Automatic identification and cropping of rectangular objects in digital images

Tomas Toss
Abstract

Automatic identification and cropping of rectangular objects in digital images

Tomas Toss

Today, digital images are commonly used to preserve and present analogue media. To minimize the need for digital storage space, it is important that the object covers as large part of the image as possible. This paper presents a robust methodology, based on common edge and line detection techniques, to automatically identify rectangular objects in digital images. The methodology is tailored to identify posters, photographs and books digitized at the National Library of Sweden (the KB). The methodology has been implemented as a part of DocCrop, a computer program written in Java to automatically identify and crop documents in digital images. With the aid of the developed tool, the KB hopes to decrease the time and manual labour required to crop their digital images.

Three multi-paged documents digitized at the KB have been used to evaluate the tool's performance. Each document features different characteristics. The overall identification results, as well as an in-depth analysis of the different methodology stages, are presented in this paper. In average, the developed software identified 98% of the digitized document pages successfully. The software's identification success rate never went below 95% for any of the three documents. The robustness and execution speed of the methodology suggests that the methodology can be a compelling alternative to the manual identification used at the KB today.
Contents

1 Introduction .......................................................... 1
   1.1 Motivation and purpose ........................................ 1
   1.2 Related work .................................................... 2
   1.3 Digitization at the KB .......................................... 3

2 Theoretical framework .............................................. 5
   2.1 Greyscale morphology .......................................... 5
   2.2 Feature extraction ............................................. 7
      2.2.1 Edge detection ............................................ 7
      2.2.2 Line detection ........................................... 11

3 Methodology .......................................................... 14
   3.1 Methodology description ........................................ 14
      3.1.1 Morphology ................................................. 16
      3.1.2 Edge detection ............................................. 17
      3.1.3 Curve detection ........................................... 18
      3.1.4 Interest point extraction and minimal rectangle genera-
      tion .............................................................. 19
      3.1.5 Quantitative analysis ..................................... 21

4 Implementation ....................................................... 22
   4.1 Implementation overview ....................................... 22
   4.2 Implementation details .......................................... 23
      4.2.1 Morphological operation ................................ 23
      4.2.2 Canny edge detection .................................... 24
      4.2.3 Line detection ............................................. 26
      4.2.4 Interest point extraction ................................ 27
      4.2.5 Minimal enclosing rectangle ......................... 28
      4.2.6 Complete document identification ..................... 29

5 Experiments and results ............................................ 31
   5.1 Experiments ..................................................... 31
   5.2 Results ......................................................... 33

6 Discussion .......................................................... 35
   6.1 Solution methodology and software ......................... 35
   6.2 Morphology ..................................................... 35
   6.3 Edge detection .................................................. 38
   6.4 Line detection .................................................. 40
   6.5 Interest point extraction ..................................... 40
   6.6 Minimal rectangle generation ................................ 42
   6.7 Quantitative analysis ......................................... 43
# Conclusions and Future Work

7.1 Conclusions

7.2 Future Work

7.3 Acknowledgements

References

A Visual tour of DocCrop

B Software execution time

C Quantitative analysis - page movements
Thesis structure

Introduction The motivation and purpose of the thesis are outlined, followed by a description of related studies and the state of current work. Finally, a description of the digitization process at the KB is presented.

Theoretical framework The theory behind the different parts of the proposed solution is explained, and the investigated methods are addressed.

Methodology The suggested solution is presented as a methodology, and the individual parts are described in detail.

Implementation The overall structure of the implemented solution is explained and the core components of the implementation are described as pseudo code.

Experiments and results The setup of the experiments is presented together with a description for how the identification quality is determined. Finally, the results of the conducted experiments are presented.

Discussion The methodology and the results of the conducted experiments are analysed and discussed. Each stage of the methodology is analysed separately.

Conclusion and future work A presentation of the conclusions drawn from the experiments is given, and topics for future work are presented.

Appendices

A Visual tour of DocCrop The developed software and its functions are displayed by images and short explanatory texts.

B Software execution time A complete presentation of the execution times for DocCrop to identify the digitized documents is given.

C Quantitative analysis - page movements The page movement of the examined multi-paged documents are visualized by a collection of graphs.
1 Introduction

1.1 Motivation and purpose

Digitizing analogue media, e.g. books, photographs, sound and video is an important process in the field of media preservation. Digital media offers several benefits compared to its physical counterpart, such as less physical storage space, and increased accessibility and functionality [17]. At the National Library of Sweden (the KB), the digitization of media is a continuous effort. The library’s experiences over the years have led the KB to design and fine tune their digitization process, striving for high digitization throughput while using the least amount of resources. By using inexpensive commodity equipment and software, the KB has reduced the investments necessary for digitization. To further decrease the cost of digitization, the KB aims to lessen the manual labour needed during the process [11].

At the time of writing, the printed media at the KB is digitized such that there is a distinct margin between the object and the borders of the digital image. The redundant area captured leads to higher digital storage needs, and may also result in difficulties in further processing steps, such as optical character recognition (OCR). To avoid this, a post processing stage which identifies the object of interest in the digital image can be applied. By identifying the object, the image can be cropped accordingly. At the KB, this post processing is to a large extent performed manually [11].

The purpose of this thesis is to investigate the possibility to automate the identification and cropping of documents digitized at the KB, and to implement the ideas in software. Focus is set to develop a robust identification algorithm. Execution speed and memory efficiency of the identification algorithm is considered where possible. The software developed in this thesis is meant to be used as a part of the digitization workflow at the KB. Thus, creating a user friendly application interface is an essential part of the development effort.

The printed media investigated in this paper are restricted to objects that have a rectangular shape, such as books, posters, and images. The media are generally digitized on top of a monochrome homogeneous surface, simplifying the identification process.
1.2 Related work

The core of the thesis is rectangle detection, a task that arises in many other practical applications. Examples of applications where rectangle detection is used include vehicle [20, 24] and building [10, 9, 22] detection in aerial images, and license plate recognition [2].

Several rectangle detection techniques have been presented over the years. However, it is still a challenge to identify such objects with high reliability and speed. The most common approach to detect rectangles in image data is to extract edge and line primitives. The properties of the primitives, such as position, length and orientation, are examined. A rectangle is detected if a collection of primitives fulfil the conditions set to constitute a rectangle.

Many of the used detection methods rely on some variation of the Hough transform (HT) [7]. The method suggested by Zhu et al. [25] to detect rectangular particles in cryo-electron microscopy images relies on a rectangle HT. However, only rectangles of one and the same size are identified. Furthermore, the size must be known before extraction, which restricts the methods usefulness in this context.

Jung and Schramm [12] proposed a rectangle detection algorithm for grey-level images that utilizes a windowed HT. The algorithm can identify objects of different sizes, but has the disadvantage that it is computational expensive. In addition, the algorithm may classify non-rectangle shapes as rectangles when separated aligned rectangles are close to each other.

Liu et al. [16] presented a Markov random field (MRF) rectangle detection method for colour images. The algorithm uses gradient direction and magnitude of edge pixels to create line segments and from these regard the rectangle detection task as an optimization problem. This implementation have considerable better detection rate, lower false alarm rate and better overall execution speed compared to the randomized Hough transform (RHT) implementation by Kälviäinen and Hirvonen [13]. Furthermore, it is possible to adjust the algorithm to detect rectangular shaped objects with certain characteristics by altering the energy function used for optimization. The energy function consists of four terms, and their individual contribution control the importance of line segment closeness, length and orientation.

In addition to identify rectangles, it is necessary to determine which of the rectangles that correspond to objects of interest, e.g. a document page or a poster, and which rectangles that do not. This task is highly application dependent, and is an important part of the document detection procedure. Even though the mentioned techniques offer some possibilities to control the properties of the rectangles to be extracted, they alone can not ensure that the identified rectangle is a document.
There are numerous algorithms that analyse the layout of documents in digital images [8, 4]. Their purpose is typically to identify and extract the images, tables and text areas of the document. In order to facilitate OCR, some of the document layout algorithms estimate and correct the skewness of the documents [18]. These algorithms are however not designed to extract a complete document page from a complex scene.

There are few reported studies of document identification in academia. The core of the related studies does however provide essential concepts for the thesis, such as edge extraction, line primitive construction and rectangle shape conditions.

1.3 Digitization at the KB

This section is based on the digitization workflow description and analysis done by Johansson et al. [11].

Since 1661, the KB has collected nearly everything that has been printed in Sweden or in Swedish. The collection consists of books, newspapers, ephemera, periodicals, posters and much more. In 2010, the library’s collection included about 4 million books, 110 million newspaper pages and 10.5 million ephemera publications.

The digitization at the KB started in 1995. At first, the digitization was focused on posters and the goal was generally preservation rather than presentation. However other types of material were soon to be subject for digitization, and over time, effort to improve the presentation was made.

As a National Library, preservation is of great importance for the KB. Many of the library’s objects are unique and at the same time both fragile and delicate. The objects do therefore not generally support a flat opening angle of 180°. Furthermore, the commercial digitization solutions often have hoods and glass plates that can put extensive pressure on the spine of the book. The KB has therefore constructed their own digitization stations. Each station consists of a table, a studio grade camera stand, a commodity digital single-lens reflex (DSLR) camera and a computer workstation (Figure 1). The height of the table can be electrically adjusted and the table is covered by a sheet of paper with a neutral grey colour. A sheet of metal is placed under the paper to allow for the use of magnetic equipment on the table.
Because the digitization stations are featured with a single camera each, and many of the objects are fragile, single pages are generally captured instead of full spreads. The object that is about to be digitized is placed with one of its covers flat on the table. The other cover, together with all of the pages, is placed upon a book stand that allows opening angles of $100^\circ$ to $180^\circ$. Colourless placeholders, secured by magnetism, are used to fixate the object, and pages that tend to curl or rise are carefully fixated by small sticks. To prevent pages from turning, a small collar is draped over both the book and the book stand.

A laser is mounted on the digitization station so that its beam is projected horizontally. The height of the table is adjusted so that the beam hits the absolute top page of the object. As the pages of the object are captured and turned, the user regularly checks that the beam still hits the absolute top page. If this is not the case, the height of the table is adjusted until the beam once again hits the top page. This guarantees that the distance between the camera and the top page is constant through the digitization procedure. In addition, the focus is also controlled at each distance check.

