FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Physical Layout Analysis of Newspaper Images

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Visual Computing

eingereicht von

Thomas Lang, BSc.
Matrikelnummer 1025705

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Robert Sablatnig
Mitwirkung: Projektass. Dr.techn. Markus Diem, MSc
Projektass. Dipl.-Ing. Dr.techn. Florian Kleber

Wien, 3. Mai 2018

Thomas Lang	Robert Sablatnig

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Physical Layout Analysis of Newspaper Images

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Visual Computing

by

Thomas Lang, BSc.
Registration Number 1025705

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Robert Sablatnig
Assistance: Projektass. Dr.techn. Markus Diem, MSc
Projektass. Dipl.-Ing. Dr.techn. Florian Kleber

Vienna, 3rd May, 2018

Thomas Lang

Robert Sablatnig

Erklärung zur Verfassung der Arbeit

Thomas Lang, BSc.
Denisgasse 8/8, 1200 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 3. Mai 2018

Thomas Lang

Danksagung

Ich möchte mich bei allen bedanken, die mich während meines Studiums und der Verfassung der Diplomarbeit unterstützt haben. Die finanzielle Unterstützung durch meine Eltern und Großeltern hat es mir erlaubt, mich ohne materielle Sorgen auf das Informatik-Studium konzentrieren zu können. Aber auch bei sonstigen Problemen konnte ich mich immer auf meine Familie verlassen.

Während der Diplomarbeit, aber auch schon beim Schreiben der Bachelorarbeit vor einigen Jahren, waren die Mitarbeiter des Computer Vision Lab stets hilfsbereit. Mit inhaltlichen und sonstigen Fragen konnte ich mich immer an meine Betreuer Robert Sablatnig, Markus Diem und Florian Kleber wenden. Insbesondere will ich mich dabei bei Markus Diem bedanken, der mir bei der Themenwahl geholfen hat und mir während der gesamten Arbeit mit Ideen, Korrekturen und Verbesserungsvorschlägen zur Seite gestanden ist.

Acknowledgements

I want to thank everyone who has supported me during my academic studies and during the writing of the master thesis. The financial support from my parents and grandparents has allowed me to concentrate on my computer science studies without having to worry about my income. But also with other issues, my family was always there for me.

During the writing of the master thesis, but also during my bachelor thesis a few years ago, the staff of the Computer Vision Lab has always been helpful. My supervisors Robert Sablatnig, Markus Diem and Florian Kleber have always been open for any questions regarding my work. I especially want to thank Markus Diem, who helped me choose the topic and provided me with ideas, corrections and suggestions for improvement during the writing of my thesis.

Kurzfassung

Obwohl moderne Zeitungen digital erstellt werden, sind explizite Informationen über ihr Layout oft nicht vorhanden, wenn sie nicht getrennt abgespeichert und mit veröffentlicht werden. Die Beschaffenheit des Layouts kann, je nach Dateiformat, aus dem digital gespeicherten Inhalt des Dokuments rekonstruiert oder aus dem Dokumentenbild erschlossen werden. Die vorliegende Arbeit stellt eine einfache Methode vor, um rechteckige Regionsgrenzen vom Typ Text oder Bild, die aus Zeitungen und anderen Dokumenten im PDF-Format extrahiert werden können, zu benutzen, um korrekte und nach Typ klassifizierte Regionsbeschreibungen zu erhalten. Notwendig ist dies, da der Inhalt von PDF-Dokumenten nur nach Text- und Bildkomponenten unterschieden werden kann, und dabei vor allem beim Auslesen der Bilder wegen Schnittmasken je nach angewandter Methode Fehler entstehen können, wodurch die resultierenden Regionsgrenzen nicht mehr mit den Grenzen der sichtbaren Bilder im Dokument übereinstimmen. Daher werden die extrahierten Regionsgrenzen, welche sich überlappen oder zu groß sein können, in Kombination mit dem Dokumentenbild benutzt, um sie an die tatsächlichen inhaltlichen Komponenten des Dokuments anzupassen. Um die Regionen anschließend klassifizieren zu können, wird ein auf HOG Features basierender Random Forest Klassifikator an einem manuell annotierten Datensatz trainiert. Für die Klassifizierung zwischen Text- und Tabellenregionen wird dabei eine error rate von 0.05 und für die Unterscheidung zwischen Bild- und Diagrammregionen eine von 0.1 erzielt. Die Segmentierung, welche nur die extrahierten Bild-Regionen betrifft, wird zuerst nach dem Maß der flächenmäßigen Überlappung zwischen annotierten und segmentierten Regionen evaluiert, womit ein Recall und eine Precision von 0.77 und 0.94 erzielt werden. Zusätzlich wird sie danach evaluiert, wie viele segmentierte bzw. annotierte Regionen genau (mit einer variablen Toleranz) einer oder mehreren annotierten Regionen aus dem jeweils anderen Datensatz entsprechen. Je nach Toleranz ergeben sich dabei F_1-Werte zwischen 0.37 und 0.45, wobei sich herausstellt, dass die Segmentierung bei Diagrammregionen wesentlich öfter scheitert als bei Bildern. Die Evaluierung des Gesamtsystems wird erneut doppelt vollzogen: nach Überlappungsfläche und nach der Anzahl sich entsprechender Regionen. Auch dabei schneiden bei der flächenbasierten Evaluierung Diagrammregionen mit einem F_1-Wert von 0.21 deutlich schlechter ab als die Regionen der anderen Klassen. Schließlich wird die Methode noch mit Tesseract hinsichtlich der Segmentierung und Klassifizierung von Text-, Bild- und Tabellenregionen verglichen. Dabei erzielt die vorgestellte Methode hinsichtlich aller Klassen einen höheren F_1-Score.

Abstract

Even though modern newspapers are born digitally, layout information is often not available if it is not stored separately and distributed alongside them. Depending on the data format, the layout structure can either be reconstructed from the content stored in the digital file or from the document image. This work proposes a method for using rectangular region bounding boxes, which can be extracted from PDF documents, in order to obtain correctly segmented and classified region descriptions. This is necessary because the content extracted from PDF files can only be distinguished as being of the text or image type, and because of clipping masks for image components, the resulting boundaries may not match the visible images in the documents, depending on the extraction method. Therefore, the resulting region boundaries, which may be too large or overlapping, are used in combination with the document image to fit them to the visible component boundaries. In order to classify the regions afterwards, a HOG classifier is trained on a manually annotated dataset. The resulting error rates are 0.05 for the distinction between text and table regions, and 0.1 for the classification between image and chart regions. The segmentation, which is only performed on the extracted image regions, is first measured according to the amounts of area overlap between annotated and segmented regions, resulting in a recall value of 0.77 and a precision of 0.94. The second kind of segmentation evaluation consists of counting how many regions of one dataset (segmented or ground-truth) fit a region of the respective other dataset, according to a variable tolerance value. Depending on this value, the resulting F_1 scores lie between 0.37 and 0.45. This evaluation also reveals that the segmentation fails significantly more often for chart regions than for images. Both measurements, region overlap and fitting region counts, are again used in the evaluation of the complete system, which shows once more that chart regions are segmented significantly worse than other classes with an overall area-based F_1 score of 0.21. The last measurement is a comparison of the proposed technique to Tesseract in terms of segmentation and classification of text, image and table regions, where the proposed system performs better in terms of F_1 scores for all classes.

Contents

CHAPTER 1

Introduction

Physical (or *Geometric*) layout analysis describes the detection and classification of different regions in a document image. Kise defines it as the combination of the segmentation of the image into separated regions and the classification of their contents such as text, images, charts or tables [Kis14]. *Logical* layout analysis is concerned with recognizing headers, footers, body text etc. [Smi09] and will be out of the scope of this thesis.

Like all document analysis tasks, layout analysis is required when there is a need for having digital representations of documents which are only available in printed or handwritten form. Document analysis makes it possible to recognize the contents of document images (acquired via scans or photographs), resulting in a digital representation. By locating and identifying the different regions of a document image, layout analysis is an essential step for subsequent tasks like handwritten text recognition (HTR), optical character recognition (OCR), word spotting, table understanding etc. Since modern newspapers and magazines are born-digital documents, meaning that they are initially created and stored digitally and only afterwards distributed in different forms, the task of page segmentation and document analysis in general can be superfluous when all the needed information is stored in the digital files. However, publications are often only available in printed format, and even if they are distributed digitally, since they are converted to different formats when they are released (for example PDF), some of the information in the resulting files can be missing or modified. Because of these problems, layout analysis on such documents can be a viable (or the only) option to reproduce the lost information.

Modern newspapers and magazines often have non-manhattan or overlapping layouts. Non-manhattan layouts have non-rectangular region boundaries (see Figure 1.1a), and overlapping layouts contain objects that are different, but not physically separated (e.g. by whitespace), for example text written on images. These difficulties cause conventional page segmentation techniques to fail, which rely on rectangular region boundaries. For non-manhattan layouts, it is not possible to segment the page into rectangular blocks, and either more complex shapes like polygons have to be used or

(a) A non-manhattan document from [ABPP09].

(b) A newspaper page with overlapping layout.

Figure 1.1: Examples of documents with complex layouts.

each pixel/superpixel is assigned to a region individually. For overlapping layouts, the process of page segmentation can no longer be separated from region classification, and regions can only be distinguished by the different kinds of content they contain. Because of these challenges, analysis of such complex layouts is still a difficult problem and an ongoing research topic.

1.1 Physical Layout Analysis of Partly Annotated Newspaper Images

In this thesis, an approach for physical layout analysis is presented which is tailored to a dataset of newspaper images containing partly correct rectangular document region descriptions of two types of content: text and images. The goal is to use the available incomplete region information to obtain correct descriptions of the locations and sizes of text, image, chart and table regions.

The main approach is described in chapter 3. Section 3.1 gives a detailed description of the properties of the region rectangles included in the dataset and shows examples of correct and of problematic cases. As a result, it turns out that the included text

regions are typically correct, while the image regions often only fit parts of actual images contained in the newspaper pages and tend to overlap each other and adjacent areas of other content types. The observed errors in the sizes and positions of the region rectangles define what the following segmentation has to achieve. Additionally, since we know that the available region descriptions contained in the dataset have been extracted from PDF files, we take a look at the structure of such files to learn how similar region rectangles could be extracted from them. This allows for generalization of the approach for other datasets.

In the sections 3.2 and 3.3, the main methodology is explained: how the inaccurate region rectangles can be segmented to correctly describe content regions and how they can subsequently be classified to identify their actual type. According to the most commonly observed errors in the available region descriptions, a simple segmentation approach is proposed which starts by finding clusters of overlapping image region rectangles and then adjusts them to connected components of foreground pixels contained in them. For the region classification, a random forest classifier is trained on a manually annotated ground truth dataset, after which it can be used to classify segmented regions. The whole process from start to finish is explained in section 3.4. Since only rectangular ground-truth information is available for the dataset, the segmentation method also adheres to this constraint and produces region bounding boxes. However, section 3.5 explains how the technique could be extended to produce exact regions.

In chapter 4, both major steps of the method are evaluated. In section 4.1, the classification performance is measured by the confusions between classes and the precision and recall values of each class. The segmentation is evaluated in section 4.2 by summing up the amounts of overlapping areas between segmented and ground truth regions. Additionally, we count how many regions of one set (segmented or ground truth) perfectly fit regions of the other set. Afterwards, the complete system is evaluated (section 4.3) by measuring the results of both segmentation evaluation methods on all regions of each class individually. Finally, in section 4.4, the same metrics are used to evaluate the results of the page segmentation component of the open source OCR engine Tesseract[1], which can then be compared to the proposed method.

1.2 Basic Definitions

The following section provides a brief introduction to the topic and defines the terms used throughout this thesis. A more comprehensive overview over the topic of page segmentation and component classification can be found in the work by Kise [Kis14].

The goal of layout analysis is to locate and identify the regions of different types that make up a document. The extracted regions containing text, images, tables etc. can then be further analyzed and processed. Layout analysis can be divided into two main tasks: the separation and identification of regions is called *physical* or *geometric* layout

[1] `https://github.com/tesseract-ocr/tesseract/`

analysis and the semantic understanding of the layout is called *logical* layout analysis. Logical layout analysis is concerned with classifying text zones according to their different meanings: title, subtitle, paragraph etc. In this thesis, we are only concerned with physical layout analysis.

Kise further groups physical layout analysis into *page segmentation* and *component classification*, the first being only the separation of different parts of the document image, and the second the recognition of the types of content they contain. We will see that these tasks can sometimes, but not always be separated, depending on the layout constraints of the document.

1.2.1 Types of Layout Constraints

Since documents are designed for human readers, they tend to follow certain rules in order to present the contained content to the reader in the desired way and in the correct order. The layout constraints of a document determine how the problem of layout analysis can be approached and how difficult it is. If the document has a known fixed layout, the problem is already solved. Generally, the more unconstrained the layout, the more sophisticated the analysis methods have to be.

The most constrained type of layout, apart from a completely fixed layout, is called *rectangular* and contains only physically separated regions of rectangular shape which are parallel or perpendicular to the page borders and to each other. *Manhattan* layouts additionally allow regions to be non-convex while still having straight borders. Layouts that contain visually separated regions with arbitrary shapes are called *non-manhattan*. Finally, *overlapping* layouts even contain intersecting regions. Examples of such complex cases are shown in Figure 1.1.

1.2.2 Page Segmentation

Page segmentation techniques take advantage of layout constraints in order to separate the document components. Therefore, knowledge about the constraints of the document at hand is required in order to get correct results. For simple page segmentation without class distinction, all document components must be visually separated in the document, so that their locations and boundaries can be identified. Additionally, it is often required that the document content is neither skewed nor warped.

The segmentation can be achieved by directly analyzing the foreground objects (meaning the actual page content), or by finding regions of background color, which are likely to separate foreground entities. Since every document consists of foreground and background, and both are complementary to each other, either one may be used to find the same region boundaries. However, as the complexity of documents increases, it becomes more difficult to decide whether a page element belongs to the foreground or to the background. The page shown in Figure 1.1b contains an image which is clearly separated from the white page background, but itself serves as background for the text and chart elements layered on top of it.

Primitives other than pixels can be used for the segmentation of a document image. All pixels in a document may be divided into small groups called superpixels, reducing the number of primitive elements and therefore the complexity of the problem. However, this always comes with the danger of losing important information, since the individual pixels contained in superpixels are no longer considered. For binary images, most individual characters contained in text result in single connected components of pixels (CCs), which is why they are often used for further analysis.

There are different strategies for page segmentation. The top-down strategy starts with the complete document and divides it into smaller components, until the desired level of detail is reached (e.g. paragraphs, text lines or words). The bottom-up approach combines the chosen document primitives until they form the desired components.

In the earlier days of layout analysis, approaches were developed using some basic ideas which still serve as the foundation of many modern methods. The run-length smearing algorithm (RLSA) by Wong *et al.* [WCW82] fills runs of white (background) pixels in the binary image which are shorter than a certain length, thereby "smearing" them. This is done horizontally and vertically (for all pixel rows and columns), after which the logical AND operation is performed between both images. The resulting binary image ideally represents a pixel mask of all text lines. The idea of X-Y-Cuts is a classical top-down method, which was initially developed by Nagy and Seth [NS84]. It finds the components of a rectangular layout by alternately dividing the document along vertical and horizontal streams of whitespace (background color). These are found by creating projection profiles of the page, whereby all foreground pixels in the binary document image are summed up along both dimensions for every row or column respectively. The valleys in the resulting distribution reveal the streams of whitespace. The Document Spectrum ("Docstrum") technique by O'Gorman [O'G93] computes all distances and angles between foreground connected components and utilizes them to determine whether the CCs belong to the same region (e.g. text line, paragraph etc.).

