INTRODUCING THE CONCEPT OF INFORMATION PIXELS AND THE SIPA (STORING INFORMATION PIXELS ADDRESSES) METHOD AS AN EFFICIENT MODEL FOR DOCUMENT STORAGE

Mohammad A. ALGhalayini

Computer and Information Unit Director
Vice Rectorate for Graduate Studies and Scientific Research
King Saud University, Riyadh, Saudi Arabia

ABSTRACT

Today, many institutions and organizations are facing serious problem due to the tremendously increasing size of documents, and this problem is further triggering the storage and retrieval problems due to the continuously growing space and efficiency requirements. This problem is becoming more complex with time and the increase in the size and number of documents in an organization. Therefore, there is a growing demand to address this problem. This demand and challenge can be met by developing a technique to enable specialized document imaging people to use when there is a need for storing documents images. Thus, we need special and efficient storage techniques for this type of information storage (IS) systems.

In this paper, we present an efficient storage technique for electronic documents. The proposed technique uses the Information Pixels concept to make the technique more efficient for certain image formats. In addition, we shall see how Storing Information Pixels Addresses (SIPA) method is an efficient method for document storage and as a result makes the document image storage relatively efficient for most image formats.

KEYWORDS

Information Pixels, Document Storage, Document Segmentation, Image Formats, SIPA Method

1. INTRODUCTION

In previous research paper [30], we have analyzed and concluded that pdfimage formatis one of the best types to store scanned documents. Our next step was to divide the KSU(King Saud University) document into segments and store only the segment that contains the necessary information, namely, the body of the document. This excludes the header and footers, date and logo. This has proved to be efficient with respect to storage and saves at least 25% storage space

[31]. In this research paper, we move a step further. We analyze the body segment of the document.
This is done by considering segment (5) of the KSU documents. In this segment, it is true with almost all documents that the shades are in black and white and there is no need for images to be stored neither as color images nor with higher memory usage, as is explained in previous papers.

The challenge is about how we can further reduce the storage size of the document. We should come up with a way that scrutinizes the document to a greater extent. This could be done by evaluating whether or not it is possible to make use of the fact that the whole document can be considered as an image, and since the information is in black, then the question should examine if we can store only the part of the segment that is black.

2. ANALYZING THE KSU DOCUMENT FULL BODY SEGMENT IN FINER DETAIL

We segmented the KSU Document sheet into several segments depending on vital data which we actually need to store. These segments can be classified based on its content. Figure (1) below shows the segmentation pattern.

It is the body of the document that contains the necessary information. As explained previously, it is necessary to analyze this part in a way by which we can incidentally reduce the size of the document by considering the segment as an image.

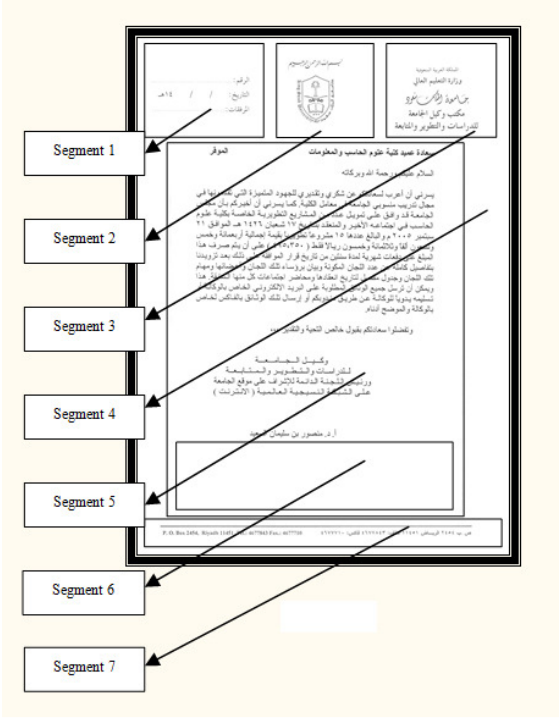

Figure (1)

An image can be considered to consist of various shades. If the images are black and white, then the image can be considered as a gray scale image that consists of (256) shades. When we further try to simplify this concept, we understand that the documents have written information

and therefore can just be considered as black and white. The images are now considered at the pixel level. The advantages of considering these images at the pixel level are:

- The pixel is the building block of an image and therefore every image can be observed in minute detail based on the pixel values.
- It is the pixel values that the computer uses to store images. The number of bits used to represent pixel values can be manipulated and hence the size of the image.
- The spatial dimensions of a pixel can be predefined. The smaller the pixel, the better the resolution and the larger the pixel, the lesser the resolution. In addition, smaller pixels in an image with specific dimensions mean more number of pixels and therefore more storage space. Larger pixels in an image mean lesser number of pixels and less storage space; therefore, storage space and image quality is a tradeoff.

One of the best application softwares that can be used to check and simulate our proposed concept on images is MATLAB. We are going to use MATLAB to analyze the outcome of our logic on these images.

3. INTRODUCING SIPA METHOD (STORING INFORMATION PIXELS ADDRESSES)

3.1 Concept of Pixel Based Segmentation

A typical KSU sheet is shown in Figure (1).We make note of a few facts: The information written in the KSU sheet is in black. It does not contain any images. We observe that when the images are stored in any format, there is lots of space used up by the white portion of the sheet, which, apparently, stores no information and may also lead to an increase in memory usage.

This portion is not of any importance to us; therefore, we need a way to only store the information, that is, nothing but the information pixels which in turn is the black portion in the image. We shall call each black pixel an information pixel.

The concept of Storing Information PixelsAddresses (SIPA) will help save memory space. This can be done by taking into account only the pixels that are black in colorwhich are the information pixels. It is important to consider the image as an ensemble of black and white shades of 256 types. This is obtained from the fact that every pixel is represented by 8unsigned bits in MATLAB. We have to observe the image on agrayscale.

Another point to be made note of is that MATLAB does not interpret images based on colors. It does so based on the intensity of pixels. If the intensity of the pixel is zero, then it is black. If the intensity of a pixel is 255 which is the maximum value of an 8-bit pixel, it is white.We use this concept for identifying colors in MATLAB using pixel intensities.