All the objects are captured with a resolution of at least 300 ppi. The master files are not subject for any post-processing other than a file format conversion (from DNG to uncompressed TIFF). Files suitable for presentation are derived from the master files and are downscaled to 1:1, colour corrected, sharpened, and the excess area around the object is removed.
2 Theoretical framework

2.1 Greyscale morphology

Mathematical morphology is a technique for extracting and enhancing geometrical structures, often in digital images, but it can also be applied to other spatial structures. Image morphology was developed in the mid-sixties by Matheron and Serra at the Ecole des Mines in Paris [19]. The technique is based on set theory, lattice theory, topology and random functions. Dilation and erosion are two of the basic morphological operations, and these are the basis for other operations, such as closing, erosion and top-hat transformation [19].

Greyscale morphology is a subclass of image morphology, where the working set is restricted to greyscale images. In greyscale morphology, images are functions, mapping the Euclidean space $E$ into $\mathbb{R} \cup \{-\infty, \infty\}$, where $\infty$ denotes an element larger than any reals, and $-\infty$ conversely denotes an element smaller than any reals.

The basic concept of greyscale morphology is to construct a simple function, referred to as a structuring function, and apply the function step-wise to the whole grid/domain of the image function. The structuring function is constructed in a way such that the features of interest are enhanced, and is itself a function from $E$ into $\mathbb{R} \cup \{-\infty, \infty\}$.

Denoting the image function by $f(x)$ and the structuring function by $b(x)$, the basic morphological operations dilation and erosion are given by

$$(f \oplus b)(x) = \sup_{y \in E} [f(y) + b(x - y)]$$

and

$$(f \ominus b)(x) = \inf_{y \in E} [f(y) - b(x - y)]$$

respectively.

A common variant of structuring function is the flat structuring function defined as

$$b(x) = \begin{cases} 0, & x \in B \\ -\infty, & \text{otherwise} \end{cases}$$

where $B \subseteq E$ is a sliding window in $E$.

In the case where $b$ is a flat structuring function, $E$ is a grid, and $B$ is bounded, dilation and erosion will return the maximum respectively the minimum value within the sliding window $B$.

Dilatation of a greyscale image, using the structuring function defined above, shrink regions with low intensity values. Erosion of an image has the opposite effect, narrowing the high intensity regions (Figure 2).
The middle image illustrate the effect of a dilating the leftmost image by a disk (green). The two shapes (marked by dotted lines) are merged into a single large shape. The rightmost image is the result of an erosion of the leftmost image by the same structuring element.

Figure 2: The middle image illustrate the effect of a dilating the leftmost image by a disk (green). The two shapes (marked by dotted lines) are merged into a single large shape. The rightmost image is the result of an erosion of the leftmost image by the same structuring element.

The closing of an image \( I \) with the structuring element function \( b \) is defined as the dilation by \( b \) followed by the erosion of \( I \) by \( b \), or \( f \bullet b = (f \oplus b) \ominus b \). A closing of a greyscale image has the effect of enhancing the bright structures that have similar form as the structuring element. An alternative viewpoint of the closing operation is that it preserves darker regions. Dark regions that have similar shape as the structuring element, or can be completely contained by the structuring element are preserved, while all other dark regions are removed. One of the applications of the closing operand is therefore removal of “pepper noise”.

The closing operator has a dual, namely the opening operator. The opening of an image \( I \) with the structuring function \( b \) is defined as the dilation by \( b \) of the eroded image \( I \) by \( b \), or \( f \circ b = (f \ominus b) \oplus b \).

An opening of a greyscale image is typically used to remove smaller unwanted bright regions, such as “white noise”. The operator can also be used to separate specific structures from each other in images. In contrast to the closing operation, the opening operation will preserve bright areas of similar shape to the structuring element, or areas containing the structuring element, while removing all other bright areas (Figure 3).
2.2 Feature extraction

In the field of image analysis, there are often cases where the amount of data to be examined is too large for analysis, and only a small part of the data is containing information of interest. Feature extraction is a technique to solve this problem by transforming the data into another, reduced, representation set of features. The features to be extracted are chosen in such a way such that the set of features still contain information necessary to solve the task at hand.

There are several categories of feature extraction in the field of image processing. Edge, corner and blob detection are examples of low-level feature extraction techniques, while thresholding, template matching and the Hough transform are examples of shape based extraction techniques. Low-level features are typically classified as features that can be extracted from the original image, whereas high-level feature extraction is based on low-level features [14].

2.2.1 Edge detection

Edge detection is a commonly used technique in the field of feature detection and extraction. It is used to identify points of the image where the differences in brightness are pronounced. The primary objective of edge detection is to reduce the complexity of the image, while maintaining the underlying structure of the former. This can be achieved since discontinuities in brightness likely are due to one of the following [1, 15]:

- Depth discontinuities
- Changes in material properties, e.g. change of surface colour
- Variation of illumination
- Surface material discontinuities
The basis of most edge detection techniques is convolution. By convolving the original image with matrix kernels, brightness discontinuities in the original image can be extracted (Figure 4). The kernels are generally designed to either approximate the gradient or the Laplacian of the image function. In the case where a gradient approximation kernel is used, the task to identify brightness discontinuities is reduced to finding extreme values. On the other hand if a Laplacian approximation kernel is used, the task is transformed into finding zero values (Figure 5).

\[
\begin{array}{ccc}
-1 & 0 & +1 \\
-2 & 0 & +2 \\
-1 & 0 & +1 \\
\end{array} \quad \begin{array}{ccc}
+1 & +2 & +1 \\
0 & 0 & 0 \\
-1 & -2 & -1 \\
\end{array} \quad \begin{array}{ccc}
0 & 1 & 0 \\
1 & -4 & 1 \\
0 & 1 & 0 \\
\end{array}
\]

Figure 4: The two kernels to the left are examples kernels used to approximate the gradient of images in the horizontal and vertical direction respectively. The rightmost kernel is a common kernel used for Laplacian approximation.

Figure 5: The leftmost graph illustrates a typical profile of an edge in a 1D image. A rapid growth of the image function \( f(x) \) corresponds to a distinct edge in the image. A pixel is typically considered to be an edge pixel if the gradient value of the pixel exceeds some threshold (middle), or if the value of the Laplacian is close to zero (right).
A successful edge detection may reduce the image complexity significantly, and hence simplifying the feature identification/extraction process. However, the result of edge detection may not capture the sought structure in the original image. This can be due to fragmentation, that is, the edge curves are not connected. Another reason can be that false edges, i.e. edges that do not correspond to an interesting element in the image, have been extracted.

Lightning conditions, noise, and brightness difference of objects and non-objects, are factors that affect the quality of edge detection. These factors can be adjusted for by changing the threshold values of the detector. There is however no general method to automatically adjust these parameters [21].

Canny edge detection

The Canny edge detector is one of the most commonly used edge detectors, and it is by many considered the standard edge detector algorithm [21]. The algorithm was developed by John Canny at MIT in 1983 as a part of his master thesis. It was designed to be an optimal detector in the sense that

- The algorithm should identify as many real edges in the image as possible
- The identified edges should be as close to the real edges in the image as possible
- The algorithm should be insensitive to noise, striving to mark each edge only once

The first step of Canny’s algorithm is to reduce the amount of noise in the image by an application of a smoothing filter. The design of the filter is based on the sum of 4 exponential functions, but is often approximated by a Gaussian. By changing the Gaussian function, supplying different values of sigma, it is possible to make the edge detector work on different scale spaces. When interested in distinct and large structures, a large and even filter is preferred, while a small and sharp filter is suitable when interested in small structures.

The next step of the algorithm is to compute the gradient of the blurred image by convolving the image with an edge detection operator, e.g. Sobel, Prewitt or Roberts. These operators does not compute the gradient directly, but instead the first derivative in x- \( (G_x) \) and y-direction \( (G_y) \) separately. The strength of an edge is approximated by \( G = \sqrt{G_x^2 + G_y^2} \) and its direction by \( \theta = \arctan \left( \frac{G_y}{G_x} \right) \). The edge direction is categorized as vertical, horizontal, or diagonal. This information is later used in the process of Non-maximum suppression, an algorithm used to thin edges.

The derivative of convolution theorem states that the derivative of the convolution of two functions is the convolution of either of the two with the derivative of the other [23]. The smoothing and gradient operations are therefore combined into a single operation in practice.
To achieve better localization of the identified edge and to reduce the number of edge responses of a single true edge, non-maximum suppression is applied (Figure 6). Only edge points that have greater gradient magnitude compared to the adjacent points in the edge normal direction are considered to be a part of a true edge.

![Figure 6](image)

Figure 6: The result of applying the Canny edge detector to a disk (left) is a circle. Before non-maximum suppression the circle is several pixels wide and it is difficult to determine the exact location of the true edge (middle). Non-maximum suppression decreases the number of edge responses, and the location of the true edge can be determined with higher accuracy (right).

Several edge detection methods rely on the assumption that strong gradients more often correspond to true edges than weak ones. Typically, the gradient magnitude of a pixel must be above a specified threshold to be considered a true edge. In most cases, it is impossible to set a single threshold for when image gradients go from representing an edge, into not doing so.

To increase the reliability of the edge thresholding, the Canny edge detector uses thresholding with hysteresis. The theory behind hysteresis thresholding is that important edges are likely to be continuous curves in the image. Instead of using a single threshold, a high and a low threshold is used. The high threshold value is used to identify edges with high gradient magnitude. These edges are most likely to correspond to true edges, and are consequently marked as such. Edges identified when the low threshold is used are only considered to be true edges if they are connected to an edge that is identified during the high threshold phase. In this way, there is a better chance to keep fine grained structures of interest in the image, while non interesting ones are removed.
2.2.2 Line detection

A commonly used approach for analysing and understanding the structure of a digital image is to search for simple figures or curves, such as straight lines and circles. A prerequisite for such analysis is often that points which may form such objects have been extracted, typically by the use of an edge detector. The points extracted in the pre-processing stage do often not fully correspond to the desired curve. There may be missing points or exist small spatial discrepancies in the edge image compared to the sought curve due to a noisy image, incorrect threshold values for the edge detector etc. The task of grouping the extracted points into appropriate objects, e.g. straight lines or circles, may therefore be difficult.

