Text-Image Segmentation and Compression using Adaptive Statistical Block Based Approach

Nidhal Kamel Taha El-Omari, Ahmad H. Al-Omari, Ali Mohammad H. Al-Ibrahim, Tariq Alwada'n

Abstract*: Images and scanned text documents are gradually more used in a vast range of applications. To reduce the needed storage or to accelerate their move through the computers networks, the document images have to be compressed. Traditional compression mechanisms, which are generally developed with a particular image type and purpose, are facing many challenges with mixed documents. This paper describes a statistical block-based technique for an automatic document image segmentation and compression. Based on the number of detected colors in each region of the image, this approach creates a new representation of the image that can produce very highly-compressed document files that nonetheless retain excellent image quality. The proposed algorithm segments the compound document image into blocks of equal size. The blocks are classified into seven different categories. Each category represents an image part that shares the same properties. A new representation of each category is formed and the similar adjacent blocks are merged to form labeled regions sharing the same properties. At the end, to achieve better compression ratio, the different regions of the image are compressed using different compression techniques.

Index Terms*: Adaptive Compression, Block-Based Segmentation, Image Document Compression, Image Segmentation, Lookup Dictionary Table (LUD).

I. INTRODUCTION

Scanned text documents are increasingly used in a wide range of applications, including but not limited to archiving systems and document management systems. Many of these documents, called compound or mixed documents, consist of a mixture of texts, pictures, graphics (drawing), and background. The storage requirements of uncompressed high quality color scanned documents are indeed quite vast. This can sometimes cause for document transformation and storage. And unfortunately, managing such uncompressed documents proves to be inefficient and creates the potential effect of substantially limiting their benefits and may perhaps never meet the ever-growing information demands of the users. As a standard A4 color page document, scanned with a resolution of 600 dpi, requires around 91 million bytes of storage space, assuming 24 bit-depth and a standard 8 X 12 inches sheet. Therefore, to reduce the space occupation or to speed up their transfer through the computers networks, the document images need to be compressed. Traditional compression mechanisms, which are generally developed with a particular image type and purpose, are facing many challenges with mixed documents. Unfortunately, these documents do not compress well using classical image compression algorithms such as JPEG-2000. This is due to the presence of sharp edges on top of the smooth surfaces of the text and graphics, typically found in natural images. What is more, compression algorithms for text facsimiles, such as JBIG2, are not suited for color or gray level images. [1,2,3,4,5,16,18,25]

Image segmentation plays an important role in compression of scanned documents, which is to part an image into different meaningful regions or clusters which have similar features [2,3,1,19,20,21,22,25]. In this paper, we tackle the problem of segmenting and compressing mixed (compound) digital documents. In order to compress it more effectively, the proposed technique segments the image into seven different types of components. Every image component is a homogenous region (or regions) having "common features" like color gamut and number, shape, pixel intensity, region formation, text occurrence, grey level, and others [20]. All segmented regions are non-overlapping [25]. In order to achieve better compression ratios, every component is compressed separately using the most appropriate compression technique. This approach differ from previous ones such as DjVu, Tiff-FX, and MRC, by being extremely simple and fast, while yielding close to and in many cases better than the state-of-the-art compression performance [6,7,9,10,14].

This work is indeed a continuation of the previous works in the area of document image segmentation and compression [2,3,16]. To explore further the arguments set out above, this paper is divided into six sections. While this section provides an introduction to the main theme of the paper, the rest of the paper is organized as follows. Section 2 looks at the related work and algorithms used. Section 3 presents the approaches developed in this research. In Section 4, the algorithm of this proposed solution is presented. Then, Section 5 presents the training results and describes the analysis of the results. Finally, Section 6 provides the conclusions and offers avenues for future work.

II. BACKGROUND

Image segmentation of a mixed document aims to separate background, text, pictures, and graphical components of a document image [1,2,3]. However, the union of these various image components generates the original document. There are different techniques proposed in the literature to solve the problem of segmenting and compressing compound documents. These techniques can be classified into three different categories that Fig. 1 illustrates.

Revised Version Manuscript Received on February 27, 2017.

Nidhal Kamel Taha El-Omari, WISE University, Faculty of Information Technology, Amman, Jordan, nidhal.omari@wise.edu.jo

Ahmad H. Al-Omari, Northeren Border University, Faculty of Science, Computer Science Division, Saudi Arabia, kefia@yahoo.com (Correspondence Author)

Ali Mohammad H. Al-Ibrahim, WISE University, Faculty of Information Technology, Amman, Jordan, ali.alibrahim@wise.edu.jo

Tariq Alwada'n, WISE University, Faculty of Information Technology, Amman, Jordan, tariq.alwadan@wise.edu.jo

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication Pvt. Ltd.

The first category of algorithms transforms the document into a black and white image. These algorithms are designed to scan and store documents in black and white colors. Then these images are decoded using lossless decoders, such as "Fax Group 3" and "Fax Group 4". Although they achieve high compression rate and preserve text legibility, they lead to the losing of contrast and color information. They may be suitable for some business and technical documents, but unsuitable for other document types such as magazines or historical documents [4,6,7,9,10,5].

