Classification of Text and Image Areas in Digitized Documents for Mobile Devices

Anne-Sophie Ettl[1,2], Axel Zeilner[2], Ralf Köster[2] and Arjan Kuijper[1,3]
[1] *Technische Universität Darmstadt, Darmstadt, Germany*
[2] *FINARX GmbH, Darmstadt, Germany*
[3] *Fraunhofer IGD, Darmstadt, Germany*

Keywords: Digitizing, Mobile Devices, Classification.

Abstract: Post processing and automatic interpretation of images plays an increasingly important role in the mobile area. Both for the efficient compression and for the automatic evaluation of text, it is useful to store text content as textual information rather than as graphics information. For this purpose pictures from magazines are recorded with the camera of a smartphone and classified according to text and image areas. In this work established desktop procedures are presented and analyzed in terms of their applications on mobile devices. Based on these methods, an approach for image segmentation and classification on mobile devices is developed, taking into account the limited resources of these mobile devices.

1 INTRODUCTION

Smartphones are becoming increasingly important and have become indispensable in professional and private life. The introduction of the iPhone in 2007 and the Android smartphone operating system in 2008 revolutionized the mobile market and, for example, now 43% of all mobile phones sold in Germany are smartphones. This trend of ubiquitous availability of computing power and network access continues. Particularly simple smartphones are now equipped with computational resources that have been part of a workgroup server less than 10 years ago. Smartphones have another feature compared to the computer. On most desktop PCs (flexible) cameras are still not widely available. Meanwhile, the mobile cameras have a zoom range with regard to the resolution and the quality of a flatbed scanner. Therefore, the processing of images plays an increasingly important role in the Mobile area (Wientapper et al., 2011; Engelke et al., 2012).

An important use case represents the digital storage of documents, magazines and documents. This can be done locally on the device or via an online storage, the cloud. In both cases, an efficient compression of the image material is advantageous. For the automatic evaluation of text it thus makes sense to store the text portion of an image not as a graphic, but as textual information. Therefore it is necessary to classify the recorded image material in text and image regions and to process them in different ways. For text fields, for example, the further processing can be done with an OCR engine.

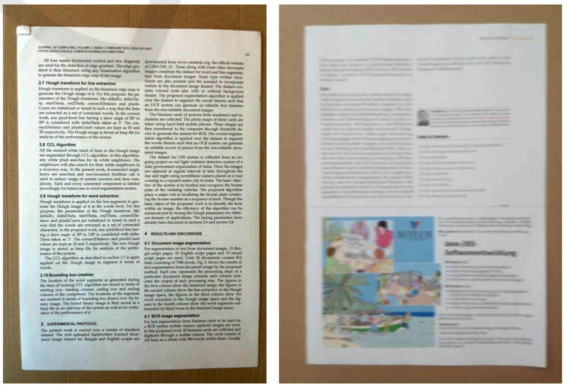


Figure 1: Typical example of a scan made using a smart phone (left) and a slightly Gaussian blurred image used in the processing pipeline (right).

For desktop computers and laptops a standard set of applications for the problem has been established (Liang et al., 2005). Because of the lower computing and storage applications, they are not directly transferable to mobile devices. Especially speed is a critical issue, since a smartphone user does not want to wait more than a few seconds for the result of an operation.

Ettl A., Zeilner A., Köster R. and Kuijper A..
Classification of Text and Image Areas in Digitized Documents for Mobile Devices.
DOI: 10.5220/0004273600880091
In *Proceedings of the International Conference on Computer Vision Theory and Applications* (VISAPP-2013), pages 88-91
ISBN: 978-989-8565-48-8

In this work, a method for image segmentation and classification is presented, which is suitable for use in mobile applications. Shots of magazines that were taken by a smartphone camera are considered as images. The method has to locate text or image regions in the image, extract them and provide them for further processing, for example for use with an OCR application. The procedure must satisfy the following claims for applicability of mobile devices:
i) High speed
ii) Low demands on resources
iii) Easy portability / Cross-platform
iv) Low complexity of the implementation to ensure the practical application.

The resulting prototype should be ported with little effort on various mobile Platforms (e.g. iOS, Android). The prototype is therefore implemented in Python using the OpenCV library. In the remainder, first existing methods are presented for image segmentation. Their performance and portability to mobile devices are discussed. Then, a suitable procedure for use on mobile devices, together with the theoretical basis is explained. The results of the prototypical implementation of the process are presented, analyzed and discussed and an outlook for future work is given.