The Hough transform

The Hough transform tries to resolve the problem of grouping edge points into suitable objects by performing a voting procedure. The voting procedure relies on the fact that the sought objects are parametrized. The more edge points that fit a specific instance of a parametrised object, the more votes the object will receive. Searching for lines or curves is thereby reduced to finding strong enough votes, or more correctly, finding tuples in the parameter space that most likely corresponds to objects in the image.

The complexity of the Hough transform depends on what type of object that is sought. The simplest form of the Hough transform arises when searching for straight lines, where a line is parametrized in the image space as $y = mx + b$.

The basic idea of the transform is to consider the line’s characteristics in terms of the parameters $(m, b)$, as opposed to spatial coordinates $(x_1, y_1)$ and $(x_2, y_2)$. Each line can thereby be identified by a single point in the $(m, b)$ plane. However, this is not a suitable parametrisation for most applications, since the parameter $m$ is unbounded in the case of vertical lines. Instead, lines are commonly parametrized in $r$ and $\theta$. The parameter $r$ represents the closest distance between the line and the origin, and $\theta$ the angle of the vector from the origin to the closest point on the line with respect to the x-axis (Figure 7).

The line can then be written as:

$$y = -\frac{\cos(\theta)}{\sin(\theta)} x + \frac{r}{\sin(\theta)}$$  \hspace{1cm} (1)

or equivalently

$$r = \cos(\theta) x + \sin(\theta) y$$ \hspace{1cm} (2)

where $\theta \in [0, 2\pi)$ and $r \in \mathbb{R}$. 

11
Figure 7: A straight line can be parametrized by a \((r, \theta)\) tuple. \(r\) represents the closest distance to the origin, while \(\theta\) represents the angle of the vector from the origin to the closest point on the line with respect to the x-axis.

Equation 2 is solved for every edge point and all values of \(\theta\). The number of occurrences of every computed combination of \(r\) and \(\theta\) is stored. A distinct line in the edge image will generate a high number of occurrences for a specific \((r, \theta)\) tuple, while other tuples will receive no, or a weak, contribution from the line (Figure 8).

Figure 8: Each edge pixel generates a line response for every possible line angle in the Hough space. The more pixels that lie along a potential line the stronger the response becomes in the Hough space. The potential red line receives a single vote, while the potential blue line receives multiple votes. The blue line does therefore more likely correspond to a line.
Although the Hough transform is a useful technique for curve detection, in practice it is restricted to curves with few parameters due to the exponential growth of storage needed with the increase of curve parameters. There are however various variations and extensions of the basic Hough transform, making it useful for more complex curves. If for example the gradient direction is available, it is possible to narrow the bounds of theta for each examined pixel, leading to significantly less computations.
3 Methodology

This section presents the suggested solution to robustly identify documents in digital images. A general description of the solution will first be provided, introducing the different methods involved in the solution design. A description of each individual stage, and the motivation behind its usage, will thereafter be given.

3.1 Methodology description

The solution methodology was designed to work with documents that have the same properties as the digitized documents at the KB. Even though the documents of interest have rectangular shape, the arbitrary document content makes the process to identify a general document a complex task. However, cropping a document only requires knowledge about the borders of the document. The task to identify a document for cropping can therefore be reduced to identify the four borders of the document. The methodology is based on this observation, and the sequential steps performed to identify a document are:

- Apply morphological operation, with the aim to remove smaller structures in the image
- Detect edges using Canny edge detector
- Search for straight lines using Hough transform
- Extract interest points, i.e. points likely corresponding to page corners
- Create a page candidate and investigate whether the page candidate meets the requirements to be a page or not. If the requirements are fulfilled, the page identification process is complete. If not, depending on the properties of the page candidate, repeat some or all steps of the identification process.

In the case of multi-paged documents, the qualitative document identification above is combined with a quantitative method. Typically, pages of a multi-paged document have some properties in common, such as material, size and location. By using information from several of the identified pages and adjusting the individual pages accordingly, a more robust identification can be achieved.
If there is any indication that some of the settings of the edge detector or morphological filter are incorrect, the settings are adjusted and the necessary steps of the algorithm are re-executed. In case where the document is multi-paged, a quantitative analysis is performed after all the pages have been analysed individually.
3.1.1 Morphology

The first step to identify a document in a digital image is to smooth, or if possibly, remove structures that can impose difficulties for the subsequent processing steps. Printed characters and photographs, together with the sticks that are used to fixate the document pages are objects that may be advantageous to remove. It is important that the removal can be performed without losing information about the document border.

Low-pass filters are not suitable for this procedure since they smooth all objects uniformly. Morphological filters do on the other hand affect objects differently depending on their shape and size.

If the size and shape of a flat structuring element are chosen correctly, a morphological opening/closing will, in contrast to low-pass filters, not affect the gradient magnitude at the document border. The intensity of the background and the document generally differs close to the document border. Furthermore, the local intensity extreme values of the document and the background generally differs considerably. Since the morphological operators only propagate the extreme values within its sliding window, the intensity difference at the border, and therefore the gradient magnitude, is unchanged.

In the case of a double paged document, the two pages often have different intensity levels. Furthermore, the area where the two sides meet, i.e. the actual border, is in most cases considerably darker than its surrounding. This is due to the fact that the light source is placed directly above the document and that the surface of the document tends to bend down slightly at the spine of the document (Figure 10).

![Figure 10: The document’s surface tends to bend slightly at spine which makes the border that separates the two pages take a darker colour tone.](image-url)
The choice of whether the image should be morphological opened or closed depends on the appearance of the document. A morphological opening will remove the bright regions that are completely contained by the structuring element, and thus merge small dark structures such as characters. A morphological closing will on the other hand remove small dark structures. Small intensity differences in images will tend to be smoothed regardless of which operator that is used.

For double paged documents, it is safer to use morphological opening compared to morphological closing. This is due to the fact that the border between the two sides in most cases is darker than the two sides themselves. Closing the image might therefore result in a complete removal of the border if the size of the structuring element is too large.

It is common that printed text is darker than the media it is printed on. In contrast to a morphological opening, a morphological closing generally removes the characters. The morphological closing is therefore the preferred operator when there is a distinct intensity difference between the page of interest and its surrounding.

The choice of structuring element greatly affects which objects that are merged, and the both the shape and size of the structuring element must therefore be chosen carefully. A too large structuring element may result in a connection of the document border and structures present in document, and thereby altering the sought characteristics of the document. Since the document’s orientation is unknown at this stage of the identification process, a square structuring element was chosen. The square shape ensures that edges are affected isotropically.

### 3.1.2 Edge detection

When the morphological operation has been applied, the next step is to extract the features characterizing the document. As mentioned above, the only requirement to crop a document is to identify its borders. Since discontinuities in the depth of the scene and changes in material properties along the borders of the document are likely, the image will probably have discontinuities in brightness at the document border. Hence, an edge detector is applied to reduce the greyscale image to a binary image of edge points.

Although general documents have some characteristics in common, the properties of a document can vary greatly. It is generally hard to set the correct threshold without any prior knowledge about the document. In other words, it is difficult to determine which pixels that are a part of an edge, and which are not. The differences in layout, content, and material of the documents suggest that the threshold values must be adapted for each document to extract the correct edges.

The edge strength does not only vary from document to document, but it also varies along the border of the single document. An edge detector that relies on a single edge strength threshold might have difficulties to extract the document border without generating a large amount of noise. The Canny edge
detector, which uses a more flexible threshold technique (threshold hysteresis), was therefore chosen. The Canny edge detector also provides the advantage of integrated edge thinning, reducing the number of edge responses from a single true edge.

### 3.1.3 Curve detection

The edge image generated by the edge detector will only in rare cases generate a closed curve that completely describes the border of the document. Instead, the border will most likely be divided into several edge segments, where each segment only constitutes a small part of the complete border. Also, other structures in the document such as characters and photographs will generate edge points that are not part of the document border. Deciding which edge points that actually form the border is a non-trivial task.

Most of the techniques reported in literature to detect rectangular shapes are based on finding parallel lines, and thereafter grouping them so that they form rectangular primitives, e.g. [12]. The proposed method to detect documents in this thesis is similar to this approach. The border of the document is ideally constituted of four lines, where each line is parallel to one of the other lines and perpendicular to the other two. The line primitives that are present in the digital image are extracted using the Hough transform (see Section 2.2.2).

The Hough transform offers a robust approach for grouping edge points into different known curves and structures. All edge points along a straight line, although they are not connected, are considered to be a part of a single line. A line in the image space is represented by a point in the Hough space and the task to identify a document is thereby reduced to finding four points in the Hough space. The four points $p_i = 1...4$ appear in pairs where $\theta$ of $p_1$, $p_2 \approx \alpha_1$, and $\theta$ of $p_3$, $p_4 \approx \alpha_2$. To ensure that each point represents a line that intersects two of the other lines perpendicular, it is required that $|\alpha_1 - \alpha_2| = 90^\circ$.

To find the points in Hough space that correspond to the document border is still a non-trivial task. Ideally, the two the parallel edges of the document should yield two equally strong responses in the Hough space. There is however no requirement in the proposed solution of equally strong responses for the document’s edges. One reason for this is that the documents where the spine of the book makes up one of the document’s edges often have significant variations in edge strength. The edge strength variations often cause only parts of the border to be extracted, which in turn results in a weaker response in the Hough space. The four edges of the document are in fact seldom completely straight, which is another reason the line responses differ in strength.

In general, it is not enough to investigate only the four strongest line responses in the Hough space. At the same time, investigating too many lines may result in finding false lines, i.e. lines that do not correspond to actual lines in the image.

To be able to choose how many lines that should be extracted, a simple algorithm was developed. The algorithm is designed to consider the strongest line responses first, and generate interest points from the extracted lines. If not
enough interest points have been generated to construct a document candidate, weaker edges are extracted. Additional lines are extracted by stepwise decreasing the line strength threshold. Thus, several lines of similar strength may be extracted at the same time. This behaviour might be beneficial in the case where the content of the document yield numerous edge pixels. In this case, it is often hard to determine which lines that correspond to document borders, and which that does not. By extracting several lines of similar line strength at the same time, the probability that a document border is extracted increases. The risk to underestimate the extent of the document is thereby reduced.

3.1.4 Interest point extraction and minimal rectangle generation

The process to translate the extracted lines into a cropping rectangle, i.e. a document page, is performed in two steps. First, interest points are extracted from the lines. Secondly, a minimal rectangle enclosing all of the interest points is created.