Even though page segmentation methods for complex layouts exist, the concept becomes problematic when page regions are overlapping, because they are no longer physically separated. Instead of first finding the page components and then classifying them, each primitive has to be classified individually and may belong to multiple classes. Therefore, in such cases, the problems of page segmentation and component classification can longer be separated.

1.2.3 Component Classification

After the document is segmented into separate components, they are classified. For OCR applications, the only distinction of interest often is whether regions contain text or something else, in order to extract all the text content. In our case, we are interested in all kinds of content commonly contained in newspapers. We classify regions into four equally important groups: text, image, chart and table.

Early methods for component classification like the ones published by Fisher *et al.* [FHD90] and by Shih and Chen [SC96] rely on fixed sets of rules for classifying the binary content of each component. They compute properties of connected pixel components, black-pixel run lengths, transitions from white to black pixels or similar features in order to compare them to a set of thresholds, which then determine whether the region contains text or something else. In the approach of Altamura *et al.* [AEM01], similar rules are used, but their parameters are generated using machine learning (in this case, decision trees).

For the segmentation of overlapping layouts, pixels can be classified using the surrounding image texture. For example, a work by Jain and Zhong [JZ96] uses a neural network to classify every pixel in a document image using its neighborhood (7×7) as either text, background or halftone (graphics).

CHAPTER 2

Related Work

A lot of progress has been made in physical layout analysis of complex documents. For the problem of overlapping layouts, Baird *et al.* proposed versatile methods for pixel-wise (instead of region- or superpixel-wise) training and classification of documents using k-NN classifiers and k-d trees [BC06] [BMN+06]. They have been complemented by a post-classification step for merging neighboring pixels to uniform regions [ABX07]. In 2009, Ray Smith published a method for detecting text and non-text regions [Smi09] as part of the Tesseract OCR engine[1], that takes advantage of the fact that publishing systems use tab-stops for laying out documents.

In the context of the biannual ICDAR (International Conference on Document Analysis and Recognition) conference, several competitions are held, including, since 2001, one on the topic of page segmentation (including region classification). These competitions are organized by the PRImA research lab[2] and aim to objectively compare layout analysis methods on a common dataset. In the course of these competitions, they have devised tools for ground-truthing document images like Aletheia [CPA11a] and an XML-based data format for storing layout information called PAGE [PA10]. This framework was furthermore used to create an annotated dataset of magazine document images with complex layouts [ABPP09]. These resources were first used in the 2009 Competition on Page Segmentation [APBP09], which was concerned with layout analysis on realistic, "everyday" documents. Four methods were submitted and compared to each other and two commercial systems, the overall winner being submitted by Iuliu Konya, Stefan Eickeler and Christoph Seibert of the Fraunhofer Institute (not published to this date). The 2011 competition [ACPP11] focused on layout analysis of a dataset of historical documents. It again included four submitted and two commercial methods; the best one was submitted by the French company JOUVE[3]. The topic of the competition held in 2013 [ACPP13]

[1] https://github.com/tesseract-ocr/
[2] http://www.primaresearch.org/
[3] http://www.jouve.com/

was historical newspaper layout analysis (HNLA), where five methods were submitted and compared to two commercial and one open-source method (Tesseract). A method submitted by Chen *et al.* [CYL13] was determined to have the best results, closely followed by the Fraunhofer method which was also submitted to the 2011 competition. In 2015, the competition was held again with the aim of analyzing documents with complex layouts [ACPP15], where four submitted methods were compared to one commercial and one open source technique in addition to previous versions of them. The results show that the 'MHS' technique submitted by Tran *et al.* from Chonnam National University had a clear overall advantage over all others. The same topic was continued in 2017 [CAP17], where five methods were submitted and compared to each other and to two state-of-the-art systems. An updated version of the MHS method again proved to be superior over the others.

Based on their experience in evaluating layout analysis methods at the PRImA research lab, Clausner *et al.* published new evaluation metrics for layout analysis competitions in 2011 [CPA11b], which are an improved form of the metrics used in the 2009 page segmentation competition [APBP09] and have been used in all PRImA layout analysis competitions since. The evaluation method uses geometric region representations (using polygons) instead of pixel masks. Regions are represented using intervals, which are decompositions of the region polygons into sets of horizontally maximal rectangles. Region overlap amounts can then be efficiently calculated using combined interval representations. Overlaps between segmented and ground truth regions are grouped into four cases. Several (more than one) ground truth regions being overlapped by one segmented region is called a *merge*. A *split* occurs if a ground truth region overlaps more than one segmented regions. If a ground truth region is not or not completely overlapped by a segmented region, it is called a *miss* or *partial miss*. A segmented region which does not overlap any ground truth regions is a *false detection*. Additionally, overlaps between segmented and ground truth regions of different types are called *misclassifications*. All errors are then quantified by the amount of areas which are not or falsely overlapped. Errors are also affected by rules, which define which cases are allowed or set weights for different kinds of errors. For example, certain merges may be allowed while others would represent a change in document structure and are penalized. Another example would be that certain confusions between region classes may be more severe than others. Some of these rules are predefined, and some are user-defined. Sets of user-defined rules and weights are called *evaluation profiles* and represent one specific layout analysis evaluation scenario. The evaluation approach used in this thesis is simpler than the one proposed by Clausner *et al.*, but includes some similar ideas (see chapter 4).

In addition to these general approaches, there are also methods for detecting specific regions in document images, which could be used in combination to produce a complete document image layout description. In 2013, a competition on table recognition was held [GHOO13], evaluating solutions for the problems of table location, table structure recognition and both tasks combined. The best-performing method in the sub-competition on table location was submitted by Nurminen [Nur13]. The technique submitted by Chen

et al. for the 2013 HNLA competition (see above) focuses on locating text regions based on the extraction of whitespace rectangles. Chiu *et al.* proposed a picture detection method [CCD10] which first uses an available OCR system to find all text pixels in order to then cluster all non-text pixels using the Normalized Cuts algorithm [SM00]. In an additional step, segmented clusters are split or merged according to the number of associated captions of each picture.

A competition was held on page object detection in the context of the ICDAR2017 conference [GYJ+17]. Even though it mainly focuses on scientific publications, the proposed techniques for detecting tables and figures could also be used in complex documents like newspapers. The NLPR-PAL method submitted by Li *et al.* involving a deep learning method achieved the best overall results.

It has already been mentioned in section 1.1 that the partly correct region rectangles included in our dataset were extracted from PDF files. Chao and Fan published a method for obtaining high-level layout information from PDF documents [CF04]. They first extract all objects (text, raster images, path objects etc.) from the document content stream. All text objects are first grouped into words, which are then combined to text lines according to their orientation, font size, style etc. Text lines are combined to text segments according to line gaps. The resulting text segments are traced in order to find their exact boundaries. The content is extracted by performing OCR on each text line contained in a segment and concatenating the results. For all image objects, the corresponding masks and clipping paths, which define the visible image boundaries, are extracted from the PDF files and stored in the SVG format along with the images themselves. Because vector graphics are described by path objects in the PDF format, which are mathematically defined, it can be difficult to group them together. The authors decided to create a bitmap image of all path objects and apply a page segmentation technique on it in order to find the logical components, which are then stored in the SVG format. As a result, the positions, boundaries and contents of all text segments, vector graphics and embedded raster images are known.

CHAPTER 3

Newspaper Layout Analysis

The proposed method has been developed based on an available dataset of newspaper images along with XML region information in the form of rectangles, which is analyzed in section 3.1 and shown to be only partly correct, but can be utilized for further processing. Since the image region rectangles are especially problematic and often only partly describe the underlying true image regions, a segmentation step is added to obtain the correct image region boundaries, which is explained in section 3.2. Section 3.3 then describes a method of classifying the resulting text and image region rectangles as text, image, chart and table regions. These steps make up the complete layout analysis system, which is summarized in section 3.4. Section 3.5 presents ideas to make the system more versatile by producing exact image boundaries instead of only rectangular bounding boxes. Finally, section 3.6 briefly shows how the system has been implemented.

3.1 Dataset

The dataset was retrieved from a digital repository of Austrian and German daily newspapers. Every page is available as PDF or JPEG image file and includes an additional XML file containing information about text and image regions, which was automatically extracted from the PDF file by the provider of the dataset. The XML files describe the document contents in terms of two types of regions: *text* and *image*. This means that other types of regions like charts, tables or separators are either described as text or image regions, or not described at all. All regions are stored as rectangular bounding boxes, even if the underlying true regions are non-rectangular. To keep the amount of data manageable, it was decided to only use documents from the publications "Heute", "Kleine Zeitung", "Kronen Zeitung", "Kurier", "Die Presse", "Der Standard" and "Süddeutsche Zeitung", all of which are daily newspapers. Additionally, to keep the focus on the pages containing the main news articles, the front pages and page numbers above 20 were left out of the dataset.

Figure 3.1: Some coordinates of the image region corners are located outside the document boundaries.

In order to assess the quality of the included XML metadata, visual inspection of the contained region descriptions is performed as a first step. This in turn requires conversion of the data into a suitable format that can easily be visualized.

3.1.1 XML Format Conversion

The XML files included in the dataset use a custom format, which contains a variable number of text and image tags and has the following structure (attributes are left out):

```
<file>
  <page>
    <text>...</text>
    ...
    <image>...</image>
    ...
  </page>
</file>
```

The attributes of the *file* tag store the number of pages (always 1), the DPI resolution of the image, the date, the file name and the publication code. The *page* tag contains the page number (always 1), the image height and the image width. Each *text* tag stores the minimum and maximum font sizes present in the region, the text rotation, a unique element ID, the region pixel area and two sets of region coordinates x1, y1, x2, y2. The attributes of an image tag are its region coordinates, the pixel area, an element ID, a generic image name like “img14” and a second ID named “objref”.

We choose to convert these XML files into the PAGE format [PA10] developed by PRImA Research[1], which is a comprehensive and flexible data format for document analysis and allows the use of existing tools such as PRImA's own PAGE viewer[2] and the Aletheia ground-truthing tool [CPA11a].

Not all of the available information is kept during the conversion. The region area is not needed and can easily be calculated from the region size afterwards. Therefore, for each region, only its type, id, rectangle corner points and text content are stored during the conversion. Additionally, some region rectangles contain points outside of the image boundaries, as illustrated in Figure 3.1. All such coordinates are clamped to the page boundaries during the conversion, so that x-values lie in the interval $[0, \text{width} - 1]$ and y-values are contained in $[0, \text{height} - 1]$. The result is a PAGE XML file with a `Page` tag enclosing a variable number of `TextRegion` and `ImageRegion` tags.

3.1.2 Visual Inspection

For each of the selected publications, 30 pages (in total 210 out of 891 labeled pages) along with their XML region descriptions are visually inspected in a systematic way. The purpose of the inspection is to manually evaluate the correctness of the regions given by the XML data and to find common patterns where the region descriptions are not correct. We find that in many cases, region correctness (meaning the stored region rectangle in the XML file) is difficult to decide, either because the type of the region is ambivalent or the region boundary cannot clearly be defined. For all targeted region types (text, image, chart and table), it is therefore attempted to find reliable assumptions that can be made about the `TextRegion` and `ImageRegion` rectangles describing them and to make note of the different cases of incorrectness and ambivalence, in order to be able to take them into account in the segmentation and classification system.

Text

In all inspected documents, text regions are described by a corresponding XML entry, except if the text is part of an image, meaning that it is not explicitly stored as text in the PDF. This is only the case when text is part of an image or advertisement; actual content of newspaper articles is always described by text regions. Figure 3.2 shows three correctly described text regions and an advertisement image containing text, which is not described in the XML file. Additionally, the visual inspection shows that every text region describes a single block of text with the same font size. While this usually also implies a homogeneous text color, there can be exceptions like single words highlighted in a different color.

The XML rectangles sometimes are larger than the actual text area, which can cause it to contain parts of neighboring areas. This problem is amplified by the fact that the region descriptions are always rectangular. The result is that although there is always

[1] `http://www.primaresearch.org/`
[2] `http://www.primaresearch.org/tools/PAGEViewer/`

Figure 3.2: Text regions described in the source XML, visualized by rectangles. The text in the image on the left is not recognized as such, because it is part of the image itself, which depicts an advertisement.

one text region for every block of text, it cannot be safely assumed that the region rectangle contains *only* this block of text. Examples of different cases of overlapping text regions are depicted in Figure 3.3. Another case, which is rare and only occurs in 1 of 30 "HEUTE" pages, is the existence of empty text regions, as shown in Figure 3.4.

Images

In contrast to the `TextRegions`, which are mostly correct, the `ImageRegion` rectangles are less reliable. Throughout this work, images are defined as graphical depictions of any kind, including photographs and advertisements, which are not designed to present data to the reader (in contrast to charts). There are several cases of incorrect or flawed image region descriptions in the inspected dataset:

- A single visible image may be described by several overlapping or neighboring regions. Some of the region rectangles are tied together with zero overlap, and some of them have very large intersections. There is no general rule of how region boxes in such "`ImageRegion` clusters" are aligned. In some cases, the union of these rectangles perfectly describes the bounding box of the underlying image; in other cases not.

- Image region boundaries, or the boundaries of image region clusters, can be larger than the actual images and may include surrounding page content.

- When text surrounds the visible image area and is included in its bounding box, there are sometimes separate image regions for these parts of text.

- Image region rectangles may contain no visible image at all. Such regions are either empty or contain content of a different type.

Figure 3.3: Figure (a) shows non-rectangular text regions where a third text region is overlapped by two others. Figure (b) shows a text region containing part of an image. In Figure (c), two image captions separated by space are described by the same text region rectangle, causing it to overlap all other regions in-between. Figure (d) shows a case where the heading region rectangle is large enough to include text of surrounding regions.

Figure 3.4: Some text regions around the image do not contain any text.

Examples of these problems can be seen in Figure 3.5. On the other hand, there are image regions which are correctly described by the rectangles in the XML. Two examples can be seen in Figure 3.6.

Charts

Charts are depictions intended to visualize data. They are usually made up of simple graphics combined with textual descriptions, e.g. single-colored bars with corresponding numbers. Some examples of charts are diagrams, graphs, maps, bar plots and pie charts. In the inspected subset of documents, text contained in charts is always labeled as a `TextRegion`, except for 6 cases out of 299. The graphical parts of charts are not always labeled as images. Figure 3.7 shows an example of bar charts, where every bar is labeled by an `ImageRegion`, whereas in Figure 3.8, only the numbers are described by `TextRegions`, but the pie graphic itself is missing `ImageRegions` (the reason for this is explained in section 3.1.3). In the inspected newspapers, charts are often combined with photographic images (examples in Figures 3.5a and 3.5c), which are labeled by `ImageRegions` like all other images. In general, a chart is only described by a single region rectangle by coincidence, if it is contained within an image or can also be interpreted as one. An example of this case can be seen in Figure 3.9. Otherwise, they are only described by free-floating `TextRegions` along with `ImageRegions`, the existence of which cannot be relied upon. This makes the localization of chart regions based on the given XML regions alone a difficult problem.

