An Agent Based Algorithm for Document Analysis (ABADA)

K. Jambi, M. Saleh, H. Barhamtooshi, F. Essa and A. Ezz
Computer Science Department
King Abdul Aziz University
Jeddah, Saudi Arabia

ABSTRACT The aim of document layout analysis is to extract the geometric structure for the document image. The introduced system is deigned to work with a variety of documents, without prior knowledge about the nature of the document. This algorithm mainly depends on dividing the document into strips or runs, these runs ease the document handling and enable the ability of handling a part of the document and then handling the other part. This is too important when handling memory, and when downloading documents from networks. It can be calssified as a hybrid tecnique of top-down, and bottom-up, to overcome the disadvantages in each staregy. Agent tecnology is used to ease the operation of the system through the network. Expermental results reveal the proposed approach is effective.

1. Introduction

Documents can be viewed as paper-based documents, or digitized documents. Paper-based documents are processed for information extraction by human which make these documents a labor force consuming. Moreover, These documents need much space for storing. On the other hand, digitized documents can be viewed as electronic images which are machine searchable. This ability makes electronic documents preferable for its fast search capabilities as well as less storage media.

Layout analysis is the extraction of geometric structure from the document image [1]. The given document image is generally in binary format. It is segmented into several objects due to location, contents, homogenity, and other attributes. Document layout analysis plays an important role in automated documet processing environment as it classifies different parts of the document before deciding which later subsystems to use[2]. Structural layout analysis can be achieved using top-down or bottom-up techniques [3].

In this work, a document image processing system is developed. This system is based on an agent technology. The document image analysis system is used to analyze and classify the document contents in details with a high accuracy. Agents are used to go through networks in parrallel to reduce the time and to collaborate the documents to get specific predefined documents. This paper deals with two different concepts. The first concept covers the intelligent agent technology, and the second one deals with the process of document analysis. In the following two sections, we will discuss these two concepts.

1.1 Document Analysis

The concept of Document Image Processing is used for performing analysis and understanding several types of documents. A document is a set of related pages to express a subject. A page is a set of objects (range from text, graphics, or images) that take part in describing the mentioned subject. The page is segmented into equi height parts called runs.

An object is a set of finite blocks that have the same characteristics, and it can take any shape, and it can contain other objects, differ in characteristics and hence in type. A block is a set of some related neighbored pixels and it is the smallest unit to express the object, and the block is the part represents the object in a single run. The white color is the default background color for any determined color. The black color is the default foreground color for any color except white [1,2]. There are many methods of document layout analysis. Most of these methods must be skew free, and/or a priori (perquisite) layout models are required [4-6].

1.2 The Concept of the Agent

An agent is an encapsulted computer system situated in some environment and capable of a flexiable, autonomous action in their environment in order to meet its design objectives [7]. Agents are designed to fulfill some specific roles, and to achieve some particular objectives. Agents receive inputs related to the states of their environment through sensors and they act on the environment through effectors. They have control over their internal state and over their own behavior. Also they are capables of exhibiting flexible problem-solving behavior in pursuit of their design objectives. These means that the agent are autonomous [8].

2. The Research Objective

This paper introduces an algorithm for a document layout analysis based on agent technology. The basic idea for document anlaysis is introduced in [9], This algorithm mainly depends on dividing the document into strips or runs, these runs ease the document handling and enable the ability of handling a part of the document and then handling the other part. This is too important when handling memory, and when downloading documents from networks. These runs are divided into blocks by verifying the horizontal and vertical thresholds. These blocks are merged or split to constitute the document objects.

The agent used in this paper is Mobil Agent. Mobile Agents are computational software processes capable of roaming wide area networks (WANs) such as the WWW, interacting with foreign hosts, gathering information on behalf of its owner and coming back home having performed the duties set by its user [8,10]. A Mobile Agent is specialized in that in addition to being an independent program executing on behalf of a network user, it can travel to multiple locations in the network. As it travels, it performs work on behalf of the user, such as collecting information or delivering requests [7,8,10].

3. The Document Analysis System (DAS)

The DAS deals with documents that are skew independent. Also, wth DAS there is no priori knowledge about documents under consideration. The DAS mainly depends on dividing the document into strips or runs, these runs enable to handle the document easily and also give the system the ability of handling modules of the document separately. This feature is very important for avoiding size limitation of the memory as well as memory management with resepect to downloading huge documents from networks. These runs are divided into blocks by verifying some horizontal and vertical thresholds. These blocks are then merged or splited to constitute the document objects. Also, the DAS uses foreground for crossing count as an indicator for object's homogeneity measurement, classification and identification [9]. The different steps for DAS are disscussed in the following sub sections

3.1 Document Preparation and Run Determination

This is the preparing step. It includes: image capture if not; image binarization while loading, noise reduction, and runs determination. Optical scanning usually capture the document image data at a resolution of 300 dpi, but the used algorithm needs only 50 to 75 dpi reducing the processing time to 1/36 to 1/16 of the processing time for the high resolution document. So, in loading the document it is read at its high resolution but processed at the low resolution.