The points of interest in the image are the corners of the document, which ideally correspond to intersection points between some of the detected lines. Due to the assumed rectangular shape of the document, only intersection points where the lines intersect each other perpendicular, or close to perpendicular, are of interest. Furthermore, the surroundings of a document corner typically have unique properties that make it possible to eliminate some of the intersection point candidates.

The document corners can generally be classified as either a true corner or a pseudo corner. A true corner refers to a corner where both the document page and its enclosing frame have a corner. A pseudo corner refers to a corner where the document has a corner, and the frame containing the document does not. Pseudo corners are typically located at the point where the spine of the book meet one of the outer edges of the document (Figure 11).
Figure 11: The image to the left contains a page of a document with four true corners (marked with green squares), while the image to the right contains a page with two true corners and two pseudo corners (marked with blue circles).

Granted that the document is put on a monochrome surface, at least three quadrants of the surrounding of a true corner will have small, or no, variation in brightness. Furthermore, these three quadrants will likely have similar colour as the monochromatic surface. The colour and the degree of brightness intensity variations of the surrounding’s fourth quadrant are generally not known since it contains the document page.

In the case of a pseudo corner, at least two of the quadrants will have a colour close to the monochromatic surface, with small brightness variations. The other two quadrants contain the page to be identified, and the page opposing the latter.

The final step of the interest point extraction process is to eliminate the intersection points that do not conform with the properties of a document corner. All intersection points that remain after this elimination are candidates to be a corner of the document.

The described properties of a corner are only necessary conditions for an intersection point to correspond to a document corner. Therefore, there may exist intersection points that do not correspond to a corner after that the interest point filtering has been performed. These points will in most cases be located inside the borders of the document. Thus, one approach to approximate the document in the image is to create a minimal rectangle that encloses all the points of interest.

The rectangle that corresponds to the identified page can then be defined by its centre point, its width and height, and finally its rotation.
3.1.5 Quantitative analysis

The objects digitized at the KB consist predominantly of multi-paged documents and the pages of the document will generally have a similar size. In the case where a single page is captured at the time, with small parts of the opposing page visible, the KB tries to keep the page position fixed throughout the digitization procedure. It is therefore natural to use information from several images from the capture of the multi-paged document to determine the cropping frame of a single page. Using the sizes and the positions of the pages that were captured just before and after every single page will most likely increase the overall identification robustness. Identified pages that deviate significantly from other identified pages can then be re-evaluated and adjusted.

The objective of the document identification for multi-paged documents is to specify a cropping frame that for each image captures the complete document page. If the document does not contain any distinct outliers, i.e., pages whose size differ from the other, the cropping frames should all have the same size.

There are several ways to compute a representative value given a certain data set, e.g., mean value, median, or an application specific weighted sum. The proposed solution does not provide any measurement of how reliable the identification of a single page is. Therefore, along with the median's implementation simplicity, and the inherited insensitivity to outliers, the page size median was chosen to represent the document size.

For some of the investigated documents, it appears to exist a correlation between the position of the page in the image and its page number in the document. The more pages that have been turned, the more the captured pages are shifted in the direction of the document’s spine (see appendix C). As a consequence the median of all the pages’ positions might not be an appropriate method to determine the position of all pages in the document.

To better capture the correlation between the position and numbering of a page, a sliding window approach was used as a compliment to the median computation. For every page in the document, only a specified number of pages are considered. The position that represents a page is constructed by computing the median position of the pages in its corresponding window, yielding a unique representative for every single page in the document.
4 Implementation

This chapter presents the implementation of the described theoretical framework. First, there is an introduction that explains the choice of programming language, the development process and general design of software. Secondly, the implementation details of the core algorithms that constitute the identification process are presented.

4.1 Implementation overview

When the project was initiated, the KB used workstations running Mac operating system. However, there was a wish to keep the developed software independent of operating system, in case other hardware and software solutions would better fit their needs in the future. Another request was that the software should be easy to extend and modify; preferably developed in a programming language used at the KB. With this in mind, Java, originally developed at Sun Microsystems, was chosen as the programming language for the project. Java provides an object-oriented approach which makes the code easy to modify and extend. Furthermore, Java compiles into byte code instead of machine code. When the byte code is run, it is translated on the fly to machine code by a platform specific Java Virtual Machine, making the Java code architecture independent.

The development of the software was performed in two stages. First, each individual algorithm that constitutes a part of the document identification process was implemented and prototyped in MATLAB. MATLAB is a numerical computing environment and programming language that provides powerful scientific computing algorithms. The large amount of built in computing algorithms, together with very concise syntax, makes MATLAB an effective tool for software development and testing.

Secondly, guided by the MATLAB prototype implementation and the experiments performed, a corresponding Java software was developed. In addition to the functionality provided by the MATLAB code, the Java implementation offers an easy to use graphical interface and tools tailored for the digitization process at the KB.

The graphical interface is built upon Java’s widget toolkit Swing [5]. The program was developed according to the model-view-controller (MVC) design pattern, in an attempt to keep the coupling between different parts of the program as weak as possible. An weak coupling does generally improve the modularity and extensibility of the software.
4.2 Implementation details

The purpose of the pseudo-code in the following section is to illustrate the general functionality and the complexity of the algorithms used in the page identification process. For clarity, some implementation details are considerably simplified, or even modified.

4.2.1 Morphological operation

The implemented morphological operator constitutes of erosion, dilation and their compositions opening and closing. Furthermore, only flat square structuring elements (described in the Theoretical framework, on page 5) can be used. However, the implementation was designed in such a way that developers can design their own structuring elements if needed. Algorithm 1 illustrates the erosion morphological operator.

**Algorithm 1** Morphological erosion with a square structuring element

\[
\begin{align*}
\text{size} &\leftarrow \text{structuring element size} \\
\text{img} &\leftarrow \text{pixel array}[\text{width, height}] \\
\text{for all pixels } P(x,y) \text{ in image do} &\quad \text{end if} \\
\text{if } P \text{ in corner then} &\quad \text{else if } P \text{ on left border then} \\
\text{if } P \text{ in top left corner then} &\quad \text{if } P \text{ in top right corner then} \\
\text{window} &\leftarrow \text{img.getData}(0, 0, \text{size, size}) \\
\text{else if } P \text{ in top right corner then} &\quad \text{else if } P \text{ in right border then} \\
\text{window} &\leftarrow \text{img.getData}(\text{width} - \text{size}, 0, \text{size, size}) \\
\text{else if } P \text{ in bottom left corner then} &\quad \text{else if } P \text{ on upper border then} \\
\text{window} &\leftarrow \text{img.getData}(0, \text{height-size}, \text{size, size}) \\
\text{else} &\quad \text{else if } P \text{ on lower border then} \\
\text{window} &\leftarrow \text{img.getData}(\text{width-size}, \text{height-size}, \text{size, size}) \\
\text{else} &\quad \text{else} \\
\text{window} &\leftarrow \text{img.getData}(\text{width-size}, \text{height-size}, \text{size, size}) \\
\text{end if} &\quad \text{window} &\leftarrow \text{img.getData}(\text{width-size}, \text{height-size}, \text{size, size}) \\
\text{end if} &\quad \text{end if} \\
\text{end if} &\quad \text{end if} \\
\text{if } P \text{ on left border then} &\quad \text{output} (P(x,y)) &\leftarrow \min (\text{window}) \\
\text{else if } P \text{ on right border then} &\quad \text{end for} \\
\text{else if } P \text{ on upper border then} &\quad \text{return} \quad \text{output} \\
\text{else if } P \text{ on lower border then} &\quad \text{end if} \\
\text{else} &\quad \text{end if} \\
\text{window} &\leftarrow \text{img.getData}(x, y, \text{size, size}) \\
\text{end if} &\quad \text{else} \\
\text{end if} &\quad \text{window} &\leftarrow \text{img.getData}(x, y, \text{size, size}) \\
\text{end if} &\quad \text{end if} \\
\text{end if} &\quad \text{end if} \\
\text{output} (P(x,y)) &\leftarrow \min (\text{window}) \\
\text{end for} &\quad \text{return} \quad \text{output} \\
\end{align*}
\]
4.2.2 Canny edge detection

Algorithm 2 is an extension of the Canny edge detection implementation described by Gibara [6]. To increase the execution speed compared to the naive implementation, the 2D Gaussian function used for image smoothing is approximated by two 1D Gaussians. One Gaussian is aligned with the x-axis, and the other Gaussian is aligned with the y-axis. The smoothed image in x-direction is differentiated by convolving the values with a one dimensional first derivative of a Gaussian, aligned with the y-axis. In the same manner, the image smoothed in the y-direction is differentiated by a convolution of the first derivative of a Gaussian aligned with the x-axis.