Tables

The inspection reveals that tables are always described by `TextRegion` elements. The only exception is an advertisement containing 7 tables as part of an image. All other instances can be grouped into three types:

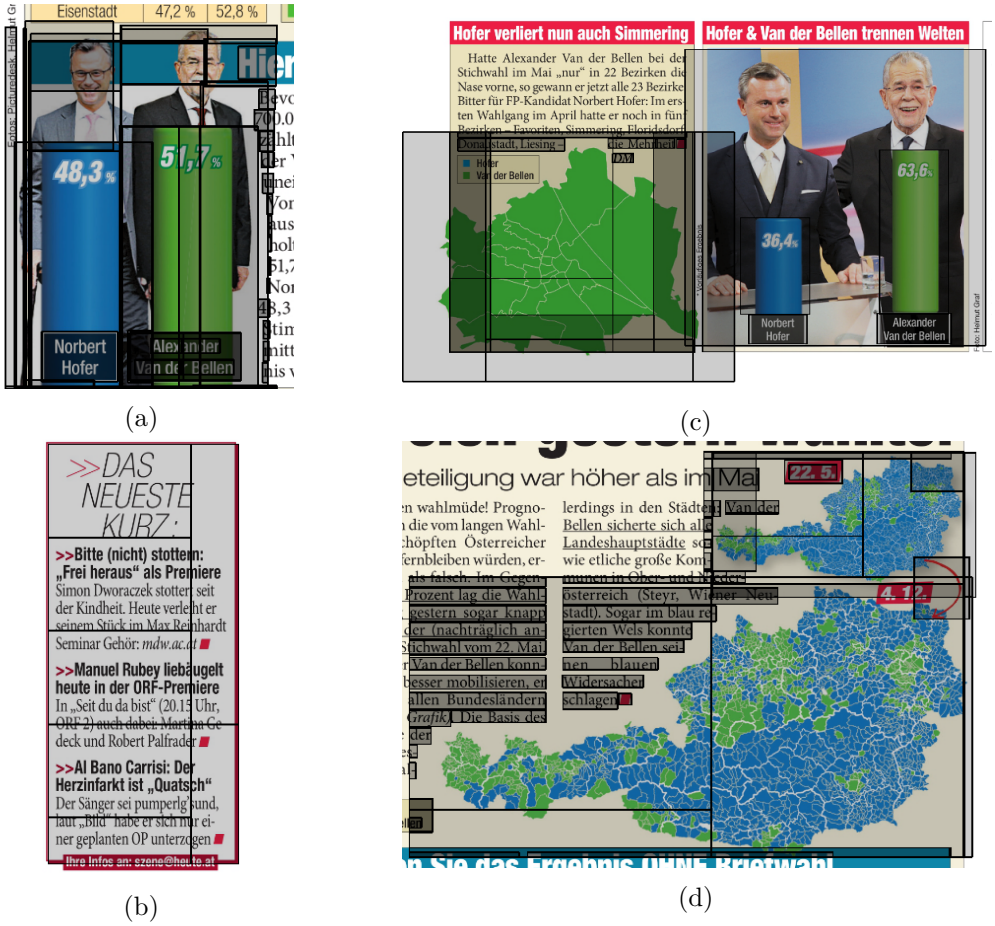
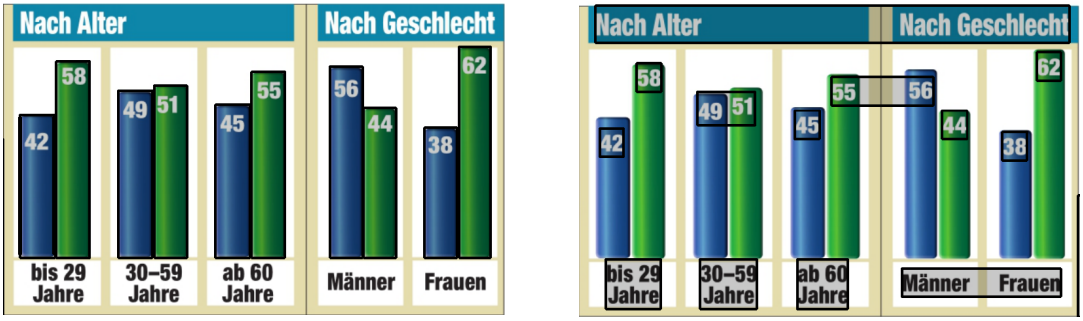


Figure 3.5: Figure (a) shows an image with a chart that is roughly described by overlapping region bounding boxes. In Figure (b), there is no visible image. Instead, text is labeled as a cluster of images. Figure (c) shows a case where the regions are bigger that the actual images and overlap into the neighboring depictions. Figure (d) shows a chart (more specifically, a map) annotated by image region rectangles which include large portions of the surrounding text.

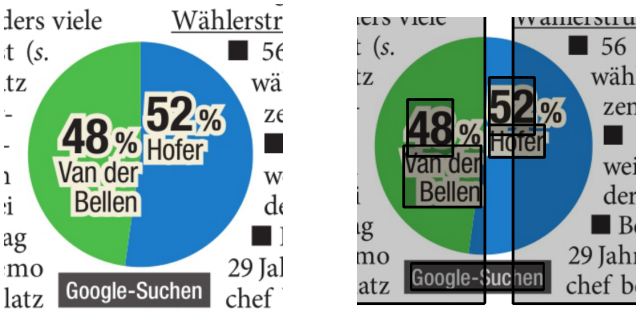
(a) (b)

Figure 3.6: A correctly described rectangular image and a non-rectangular, overlapping one with a correct bounding box.



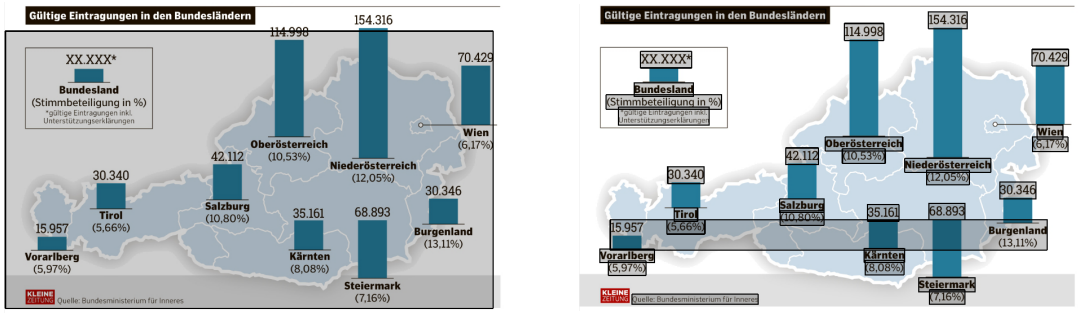
(a) (b)

Figure 3.7: Image region (a) and text region (b) labels in a collection of bar charts.



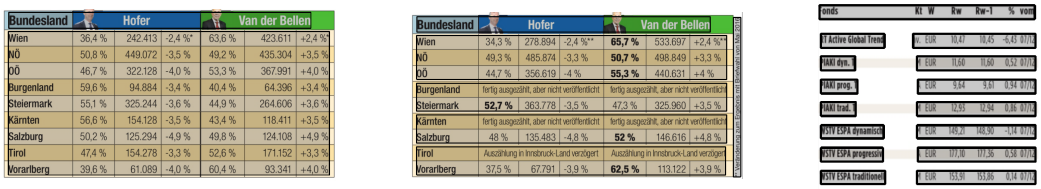
(a) (b)

Figure 3.8: A pie chart, which is only described by `TextRegions` (b). Figure (a) shows the `ImageRegions`, of which there are none.



(a) (b)

Figure 3.9: A chart described by a single image region (a) and several text regions (b).



(a) (b) (c)

Figure 3.10: Figure (a) shows a table described by a single text region. Figure (b) shows a table described by multiple text regions, the contents of which can be considered tables themselves. Figure (c) shows a table that is insufficiently described by text regions.

1. Tables that are described by a single text region element. This is the optimal case.

2. Tables that are described by multiple text region elements, themselves contain large enough parts of the table to be considered tables themselves. Each `TextRegion` contains at least one whole row or one whole column of the table.

3. Tables whose elements are included in `TextRegions`, but they are disconnected from each other and do not contain whole rows or columns.

Examples of all three cases are depicted in Figure 3.10. The first two types we will consider valid, because the rectangles, if correctly classified, describe table regions, even if not always in full. Regions of the third type leads to false results, because if the true table area is only described by individual `TextRegion` rectangles for single or small groups of table elements, their semantic connection as parts of a table is lost. 45 out of the 633 total table regions (7.11 %) in the inspected dataset belong to this third class. However, it may be worth noting that 37 of them occurred on only 3 pages in the same newspaper, containing tables with financial data. Therefore, if we only look at the remaining pages, the ratio of "bad" `TextRegions` on tables is much smaller (1.3 %).

Layout Elements

Separators and other elements meant to convey the page layout to the reader, like differently colored backgrounds for headings, lines between articles etc. are not described by XML `ImageRegions`, even though they are graphical elements. Even though it would be useful to know the location of such separators, them being excluded from `ImageRegion` elements allows us to effectively ignore them. We can assume that `ImageRegion` bounding boxes only contain actual foreground elements that are part of the newspaper content.

Conclusion

Despite the various flaws of the available `TextRegion` and `ImageRegion` elements revealed by the visual inspection, we still consider them usable, because a large proportion of them correctly describes document components, and it also turned out that only content-related images are annotated as such in the XML, whereas layout elements are not, which makes it possible to use the region locations to find the former while ignoring the latter. In addition, the distinction between text and image elements is, as we have seen, mostly correct, and can, at least in some cases, be used as a basis to localize table and chart regions in the document. Hence this available knowledge contained in the XML data is used in the following sections for building a layout analysis system based on it.

3.1.3 Content Extraction from PDF Files

Using the available images of newspaper pages along with the additional XML information included in the dataset, we have drawn general conclusions about how such low-level element descriptions may be viable for gaining information about the document layout. Since other datasets may not include similar meta-information, but may be available in the PDF format, we take a look at how region information can be extracted from PDF files. A short introduction into how text and raster images are stored in the PDF format will give a general idea of how their positions and sizes can be determined. A look at some example image regions in PDF files will also show that many of the previously discussed errors in the XML rectangles may be the result of the way publishing software stores the images in the documents and how the region bounding boxes are extracted.

PDF files are used to store and display digital documents in a device-independent way. This is achieved by storing all low-level elements contained in the document like characters, lines, shapes, images etc. individually at a fixed relative size and location. Therefore, high-level information about the layout structure like the locations of paragraphs or separators is lost. The syntax of PDF files is explained in detail in the official specification[3]. A PDF file is mainly made up of *objects*, some of which have the special purpose of describing the file and document structure. The *document catalog* object describes the outline and pages of the document. The appearance of individual pages is described by special objects

[3] `http://www.adobe.com/devnet/pdf/pdf_reference.html`

called *content streams*, referenced in each page object. They contain the instructions for drawing the page contents, and are acted upon sequentially. Instructions are used to construct and paint vector graphics or clipping paths, show text, and to include *XObjects*, which can be used, among other purposes, to include raster images. The following snippet from a content stream displays the string "Hello World" on the document:

```
BT
/F1 24 Tf
100 700 Td
(Hello World) Tj
ET
```

The instructions `BT` and `ET` mark the beginning and end of a text object to be drawn. The `Tf` operator takes two parameters: the font, which is specified by a reference to a previously defined font resource (F1) and its size (24). All operators are written in postfix notation, meaning that they are preceded by their arguments. The `Td` command moves the current text position (stored in a 3×3 transformation matrix) to the next line, offset from the current line by the specified parameters (x=100, y=700). The string "Hello World" (strings are always specified by parentheses) is then drawn by the `Tj` command. The following snippet displays a raster image:

```
/GS0 gs
99 0 0 127 82 511 cm
/Im0 Do
```

The `gs` command sets parameters of the graphics state, which are specified in the referenced dictionary object (GS0). The `cm` command modifies the current transformation matrix (CTM), which is applied to all coordinates, meaning that it modifies the coordinate system. As specified in section 8.3.2 of the PDF specification, the CTM is modified by premultiplication of the additional transformation, which is specified by the parameters `a b c d e f`:

$$M' = M_T \times M, \quad M_T = \begin{bmatrix} a & b & 0 \\ c & d & 0 \\ e & f & 1 \end{bmatrix}, \tag{3.1}$$

where M is the old and M' is the new CTM. Coordinates are transformed by a transformation matrix M as follows:

$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \times M. \tag{3.2}$$

The referenced image resource Im0, which refers to an object of type *XObject* containing the image data, is then drawn by the `Do` operator.

In both examples, the positions of the drawn objects can be learned from the transformations specified by the `Td` and `cm` commands respectively. However, one has to keep in mind, that, since all operations in the content stream are performed sequentially, and the transformations are applied to the coordinate systems instead of the objects

themselves, the correct transformation matrix for a specific object always includes all previous transformations applied to objects of the same type. Therefore, the position of a single object of a certain type cannot be determined without also computing the positions of all other objects of the same type.

Furthermore, in order to obtain the *sizes* of the displayed objects, additional steps are necessary. For XObjects, the width and height are stored directly as object entries `Width` and `Height`. However, the position and size of an embedded image do not necessarily describe the visible image in the document, because the visible portion can further be defined by clipping paths. Therefore, if the exact boundaries of an image are to be obtained, these paths must also be taken into account. The extents of text objects have to be computed manually from the character sizes defined by the specified font and, if the text is additionally transformed in any way, the text matrix, which would then be specified by the `Tm` operator before the text is drawn.

Figure 3.11: What appears as a single image is, in this case, actually made up of several smaller image parts, which explains some of the structure of the XML image regions.

We now have an idea of how the positions and sizes of text and image objects can be determined, but they may not correspond to meaningful logical units of text or image regions in the document. Images are sometimes broken up into smaller parts by the publishing software used to create the documents. Figure 3.11 shows an example of a rectangular image, which actually consists of several smaller image fragments placed adjacent to each other. This explains some of the image region boundaries in the XML files of our dataset. They simply describe the borders of all image objects (XObjects) in the PDF document at their respective locations and sizes. Cases with several overlapping XML regions can also be caused by composite images. Figure 3.12 shows an example of what appears to be a single image in the document, but is actually a combination of multiple layers of images with clipping paths. The image in the example is displayed smaller than its real size because of the clipping path applied to it, which may cause extracted image regions to be too large if they describe the bounding box of the full image instead of the clipped one. The example also shows that clipping paths were created to make text visible that would otherwise be covered by the raster image because of its rectangular shape. Such cut-out regions result in separate image regions in the XML data. Therefore, such image regions containing text, which were previously discussed in section 3.1.2, actually represent cut-outs in other images.

