# Sketch-a-Doc: Sketch a Document to Find It

Filipe Rodolfo Alves, Manuel João Fonseca, Daniel Gonçalves
Dep. Engª. Informática, IST
Av. Rovisco Pais, 1000 Lisboa
filipe.alves@ist.utl.pt, {mjf,daniel.goncalves}@inesc-id.pt

**Summary**
*With the vast amount of documents users tend to accumulate in their hard drives, it is natural that they often forget where certain file is stored or even its name. However, sometimes they still recall a mental image of the document's layout. We developed a new approach to document retrieval that capitalizes on human visual memory to help users find their personal documents. The users can sketch the layout of the document using a calligraphic interface and the system will present them with those that match that sketch. Documents are processed to extract its relevant features, blocks are segmented and classified according to their contents and a description of the layout is created. We described the document in two different ways: grid-based and semantic-based. The interface allows the user to choose from each of this search methods and includes also a complementary query by example mechanism.*

**Key Words**
*Document Retrieval, Sketches, Calligraphic Interfaces, Personal Information Management*

## 1. INTRODUTION

Nowadays, we sometimes find it hard to find that document that we know to be somewhere in our hard drive. Usually, this is not a trifling task, especially for older documents. It is often the case where, when looking for a document, we can't remember its name, location, or any other kind of attribute usually used in a common search task. However, occasionally we can recall its appearance, how the first page looked like (it was written on two columns, it had a picture downloaded from the Internet at the top of its rightmost column and an Excel table at the bottom of the document with the results of the work, etc.). Even if most of the documents in our possession are similar, filtering them by appearance provides a simple yet effective of greatly narrowing down the possible choices when looking for a particular one.

The visual memory of the user plays a crucial role in the recognition of objects and also documents. Human beings can easily remember and perceive images rather than words. So we use this fact to construct a new kind of document retrieval model.

One of the best and easier ways to describe a visual representation of something, in this case of a document, is to sketch it. Calligraphic interfaces are much enjoyed by users because they feel like they can express their ideas and intentions easily and without too many constraints. Also, touch-screen-based devices, such as PDAs, Tablet PCs and even Smartphones, are becoming more and more available to the general public.

This paper presents a system capable of retrieving documents similar to a sketch drawn by the user in a calligraphic interface. In order to accomplish this we had to process the documents so they could be effectively indexed. This involved transforming the first page of the document to an image. This image was then processed and segmented into blocks classified according to their content: text, image, graphic, table, horizontal line and vertical line.

From each block, relevant features were extracted. We used those features to create two distinct indexation methods: a grid method, based on the spatial distribution of the blocks, and a semantic method, that makes use of more high level characteristics of the blocks. This allows users to specify queries using high-level semantic descriptions of their document appearances ("a two-column document with an image on the top of the right column…"), rather than simply comparing sets of pixels.

Finally, we created a retrieval interface, capable of query by sketch and query by example. It allows the use of both indexation methods. It was initially conceived with a system of rectangles and colors to describe the document to search for. Then CALI [Fonseca00] library was incorporated to make the recognition of free-form user sketches, according to some descriptive language [Albuquerque00]. To make this interface a reality, we had to study how users describe the layout of their documents, how many details they can recall and how accurately they are able to draw them.

Throughout the rest of this document we will describe every step of the retrieval process and the design and development of the calligraphic interface. Section 2 references some previous projects in this area and their ac-

complishments. In Section 3 we describe the algorithms used to process the document image and extract its relevant features. Section 4 details the document description methods implemented and the indexation technique. Section 5 explains the design and functionalities of the interface. Section 6 summarizes the work done and possible future work in the area.

## 2. RELATED WORK

In the last decades, as an attempt to help users retrieve their documents based on their appearance, some systems were studied and developed. Some address only part of the problem (document processing and segmentation, image indexation, etc.) while others tried to produce complete document retrieval solutions.

This subject is addressed essentially in Content-Based Image Retrieval (CBIR) systems, more specifically the Query by Sketch category. These systems exploit information gathered from the contents of images. A certain number of methodologies, techniques and tools related with image processing were studied with the aim of identifying and comparing features useful to the development of classification and retrieval systems based on the (almost) automatic interpretation of image contents.