2 RELATED WORK

In the literature, there are a variety of approaches for the segmentation of documents. They can be divided into three categories: bottom-up, top-down and hybrid methods (Lin et al., 2006). See (Ettl, 2012) for more details.
Yuan & Tan (Yuan and Tan., 2000) segment the source image into text and non-text areas and let the OCR engine carry out a refined analysis. For this purpose they require certain properties of text fields: It can be assumed that words in text fields have similar height, alignment, and spacing. Unfortunately, they do not provide runtime data. Single statement: The algorithm is ”relatively fast”.
Mollah et al. (Mollah et al., 2010) describe a method of text segmentation on Business cards, which is specially designed for use on mobile devices. First, the background is removed. Then, detection of text components is performed using Connected Components in isolated information areas. This procedure takes between 0.06 seconds (0.3 megapixels) and 0.6 seconds (3 megapixels) on a dual-core 1.73 GHz processor with 1 GB RAM.
Lin, Tapamo, and Ndovie (Lin et al., 2006) present a method that segments a document using a gray scale matrix for the detection of textures. Regions are classified in text, image and empty areas. The running time of this process on a Pentium 4 is about 3 seconds for an image with a resolution of 1449x2021 pixels.

3 IMAGE SEGMENTATION ON MOBILE DEVICES

In this work, some algorithms that are frequently used in the image segmentation are combined and tested. The aim is to develop a two-stage process:

In the first step, a pre-processing method is carried out and all the foreground objects are separated from the background (*object extraction*). Foreground objects can be text, images, or drawings. The algorithms are applied to the entire original image.

In the second step, the detected objects are classified into text and non-text objects (*object classification*). The algorithms are applied only to certain areas of the source image’s regions of interest (ROI). This has the advantage that individual objects may be examined according to need with various methods. In addition, the running time is reduced when small areas of the image are edited instead of the whole image.

From the available approaches a selection of appropriate algorithms (Burger and Burge, 2009) for rapid segmentation was selected. An important criterion based on which algorithms were selected, was the performance. The *histogram analysis* was deferred in favor of heuristic lines of text recognition, because the running time is worse and it has other weaknesses in the distinction between text and drawings. The *Canny edge detector* was rejected because it didn’t add value to the Connected Components Analysis. The edge detector produces a binary edge image, whereas the CCA supplies object contours that are stored in a list or a tree structure that can easily be further processed.

The original image is first smoothed by a Gaussian filter. This is to reduce the noise on the one hand, by filtering out small noise pixels. On the other hand object contours are highlighted. Depending on the degree of blurring, individual words blur into lines of text, lines of text into blocks, and smaller objects are grouped into larger areas. This allows text lines, blocks of text, photos, and graphics to delimit easily from the background of the picture. This effect is enhanced by the application of the Gaussian pyramid, which also increased the processing speed by reducing the image resolution. In the next step, the image is binarized by an adaptive thresholding method and thus a binary mask is created. Background pixels are set to 0 and foreground or object pixels to 255.

On the mask Connected Components Analysis

(CCA) analysis is performed to identify foreground objects and to draw bounding boxes around them. In the Connected Components Analysis, the contour extraction of (Suzuki and Abe, 1985) is used, as the OpenCV library (the language used in the implementation) provides a very fast implementation of the algorithm. On the mask an erosion is applied to again remove small noise pixels and to ensure that in the subsequent dilatation no objects are merged that do not belong together. By means of the dilation of small holes within objects are closed and objects lying very closely together, such as lines of text or objects within a graphic, are merged into a larger object (Figure 2). A closing operation does not bring the desired effect, since the use of different structural elements gives better results. On the mask, a second Connected Components Analysis is performed. The identified objects are then used to extract the ROIs from the original image.

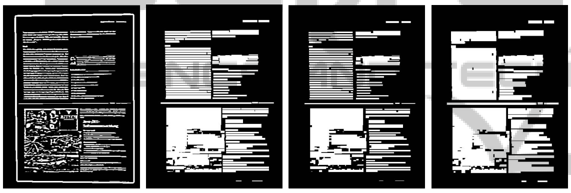


Figure 2: Segmentation Pipeline: Binarization of the slightly blurred image, CC-Analysis, erosion and dilation (left to right).

The next step is to decide whether the extracted ROIs are text or non-text areas. The following heuristics are adopted (Mollah et al., 2010):
i) Lines of text have a certain ratio of height to width;
ii) Lines of text have a certain distance from each other;
iii) Text fields have a certain ratio of background pixels to foreground pixels.

To extract text lines the image region is filtered with a Gaussian and then binarized. Next, using CCA bounding boxes are drawn around the objects. Using heuristics, it is decided whether objects are text lines or not. When the objects within a ROI are exclusively or predominantly clearly marked as lines of text, then the ROI is classified as a text area. This is not the case, then the ROI initially is classified as a non-text and with refined heuristics or with other methods, for example a texture analysis, it can be tested if it contains any text components.

