

IMPACT – Technical deliverable documentation

D-TR2 - Segmentation and classification toolkit

1. Partners

NCSR (National Centre for Scientific Research "Demokritos"),
 USAL (University of Salford),
 ABY (ABBYY Production LLC).

2. Deliverable

D-TR2: Segmentation and Classification Toolkit

Table of contents

1. Partners	2
2. Deliverable	2
3. Background	3
3.1 Block Segmentation and Classification.....	3
3.2 Text Line and Word Segmentation.....	3
3.3 Character Segmentation.....	4
4. Outline of Functionality	6
4.1 Block Segmentation.....	6
4.2 Text Line and Word Segmentation.....	10
4.3 Character Segmentation.....	18
5. Evaluation	24
5.1 Block Segmentation.....	24
5.2 Text Line and Word Segmentation.....	26
5.3 Character Segmentation.....	28
6. License and IPR Protection	30
6.1 Block Segmentation.....	30
6.2 Text Line and Word Segmentation.....	30
6.3 Character Segmentation.....	30
References	31

3. Background

Segmentation is a major function in an OCR system. During this step, the main document components (text / graphic areas, text lines, words and characters or glyphs) are automatically extracted. In the literature, the problem of segmenting archived historical machine-printed documents is tackled by the use of techniques which are mainly designed for contemporary documents.

As a result, several problems inherent in historical documents such as general low quality, complex, dense and irregular layouts, artefacts not completely corrected during pre-processing (noise between characters, ink diffusion and local skew) seriously affect the segmentation and, consequently, the recognition accuracy of OCR. Furthermore, publication-specific rules are usually used for segmenting historical machine-printed documents. The above limitations are restrictive for mass full-text digitisation and necessitate the development of new approaches.

IMPACT introduces novel hierarchical segmentation models that allow the discrete problems of text block, text line, word and character segmentation to be addressed separately while at the same time allowing for interplay between all levels.

In this section, we present the current technology that is considered as background for all individual segmentation steps.

3.1 Block Segmentation and Classification

The new IMPACT Block segmentation and classification toolkit is released on the basis of the ABBYY FineReader Engine 10 for Windows. Since ABBYY uses the same technology base for many different products, the new algorithms developed for the IMPACT project are already available in Recognition Server 3.0. Some, but not all of them will soon be available in a desktop product, FineReader 12.

3.2 Text Line and Word Segmentation

Segmentation in the context of document analysis is the partitioning of images into meaningful regions in order to relay them to the next processing step depending on their particular type. In a typical text recognition scenario, text line segmentation followed by word segmentation are the next steps after text regions have been identified in the original. Segmented words are subsequently passed on to character segmentation which provides the input for the actual character classifier which is part of any OCR (Optical Character Recognition) software.

As for recognition-free document image segmentation (i.e. based only on geometrical features) there are two major approaches:

- Based on dividing image regions according to gaps (space)
- Based on merging image parts according to connected components / neighbouring objects

Both approaches have been implemented and further developed for the initial text line and word segmentation toolkit in order to study and improve segmentation results of this stage.

3.3 Character Segmentation

In this work, we focus on the task of isolating characters in historical machine printed documents. We consider as input isolated words and our aim is to detect character areas (see Fig.1).

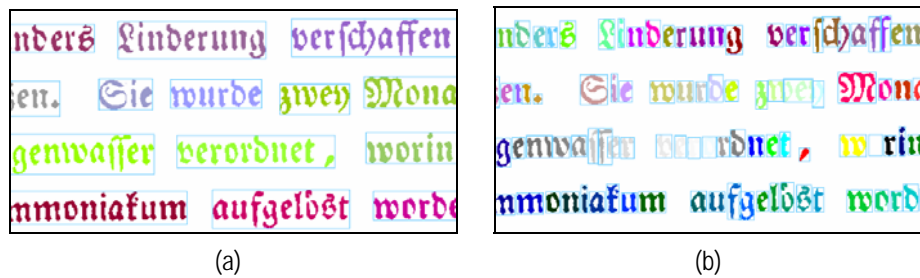


Figure 1. Character segmentation example. (a) isolated words used as input; (b) segmented characters

Existing methodologies in the literature

Antonacopoulos and Karatzas [1] use the horizontal projection profile (summing the foreground pixels in vertical direction) of each word segment for character segmentation in historical machine-printed documents (see Fig.2). The first split point position is predicted based on the expected character box width and refined according to the location and strength of the projection minima. The next splitting positions are derived in the same manner using the information from the location of the previous separator. This approach cannot handle the case of overlapping characters that is characters that their bounding boxes overlap horizontally.

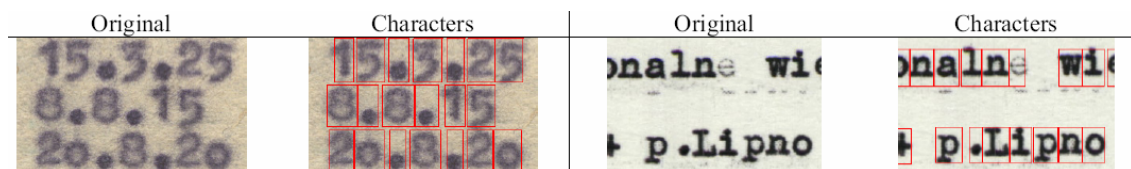


Figure 2. Character segmentation using projection profiles [1].

In the work of Nomura et al. [2] character segmentation and feature extraction is applied on degraded images of license plates. Horizontal projection profile is used to merge character fragments and final character segmentation is guided by morphological operations. Horizontal projection profile is also used in the work of Jia et el [3] for character segmentation of degraded license plates (see Fig.3).



Figure 3. Character segmentation of degraded license plates [3].

A well known methodology for document image segmentation is the Run Length Smoothing Algorithm (RLSA) [4]. RLSA examines the white runs (successive white pixels) existing in the horizontal and vertical directions. For each direction, white runs with length less than a threshold are eliminated. For the case of character segmentation, RLSA is applied mainly in vertical direction. The application of RLSA results in a binary image where pixels of the same character become connected to a single connected component (Fig.4(a)). In the sequel, a connected component analysis is applied in order to extract the final character segmentation result (Fig.4(b)).



Figure 4. Example of applying RLSA. (a) result after smoothing; (b) character segmentation result.

In Liang et al. [5] work, character segmentation is applied on modern machine-printed documents. Here, the touching characters problem is confronted using discrimination functions based on pixel and component profile projections. Also, contextual information and spell checking are used to improve recognition accuracy. Kavallieratou et al. [6] proposed a technique for handwritten character segmentation. A pre-segmentation stage locates all possible splitting points by detecting local minima in the horizontal projection profile. Final splitting points are determined by a transformation-based learning procedure (see Fig. 5).

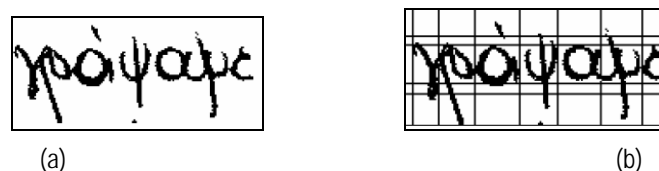


Figure 5. Handwritten character segmentation using [6]. (a) word image; (b) character segmentation result.

A cursive script character segmentation method is proposed in [7] in which knowledge of the structure of English letters and the structural characteristics between background and characters is investigated. Yanikoglu and Sandon [8] proposed a character segmentation technique for cursive handwritten text. Using linear programming, the weights of the extracted features are determined and a segmentation cost function for all possible character splitting points is formed. The final splitting points result after the evaluation of each cost function. The concept of water reservoir is introduced in the work of Pal et al. [9] where touching numeral segmentation is performed (see Fig. 6). A reservoir is the region that is formed around the touching point of two numerals and the analysis of its boundary determines the cutting point.

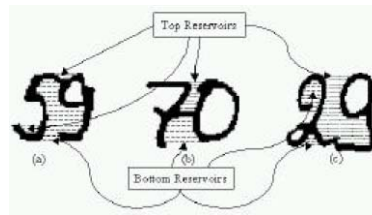


Figure 6. Water reservoir concept for touching numeral segmentation.

Finally, Chang and Chen [10] use the convex-hull techniques and typographical structure to segment touching characters without the guidance of recognition. Features based upon a convex hull are insensitive to character fonts and sizes, the touching-character problem of various fonts and sizes can be handled even for heavily touching characters or italic-type overlapping characters without slant correction.

Commercial products

Character segmentation is part of all commercial OCR software (e.g. FineReader Engine [14], OCRXpress [15]) or open source OCR (OCRopus library [16]).

