Refinement of digitized documents through recognition of mathematical formulae

Toshihiro KANAHORI
Research and Support Center on Higher Education
for the Hearing and Visually Impaired,
Tsukuba University of Technology,
Kasuga 4-12-7, Tsukuba,
Ibaraki, Japan 305–8521
Email: kanahori@k.tsukuba-tech.ac.jp

Masakazu SUZUKI
Faculty of Mathematics,
Kyushu University,
Hakozaki 6-10-1, Higashi-ku,
Fukuoka, Japan 812–8581
Email: suzuki@math.kyushu-u.ac.jp

Abstract

We are developing a recognition system, named 'Infty', for scientific documents including those with mathematical formulae. In this paper, we propose a new system that can refine a text embedded PDF document recognizing the PDF as images and integrating its text information into the recognition results of Infty. This system can be combined with other OCR systems that output recognition results as text embedded in a PDF document. Using this system, mathematical information can be added to books, journals and papers in existing digital libraries. We evaluate effects of this system, comparing its recognition rates with those of ABBYY FineReader. The evaluation shows that this system can add mathematical information to PDF documents generated by FineReader without loss of quality of the ordinary text parts.

1. Introduction

World wide, there are many project digitizing printed documents. However, except for some specialized digital libraries [1], some components of a document, for example, chemical and mathematical formulae, can not be completely or accurately digitized because of technical difficulties. Generally, it is expensive to correct recognition errors, and so incorrect information often remains in a digital library. It is also expensive to input information of components that can not be automatically recognized. Mathematical formulae are either skipped and have no hidden text (fig. 1), or they are recognized as ordinary text and they are given cryptic text (fig. 2). In the both cases, the mathematical information can not be reused such as in searches for a mathematical formula in a document. Mathematical formulae also cause recognition errors in the ordinary text around them. In the figure 3, the formula "$\zeta \to \log((\zeta - i\eta_0)^{-1})$" caused an estimation error of the parameters of the line, and so the words 'and' and 'applying' after the formula were not recognized correctly. It is important and necessary to digitize printed documents; therefore, we can not wait for long to resolve these technical and cost problems.

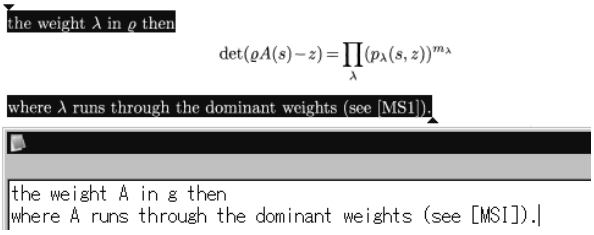

Figure 1. The mathematical formula on the middle line is skipped and there is no hidden text for the formula. The black painted parts have hidden text in the upper area. In the lower area, the hidden text is shown.

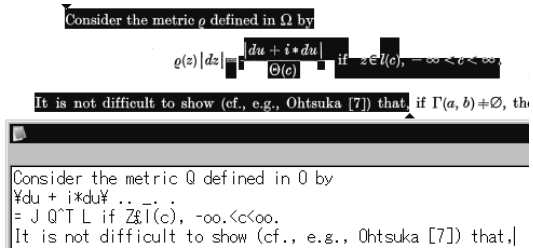

Figure 2. The hidden text for mathematical formulae do not have correct mathematical information.

In this paper, we propose a new system that can generate higher quality document recognizing text embedded PDF documents and adding the recognition results of mathematical formulae to them. From a text embedded PDF document, an image file is generated and the text information extracted as a text file. Then, by utilizing the text information to correct the recognition results, the image file is recognized by Infty, a scientific document recognizer we

are developing that also recognizes mathematical formulae. The recognition results of Infty can be output into LaTeX, MathML, HTML and so on, and can be edited with *InftyEditor*, which has rich inputting interfaces for mathematical formulae [2]. We can combine Infty with other OCR systems that output recognition results in text embedded PDFs.