Since text objects just represent low-level character strings of certain font styles and sizes at specified locations, there is no notion of a ‘text region’, column or paragraph in PDF. Text lines are nothing more than characters drawn next to each other. Figure 3.13 shows

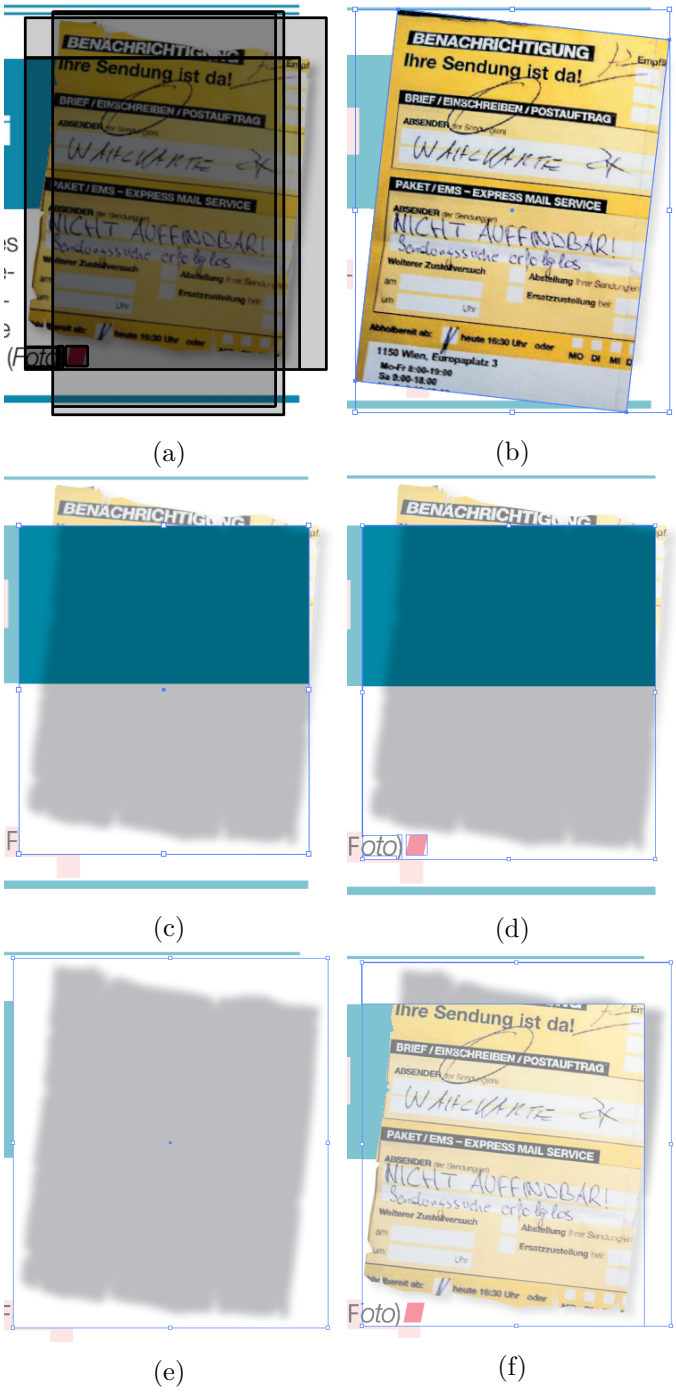


Figure 3.12: The first row shows the XML image regions and the full image without the clipping path. The second row shows one part of the background and the drop shadow of the image. Subfigure (d) shows how a clipping path is used to make text visible which would otherwise be occluded by the rectangular image. The third row shows the drop shadow image and how it is used in combination with a clipping path to complement the background shown in the second row.

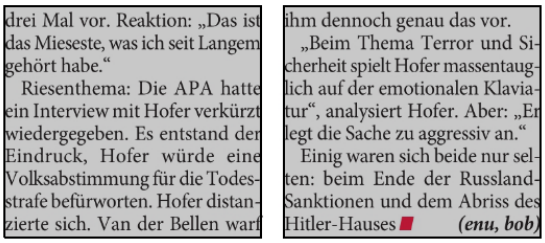

drei Mal vor. Reaktion: „Das ist das Mieseste, was ich seit Langem gehört habe."
Riesenthema: Die APA hatte ein Interview mit Hofer verkürzt wiedergegeben. Es entstand der Eindruck, Hofer würde eine Volksabstimmung für die Todesstrafe befürworten. Hofer distanzierte sich. Van der Bellen warf ihm dennoch genau das vor.
„Beim Thema Terror und Sicherheit spielt Hofer massentauglich auf der emotionalen Klaviatur", analysiert Hofer. Aber: „Er legt die Sache zu aggressiv an."
Einig waren sich beide nur selten: beim Ende der Russland-Sanktionen und dem Abriss des Hitler-Hauses ■ *(enu, bob)*

Figure 3.13: Text, which is spatially separated into two columns, described by XML text regions.

two columns of text, each described as an XML text region. The following snippet shows how the transition from the first text column to the second one is described in the PDF file:

```
10.6704 0 0 11 22.677 324.07 Tm
(strafe bef\374rworten. Hofer distan-)Tj
-0.01 Tc
0.074 Tw
10.78 0 0 11 22.677 312.22 Tm
[(zierte sich. Van der Bellen warf) -10 ( )] TJ
0 Tw
13.639 9.695 Td
(ihm dennoch genau das vor.)Tj
0 Tc
11.109 0 Td
( )Tj
-0.014 Tc
0.113 Tw
-10.32 -1.077 Td
(\204Beim Thema Terror und Si-)Tj
```

The snippet is contained in a text object which starts at the beginning of the first column ("drei Mal vor.") and ends at the bottom of the second column ("(enu, bob)"). The text strings are drawn using `Tj` and `TJ` operators, the latter of which allows to additionally specify the spacing between characters or words. The `Tw` and `Tc` operators are used to define the word and character spacing respectively. The placement of the strings is managed by `Tm` and `Td` operators, which modify the text transformation matrix. The line `13.639 9.695 Td` moves the text position to the specified offset from the current line, where the second column starts. This illustrates how the columns are not specified by syntactic elements in the PDF language, but are merely spatially close portions of text. Since there are XML text regions for each column, it is clear that the provider of the dataset has already applied some text segmentation rules to obtain the rectangular bounding boxes of text blocks. Chao and Fan [CF04] explain how this can be achieved using a bottom-up approach. Their method starts with characters to form words, combines them to text lines, and finally forms text segments by grouping nearby

lines together (see chapter 2).

Besides text and raster images, the PDF format also has the capability to store vector graphics, which are described by *path* objects. The following snippet draws a green rectangle on the page.

```
0.705 0 0.896 0 k
187 699.89 -92 41 re
f
```

The first operator `k` sets the fill color using the arguments `c m y k`. The `re` operator appends a rectangle with the properties `x y width height` to the current path. Finally, the `f` operator fills the current path using the color specified by `k`. Using different operators, much more complex paths can be specified. In section 3.1.2, we have seen that layout separation elements, but also some parts of diagrams are not represented by `ImageRegion`s in the XML files. This can now be explained by path objects contained in the original PDF files not having been extracted as either text or image regions. Therefore, vector graphics are effectively ignored.

We have seen that the positions and sizes of text and image objects can be extracted from PDF files, with the drawback of them being stored in a fragmented way and having no logical relation to each other. In order to obtain a meaningful document layout description, the resulting object boundaries still have to be merged to full document regions according to an algorithm or a set of rules (this is the goal of the method presented in the next section). It has also become clear that the image region rectangles contained in our dataset, which we have inspected in section 3.1.2, are the result of naïve bounding box extraction, while the text regions describe portions of text which have already been processed and determined to be spatially and logically connected.

3.2 Image Region Segmentation

In the previous sections, we have seen that the given XML image regions are located on actual content images in the document, but are often too large or do not cover the image completely, and are directly adjacent or have large intersections between them, sometimes resulting in a single image in the document being described by several image region rectangles. The first step in the layout analysis procedure is to find such clusters of image regions, analyze their content, and produce a single image region bounding box for every contained isolated image object.

This is done by first using the known text regions, which we have determined to be mostly correct (see section 3.1.2), and removing the text contained in them to obtain an image which only contains non-text elements. This page image is then used to find the image regions by locating the remaining foreground components inside clusters of known partly-correct region rectangles. The input image with width W and height H is assumed to be in the RGB format using 8 bit for each channel c.

3.2.1 Text Removal

From the previous visual inspection in section 3.1.2, we know that the available XML image region rectangles roughly describe the true image regions, often containing surrounding document content, which in some cases can be explained by them being inaccurate, in other cases they are caused by the fact that the underlying images are non-rectangular. Using the known XML text bounding boxes, all lines of text can be removed, resulting in a document image containing only image and background pixels, which allows the final localization of images using simple thresholding operations and connected component analysis inside the rectangular XML image regions (see the next section, 3.2.2).

The visual inspection showed that even though text can occur in different colors and sizes in the document image, the XML text regions always contain single paragraphs or headings of a homogeneous font size and color. This knowledge can be used to find text pixels based on the average foreground connected component (CC) properties in every region, considering components as non-text if they deviate from these properties. The results of the text removal step do not have to be perfect, because remaining letters and other imperfections can easily be removed or ignored afterwards. For this reason, the procedure was developed with the main focus on simplicity and runtime performance. Specific values for thresholds etc. have been determined experimentally. The first step is to find a threshold T on the gray-scale image I_{gray} using the Otsu method [Ots79]. The larger of the resulting complementary sets of pixel coordinates is considered the set of probable background pixels

$$M_{pbg} = \underset{M_i \in \{M_{low}, M_{high}\}}{\arg\max} |M_i|\,, \tag{3.3a}$$

$$M_{low} = \{(x, y) \mid I_{gray}(x, y) \leq T\}\,, \tag{3.3b}$$

$$M_{high} = \{(x, y) \mid I_{gray}(x, y) > T\}\,. \tag{3.3c}$$

Histograms are calculated to find the most frequent values v_R, v_G, v_B at the coordinates of M_{pbg} in each color channel I_R, I_G, I_B. The sets of foreground and background pixels are then computed based on the deviation from these values:

$$M_{bg} = \Big\{(x, y) \;\Big|\; |v_c - I_c(x, y)| \leq 50 \;\forall\, c \in \{R, G, B\}\Big\} \tag{3.4}$$

$$M_{fg} = \{(x, y) \notin M_{bg}\}. \tag{3.5}$$

Connected component analysis is performed on the binary foreground image masked by M_{fg} and the areas of all connected components (CCs) are computed. Outliers larger than 4 times the upper quartile $Q_{0.75}$ are considered non-text components. If this threshold is chosen too low, merged letters, which have greater pixel areas, may be falsely added to this set. All other CCs are possible text components. The RGB image is then converted to the L*a*b* color space and k-means clustering with $k = 3$ is performed on the a* and b* channels to find the dominant text color $c_{dom} = (a*, b*)$, which is the cluster center with the most pixels assigned to it. Cluster centers c_i with $\|c_{dom} - c_i\| > 15$ are treated as deviating colors. All CCs that consist of more than 65 % pixels assigned to clusters

(a) (b)

Figure 3.14: Figure (a) shows the text regions in the original image, (b) shows the result after the text removal step. Some components with larger size or different colors like the red parallelograms at the end of paragraphs are not considered text by the algorithm and remain in the image. Text removal fails for the date in the upper-right corner because of the separator lines below it, which connect all the letters, making it a single connected component.

of deviating colors are considered non-text components. Afterwards, a binary image B_{text} is created, containing all pixels that are part of the remaining text components. This text mask is then dilated with a circle-shaped structuring element of diameter 3 to take smooth text edges into account, which may have been classified as background in previous steps. The last step is to replace all pixels in I_{RGB} masked with $B_{text,\,dilated}$ with the background color (v_R, v_G, v_B). An example of an image with removed text can be seen in Figure 3.14.

3.2.2 Merging and Separation

Because image regions are often clustered, meaning that they are overlapping or laid out like tiles (see Figure 3.5), the first step is to group together all regions belonging to the same region cluster. Regions are considered part of the same cluster if they have significant overlap or if they are adjacent to each other. Since the regions are rectangular, they are defined by the four values x_{left}, x_{right}, y_{top}, y_{bottom}, which describe the positions of the upper-left and the lower-right corner. All pairs of regions in the image are tested for three criteria:

1. The area of their intersection is greater than 5 % of the area of the smaller of both regions.

2. The horizontal distance between them is $\leq T_{adj}$ and they have vertical overlap > 0.

(a) The original XML regions.

(b) The result of the image region segmentation.

Figure 3.15: After the partly incorrect image regions have been recalculated, every image in the result shown in (b) has a single region bounding box. Because of its background color, the whole box on the left side, including the text, was detected as an image/chart region.

3. The vertical distance between them is $\leq T_{adj}$ and they have horizontal overlap > 0.

T_{adj} is an adjacency tolerance, which we set at 2 px (regions count as directly adjacent if their distance is $\leq T_{adj}$). If one of these criteria is fulfilled, they belong to the same region cluster. The result is a set of region sets, representing the region clusters, including possible clusters of only one isolated region. Algorithm 3.1 shows how the sets of clustered regions can be found by first finding pairs of clustered regions using the specified criteria and then merging them together if they contain at least one common region.

The next step is to separate these regions where background-color segments run through them. Since the document foreground must be recognized by the reader, it can be assumed that it is clearly separated from the background in terms of brightness. Also, because of the spacing in documents, it can be assumed that the total number of foreground pixels is always less than the number of background pixels. Therefore, the background gray-value v_{bg} is determined as the most frequent value in the histogram of all pixel values of the gray-scale image I_{gray}. The foreground mask is created as

$$B_{fg}(x,y) = \begin{cases} 1 & \text{if } I_{gray}(x,y) \geq v_{bg} + T_{bg} \\ 0 & \text{otherwise} \end{cases} \tag{3.6}$$

if $v_{bg} < 127$, meaning that the background is dark and the foreground is bright, where T_{bg} is an additional tolerance. Otherwise it is created using the inverse operation with the threshold $v_{bg} - T_{bg}$. For T_{bg}, we choose a small value of 10 to account for minor noise and smooth borders.

For every region cluster, a binary image mask $B^*_{cluster}$ of all pixels inside the regions contained in the cluster is created. To account for the tolerance T_{adj} in the region

Algorithm 3.1: FindRegionClusters

Input: list of image region rectangles $R = [r_1, r_2, \dots]$
Output: list of region cluster sets $C = [\{r_i, r_j, \dots\}, \{r_k, r_l, \dots\}, \dots]$

```
1  I ← {0, 1, ..., R.length − 1};        /* stores indices of isolated clusters */
2  C ← [];                                /* stores cluster sets */
   /* pair up regions according to the criteria */
3  for i ← 0 to R.length − 1 do
4  |  for j ← i + 1 to R.length − 1 do
5  |  |  if at least one criterion is fulfilled for (R[i], R[j]) then
6  |  |  |  C.add({R[i], R[j]});
7  |  |  |  I ← I \ {i, j};
8  |  |  end
9  |  end
10 end
11 M ← ∅;                                 /* stores indices of merged clusters */
   /* merge pairs of clusters containing common elements */
12 for k ← 0 to C.length do
13 |  if k ∈ M then
14 |  |  continue;
15 |  end
16 |  isActive ← true;
17 |  while isActive do
18 |  |  isActive ← false;
19 |  |  for l ← k + 1 to C.length do
20 |  |  |  if l ∈ M then
21 |  |  |  |  continue;
22 |  |  |  end
23 |  |  |  if C[k] ∩ C[l] ≠ ∅ then
24 |  |  |  |  C[k] ← C[k] ∪ C[l];
25 |  |  |  |  M ← M ∪ {l};
26 |  |  |  |  isActive ← true;
27 |  |  |  end
28 |  |  end
29 |  end
30 end
   /* remove all merged clusters */
31 for idx ∈ M do
32 |  C.removeAt(idx);
33 end
   /* add isolated regions */
34 for idx ∈ I do
35 |  C.add({R[idx]});
36 end
37 return C;
```

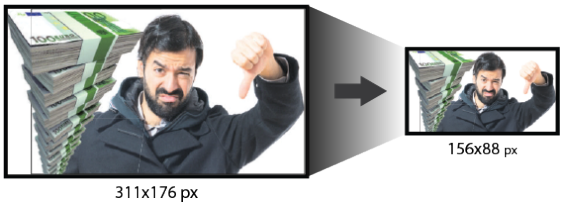


Figure 3.16: The image is downsampled to the scale of the HOG window.

adjacency test, a binary closing with a quadratic structuring element of size $2 \times T_{adj} + 1$ is performed afterwards, resulting in the region cluster mask $B_{cluster}$. The cluster foreground mask is then defined by

$$B_{cluster,fg} = B_{cluster} \wedge B_{fg}. \tag{3.7}$$

Connected component analysis is performed on $B_{cluster,fg}$, and all CCs contained in the bounding box of another CC are discarded (this can happen when CCs contain holes, allowing for nested components). Additionally, CCs with a bounding box of size $(\text{width}, \text{height})$ where $\max(\text{width}, \text{height}) < 0.01 \times \min(W, H)$ are discarded as well. The bounding boxes of the remaining components are the boundaries of the new image regions. An example is depicted in Figure 3.15. It also shows a weakness of this method: because the background value v_{bg} is estimated for the whole image, text regions with deviating background colors may be regarded as images.

