International Journal of Signal Processing, Image Processing and Pattern Recognition
Vol.7, No.4 (2014), pp.23-36
http://dx.doi.org/10.14257/ijsip.2014.7.4.03

Practical Segmentation Methods for Logical and Geometric Layout Analysis to Improve Scanned PDF Accessibility to Vision Impaired

Azadeh Nazemi, Iain Murray and David A. Mc Meekin

Electrical and Computer Engineering Spatial Sciences,
Curtin University ,Perth ,WA,Australia
Azadeh.nazemi@postgrad.curtin.edu.au I,murray@curtin.edu.au.
D.McMeekin@curtin.edu.au

Abstract

The use of electronic documents has rapidly increased in recent decades and the PDF is one the most commonly used electronic document formats. A scanned PDF is an image and does not actually contain any text. For the vision–impaired user who is dependent upon a screen reader to access this information, this format is not useful. Thus addressing PDF accessibility through assistive technology has now become an important concern. PDF layout analysis provides precious formatting information that supports PDF component classification. This classification facilitates the tag generation. Accurate tagging produces a searchable and navigable scanned PDF document. This paper describes several practical segmentation methods which are easy to implement and efficient for PDF layout analysis so that the scanned PDF document can be navigated or searched using assistive technologies.

Keywords: *PDF layout analysis**,** Optical character recognition (OCR), Vision-impaired*

1. Introduction

To generate a scanned PDF document that is accessible withan assistive technology such asa screen readers Optical Character Recognition (OCR) software is required. OCR extracts text from images and creates a plain text format from a scanned PDF document [1]. OCR software output does not include any mark up or tags to help with the documents navigability with assistive technologies.

HTML OCR an open standard which defines a data format for representation of OCR output. The standard aims to embed layout, recognition confidence, style and other information into the recognized text itself. Embedding this data into text in the standard HTML format is used to improve accessibility and achieve navigation ability. [2]

OCRopus is an OCR software package which saves output results in an html (hOCR) file. However, in some cases all of the different components within the PDF document are not indicated. For example hOCR does not have a specific tag for mathematics formulae [3].

Document layout analysis is the process of identifying and categorizing the regions of interest in the scanned image of a text document. It requires the segmentation of text zones from non-textual ones and the arrangement in their correct reading order [4] Detection and labeling of the different zones (or blocks) as text body, illustrations, math symbols, and tables embedded in a document is called geometric layout analysis. Semantic labeling and classification text zones based on different logical roles inside the document (titles, captions, footnote, *etc.*,) the logical layout analysis [5].

Document layout analysisis typically performed before a document image is sent to an OCR engine. The approaches described in this paper aim to improve layout analysis performance and add more specific and detailed tags to hOCR tagging system.

ISSN: 2005-4254 IJSIP

To add more specific and detailed tags to the hOCR tagging system.

Segmentation Modules

Running PDF document component segmentation can provide tags for different parts of the PDF document based on PDF component properties. The layout analysis is the factor, when working with scanned PDF documents that provides the ability for the document to be searched and navigated. This ability is based on the discovery of the bounding boxes of various components from within the scanned PDF document. Performing multiple layer segmentation facilitates the discovery and extraction of the data from the bounding boxes in the scanned PDF document's images. These different segmentation layers are:

- Block segmentation
- Text–image–segmentation
- Line–segmentation
- Word–segmentations
- Vertical–Horizontal–recursive–segmentation

3. Pre Processing

The PDF document image can be either colored or grey scale. The majority of images have a top/bottom, left/right margin. In some cases during the scanning process, the PDF document may rotate slightly. Image color, size, margins and skew are effective factors in the segmentation result. Therefore, the preprocessing module before starting segmentation is essential to generate smooth input data for the first segmentation layer.

In developing an application that automatically pre-processes all of the required steps for the segmentation process, in this research project, the Open Source image processing package, Image Magick was used. The pre-processing includes the following stages:

3.1. Format Conversion

The scanned PDF document is converted to the Magick Persistent Cache (mpc) image file format. MPC is the native in-memory uncompressed file format. This file format is an identical representation of the image in memory in which the file read is directly mapped into memory. The MPC file format is not portable and is not a suitable format for archiving. However, it is a suitable file format as an intermediate step for high-performance image processing. The MPC format requires two files to support a single image. The image attributes are written to one file, with the extension. mpc and the image pixels are written to a second file with the extension cache. In converting the PDF document to. mpc it is first converted to a portable pixel map format (ppm):

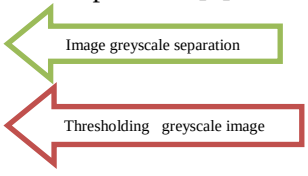
1. *convert document.pdf to document.ppm*
2. *convert document.ppm to document.mpc*

Converting PDF format toMagick Persistent Cache image file format(mpc). MPC is the native in-memory uncompressed file format. This file format is identical to represent images *c*