QBIC [Faloutsos94] was the first system of this type, making use of global features like area, circularity and eccentricity in shape comparison. Query by Example [Kato92] is a complementary method to query by sketch, taking advantage of some image already in the database chosen by the user similar to the result intended.

Most of these solutions make explicit reference to images. More recently it has emerged a larger awareness of the problems of document retrieval, whether through key-terms or from the document image obtained from its layout. Our approach follows a series of methods designated by SBIR (Sketch-Based Image Retrieval) since it starts with a sketch of an image to try to recover it. The objective is to expand these sorts of algorithms so that a sketch may retrieve not only images but also documents, calling then SBDR (Sketch-Based Document Retrieval).

One system that already tries to employ such a method is WISDOM++ [Berardi04]. They present an approach for semantic structure extraction in document images. They first extract layout structures and then use textual content to automatically label these structures, applying machine learning techniques to support the process.

## 3. THE SKETCH-A-DOC INTERFACE

We have created two different interfaces for our system. The first can be seen in Fig. 1. Its main area is a canvas, where users can draw a sketch representing the layout of the documents they want to retrieve as a series of rectangles. To the left of the canvas is a tool palette where the type of block to be drawn can be chosen by the user. Each block will have a color, predefined according to its type, as follows:

- Red: text block;
- Green: image;

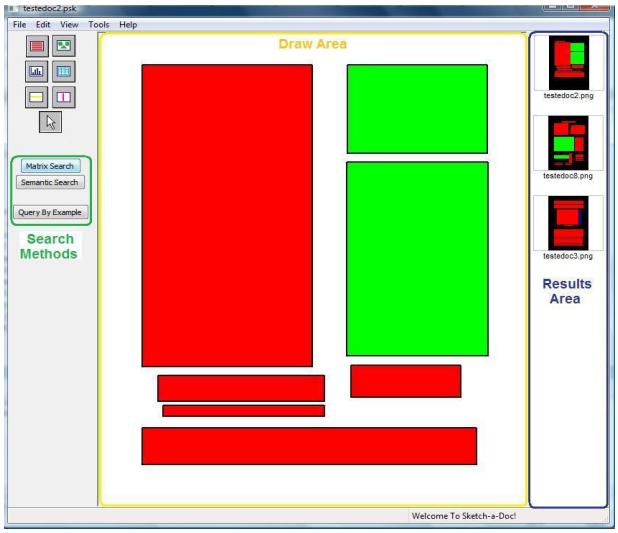


**Fig. 1 – Rectangle-based Interface**

- Blue: graphic;
- Yellow: horizontal line;
- Magenta: vertical line;
- Cyan: table.

The user, of course, needs only select the type of block, and not the color. This method has the disadvantage of being a little restrictive than free-form hand-drawn sketches, but it can be more precise and less error-prone.

Also, a selection tool allows the users to select already drawn blocks and delete or move them. Three buttons allow the user to choose the type of search to be performed grid- or semantic-based (described below), and to provide some similar document to start the search instead of drawing a sketch (query by example). To the right of the canvas, the interface also includes an area to display the best ranked results. There are two preview modes to show the results: Normal: shows the thumbnails with the result documents; Sketch: shows the thumbnails with sketch representation (obtained with the rectangle and colors system) of the result documents. Double-clicking a thumbnail will open the respective document. The users can also save their sketches and after a while load them to perform a new search, editing or not the original sketch.

