IPSJ Transactions on Computer Vision and Applications Vol.6 132–142 (Dec. 2014)

[DOI: 10.2197/ipsjtcva.6.132]

Research Paper

Mathematical Information Retrieval (MIR) from Scanned PDF Documents and MathML Conversion

Azadeh Nazemi[1,a)] Iain Murray[1] David A. McMeekin[2]

Received: April 4, 2014, Accepted: September 22, 2014, Released: December 10, 2014

Abstract: This paper describes part of an ongoing comprehensive research project that is aimed at generating a MathML format from images of mathematical expressions that have been extracted from scanned PDF documents. A MathML representation of a scanned PDF document reduces the document's storage size and encodes the mathematical notation and meaning. The MathML representation then becomes suitable for vocalization and accessible through the use of assistive technologies. In order to achieve an accurate layout analysis of a scanned PDF document, all textual and non-textual components must be recognised, identified and tagged. These components may be text or mathematical expressions and graphics in the form of images, figures, tables and/or diagrams. Mathematical expressions are one of the most significant components within scanned scientific and engineering PDF documents and need to be machine readable for use with assistive technologies. This research is a work in progress and includes multiple different modules: detecting and extracting mathematical expressions, recursive primitive component extraction, non-alphanumerical symbols recognition, structural semantic analysis and merging primitive components to generate the MathML of the scanned PDF document. An optional module converts MathML to audio format using a Text to Speech engine (TTS) to make the document accessible for vision-impaired users.

Keywords: math recognition, graphics recognition, Mathematical Information Retrieval (MIR), Support Vector Machine (SVM)

1. Introduction

Lines in Mathematical PDF documents are categorized into different three types.

1. Text lines (the top of **Fig. 1**)
2. Embedded formulae (EF) lines which are mixed with text (the middle of Fig. 1)
3. Isolated formulae (IF) lines (the bottom of Fig. 1)

Currently there are several methods that facilitate math formulae to be extracted from scanned documents. Jin et al. [1] proposed a Mathematical Formulas Extraction method. The method classifies each line as either Isolated Formulae (IF) or non-Isolated Formulae (Non-IF) using the Parzen Window Classifier technique. The Parzen Window Classification [2] method is a technique for nonparametric density estimation that can also be used for classification. Each class density is separately approximated and a test point, with the maximal posterior probability, is assigned to each class. The resulting algorithm is extremely simple and closely related to Support Vector Machines (SVM). Mathematical expressions are either one-dimensional (1-D) or two-dimensional (2-D). After line classification, the Jin method uses 2-D structure detection. This method does not support 1-D structured EFs [2]. A different approach to Jin et al.'s method for Mathematical Formulas Extraction was presented by Kacem et al. [3]. The Kacem et al. [3] method is based on connected component identification and uses several features, such as boundary boxes, for IF detection and extraction. The EF method is dependent upon the results from both symbol recognition and Optical Character Recognition (OCR). This method specifies the most significant symbol's location and extends it to adjoining symbols using contextual rules. This is continued until the entire formula is delimited, usually through the use of white spaces [3], [4]. Lin et al. [5] identified of embedded mathematical formulas in PDF documents using SVM. This method carries out a word segmentation over the scanned document and then classifies words as either ordinary text or as a formula fragment using SVM classifier [5]. Since an isolated expression has a recognizable geometric layout, most of the previous methods for IF detection and extraction are rule-based. However, embedded expressions are often short and separating equations from text is problematic. Character-based separation is a method to recognize the location of embedded expressions. This method works in conjunction with OCR and specifies the location of some symbols (e.g., the equal sign) position within the line. In cases where the line does not

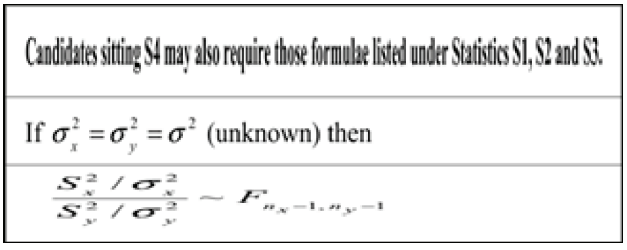


Fig. 1 Different types of lines in mathematics documents.

[1] Electrical and Computer Engineering Department, Curtin University, Perth, Western Australia, Australia
[2] Department of Spatial Sciences, Curtin University, Perth, Western Australia, Australia
[a)] azadeh.nazemi@postgrad.curtin.edu.au

contain the specified symbol, the EF's position remains ambiguous. The method undertaken in this paper to extract mathematical expressions from scanned PDF uses the following steps:

1. Line segmentation
2. Classify line segments to IF and non-IF
3. Apply word segmentation for non-IF lines
4. Classify word blocks to text block and maths block
5. Send IF obtained by step 2, and EF by step 4 to Recursive Components Extraction (RCE) to convert them to linear form

2. Line Segmentation Issues in Mathematical Documents

In terms of line segmentation accuracy in mathematical PDF documents, lines are divided to four categories as follows:

1. Full-Segmented: the line is completely segmented with its subscript or superscript (**Fig. 2**).
2. Over-Segmented: the line can be split into more than one line, or partially detected. **Figure 3** illustrates the 2-dimensional IF line, and shows over-segmentation occurrence during ordinary line segmentation, which splits one line into two separate segments.
3. Under-Segmented: the line is merged with some other lines. **Figure 4** indicates two IF lines and shows under-segmentation occurrence during ordinary line segmentation, which merges the bottom line with the superscript of the previous line and subscript of the next line.
4. Broken symbol or character: **Fig. 5** shows a 2-D EF line with a symbol, which is broken during ordinary line segmentation.

3. Mathematical Information Retrieval Overview

The flowchart shown in **Fig. 6** illustrates several modules involving MIR processing. These modules use image processing techniques and PDF layout analysis.

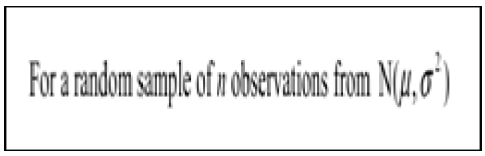


Fig. 2 A Full-Segmented EF line.

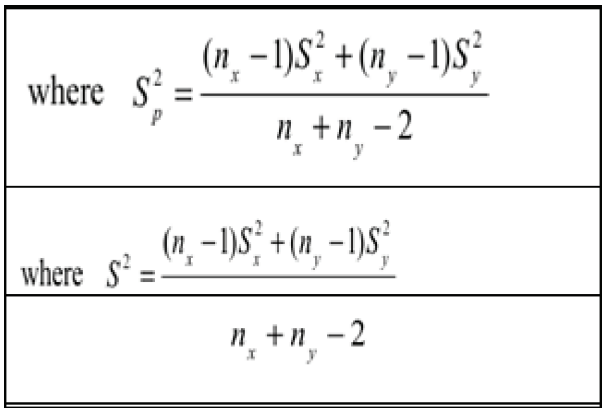


Fig. 3 A 2-D IF line and over segmentation.

4. MIR Modules

4.1 Preprocessing

This module includes:

1. Document image binary conversion
2. Margin removal

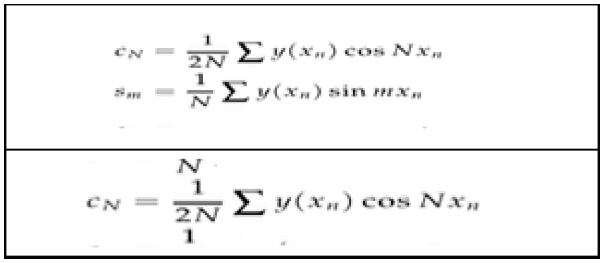


Fig. 4 Under-Segmented 2-D IF line.