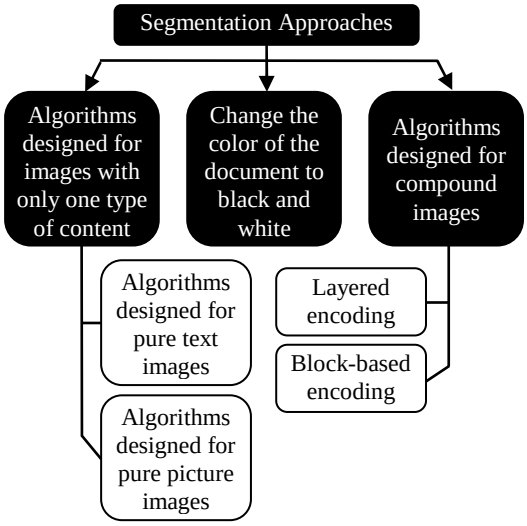

Figure 1: Compressing Compound Documents

The other category uses algorithms that are only designed for one type of content. Some of them are designed to compress pure text images which contain only text on pure color background of the whole image. These algorithms show bad performance on pure picture images. An example of such an algorithm is Lempel-Ziv algorithm. Others algorithms are designed for pure picture images which do not have any text in the whole image. Alternatively, they have bad performance on pure text images. An example of them is JPEG. [11,12,13,14,16,18,23,25]

Since no single algorithm gives good results across all image types or applications, a third category of algorithms is needed to compress compound images with different content types: picture, graphics, and text. Although these algorithms are proposed to solve the drawbacks of the previous two categories, they do not reach the ideal situation. The algorithms of this category are further categorized into two groups: the Layered encoding and the block-based encoding. [2,1,23]

The Layered encoding methods separates the images into different layers and each layer is being encoded independently from the other layers. Most Layered encoding methods use the standard three layers Mixed Raster Content (MRC). As illustrated in Fig. 2, the three layers are: an image BackGround layer (BG), an image ForeGround layer (FG), and a binary mask layer. The mask classified the image components as either ForeGround or BackGround components. While the ForeGround components are coded by the ForeGround coder, the BackGround components are coded by the BackGround coder. Examples of this group of methods are LuraDocument and DjVu techniques. [2,6,14,7,9,4,10,15,11,12]

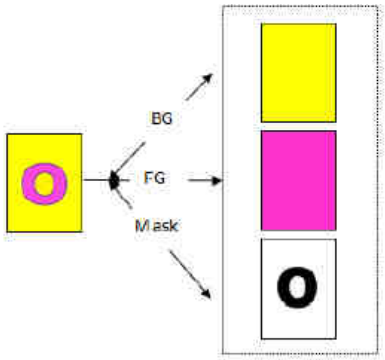

Figure 2: Example of Mixed Raster Content Layers

However, the Layered encoding methods still have some drawbacks. The complexity of layer generation is high, that makes it unsuitable for many embedded and real time applications [14,13,12,11]. These techniques tend to classify the text in the image as the ForeGround and all other details as the BackGround. Binary representation of the mask layer, that encodes the text and the graphics contents, tends to distort some fine document details, such as text edges and thin lines [9,6,7]. Although ForeGround and BackGround layers may not be used, they should also be coded; this adds some inefficiency [12,11,14,6]. Unfortunately, some foreground components may be classified as belonging to the background layer [4,9,12,11,14,6]. By contrast, some background components may be classified as belonging to the ForeGround layer [4,9,12,11,14,6].

Moreover, layer based approaches work well on simple compound images. But when the content is very complex, they show poor performance. For example, it is difficult to separate text from backgrounds when the text overlaps with background or the text has surrounding shadow. [11,12,15,6,14,7]

The block-based approaches, which are generally used for their low complexity, classify the compound image into blocks of different types. Then each type is compressed individually with the most off-the-shelf appropriate encoder technique. Although these methods give better results than the previous group, there are still some drawbacks. In case of strong edges in the textual area, they lead to hybrid blocks. These hybrid blocks contain mixed text and pictures that cannot be handled effectively. Even if the block contains a boundary between two regions, all of its pixels are classified in the same manner and given the same label. Although the complexity is lower than Layered encoding techniques, both the classification and compression algorithms of block-based encoding still have high calculation complexity, which makes them not suitable for real time applications. [12,11,13,17,23] Furthermore, the block-based segmentation approaches can be further divided into two groups: variable-size and fixed-size blocks [2,23]. This paper indeed use the equal-size-square blocks.

Accordingly, there is still much room for improving existing algorithms or coming up with new effective algorithms and techniques which is described in this research paper. However, there is a need for an effective way to classify image components and to compress its content.

III. THE TECHNIQUE DESCRIPTION

The proposed technique divides the scanned image into equal-size-square blocks and compresses them in a way that can

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication Pvt. Ltd.

restore the blocks again such that each and every piece of data that was initially in the blocks stays after the document is decompressed. It works in a sequence of five phases: preprocessing phase, image segmentation and classification phase, rearrangement phase, merging phase, and compression phase. These phases are illustrated through the flowchart of Fig. 3 which they form the backbone framework for the proposed technique. Since each and every bit is returned back to its original form after the document is decompression, this proposed algorithm is a lossless compression technique [2,23,25]. Therefore, this algorithm is suitable to be used where losing data or monetary information could represent an issue [25].

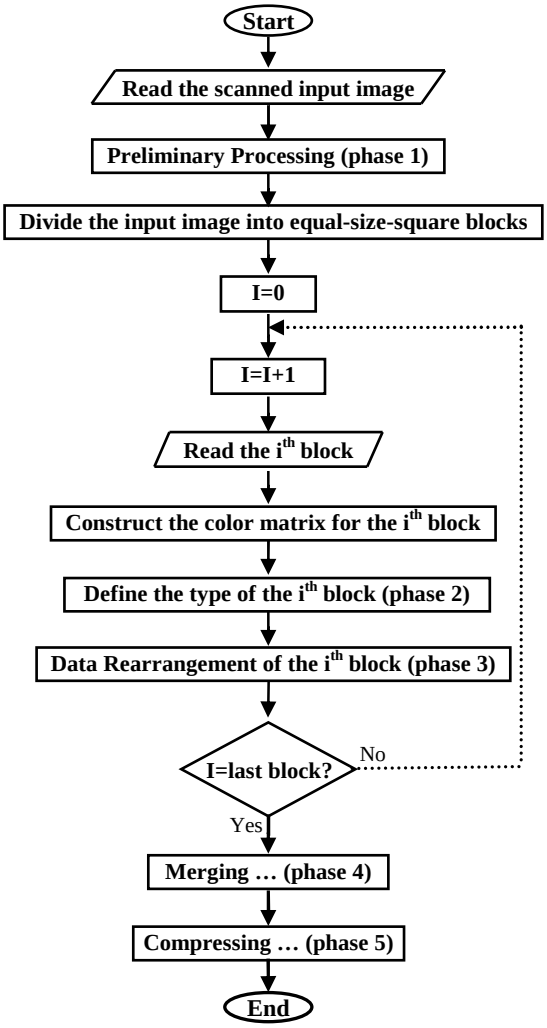

Figure 3: Proposed Technique Flowchart

A. Phase I: Data Perpetration

The original data set is subjected to a number of preliminary processing steps in order to make it operates accurately and usable by the next phase. Therefore, this stage determines the success of this technique. This includes data collection and partitioning, pre-processing, post-processing (i.e. De-normalization), and all the primarily operations that are used for reducing noise or variations inside the scanned image.