The MATLAB program below is for a jpeg format image. This program is used to store the addresses of information pixels.

```
% Provides information of the image
Imfinfo('C:\Users\ALGhalayini\Desktop\sample.jpg');
% Start counter to keep a tab of the time
Tic;
% Read the image in MATLAB
RGBimage=imread('C:\Users\ALGhalayini\Desktop\sample.jpg','jpg')
% Convert it to gray format
```

```
GRAYimage= rgb2gray(RGBimage);
% Display the image
Figure(1);
Imshow(GRAYimage);
% Find the dimensions of the image
[m n]=size(GRAYimage);
total_pixels=m*n;
% Store address of information pixels in Array1 and Array2
K =1;
forI =1:m
for j=1:n
ifGRAYimage( I ,j)<128
        Array1(1,k)= I ; Array2(1,k)=j;
K =k+1;
end
end
end
%Form N X 2 array that stores the address of information pixels
Array=[Array1' Array2'];
% End of time counter
Toc;
```

First, we execute the *imfinfo* command and make sure that the image format we are using is compatible withMATLAB. The image is then read into MATLAB using the *imread* command.

This command reads the image asaRGB image. The RGB color model is *additive* in nature. That is, when R,G and B light beams are added together, their light spectra add to make the final color's spectrum. Zero intensity for each component gives the darkest color (no light-considered *black*), and full intensity of each gives a white; the *quality* of this white depends on the nature of the primary light sources. When the intensities for all the components are the same, the result is a shade of gray, darker or lighter depending on the intensity. When the intensities are different, the result is a colorized. Since our image is black and white, we can use an image model that uses equal intensities of the 3 primary colors to produce shades between black and white. This is called the grayscale format.

3.2 Thresholding and Image Enhancement

Thresholding is the simplest form of image segmentation. It is a form of scaling the intensities in the image. It can be done on grayscale images.

During the thresholding process,individual pixels in an image are marked as “object” pixels if their value is greater than some threshold value (assuming an object to be brighter than the background) and as “background” pixels otherwise. Typically, an object pixel is given a value of “1” while a background pixel is given a value of “0.” Finally, a binary image is created by coloring each pixel white or black, depending on a pixel's label. Since the grayscale image has values from 0 to 255, we can consider the threshold value to be 127, halfway between the highest and lowest intensities. In this scale, pixel values above 127 are considered white and below 127 are considered black.This process of thresholding also enhances the image because it removes unwanted gray shades in the image formed during scanning.

Let *m* be the number of pixels row wise and *n* is the number of pixels column wise. MATLAB has the ability to handle images with specific formats. These formats are presented in Table (1) below.

The execution time is recorded in MATLAB for each format. The code for JPEG format is displayed above. Observe that there should be only small differences in the code. It is in the part where the image is read. MATLAB must be given information about the type of format of the image it is supposed to read. This can be accomplished by replacing the filename in the *imread* command.

No.	Image Type	Type meaning
1	BMP	Windows Bitmap
2	CUR	Cursor File
3	GIF	Graphics Interchange Format
4	HDF4	Hierarchical Data Format
5	ICO	Icon File
6	JPEG	Joint Photographic Experts Group
7	PBM	Portable Bitmap
8	PCX	Windows Paintbrush
9	PGM	Portable Graymap
10	PNG	Portable Network Graphics
11	PPM	Portable Pixmap
12	RAS	Sun Raster
13	TIFF	Tagged Image File Format
14	XWD	X Window Dump

Table (1)

4. APPLYING THE SIPA METHOD ON THE WHOLE KSU A4 DOCUMENT IMAGES

4.1 Program Algorithm

The program algorithm shown below is for the Whole KSU A4 document sheet.

1 - clear all; close all; clc;

- In this line, clear all:clears the memory and hence does not store values of variables used in any of the previous commands.
- Close all :closes the execution of the previous commands.
- Clc :Clears the screen.

2 - The image is then converted from RGB format to Gray scale format.
3 - The number of rows is taken to be m and the number of columns is taken to be n.
4 - The total number of pixels is m*n.
5 -forI =1:m

```
for j=1:n
ifGRAYimage(I ,j)<128
        Array1(1,k)= i ; Array2(1,k)=j;
K =k+1;
end
end
end
```

The for-loops help check row-wise, and then column-wise, the value of the pixel. If the value of the pixel is less than 128, it is an information pixel (is black) and its address is stored in an array. Both the row and column addresses are stored. The arrays, Array1 and Array2 are of type double.

6 -Array1 and Array2 are together combined to form Array that is a 2-dimensional array. The first column of the array represents the row address and the second column of the array represents the column address corresponding to each information pixel.

7 - Tic and Toc are commands that are used to record the time taken for the whole execution.

4.2 Execution Results and Analysis

Table (2) below represents the result of the analysis applied on the Whole KSU A4 page. It also shows the computed result using MATLAB. It is executed for four image format types :

- Jpeg
- Bmp
- TIFF
- Gif

The result for these four formats is tabulated. The first column in Table(2) indicates the image formats on which the MATLAB program have been executed. The second column indicates the pixels per inch(resolution). For every image type, we execute the code for 4 different pixel densities- 75, 100, 150, and 200. The pixel densities vary from less resolution with merely intelligible information to more resolution with better clarity. Third column shows the size of the original image in kilobytes corresponding to the image format in column 1 and pixel density in column 2. The fourth column shows the time in seconds taken to complete the execution. This process includes reading the image, locating and storing the address of information pixels. The fifth column indicates the total number of information pixels. The sixth column indicates the array dimension. The seventh column indicates the size of the resulting array. As indicated in the sixth column, the array has 2 dimensions, one that represents the row address and the other that represents the column address.

Example: If an array has the element [1790 , 34], it means that the 1790th row and 34th column is an information pixel. Since the number of pixels either row wise or column wise for even the least pixel density is greater than 255, the pixel addresses have to be represented with numbers greater than 255 at times and is therefore not of type integer whose range is from 0 to 255 but is of type double which indicates double precision and can go up to 65536.

