Journal of Computer Science and Cybernetics, V.32, N.2 (2016), 153–167
DOI no. 10.15625/1813-9663/32/2/8546

A HYBRID PARAGRAPH-LEVEL PAGE SEGMENTATION

HA DAI TON[1], NGUYEN DUC DUNG[2]

[1]*Ha Long Gifted High School, Quang Ninh Province, Viet Nam*
[2]*Institute of Information Technology, Vietnam Academy of Science and Technology;*
[1]*hadaiton83@gmail.com;* [2]*nddung@ioit.ac.vn*

Abstract. Automatic transformation of paper documents into electronic forms requires geometry document layout analysis at the first stage. However, variations in character font sizes, text-line spacing, and layout structures have made it difficult to design a general-purpose method. Page segmentation algorithms usually segment text blocks using global separation objects, or local relations among connected components such as distance and orientation, but typically do not consider information other than local component's size. As a result, they cannot separate blocks that are very close to each other, including text of different font sizes and paragraphs in the same column. To overcome this limitation, we proposed to use both separation objects at the whole page level and context analysis at text-line level to segment document images into paragraphs. The introduced hybrid paragraph-level page segmentation (HP2S) algorithm can handle difficult cases where the purely top-down and bottom-up approaches are not sufficient to separate. Experimental results on the test set ICDAR2009 competition and UW-III dataset show that our algorithm boosts the performance significantly comparing to the state of the art algorithms.

Keywords. Page segmentation, text-lines, homogenous regions, separation objects, paragraphs, evaluation result.

1. INTRODUCTION

Document layout analysis is one of the main components of any OCR (optical character recognition) system. The task of structural analysis includes automatically detecting image zones on a document image (physical structure analysis) and classifying them into different types such as: text, images, tables, header, footer... (logical structure analysis). The results of page segmentation are used as an input to the process of recognition and automatic data entry of image processing systems in general.

Compared with the analysis of the logical structure analysis, the physical structure analysis (page segmentation) has attracted more attention due to the diverse and complex structures of different types of document. Not only the specific types of document (books, newspapers, magazines, reports...) but also the other factors of a page such as editors and font size, layout, alignment constraints... affect detection and segmentation accuracy of the algorithm.

Document layout analysis algorithms are primarily divided based on their order of processing into three approaches: bottom-up, top-down and hybrid.

Bottom-up algorithms are both the oldest [17] and more recently published [6, 7, 13] algorithms. They classify small parts of the image (pixels, groups of pixels, or connected components), and gather those of the same type together to form regions. The key advantage of bottom-up algorithms is that they can handle arbitrarily shaped regions with ease (rectangular or non-rectangular). But the fact that they are really sensitive to the measure used to form higher-level entities is the key disadvantage; this often leads to error of over-segmentation in the page with many changes in font sizes and styles, especially in the titles with large or extra-large font size.

Top-down algorithms, e.g. [5, 12], cut the image recursively in vertical and horizontal directions along white spaces that are expected to be column boundaries or paragraph boundaries. Although top-down algorithms have the advantages that they start by looking at the largest structures on the page, they are not really able to handle free-formed layouts that often occur in magazines, such as non-rectangular regions and cross-column headings that blend seamlessly into the columns below.

A third type of algorithm [14] is based on bottom-up method to find delimiters such as rectangular whitespaces (the Fraunhofer method [2]), tab-stops (the Tab-Stop method [16]). This reduces top-down structure. And then, it is to use a combination of bottom-up method and top-down structure to detect text regions. So, these approaches can overcome over-fragmentation error of bottom-up algorithms as well as perform better with non-rectangular regions. However, it is not trivial to detect exactly delimiters by many following reasons: text regions are very close to each other, text regions are not left aligned or right aligned, large space of connected components is also the reason to lead misidentified separations,... The current hybrid algorithms almost failed when facing to these difficulties. Besides, variations in character font sizes, text-line spacing, and layout structures have made them difficult to group connected components into text blocks.

In this paper, the authors propose a new page segmentation algorithm, called Hybrid Paragraph-level Page Segmentation (HP2S), that can not only handle text blocks of any shape and variation in font sizes, but also split document images at the level of paragraph. Firstly, character-like connected components are grouped into text-lines. Before merging the text-lines into regions, under-segmented and over-segmented text-lines are treated specially using local context information like size of characters and global white columns. The processed text-lines are then grouped into homogeneous text-regions of similar font-size characters. To reduce the under-segmentation errors, the text-lines located at the beginning and ending of each paragraph are found. These local separation text-lines are finally used in combination with the global white columns to split the discovered text-regions into paragraphs. As a result, HP2S can segment page images of very hard and complicated structure. Experiments on the test set ICDAR2009 competition and UW-III dataset show that HP2S algorithm achieves the highest performance compared to the state of the art algorithms.

This paper is organized as follows. Section 2 describes in detailed the HP2S algorithm. Section 3 gives experimental results and analysis. Finally, conclusion is given in section 4.

2. PAGE LAYOUT ANALYSIS VIA SEPARATION OBJECT DETECTION

The proposed page segmentation algorithm is based on the mix of the bottom-up approach and the top-down approach. Figure 1 outlines two stages and main processing steps of the proposed HP2S algorithm. The first stage aims to group connected components into homogeneous text regions. The second stage divides homogeneous text regions into para-

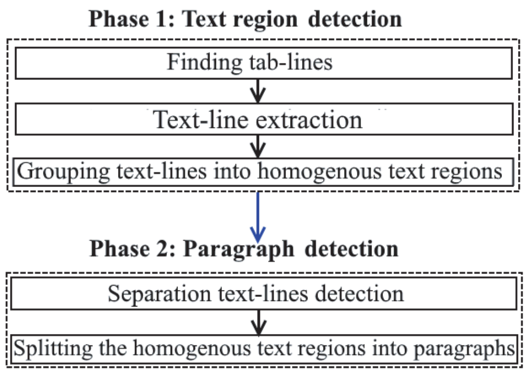

Phase 1: Text region detection
Finding tab-lines
Text-line extraction
Grouping text-lines into homogenous text regions
Phase 2: Paragraph detection
Separation text-lines detection
Splitting the homogenous text regions into paragraphs

Figure 1. Main steps of the HP2S document layout analysis

graphs. The morphological processing [16] firstly detects vertical, horizontal lines and image regions. The detected elements are subtracted from the image before passing to connected components analysis. The connected components are then filtered by their size into small considered as noise, large seen as image halftone and the rest are medium considered as text (each character is referred to CC, a set of character is referred to CCs). The array of CCs at the left or the right of column is linked into tab-lines. These tab-lines are liked as delimiters. The CCs are grouped into text-lines, homogeneous text region based the mix of parameter adaptive and tab-lines. Finally, the homogenous text regions are split into paragraph based on context analysis at text-line level.