This particular implementation of the algorithm does not contain any explicit computation of the edge direction during the non-maximum suppression phase. Instead, the edge magnitude of the neighbours is computed by first examining the sign of the partial derivative components, and then comparing their absolute values. By interpolation, it is then possible to approximate the edge magnitude of neighbouring pixels that are perpendicular to the edge orientation. If any of these pixels have generated a stronger edge response than the current analysed pixel, the latter is no longer considered to be an edge pixel.
Algorithm 2 Canny edge detection algorithm

\[ \text{img} \leftarrow \text{pixel array}[\text{width}, \text{height}] \]

\[ \text{lowThresh} \leftarrow \text{Gradient strength to continue an edge} \]

\[ \text{highThresh} \leftarrow \text{Gradient strength to start an edge} \]

\[ \text{kernel} \leftarrow \text{gaussian(size of kernel, gaussian radius)} \]

\[ \text{blurImg} \leftarrow \text{convolve(img, kernel)} \]

\[ \text{gradImg} \leftarrow \text{convolve(blurImg, diff(kernel))} \]

\[ \text{for all pixels } P \text{ in gradImg do} \]

\[ \text{gradMag} \leftarrow \text{hypot}(P) \]

\[ P', P'' \leftarrow \text{neighbouring pixels perpendicular to edge orientation} \]

\[ \text{if } \text{gradMag} \geq \text{hypot}(P') \text{ or } \text{gradMag} \geq \text{hypot}(P'') \text{ then} \]

\[ \text{edgeImg}(P) \leftarrow \text{gradMag} \]

\[ \text{else} \]

\[ \text{edgeImg}(P) \leftarrow 0 \]

\[ \text{end if} \]

\[ \text{end for} \]

\[ \text{output} \leftarrow \text{empty array}[\text{width}, \text{height}] \]

\[ \text{for all pixels } P \text{ in edgeImg do} \]

\[ \text{if } \text{magnitude}(P) \geq \text{highThresh and output}(P) == 0 \text{ then} \]

\[ \text{follow}(P, \text{lowThresh}, \text{edgeImg}) \]

\[ \text{end if} \]

\[ \text{end for} \]

Algorithm 3 FOLLOW(P, threshold, image)

\[ \text{output}(P) \leftarrow \text{magnitude}(P) \]

\[ \text{for all neighbours } P' \text{ of } P \text{ in image do} \]

\[ \text{if } \text{output}(P') == 0 \text{ and magnitude}(P') \geq \text{threshold} \text{ then} \]

\[ P' \leftarrow \text{edge point} \]

\[ \text{follow}(P', \text{threshold}, \text{img}) \]

\[ \text{break} \]

\[ \text{end if} \]

\[ \text{end for} \]
4.2.3 Line detection

Algorithm 4 illustrates a straight-forward implementation of the classic Hough transform, with the lines parametrized as

\[ r = \cos(\theta) \left( x - \frac{\text{width}}{2} \right) + \sin(\theta) \left( y - \frac{\text{height}}{2} \right). \]

---

**Algorithm 4** Hough transform algorithm

\[ \text{image} \leftarrow \text{edge image} \]
\[ \text{for } x = 1 \text{ to image width do} \]
\[ \quad \text{for } y = 1 \text{ to image height do} \]
\[ \quad \quad \text{if } (x, y) \text{ is an edge pixel then} \]
\[ \quad \quad \quad \text{for all } \theta \text{ do} \]
\[ \quad \quad \quad \quad \text{dist} \leftarrow \text{distance}(\theta, x, y) \]
\[ \quad \quad \quad \quad \text{houghSpace}[\text{dist}, \theta]++ \]
\[ \quad \quad \quad \text{end for} \]
\[ \quad \quad \text{else} \]
\[ \quad \quad \quad \text{continue} \]
\[ \quad \quad \text{end if} \]
\[ \quad \text{end for} \]
\[ \text{end for} \]
\[ \text{return houghSpace} \]

Once the Hough transform has been executed, only \((r, \theta)\) tuples that have values that that are higher than a specified threshold are considered lines. To reduce the number of responses from a single line, a strong line response also needs to be a local maxima in the neighbourhood of \((r, \theta)\) to be registered as a line (Algorithm 6).

---

**Algorithm 6** Hough transform line extraction algorithm

\[ \text{houghSpace} \leftarrow \text{Hough space array} \]
\[ \text{nbhdSize} \leftarrow \text{size of neighbourhood where to search for local maxima} \]
\[ \text{lines} \leftarrow \text{list containing lines to return} \]
\[ \text{for all values of } \theta \text{ do} \]
\[ \quad \text{for all valid values of } \rho \text{ do} \]
\[ \quad \quad \text{if } \text{houghSpace}[\theta, \rho] \geq \text{threshold then} \]
\[ \quad \quad \quad \text{peak} \leftarrow \text{houghSpace}[\theta, \rho] \]
\[ \quad \quad \quad \text{if } \text{peak} \leq \max(\text{neighbourhood}(\text{houghSpace}[\theta, \rho], \text{nbhdSize})) \text{ then} \]
\[ \quad \quad \quad \quad \text{continue} \]
\[ \quad \quad \quad \text{end if} \]
\[ \quad \quad \text{lines.add}(\text{line}(\theta, \rho)) \]
\[ \quad \text{end if} \]
\[ \quad \text{end for} \]
\[ \text{end for} \]
\[ \text{return lines} \]
4.2.4 Interest point extraction

An interest point is a point where two detected lines in the image intersect each other perpendicularly. Another requirement for a point to be an interest point is that the two intersecting lines individually generate a response stronger than a set line strength threshold in the Hough space.

The extraction of interest points (Algorithm 7) are conducted in two steps. First, a line strength threshold based on the strongest response in the Hough space, is set. The line strength threshold is then decreased until either of the two conditions is true:

- At least four interest points are non-collinear
- The line strength threshold is lower than a preset threshold

In the latter case, a flag weakEdges is set, and the algorithm returns. The flag is used in later processing steps to determine what adjustments that should be made to increase the line strength in the image.

In the case where at least four intersection points have been found, the intersection points are filtered. The neighbourhood of each intersection point is investigated and intersection points unlikely to correspond to a document corner are removed. If enough intersection points remain after filtration, the algorithm returns. Otherwise, the line strength threshold is decreased, and the interest point extraction process is restarted.

**Algorithm 7** Interest point extraction algorithm

```
\texttt{houghSpace} \leftarrow \text{Hough transform of edge image}
\textbf{for} \ \texttt{lineThreshold} = \text{high} \ \textbf{to} \ \text{low} \ \textbf{do}
\texttt{lines} \leftarrow \text{getLines(houghSpace, lineThreshold, neighbourhoodSize)}
\texttt{inter} \leftarrow \text{perpendicularIntersections(lines)}
\texttt{nonCollinearPoints} \leftarrow \text{removeCollinear(inter)}
\textbf{if} |\texttt{nonCollinearPoints}| \geq 4 \ \textbf{then}
\hspace{1em} \textbf{for all} \ \texttt{intersection points} \ p \ \textbf{in} \ \texttt{inter} \ \textbf{do}
\hspace{2em} \textbf{if} \ \text{all or} \leq 1 \ \texttt{quadrants} \ \text{surrounding} \ p \ \sim \ \text{background} \ \text{colour} \ \textbf{then}
\hspace{3em} \text{remove} \ p \ \textbf{from} \ \texttt{inter}
\hspace{1em} \textbf{end if}
\hspace{1em} \textbf{end for}
\hspace{1em} \textbf{if} |\texttt{inter}| \geq 4 \ \textbf{then}
\hspace{2em} \textbf{return} \ \texttt{inter}
\hspace{1em} \textbf{end if}
\textbf{end if}
\textbf{if} \ \text{line strength} \leq \text{threshold} \ \textbf{then}
\hspace{1em} \texttt{weakEdges} = \text{true}
\hspace{1em} \textbf{return}
\hspace{1em} \textbf{end if}
\textbf{end if}
\textbf{end for}
```
4.2.5 Minimal enclosing rectangle

The first step to create a minimal rectangle that encloses all of the extracted interest points is to create the convex hull (Algorithm 8). Then, since at least one of the edges of the convex hull must be contained by one of the edges of the minimal rectangle, the convex hull is rotated such that one of its edges are parallel to the y-axis [3]. The minimal bounding rectangle can simply be computed by finding the most right, most left, most down and most up points of the rotated convex hull. The rotation and minimal bounding rectangle procedure is repeated until all edges of the convex hull have been aligned with the y-axis at least once. The minimal bounding rectangle is attained by reversing the rotation of the smallest of the bounding rectangles (Algorithm 9).

Algorithm 8 (Gift wrapping) A convex hull generation algorithm

\[
\begin{align*}
\text{pointOnHull} & \leftarrow \text{leftmost point in the set } S \\
i & \leftarrow 0 \\
\textbf{while} \ \text{pointOnHull} \neq P[0] \ \textbf{do} \\
& \text{\hspace{1cm} } P[i] \leftarrow \text{pointOnHull} \\
& \text{\hspace{1cm} } \text{end} \leftarrow S[0] \\
& \text{\hspace{1cm} } \textbf{for} j \ \text{from} \ 1 \ \text{to} \ |S| - 1 \ \textbf{do} \\
& \text{\hspace{2cm} } \text{\hspace{1cm} } \text{\hspace{1cm} } \text{if} (\text{end} = \text{pointOnHull}) \ \text{or} \ (S[j] \text{ is on left of line from } P[i] \ \text{to} \ \text{end}) \ \text{then} \\
& \text{\hspace{2cm} } \text{\hspace{1cm} } \text{\hspace{1cm} } \text{\hspace{1cm} } \text{end} \leftarrow S[j] \\
& \text{\hspace{2cm} } \text{\hspace{1cm} } \text{\hspace{1cm} } \text{end if} \\
& \text{\hspace{1cm} } \text{\hspace{1cm} } \text{end for} \\
& \text{\hspace{1cm} } i \leftarrow i + 1 \\
& \text{pointOnHull} \leftarrow \text{end} \\
\textbf{end while} \\
\text{return } P
\end{align*}
\]

Algorithm 9 Minimal area enclosing rectangle algorithm

\[
\begin{align*}
hull & \leftarrow n \times 2 \ \text{matrix containing all points of the convex hull} \\
angles & \leftarrow \text{the angles for all the edges of the convex hull wrt the x-axis} \\
minimumArea & \leftarrow \infty \\
\textbf{for all} \ \text{edges } e \ \text{in} \ hull \ \textbf{do} \\
& \ \text{rotationMatrix} \leftarrow \text{rotation matrix for edge } e \\
& \ \text{rotatedHull} \leftarrow hull \cdot \text{rotationMatrix} \\
& \ l, r, d, u \leftarrow \text{the left, right, down and upmost points of } \text{rotatedHull} \\
& \ \text{area} \leftarrow |((l.x - r.x)(d.y - u.y))| \\
& \ \text{if} \ \text{area} \leq \text{minimumArea} \ \text{then} \\
& \ \ \text{R} \leftarrow \text{minimal bounding rectangle enclosing } l, r, d \ \text{and} \ u \\
& \ \ \text{minimalRect} \leftarrow R \cdot \text{rotationMatrix}^T \\
& \ \ \text{end if} \\
\textbf{end for} \\
\text{return } \text{minimalRect}
\end{align*}
\]
4.2.6 Complete document identification

The final step of the document identification process is to verify that the generated rectangle corresponds to a document (Algorithm 10). The properties examined are: area of the document candidate, line strength of the detected lines in the image and the number of identified lines. First and foremost, the quality of the line detection is examined. If the flag `weakEdges` was set during the interest point extraction, the edge detector sensitivity level is increased one step, and the edge image is recomputed. If the sensitivity of the edge detector is already set to the highest setting, the morphological filter size is adjusted according to

\[ ns = os + \text{sign}(d) \left( 1 + |c \cdot \ln (1 + |d|) | \right) \]

where \( os \) denotes the old structuring element size, \( d \) the difference between the number of desired and current number of identified lines, and \( c \) is a constant.