B. Phase II: Assigning Labels

The image is divided into equal-size-square blocks. A matrix of an RGB color map of each block is generated. This matrix represents the colors and their frequencies. However, colors with low frequency may be considered as noise. As a consequence these low frequency colors will be eliminated. Each block is assigned a label or a type. This assignation is based on the analysis of the distribution and number of colors; such that pixels with the same label share certain characteristics [18,19,20]. Typically, all blocks that make up of the same number of colors are given the same label or type. Based on this, there are seven types:

1. Type "**A**" represents the blocks that contain only one color, considered as background. These blocks represent, in general, the background of a document image which is a large expanse of a single color.
2. Type "**B**" represents the blocks that contain two colors. The blocks of this category usually represent the text regions.
3. Type "**C**" represents the blocks that contain three or four colors.
4. Type "**D**" represents the blocks that contain from 5 to 16 colors.
   Practically, the last two categories "**C**" and "**D**" represent mainly the drawing parts of the documents where we find generally the graphs, charts, and curves.
5. Type "**E**" represents the grey blocks that contain from 17 to 256 grey colors. These blocks are mainly the grey part of the image.
6. Type "**F**" represents the blocks containing from 17 to 256 RGB colors. The blocks of this type usually represent the picture regions.
7. Type "**X**" represents all the other cases where each block contains more than 256 colors. These blocks represent in general the millions of colors pictures found in the images.

For the scanned image, N, let the numbers of blocks for the types "A" , "B", "C", "D", "E", "F", and "X" are: NA , NB , NC , ND , NE , NF , and NX , respectively. Then, the total number of blocks is defined as:

$$N = N_A + N_B + N_C + N_D + N_E + N_F + N_X \qquad (1)$$

C. Phase III: Tables Forming

The eventual goal of this phase is to get a new representation of each block. The new generated data of each block is based on the content of each block. The output of this phase is a table where each row represents a single block of the original input image. The content of this table depends on each block type. To explore further details, this phase is described in the following subsections (1. through 3.) below:

1. Types "C", "D", "E" and "F"

This phase for these types depends on storing the detected colors within each block inside a special dictionary constructed specifically at the level of that block. Then, rather than storing the colors, the reference pointer indexes are used. Each reference pointer points out to a specific color inside this dictionary. These reference pointers are typically implemented by means of a Lookup Dictionary Table (LUD). Each pointer is used as an indication of where to decompress the original block. The numbers of needed bits for each pointer are 1,2,4,8 for the types "B", "C", "D", and "F", respectively.

At the decoder side and through the decompression process, when the computer read the compressed file and encounters a pointer, it interprets that pointer by retrieving the corresponding color from its place in

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication Pvt. Ltd.

the dictionary index of that block. So the original image is retrieved up to the last bit.

Since type "B" has two color layers, the dictionary contains six cells one cell for every basic color component of the RGB color model. Fig. 4 shows the representation of the data structure of type "B". The blocks in Fig. 4 are represented by the address (I, J) for each block, the 2-color dictionary, called BackGround (BG) and ForeGround (FG) colors, and only one bit for each individual pixel to indicate whether it is assigned to either the BackGround or the ForeGround colors. This individual bit is set to either zero if it belongs to the BackGround color or one if it belongs to the ForeGround color.

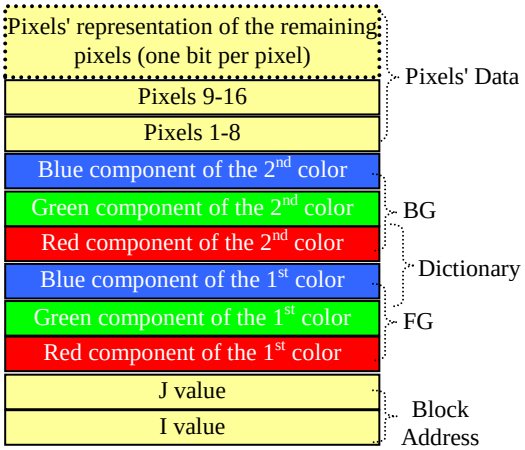

Figure 4: Data Structure for Type "B"

The three types, "C", "D", and "F", are similar to type "B". Fig. 5 illustrates the representation of data in these types. The following points should be noted:

- Since every color has three RGB components, the dictionary of type "C" has 4 * 3 = 12 cells (bytes). In view of that, each block is represented by the pair (I, J) where I and J represent the column and row numbers respectively; as well as the 4-color dictionary, and two-bit reference pointer for each individual pixel to designate a specific color from the four colors of the dictionary. The value $(00)_2$ points out to the first row in the color map, the value $(01)_2$ points out to the second row, the value $(10)_2$ points out to the third row and the value $(11)_2$, which is $(3)_{10}$, points out to the last row. In case of there are only three colors, the fourth color is assumed to be null.
- As discussed beforehand, the dictionary of type "D" blocks has 16 * 3 = 48 cells. Once more, each block is represented by the pair (I, J), the 16-color dictionary, and four-bit reference pointer for each individual pixel to designate a specific color from the sixteen colors of the dictionary. The value $(0000)_2$ points out to the first color in the dictionary, the value $(0001)_2$ points out to the second color and so on up to the value $(1111)_2$, which points out to the last color. In the special dictionary, if there are colors less than 16 and more than 4, they are fulfilled to 16 colors using null values.
- The dictionary of type "F" blocks has 256 x 3 = 768 cells (bytes). Like other types, if there are colors less than 256 and more than 17, they are completed to 256 colors using null values. Each block is represented by the pair (I, J), the 256-color dictionary, and eight-bit reference pointer for each individual pixel to designate a specific color from the 256 colors of the dictionary. Thus, the value $(00000000)_2$ points out to the first color, the value $(00000001)_2$ points out to the second color and so on up to the value $(11111111)_2$, which points out to the last color.