2.1. Phase 1: Text regions detection

Instead of using some parameters estimated in whole page, e.g [13]. The HP2S algorithm takes advantage of both high-level information tab-stops [16] and adaptive parameters to gather connected components into text-lines, homogeneous text regions.

2.1.1. Finding tab-lines as line segments

Tab-stops [16] represent the vertical alignments that can be found on text-lines edges and that are inferred by margins and column edges which are all placed at a fixed x-position. They are used to present physical separators and white rectangles in finding column structures of a page. In this paper, a simple method is used to determine the tab-stops, which consists of fewer steps, simpler and easier to implement as follows.

For each CC, we define rectangles next to the left, right, top and bottom as illustrated in Figure 2. Since then, the CCs are considered as a candidate tab-stop (tabstop-cand) if their left or right adjacency does not intersect with any others (Figure 3b). Next, tabstop-cand intersecting with their top and bottom adjacency is accepted as a tab-stop (Figure 3c). We then revise once again to find tab-stops at the beginning and the ending of the each chain of the tab-stops (Figure 3d). Finally, it is to connect the tab-stops from top to bottom together into groups, provided that between two adjacent tab-stops are just white space. Each group of tab-stops with three or more numbers will be retained and straight lines connecting the tab-stops are the last tab-lines detected by our algorithm (Figure 3e).

The detected tab-lines will be used as delimiters for text-lines detection as well as grouping text-lines into homogeneous text regions and dividing the homogeneous text regions into paragraphs.

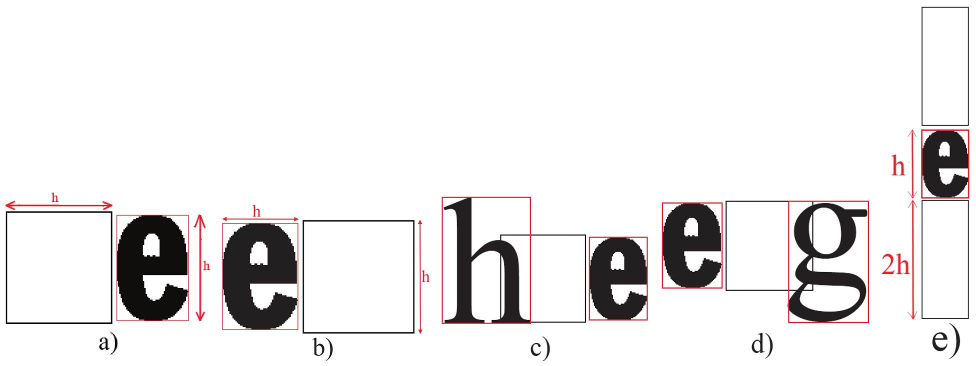


Figure 2. Illustration of rectangles next to the left, right, top and bottom of connect components: a) and b) Left and right adjacency of a tabstop-cand "e"; c) and d) "e" is not a tabstop-cand as its adjacency intersects a CC; e) top and bottom adjacency of "e"

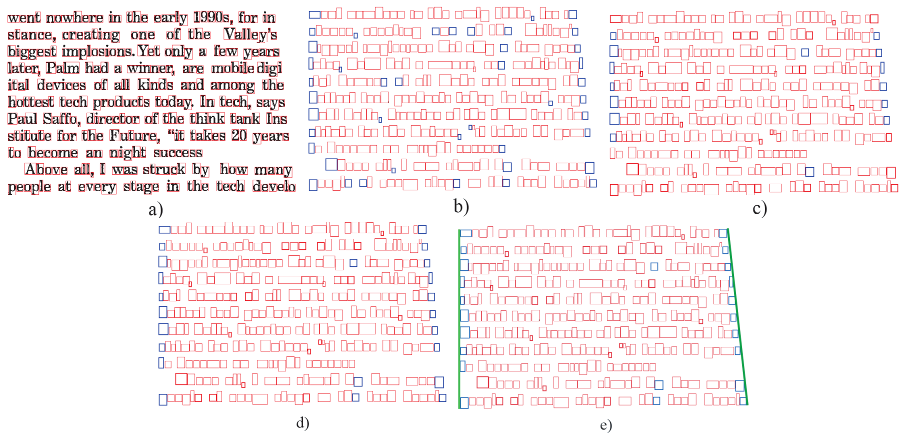
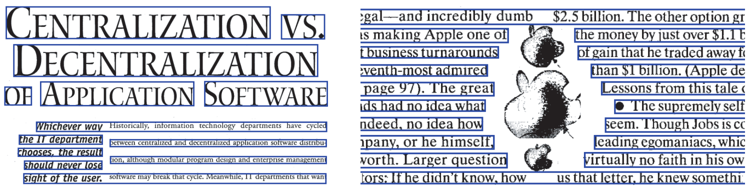

went nowhere in the early 1990s, for in
stance, creating one of the Valley's
biggest implosions. Yet only a few years
later, Palm had a winner, are mobile digi
ital devices of all kinds and among the
hottest tech products today. In tech, says
Paul Saffo, director of the think tank Ins
stitute for the Future, "it takes 20 years
to become an night success
Above all, I was struck by how many
people at every stage in the tech develo

Figure 3. illustrates the steps of tab-line detection: a) Original image with bounding boxes of CCs, b) candidate tab-stops, c) tab-stops, d) extension tab-stops, e) the yellow-green lines are the tab-lines detected by our algorithm

2.1.2. Text-lines extraction

Text-lines extraction is one of the most important step in the process of a bottom-up page segmentation algorithm as it directly affects the final result. Docstrum [13], Voronoi [9], Smearing [18] detect text-lines by grouping CCs based on their nearest neighbors. However, these methods usually produce over-fragmented text-line failures in document images with large difference in font sizes and styles (especially the titles) and merging text-line failures


CENTRALIZATION VS.
DECENTRALIZATION
OF APPLICATION SOFTWARE
Whichever way the IT department chooses, the result should never lose sight of the user.