The size of the array is calculated by using the formula :

Image Format	dpi	Original Image Size (in KB)	Execution Time (in s)	Number of Information pixels	Array Dimensions		Array Size (in KB)	Compression Ratio	Average Compression Ratio
jpeg	75	80.00	2.43	544,617	3,728	X 2	116.50	0.687	0.73
	100	112.00	2.17	981,618	7,719	X 2	120.60	0.929	
	150	240.00	10.50	2,208,015	21,849	X 2	341.38	0.703	
	200	384.00	31.22	3,926,472	41,548	X 2	649.18	0.592	
bmp	75	1,600.00	1.09	544,617	3,570	X 2	55.77	28.687	22.53
	100	2,880.00	2.38	981,618	7,551	X 2	117.98	24.412	
	150	6,480.00	11.23	2,208,015	21,633	X 2	338.02	19.171	
	200	11,504.00	32.35	3,926,472	41,214	X 2	643.97	17.864	
gif	75	144.00	1.89	544,617	5,928	X 2	92.62	1.555	1.23
	100	240.00	4.80	981,618	11,873	X 2	185.51	1.294	
	150	496.00	19.60	2,208,015	30,092	X 2	470.18	1.055	
	200	864.00	79.85	3,926,472	55,066	X 2	860.40	1.004	
tif	75	528.00	1.00	544,617	3,570	X 2	55.77	9.467	6.14
	100	704.00	2.13	981,618	7,551	X 2	117.98	5.967	
	150	1,616.00	10.31	2,208,015	21,633	X 2	338.02	4.781	
	200	2,800.00	25.61	3,926,472	41,214	X 2	643.97	4.348	

Table (2)

$$Size = \varepsilon * 2 * 8 / (1024) \quad \ldots\ldots \text{ Formula } (6 - 1)$$

ε indicates the array dimension. Since each pixel is associated with a row and column pixel address, each of which are of type double, there are 2 integers of type double associated with every address. Every pixel address has a size of 2*8 bytes because each number in the array is of type double which is 8 bytes. It has to be multiplied by the length of the array to get the whole array size. This is then divided by 1024 to get the size in KB. The eighth column gives the ratio of original image size to array size. If the value in this column is greater than 1, it means that the size of the array is smaller than the size of the image and using the concept of information pixels turns out to be successful. If the value is less than 1, it means that the size of the array is greater than the size of the image and the method fails in compressing the image further. The last column indicates the average percentage of compression for a specific format that averages over all pixel densities. The ratio of array to image compression is given by:

$$Ratio = Array\ Size / Image\ Size * 100$$

A few important observations that can be made from the table are:

- As the pixel density increases, the storage size of the image increases as a result of more pixels in the image.
- As the number of pixels and storage size increases, the execution time increases because the computational complexity increases.
- The number of elements in the array used to store addresses of information pixels increases with increase in number of pixels or increase in dpi. This is because, as the dpi increases, pixels become smaller and represents smaller portions of the image.

For the JPEG image format, we observe that the applying SIPA method does not turn out to be successful. The average compression is 0.73 which means that the size of the array is 1/0.73 times the size of the image, that is, the size of the array is larger than the size of the image. Therefore, using SIPA method for JPEG image formats is not a good idea. Also, we can see that as the dpi increases, the compression ratio gets smaller. For 75 dpi, the array is 1/0.687 times the original image while for 200 dpi, the array is 1/0.592 times the original image. This shows that the inefficiency of this technique increases with increase in pixel density.

For the BMP image format, we observe that applying SIPA method of storing only information pixels is highly efficient. The average compression is 22.53 which means that the size of the array is just about 1/22.53 times the size of the original image, that is, the size of the array is very small compared to the size of the image. Therefore, using this scheme for BMP image formats is a good idea. Also, we can see that as the dpi increases, the compression ratio decreases. For 75 dpi, the compression ratio is 28.687 while for 200 dpi, the compression ratio is 17.864. This says that the efficiency of this technique decreases with increase in pixel density.

For the GIF image format, we observe that applying SIPA method does turns out to be good. The average compression ratio is 1.23 which means that the size of the array is about 1/1.23 times the size of the original image, that is, the array size is a little smaller than the image size. Therefore, using this scheme for GIF image formats is acceptable. Also, we can see that as the dpi increases, the compression ratio decreases. For 75 dpi, the compression ratio is 1.555 while for 200 dpi, the array to image size ratio is 1.004. This says that the efficiency of this technique decreases with increase in pixel density.

For the TIFF image formats, we observe that applying SIPA method is very successful. The average compression is 6.14 which means that the size of the array is about 1/6.14 the size of the original image, that is, the size of the array is smaller than the size of the image. Therefore, using this scheme for TIFF is a good idea. Also, we can see that as the dpi increases, the compression

ratio decreases. For 75 dpi, the compression ratio is 9.467 while for 200 dpi, the compression ratio is 4.348. This says that the efficiency of this technique decreases with increase in pixel density.

From Table (2), we observe that irrespective of the image format, the array size for a specific dpi is almost the same. This can be explained rationally as follows. When we work on images pixel wise in MATLAB, the location of information pixels in different image formats does not change. Therefore, the number of information pixels remains almost the same taking into consideration the fact that for a specific dpi, the pixel dimension is the same. The proof to this statement can be observed from the table. For example, for 75 dpi, the array dimension is 3728 for the JPEG, 3570 for both the BMP and the TIF, and it is 5928 in the GIF.

4.3 Generalizing The SIPA Method Over All Image Formats

The disparity of the number of information pixels in GIF image can be attributed to the fact that GIF images are structured in a different way when compared to the other 3 image formats.

The images in other image formats have an R,G and B scale whereas GIF has only 1 dimension. We can now consider the median value as the number of information pixels in any image format in general. We are taking the median and not the average because we can avoid the disparity in values caused by the GIF image. The median can be calculated as follows.

Consider the values in ascending order :3728, 3570, 3570, 5928

The middle value is the median. If the number of elements during the calculation of the mean is even, then we consider the average of the two middle values. Here the mean is therefore 3649. Similarly, we can calculate the mean for 100, 150 and 200 dpi.

We now plot a table to compute the size of the array for all image formats for specific dpi.

dpi	Median Array Dimension	Array Size (in KB)
75	3,649	57.01
100	7,635	119.29
150	21,741	339.70
200	41,381	646.57

Table (3)

We can now use this generalization of array sizes to all image formats. We now plot a table for the rest of the considered image formats.

