

Anveshak - A Groundtruth Generation Tool for Foreground Regions of Document Images

Soumyadeep Dey, Jayanta Mukherjee, and Shamik Sural

Department of Computer Science and Engineering, Indian Institute of Technology
Kharagpur, Kharagpur 721302

Abstract. We propose a graphical user interface based groundtruth generation tool in this paper. Here, annotation of an input document image is done based on the foreground pixels. Foreground pixels are grouped together with user interaction to form labeling units. These units are then labeled by the user with the user defined labels. The output produced by the tool is an image with an *XML* file containing its metadata information. This annotated data can be further used in different applications of document image analysis.

1 Introduction

Document digitization has attracted attention for several years. Conversion of a document image into electronic format requires several types of document image analysis. Typical document image analysis includes different types of segmentation, optical character recognition (*OCR*), etc. Numerous algorithms have been proposed to achieve these objectives. The performance of these algorithms can be measured with the help of groundtruth. The data with groundtruth is of immense importance in document image analysis. It is required for training, machine learning based algorithms, and it is also used for evaluation of various algorithms. The generation of groundtruth is a manual and time consuming process. Hence, the groundtruth generation tool should be user friendly, reliable, effective, and capable of generating data in a convenient manner.

Several systems for groundtruth generation have been reported in the literature for producing benchmark datasets to evaluate competitive algorithms. Pink Panther [21] is one such groundtruth generator, and is mainly used for evaluation of layout analysis. PerfectDoc [19] is a groundtruth generation system for document images, based on layout structures. Various layout based groundtruth generation tools are present in the literature [9], [15], [20]. These groundtruth generators [7], [9], [19], only support rectangular regions for annotation. Hence, they fail to generate groundtruth for documents with complex layout.

A recent groundtruth generator *GED*I [5], supports annotation by generating a polygonal region. However, it is observed that the tool is quite inefficient for images of larger size (600*dpi*). PixLabeler [13] is an example of pixel level groundtruth generator. Similar tools are also reported in [11], [14], [18]. Pixel level annotation gives more general measure for annotation, but it involves more time for completing the annotation task.

In this paper, we propose a tool to annotate a document image at pixel level. The main objective of the tool is to efficiently annotate data using less amount of time. Towards this, we have provided a semi-automatic interactive platform to annotate document images efficiently. Since our main goal is to annotate foreground pixels, we segment foreground pixels from its background with user assistance. Next, we group foreground pixels such that neighboring pixels of similar types get connected. Finally, annotation of each such group of pixels is performed with a predefined set of labels.

The system is called Anveshak and its functionality is described in Section 2. Implementation details are discussed in Section 3. Section 4 provides the details of groundtruth generation with Anveshak. Finally, we conclude in Section 5.

2 Functionality

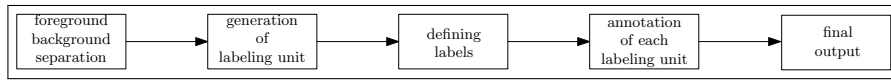


Fig. 1. Work flow of Anveshak system

The work-flow of the Anveshak system is shown in Figure 1. Some semi-automated modules are implemented to speed up the annotation process.

2.1 Foreground Background Separation

We are mainly concerned with the annotation of foreground pixels of a document image. A module is integrated with Anveshak to efficiently segment foreground pixels from its background. This task can be performed with three types of thresholding techniques, first, *GUI* based thresholding, second, a *GUI* based adaptive thresholding technique [1], and third, the *Otsu's* thresholding technique [12]. Here, a user can segment foreground from its background efficiently, using either of these three thresholding techniques. An example of foreground background separation module using *GUI* based thresholding is shown in Figure 2.

2.2 Generation of Labeling Units

Anveshak has a unique technique to predefine labeling units. Labeling units are generated using *GUI* based morphological operations. Morphological operations included in Anveshak are, *erosion*, *dilation*, *closing*, *opening*, *gap-filling*, and *smoothing*.