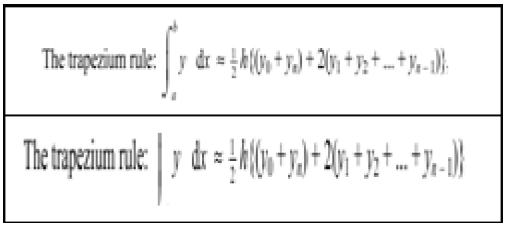


Fig. 5 2-D EF line (top) and broken symbol (bottom).

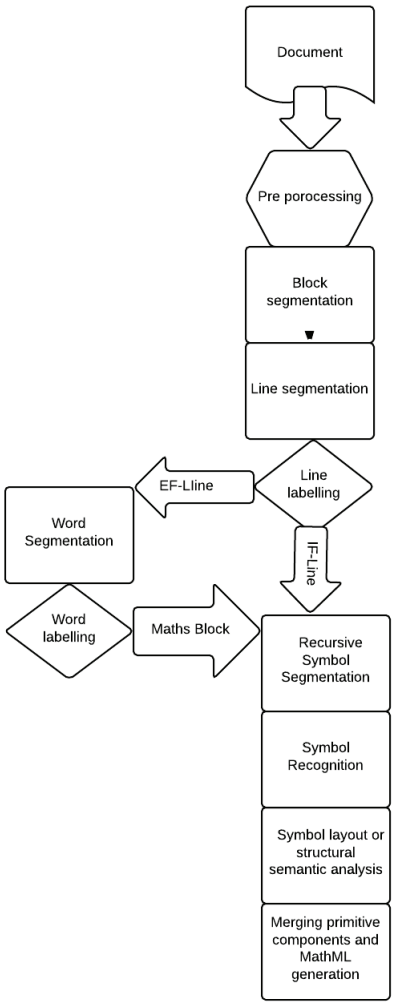


Fig. 6 MIR overview flowchart.

3. Skewing detection and correction
4. Scaling or reducing a large image document by 50% improves image processing speed during segmentation [6]
5. Converting PDF to PNG using the following commands:

```
pdftoppm math.pdf math
convert math.ppm math.png
```

4.2 Block Segmentation

Block segmentation preserves the reading order. In a multi-column document the OCR results would not keep original reading order. Block segmentation is responsible for identifying the extended vertical black lines or extended vertical whitespace. Block segmentation divides the multi-column document image into blocks using combination of morphological operations [6] or by the Recognition by Adaptive Subdivision of Transformation Space (RAST) method to retain the reading order [7].

4.3 Line Segmentation

In this research One-Column-Projection (1CP) is used to segment the lines. It specifies the position of the largest rectangle of whitespace area. **Figure 7** illustrates a mathematical document and its line segmentation. Using the 1CP method for line segmentation solves the issues mentioned in Section 2, even though hand written document pages have been successfully segmented with this method. **Figure 8** illustrates a hand written mathematical page and its 1CP line segmentation result. The following snippet (bash script source codes under Linux) using the open source ImageMagick package performs 1CP and provides bounding boxes

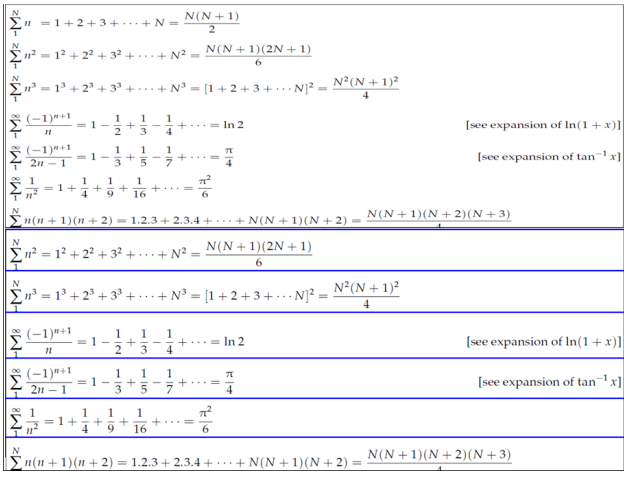


Fig. 7 Line segmentation.

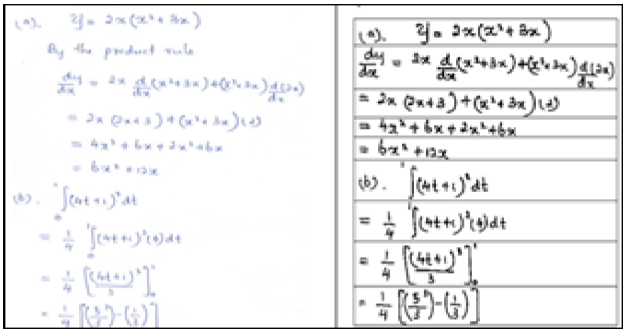


Fig. 8 Line segmentation of a mathematic hand written document.

information for lines. Then make it executable by: `chmod +x`

```
in=$1
convert "$in" -scale 50% bin.mpc
width=`identify -format "%w" bin.mpc`
height=`identify -format "%h" bin.mpc`
convert bin.mpc txt:-|sed 's/:.*#/ /g;1d;s/,/
/g;s/white/1/g;s/black/0/g'|awk '{print $1,$2,$4}
'|sort -b -k2n,2|awk '$3==1'|awk '{ a[$2]++}END
{ for(i in a) print i,a[i]}'|sort -b -k2n,2|awk
''$width'==$2'|sort -b -k1n,1|awk '{print $1}'|
awk 'p{print $1,$1-p,p}{p=$1}'|awk '$2!=1'|sort
-b -k3n,3| awk '{print $1,$3}'>yy.tmp
convert bin.mpc txt:-|sed 's/:.*#/ /g;1d;s/,/ /
g;s/white/1/g;s/black/0/g'|awk '{print $1,$2,$4}'
|sort -b -k2n,2|awk '$3==1'|awk '{ a[$2]++}END
{ for(i in a) print i,a[i]}'|sort -b -k2n,2|awk
''$width'==$2'|sort -b -k1n,1|awk '{print $1}'
|awk 'p{print $1,$1-p,p}{p=$1}'|awk '$2!=1'|sort
-b -k3n,3| awk '{print $1,$3,NR}'|awk '{print
"convert bin.mpc -crop 0x"$1-$2"+0+"$2" " $3"
-hor.mpc"}'>crop.sh
chmod +x crop.sh
./crop.sh
nox=$(cat crop.sh |wc -l)
for (( e=1;e<=$(($nox));e++ ))
do
convert $e-hor.mpc txt:-|sed 's/:.*#/ /g;1d;
s/,/ /g;s/white/1/g;s/black/0/g'|awk '{print
$1,$4}'|awk '$2!=1'>tmp.tmp
xs=$(cat tmp.tmp|sort -b -k1n,1|awk 'NR==1
{ print $1}')
xe=$(cat tmp.tmp|sort -b -k1n,1|awk 'END
{ print $1}')
ye=$(cat yy.tmp |awk 'NR=='$e' { print $1}')
ys=$(cat yy.tmp |awk 'NR=='$e' { print $2}')
echo $xs $xe $ys $ye >>tmp.box
done
cat tmp.box|awk 'p{print NR,$1, '$width'-$2,
$4-$3,$2-$1,$3-p,($4-$3)/($2-$1)}{p=$4}{if
(NR==1)print NR,$1, '$width'-$2,$4-$3,$2-$1,0,
($4-$3)/($2-$1)}'
convert $(ls *-hor.mpc|sort -b -k1n,1)
-background red -splice 0x1+0+1 -append x:
```

