Proceedings of the IASTED International Conference
HUMAN-COMPUTER INTERACTION
November 14-16, 2005, Phoenix, AZ, USA

ENABLE BETTER ACCESS TO MATH FOR BLIND AND VISUALLY IMPAIRED INDIVIDUALS – A MATH EXPRESSION READER

Wenfeng Li and Baoxin Li
Department of Computer Science & Engineering
Arizona State University, Tempe, AZ 85281
USA
Wenfeng.Li@asu.edu, Baoxin.Li@asu.edu

ABSTRACT
Current optical character recognition (OCR), speech synthesis, and video technologies have made it possible to build reading aid devices for blind and visually impaired readers. However, most existing systems can read only plain text from printed materials. For scientific books such as mathematical or engineering textbooks that are full of equations, formulae, tables, and graphical charts, a conventional OCR-based system will not be able to provide the desired assistance. This paper presents a completely automatic method for converting a scanned or captured mathematical expression into speech. In addressing the challenges in automatic symbol recognition, we present several new techniques that make the proposed system more robust. The symbol-to-speech conversion is facilitated by utilizing a MathML-compliant player.

KEY WORDS
Interaction design for people with disabilities, formula recognition, MathML

1. Introduction

Most people read through visual sense, but such capability is partially or completely limited for the blind and visually impaired readers, who typically read through tactile (touching) or auditory (hearing) senses. Braille is universal tactile media for blind person, which enables blind persons to read and write text. Since most scientific materials include mixed text, formulas, charts and graphs, how to enable blind readers to understand these materials has been a challenging problem. There are many standards for mathematical expressions in Braille. For example, Nemeth code [11] is most commonly used in the U.S., and Dots-plus [12] is a 2-D Braille math format, which is still in experimental stage. In either case, aids from sighted person are typically required to translate existing materials into Braille.

In the case of reading mathematical expressions, with a Braille-enabled device, the problem seems to be partially solved as long as the expressions are first converted to Braille via some means. However, this conversion is typically time-consuming if it is done by humans, and there is no existing complete system that can achieve this automatically and reliably, although there have been much reported research on automatic method for automatic recognition of mathematical expressions. Additionally, as a matter of fact, recent technology development may render other means other than Braille more appropriate. For example, studies from [1] show that, in the case of receiving and perceiving information, the auditory sense is two orders of magnitude faster than the tactile sense. Also, in [2], J. Halliday discussed the strengths and weaknesses of speech and Braille from cognitive psychology point of view and stressed the importance of combining multiple forms of media (speech, Braille, and tactile graphics) for blind student education. This becomes feasible with the development of new video, audio and computer technology in recent years, especially the speech synthesis and OCR techniques. Nowadays, OCR-based reading systems for the blind population are commonly available for books.

One of the remaining problems is how to make mathematical expressions accessible to a blind reader, which has not been handled by existing commercial OCR systems. One core task in achieving such an objective is the automatic recognition of printed mathematical expressions, which has been heavily researched since the 1960s and many methods were published. Among the well-known methods are the projection profile cutting [3, 4, 5], attributed string grammars [14, 15], structure specification schemes [16], graph rewriting [17], stochastic grammars [18], and procedural translation [19]. A survey of these and other methods can be found in [6]. It would be difficult to declare any of the methods as the best, since each has its advantages and disadvantages. Although many methods claim to have accuracy of over 90%, even a single error may be fatal to the whole recognition since a lexical error can cause the total failure of syntactic and semantic analysis, which is a core step in most of the methods.

In this paper, we present a complete method for converting printed mathematical expressions into speech, to enable a blind reader to read through equations, formulae, etc that are ubiquitous in scientific books. The

proposed method employs the special features of mathematical expressions to circumvent the difficulty in syntactic and semantic analysis: we simply provide a speech representation of the underlying mathematical expression without explicit analysis of the meaning of the expression. The interpretation is up to the reader, who is presumably in a better position to understand the meaning of the expression. Consequently, one advantage of the proposed method is that the reliability of the system does not rely on the correctness of lexical processing. The reader may detect an error and/or solve the ambiguity through his own judgment.