4. Outline of Functionality

4.1 Block Segmentation

In comparison to the state-of-art (release 9 of ABBYY FineReader Engine) the Block Segmentation and Classification Toolkit contains the following improvements:

1. Better picture detection. To improve the picture detection algorithm, a new set of various criteria was developed. The criteria are based on different features, such as gradients, color variance and many others.
2. Better separators extraction. By improvement in the separators extraction it became possible to detect tables, columns and running titles more reliably.
3. Segmentation has better overall quality. Results were achieved by examining the new datasets made available by IMPACT and by correcting some models which relate the positions of text and pictures.
4. The new models for better segmentation of historic newspapers.

Here are some details of the toolkit.

Picture detection

The task of finding a picture on a page was split into two parts. First, we detect pictures among noise or garbage. Second, when the picture is detected, we try to detect its borders properly. To improve the picture detection we developed the following criteria:

- Number of connected regions in a rectangle
- Number of white connected regions
- Average length of a white/black stroke
- Some criteria from a greyscale image

Some new picture models were added as well. Here are some of them:

- Picture in a frame. Sometimes a picture is bordered by a "box" of black or gradient separators. Often such a box has low contrast. The detection of such images was improved.
- Picture in a shadow. Pictures are now detected better if the shadow and picture overlap. Shadows and pictures differ when using criteria such as the number of gray tints or different colors. Next, shadows with straight borders are given higher weights. And at last, when we cut shadows, we check some of them for a "pictureness". Usually a picture is a rather big object with not such a big width/height ratio which contains high- and low-contrast regions.
- Picture detection in a column. A special algorithm to detect pictures in a multi-column layout was developed. Such pictures are formed by objects which remain after the text-line detection. All such objects are collected, then by changing the binarisation threshold we enlarge the collected regions until there is a small space between them and the text.
- Big pictures on a page. There are cases when a big picture occupies most of a page. A special model to treat such situations was created.

Separators extraction

First, we improved the treatment of long white separators on newspapers. Now instead of many short unconnected separators, a long one is detected.

Second, a large amount of work was done on separator roles. Now all detected separators are related to page elements, so the toolkit can detect the page structure much better. There are the following relations:

- Separators around paragraphs
- Separators, which separate footnotes and body text
- Separators between running titles and body text
- Separators between text columns
- Separators between text sections

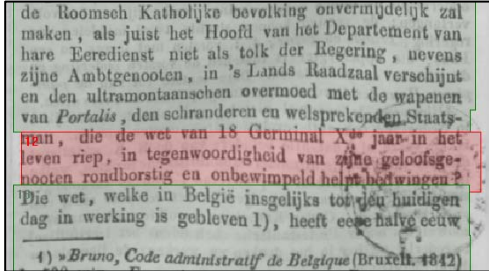
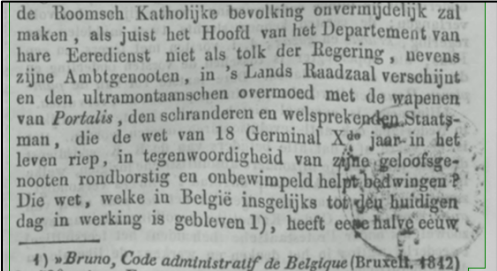




Better segmentation of historical newspapers

The requirement to better segment historic newspaper was not evident when IMPACT started. It appeared after a discussion with the libraries about the issues that affect their performance during mass digitisation tasks.

To address this challenge, a special segmentation model was developed to better segment historic newspapers. This model can handle the typical layouts of the historic newspapers better: they usually contain many straight columns of text and large headers. Often, the lines of the text to the right or to the left are distorted because newspapers are scanned without unbinding the whole set of sheets at the library.

The model that was developed is included in the segmentation toolkit, and a set of tests was performed to evaluate the performance of the algorithm.

Here are some examples of the improved newspaper segmentation:

FineReader Engine 9	New segmentation toolkit / FineReader Engine 10
<p>Part of the text was misdetected as an image</p> 	
<p>A column of text was incorrectly split into many regions</p> 	
<p>Text was lost</p> 	

Information on how to use the toolkit

The Segmentation Toolkit was released as part of ABBYY FineReader Engine 10 SDK. This SDK is a tool that is already used for mass digitisation. It is thoroughly documented and contains many samples that demonstrate how to use this toolkits in different platforms and different programming languages.

Here is a brief extract taken from the sample code that goes along with the toolkit:

```
// Create document from image file
printf( "Loading image...\n" );
CWSTR imagePath = L"SampleImages\\sample.tif";
CSPtr<IFRDDocument> frDocument = 0;
CheckResult( Engine->CreateFRDocumentFromImage( imagePath, 0,
frDocument.GetBuffer() ) );
//Recognize document

printf( "Recognizing...\n" );
CSPtr<IPageProcessingParams> ppp;
CheckResult( Engine->CreatePageProcessingParams( ppp.GetBuffer()
) );
CSPtr<IPageAnalysisParams> pap;
CheckResult( ppp->get_PageAnalysisParams( pap.GetBuffer() ) );
CheckResult( pap->put_DetectOrientation( VARIANT_FALSE ) );
CheckResult( frDocument->Analyze( ppp ) );

// Iterate over segmentation results

CSPtr<IFRPages> frPages;
CheckResult( frDocument->get_Pages( frPages.GetBuffer() ) );
long nPages;
CheckResult( frPages->get_Count( &nPages ) );
for( int iPage = 0; iPage < nPages; iPage++ ) {

    CSPtr<IFRPage> frPage;
    CheckResult( frPages->get_Element( iPage,
frPage.GetBuffer() ) );
    CSPtr< ILayout > layout;
    CheckResult( frPage->get_Layout( layout.GetBuffer() ) );
    CSPtr< ILayoutBlocks > blocks;
    CheckResult( layout->get_Blocks( blocks.GetBuffer() ) );
    long nBlocks;
```



```

        CheckResult( blocks->get_Count( &nBlocks ) );
        for( int iBlock = 0; iBlock < nBlocks; iBlock++ ) {
            CSafePtr< IBlock > block;
            CheckResult( blocks->get_Element( iBlock,
block.GetBuffer() ) );
            CSafePtr< IRegion > region;
            CheckResult( block->get_Region( region.GetBuffer()
));

            long nRects;
            CheckResult( region->get_Count( &nRects ) );
            printf( "Block %d, %d rects\n", iBlock, nRects );

            for( int iRect = 0; iRect < nRects; iRect++ ) {
                long left, top, right, bottom;
                CheckResult( region->get_Left( iRect, &left )
);

                CheckResult( region->get_Top( iRect, &top ) );

                CheckResult( region->get_Right( iRect, &right
) );

                CheckResult( region->get_Bottom( iRect,
&bottom ) );

                printf( "  Rect %d, %d - %d, %d\n", left, top,
right, bottom );
            }
        }
    }
}

```

4.2 Text Line and Word Segmentation

The final segmentation toolkit introduces now improved text line and word segmentation algorithms resulting from the research and optimisation work done in IMPACT. For text line segmentation a hybrid method has been developed which combines elements from connected component merging and projection profile analysis in a novel way. As for word segmentation a pixel density method has been developed which is based on projection profile analysis extended by a novel adaptive whitespace detection algorithm.

Projection Profile Based Text Line Segmentation

For a given text region a horizontal-projection profile is calculated, which is then used to find delimiting whitespaces (valleys within the profile). Finally, all connected components are assigned to the text line (delimited by whitespaces) they most overlap. Based on the grouped connected components, the text line outlines are calculated by an approach using smearing and shape tracing.

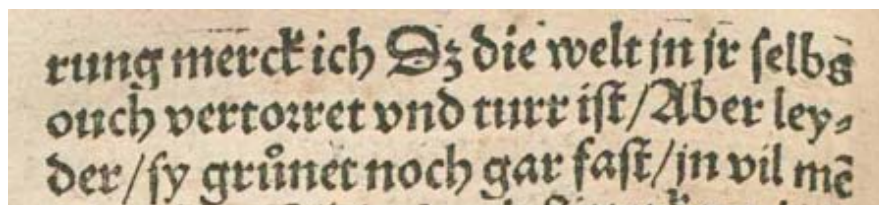
Connected Component Based Text Line Segmentation

For a given text region the connected components are retrieved and for each component the four nearest neighbours are detected. Based on angles between neighbouring components, the components are then grouped to lines. The resulting outlines are again calculated by an approach using smearing and shape tracing.

IMPACT Hybrid Text Line Segmentation

The hybrid approach combines grouping of connected components and local projection profiles to improve the results in comparison to using each of the two techniques on their own. That way common problems related especially to historical documents can be solved.

