

A DOCUMENT IMAGE ANALYSIS SYSTEM FOR EDUCATIONAL PURPOSES

Andrei Tigora¹

ABSTRACT

The Document Image Analysis System, shortly DIAS, is a modular software framework for image processing and information extraction. The system is comprised of various elementary executables, that form the processing backbone, as well as several GUI applications aimed at simplifying the configuration process and fixing errors that may appear during the actual image processing. The system is modular, and can easily be extended with more executables that fit into one of the predefined categories.

Keywords: Document Image Analysis, Retroconversion, Binarization, Skew Detection, Layout Analyze, Document Hierarchy, Document Export

1. INTRODUCTION

Storing information digitally has become a viable alternative to printing on paper; documents that would otherwise require a special room for safekeeping can be stored digitally and carried around in handbags. However, the simple fact that there is a better alternative for storage does not mean that the “old ways” are gone. There still exist many rooms whose only purpose is to act as storage space for these documents created before the dawn of the digital age. Still, storage itself is not the greatest of problems – after all, they have been there for quite some time and there was space for them. The greatest problem is in fact access to the information contained within them.

Some of these collections of documents are written or printed archives, for whom indexing is of utmost importance. Others are stored as microfilms, a first attempt at solving the problem of storage space and facilitating access. Just as numerous, but perhaps even more valuable, are manuscripts and old prints. No matter what kind of collection one is dealing with, they all exhibit the same restricted access to information, whether it is due to a prohibitively large volume of data or access restrictions imposed by the value of the documents themselves.

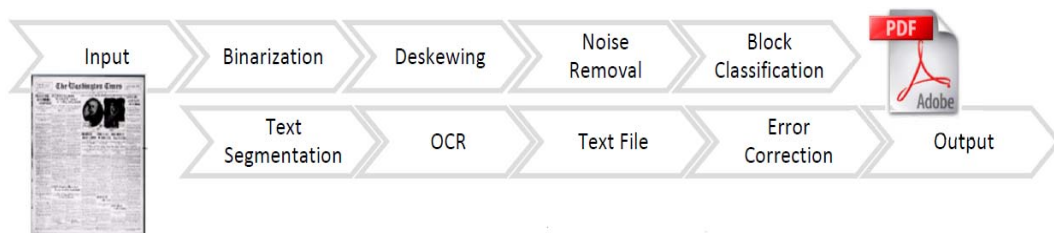


Figure 1. The typical processing flow in document image analysis software

¹ **Engineer**, Department of Computer Science and Engineering, Faculty of Automatic Control and Computers Science, University “Politehnica” of Bucharest, Splaiul Independenței 313, Bucharest, 060042, Romania, andrei.tigora@cti.pub.ro

The task of solving these problems falls to document image analysis software. These are complex applications that offer support for:

- digitally enhancement of scanned images of the documents
- identifying entities within the documents and indexing them
- grouping various entities and organizing them in a hierarchy
- automatic text recognition
- many others

2. HANDLERS

Processing is performed by small independent applications called handlers.

Each handler may accept either one or two parameters: the first one represents the processing file that describes the task that should be performed by the handler, whereas the second one represents the name of the report file that indicates whether the processing was successful or not. If the second parameter does not exist, no report file is created. Both the processing and report file are XML documents.

The format of the processing file is specified through an xml schema file that accompanies each handle. Apart from the elements that describe the structure of the processing file, it also contains auxiliary information that describes the handler's utility. These identification elements can be found within three data types:

- **execName**: the name of the handler corresponding to this schema
- **execType**: the nature of the processing performed by the associated handler
- **execDescription**: a short description of the handler's capabilities

All processing files contain the following types of elements:

- **inputFile**: usually a single one, but some handlers may use several, that corresponds to an actual file to be processed by the handler
- **outputFile**: a single one, corresponding to the output file after the handler's processing is done

Ideally, an image handler should not require more than one (or possibly more) input and one output file entries in its processing request. However, *not all handlers are written equal*, and depending on the quality and nature of the input material, some handlers might require some fine tuning for their parameters. Moreover, GUI specific handlers, since they are not usually part of an automated processing workflow and are run one at a time, have several extra parameters to answer a user's every whim.