(a) (b)

Figure 4. Fragmented text-lines in title and merging text-lines in columns

in the columns that are very close to each other, as shown in Figure 4(a).

Tab-Stop [16], RAST [5] avoid the fusion error of text-lines by using delimiters between column like as tab-lines [16] or white-spaces [5]. Scanning the CCs from left to right and top to bottom, runs of similarly classified CCs are gathered into text-lines, subject to the constraint that no text-line may cross a tab-line (rectangular white space). These algorithms are often limited either in the case where the columns are not left aligned or right aligned or in the case of angle-page images caused by scanning process. Because there are no delimiters between columns, the horizontally scan for text-lines will completely fail, see Figure 4(b).

To overcome the above limitations, the following procedure is used to determine text-lines of different font sizes and line segments in very close columns.

Firstly, the Hough transform [10] is performed the set of the bottom-center points of CCs to find aligned CCs. These aligned CCs are candidates to create text-lines, we called them textline-cand (Figure 5 and 6). For each textline-cand, we build a histogram of distances between adjacent CCs. The highest peak of the histogram is considered as the distance between characters and the second peak is the distance between words, d_w. The d_w is used together with tab-lines to split textline-cands into text-lines as follows.

Two horizontally adjacent CCs belong to the same text-line if they are not located on different sides of any tab-line and the horizontal distance between them is not larger than two times of d_w. The combination of the top-down analysis (tab-lines) with the traditional bottom-up text lines finding helps to separate columns that are very close to each other. As illustrated in Figure 5a, the gap between two columns is nearly the same as the distance between words on the textline-cands. However, there exits a vertical tab-lines between them and the textline-cand are broken down into text lines that belongs to two different columns (Figure 5b). When text columns are not left and/or right aligned, tab-lines will not exist and the parameter d_w will prove itself very useful in separating text-lines. Most of the cases, the distance between text-lines d is larger than the distance between words d_w (Figure 6). Different from other bottom-up algorithms, we don't use one value of d_w for all textline-cands. The parameters d_w are estimated locally on each set of characters with similar font size (textline-cand). Therefore, this procedure helps to reduce the over-fragmented text-line errors, especially text-lines with large font size in the title (Figure 5b).

In some cases, such as regions of references or paragraphs beginning with symbols, the text regions are often aligned and indented with indicators and symbols. Thereby, the tab-lines will split indicators or symbols out of text-lines (Figure 7).

For dealing with this type of error, we firstly find additional right tabstop-cands using the same method in section 2.1.1 with the width of the right adjacent rectangle of each CC being only half as height of the respective CC considered. After that, these right tabstop-

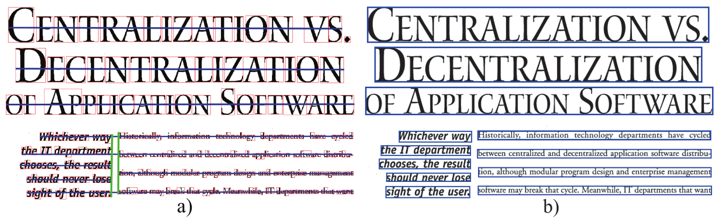


Figure 5. illustrates the tab-lines used as delimiters for text-line detection. a) shows Textline-cands, the yellow-green lines are the tab-lines. CCs lying on different sides of a tab-line will belong to different text-lines. b) The text-lines outcome of our algorithm

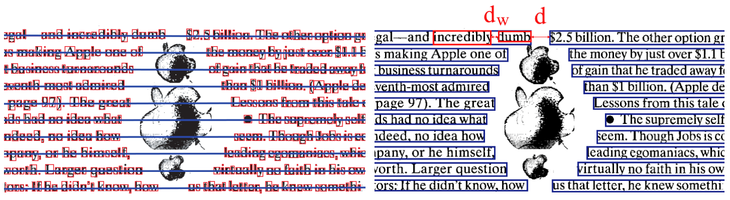


Figure 6. a) Aligned CCs as textline-cands, b) When the tab-line does not exist, d_w is used to separate CCs into text-lines

cands that intersect the left tabstop-cands determined in section 2.1 will be accepted as the tab-stops at the references and symbols, denoted by m_tabs (Figure 8). The m_tabs are CCs that have been separated out of text-lines by tab-lines. Finally, we combine m_tabs with left adjacent text-lines and marked them as separation text-lines (Figure 8b). These separation text-lines will be used again in section 2.2.2 to detect paragraphs.

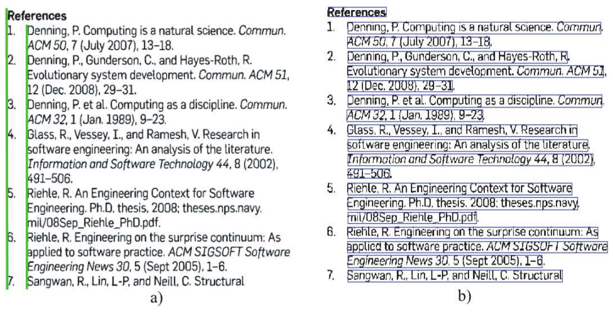
2.1.3. Grouping text-lines into homogenous text regions

In this section, the process of grouping text-lines into homogeneous text regions is presented. The bottom-up approach is used to gather neighbor text-lines together to form text regions of any shape.

The set of text-lines discovered in previous section is arranged in order from left to right and from top to bottom. A pair of text-lines $(line_i, line_j)$ satisfies simultaneously the following conditions then they will be grouped into the same region:

$$\begin{cases} (i) & DistHoriz\,(line_i,\ line_j) \le AvgHoriz, \\ (ii) & CheckRulling\,(line_i,\ line_j)\ \ is\ \ false, \\ (iii) & CheckTabline\,(line_i,\ line_j)\ \ is\ \ false, \\ (iv) & |y_i - y_j|\ \le (1+\theta) * x\_height_{ij} \end{cases}$$

In the above conditions, $DisHoriz(.,.)$ is the horizontal distance between text-lines. $AvgHoriz$ is the average horizontal distance of text-lines. y_i and y_j are respectively y-coordinates of the center of text-lines $line_i$ and $line_j$. $x - height_{ij}$ is the minimum of the x-heights. $CheckTabline\,(.\,,\,.)$ returns true if two text-lines are located on two sides of a

