Document layout analysis, Classical approaches

(5 Papers / 1992 : 2001 - International Conference on Document Analysis and Recognition)

A Resume

November 23, 2018

Abstract

Classical approches for document segmentation and layout analysis are discussed (paper discussed date back to 2001 and earlier).

1 Document layout/structure analysis

All documents have some structure, either perceived by the reader or defined by the basic publishing units such as letters, words, lines and blocks. The first type of structure is referred to as the **logical structure** of the document, while the second is termed the **physical structure** of the page. The logical structure of a document can vary for both subjective reasons (since different people can interpret the same document differently) and objective reasons (different types of document contain different logical units). The physical structure of a document is more clearly defined than the logical structure. It contains fewer elements, that are all well-defined (characters, words, lines etc.) and are independent of their contents. The series of procedures that detect both the physical and the logical structure of a document, and map the two structures onto each other is termed **document layout analysis and understanding**.

Document layout analysis finds the components of the page, i.e., the words, lines and blocks, and forms one of the first steps of all the processing. Results from this stage are important for all the other processes performed after layout analysis: the word and line information is essential for optical character recognition (OCR), the block information is essential for the extraction of the logical structure of the document, i.e., for understanding the document layout [4].

Wikipedia definition: **Document layout analysis** is the process of identifying and categorizing the regions of interest in the scanned image of a text document. Detection and labeling of the different zones (or blocks) as text body, illustrations, math symbols, and tables embedded in a document is called geometric layout analysis (physical structure ?). But text zones play different logical roles inside the document (titles, captions, footnotes, etc.) and this kind of semantic labeling is the scope of the logical layout analysis.

The logical structure of a document associates semantic information to the physical document. It reflects the way information is organized in terms of logical objects, i.e., chapters, titles, paragraphs, figures, headers, etc. The logical structure specifies the function and meaning of the physical objects forming the document and the relationships between them. More precisely, the relationships between the logical objects in the logical structure are typically in the form of hierarchichal include and sequence relations.
For example, a document includes a title, authors, summary, and a sequence of chapters, each of

them including a title and a sequence of sections, and so on. Thereby, the logical structure is commonly represented as a tree structure in order to transcribe the hierarchial relationships existing between the various logical objects.

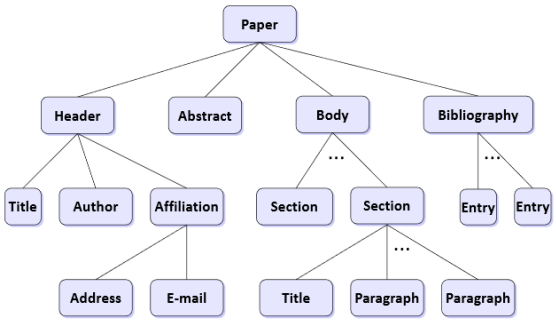


Figure 1: The logical structure of the document shown as a tree structure and formalized in an XML-based representation

In a physical document, the logical structure is no more explicitly written, but it is revealed thanks to the documents typographical attributes and layout, i.e., **the physical structure** . The physical structure of a document corresponds to the organization of the page in terms of a hierarchy of regions delimited by images, graphics, and text blocks that can be further split into text lines, words, and characters.

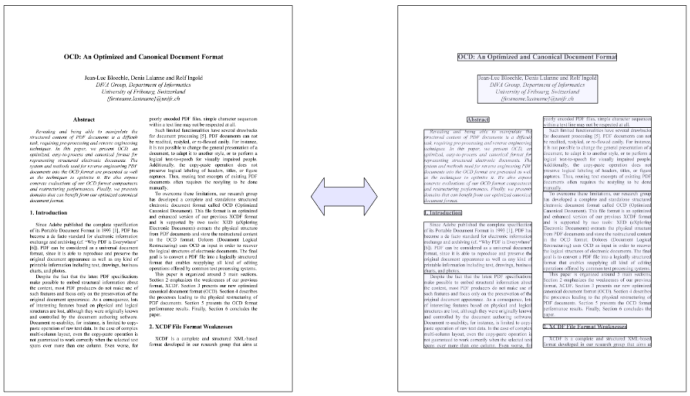


Figure 2: On the left, the first page of a conference paper, and on the right, the text blocks segmentation corresponding to the physical structure of the conference paper

Thus, the visual appearance of a printed document simply reflects its physical structure, without any semantic interpretation. As the logical structure, the physical structure is often represented as a tree structure in order to transcribe the hierarchial relationships existing between the various physical objects . Source: Link

2 Bottom-up vs. Top-down

Top-down Attempt to iteratively cut up a document into columns and into blocks, and the blocks into lines and the lines into words, based on white space and geometric global information.

Very elegant, fast and efective, but good results can be obtained only for documents with fixed and well known structure. It uses the knowledge of logical structure of the document described in a high level language and verifies the localisation of the entities extracted from the document contents and the physical entities extracted from document layout.