Image Format	dpi	Image Size (in KB)	Array Size (in KB)	Compression Ratio	Average Compression Ratio
.exe	75	496	57.01	8.70	4.22
	100	544	119.29	4.56	
	150	736	339.70	2.17	
	200	944	646.57	1.46	
.fpx	75	160	57.01	2.81	1.66
	100	208	119.29	1.74	
	150	384	339.70	1.13	
	200	608	646.57	0.94	
.htm	75	192	57.01	3.37	2.08
	100	256	119.29	2.15	
	150	528	339.70	1.55	
	200	800	646.57	1.24	

Image Format	dpi	Image Size (in KB)	Array Size (in KB)	Compression Ratio	Average Compression Ratio
.max	75	144	57.01	2.53	1.49
	100	192	119.29	1.61	
	150	336	339.70	0.99	
	200	544	646.57	0.84	
.pdf	75	96	57.01	1.68	1.11
	100	144	119.29	1.21	
	150	288	339.70	0.85	
	200	464	646.57	0.72	
.png	75	512	57.01	8.98	5.73
	100	640	119.29	5.37	
	150	1520	339.70	4.47	
	200	2640	646.57	4.08	
TIFF class f	75	32	57.01	0.56	0.29
	100	32	119.29	0.27	
	150	64	339.70	0.19	
	200	96	646.57	0.15	
TIFF group 4	75	32	57.01	0.56	0.29
	100	32	119.29	0.27	
	150	64	339.70	0.19	
	200	96	646.57	0.15	
TIFF lzw	75	528	57.01	9.26	6.06
	100	704	119.29	5.90	
	150	1616	339.70	4.76	
	200	2800	646.57	4.33	
TIFF wwww	75	1600	57.01	28.06	22.28
	100	2880	119.29	24.14	
	150	6480	339.70	19.08	
	200	11536	646.57	17.84	
dcx	75	1312	57.01	23.01	14.44
	100	1504	119.29	12.61	
	150	3872	339.70	11.40	
	200	6944	646.57	10.74	
pcx	75	1312	57.01	23.01	14.44
	100	1504	119.29	12.61	
	150	3872	339.70	11.40	
	200	6944	646.57	10.74	

Table(4)

For the EXE image format, we observe that applying SIPA method turns out to be successful for dpi 75,100, 150, and 200. The average compression ratio is 4.22, that is, size of the array is about (1 / 4.22) times the size of the original image, that is, the size of the array is smaller than the size of the image. Therefore, using this scheme for EXE images is a good idea. Also, we can see that as the dpi increases, the compression ratio decreases. For 75 dpi, the array to image size ratio is 8.7 while for 200 dpi, the array to image size ratio is 1.46. This says that the technique is acceptable when used for 200 dpi. As discussed in chapter (2), 150 dpi has an acceptable image quality and therefore this method can be applied on EXE image formats with 150 dpi for which it is successful.

For the FPX image format, we observe that applying SIPA method is efficient for all dpi except the 200. The average compression is about 1.66 which means that the size of the array is 1/1.66 times the size of the original image, that is, the array size is smaller than the image size. Therefore, using this scheme for FPX images is acceptable. Also, we can see that as the dpi increases, the compression ratio gets smaller. For 75 dpi, the array to image size ratio is 1.74 while for 200 dpi, the array to image size ratio is 0.94.

For the HTM image format, we observe that applying SIPA method turns out to be successful for all dpi. The average compression is 2.08 which means that the size of the array is about 1/2.08 times the size of the original image, that is, the size of the array is about half of the size of the image. Therefore, using this scheme for HTM images is a good idea. Also, we can see that as the dpi increases, the compression ratio reduces. For 75 dpi, the array to image size ratio is 3.37 while for 200 dpi, the array to image size ratio is 1.24. This says that the technique is efficient when used for 75,100,150, and 200 dpi.

For the MAX image format, we observe that applying SIPA method is efficient only for 75 and 100 dpi. The average compression is about 1.49 which means that the size of the array is 1/1.49 times the size of the original image, that is, the array size is smaller than the image size; however, using this scheme for MAX images is not a good idea since images with resolution 75 and 100 dpi are unacceptable due to their quality and it is only for those dpi that this method is successful. Also, we can see that as the dpi increases, the compression ratio gets smaller. For 75 dpi, the array to image size ratio is 2.53 while for 200 dpi, the array to image size ratio is 0.84.

Although we previously stated that the PDF image format is preferred due to its relatively small size and better image quality, we observe that applying SIPA method is not very efficient but it is just acceptable. The average compression is 1.11 which means that the size of the array is about 1/1.11 times the size of the original image, that is, the size of the array is little smaller than the size of the image. Also, we can see that as the dpi increases, the compression ratio reduces. For 75 dpi, the array to image size ratio is 1.68 while for 200 dpi, the array to image size ratio is 0.72.

For the PNG image format, we observe that applying SIPA method is efficient. The average compression is 5.73 which means that the size of the array is about 1/5.73 times the size of the original image, that is, the size of the array is smaller than the size of the image. Also, we can see that as the dpi increases, the compression ratio reduces. For 75 dpi, the array to image size ratio is 8.98 while for 200 dpi, the array to image size ratio is 4.08.

For the TIFF CLASS F and the TIFF GROUP 4 image formats, we observe that the SIPA method is not efficient. The average compression is 0.29 which means that the size of the array is about 1/0.29 times the size of the original image, that is, the size of the array is larger than the size of the image. Also, we can see that as the dpi increases, the compression ratio reduces. For 75 dpi, the array to image size ratio is 0.56 while for 200 dpi, the array to image size ratio is 0.15.

For the TIFF LZW image format, we observe that applying SIPA method is efficient. The average compression is 6.06 which means that the size of the array is about 1/6.06 times the size of the original image, that is, the size of the array is larger than the size of the image. Also, we can see that as the dpi increases, the compression ratio reduces. For 75 dpi, the array to image size ratio is 9.26 while for 200 dpi, the array to image size ratio is 4.33.

For the TIFF UNCOMPRESSED image format, we observe that applying SIPA method is efficient. The average compression is 22.28 which means that the size of the array is about 1/22.28 times the size of the original image, that is, the size of the array is larger than the size of the image. Also, we can see that as the dpi increases, the compression ratio reduces. For 75 dpi, the array to image size ratio is 28.06 while for 200 dpi, the array to image size ratio is 17.84.
For the DCX, and the PCX image formats, we observe that the SIPA method is efficient. The average compression is 14.44 which means that the size of the array is about 1/14.44 times the

size of the original image, that is, the size of the array is larger than the size of the image. Also, we can see that as the dpi increases, the compression ratio reduces. For 75 dpi, the array to image size ratio is 23.01 while for 200 dpi, the array to image size ratio is 10.74.

TIFF Multi-Page Class F and TIFF Multi-Page Class 4 have the have the same compression ratio and properties as TIFF Class F and TIFF Class 4 respectively. TIFF Multi-Page LZW and TIFF Multi-Page have the have the same compression ratio and properties as TIFF LZW. TIFF Multi-Page UNCOMPRESSED has the same compression ratio and properties as TIFF UNCOMPRESSED.