Generally, a commercial OCR uses linguistic information for recognition. Thus, it can provide high recognition rates for ordinary text documents written in a single language, but for documents that include unexpected terms like mathematical formulae, or words from other languages, it can not recognize them with that performance, and so the errors causes result in further following errors. On the other hand, Infty does not use linguistic information even for parts of a document with ordinary text (fig. 4). Hence, it can not recognize ordinary text as well as other commercial OCR, but it can provide higher recognition rates for multilingual documents, including those with mathematic formulae (fig. 3).

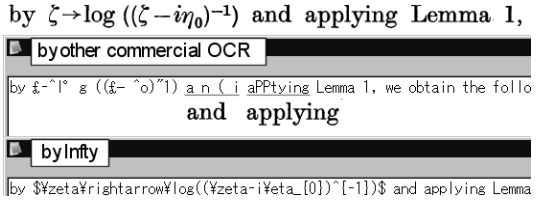

Figure 3. Recognition errors by other commercial OCR around mathematical formulae (upper text), but Infty recognizes correctly (lower). Infty is embedding mathematical information in LaTeX format.

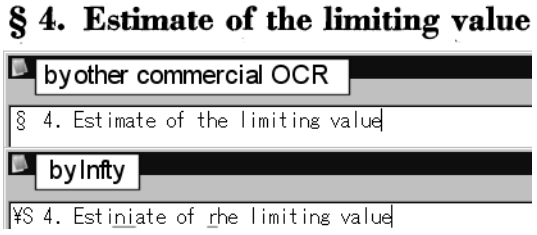

Figure 4. Recognition errors by Infty in ordinary text (lower text). Infty does not use linguistic information, so Infty can not find easy 'spell misses.'

Using this system in combination, we can get high quality scientific documents generated from the good recognition results for ordinary text using a commercial OCR and get mathematical information by using Infty. We evaluate the effects of this system for mathematical documents in PDF form as generated by ABBYY FineReader. The evaluation results show that this system adds mathematical information to PDF documents without loss of quality for ordinary text parts and corrects some recognition errors caused by the mathematical formulae.

2. Combining other OCR system with Infty via PDF

Recent general OCR systems can output recognition results as text embedded PDF documents that also include scanned page images of original documents. We use *GhostScript* to combine Infty with other OCR systems via PDFs.

This system recognizes a text embedded PDF document as follows (fig. 5); 1) by extracting text information from a PDF document as a text file, 2) by converting the PDF document to a Multi-TIFF image file, 3) by recognizing the image file by Infty, 4) by correcting the recognition results by matching them with the extracted text information, 5) by generating a text embedded PDF document from the corrected recognition results, in which mathematical formulae are output in LaTeX format. These steps are detailed in the following sections.

An advantage of this method is that it needs only a text embedded PDF document to combine Infty with other OCR systems. Namely, by recognizing a PDF document, we do not need knowledge of the OCR system that generated the PDF. However, knowledge of the OCR system helps us to tune the method of matching Infty's recognition results with the OCR system's results.

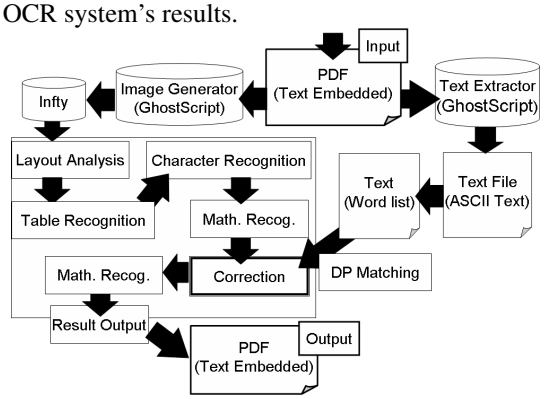

Figure 5. Recognition process by Infty via PDF

3. Preprocess for PDF

The preprocessing for a PDF document is separated into tasks of extracting text information and then converting it to an image file. Both tasks are done by utilizing *GhostScript*, a popular PostScript interpreter [3], via a batch file, as follows; 1) Extracting text information from a PDF document: By calling Ghostscript, Infty gets the text information from a PDF document as a ASCII-text file. Infty generates a *word* list segmenting character strings of the extracted text file at space characters into pieces, each of which is called '*word*'. Utilizing the word list, Infty corrects its recognition results. 2) Generating an image file from a PDF document: GhostScript can generate a Multi-TIFF image file which has

