Assignment 1 - Page Detection

Document Analysis

Timon Höbert(01427936)
Manuel Mayerhofer (01328948)
Stefan Stappen(01329020)

June 3, 2018

1 Definition of Task

Optical character recognition is the task of transforming digitalized documents again into a machine readable format. Precisely, the content of images of documents is analysed, characters are recognized and the text is rebuilt from the characters. The ICDAR2015Competition on Smartphone Document Capture and OCR (SmartDoc) [1] is the first competition for document page detection and Smartphone OCR. The second assignment of this competition was OCR.
The input is a scanned in document, a subset of the "I AM printed" dataset. The output represents the text inside the document with x and y coordinates for each character.

2 Dataset

The dataset consists of a subset of the I AM PRINTED dataset. The images are all 2 to 7 lines of text with negligible skew. For this reason we omitted the skew detection. Every text has a enclosing lines at the top and the bottom and sometimes there is handwritten text at the bottom. All the text samples are written with the same style, with serifs, none-bold and the same font-art. There are only minor changes in resolution. The whole dataset consists of 100 images and 100 xml files containing the ground truth. In each xml file x and y coordinates for each line, word and character can be found. Though, these seem not be correct as width and height exceed the resolution of the images!

3 Implementation

As recommended, we implemented for the line, word and character segmentation projection profiles. Our OCR is based on a trained neural network. For this reason, we created our own training data and annotated them semi-automatically. TODO For the neural networks we used the python library TODO

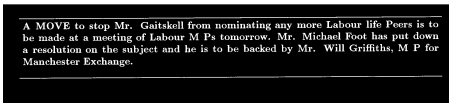
3.1 Line, Word and Character Segmentation

In a preprocessing step, we eliminate noise with a Gaussian blur. Afterwards, we threshold the image with an Otus-Threshold and invert the image. In the next step we dilate the image for better line segmentation.

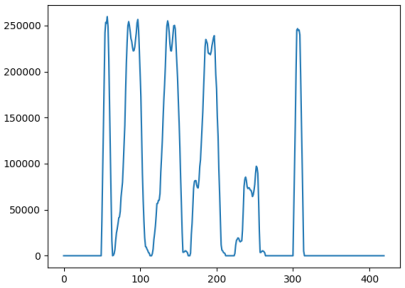
After the preprocessing we are ready to compute the horizontal projection profile. In Figure 1 a preprocessed image and its horizontal projection profile can be observed. The segmentation process is now straight forward. Zeros represent space and all connected non-zeros are one line. After detecting all lines the medium line height is computed and lines smaller then 45 % of the medium line height are rejected. These processed lines are now further processed to detect words inside a line. To better detect words and not detect single characters we apply a further dilation on the extracted line. Then we apply a vertical projection and do the same segmentation as with lines. An example for such an post-processed line and its vertical projection profile can viewed in Figure 2. In the last step the so detected words are then segmented into characters. As characters are very near to each other, sometimes there is only one pixel space between them, we use not the preprocessed image for them. Instead we apply an Otsu threshold and invert the original image, without any dilation and blurring. An example for this can viewed in Figure 5. The line and word segmentation works flawlessly. The character segmentation has some failures. We first tried to use the preprocessed image we used for the other segmentations too. Though, this leads to merged characters in many cases, see Figure **??**. We tried the same without the dilation but it yields the same results because the blur already merges the characters. Another approach we tried to prevent these merged characters as to use skeletonization with the code from `http://opencvpython.blogspot.com/2012/05/skeletonization-using-opencv-python.html`. This method works quite good to resolve the merged characters. Though, it has the disadvantage of split characters because some lines of characters are very thin, see Figure **??**. For this reason, we rejected this approach as it introduces more errors than it fixes. Our final method uses no blur and no dilation and works on the original image, thresholded and inverted. Then we cut the top and bottom space to remove overlappings from other characters, like an 'f' and apply the vertical projection profile. With this approach we achieve the correct results in about 90 % of the cases. Some errors remain, especially if characters overlap or are connected.

4 Performance

TODO One big drawback of the algorithm is the execution time that depends on the number of founded line segments. We calculate for every combination of two horizontal and two vertical lines a bounding box. This is very expense even with a small number of lines due to the permutations. Another problem is that the LSD algorithm takes only an input image path as input. So we have to create an image for every frame and write it out.



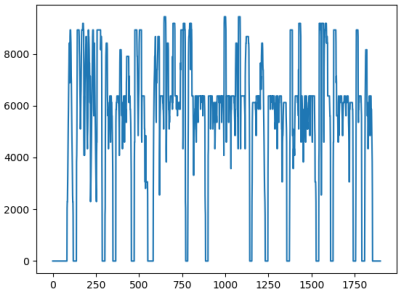
(a) preprocessed image



(b) horizontal projection profile

Figure 1: Example of preprocessed image and line projection profile.

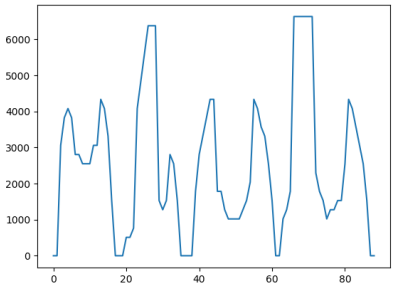
(a) preprocessed image



(b) horizontal projection profile

Figure 2: Example of further dilated extracted line and word projection profile.

(a) preprocessed image



(b) horizontal projection profile

Figure 3: Example images of extracted word without blurring and dilating and its vertical projection profile.