An alternative to the rectangle and color system is to use the CALI recognizer library [Fonseca00], publicly available and with verified high recognition rate, coupled with a visual grammar capable of identifying any of those six elements (Fig. 2). This grammar [Albuquerque00] was compiled with the help of studies about the most typical ways users draw a set of shapes and what those shapes represent. In this manner users can draw as they are used to and the system will recognize the layout they meant to sketch:

- Text block -> {WavyLine}
- Image -> {Rectangle Line}
  *If* Contains (Rectangle, Line)
  *And* Oblique (Line)

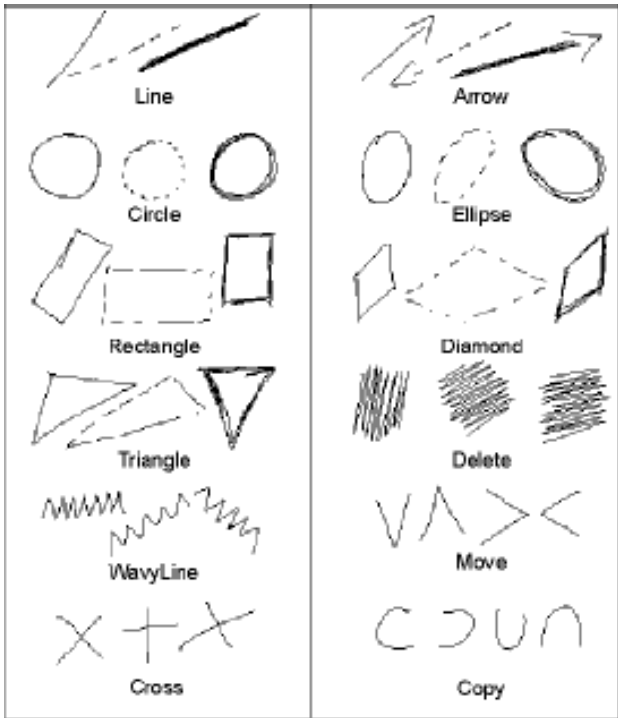


**Fig. 2 – Grammar Shapes**

- Graphic -> {Rectangle Triangle}
  *If* Contains (Rectangle, Triangle)
- Table -> {Rectangle, Cross}
  *If* Contains (Rectangle, Cross)
- Hor. Line -> {Line}
  *If* Horizontal (Line)
- Vert. Line -> {Line}
  *If* Vertical (Line)

In this case the rectangle icons disappear since only a scribble, besides the selection tool, is needed.

To perform a search using a similar document as a starting point, the user can click the query by example option. An open dialog is shown, from which the sample document is chosen. This document is processed as if were to be indexed, all of its features are extracted and used to find documents that resemble this one. It is also possible to select a document from the results of a query and ask for others similar to that document, thus helping the user to iteratively refine the search process.

### 3.1 Scoring

After a query is entered, a scoring algorithm is applied according to the index method chosen to search in – the user can choose either one of the two indexation methods, grid- or semantic-based. In the first, the document's page is divided into several grid cells, and the type of element(s) in the cells is recorded. In the second, a high-level description of the content blocks, their size, and relative positions is stored.

If the grid-based method was chosen, for each block given by the query, the system looks for every document that has the same type of block in the same grid cells. For each match the document is awarded one point. After all blocks have been processed the documents with the highest scores are presented to the user.