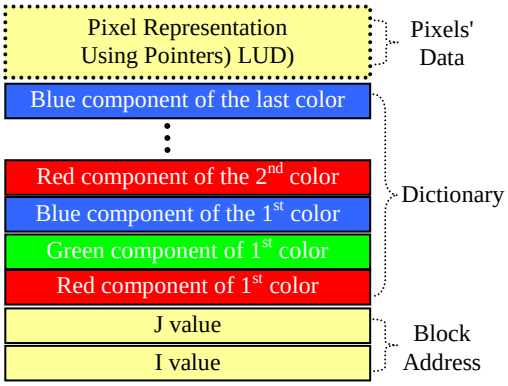

Figure 5: Data Structure for Types "C", "D", and "F"

2. Type "A"

Since type "A" blocks have only one color, the dictionary contains only three cells, one for every basic color component of the single RGB color. Rather than saving the same information for every individual pixel that makes up the BackGround, this approach stores the color data for the BackGround color only once to refer to all pixels of that block. Fig. 6 illustrates how the data is constructed in this type.

For this type, each block is represented by its address (I, J), and the three RGB components of its unique color. As the image is equal-size-square blocks, the size information, "blocklength", is only stored once at the first location of the compressed file of this type.

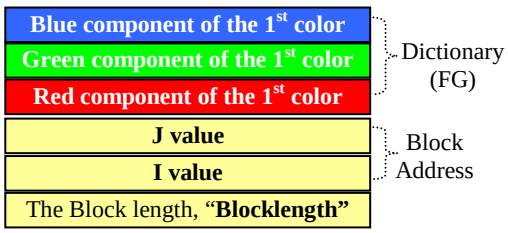

Figure 6: Data Structure for Type "A"

3. Types "E" and "X"

A block is obviously identified as grey if the values of the three basic RGB components in all pixels of the block are almost equal. Rather than repeating the same information for the three repeated RGB color components, one component is enough to represent the other two components. The red component is therefore used to represent the other two components.

Accordingly, neither the special dictionary, nor the reference pointers (LUD) are needed for type "E" blocks. Rather, the actual red component of the original block is selected and directly stored as it is without any reshaping or rearrangement. Fig. 7 illustrates how the data is constructed in this type of blocks. Each block is represented by its address (I, J) and the actual red component of its pixels, where each individual pixel requires a single byte.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication Pvt. Ltd.

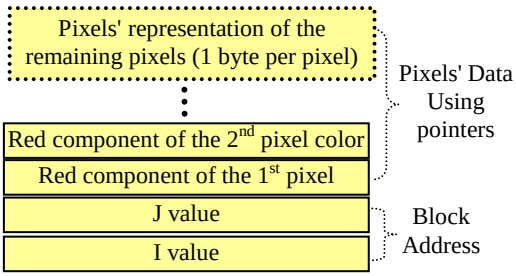

Figure 7: Data Structure for Type "E"

Type "X" is like type "E" but all the three basic RGB components of the original block are stored while the red component is only stored in type "E". The representation of these blocks is saved by storing the address (I, J) of the block and the actual pixels' data, where each individual pixel requires three bytes. Fig. 8 shows the representation of this type of blocks.

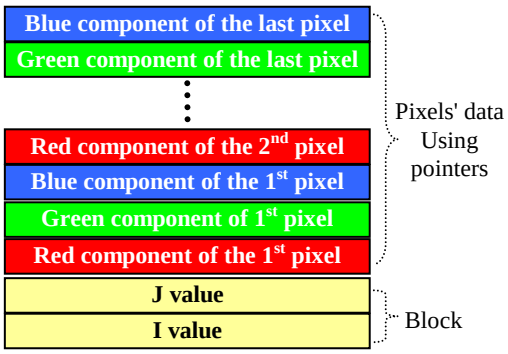

Figure 8: Data Structure for Type "X"

D. *Blocks Merging*

The merging phase aims to put together the adjacent equal-type blocks that have the same dictionary of colors into a larger arrangement of blocks to form higher-level regions. However, the blocks belonging to the same type don't necessary have the same colors, but they may have the same number of colors. As in Fig. 9, block neighborhoods can be defined in terms of one of the followings:

- 4-connectivity: in which the two blocks share a common side.
- 8-connectivity: in which the two blocks share either a common side or a common corner.

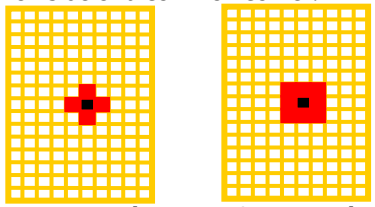

Figure 9: 4-Connected & 8-Connected neighbor Blocks

E. *Compression*

This is last phase in which every region (blocks of similar features) is compressed separately using the most off-the-shelf appropriate compression technique.

IV. THE ALGORITHMS

This proposed technique consists of two algorithms: Image Color Statistic and Color Counts Block-Based Segmentation.

A. *The First Algorithm*

Algorithm 1 is designed to generate a Color Statistic Table (CST) for the colors and their frequencies to either the whole image or one of its blocks. If the pixels of an input block (I, J) of "blocklength x blocklength" in size and its pixels are distributed among "n" 3-component colors, then Table 1 represents the output of this algorithm:

Table 1: Output of Algorithm 1

Red	Green	Blue	Frequency
R_{001}	G_{001}	B_{001}	F_{001}
R_{002}	G_{002}	B_{002}	F_{002}
:::::	:::::	:::::	:::::
R_{i-1}	G_{i-1}	B_{i-1}	F_{i-1}
R_i	G_i	B_i	F_i
:::::	:::::	:::::	:::::
R_{n-1}	G_{n-1}	B_{n-1}	F_{n-1}
R_n	G_n	B_n	F_n
Total			$Blocklength^2$

This table is arranged in descending order according to the last column, "Frequency", from "F001" to "Fn". At the beginning of this algorithm, an empty 4-column table is created. As the image file is read, this table is altered whenever a new color is encountered. If the encountered color is already in the table, its corresponding frequency is increased by one. Otherwise, a new row corresponding to this color is created with a frequency equals to one.

