Mathematical Formula Detection in Heterogeneous Document Images

Wei-Ta Chu
Department of Computer Science and Information Engineering
National Chung Cheng University
Chiayi, Taiwan
wtchu@cs.ccu.edu.tw

Fan Liu
Department of Computer Science and Information Engineering
National Chung Cheng University
Chiayi, Taiwan
a4623671@gmail.com

***Abstract*—This paper presents mathematical formula detection in heterogeneous document images that may contain figures, tables, text, and math formulas. We adopt the method originally proposed for sign detection in natural images to detect non-homogeneous regions and accordingly achieve text line detection and segmentation. Novel features based on centroid fluctuation information of non-homogeneous regions are proposed to more appropriately characterize both displayed formulas and embedded formulas. By comparing the proposed method with previous works, we demonstrate the effectiveness of the proposed features.**

Keywords-mathematical formula detection; heterogeneous document images; text line segmentation

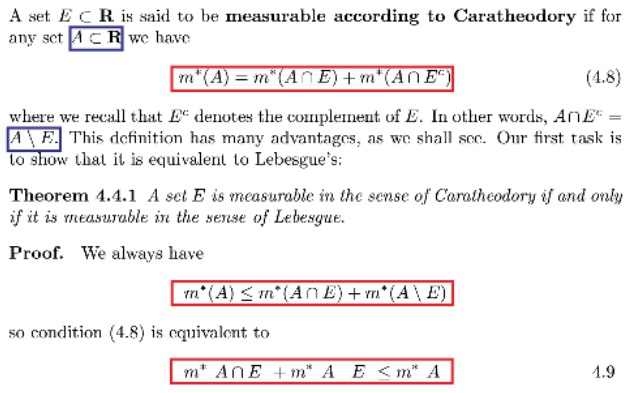
I. INTRODUCTION

As large amounts of technical documents have been published in recent years, efficiently retrieving relevant documents and identifying locations of targeted terms are urgently needed. We have already widely utilized search engines like Google to find technical documents. However, currently only text-based keywords are used for retrieving documents having related text in title, abstract, or main body. We argue that mathematical formulas have been overlooked in technical document retrieval for a long time. Scientists severely express their ideas in math, following some conventions or unwritten customs to define notations, which make *retrieving technical documents by math* a feasible idea. For example, when a beginner knows nothing about the formula of Fourier transform when he reads a paper, he may be able to find relevant online textbooks or tutorial by uploading the math formula image to a dedicated search engine. This idea can be realized only if math formulas can be identified and recognized in a scalable and automatic way. Therefore, in this work, we focus on automatic math formula detection in technical documents, which may consist of text, math, figures, and tables.

From the perspective of document analysis and recognition, math expression identification or recognition has been studies for over a decade. Results of math expression identification could largely aid optical character recognition (OCR) systems to convert scientific documents into electronic forms. Currently, some works utilize the file structure embedded in PDF files to achieve accurate detection results, e.g., [1]. Without the limitation of PDF documents, more works identify math formulas in scanned document images. However, many works focus only on detecting displayed expressions (or isolated formulas), which are displayed on lines separated from the main text and are relatively easier to be detected [2]. Garain [3] proposed a bunch of features to identify both displayed formulas and embedded formulas, which are displayed on lines conveying the main text and are also known as inline formulas. Nevertheless, heuristic decision with empirically thresholding was adopted, which diminished generality of the proposed method. Figure 1 shows some examples of displayed formulas (marked in red) and embedded formulas (marked in blue). From this figure we see high variations of math formulas, and it is sometimes quite challenging to detect them, especially the embedded formulas.

In this work we aim to identify both types of math formulas in heterogeneous document images. The contributions of this work are threefold. First, the method originally proposed for sign detection in natural images are adopted to conduct text localization and text line segmentation, so that text lines can be robustly detected in heterogeneous document images. Second, one novel feature is proposed to describe text lines and words so that better performance can be achieved. Third, classifiers based on statistical learning, i.e., support vector machine (SVM) classifier, are constructed to identify math expressions, avoiding ad hoc threshold settings.

The rest of this paper is organized as follows. Related literature is surveyed in Section II. The system framework and details of each component are described in Section III. Evaluation results are presented in Section IV, and Section V gives concluding remarks of this work.