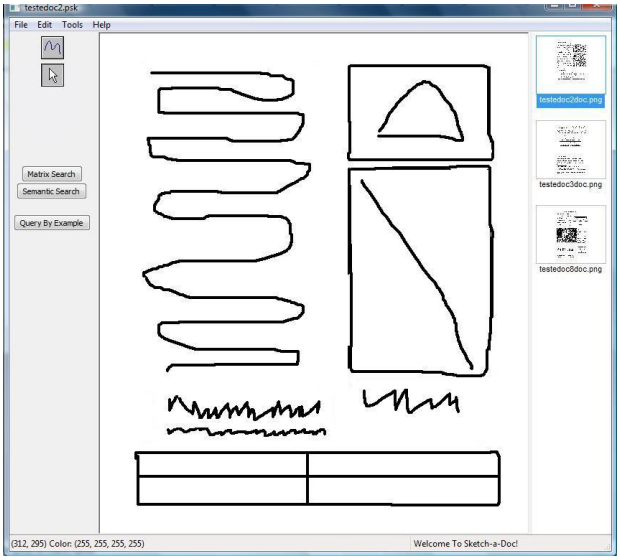


**Fig. 3 – Sketch-based Interface**

The scoring in the second method is a bit more complex. For every feature, it seeks the entries that match the query and then the ones in the immediate feature neighborhood. For instance, consider a block with the following features: size="small", type="text", xOrigin="centerLeft", yOrigin="3/10". The algorithm would look for blocks with-size="verySmall"/"small"/"medium", type="text" / "image"/"graphic"/"table"/ vLine"/ "hLine", xOrigin="left" / "centerLeft"/"centerRight", yOrigin="2/10"/"3/10"/ "4/10", varying one feature at a time. It bestows ten points to a document that exactly matches the query in the same position, two points for each matched feature in the feature neighborhood of the query, and one point to every block of the same type in the feature neighborhood, even if no other feature is matched. This accounts for possible errors remembering the block, and allows documents that only partially match the query to be nevertheless found.

## 4. DOCUMENT ANALYSIS

Underlying the interface we've just described is a representation of the appearance of all indexed documents. To obtain it, the main challenge was to get a high-level description of the documents according to their layout.

To extract the required features from a document we first have to transform it into a format more amenable to processing, abstracting from the underlying file format (pdf, etc.). We grab the first page of the document and turn it into to an image, since there are many techniques capable of image processing and block segmentation, to identify relevant areas in the images.

### 4.1 Image Pre-Processing

Using a module developed in the personal document retrieval project Quill [Gonçalves08] to extract an image from most document types, we produce images of the cover pages of documents. As our objective is to produce a high-level description of document appearances, we then process those images. The first step is to convert the

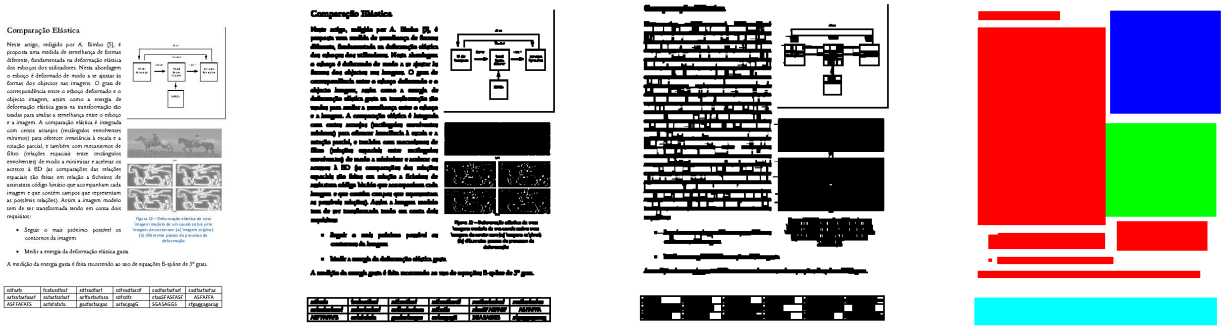

**Fig. 4 – Document Analysis Process. Left to right: the original document; the document after thresholding and applying the erosion filter; the result of the RSLA; the final result, with all blocks colored according to their type.**

image to black and white pixels only, using a basic threshold algorithm (Fig.4).

This led to an image reflecting the overall structure of the document page, but also resulted in several "impurities", stranded pixels created by the thresholding process. When trying to identify relevant content blocks in the image, one block could be, mistakenly, included in an adjacent one or its type mistaken, just because there was some small group of pixels, "noise" data, in some inappropriate location in the document image. To solve this problem, and since we are only concerned with the overall features of the page, we decided to apply an erosion filter (Fig. 5) with a simple cross flat structuring element to minimize those spurious pixels and improve the efficiency the next processing step: block segmentation.