A labeling unit is a collection of foreground pixels, grouped together using a suitable morphological operator. Pixels are grouped together by choosing either

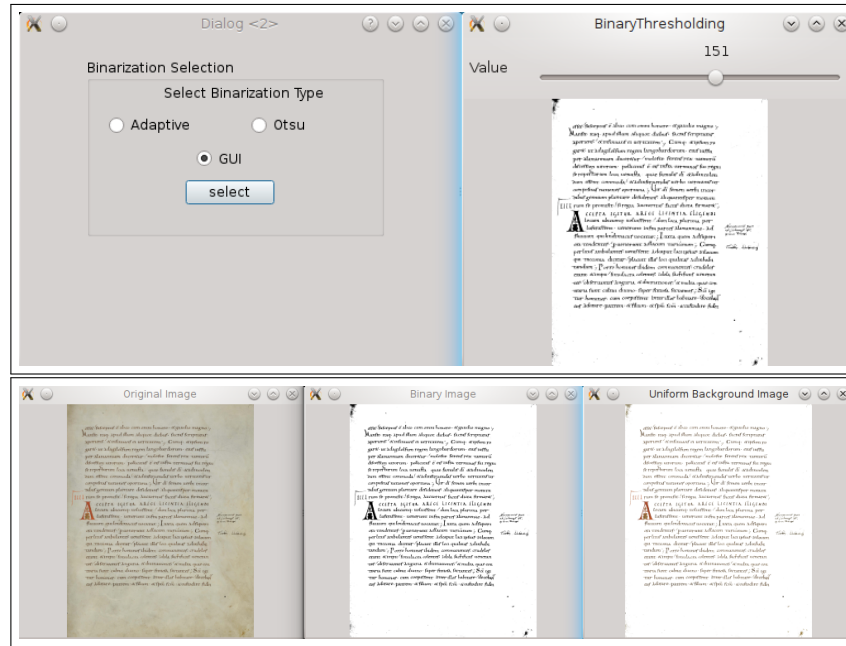


Fig. 2. Foreground background separation module.

of these morphological operations - *erosion*, *dilation*, *closing*, and *opening* [8]. The user can select an ideal element size and element type, in order to group pixels. A user can also accumulate pixels to form a group by a smoothing operation [17], where choosing of run length parameter is an interactive process. Foreground pixels can also be grouped together using gap filling operation [4], where selection of the parameter, gap size in horizontal and vertical directions, is a user driven process. An instance of Anveshak for generating labeling units is shown in Figure 3.

After grouping the pixels, contours of each group is obtained using the method described in [16]. Each contour is then approximated to a polygon by applying Douglas-Peucker algorithm [6]. The polygons thus computed are the basic units for annotation in Anveshak. An example of a collection of labeling units is shown in Figure 4, where each labeling unit is represented using a unique colors.

2.3 Defining Labels

There are some predefined labels in Anveshak. The tool provides an option to add and delete labels, as shown in Figure 5. After defining all the labels, a user can annotate the labeling units of the input document with the defined labels. A unique index number and a color is assigned to each label, which are used in the later stages of annotation.