A set $E \subset \mathbf{R}$ is said to be **measurable according to Caratheodory** if for any set $A \subset \mathbf{R}$ we have

$$m^*(A) = m^*(A \cap E) + m^*(A \cap E^c) \tag{4.8}$$

where we recall that E^c denotes the complement of E. In other words, $A \cap E^c = A \setminus E$. This definition has many advantages, as we shall see. Our first task is to show that it is equivalent to Lebesgue's:

Theorem 4.4.1 *A set E is measurable in the sense of Caratheodory if and only if it is measurable in the sense of Lebesgue.*

Proof. We always have

$$m^*(A) \leq m^*(A \cap E) + m^*(A \setminus E)$$

so condition (4.8) is equivalent to

$$m^* \; A \cap E \; + m^* \; A \quad E \; < m^* \; A \tag{4.9}$$

Figure 1. Examples of displayed formulas (in red) and embedded formulas (in blue). Note that we just mark a few of them to maintain readability.

II. RELATED WORKS

As a universal language, mathematical expression extraction/detection from documents has been studied for over a decade. Jin et al. [5] proposed one of the earliest systems to extract displayed formulas and embedded formulas. Displayed formulas were detected by a Parzen classifier constructed based on line height and indent features. Based on horizontal projection characteristics, embedded formulas were detected based on a heuristic method. Chowdhury et al. [6] focused on segmentation of math zones by decision trees, which were constructed according to several observations, such as appearance of subscript/superscript and heights of math symbols. Given a text line, Drake and Baird [7] modeled it as a graph consisted of characters as nodes and relationships between neighboring characters as edges, and then classify a text line into math expression or English text. With a similar concept, Guo et al. [9] decomposed a math expression into sub-components, and developed a Gaussian mixture model to describe the spatial relationship between them. More specifically, Tian et al. [8] studied layout difference between math formulas typed by LaTex and Microsoft Word. Recently, Garain [3] proposed features dedicated to displayed formulas and embedded formulas separately, and conducted a large-scale experiment to verify detection performance. Lin et al. [1] utilized syntax information in PDF files and accurately identified math formulas by combining a rule-based method and a learning-based method. Yamazaki et al. [2] focused on embedding their math formula identification components, which are mainly modified from the work in [3], into an open source OCR system [10]. For a general document recognition and understanding problem, PDF syntax information was also adopted to detect and describe document objects [11].

In this work, we focus on robust text line segmentation in heterogeneous document images, which influence on performance of an OCR system was underestimated before, and novel features to identify math formulas in a systematic framework, i.e., machine learning method.

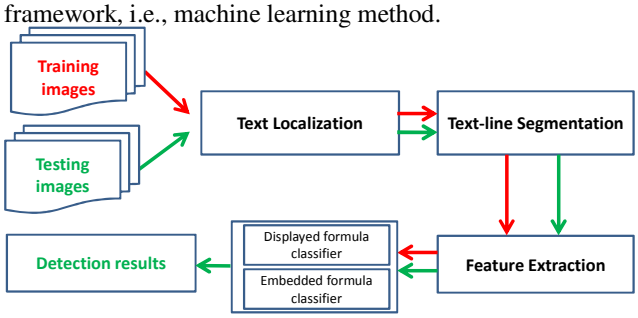


Figure 2. Flowchart of the proposed math formula detection system.

III. THE PROPOSED SYSTEM

A. System Overview

Figure 2 shows the flowchart of the proposed math formula detection system, where the red arrows denote the flow path of classifier training, and the green arrows denote the flow path of detecting formulas in test images. For both training and testing, the first step is text localization, which identifies location of text in heterogeneous documents to avoid the influence of figures and tables. We then segment the text regions into horizontal text lines, which are the basic units to detect displayed formulas. The text lines not identified as displayed formulas are further segmented into blocks, where each block represents a bounding box of a word, and is the basic unit to detect embedded formulas. We design several features to describe text lines and text blocks. Based on these features, an SVM classifier to discriminate displayed formulas from the main text and an SVM classifier to discriminate embedded formulas from a common text block are constructed, respectively.