In the experiments conducted as a part of this thesis, \( c \) was set to 1 and \( d \) to 20.

If the interest point extraction was successful, i.e. the flag `weakEdges` was set to false, a document candidate is created. All candidates that are smaller than a specified minimum document size are rejected. Depending on the number detected of lines, the edge detector and the morphological filter are adjusted, and a new document candidate is computed.

If the size of the candidate is larger than the specified minimum document size, the candidate is accepted and the identification process is considered successful.
Algorithm 10 Interest point extraction algorithm

while $tries \leq maxTries$ do

    $image \leftarrow$ morphologicOperation($originalImage$)
    $edgeImage \leftarrow$ edgeDetector($image$)
    $houghSpace \leftarrow$ hough($edgeImage$)
    $inter \leftarrow$ result from interest point extraction

    if $weakEdges = true$ then
        tries++
        if number of lines $\geq$ threshold then
            increase size of structuring element
            goto morphologicOperation
        else
            increase edge detector sensitivity
            goto edgeDetector
        end if
    end if

    $minimalRectangle \leftarrow$ enclosingRectangle($inter$)
    if area of rectangle $\leq$ areaThreshold then
        if number of lines $\geq$ threshold then
            decrease edge detector sensitivity or increase size of structuring element
        else
            increase edge detector sensitivity or decrease size of structuring element
        end if
    else
        return
    end if
    tries++
end while
5 Experiments and results

This section presents the results obtained from experiments. The data set used for the experiments and the criteria to classify the quality of the document candidates are presented. All images of the data set were analysed by DocCrop, and the results of the execution are presented in the end of this section.

5.1 Experiments

Three different data sets ($D_1$, $D_2$ and $D_3$), supplied by the KB, have been investigated (Figure 12). $D_1$ is a hardcover book which contains text, monochrome images and numerical data tables. It was captured with a resolution of 1501 x 2253 pixels. $D_2$ is price information booklet, where the pages for the most part are covered by monochrome tables. $D_2$ was captured with a resolution of 1831 x 2773 pixels. The last data set, $D_3$, was captured with a resolution of 3744 x 5616 pixels. It is a hardcover book which contains text passages, complex grey-scale images, and coloured headlines. The characteristics of the data sets are given in Table 1.

The size of the images is automatically adjusted by the program before the identification process begins. In the performed experiments, the images have been shrunk such that the longest side of the images consists of 600 pixels.

The experiments were performed on a laptop with an Intel Core i5 M460 (2.53 GHz) CPU, 3 MB cache memory and 4 GB DDR3 RAM. The computer runs on a Windows 7 home edition operating system.

<table>
<thead>
<tr>
<th>Data set</th>
<th>colour</th>
<th>fixating stick</th>
<th>photos</th>
<th>illustrations</th>
<th>tables</th>
<th>lines</th>
<th>hardcover</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_1$</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>$D_2$</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>$D_3$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: The characteristics of the images in the data sets $D_1$, $D_2$ and $D_3$ are marked by crosses in the corresponding column. $D_1$, for example, contains images where some of the pages needed to be fixed by sticks (due to the rise or curl of those pages). Furthermore, some of the images in $D_1$ have illustrations, tables and/or lines.
Figure 12: Representative pages of $D_1$ (top left), $D_2$ (top right) and $D_3$ (bottom centre).
Each image that is analysed by the software is classified into one of the three categories denoted by the subscript $s$, $c$ and $f$. The subscript $s$ indicates that the page has been successfully identified, the subscript $c$ that the identification is close to successful, and the subscript $f$ that the identification has failed.

A page is considered to be successfully identified if:

- All the information in the true page, such as text, images and tables, is contained within the border of the cropping frame
- There is less than a 10% difference in size of the cropping frame and the true page

For a page to be considered close to successfully identified:

- Most of the information of the page is retained. Parts of tables, photos and characters close to the page border may be missing
- There is less than a 10% difference in size of the identified page and the true page

Otherwise, the page identification has failed.

5.2 Results

Table 2 presents the number of successful, close to successful, and failed document identifications. Without any quantitative adjustments, the solution shows an overall identification success rate of 96%. Out of the three data sets, $D_1$ is the most difficult document to identify, with a identification success rate of 92%. The identification failed completely only for 1 of the 120 analysed document pages in the three sets.

<table>
<thead>
<tr>
<th>Data set</th>
<th>$N_s$</th>
<th>$N_c$</th>
<th>$N_f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_1$ (50 images)</td>
<td>46 (92%)</td>
<td>4 (8%)</td>
<td>0 (0%)</td>
</tr>
<tr>
<td>$D_2$ (22 images)</td>
<td>21 (95%)</td>
<td>0 (0%)</td>
<td>1 (5%)</td>
</tr>
<tr>
<td>$D_3$ (48 images)</td>
<td>48 (100%)</td>
<td>0 (0%)</td>
<td>0 (0%)</td>
</tr>
<tr>
<td>Total (120 images)</td>
<td>115 (96%)</td>
<td>4 (3%)</td>
<td>1 (1%)</td>
</tr>
</tbody>
</table>

Table 2: The identification results prior to quantitative analysis and adjustments. $N_s$ denotes the number of successful page identifications, $N_c$ the number of close to successful page identifications and $N_f$ the number of failed page identifications.

When quantitative adjustments were made, the identification success rate increased for two out of the three data sets, and the overall identification success rate reached 98%. Furthermore, none of the identification failed completely, as shown in Table 3.
Table 3: The identification results after quantitative analysis and adjustments have been made. $D_s$, $D_c$ and $D_f$ denotes the difference between the number of page candidates before and after quantitative analysis for the categories $s$, $c$ and $f$.

Table 4 presents the execution times to analyse the different data sets. The average time to identify a document differs greatly depending on data set; pages in $D_1$ and $D_2$ are identified in well below two seconds, while pages that belong to $D_3$ are identified in just over three seconds.

Table 5 displays the execution time distribution over the different parts of the identification procedure. The amount of work required to identify a page consists predominantly of edge detection, morphological operations, and image loading. The edge detection constitutes the largest part of the total workload in $D_1$ and $D_2$ (64% and 73%), while image loading constitutes the largest workload for $D_3$ (51%). Less than 2% of the total execution time composes of Hough transform, interest point extraction and minimal rectangle generation computations.
6 Discussion

In this chapter the results of the performed experiments are discussed. First, a brief discussion on the solution methodology and the developed software is presented. Then, the observations made during the conducted experiments of the methodology’s basic parts are presented and discussed.

6.1 Solution methodology and software

The experimental results of the proposed method are promising. The results suggest that the methodology may be a viable alternative for automatic document identification.

The most striking characteristic of the methodology is its robustness. This is illustrated by the fact that the composite identification success rate, i.e. the identified pages were categorized as 0 or c, exceeds 95% for all of the examined documents. The identification success rate could possibly be increased even further if some user defined variables, e.g. initial morphological structuring element size and shape, are set for each document separately. The interaction between the appearance of the document and the variables are however non-trivial. For casual users it may be beneficial to provide a limited number of preset variable configurations.

The majority of the computational work to identify a document is spent extracting the edge primitives within the image. However, as the time to load the image into memory also constitute a significant part of the computational work, optimization of the single thread edge detector implementation is not deemed worthwhile.

Most modern computers have multiple cores and to re-write the code for parallel execution can be an efficient way to decrease the execution time. In fact, large parts of the Canny edge detector and the Hough transform can by quite simple means be parallelised.

The methodology relies on a few basic properties, such as minimal document size, number of detected lines and number of extracted interest points, to adapt to the documents’ various appearances. These properties are generally not sufficient to determine whether the document identification is successful or not. A manual inspection of the identification is currently necessary to verify its correctness. To fully utilize the advantages of automatic identification, a more reliable technique to verify the identification is required.

6.2 Morphology

To a large extent, the morphological operator achieved the objective to even out structures of no interest, while maintaining the important features of the investigated document. The morphological operator was proved to be particularly important when the document contains photographs along its border, as seen in Figure 13. Without the application of the operator, the subsequent edge detection captures the small brightness differences in the photograph instead of
the border of the document. The response in the Hough space of the document borders therefore becomes weaker, and the risk of failed document identification increases.

Figure 13: The image at the top left is the original image file, and the top right image is the edge image generated when no morphological operator was applied. At the bottom: the left image is the result of a greyscale conversion of the original image, followed by an opening operation. The right image is the edge image that corresponds to bottom left image.
The proposed solution generally relies on an opening operator, but, as can be observed in Figure 14, the closing operator can produce a better result. An observation made from the experiments was that the use of a closing operator however increases the risk to remove the border along the spine of the document. This was primarily due to the fact that the spine of the documents generally has a lower brightness intensity compared to its surrounding. Parts of such borders, which can be completely enclosed by the structuring element associated with a closing operator, can disappear in the output image when the closing operator is applied.

Figure 14: To the left: the original image in figure 13 has been converted to greyscale and closed. While the smaller structures, such as text and parts of the photograph, has been evened out, the parts that form the border of the document has been unaffected. To the right: the edge image generated from the closed image contains significantly less noise than the edge images generated from both the raw and opened images.
6.3 Edge detection

For most of the investigated images, the Canny edge detector produced edge maps that captured sufficient parts of the document borders to allow for successful document identification. However, the need of a priori knowledge of the border edge strength to produce a good result is a clear disadvantage of this method. The solution suggested in this paper does not collect any knowledge about the border edge strength before the edge detection phase. Instead it relies on posterior knowledge, e.g. number of found lines and document candidate area, to readjust the edge thresholds. Because of this, and the fact that the edge strength of the document border vary from image to image, several iterations of the edge detector might be needed to create a good edge map.

Furthermore, the implemented Canny edge detector only works on single channel images, and can therefore not utilize all the information of RGB images. An implementation, similar to colour Canny edge detector presented by Nadernejad et al. [21], could possibly increase the reliability of the edge detection process.

There exist edge detectors that are less reliant on the edge strength compared to the Canny edge detector. The boolean function based edge detector adapts the edge strength threshold for each pixel by studying its surrounding pixels’ intensity [21]. This method does however often produce spotty or double edges.