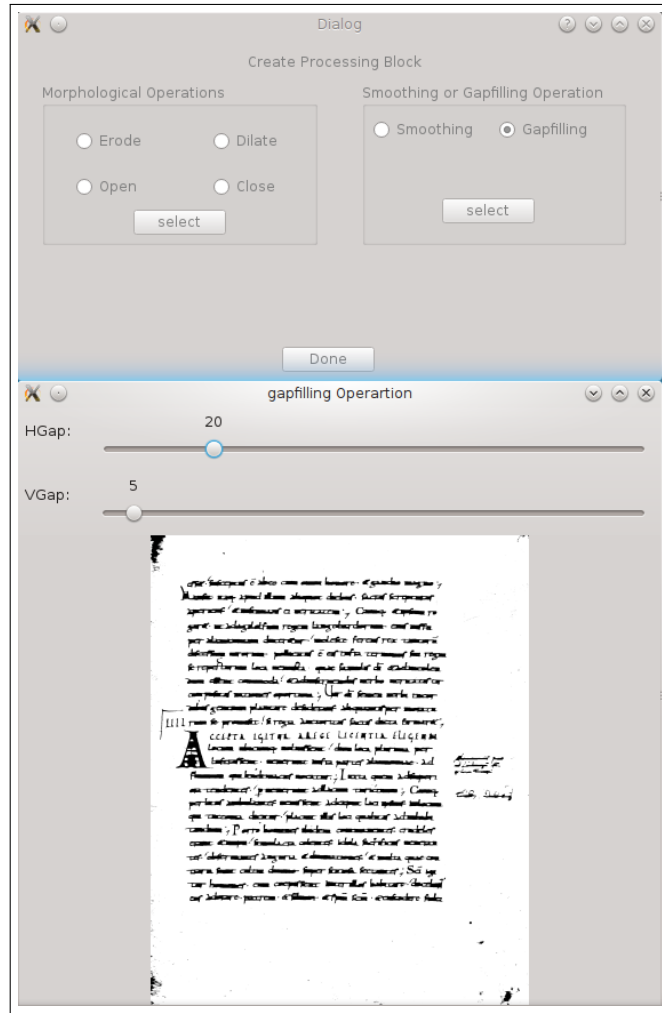


Fig. 3. Module of Anveshak for generating labeling unit with gap filling operation

2.4 Annotation of Labeling Units

Overall annotation process can be summarized using a flow chart given in Figure 6. Annotation of labeling units is performed in two ways as shown in Figure 7. A user can label unlabeled units one by one with the predefined labels. In this case, an unlabeled unit is displayed in a window and the user is prompted for a label for the displayed unit. This process continues until each of the units is labeled, or the user chooses to label the units by selecting a region of interest (ROI).

Another method of labeling units is to select a region of interest. In this module, a user can select an ROI, which can be annotated with the defined

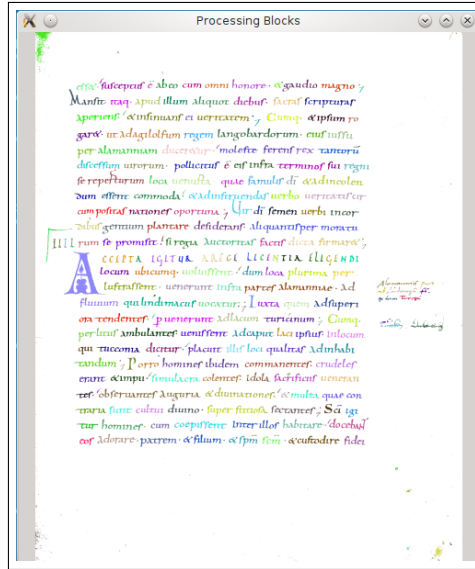


Fig. 4. Labeling units generated by Anveshak with user intervention

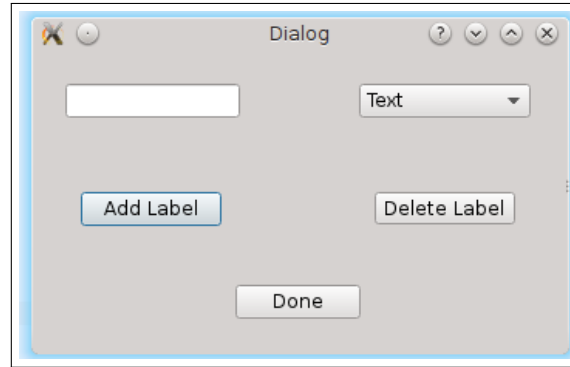


Fig. 5. An instance of Anveshak to set labels

labels. At first, all units are determined which are completely present within the selected *ROI*. After selection of an *ROI*, units present within the *ROI* can be labeled using three different modes (Figure 8). A user can annotate all units within the *ROI* with one label, and update all units with the selected label. Another way of annotation is by labeling all units belonging to the selected *ROI* with a particular type. Lastly, a user can annotate each unit belonging to the selected *ROI* individually with a label. Pixels belonging to a particular labeling unit are updated with the unique index corresponding to the label of *ROI*, and color of those pixels is updated with the color of that label. Belongingness of a pixel to a particular labeling unit is computed through point-polygon test. At