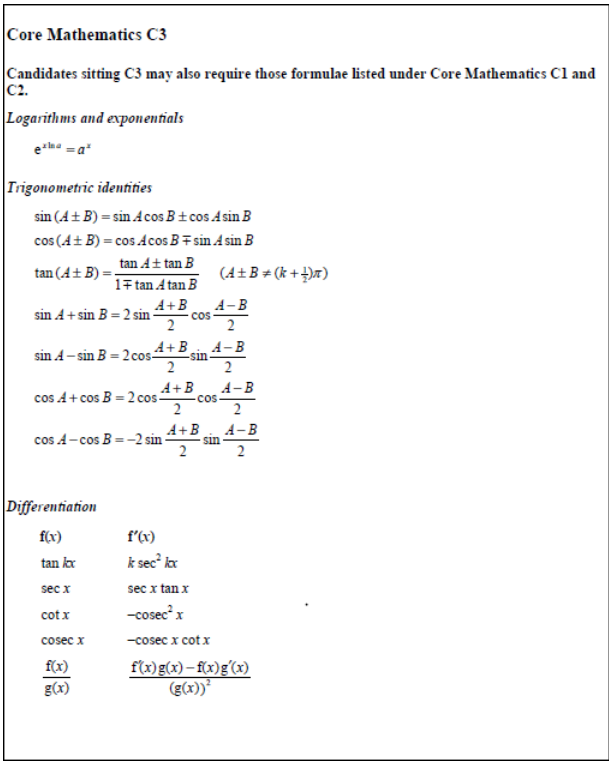
4.4 Global Line Labelling by Data Collection and SVM

Completed line segmentation using 1CP provides lines bounding boxes which are defined by lower left and upper right positions. Suppose bounding box for a line is represented as: Bounding box= (Xmin, Ymax, Xmax, Ymin)

Aspect_Ratio=AR=h/w
Area=w*h
Left margin=LM=Xmin
Number of black pixels=nobp
Density=nobp/area
Right Margin=w-Xmax
vertical_space=vs=Ymin(i)-Ymax(i-1) for two adjacent lines

Table 1 Line features.

Line	LM	RM	H	W	VS	AR
1	13	221	7	74	0	0.094595
2	13	20	8	275	11	0.029091
3	13	286	7	9	0	0.777778
4	13	213	8	82	6	0.097561
5	26	257	7	25	6	0.28
6	14	228	8	66	12	0.121212
7	26	181	8	101	5	0.079208
8	26	180	8	102	3	0.078431
9	26	145	17	137	3	0.124088
10	26	183	17	99	3	0.171717
11	26	184	17	98	2	0.173469
12	26	180	17	102	3	0.166667
13	26	178	17	104	2	0.163462
14	13	254	8	41	20	0.195122
15	29	227	7	52	6	0.134615
16	29	216	9	63	3	0.142857
17	29	211	6	68	6	0.088235
18	29	213	8	66	4	0.121212
19	29	201	6	78	6	0.076923
20	30	182	19	96	5	0.197917



Core Mathematics C3

Candidates sitting C3 may also require those formulae listed under Core Mathematics C1 and C2.

Logarithms and exponentials

$e^{x \ln a} = a^x$

Trigonometric identities

$\sin(A \pm B) = \sin A \cos B \pm \cos A \sin B$

$\cos(A \pm B) = \cos A \cos B \mp \sin A \sin B$

$\tan(A \pm B) = \dfrac{\tan A \pm \tan B}{1 \mp \tan A \tan B} \quad (A \pm B \neq (k + \frac{1}{2})\pi)$

$\sin A + \sin B = 2 \sin \dfrac{A+B}{2} \cos \dfrac{A-B}{2}$

$\sin A - \sin B = 2 \cos \dfrac{A+B}{2} \sin \dfrac{A-B}{2}$

$\cos A + \cos B = 2 \cos \dfrac{A+B}{2} \cos \dfrac{A-B}{2}$

$\cos A - \cos B = -2 \sin \dfrac{A+B}{2} \sin \dfrac{A-B}{2}$

Differentiation

f(x)	f′(x)
$\tan kx$	$k \sec^2 kx$
$\sec x$	$\sec x \tan x$
$\cot x$	$-\mathrm{cosec}^2 x$
$\mathrm{cosec}\, x$	$-\mathrm{cosec}\, x \cot x$
$\dfrac{f(x)}{g(x)}$	$\dfrac{f'(x)g(x) - f(x)g'(x)}{(g(x))^2}$

Table 1 indicates line features extracted by line segmentation of the sample mathematical page.

Segmented lines by single column projection technique are divided in to several categories. These categories are:

Text only
Linear Embedded Formula
Multi-Dimensional Embedded Formula (EF)
Multi-Dimensional Isolated Formula (IF)
Linear Isolated Formula
Caption
Running Footer
Running Header
Heading Title

The lines are classified using features such as: Left Margin, Right Margin, Aspect Ratio, Height, Density, and Vertical Spaces. Then the lines which are labelled as IF or EF are sent to the Mathematical Information Retrieval (MIR) module for further investigation. The threshold value of each feature is used to compare features in the datasets. Mean, mode and median are used to find the threshold value. These three values are calculated with the following formulae:

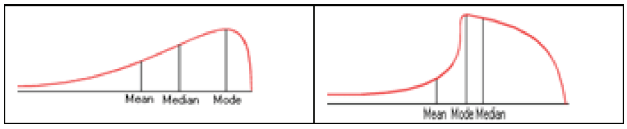


Fig. 9 Threshold value=Mode(x) (left), Threshold value=Median(x) (right).

$$Mean(x) = \frac{\sum_{i=1}^{n}(x_i)}{n} 1 < i < n$$

$$Median(x) = \frac{x_{\frac{n}{2}} + x_{\frac{n}{2}+1}}{2}$$

$$Mode(x) = most\ frequent\ x$$

If Mode(x) >Mean(x) and Mode(x)>Median, the data set is skewed to the left or it is negatively skewed. In this situation the mean and the median are both less than the mode. Generally, most of the time when the data is skewed to the left, the mean will be less than the median.

If: Mean(x) <=Median(x) <=Mode(x) then Threshold-value(x) =Mode(x) (**Fig. 9** left)

If: Mode (x) <Mean(x) and Mode(x) <Median(x) it means the data set is skewed to the right then Threshold-value(x) = Median(x) (Fig. 9 right)

The below snippets for feature i represented by f_i have been collected in a file (file.dat) in order to obtaint the mode, frequency, mean, median, minimum and maximum.

```
Mode(fi)=$(cat file.dat| awk '{ a[$1]++}END {
for(i in a) print i,a[i]}' |sort -b -k1n,1|awk
'END {print $1}')
Freq(fi)=$(cat file.dat| awk '{ a[$1]++}END {
for(i in a) print i,a[i]}' |sort -b -k2n,2|awk
'END {print $2}')
mean(fi)=$(awk '{sum+=$1} END { print sum/NR}'
file.dat)
median(fi)=$(sort -b -k1n,1 cat file.dat | awk
' { a[i++]=$1; } END { x=int((i+1)/2); if (x <
(i+1)/2) print (a[x-1]+a[x])/2; else print
a[x-1]; }')
min(fi)=$(cat file.dat|sort -b -k1n,1|awk
'$1!=0'|awk 'NR==1{print $1}')
max(fi)=$(cat file.dat|sort -b -k1n,1|awk
'END {print $1}')
```

The line labelling module first focuses on mathematical expression attribute to classify IF.

