Comparison of Fax Image Compression Techniques

Vishweswaran K
Electronics and Communication
PES University
Bangalore, Karnataka

Sreenivas K
Electronics and Communication
PES University
Bangalore,Karnataka

Alisha Reji
Electronics and Communication
PES University
Bangalore,Karnataka

B K Shyam Santhosh
Electronics and Communication
PES University
Bangalore,Karnataka

Mr. Dhruva Kumar
Technical Manager
Consilient Technologies
Bangalore, Karnataka

Dr.Ajey SNR
HOD - ECE Department
PES University
Bangalore,Karnataka

***Abstract*—This project aims to emphasize the need for FAX machines and the underlying data compression algorithms it employs. Data compression is an essential component while transmitting images, be it via a telephonic cable or in the present scenario wirelessly. Data compression is an essential part of data transmission, and it is required to reduce noise and errors induced while transmitting data as well as to avoid storage and transmission of the huge amounts of data that is more susceptible to noise making it less accurate. FAX machine protocols are described in the ITU-T standards.**

The focus of this study is to understand, implement and analyze the facsimile algorithms stated in the ITU-T. The compression ratio achieved using all the different algorithms is analyzed and insightful inferences are made. The advantages of FAX image Compression algorithms are understood and structures that optimize the compression ratio for each algorithm are identified.

***Index Terms*—Image Compression, Lossless Compression, Huffman algorithm, T.4 1D, T.4 2D, T.6 2D, TIFF Files**

I. Introduction

Fax or facsimile use the concept of Plain Old Telephone Services (POTs) for transmitting and replicating documents via radio waves or cable lines. Fax machines are primarily configured to transmit scanned textual and graphical data through the telephone network to similar facsimile machines, where facsimiles reproduce the documents to get back the original documents.

The document data is translated to a series of binary digits, and the bitstream is passed to a source encoder, which compresses all the bits and substitutes long runs of white or black spots with codewords, and hence ensures compression. Transmission of the encoded bitstream is done via an analog carrier, through the telephone network.

Modern fax machines have expanded in terms of speed, compression ratio and total number of scan lines possible per inch.

This project aims to implement the compression algorithms mentioned in Group 3 and Group 4 facsimile protocols based on the ITU-T model. Analyzing these algorithms and comparison of the compression ratio achieved with each of the algorithms is performed. Group 3 1 Dimensional Encoding scheme exploits the idea of repetitive run lengths and uses smaller codewords based on statistical experiments. 2-Dimensional Encoding, on the other hand, uses the preceding line as a reference line to encode data to reduce the number of errors that occurred during transmission. These results have been analyzed, and important conclusions have been made.

The project also uses TIFF files as they are flexible and are supported in multiple applications. Data is extracted from these TIFF files and the compression algorithms are performed on this data.

II. Background Knowledge

This section talks about the kind of data that is used in the project. This section is intended to give the reader a brief understanding of the input data, the file format and its properties. The input data used in the project are Images that are stored in the form of a TIFF file.

"Tagged Image File Format", popularly known as TIFF is typically used to store black and white images.TIFF files have multiple applications and can be found in paint, imaging, and desktop publication applications.

The flexible nature and its support for numerous compression algorithms make this file format easy to use by developers and exploit its properties and use it for their needs. It is ideally used for image storage as it allows multiple bitmap images to exist in a single file.

A TIFF file has 3 main components in it, These are the

- Image file Header
- Image file Directory
- Raw Bitmap data

These 3 components of the tiff file can be physically arranged in the memory in multiple ways. This is indicated in the **Fig 1.Different Physical arrangements of data in a TIFF file**

All the data in the tiff file is accessed using multiple offsets used as pointers to the memory address of the required data. No matter how the data is arranged physically the logical

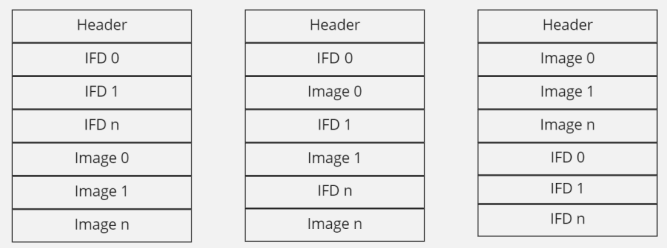

Fig. 1. Different Physical arrangements of data in a TIFF file

arrangement of tiff files is always the same. Hence it is important for us to understand how the logical arrangement of the tiff file is. The logical arrangement of the tiff file is shown below in **Fig 2. Logical organization of TIFF File.**

