Picture Detection in Document Page Images

Patrick Chiu, Francine Chen, Laurent Denoue
FX Palo Alto Laboratory
3400 Hillview Ave., Bldg. 4, Palo Alto, CA 94304, USA
{chiu, chen, denoue}@fxpal.com

ABSTRACT
We present a method for picture detection in document page images, which can come from scanned or camera images, or rendered from electronic file formats. Our method uses OCR to separate out the text and applies the Normalized Cuts algorithm to cluster the non-text pixels into picture regions. A refinement step uses the captions found in the OCR text to deduce how many pictures are in a picture region, thereby correcting for under- and over-segmentation. A performance evaluation scheme is applied which takes into account the detection quality and fragmentation quality. We benchmark our method against the ABBYY application on page images from conference papers.

Categories and Subject Descriptors
H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing.

General Terms
Algorithms, Performance.

Keywords
Picture detection, page image, entity extraction, OCR, document image analysis.

1. INTRODUCTION
Picture detection and other object recognition technologies are useful for supporting indexing and browsing of document pages. A universal way to handle various document formats is to create images of the pages by using scanners and digital cameras for papers and books, or rendering page images for electronic files. Companies and libraries have already scanned millions of pages, with Google being a well-known example. For electronic document files, rendering them as images can either provide a temporary interface between application software systems, or as a way to mitigate the "bit rot" problem in archive content where files made by obsolete software can no longer be interpreted. An example of a document management system that emphasizes page image representation is UpLib [10].

In general, detecting pictures in page images is a difficult problem. Examples of one extreme are magazine articles with complicated and overlapping graphic designs. In this paper, we focus on the more tractable problem of picture detection in page images from technical papers and reports, which have fairly straightforward layout designs. By pictures, we include natural images like photos and synthetic images like block diagrams, graphs, screenshots, etc. Decorative graphics such as long lines or "rules" are excluded.

A related problem is separating text from graphics, for which a number of methods have been developed (e.g. [3], [6]). These methods are low-level analyses commonly based on geometric features of connected components or on mathematical morphology. Another similar problem is document layout analysis, for which various algorithms using geometric features have been developed (see [13]). In layout analysis, the focus is more on determining the structure and categorization of regions in a page layout than the detection of specific objects and their precise boundaries. When addressing these related problems, higher level semantic information of the text is not usually employed and the regions are susceptible to under- and over-segmentation.

Recently, machine learning techniques have been applied to page segmentation and region discovery. For example, in [4], patch-based codebook learning is employed to discover different region types in degraded page images. In an evaluation with 100 page images, while certain types of regions ("text", "whitespace", "images") were found with high accuracy, the type "figure" was not reliably detected.

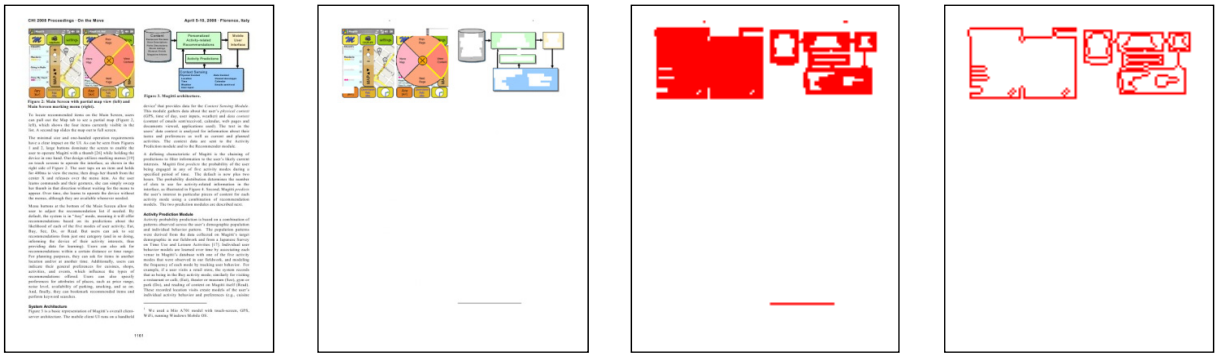
More specialized techniques based on the JPEG 2000 format have been developed to identify image segments. Hand-developed rules are used to assign each segment to either a text segments list or image segments list [2].