The report file succinctly describes the results of the performed processing. A report file looks as follows:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<report>
<result>failure</result>
```

<details>Unexpected file format</details>
</report>

The value of the *result* element may be either *success* or *failure* corresponding to the two possible outcomes of running the handler. The *details* section presents in case the processing failed a reason for the failure, and a simple success message otherwise.

3. PROCESSING

Before talking about the actual processing, it must be clear what type of the images can be processed by DIAS. Based on how the information is stored, an image can either be a raster image or a vector image. The latter are made of geometric objects, precisely described through mathematical formulas, that allow them a high display accuracy regardless of the requested level of detail. Unfortunately, these images are not the standard output for devices that capture the physical world; most of the times they are the product of direct human intervention, through dedicated software.

Therefore, it comes as no surprise that the images that are of interest for image processing are raster (also known as bitmap) images. These store the information as a matrix of pixels, which is in most cases more than enough for an accurate of the world for regular human use. Yet, it does pose some problems for automated processing and those problems are exactly what DIAS attempts to solve.

The handlers perform various processing tasks that are necessary for extracting information from a regular image and storing it in a more appropriate format. The type of the processing performed by a certain handler is indicated by the *execType* data type in the xsd file associated to the handler. The major processing components can be grouped under seven categories:

- preprocessing
- binarization
- layout analysis
- paging
- OCR
- hierarchy analysis
- PDF exporter

Apart from these seven processing indicators, there are also some auxiliary indicators; one of them indicates whether the handler is GUI friendly, whilst the other represent minor processing categories.

The following sections describe the reason for using each of these types of handlers.

4. PREPROCESSING

Image enhancement occurs before actual processing is performed, so it corresponds to the *preprocessing* phase [1]. While regular images of scanned documents have a good enough quality to be subjected to content extraction, the images of documents of high interest, such as archives and valuable old writings, do not meet the standards. There are a couple of reasons why that might happen:

- the available scanned image (or microfilm) is lacking in quality, and accessing the original document is not possible
- the document itself is in poor shape, having been affected by environmental conditions (extremes in air humidity, light exposure) or improper handling

A substandard document image exhibits one or more of the following characteristics, either globally or locally:

- poor contrast between foreground and background [1]
- random noise [11]
- skewed images [10]

Fixing, or at least reducing these aberrations is important before carrying on with higher level processing. Even here, the saying *garbage in, garbage out* still holds true.

Input: 8BPP grayscale image

Output: 8BPP grayscale image

5. BINARIZATION

The aim of the whole system is to extract information from the document image and obtain a file enhanced with structure and contents information. [2][4] Binarization, on the other hand, does not aim to add to the information, but to reduce the image to the bare essentials. Binarization is the process of separating between background and foreground [3]; during binarization, the pixels of the raster images are assigned to one of the two groups and from thereon are represented as either white or black, depending on whether they are foreground or background.



Figure 2. Different binarization results corresponding to different algorithms
The first line contains 3 different grayscale input images
Every column contains 6 distinct binarization results

The information loss might seem rather radical: starting from an 8BPP (or even 24/32BPP image) and ending up with a 1BPP image is radical. Nevertheless this is necessary, since the next processing steps are rather complex themselves, and handling complicated images would only complicate the algorithms further.

Input: 8BPP grayscale image

Output: 1BPP grayscale image

6. LAYOUT ANALYSIS

Segmentation is perhaps most well known through binarization, but it covers a wide array of algorithms aimed at cataloguing pixels for the aim of *segmenting* the image into smaller entities. The *layout analysis* performed by DIAS's handlers relies on segmentation to identify text lines and group them into paragraphs, pictures within the image as well as graphic elements that do not fit within either of the previous categories [5][6][7].

Input: 1BPP grayscale image

Output: layout specific XML file

7. PAGING

While the page number in itself is not that relevant for a single document image, it is important when handling multiple images. The paging information may be used by the hierarchy analysis handler to determine the correct order to position the pages.

Input: layout specific XML file

Output: layout specific XML file

8. OCR