5 Evaluation

TODO To measure the performance the Jacard index measure [2] is used. The Jacard index is the intersection of the detected bounding box with the ground truth document bounding box divided through the union of the bounding boxes (see equation 1).

$$JI(frame) = \frac{area(G \cap S))}{area(G \cup S))} \tag{1}$$

The score for one video is the average of the frame scores, and the overall accuracy of the implemented method is the average of all videos with different backgrounds.

6 Results

The computed Jacard indices for the given dataset with our method can be viewed in table 1.

We tried to optimize our approach for the usual case of page detection with a smartphone. For this reason we optimized our technique for the first three datasets. The fourth background is not really in a natural environment because usually the flash light is turned of during video page detection with a smartphone, though this case is arguable. The fifth background does not make any sense as the later for OCR needed information is anyway overlayed by other stuff. Although, there are more than one document and it can be argued to decide which document to detect.

Figure 4: Example images of merged characters.

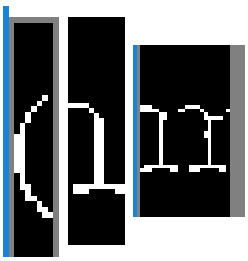


Figure 5: Example images of split characters.

Table 1: Jacard Index Results

	Background				
Document Type	01	02	03	04	05
Datasheet	0.961804	0.933321	0.956437	0.338121	0.419252
	0.969550	0.949776	0.953702	0.670783	0.204255
	0.958112	0.926158	0.954332	0.460790	0.407288
	0.963166	0.942556	0.957530	0.406976	0.279959
	0.956819	0.931061	0.964136	0.480986	0.373999
Letter	0.966585	0.931361	0.964660	0.548722	0.488031
	0.962465	0.950924	0.939243	0.637083	0.454185
	0.962962	0.943569	0.901809	0.444535	0.463177
	0.955535	0.945724	0.939415	0.690977	0.405386
	0.959804	0.939154	0.941210	0.414960	0.624913
Magazine	0.961872	0.906609	0.945897	0.169548	0.042877
	0.959672	0.937086	0.919084	0.148990	0.110548
	0.922668	0.900296	0.941420	0.693086	0.000000
	0.886081	0.863409	0.915656	0.389279	0.552460
	0.922554	0.813186	0.964260	0.458704	0.175053
Paper	0.956659	0.967295	0.952133	0.667463	0.410692
	0.954750	0.959377	0.904080	0.610046	0.015565
	0.943191	0.955013	0.950578	0.557951	0.408258
	0.923238	0.947480	0.971184	0.643748	0.447700
	0.954222	0.934069	0.962020	0.614676	0.453615
Patent	0.944150	0.950614	0.915164	0.629440	0.464737
	0.934623	0.942731	0.906549	0.634424	0.380177
	0.939969	0.935944	0.956033	0.538994	0.502359
	0.944970	0.918508	0.897266	0.226930	0.358804
	0.914572	0.837318	0.954834	0.548980	0.311188
Tax	0.949686	0.902505	0.944010	0.342882	0.490883
	0.940098	0.937493	0.956152	0.575945	0.016099
	0.867588	0.938492	0.933569	0.466925	0.062252
	0.903321	0.880571	0.936461	0.419935	0.047539
	0.951203	0.932429	0.930931	0.249198	0.500233
Overall/Background	0.9431	0.9251	0.9410	0.4894	0.3290
Overall	0.7255				

Anyway, we tried to improve our solution even for these unrealistic cases. First we had to tackle that because of missing contrast and overlays not the right lines were detected by the LSD. To overcome this proble, we convert the image in a gray value image. Then we use the canny edge detector with a sigma of 3 to create a binary image of the document. We chose the canny detector because it found most of the edges in the document frames, unlike Sobel or Prewitt. However, there were many distortions on the edges, so we increased the sigma from $\sqrt{2}$ to 4 to smooth the image a bit. The result of the preprocessing steps looks like Figure **??**.

With this preprocessing step the LSD again finds reasonable lines for the datasets with background four and five. Though, the results only improved to about 60 %. For the realistic backgrounds, one, two and three this preprocessing step has a severe impact. Due to the loss of information by converting the image into an edge image, the important lines get lost for the realistic data sets. The results are reduced from over 90 % to about 60 %. For this reason we stayed without the preprocessing step and therefore created a method which works quite well for realistic scenarios (over 90 %) and fails at unrealistic scenarios, a justifiable trade-off.

It is still to be noted, that for none of the described methods using LSD a paper could be found. Maybe they changed the method name or someone else in their research group published the paper with another name because we searched for the method names and the authors and could not find the papers. If they really did not publish there methods it is arguable that the description is either not complete and some other criteria for bounding box selection is used or that they optimized the parameters for the given datasets to achieve these results.

References

[1] Jean-Christophe Burie, Joseph Chazalon, Mickaël Coustaty, Sébastien Eskenazi, Muhammad Muzzamil Luqman, Maroua Mehri, Nibal Nayef, Jean-Marc Ogier, Sophea Prum, and Marçal Rusiñol. Icdar2015 competition on smartphone document capture and ocr (smartdoc). In *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, pages 1161–1165. IEEE, 2015.

[2] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[3] primetang. Lsd-opencv-matlab. https://github.com/primetang/LSD-OpenCV-MATLAB, commit: 0b035a8c5b105091604d71f1570ac4d4526aec8d, 2014.

[4] Rafael Grompone Von Gioi, Jeremie Jakubowicz, Jean-Michel Morel, and Gregory Randall. Lsd: A fast line segment detector with a false detection control. *IEEE transactions on pattern analysis and machine intelligence*, 32(4):722–732, 2010.