all the page image files from a PDF document. The generated image file is recognized by Infty.

4. Recognizing by Infty

For an image file generated from a PDF document, Infty analyzes page layouts and recognizes characters, tables and mathematical formulae of the document. Then, comparing Infty's recognition results with the text extracted from the PDF documents, Infty corrects its recognition results. The outline of the correction is done as follows: In parts with ordinary text, the extracted text is used as the final recognition results, when they are not *incompatible* with the page image *by Geometric Position Checking*, because other commercial OCR systems usually perform with higher recognition rates than Infty for ordinary text. In parts with mathematical formulae, Infty's recognition results are used as the final results.

4.1. Segmentation of ordinary text / math parts of a document

After character recognition, Infty checks whether each word is ordinary text or not by *Geometric Position Checking* (defined in the section 4.2). Each word determined as not ordinary text is recognized as mathematical formulae for analysis. In the analysis, there are possibilities that character recognition results are changed ([4]). Therefore, after the analysis Infty again checks whether each word analyzed as a mathematical formula is ordinary text or not by Geometric Position Matching. Thus, character recognition results are segmented into ordinary text or mathematical formulae. After segmentation, Infty corrects all the recognized results by *Word Matching Method* (defined in the section 4.3), except for those clearly defined mathematical parts having upper and lower structures such as a fraction or a summation.

4.2. Geometric Position Checking

After character recognition, each character has a bounding rectangle. For each line, Infty estimates the character position features – a top line position, a center line position, a base line position and a bottom line position, and for a lower letter 't', its top line position (t-top line) is also estimated (fig. 6). For each character, these features are estimated from its character recognition result. The features of a line are estimated by majority voting of each character's features. When the numbers of characters of a line are enough, the line's features are deemed reliable because of the high recognition rates of Infty.

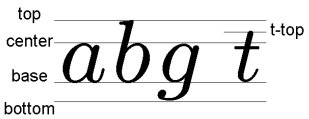

Figure 6. Character position features

Using these parameters, Infty checks whether a word W is ordinary text or not. Infty estimates these position features for each character of W according to its character recognition results. If gaps of all characterfs position features of W are less than some threshold, namely all the characters are on the same baseline with proper sizes, Infty determines that W is ordinary text. This check method is named *Geometric Position Checking Method*.

4.3. Word Matching

Each line of recognition results of Infty is cut at spaces or borders between ordinary text and mathematical formulae, and is segmented into words. Matching the words with the word list of the extracted text from the PDF, Infty corrects the recognition results. Infty distinguishes italic types from roman types, but in the following matching process they are identified and comparing as a simple character string. Each special symbol (ex. a mathematical symbol, etc.) is replaced '\'(backslash).

1) Fixing recognition results by completely matching: If a word w_i in the recognition results of Infty is completely matched with a word W_p in the word list of the extracted text from the PDF, the recognition results of w_i are fixed as the final results and the previous word w_{i-1} of w_i in Infty results is compared with the previous word W_{p-1} of W_p in the word list. If w_{i-1} and W_{p-1} are also completely matched, the length of w_{i-1} is added to N_P, and the previous word w_{i-2} of w_{i-1} is compared with the previous word W_{p-2} of W_{p-1}. Then, if w_{i-2} and W_{p-2} are completely matched, the length of w_{i-2} is added to N_P. Following these steps, while both of the n-th previous word w_{i-n} of w_i and the n-th previous word W_{p-n} of W_p are completely matched, the length of w_{i-n} is added to N_P. In the same manner, for next word w_{i+n} from w_i, its length is added to N_N while w_{i+n} is completely matched with W_{p+n}.