OCR is the acronym for Optical Character Recognition and its aim is, as the name suggests, to identify the corresponding characters from images of texts. The character identification is a rather complex process that involves comparing the sample image of the character with predefined bitmaps or feature lists image corresponding to the candidate characters. Most OCR engines do not rely on character level matching, instead they attempt to find matches at word level, with the help of dictionaries.

OCR's are anything but perfect; in fact they even struggle with non English Latin script texts which tend to make more heavy use of diacritical marks than English does. Support for less common scripts is often non-existent, and even when it is available, it might not have an appropriate dictionary for improved character extraction.

Since OCRs are very complex applications, developing an independent engine from scratch is very difficult [8]. As a result, for now, the available handlers act as wrappers that make use of preexisting OCR software [12]. However, it should be noted that the only information

used from the OCR engines' output is the text itself, not the structure details. The document image structure is obtained through the various available handlers.

Input: layout specific XML file

Output: layout specific XML file

9. HIERARCHY ANALYSIS

While identifying text lines and paragraphs is no easy feat, a greater degree of organization is usually expected. A paragraph does not mean much by itself, but if it can be added in a logical succession with other paragraphs and grouped together with pictures, then they become an article, which, as an entity, can stand on its own. The article is relevant enough to be referenced in a catalogue and therefore ease future search operations.

In constructing the hierarchy of elements, the only aspects taken into consideration should be the geometric characteristics of the entities [7]. The handlers should not be required to "understand" the contents of the document. This is not as far-fetched as it might initially sound; after all, human beings are capable of identifying a title regardless of the language, because the organization criteria are universal.

Input: one or more layout specific XML file

Output: hierarchy specific XML file

10. PDF EXPORTER

The ultimate goal of the Document Image Analysis System is to obtain a content enhanced image. Since images format were not conceived with the purpose of attaching non image data to them, a different format was chosen to store the result: PDF. The Portable Document Format is an open source standard developed by Adobe. PDF began as an internal document exchange format for Adobe, gradually became the defacto standard for image exchange, and it is nowadays a de jure standard having been transferred under the authority of the International Organization for Standardization.

The PDF exporter handlers convert the XML hierarchy format into a format that is in accordance with the PDF standard. The end result is a PDF document that has as pages the images used for generating the hierarchy XML and allows content based navigation and selection of the text.

11. LAYOUT AND HIERARCHY XML FILES

Although the handlers are independent and new ones can be added without affecting the overall functionality of the DIAS, they must be able to communicate with the other handlers, that is, they must be able to parse the XML files generated by other handlers and in turn generate valid XML files that may be used by the other handlers. For that purpose, two different xsd files were created, to validate the layout specific XML files and hierarchy specific layout files.

12. GUI APPLICATIONS

While the actual image and xml processing are the main focus of the system, from the human direct human interaction with the handlers does have its drawbacks. Firstly, creating the task input files for the handlers requires familiarity with the XML standard and the Schema specification. If this entry level barrier is passed, there is the problem of productivity, as each new processing task requires a new input file. Just as important, is the correctness aspect of the performed processing. Most of the analyzing algorithms are based on heuristics and their performances are greatly influenced by the quality of the input material [9]. Therefore, on some occasions, various errors might occur that require user intervention to have them corrected.

All GUI applications rely on some extent on handlers, and selecting the appropriate handler to be used is crucial. For ease of use, a special folder is designated for storing the handlers, and another one for storing the schemas. To identify the handlers relevant for a particular GUI application, the application has to parse all the schema files and identify the ones which have the appropriate `execType` specified. If the `execType` does not match the expected one, then the schema is ignored

To solve these problems, a series of GUI applications were developed, and they are presented in the following sections.

13. DOCUMENT PROCESSING MANAGER

The Document Processing Manager was created with two purposes in mind:

- automating the generation of tasks files
- executing complex workflows of multiple processing tasks