The rest of the paper is organized as follows. In Section 2, we describe the proposed method, with accompanying examples to illustrate key steps. Section 3 presents additional experimental results, and Section 4 concludes with discussion on continuing and future work.

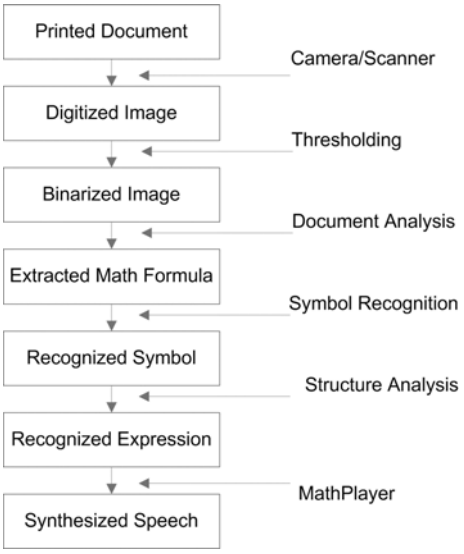


Figure 1. Data flow and major processing steps in the proposed method.

2. Proposed Method

The proposed method is a system for converting printed mathematical expressions into speech so that a blind reader can "read" them through listening. The major processing steps of the proposed system and the data flow are illustrated in Figure 1. The system digitizes the input page, a printed document containing math expressions, via a scanner or a digital camera. After some pre-processing including the segmentation of the math expressions from other texts, symbol recognition and structure analysis are applied to the extracted region. Further syntax and semantic analysis can be performed without any transforming since most of the grammar operations use tree structure and XML DOM (Document Object Model) can present MathML string as a tree structure. Finally, the recognized expression is formatted into a MathML string, and MathPlayer is utilized to read out the math expression. Note that the MathPlayer is used here only for illustration purpose and for fast prototyping. In a practical system, it is not difficult to develop a stand-alone symbol-to-speech subsystem. Details of these processing steps are explained in the following sub-sections.

2.1 Image Capture and Pre-processing

The prototype system is implemented on a personal computer platform with commodity peripherals , and the majority processing is done via software. This would facilitate easy adoption of the developed technology, assuming that the target user, a blind reader, will have access to a reasonably equipped PC.

Image Acquisition: The first step is to acquire the printed document into the system as a digitized image. Two approaches are used in our experiments, video-camera-based and scanner-based. PC-based video camera and digital camera are very popular nowadays and are inexpensive. Thus a camera-based acquisition module would be preferred for portability of the system, although a scanner-based one would easily provide image of better quality and resolution. For a camera-captured image, we first obtained a binary image through thresholding the gray-level input image. For scanner-based acquisition, we use only a low resolution of 100dpi. Currently, emphasis is on building a complete end-to-end system, and we have to take extra measures to ensure the captured data is clean without much noise. So one of the major pre-processing steps is to detect and fix any potential misalignment of the captured document, as detailed in the below.

Document de-skewing: Even if cares have been taken during the acquisition to align the document, a minor residual skew may still exist. Most OCR systems require horizontally aligned text and are sensitive to skew angles. This will also be true in our algorithm for symbol and equation recognition since our system is working on certain baselines, which require all text lines are properly aligned. A simple anti-skew approach is to utilize the homogeneous feature of aligned texts. Projection profile is the most straightforward method for skew detection and it works in the following way [13]. First, we count the number of black pixels of each row as a horizontal projection profile. It is easy to imagine that, for a document image without skew, this profile would reach its peaks for text lines and valleys for space lines. For a document with minor skew angel, the contrast between the peaks and the valleys is reduced. So the problem of de-skewing turns out to be an optimization problem where the sum of squared profile function is used as the objective function and the task is to maximize the objective function with respect to the rotation angle. An exemplar curve of the objective function with a document image originally skewed by -5.85° is shown in Figure 2, where the global maximum at 5.85°, indicating the skew angle. After the skew angle is detected, de-skewing is

straight forward: we will simply undo the skew by rotating the image by the same angle at the opposite direction.