3.3 Region Training and Classification

Since the XML dataset described in section 3.1 contains only text and image region rectangles, in order to distinguish between more than these two types, region classification is necessary. Histograms of Oriented Gradients (HOG) [DT05] are used as feature descriptors for classification, which is possible because of the rectangular shape of the document region descriptions. In experiments with different parameters, a window size of 64×64 pixels, cell size of 8×8 pixels and block size of 8×8 cells gave the best results. Using these parameters, each feature descriptor is a real-valued vector of length 576.

For each of the labeled pages in the training set, an image pyramid is computed. The feature computation routine iterates over every region in every page and computes at least one feature per region. There is also the special case of regions that are smaller than the HOG window, which can therefore not be classified or used for training. In order to avoid losing all such regions, it was decided to keep them with their initial type: text or image. It can be argued that very small regions are unlikely to be charts or tables anyway, which usually consist of multiple text and/or image elements and therefore require are larger area. However, there may be fringe cases for which this limitation still produces errors in the resulting region annotation. An additional case are region rectangles which are too small to describe meaningful text, image, chart or table regions. A conceivable example would be thin raster images serving as separator elements. We define a threshold of 4 px and remove all region rectangles with a height or width smaller than this value.

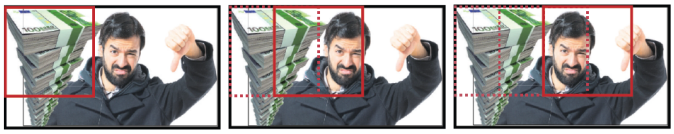


Figure 3.17: The HOG window is slid over the downsampled image.

For every remaining region rectangle, the highest image pyramid level (where level 0 contains the original size) is selected where the downscaled region can still fit the HOG window, as illustrated in Figure 3.16. This step is made to approximately bring all regions to the same scale. The resulting two sides of the region in the down-sampled image are

$$64 \leq dim_1 < 128 \tag{3.8}$$

$$64 \leq dim_2, \tag{3.9}$$

where dim_1 and dim_2 are either width or height respectively. To use as much of the region as possible for feature computation, a sliding HOG window can be used in each dimension with a step size of 32. This allows $\frac{dim_i - 64}{32}$ shifts in each dimension i, resulting in $(1 + \frac{dim_1 - 64}{32}) \times (1 + \frac{dim_2 - 64}{32})$ feature descriptors for the whole region. Figure 3.17 illustrates the sliding HOG window. After all features for the labeled regions contained in the subset of the dataset selected for training have been computed, they can be used to train a random forest model.

For the classification, the feature descriptors are computed the same way for all known regions in the test set. However, the classifier predicts one class label per feature descriptor, which would result in more than one label per region. For such cases, we can take advantage of the fact that the random forest classifier consists of multiple bagged trees. For every feature descriptor, each tree in the forest votes for one of the four classes. The final prediction of the random forest for the feature is the class with the most votes from all trees. In order to get one decision for all feature descriptors of the region, we add all votes of the bagged trees for every feature descriptor. The class with the most overall votes is the final class prediction for the region.

3.4 Newspaper Layout Analysis System

The complete layout analysis system consists of all the previously described steps. It starts with the input of a newspaper image along with an XML file containing extracted rectangular regions. The XML file is then parsed and converted, after which the main layout analysis is performed. Since the positions and sizes of the text regions are left unchanged, they can immediately be classified as either table or text regions. The image region rectangles first have to be segmented, after which the resulting regions are classified as images or charts. Finally, the result is written to an output PAGE XML file. The complete process is illustrated in Figure 3.18.

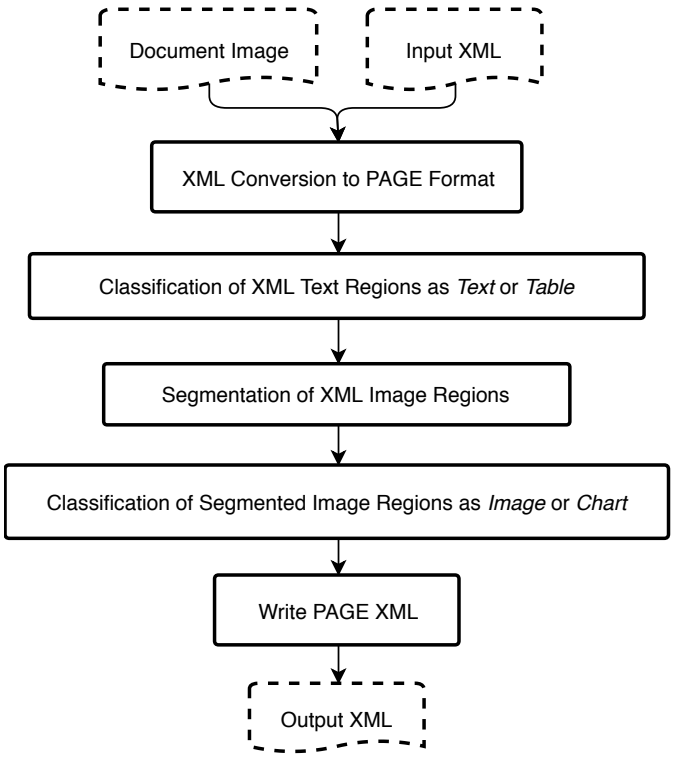


Figure 3.18: Illustration of the complete layout analysis system.

3.5 Extension for Non-Rectangular Regions

Because of our limited ground-truth information, the presented layout analysis technique mainly focuses on retrieving rectangular document regions. However, the method can easily be extended to produce polygonal or pixel-precise non-rectangular regions.

In the image segmentation step, the final image rectangles are obtained by finding the bounding boxes of foreground connected components. Without this last step, the foreground components, if their holes are filled, represent the exact image regions, which could then be approximated by a polygon or be stored as pixel masks.

In the text removal step, image processing operations are applied to find a pixel mask of all letters in each text region. Again, the main problem is already solved. For each text region, the outline of all detected text pixels can be approximated by a polygon representation.

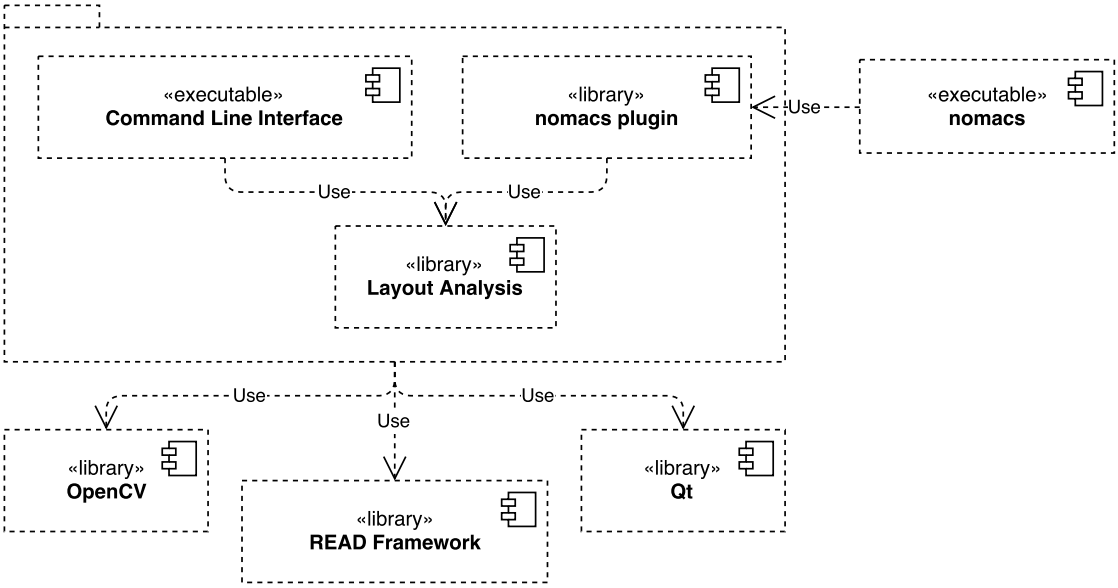


Figure 3.19: Implementation components.

3.6 Implementation

The method described in 3.2 and 3.3 has been implemented in the C++ programming language using the Qt library[4], OpenCV[5] and the READ framework[6], which supports the use of PAGE XML files and provides a number of document analysis algorithms and utility functions.

Figure 3.19 shows a diagram of all program components which have been developed and implemented. The main functionality is contained in a library, which can be used via a command-line interface or through a plugin for the nomacs image viewer[7], which allows for direct visualization of results, contains batch processing functionality and can be extended by additional plugins. One of these plugins is the PageVis READ module[8], which can visualize PAGE image regions and was used to generate many of the Figures in this thesis.

[4] https://www.qt.io/
[5] https://opencv.org/
[6] https://github.com/TUWien/ReadFramework/
[7] https://nomacs.org/
[8] https://github.com/TUWien/ReadModules/

CHAPTER 4

Evaluation

In order to evaluate the viability of the proposed page segmentation and region classification method, the correctness of the resulting region descriptions has to be measured. These results then provide insights about the performance of the technique for different cases and for each region class, and make it comparable to other layout analysis methods. The correctness is measured by using human-generated data as ground truth, to which the resulting data is then compared. First, in section 4.1, the region classification alone is evaluated by only considering the content classes which are assigned to each region rectangle by our algorithm. Confusion matrices are used to display how often each ground-truth class was assigned correctly or to a different class, revealing the weaknesses of the classifier in distinguishing each pair of classes. The second kind of evaluation, described in section 4.2, concerns the correctness of the image region segmentation. This involves the measurement of all pixel overlaps between ground-truth regions and computed ones, but it also includes counting all spatial matches between ground-truth and segmented regions. The third step is to measure the performance of the whole layout analysis system by evaluating the correctness of all produced region rectangles for each class. The results are shown in section 4.3. Finally, in section 4.4, a second method is evaluated on our ground truth data in order to compare the results to the proposed technique.

4.1 Region Classification

Since the converted XML files only contain text and image regions, manual labeling was performed to add information about chart and table regions. For this purpose, region rectangles have been added to 6211 pages of the selected publications. The annotation resulted in 891 pages containing at least one table or chart. Where tables were exactly described by XML text regions, the text regions have been removed and replaced with table regions. XML image regions describing charts have been treated accordingly. These

pages represent the final classification dataset: 891 page images with corresponding PAGE XML files, containing rectangular region descriptions of 15870 text, 8187 image, 299 chart and 633 table regions in total, with the image regions being incomplete, redundant or false in some cases, as is known from section 3.1.2.

The classification is evaluated using 70 % of the dataset for training and the remaining pages as test set (624 training pages, 267 test pages). In the training step, feature descriptors are computed for all original text and image regions from the source XML to be used as text and image features. The manually labeled chart and table regions are used to compute the respective chart and table feature descriptors. To avoid an unbalanced training set, the number of feature descriptors for each class is also reduced to the minimum class size. The random forest predictor included in OpenCV 3.3 is used with a tree depth limit of 25 and a maximum number of trees of 150. For the evaluation, the number of classified regions for each class is reduced to the minimum class size, in order to better be able to compare the results. The following table shows the confusions of all four classes (351 regions per class), where the rows represent the true labels and columns represent the predictions:

	Text	Image	Chart	Table	Re.
Text	320	23	3	5	0.91
Image	27	306	18	0	0.87
Chart	14	34	285	18	0.81
Table	29	10	6	306	0.87
Pr.	0.82	0.82	0.91	0.93	

The average recall and precision both are 0.87, the rate of wrongly classified regions is 0.13. However, in the final layout analysis system, only the distinction between text/table and image/chart regions is needed. When we train models for this purpose, the results for the text/table classification (1084 regions per class) are

	Text	Table	Re.
Text	1048	36	0.97
Table	63	1021	0.94
Pr.	0.94	0.97	

with an average recall of 0.95 and an average precision of 0.96. Out of 2168 regions, 99 are wrongly classified, resulting in an overall error rate of 0.05. The results for the image/chart classification (351 regions per class) are

	Image	Chart	Re.
Image	328	23	0.93
Chart	50	301	0.96
Pr.	0.87	0.93	

(a) The "Krone Service" graphic overlaps with the photograph, even though they are separate images.

(b) The segmentation method outputs three image regions. This could be considered correct, because the three image parts are separated by whitespace, but they also belong to the same advertisement.

Figure 4.1: Two examples of the problem of deciding whether images are connected or separated.

with an average recall and precision of 0.9. Out of 702 regions, 73 are wrongly classified, resulting in an overall error rate of 0.1.

4.2 Segmentation

Since segmented region rectangles are either classified as image or chart regions, the segmentation step is evaluated by comparing the region rectangles resulting from the segmentation to the ground-truth image and chart regions. Because the exact boundaries of images are dependent on the subjective interpretation of the viewer, the ground truth generation poses a challenge:

- Regions with different meanings or content can overlap each other, which therefore could be interpreted as either one single region or several separate ones.
- Images containing segments of the document background color (usually whitespace) could be interpreted as either one or multiple separated images.
- Advertisements may consist of images and text elements, where it can be ambiguous if the text can be considered a part of the image.
- Charts are often combined or overlapped with photographs or other depictions, raising the question if only the image or the whole combined region should be considered an image.

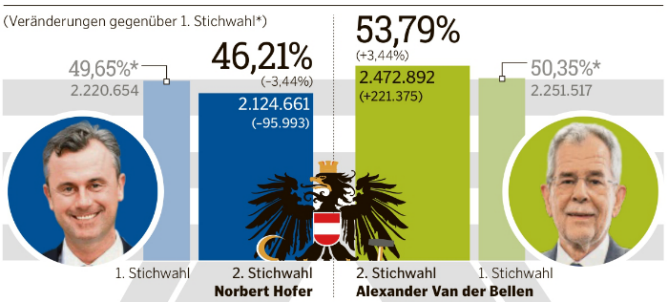


Figure 4.2: The depicted region contains a bar plot, combined with the coat of arms and two photographs. Can the whole region be considered a single image, is it a combination of image, chart and vector graphics elements, or something else?