Warped images for instance cause considerable difficulties for projection profile based approaches, because the resulting profile is indistinct and text lines cannot be determined reliably:



Text descenders which touch glyphs on the next text line are problematic for connected component based approaches as they lead to components reaching across two or more lines:

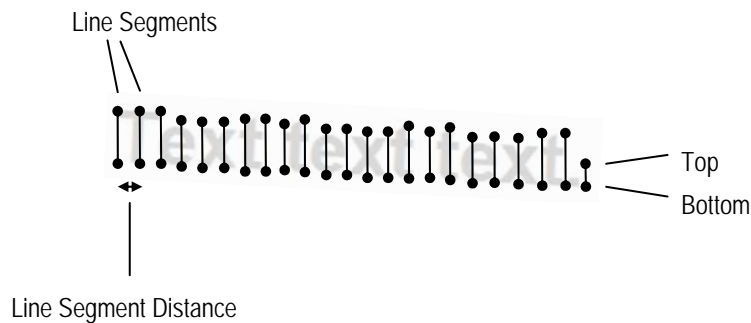


- **Pre-processing**

Firstly the connected components of the current region are detected. Following, noise is filtered from the components. For this purpose, noise is defined as connected components with an area less or equal three pixels and with no non-noise component as one of its four nearest neighbours. This way filtering of small but meaningful objects like i-dots etc can be avoided.

o **Representation of Text Lines**

The algorithm uses a text line data structure called Flex-Text-Line. It consists of horizontally aligned line segments, which have a top and a bottom point. Segments have a predefined distance to each other. The advantage in comparison to just using a bounding box is that also skewed and warped text lines can be represented precisely.



In addition, a flex text line has a list of assigned connected components and each line segment has a flag pointing out if it is a whitespace or not (i.e. text).

o **Segmentation Process**

The segmentation consists of three steps:

- Grouping the connected components of the region to lines
- Find and split under-segmented lines using local projection profiles
- Merge lines containing only dots, commas etc. to their nearest neighbour

o **Multi Threading**

The segmentation process uses a two stage multi threading model. For both stages a limited number of thread slots is available. This number can be specified using the parameter max_threads. If set to zero, the number of threads will automatically be set to the number of available processor cores.

The first thread stage is on layout region level. Since each region is processed individually, they can be segmented within threads.

The second thread stage applies for the step where connected components are grouped to lines. The set of components of large regions (more than 500 components) is split into two subsets. Each subset is then processed in its own thread. Once both threads are finished, the results are further processed to the final lines.

o **Parameters**

There are several parameters to change the behaviour of the segmenter. See the following table for an overview:

Parameter	Description	Reasonable Range	Default Value
immediate merge threshold	Threshold in the grouping stage for the match score of two flex text lines. Lines will immediately be merged if their match score is higher than this threshold.	min merge threshold .. 1.0	0.4

Parameter	Description	Reasonable Range	Default Value
min merge threshold	Threshold in the grouping stage for the match score of two flex text lines. Lines will only be merged if their match score is higher than this threshold.	0.1 .. immediate merge threshold	0.35
x distance score coeff	Coefficient for the match score formula. Influences the maximal allowed horizontal distance of two flex line segments.	0.0 .. 10.0	2.7
min component count for split	Threshold for the step of splitting under-segmented flex text lines. Only lines with more components than the threshold will be considered for a split.	0 .. 20	5
under-segmentation coeff	Coefficient for the step of splitting under-segmented flex text lines. This coefficient is used to calculate a threshold in order to decide whether a flex text line is under-segmented or not.	0.1 .. 1.0	0.6
split component coeff	Coefficient for the step of splitting under-segmented flex text lines. This coefficient is used to calculate a threshold to decide whether a component is large and therefore considered to be split.	1.1 .. 3.0	1.7
projection width coeff	Coefficient for the local projection profile when splitting large components. This coefficient influences how broad the projection area around the component is.	1.0 .. 20.0	9
projection whitespace coeff	Coefficient for the local projection profile when splitting large components. This coefficient is used to classify a single projection line as whitespace or not.	0.1 .. projection text coeff	0.4
projection text coeff	Coefficient for the local projection profile when splitting large components. This coefficient is used to classify a single projection line as text area or not.	projection whitespace coeff .. 1.0	0.7
projection min whitespace height coeff	Coefficient for the local projection profile when splitting large components. This coefficient is used to calculate a whitespace height threshold. Only consecutive projection profile lines classified as whitespace with a total height greater or equal the threshold is considered to be line separating whitespace.	0.1 .. 1.0	0.2
split iterations	Number of iterations for the complete split step. When a flex text line could not be split, the split process will be repeated with refined parameters.	1 .. 3	2
small component area threshold	Threshold for the first part of the step of merging loose elements to their nearest neighbour. Only lines with an average component size less than the threshold are considered for a merge.	0.0 .. 20.0	7.0
line segment x dist influence	Coefficient to set the influence of the horizontal distance of line segments for the computation of the minimal distance between two flex text lines.	0.5 .. 1.5	0.85

Parameter	Description	Reasonable Range	Default Value
dot height coeff	Coefficient for the second part of the step of merging loose elements to their nearest neighbour. This coefficient is used to classify a line as line containing dots, commas, ... by comparing the line height to the average component height of the current region.	0.1 .. 0.9	0.5
max dot dist coeff	Coefficient for the second part of the step of merging loose elements to their nearest neighbour. This coefficient is used to calculate a threshold for a maximal distance of a line containing dots, commas, ... to their nearest neighbour to merge them.	0.0 .. 2.0	1.0
dot line height diff coeff	Coefficient for the second part of the step of merging loose elements to their nearest neighbour. This coefficient is used to calculate a threshold for a maximal difference of the heights of a line containing dots, commas, ... and its nearest neighbour.	0.1 .. 1.0	0.8
comma min merge threshold fact	Factor for the second part of the step of merging loose elements to their nearest neighbour. This factor is used to adapt the 'min merge threshold' for calculating the match score of dots, commas, ... at the end of a text line.	0.5 .. 1.0	0.75

o **Creating the Final Text Lines**

The final text line outlines are generated using a smearing approach. In a loop it is tried to connect all components of a flex text line to one big component by applying horizontal and vertical smearing with increasing smearing distances. If that is not successful, an adaptive diagonal smearing is used to connect the remaining components. Then the contour of the resulting connected component is traced and the text line polygon is created.

Projection Profile Based Word Segmentation

Starting with a given text line, a vertical projection profile is calculated, which is then used to find delimiting whitespaces (valleys within the profile). Finally, all connected components are assigned to the word (delimited by whitespaces) they overlap most with. Based on the grouped connected components, the word outlines are calculated by the aforementioned approach using smearing and shape tracing.

Connected Component Based Word Segmentation

For a given text line the connected components are retrieved and sorted by x-position. An inter-component-distance histogram is used to group the components to words. The resulting outlines are then calculated using the standard smearing and shape tracing approach.

IMPACT Pixel Density Based Adaptive Word Segmentation

Before the actual segmentation is performed, the connected components of the current line are determined.

The segmentation process consists of three steps:

- Calculation of the vertical projection profile
- Finding delimiting whitespaces
- Grouping of connected components into words using the delimiting whitespaces

○ **Parameters**

There are several parameters to change the behaviour of the segmenter. See the following table for an overview:

Parameter	Description	Reasonable Range	Default Value
whitespace coeff	Used to adjust the threshold for whitespace detection within the projection profile. Lower values will result in more delimiting whitespaces.	0.1 .. 0.5	0.2
gap width coeff	Used to adjust the threshold to differentiate between inter glyph gaps and inter word gaps. Low values will cause over-segmentation whereas high values will cause under-segmentation.	1.5 .. 3.5	2.0
min gap width coeff	Used to calculate a minimal inter word gap width using the average line height. Use 0 to not use the minimum. High values can cause under-segmentation.	0.0 .. 0.3	0.2

○ **Creating the Final Words**

The final word outlines are generated using the already mentioned smearing approach. In a loop it is tried to connect all components of a word to one big component by applying horizontal and vertical smearing with increasing smearing distances. If that is not successful, an adaptive diagonal smearing is used to connect the remaining components. Then the contour of the resulting connected component is traced and the word polygon is created.

The deliverable is in the form of a zip file (tlwSegmenter-1.1.zip) containing the command line executable **segmenter-1-1-395.exe** for Windows operating systems (including required third party dlls) plus two examples (images and PAGE files) and the batch file **examples.bat** to run the tool in all modes (text line segmentation: component based method, profile based method, IMPACT hybrid approach; word segmentation: component-based method, profile-based method, IMPACT pixel density approach) on the examples provided.

The tool can be obtained from the PRImA Research Group (www.primaresearch.org).

For ease of use, text line segmentation and word segmentation have been compiled into one single command line tool which can be controlled by a set of parameters. The tool requires as input a bitonal (black-and-white) TIFF image as well as a PAGE (<http://schema.primaresearch.org/PAGE/qts/pagecontent/2010-03-19/>) file describing the outline of text regions to run text line segmentation on (text regions plus text lines to run word segmentation on). Output is again a PAGE file, enhanced with text line descriptions embedded in the text regions (word descriptions within text lines).

Figure 7 shows input (manually created ground truth) and output of the text line segmentation tool for image 5177 (IMPACT dataset ID) displayed in Aletheia (ground-truthing tool supporting PAGE). Corresponding screenshots for input and output of word segmentation can be found in Figure 8.