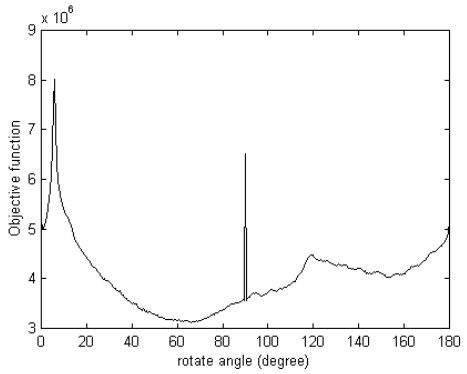


Figure 2. Projection profile: the objective function computed from the projection profile, with respect to the rotation angle.

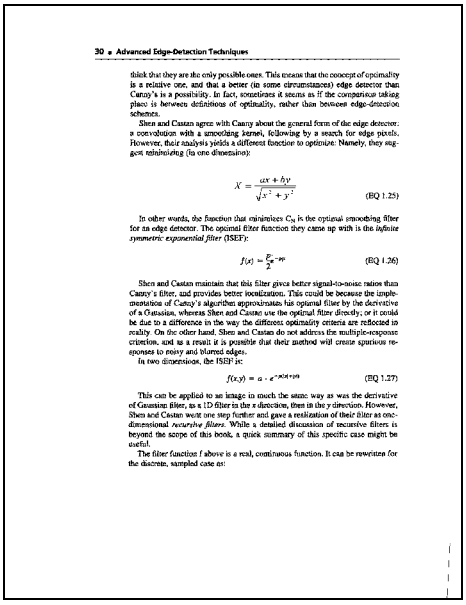
Although projection profile is simple and effective, it is computational expensive since the image must be rotated many times, some improvements can reduce the computation such as downsampling the image first and detecting a coarse skew angle at the lower resolution first and then refining in the original resolution.

2.2 Segmentation and Formula Extraction

Before recognition can be applied to an object, either texts or formulae, we need to first segment the different components so that different processing can be applied accordingly. A run length smoothing algorithm from [7] can turn the image into blocks of components as illustrated in Figure 3. The text blocks and formula blocks can be classified according to their different characteristics. For example, text lines are grouped into thin and long lines together, while formulas are grouped into shorter blocks and the margins above and below are normally larger than text line spaces. To avoid the possible interference of other graph blocks, the extracted candidate formula blocks will be further verified in the later recognition stage: If a block cannot be recognized to be a set of mathematical symbols, they will be rejected.

2.3 Symbol Recognition

It is natural to consider choosing commercial OCR software for the recognition task in such a system. However, commercial OCR components are designed to recognize texts and trained to understand English literatures. In math expressions, many different fonts and glyphs are in use. In addition, the symbols may stand alone and do not conform to the English syntax according to some context, which is typically exploited in the OCR algorithms. Therefore, we need to design our own OCR component. To this end, we first apply the component labeling method of [8] to the digitized image to get connected components, and then we adopt the feature vector method as described in [9], as follows. A feature vector for each symbol is defined as a 27-dimension vector: evenly divide the symbol into 5x5 subdivisions; the percentage of black pixels in each subdivision makes up the first 25 elements of the vector; the whole percentage of black pixels and the ratio of block height to width are put into the remaining two elements. Euclidean distance is used to calculate the difference between two characters for recognition. In our initial testing, this method given in [9] works quite well.


30 ■ Advanced Edge-Detection Techniques

think that they are the only possible ones. This means that the concept of optimality is a relative one, and that a better (in some circumstances) edge detector than Canny's is a possibility. In fact, sometimes it seems as if the comparison taking place is between definitions of optimality, rather than between edge-detection schemes.

Shen and Castan agree with Canny about the general form of the edge detector: a convolution with a smoothing kernel, following by a search for edge pixels. However, their analysis yields a different function to optimize. Namely, they suggest *minimizing* (in one dimension):

$$\lambda' = \frac{ax + by}{\sqrt{x^2 + y^2}} \qquad \text{(EQ 1.25)}$$

In other words, the function that *minimizes* C_N is the optimal smoothing filter for an edge detector. The optimal filter function they came up with is the *infinite symmetric exponential filter* (ISEF):

$$f(x) = \frac{p}{2}e^{-p|x|} \qquad \text{(EQ 1.26)}$$

Shen and Castan maintain that this filter gives better signal-to-noise ratios than Canny's filter, and provides better localization. This could be because the implementation of Canny's algorithm approximates his optimal filter by the derivative of a Gaussian, whereas Shen and Castan use the optimal filter directly; or it could be due to a difference in the way the different optimality criteria are reflected in reality. On the other hand, Shen and Castan do not address the multiple-response criterion, and as a result it is possible that their method will create spurious responses to noisy and blurred edges.