Bottom-up Iteratively parse a document based on the raw pixel data. Starting with local information (concerning the black pixels, or the connected components). These approaches typically first parse a document into connected regions of black and white, then these regions are grouped into words, then into text lines, and finally into text blocks.

Time consuming and less elegant, but it can be used for reading any type of unconstrained documents. It analyses the image to extract all connected components, then merges the components into words, lines, and then into paragraphs, all unclassified componenets are rejected and considered as graphics. An intermediate processing can be used to separate graphics from text.

Hybrid approaches Some approaches cannot easily be classified as either top-down or bottom-up, and have been described as hybrid methods since they use elements of both approaches, such as [12]

3 Paper 1: Bottom-up

A fast and efficient method for extracting text paragraphs and graphics from unconstrained documents [1]

This paper outlines fast and efficient method for extracting graphics and text pragraphs from printed documents using a bottom-up approach, thus achieving good performances in most cases. This is done in multiple processing steps.

> **Side note : Run Length Smoothing Algorithm (RLSA)** The basic RLSA is applied to a binary sequence in which white pixels are represented by 0s and black pixels by 1s. The algorithm transforms a binary sequence x into an output sequence y according to the following rules:
> **1.** 0s in x are changed to 1s in y if the number of adjacent 0s is less than or equal to a predefined limit C. **2.** 1s in x are unchanged in y .
> For example, with C = 4 the sequence x is mapped into y as follows:
> x : 0 0 0 1 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 0
> y : 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 1 1 1 1 1
> When applied to pattern arrays, the RLSA has the effect of linking together neighboring black areas that are separated by less than C pixels. With an appropriate choice of C, the linked areas will be regions of a common data type.

1- Preprocessing: Linking characters and graphics to form blocks during preprocessing.

RLSA is applied to the image as a preprocessing step, it links together neighbouring black areas that are seperated with that a given number of pixels, this is done both vertically and horizentaly, and we take their logical intersection afterwards.

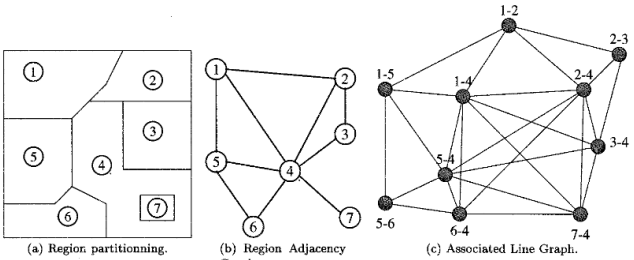
Problems : Memory requirements, processing time and possible linking lines when applying horizental smoothing.

The paper proposes **Reduction-Dilation**: Only applying vertical smoothing and using horizental dilation, and reducing the size of the image (factor of 8 and 3 H-ly and V-ly) to store the image in the memory and speeding up the processing.

> **Side note: Erosion & Dilation** Two basic operators in mathematical morphology. They are typically applied to binary images, but can work on grayscale images. The basic effect of the operators on a binary image is to gradually erode /enlarge the boundaries of regions of foreground pixels
> Definition : Suppose that X is the set of Euclidean coordinates corresponding to the input binary image, and that K is the set of coordinates for the structuring element.
> - Let Kx denote the translation of K so that its origin is at x.
> Erosion : The erosion of X by K is simply the set of all points x such that Kx is a subset of X.
> Dilation : The dilation of X by K is simply the set of all points x such that the intersection of Kx with X is non-empty.

2- Segmenting and labeling : Labelling blocks and simultanously extracting graphics from the image.

Side note: Line Graph of the Region Adjacency Graph Region adjacency graph (RAG) consists of associating a vertice at each region and an edge at each pair of adjacent regions. By definition, this region adjacency graph provides a simple-connectivity view of the image.
Another way to study regions connectivity relationships consists of analyzing the line graph (LG) of the RAG. The line graph G' of a graph G is defined as follows:
- at each vertice of G' corresponds an edge of G;
- at each edge ofG' corresponds a pair of adjacent vertices in G'.
This structure enables to compare each pair of adjacent regions $\{R_i; R_j\}$ to any other pair of adjacent regions $\{R_j; R_k\}$ which have a region R_j in common. At each region R_i corresponds a color value μ_i and a size parameter s_{R_i}. Consequently, at each vertice v_{ij} of G' corresponds a pair of adjacent region $R_i; R_j$, a color distance $d_{ij} = d^2(R_i; R_j)$, and a weight $w_{ij} = \min(s_{R_i}, s_{R_j})/(s_{R_i} + s_{R_j})$



(a) Region partitionning. (b) Region Adjacency Graph. (c) Associated Line Graph.

Based on line adjacency graph method, capable of working on two lines simultaneously and requires less memory. Both segmentation and block labeling are done at the same time. With the minimum block sizes $(X_{Bmin}, X_{Bmax}, Y_{Bmin}, Y_{Bmax})$ and a density block (DS_B), the following semgmentation rules are applied:
- Block within the minimum range are treated as text/graphics.
- Divide block into subparts if the difference between each line becomes greater than 25%.
- Reject block if it contains other blocks.
- Treat block as lines of text if it's density is in the range we defined.