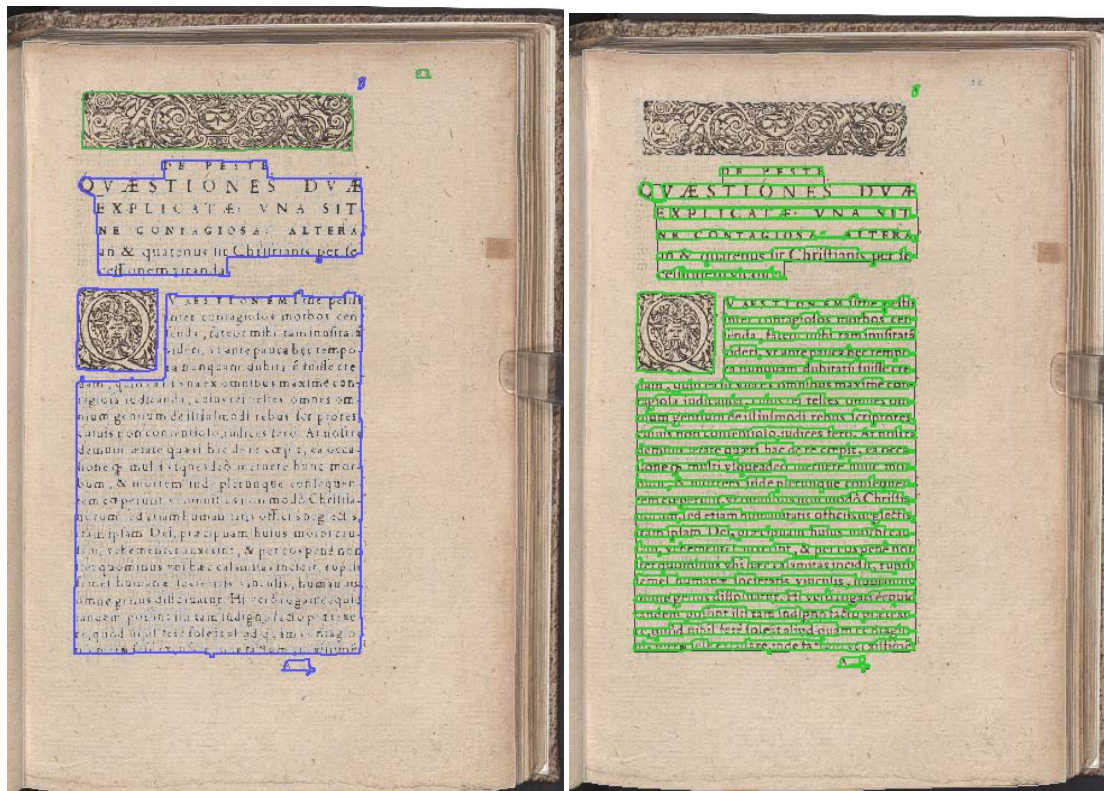


Figure 7. left – blue regions represent text regions which serve as input for the text line segmenter; right – output of the tool: light green regions represent the segmented text lines.

It has to be noted that the overall quality of text line and word segmentation depends on the accuracy of the respective input (i.e. text regions as input for text line segmentation and text lines as input for word segmentation).

The usage of the segmenter executable is as follows:

```
segmenter-1-1-395 <segm> <method> <binimg> <inputxml> <outxml> [<opts>] [<paramfile>]
```

Where:

<segm> one of:

line - text line segmentation

word - word segmentation

<method> one of:

component - A technique based on connected component aggregation (text line and word detection)

profile - A profile-based technique which uses profile minima & maxima to detect text lines / words

hybrid – A hybrid method based on connected component aggregation and projection profile analysis (text line detection only)

density – A projection profile based method with adaptive thresholds (word detection only)

- < **binimg** > Filename of the 1-bit TIFF image to be segmented
- < **inputxml** > Filename of an XML file containing a region-level or text line segmentation layout
- < **outxml** > Filename to which an XML file containing the resulting segmentation should be written
- < **opts** > (optional):
 - r** - Removes any existing text lines or words from the document before segmenting (this can e.g. be used to run the tool directly on ground truth files already containing all segmentation levels or to overwrite any other existing segmentation results in a PAGE instance)
 - p** - For parameter file usage (hybrid/line or density/word only).
- < **paramfile** > Full path to *.ini file containing a parameter set for the hybrid line segmenter or the density word segmenter. (optional)

Example parameters for hybrid text line detection:

```
[main]
ImmediateMergeThreshold=0.4
MinMergeThreshold=0.3
XDistanceScoreCoeff=2.5
UndersegmentationCoeff=0.5
SplitComponentCoeff=1.7
ProjectionWidthCoeff=9
ProjectionWhitespaceCoeff=0.4
ProjectionTextCoeff=0.7
ProjectionMinWhitespaceHeightCoeff=0.2
SplitIterations=2
MinComponentCountForSplit=5
SmallComponentAreaThreshold=7
DotHeightCoeff=0.6
MaxDotDistCoeff=0.9
DotLineHeightDiffCoeff=0.7
CommaMinMergeThresholdFact=0.75
LineSegmentXDistInfluence=0.75
MaxThreads=3
```

Example parameters for density word detection:

```
[main]
WhiteSpaceCoeff=0.2
GapWidthCoeff=2.0
MinGapWidthCoeff=0.2
```

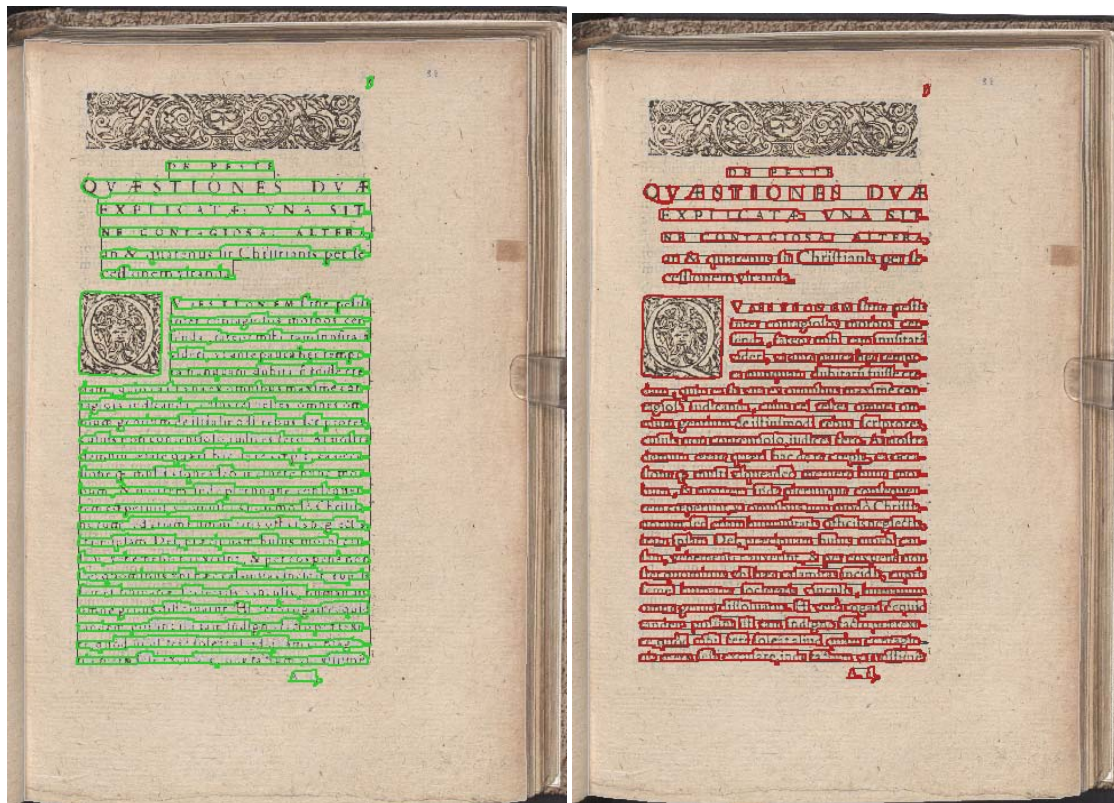


Figure 8. left – input for word segmentation: green regions represent text lines; right – output of the tool: red regions represent segmented words.

4.3 Character Segmentation

The developed character segmentation algorithm is based on skeleton segmentation paths which are used to isolate possible connected characters. The basic idea is that we can find possible segmentation paths linking the feature points on the skeleton of the word and its background. Similar approach has been proposed by Chen et al [11] for the segmentation of touching numerals.

We calculate the connected components (CCs) of a word and apply the following steps in all the CCs that have their height to width ratio less or equal to 0.5, in order to separate them into letters (Figs. 9(a) - 9(b)).

Step 1: We calculate the skeleton of the CC and its background (Fig. 9(c)).