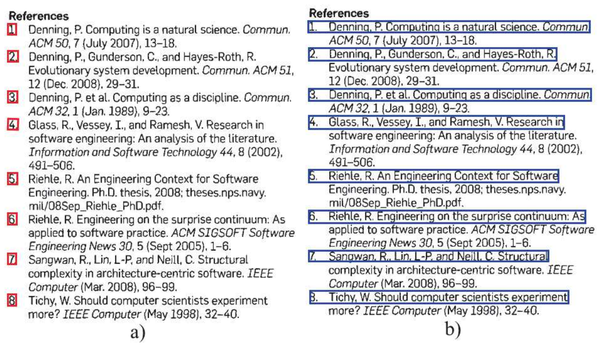

References
1. Denning, P. Computing is a natural science. *Commun. ACM 50*, 7 (July 2007), 13–18.
2. Denning, P., Gunderson, C., and Hayes-Roth, R. Evolutionary system development. *Commun. ACM 51*, 12 (Dec. 2008), 29–31.
3. Denning, P. et al. Computing as a discipline. *Commun. ACM 32*, 1 (Jan. 1989), 9–23.
4. Glass, R., Vessey, I., and Ramesh, V. Research in software engineering: An analysis of the literature. *Information and Software Technology 44*, 8 (2002), 491–506.
5. Riehle, R. An Engineering Context for Software Engineering. Ph.D. thesis. 2008; theses.nps.navy.mil/08Sep_Riehle_PhD.pdf.
6. Riehle, R. Engineering on the surprise continuum: As applied to software practice. *ACM SIGSOFT Software Engineering News 30*, 5 (Sept 2005), 1–6.
7. Sangwan, R., Lin, L-P, and Neill, C. Structural

a)

References
1. Denning, P. Computing is a natural science. *Commun. ACM 50*, 7 (July 2007), 13–18.
2. Denning, P., Gunderson, C., and Hayes-Roth, R. Evolutionary system development. *Commun. ACM 51*, 12 (Dec. 2008), 29–31.
3. Denning, P. et al. Computing as a discipline. *Commun. ACM 32*, 1 (Jan. 1989), 9–23.
4. Glass, R., Vessey, I., and Ramesh, V. Research in software engineering: An analysis of the literature. *Information and Software Technology 44*, 8 (2002), 491–506.
5. Riehle, R. An Engineering Context for Software Engineering. Ph.D. thesis. 2008; theses.nps.navy.mil/08Sep_Riehle_PhD.pdf.
6. Riehle, R. Engineering on the surprise continuum: As applied to software practice. *ACM SIGSOFT Software Engineering News 30*, 5 (Sept 2005), 1–6.
7. Sangwan, R., Lin, L-P, and Neill, C. Structural

b)

Figure 7. illustrates miss line error. a) The yellow-green lines show tab-lines, b) The indicators have been split out of text-lines by tab-lines


References
1. Denning, P. Computing is a natural science. *Commun. ACM 50*, 7 (July 2007), 13–18.
2. Denning, P., Gunderson, C., and Hayes-Roth, R. Evolutionary system development. *Commun. ACM 51*, 12 (Dec. 2008), 29–31.
3. Denning, P. et al. Computing as a discipline. *Commun. ACM 32*, 1 (Jan. 1989), 9–23.
4. Glass, R., Vessey, I., and Ramesh, V. Research in software engineering: An analysis of the literature. *Information and Software Technology 44*, 8 (2002), 491–506.
5. Riehle, R. An Engineering Context for Software Engineering. Ph.D. thesis. 2008; theses.nps.navy.mil/08Sep_Riehle_PhD.pdf.
6. Riehle, R. Engineering on the surprise continuum: As applied to software practice. *ACM SIGSOFT Software Engineering News 30*, 5 (Sept 2005), 1–6.
7. Sangwan, R., Lin, L-P, and Neill, C. Structural complexity in architecture-centric software. *IEEE Computer* (Mar. 2008), 96–99.
8. Tichy, W. Should computer scientists experiment more? *IEEE Computer* (May 1998), 32–40.

a)

References
1. Denning, P. Computing is a natural science. *Commun. ACM 50*, 7 (July 2007), 13–18.
2. Denning, P., Gunderson, C., and Hayes-Roth, R. Evolutionary system development. *Commun. ACM 51*, 12 (Dec. 2008), 29–31.
3. Denning, P. et al. Computing as a discipline. *Commun. ACM 32*, 1 (Jan. 1989), 9–23.
4. Glass, R., Vessey, I., and Ramesh, V. Research in software engineering: An analysis of the literature. *Information and Software Technology 44*, 8 (2002), 491–506.
5. Riehle, R. An Engineering Context for Software Engineering. Ph.D. thesis. 2008; theses.nps.navy.mil/08Sep_Riehle_PhD.pdf.
6. Riehle, R. Engineering on the surprise continuum: As applied to software practice. *ACM SIGSOFT Software Engineering News 30*, 5 (Sept 2005), 1–6.
7. Sangwan, R., Lin, L-P, and Neill, C. Structural complexity in architecture-centric software. *IEEE Computer* (Mar. 2008), 96–99.
8. Tichy, W. Should computer scientists experiment more? *IEEE Computer* (May 1998), 32–40.

b)

Figure 8. a) The CCs that are labeled red are m_tabs, b) the red text-lines are recovered

tab-line, otherwise it returns false. $CheckRulling(.,.)$ returns true if two text-lines are located on two sides of a horizontal line, otherwise it returns false.

The conditions (i) and (ii) ensure separating text-lines in different columns. This is done by using the combination of both top-down delimiters and strictly bottom-up merging condition. The condition (iii) then permits grouping only text-lines of similar font size that 'overlap' vertically.

It is noteworthy that the condition (iii) favors text lines of the same font size and becomes stricter when their sizes are different. In the latter case, the distance between two line centers on the left-hand side counts the large font size whereas the right-hand side counts the small font size. We will show experimentally that the performance of HP2S is very insensitive to the value of θ (Figure 19). Our primary experiment indicates that the suitable values for θ are in the range between 1.4 and 1.6. Therefore, the default value of 1.5 is used for θ in all of our experiments.