Another relevant technology to picture detection is OCR. OCR programs typically segment the page image into zones, primarily with the purpose of finding zones that contain text for character recognition. Thus, there is not a focus on good picture identification. More specifically, commercial OCR software (e.g. ABBYY) does not do a good job of detecting pictures: the zones often do not match the picture regions (cf. Section 3.2 below). In particular, they have problems with block diagrams and screenshots which contain sub-regions of text that makes it complicated for the OCR programs to handle. Other researchers have also observed that the zones are not well detected [8].

In our approach, we utilize OCR for what it does well, which is text extraction, and then employ both low-level and higher-level techniques that are focused on correct picture identification, both in locating regions and in determining the number of pictures. In particular, we mask out the text from the page image, and perform image segmentation on a simplified page image by clustering the non-text pixels.

To resolve under- and over-segmentation of the picture regions, we look for captions in the extracted text and match them to the picture regions, splitting and merging the regions as needed.

DocEng2010, September 21–24, 2010, Manchester, United Kingdom.



(a) (b) (c) (d)

Figure 1. (a) Example of a document page image. (b) The masked image created from using the OCR text bounding boxes. (c) The scaled-down binarized image created from the smoothed masked image. (d) The reduced binarized image, with specks and line graphics removed.

In the following sections we explain the method in more detail, and present the results of a performance evaluation on two sets of conference papers with originally scanned and electronically produced page images.

2. DETECTION METHOD

We assume that documents in other formats have been converted to page images. To obtain images from electronic files, software tools for rendering them are widely available (e.g. Xpdf). For images from scanners or cameras, a preprocessing step may be necessary to correct for skewing, warping, and uneven lighting.

2.1 OCR Masked Image

We use the Microsoft Office Document Imaging OCR technology to extract text tokens ("words") along with their bounding boxes. By setting the pixels inside these bounding boxes on the page image to the background color, we create a masked image (see Fig. 1b).

Since the input page image usually has fairly high resolution and this level of precision is not necessary for the computations, we downsample the masked image. Typically, the resolution of an 8.5" by 11" page at 200 dpi is 1700x2200. For our computations, we are using images of size 120x155. In order to avoid aliasing, we smooth the image before downsampling.

From the scaled-down masked image, we create a binarized image (Fig. 1c). The pixels that are not of the background color (differing by some small threshold) are labeled as *1-pixels*. All other pixels are labeled as *0-pixels*.

To improve efficiency, we reduce this binarized image by removing some of the interior 1-pixels (Fig. 1d). More precisely, the 4-connected points are removed using a morphological erosion residue edge detector [5]. By removing the interior points, we reduce the number of 1-pixels by a factor of a square root.

Another step that is required in practice is to clean up the image. Small specks and noise are filtered out by removing connected components whose width and height are below a certain threshold. Decorative graphics such as isolated lines with high aspect ratio are also removed. See Fig 1d.

2.2 Clustering by Normalized Cuts

To cluster the non-text pixels (1-pixels), we apply the Normalized Cuts algorithm ([14], [11]), or *NCUT* for short. As a spectral method, it has certain theoretical advantages over earlier generation clustering methods such as K-means [7].

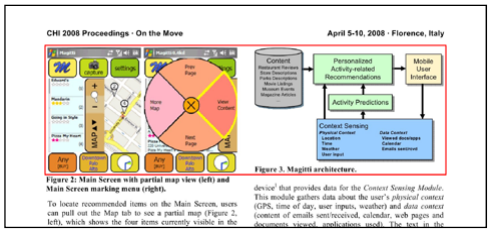
We note that NCUT can be directly applied to image segmentation by taking each pixel in the image as a node in a weighted graph [14]. This can work well on natural images like photographs; however, when we attempted to use it to segment document page images, it had difficulty segmenting out pictures (e.g. block diagrams, line graphs) that do not have dense and uniform patches of color or intensity. Moreover, by using all the pixels, the computation time for segmentation takes much longer than for clustering the smaller set of 1-pixels.