Step 2: We classify the skeleton in four different segments as follow (Fig 9(d)):

- Top-segment: The segment generated from the upper part of the background region.
- Bottom-segment: The segment generated from the lower part of the background region.
- Stroke-segment: The segment generated from the black pixels of the CC.
- Hole-segment: The segment generated from the hole-region of the background.

Step 3: We locate the feature points of the skeleton (Fig. 9(e)). The different kinds of feature points are defined as follows:

- Fork point: The point on a segment which has more than two connected branches.
- End point: The point on a segment that has only one neighbor pixel.
- Corner-point: The point on a segment where the curvature of the segment changes sharply.

Step 4: In this step we construct all the possible segmentation paths (Fig 9(f)). We simultaneously apply two different searches, downward search and upward search. In downward search, we construct all the segmentations paths which start from the feature points on the top-segment. Each segmentation path should start from a feature point on the top-segment, pass through one or two feature points on the stroke-segment and end at a feature point on the bottom-segment. The distance between two feature points (top-stroke, stroke-bottom or stroke-stroke) must be less than 0. Therefore, if no one feature point on the stroke-segment matches a feature point on the top-segment, a vertical 8xaverage character height path is constructed starting from this feature point on the top-segment until it touches the bottom-segment. A similar process is applied to upward search in order to construct possible segmentation paths from bottom-segment to top-segment.

Step 5: After locating all the possible segmentation paths we select the best segmentation paths (Fig. 9(g)). In order to achieve this, starting from the beginning of component or from the last segmentation path that was selected, we take into consideration only segmentation paths that result to characters with width in the limit of [MinCharWidth, MaxCharWidth]. Among these, the best segmentation path is selected as the one that minimizes the following criteria:

- The divergence of resulting letter's width from the expected width.
- The divergence of resulting letter's height from the expected height.
- The length of the segmentation path.
- The width of the segmentation path.

We repeat this process until the CC cannot be segmented into other letters or no other possible segmentation paths exist.

The two main parameters of the character segmentation technique are the *MinCharWidth* and *MaxCharWidth*. These parameters define the area in which the algorithm searches for the best segmentation path. During this procedure, we take into consideration only segmentation paths that result to characters with width in the limit of [MinCharWidth*average character height, MaxCharWidth*average character height]. For the MinCharWidth and MaxCharWidth parameters the following pairs are used: (0.3, 0.4), (0.4, 0.5), (0.5, 0.6), (0.6, 0.7), (0.7, 0.8), (0.8, 0.9). As a result we have several segmentation variations with and without applying noise removal. Each segmentation result is represented by a confidence which is based on the difference between average and dominant ratio of height to width of possible characters. Examples of the developed character segmentation methodology are given in Fig. 10 and 11.

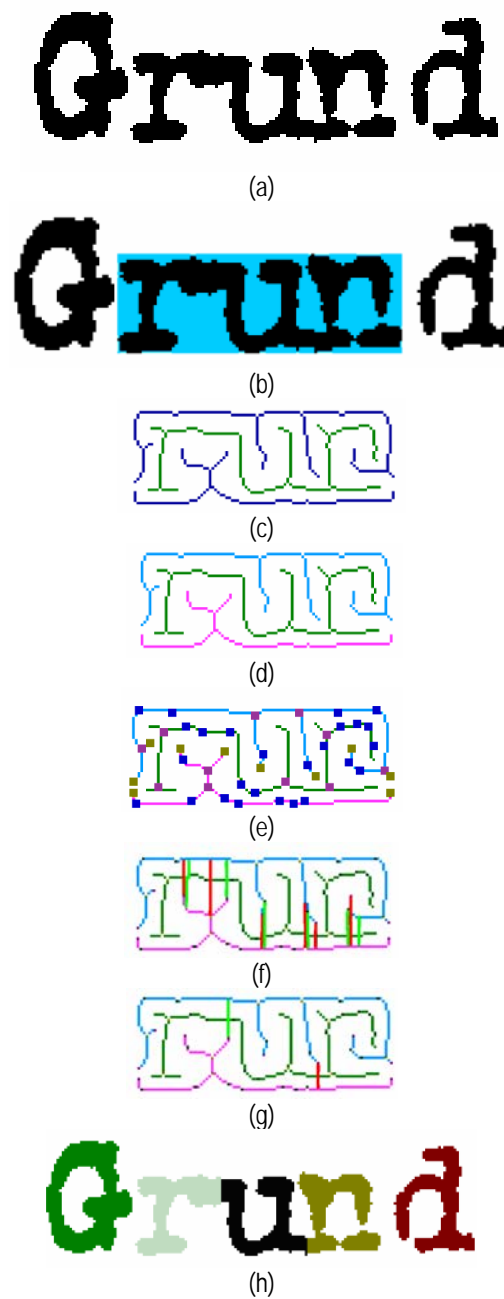


Figure 9. Example of character segmentation: (a) Original image; (b) Candidates CCs to be split; (c) skeleton of CCs (green) and their background (blue), (d) classification of skeletons: top-segment (blue skeleton), bottom-segment (pink skeleton) and stroke-segment (green skeleton); (e) feature points: fork points (red points), end points (brown points) and corner points (blue points), (f) possible segmentation paths, (g) best segmentation paths, (h) final result of character segmentation.

benzuzwohnen,

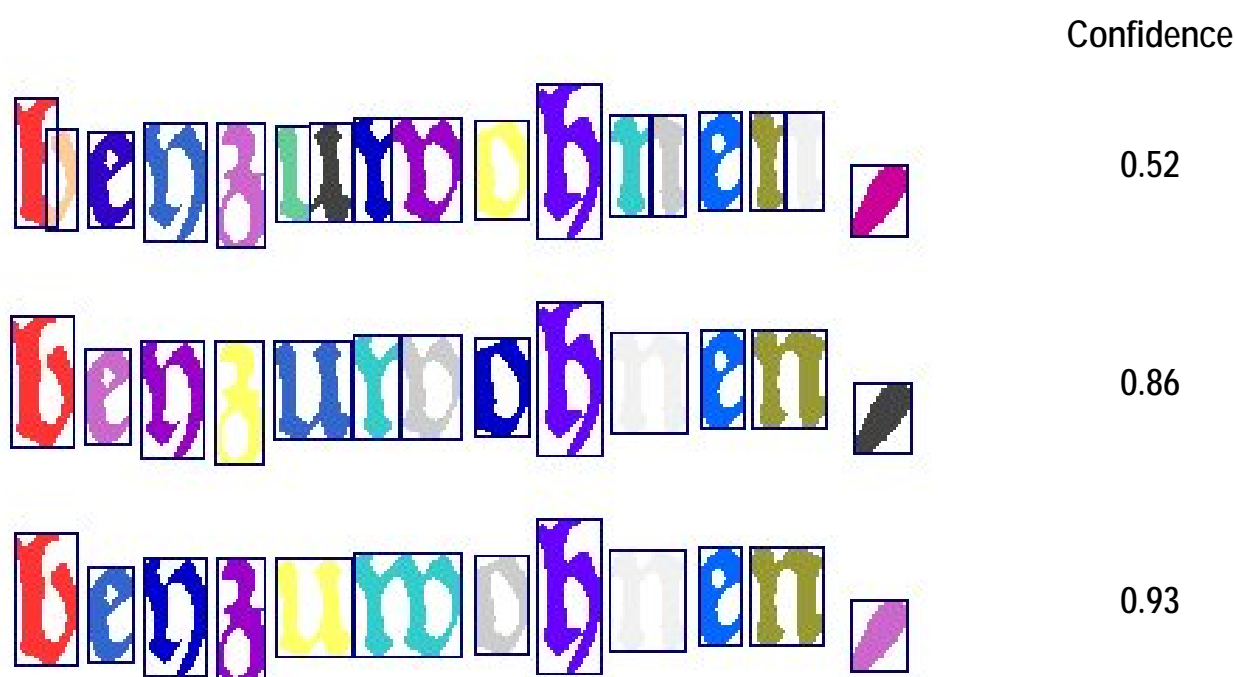


Figure 10. Example of several segmentation variations with their confidence.

würden.

