Ben Gurion University

BITeR
Bidirectional Image Text Reflow

Method and utility for re-factoring scanned documents for optimal display on multiple screen sizes

Avi Hayoun
Boaz Arad

Project Report

March 4rd, 2012

Abstract: In this project, we present an automated method for segmenting scanned documents into ordered components for optimal viewing on any screen size. Our method gracefully handles segmentation errors, and produces cross platform files viewable on nearly any mobile device.

Table of Contents

1. Motivation

Even today, while digital publishing has become widespread, a huge amount of the worlds information still exists in printed form. As part of their “Books” project, Google estimated that at least 129 Million distinct books exists today. “Google Books” is only one among several such endeavors to digitize the worlds written knowledge. Others include the “Million book project”, “Project Gutenberg” and the library of congresses “American Memory”.

Due to the current state of Optical Character Recognition (OCR) technology, automated digitization of printed documents is unreliable. And so, even today, we must frequently view scanned documents on computer systems. This is not a problem on desktop displays measuring anywhere from 19 to 24 inches, but viewing a scanned document on a 3-5 inch screen is unpleasant since it is rarely possible to view a complete line of text without reducing it to an unreadable size.

We aim to provide an OCR-independent method for comfortably viewing scanned documents on small screens while preserving the document structure, and without damaging important features such as diagrams, equations or images.

2. Problem domain and Assumptions

Our application should be able to handle most 'properly scanned' printed documents.
A 'properly scanned' document does not contain excessive noise, and has horizontal rows of text. Since the document was printed, we assume that the text lines are properly spaced, and that no characters share the same vertical coordinates as characters from adjacent text rows.

Also, we assume that text and non-textual objects (images, graphs etc.) do not occupy the same vertical space and that each input file contains a single column of images and text (i.e. a 'book' layout as opposed to a 'magazine' layout).

2.1. On differentiating text form non-textual elements

A few key observations regarding printed documents:

1. Text tends to be organized in discrete paragraphs with a fixed line height.
2. Due to curved nature of letters, the image gradient direction exhibits a very high variance in areas containing text.
3. Spacing between words is directly related to line height.
4. Paragraph margins will usually be smaller than image/graph margins.

We will use these properties to differentiate text from non-textual elements, and to identify individual words.

3. Method

Our algorithm attempts to segment the document image into sections containing either text or other elements (images, graph, equations, etc.). Text sections are further segmented into individual words, while other sections are trimmed of white space and rescaled if necessary. The original order of the segments is preserved and the document is rephrased into an HTML file containing multiple small images. The resulting HTML is easily viewable in most common mobile browsers and will scale to fit the screen width automatically.

3.1. Preprocessing

The input image is converted to black and white, noise is removed by deleting pixels which have only a small amount of connected components and the resulting image is cropped to remove empty margins. The resulting image will be used for most calculations, while a cropped version of the original image is preserved, and will be used for output.

3.2. Estimation of line height

Line Height is an important metric for identifying text segments, as well as segmenting text into individual words.
To estimate line height – we sum the value of image pixels horizontally, then determine the median value of the resulting vector. We then measure the length of all continuous sequences in the resulting vector with a value greater than the median.
For most inputs, the median of the resulting sequence length is a good estimation for the documents line height.
In images containing large amounts of white space, the above result may quickly approach zero, in such cases, and attempt is made to estimate line height using a smaller part of the input image. Smoothing the original image with a Gaussian, we search for the area of the image with the highest horizontal pixel density, then repeat our original process for a window half the size of the input image around the detected peak. This process is repeated until a satisfactory result is achieved.

3.3. Image Segmentation

The image must now be segmented into sections categorized as either textual or non-textual. We achieve this by relying on the variance of image gradient direction along the horizontal axis.
We have found that even graphs and diagrams exhibit low horizontal gradient direction variance when compared to text in the same document.
With this in mind – we compute the gradient direction for each image pixel, then find the variance along the horizontal axis.
The median of the resulting variance vector is calculated, and used to create a cut-off factor. Lines with a variance greater than the cutoff factor are considered text, while lines with variance below the cutoff factor are considered non-textual.
The resulting segmentation is further improved by filling gaps smaller than the computed line height, as it is unlikely that two textual segments are separated by an image segment smaller than the height of one line.