The extraction of binary character/graphics images from gray scale document images with background pictures, highlight, shadows, smeared is a critical image processing operation [11]. We handle the problem with simple issue in Java language. Gray scale and colored images can be treated by tuning the threshold value between the foreground and the background by the user.

Noise is the result of unclear document, imperfect reproduction, or digitization. Such noise does not disturb human readers but complicates automated analysis [12]. Krishnamoorthy et al, [12] suggested making the grammars sufficiently robust to ignore such noise. Or removing all specks smaller than a given size in a preliminary pass using connected components algorithm. But handling noise in our algorithm is done automatically by loading the image at lower resolution.

To split the document into a number of runs (strips), the following criteria is used:

$$R_n = D_h / R_h$$

Where; R_n represents the number of runs, D_h is the document height, and R_h is the run height. Now, the processed document is splitted into runs and it is ready for manipulation to get blocks in each run.

3.2 Blocks Determination

In this step, a single run is divided into unrelated blocks. If a run contains more than one block, this indicates that these blocks belong to different objects. If more than one block is contained in one object, and they belong to one run, this means that they aren't related in that run and at least the horizontal background gap should be used to separate them. Figure (1) shows some runs , with different blocks

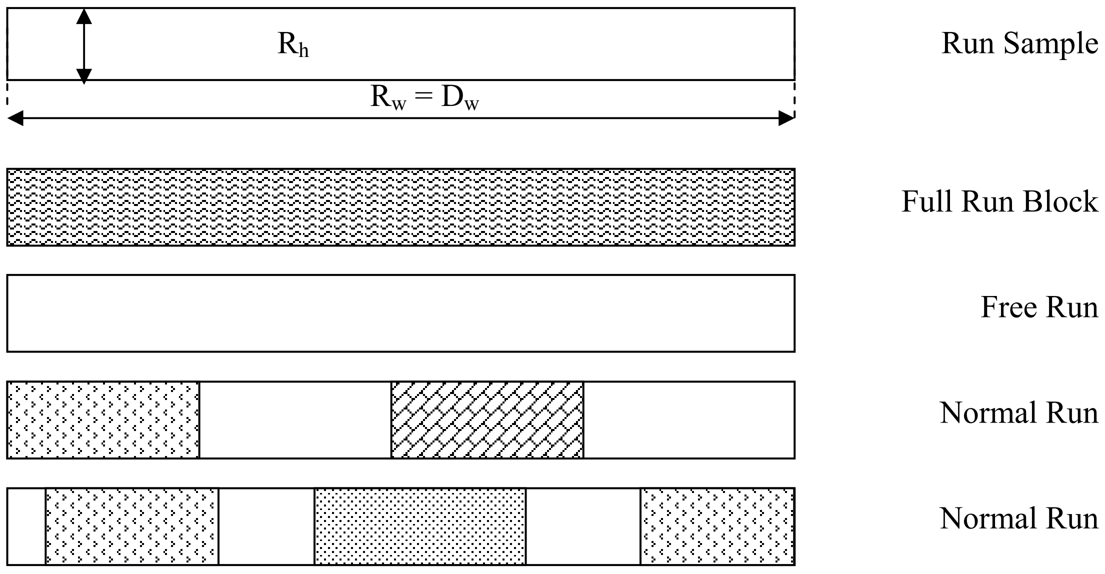


Fig. (1): Block Determination Based on Horizontal Threshold H_t

After determining the block coordinates (x1,y1) , (x2,y2). It is required to classify the block as a text or a halftone block. In fact, homogeneity factor for the block_n is H_n is measured by:

$$H_n = \frac{F_n}{Cc_n}$$

Where; F_n is the number of foreground pixels in the block$_n$.,Cc_n is the number of variations (transitions) from foreground to background.
The number of foreground pixels F_n is determined by:
Where:

$$F_n = \sum_{j=y_1}^{y_2} \sum_{i=x_1}^{x_2} p(i,j)$$

$p(i, j)$ is the foreground pixel in the domain of tested block.
y_1 is the starting vertical coordinate of the block$_n$
y_2 is the ending vertical coordinate of the block$_n$
x_1 is the starting horizontal coordinate of the block$_n$
x_2 is the ending horizontal coordinate of the block$_n$

Crossing count is the number of times the pixel value is turned from background to foreground background along the whole block. Consequently, the crossing count can be measured horizontally according to the following formula:

Where $\overline{p(i,j)}$ is a background pixel, and $p(i+1,j)$ is a foreground pixel.

$$Cc_n = \sum_{j=y1}^{y2} \sum_{i=x1}^{x2} \overline{p(i,j)}p(i+1,j)$$

3-3 Block Classification

Text/nontext objects(blocks) tend to group (cluster) in space with respect to some features, a threshold or a discrimination function is selected for separation. Wong, el. [13] used a two dimensional plane consisting of mean value of the block height verses run length of the block mean black pixel to classify document blocks into text, non-text, horizontal lines, and vertical lines. Fisher el. [14] used a rule based classification technique where some features such as height, aspect ratio, density, perimeter, and perimeter/width ratio. Wang, and Srihari, used a method to create the black-white pair run-length matrix and black-white-black combination run-length matrix to derive three features: short run, emphasis, long run, emphasis, and extra long run, emphasis for clustering [14].

The object in any orientation has nearly the same number of foreground pixels, all the objects we are working with may or may not be regular or skewed. So, we must check a right method for detecting the features of the line. A good feature for object classification is the foreground/crossing count ratio [9,15].

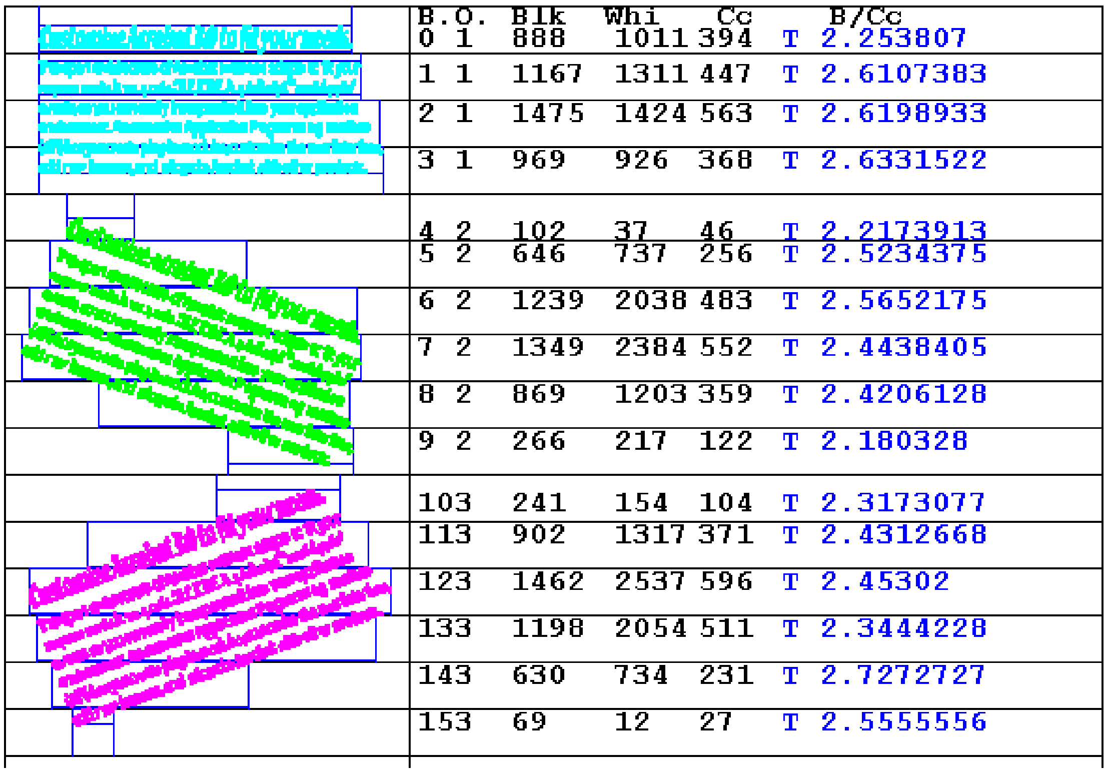
For example, Figure (2) shows a text object in normal and skewed form, and on the right is the characteristics of each block. These results indicate that the foreground to crossing count ratio is nearly constant for the same object horizontally aligned or skewed.

The classification is done according to the following rules:

- If the block F/Cc < 10 and Background > 0
  Then the block is text.
- If the block F/Cc < 10 and Background < 0
  Then the block is a horizontal line.
- If the block F/Cc > 10 and Background < 0
  Then the block is classified as half-tone until merging step.

- If the block to be merged with a half-tone block it will be half-tone
- Else it will be a vertical or sloped line.