3.2. Binarization

A greyscale image can be used to create a binary image through thresholding [6]. Thresholding a colour image is done through designating a separate threshold for each RGB component in the image and then combining them with an AND operation [7]. The following steps demonstrate the process:

1. *convert document.mpc to separate.png*



2. *convert separate.png using an 80% threshold to binary.mpc*

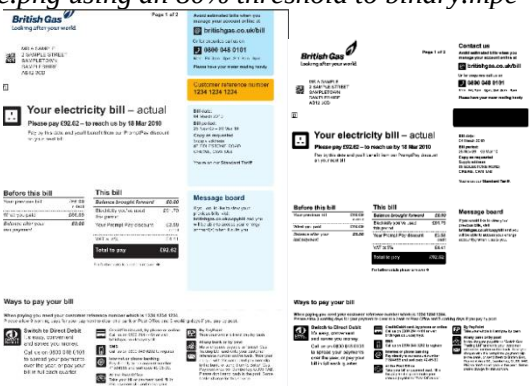


Figure 1. Binarization

3.3. Scaling Image

Reducinga large image document by 50% increases the image processing speed during the segmentation process.

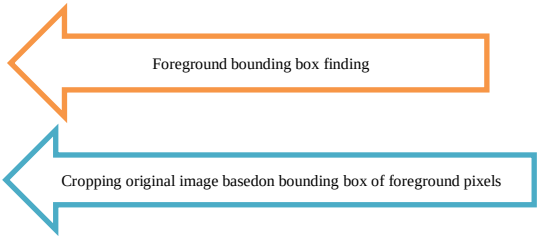
1. *convert binary.mpc -scale by 50% half.mpc*

3.4. Margin Removal

A PDF document layout analysis is based on block segmentation. Accurate recognition of white space areas leads to access block separator segments. Top-bottom and left-right margins often interfere with the accurate recognition of white space areas as block segments separator. Use of a margin removal module removes all margins prior to running the block segmentation. The main tasks for the margin removal module are:

- Finding the bounding box of the foreground pixels:
  (Xmin_foreground _pixels,Xmax_foreground _pixels)
  (Ymin_foreground _pixels,Ymax_foreground _pixels)
- Converting the original image to an image without a margin by cropping the original image using the obtained bounding box

```
in="$1"
ext=$(echo "$in"|sed 's/\./!/g'|sed 's/.*!//g'    )
name=$(echo "$in"|sed 's/\./!/g'|sed 's/!.*//g')
if [[ $ext == "pdf" ]];then
pdftoppm $in ppm
convert ppm-1.ppm  -threshold 80%  $name.png
else
convert $in  -threshold 80%  $name.png
fi
convert  $name.png txt:-|grep -Evwhite|sed 's/:.*#/ /g;1d;s/,/ /g' > pixel.txt
cat pixel.txt |awk '{print $1}'|sort -b -k1n,1 >x.txt
cat pixel.txt |awk '{print $2}'|sort -b -k1n,1 >y.txt
xs=$(cat x.txt |awk 'NR==1')
xe=$(cat x.txt |awk 'END{print }')
ys=$(cat y.txt |awk 'NR==1')
ye=$(cat y.txt |awk 'END{print }')
echo $ye $ys
x=$(($xe-$xs))
y=$(($ye-$ys))
```



convert $name.png -crop $x"x"$y"+"$xs"+"$ys $name"_without_margin.p f

3.5. Skew Detection and Correction

In some cases during the process of scanning into a PDF document,the document may rotate slightly, reducing the PDF document layout analysis' accuracy. To address this issue askew detection and correction module is applied; this is performed before running the PDF document layout analysis. The application of this module, rather than obtaining the bounding box of the foreground pixels in the image, it extracts the following data from. Mpc image:

- X_in _Ymax_foreground_pixels
- Y_in_Xmin_foreground_pixels
- H=Height of the image

```
#!/bin/bash
in="$1"
ext=$(echo "$in"|sed 's/\./!/g'|sed 's/.*!//g'    )
name=$(echo "$in"|sed 's/\./!/g'|sed 's/!.*//g')
if [[ $ext == "pdf" ]];then
pdftoppm $in ppm
convert ppm-1.ppm  -threshold 80%  $name.png
else
convert $in  -threshold 80%   $name.png
fi
convert  $name.png txt:-|grep -Evwhite|sed 's/:.*#/ /g;1d;s/,/ /g' > pixel.txt

cat pixel.txt |awk '{print $1}'|sort -b -k1n,1 >x.txt
cat pixel.txt |awk '{print $2}'|sort -b -k1n,1 >y.txt

xs=$(cat x.txt |awk 'NR==1')
xe=$(cat x.txt |awk 'END{print }')
ys=$(cat y.txt |awk 'NR==1')
ye=$(cat y.txt |awk 'END{print }')

x=$(($xe-$xs))
y=$(($ye-$ys))
convert $name.png -crop $x"x"$y"+"$xs"+"$ys $name"_without_margin.mpc"
convert $name.png -crop $x"x"$y"+"$xs"+"$ys wm.txt
cat  wm.txt|grep -Evwhite|sed 's/:.*#/ /g;1d;s/,/ /g'|awk '{print $1, $2}'   > wpixel.txt
height=$(identify -format "%h"  $name"_without_margin.mpc")

xs=$(cat wpixel.txt|sort -b -k1n,1|awk 'NR==1'|awk '{print $1}')
ye=$(cat wpixel.txt|sort -b -k2n,2|awk '{print $2}'|awk 'END{print}')
yxs=$(cat wpixel.txt|awk '$1=='$xs''|awk '{print $2}'|awk '{sum+=$1} END { print sum/NR}'|bc -l)
xye=$(cat wpixel.txt|awk '$2=='$ye''|awk '{print $1}'|awk '{sum+=$1} END { print sum/NR}'|bc -l)

delx=`echo "$xye" | bc -l`
dely=`echo "$height-$yxs" | bc -l`
tangant=`echo "$dely/$delx" | bc -l`
roting=$(echo "scale=3;a($angle)/0.017453293" | bc -l)
rotating_angle=$(($(echo $roting |awk '{printf "%d" ,$1+1}')))

convert $name"_without_margin.mpc" -rotate  -$rot $name"_unrotate.png"
```

Finding x in y _max and y in x_min

Rotating angle Tangent calculation

Arctangent calculation & rotating angle finding

Rotating original image(–rotating angle)

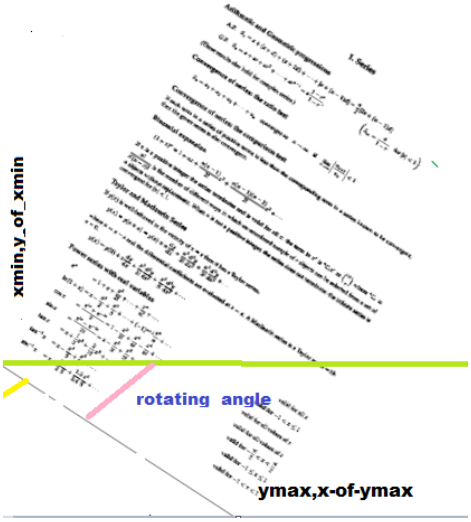


Figure 2. Rotating PDF Document

4. Block Segmentation

Block segmentation divides the page into logical blocks, preserving the reading order. In a double column document the OCR results would not keep the reading order. Block segmentation is responsible for identifyingthe extended vertical black lines or extended vertical whitespace as shown in Figure 3.

Figure 3. Image Morophlogy Dilated and Eroded, Horizontal Block Segnentatuion and Combination of Vertical Horizontal Segmentation

Block Segmentation utilizes the combination of morphological operations it is through the Image Morphology method that the structure of the shapes within an image is able to be cleaned up and studied. This happens by comparing each pixel in the image with its neighbors in multiple ways, so as to add or remove, brighten or darken that pixel. When this is applied over a whole image, even repetitively, specific shapes can be found and/or removed and modified. If a pixel is white and is completely surrounded by other white pixels, this means

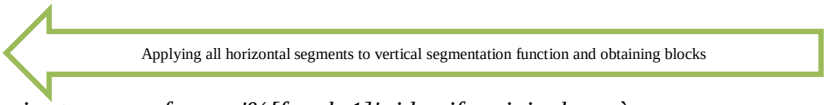
that that pixel is not on the edge of the image [8]. The entire process depends on the definition Kernel that defines what pixels are to be classed as neighbors within each specific morphological method. The *dilate* operation returns the maximum value in the neighborhood while the *erode* operation returns the minimum value in the neighborhood using the following steps:

1. *convert bin.mpc -morphology dilate:1 diamond dilate.mpc*
2. *convert dilate.mpc -morphology erode:5 diamond -clip-mask bin.mpcerode.mpc*

Morphology helps to remove non-interested white space between texts and recognizeremained white space area as blocks separator.