We also do a rescaling operation to improve the behavior of NCUT on elongated clusters that we observed in our experiments. It has been noted by others that NCUT may be viewed as clustering with hyperplanes and that it can sometimes split elongated clusters [12]. To address this, we rescale the binarized image in x and y to equalize the average width and height of the connected components.

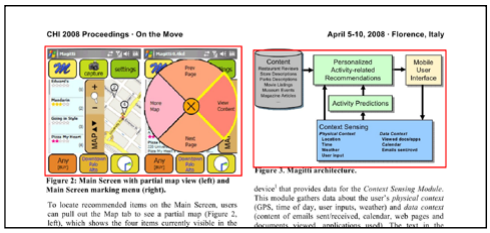
When applying NCUT to the rescaled 1-pixels, we specify a sequence $k = \{1, 2, 3, \ldots, N\}$ where k is the number of output clusters. This produces a sequence of clusterings $\{C_k\}$, where each clustering C_k is made up of clusters $\{c_{k,i}\}$. From this sequence, we use a heuristic to find the optimal clustering by looking for the k that maximizes the jump in the distance between the clusters in C_k and C_{k-1}.

2.3 Correcting for Under- and Over-Segmentation

After the clustering step, the clusters are refined to correct for under- and over-segmentation. A cluster may consist of two or more actual pictures and needs to be split, or several clusters may be part of a single picture and need to be merged. From the image properties alone, it may be ambiguous whether nearby clusters need to be split or merged. By using the caption information found in the OCR text, a better determination can be made. For example, a cluster that has two "Figure" captions nearby is likely to contain two pictures and therefore should be split. See Fig. 2a and 2b.



(a)



(b)

Figure 2. (a) The red bounding box shows the optimal clustering obtained from applying NCUT. (b) After refinement of the clustering using captions found in the OCR text, we obtain the final result.

To find the captions for the clusters from the OCR text, we do a string match for caption label words like "Figure", "Fig.", "Table", etc. The match can be approximate to handle the few characters that may be mistakenly recognized by the OCR. Using font information (e.g. bold style) can also improve accuracy. Checking the location of the text tokens, we eliminate those that are too far away from a cluster or that do not begin a line of text. Clusters associated with captions labeled as "Table" are eliminated.

When splitting a cluster that has more than one associated caption, we look at the connected components within the cluster. Because these do not always correspond to sub-optimal clusterings with a higher *k* index due to the closeness of the 1-pixels, the connected components produce a more effective splitting.

Merging clusters that have fewer associated captions is comparatively straightforward. For each cluster *c* that has a single associated caption, if there is another cluster that is nearby and aligned with *c* in either the horizontal or vertical direction, these clusters are merged. The process is iterated until no more merges are made.

Finally, a "picture" is identified with the bounding box of each refined cluster. The whole picture detection process is summarized in the block diagram in Fig. 3.

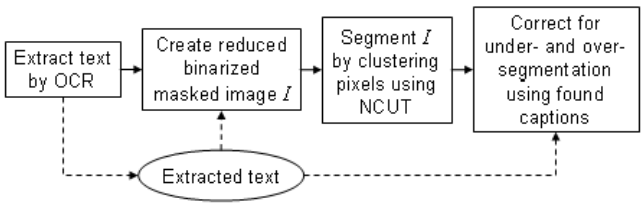


Figure 3. Summary of picture detection process.

3. PERFORMANCE EVALUATION

We adopt a performance evaluation scheme that is based on existing work for text region detection in documents and videos [9]. We give a brief description and a complete set of formulas.

It is also possible to apply more elaborate performance analysis schemes such as [1] developed for document layout analysis. However, these take into account the interplay between different types of regions and document structure features that are not germane to our problem, which is more akin to object detection.

3.1 Definitions

The basic idea presented in [9] is that if *x* is a ground truth region, then how well it is covered by the set of detected regions gives an indication of the correct detection rate (or true positive rate) *CD*. This measures "recall" in a more general way that is continuous. Similarly, if *x* is a detected region, how poorly it intersects the set of ground truth regions gives an indication of false discovery rate *FD*. This measures the complement of "precision". Another important characteristic is how much *x* is fragmented by the intersections, which gives some indication of under- and over-segmentation. By combining *CD* and *FD*, a score *TF1* is defined which is analogous to the *F1* score in information retrieval.