- Else If the block F/Cc > 10
  Then the block is classified as half-tone



B.O.	Blk	Whi	Cc		B/Cc
0 1	888	1011	394	T	2.253807
1 1	1167	1311	447	T	2.6107383
2 1	1475	1424	563	T	2.6198933
3 1	969	926	368	T	2.6331522
4 2	102	37	46	T	2.2173913
5 2	646	737	256	T	2.5234375
6 2	1239	2038	483	T	2.5652175
7 2	1349	2384	552	T	2.4438405
8 2	869	1203	359	T	2.4206128
9 2	266	217	122	T	2.180328
103	241	154	104	T	2.3173077
113	902	1317	371	T	2.4312668
123	1462	2537	596	T	2.45302
133	1198	2054	511	T	2.3444228
143	630	734	231	T	2.7272727
153	69	12	27	T	2.5555556

Fig. (2): Three alignments of the same object (Text Object)

3.4 Blocks Concatenation

After the run ends, and before going to the next run, linking process is done between the current run blocks and the previous run to constitute a single type objects. The linking process is done according to the following sub algorithm:

Sub-Algorithm: Check Object( )

```
{   At start let the starting vertical coordinate of the tested block by y1,
    and the ending coordinate by y2.
    The upper block is i, and the lower is j.
    If (y1[j] - y2[i] < Vt)
    Then the blocks i and j may be merged go to Final step.
    If (y1[j] - y2[i] >= Vt)
    Then the two blocks belong to different objects, and this needs the next step.
    If one of the two blocks has Block Object $B_o$ = -1
    Then index the block with $O_i$ and Increment $O_i$
    If both of the two blocks has Block Object $B_o$ = -1
    Then index the first block with $O_i$ and Increment $O_i$
    And index the second block with $O_i$ and Increment $O_i$
    If the two blocks are inhomogeneous, index them with two different index
    as the previous step.
    Finally, the two blocks  i,j are merged
    If
            xi1 <= xj1 <= xi2  OR
            xi1 <= xj2 <= xi2  OR
            xj1 <= xi1 <= xj2  OR
            xj1 <= xi1 <= xj2
    }
```

3.5 Document Representation

After getting the document objects, and identifying them, the final stage is to represent the analyzed document in a simple form suitable for other modules of document image processing. This is done by producing a file or an array that hold the analyzed blocks data such that in Table (1).

Table (1) : Representation of Analyzed Document

Block index	Object index	Block type	Block Features			Block Dimensions			
			Black	White	Cross Count	x_1	x_2	y_1	y_2
0	1	Text	1731	1578	598	61	207	13	72
1	1	Text	2582	3697	1170	49	199	73	145
2	2	Image	6223	1263	395	29	188	146	218
3	2	Image	8288	-32	165	16	145	219	291
4	2	Image	4252	29	100	9	132	292	339

4. System Implementation

In this work, we use an agent framework called Concordia [10]. Concordia is a software framework for developing, running and administrating mobile agents. Concordia mobile agents are Java programs which travel in the network to perform their function. According to figure (3), there are three types of agents used in the system: user agent, boss agent, and the agent server.

The User Agent (the requestor) for the operation. Its job is to determine exactly what users are looking for, what they want, if they have any preferences with regard to the information needed.