Thus, the sum of the lengths of the previous or next words of w_i, which are completely matched with the corresponding words of the word list of the PDF, is defined as $N_M(w_i, W_p) := N_P + N_N$. If there are several words of the word list completely matching with w_i, we define as follows; $N_M(w_i) = N_M(w_i, W(i))$
$= \max\{N_M(w_i, W_p) | W_p \text{ is completely matching with } w_i\}$, where $W(i)$ is one of the word list of the PDF text and attains the $N_M(w_i)$. In this step, we do not use a position of a word in the word list, but only use local adjacency of words, because the order of lines is sometimes not correct due to errors of layout analysis.

2) Matching neighbor words of fixed words: Each previous or next unfixed word w_i of a fixed word is matched with a word W_p of the word list by Dynamic Programming (DP-matching). The word W_p is obtained as follows;
1) if the previous word w_{i-1} of w_i is fixed, then W_p is the next word of $W(i-1)$,

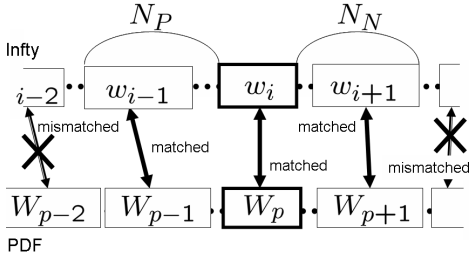

Figure 7. The meaning of $N_M(w_i, W_p)$. In this figure, each box represents a word.

2) if the next word w_{i+1} is fixed, then W_p is the previous word of $W(i-1)$,
3) if both w_{i-1} and w_{i+1} are fixed,

$$W_p = \begin{cases} \text{the next word of } W(i-1), \\ \quad \text{for } N_M(w_{i-1}) \geq N_M(w_{i+1}) \\ \text{the previous word of } W(i+1), \\ \quad \text{for } N_M(w_{i-1}) < N_M(w_{i+1}) \end{cases}$$

where we let $N_M(w_{i-1}) := N_M(w_{i-1}, W(i-1))$.
An unfixed word w_i is DP-matched with W_p obtained as above. If the score of the DP-matching is less than a threshold in respect to the length of the word, the pair (w_i, W_p) is checked by *Geometric Position Checking* method to decide which recognition results, w_i or W_p, should be used as the final results. In the DP-matching that we are using in this process, some sets of similar characters, for example, '1'(one) and 'l'(alphabet lower L), '0'(zero), 'O'(alphabet capital O) and 'O'(alphabet lower O), etc., are identified. A word w_i has result candidates, so the top three candidates are used and then costed according to the candidate order in the DP-matching. For pairs of characters that tend to be mistaken by character recognition, for example, n and π, etc. their DP-costs are tuned.

3) Matching unfixed words: For each word w in Infty whose length $L(w)$ is greater than 3, Infty searches for a word W in a word list such that $|L(w) - L(W)| \leq 2$ and $S_{DP}(w, W) = \min\{S_{DP}(w, W') | |L(w) - L(W')| \leq 2\}$, where $S_{DP}(w, W)$ is a DP-matching score of w and W. If the $S_{DP}(w, W)$ is less than some threshold and w are matched with W by the Geometric Position Checking method, recognition results of w are replaced with W.

For unfixed words whose lengths are less than 4, their recognition results are not corrected.

4) Final checking: For the matched words pair (w, W), Infty determines which word is used as the final recognition results. A bounding rectangle is estimated and set to each character C of W according to the matching result. If the word W is determined as ordinary text by Geometric Position Checking, the recognition results of the word w are changed to W's results.