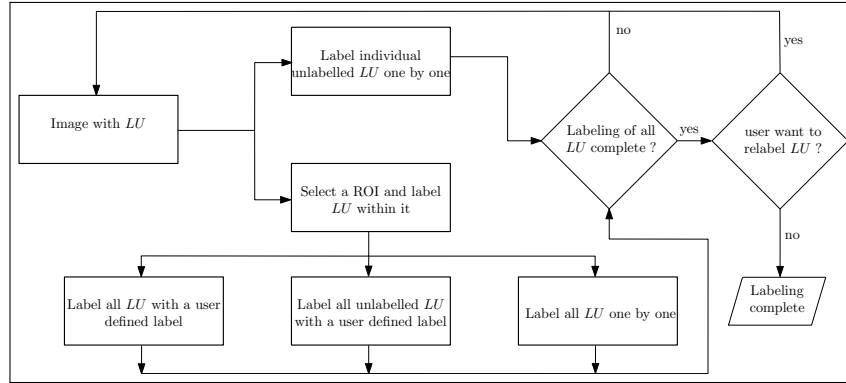


Fig. 6. Work flow of the annotation unit of Anveshak

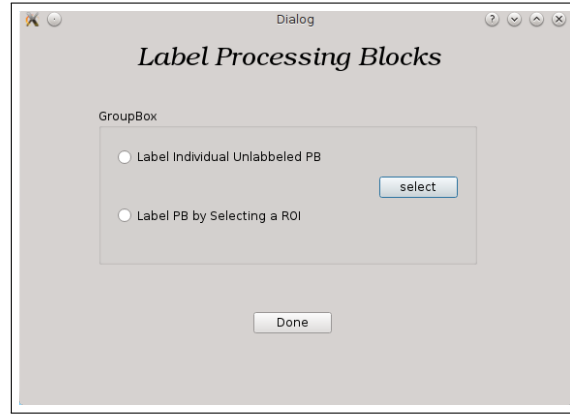


Fig. 7. Instance of Anveshak for choosing annotation mode

each stage of the annotation process, the updated color image is displayed, where labeled pixels are displayed with color of the corresponding label, and unlabeled pixels are displayed with their original color value.

The process of annotation continues until all labeling units are marked. After completion of annotation, the user is asked, whether he/she wants to update any label, or finalize the labels. After finalizing the labels, output labeled image and its corresponding *XML* file are generated. An example of different stages of labeling is shown in Figure 9.

3 Implementation Details

Anveshak is implemented in *C++*, using cross-platform application framework *Qt* for graphical user interface and with customized modules developed using *OpenCV* [1]. Annotation of an image is achieved through the user interface

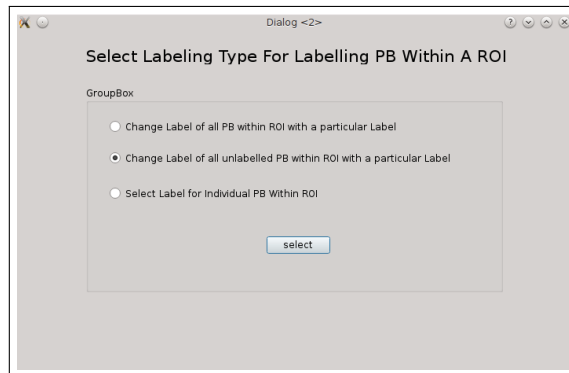


Fig. 8. Options prompted for choosing labeling mode for a *ROI*.

and after completion, a single image in *.png* format is generated. Each pixel of the output image is represented with an index corresponding to a particular annotation.

The metadata of the concerned image is stored in an *XML* file, which also includes the information of the source image along with the annotated image. In the *XML* file, an index corresponds to the unique pixel value for a particular label in the annotated image. Examples of two different annotated images and their corresponding *XML* files are respectively shown in Figures 9 (c) and (d) and Figures 10 (a) and (b). Anveshak is tested to annotate 344 images from the dataset reported in [10]. It has been observed by the annotator that, the labeling can be performed in a much easier and faster way than it could be performed with PixLabeler [13] or *GEDI* [5].

In our present implementation of Anveshak, only one annotation per block is supported. In many scenarios, it is desirable to have multiple annotations per block, mainly in case of overlapping regions. In future, we plan to support more than one annotation per block.

4 Generation of Groundtruth using Anveshak

Anveshak is used to generate groundtruth for the dataset reported in [10]. The images in the dataset consist of various regions like logo, headers, text, signature, headline, bold text, etc. However, annotation of stamp regions is only available with the original dataset. The dataset consists of 425 scanned images in 600, 300, and 200 *dpi* resolutions. Out of these 425 images, 344 images contain non overlapping regions. Anveshak is used to annotate these 344 images of 300 *dpi* resolution, and the groundtruth data has been made available online¹. These 344 images are annotated using Anveshak with the help of 6 users. There are on an average 5 labels, and 148 segments per image in the given dataset. Users