5. APPLYING THE SIPA METHOD ON THE FULL BODY SEGMENT

The only change in this program when compared to the previous program is the consideration of addresses for the information pixels only from the full body segment.In this paper, the concept of segmentation helps reduce memory occupied by information pixels because only segment (5) is considered and the information pixels in the other segments are ignored. This way, the total number of information pixels in the whole image reduces and hence the memory required for its storage. For this to be executed, we need to define the pixel boundaries for segment (5). This can be explained as follows:Length of the A4 sheet: 29.7 cm. Width of the A4 sheet is 21 cm.

5 .1 Defining Pixel Boundaries For The Full Body Segment

The information pixels in segment (5) start 7.7 cm from the top of the sheet. Let ‘m’ represent the address of row pixels. Since the resolution of the image is 150 dpi and 1cm=0.39370078 inches, the start address of m is :7.7*150* 0.39370078 = 454.

The information pixels of segment (5) end 5cm above the bottom of the sheet, which is 24.7cm from the top of the sheet. Since the resolution of the image is 150 dpi and 1cm=0.39370078 inches, the end address of m is :24.7*150* 0.39370078 = 1459
Let n represent the address of the column pixels. The information pixels in segment (5) start 2 cm to the right of the left end of the A4 sheet. Since the resolution of the image is 150 dpi and 1cm=0.39370078 inches, the start address of n is :2*150* 0.39370078 = 118.

The information pixels in segment (5) end 2 cm to the left of the right end of the A4 sheet, which is 19cm to the right of the left end of the sheet. Since the resolution of the image is 150 dpi and 1cm=0.39370078, the end address of n is :19*150* 0.39370078 = 1122.

Table (5) below shows the starting and ending row and column pixels for different dpi as computed above.

Dpi	Starting row	Ending row	Starting column	Ending column
75	227	729	59	561
100	303	972	79	748
150	454	1459	118	1122
200	606	1944	157	1496

Table (5)

5. 2 Program Algorithm

A typical program for a JPEG image with 150 dpi that works only on the body of the KSU sheet is shown below.

```
clear all; close all; clc;
tic;
RGBimage=imread('C:\Users\ALGhalayini\Desktop\algha\a4 size\jpg\full page 256 colors 8bit
150.jpg','jpg');
GRAYimage= rgb2gray(RGBimage);
Figure(1);
Imshow(GRAYimage(454:1459,118:1122));
M =length(GRAYimage);
N =length(GRAYimage(1,:));
total_pixels=m*n;
k=1;
forI =454:1459
for j=118:1122
ifGRAYimage( I ,j)<128
        Array1(1,k)= I ; Array2(1,k)=j;
K =k+1;
end
end
end
 Array=[Array1' Array2'];
toc
```

We observe that the program is different from that of the JPEG 150 dpi A4 image size program in the '*for*' loop. The snippet is shown below :

```
forI =454:1459
for j=118:1122
ifGRAYimage( I ,j)<128
        Array1(1,k)= I ; Array2(1,k)=j;
K =k+1;
end
end
end
```

The concept of selecting the starting and ending row and column which is nothing but the range of values of *i* and *j*is explained in Table(5). We now record the observations from these executions and display them in a table as below.

5. 3 Execution Results and Analysis

Table (6) below represents the result of the analysis applied on the Full Body Segment. It also shows that the technique of storing the information pixels addresses (SIPA) is successful for the bmp, the gif, and the TIFF image formats. We observe that Table (6) is very similar to

Table (2). The only difference is that we consider the whole A4 KSU scanned document page and search for information pixels in Table (2) while we consider only the Full Body Segment of the A4 KSU scanned document page and search for information pixels in Table (6). Column 3 in Table (6) is the image size. Column 7 in Table (6) indicates the size of the array formed by storing information pixels contained in the body of the KSU document page while column 7 in Table (2) indicates the size of the array formed by storing information pixels contained in the whole KSU document page. By intuition, we know that the size of the array used for storing information pixels addresses of the whole KSU document page should be larger than the size of the array used for storing information pixels of the Body Segment of KSU document page. This intuition is verified when we see that values in column 7 in Table (6) are lesser than corresponding values in Table (2)

Image Format	dpi	Image Size (in KB)	Execution Time (in s)	Number of Information Pixels	Array Dimensions		Array Size (in KB)	Compression Ratio	Average Compression Ratio
jpeg	75	60	0.86	252,004	3,453.5	X	53.96	1.11	0.74
	100	84	1.96	447,561	7,154.5	X	111.79	0.75	
	150	180	8.22	1,009,020	19,407.5	X	303.24	0.59	
	200	288	17.16	1,791,582	35,726	X	558.22	0.52	
bmp	75	992	0.98	252,004	3,313.5	X	51.77	19.16	15.11
	100	1,744	1.94	447,561	7,022	X	109.72	15.90	
	150	3,904	8.22	1,009,020	19,304.5	X	301.63	12.94	
	200	6,944	16.91	1,791,582	35,696.5	X	557.76	12.45	
gif	75	96	1.05	252,004	3,928.5	X	61.38	1.56	1.18
	100	144	2.37	447,561	7,987.5	X	124.80	1.15	
	150	304	9.19	1,009,020	2,0360	X	318.13	0.96	
	200	592	17.10	1,791,582	36,695.5	X	573.37	1.03	
tif	75	304	0.84	252,004	3,313.5	X	51.77	5.87	4.73
	100	496	1.87	447,561	7,022	X	109.72	4.52	
	150	992	8.18	1,009,020	19,304.5	X	301.63	3.29	
	200	2,912	17.24	1,791,582	35,696.5	X	557.76	5.22	

Table (6)

5.4 Generalization The SIPA Method Over All Image Formats

We now use Formula (1) to calculate the median array dimension and the array size and generalize it for all image formats as we did previously for the A4 size image formats.

dpi	Median Array Dimension	Array Size (in KB)
75	1,691.75	26.43
100	3,544.13	55.38
150	9,678.00	151.22
200	17,855.63	278.99

Table (7)

We now use the generalized array sizes for different dpi for all image types as explained previously for the A4 size image formats.

We recall that for all image formats and all dpi, whenever the compression ratio is less than 1, It implies that the array sizes formed by pixel based segmentation are larger than the image itself and therefore applying the SIPA technique is unsuccessful.

