planet-size of documents; however, it's hard to retrieve relevant images in a large-scale digital libraries. The OCR technology translates image-based content into readable and searchable data, where the mathematical expression recognition has been studied for over a decade. Based on their works, we develop an automatic recognition tool EqnEye which leverages the OpenCV library to perform image processing and Tesseract tool to recognize mathematical symbols. We also apply yaw and segmentation corrections before the recognition stage to improve the recognition accuracy. Experimental results exhibit the success of our correction methods to enhance the accuracy. Moreover, the simulation shows great influence of the shadow. A future work is to moderate the shadow effect by the technique in the image processing.

Acknowledgements

This work is partially supported by the Ministry of Science and Technology under contract number MOST 104-2633-S-035-001.

References

[1] OpenCV, http://opencv.org/

[2] tesseract-ocr, https://github.com/tesseract-ocr

[3] K.-F. Chan and D.-Y Yeung, "Mathematical Expression Recognition: A Survey," *International Journal on Document Analysis and Recognition*, vol. 3, no. 1, pp. 3-15, 2000.

[4] R. Zanibbi, D. Blostein, and J.R. Cordy, "Recognizing Mathematical Expressions using Tree Transformation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 11, pp. 1455-1467, 2002.

[5] A. Raja, et al., "Towards a Parser for Mathematical Formula Recognition," *Mathematical Knowledge Management*, vol. 4108 of LANI, pp. 139-151, 2006.

[6] F. Álvaro and J.A. Sánchez, "Comparing Several Techniques for Offline Recognition of Printed Mathematical Symbols," *20th International Conference on Pattern Recognition (ICPR)*, Istanbul, Turkey, 2010.

[7] Y.-S. Guo, L. Huang, and C.-P. Liu, "A New Approach for Understanding of Structure of Printed Mathematical Expression," *International Conference on Machine Learning and Cybernetics*, Hong Kong, China, 2007.

[8] X. Lin, et al., "A Text Line Detection Method for Mathematical Formula Recognition," *12th International Conference on Document Analysis and Recognition (ICDAR)*, Washington, DC, 2013.

[9] W.-T. Chu and F. Liu, "Mathematical Formula Detection in Heterogeneous Document Images," *2013 Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, Taipei, Taiwan, 2013.

[10] P.P. Kumar, A. Agarwal, and C. Bhagvati, "A Knowledge-Based Design for Structural Analysis of Printed Mathematical Expressions," *8th International Workshop on Multi-disciplinary Trends in Artificial Intelligence*, vol. 8875 of LNCS, pp. 112-123, 2014.

[11] M. Sezgin and B. Sankur, "Survey over Image Thresholding Techniques and Quantitative Performance Evaluation," *Journal of Electronic Imaging*, vol. 13, no. 1, pp.146-168, 2004.

[12] J.R. Parker, *Algorithms for Image Processing and Computer Vision, 2nd ed.*, Wiley, 2010.

[13] S. Li, Q. Shen, and J. Sun, "Skew Detection using Wavelet Decomposition and Projection Profile Analysis," *Pattern Recognition Letters*, vol. 28, pp. 555-562, 2007.