The line may be IF: if the line meets one the following conditions:

1. If height >median (normal text characters height).
2. Mathematical expressions include subscript and superscripts (S&S) or tall symbols (aspect ratio > 1.3) are often written using 2–3 row to accommodate them).
3. If the line includes a horizontal line separating the numerator

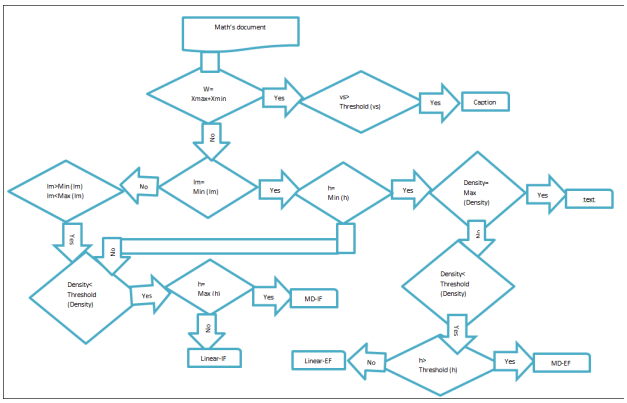


Fig. 10 Line labelling overview flowchart.

and denominator portions.

4. If Density<threshold (density).

Figure 10 illustrates a flowchart to show an overview of the line labelling module based on feature extracted.

W=Line-width
H=Line-height
LM=Left-margin
IF=Isolated-formula
EF=Embedded-formula
VS=Vertical-space
I=Line-number
MD=Multi-dimension

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. SVM algorithms calculate the optimal hyperplane to categorize new examples using labelled training data. This is known as supervised learning [8]. Utilizing SVM as a classifier needs a training data file and a model file. In order to generate a training data file for binary classifier, features (Vector= [W, H, LM, AR, VS, Density] [9]) were collected from more than 40 pages (about 700 lines) of mathematical expressions from the Mathematical Formula Handbook and the N38210A-GCE Mathematical Formulae Statistical Tables. Then target values were manually assigned to each line. $(1, -1)$ are assigned to non-IF, and IF respectively to classify IF or non-IF. To get optimal results heading lines (title lines) are not included in the tetraining data file.

The lines with the following properties are considered heading line and labelled as non-IF lines:

left_margin !=Mode(lm)
Density>Threshold (density)
Vertical Space>Threshold (vertical-space)

The next step is the SVM model file generation utilizing this command:

```
svm_learn traindatat.dat model
```

To apply a new line to the classifier, SVM model file and data file contains feature vector are used by this command:

```
svm_classify data.dat model Result.txt
```

"Resault.txt" includes classifier result for data file.

4.5 EF Extraction Using Word Segmentation

Isolated formulae detection is performed based only on features of the geometric layout and SVM classification because

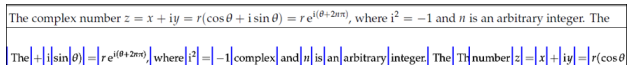


Fig. 11 Word segmentation.

text lines and isolated expressions are significantly different in properties such as height, separation, character sizes and symbol layout [10] whereas position identification of embedded mathematical expression between ordinary text is more complicated. Unlike ordinary text, mathematical expressions in most cases are 2-Dimentional. Various types of 2-D math's syntax include:

- Fraction
- Square root $\sqrt{}$
- Integral $\int$
- Symbol with super and/or subscript right above and/or right under it. $\prod$ $\sum$
- lim
- Accent sign $\bar{}$

After global line labelling is completed using SVM classification; each non-IF labeled line is either plain text or contains an embedded formula is applied to the noise removal module in order to reduce the possible noise. Noise removal is used to perform central smoothing by the following formulae [11]. Supposing each pixel is represented by

P(Xi,Yi)
Xi= 0.25Xi+1+ 0.5Xi+ 0.25Xi-1
Yi= 0.25Yi+1+ 0.5Yi+ 0.25Yi-1

Figure 11 shows three samples for word segmentation.

The following snippet is used to perform word segmentation for segmented lines. Word segmentation specifes the white space area between characters or mathematical symbols.

```
#!/bin/bash
nam=$(echo "$in"|sed 's/\./!/g'|sed 's/!.*//g' )
width='identify -format "%w" bin.png'
height='identify -format "%h" bin.png'
nox=$(convert bin.png txt:-|sed 's/:.*#/ /g;1d;
s/,/ /g;s/white/1/g;s/black/0/g'|awk '{print $1,
$2,$4}'|sort -b -k2n,2|awk '$3==1'|awk '{
a[$2]++}END { for(i in a) print i,a[i]}'|sort -b
-k2n,2|awk ''$width'-$2<2'|sort -b -k1n,1|awk
'{print $1}'|awk 'p{print $1,$1-p,p}{p=$1}'|awk
'$2!=1'|wc -l)
convert bin.png txt:-|sed 's/:.*#/ /g;1d;s/,/
/g;s/white/1/g;s/black/0/g'|awk '{print $1,$2,
$4}'|sort -b -k2n,2|awk '$3==1'|awk '{ a[$2]++}
END { for(i in a) print i,a[i]}'|sort -b -k2n,2
|awk ''$width'==$2'|sort -b -k1n,1|awk '{print
$1}'|awk 'p{print $1,$1-p,p}{p=$1}'|awk '$2!=1'
|sort -b -k3n,3| awk '{print $1,$3,NR}'|awk
'{print "convert bin.png -crop 0x"$1-$2"+0+"$2"
-rotate -90 " $3".mpc"}'>$in.sh
chmod +x $in.sh
./$in.sh
convert $(ls *.mpc|sort -b -k1n,1) -background
red -splice 0x1+0+1 -append x:
```

Then SVM classifies each word block as either a text block or a mathematical block using the following features:

w=Xmax-Xmin h=Ymax-Ymin

Horizontal−space=hs= Xmin(i)- Xmax(i-1) supposing w(i) and w(i+1) are two adjacent words

Aspect-Ratio =h/w

Area=w*h Density(i)=Number of black pixels/area(i)

- *x-deviation-word* $= \sigma_x = \sqrt{\frac{\sum_{i=1}^{n}(x_i - mean(x))^2}{n}}$
- *y-deviation-word* $= \sigma_y = \sqrt{\frac{\sum_{i=1}^{n}(y_i - mean(y))^2}{n}}$

After geometric classification, text blocks are applied to character based classification.

If a text block contains one of the following items then it is considered as a part of the mathematical block.

1. Greek alphabet
2. Latin alphabet
3. Mathematic standard functions such as: sin, cos, tan, csc, sec, cot, sinh, cosh, tanh, log, ln, det, dim, lim, mod, gcd, lcm

4.6 Recursive Symbol Segmentation and Bracket Rule

Since in many cases mathematical expressions are multi-dimensional, symbol segmentation or primitive component extraction must be accomplished both vertically and horizontally. For this purpose, several segmentation techniques were used, compared and evaluated. The Voronoi-diagram based segmentation algorithm is a bottom-up algorithm, which extracts sample points from the boundaries of the connected components. A Voronoi diagram is then generated using sample points obtained from the borders of the connected components. The Voronoi edges that pass through a connected component are then deleted to obtain a Voronoi diagram area. Unnecessary Voronoi edges are deleted to obtain boundaries of document components. The output of the algorithm consists of arbitrarily shaped regions bounded by Voronoi edges. Voronoi performs segmentation by connected components [12]. **Figure 12** proves that using Voronoi solves the Impervious Component Extraction (ICE) issue for some symbols like the radical symbol [13]. Additionally the Voronoi result provides bounding boxes and geometric properties for all mathematical components, which are sufficient to symbolize role recognition and semantic structure analysis.