Figure 4.1a shows an example of the first case: a photograph is overlapped by a graphic. Both images belong to the corresponding article, but serve different purposes. The photograph is relevant to the topic of the article, while the graphic is intended to highlight the article as being especially informative or useful to the reader. Therefore, they are clearly two separate images – while being spatially not separated at all. An example of the second case can be seen in Figure 4.1b, which shows image region annotations resulting from the segmentation of the algorithm. The three images are actually part of a single advertisement, but the automatic segmentation could also be considered correct, since the advertisement consists of three objects separated by whitespace (background). An example of the fourth case can be seen in Figure 4.2.

In order to evaluate the image region segmentation, all depicted images in 70 newspaper pages (10 of each publication in the dataset) have been manually annotated. All selected newspaper pages are also contained in the dataset of 891 pages described in section 4.1, meaning that as a result, they contain manually labeled image, chart and table regions, which will be of use in the following evaluation of the complete layout analysis system. Since they will also be used for the evaluation of the complete system in section 4.3, we use only pages which are not contained in the set used to train the classifier. The purpose of the segmentation is to use the partly correct image region rectangles to find the boundaries of document regions containing one type of graphical content. For this reason, overlapping images like the ones shown in Figure 4.1a (graphics and photographs are both considered subcategories of images) have been annotated as single image regions. Images containing background color (whitespace) between them have been annotated as a single image if they are spatially close and clearly belong together, or if they are somehow connected, e.g. by a border around them. Photographs contained in charts are considered images. As in all human ground truth generation, there is a subjective element to these decisions.

The correctness of the computed region rectangles is determined by comparing them to the ground truth regions. Two kinds of measurements are conducted. First, the regions are compared solely based on the amount of area of ground-truth regions that is

covered by computed regions, and vice versa. Second, based on the overlapping areas, the attempt is made to find matching computed and ground truth region rectangles. Each page contains a set of segmented region rectangles S and a complementary set of ground-truth region rectangles T, with $S \cap T = \varnothing$. Regions $r \in (S \cup T)$ are sets of document pixels, forming a rectangular shape. However, unions and intersections of such rectangles may produce arbitrary pixel sets. The number of pixels contained in a region r is its area(r). The evaluation includes measurements on the sets of all computed and ground-truth region rectangles in all n pages, which are defined by:

$$R_S = \bigcup_{i=0}^{n} S_i, \tag{4.1}$$

$$R_T = \bigcup_{i=0}^{n} T_i. \tag{4.2}$$

As previously discussed in section 3.1.2, many XML image rectangles contain parts of charts, but not all graphical elements of charts are fully described by XML image rectangles, and there rarely exists a single XML rectangle describing the bounding box of a complete chart document region. Because of the simple nature of the segmentation method, it can be assumed that this leads to many charts not being properly described by segmented region rectangles as a result. In order to reveal the influence of chart regions on the evaluation results, all measurements of the segmented image regions are taken twice: once with all ground-truth image and chart regions, and once using only the ground-truth image regions. For the first case, the set T for each page contains all manually annotated image and chart region rectangles, and for the second case, T only contains the manually annotated image regions.

4.2.1 Measuring the Amounts of Overlap

The first kind of measurements are based on the amount of area coverage between computed and ground-truth regions contained in the sets S and T for each page. Since computed regions are compared to ground-truth regions and vice versa, we define the 'opposite' set of a region in a page as the complementary set to the one containing it:

$$\mathrm{Opp}(r) = \begin{cases} S & \text{if } r \in T \\ T & \text{if } r \in S \end{cases}, \tag{4.3}$$

where $r \in S \vee r \in T$. Each ground-truth or computed region r is covered by regions of the opposite set by an amount

$$\mathrm{cov}(r) = \mathrm{area}\left(\bigcup_{r_o \in \mathrm{Opp}(r)} r_o \cap r \right). \tag{4.4}$$

The *coverage ratio* $\frac{\mathrm{cov}(r)}{\mathrm{area}(r)}$ defines the covered proportion of a region. For ground-truth regions, this ratio can be interpreted as the region-wise Recall_r (proportion of ground-truth area covered by computed regions), whereas for computed regions, it is equivalent

to the region-wise Precision$_r$ (proportion of computed region area matching the ground truth). The overall recall and precision values are computed similarly:

$$\text{Recall} = \frac{\sum_{r \in R_T} \text{cov}(r)}{\sum_{r \in R_T} \text{area}(r)}, \tag{4.5a}$$

$$\text{Precision} = \frac{\sum_{r \in R_S} \text{cov}(r)}{\sum_{r \in R_S} \text{area}(r)}. \tag{4.5b}$$

Since there are pages with special layouts (e.g. whole pages covered by a single background image), it may also be interesting to know the precision and recall results for all individual pages:

$$\text{Recall}_i = \frac{\sum_{r \in T_i} \text{cov}(r)}{\sum_{r \in T_i} \text{area}(r)}, \tag{4.6a}$$

$$\text{Precision}_i = \frac{\sum_{r \in S_i} \text{cov}(r)}{\sum_{r \in S_i} \text{area}(r)}. \tag{4.6b}$$

First, the measurements are taken with the ground-truth set R_T containing all manually annotated image and chart regions, which equals a total of 389 ground-truth regions. The segmentation produced 461 region rectangles (R_S). Table 4.1 shows the evaluated results according to the above definitions. It includes the overall F_1 score, which is the harmonic mean of the precision and recall.

	Rec.$_r$	Pre.$_r$
μ	0.67	0.63
σ	0.45	0.46

(a) Region-wise

	Rec.$_i$	Pre.$_i$
μ	0.74	0.95
σ	0.3	0.09

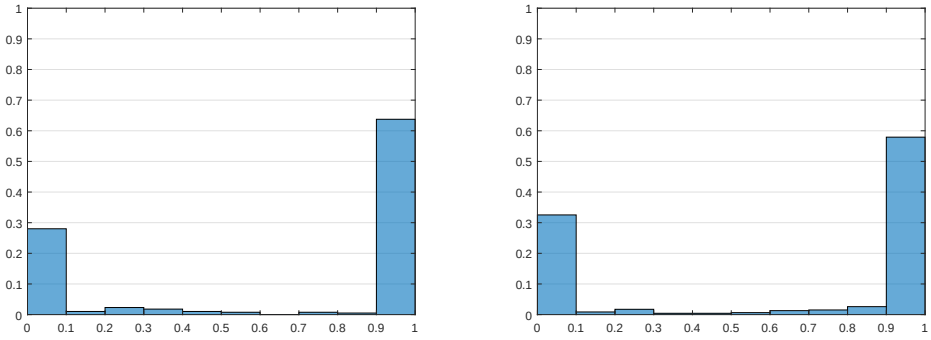
(b) Page-wise

Rec.	Pre.	F_1
0.77	0.94	0.85

(c) Overall

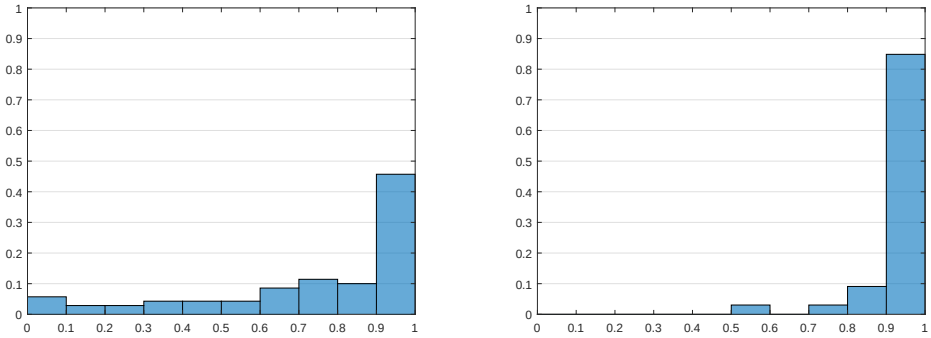
Table 4.1: Area-based recall and precision values.

Since the standard deviation σ is quite large, it might be interesting to look at the distribution of the values. The histogram in Figure 4.3, containing 10 bins, shows the region coverage ratios (the Recall$_r$ and Precision$_r$ values). Most regions are either completely covered by regions of the opposite set or not covered at all. 28 % of ground-truth regions have a coverage ratio of < 0.1, 64 % are covered by a proportion ≥ 0.9 and all others are somewhere in between. 33 % of computed regions have a coverage ratio of < 0.1, 58 % are covered by a proportion ≥ 0.9. Figure 4.4 shows the distributions of the recall and precision values of all pages, which for the precision values are only 66, because for the other four, the segmentation did not produce any region rectangles. The values are calculated according to Equation 4.6. While 85 % of the documents have a precision ≥ 0.9, only 46 % have a recall ≥ 0.9. The distribution of the recall values shows that for the rest of the documents, there is more than 10 % of ground-truth area which is not covered by computed regions. The high precision values imply that for most documents (85 %), a high proportion of the computed regions (≥ 0.9) is correct. In conclusion, the



(a) Recall$_r$ values (b) Precision$_r$ values

Figure 4.3: The distributions of the coverage ratios of ground-truth (a) and computed (b) regions (using images and charts as ground-truth data).



(a) Recall$_i$ values (b) Precision$_i$ values

Figure 4.4: The distributions of the recall (a) and precision (b) values of all 70 and 66 pages respectively (using images and charts as ground-truth data).

precision values are satisfying, but large parts of the ground-truth regions are not covered by computed regions.

The second set of measurements is taken with only image regions (and no charts) contained in the ground-truth set R_T, which reduces its size from 389 to 269. The set of segmented regions R_S stays unchanged (containing 461 segmented region rectangles). The resulting values and how they have changed are shown in Table 4.2.

Figure 4.5 again shows the distribution of the coverage ratios of all regions. As before, most regions are either almost fully (≥ 0.9) covered or not covered at all (< 0.1). Without the chart regions in the ground truth set, the proportion of almost fully covered ground-truth regions (84 %) is higher than with them included (64 %), and the amount of regions with a coverage ratio < 0.1 is 11 %, compared to 28 % before. On the other hand,

	Rec.$_r$	Pre.$_r$
μ	0.86 ↑	0.55 ↓
σ	0.33 ↓	0.48 ↑

(a) Region-wise

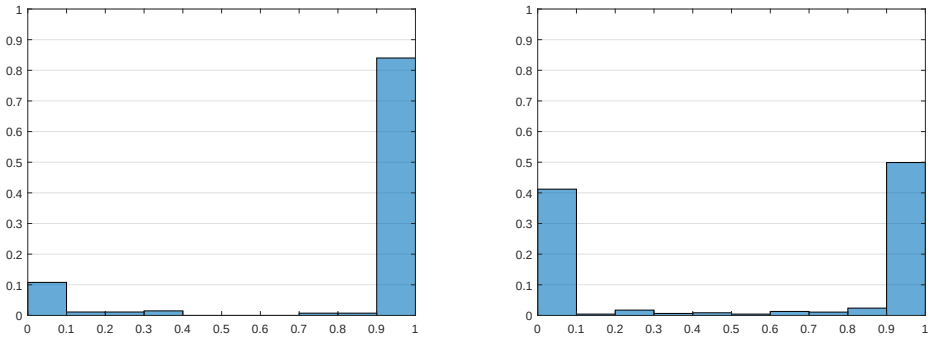
	Rec.$_i$	Pre.$_i$
μ	0.84 ↑	0.91 ↓
σ	0.28 ↓	0.17 ↑

(b) Page-wise

Rec.	Pre.	F_1
0.85 ↑	0.9 ↓	0.88 ↑

(c) Overall

Table 4.2: Area-based recall and precision values (with the GT set containing image regions only).



(a) Recall$_r$ values

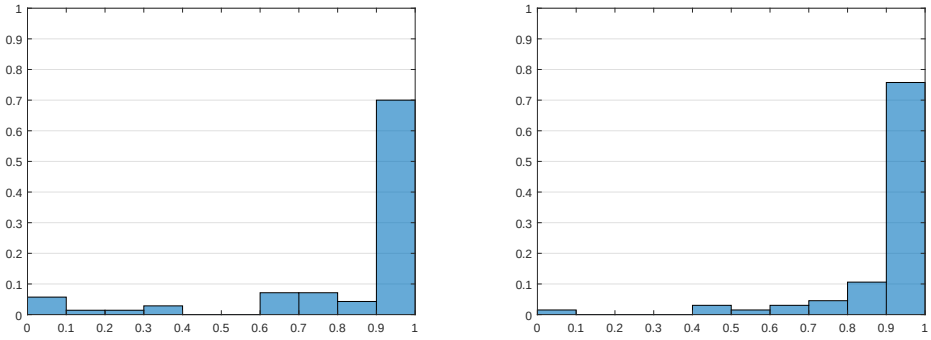
(b) Precision$_r$ values

Figure 4.5: The distributions of the coverage ratios of ground-truth (a) and computed (b) regions (using only images as ground-truth data).

the proportion of almost fully covered computed regions has gone down from 58 % to 48 %, while the bin < 0.1 now contains 41 % of regions, compared to 33 % from before. Accordingly, the overall recall value has increased from 0.77 to 0.85, and the overall precision has slightly decreased from 0.94 to 0.9. The distributions of page-wise recall and precision values can be seen in Figure 4.6. While the proportion of pages with precision values ≥ 0.9 has decreased from 85 % to 76 %, the proportion of recall values ≥ 0.9 has increased significantly from 46 % to 70 %. The improved recall values confirm that for many charts in the newspaper pages, there are insufficient XML image region rectangles which could be segmented to cover the chart region. The decreased precision values are explained by the fact that often, there exist XML image region rectangles for parts of chart regions, which are counted as false in this case, because the ground-truth set does not contain the corresponding chart regions.

4.2.2 Counting the Number of Matching Regions

In order to determine the correctness of the computed regions, it is not enough to measure the amount of coverage of ground-truth regions by computed rectangles. Computed rectangles may describe completely different regions while containing smaller ground-truth regions by coincidence, or they may intersect ground-truth regions by coincidence. Furthermore, the histograms of region intersections in the previous evaluation (Figures



(a) Recall$_i$ values (b) Precision$_r$ values

Figure 4.6: The distributions of the recall (a) and precision (b) values of all 70 and 66 pages respectively (using only images as ground-truth data).

4.3 and 4.5) have shown that most regions are either intersected by at least 90 % or by less than 10 %. The goal of the following evaluation is to find *matching* computed and ground-truth region rectangles describing the same image regions.