Algorithm 1: Image Color Statistic.

Description: This algorithm is designed to build a statistic about the detected colors and their frequencies that are found within every block. This statistic represents the color map or the dictionary of colors.

Input: Either an MxNx3-size BMP image or one of its blocks.

Output: a colour statistic table (CST) of four columns; three of them correspond to the three basic RGB components of each colour and the last one corresponds to the frequency of that colour. Every detected colour has one row.

Method:

1) *Initialization: construct an empty table "CST" of 4 columns.*
2) ***Read*** *the input image pixels from left to right and top to bottom.*
3) ***Repeat*** *for each individual pixel of the input file:*
   If *the three basic RGB components are in "CST"* ***Then***
   Add 1 to the frequency that corresponds to that color.
   Else
   Insert this color in the table "CST" with a frequency equals to one.
   End If

B. *The Second Algorithm*

Algorithm 2 represents the main steps of the technique discussed in this paper.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication Pvt. Ltd.

Algorithm 2: Color Counts Block-Based Segmentation.

Description: Through this algorithm, the bitmap table of the original image is divided into seven types according to the number of colors that are used inside.

Input: Any BMP image of size MxNx3 that represents a scanned document.

Output: Compressed Image File.

Method:

1. *Initialization: create seven empty tables correspond to the seven types. Each* table is of unsigned integer type with 8-bit (uint8) length.
2. *Preliminary processing for reducing noise or variations inside the scanned image.*
3. *Divide the image into equal-size-square blocks.*
4. *For each block:*
   a) *Using* Algorithm 1, *construct the* color statistic table "CST".
   b) *Check the colors* frequencies *of the previous table, "CST". Practically, colors with low frequency may be considered as noises and then eliminated.*
   c) *Determine the type of the block as:*

$$\begin{cases} \textbf{A}: \text{if the block contains one color.} \\ \textbf{B}: \text{if the block contains two colors .} \\ \textbf{C}: \text{if the block contains three or four colors.} \\ \textbf{D}: \text{if the block contains 5 - 16 colors.} \\ \textbf{E}: \text{if the block contains 17 - 256 grey colors.} \\ \textbf{F}: \text{if the block contains 17 - 256 RGB colors.} \\ \textbf{X}: \text{otherwise.} \end{cases}$$

If *the block type is "**A**"* ***Then***

*Create a new row in the table "**A**" for this block. This row contains:*

- *The block length, "**Blocklength**".*
- *The address of the block: "I" and "J".*
- *The values of the three RGB components of the single detected color of the block.*

Else If *the block type is "**B**"* ***Then***

*Create a new row in the table "**B**". This row contains:*

- *The address of the block: "I" and "J".*
- *A special dictionary for the two detected colors.*
- *1-bit-reference-pointer index to designate one of the two colors of the dictionary; using zero for the pixels having the first color and one for the second color. As a result, one byte can hold the information of 8 pixels.*

Else If *the block type is "**C**"* ***Then***

*Create a new row in the table "**C**" for this block. This row contains:*

- *The address of the block: "I" and "J".*
- *A special dictionary for the four detected colors. In the case of three colors, the fourth color is assumed to be null.*
- *2-bit-reference-pointer index to designate a specific color from the four colors of the stored dictionary. One LUD value from the binary list {00, 01, 10, 11} is used for each pixel to point out to its color. One byte can therefore hold the information of 4 pixels.*

Else If *the block type is "**D**"* ***Then***

*Create a new row in the table "**D**" for this block. This row contains:*

- *The address of the block: "I" and "J".*
- *A special dictionary for the 16 detected colors. If there are colors less than 16 and more than 4, they are fulfilled to 16 colors using null values.*
- *4-bit-reference-pointer index to designate a specific color from the sixteen colors of the stored dictionary. Every 2 pixels require one byte.*

Else If *the block type is "**E**"* ***Then***

*Create a new row in the table "**E**" for this block. This row contains:*

- *The address of the block: "I" and "J".*
  - *For each pixel of the block, store only its red color component. Each pixel, in turn, requires a single byte.*

Else If *the block type is "**F**"* ***Then***

*Create a new row in the table "**F**" for this block. This row contains:*

- *The address of the block: "I" and "J".*
- *A special dictionary for the 256 detected colors. If there are colors less than 256 and more than 17, they are fulfilled to 256 colors using null values.*
- *8-bit-reference-pointer index to designate a specific color from the 256 colors of the stored dictionary. Obviously, each individual pixel requires a single byte.*

Else If *the block type is "**X**" Then*

*Create a new row in the table "**X**". This row contains:*

- *The address of the block: "I" and "J".*
- *The pixels' data that are detected in that block. Each individual pixel requires three bytes.*

End If

5. *Do Merging.*
6. *Do compression.*
7. *Combine all the seven tables into one table.*

V. EXPERIMENTAL RESULTS AND ANALYSIS

Since a reliable system should be experimented on a large number of samples, a special database that includes different images was created [24,2]. As illustrated in Table 2, this database contains 1821 24-bit-RGB-bitmap images of different resolutions distributed among five classes. Using MATLAB® version 9.0 release R2016a environments, the proposed technique has been implemented on this source database.

Table 2: Classes of the Special Database.

Image Class	No. of images
Pure Background	142
Pure Text	320
Pure Graph	350
Pure Picture	370
Mixed Image	639

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication Pvt. Ltd.