B. Text Localization and Text Line Segmentation

To detect math formulas, we have to first identify text lines, from which features are extracted to determine whether a text line is a displayed formula or not. In this section, we focus on detecting text lines.

We adopt a low-cost text localization method [4] originally proposed for sign detection in natural images to identify text regions. There are already some text localization methods especially designed for document images. However, existing methods are often not robust when documents contain photographs, figures, clip arts, and tables.

A document image is first segmented into $k \times k$-pixel blocks. Each block is then examined for homogeneity to determine whether it is in the background region, i.e., homogeneous region. The process is illustrated in Figure 3. Let $\boldsymbol{b}$ denote a k^2-dimensional vector constituted by the intensity values in a $k \times k$-pixel block. The degree of homogeneity of this block is measured by

$$\delta^{(m)} = \left| \frac{2}{k^2} \sum_{i=1}^{k^2} \boldsymbol{b}_i \boldsymbol{w}_i^{(m)} \right|, \quad (1)$$

where $\boldsymbol{b}_i$ denotes the ith component of $\boldsymbol{b}$, $\boldsymbol{w}^{(m)}$ is the mth weighting matrix, $0 \le m < M$, with binary entries (i.e., each entry is 1 or -1) that sum to zero. In this work, three weighting matrices are designed to evaluate the gradients along the horizontal, vertical, and diagonal directions, respectively, i.e., $M = 3$, which are illustrated in Figure 3. The entry in black has the value -1, and the entry in white has the value +1. Generally speaking, $\delta^{(m)}$ is an approximate gradient of a block towards a specific direction. The three weighting matrices illustrated in Figure 3 are used to check gradients in diagonal, horizontal, and vertical directions, respectively.

After calculating all $\delta^{(m)}$'s, we classify a block as homogeneous if the L_1-norm of the vector $(\delta^{(1)}, \delta^{(2)}, \ldots, \delta^{(M)})$ is less than a threshold, and at least one of its four neighboring blocks meets this criterion.

To find text lines, we first binarize the document image by setting intensity values of pixels in homogeneous blocks as 0, and setting that in non-homogeneous blocks as 255. Intensity values of pixels are then projected in the horizontal direction to construct a horizontal profile. The row with profile value larger than a threshold is determined to be a part of a text line. In this work, the threshold is set as the 10% of the average horizontal profile value.

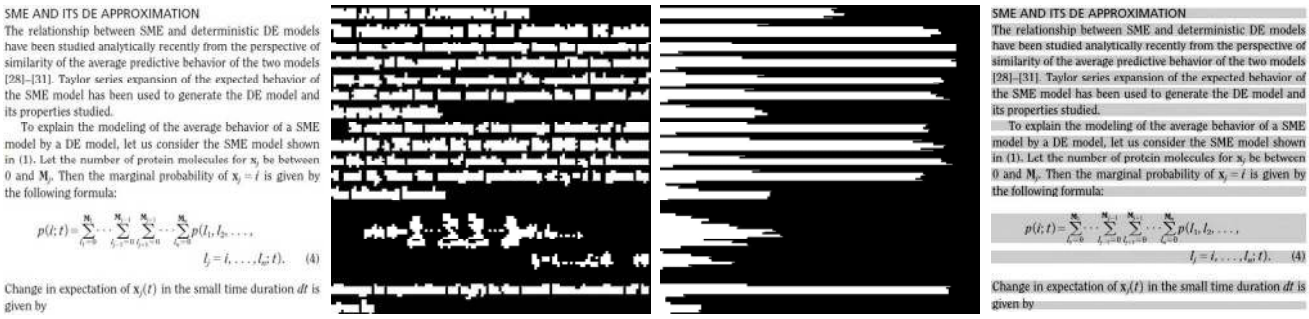


Figure 4. An example of text localization and text line segmentation.

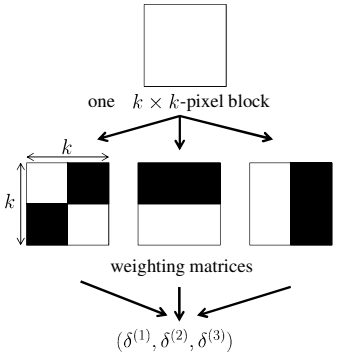


Figure 3. Checking homogeneity of a block by convolving it with three weighting matrices.