Since PDF document may contain different logical layout such as footer and multi columns, block segmentation must be done both vertical and horizontal as following:

- Considering horizontal white space area
- Running horizontal segmentation
- Considering vertical white space area
- Applying vertical segmentation for all extracted horizontal segments




```
_vertical()
{
n=$1
height=`convert $n"-hor.mpc" +gravity +repage -format '%[fx:u.h-1]' -identify original.mpc`
width=`convert $n"-hor.mpc" +gravity +repage -format '%[fx:u.w-1]' -identify original.mpc`
convert $n-hor.mpc -morphology dilate:1 diamond dilate.mpc
convertdilate.mpc -morphology erode:5 diamond -clip-mask $n-hor.mpcerode.mpc
convert erode.mpc txt:-|sed 's/:.*#/ /g;1d;s/,/ /g;s/white/1/g;s/black/0/g'|awk '{print $1,$2,$4}'|awk '{if
( $3!=1) $3=0}{print $1,$2,$3}'>erode.txt
y=$(cat erode.txt|sort -b -k2n,2|awk '$3==0'|awk '{ a[$2]++}END { for(i in a) print i,a[i]}'|sort -b -
k2n,2|awk 'END{print $1}')

xofy=$(cat erode.txt|sort -b -k2n,2|awk '$3==0 && $2=='$y''|sort -b -k1n,1 |awk '{print $1}'|awk
'p{print $1,$1-p,p}{p=$1}{if (NR==1)print $1,$1,0}'|sort -b -k2n,2|awk '{print $2}'|uniq|tail -n
3|head -n 1)

x=($(cat erode.txt|sort -b -k2n,2|awk '$3==0 && $2=='$y''|sort -b -k1n,1|awk '{print $1}'|awk
'p{print $1,$1-p,p}{p=$1}{if (NR==1)print $1,$1,0}'|awk '$2>'$xofy''|awk '{print $1}'))
noy=${#x[@]}
if [[ $noy -gt 0 ]];then
for (( e=0;e<$(($noy));e++ ))
do
l=$(($e+1))
if [[ $e -ne $(($noy-1)) ]] ;then
convert $n-hor.mpc -crop $((${x[$l]}-${x[$e]}))"x0+"${x[$e]}"+0" $n-ver$e.mpc
else
convert $n-hor.mpc -crop $(($width-${x[$e]}))"x0+"${x[$e]}"+0" $n-ver$e.mpc
fi
done

else
convert $n-hor.mpc $n-ver1.mpc
fi
convert $n-ver*.mpc -rotate 90 -background blue -splice 1x3+0+0 -append -rotate -90 result$n.mpc
}
in="$1"
```



```
height=`convert "$in" +gravity +repage -format '%[fx:u.h-1]' -identify original.mpc`
convert "$in" -threshold 60% -fuzz 1% -trim +repage  -scale 50% bin.mpc
convertbin.mpc -morphology dilate:1            diamond   dilate.mpc
convertdilate.mpc   -morphology erode:5 diamond -clip-mask bin.mpcerode.mpc

mh=$(convert erode.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 'END{print $2}')
echo $mh
x=($(convert  erode.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[$2]++}END { for(i in a)  print i,a[i]}'|sort -b -
k2n,2|awk "$mh'-$2<5'|sort -b -k1n,1|awk '{print $1}'|awk 'p{print $1,p}{p=$1}'|awk '$1-$2!=1'|awk
'{print $1}'))
nox=${#x[@]}

for (( e=0;e<$(($nox));e++ ))
do
l=$(($e+1))
if [[ $e  -ne  $(($nox-1)) ]] ;then
convertbin.mpc -crop "0x"$(($(x[$l]}-${x[$e]}))"+0+"${x[$e]} $e-hor.mpc
else
convertbin.mpc -crop "0x"$(($height-${x[$e]}))"+0+"${x[$e]} $e-hor.mpc
fi

done
for (( v=0;v<$(($nox));v++ ))
do
_vertical $v
done
convert *-hor.mpc  -background red -splice 1x1+0+0 -append horizon.png
convert result*.mpc  -background blue -splice 1x1+0+0 -append x:
```

Horizontal space area finding and performing horizontal segmentation

As in Figure 4 is shown this method is suitable for horizontal block segmentation For a bill document but during vertical segmentation over segmentation is occurred.

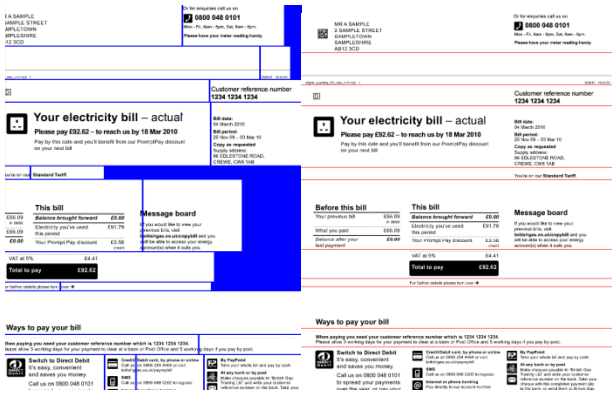


Figure 4. Sample Bill Segmentation

5. Text-Image Segmentation

Morphology is also used in the separation textual components from graphical components in image documents. To separate the text from the image, first the binary image is dilated,