Bounding rectangles for each character of a word W are obtained as follows:

A character c of w is matched with a character C of W: The bounding rectangle of c_i is just set to C_p's.

Several characters $\{\mathbf{c_i}, \cdots, \mathbf{c_{i+n}}\}$ of w are matched with a character $\mathbf{C_p}$ of W:
1) If c_i is matched with not only C_p and C_{p+1} is just matched with a character $c(p+1)$ of w, the sum of bounding rectangles of $c_i, c_{i+1}, \cdots, c_{i+n}$ is set to C_p (fig. 8).
2) Otherwise, for $c_j \in \{c_i, c_{i+1}, \cdots, c_{i+n}, c_{i+n+1} = c(p+1)\}$, if a center point of c_j's bounding rectangle is on the left side of the right edge of c_i's bounding rectangle, then c_j's bounding rectangle is added to C_p's bounding rectangle. If a center point of c_j is on the right side of the left edge of $c(p+1)$, then c_j's bounding rectangle is added to C_{p+1}'s bounding rectangle (fig. 9).

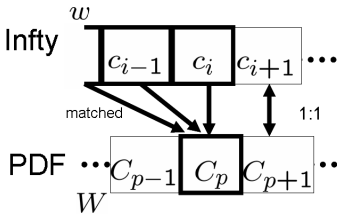

Figure 8. Several characters in Infty are matched with one character in PDF (case 1).

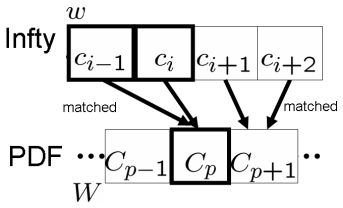

Figure 9. Several characters are matched with one character (case 2).

Several characters $\{\mathbf{c_{i-n}}, \cdots, \mathbf{c_{i-1}}, \mathbf{c_i}\}$ of w are matched with a character $\mathbf{C_p}$ of W: In the manner similar to the above case, Infty sets a bounding rectangle to C_{p-1} and C_p.

There is no character in w corresponding to $\mathbf{C_p}$:
1) If the previous character C_{p-1} corresponds to $\{c_{i-n}, \cdots, c_i\}$ and C_{p+1} corresponds to c_{i+1}, the bounding rectangle of C_{p-1}, which is already set, is vertically divided into two equal rectangles, and they are set to C_p and C_{p-1}, respectively(fig. 10). Then according to the characters of C_p and C_{p-1}, their vertical positions and heights are adjusted using the character position feature (fig. 6).
2) If C_{p-1} corresponds to c_{i-1} and C_{p+1} corresponds to $\{c_i, \cdots, c_{i+n}\}$, C_{p+1}'s bounding rectangle is divided to C_p and C_{p+1}, and their vertical positions and heights are adjusted as above. Thus, a bounding rectangle is set for all characters of each word of a word list.

4.4. Recognition of parts with mathematical formulae

After the above matching steps, Infty undertakes mathematical recognition and again checks whether the results

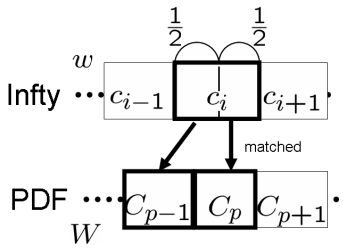

Figure 10. A character in Infty is matched with several characters in a PDF

are mathematical parts or not. Then, the recognition results are used as the final results, except for words corrected as ordinary text parts in the above matching steps.

5. Output of recognition results

Final recognition results of Infty corrected by matching with a PDF text are output into a text embedded PDF again. In order to produce a text embedded PDF, Infty first converts recognition results into LaTeX format. Second, the LaTeX source file is compiled by a 'platex' command to make a DVI file. Finally, a PDF file is generated from the DVI file. The recognition results are embedded under generated page images from an original PDF file using LaTeX's *picture* environment. For each word in parts of ordinary text, its recognition results are put on the image as *white* text according to its bounding rectangle by a 'put' command. Each mathematical formula part is embedded as in LaTeX format (fig. 3). Infty can output recognition results into MathML, so parts with mathematical formulae can be embedded as in MathML.