3- Merging line of text into paragraphs. Two rules can be applied : typographical (depends on the parameters of the paragrah : alignement, justification, interline...) and the geographical rules (aggregate similar/close blocks using information about logical structure).

In this paper only second rule is used, the merging is performed is three steps: - x-y coordinates of blocks are stored in descending order.
- lines which are close enough are merged using geographical rules.
- logical analysis is carried out on paragraph composition and achieved coherence is verified.

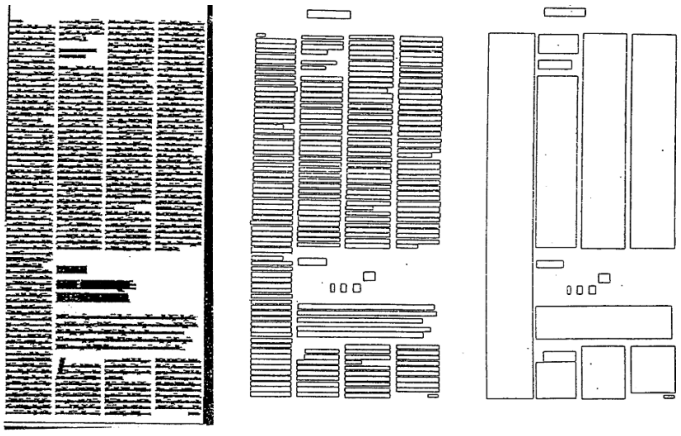


Figure 3: The results at each step

4 Paper 2: Top-down

Document page decomposition by the bounding-box projection technique [2]

In this pape the problem of extracting words, textlines and text blokcs is viewed as a clustering problem in the 2-dimentionnal discrete domain, using profiling analysis to measure horizontal or vertical gaps of componenets during the segmentation.

Step-by-step process:

1- Bounding boxes of connected componenets Apply a connected component analysis algorithm to the foreground to the input binary image I producing a set of connected componenets. Then, associate to the smallest rectangular box which circumscribes the component a bounding box.

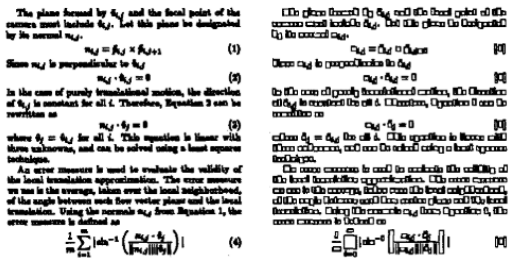


Figure 4: The results of the first step

2- Projections of Bounding Boxes By projecting the bounding boxes onto the horizental and/or vertical line, the projection will accumulate onto the same line resulting in the projection profile. A projection profile is a frequency distribution of the projected bounding boxes, providing information about the number of bounding boxes aligned along the projection direction.

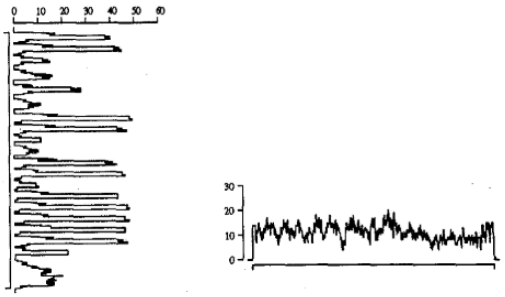


Figure 5: The results of the second step

3- Extraction of Textlines From the above Figure, it is easy to observe that textlines are horizontally oriented, with distinct high peaks and deep valleys at regular intervals. The coordinate of the bounding boxes spaning the whole lines are then obtained for the high peaks.

4- Extraction of words

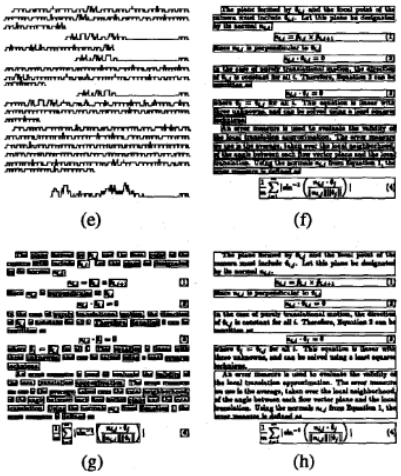


Figure 1 (continued): (e) vertical projection profiles, (f) textline bounding boxes, (g) word bounding boxes, (h) text-block bounding boxes.

Figure 6: Words & Paragraphs extraction

After having bounding boxes regrouping each line at a time, this the algorithm groups them into boxes of words. First compute the projection profiles within each of the textline bounding boxes, each projection profile is considered as a one-dimensional gray-scale image, and then thresholds each of the images with a threshold of one to produce a binary image.
After such binarization, the algorithm performs a morphological closing operation on each textline with a the appropriate structuring elementn, in the valey between the inter-character spacings within words and inter-word spacing, to close the gaps between charaters of the same words, Otsu's automatique can be used to find the optimal threshold in the said valey.