Figure 4 shows a sample result of the whole process. Figure 4(a) is the original image, Figure 4(b) is the result of binarization, and Figure 4(c) is the horizontal profile showing the distribution of homogeneity. Figure 4(d) shows the text line segmentation result, where the lines with shaded background are determined as text lines.

C. Word Segmentation

To detect embedded formulas, the text lines determined not to be displayed formulas are further segmented into words. In the binarized image, intensity values of pixels corresponding to non-homogeneous regions are set as 255. These pixels are generally aggregated together, separated by spaces between words, as shown in Figure 4(b). We detect the minimum bounding box of each non-homogeneous region based on pixels' intensities. The horizontal displacement between two neighboring boxes is checked. If the horizontal displacement is less than a threshold, two bounding boxes are merged because they probably belong to the same word or embedded formula. Next, if a bounding box which left neighboring box contains only one alphabet[1] and right neighboring box contains only one alphabet, these three spatially adjacent boxes are merged. This process is designed because a pure text word rarely contains only one alphabet. Isolated alphabets in text lines are often parts of an embedded formula, such as the symbols "=" and "+".

[1] We check whether the non-homogeneous regions bounded by this box are fully connected, i.e., only one connected component. If so, the bounding box is determined to contain only one alphabet.

D. Feature Extraction

To determine whether a text line includes a displayed formula, we extract the following features to describe each text line.

- Density f_d

The density value of a text line is defined as

$$f_d = \frac{\#\text{pixels}_t}{\#\text{pixels}_r}, \tag{2}$$

where $\#\text{pixels}_r$ denotes the number of pixels in the bounding box of a text line, and $\#\text{pixels}_t$ denotes the number of pixels in the bounding boxes of non-homogeneous regions in the text line. This feature basically describes the density of non-homogeneous region's pixels in a text line. When a text line corresponds to a displayed formula, its f_d is generally smaller [1].

- Height of text line f_h

This feature describes the ratio of the height of a text line to the height of the whole document image, and is defined as

$$f_h = \frac{t_h}{H}, \tag{3}$$

where t_h is the height of a text line, which is the height of the minimum bounding box covering of this text line. The value H is the height of the whole document image. Generally, pure text lines have similar heights, while a text line containing a displayed formula may have a different height [1].

- Left indent f_l and right indent f_r

A displayed formula often does not occupy the whole text line, and thus the values of left indent and right indent are often used to describe this characteristic [5]. The left indent feature is defined as

$$f_l = \frac{ls_w}{W}, \tag{4}$$

where ls_w denotes the distance between the left border of the document image[2] and the left boundary of the leftmost minimum bounding box in a text line. The value W is the width of the whole document image. Similarly, the right indent feature is defined as

$$f_r = \frac{rs_w}{W}, \tag{5}$$

where rs_w denotes the distance between the right border of the document image and the right boundary of the rightmost minimum bounding box in a text line.

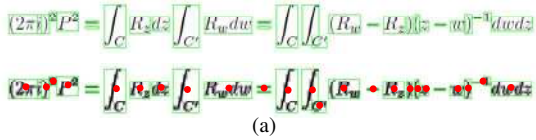
- Centroid fluctuation f_u

To more accurately describe display formulas, we propose the fluctuation feature to describe how the centroids of non-homogeneous regions in a text line move. It is mathematically described as

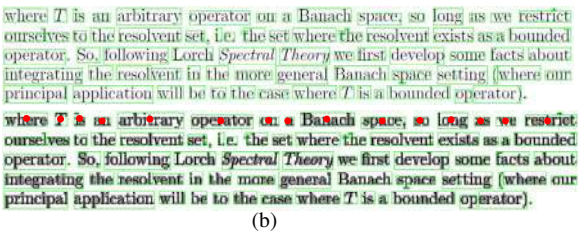
$$f_u = \frac{1}{n}\sum_{i=1}^{n-1}\left|\cos^{-1}\frac{\boldsymbol{v}_i \cdot \boldsymbol{v}_h}{\|\boldsymbol{v}_i\|\|\boldsymbol{v}_h\|}\right|, \tag{6}$$

where $\boldsymbol{v}_i$ is the vector from the centroid of the ith word to the centroid of the $(i+1)$ th word. Assuming that a document image forms a coordinate system where the origin locates the left-top corner of this document, and each pixel in this document can be located at a (x, y) coordinate. A word's centroid is calculated as the mean x coordinate and the mean y coordinate of the minimum bounding box of this word. The vector $\boldsymbol{v}_h$ is the horizontal vector passing through the mean x coordinate of all words in the text line.