thenthe conditional eroding is performed on the dilated image using the original image as the mask until the image part is complete again. The original image is used as a mask to protect parts of the image from changes.It will restore all shapes that still have some seed part left, so only the image has been left. Since image parts contain fatter lines and larger blobs than text parts, images can be extracted by dilating the image until all letters are gone, but some parts of the image still remain

convertBLOCKsegment.mpc -morphology dilate: 3 diamond dilated.mpc

Dilating the BINARY block segmented

convertBLOCK_segment.mpc -morphology dilate:3 diamond mpc:-|convert mpc:-txt:-|grep -Ev '#FFFFFF'|sed '1d;s/:.//g;s/,/ /g'>rgb.txt*
xs=$(cat rgb0.txt|awk '{print $1}'|sort -b -k1n,1|awk 'NR==1')
xe=$(cat rgb0.txt|awk '{print $1}'|sort -b -k1n,1 |awk END'{print}')
ys=$(cat rgb0.txt| awk '{print $2}'|sort -b -k1n,1|awk 'NR==1')
ye=$(cat rgb0.txt|awk '{print $2}'|sort -b -k1n,1|awk END'{print}')
x=$(($xe-$xs));y=$(($ye-$ys))

Obtaining bounding box of graphical part remaining in dilated image

convertBLOCK_segment .mpc -draw "fill white rectangle $xs,$ys $xe,$ye" textonly.mpc

Making non textual part white, what left from original image is text

convertBLOCKsegemnt.mpc -crop $x"x"$y"+"$xs"+"$ysimageonly .mpc

Extracting image only from original image using its bounding box

Another method to extract textual part is:

convertdilated.mpc -morphology erode:20 diamond -clip-mask BLOCKsegment.mpceroded.mpc

Eroding dilated image

Finally using image contains only image and original image to obtain text only image*:*
convert eroded.mpc -negate BLOCK_segment.mpc -compose plus -composite tesxtonly.png
Figue.5. illustrates text and graphical component segmentation

Figure 5. Text-Image Segmentation

6. Line Segmentation

Once the block segmentation and the text-image segmentation hasfinished executing,the line segmentation modules is executed generating all of the lines and bounding boxes.
#!/bin/bash


```
#One_Column_Page_Line_Segmentation
in="$1"
convert "$in" -threshold 60% -scale 50% bin.mpc
convert bin.png txt:-|sed 's/:.*#/ /g;1d;s/,/ /g;s/white/1/g;s/black/0/g'|awk '{print
$1,$2,$4}'>data.dat
```

Binarization and scaling

Pixel coordinate value(x,y) and color extraction, considering 1 for background and 0 for foreground

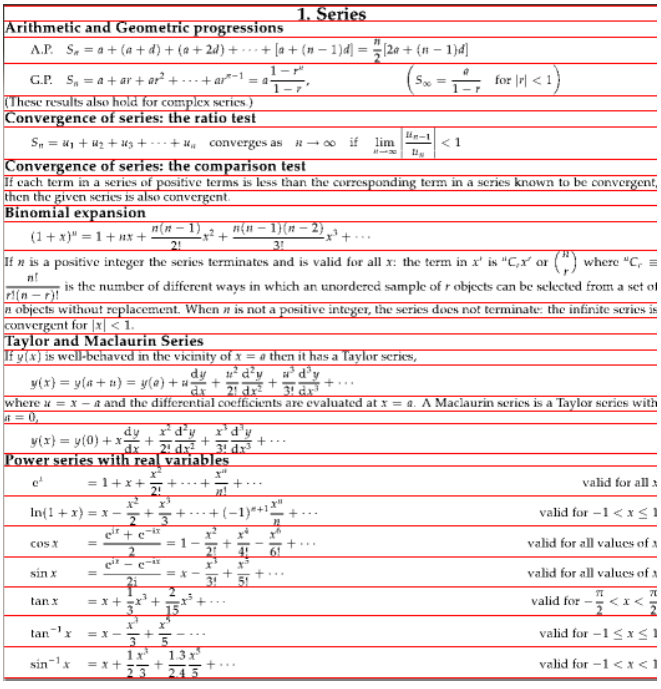
```
height=`convert bin.png +gravity +repage -format '%[fx:u.h-1]' -identify original.mpc`
width=`convert bin.png +gravity +repage -format '%[fx:u.w-1]' -identify original.mpc`
for (( i=0;i<=$height;i++ ))
do
hgap=$(cat data.dat|awk '$2=='$i''|awk '$3==1'|wc -l)
diff=$(($hgap-$width))
if [[ ( $diff -lt 3)&&( $diff -gt -3) ]]; then
echo $i>>hgap.dat
fi
done
```

Obtain exact image size

Finding all lines with background colour as horizontal gaps

```
'base="bin.png"
cat hgap.dat|awk 'p{print $1,p,$1-pf}{p=$1}'|awk '$3>5'|awk 'p{print p,$2,NR}{p=$2}'|awk
'{print "convert '$base' -crop 0x"$2-$1"+0+"$1" line"$3".png"}'>>crop.sh
chmod +x crop.sh
./crop.sh
convert line*.png -background red -splice 0x1+0+0 -append x:
```

Crop original image based on horizontal gaps position to create line segments.