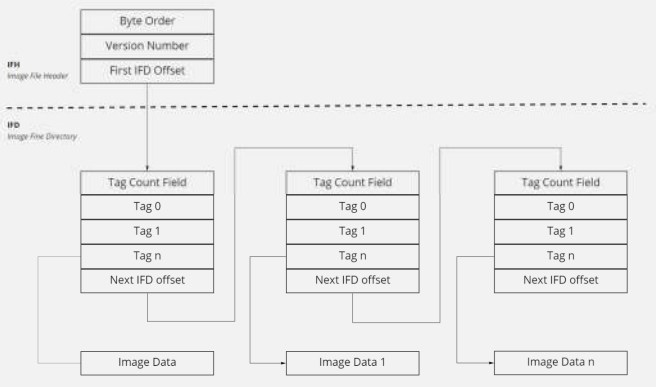

Fig. 2. Logical organization of TIFF file

Details about the 3 main components are described in detail below.

A. Image file Header

- The only constant segment in all tiff files. It has a fixed size of 8 bytes.
- These 8 Bytes are further divided into sub-fields
- First 2 Bytes - Identifies the Endianness of the entire TIFF file.
- A value of 4949h(II) indicates little-endian format and a value of 4D4D(MM) indicates big-endian format.
- The Next 2 Bytes - Identifies the file format, This is always fixed to the value 42 for TIFF files. As the value 42 indicates that the file format is “TIFF”.
- The last 4 Bytes - This field stores the offset to the first IFD. Using this offset we can move to the address where the First IFD is stored and read the data.
- This is also indicated in Fig . X: Logical arrangement of the tiff file.

B. Image file Directory

- Each Image file directory corresponds to an Image in the tiff file.
- The First 2 Bytes of the IFD indicate the number of tags present in the IFD.
- Depending on the number of tags present the next field of the IFD is a sequence of 12 Byte tags. If there are N tags present there will be a field of N*12 bytes for the tags.
- Each tag holds some aspect of Information about the image like Image width, Image Height, Where the actual image data is stored in the tiff file, and so on.
- After the sequence of 12-byte fields for the tags, there is a 4 Byte field that holds the offset value to the next IFD in case there is another image present in the tiff file. If there are no more images present in the tiff file, Then this field holds the value 0.

C. The raw bitmap data

- The raw bitmap data holds the actual image in binary. This data can be accessed using a few of the mandatory tags that are present in every IFD.
- These mandatory tags are RowsPerStrip, StripOffsets, and StripByteCounts.
- The data that is stored is in binary format and should be interpreted to form the image based on the colour scheme of the image and other details given in the tags for the respective image.

With this knowledge, we can further look into the various compression algorithms.

III. Group 3: 1-Dimensional Encoding

A. Overview

One dimensional run length encoding (RLE) is a technique that is used for compression of bits(data) using a predefined table of codewords, which is based on the frequency of the run length in statistically averaged data. Run length of each black or white pixel is calculated and the corresponding codeword is looked up in the Terminating Codewords table. For example, two, three and four runs of black pixels are found to be the most frequent in handwritten data, thus they have the code words of two (11), two (10) and three bits (011) respectively. Thus, the resulting compressed sequence will be optimized to occupy the least space. With large runs (¿64) it was it was found efficient to use certain make-up codewords, these codewords act as a prefix to the terminating code. In such a case, a makeup codeword from the Makeup Codeword table is prefixed for the closest multiple of 64. The terminating codeword for the remaining length is then appended. For example, a run of 585 white pixels, we use the makeup code for 576 W (01101000) followed by the terminating code for 585-576 = 9 White (10100). Thus, the size of the file is decreased from 585 bits to just 13 bits.