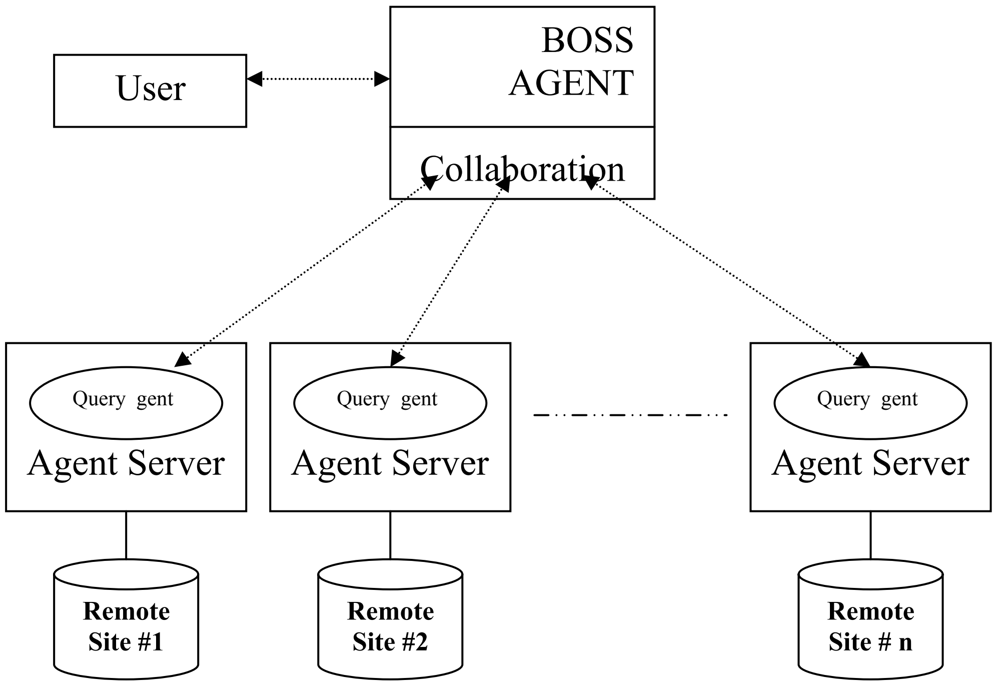


Fig. (3): Mobile Agent for Document Analysis

The Boss Agent: plays a role of intermediate agents. The job of the boss agent is to receive the user request of infornation and determine the specified classification of this information. Also, to seacrh the documents database to find the related document(s) and its location(s) according to the specified classification. Finally, to receive the results of the doucments agents and save the classified data in the documents database. The classification of the document data are used to get an accurate document(s) according to user seraching requirements. And to send the required results to the calling (user) agent.
The scenario of document analysis can be summarized in the following steps:

1- Reading the document image
2- Running the Run Block Algorithm
3- Saving the results in a vector or in a Database
4- Reporting the results by reading the vector or the database

4.1 The DocAnalysis Agent implementation

The DocAnalysis class is the main class of the system. It is implemented as Concordia Agent and includes the following methods:

- getPixels() to Reading the document image into an array of pixels
- process_horizontally() to process each run (strip)
- process_blocks() to put block information
- block_classify() to classify the block

`its Results`, a vector to hold the results

```
// the main class for document analysis
public class DocAnalysis extends Agent
{
        // the vector to hold the results
          Vector itsResults;
          public DocAnalysis ()
          {
                    itsResults = new Vector();
          }
        //the method to print the results
          public void reportResults()
          {
                    Enumeration enum = itsResults.elements();
                    while (enum.hasMoreElements())
                     {
                              QueryResult result = (QueryResult)enum.nextElement();
                               // printing the results or saving it to file
                     }
          }
        public void run_block_algorithm()
        {
                GetPixels();
                for (int j=0;j< regular_height ;j+=run_height)
                {
                        process_horizontally();
                        process_blocks();
                        block_classify();
                }
        }
        public void GetPixels()
        {  //Implementation code }
        public void process_horizontally()
        {  //Implementation code }
        public void process_blocks ()
        { //Implementation code }
        public void block_classify ()
        { //Implementation code }
}
```

4.2 The QureyResult class

The DocAnalysis class contains the document analysis data that are gathered by the agents. The agent firstly travels to the desired location, opens the document, analyzes it, and finally puts the analysis data in this class. This data can be used by other methods in other locations. The QueryResult class is implemented as serial object.

```
class QueryResult
{// to store the block information as in table (1) }
```

4.3 The Agent Launching

The bootstrap command is used to launch Agents [10]. Launching an Agent refers to the process of constructing an Agent, setting up its travel Itinerary and beginning its travels through the network. Before calling a launcher, we build an Itinerary for the Agent and populate the Itinerary with Destinations using the following method:

```
itinerary.addDestination(new Destination("server", "method to call "));
```

1- At start the Boss Agent (main machine) needs to run the DocAnalysis on the remote site#1
2- The agent travels to machine called "Site#1" and apply " run_block_algorithm()",
3- Then the agent returns to Boss Agent and apply "ReportResults " method , to display the image document analysis results.