Figure 5 shows examples of centroid fluctuation characteristics of a displayed formula and pure text lines. From Figure 5(a), we see centroids of non-homogeneous regions (marked as red dots) often fluctuates because of the integral symbol or superscripts/subscripts. On the contrary, centroids of non-homogeneous regions in Figure 5(b) are generally close to the middle line cross the whole text line.



(a)



(b)

Figure 5. Examples showing the centroid fluctuation of (a) a displayed formula; and (b) a pure text line.

[2] We assume the document is in one-column style. If the document contains multiple columns, we would first segment the document into individual columns. The left indent feature is then defined as the left border of the column and the left boundary of the leftmost minimum bounding box in a text line. The right indent feature is defined similarly.

The features mentioned above are concatenated as a vector $(f_d, f_h, f_l, f_r, f_u)$, which is then input to an SVM classifier that determines whether the text line is a displayed formula or not. If not, this text line would be a pure text line or would contain some embedded formulas. Some words segmented from a text line (described in Section III.C) may correspond to embedded formulas. To determine whether a word is an embedded formula, we extract the following features and then fed them into another SVM classifier. These features are basically counterparts of that extracted from text lines.

- Density g_d

The density value of a word is defined as

$$g_d = \frac{\#\text{pixels}_{t'}}{\#\text{pixels}_{r'}}, \tag{7}$$

where $\#\text{pixels}_{r'}$ denotes the number of pixels in the minimum bounding box of a word, and $\#\text{pixels}_{t'}$ denotes the number of pixels in the minimum bounding boxes of non-homogeneous regions in the word.

- Height of word g_h

This feature describes the ratio of the height of a word to the height of the whole document image, and is defined as

$$g_h = \frac{b_h}{H}, \tag{8}$$

where b_h is the maximum height of non-homogeneous regions belonging to this word. The value H is the height of the whole document image.

- Centroid fluctuation g_u

The fluctuation feature of a word is defined as

$$g_u = \frac{1}{m}\sum_{i-1}^{m-1}\left|\cos^{-1}\frac{\boldsymbol{u}_i \cdot \boldsymbol{u}_h}{\|\boldsymbol{u}_i\|\|\boldsymbol{u}_h\|}\right|, \tag{9}$$

where $\boldsymbol{u}_i$ is the vector from the centroid of the ith non-homogeneous region to the centroid of the $(i+1)$th non-homogeneous region in this word. The vector $\boldsymbol{u}_h$ is the horizontal vector passing through the mean x coordinate of all non-homogeneous regions in this word.

IV. Evaluation Results

A. Dataset

We utilize the dataset that was also used in [1] to verify the proposed method. The document images are converted from the PDF pages of mathematics textbooks [3]. We randomly selected 100 pages as the training dataset, and 96 pages for testing. For the training dataset, we manually define the positions of displayed formulas and embedded formulas. After feature extraction, an SVM classifier was built to determine whether a text line contains a displayed formula or not. Another SVM classifier was built to

[3] http://www.math.harvard.edu/~shlomo/

determine whether a word corresponds to an embedded formula or not.

B. Results of Text Line Segmentation

The reason to develop a text line segmentation method modified from a sign detection method for natural images is higher robustness for heterogeneous document images. We compare the proposed method with the default segmentation method in the OCRopus system [10], based on document images consisting of text, figures, and tables.

Figure 6 shows sample text line segmentation results by the OCRopus system (top) and our system (bottom). From Figure 6 we see our method works better because (i) the whole embedded figure is detected as a text line, which can be easily eliminated by the following process; and (ii) text lines are more accurate segmented, especially in two-column documents.