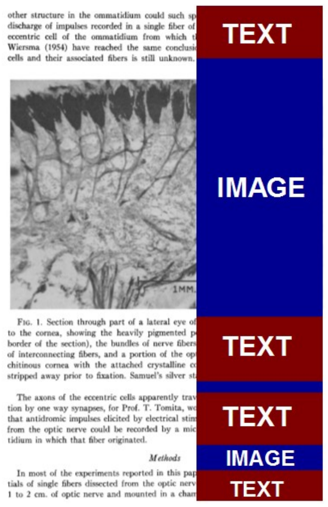


Illustration 1: Results of image segmentation and verification

3.4. Segment Verification and Grouping

Since our segmenter may still return false positives for textual areas (in fact, it is designed to do so, rather than mistakenly identify textual areas as non-textual) – we now verify that all textual segments

indeed include text. Segments with large margins, or segments that are not at least twice their local line height in size are reclassified as non-textual.
Reclassified segments are then grouped with their preceding and trailing image segments.

3.5. Word Identification

Textual segments are further segmented into individual words by first applying a Gaussian with a magnitude proportional to the segments detected line height (slightly modified to smooth horizontally more than it smooths vertically). The resulting image is converted to binary connected components are detected. The result is a good segmentation into individual words.

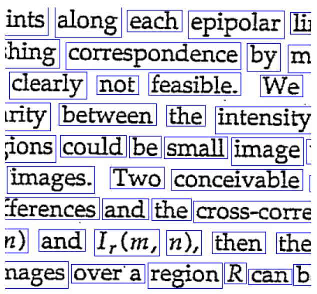

ints along each epipolar li
hing correspondence by m
clearly not feasible. We
rity between the intensity
ions could be small image
images. Two conceivable
ferences and the cross-corre
n) and $I_r(m, n)$, then the
nages over a region R can b

Illustration 2: Results of word identification

3.6. Text Baseline Detection

To maintain proper order between detected words, we must be able to differentiate between discrete lines of text. Using a similar methodology to that of line height estimation, we determine that text baselines exist wherever the image transitions from areas where the horizontal sum of pixel values is greater than the median, to areas where it is below the median.
Further accuracy is achieved by demanding that at least 'line-height' above median pixels are detected before reporting a transition.

3.7. Word order

Once both words and text baselines are detected, words are ordered first by the lowest baseline they appear above, then by their X co-ordinate. We can easily accommodate Right-to-Left text by selecting descending order for the X co-ordinates.

3.8. Word Alignment

Since each detected word is closely cropped, each words baseline will effectively become it's descender line (the lowest portion of the words lowest letter). In order for our segmented words to be properly aligned, we must detect each words true baseline, and correct its position.
This is achieved by computing the vertical gradient of the word image, summing the values along the X axis, and finding the local maxima in the lower portion of the image.

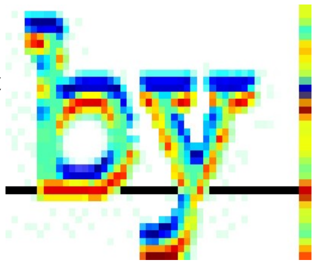


Illustration 3: Example of gradient based baseline detection

3.9. Final output

We have chosen to output our result as a HTML file, this is useful as a proof-of-concept format, since it is easily writable as well as easily viewable in both mobile and desktop browsers. Words and image segments are output as individual images, and embedded in the HTML file. Appropriate margins and relative positions are set using the data obtained by line height and word alignment detection.

Although line breaks in the original image are detected, they are not preserved in the output – since our aim is to include the largest possible amount of text in a limited display area.

4. Experimental Results

4.1. Methodology

In our test data set, comprised of various scanned documents, we measured the accuracy of segment type identification, and word segmentation. Total document-segmentation accuracy was measured using the following formula:

$$\frac{correctly\,identified}{total\;segments}=accuracy$$

Word-Segmentation accuracy was measured using the following formula:

$$\frac{(correctly\;segmented)+0.5\cdot(baseline/segmenting\;erros)-(dropped/misplaced\;words)^2}{total\,words}=accuracy$$

Baseline/segmentation errors are only counted as 'half' incorrect, as they do not usually reduce the result readability significantly, while the weight of dropped / misplaced words is grows exponentially to reflect the gravity of such errors.