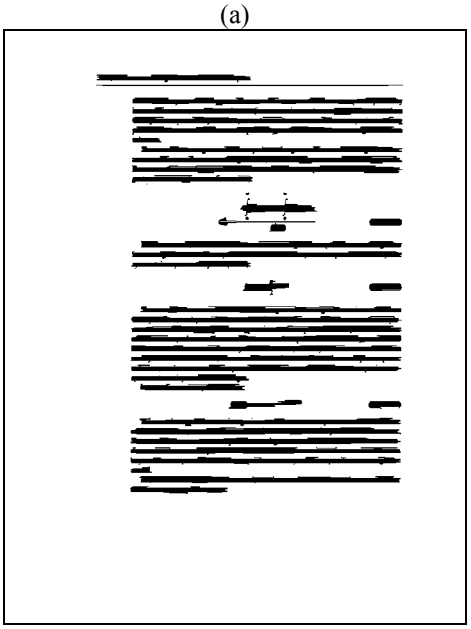
In two dimensions, the ISEF is:

$$f(x,y) = \alpha \cdot e^{-p(|x|+|y|)} \qquad \text{(EQ 1.27)}$$

This can be applied to an image in much the same way as was the derivative of Gaussian filter, as a 1D filter in the x direction, then in the y direction. However, Shen and Castan went one step further and gave a realization of their filter as one-dimensional *recursive filters*. While a detailed discussion of recursive filters is beyond the scope of this book, a quick summary of this specific case might be useful.

The filter function f above is a real, continuous function. It can be rewritten for the discrete, sampled case as:

(a)



(b)

Figure 3. A sample document (a) and its segmentation into object blocks (b).

For training data, we use images from two sources: images generated by software directly from Microsoft Word document and TEX files. These images are 300dpi, black and white, and noise-free. We generate our database from all fonts that are typically used in Word and TEX, and they have been found to cover the general printed documents we have tested.

When a new document image is processed for recognition, the same component labeling method stated above will be applied. An example of the detected connected components is illustrated in Figure 4, where the bounding boxes specify the components. These components will then go through the above-described feature computation process and then be recognized through comparing the computed feature vector against the stored symbol database.

$$X = \frac{ax + by}{\sqrt{x^2 + y^2}}$$

Figure 4. Extracted and segmented formula. Each bounding box represents a potential symbol.

2.4 Expression Recognition

As discussed earlier, there are many formula recognition methods. We use a *procedural translation* [19] method for the following reasons. Firstly, what our system handles are printed reading materials and we assume that the image quality is good enough and thus for such printed documents, all texts including formulas are aligned to specific horizon baselines which can be utilized by procedural translation. Secondly, the result generated in our system is an XML string, which grows from top to bottom by depth first search. A procedural translation matches this order naturally and no tree operation is required for grammar processing. In addition, the speed performance of such as a method is attractive.

The proposed method works as follows. First, we form baselines for all symbols: each symbol is associated with a horizontal line position, called its baseline. For all normally printed characters, they are aligned to certain baselines. New levels such as subscript and superscript will start a new baseline. The baseline of symbol is determined from the trained data and stored in the database. Figure 5 illustrates a formula with baselines depicted.

$$X = \frac{ax + by}{\sqrt{x^2 + y^2}}$$

Figure 5. A sample formula with formed baselines

Next, we organize and format the detected symbols. MathML and LaTex are two commonly used math notation standard. We use MathML in our system since it is more widely supported especially in Web-based applications. The steps of formatting the detected symbols consist of the following:

1. Start with the longest baseline. Do following steps on symbols (or virtual component, see step 2) belonging to this baseline.