Table (8) below shows that applying SIPA technique is successful for all considered image formats except the TIFF Class F and TIFF Class 4. It also shows the array to image average percentage for various image formats when scanned over the Full Body Segment only. TIFF Multi-Page Class F and TIFF Multi-Page Class 4 has the properties of TIFF Class F or TIFF Class 4.

TIFF Multi-Page LZW and TIFF Multi-Page has properties of TIFF LZW. TIFF Multi-Page UNCOMPRESSED has properties of TIFF UNCOMPRESSED.

Image Format	dpi	Image Size	Array Size	Compression Ratio	Average Comp.Ratio
exe	75	464	26.43	17.55	8.53
	100	512	55.38	9.25	
	150	640	151.22	4.23	
	200	864	278.99	3.10	
fpx	75	120	26.43	4.54	2.72
	100	156	55.38	2.82	
	150	288	151.22	1.90	
	200	456	278.99	1.63	
htm	75	160	26.43	6.05	3.73
	100	208	55.38	3.76	
	150	400	151.22	2.65	
	200	688	278.99	2.47	
max	75	112	26.43	4.24	2.58
	100	160	55.38	2.89	
	150	240	151.22	1.59	
	200	448	278.99	1.61	
pdf	75	72	26.43	2.72	1.84
	100	108	55.38	1.95	
	150	216	151.22	1.43	
	200	348	278.99	1.25	
png	75	336	26.43	12.71	9.67
	100	512	55.38	9.25	
	150	976	151.22	6.45	
	200	2864	278.99	10.27	
TIFF Class F	75	16	26.43	0.61	0.46
	100	32	55.38	0.58	
	150	48	151.22	0.32	
	200	96	278.99	0.34	
TIFF Group 4	75	16	26.43	0.61	0.48
	100	32	55.38	0.58	
	150	48	151.22	0.32	
	200	112	278.99	0.40	
TIFF LZW	75	304	26.43	11.50	9.36
	100	496	55.38	8.96	
	150	992	151.22	6.56	
	200	2912	278.99	10.44	
TIFF uncompressed	75	992	26.43	37.53	29.97
	100	1744	55.38	31.49	
	150	3920	151.22	25.92	
	200	6960	278.99	24.95	
dcx	75	752	26.43	28.45	23.76
	100	1168	55.38	21.09	
	150	2304	151.22	15.24	
	200	8448	278.99	30.28	
pcx	75	752	26.43	28.45	23.76
	100	1168	55.38	21.09	
	150	2304	151.22	15.24	
	200	8448	278.99	30.28	

Table (8)

From Table (2) and Table (6) shown above, we note few important observations :

- As seen from the table, the time elapsed for all the cases for execution of the A4 image size is more than the time elapsed for execution of full body segments.
- It is also observed that as the dpi increases, the time elapsed during the execution of the Full Body Segment is comparatively lesser than the time elapsed during the execution of A4 size image of the same dpi. This can be explained from the fact that the MATLAB code needs to run over lesser number of pixels during the execution of the Full Body Segment than the A4 size image.

Applying SIPA method on different image formats with different dpi proved to be efficient for most formats and yielded a considerable memory savings (if we ignored the execution time of the storage process) which is an important issue to us.

Table (9) below shows the amount and percentage of memory saved by using SIPA model on different Image formats of the Whole KSU A4 Document scanned Images

In Table (9) below, whenever the value of the memory saved is negative it implies that applying SIPA method is inefficient and whenever it is positive applying SIPA method is efficient. Although the memory saving percentage turned out to be negative for certain dpi the average saving could be positive which means applying SIPA method can be used for those other dpi where there is a considerable memory saving.

In general we can see that with most image formats applying SIPA method turned out to be highly efficient with the Whole KSU Document page as well as with the Full Body Segment as shown in Table (10).

Table (10) below shows the amount and percentage of memory saved by using SIPA method on different Image formats of the Full Body Segment scanned Images.

Image Format	dpi	Original Image Size (in KB)	Array Size (in KB)	Memory Saved using SIPA	Percentage Of Memory Saved Using SIPA	Average Percentage Of Memory Saved Using SIPA
jpeg	75	80.00	58.25	21.75	27.19%	-22.95%
	100	112.00	120.60	-8.60	-7.68%	
	150	240.00	341.38	-101.38	-42.24%	
	200	384.00	649.18	-265.18	-69.06%	
bmp	75	1600.00	55.77	1544.23	96.51%	95.40%
	100	2880.00	117.98	2762.02	95.90%	
	150	6480.00	338.02	6141.98	94.78%	
	200	11504.00	643.97	10860.03	94.40%	
gif	75	144.00	92.62	51.38	35.68%	16.00%
	100	240.00	185.51	54.49	22.71%	
	150	496.00	470.18	25.82	5.21%	
	200	864.00	860.40	3.60	0.42%	
tif	75	528.00	55.77	472.23	89.44%	82.19%
	100	704.00	117.98	586.02	83.24%	
	150	1616.00	338.02	1277.98	79.08%	
	200	2800.00	643.97	2156.03	77.00%	
.exe	75	496	57.01	438.99	88.51%	62.98%
	100	544	119.29	424.71	78.07%	
	150	736	339.70	396.30	53.85%	
	200	944	646.57	297.43	31.51%	