2.2. Paragraph detection

The difficulties in page segmentation are not only the change in structure of document images, changes of font sizes and styles, but also the closeness of text-lines in different text blocks. Since the separation attributes such as distance or angle between CCs are not enough

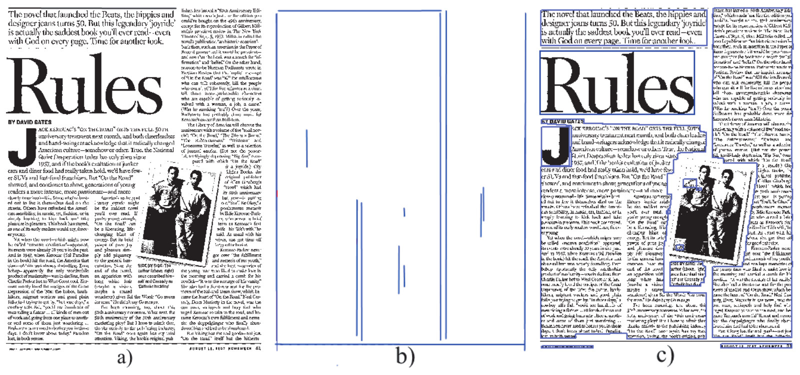


Figure 9. a) Original image, b) tab-lines are used as delimiter objects, c) homogeneous text regions are detected in Phase 1

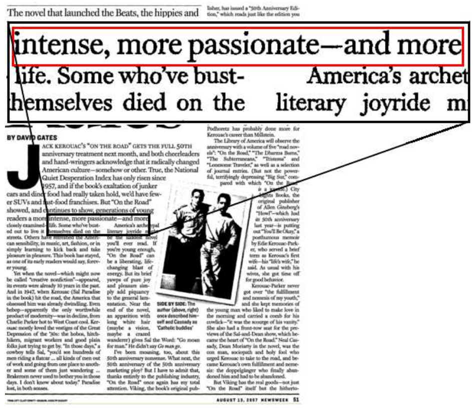

intense, more passionate—and more
life. Some who've bust- America's archet
hemselves died on the literary joyride m

Figure 10. The red text-lines that lay across two columns are very close to text-lines under them

to separate paragraphs, we rely on additional separation objects found by logical analysis. These objects are text-lines starting of each paragraphs and text-lines laying across two columns (Figure 10).

In this section, the step of paragraph detection is presented. The tab-stops and separation text-lines are used to split the homogeneous text regions into paragraphs.

2.2.1. Separation text-lines detection

For each text-line (*current_line*), the *pre_line* and *next_line* are called as the nearest top-neighbor and the nearest bottom-neighbor of the *current_line* if the vertical distance to *current_line* is smaller than two times the height of the *current_line*. If there is no such *pre_line* or *next_line*, we consider it coincide with *current_line*, (Figure 12).

a) Finding starting text-line of a paragraph: denote d_{tl} as the distance from current line (*current_line*) to the nearest tab-line, h is height of *current_line*. Then, if d_{tl} is large enough, and the left edges of *pre_line* and *next_line* nearly intersects the tab-line, the

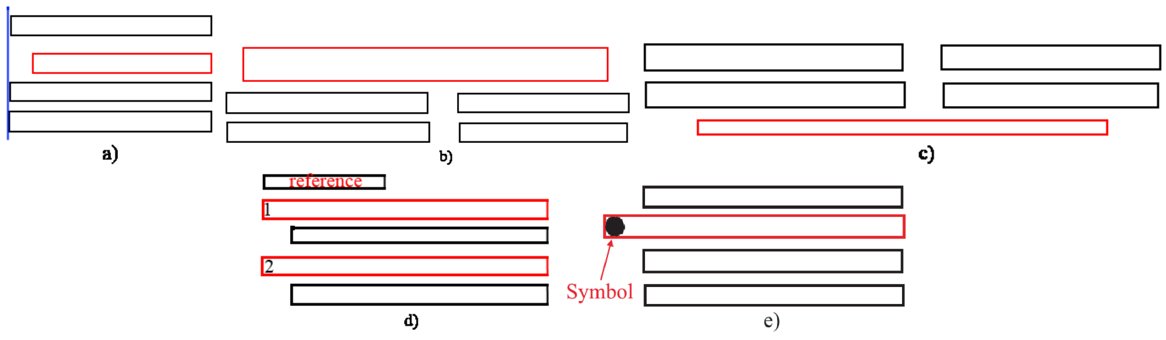


Figure 11. Red rectangles show separation text-lines, the blue line is a tab-line. a) The start of a paragraph, b) and c) Line is laid across two columns, d) and e) special cases

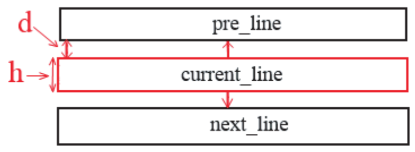


Figure 12. *Pre_line* and *next_line*

current_line is accepted as the starting text-line of a paragraph (Figure 13).

b) Finding text-lines lay across two columns: a text-line (*current_line*) is considered as laying over/under two columns (Figure 14) if these following conditions are satisfied:

- Heights of *current_line* and *pre_line* are similar,
- Heights of *next_line*, *line_i*, *line_j*, *line_k* are similar,
- Horizontal distances from pairs *next_line* to *line_i* and *line_k* to *line_j* are similar,
- Vertical distances from *next_line* to *line_k* and *line_i* to *line_j* are similar.

2.2.2. Splitting homogeneous text regions into paragraphs

The separation text-lines are used to sub-divide the homogeneous text regions detected in section 2.1.3. into paragraphs as illustrated in Figure 15. Firstly, we scan from top to

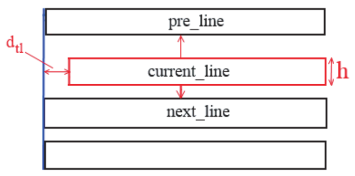


Figure 13. The red text-line is the starting of a paragraph

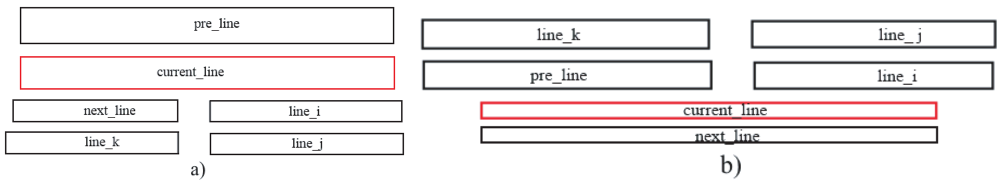


Figure 14. The detected red text-lines are laid-over a) and laid-under b) text columns

bottom and from bottom up to top for across separation text-lines (Figure 11b or 11c) if exist and then divide the region into sub-regions (step 1 in Figure 15.c). After that, text-lines in those sub-regions are grouped into columns using conditions (i), (ii) in section 2.1.3. (step 2). Finally, each column is divided into paragraphs using separation text-lines in Figures 11a, 11d or 11e (step 3).