Generating the task files used by the handlers can be tedious, especially when a large number of tasks must be run. Therefore, this mechanism is automated by the Document Processing Manager. The user first selects one or more files, which depending of the processing to be performed may be image files in various formats and BPPs or XML files. Then, appropriate handlers are chosen from each of the processing categories. The processing takes place in a pipeline fashion; it may begin with any of the existing processing steps, but it may not skip any intermediate step until the final one. To be more precise, a workflow can begin with binarization and end with hierarchy analysis, but it cannot skip layout analysis along the way.

Once the task input files are created, the application also manages executing the handlers. For each of the tasks it must make sure that the processing did not encounter any problems, by analyzing the report XML. If any errors occurred, the processing ends.

Another feature of the application is to run previously created workflows, or template workflows. The first option allows experienced users to edit their own workflows outside the application environment and then have the manager control the actual execution. The second one on the other hand permits the creation of a template workflow, with specific processing components, so that the same type of processing can be performed over and over again without having to select the components every time the application is launched.

14. WEB BASED DOCUMENT PROCESSING MANAGER

This application is a simplified version of the Document Processing Manager that is accessed through a web interface. The user loads on or more images and then chooses the appropriate handlers. The processing starts with binarization (preprocessing is optional) and ends with a PDF-exporter handler. Once the processing ends, a link is provided to the user, allowing the retrieval of the resulting file.

This application is more of a proof of concept, that is most likely to be used with high quality input files, where little or no human intervention is necessary. For development and testing purposes, the desktop version of the application is used instead.

15. PREPROCESSING GUI

Having a high quality input image is ideal, but most images that are of interest are far from being that. Processing an image in its raw form gives poor results, so enhancing certain features could greatly improve the results of the high level analyzers.

Unfortunately, deciding what kind of enhancement yields better results is sometimes hard to estimate based on algorithm alone. In some cases, it may even be necessary to run the entire workflow to have a clear picture of what works best and what does not. However, most of the times, it can be guessed from the binarized image.

The current GUI application loads a grayscale image, or a color image that it converts to grayscale and displays it. Then, it can choose preprocessing, support and binarization handlers to determine how the image would look after various processing steps with different parameters. All the resulting images are displayed in a side panel for visual comparison; furthermore, the user may choose a pair of the binarized images displayed in the side panel and have them investigated more closely. The user may switch between the two images for comparison purposes and may also perform bitwise operations between the two images to determine the differences and similarities between the two of them. The user may choose to save one, more or none of the images present in the side pane.

As mentioned above, the user may use preprocessing, support and finally binarization handlers. The handlers marked as *support* offer functionalities similar to those of the preprocessing but cannot be easily automated. One such example is cropping an image; while it is fairly simple to crop an image, this means that the xml file describing this tasks requires more than just *inputFile* and *outputFile* elements. Since the aim of the Document Processing Manager was to automate input XML creation, managing tags other than the previously mentioned ones would have added unnecessary complexity. Moreover, cropping tends to be a lot more input dependent than any of the other processing, so it needs configuring on a file by file basis. Setting the extra parameters for the support type handlers is also automated, and relies solely on the contents of the schema file.

16. LAYOUT ANALYSIS CORRECTION GUI

As mentioned throughout this document, the results of the processing are sensible to the input, meaning that errors may occur, especially in “ambiguous” regions of the image. As

a result, such regions may end up being catalogued incorrectly – such as being tagged as image instead of text. While directly editing the resulting XML file is not difficult, the lack of visual clues to the actual image itself makes it impossible.

This is exactly what the current application aims to solve, by visually representing the image and the various blocks on top of it. The user may delete, add, change or retag the existing blocks to better reflect the structure of the image. The user may also rerun character extraction on select blocks to replace the text automatically, or edit the text by hand if the identified text is not correct. Also, a paging handler may be run to identify the number of the page, if one exists. If the user is unsatisfied with the identified block, another one may be designated in its place.