Image Format	dpi	Original Image Size (in KB)	Array Size (in KB)	Memory Saved using SIPA	Percentage Of Memory Saved Using SIPA	Average Percentage Of Memory Saved Using SIPA
.fpx	75	160	57.01	102.99	64.37%	28.05%
	100	208	119.29	88.71	42.65%	
	150	384	339.70	44.30	11.54%	
	200	608	646.57	-38.57	-6.34%	
.htm	75	192	57.01	134.99	70.31%	44.64%
	100	256	119.29	136.71	53.40%	
	150	528	339.70	188.30	35.66%	
	200	800	646.57	153.43	19.18%	
.max	75	144	57.01	86.99	60.41%	19.58%
	100	192	119.29	72.71	37.87%	
	150	336	339.70	-3.70	-1.10%	
	200	544	646.57	-102.57	-18.86%	
.pdf	75	96	57.01	38.99	40.61%	0.12%
	100	144	119.29	24.71	17.16%	
	150	288	339.70	-51.70	-17.95%	
	200	464	646.57	-182.57	-39.35%	
.png	75	512	57.01	454.99	88.86%	80.85%
	100	640	119.29	520.71	81.36%	
	150	1520	339.70	1180.30	77.65%	
	200	2640	646.57	1993.43	75.51%	
TIFF Class F	75	32	57.01	-25.01	-78.16%	-338.81%
	100	32	119.29	-87.29	-272.78%	
	150	64	339.70	-275.70	-430.78%	
	200	96	646.57	-550.57	-573.51%	
TIFF Group 4	75	32	57.01	-25.01	-78.16%	-338.81%
	100	32	119.29	-87.29	-272.78%	
	150	64	339.70	-275.70	-430.78%	
	200	96	646.57	-550.57	-573.51%	
TIFF LZW	75	528	57.01	470.99	89.20%	82.04%
	100	704	119.29	584.71	83.06%	
	150	1616	339.70	1276.30	78.98%	
	200	2800	646.57	2153.43	76.91%	
TIFF uncompressed	75	1600	57.01	1542.99	96.44%	95.36%
	100	2880	119.29	2760.71	95.86%	
	150	6480	339.70	6140.30	94.76%	
	200	11536	646.57	10889.43	94.40%	
Dcx	75	1312	57.01	1254.99	95.65%	92.41%
	100	1504	119.29	1384.71	92.07%	
	150	3872	339.70	3532.30	91.23%	
	200	6944	646.57	6297.43	90.69%	
pcx	75	1312	57.01	1254.99	95.65%	92.41%
	100	1504	119.29	1384.71	92.07%	
	150	3872	339.70	3532.30	91.23%	
	200	6944	646.57	6297.43	90.69%	

Table (9)

Image Format	dpi	Original Image Size (in KB)	Array Size (in KB)	Memory Saved using SIPA	Percentage Of Memory Saved Using SIPA	Average Percentage Of Memory Saved Using SIPA
jpeg	75	60	53.96	6.04	10.07%	-46.33%
	100	84	111.79	-27.79	-33.08%	
	150	180	303.24	-123.24	-68.47%	
	200	288	558.22	-270.22	-93.83%	
bmp	75	992	51.77	940.23	94.78%	93.18%
	100	1744	109.72	1634.28	93.71%	
	150	3904	301.63	3602.37	92.27%	
	200	6944	557.76	6386.24	91.97%	
gif	75	96	61.38	34.62	36.06%	11.97%
	100	144	124.80	19.20	13.33%	
	150	304	318.13	-14.13	-4.65%	
	200	592	573.37	18.63	3.15%	
TIFF	75	304	51.77	252.23	82.97%	77.82%
	100	496	109.72	386.28	77.88%	
	150	992	301.63	690.37	69.59%	
	200	2912	557.76	2354.24	80.85%	
exe	75	464	26.43	437.57	94.30%	81.89%
	100	512	55.38	456.62	89.18%	
	150	640	151.22	488.78	76.37%	
	200	864	278.99	585.01	67.71%	
fpx	75	120	26.43	93.57	77.97%	57.20%
	100	156	55.38	100.62	64.50%	
	150	288	151.22	136.78	47.49%	
	200	456	278.99	177.01	38.82%	
htm	75	160	26.43	133.57	83.48%	69.62%
	100	208	55.38	152.62	73.38%	
	150	400	151.22	248.78	62.20%	
	200	688	278.99	409.01	59.45%	
max	75	112	26.43	85.57	76.40%	54.13%
	100	160	55.38	104.62	65.39%	
	150	240	151.22	88.78	36.99%	
	200	448	278.99	169.01	37.72%	
pdf	75	72	26.43	45.57	63.29%	40.46%
	100	108	55.38	52.62	48.73%	
	150	216	151.22	64.78	29.99%	
	200	348	278.99	69.01	19.83%	
png	75	336	26.43	309.57	92.13%	89.02%
	100	512	55.38	456.62	89.18%	
	150	976	151.22	824.78	84.51%	
	200	2864	278.99	2585.01	90.26%	
TIFF Class F	75	16	26.43	-10.43	-65.21%	-135.98%
	100	32	55.38	-23.38	-73.05%	
	150	48	151.22	-103.22	-215.04%	
	200	96	278.99	-182.99	-190.62%	
TIFF Group 4	75	16	26.43	-10.43	-65.21%	-125.60%
	100	32	55.38	-23.38	-73.05%	
	150	48	151.22	-103.22	-215.04%	
	200	112	278.99	-166.99	-149.10%	

Image Format	dpi	Original Image Size (in KB)	Array Size (in KB)	Memory Saved using SIPA	Percentage Of Memory Saved Using SIPA	Average Percentage Of Memory Saved Using SIPA
TIFF LZW	75	304	26.43	277.57	91.30%	88.83%
	100	496	55.38	440.62	88.84%	
	150	992	151.22	840.78	84.76%	
	200	2912	278.99	2633.01	90.42%	
TIFF uncompressed	75	992	26.43	965.57	97.34%	96.57%
	100	1744	55.38	1688.62	96.82%	
	150	3920	151.22	3768.78	96.14%	
	200	6960	278.99	6681.01	95.99%	
dcx	75	752	26.43	725.57	96.48%	95.47%
	100	1168	55.38	1112.62	95.26%	
	150	2304	151.22	2152.78	93.44%	
	200	8448	278.99	8169.01	96.70%	
pcx	75	752	26.43	725.57	96.48%	95.47%
	100	1168	55.38	1112.62	95.26%	
	150	2304	151.22	2152.78	93.44%	
	200	8448	278.99	8169.01	96.70%	

Table (10)

6. CONCLUSION

Our goal in this research paper was to introduce Information Pixels and the concept of Storing Information Pixels Addresses (SIPA) and show practically that it is an efficient model for document storage for most image formats.

We observed that the stored images for most image formats are better in size if they are compared to their original sizes, which yielded in a considerable amount of memory savings.

- The time taken to store an image in a specific format with lesser dpi is less than the time taken to store an image in the same format with larger dpi. Hence, this is a time/complexity-quality tradeoff.
- Applying SIPA method on Full Body Segment images of a specific format and dpi take lesser time to be executed than its A4 size image counterpart.

Considering the above 2 factors, the complexity and time of image storage increases as we move from 75 to 200 dpi. It takes lesser time to store a Full Body Segment scanned image than the Whole A4 Document scanned image.

We can therefore conclude that using 150 dpi Full Body Segment scanned images produces an optimum result because :

- Storage of images with 150 dpi do not take as much time as images with 200 dpi;
- Storage of an image with 150 dpi is not as complex as an image with 200 dpi because the number of information pixels are lesser in images with 150 dpi and hence lesser the size of the array of addresses.
- The quality of images is certainly better than images with 75,100

REFERENCES

[1] O. Lezoray and H. Cardot, “Cooperation of color classification schemes and color watershed: a study for microscopic images,” IEEE Trans. on Image Processing, no. 7, vol. 11, 2002.