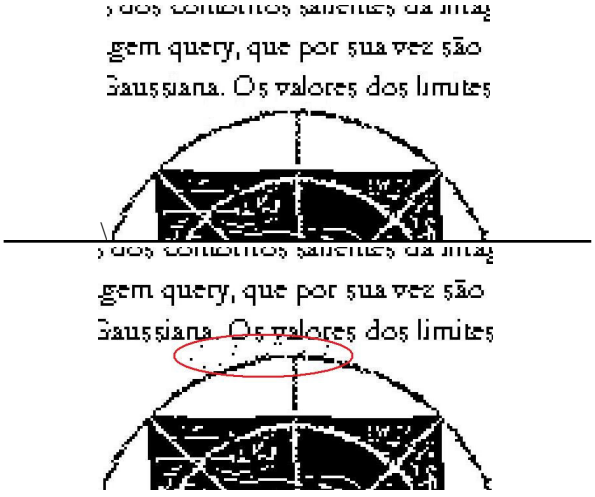


**Fig. 5 – Example of erosion**

## 4.2 Block Segmentation

Many studies have already been made about image and document block segmentation and classification. Some of them are strict rule-based approaches and others more dynamic, resorting to machine learning techniques. For simplicity and effectiveness' sakes, we employed the RLSA (Run-Length Smoothing Algorithm), an adaptive rule-based algorithm. Instead of the basic version, an improved two-step block segmentation version [Shih96] was adopted. This algorithm is able to detect content blocks in a document image, while at the same time classifying those blocks according to their type. To detect content blocks, the RLSA algorithm starts by finding content lines (uninterrupted horizontal sequences of pixels), and then grouping those closer than a predefined threshold. However this block classification method does not account for tables, which were one of the block types we needed to identify. It only classified in according to five categories: text, image, graphic, horizontal line and vertical.

To adapt the algorithm to our needs, its rules and parameters were modified, although the same basic features are used for block classification:

- Height of each block - $H$;
- Ratio of width to height (aspect ratio) - $R$;
- Density of black pixels in a block - $D$;
- Horizontal transitions of white-to-black pixels per unit width – $TH_x$;
- Vertical transitions of white-to-black pixels per unit width – $TV_x$;
- Horizontal transitions of white-to-black pixels per unit height – $TH_y$.

The width to height ratio is used to detect horizontal and vertical lines. $TH_x$ and $TV_x$ are used for table and text discrimination. $TH_y$ is also used in table recognition. Both density $D$ and height $H$ allow the discovery of images and graphics, according to some threshold. This part was done with caution because it was an innovation to the RLSA classification algorithm. Unlike the original method we decided to identify tables, images and graphics first and let the text blocks be the "otherwise" rule, as blocks of other types are easier to identify than text blocks, that can take many different shapes. Images are easily classified since they usually have more "ink" density and are larger than most blocks. Graphics frequently occupy an identical space to images; the difference is that their density is much lower. To identify tables, we created a new set of rules:

- $H <= 30$ and $D < 0.9$ and $0.15 < TH_x < 1.7$ and $1.7 < TV_x < 4.8$ and $TH_y > 5.0$
- $30 < H <= 60$ $D < 0.8$ and $1.8 < TH_x < 3.8$ and $2.4 < TV_x < 4.8$
- $60 < H <= 90$ and $0.25 < D < 0.85$ and $2.2 < TH_x < 4.8$ and $3.0 < TV_x > 6.2$ and $TH_y < 12.5$
- $H > 90$ and $0.15 < D < 0.6$ and $TH_x > 3.7$ and $6.0 < TV_x > 20.0$ and $TH_y > 10.0$

As can be seen, the block detection and classification algorithm is sensitive to the values of several parameters. To choose the most appropriate values for these parameters, mentioned above, we performed an experiment in which several values were tried and the quality of the results evaluated. We used a set of 46 documents, representative of different block types, sizes and combinations commonly found in personal documents. Text blocks were formatted with character sizes varying from 8 to 32 points mixed with commonly used fonts with and without serifs, like Times New Roman, Arial, Garamond, Bookman Old Style and Comic Sans MS. For some of these fonts the character sizes are almost the same and so the results are also equal, but for others the spacing, thickness and size are slightly different. We did this so that the estimated parameter values would adequately encompass a wide range of documents. Tables with a wide variety of number of columns and number of rows and also more or less filled with text and varying sizes were also used to allow the estimation of the parameters necessary to identify them.

We were able to infer the parameter values that give us, on average, the best results. The process was a bit arduous since the original algorithm didn't take on account tables, which are easily mistaken for text blocks or graphics, depending on whether they are more or less filled with text. Overall, we found that our algorithm is able to correctly identify and classify 87.5% of all blocks.