The application may either load an image and then choose a layout analysis handler to generate the layout XML, or generate blocks by hand through the GUI directly over the image, or load an existing layout XML and modify it accordingly. Eventually, the user saves a layout specific XML file.

17. HIERARCHY ANALYSIS CORRECTION GUI

The aim of this application, just like the previous one, is to correct errors following hierarchy analysis. This time however, the user has to deal with multiple images, and the main action will be regrouping and reassigning blocks to higher level blocks.

The user either loads a hierarchy XML files, or generates the file right away through a hierarchy analysis handler from several layout XML files. The user is presented with multiple images, one after the other, with rectangles, and perhaps more complex polygons if necessary [6], displayed above these images. Similar block manipulation actions are possible as the ones described for the previous correction application. Eventually, the user saves a hierarchy specific XML file.

18. FUTURE WORK

Although the system is functional, it is far from being complete and requires a lot more work to be done.

In the next period, the main focus will be extending the number of handlers available. Right now, for some of the processing categories there is only a single handler available, so there can be no comparative analysis of performance and trade-offs. As a side objective, it would be worth analyzing if adding more processing categories is necessary; it is currently being evaluated if a grayscale conversion [13] category could be useful to supplement the default luminance conversion mechanism.

A less obvious intervention would be to refine the intermediate XML schemas generated by the layout and hierarchical analysis. The current specifications were developed in parallel with the analyzers themselves, so if more information is required by future more complex analyzers, the schemas should be changed to allow the inclusion of the extra information.

Another possible expansion of the system would be to integrate all, or at least some, of the GUI Applications into a single one. The most obvious one would be to merge the layout correction GUI with the hierarchy correction GUI. Though there are some differences between the formats of the output XML files following the two processing phases, the nature of the corrections is similar enough to permit a merging of the two. A different change to the GUI applications would be to create a central GUI that would then launch all the other applications. Apart from the benefit of keeping all the applications together, this could also allow more complex processing workflows, interleaving the correction applications with the handlers.

19. BIBLIOGRAPHY

- [1] C. A. Boianiu, A. I. Dvornic. *Modern Preprocessing Techniques for Automatic Content Conversion Systems*, Annals of DAAAM for 2008, Proceedings of the 19th International DAAAM Symposium, pp. 0123–0124
- [2] C. A. Boianiu, I. Bucur, A. Tigora, *The Image Binarization Problem Revisited: Perspectives and Approaches*, The Proceedings of Journal ISOM Vol. 6 No. 2 / December 2012 pp. 419-427
- [3] C. A. Boianiu, A. Olteanu, A. V. Stefanescu, D. Rosner, A. I. Egner, *Local Thresholding Image Binarization using Variable-Window Standard Deviation Response*, Annals of DAAAM for 2010, Proceedings of the 21st International DAAAM Symposium, 20-23 October 2010, pp. 133-134
- [4] C. A. Boianiu, A. I. Dvornic, D. C. Cananau, *Binarization for Digitization Projects Using Hybrid Foreground-Reconstruction*, Proceedings of the 2009 IEEE 5th International Conference on Intelligent Computer Communication and Processing", Cluj-Napoca, August 27-29, pp.141-144
- [5] C. A. Boianiu, D.C. Cananau, B. Raducanu, I. Bucur, *A Hierarchical Clustering Method Aimed at Document Layout Understanding and Analysis*, International Journal of Mathematical Models and Methods in Applied Sciences, Issue 1, Volume 2, 2008, pp. 413-422
- [6] C. A. Boianiu, B. Raducanu, S. Petrescu, I. Bucur, *Ensure Non-Overlapping in Document Layout Analysis*, 20th DAAAM World Symposium, Austria Center Vienna (ACV), 25-28th of November 2009
- [7] C. A. Boianiu, D. C. Cananau. *Combined Approaches in Automatic Page Clustering for Content Conversion*, Annals of DAAAM for 2008, Proceedings of the 19th International DAAAM Symposium, pp. 0121-0122
- [8] Geometric Methods for Optical Character Recognition, G. N. Sazaklis, dissertation, 1997
- [9] C. A. Boianiu, A. V.Stefanescu, D. Rosner, A. Olteanu, A. Morar, *Automatic Slanted Edge Target Validation in Large Scale Digitization Projects*, Annals of DAAAM for 2010, Proceedings of the 21st International DAAAM Symposium, 20-23 October 2010, pp. 131-132
- [10] C. A. Boianiu, A. V.Stefanescu, D. Rosner, A. Olteanu, A. V. Stefanescu, A. D. B. Moldoveanu, *Confidence Measure for Skew Detection in Photographed Documents* Annals of DAAAM for 2010, Proceedings of the 21st International DAAAM Symposium, 20-23 October 2010, Zadar, Croatia, pp. 129-130
- [11] T. Saba, G. Sulong, A. Rehman, Document image analysis: issues, comparison of methods and remaining problems, Artificial Intelligent Review, Vol. 35, No. 2, February 2011
- [12] R. Smith, An Overview of the Tesseract OCR Engine, Ninth International Conference on Document Analysis and Recognition, Vol 2, pp. 629–633, 2007
- [13] Yongjin Kim, Cheolhun Jang, Julien Demouth, Seungyong Lee, *Robust Color-to-gray via Nonlinear Global Mapping*, ACM Transactions on Graphics (SIGGRAPH ASIA 2009), Vol. 28, No. 5, December 2009