¹ http://www.facweb.iitkgp.ernet.in/~jay/anveshak_gt/anveshak_gt.html

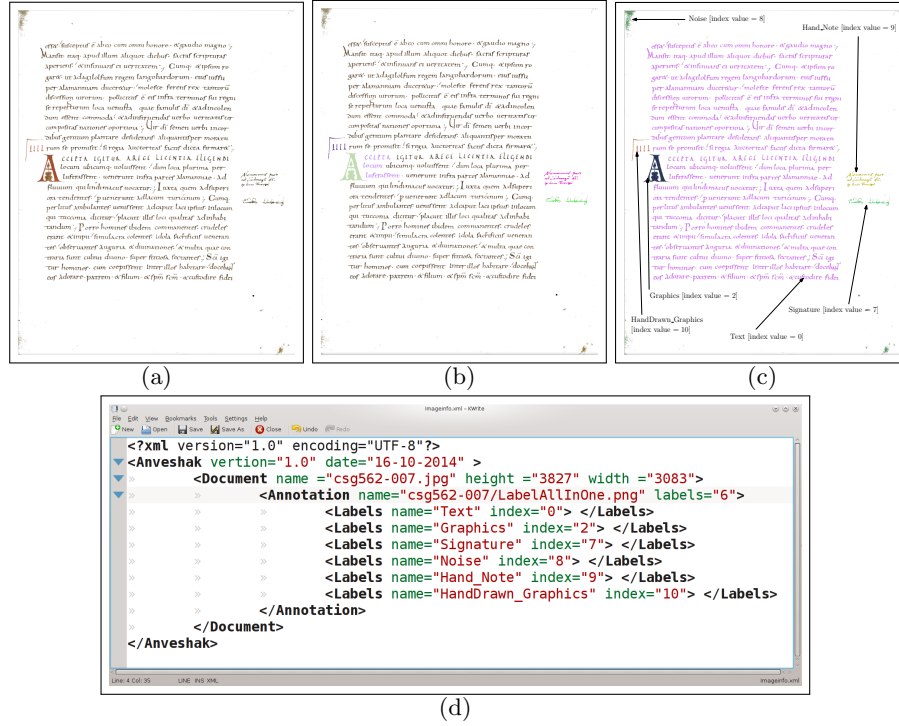


Fig. 9. (a) Unlabeled foreground pixels; (b) Half annotated foreground pixels; (c) Full annotated foreground pixels with a unique color per label; (d) *XML* output after annotation of Figure 9(a)

involved in annotation are initially trained to annotate data with one random image. Average time taken by a user to annotate an image with Anveshak is about 3 – 4 minutes. The annotated dataset has been used in the works reported in [2] and [3].

5 Conclusion

The primary target of Anveshak is to annotate an input document image in an efficient manner. Our tool produces an *XML* file containing the metadata information, along with an annotated image. We have developed a user friendly groundtruth generation tool, with some semi-automatic modules which make the annotation process faster. We hope that Anveshak will serve the document analysis community in an effective manner by simplifying groundtruth generation procedure.