1. Series

Arithmetic and Geometric progressions

A.P. $S_n = a + (a+d) + (a+2d) + \cdots + [a+(n-1)d] = \frac{n}{2}[2a+(n-1)d]$

G.P. $S_n = a + ar + ar^2 + \cdots + ar^{n-1} = a\frac{1-r^n}{1-r}, \quad \left(S_\infty = \frac{a}{1-r} \quad \text{for } |r|<1\right)$

(These results also hold for complex series.)

Convergence of series: the ratio test

$S_n = u_1 + u_2 + u_3 + \cdots + u_n$ converges as $n \to \infty$ if $\lim_{n\to\infty}\left|\frac{u_{n-1}}{u_n}\right| < 1$

Convergence of series: the comparison test

If each term in a series of positive terms is less than the corresponding term in a series known to be convergent, then the given series is also convergent.

Binomial expansion

$(1+x)^n = 1 + nx + \frac{n(n-1)}{2!}x^2 + \frac{n(n-1)(n-2)}{3!}x^3 + \cdots$

If n is a positive integer the series terminates and is valid for all x: the term in x^r is $^nC_r x^r$ or $\binom{n}{r}$ where $^nC_r \equiv \frac{n!}{r!(n-r)!}$ is the number of different ways in which an unordered sample of r objects can be selected from a set of n objects without replacement. When n is not a positive integer, the series does not terminate: the infinite series is convergent for $|x| < 1$.

Taylor and Maclaurin Series

If $y(x)$ is well-behaved in the vicinity of $x = a$ then it has a Taylor series,

$y(x) = y(a+u) = y(a) + u\frac{dy}{dx} + \frac{u^2}{2!}\frac{d^2y}{dx^2} + \frac{u^3}{3!}\frac{d^3y}{dx^3} + \cdots$

where $u = x - a$ and the differential coefficients are evaluated at $x = a$. A Maclaurin series is a Taylor series with $a = 0$,

$y(x) = y(0) + x\frac{dy}{dx} + \frac{x^2}{2!}\frac{d^2y}{dx^2} + \frac{x^3}{3!}\frac{d^3y}{dx^3} + \cdots$

Power series with real variables

e^x	$= 1 + x + \frac{x^2}{2!} + \cdots + \frac{x^n}{n!} + \cdots$	valid for all x
$\ln(1+x)$	$= x - \frac{x^2}{2} + \frac{x^3}{3} + \cdots + (-1)^{n+1}\frac{x^n}{n} + \cdots$	valid for $-1 < x \le 1$
$\cos x$	$= \frac{e^{ix} + e^{-ix}}{2} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots$	valid for all values of x
$\sin x$	$= \frac{e^{ix} - e^{-ix}}{2i} = x - \frac{x^3}{3!} + \frac{x^5}{5!} + \cdots$	valid for all values of x
$\tan x$	$= x + \frac{1}{3}x^3 + \frac{2}{15}x^5 + \cdots$	valid for $-\frac{\pi}{2} < x < \frac{\pi}{2}$
$\tan^{-1} x$	$= x - \frac{x^3}{3} + \frac{x^5}{5} - \cdots$	valid for $-1 \le x \le 1$
$\sin^{-1} x$	$= x + \frac{1}{2}\frac{x^3}{3} + \frac{1\,3}{2\,4}\frac{x^5}{5} + \cdots$	valid for $-1 < x < 1$

Figure 6. Matematical PDF Line Segmentation

The bounding box for each line is represented by a pair of coordinate values*:*

*(x_0, y_0)(x_1, y_1) or (x_0 ,y_0,x_1,y_1)*all following lines properties can be collected and used to logical layout analysis and generate appropriate tag for PDF components.

Line Intention =Left Margin=lm=x_0

h=$y_1 - y_0$ or h=$(identify-format "%h" line.bin.png)

w=$x_1 - x_0$ or w=$(identify-format "%w" line.bin.png)

Right Margin=rm=w- x_1

White Spaces=ws

Vertical Spaces Between Lines=vs

no of recognized character=$(cat line.txt|sed 's/ //g'|sed 's/!//g'|wc –c)

Character Recognition Ratio= crr = number of recognized character /w

aspect-ratio =ar=w/h

mean (vs)=1/n($\sum_{i=1}^{n}$ $y_{0_n} - y_{1_{n-1}}$)

ws=$(convert line.png txt:-|grep –c FFFFFF) : based on number of white pixels in line segment

ws_{mean}=(ws_n+ws_{n+1})/2

ratio(lm) =x_0/min(lm)

h_{mean} =1/n$\sum_{i=1}^{n}$ h_i

ratio(h)=h/mean(h)

In terms of line segmentation accuracy in mathematical PDF documents, the lines have been divided into four categories as follows:

- Fully-Segmented: the line is completely segmented with its subscript or superscript.