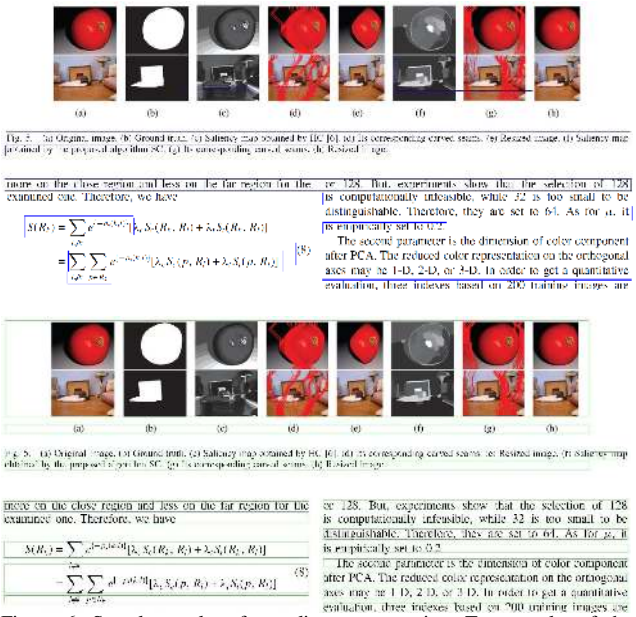


Figure 6. Sample results of text line segmentation. Top: results of the OCRopus system; bottom: our results.

C. Results of Displayed Formula Detection

In detecting displayed formulas, we compare our work with [2]. To make fair performance comparison, the evaluation dataset and all components (text line segmentation, word segmentation, and SVM-based classification) are the same for two works, except for the features used to describe text lines.

Table 1 shows performance comparison. The precision rate is the ratio of the number of correctly detected formulas to the number of all detected formulas. The recall rate is the ratio of the number of correctly detected formulas to the number of all relevant formulas. The F-measure value jointly considers precision and recall, and is defined as

$$\text{F-measure} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}. \qquad (10)$$

From this table we clearly see the performance superiority of the proposed method, more specifically the proposed feature. The work in [2] also adopts the density, height, and indent features similar to ours, and thus the performance improvement is mainly brought by the centroid fluctuation feature f_u.

Figure 7 shows one sample page of displayed formula detection results, where red bounding boxes denote the ground truths and blue bounding boxes denote the detection results. Actually, from these examples all detected results coincide with ground truths, and we see most are purple bounding boxes that are mixture of red and blue.

Table 1. Performance comparison of displayed formula detection.

	Precision	Recall	F-measure
[2]	0.84	0.83	0.84
Our	0.95	0.91	0.93

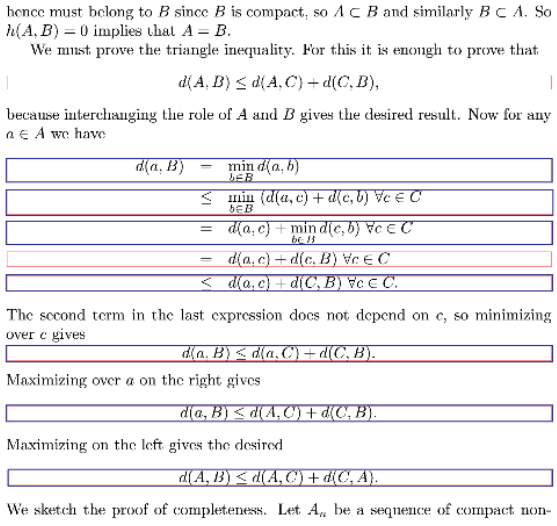


Figure 7. One sample page of displayed formula detection results.