Side note: Otsu Thresholding Otsu's thresholding method involves iterating through all the possible threshold values and calculating a measure of spread for the pixel levels each side of the threshold, i.e. the pixels that either fall in foreground or background. The aim is to find the threshold where the sum of foreground and background spreads is at its minimum.
We calculte with within class variance for various thresholds, and choose the one with the lowest variance.

$$\sigma_{within-class} = W_f \times \sigma_f + W_b \times \sigma_b$$

$$W_f = \frac{\#foreground\ pixels}{\#pixels}$$

$$W_b = \frac{\#background\ pixels}{\#pixels}$$

$$\sigma_f = \frac{1}{\#foreground\ pixels} \sum_{i=1}^{n} (x_i - mean_f)^2$$

$$\sigma_b = \frac{1}{\#background\ pixels} \sum_{i=1}^{n} (x_i - mean_b)^2$$

5- Extraction of paragraphs In this step, the extracted bounding boxes of text-blocks are grouped together. Based on the physical appearance of the document image, the logical hierarchy can be deduced, when a significant change in the textline heights or the inter-textline spacings occurs, we might sat a new paragraph begins.

5 Paper 3: Top-down

Recursive x-y cut using bounding boxes of connected components [3]

The paper proposes a top-down page segmentation technique known as recursive X-Y cut, by recursively decomposing a document image into a set of rectangular blocks using bounding boxes of connected components of black pixels instead of image pixels.

Traditionally, some effort were made to initiate document image understanding by projecting image pixels as a top-down segmentation method, to decompose document images, local peaks and valleys of pixels projection can be used to determine where to apply the division. But it is computationally inefficient, why not projecting bounding boxes instead of image pixels? it is less computationnaly involved with the possibility to infer how the bounding boxes are alinged.

Projection and profiles V and H projections and profiles can be formulated as follows. for a given bounding boxes $B = \{b_1, b_2, ..., b_N\}$ where each b_i encloses the region:

$R_{bi} = \{(x, y)|x_{bi,min} \leq x \leq x_{bi,max}$ and $y_{bi,min} \leq y \leq y_{bi,max}\}$

The horizental projection of bb b_i is the association of b_i with a scalar function H_{bi} which is called the height function of b_i:

$$H_{bi}(y) = \begin{cases} 1 \text{ if } y_{bi,min} \leq y \leq y_{bi,max} \\ 0 \text{ otherwise} \end{cases}$$

And the same of the vertical projection.

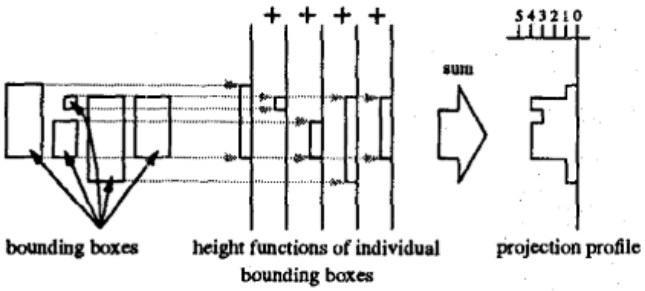


Figure 7: Horizental projection profile

The projection profile of a set of bb can be defined using the abose two projection profiles. the horizental projection profile of B is a function form Z to Z: $H_B = \sum_{i=1}^{N} H_{bi}$, same for vertical projection : $V_B = \sum_{i=1}^{N} V_{bi}$

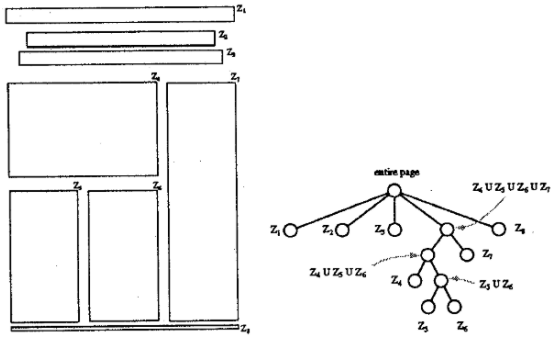


Figure 8: X-Y tree of the document, recursively being segmented

Segmentation by Recursive X-Y cut At each step, the pixel-projection profiles are calculated in both horizental and vertical direction, then a zone division is performed ar the most prominenet valley in either projection profile, and the process is repeated recursively until wide valleyts are left in both profiles.

The structured subdivsion of a document image by recursive X-Y cut makes the document to be represented in the form of a tree with nested rectangular blocks; A X-Y tree, the root node is the

bb of the full pagen, each node represents a rectangle in the page, the children are obtained by subdividing the rectangle of the parent node either H-ly or V-ly, both being used alternately levle by level.