A region $r_1 \in (S \cup T)$ is defined to 'fit' another region $r_2 \in (S \cup T)$ if their Jaccard index is greater than a threshold $1 - tol$ (tol is the tolerance):

$$\text{fits}(r_1, r_2) = \begin{cases} true & \text{if } J(r_1, r_2) > (1 - tol) \\ false & \text{otherwise} \end{cases}, \tag{4.7}$$

$$J(r_1, r_2) = \frac{\text{area}(r_1 \cap r_2)}{\text{area}(r_1 \cup r_2)}. \tag{4.8}$$

This defines the set of fitting region rectangles

$$F = \{r \in (S \cup T) \mid \exists\, r_o \in \text{Opp}(r) : \text{fits}(r, r_o)\}, \tag{4.9a}$$

$$F_S = F \cap S, \tag{4.9b}$$

$$F_T = F \cap T. \tag{4.9c}$$

Using this criterion, the number of correct computed regions can be determined by counting all instances where $\text{fits}(r_1, r_2) = true$. To account for ambiguous cases where regions could be interpreted as a single image region or as multiple neighboring image regions, each region is also tested for whether it is 'covered' by the union of other regions:

$$C = \left\{ r \in (S \cup T) \setminus F \;\middle|\; \exists\, O \subseteq \text{Opp}(r) : \text{fits}\left(r, \bigcup O\right) \right\}, \tag{4.10a}$$

$$C_S = C \cap S, \tag{4.10b}$$

$$C_T = C \cap T. \tag{4.10c}$$

Algorithm 4.1: IsRegionCovered

Input: a region r, a set $X \subseteq \text{Opp}(r)$ intersecting r, the fitting tolerance tol
Output: $true$ if the region is covered, $false$ otherwise

```
1  if |X| ≤ 1 then
2  |  return false;
3  end
4  S ← [X];        /* contains the candidate sets of intersecting regions */
5  hasBranch ← true;
6  while hasBranch do
7  |  hasBranch ← false;
8  |  S_new ← [];
9  |  for X_c ∈ S do
   |  |  /* test if r could fit X_c or subsets of it                    */
10 |  |  if area((⋃X_c) ∩ r) > (1 − tol) × area(r) then
   |  |  |  /* test if r fits the candidate set X_c = {r_0, r_1, ...}   */
11 |  |  |  if fits(r, ⋃X_c) then
12 |  |  |  |  return true;
   |  |  |  /* otherwise add subsets of X_c to S_new, which represent the
   |  |  |     branches visited in the next iteration                   */
13 |  |  |  else if |X_c| > 2 then
14 |  |  |  |  hasBranch ← true;
15 |  |  |  |  for r_c ∈ X_c do
16 |  |  |  |  |  S_new.add(X_c \ r_c);
17 |  |  |  |  end
18 |  |  |  end
19 |  |  end
20 |  end
21 |  S ← S_new;
22 end
23 return false;
```

This concept of covered regions is different, but related to the previously defined region coverage function $\mathrm{cov}(r)$: instead of just measuring the amount of overlap of other regions over r, we look for sets of region rectangles whose union area 'fits' (according to the fits function) the area of r. The idea is similar to the measures used by Clausner *et al.* [CPA11b], but less complex, since we do not use additional weights and are only interested in the number of fully 'covered' regions. Using their terms, the set C_S corresponds to the set of 'merges' and C_T represents the set of 'splits'.

Searching for a set of fitting regions in $\mathrm{Opp}(r)$ requires taking all possible subsets into account, which is equivalent to iterating over the power set $2^{\mathrm{Opp}(r)}$. The complexity of the search can be reduced by only considering the subset of intersecting regions $X = \{r_o \in \mathrm{Opp}(r) \mid r_o \cap r \neq \varnothing\}$ and by using a branch-and-bound approach. Algorithm 4.1 shows a simple solution relying on the knowledge that for $O \subseteq \mathrm{Opp}(r)$,

$$J(r, \bigcup O) = \frac{\mathrm{area}(r \cap (\bigcup O))}{\mathrm{area}(r \cup (\bigcup O))} \leq \frac{\mathrm{area}(r \cap (\bigcup O))}{\mathrm{area}(r)}, \tag{4.11a}$$

$$\exists\, O^* \subseteq O : \mathrm{fits}(r, \bigcup O^*) \implies \frac{\mathrm{area}(r \cap (\bigcup O))}{\mathrm{area}(r)} > (1 - tol). \tag{4.11b}$$

Finally, the set of 'matched' regions contains all regions which are either fitting or covered:

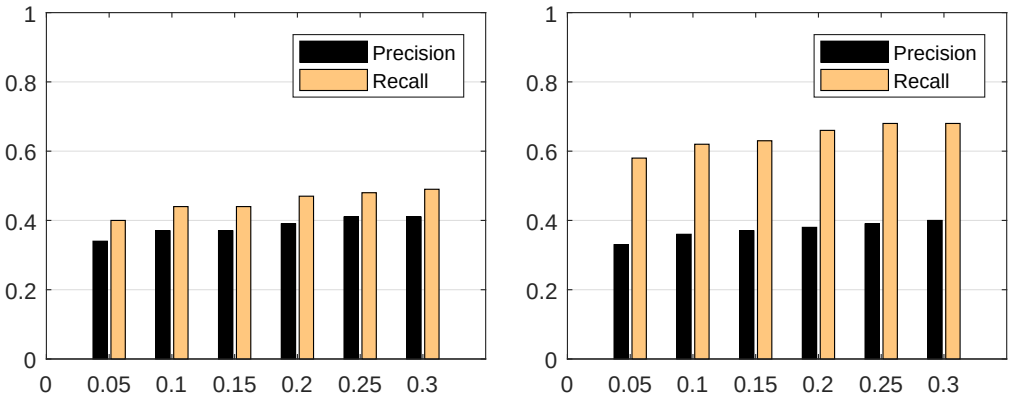
$$M = F \cup C, \tag{4.12a}$$

$$M_S = M \cap S, \tag{4.12b}$$

$$M_T = M \cap T. \tag{4.12c}$$

The ratios of matched regions $|M_T|/|T|$ and $|M_S|/|S|$ can be interpreted as the match-based recall and precision values of a page. Based on these definitions, the results are computed for different values of T. As in the previous section, all measurements are taken twice: once for the ground-truth set containing image and chart regions, and once with it containing only image regions. The first set of measurements is made on the ground-truth set R_T containing all image and chart regions, which is a total of 389 regions. The computed set R_S contains 461 segmented region rectangles. The following table shows the results summed over all pages in the dataset, where $\mathrm{Rec.}_M$ and $\mathrm{Pre.}_M$ are the overall match-based recall and precision values $\sum |M_T| / \sum |T|$ and $\sum |M_S| / \sum |S|$. A graphical representation of the results can be seen in Figure 4.7a.

tol	$\sum \lvert F_T \rvert$	$\sum \lvert F_S \rvert$	$\sum \lvert C_T \rvert$	$\sum \lvert C_S \rvert$	**Rec.**$_M$	**Pre.**$_M$	F_1
0.05	156	156	1	0	0.4	0.34	0.37
0.1	169	169	1	0	0.44	0.37	0.4
0.15	172	172	1	0	0.44	0.37	0.41
0.2	181	181	1	0	0.47	0.39	0.43
0.25	187	186	1	2	0.48	0.41	0.44
0.3	188	187	3	2	0.49	0.41	0.45



(a) With images and charts in the GT set. (b) With only images in the GT set.

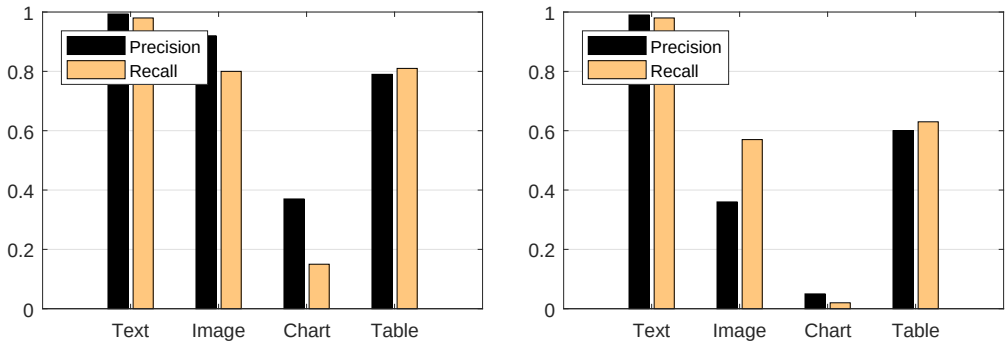
Figure 4.7: Match-based precision and recall values of the segmentation evaluation for different tolerance values.

The numbers of matched regions $|M_T|$ and $|M_S|$ are $|F_T|+|C_T|$ and $|F_S|+|C_S|$ respectively. As one would expect, the rate of matched ground-truth regions increases with a higher tolerance value.

For the second set of measurements, the 461 segmented region rectangles ($|R_S|$) are measured on the ground-truth set R_T containing only image regions, with a cardinality of 269.

tol	$\sum\|F_T\|$	$\sum\|F_S\|$	$\sum\|C_T\|$	$\sum\|C_S\|$	**Rec.**$_M$	**Pre.**$_M$	F_1
0.05	154	154	1	0	0.58	0.33	0.42
0.1	166	166	1	0	0.62	0.36	0.46
0.15	169	169	1	0	0.63	0.37	0.46
0.2	177	177	1	0	0.66	0.38	0.49
0.25	181	181	1	1	0.68	0.39	0.5
0.3	182	182	2	1	0.68	0.4	0.5

The precision and recall values are also displayed in Figure 4.7b. The fact that the absolute numbers of fitting and matched regions $|F_T|$, $|F_S|$, $|M_T|$ and $|M_S|$ are almost identical to the results of the previous measurements shows that correctly segmented regions are almost exclusively of the image type (instead of chart). Considering the almost unchanged ratios of matched computed regions (average decrease: 0.01), the increase in recall of 0.19 on average is mainly caused by the reduced size of the ground-truth set (30.8 % smaller). Therefore, these results confirm what the histograms of region intersections in Figures 4.3 and 4.5 already indicate.



(a) Area-based values. (b) Match-based values ($tol = 0.15$).

Figure 4.8: Overall precision and recall values of all classes.

4.3 Complete Layout Analysis System

In addition to the evaluation of the classification and segmentation steps, a final evaluation of the complete system is conducted on the dataset of 70 newspaper page image files described in section 4.2. The input of the layout analysis system are the images contained in the dataset, along with the included XML data containing text and image region rectangles as described in section 3.1. The results are PAGE XML files containing segmented and classified text, image, chart and table regions. Since the images, charts and tables in the 70 selected pages have already been annotated for the classification and segmentation evaluations, the only major content type missing in the ground-truth data are the text regions. But, since the text regions are mostly correct (as described in section 3.1.2), we reuse them as ground-truth data, because a full manual annotation of all text regions would require a lot of additional effort, seeing as they are the most frequent of all region types (4425 regions). However, this does not mean that all the text regions of the input set and those of the ground-truth set are identical, since many of them have been replaced by table regions in the manual annotation, as described in section 4.1. Like in the segmentation evaluation in section 4.2, we are interested in the amount of pixel overlap between ground-truth and computed regions and in the number of regions describing the same document components. The difference is that now, the measurements have to be taken for each of the four region classes. Figure 4.8 shows a graphical comparison of the overall area- and match-based precision and recall values of all classes.

4.3.1 Text Regions

The ground truth set R_T for the text class contains 4425 text regions, while 4375 regions are segmented and classified as text regions (R_S). 98 % of their area is covered by text regions resulting from the layout analysis system (which is the overall area-based recall), and of those, 99.34 % are covered by ground-truth regions (which is the overall area-based

precision). The resulting area-based F_1 score is 0.98. Both values are calculated as previously defined in Equation 4.5.

tol	$\sum \lvert F_T \rvert$	$\sum \lvert F_S \rvert$	$\sum \lvert C_T \rvert$	$\sum \lvert C_S \rvert$	**Rec.**$_M$	**Pre.**$_M$	F_1
0.05	4346	4346	0	0	0.98	0.99	0.99
0.1	4346	4346	0	0	0.98	0.99	0.99
0.15	4346	4346	0	0	0.98	0.99	0.99
0.2	4346	4346	0	0	0.98	0.99	0.99
0.25	4346	4346	0	0	0.98	0.99	0.99
0.3	4346	4346	0	0	0.98	0.99	0.99

The high precision and recall values are not surprising, because the set of resulting text regions is a subset of the set of ground-truth text regions, as previously mentioned. This means that since no segmentation is performed on these regions, the results actually only represent the classification accuracy and all errors are caused by wrongful classifications of text regions as tables.

4.3.2 Image Regions

The ground truth set R_T contains 269 image regions, while the set of computed regions R_S has a size of 422. The overall area-based recall and precision values are 80 % and 92 % respectively, resulting in an area-based F_1 score of 0.86.

tol	$\sum \lvert F_T \rvert$	$\sum \lvert F_S \rvert$	$\sum \lvert C_T \rvert$	$\sum \lvert C_S \rvert$	**Rec.**$_M$	**Pre.**$_M$	F_1
0.05	141	141	1	0	0.53	0.33	0.41
0.1	152	152	1	0	0.57	0.36	0.44
0.15	154	154	1	0	0.58	0.36	0.45
0.2	162	162	1	0	0.61	0.38	0.47
0.25	166	166	1	1	0.62	0.4	0.48
0.3	167	167	2	1	0.63	0.4	0.49

The lower precision and recall values compared to the area-based results can be explained by the segmentation performance, as we have already seen in section 4.2. Even though most of the areas of computed image regions lie on ground-truth areas (92 %), the region boundaries often don't match the ground-truth rectangles.

4.3.3 Chart Regions

The ground truth set R_T contains 120 chart regions. The set of segmented chart regions R_S has a cardinality of 39. The overall area-based recall is 15 %, whereas the overall area-based precision value is 37 %. The resulting area-based F_1 score is 0.21.

tol	$\sum \lvert F_T \rvert$	$\sum \lvert F_S \rvert$	$\sum \lvert C_T \rvert$	$\sum \lvert C_S \rvert$	**Rec.**$_M$	**Pre.**$_M$	F_1
0.05	1	1	0	0	0.01	0.03	0.01
0.1	2	2	0	0	0.02	0.05	0.03
0.15	2	2	0	0	0.02	0.05	0.03
0.2	3	3	0	0	0.03	0.08	0.04
0.25	4	4	0	0	0.03	0.1	0.05
0.3	4	4	0	0	0.03	0.1	0.05

Here, the main weakness of the method becomes clear. Neither the precision and recall values, nor the number of region rectangle matches are satisfactory. Again, the results confirm the problems discussed in the previous section (4.2). Even though the recall and precision values for class distinction between images and charts are high (0.96 and 0.93), when using the output rectangles of our method, the bulk of the chart areas are not covered by resulting chart regions, because for most charts, only small parts are described by XML image regions. This becomes especially apparent in the low rate of matched regions, which is almost zero even for higher tolerance values.

4.3.4 Table Regions

The ground truth set R_T contains 191 table regions, while there are 200 segmented table regions in R_S. The overall area-based recall and precision values 81 % and 79 % respectively, resulting in an F_1 score of 0.8.

tol	$\sum \lvert F_T \rvert$	$\sum \lvert F_S \rvert$	$\sum \lvert C_T \rvert$	$\sum \lvert C_S \rvert$	**Rec.**$_M$	**Pre.**$_M$	F_1
0.05	120	120	0	0	0.63	0.6	0.61
0.1	120	120	0	0	0.63	0.6	0.61
0.15	120	120	3	0	0.64	0.6	0.62
0.2	120	120	4	0	0.65	0.6	0.62
0.25	121	121	7	0	0.67	0.61	0.64
0.3	121	121	7	0	0.67	0.61	0.64

While 19 % of the overall table area is not identified as such, 79 % of the output table area is correct. The ratios of matched rectangles shows that the positions and sizes of the text regions, which are re-used without further segmentation, are largely correct, almost reaching the area-based values.