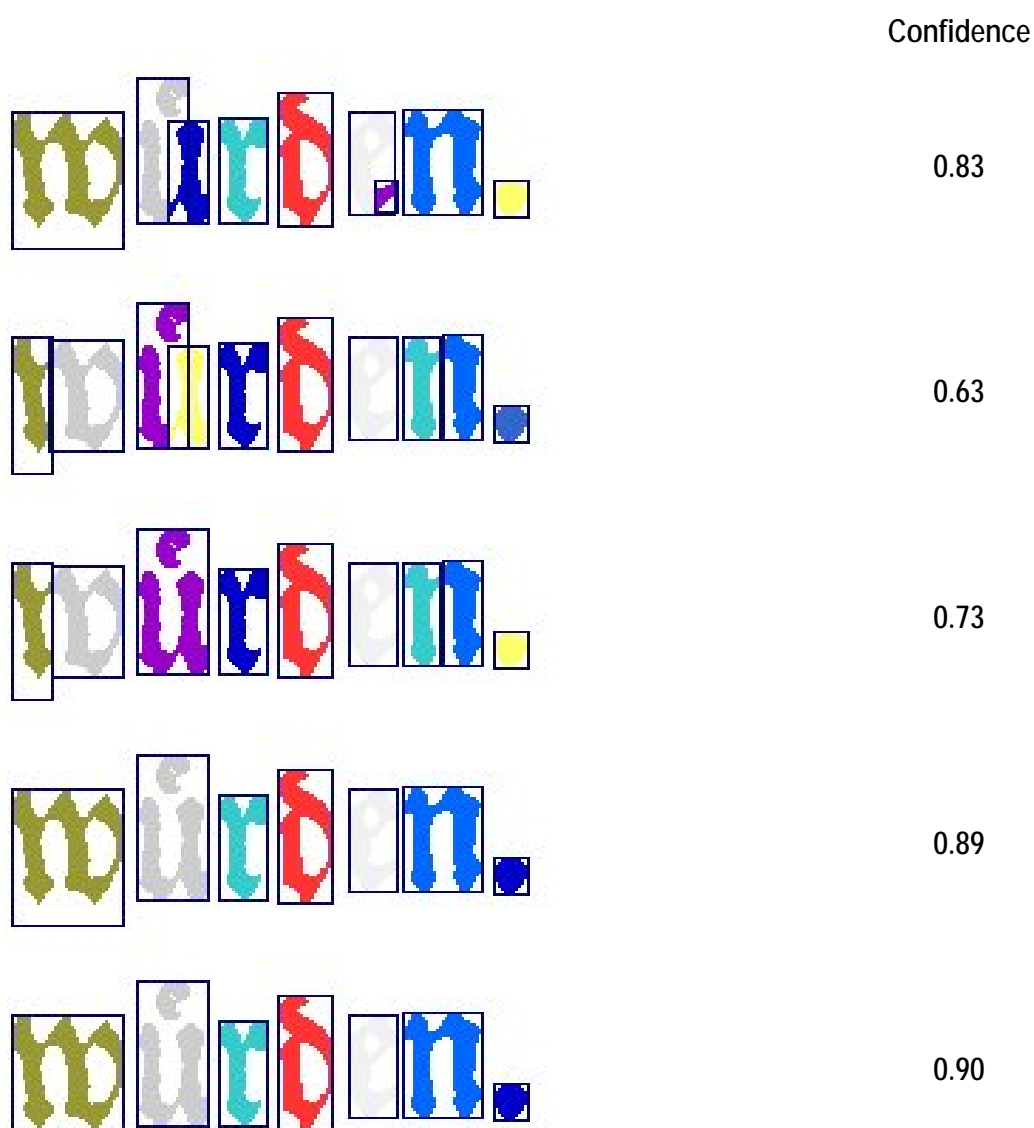


Figure 11. Example of several segmentation variations with their confidence.

The final version of the character segmentation toolkit (v.4) was delivered in a DLL version (can be downloaded from http://users.iit.demokritos.gr/~bgat/IMPACT_temp/CharacterSegmDLL_v3.rar). The call to the DLL has to be done as follows:

```
HINSTANCE CharSegmDLL = NULL;
typedef unsigned char __stdcall __declspec(dllimport) (*CharSegmFuncDLL) (int *IM,int lx,int ly,int hpitch,int bpp,int
*results,int *Rs1,float *conf1,int *Rs2,float *conf2,int *Rs3,float *conf3,int *Rs4,float *conf4,int *Rs5,float *conf5,int
*Rs6,float *conf6,int *Rs7,float *conf7);
AnsiString CharSegmDLLFileName = "Character_Segmentation_v3.dll";
CharSegmDLL = LoadLibrary(CharSegmDLLFileName.c_str());
CharSegmFuncDLL CharSegmFunc;

CharSegmFunc = (CharSegmFuncDLL) GetProcAddress(CharSegmDLL, "Character_Segmentation");
int res = CharSegmFunc
(IM,lx,ly,hpitch,bpp,results,Rs1,conf1,Rs2,conf2,Rs3,conf3,Rs4,conf4,Rs5,conf5,Rs6,conf6,Rs7,conf7);
where
INPUT
IM: The pointer to the image (255 corresponds to white and 0 to black),
lx: Image Width
ly: Image Height
hpitch: Horizontal Pitch
bpp: Number of bits per pixel
OUTPUT
results: Number of different segmentation results (1-7)
Rs1: Pointer to the labelled image that corresponds to the first result
conf1: Confidence of the first result
.
```

A labelled image is a raw image file where an integer is used for every pixel value and 0s (or <0) correspond to the background while all other IDs (>0) define the resulting different segmentation regions (see Fig. 12).

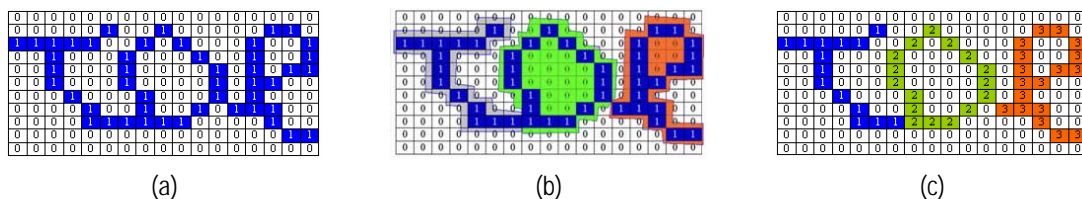


Figure 12. Labelled image example. (a) original image; (b) character segmentation result; (c) labelled image with character segmentation information.

5. Evaluation

5.1 Block Segmentation

After the active research and development stage was finished, a set of acceptance tests was performed in order to measure the segmentation quality improvements on documents specific for the IMPACT project.

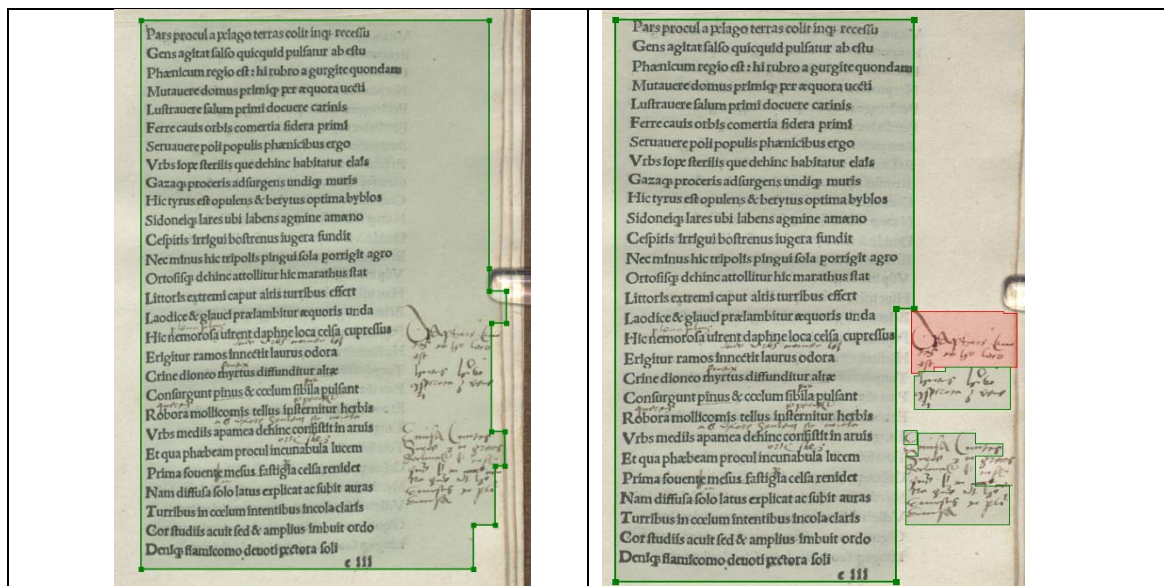
The first test was a human inspection of segmentation results on different types of images, many of them from the random sets collected at the beginning of the project. During the inspection, each page was given a score that describes its segmentation quality.

The scoring is as follows:

- 0: Segmentation is correct. No human intervention is needed
- 1: There are some segmentation problems. Slight manual work is required to fix it.
- 2: Huge segmentation problems. It is easier to segment the image manually rather than try to fix it.

After all the images in a testset were rated, the sum of all the values was calculated and compared with the result of the pre-IMPACT version. The new segmentation toolkit in such a test demonstrated 20% decrease in the number of errors compared to the previous version.