Total accuracy was computed as:

$$0.75\cdot(word\;segmentation)+0.25\cdot(document\;segmentation)=total\,accuracy$$

4.2. Individual results

Excerpt from V.S. Nalwa's "A guided tour of computer vision" **, 1993 (target1.jpg)**
Document-Segmentation:

Correctly identified	Incorrectly Identified as textual	Incorrectly Identified as non-textual	Accuracy
6	0	1	85.7%

Word-Segmentation

Correctly segmented	Baseline/Segmenting errors	Dropped/Misplaced Words	Accuracy
386	3	0	99.6%

Second Excerpt from V.S. Nalwa's "A guided tour of computer vision" **, 1993 (target2.jpg)**
Document-Segmentation:

Correctly identified	Incorrectly Identified as textual	Incorrectly Identified as non-textual	Accuracy
2	1	0	66.6%

Word-Segmentation

Correctly segmented	Baseline/Segmenting errors	Dropped/Misplaced Words	Accuracy
207	0	0	100%

2010 English Journal article (target3.jpg)

Document-Segmentation:

Correctly identified	Incorrectly Identified as textual	Incorrectly Identified as non-textual	Accuracy
4	0	0	100%

Word-Segmentation

Correctly segmented	Baseline/Segmenting errors	Dropped/Misplaced Words	Accuracy
136	2	1	97.1%

1907 German journal article (german1.jpg)

Document-Segmentation:

Correctly identified	Incorrectly Identified as textual	Incorrectly Identified as non-textual	Accuracy
2	0	0	100%

Word-Segmentation

Correctly segmented	Baseline/Segmenting errors	Dropped/Misplaced Words	Accuracy
136	1	0	99.6%

Hebrew sample text (hebrew1.jpg)

Document-Segmentation:

Correctly identified	Incorrectly Identified as textual	Incorrectly Identified as non-textual	Accuracy
9	0	1	90%

Word-Segmentation

Correctly segmented	Baseline/Segmenting errors	Dropped/Misplaced Words	Accuracy
173	3	0	99.1%

Hebrew sample text 2 (hebrew2.jpg)

Document-Segmentation:

Correctly identified	Incorrectly Identified as textual	Incorrectly Identified as non-textual	Accuracy
2	0	1	66.6%

Word-Segmentation

Correctly segmented	Baseline/Segmenting errors	Dropped/Misplaced Words	Accuracy
175	11	0	91.1%

1953 Article by Kuffler (kuffler.jpg)

Document-Segmentation:

Correctly identified	Incorrectly Identified as textual	Incorrectly Identified as non-textual	Accuracy
6	0	2	75%

Word-Segmentation

Correctly segmented	Baseline/Segmenting errors	Dropped/Misplaced Words	Accuracy
183	0	0	100%

1925, Gestalt theory by Max Wertheimer (target4.jpg)
Document-Segmentation:

Correctly identified	Incorrectly Identified as textual	Incorrectly Identified as non-textual	Accuracy
1	0	2	33.3%

Word-Segmentation

Correctly segmented	Baseline/Segmenting errors	Dropped/Misplaced Words	Accuracy
416	1	13	57.3%

1983 Excerpt from Human and Machine Vision, Vitkin & Tenenbaum (vitkin1.jpg)
Document-Segmentation:

Correctly identified	Incorrectly Identified as textual	Incorrectly Identified as non-textual	Accuracy
3	0	0	100%

Word-Segmentation

Correctly segmented	Baseline/Segmenting errors	Dropped/Misplaced Words	Accuracy
86	4	0	97.7%

4.3. Summary:

Test Set	**Accuracy**
Exert from V.S. Nalwa's "A guided tour of computer vision" , 1993 (1)	96.1%
Exert from V.S. Nalwa's "A guided tour of computer vision" , 1993 (2)	91.6%
2010 English Journal article	97.8%
1907 German journal article	99.7%
Hebrew sample text	96.8%
Hebrew sample text 2	84.9%
1953 Article by Kuffler	93.7%
1925, Gestalt theory by Max Wertheimer.	57.3%
1983 Excerpt from Human and Machine Vision, Vitkin & Tenenbaum	98.2%

It is important to note that our algorithm usually fails in a graceful fashion. The most common mistakes are either identification of textual elements as non-textual, or incorrect word baseline detection. In the former case – the only adverse effect is reduced reflowability, while in the latter case, the resulting offset is rarely enough to prevent the word from being readable.