As illustrated in Figure 15, separation text-lines are very effective in separating close text blocks of similar font size that transitive closure operations of the bottom-up algorithms, e.g. Docstrum, Voronoi, usually fail. The global separation objects like tab-stops or white spaces can handle the close text columns, but they limit themselves to rectangular text blocks only. The proposed combination of global and local separation objects help to solve thoroughly this problem.

3. EXPERIMENT

In this section, we evaluate and compare empirical performance of HP2S with the proposed methods in ICDAR2009 page segmentation competition [2]. The tool LayoutEval 1.5 [8] is used for this study.

3.1. Data set

The UW-III dataset [15] and the test set of ICDAR2009 competition (ICDAR2009 dataset) are used for the performance evaluation. These data sets have text-line and paragraphs level ground-truth for each document image. The text-line and paragraphs level ground-truth are represented by non-overlapping polygons. The UW-III dataset has 1600 deskewed binary document images at 300 dpi resolution. It provides a good basis for comparative evaluation of page segmentation algorithms because the majority of documents available today like books, journals, magazines, letters etc. has Manhattan layouts and many document images with heavy noise (speckles, margin noise or undesired text parts from the neighboring page...). The PRImA dataset [3] has 305 document images at 300 dpi resolution. It contains a wide variety of different document types, reflecting the various challenges in layout analysis. The layouts of these pages contain a mixture of simple and complex layouts, including many instances of text wrapping tightly around images, varying font sizes and other characteristics which are useful to evaluate layout analysis methods on. This dataset is divided into two subset: training dataset contains 250 images and testing dataset contains 55 images. The testing dataset is the test set of ICDAR2009 page segmentation competition (ICDAR2009 dataset).

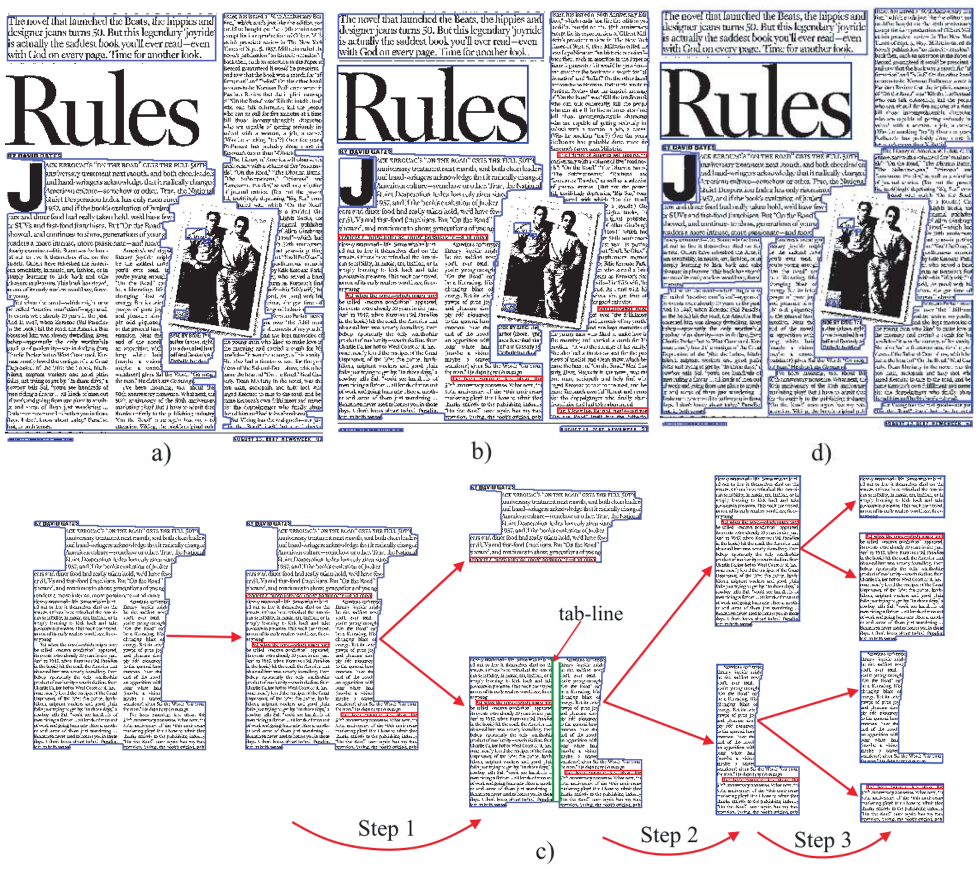


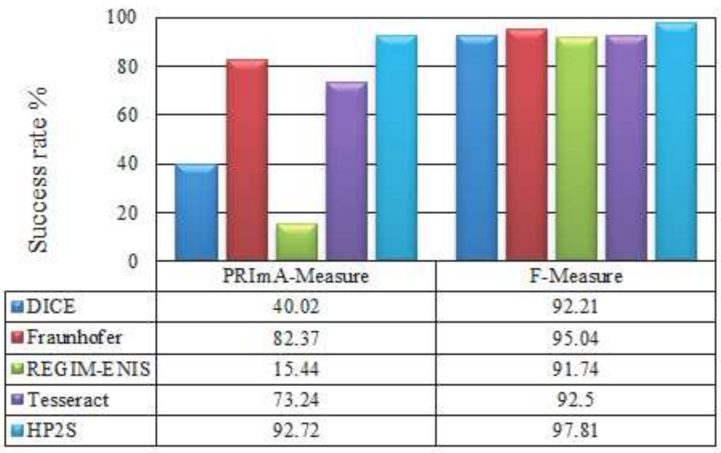
Figure 15. a) The segmentation result without using separation text-lines, b) Separation text-lines are detected, d) Final page segmentation result, c) Detailed steps of sub-dividing a text regions into paragraphs

3.2. Performance evaluation

The evaluation of document layout analysis is always complicated because they depend heavily on dataset, ground-truth and evaluation methodology. There are many different datasets and evaluation methodologies. The ICDAR page segmentation competitions [2, 4] and [1] provide an evaluation methodology which has been used successfully to compare the entrants of the competitions. The methodology combines different types of segmentation errors (split, merge, miss, false detection, misclassification and reading order) using adjustable weights accounting for different application scenarios. Mao et al. [11] presented an empirical benchmarking methodology based on text-line measure of page segmentation accuracy. This measure called PSET-measure is particularly useful because it does not make assumptions about the layout of the document. Besides, it requires only text-line level ground-truth. Using both of these methodologies, we evaluated the performance of our method on the test set of ICDAR2009 competition (ICDAR2009 set) and UW-III dataset in two measures: PRImA-measure and PSET-measure. The details of these measure can be found in [2, 8] and [11].