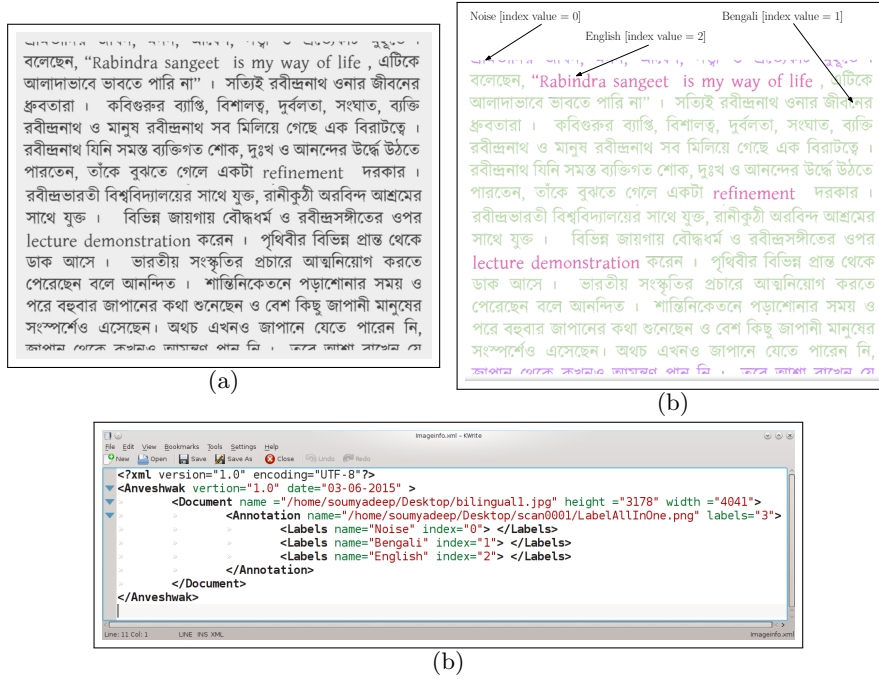


Fig. 10. (a) An input bilingual image (b) Annotated image for Figure 10 (a) with a unique color per label; (c) XML output after annotation of Figure 10 (a)

Acknowledgments

This work is partly funded by TCS research scholar program and partly by Ministry of Communications & Information Technology, Government of India; MCIT 11(19)/ 2010-HCC (TDIL) dt. 28-12-2010.

References

1. G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.
2. S. Dey, J. Mukherjee, and S. Sural. Stamp and logo detection from document images by finding outliers. In *2015 Fifth National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG)*, pages 1–4, Dec 2015.
3. S. Dey, J. Mukherjee, and S. Sural. Consensus-based clustering for document image segmentation. *International Journal on Document Analysis and Recognition (IJDAR)*, pages 1–18, 2016.
4. S. Dey, J. Mukhopadhyay, S. Sural, and P. Bhowmick. Margin noise removal from printed document images. *Workshop on Document Analysis and Recognition*, pages 86–93, 2012.
5. D. Doermann, E. Zotkina, and H. Li. GEDI - A Groundtruthing Environment for Document Images. <http://lampsrv02.umiacs.umd.edu/projdb/project.php?id=53>, 2010.

6. D. H. Douglas and T. M. Peucker. Algorithm for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122, 1973.
7. G. Ford and G. R. Thoma. Ground truth data for document image analysis. pages 9–11, 2003.
8. R. C. Gonzalez and R. E. Woods. *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2009.
9. L. C. Ha and T. Kanungo. The architecture of trueviz: a groundtruth/metadata editing and {VIsualiZing} toolkit. *Pattern Recognition*, 36(3):811 – 825, 2003.
10. B. Micenкова and J. V. Beusekom. Stamp detection in color document images. *11th International Conference on Document Analysis and Recognition*, pages 1125–1129, 2011.
11. M. Moll, H. Baird, and C. An. Truthing for pixel-accurate segmentation. In *Document Analysis Systems, 2008. DAS '08. The Eighth IAPR International Workshop on*, pages 379–385, Sept 2008.
12. N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979.
13. E. Saund, J. Lin, and P. Sarkar. Pixlabeler: User interface for pixel-level labeling of elements in document images. *10th International Conference on Document Analysis and Recognition*, pages 646–650, 2009.
14. F. Shafait, D. Keysers, and T. Breuel. Pixel-accurate representation and evaluation of page segmentation in document images. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 1, pages 872–875, 2006.
15. T. Strecker, J. van Beusekom, S. Albayrak, and T. Breuel. Automated ground truth data generation for newspaper document images. *10th International Conference on Document Analysis and Recognition*, pages 1275–1279, July 2009.
16. S. Suzuki and K. Abe. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32–46, 1985.
17. F. M. Wahl, K. Y. Wong, and R. G. Casey. Block segmentation and text extraction in mixed text/image documents. *Computer Graphics and Image Processing*, 20(4):375 – 390, 1982.
18. L. Wenying and D. Dori. A protocol for performance evaluation of line detection algorithms. *Machine Vision and Applications*, 9(5-6):240–250, 1997.
19. S. Yacoub, V. Saxena, and S. Sami. Perfectdoc: a ground truthing environment for complex documents. *8th International Conference on Document Analysis and Recognition*, pages 452–456, 2005.
20. L. Yang, W. Huang, and C. Tan. Semi-automatic ground truth generation for chart image recognition. In *Document Analysis Systems VII*, volume 3872 of *Lecture Notes in Computer Science*, pages 324–335. Springer Berlin Heidelberg, 2006.
21. B. Yanikoglu and L. Vincent. Pink panther: A complete environment for ground-truthing and benchmarking document page segmentation. *Pattern Recognition*, 31(9):1191–1204, 1998.