2. Search all pivotal characters, including fraction "—", root "√", summation "∑", integration "∫" and product "∏"; Group all symbols within their effective area and start a recursive process from step 1 to analyze the sub-structure of the expression; These symbols will be replaced by a void component in future process so they will not be analyzed again by other process.

3. Start from the left to right on symbols of current baseline, check around area to check subscripts and supscripts, and if there exists any, start a recursive process for sub-expression; Form a MathML node for each symbol and append to the MathML string.

4. Return the MathML string.

A MathML string returned from above processes for the formula in Figure 4 is given by

```
<m:math>
    <m:mi>X</m:mi>
    <m:mo>=</m:mo>
    <m:mfrac>
        <m:mrow>
            <m:mi>a</m:mi>
            <m:mi>x</m:mi>
            <m:mo>+</m:mo>
            <m:mi>b</m:mi>
            <m:mi>y</m:mi>
        </m:mrow>
        <m:mrow>
            <m:msqrt>
                <m:msup>
                    <m:mi>x</m:mi>
                    <m:mn>2</m:mn>
                </m:msup>
                <m:mo>+</m:mo>
                <m:msup>
                    <m:mi>y</m:mi>
                    <m:mn>2</m:mn>
                </m:msup>
            </m:msqrt>
        </m:mrow>
    </m:mfrac>
</m:math>
```

2.5 Speech Synthesis

After a mathematical expression is recognized from a document image, we have its syntax notation which is understandable by the machine. The last step is to let it to be understandable by blind reader. As discussed earlier, we consider speech as the best way for the following reasons: it is easy to understand and accessible to all blind and visually impaired readers; it is inexpensive, a PC with a camera will be able to achieve this function without the requirement of other specialized equipment. Further, the methodology can be easily adopted to build a stand alone reading device, which will be the research task of the Phase II of the project.

Another important reason we propose to use speech is the availability of the freely-available software, the MathPlayer [10]. MathPlayer is a Microsoft Internet Explorer plug-in to display mathematical notation in web pages, introduced by Design Science Inc. It is based on MathML technology and can not only convert the MathML notation to graphics, but also to speech. In the example of Figure 6, after feeding the string into a MathPlayer, the expression will be read out as

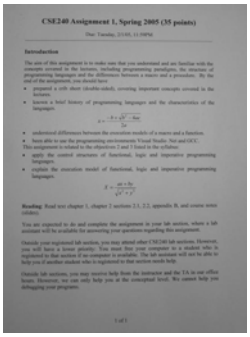
" Capital X equals,
begin fraction,
a x plus b y,
over
begin square root,
x square plus y square,
end square root,
end fraction "

It is worth point out again that, while we choose MathPlayer as the rendering scheme in the current research, it is intended for a quick demonstration of the feasibility. It will be straightforward and easy to build a stand-alone symbol-to-speech synthesizer after the string has been detected and formatted.

3. Experimental Results

We have implemented and tested all the algorithms of the proposed system. Current testing has demonstrated the effectiveness and efficiency of the proposed method. We plan to demonstrate the prototype system live during the conference. In the following, a couple of sample results are listed for illustration only.

In current experiments, the input data include both camera-captured and scanned images from general math textbook and scientific papers. During the testing, the performance metric was to examine if the math expression in the images are recognized correctly and transformed to speech successfully. Figure 6 illustrates the typical example of an input image in the testing, where the left-hand column is the input image captured by a camera, and on the right-hand side are the screen shot of the MathPlayer display of these recognized expressions (that is, the system first detects and recognizes the equations and then converts them into MathML streams, and then MathPlayer is employed to visualize the equations to verify the correctness.)

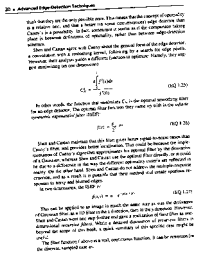


$$x = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

$$X = \frac{ax + by}{\sqrt{x^2 + y^2}}$$

Figure 6. An example illustrating the recognized expressions from the input document image.