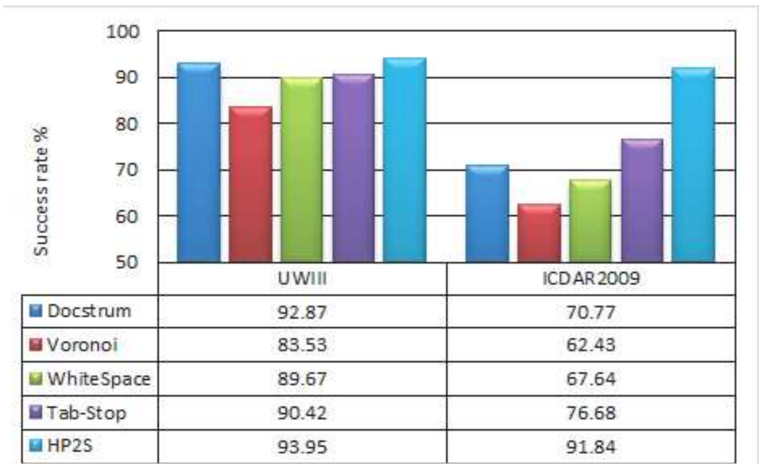
3.3. Result and discussion

Evaluation results are presented in this section, on two datasets ICDAR2009 and UW-III, in the form of graphics with corresponding tables. Firstly, Fig. 16 the F-measure and PRImA-measure are reported for the four algorithms get the highest results in ICDAR2009 competition and HP2S algorithm on dataset ICDAR2009. Secondly, Fig 17 the PSET-measure is reported for the state of the art algorithms and our algorithm on two datasets ICDAR2009 and UW-III. Finally, different error types (split, merge and miss) of the state of the art algorithms and HP2S algorithm is compared in Fig 18, on ICDAR2009 set.



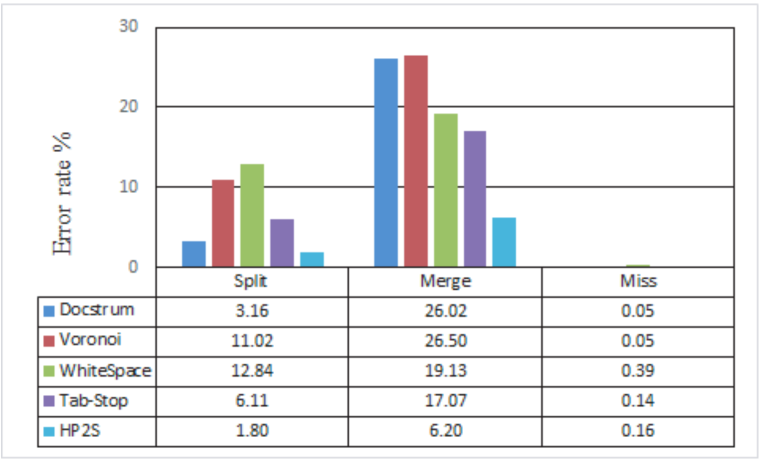
	PRImA-Measure	F-Measure
DICE	40.02	92.21
Fraunhofer	82.37	95.04
REGIM-ENIS	15.44	91.74
Tesseract	73.24	92.5
HP2S	92.72	97.81

Figure 16. F-measure and PRImA-measure of HP2S and the published algorithms in ICDAR2009



	UWIII	ICDAR2009
Docstrum	92.87	70.77
Voronoi	83.53	62.43
WhiteSpace	89.67	67.64
Tab-Stop	90.42	76.68
HP2S	93.95	91.84

Figure 17. Results using PSET-measure and evaluating on two dataset: UW-III dataset, test set of ICDAR2009 competition



	Split	Merge	Miss
🟦 Docstrum	3.16	26.02	0.05
🟥 Voronoi	11.02	26.50	0.05
🟩 WhiteSpace	12.84	19.13	0.39
🟪 Tab-Stop	6.11	17.07	0.14
🟦 HP2S	1.80	6.20	0.16

Figure 18. PSET-measure for Errors of different algorithms on the ICDAR2009 dataset

The evaluation results on different datasets with various evaluation methodologies show that our algorithm has an overall advantage. The algorithm HP2S has superior accuracy compared to the state-of-the-art algorithms, 92.72% of HP2S compare to 82.37% of Fraunhofer, and 91.84% of HP2S compare to 76.68% of Tab-Stop (see Fig 16 and 17). Figure 17 shows that our algorithm gives the best performance on both of datasets and is more stability than the state-of-the-art algorithms. HP2S algorithm significantly reduces split and merge errors (see Fig 18). These errors usually occur on region images with large font size or region images are very close to each other.

Figure 20 shows the average running time for one page of Docstrum, Voronoi, WhiteSpace, TabStop and HP2S on the ICDAR 2009 dataset. The experiment was conducted on an Intel Core i5 Processor 3.2GHz machine. HP2S takes about 0.7 second to analyze one document image, faster than Voronoi, WhiteSpace, TabStop and slower than Docstrum.