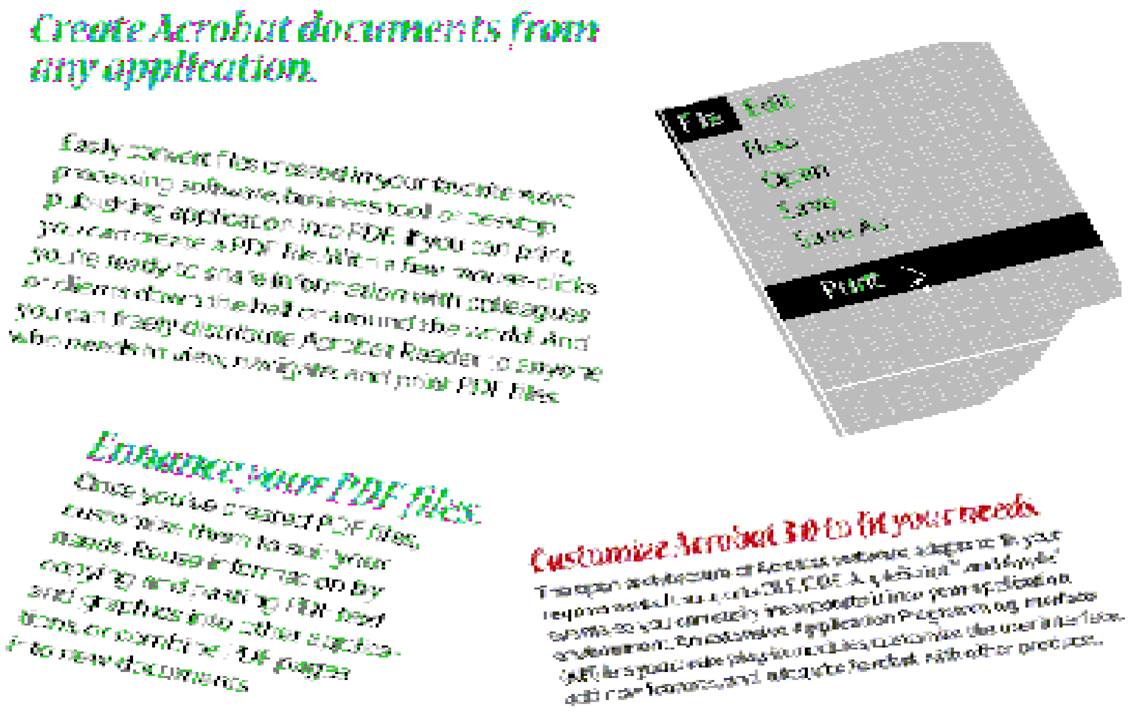
The code of this launch is shown in the following lines.

```
public class TestLaunch {
 public static void main(String args[ ]) {
   DocAnalysis agent = new DocAnalysis ();
   Itinerary itinerary = new Itinerary();
   itinerary.addDestination(new Destination("Site#1,” run_block_algorithm”);
itinerary.addDestination(new Destination("boss", "reportResults"));
       String code = "file:C:\MyAgent";
       String relatedClasses[ ] = {"QueryResult"};
       BootStrap.launchAgent(agent, itinerary, code, relatedClasses);
   }
}
```

5. Typical Example

Figure (4) shows a document images to be analyzed. It contains a mix of text, and graphical figure. As a good feature to distinguish between graphics, half-tones, and text is the foreground to crossing count ratio which is lower for text, and higher for half-tone figures.

The layout analysis starts by segmenting the document into a number of runs. Then these runs are further divided into unrelated runs. Figure (5) displays the result of portioning the document into runs (horizontal lines), and hence dividing these runs into unrelated blocks (these blocks are displayed with the vertical lines intersected with runs lines). The next step is to classify these blocks according to their characteristics, and finally merging these blocks into unary type objects as shown in Figure (6).



Create Acrobat documents from any application.

Easily convert files created in your favorite word processing software, business tool or desktop publishing application into PDF. If you can print, you can create a PDF file. With a few mouse-clicks you're ready to share information with colleagues or clients down the hall or around the world. And you can freely distribute Acrobat Reader to anyone who needs to view, navigate, and print PDF files.

Enhance your PDF files.
Once you've created PDF files, customize them to suit your needs. Reuse information by copying and pasting PDF text and graphics into other applications, or combine PDF pages into new documents.

Customize Acrobat 3.0 to fit your needs.
[illegible]