The algorithm for the recursive X-Y cut is described as follows, given a document image;
1- Do preprocessing (e.g noise removal, skew correction, ect.),
2- Apply a connected component labeling algorithm,
3- Obtain the bounding boxes of connected components,
4- Create a root node,
5- Calculate the horizontal and vertical projection profiles within the region of interest,
6- Do division at large gaps in the projection profiles whose widths exceed a threshold value V_{thr}, each time creating child nodes,
7- Do steps 5-6 recursively until no futher divisions are possible (H and V division altenate).

6 Paper 4: Bottom-up

A fast algorithm for bottom-up document layout analysis [4]

A method based on Kruskal's algorithm, and uses a special distance-metric between the components to construct the physical page structure
Advantages of the proposed method: has the advantages of bottom-up methods such as independence from different text spacing and independence from different block alignments, no necessity of a prior knowledge about character size of line-spacing, detection of all elements (words, lines, blocks, columns, stripes); efficient processing of multicolumned images involving complicated text and graphics arrangements, and the complexity is linear using heuristics and path-compression.

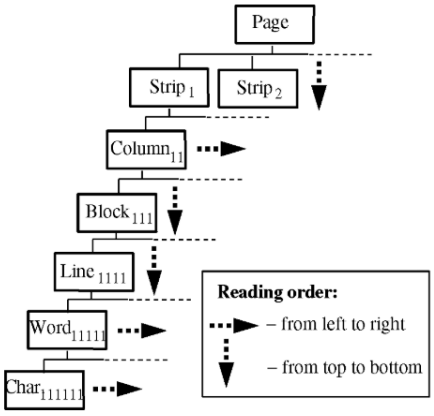


Figure 9: Physical structure represented as a graph.

The document layout analysis starts by processing the connected components of the image, and results in a list of words, text lines, text and graphic blocks, columns and strips. After the image has been loaded, the connected components of the page are found and the noise-like connected components are removed. Document layout analysis starts with calculation of the distances between the pairs of the connected components. The connected components represented as the vertices of a graph, and the distances between them as the weighted edges of the same, then words, lines, blocks, etc., can be derived from the minimal-cost spanning tree, built with Kruskals algorithm.

Side note: Binary/ N-ary Trees A binary tree is a structure comprising nodes, where each node has the following 3 components:

- Data element: Stores any kind of data in the node,
- Left pointer: Points to the tree on the left side of node,
- Right pointer: Points to the tree on the right side of the node.

As the name suggests, the data element stores any kind of data in the node. The left and right pointers point to binary trees on the left and right side of the node respectively. If a tree is empty, it is represented by a null pointer.

Side note: Weighted graphs In many applications, each edge of a graph has an associated numerical value, called a weight. Usually, the edge weights are non- negative integers. Weighted graphs may be either directed or undirected. The weight of an edge is often referred to as the "cost" of the edge. In applications, the weight may be a measure of the length of a route, the capacity of a line, the energy required to move between locations along a route, etc.
Given a weighted graph, and a designated node S, we would like to find a path of least total weight from S to each of the other vertices in the graph. The total weight of a path is the sum of the weights of its edges. Dijkstra's SSAD Algorithm is a simple, greedy algorithm that will solve this problem.
We assume that there is a path from the source vertex s to every other vertex in the graph. Let S be the set of vertices whose minimum distance from the source vertex has been found. Initially S contains only the source vertex. The algorithm is iterative, adding one vertex to S on each pass.

Side note: Kruskal's Algorithm Kruskal's algorithm is a minimum spanning tree algorithm that takes a graph as input and finds the subset of the edges of that graph which
- form a tree that includes every vertex
- has the minimum sum of weights among all the trees
- that can be formed from the graph

We start from the edges with the lowest weight and keep adding edges until we we reach our goal.

The steps for implementing Kruskal's algorithm are as follows:

-Sort all the edges from low weight to high
-Take the edge with the lowest weight and add it to the spanning tree. If adding the edge created a cycle, then reject this edge.
-Keep adding edges until we reach all vertices.

Definition of the distance The distance between any two componenets (word, line block or CC) of a page is expressed with the distance between their enclosing boxes.

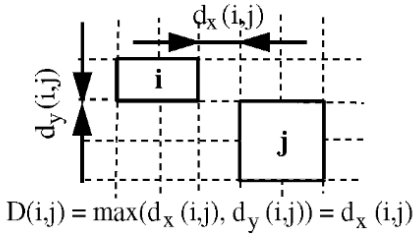


Figure 10: Distance calculations.

$$D(i,j) = \max(d_x(i,j), d_y(i,j))$$

Where:

$$d_x(i,j) = \begin{cases} 0 & \text{if } L(i,j) < R(i,j) \\ L(i,j) - R(i,j) & \text{otherwise} \end{cases}$$

$$L(i,j) = \max(CC_i.left_x, CC_j.left_x)$$
$$R(i,j) = \min(CC_i.right_x, CC_j.right_x)$$

Applied Kruskals Algorithm Let us consider the connected components of an image to be the vertices of an undirected graph, and the distances between any of them as the edges. A complete graph is therefore $G = (V, E)$ with $V = |n|$ and $E = \binom{2}{n}$, where n is the number of the connected componenets, this is the page graph. Each edge (i, j) has a cost that is $D(i, j)$.