Below are some examples of better segmentation results:



Marginal notes detected (old left, new right)

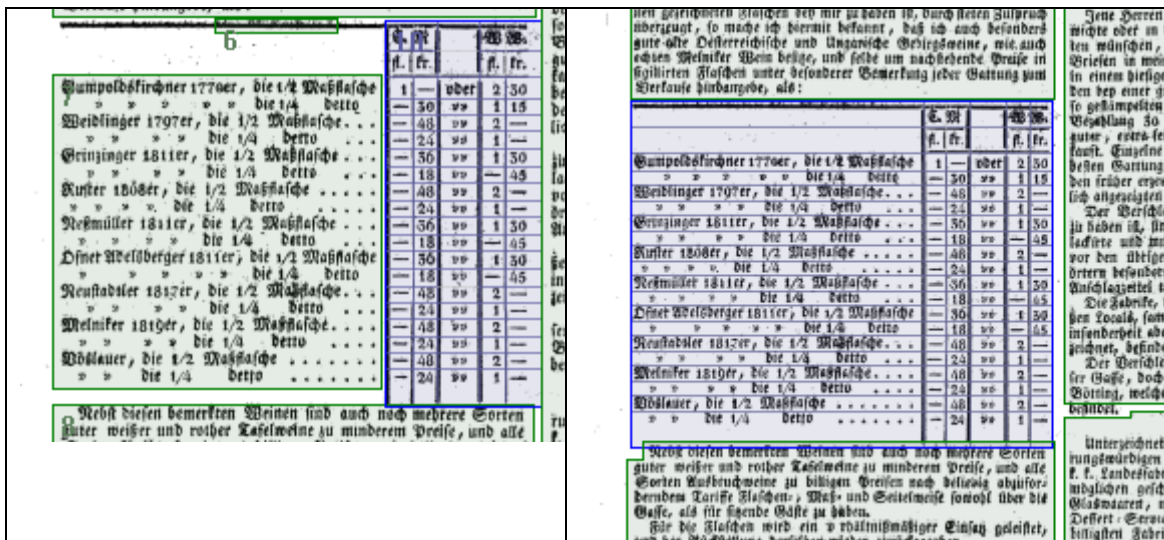


Table detected (old left, new right)

Evaluation of newspaper segmentation

With the availability of the segmentation ground truth in the second part of the IMPACT project it became possible to perform automatic segmentation evaluation on the historic newspapers.

The GT we used for the evaluation contains text and geometric data. For the proposed evaluation methodology, only geometric data is necessary. Every piece of text is contained in a corresponding region. Below is an example GT for a newspaper.



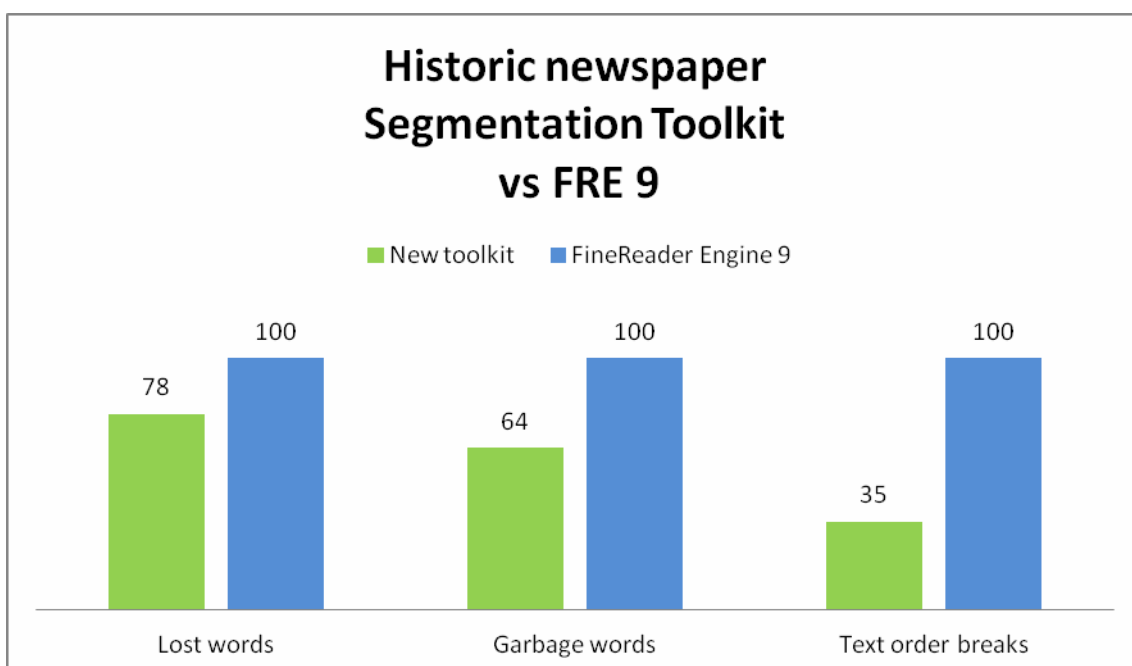
The main idea of the segmentation evaluation was to measure its quality via the recognition quality. There are three possible consequences of bad segmentation:

- First, when the segmentation does not detect any part of the text, this text is not available to the OCR;
- Second, when the segmentation detects a region that does not contain text as text regions, this slows down the recognition and, which is more important, the OCR result usually contains garbage;
- And the last possible problem occurs when the segmentation merges two or more columns of text into one. In this case all text is present in the OCR output but is completely useless, since the different parts of it are mixed into one unreadable mess.

So, the idea of our segmentation evaluation was to perform the recognition twice: the first time on the reference segmentation from the ground truth, and the second time on the segmentation of the toolkit, and to compare the OCR results.

After the comparison we obtain the mentioned three numbers: the number of lost characters after automatic segmentation, the number of extra (i.e. garbage) characters, and the number of text order breaks.

The testing results are shown in a chart below: there is a 22% improvement in the number of lost words, 36% less garbage text and 65% less text order breaks.

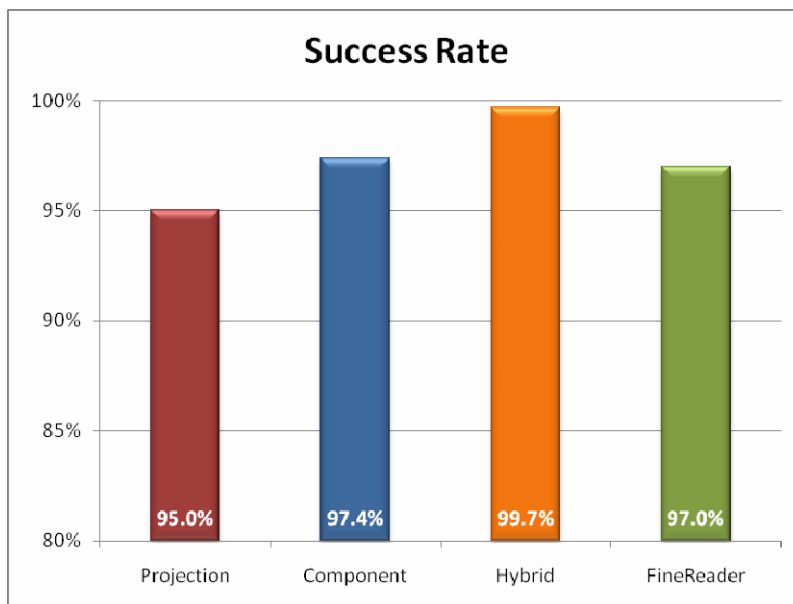


5.2 Text Line and Word Segmentation

The new IMPACT text line and word segmentation methods were evaluated on a carefully selected set of 66 documents, representing all major types of documents contained in the IMPACT image repository. Ground truth for those 66 documents comprises 809 regions, 2807 text lines and 18825 words. The performance evaluation method uses a measure based on merge, split, miss and false detection errors (see also D-OC3.1(i) Evaluation scenarios and metric definition and D-OC3.3(i) Evaluation tools). With the used approach individual problems can be highlighted or ignored by using specific set of weights (evaluation profiles). Input for the text line segmentation methods were text region snippets extracted from the original ground truth. For word segmentation, text lines snippets were used as input.