Fig. (4): Sample Document with different objects orientation

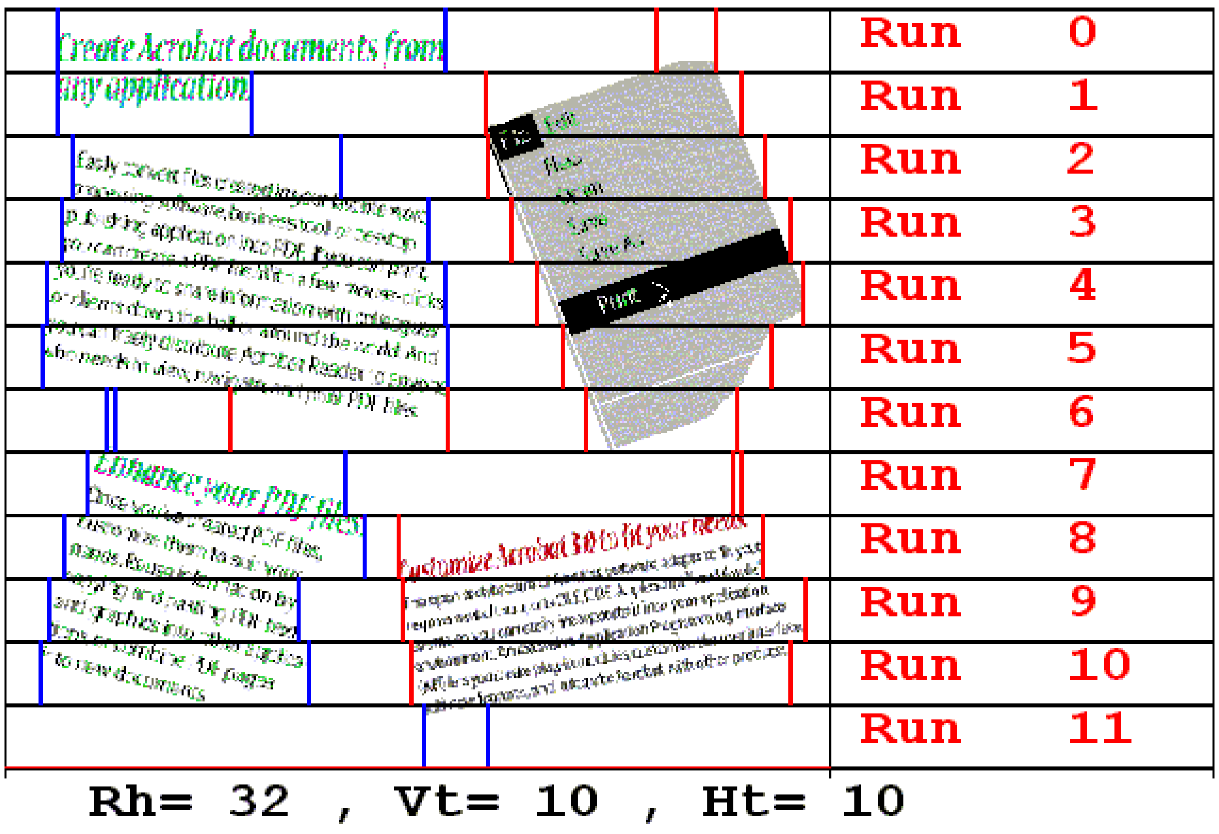
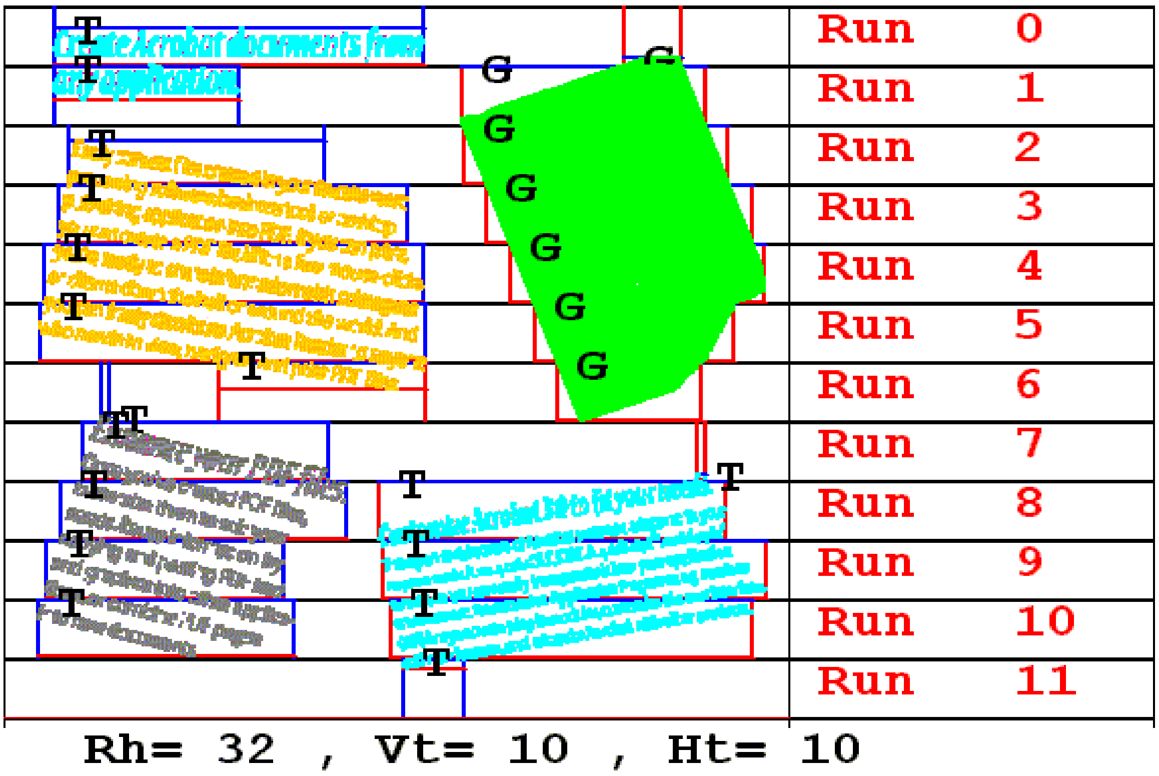