Run Length Smearing Algorithm (RLSA) [14] is another segmentation algorithm tried within this module. The algorithm transforms a binary sequence x into y. Except in cases, which mathematical expressions contain impervious components RLSA provides better results compared to Voronoi and under-segmentation occurrence is less than that of Voronoi. To improve primitive component extraction accuracy, Recursive Component Extraction (RCE) has been developed which contains two sub modules: Vertical Component Extraction (VCE) and Horizontal Component Extraction (HCE). To reduce ambiguity, each VCE output first must be surrounded by ( ), then recursively applied to the HCE module until all components are extracted [13]. The

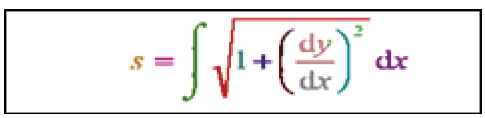


Fig. 12 Voronoi result for sample EF.

brackets Rule or, considering brackets ( ) around each VCE segment, is designed to investigate order, precedence and associativity of operators and addressing symbols relations ambiguity.

For example, without the brackets rule some expressions such as $\frac{a+b}{c}$ and $a + \frac{b}{c}$ are not recognizable. Using the brackets rule changes them to $\left(\frac{a+b}{c}\right)$ and $(a) + \left(\frac{b}{c}\right)$ which are totally different. The bracket rule covers the following concepts to find operators' order in expressions.

1) Operator range defines legal spatial locations for arguments of an operator (e.g., for '+', or fractions).
2) Operator dominance [15], defines a partial ordering on the application of operators' predicates. An operator which nests completely within the range of another operator/relation is called dominated. For example, the + in $\frac{a+b}{c}$ is dominated by the fraction line. Dominating operators are applied after the operators they dominate. One operator dominates another if and only if the latter is in the range of the former and the converse is false.
3) Operator associativity is employed when two or more of the same operator appear in each other range. For example, addition is normally left-associative: $a + b + c = a + (b + c)$.
4) Operator precedence is applied to different operators when they are within each other's range. For example, $a + b \times c = a + (b \times c)$ or by ordering the operators "+" and "/" of the string $a + \frac{b}{c} \Rightarrow a + \left(\frac{b}{c}\right)$

The following snippets represent VCE and HCE functions which call each other in order to provide primitive components. They are used to convert multi-dimensional mathematical expression to linear form.

```
  #!/bin/bash
  #Vertical component extraction
  fin="$1"
  name=$(echo "$fin"|sed 's/\./!/g'|
sed 's/!.*//g' )
  fb="fbin"$name convert "$fin" -threshold 60%
fbin$name.png
  width='identify -format "%w" fbin$name.png'
  height='identify -format "%h" fbin$name.png'
  if [[ $(convert fbin$name.png txt:-|sed
's/:.*#/ /g;1d;s/,/ /g;s/white/1/g;s/black/0/g'|
awk '{print $1,$2,$4}'|sort -b -k1n,1|awk '$3==1'
|awk '{ a[$1]++}END { for(i in a) print i,a[i]}'
|sort -b -k1n,1|awk ''$height'-$2==0'|awk
'{print $1}'|awk 'p{print $1,$1-p,p}{p=$1}'|awk
'$2>2'|wc -l) -gt 2 ]];then
  convert fbin$name.png txt:-|sed 's/:.*#/ /g;
1d;s/,/ /g;s/white/1/g;s/black/0/g'|awk '{print
$1,$2,$4}'|sort -b -k1n,1|awk '$3==1'|awk '{
a[$1]++}END { for(i in a) print i,a[i]}'|sort
-b -k1n,1|awk ''$height'-$2==0'|awk '{print $1}'
|awk 'p{print $1,$1-p,p}{p=$1}'|awk '$2>2'|sort
-b -k3n,3| awk '{print $1,$3,NR"'$name'","'$fb'",
'$width','$height'}'|awk '{print "convert
"$4".png -shave 0x0 -repage " $5"x"$6"+0+0 png:
-|convert png:- -crop " $1-$2"x0+"$2"+0 "$3".png
"}'>$fin.sh convert fbin$name.png txt:-|sed
```

```
‘s/:.*#/ /g;1d;s/,/ /g;s/white/1/g;s/black/0/g‘
|awk ‘{print $1,$2,$4}‘|sort −b −k1n,1|awk
‘$3==1‘|awk ‘{ a[$1]++}END { for(i in a) print
i,a[i]}‘|sort −b −k1n,1|awk ‘‘$height‘−$2==0‘
|awk ‘{print $1}‘|awk ‘p{print $1,$1−p,p}{p=$1}‘|
awk ‘$2>2‘|sort −b −k3n,3| awk ‘{print $1,$3,NR
"‘$name‘"}‘|awk ‘{print "./hce "$3".png "}‘
>>$fin.sh
  chmod +x $fin.sh
  echo $fin.sh
  ./$fin.sh fi
   #!/bin.bash
  #Horizontal component extraction
  in="$1"
  nam=$(echo "$in"|sed ‘s/./!/g‘|sed ‘s/!.*//g‘ )
  convert "$in" −threshold 80% bin$nam.png
fbi="bin"$nam
  width=‘identify −format "%w" bin$nam.png‘
  height=‘identify −format "%h" bin$nam.png‘
  if [[ $(convert bin$nam.png txt:−|sed
‘s/:.*#/ /g;1d;s/,/ /g;s/white/1/g;s/black/0/g‘|
awk ‘{print $1,$2,$4}‘|sort −b −k2n,2|awk ‘$3==1‘
|awk ‘{ a[$2]++}END { for(i in a) print i,a[i]}‘|
sort −b −k2n,2|awk ‘‘$width‘==$2‘|sort −b −k1n,1
|awk ‘{print $1}‘|awk ‘p{print $1,$1−p,p}{p=$1}‘|
awk ‘$2>2‘|awk ‘$1−$3>2‘|wc −l) −gt 2 ]];then
  convert bin$nam.png txt:−|sed ‘s/:.*#/ /g;1d;
s/,/ /g;s/white/1/g;s/black/0/g‘|awk ‘{print
$1,$2,$4}‘|sort −b −k2n,2|awk ‘$3==1‘|awk ‘{
a[$2]++}END { for(i in a) print i,a[i]}‘|sort −b
−k2n,2|awk ‘‘$width‘==$2‘|sort −b −k1n,1|awk
‘{print $1}‘|awk ‘p{print $1,$1−p,p}{p=$1}‘|awk
‘$2>2‘|awk ‘$1−$3>2‘|sort −b −k3n,3| awk
‘{print $1,$3,NR"‘$nam‘","‘$fbi‘"}‘|awk ‘{print
"convert "$4".png −crop 0x"$1−$2"+0+"$2"
" $3".png "}‘>$in.sh
  convert bin$nam.png txt:−|sed ‘s/:.*#/ /g;1d;
s/,/ /g;s/white/1/g;s/black/0/g‘|awk ‘{print
$1,$2,$4}‘|sort −b −k2n,2|awk ‘$3==1‘|awk ‘{
a[$2]++}END { for(i in a) print i,a[i]}‘|sort −b
−k2n,2|awk ‘‘$width‘==$2‘|sort −b −k1n,1|awk
‘{print $1}‘|awk ‘p{print $1,$1−p,p}{p=$1}‘|awk
‘$2>2‘|awk ‘$1−$3>2‘|sort −b −k3n,3| awk ‘{print
$1,$3,NR"‘$nam‘"}‘|awk ‘{print " ./vce "$3".png "
}‘>>$in.sh"
  chmod +x $in.sh
  ./$in.sh
```

Figure 13 shows RCE results for a 2D mathematical expression.