4.4 Comparison

As an additional evaluation, a comparison of the proposed method to the layout analysis component of the well-known open-source Tesseract OCR engine[1] developed by Google [Smi07] is conducted. Using the converter developed by PRImA Research[2], it is possible

[1] `https://github.com/tesseract-ocr/tesseract/`

[2] `http://www.primaresearch.org/tools/TesseractOCRToPAGE`

(a) (b)

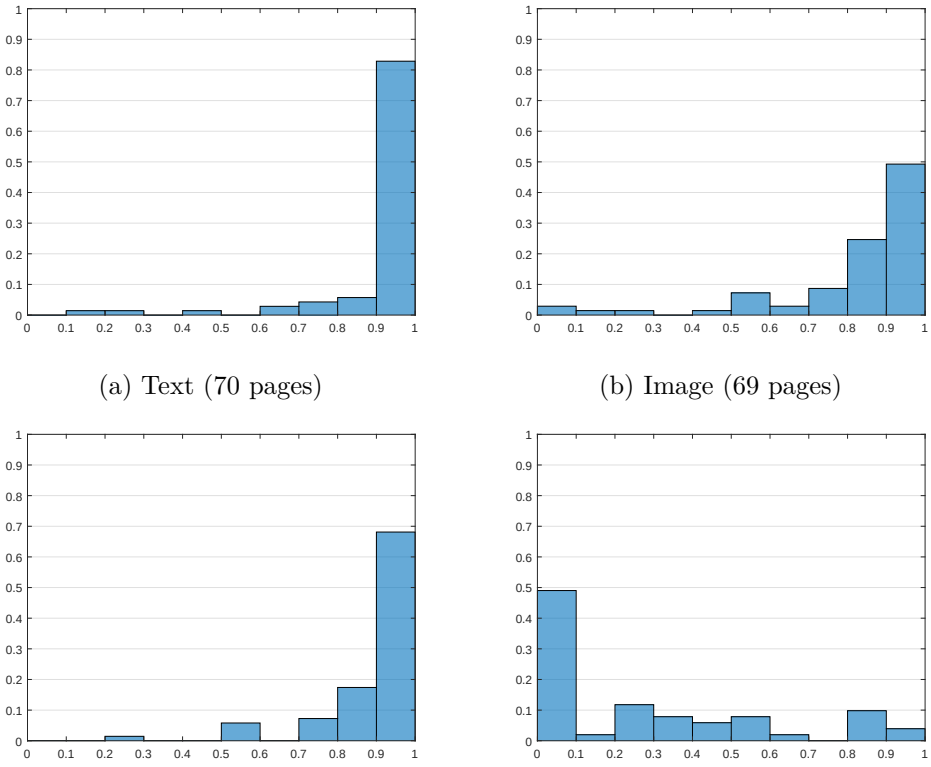
Figure 4.9: Example of a ground-truth region rectangle (a) and a non-rectangular polygon resulting from Tesseract page segmentation (b).

to store the Tesseract page segmentation and classification results in the PAGE format. We use version 3.04 of the Tesseract engine. However, even with the same data format, a direct comparison of the results is not trivial, because Tesseract produces non-rectangular (but manhattan) region polygons.

The less constrained region shape allows for more exact region descriptions than our ground-truth rectangles. Therefore, without changing their polygon shape, it only makes sense to measure the (area-based) precision values, but not the recall, since not all parts of the ground truth rectangles are necessarily also part of the actual content regions, which could lead to recall penalties in cases where the evaluated region polygons are actually correct and more accurate. An example of this problem is shown in Figure 4.9. Additionally, the Tesseract engine only produces text, image, table and separator polygons. Because charts are mostly made up of graphical elements, the bulk of the chart regions should be classified as images. The precision measurements for the image class are therefore taken twice: once using only image rectangles as ground-truth and once with all charts added to the set (Image*):

Text	Image	Image*	Table
0.94	0.85	0.91	0.4

Figure 4.10 shows the histograms of the resulting precision values of the pages in the evaluation set. Each class is only evaluated on pages containing regions of the respective type. For 83 % of the documents, the Tesseract precision for text regions is ≥ 0.9. The overall text precision is 0.94. Continuing with the image class, for one of the 70 pages, no Tesseract image polygons are produced. When only the ground-truth image regions are taken into account, 49 % of the remaining pages have precision values ≥ 0.9, and the overall image precision is 0.85. These results increase when the chart regions are added to the ground-truth sets, resulting in an overall precision of 0.91, with 68 % of the pages having a precision of ≥ 0.9. The overall table precision is the lowest with 0.4 and the histogram showing a more even distribution with the ≥ 0.9 bin only containing 4 % of the pages. The histogram only shows the values for 51 of the 70 pages, because in the other 19, no table regions are found by Tesseract. The < 0.1 bin contains 49 % of the pages, meaning that almost all of the pixel area (≥ 90 %) contained in these pages which is labeled as table is false.



(a) Text (70 pages)

(b) Image (69 pages)

(c) Image (GT with charts, 69 pages)

(d) Table (51 pages)

Figure 4.10: Histograms of per-document precision values of Tesseract polygons compared to ground-truth rectangles.

(a)

(b)

Figure 4.11: Adjacent image region polygons produced by Tesseract (a) and rectangular region resulting from them (b).

In order to take further measurements, it is necessary to make the Tesseract output comparable to the available ground-truth data. The polygons produced by Tesseract can easily be converted to rectangles using their bounding boxes, whereby the additional accuracy is lost. However, if they are correct, their bounding boxes should also match the ground-truth rectangles. Since the region polygons produced by Tesseract are often directly adjacent to each other, the bounding boxes for each region class (text, image, table) are obtained by first creating a binary pixel mask with all pixels enclosed by polygons to 1 and then finding the bounding box around all connected components. An example can be seen in Figure 4.11.

The following table shows the precision and recall values of the bounding boxes of the Tesseract regions for all classes and compares them to the results of the proposed method, which have already been shown in section 4.3.

Type	Precision		Recall		F_1	
	Tess.	Prop.	Tess.	Prop.	**Tess.**	**Prop.**
Text	0.92	**0.99**	0.86	**0.98**	0.89	**0.98**
Image	0.8	**0.92**	0.78	**0.8**	0.79	**0.86**
Image*	0.87	**0.94**	0.73	**0.77**	0.79	**0.85**
Table	0.4	**0.79**	0.81	0.81	0.54	**0.8**

The “Image*” class refers to the combination of image and chart regions to one class. For the Tesseract regions, this means that the set of computed image regions is evaluated using the set of all ground-truth image and chart regions. For the proposed method, it means that the same ground-truth set is used to evaluate the set containing all computed image and chart regions (all charts are treated as images in both sets).

In comparison to the values of the exact Tesseract output regions, the precision results have deteriorated by an average of 0.03. For all classes, the precision of the proposed method is higher. While for the table class, the recall values are equal, the Tesseract precision is only 0.4, which is about half of the result of the proposed technique. This means that a very large amount of table areas is produced by Tesseract, which cover large portions (81 %) of the ground truth, but also much more than that. Using the F_1 score, the results can be summarized in a single number for each class. The table shows that the proposed method has slightly better results for the Text, Image and Image* classes, and a significantly better overall result for the Table class. However, one has to keep in mind that the ground-truth text regions have not been manually annotated, but are identical to the XML text regions, which are also used to produce the result. Therefore, the results for the text class may not be as high for a manually annotated ground truth set, because some of the XML text region boundaries contain errors (as explained in section 3.1.2).

CHAPTER 5

Conclusion

A technique for segmentation and classification of partly correct, rectangular text and image regions of a dataset of contemporary newspaper pages has been presented. The evaluation results show that even with the simple methods proposed in this thesis, data extracted from PDF files can be used to obtain a viable page layout description.

Visual inspection of the available dataset showed the need to segment the given image regions, which are often overlapping, too large or contain only parts of images. It also showed that while tables are usually described by text regions, the graphical components of chart regions are not always described by image regions. A look at the PDF format and how it stores content has given an explanation for the properties of the included document regions and how such information can be extracted from PDF files. We have obtained segmented image regions by using the available information and adjusting the boundaries to non-text foreground objects occurring in the page. In order to distinguish between text, image, chart and table regions, we have trained a random forest classifier using HOG feature descriptors.

The classification has achieved low overall error rates of 0.05 for text/table distinction and 0.1 for image/chart distinction. In the area-based segmentation evaluation, the computed regions fit the ground truth with an overall recall of 0.77 and precision of 0.94. Without the chart regions in the ground truth set, the recall improved to 0.85 and the precision deteriorated to 0.9, confirming what the visual inspection already showed for a smaller subset of newspapers. Additionally, matches between segmented and ground-truth region boundaries have been evaluated, resulting in recall and precision values between 0.35 and 0.5, depending on the tolerance value. Again, the recall increased by 0.19 on average when the charts were left out of the ground truth set, while the precision only decreased by 0.01 on average. Without the charts in the ground-truth set, over 60 % of image bounding boxes are accurately detected with evaluation tolerance values ≥ 0.1. The high precision values of the area-based evaluation (0.94 with charts and 0.9 without charts) in

contrast to the lower values of the match-based evaluation showed that most segmented image regions do lie on the ground-truth regions, but don't fit them exactly.

The results of the complete system have confirmed that regions of all classes have large intersections with the ground-truth regions, but often do not fit them exactly. Additionally, they reveal that the proposed method is not viable for the purpose of locating charts.

The comparison to Tesseract shows that, having the advantage of preexisting knowledge about document regions extracted from PDF files, the proposed system outperforms it in the segmentation and classification of text, image and table regions.

In order to better detect chart regions, the method would have to be extended by additional steps like classifying combinations of images and text as possible regions. Also, in addition to raster images, it would be necessary to also consider vector graphics contained in the document, which could also be extracted from PDF files, but are not included in the dataset used and described in this thesis.

Since the presented results are limited to rectangular region boundaries, it would be interesting to extend the method to generate non-rectangular region polygons or pixel masks, as explained in section 3.5. This would allow more precise evaluation of the results and a direct comparison to other contemporary page segmentation techniques. However, in order to evaluate it, exact ground-truth data would be required. Instead of the newspaper page images used in this thesis, the dataset proposed for the ICDAR layout analysis competitions by Antonacopoulos *et al.* [ABPP09] [APBP09] could be used, which includes detailed ground-truth information.

Bibliography

[ABPP09] Apostolos Antonacopoulos, David Bridson, Christos Papadopoulos, and Stefan Pletschacher. A realistic dataset for performance evaluation of document layout analysis. In *Document Analysis and Recognition, 2009. ICDAR'09. 10th International Conference on*, pages 296–300. IEEE, 2009.

[ABX07] Chang An, H Baird, and Pingping Xiu. Iterated document content classification. In *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, volume 1, pages 252–256. IEEE, 2007.

[ACPP11] Apostolos Antonacopoulos, Christian Clausner, Christos Papadopoulos, and Stefan Pletschacher. Historical document layout analysis competition. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 1516–1520. IEEE, 2011.

[ACPP13] Apostolos Antonacopoulos, Christian Clausner, Christos Papadopoulos, and Stefan Pletschacher. Icdar 2013 competition on historical newspaper layout analysis (hnla 2013). In *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*, pages 1454–1458. IEEE, 2013.

[ACPP15] Apostolos Antonacopoulos, Christian Clausner, Christos Papadopoulos, and Stefan Pletschacher. Icdar2015 competition on recognition of documents with complex layouts-rdcl2015. In *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, pages 1151–1155. IEEE, 2015.

[AEM01] Oronzo Altamura, Floriana Esposito, and Donato Malerba. Transforming paper documents into xml format with wisdom++. *International Journal on Document Analysis and Recognition*, 4(1):2–17, 2001.

[APBP09] Apostolos Antonacopoulos, Stefan Pletschacher, David Bridson, and Christos Papadopoulos. ICDAR 2009 page segmentation competition. In *2009 10th International Conference on Document Analysis and Recognition*. IEEE, 2009.

[BC06] Henry S Baird and Matthew R Casey. Towards versatile document analysis systems. In *International Workshop on Document Analysis Systems*, pages 280–290. Springer, 2006.

[BMN+06] Henry S Baird, Michael A Moll, Jean Nonnemaker, Matthew R Casey, and Don L Delorenzo. Versatile document image content extraction. In *Electronic Imaging 2006*, pages 60670R–60670R. International Society for Optics and Photonics, 2006.

[CAP17] Christian Clausner, Apostolos Antonacopoulos, and Stefan Pletschacher. ICDAR2017 competition on recognition of documents with complex layouts - RDCL2017. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, nov 2017.

[CCD10] Patrick Chiu, Francine Chen, and Laurent Denoue. Picture detection in document page images. In *Proceedings of the 10th ACM symposium on document engineering*, pages 211–214. ACM, 2010.

[CF04] Hui Chao and Jian Fan. Layout and content extraction for pdf documents. In *International Workshop on Document Analysis Systems*, pages 213–224. Springer, 2004.

[CPA11a] Christian Clausner, Stefan Pletschacher, and Apostolos Antonacopoulos. Aletheia-an advanced document layout and text ground-truthing system for production environments. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 48–52. IEEE, 2011.

[CPA11b] Christian Clausner, Stefan Pletschacher, and Apostolos Antonacopoulos. Scenario driven in-depth performance evaluation of document layout analysis methods. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 1404–1408. IEEE, 2011.

[CYL13] Kai Chen, Fei Yin, and Cheng-Lin Liu. Hybrid page segmentation with efficient whitespace rectangles extraction and grouping. In *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*, pages 958–962. IEEE, 2013.

[DT05] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[FHD90] James L Fisher, Stuart C Hinds, and Donald P D'Amato. A rule-based system for document image segmentation. In *Pattern Recognition, 1990. Proceedings., 10th International Conference on*, volume 1, pages 567–572. IEEE, 1990.

[GHOO13] Max Göbel, Tamir Hassan, Ermelinda Oro, and Giorgio Orsi. Icdar 2013 table competition. In *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*, pages 1449–1453. IEEE, 2013.

[GYJ+17] Liangcai Gao, Xiaohan Yi, Zhuoren Jiang, Leipeng Hao, and Zhi Tang. Icdar2017 competition on page object detection. In *Document Analysis and Recognition (ICDAR), 2017 14th IAPR International Conference on*, volume 1, pages 1417–1422. IEEE, 2017.

[JZ96] Anil K Jain and Yu Zhong. Page segmentation using texture analysis. *Pattern recognition*, 29(5):743–770, 1996.

[Kis14] Koichi Kise. Page segmentation techniques in document analysis. In *Handbook of Document Image Processing and Recognition*, pages 135–175. Springer, 2014.

[NS84] G. Nagy and S. Seth. Hierarchical representation of optically scanned documents. *Proc. 7th Int. Conf. Pattern Recognition*, pages 347–349, 1984.

[Nur13] Anssi Nurminen. Algorithmic extraction of data in tables in pdf documents. Master's thesis, Tampere University of Technology, 2013.

[O'G93] Lawrence O'Gorman. The document spectrum for page layout analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1162–1173, 1993.

[Ots79] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, 9(1):62–66, 1979.

[PA10] Stefan Pletschacher and Apostolos Antonacopoulos. The page (page analysis and ground-truth elements) format framework. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 257–260. IEEE, 2010.

[SC96] Frank Y Shih and Shy-Shyan Chen. Adaptive document block segmentation and classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(5):797–802, 1996.

[SM00] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

[Smi07] Ray Smith. An overview of the tesseract ocr engine. In *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, volume 2, pages 629–633. IEEE, 2007.

[Smi09] Raymond W Smith. Hybrid page layout analysis via tab-stop detection. In *Document Analysis and Recognition, 2009. ICDAR'09. 10th International Conference on*, pages 241–245. IEEE, 2009.

[WCW82] Kwan Y. Wong, Richard G. Casey, and Friedrich M. Wahl. Document analysis system. *IBM journal of research and development*, 26(6):647–656, 1982.