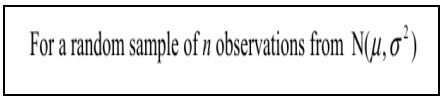


Figure 7. Full-Segmented

- Over-Segmented: the line is either split into more than one text-line, or partially detected. Figure 8 shows over -segmentation occurrence during ordinary line segmentation, which splits one line into two separate segments.

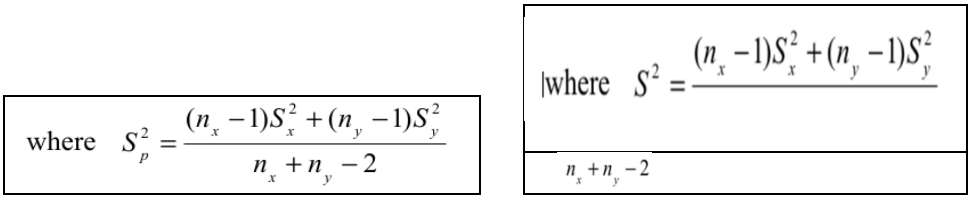


Figure 8. Over-Segmentation

- Under-Segmented: the line is merged with some other lines. Figure 9 illustrates a line segment which has been merged with the superscript of the previous line and subscript of the next line.

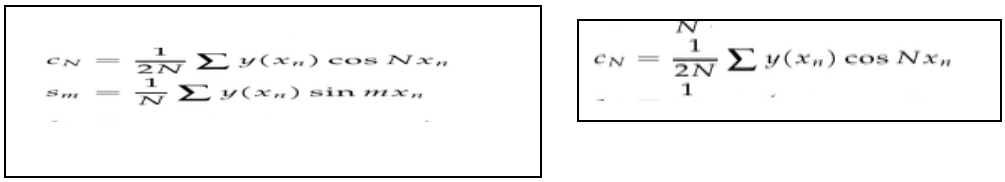


Figure 9. Under-Segmentation

- Broken symbol or character:

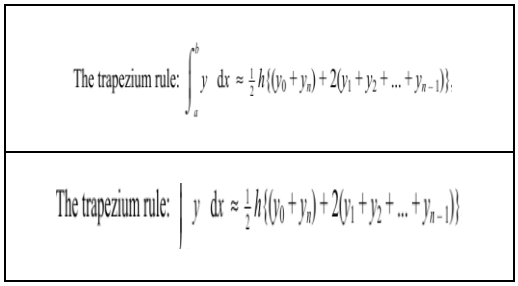


Figure 10. Broken Symbol

Using the currentline-segmentation method solves the problems of mathematical PDF. Additionally, hand written document pages have been successfully segmentedwith this method.

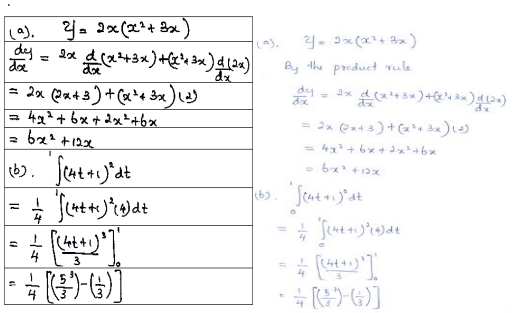


Figure 11. Line Segmentation Mathematic Hand Writing Document

7. Word Segmentation

Since line segments may contain embedded mathematics formulae or non-alphanumeric components which OCR cannot recognize them, applying line segments to word segmentation module supports to classify ordinary words and other components.

```
in="$1"
nogap=8
convert "$in"    -fuzz 1% -trim +repage  -threshold 60%  -rotate 90 bin.png
width=`convert bin.png  +gravity +repage -format '%[fx:u.h-1]' -identify original.mpc`
convertoriginal.mpc -crop 0x1 -format '%k %[pixel:s]' info:pixels
exec<pixels
x=0  ys=$x
segment=0
readys_xsys_color
while x=`expr $x + 1`; read xs color; do
if [ $ys_xs -eq 1 -a \( $xs -ne 1 -o "$ys_color" != "$color" \) ] ||   [ $ys_xs -ne 1 -a $xs -eq 1
]; then
if [ $ys_xs -ne 1  ]; then
    w=`expr $x - $ys`

xposition_start[$segment]=$ys
xposition_end[$segment]=$(($ys+$w))
```