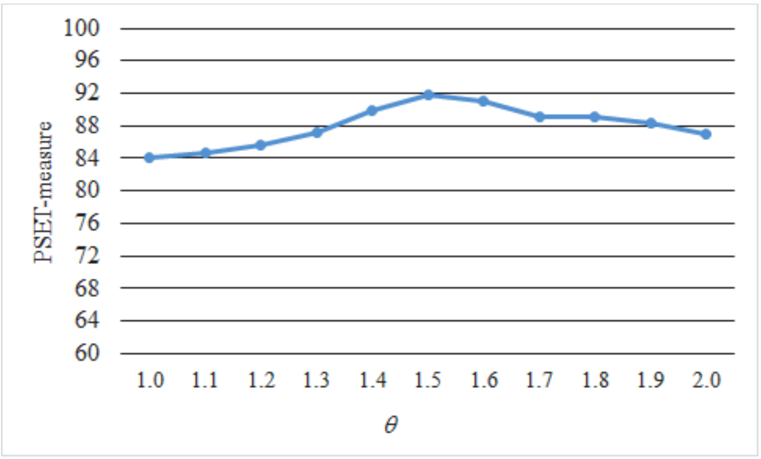


Figure 19. PSET-measure of HP2S with different values of θ on the ICDAR2009 dataset

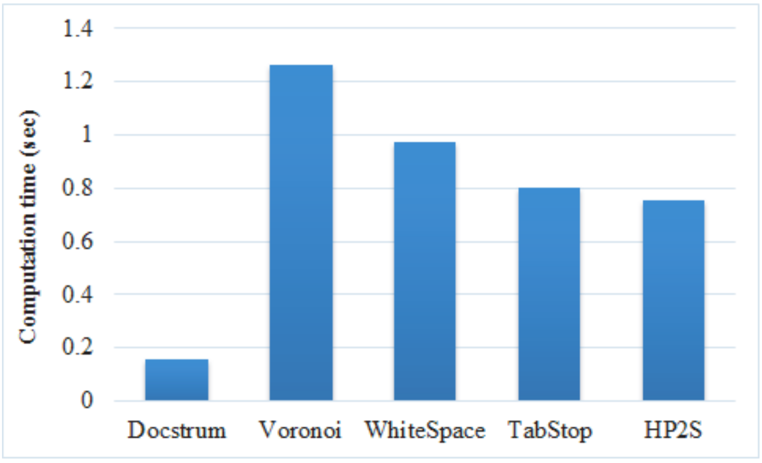


Figure 20. Average running time for one page segmentation of representative algorithms

4. CONCLUSIONS

The authors present a hybrid paragraph-level page segmentation (HP2S) algorithm that uses combination of global separation properties and local information analysis to find separation objects, thereby segment page document images at the level of paragraph. The outcomes of experimental show that HP2S has achieved high performance on set test of ICDAR2009 competition and UW-III dataset regardless of changes in character font sizes and complex document layout.

ACKNOWLEDGMENT

This work was supported in part by the Vietnam Academy of Science and Technology under the grand VAST01.08/15-16 "Development of document analysis and recognition methods for automatic data entry".

REFERENCES

[1] A. Antonacopoulos, C. Clausner, C. Papadopoulos, and S. Pletschacher, "Competition on historical newspaper layout analysis (hnla 2013)," in *Proc. of 12th International Conference on Document Analysis and Recognition (ICDAR)*, 2013, pp. 1454–1458.

[2] A. Antonacopoulos, S. Pletschacher, D. Bridson, and C. Papadopoulos, "Page segmentation competition," in *2009 10inth International Conference on Document Analysis and Recognition (ICDAR 2009)*, no. 5, 2009, pp. 1370–1374.

[3] A. Antonacopoulos, D. Bridson, C. Papadopoulos, and S. Pletschacher, "A realistic dataset for performance evaluation of document layout analysis," in *2009 10th International Conference on Document Analysis and Recognition.* IEEE, 2009, pp. 296–300.

[4] A. Antonacopoulos, C. Clausner, C. Papadopoulos, and S. Pletschacher, "Historical document layout analysis competition," in *2011 International Conference on Document Analysis and Recognition.* IEEE, 2011, pp. 1516–1520.

[5] T. M. Breuel, “Two geometric algorithms for layout analysis,” in *International workshop on document analysis systems.* Springer, 2002, pp. 188–199.

[6] M. Chen, X. Ding, and Y. Wu, “Unified hmm-based layout analysis framework and algorithm,” *Science in China Series F: Information Sciences*, vol. 46, no. 6, pp. 401–408, 2003.

[7] S. Chowdhury, S. Mandal, A. Das, and B. Chanda, “Segmentation of text and graphics from document images,” in *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, vol. 2. IEEE, 2007, pp. 619–623.

[8] C. Clausner, S. Pletschacher, and A. Antonacopoulos, “Scenario driven in-depth performance evaluation of document layout analysis methods,” in *2011 International Conference on Document Analysis and Recognition.* IEEE, 2011, pp. 1404–1408.

[9] K. Kise, A. Sato, and M. Iwata, “Segmentation of page images using the area voronoi diagram,” *Computer Vision and Image Understanding*, vol. 70, no. 3, pp. 370–382, 1998.

[10] G. Louloudis, B. Gatos, I. Pratikakis, and K. Halatsis, “A block-based hough transform mapping for text line detection in handwritten documents,” in *Tenth International Workshop on Frontiers in Handwriting Recognition.* Suvisoft, 2006.

[11] S. Mao and T. Kanungo, “Software architecture of pset: A page segmentation evaluation toolkit,” *International Journal on Document Analysis and Recognition*, vol. 4, no. 3, pp. 205–217, 2002.

[12] G. Nagy, S. Seth, and M. Viswanathan, “A prototype document image analysis system for technical journals,” *Computer*, vol. 25, no. 7, pp. 10–22, 1992.

[13] L. O’Gorman, “The document spectrum for page layout analysis,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 11, pp. 1162–1173, 1993.

[14] T. Pavlidis and J. Zhou, “Page segmentation and classification,” *CVGIP: Graphical models and image processing*, vol. 54, no. 6, pp. 484–496, 1992.

[15] I. Phillips, “Users reference manual,” *CD-ROM, UW-III Document Image Database-III*, 1995.

[16] R. W. Smith, “Hybrid page layout analysis via tab-stop detection,” in *2009 10th International Conference on Document Analysis and Recognition.* IEEE, 2009, pp. 241–245.

[17] F. M. Wahl, K. Y. Wong, and R. G. Casey, “Block segmentation and text extraction in mixed text/image documents,” *Computer graphics and image processing*, vol. 20, no. 4, pp. 375–390, 1982.

[18] K. Y. Wong, R. G. Casey, and F. M. Wahl, “Document analysis system,” *IBM journal of research and development*, vol. 26, no. 6, pp. 647–656, 1982.

Received on July 18 - 2016
Revised on October 31 - 2016