4 RESULTS

The times were measured on a MacBook Pro 2.4 GHz Intel Core 2 Duo processor. Shown here are the pure running times of the algorithm, without loading the image, or displaying it on the screen. To normalize the runtime, the algorithm was run 50 times and the mean value was obtained.

The segmentation using the Hough algorithm was discarded after a few test runs, as it has the following disadvantages compared to a Connected Components Analysis:
1) The algorithm is much more computationally intensive. The pure running time for an image with 612x816 pixels is 0.21 seconds (threshold of the accumulator array = 150) and 0.49 seconds (threshold = 100). Compared to the complete CCA including histogram analysis (0.21 seconds), the algorithm is 1-2 times as computationally intensive.
2) In order to obtain approximately equivalent results a significantly more complex implementation is required. The output image must first be prepared with the Canny edge detector with appropriate parameters, and thereafter an appropriate parameterization of the Hough algorithm must be determined.

The histogram analysis yields qualitatively similar results as heuristic analysis, but was deferred in this work however:
1) It is computationally more intensive than the use of heuristics to recognize the text: For an image with 612x816 pixels, the method with histogram analysis requires 0.21 seconds, with heuristics 0.042 seconds.
2) It is difficult to distinguish between text fields and drawings.

The method has been tested on various sample images with different resolutions, taken with an iPhone 3 and an iPhone 4. Representative results on 5 images are shown in Figures 3 and 4.

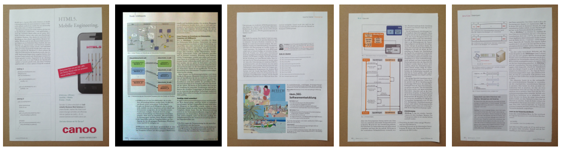


Figure 3: Original images.

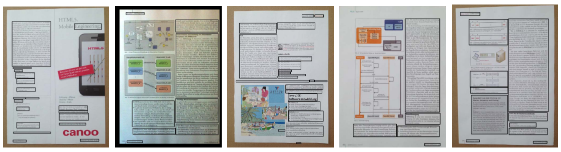


Figure 4: Results of object classification.

To evaluate the classification of the image objects following cases were evaluated (Mollah et al., 2010):
1) True positive (TP): Text areas are classified as text
2) True negative (TN): Non-text is classified as non-text

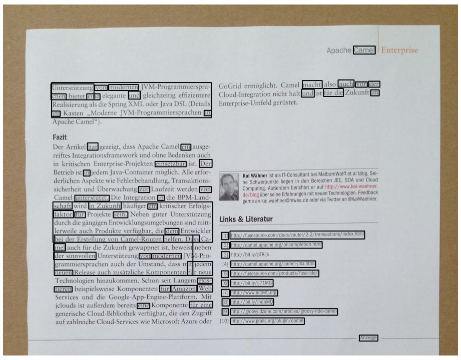
Table 1: Evaluation of the shown test images.

Image	Size	Recall	Prec.	Acc.
1	734x979	0.75	0.88	0.70
2	623x921	0.73	1.00	0.77
3	734x979	0.90	0.82	0.76
4	612x816	0.56	1.00	0.64
5	612x816	0.89	0.73	0.69

3) False negatives (FN): Text areas are classified as non-text
4) False Positive (FP): Non-text is classified as text

From the number of occurrences of the different cases one can calculate Recall, Precision and Accuracy. *Recall* indicates how many text components are correctly identified from any existing text components of an image. *Precision* indicates how many of the identified text components also are actually text components. *Accuracy* is the proportion of all the objects that have been classified correctly. The values of the three dimensions are between 0 and 1 In an ideal classification they take the value 1.
Recall = TP / (TP + FN)
Precision = TP / (TP + FP)
Accuracy = (TP + TN) / (TP + FP + TN + FN)

The average recall is thus 76.6% and average precision 88.6%, and average accuracy at 71.2%.



Apache Camel | Enterprise

Unterstützung von modernen JVM-Programmiersprachen bietet eine elegante und gleichzeitig effizientere Realisierung als die Spring XML oder Java DSL (Details im Kasten „Moderne JVM-Programmiersprachen in Apache Camel“).