A spanning tree of a connected graph is a tree that contains all the vertices of the graph and those vertices are all connected. A minmal-cost spanning tree of a graph is the one spanning tree, for which the sum of the edges is minimal compared to the other spanning trees of the same graph.

A minimal spanning tree of the page graph, generated with the Kruskal algorithm maps extacly the closest connection between the physical unit of the page.

The minimal-cost spanning tree is built by always inserting the smallest of the remaining unused distances. Hence in each step of the algorithm, the actual state contains some number of components, that have the smallest internal distance at the current level (initially all the vertices are in different components). To avoid connecting two lines (sometimes even with 0 distance they don't belong to the same line) there must be at least a horizontal intersection of 70%. So starting from the page we only connect close vertices, the character of the same word connected to the same parent, and this nodes is connected to the closest words in the same line and al of the line node, which is connected to the paragraph node and to the page node.

Path-Compression Algorithm and Some Heuristics for Reducing Processing Time The calculation of all the distances (edges) require $O(n^2)$, where $n = |V|$ is the number if the connected components, so to reduce the processing time we can only calculate the edges that are necessary, i.e. only components with vertical intersection and with similar y-coordiantes. This is especially important in the first two levels (words and lines) where the number of vertices is much larger that the upper levels. So only calculate the edges of the words components that are above some threshold of vertical intersection. ect. This can reduce the time to $O(n)$.

Classification of the Segments The class of a segment is inherited from the previous, lower layers, i.e., if a word has been classified as graphic, then by ex- panding it into a line it will automatically become a graphic line, thereafter a graphic block. It is important to classify the segments already at the words level, as the necessary edges heuristics are different for text and graphics. Text words must have a significant vertical intersection to be considered as belonging to the same line, while graphic words are merged with any other word which is close enough to their enclosing boxes.

7 Paper 5: Bottom-up

Page segmentation and classification utilising a bottom-up approach

This paper presents the use of analyzing the connected components extracted from the binary image of a document page (**even skewed images** !). And they describe two new algorithms, one for skew detection and one for skew correction.

Possible solutions to skewed images : Hough transform, Projection profile technique, Two dimentional Fourier transform, Correlation and regression methods, Complexity variance, For a comprehensive review of these techniques : Link

Digitization and Thresholding The program generates a histogram from all pixel values and smoothes it by a sliding average of the number of pixels of each gray scale intensity. Then, it checks for local maxima. The two maxima with the greatest number of pixels are then chosen to represent the gray scale levels for the foreground objects such as characters and background image (M1 and M2, respectively). To find the cutoff level for the background, Otsus algorithm is then applied to find a threshold between M1 and M2 (They check for M3, an intermediate value).

Segmentation The segmentation algorithm firstly determines the $cc's$ of the page and then applies a merging technique using a nearest-neighbor merging procedure to group together $cc's$ of similar dimensions.

1- Creation of connected components The objective of the connected component analysis of an image is to form rectangles around distinct components on the page, whether they are textual elements such as characters or nontextual elements such as images. These bounding rectangles then form the skeleton for all future analysis on the page.
This is an iterative procedure, comparing successive scanlines of an image to determine whether black pixels in any pair of scanlines are connected together. Bounding rectangles are extended to enclose any groupings of connected black pixels between successive scanlines.

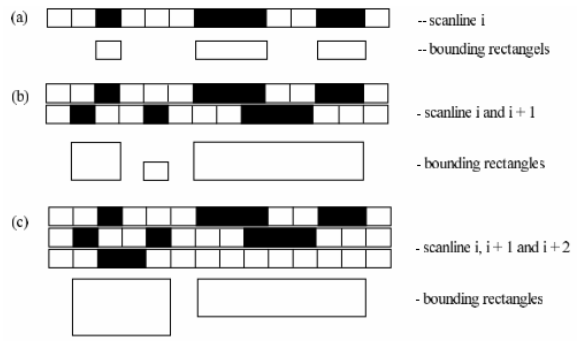


Figure 11: Creating of bounding boxes.

2- Grouping All the $cc's$ of documents fall into one of three categories, small, medium or large depending on their size. The $cc's$ are then merged in accordance to the category they fall under using different thresholds.
Taking one cc at a time and tries to merge it into a group from a set of existing groups. If it succeeds, the group's dimensions are altered so as to cater for the new cc, that is the group encompassing the rectangle is expanded so as to accommodate the new cc along with the existing $cc's$ already forming the group. Success is achieved when the cc and a group (both of the same category) are in close proximity to each other and are thus merged.

Skew Detection and Correction Before a page may be correctly segmented, it is necessary to have zero or almost zero skew.