Figure 7 shows a similar example, with a scanned document as the input.



$$C_N^2 = \frac{4\int_0^2 f^2(x)\,dx}{f^4(0)}$$

Figure 7. Results with a scanned document.

4. Conclusion

In this paper, we present an automatic end-to-end system which can read formulas from captured printed documents. This would greatly extend the reading capability of blind and visually impaired readers beyond simple texts. It is also a first step towards the general goal of helping them to understand other non-textual visual contents such as complex graphs, charts, plots, and even images etc.

This current system is under continued development. In particular, efforts are being made to make the system more robust especially with camera-based capture where the input image quality could be fairly poor unless extreme care is taken. Also, work is being performed to expand the capability of the system for processing more complex, multi-line formulas. A systematic evaluation of the system by blind readers will also be carried out and their feedbacks will be incorporated into the next step of development of the system.

References

[1] K.J. Kokjer, The information capacity of the human fingertip. *IEEE Trans. on Systems, Man, and Cybernetics, 17*(1), 1987, 100-102.

[2] J. Halliday, Braille vs. Speech: Making Sense of the Debate. *Closing the Gap, 17*(6), 1999, 6-7, 26-27, 36.

[3] M. Okamoto & A. Miyazawa, An experimental implementation of document recognition system for papers containing mathematical expressions. *Structured Document Image Analysis* (Springer 1992), 36–53.

[4] J. Ha, R. Haralick & I. Phillips, Understanding mathematical expressions from document images. *Proc. Int. Conf. on Document Analysis and Recognition*, Montreal, Canada, 1995, 956–959.

[5] H. Twaakyondo & M. Okamoto, Structure analysis and recognition of mathematical expressions. *Proc. Int. Conf. on Document Analysis and Recognition*, Montreal, Canada, 1995, 430–437.

[6] K.F. Chan & D.Y. Yeung, Mathematical expression recognition: A survey. *International Journal of Document Analysis and Recognition, 3*(1), 2000, 3-15.

[7] F.M. Wahi, K.Y. Wong & R.G. Casey, Block segmentation and text extraction in mixed text/image documents. *Computer Graphics and Image Processing,* 22, 1982, 375-390.

[8] A. Rosenfeld & A.C. Kak. *Digital Picture Processing, Vol. 2* (New York: Academic Press, 1982).

[9] R.J. Fateman, T. Tokuyasu, B.P. Berman & N. Mitchell, Optical character-recognition and parsing of typeset mathematics. *Journal of Visual Communication and Image Representation, 7*(1), 1996, 2-15.

[10] MathPlayer. Design Science Inc. http://www.dessci.com/en/products/mathplayer/

[11] A. Nemeth, *The Nemeth code of Braille mathematics* (American Printing House for the Blind, 1956).

[12] J.A. Gardner, Dotsplus-Better than Braille? *Proc. Int. Conf. on Technology and Persons with Disabilities,* Los Angeles, CA, 1993.

[13] W. Postl, Detection of linear oblique structures and skew scan in digitized documents. *Proc. Int. Conf. on Pattern Recognition*, Paris, France, 1986, 687-689.

[14] A. Belaid & J. Haton, A syntactic approach for handwritten mathematical formula recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence, 6*(1), 1984, 105–111.

[15] Y. Dimitriadis, J. Coronado & C. de la Maza, A new interactive mathematical editor, using on-line handwritten symbol recognition and error detection-correction with an attribute grammar. *Proc. Int. Conf. on Document Analysis and Recognition,* Saint Malo, France, 1991, 242–250.

[16] S. Chang, A method for the structural analysis of two-dimensional mathematical expressions. *Information Sciences, 2*(3), 1970, 253–272.

[17] A. Grbavec & D. Blostein, Mathematics recognition using graph rewriting. *Proc. Int. Conf. on Document Analysis and Recognition*, Montreal, Canada, 1995, 417–421.

[18] P. Chou, Recognition of equations using a two-dimensional stochastic context-Free grammar. *Proc. SPIE Conf. on Visual Communications and Image Processing IV*, Philadelphia PA, 1989, 852–863.

[19] H. Lee & J. Wang. Design of a mathematical expression understanding system. *Pattern Recognition Letters*, *18*, 1997, 289–298.