Fazit

Der Artikel hat gezeigt, dass Apache Camel ein ausgereiftes Integrationsframework und ohne Bedenken auch in kritischen Enterprise-Projekten einsetzbar ist. Der Betrieb ist in jedem Java-Container möglich. Alle erforderlichen Aspekte wie Fehlerbehandlung, Transaktionssicherheit und Überwachung zur Laufzeit werden von Camel unterstützt. Die Integration in die BPM-Landschaft wird in Zukunft häufiger ein kritischer Erfolgsfaktor für Projekte sein. Neben guter Unterstützung durch die gängigen Entwicklungsumgebungen sind mittlerweile auch Produkte verfügbar, die dem Entwickler bei der Erstellung von Camel-Routen helfen. Dass Camel auch für die Zukunft gewappnet ist, beweist neben der sinnvollen Unterstützung von modernen JVM-Programmiersprachen auch der Umstand, dass mit jedem neuen Release auch zusätzliche Komponenten für neue Technologien hinzukommen. Schon seit Langem existieren beispielsweise Komponenten für Amazon Web Services und die Google-App-Engine-Plattform. Mit jclouds ist außerdem bereits eine Komponente für eine generische Cloud-Bibliothek verfügbar, die den Zugriff auf zahlreiche Cloud-Services wie Microsoft Azure oder GoGrid ermöglicht. Camel macht also auch vor der Cloud-Integration nicht halt und ist für die Zukunft im Enterprise-Umfeld gerüstet.

Kai Wähner ist als IT-Consultant bei MaibornWolff et al tätig. Seine Schwerpunkte liegen in den Bereichen JEE, SOA und Cloud Computing. Außerdem berichtet er auf http://www.kai-waehner.de/blog über seine Erfahrungen mit neuen Technologien. Feedback gerne an kai.waehner@mwea.de oder via Twitter an @KaiWaehner.

Links & Literatur

[1] http://fusesource.com/docs/router/2.2/transactions/index.html
[2] http://camel.apache.org/oncompletion.html
[3] http://bit.ly/pTKijk
[4] http://camel.apache.org/camel-jmx.html
[5] http://fusesource.com/products/fuse-ide/
[6] http://bit.ly/ij71MG
[7] http://www.activiti.org
[8] http://bit.ly/lhdbMQ
[9] http://groovy.dzone.com/articles/groovy-ride-camel
[10] http://www.grails.org/plugin/camel

Anzeige

Figure 5: Figure 3, 1714x2285 pixels: result of the classification at too high resolution.

As part of the implementation it turns out that the algorithm for an A4 page of a magazine photographed only with a resolution from 0.3 to 1 megapixel (MP) returns most useful results. When the resolution is lower, then in the Connected Components Analysis and the dilatation too many objects fuse to a region, which then can not be clearly classified. When the resolution is higher, possibly related objects are not merged and only individual words are recognized (Figure 5). In both cases, the algorithm does not lead to meaningful results. In the implementation, therefore, the original image is scaled in the pre-processing in the above-mentioned range, the algorithm is applied to the scaled image, and then extrapolated to the original image, where the classified objects are marked or extracted.

Table 2: Running times in milliseconds for image 3 for different sizes (megapixels), including preprocessing, extraction and classification.

Size	MP	tot.	pre.	extr.	class.
2448x3264	8.0	747	29	75	643
1714x2285	4.0	378	15	40	323
1224x1632	2.0	161	6.6	18	136
857x1142	1.0	75	4.2	9	62
612x816	0.5	42	2.3	4.7	35

From the measurements it can be seen that the running time of the process is nearly linear. Because the algorithm provides good classification results for scaled images in the range between 0.3 and 1 megapixel, this area is also of particular interest for running time analysis. Here the duration of the process is 20 to 70 ms.

REFERENCES

Burger, W. and Burge, M. (2009). *Principles of Digital Image Processing*. Springer, London.

Engelke, T., Becker, M., Wuest, H., Keil, J., and Kuijper, A. (2012). MobileAR browser - a generic architecture for rapid AR-multi-level development. *Expert Systems with Applications*, x(x):xx–xx. in press, DOI=10.1016/j.eswa.2012.11.003.

Ettl, A.-S. (2012). Klassifikation von bildbereichen in digitalisierten dokumenten zur andendung auf mobilen geraeten. Technical report, TU Darmstadt.

Liang, J., Doermann, D., and Li, H. (2005). Camera-based analysis of text and documents: a survey. *International Journal on Document Analysis and Recognition*, 7:84–104.

Lin, M.-W., Tapamo, J.-R., and Ndovie, B. (2006). A texture-based method for document segmentation and classification. *South African Computer Journal*, 36:49–56.

Mollah, A., Basu, S., and Nasipuri, M. (2010). Text/graphics separation and skew correction of text regions of business card images for mobile devices. *Journal of Computing*, 2(2):96–102.

Suzuki, S. and Abe, K. (1985). Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32–46.

Wientapper, F., Wuest, H., and Kuijper, A. (2011). Composing the feature map retrieval process for robust and ready-to-use monocular tracking. *Computers & Graphics*, 35(4):778–788.

Yuan, Q. and Tan., C. (2000). Page segmentation and text extraction from gray scale image in microfilm format. In *Proceedings SPIE vol. 4307*, pages 323–332.