First, the page is divided into vertical section with a width equal the average width of the connected components, the cc closest to the baseline remains in the bin and others are discarded.

To find the skew angle, Hough Transform is applied to each point at the center of the bottom of each rectangle, the resulting Hough accumulator is examined to find the two points at which most curves cross. The maximum of these values corresponds to the dominant skew angle of the page.

Given that some old pixels will point to the same location after rotation, and some spaces will remain blank, the image can be degraded if we simply tranformed the skew image pixel to pixel. Therefore for each (x_{skewed}, y_{skewed}) the correct value for a pixel at location (x, y) is calculated based on the four neighbooring pixels. Then the transformation is applied to (x, y) to obtain the

old coordinates.

$$x_{old} = x\cos\alpha + y\sin\alpha$$

$$x_{old} = y\cos\alpha - x\sin\alpha$$

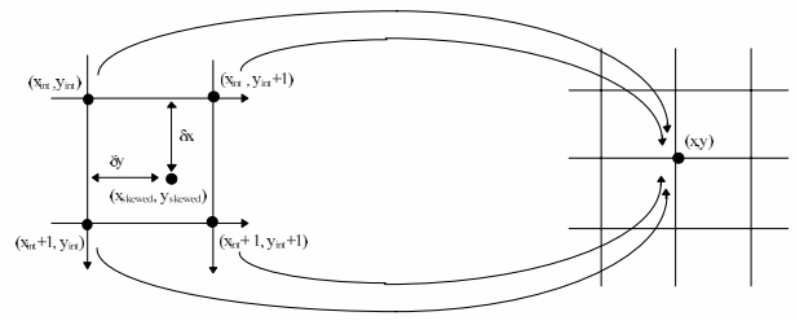
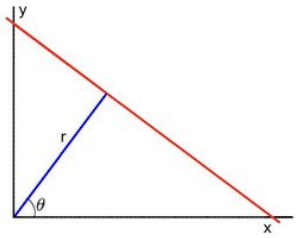


Figure 12: Weighted contribution to (x,y).

Side note: Hough Transform The Hough Line Transform is a transform used to detect straight lines. To apply the Transform, first an edge detection pre-processing is desirable.

a line in the image space can be expressed with two variables. For example:
In the Cartesian coordinate system: Parameters: (m, b). In the Polar coordinate system: Parameters: (r, θ)



For Hough Transforms, lines are expressed in the Polar system. Hence, a line equation can be written as:

$$y = \frac{-\cos\theta}{\sin\theta}x + \frac{r}{\sin\theta}$$

After re-arranging the terms : $r = x\cos\theta + y\sin\theta$
In general for each point (x_0, y_0), we can define the family of lines that goes through that point as:

$$r_\theta = x_0 \cdot \cos\theta + y_0 \cdot \sin\theta$$

Meaning that each pair (r_θ, θ) represents each line that passes by (x_0, y_0).

So a line (in Cartesian system) can be detected by finding the number of intersections between curves (In polar system). The more curves intersecting means that the line represented by that intersection have more points. In general, we can define a threshold of the minimum number of intersections needed to detect a line.

Classification After a page is partitioned into coherent blocks, the next step is to classify each block as text or graphics.
A frequency histogram of $cc's$ is used, along with the ratio b/w of the total area b covered by the $cc's$ over the total area covered by the white pixels w. These two criteria are tied together with the use of a third technique, which makes use of the geometric observation that $cc's$ representing graphical entities usually contain internal $cc's$. This internal $cc's$ are bounding rectangles for either text or graphics. The combination of all three discriminators is applied so as to distinguish between text and graphics.

Side note: Image transformations An affine transformation is any transformation that preserves collinearity (i.e., all points lying on a line initially still lie on a line after transformation) and ratios of distances (e.g., the midpoint of a line segment remains the midpoint after transformation).

In general, an affine transformation is a composition of rotations, translations, magnifications, and shears.

$$u = c_{11}x + c_{11}y + c_{13}$$
$$v = c_{21}x + c_{22}y + c_{23}$$

c_{13} and c_{23} affect translations, c_{11} and c_{22} affect magnifications, and the combination affects rotations and shears.

Example: for rotation, A rotation is produced by θ:

$$u = xcos(\theta) + ysin(\theta)$$
$$v = -xsin(\theta) + ycos(\theta)$$

Combinations of Transforms Complex affine transforms can be constructed by a sequence of basic affine transforms.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Composite Affine Transformation The transformation matrix of a sequence of affine transformations, say T_1 then T_2 then T_3 is :

$$T = T_3T_2T_3$$

Projective transformation
- The projective transformation does not preserve parallelism, length, and angle. But it still preserves collinearity and incidence.
- Since the affine transformation is a special case of the projective transformation, it has the same properties. However unlike projective transformation, it preserves parallelism.