D. Results of Embedded Formula Detection

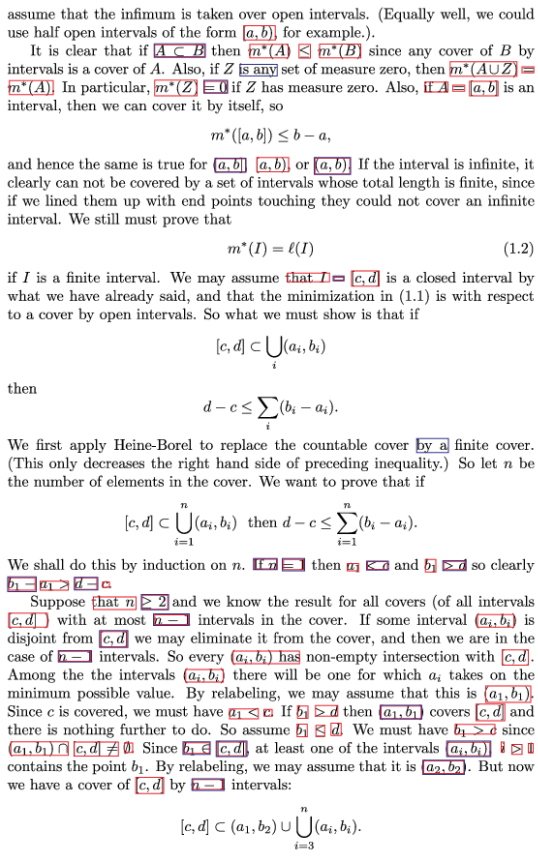
Generally, detecting embedded formulas is much harder than detecting displayed formulas. Table 2 shows performance comparison between our method and [3], which was one of the few works that tackle embedded formula detection. It can be seen that although our method obtains slightly worse performance in precision, we achieve significantly better performance in recall. The work in [3] adopts features similar to the concept of density and height. This further demonstrates that the proposed centroid fluctuation feature g_u is able to more appropriately describe embedded formulas and thus improves the recall rate.

Figure 8 shows samples of embedded formula detection results. From this figure we clearly see detected embedded formulas are highly accurate. However, many embedded formulas are not detected, which is also reflected as the lower recall rate in Table 2. The main reasons for low recall are twofold. First, some non-homogeneous regions are not correctly merged as an embedded formula. For example, the

formula "$(a_1, b_1) \cap [c, d] \neq \emptyset$" is divided into three pieces, "$(a_1, b_1)\cap$", "$[c, d] \neq$", and "$\emptyset$". Second, some embedded formulas are quite confused with common text, like "(a_1, b_1)". For the first shortage, we need to develop a more robust word segmentation method. For the second shortage, novel features that jointly consider information from neighboring words would be needed. The idea of the language model used in natural language processing may be adopted to model relationships between words.

Table 2. Performance comparison of embedded formula detection.

	Precision	Recall	F-measure
[3]	0.86	0.32	0.46
Our	0.80	0.48	0.60



assume that the infimum is taken over open intervals. (Equally well, we could use half open intervals of the form $[a, b)$, for example.).

It is clear that if $A \subset B$ then $m^*(A) \leq m^*(B)$ since any cover of B by intervals is a cover of A. Also, if Z is any set of measure zero, then $m^*(A \cup Z) = m^*(A)$. In particular, $m^*(Z) = 0$ if Z has measure zero. Also, if $A = [a, b]$ is an interval, then we can cover it by itself, so

$$m^*([a, b]) \leq b - a,$$

and hence the same is true for $(a, b]$, $[a, b)$, or (a, b). If the interval is infinite, it clearly can not be covered by a set of intervals whose total length is finite, since if we lined them up with end points touching they could not cover an infinite interval. We still must prove that

$$m^*(I) = \ell(I) \tag{1.2}$$

if I is a finite interval. We may assume that $I = [c, d]$ is a closed interval by what we have already said, and that the minimization in (1.1) is with respect to a cover by open intervals. So what we must show is that if

$$[c, d] \subset \bigcup_i (a_i, b_i)$$

then

$$d - c \leq \sum_i (b_i - a_i).$$

We first apply Heine-Borel to replace the countable cover by a finite cover. (This only decreases the right hand side of preceding inequality.) So let n be the number of elements in the cover. We want to prove that if

$$[c, d] \subset \bigcup_{i=1}^{n} (a_i, b_i) \quad \text{then } d - c \leq \sum_{i=1}^{n} (b_i - a_i).$$

We shall do this by induction on n. If $n = 1$ then $a_1 < c$ and $b_1 > d$ so clearly $b_1 - a_1 > d - c$.