Total number of images	1821

The Saving Ratio Percentage (SRP) is used as a measure to evaluate the performances of the proposed technique. It is defined as follows:

$$SRP = \left[1 - \frac{\text{compressed image size}}{\text{original image size}}\right] \times 100\% \qquad (2)$$

This measure depends on the image content that leads to the distribution of the original table on the seven types. Since the compression ratios are dependent on the type of each block which can in turn affected the SRP, the best case is obviously seen whenever all the blocks are of type "A", which means that the entire image is a BackGround of one color. The next best case is whenever all the blocks are of type "B", and so on. However, the worst case is whenever all the blocks are of type "X", which means that the entire image is a picture. In this case, the encoding of this approach is not appropriate and the system will be flexible to cancel the encoding process and use another proper encoder.

In case of blocks of type "A", there is a need to store an additional one-byte cell to represent the block length, "blocklength". Moreover, since the blocks of type "A" have single color, which is classified as background, there is no need to store more data about pixels contained in the block. Hence, neither the special dictionary, nor the reference pointers (LUD) are needed, only six bytes are required to store the whole block no matter how much its size. However, this solves one of the drawbacks of Layered encoding mentioned at Section 2. The SRP per block of this type is given by the equation:

$$SRP(A) = \left[1 - \frac{1+2+3}{3*\ \text{blocklength}^2}\right] *100\% \qquad (3)$$

$$= \left[1 - \frac{2}{\text{blocklength}^2}\right] *100\%$$

Where:

1. Number "1" of the numerator means that one byte is required to store the "blocklength".
2. Number "2" of the numerator means that two bytes are required to store the address of each block, one byte for the I^{th} address and one byte for the J^{th} address.
3. Number "3", in the numerator and the denominator, means that there are three basic RGB color components.
4. "blocklength" stands for the block length and is given in pixels. Since the image is divided into equal-size-square blocks, the size of each block is "blocklength^2". Thus, the denominator stands for the size of the original block before compression.

For the blocks of types "B", "C", "D", and "F", the compression is done by storing pointers for the special dictionary. However, the SRP per block is given by the equation:

$$SRP(B\,|\,C\,|\,D\,|\,F) = \left[1 - \frac{2+3*2^b + \frac{\text{blocklength}^2}{8/b}}{3*\ \text{blocklength}^2}\right] *100\% \quad (4)$$

Where:

1. "2^b" stands for the number of bits required to store the reference pointers (LUD); "**b**" is 1,2,4,8 for the types "**B**", "**C**", "**D**", "**F**", respectively.
2. The number of pixels that can be stored in a single byte is (8 / b), which gives 8,4,2,1 for the types "**B**", "**C**", "**D**", "**F**", respectively. So the expression (blocklength^2/ (8/b)) is used to determine the number of bytes that are required to store the data of each block.
3. The rest of this equation is like equation 3.

In type "**E**", the SRP per block is given by the equation:

$$SRP(E) = \left[1 - \frac{2 + \text{blocklength}^2}{3*\text{blocklength}^2}\right] *100\% \qquad (5)$$

The major difference between the last two equations, 4 and 5, is that the dictionary is not needed in equations 5 and therefore is cancelled.

For the blocks of type "X", the SRP per block is given by the equation:

$$SRP(X) = \left[1 - \frac{2+3*\text{blocklength}^2}{3*\ \text{blocklength}^2}\right] *100\% \qquad (6)$$

Consequently, a summary of all types is provided in Table 3. Table 4 illustrates these remarks and results for different block types and lengths, "blocklength". However, the strikethrough bolded cells are introduced in this table, Table 4, to show the cases where the compression ratio is inappropriate due to the fact that:

*If the block is of type "**X**", the actual data is saved, as it is, in conjunction with the block address, (I, J).*

Table 3: Comparison of the seven types

Type	Block Length	Block Address	Dictionary Size	LUD
"A"	✓	✓	1*3=3	×
"B"	×	✓	2*3= 6	✓
"C"	×	✓	4*3=12	✓
"D"	×	✓	16*3=48	✓
"E"	×	✓	Null=0	×
"F"	×	✓	256*3=768	✓
"X"	×	✓	Null=0	×

Fig. 10 shows the possible ranges (minimum and maximum) of SRP for these seven types. Fig. 11 shows the evolution of the SRP in function of the block length. However, the SRP is improved while the size of the block is increased.

By analyzing these results, we found that the block length, "blocklength", affects moderately the SRP. When the block length is increased, SRP increases, too. Moreover, in case of type "X" blocks, this technique gives negative results. As a result, this technique should be dynamic enough, so that when the decomposition using this technique is not appropriate for a particular image, the system should be flexible enough to cancel the operation and use another compressor.

As a final point, if the logical operation "XOR" is applied between the encoded input image and the decoded output image, the result is zero. Therefore, the output quality of this phase is 100% which, in turn, leads to the conclusion that this technique is a lossless one [2,23,25].

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication Pvt. Ltd.

Table 4: Saving Rates Percentage per Block for the different types of blocks

Block length	Type "A"	Type "B"	Type "C"	Type "D"	Type "E"	Type "F"	Type "X"
020	99.50	95.17	90.50	79.17	66.50	02.50	~~-0.17~~
030	99.78	95.54	91.15	81.48	66.59	38.15	~~-0.07~~
040	99.88	95.67	91.38	82.29	66.63	50.63	~~-0.04~~
050	99.92	95.73	91.48	82.67	66.64	56.40	~~-0.03~~
060	99.94	95.76	91.54	82.87	66.65	59.54	~~-0.02~~
070	99.96	95.78	91.57	82.99	66.65	61.43	~~-0.01~~
080	99.97	95.79	91.59	83.07	66.66	62.66	~~-0.01~~
090	99.98	95.80	91.61	83.13	66.66	63.50	~~-0.01~~
100	99.98	95.81	91.62	83.17	66.66	64.10	~~-0.01~~
110	99.98	95.81	91.63	83.20	66.66	64.55	~~-0.01~~
120	99.99	95.81	91.63	83.22	66.66	64.88	0.00
130	99.99	95.82	91.64	83.23	66.66	65.15	0.00
140	99.99	95.82	91.64	83.25	66.66	65.36	0.00
150	99.99	95.82	91.65	83.26	66.66	65.53	0.00
160	99.99	95.82	91.65	83.27	66.66	65.66	0.00
170	99.99	95.82	91.65	83.28	66.66	65.78	0.00
180	99.99	95.83	91.65	83.28	66.66	65.87	0.00
190	99.99	95.83	91.65	83.29	66.66	65.96	0.00
200	99.99	95.83	91.66	83.29	66.67	66.03	0.00
210	99.99	95.83	91.66	83.30	66.67	66.08	0.00
220	99.99	95.83	91.66	83.30	66.67	66.14	0.00
230	99.99	95.83	91.66	83.30	66.67	66.18	0.00