To be more explicit, we provide a complete set of simplified formulas to compute the evaluation scores; for further explanation and details, see [9]. Let X and Y be two sets of regions, then define the detection quality $Q(x)|\mathrm{Y}$ of a region $x \in \mathrm{X}$ against the set Y as a product of basic quality $Q_\mathrm{b}(x)|\mathrm{Y}$ and fragmentation quality $Q_\mathrm{f}(x)|\mathrm{Y}$:

$$Q(x)|\mathrm{Y} = Q_\mathrm{b}(x)|\mathrm{Y}\ Q_\mathrm{f}(x)|\mathrm{Y},$$

with

$$Q_\mathrm{b}(x)|\mathrm{Y} = \frac{\sum_{y\in Y} Q_D(y,x)\mu(y\cap x)}{\max(\mu(x), \sum_{y\in Y}\mu(y\cap x))}$$

$$Q_\mathrm{f}(x)|\mathrm{Y} = \frac{\sqrt{\sum_{y\in Y}\mu(y\cap x)^2}}{\sum_{y\in Y}\mu(y\cap x)}$$

where $\mu(\mathrm{x})$ is some measure such as the area of *x* or the number of Sobel edge points in *x*, and $Q_\mathrm{D}(y, x)$ provides the detection quality of overlapping regions with respect to *x*. Let $Q_\mathrm{D}(y, x) = \mu(y \cap x)/\mu(y)$. In the evaluations in this paper, we use Sobel edge points in $\mu(\mathrm{x})$.

In [9], an exponent is also used with $Q_\mathrm{D}(y, x)$ to account for the detection difficulty index of *x* and *y* based on the contents of those regions; for simplicity we leave out this exponent (i.e. set it to 1).

The total scores will be weighted averages over X. For notational convenience, we define a functional:

$$W_\mathrm{X}(\rho|_\mathrm{Y}, \tau) = \frac{\sum_{x\in X}\rho(x)|_Y\ \tau(x)}{\sum_{x\in X}\tau(x)},$$

where $\tau(x)$ measures the detection importance level, which we set to be $\mu(x)$, and $\rho(x)|_\mathrm{Y}$ will be defined in terms of the detection quality $Q(x)|_\mathrm{Y}$.

Then the correct detection rate *CD* is defined by

$$CD = W_\mathrm{X}(Q|_\mathrm{Y}, \mu),$$

in which X is the set of ground truth regions and Y is the set of detected regions.

And false discovery rate *FD* is defined by

$$FD = W_X (1 - Q|_{Y,\mu}),$$

in which Y is the set of ground truth regions and X is the set of detected regions.

Finally, the combined detection index *TF1* is defined as the harmonic mean of *CD* and (1 – *FD*).

3.2 Evaluation and results

We evaluated our method on two sets of page images from conference papers. From the older ACM UIST 1988-1997 proceedings in the ACM digital library, which has scanned page images of high quality but noticeable defects (embedded in PDF files), we selected 10 papers with a random number generator. These provided 82 page images containing 69 pictures. A second set was obtained from the IEEE ICME '09 proceedings. In this collection, the PDF files were created directly from software (without scanning defects). We randomly selected 20 papers, providing 80 page images containing 84 pictures.

The PDF files in both sets were converted to JPG images at 1700x2200 resolution using Xpdf software. Microsoft Office Document Imaging, which has an easily accessible API, was used to OCR the images. We hand-labeled the picture bounding boxes.

For benchmarking the results, ABBYY FineReader 10.0 is used as a baseline. This commercial system has been used for benchmarking research systems (e.g. ICDAR Page Segmentation Competition). It produces regions with labels {Text, Picture, Table, Barcode}. We take the bounding boxes of the ABBYY Picture regions for computing the performance evaluation.

The results are shown in Fig. 4. For the overall *TF1* score, our PicDetect method shows an improvement over ABBYY by 35% (UIST) and 40% (ICME). For the *CD* score, our method also performed much better; however, for the *FD* score, the results were mixed.