A simple edge detector was developed to investigate whether the gradient orientation is a sufficient condition to determine whether a pixel belongs to the document border or not. Instead of edge strength, the edge detector relies solely on edge orientation. First, the image gradient direction is computed for every pixel in the image. Then, the pixels where the gradient direction is within a specified interval are stored as edge pixels. To reduce noise, edge structures whose orientation does not conform with the investigated gradient direction are removed. This is done by morphologically open the edge image with a structuring element that extends perpendicular to the gradient direction.

If several edge orientations are to be examined in an image, a edge map for each orientation is generated. For each and every edge map, a morphological opening, with a suitable oriented structuring element, is applied. The edge maps are then merged. Finally, non-maximum suppression is applied to achieve better edge locality.

This method relies on the fact that the edge orientation of the document is either known beforehand, or can be computed. Due to the nature of the documents, only two edge directions needs to be investigated. The technique used to determine the document orientation in Figure 15 first extracted the 1000 pixels that had the strongest gradient magnitude. The orientation of the strongest line in the resulting image was then used to determine the document orientation.
Figure 15: The edge detector that does not rely on edge strength was applied to the left image. Pixels where the gradient are horizontal or vertical was extracted, and these are illustrated in middle image. The image to the right is the final result of the edge detector. Morphological opening and non-maximum suppression has been applied to reduce noise and improve edge localization.

The edge detector which does not rely on edge strength captures the document border in the majority of the investigated images. Although some of these documents contain complex objects, such as photographs, the noise levels in the edge images are low.

This detector is however not suitable for documents where the brightness intensity vary significantly along its borders. This occurs for example when a photograph, see Figure 16, is located next to the document border. The gradient direction along such borders tend to vary, and thus yield a poor result.

Figure 16: The photograph printed on the document (left image) extends to the document border. This results in brightness variations along parts of the document border, which in turn leads to that the gradient direction varies along the border. The output of the edge detector (right image) is therefore very poor.
6.4 Line detection

One of the Hough transform’s strengths is to extract shapes, even if parts of the shape is missing, possibly due to image or edge imperfections. This is an important feature when document borders are to be extracted since the edge pixels that constitute the border seldom are cohesive.

To detect the document border without extracting to many non-border segments, it is important to set an appropriate line threshold value. This is a nontrivial task which in general requires information about the document, such as its size, contents, and regularity of its border.

The proposed method initially extracts lines with a high preset threshold. Generally, if few lines and/or interest points was extracted, the threshold is decreased. This approach works well for almost all of the examined images. However, for documents with slightly curved borders, a tendency where many non-border segments are extracted is observed. This is due to the fact that the response for a curved border in the Hough space is significantly weaker compared to the response of a straight border. The weak line response of the borders requires a low threshold value, causing other weak line segments to be extracted as well. This is an effect of the assumption that each side of a document page can be approximated by a single straight line. In the case of documents with curved borders, a more flexible border parametrisation would be preferable.

Liu et al. [16] proposed an algorithm that merges straight line segments into more general structures by using gradient information of the edge pixels. Thus, the short straight line segments of a curved border could be merged into a single strong line segment. The weak line segments that remain could thereafter be eliminated without losing information about the document border.

In addition to produce stronger line responses of curved borders, such approach would make it possible to compute the endpoints of the line segments. The line extension information could be used in the interest point extraction procedure to reduce the number of interest points that does not correspond to a document corner.

Due to time constraints and uncertainty of the effects of such a solution, the algorithm was not integrated into the suggested solution.

6.5 Interest point extraction

For many investigated images, the interest point extraction procedure generates several points that do not correspond to a document corner. This is partly due to the difficulty of describing an arbitrary document corner, and partly due to the fact that the suggested method fails to use some of the available information. The primitives that are extracted in the line detection process contain information about their own direction, position and strength, but no information of their extension. To be able to use line extension, a reliable line combination technique similar to the one suggested by Liu et al. [16] needs to be integrated into the solution. Hence the initial interest point extraction procedure
may identify points as corners, even though there are no intersecting lines at this location in the actual scene. These false interest points arise, for example, when an image contains several small rectangular objects such as photographs, see Figure 17.

Many of the initially extracted interest points can be removed since their surrounding does not match the characteristics of a document corner. In Figure 17, all but two of the final extracted interest points are located along the document corner, and none of these two interest points affected the result of the minimal rectangle procedure.

Figure 17: The line primitives, marked as red lines, were extracted with the help of the line detection procedure. In the left image the intersection points where no true line intersection occur are marked with red squares. The blue circles in the right image are the interest points which were removed because of the properties of their surroundings. The remaining interest points in the right image are marked by green crosses.

A process where the neighbouring edge pixels of the interest points are analysed might reduce the number of incorrect extracted interest points. However, this requires an edge detector that has high reliability and can produce high quality edge maps. Since a weak edge response of a document corner can lead to erroneous elimination of correct interest points.

When the Canny edge detector is used as in the presented solution, complex document borders, e.g. where photographs are a part of the border, can cause unreliable results. Figure 18 illustrate the importance of correct detector settings. The variation of the document border’s edge strength makes it difficult to capture the complete border without a significant amount of noise. In the rightmost image a too low edge threshold was used, resulting in a loss of edge information.
An alternative to use a single edge map to perform the interest point analysis is to use local edge maps for every interest point. Thus it is possible to adapt the settings of the edge detector to the different conditions in the image. If a threshold reliant edge detector is used, the threshold can for example be dynamically adjusted until a certain number of edge pixels have been identified close to the interest point. A non edge strength reliant edge detector (similar to the edge detector described in section 6.3) can be adapted by altering the interval of gradient orientations for which pixels are considered edge pixels.

![Figure 18](image)

**Figure 18:** The border of the document in the leftmost image vary in strength and it is a delicate task to set correct threshold values. The middle picture is the outcome of an edge detection where the threshold values have been set adequately, while the rightmost image is the outcome where the threshold values were set to low.

### 6.6 Minimal rectangle generation

For all of the investigated images, the edges of the minimal rectangle are almost parallel to the document borders, and due to the digitization setup, also close to parallel to the image borders. In fact, the minimal rectangle is in most cases very similar to the rectangle defined by the corners \([x_{\text{min}}, y_{\text{min}}], [x_{\text{min}}, y_{\text{max}}], [x_{\text{max}}, y_{\text{min}}] \) and \([x_{\text{max}}, y_{\text{max}}]\), where \(x_{\text{min}}/x_{\text{max}}\) and \(y_{\text{min}}/y_{\text{max}}\) are the minimum/maximum \(x\)-respectively \(y\)-coordinate of all the interest points.

Even though there are no clear advantages to use a rotational independent rectangle generation algorithm when the document is easy to align, there are situations where it may be useful. Large posters, for example, may be difficult to align perfectly. With rotational independent rectangle generation, the program can create a better approximation of the document (Figure 19).
To illustrate the advantage of a rotational independent rectangle generation the document has been tilted significantly. From the same set of interest points two document candidates were generated (marked by a green rectangle) by the program. The candidate to the left is generated with rotational independent method, while the right candidate is generated by a rotational dependent method.

The relative low computational cost (see Table 4) of the minimal rectangle generation procedure suggests that it is impossible to increase the performance using another rectangle generation method.

6.7 Quantitative analysis

The quantitative analysis improved the reliability of the document identification for all of the three investigated data sets. Figures 34 - 39 in Appendix C show that the true positions of the document pages often differ from the identified positions.

The positional error is considerably less in the vertical direction. This is most likely due the fact that the vertical position primarily is determined by the horizontal borders. These borders are adjacent to the monochrome background and are therefore easy to identify. Generally, there is almost no vertical displacement of the document pages in their respective image series. The true vertical position is therefore close to constant, and the vertical position outliers in the identified document data are generally erroneous. The (few) outliers are effectively removed by the quantitative analysis procedure and the adjusted identified vertical position conform very well with the true vertical position.

The horizontal position of the documents varies more than vertical position
in all of the three data sets. This is because the spine of the document moves as the pages are flipped during the digitization process. In addition to the horizontal displacement of the document, the width of the pages varies. For data set $D_3$ there is a width difference of about 10\%, or approximately 100 pixels, between the smallest and the widest page of the set. The fixed size page frame, calculated as the median of all the document pages, is therefore considerably larger than some of the actual pages. A slight horizontal displacement of the cropping frame, i.e. positional error of the identified page, does therefore in most cases not lead to an incorrect cropping.

Out of the 120 analysed images, only 3 of them were cropped so that information of the page was lost. The information that was lost, was in each case close to the spine of the document. This consistent result suggests that an expansion of the cropping frame in the direction towards the document spine could increase the reliability of the identification process. Alternatively, a biased quantitative analysis scheme which displaces the cropping frame in direction towards the spine of the document could be used.

Figure 20: The 3D graph illustrates the positional movement of the left and right pages in the data set $D_3$. More detailed results of the page movement are presented in Appendix C.
7 Conclusions and Future Work

This section provides conclusions made from the software development and the conducted experiments. At the end, suggestions of future work to improve the solution methodology are presented.

7.1 Conclusions

The solution methodology provides, given a setup that conforms with the restrictions that apply to KB’s digitization process, a robust automatic solution to identify documents in digital images. This allows the KB to improve the efficiency and decrease the amount of manual labour needed in their digitization process. The interface of the developed software enables the user to easily process large batches of images without almost any manual work.

The solution methodology achieved a 98% overall identification success rate when 120 images, from three different data sets, were analysed. However, the methodology does not provide any quality measure of the identification. A manual inspection of the identification is therefore required to verify that the identification is correct. Until a reliable quality measure has been developed, the practical applications of the software are slightly limited.

In terms of efficiency the solution methodology is a viable alternative to manual document identification. For the majority of the studied images the software was able to identify the digitally captured documents within a few seconds. Furthermore, the accuracy of the identification of individual pages is in most cases very good. For some multi-paged documents, the identification may be even more accurate than the manual identification done at the KB. This is due to the fact that the software computes individual positions for the cropping frame in each image, while one and the same cropping frame position is used in the manual identification.

The suggested solution relies inter alia on the placement of the document and the background surface colour in the images of the captured images at the KB. The proposed method may therefore not be a suitable solution when interested to identify documents in a setup that differ from the KB’s.
7.2 Future Work

The proposed solution performed very well for all of the three investigated data sets. Below, possible topics of further research together with concrete adjustments to the implemented software are presented.