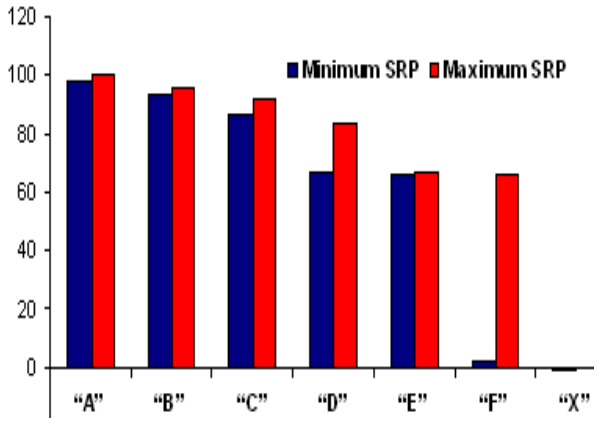

Figure 10: Possible SRP ranges per block type

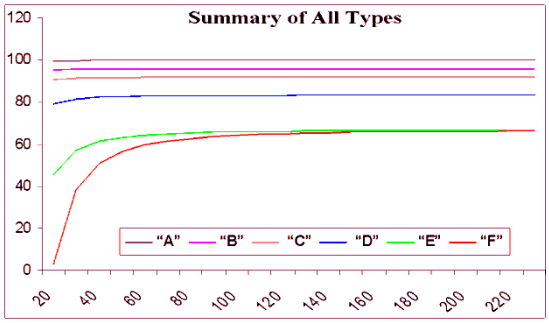

Figure 11: Evolution of the SRP in function of the block length

VI. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a five-phase image segmentation and compression scheme based on the number of colors detected in each region of the image. This proposed technique benefit from the use of information regarding the number of detected colors in each region of the scanned image to be segmented. It aims to segment the original image into consistent and homogeneous non-overlapping regions and then, each region is compressed by using the most off-the-shelf appropriate compression technique. This approach combines different compression concepts in order to achieve better compression of the scanned documents. To test the performance of the proposed algorithm, a special database was created and for security motivation, an integrate encryption can be applied at the encoder side and decryption at the decoder side; this help in creating secure data storage for the scanned document.

ACKNOWLEDGMENT

This work is encouraged by the World Islamic Science and Education University (WISE), Amman Jordan, and the Northern Border University Arar, Kingdom of Saudi Arabia..

REFERENCES

1. Acharyya, M. and Kundu, M.K. (2002). "Document Image Segmentation Using Wavelet Scale-Space Features", IEEE Transactions Circuits Syst. Video Technol., Volume 12, Issue 12, pp. 1117–1127.
2. Nidhal Kamel Taha El Omari. (2008). "A Hybrid Approach for Segmentation and Compression of Compound Images", PhD Dissertation, the Arab Academy for Banking and Financial Sciences.
3. Nidhal Kamel Taha El-Omari and Arafat A. Awajan. (December 20-22, 2009). "Document Image Segmentation and Compression Using Artificial Neural Network Based Technique", International Conference on Information and Communication Systems (ICICS09), pp. 320-324, Amman, Jordan.
4. Kai Uwe Barthel et al., (January 2000). "New Technology for Raster Document Image Compression", SPIE. The International Society for Optical Engineering, Volume 3967, pp. 286-290, San Jose, CA.
5. Patrice Y. Simard et al., (March 23-25, 2004). "A Foreground/Background Separation Algorithm for Image Compression", IEEE Data Compression Conference (DCC), pp. 498–507, Snowbird, UT, USA.
6. Ricardo L. de Queiroz et al., (February 1999). "Mixed Raster Content (MRC) Model for Compound Image Compression", SPIE the International Society for Optical Engineering, Volume 3653, pp. 1106-1117.
7. Ricardo L. de Queiroz. (October 8-11, 2006). "Pre-Processing for MRC Layers of Scanned Images", Proceedings of the International Conference on Image Processing (ICIP), Atlanta, Georgia, USA, pp. 3093–3096.
8. Lihong Zheng and Xiangjian He. (2004). "Edge Detection Based on Modified BP Algorithm of ANN", Conferences in Research and Practice in