## 5. DOCUMENT DESCRIPTION

Instead of following only one approach we decided to describe the documents in two ways: grid-based and semantic-based.

### 5.1 Grid Description

This is the most straightforward approach; it is based in spatial organization features only.

We divide the document layout according to a pre-established 4x10 grid. We chose to partition the document in 40 units (4 columns and 10 rows) because after some analysis we concluded that almost no document was formatted in more than 3 columns and they were not split vertically with more than one type of block. The number of rows was chosen empirically, and 10 was the number held expressive enough.

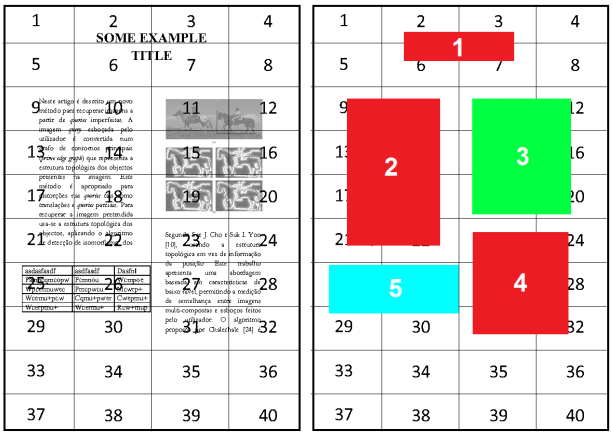


**Fig. 6 – Grid of a document layout**

For instance, in Fig.6 we have a document with the grid overlaid on it. According to this kind of description we would get something like: There is a text block in cells [2,3,6,7], another in [9,10,13,14,17,18,21,22] and another one at [23, 24, 27, 28, 31, 32]; there is an image at [11, 12, 15,16,19,20]; and there is also a table located at cells [25,26,29,30]. This description is very simplistic, but good enough to portray the layout of the first page of a document. In this case there aren't any intersections on the same grid cell but if there were the algorithm would use the same cell number on the description of different blocks. This provides a description of the page in which the type of element present in each area of it is known. We can then match this description to that of a sketch drawn by the user.

### 5.2 Semantic Description

This approach is more high-level than the one described in section 4.1. It does not depend on a pre-determined grid, as it is based mainly on parameters such as the types, proximity and relative sizes of blocks. This description, based on semantically relevant entities and concepts, can be used as the basis for a description of documents in simple English, making it self-explanatory. A user who reads such a description would have no trouble understanding it. This allows applications in which a high-level or natural language-based description provided by the user can be compared to the indexed documents.

The description for the topmost part of the document in Fig.4 would look like this (the number between brackets represents the id of the block): [2] – element of type *text* with *32,81* % of page width and *35,62* % of page height, with origin at *left* and *3/10* of page height. It's below [1], on top of [5] and left of [3, 4]; [3] – element of type *image* with *32,43* % of page width and *26,46* % of page height, with origin at *centerRight* and *3/10* of page height. It's below [1], on top of [4], and right of [2].

Since every block is described this way, the neighborhoods are also well identified with relevant features like block size and type.

### 5.3 Indexing

After describing the documents, all we had to do was transcribe the document descriptions to some simple, easy

and helpful format. XML seemed to be the adequate choice, even for the reason that it is so widely spread.

At description time. each document results in one XML file containing all high level features obtained from the processing stage. Then, by the time the indexation function is called to process all documents intended, two other XML files are created, one for each type of document description – grid and semantic – containing information about all documents in the index.

Both files depict a tree where leafs are the document file locations. The grid description XML file follows the simple tree: *grid cell number -> block type –> file*. The semantic description XML file follows a more sophisticated tree involving the content block sizes, types and coordinates. This index can be used to filter documents according to the possible values of all different features. Each entry references the block size ("verySmall", "small", "medium", "big", "veryBig"), its type, its origin in the x coordinate ("left", "centerLeft", "centerRight", "right") and its origin in the y coordinate ("1/10", "2/10", … , "9/10", "1"). It also contains references to the documents classified in the feature neighborhood, easily allowing the navigation in the feature space to look for similar, but not exactly equal, documents.