6. Experimental evaluation

Using this recognition system for PDF documents, including those with mathematical formulae, as being images but ones without their mathematical information, we can recognize documents and obtain higher quality PDF documents by adding the mathematical information into the original PDF. In this section, we show experimental results to evaluate this system's effectiveness for digitized documents, including those with mathematical formulae.

ABBYY FineReader, one of the most popular OCR systems, is often used for creating digitized libraries and provides very high recognition rates for ordinary text documents. We prepared text embedded PDF documents generated from papers of mathematical journals and textbooks with FineReader7.0 without any correction, and used them as the documents contained in a digital library for the following experiment.

We used our ground truth data base of scientific documents including mathematical formulae created by Infty. The ground truth has images of the pages of the documents scanned in 600 dpi, and recognition results and bounding rectangles of characters, which are distinguished parts with ordinary text / parts with mathematical formula, and roman type / italics / bolds, and mathematical and logical structures (title / header / footer / bibliography / etc.) of documents, and so on ([5], [6]). In this experiment, we used only the character recognition results. Among the ground truth, we used 31 papers from 15 journals or textbooks of mathematics written in English. The papers have 787,421 characters – ordinary text part: 621,998 characters / mathematical part: 165,423 characters.

We used texts embedded in PDF files generated by FineReader to obtain recognition results. To compare our system's recognition rates with FineReader's recognition rates, it was necessary to convert FineReaderfs recognition results into the same format as Inftyfs. Infty's results format has bounding rectangles for each character, but the text files extracted from PDF documents have only character strings. We added bounding rectangles to the extracted text files in a manner similar to that set out in section 4, where FineReader's texts were used as the final recognition results for all parts with ordinary text. Moreover, there is no information about font types (roman type / italics / bolds), font faces (Script, Fraktur, etc.), Latin characters, distinctions of point components like ' " ' (begin double quotation) and ' " ' (end double quotation), and '−'(minus) and '-'(hyphen), and so on, because we used FineReader's recognition results via ASCII-text files. So, evaluating FineReader's recognition rates, we did not distinguish the font types and the font faces, and '−'(minus) and '-'(hyphen), and reduced Latin characters and point components; namely, we used ASCII-text information.

Thus, for parts with ordinary text (TEXT), we obtained FineReaderfs recognition rates using text files extracted from PDF documents. Next, we obtained Infty's recognition rates for parts with ordinary text (TEXT), mathematical formula parts (MATH) and parts with both (TOTAL) under the same conditions as for FineReader. Finally, we obtained the recognition rates of the recognition system for the PDF documents generated by FineReader in a manner similar to that for the evaluation of Infty. The evaluation results are shown in Table 1. Table 2 shows the recognition results for Infty and FineReader + Infty under strict conditions wherein font types (except for bold type) and font faces are distinguished, and all characters and symbols are used for the evaluation.

We could confirm that the FineReaderfs results do not show its intrinsic performance because we used ASCII-text files to get the recognition results for FineReader. However, the main causes are that there are several recognition errors, especially in respect to the parts with mathematical formulae, which were classified into the 3 following patterns: 1) Ordinary text around mathematical formulae that were not recognized by FineReader, for example, formula numbers. 2) Similar character sets (ex. 'o' and 'O', 'w' and 'W', etc.) after mathematical formulae were not recognized properly,

RECOGNIZER	TOTAL	TEXT	MATH
FineReader	–	99.75%	–
Infty	99.17%	99.71%	97.15%
FineReader + Infty	99.19%	99.80%	96.91%

Table 1. Character recognition results for only ASCII characters

RECOGNIZER	TOTAL	TEXT	MATH
Infty	98.99%	99.53%	96.89%
FineReader + Infty	99.00%	99.64%	96.54%

Table 2. Character recognition results for all characters and fonts

because the formulae caused linguistic information errors. For example, almost of all 'w' after formulae were recognized as 'W'. 3) After a large formula, a character was recognized as a superscript by FineReader. Then linguistic information was not applied to the next words (fig. 3). Thus, mathematical formulae often cause linguistic correction errors for a word.