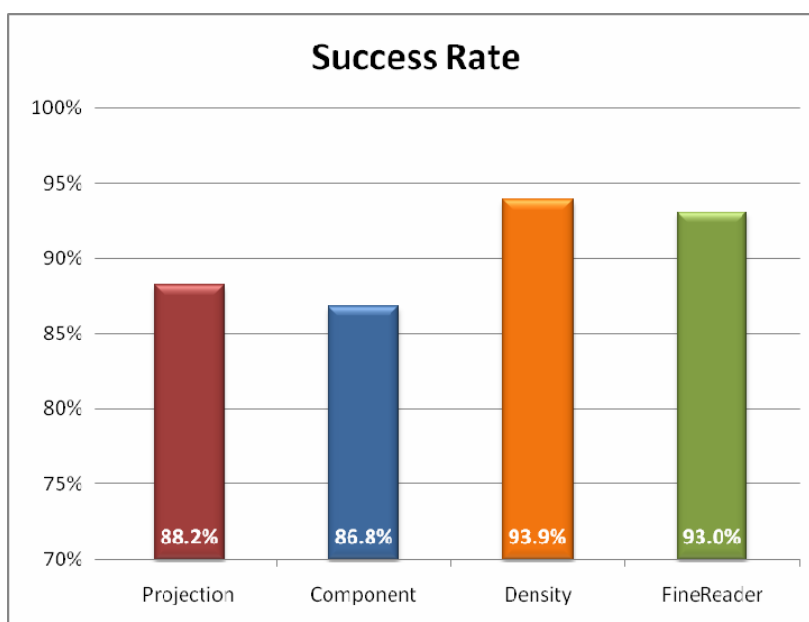
Evaluation results

The hybrid text line segmentation approach clearly outperforms the simple approaches based on projection profiles and connected component grouping. It also performs better than ABBYY FineReader Engine 9.0 which was chosen as commercial state-of-the-art system. See the following figure for an overview of the four methods:



Performance of text line segmentation methods

Similar results can be reported for the Pixel Density Word Segmenter which performs better than all other methods, including the commercial system:



Performance of word segmentation methods

It has to be noted that the IMPACT segmentation methods work solely on geometric features and do not rely on any form of text recognition or lexicon based analysis. This distinguishes the potential of the approaches even more, especially in cases where OCR is not feasible or where segmentation is to be carried out independently and prior to recognition tasks.

For more detailed evaluation results (obtained in the scope of T-TR2.5 Methodology evaluation) see also D-TR2.4(i) Evaluation reports.

Regarding quality assurance, there have been three layers of testing to ensure stability and compatibility of the tools:

1. Internal – by the main developer (USAL)
2. Internal – by an independent person who is not directly involved in the development (USAL)
3. External – by a technical project partner (NCSR – via integration into their demonstration platform)

Feedback from testing is centrally collected and managed by means of a server based bug tracker (Mantis).

Source code as well as releases are managed by a version control system (Subversion) which enables tracking of changes as well as reverting to earlier development stages.

5.3 Character Segmentation

In order to record the efficiency of the developed character segmentation toolkit we followed a well established evaluation approach that is also employed by several document segmentation contests we recently organized ([12], [13]). The performance evaluation method is based on counting the number of matches between the entities detected by the algorithm and the entities in the ground truth. We use a MatchScore table whose values are calculated according to the intersection of the ON pixel sets of the result and the ground truth.

Let I be the set of all image points, G_j the set of all points inside the j ground truth region, R_i the set of all points inside the i result region, $T(s)$ a function that counts the elements of set s . Table $MatchScore(i,j)$ represents the matching results of the j ground truth region and the i result region:

$$MatchScore(i, j) = \frac{T(G_j \cap R_i \cap I)}{T((G_j \cup R_i) \cap I)} \quad (1)$$

We consider a region pair as a one-to-one match only if the matching score is equal to or above the evaluator's acceptance threshold T_a (in our experiments we used $T_a = 0.95$). If N is the count of ground-truth elements, M is the count of result elements, and $o2o$ is the number of one-to-one matches, we calculate the detection rate (DR) and recognition accuracy (RA) as follows:

$$DR = \frac{o2o}{N}, \quad RA = \frac{o2o}{M} \quad (2)$$

A performance metric FM can be extracted if we combine the values of detection rate and recognition accuracy:

$$FM = \frac{2DR \cdot RA}{DR + RA} \quad (3)$$

For the evaluation of the character segmentation algorithm we manually cropped 21373 word images from randomly selected historical images which consist of 108537 characters. Then, we manually marked the correct character segmentation regions in the original b/w word images using polygons in order to create the ground truth set (see Fig. 13).



Figure 13. Example of creating character segmentation ground truth.

The developed segmentation algorithm produces several segmentation variations with confidence. To verify the validity of our methodology we applied two different evaluation scenarios. At the first one we evaluated the segmentation result with the highest confidence and, at the second one, the best possible segmentation result. For comparison purposes, we applied at the same set the commercial products FineReader Engine 8.1 [14] and OCRXpress [15], with the open source OCRopus library [16] as well as with 2 state-of-the-art methods based on RLSA [4] and on Projection Profiles (similar approach is used in [2],[3]). Table 1 presents the overall evaluation results.

Table 1. Evaluation results

Method	N	M	o2o	DR	RA	FM
FineReader	108537	114664	96199	88,63%	83,90%	86,20%
OCROpus	108537	126672	88666	81,69%	70,00%	75,39%
OCRXpress	108537	78461	69013	63,58%	87,96%	73,81%
RLSA	108537	109704	93567	86,21%	85,29%	85,75%
Projection Profiles	108537	114393	81901	75,46%	71,60%	73,48%
IMPACT (Highest Confidence)	108537	114641	97905	90,20%	85,40%	87,74%
IMPACT (Best Result)	108537	113370	102827	94,74%	90,70%	92,68%

As it can be observed:

- Concerning state-of-the-art techniques, best performance is achieved using FineReader (FM = 86.20%)
- The developed segmentation algorithm outperforms all the others methods and achieves FM= 92.68% when we evaluate the best possible segmentation result and FM= 87.74% when we evaluate the segmentation result with the highest confidence.

6. License and IPR Protection

6.1 Block Segmentation

ABBYY is not going to patent any work which is done within the IMPACT project.

All parts of the toolkit will be available as part of FineReader Engine product. Licensing of the toolkit is the same as actual licensing of FineReader Engine.

During the IMPACT project the FineReader Engine and toolkits based on it are available to the participants of the project in accordance with the Consortium Agreement.

6.2 Text Line and Word Segmentation

Copyright of this software lies with the PRImA Research Group, University of Salford, UK. Individual licensing agreements for commercial and non-commercial use are required; in case of solely non-commercial use fees can be waived.

For copyright of the used third party libraries libxml2 (<http://xmlsoft.org/>) and LibTIFF (<http://www.remotesensing.org/libtiff/>) see the respective websites.

6.3 Character Segmentation

This part of work is owned by the National Centre for Scientific Research "Demokritos", Greece (NCSR).

Free use is allowed for no-commercial purposes from the IMPACT partners during the project life time.

After the project ends, individual licensing agreements for commercial and non-commercial use will be required.

References

- [1] A. Antonacopoulos, D. Karatzas, Semantics-based content extraction in typewritten historical documents, in: Eighth International Conference on Document Analysis and Recognition, 2005, pp. 48–53.
- [2] S. Nomura, K. Yamanaka, O. Katai, H. Kawakami, T. Shiose, A novel adaptive morphological approach for degraded character image segmentation, *Pattern Recognition* 38 (11) (2005) 1961–1975.
- [3] X. Jia, X. Wang and W. Li, H. Wang, A Novel Algorithm for Character Segmentation of Degraded License Plate Based on Prior Knowledge, *Proceedings of the IEEE International Conference on Automation and Logistics*, 2007, pp. 249-253.
- [4] F.M. Wahl, K.Y.Wong and R.G. Casey. Block segmentation and text extraction in mixed text/image documents. *Comput. Graph. Image Process.*, 20: 375–390, 1982.
- [5] S. Liang, M. Shridhar, M. Ahmadi, Segmentation of touching characters in printed document recognition, *Pattern Recognition* 27 (6) (1994) 825–840.
- [6] E. Kavallieratou, E. Stamatatos, N. Fakotakis, G. Kokkinakis, Handwritten character segmentation using transformation-based learning, in: 15th International Conference on Pattern Recognition, vol. 2, 2000, pp. 634–637.
- [7] X. Xiao, G. Leedham, Knowledge-based English cursive script segmentation, *Pattern Recognition Letters* 21 (10) (2000) 945–954.
- [8] B. Yanikoglu, P.A. Sandon, Segmentation of off-line cursive handwriting using linear programming, *Pattern Recognition* 31 (12) (1998) 1825–1833.
- [9] U. Pal, A. Belaid, Ch. Choisy, Touching numeral segmentation using water reservoir concept, *Pattern Recognition Letters* 24 (1–3) (2003) 261–272.
- [10] Chang, T.C., Chen, S.Y., 1999, Character Segmentation Using Convex-Hull Techniques, *International journal of pattern recognition and artificial intelligence*, 833-858.
- [11] Y. Chen, J. Wang, Segmentation of single- or multiple-touching handwritten numeral string using background and foreground analysis, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22 (11) (2000) 1304–1317.
- [12] B. Gatos, N. Stamatopoulos and G. Louloudis, "ICDAR2009 Handwriting Segmentation Contest", 10th International Conference on Document Analysis and Recognition (ICDAR'09), pp. 1393-1397, Barcelona, Spain, July 2009.
- [13] B. Gatos, A. Antonacopoulos and N. Stamatopoulos, "ICDAR2007 Handwriting Segmentation Contest", 9th International Conference on Document Analysis and Recognition (ICDAR'07), pp. 1284-1288, Curitiba, Brazil, September 2007.
- [14] FineReader v.8.1 Engine: <http://www.abbyy.com/>
- [15] OCR Xpress v.1: <http://www.pegasusimaging.com/>
- [16] OCRopus open source library: <http://www.ocropus.org>