[2] D. Comaniciu, “Image segmentation using clustering with saddle point detection,” IEEE Int. Conf. Image Processing (ICIP'02), Rochester, NY, Vol. 3, 297-300, 2002

[3] G. Herman, and B. Carvalho, “Multiseeded segmentation using fuzzy connectedness,” IEEE Tans. On Pattern Analysis and Machine Intelligence, no. 5, vol. 23, May 2001.

[4] R. Nock, F. Nielsen, “Semi-supervised statistical region refinement for color image segmentation,” Pattern Recognition no. 6 vol. 38 pp. 835-846, 2005.

[5] Lee, and M. Lewicki, “Unsupervised classification, segmentation, and enhancement using, ICA mixture model,” IEEE Trans. Image Proc. No. 3 vol 11 pp. 270-279, 2002.

[6] H. Shah-Hosseini and R. Safabakhsh, “Automatic multilevel thresholding for image segmentation by growing time adaptive self-organizing map,” IEEE Tans. On Pattern Analysis and Machine Intelligence, no. 10, vol. 24, 2002.

[7] T. HORIUCHI, “Grayscale image segmentation using color space,” IEICE Transactions on Information and Systems, E89-D(3) pp. 1231-1237, 2006.

[8] T. Horiuchi and S. Hirano, "Colorization algorithm for grayscale image by propagating seed pixels," Proc. IEEE ICIP 2003, pp.457–460, 2003.

[9] Y. Deng, and B. Manjunath, “Unsupervised segmentation of color-texture regions in image and video,” IEEE Tans. On Pattern Analysis and Machine Intelligence, no. 3, vol, 23, pp. 800-810, 2001.

[10] Z. Tu, and S. Zhu, “Image segmentation by data-driven markov chain montecarlo,” IEEE Tans. On Pattern Analysis and Machine Intelligence, no 5, vol. 24, pp. 657-673, 2002.

[11] J. Rushing, H. Ranganth, T. Hinke, and S. Graves, “Image Segmentation using association rule features,” IEEE Trans. on Image Processing, no. 5, vol. 11, 2002.

[12] M. Ozden, E. Polat, “A color image segmentation approach for content-based image retrieval,” Pattern Recognition,vol. 40 , issue 4, pp. 1318-1325, 2007.

[13] S. Panda, P. Nanda, and P Mohapatra, “Multiresolution Approach for Color Image Segmentation using MRF Model,” Proceedings of the National Conference on Smart Communication Technologies and Industrial Informatics, SCTII, 03-04 pp. 34-42, Rourkela, Feb 2007.

[14] S. Kang, S. Park, Y. Shin, H. Yoo, and D. Jang, “Image segmentation using statistical approach via perception-based color Information,” International Journal of Computer Science and Network Security, no. 4, vol. 8 No.4, April 2008.

[15] Bocij, P. et al., Business information systems: Technology, development and management. (London: Financial Times Management, 1999).

[16] Richard Casey, " Document Image Analysis", available at http://cslu.cse.ogi.edu/HLTsurvey/ch2node4.html

[17] H Bunke And P S P Wang, "Handbook Of Character Recognition And Document ImageAnalysis", available at:http://www.worldscibooks.com/compsci/2757.html

[18] M Simone, " Document Image Analysis And Recognition" , available at :http://www.dsi.unifi.it/~simone/DIAR/

[19] "Document Image Analysis",availableat:http://elib.cs.berkeley.edu/dia.html

[20] Richard Casey, " Document Image Analysis", available at:http://cslu.cse.ogi.edu/HLTsurvey/ch2node4.html

[21] H Bunke And P S P Wang, "Handbook Of Character Recognition And Document Image Analysis", available at :http://www.worldscibooks.com/compsci/2757.html

[22] M Simone, " Document Image Analysis And Recognition" http://www.dsi.unifi.it/~simone/DIAR/

[23] "Basic Images Processing" : "Some general commands related to handling Matlab graphics and printing" "Simple image processing operations that you can do with Matlab" available at:http://noodle.med.yale.edu/~papad/ta/handouts/matlab_image.html

[24] "Document Image Analysis" availableat :http://elib.cs.berkeley.edu/dia.html

[25] "Getting Started with MATLAB" available at :http://www.stewart.cs.sdsu.edu/cs205/module7/getting6.html

[26] "Matlab Image Processing Toolbox" available at :http://homepages.inf.ed.ac.uk/rbf/HIPR2/impmatl.htm

[27] Matlab Resources" available at :http://www.cse.uiuc.edu/heath/scicomp/matlab.htm

[28] "Read and Display an Image" available at :http://www.mathworks.com/access/helpdesk/help/toolbox/images/getting8.html
[29] "Reading and displaying images" matlab toutorial",03/25/2005, available at :http://ai.ucsd.edu/Tutorial/matlab.html#images
[30] ALGhalayini M., Shah A., "Introducing The (POSSDI) Process : The Process of Optimizing the Selection of The Scanned Document Images". International Joint Conferences on Computer, Information, Systems Sciences, and Engineering (CISSE 06) December 4 - 14, 2006.
[31] Mohammad A. ALGhalayini and ELQasemALNemah, "An Efficient Storage and Retrieval Technique for Documents Using Symantec Document Segmentation (SDS) Approach"; CISSE 2007, The Third International Joint Conferenceson Computer, Information, and Systems Sciences, and Engineering (CISSE 07).
[32] Mohammad A. ALGhalayini "Introducing The Concept Of Information Pixels and the SIPA (Storing Information Pixels Addresses) Method As a Model for Document Storage"; CISSE 2011, The Seventh International Joint Conferenceson Computer, Information, and Systems Sciences, and Engineering (CISSE 11).

AUTHOR

Mohammad A. ALGhalayini, Bs.C. in Computer Science and Mathematics, Iowa, USA. 1987. Ms.C. in Computer Science, Alabama, USA. 1999. Ph.D. in Information Management Systems, London, UK. 2010. The author lectured at the College of Information & Computer Sciences of King Saud University, KSA for more than 14 years. Specialized in document management and databases, the author developed several imaging database systems for various KSU offices and departments, currently working as an academic researcher and supervisor of computer and Information unit at the KSU Vice Rectorate of Graduate Studies and scientific Research.