Projective is represented as follows:

$$\begin{bmatrix} u' \\ v' \\ w \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ j & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Such that : $u = \frac{u'}{w}$ and $v = \frac{v'}{w}$
Why use homogenous coordiannates? They simplify and unify the mathematics used in graphics:
- They allow you to represent translations with matrices.
- They allow you to represent the division by depth in perspective projections.

8 Other methods

(from [4])

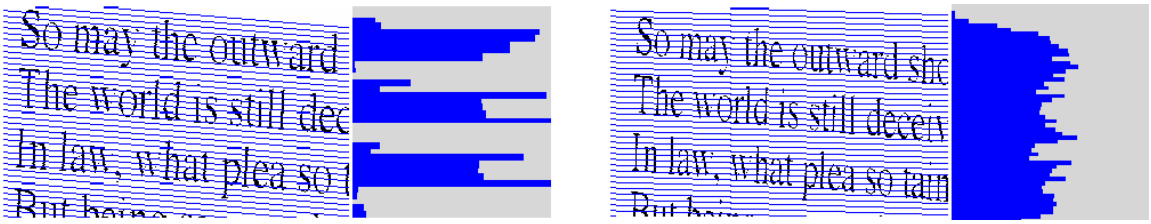
Top-down The global information in the case of the top-down method given by Krishnamoorthy et al. [6] is the white gap between the segments, that is used in the proposed recursive X-Y cut algorithm. Pavlidis and Zhou [7] introduced a method, that even though employs the white streams and projection profile analysis, still does not require preliminary skew-correction, as the analysis is done on short-height segments. The implied assumption for this method is that the page does not have severe skew. A robust, multi-lingual top-down method was proposed by Ittner and Baird [8]. After correction of the skew of the image, this method detects the blocks on the basis of the white streams. It generates the minimal spanning tree for detection of the text line orientation, and finally uses the projection profiles of the blocks to find the text lines. All of these methods, as is typical for the top-down technique, work only for Manhattan layouts, i.e., for pages with clear horizontal and vertical white gaps between and within the blocks. Even with this constraint,

it is acknowledged that the top-down techniques are important, as significant proportion of the documents have Manhattan layouts. In addition top-down analysis methods are generally fast, as finding the white gaps is usually a linear algorithm.

Bottom-up Bottom-up techniques use local information to build up higher information units. OGorman [9] has proposed the docstrum method, where the relationship of the objects are expressed with polar coordinates (distance and angle). The image is segmented by finding the k nearest-neighbor pairs between the components. Text orientation and the spacing parameters are estimated. With these features the algorithm works for most layouts. Tsujimoto and Asada [10] have proposed a segmentation method, that performs the calculations on the runlength representation of the image, and is more efficient than a bitmap representation. Connected components are extracted, and segments iteratively merged and classified. Saitoh et al. [11] have proposed a similar method, but the reduction of the image is done in both, x and y, directions, i.e., the pixels in the reduced image correspond to a square in the original image.

Skew correction There are several commonly used methods for detecting skew in a page, some rely on detecting connected components (for many purposes, they are roughly equivalent to characters) and finding the average angles connecting their centroids. The method we employed (after observing it in Fateman's program) was to project the page at several angles, and determine the variance in the number of black pixels per projected line.

The projetion parallel to the true alignment of the lines will likely have the the maximum variance, since when parallel, each given ray projected through the image will hit either almost no black pixels (as it passes between text lines) or many black pixels (while passing through many characters in sequence). Oblique projections will commonly pass both through lines of text, and spaces between lines, the variance in the number of pixels hit by the individual rays will thus be smaller than in the parallel case.



References

[1] Lebourgeois, F.Bublinski, Z.Emptoz, H., *A fast and efficient method for extracting text paragraphs and graphics from unconstrained documents*

[2] Ha, Jaekyu Ha Jaekyu, Haralick, R.M.Phillips, I.T., *Document page decomposition by the bounding-box projection technique*

[3] Ha, Jaekyu Ha Jaekyu, Haralick, R.M.Phillips, I.T., *Recursive xy cut using bounding boxes of connected components*

[4] Simon, Anik, Pret, J.-C., Johnson, A.P., *A fast algorithm for bottom-up document layout analysis*

[5] D. Drivas ; A. Amin, *Page segmentation and classification utilising a bottom-up approach*

[6] M. Krishnamoorthy, G. Nagy, S. Seth, and M. Viswanathan, *Syntactic Segmentation and Labeling of Digitized Pages from Technical Journals*

[7] T. Pavlidis and J. Zhou, *Page Segmentation and Classification*

[8] D.J. Ittner and H.S. Baird, *Language-Free Layout Analysis*

[9] L. OGorman, *The Document Spectrum for Page Layout Analysis*

[10] S. Tsujimoto and H. Asada, *Major Components of a Complete Text Reading System*

[11] T. Saitoh, T. Yamaai, and M. Tachikawa, *Document Image Segmentation and Layout Analysis*

[12] K. Kida, O. Iwaki, and K. Kawada, *Document Recognition System for Office Automation*