4.7 Generating a Symbol Dictionary Using InftyMDB

In the Infty Project, InftyMDB-1 was collated to be used for development. The formulae were collected from 32 pure mathematical articles. Each mathematical formula in the database consists of 10 or more symbols. Original images of the formulae are available [16]. The RCE run for 1,000 mathematical expression

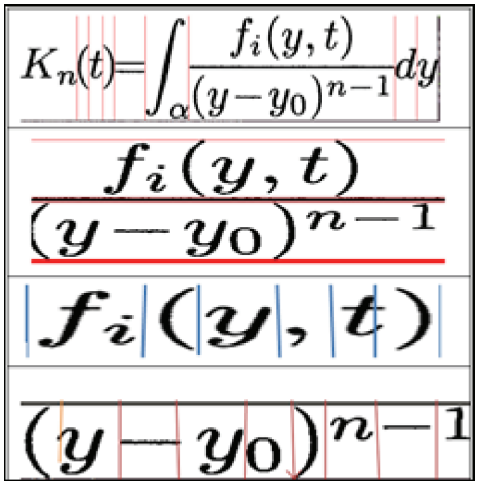
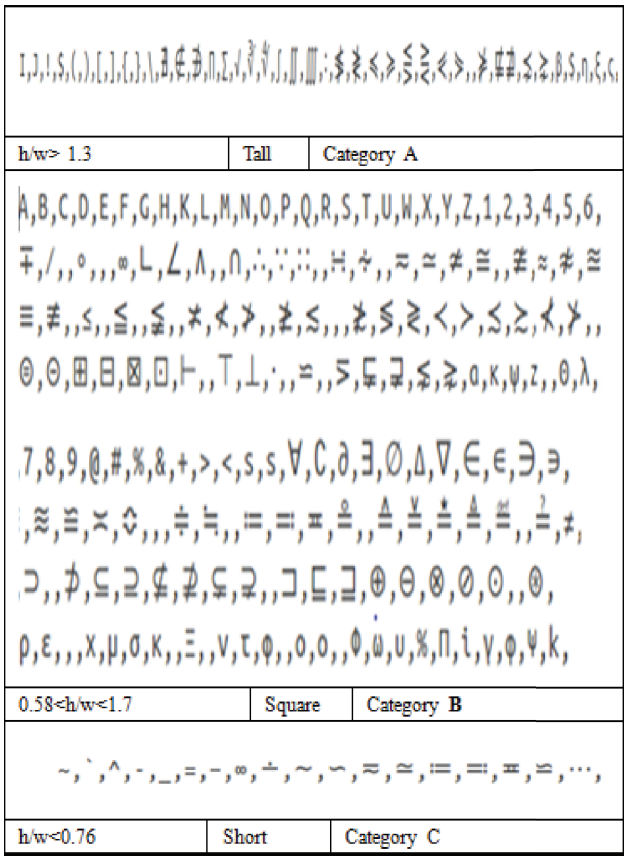


Fig. 13 VCE and HCE result for a sample.

Table 2 Mathematic symbol categories based on mathematic symbol aspect ratio.



h/w> 1.3	Tall	Category A
0.58<h/w<1.7	Square	Category **B**
h/w<0.76	Short	Category C

image files included segmenting them to primitive symbols so as to provide a mathematical symbol dictionary. This dictionary was divided to 3 categories considering the symbols'aspect ratio. Any symbol, belongs to one or two (but never three) categories [17]. **Table 2** shows 3 categories for 258 different symbols and characters considering their aspect ratio. In the end there were, 18 symbol which are short, 206 square and 43 tall symbols. This dictionary is utilized for symbol recognition. For symbol recognition Vector-based Symbol Recognition collects the Aspect Ratio, Area, Density and Norm as the symbol features.

Area (i) =WxH

Density (i) =Number of black pixels/Area (i)

Aspect Ratio = H/W

$$x\text{-}deviation = \sigma_x = \sqrt{\frac{\sum_{i=1}^{n}(x_i - mean(x))^2}{n}}$$

$$y\text{-}deviation = \sigma_y = \sqrt{\frac{\sum_{i=1}^{n}(y_i - mean(y))^2}{n}}$$

$$Norm\ (i) = \sqrt{\sigma_x(i)^2 + \sigma_y(i)^2 + ar(i)^2 + density(i)^2}$$

$$Vector\ (i) = [\sigma_x, \sigma_y, ar, density, norm]$$

The matching function calculates the Euclidean distance between symbol i and all symbols in the dictionary of its category. The matching function is responsible for comparing the Euclidean distance values and finding the smallest one as a result of symbol i recognition.

Euclidean Distance

$$= \sqrt{\begin{array}{c}((\sigma_x(i) - \sigma_x(model))^2 + (\sigma_y(i) - \sigma_y(model))^2 \\ +(ar(i) - ar(model))^2 \\ +(density(i) - density(model))^2\end{array}}$$

4.8 Symbol Layout or Structural Semantic Analysis Issues Regarding Mathematic Layout Analysis Include

1. Two-dimensional structure
2. Non-ordering symbol arrangements
3. No dictionary of all math expressions
4. Large variations in symbol scale
5. Many more symbol classes

The first two issues have been already addressed by the previously described approach (RCE) and using the brackets rules. The mathematic dictionary is generated, as described in the previous section reduces the lack of availability of complete and comprehensive dictionaries. The 4th and 5th issues are addressed by using semantic structure analysis by investigating relationships between two adjacent symbols. Mathematical expressions generally represent an application of functions, operators and relations to arguments. Multiple mathematical statements may be represented by a single expression; in other words, mathematical expressions are polysemic [18]. It means the definition and role of symbols frequently change. Even when the domain is clear, symbol definitions are often ambiguous. Analysis of the spatial relationships between symbols is the symbol layout or structural semantic analysis. The spatial relationships are composed of baseline, subscript, superscript, upper, and lower relations. Since spatial relationships convey the meaning of mathematical expressions, its identification is critical to recognize mathematical expressions. Symbol semantic analysis is considered when solving the ordering and relationship problems noting the symbol role and situation concentration instead of symbol isolation. Semantic analysis detects upper, lower, subscripts and superscripts components, assigning them to their parent symbols and merging them as a single unit [19]. Identifying relationships between each symbol and its previous adjacent symbol has been implemented by measuring the line slope between bounding box corners [20]. As shown in **Fig. 14**, these slopes are captured by the following formulae:

$$Green\ line\ slope\ (i) = Ymax_i - Ymax_{i-1}/Xmin_i - Xmax_{i-1}$$

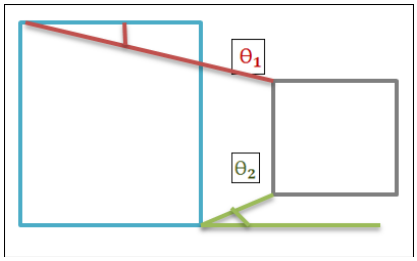


Fig. 14 Slopes between two adjacent symbols.