Edge detection

The proposed solution uses only brightness levels present in the input image, and not the complete colour information, to create the edge map. This simplifies the edge detection implementation, but might reduce the quality of the edge detection since the level of detail in the input image decreased. A thorough study of the effects of multi-channelled edge detection is desirable, since it is far from certain that increased level of detail improves the document border detection.

The current morphological pre-processing relies on the fact that it is possible to find extreme (brightness) values in the neighbourhood of every pixel. Finding such extreme values in a multi-channel image is considerably more difficult task compared to finding extreme values in a single channel image. A multi-channel image requires that the significance of the different channels are weighted, which may be a delicate task. Whether the morphological pre-processing still can be used or not in the case of a multi-channelled edge detector needs to be clarified.

The possibility to use an edge detector that relies on edge orientation in contrast to edge strength was briefly investigated during the development of the proposed solution. The results obtained from the experiments of such an edge detector are promising and suggests that this category of edge detector can be a viable alternative to the Canny edge detector.

Line detection

The Hough transform implementation used to detect lines does not utilize edge pixels orientation. For every edge pixel there are as many contributions to the Hough space as there are permissible line angles. In the case where the generated edge map is noisy and contains large amounts of edge pixels, both the efficiency and reliability of the line detection decreases.

The results obtained from the investigations of the edge detector that relies on edge orientation hints that edge orientation could be used to detect lines. Edge pixels that all constitute a single document border tend to have the same orientation. An implementation of the Hough transform that only generate line responses for angles that are, or are close to, perpendicular to the edge pixel's orientation is therefore an option which should be explored.

Interest point extraction

All points where at least two of the detected lines intersect perpendicularly are initially considered to be interest points. The candidate interest points where the colour of their surrounding does not match the characteristics of a corner
are removed. However, this is often not sufficient to remove all false interest points. A method which also use additional information, e.g. pattern of colour and brightness differences, of the interest point candidate surrounding’s may reduce the number of false interest points.

**Identification quality measure**

A reliable measure to determine the quality of the document identification is a very important feature for the identification to be handled automatically. The measure could, in addition to determine which of the identified pages that needs to be manually validated, also be used as a tool to improve the quantitative analysis.

Currently, the quantitative analysis works well to correct documents where the document pages are of the same size. The analysis is however of limited use when the document pages vary in size.

A measure of the identification quality could be integrated into the quantitative analysis to determine which of the identifications that needs to be adjusted. Identified pages with high quality measure could for example be kept unchanged, and possibly even be categorized into different page types. These pages could then constitute the basis the quantitative analysis of the remaining pages. Thus, documents with pages of various sizes could be analysed in a more reliable manner.

There are a few areas in the image that most likely should be examined to measure the identification quality of a document reliably. First and foremost the document’s immediate surrounding and the document border needs to be inspected. Depending on whether a page is a part of a document spread or not, there should be limited brightness variations along three or four of the page’s sides. Furthermore, the area adjacent to these sides should have the same colour as the material the document is placed upon.

It is rare that the border of the document coincide perfectly with the rectangular shaped border of the identified page. An edge tracing algorithm that is guided by the border of the identified page, combined with an analysis of the edge’s strength and orientation, could enable a more accurate analysis of the document border.

To further ensure that the identified document correspond to real document, the structure of the content in the identified should be analysed. This task is most likely very complex, and might not be necessary to provide a reliable identification measure.
7.3 Acknowledgements

I would like thank my supervisor Henrik Johansson for his support during the methodology development and his invaluable assistance revising this report. I would also like to thank Anders Brun at the Centre of Image Analysis for his insightful comments on the draft of this report and his ideas to improve the methodology. Finally, I would like thank the KB for giving me permission to use some of their digitized material in my report.
References


A Visual tour of DocCrop

DocCrop is software that can identify printed media in digital images. The program was primarily developed to identify single- and multi-paged documents, but it can also identify photographs and posters. DocCrop is a part of Tomas Toss master thesis, and the theory on which DocCrop relies are described in detail in the paper “Automatic identification and cropping of rectangular objects in digital images” written by Tomas Toss.

This guide presents through pictures and short text passages the DocCrop interface and its core functionality. The pictures describe a typical use of the program; from choosing the image files and adjusting program settings to inspecting the output.

![Image of DocCrop interface]

Figure 21: The main screen of DocCrop. The screen features the action panel to the left, the status panel in the top right, and document settings at the bottom right.
Figure 22: The document choosing dialogue. The dialogue is accessed from the main screen using the corresponding button in the action panel. It is possible to select individual image files and/or folders for processing. When selecting folders, all the image files inside are considered to be from a single multi-paged document. Files that are selected individually are considered to be single-paged documents.

Figure 23: The main screen of DocCrop with the image batches to be analysed now shown in the status panel. Specific batch settings can be made from the settings panel at the bottom right. General program settings can be adjusted by opening a separate settings window from the menu bar.
Figure 24: The settings window. Sub-windows can be accessed for general application settings, standard document settings and algorithm settings.

Figure 25: The main screen of the settings window. In this screenshot the output file location and document padding have been altered.
Figure 26: The algorithm screen of the settings window, displaying fast, normal and custom algorithm setting mode. In the customized mode it is possible to freely adjust the algorithm threshold and constant values, e.g. maximal number of iterations, working image size and minimal document size.
Figure 27: The document preview window illustrate the most recent document candidate by a coloured semitransparent rectangle. This window is updated as new document pages are identified.
Figure 28: A document candidate for one of the left pages in the image batch is displayed in the preview window. In the subsequent quantitative analysis, the left and right pages are considered separately.

Figure 29: As the identification process proceeds, the status panel is updated continuously.
Figure 30: The identification process is complete. The coordinates of the four corners of each document are stored, together with the absolute path of the image file, in a text file. In the status panel of the main window, the user have been notified that the identification is complete.

Figure 31: If the user so chooses, the cropping recommendations are stored as images.
Figure 32: Once the identification process is complete, additional batches can be chosen for further execution. Here, two multi-paged documents, visualized as two folders, have been chosen.

Figure 33: DocCrop supports several file formats, e.g. tiff, png and bmp.
<table>
<thead>
<tr>
<th>Page number</th>
<th>Total time (ms)</th>
<th>Rectangle generation (ms)</th>
<th>Interest point (ms)</th>
<th>Homography (ms)</th>
<th>Edge detection (ms)</th>
<th>Morphological (ms)</th>
<th>Image loading (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>42.569</td>
<td>27</td>
<td>15</td>
<td>16</td>
<td>0.2</td>
<td>4.2</td>
<td>0.7</td>
</tr>
<tr>
<td>1</td>
<td>17.0</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>20.0</td>
<td>12</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>18.0</td>
<td>12</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>19.0</td>
<td>12</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>18.0</td>
<td>12</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>19.0</td>
<td>12</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>18.0</td>
<td>12</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>19.0</td>
<td>12</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>18.0</td>
<td>12</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>19.0</td>
<td>12</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>18.0</td>
<td>12</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>19.0</td>
<td>12</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>18.0</td>
<td>12</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>19.0</td>
<td>12</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>18.0</td>
<td>12</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>19.0</td>
<td>12</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>18.0</td>
<td>12</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>19.0</td>
<td>12</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>18.0</td>
<td>12</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>19.0</td>
<td>12</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>21</td>
<td>18.0</td>
<td>12</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>22</td>
<td>19.0</td>
<td>12</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>#</td>
<td>Page number</td>
<td>Time (ms)</td>
<td>Total (ms)</td>
<td>Time generation</td>
<td>Edge detection</td>
<td>Hough transform</td>
<td>Image transformation</td>
</tr>
<tr>
<td>----</td>
<td>-------------</td>
<td>-----------</td>
<td>------------</td>
<td>----------------</td>
<td>----------------</td>
<td>-----------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>0</td>
<td>2880</td>
<td>0</td>
<td>10</td>
<td>108</td>
<td>117</td>
<td>119</td>
<td>120</td>
</tr>
<tr>
<td>1</td>
<td>2910</td>
<td>0</td>
<td>10</td>
<td>108</td>
<td>117</td>
<td>119</td>
<td>120</td>
</tr>
<tr>
<td>2</td>
<td>3256</td>
<td>0</td>
<td>10</td>
<td>108</td>
<td>117</td>
<td>119</td>
<td>120</td>
</tr>
<tr>
<td>3</td>
<td>3400</td>
<td>0</td>
<td>10</td>
<td>108</td>
<td>117</td>
<td>119</td>
<td>120</td>
</tr>
<tr>
<td>4</td>
<td>2890</td>
<td>0</td>
<td>10</td>
<td>108</td>
<td>117</td>
<td>119</td>
<td>120</td>
</tr>
<tr>
<td>5</td>
<td>3256</td>
<td>0</td>
<td>10</td>
<td>108</td>
<td>117</td>
<td>119</td>
<td>120</td>
</tr>
<tr>
<td>6</td>
<td>3315</td>
<td>0</td>
<td>10</td>
<td>108</td>
<td>117</td>
<td>119</td>
<td>120</td>
</tr>
<tr>
<td>7</td>
<td>3315</td>
<td>0</td>
<td>10</td>
<td>108</td>
<td>117</td>
<td>119</td>
<td>120</td>
</tr>
</tbody>
</table>

The table represents 48 uncompressed TIF images with a resolution of 744x5616 pixels.
C Quantitative analysis - page movements

Quantitative analysis is throughout this appendix denoted by QA.

Figure 34: The right pages of $D_1$'s horizontal (top) and vertical (bottom) positions over time are illustrated by two graphs.
Figure 35: The left pages of $D_1$’s horizontal (top) and vertical (bottom) positions over time are illustrated by two graphs.
Figure 36: The left pages of $D_1$’s horizontal (top) and vertical (bottom) positions over time are illustrated by two graphs.
Figure 37: The right pages of $D_2$'s horizontal (top) and vertical (bottom) positions over time are illustrated by two graphs.
Figure 38: The right pages of $D_3$’s vertical (top) and horizontal (bottom) positions over time are illustrated by two graphs.
Figure 39: The left pages of $D_3$'s horizontal (top) and vertical (bottom) positions over time are illustrated by two graphs.