In the table 1, comparing FineReader with FineReader + Infty in respect to the recognition rates in parts with ordinary text, we can see that FineReader+Infty not only maintains FineReader's recognition rate, but also corrects the misrecognitions cased by mathematical formulae. In the tables 1 and 2, by combining FineReader, Infty's recognition rates of the parts with ordinary text was improved, but Infty's recognition rates for parts with mathematical formulae were slightly worsened. This was caused because FineReader's recognition errors for parts with mathematical formulae were used as the final recognition results in the matching steps.

7. Conclusion

In this paper, we proposed a new system that can be combined with another OCR system via text embedded PDF files. This system can add mathematical information to PDF files generated by another OCR system and correct the recognition errors around mathematical formulae, by utilizing the extracted text files as their recognition results. By using this system, it is also possible to add mathematical information to already digitized documents in digital libraries. To evaluate our system, we recognized PDF files generated by ABBYY FineReader 7.0. In the evaluation, the system was shown to be able to add mathematical information to PDF files without loss of recognition rates for the parts with ordinary text. Moreover, the system also could improve the recognition results for the parts with ordinary text by correcting the recognition errors that occurred around mathematical formulae. However, all the recognition results of FineReader – font information, special characters, bounding rectangle, etc. – could not be used, because only ASCII-text files are used to obtain FineReader's results. Greater improvements can be expected by extracting recognition results directly from PDF files. For mathematical formulae, Inftyfs recognition rates combined with FineReader was not as good as Infty's on their own because the matching method was applied to correctly recognized formulae and so altered their recognition results. It will be necessary to consider the extent to which the matching method is applied.

This system uses another OCR system via PDF files. It appears possible to integrate the best recognition results of several OCR systems, or to update digital libraries whenever OCR systems are upgraded with advances in technology. In this way, documents in digital libraries can be refined.

In this paper, we did not evaluate our system in respect to bibliographies, which are not well recognized by commercial OCR systems because bibliographies are generally multilingual. However, as our system does not use linguistic information, some improvements can also be expected for bibliographies.

Acknowledgments This work was partially supported by the Ministry of Education, Culture, Sports, Science and Technology of Japan under the Kyushu University 21st Century COE Program (Development of Dynamic Mathematics with High Functionality) and a Grant-in-Aid for Scientific Research No.14380182 of the Japan Society for the Promotion of Science.

References

[1] G. O. Michler, "Report on the retrodigitization project "Archiv der Mathematik"," *Archiv der Mathematik*, 77:116–128, 2001.

[2] M. Suzuki, F. Tamari, R. Fukuda, S. Uchida and T. Kanahori, "INFTY — An Integrated OCR System for Mathematical Documents", *ACM Symposium on Document Engineering*, 95–104, 2003.

[3] "Ghostscript, Ghostview and GSview", `http://www.cs.wisc.edu/~ghost/`

[4] Y. Eto and M. Suzuki, "Mathematical formula recognition using virtual link network," *Proc. ICDAR*, 762–767, 2001.

[5] S. Uchida, A. Nomura and M. Suzuki, "Quantitative Analysis of Mathematical Documents", *International Journal on Document Analysis and Recognition*, Vol. 7, No. 4, Springer (2005), 211–128.

[6] M. Suzuki, S. Uchida and A. Nomura, "A Ground-Truthed Mathematical Character and Symbol Image Database", *Proceedings of 8th International Conference on Document Analysis and Recognition*, ICDAR 2005, Seoul, Korea.