Fig. (5) : The Document After Portioning into Runs and Blocks



Legends:
Rh:= Run hight Vt :=Vertical threshold Ht:= horizontal threshold

Fig. (6): Object Construction

6. Conclusion

This paper represented an important step toward office automation. In this work we build mobile agent for document analysis Due to the nature of the mobile agent technonlogy, the system can go to a differnet machine and handles different documen images in the same time. In this work we build a single mobile agent. In the future work we will extend this single mobile agent to multi agents system. The multi agents system can improve the roles of the document analysis. The agents can be coordinated and collorbrated to obtain a good results from different documents in different machine at the same time.

Acknwledgement:
The research team is thankful for the grant no. At-21-96 from King Abdulaziz City for science and Technology.

References

[1] **Tang, Y. Y., Lee, S.,** and **Suen, C. Y.**, “Automatic Document processing A Survey”, *Pattern Recognition*, Vol. **29** No. **12** pp. 1931-1952, 1996.
[2] **Liu, F., Luo, Y., Yoshikawa, M.,** and **Hu, D.**, “A New Component based Algorithm for Newspaper Layout Analysis”, *Proc. 6th IEEE ICDAR*, 2001.
[3] **O’Gorman L.**, and **Kasturi R**. (eds.), Document Image Analysis, IEEE Computer Society Press, New York, 1995
[4] **Gross, A.,** and **Latecki, L.,** “Digital Geometric Methods in Document Image Analysis”, *Pattern Recognition*, Vol. **32** No. **3** pp. 407-424, 1999.

[5] **Nagy, G**., “Twenty Years of Document Image Analysis”, *IEEE Trans. PAMI*, Vol. 22, No.**1** pp. 38-62, 2000.

[6] **Liang, J.,** “Document Structure Analysis and Performance Evaluation”, PhD Dissertation, University of Washington, 1999.

[7] **Wooldridge, M**., “Agent-based software engineering”, *IEEE proceedings of Software Engineering* 144, 1999, 26-37.

[8] **Nicolas R. Jennings**, “An Agent-based Approach for building Complex Software Systems”, *Communications of the ACM magazines*, April 2001-Vol. **44**, No. **4**.

[9] **Saleh, M**., “An Intelligent Technique for Document Coding”, Ph. D. Thesis, Mansoura University, Egypt, 2000.

[10 A white paper, “Mobile Agent Computing”, Mitsubishi Electric, ITA, January 19, 1998.

[11] **Kamel, M.,** and **Zhao, A.,** “Extraction of Binary Character/Graphics Images from Grayscale Document Images”, *Graphical Models and Image Processing* (CVGIP), Vol.**55**, No.**3**, 1993.

[12] **Krishnamoorthy, M., Nagy, G., Seth, S.,** and **Viswanathan, M**., “syntactic Segmentation and Labeling of Digitized Pages from Technical Journals”, *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. **15**, No. **7**, 1993.

[13] **Wong, K. Y., Casey, R. G.** and **Wahl, F. M**., “Document Analysis System”, *IBM J. Res. Develop*, vol. **6** pp.642-656, Nov. 1982.

[14] **Fletcher, L. A.** and **Kasturi,** R., “A Robust Algorithm for Text String Separation from Mixed text/graphics Images”, *IEEE Trans. Pattern Analysis and Machine Intelligence*., Vol. **10**, No. **6**, pp. 910-918, 1988.

[15] Wang, D. and Srihari, S. N., “Classification of Newspaper Image Blocks Using Texture Analysis”, *Computer Vision, Graphics, Image Processing*, Vol. **47**, pp.327-352,1989.

نظام تحليل المستندات المبني على تقنية الوكيل

كمال جمبي، مصطفى صالح ، حسنين البرهمتوشي ، فتحي عيسى ، أحمد عز

قسم هندسة الحاسبات ، كلية العلوم

جامعة الملك عبدالعزيز ، جدة ، المملكة العربية السعودية

المستخلص . يهدف هذا البحث إلى تحليل صور المستندات بالاعتماد على تقنية الوكيل، ومن ثم اســتنتاج التركيب الهندسي للمستند. وتبنى فكرة التحليل على تقسيم المستند لمجموعة من الشرائح الأفقية (الأجزاء)، والتي تسهل عملية التحليل حيث يعالج كل جزء دون الحاجة لوجود الجزء الآخــر، بغــرض الاســتفادة القصوى من الذاكرة أثناء التشغيل وأثناء تحميل المستند من الإنترنت. واستخدمت آلية الوكيــل لإمكانيــة تشغيل النظام خلال شبكة. وأظهرت النتائج كفاءة النظام المقدم.