Pixel information extraction

Read data pixel by pixel to find connected components properties until reaching gap

```
if [[ $segment -gt 1 ]];then
gap=$((${xposition_start[$segment]}-${xposition_end[$(($segment-1))]}))
else
gap=${xposition_end[$segment]}
fi
if [[ $gap -lt $nogap ]]; then
xposition_start[$segment]=${xposition_start[$(($segment-1))]}
fi
echo $segment ${xposition_start[$segment]} ${xposition_end[$segment ]} >>gap.dat
segment=`expr $segment + 1`
echo "component $segment:width $w"
fi
ys=$x ys_xs=$xsys_color="$color"

fi
done
catgap.dat|awk 'p{print $2,$3,$2-p}{p=$2}'|awk '$3!=0'|awk 'p{print p ,$2}{p=$2}'
>ggap.dat
crop=$(($(cat ggap.dat|wc -l)-1))
for (( j=0;j<=$crop;j++ ))
do
catggap.dat|awk 'NR=='$j'"|awk '{print "convert '$in' -crop ",$2-$1"x0+"$1"+0 ",
"word"'$j'".png"}'>>crop.sh
done
chmod +x crop.sh
./crop.sh
convert -size 0x0 canvas:white result.png
for (( j=1;j<=$crop;j++ ))
do
convert result.png word$j.png -background red -splice 1x0+0+0 +append result.png
done
```

Ignoring non interested gaps between characters

Save words bounding boxes information ina database to use for words classification based on feature extracted and accurate layout analysis

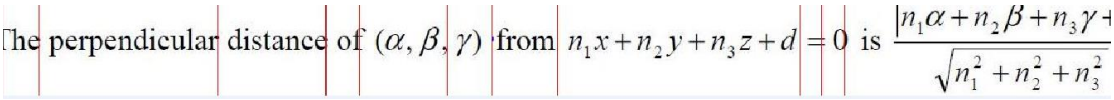


Figure 12. Word Segmentation a Line with Embedded Formulae

8. Vertical-Horizontal-Recursive-Segmentation

If a mathematical formulae is extracted by the word segmentation module, prior to sending this segment to the mathematical OCR for character recognition, it must be segmented into primitive components .Since mathematical formulae are mostly multi-dimensional and not linear, thus to recognize the order of primitive components, another segmentation method was used in this research.Vertical-Horizontal–recursive-segmentation covers all of the primitive components and saves their bounding boxes to a database. This database is used to define relationship between primitive components and finding their order.

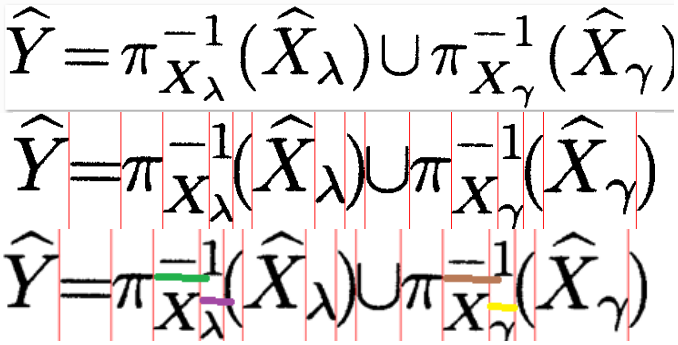


Figure 13. Vertical-Horizontal–Recursive-Segmentation

8. Conclusion

The research undertaken provides open source bash script codes for PDF document segmentation which easily can be run from command line in ordinary Linux machine or virtual machine. In order to design affordable, stand-aloneand simple to use software/hardware embedded system for reading electronic documents to vision impaired further development contains running these scripts in System on Chip (SoCs) platform such as ODROID-X2 with UbuntueLinaro operating system OS .

References

[1] S. O Brein, "Optical Character Recognition", Worcester Polytechnic Institute, **(2012).**
[2] T. M. Breuel and U. Kaiserslautern, "The hOCR Micro format for OCR Workflow and Results", Document Analysis and Recognition, 2007.ICDAR 2007.Ninth International Conference on, **(2007)**.
[3] T. M. Breuel, "The OCRopus open source OCR system", In Proc. SPIE Document Recognition and Retrieval XV, pages 0F1–0F15, San Jose, CA, USA, doi:10.1117/12.783598, **(2008)**
[4] H. S. Baird, "Anatomy of a versatile page reader", Proc. of IEEE, vol. 80, no. 7, **(1992),** pp. 1056-1065.
[5] R. M. Haralick, "Document image understanding: geometric and logical layout", Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94, 1994 IEEE Computer Society Conference on, **(1994).**
[6] G. Shapiro, G, Linda and G. C. Stockman, "Computer Vision", Prentice Hall, ISBN 0-13-030796-3, **(2002).**
[7] N. Pham, A. Morrison and J. Schwock, "Quantitative image analysis of immunohistochemical stains using a CMYK color model. DiagnPathol, vol. 2, **(2007),** pp. 8.
[8] E. R. Dougherty, "An Introduction to Morphological Image Processing , ISBN 0-8194-0845-X, **(1992)**.