Information Technology (RPIT), Volume 36, pp. 119–122.
9. Guotong Feng and Charles A. Bouman. (October 2006). “High Quality MRC Document Coding”, IEEE Transactions Image Processing, Volume 15, Issue 10, pp. 3152-3169.
10. Leon Bottou, Patrick Haffner et al., (July 1998). “High Quality Document Image Compression with DjVu”, Journal of Electronic Imaging, Volume 07, Issue 3, pp. 410-425.
11. Wenpeng Ding et al., (January 30, 2007). “Rate-Distortion Optimized Color Quantization for Compound Image Compression”, Visual Communications and Image Processing Conference, SPIE Proceedings, Volume 6508, pp. 65082Q1-65082Q9, San Jose, CA, USA.
12. Tony Lin and Pengwei Hao. (August 2005). “Compound Image Compression for Real Time Computer Screen Image Transmission”, IEEE Transactions on Image Processing, Volume 14, Issue 8, pp. 993-1005.
13. Wenpeng Ding et al., (2006). “Block-based Fast Compression for Compound Images”, ICME, paper ID 1722, pp. 809–812.
14. Debargha Mukherjee et al., (June 2002). “JPEG2000-Matched MRC Compression of Compound Documents”, IEEE International Conference on Image Processing (ICIP), Volume 3, pp. 225-228.
15. Cheng H. and Bouman C. A. (April 2001). “Document Compression Using Rate-Distortion Optimized Segmentation”, Journal of Electronic Imaging, Volume 10, Issue 2, pp. 460–474.
16. Nidhal Kamel Taha El-Omari et al., (2012). “Innoviate Text-Image Compression Technique”, European Journal of Scientific Research, © EuroJournals Publishing Inc., Volume 88, Issue 4, pp. 603-616.
17. Gnana King, G.R.1 and Seldev Christopher, C.2. (2014). “Improved block based segmentation algorithm for compression of compound images”, Journal of Intelligent & Fuzzy Systems, Volume 27, Issue 6, pp. 3213-3225.
18. Qindong Sun et al., (2015). “A Method of Image Segmentation based on the JPEG File Stream”, Journal of Computational Methods in Sciences & Engineering, Volume 15, Issue 3, pp. 467-475.
19. Bo Chen et al., (June 2015). “A new image segmentation model with local statistical characters based on variance minimization”, Applied Mathematical Modelling, Volume 39, Issue 12, pp. 3227-3235.
20. Gagan Jindal and Sikander Singh Cheema, (2016), “Review Paper of Segmentation of Natural Images using HSL Color Space Based on K-Mean Clustering”, International Journal of Innovations & Advancement in Computer Science, Volume 5, Issue 7, pp. 26-29.
21. Zhanjiang Zhi et al., (2016), “Two-Stage Image Segmentation Scheme Based on Inexact Alternating Direction Method”, Numer. Math. Theor. Meth. Appl., Volume 9, Issue 3, pp. 451-469.
22. Haifeng Sima et al., (2016), “Objectness Supervised Merging Algorithm for Color Image Segmentation”, Mathematical Problems in Engineering, Volume 2016, Article ID 3180357, pp. 1-11.
23. S.Thayammal, and D.Selvathi., (2013), “A Review On Segmentation Based Image Compression Techniques”, Journal of Engineering Science and Technology Review, Volume 6, Issue 3, pp. 134-140.
24. Ian Sommerville, (2015), “Software Engineering”, 10th Edition, Pearson Education, Inc., ISBN-13: 978-0133943030, New York, USA.
25. Er. Kuldeep Kaur et al., (2016), “Comparative Analysis of Compression Techniques: A Survey”, International Research Journal of Engineering and Technology (IRJET), Volume 03, Issue: 04, pp. 1042-1046.

Nidhal Kamel Taha El-Omari He obtained B.Sc. 2005. In 2008, he obtained his Ph.D. in Computer Information Systems in Image Processing from Arab Academy for Banking and Financial Science (AABFS), Amman-Jordan.
He joined the Information Technology Directorate of the Jordanian Ministry of Defense in 1986 and retired in 2009. During 1986-1989, he worked as software developer. During 1989-1995, he was systems analyst and systems engineer. During 1995-2004, he was the chief of IT instructors, head of many sections, and the project manager of many computer projects. During 2004-2009, he chaired a number of IT-related departments including: System Follow up Department, Technical Support Department, and Automation Department. During 2009-2011, he was the director of the Computer Center and the Chair of the Department of Computer Science and Basic Science at Faculty of IT at WISE University in Jordan. During 2009-2014, he was an assistant Professor.
Since 2015, he is an Associate Professor and the head of Department of Software Engineering at the Faculty of IT, WISE University. His research interests include: Image Compression & Segmentation, Artificial Neural Network (ANN), Artificial Bees Colony System (ABC), Wireless Networks, and Programming Languages and Methodologies for building correct, secure and efficient software. Dr. El-Omari authored/co-authored two computer books and more than twenty five research papers in international journals and conferences. nidhal.omari@wise.edu.jo; omari_nidhal@yahoo.com;

Ahmad H. Al-Omari, received the B. Sc. In Computer Science in 1985, M of Computer Science in 2001, and he received his Ph.D. in in Computer Information Systems in 2004, he had long working experience in the field of information technology in many working areas like, systems analysis, programming, tendering, network design, management and trainer. After he joined the academic area, he was the acting dean, dean, department head in the faculty of Information Technology FIT, Applied Science University. He supervised many master students, he participated in master examination and discussion committees, and he also published more than 13 research work in his field.

Al-Ibrahim was born in Jordan on june 1th, 1966. He obtained the B.Sc. in Computer Science in 1988 from Yarmouk University, Irbid, Jordan. In 2008, he obtained his Ph.D. in Computer Information Systems in Image Processing from Arab Academy for Banking and Financial Science (AABFS), Amman, Jordan. He joined the Information Technology Directorate of the Arab Potash Company in 1991 and retired in 2009. During 1991-1996, he worked as software developer. During 1996-1999, he was systems analyst and systems engineer. During 1999-2005, he was the chief of IT Development and Technical Support unit (head of many sections, and the project manager of many computer projects). During 2006-2010, he was Human Resources Manager (HR Manager). During 2014-2015 he was Chair of the Department of Computer Science and Basic Science at Faculty of IT at WISE University in Jordan. Since 2010, he is an Assistant Professor at the Faculty of IT in Jadara University then from 2011-2016 Assistant Professor Then 2016 – Associate Professor in WISE University. His research interests include: Image compression & segmentation ,Artificial Intelligence (AI), Database(DB),e-Government , Information Storage and Retrieval (IR), Artificial Neural Networks-Based Decision Support System ,Strategic Information System, and programming languages and methodologies for building correct, secure and efficient software. Dr. Al-Ibrahim authored to computer books (1-Discrete Mathematics for IT Students and 2-Introduction of programming languages / Theory of Computation**)** and more than fifteen research papers in international journals and conferences. ali.alibrahim@wise.edu.jo; ali.alibrahim66@yahoo.com;.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication Pvt. Ltd.