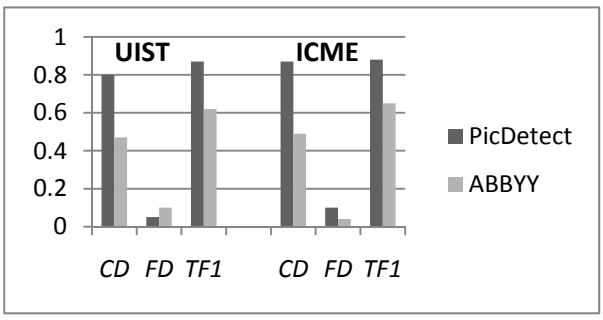


Figure 4. Evaluation results.

3.3 Discussion

The evaluation indicates that the basic method works well without much additional rules for handling special cases. One source of error is that a bunch of equations on adjacent lines may not be sufficiently masked out, leading to a false alarm. This is due to limitations of the OCR engine in recognizing mathematical symbols. A remedy is to add a module specialized to recognize regions with equation features and eliminate them to improve performance.

Gains can also be made by domain knowledge about the formatting. For example, a rule that checks for consistent placement of captions above or below pictures can lead to more accurate splitting and merging of regions.

4. CONCLUSION & FUTURE WORK

We presented an approach for picture detection in document page images that leverages OCR and pixel clustering algorithms. As a document engineering solution, it can be built by utilizing existing software libraries for the OCR and clustering functions. Our PicDetect method performs well compared to ABBYY FineReader, and the evaluation also suggests some ways that PicDetect can be enhanced.

For future work, we plan to do more extensive testing on more types of documents. Another direction is to make use of the extracted caption text for tagging the detected pictures. These can be used to facilitate social and collaborative applications around documents.

5. REFERENCES

[1] Antonacopoulos, A., Bridson, D. Performance analysis framework for layout analysis methods. *Proc. ICDAR '07*, pp. 1258-1262.

[2] Berkner, E.L., Schwartz, C., Marle, C. SmartNails - Image and display dependent thumbnails. *Proc. of SPIE '04*, vol. 5296, pp. 53-65.

[3] Bloomberg, D.S., Chen, F.R. Extraction of text-related features for condensing image documents. *Proc. of SPIE '96*, *Document Recognition III*, pp. 72-88

[4] Burns, T.J., Corso, J.J. Robust unsupervised segmentation of degraded document images with topic models. *Proc. of CVPR '09*, pp. 1287-1294.

[5] Chanda, B., Kundu, M., Padmaja, Y. A multi-scale morphologic edge detector. *Pattern Recognition*, 31 (10): 1469-1478, Oct. 1998.

[6] Fletcher, L.A., Kasturi, R. A robust algorithm for text string separation from mixed text/graphics images. *IEEE TPAMI*, 10 (6): 910-918, Nov. 1988.

[7] Gong, Y., Xu, W. *Machine Learning for Multimedia Content Analysis*. Springer, 2007.

[8] Hauser, S.E., Le, D.X., Thoma, G.R. Automated zone correction in bitmapped document images. *Proc. of SPIE '00*, *Document Recognition and Retrieval VII*, pp. 248-258.

[9] Hua, X.-S., Liu, W., Zhang, H.-J. Automatic performance evaluation for video text detection. *Proc. of ICDAR '01*, pp. 545-550.

[10] Janssen, W.C., Popat, K. UpLib: A universal personal digital library system. *Proc. of DocEng '03*, pp. 234-242.

[11] Normalized Cuts software. http://www.cis.upenn.edu/~jshi/software/.

[12] Rahimi, A., Recht, B. Clustering with Normalized Cuts is clustering with a hyperplane. *Statistical Learning in Computer Vision 2004*.

[13] Shafait, F., Keysers, D., Breuel, T.M. Performance evaluation and benchmarking of six page segmentation algorithms. *IEEE TPAMI*, 30 (6): 941-954, Jun. 2008.

[14] Shi, J. Malik, J. Normalized cuts and image segmentation. *IEEE TPAMI*, 22 (8): 431-439, Aug. 2000.