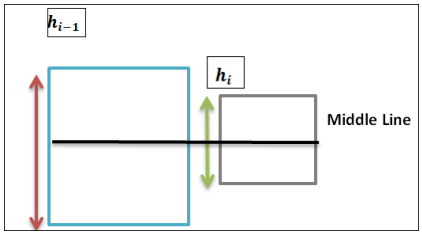


Fig. 15 Relationship between two adjacent mathematic symbols.

Table 3 Different relations between two adjacent symbols in mathematical expressions.

Formula	Extracted features	Relation	Example	Shape
$cc_{i-1}\ cc_i$	HR=1 MLR=0 $0 < \theta_1 < 1$ $\theta_2 < -1$	Left	$M =$	
	$-1 < \theta_1 < 0$ $\theta_2 > 1$	Left	$-\nu_0$	
	$0 < \theta_1 < 1$ $\theta_2 = 0$	Left	$b\Omega$	
$(cc_{i-1}^{cc_i})$	$\theta_1 < -1$ $\theta_2 > 1$	Below	$\sum^{n}$	
$cc_{i-1}\ cc_i$	$0 < MLR < 1$ $HR < 1$ $0 < \theta_1 < 1$ $\theta_2 > 1$	Above-Left	D_i	
$cc_{i-1}^{cc_i}$	$-1 < \theta_1 < 0$ $\theta_2 < 1$	Below-left	x^2	
$\sqrt{cc_i}$ when cc_{i-1} is $\sqrt{\square}$	$0 < MLR < 1$ $HR < 1$ $-1 < \theta_1 < 0$ $\theta_2 < 1$	Inside	$\sqrt{1+\beta^2}$	

$$Red\ line\ slope\ (i) = Ymin_i - Ymin_{i-1}/Xmin_i - Xmin_{i-1}$$

Other useful features to recognize adjacent symbols' relationships and classify them are (**Fig. 15** [21])

$$HR = H\_Ratio(i) = \frac{h_i}{h_{i-1}}$$

$$MLR = middle.line.ratio(i) = \frac{Ymin_i + Ymax_i - Ymax_{i-1} - Ymin_{i-1}}{h_i}$$

Table 3 indicates different relationships between two adjacent symbols.

4.9 Merging Primitive Components and MathML Generation

In order to comply with symbol layout analysis, symbols are inspected from left to right [22]. Symbol layout analysis and symbol classification are based on extracted features format. One of the most critical tasks in this module is detecting dependent components and merging them as a single unit using brackets to sur-

round them. Dependent components may be:

Operands of an individual operator

Parameters of a function

Subscript and superscript of a symbol

Upper and lower part of a symbol

Math symbols are divided to 13 classes, this classification supports symbol layout analysis:

1) simple/ordinary A, x
2) prefix operator $\sum \prod \int$
3) binary operator (conjunction) +−
4) relation/comparison (verb) = < ?
5) Open left/opening delimiter ([{
6) Close right/closing delimiter)] }
7) postfix/punctuation . , ; ! and Ellipses $a_0, a_1, \ldots, a_n$
8) Subscripts and superscripts
9) Accents
10) Binomial
11) Matrices
12) Roots
13) Text or Named operator. These operators are represented by a multi letter abbreviation:
14) arccos, arcsin, arctan, arg, cos, cosh, cot, coth, csc, deg, det, dim, exp, gcd, hom, inf, inj, lim, lg, ker, ln, log, max, min, Pr, sec, sin, sinh, sup, tan, tanh

These parts of mathematical expressions can be recognized using OCR. Symbols of class 3, notably the minus sign, are automatically coerced to class 1, if they do not have a suitable left operand. $cc_{i-1}cc_icc_{i+1}$ supposing cc_i is in the prefix class, if cc_i be located at below or left below of cc_{i+1} and above or above left of cc_{i-1}, then it can be the parent for cc_{i-1}, cc_{i+1} and so $cc_{i-1}cc_icc_{i+1}$ can be placed in the same bracket.

For operands finding of an operator this algorithm must be run:

- Parse from left to right until meeting class 6 in position A
- Parse right to left from position A until reaching class 5 in position B
- Extract the most inner layer between B and A
- Parse from B to A until getting one of the existing symbols in class 3 in position C
- Parse back between C and B to get the first operand
- Parse forward between C +LENGTH (symbol) and A to get the second operand
- Result of RCE is a series of symbol images without ordering. Thus to reorder them so as to keep conceptual meaning, bracket rule, symbol layout analysis and merging algorithm are used as described in the following examples:

Example 1: $\frac{x^2+z}{y}$

VCE and bracket rule convert formula to $(\frac{x^2+z}{y})$

Brackets which are located around only one single component are removed

It is converted to : $\frac{x^2+z}{y}$

HCE converts $\frac{x^2+z}{y}$ to $x^2 + z$, $fraction\,bar$, y

VCE and bracket rule convert $x^2 + z$, $fraction\,bar$, y to $(x)(^2)(+)(z)$, $fraction\,bar$, y

Brackets which are located around only one single component are removed

Component segmentation is completed: $x^2 + z$, $fraction\,bar$, y

Parse from left to right until reach class 2, 3, 4, 5

Reach + in class 3

Parse back looking for first operand for +

Reach 2 which is not at + level (considering Ymax)

Parse back

Reach x at the + level

Calculation 1, 2 shows x is located in the below left of 2

x is parent of 2 and x, 2 should be put in bracket and treated as an unit (x^2)

There is nothing before (x^2)

(x superscript 2) is tagged for being first operand for +

Parse forward after +

Reach z at the (x^2) and + levels

Parse forward

Reach fraction which is not at 3 level

Parse back and tagged z as second operand for +

Consider (x^2) +z as an unit and put them in bracket $((x^2)+z)$

Parse forward after z

Reach fraction bar

Parse back looking for first operand for fraction bar

Reach unit $((x^2)+z)$ and tag it a as numerator

Parse forward after fraction bar

Reach y

There is nothing after y so y is the denominator

Numerator $((x^2)+z)$

Fraction bar

Dominator y

Example 2: $\frac{x^{2+z}}{y}$

VCE and bracket rule convert formula to $(\frac{x^{2+z}}{y})$

Brackets which are located around only one single component are removed

It is converted to: $\frac{x^{2+z}}{y}$

HCE convert it to x^{2+z} $fraction\,bar$, y

VCE and bracket rule convert formula to $x, 2, z, fraction\,bar, y$

All single component are extracted

Parse from left to right until reach class 2, 3, 4 ,5

Reach + in class 3

Parse back looking for first operand for +

Reach 2 which is at + level

Parse back reach x

x is not at the same as 2

2 is tagged as first operand for +

Parse forward after +

Reach z at the 2 same level

Parse forward

Reach $fraction\,bar$, which is not at the same level as z

z is second operand for +

2+z tagged as unit and put in brackets (2+z)

Calculate O_1, O_2 shows x is located in the below left of (2+z)

x is parent of (2+z) and x, (2+z) should be put in bracket and treated as an unit

Parse forward after unit $x^{(2+z)}$

Reach $fraction\,bar$

Parse back looking for first operand for $fraction\,bar$

Reach unit ($x^{(2+z)}$) and tag it as numerator
Parse forward after fraction bar
Reach y
There is nothing after y so y is the denominator
Numerator ($x^{(2+z)}$)
Fraction bar
Dominator y

5. Conclusion and Further Development

This research presented to several approaches for MIR from scanned PDF and converted mathematical expressions to MathML, which has the potential to make mathematics accessible to those with visual and print disabilities. It will allow mathematical content to be reused and exchanged with technical computing systems for further manipulation such as rendering with MATHSPEAK (an application to transfer MathML to audio). MATHSPEAK has the ability to preserve conceptual information of formulae, discard the visual description and reduce ambiguity [23]. Further research is required to deal with handwritten mathematical document.