Suppose that $n \geq 2$ and we know the result for all covers (of all intervals $[c, d]$) with at most $n - 1$ intervals in the cover. If some interval (a_i, b_i) is disjoint from $[c, d]$ we may eliminate it from the cover, and then we are in the case of $n - 1$ intervals. So every (a_i, b_i) has non-empty intersection with $[c, d]$. Among the the intervals (a_i, b_i) there will be one for which a_i takes on the minimum possible value. By relabeling, we may assume that this is (a_1, b_1). Since c is covered, we must have $a_1 < c$. If $b_1 \geq d$ then (a_1, b_1) covers $[c, d]$ and there is nothing further to do. So assume $b_1 < d$. We must have $b_1 > c$ since $(a_1, b_1) \cap [c, d] \neq \emptyset$. Since $b_1 \in [c, d]$, at least one of the intervals (a_i, b_i) contains the point b_1. By relabeling, we may assume that it is (a_2, b_2). But now we have a cover of $[c, d]$ by $n - 1$ intervals:

$$[c, d] \subset (a_1, b_2) \cup \bigcup_{i=3}^{n} (a_i, b_i).$$

Figure 8. Samples of displayed formula detection results.

V. CONCLUSION

We have presented a mathematical formula detection system able to detect displayed formulas and embedded formulas in heterogeneous document images. We adopt the segmentation method originally proposed for sign detection in natural images to detect text lines. A text line is further segmented into words if necessary. In addition to density, height, and indent features that were commonly used to describe formulas, we propose the centroid fluctuation feature to more appropriately describe characteristics of math formulas. The experimental results demonstrate performance superiority of the proposed features.

In the future, a much more large-scale evaluation will be conducted to verify generality of the proposed method. We would include various types of technical documents that may contain math formulas. In addition, a math retrieval system mentioned in the introduction section would be a high-level goal in the future.

ACKNOWLEDGMENT

The work was partially supported by the National Science Council of Taiwan, Republic of China under research contract NSC 101-2221-E-194-055-MY2.

REFERENCES

[1] X. Lin, L. Gao, Z. Tang, X. Lin, and X. Hu, "Mathematical formula identification in PDF documents," Proc. of International Conference on Document Analysis and Recognition, pp. 1419-1423, 2011.

[2] S. Yamazaki, F. Furukori, Q. Zhao, K. Shirai, and M. Okamoto, "Embedding a mathematical OCR module into OCRopus," Proc. of International Conference on Document Analysis and Recognition, pp. 880-884, 2011.

[3] U. Garain, "Identification of mathematical expressions in document images," Proc. of International Conference on Document Analysis and Recognition, pp.1340-1344, 2009.

[4] K. Bouman, G. Abdollahian, M. Boutin, and E. Delp, "A low complexity sign detection and text localization method for mobile applications," IEEE Transactions on Multimedia, vol. 13, no. 5, pp. 922–934, 2011.

[5] J. Jin, X. Han and Q. Wang, "Mathematical formulas extraction," Proc. of International Conference on Document Analysis and Recognition, pp. 1138-1141, 2003.

[6] S.P. Chowdhury, S. Mandal, A.K. Das, and B. Chanda, "Automated segmentation of math-zones from document images," Proc. of International Conference on Document Analysis and Recognition, pp. 755-759, 2003.

[7] D.M. Drake and H.S. Baird, "Distinguishing mathematics notation from english text using computational geometry," Proc. of International Conference on Document Analysis and Recognition, pp. 1270-1274, 2005.

[8] X.-D. Tian, Y. Zhao, H. Wang, and Q.-J. Wang, "Layout identification of printed mathematical formula for recognition," Proc. of International Conference on Information Engineering and Computer Science, pp. 1270-1274, 2005.

[9] Y.-S. Guo, L. Huang, and C.-P. Liu, "A new approach for understanding of structure of printed mathematical expression," Proc. of International Conference on Machine Learning and Cybernetics, pp. 2633-2638, 2007.

[10] OCRopus, http://code.google.com/p/ocropus/.

[11] V. Loia and S. Senatore, "An alternative, layout-driven approach to the clustering of documents," International Journal of Intelligent Systems, vol. 23, no. 7, pp. 795-821, 2008.