Major failures mostly occur when either the text contrast is not high enough to be properly preprocessed – resulting in dropped words, or when the word detection filter is too aggressive – resulting in several words being detected as a single word.

5. Future work

At the moment, all detection parameters are based on the detected line height of the processed segment. Adaptively modifying parameters such as the size of the Gaussian used in word-detection is expected to improve result accuracy.

While non-textual segments mistakenly labeled as text are usually accurately detected, a similar detection method should be implemented for textual segments mistakenly labeled as non-textual. Local gradient directional variance is probably a good metric to use for this purpose.

In order to provide a more robust product, support for multi-column and text-beside-images document layouts can be added by detecting white (or nearly white) horizontal and vertical lines, recursively bisecting the document and supplying our algorithm with the resulting ordered segments.

Finally, detection for special text formats such as lists, where preserving line breaks is desirable, would significantly improve the user experience.

6. References

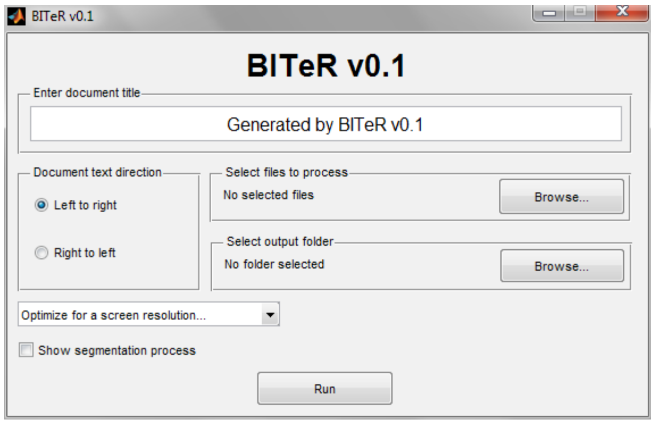
1. Y. Pan, X. Hou, and C. Liu. **A robust system to detect and localize texts in natural scene images**. In International Workshop on Document Analysis Systems, 2008.
2. Boontee Kruatrachue, Narongchai Moongfangklang, Kritawan Siriboon. **Fast Document Segmentation Using Contour and X-Y Cut Technique**. In Proceedings of WEC (5)'2005. pp.27~29
3. "Books of the world". Google. August 5, 2010.
4. Ben Shahar, Ohad, Lecture notes from Introduction to Computational and Biological Vision course, 2011.

7. Appendix: Using the BITeR utility

The BITeR Utillity has been packaged as a Matlab Executable Package and you should be able to run it by simply double clicking BITeR.exe on a 64-bit windows based system.

If you have trouble loading the executable, you may open 'runBiter.m' from the provided source package, and run it within Matlab (be sure to extract all files, as they are required by runBiter.m).

1. Select an input image file (JPEG and PNG are supported), you may also select multiple files at once.
2. Select an output directory where the resulting HTML and image files will be stored.
3. Select the desired text direction (i.e. Right-to-Left for Hebrew, Left-to-Right for English).
4. Enter a document title – this will be displayed as the HTML files title.
5. Optionally select a screen size for optimization, BITeR will rescale all images to be compatible with the selected display size.
6. If 'Show segmentation process' is selected the process will be slowed down, and verbose information will be displayed while BITeR is processing your file.
7. Click run, once the process is complete, your default browser should display the processed document.


BITeR v0.1

Enter document title: Generated by BITeR v0.1

Document text direction: [✓ Left to right] Right to left

Select files to process: No selected files

Select output folder: No folder selected

Optimize for a screen resolution...

Show segmentation process

Illustration 4: Screen-shot of the BITeR utility