References

[1] Jianming, J., Xionghu, H. and Qingren, W.: Mathematical formulas extraction, *Proc. 7th International Conference on Document Analysis and Recognition* (2003).

[2] Duda, R.O. and Hart, P.E.: *Pattern Classification and Scene Analysis*, Wiley (1973).

[3] Kacem, A., Belaïd, A. and Ahmed, B.M.: Automatic extraction of printed mathematical formulas using fuzzy logic and propagation of context, *International Journal on Document Analysis and Recognition*, Vol.4, No.2, pp.97–108, DOI: 10.1007/s100320100064 (2001).

[4] Kacem, A., Belaid, A. and Ahmed, M.B.: Embedded formulas extraction, *Proc. 15th International Conference on Pattern Recognition* (2000).

[5] Lin, X., Liangcai, G., Tang, Z. and Hu, X.: Identification of embedded mathematical formulas in PDF documents using SVM, *Proc. SPIE 8297, Document Recognition and Retrieval XIX*, *82970D*, DOI: 10.1117/12.912445 (2012).

[6] Nazemi, A., Murray, I. and McMeekin, D.A.: Practical segmentation methods for logical and geometric layout analysis to improve scanned PDF accessibility to Vision Impaired, *International Journal of Signal Processing, Image Processing and Pattern Recognition* (Aug. 2014).

[7] Nazemi, A., Murray, I. and McMeekin, D.: Layout analysis for Scanned PDF and Transformation to the Structured PDF Suitable for Vocalization and Navigation, *J. Comput. Inf. Sci.*, Vol.7, No.1 (Feb. 2014).

[8] Tong, S. and Koller, D.: Support vector machine active learning with applications to text classification, *Journal of Machine Learning Research Archive*, Vol.2, pp.45–66, DOI: 10.1162/153244302760185243 (2002).

[9] Garain, U. and Chaudhuri, B.B.: A syntactic approach for processing mathematical expressions in printed documents, *Proc. 15th International Conference Pattern Recogn.*, Vol.4, pp.523–526 (2000).

[10] Garain, U. and Chaudhuri, B.B.: OCR of printed mathematical expressions, *Digital Document Processing*, pp.235–259, Springer (2007).

[11] Keshari, B. and Watt, S.M.: Hybrid Mathematical Symbol Recognition Using Support Vector Machines, *9th International Conference on Document Analysis and Recognition*, *ICDAR 2007* (2007).

[12] Shafait, F., Keysers, D. and Breuel, T.M.: Performance Evaluation and Benchmarking of Six-Page Segmentation Algorithms, *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol.30, No.6, pp.941–954, DOI: 10.1109/TPAMI.2007.70837 (2008).

[13] Nazemi, A. and Murray, I.: Mathematical Formula Recognition and Transformation to a Linear Format Suitable for Vocalization, *International Journal on Computer Science and Engineering*, Vol.5, No.9 (2013).

[14] Wong, K.Y., Casey, R.G. and Wahl, F.M.: Document analysis system, *IBM Journal of Research and Development*, Vol.26, No.6, pp.647–656 (1982).

[15] Chang, S.-K.: A method for the structural analysis of two-dimensional mathematical expressions, *Inf. Elsevier Sci.*, Vol.2, No.3, pp.253–272, DOI: 10.1016/S0020-0255(70)80052 (1970).

[16] Fujiyoshi, A., Suzuki, M. and Uchida, S.: Verification of mathematical formulae based on a combination of context-free grammar and tree grammar, *Proc. MKM 2008*, *LNCS* (*LNAI*), Vol.5144, pp.415–429 (2008).

[17] Malon, C., Uchida, S. and Suzuki, M.: Support Vector Machines for Mathematical Symbol Recognition, Yeung, D.-Y., Kwok, J.T., Fred, A., Roli, F. and Ridder, D. (Eds.), *Structural, Syntactic, and Statistical Pattern Recognition*, Vol.4109, pp.136–144, Springer Berlin Heidelberg (2006).

[18] Zanibbi, R. and Blostein, D.: Recognition and retrieval of mathematical expressions, *International Journal on Document Analysis and Recognition* (*IJDAR*), Vol.15, No.4, pp.331–357, DOI: 10.1007/s10032-011-0174-4 (2012).

[19] Aly, W., Uchida, S., Fujiyoshi, A. and Suzuki, M.: Statistical classification of spatial relationships among mathematical symbols, *Proc. 10th International Conference on Document Analysis and Recognition*, *ICDAR 2009*, Barcelona, Spain, pp.1350–1354 (2009).

[20] Aly, W., Uchida, S., Fujiyoshi, A. and Suzuki, M.: Identifying Subscripts and Superscripts in Mathematical Documents, *Mathematics in Computer Science*, Vol.2, No.2, pp.195–209, DOI: 10.1007/s11786-008-0051-9 (2008).

[21] Labahn, G., Lank, E., MacLean, S., Marzouk, M. and Tausky, D.: MathBrush: A System for Doing Math on Pen-Based Devices, *8th IAPR International Workshop on Document Analysis Systems*, *DAS '08* (2008).

[22] Li, Y., Wang, K., Tang, L. and Wu, J.: A Two-dimensional Structure Analysis Method of Printing Mathematical Formula Based On Characteristic Characters, *International Conference on Mechatronics and Automation*, *ICMA 2007*, pp.953–957 (2007).

[23] Nazemi, A. and Murray, I.: Mathspeak: An Audio Method for Presenting Mathematical Formulae to Blind Students, (2012), available from ⟨http://hsi2012.debii.edu.au⟩.

Azadeh Nazemi received her B.S. degree in Computer Hardware Engineering from Shiraz University, Iran. She started Master by Research and Ph.D. in September 2010 and 2012 in Curtin University, WA, Australia respectively. She was awarded a Curtin University Postgraduate Scholarship (CUPS) and an Australian Postgraduate Award (APA) scholarship in 2012, respectively. She is presently working towards a Ph.D. degree in Computer engineering at Curtin University, Perth, Australia. Her research area is Assistive Technology and specifically she is working to design Complete Reading System for vision impaired. On November 2012 her research paper has been awarded "Best Student Paper" in 3rd Annual International Conference on Computer Science Education Innovation & Technology (CSEIT 2012) in Singapore. She has nine years of industry experience as an engineer working in consultant companies.

Iain Murray received his B.Eng. (Hons) in Computer Systems Engineering in 1998 and his Ph.D. titled "Instructional eLearning technologies for the vision impaired" in 2008 both at Curtin University. He has worked in the field of assistive technology for more than 25 years both as a practitioner and researcher. Currently employed as a senior lecturer in the Department of Electrical and Computer Engineering, his research interest include learning environments for people with vision impairment, embedded sensors in health applications and assistive technology. He founded the "Cisco Academy for the Vision Impaired" in 2002 to deliver ICT training to vision impaired people globally. He is a Fellow of the Australian Computer Society and a member of the IEEE.

David A. McMeekin received his Ph.D. degree in Software Engineering from Curtin University. He is currently a senior research fellow in the Cooperative Research Centre for Spatial Information based in the Department of Spatial Sciences at Curtin University. His research interests are in the areas of the use of technology for people with differing physical and mental challenges as well as the real time delivery of spatially enabled data through semantic web technologies. He is a member of the IEEE Computer Society and Electronics Society.

(Communicated by *Dong Xu*)