Since it is not efficient to parse the XML files every time a query is introduced, this is done only once, at the time of the first query, and its elements are stored in a nested dictionary, that provides an easy and effective way to find documents that match the query. So, for the semantic approach, there will be something like: *semanticIndex [size][type][xOrigin][yOrigin]*; where the last dictionary also contains four more dictionaries for each of the block neighborhood directions: top, bottom, left and right. While for the grid approach the structure is more straightforward: *gridIndex[type][cellNumber]*.

Providing the right indices the results are obtained right away.

## 6. CONCLUSIONS

The users often resort to their visual memories when describing documents. However, modern operating systems and applications do not allow them to use those memories to retrieve their documents. We presented a document retrieval system in which a calligraphic interface allows the users to draw sketches of document appearances in order to retrieve them.

To do so, we had to process the images representing the first pages of documents. This required some modifications to the RLSA algorithm, and the tuning of its parameters to include a new type of block: tables. We also developed two separate document descriptions. One based simply on block spatial distribution – grid-based – and the other, more complex, based not only on attributes like size and location but also block adjacency, giving a semantic-based description of documents that can be used in high-level applications. The interface we designed allows the use of both indexing methods and query-by-example. User tests will allow us to determine which method provides better results. If it is possible to identify situations or types of documents in which a method outperforms the other, it would be interesting to let the system automatically decide the best method to adopt in each case.

One of the next steps in this research is to improve the grid description to include the percentage or ratio of how much of the grid cell is occupied by the block. Also, there are some aspects to be explored about the semantic index. The retrieval process will be adjusted to consider more neighborhood features and adapt them to the scoring algorithm.

As for the interface, it will undergo of usability tests, resulting in possible modifications. We would as well like to include a relevance feedback facet to improve the quality of the results and another search method – for each feature, one would choose from a predefined set of values, as if constructing a semantic description of the document. Also, we intend to test the interface and the retrieval process as a whole, including each of the methods (grid and semantic), to see which one presents the best performance in terms of retrieval success.

## 7. REFERENCES


[Albuquerque00] Albuquerque, Maria. Fonseca, Manuel J. Jorge, Joaquim A. Visual Languages for Sketching Documents*, IEEE Symposium on Visual Languages, IEEE Computer Science Press*, 09/2000

[Berardi04] Berardi, M. Lapi, M. Malerba, D. An integrated approach for automatic semantic structure extraction in document images, *In S. Marinai & A. Dengel (Eds.), Document Analysis Systems VI. 6th International Workshop, DAS 2004, Lecture Notes in Computer Science, Vol. 3163*, 179-190, 2004

[Faloutsos94] Faloutsos, C. Equitz, W. Flickner, M. Niblack, W. Petkovic, D. Barber, R. Efficient and Effective Querying by Image Content, *In Journal of Intelligent Information Systems,* 3:231-262, 1994

[Fonseca00] Fonseca, Manuel J. Jorge, Joaquim A. CALI: A Software Library for Calligraphic Interfaces. *Actas do Nono Encontro Português de Computação Gráfica*, Marinha Grande, Portugal, 02/2000

[Gonçalves08] Daniel Gonçalves, Joaquim A. Jorge, In Search of Personal Information: Narrative-Based Interfaces. *In Proceedings International Conference on Intelligent User Interfaces (IUI'2008)*, 13-16 January, Maspalomas, Canary Islands, Spain. 2008

[Kato92] Kato, T. Kurita, T. Otsu, N. Hirata, K. A Sketch Retrieval Method for Full Color Image Database, *In Proc. of the 11th Intl. Conf. On Pattern Recognition,* pages 530-533, The Netherlands,Aug. 1992.

[Shih96] Shih, Frank Y. Chen, Shy-Shyan. Adaptive document block segmentation and classification, *Systems, Man, and Cybernetics, Part B, IEEE Transactions on, Volume: 26, Issue: 5*, 797-802, 10/1996