B. Encoding Methodology

The length of the continuous sequence of a particular value is known as runlength. Runlength Encoding scheme (Group 3 1 D) predominantly uses this recurrence of similar data to compress data. The set of terminating and makeup codewords are documented in the ITU-T T.4 series and are a set of predefined values determined by statistical experiments. The

terminating and makeup codewords are given in Table 1.1. After encoding each line, an end of line codeword given by 000000000001 is used. After this EOL codeword is read, the decoder is aware that the following encoded data corresponds to a new line. This way data at the receiver end is decoded synchronously. Fill Bits are Extra bits are added between the encoded data and end of line (EOL) codeword to ensure that every new line of encoded data will begin from a new byte boundary. This is done to avoid errors during transmission and during decoding at the receiver variable number of 0s are used for this purpose. Return to control signal is used to indicate that the entire data has been encoded. To indicate this end of document transmission, the encoder appends a series of 6 consecutive EOLs. Thus, denoting the end of document transmission, when the decoder reads continuous 6 EOL codewords.

White run length	Code word	Black run length	Code word
0	00110101	0	0000110111
1	000111	1	010
2	0111	2	11
3	1000	3	10
4	1011	4	011
5	1100	5	0011
6	1110	6	0010
7	1111	7	00011
8	10011	8	000101
9	10100	9	000100
10	00111	10	0000100
11	01000	11	0000101
12	001000	12	0000111
13	000011	13	00000100
14	110100	14	00000111
15	110101	15	000011000
16	101010	16	0000010111
17	101011	17	0000011000
18	0100111	18	0000001000
19	0001100	19	00001100111
20	0001000	20	00001101000
21	0010111	21	00001101100
22	0000011	22	00000110111
23	0000100	23	00000101000
24	0101000	24	00000010111
25	0101011	25	00000011000
26	0010011	26	000011001010
27	0100100	27	000011001011
28	0011000	28	000011001100
29	00000010	29	000011001101
30	00000011	30	000001101000
31	00011010	31	000001101001
32	00011011	32	000001101010
33	00010010	33	000001101011
34	00010011	34	000011010010
35	00010100	35	000011010011
36	00010101	36	000011010100
37	00010110	37	000011010101
38	00010111	38	000011010110
39	00101000	39	000011010111
40	00101001	40	000001101100
41	00101010	41	000001101101
42	00101011	42	000011011010
43	00101100	43	000011011011
44	00101101	44	000001010100
45	00000100	45	000001010101
46	00000101	46	000001010110
47	00001010	47	000001010111
48	00001011	48	000001100100
49	01010010	49	000001100101
50	01010011	50	000001010010
51	01010100	51	000001010011
52	01010101	52	000000100100
53	00100100	53	000000110111
54	00100101	54	000000111000
55	01011000	55	000000100111
56	01011001	56	000000101000
57	01011010	57	000001011000
58	01011011	58	000001011001
59	01001010	59	000000101011
60	01001011	60	000000101100
61	00110010	61	000001011010
62	00110011	62	000001100110
63	00110100	63	000001100111

Fig. 3. Terminating Code Words

IV. Group 3: 2-Dimensional Encoding

A. Overview

The two-dimensional coding scheme is an optional extension of the one-dimensional coding scheme. It is more optimized when compared to the One-dimensional encoding. This is a line-by-line coding method wherein the changing

White run length	Code word	Black run length	Code word
64	11011	64	0000001111
128	10010	128	000011001000
192	010111	192	000011001001
256	0110111	256	000001011011
320	00110110	320	000000110011
384	00110111	384	000000110100
448	01100100	448	000000110101
512	01100101	512	000000110100
576	01101000	576	000000110101
640	01100111	640	000000100101 0
704	011001100	704	000000100101 1
768	011001101	768	000000100110 0
832	011010010	832	000000100110 1
896	011010011	896	000000111001 0
960	011010100	960	000000111001 1
1024	011010101	1024	000000111010 0
1088	011010110	1088	000000111010 1
1152	011010111	1152	000000111011 0
1216	011011000	1216	000000111011 1
1280	011011001	1280	000000101001 0
1344	011011010	1344	000000101001 1
1408	011011011	1408	000000101010 0
1472	010011000	1472	000000101010 1
1536	010011001	1536	000000101101 0
1600	010011010	1600	000000101101 1
1664	011000	1664	000000110010 0
1728	010011011	1728	000000110010 1
EOL	000000000001	EOL	000000000001

NOTE – It is recognized that terminals exist which accommodate larger paper widths maintaining the standard horizontal resolution. This option has been provided for by the addition of the make-up code set defined in this table.

Fig. 4. MakeUp Code Words

elements' position in the current line is compared with the changing elements' position present in the previous line and based on which a coding mode is chosen to perform 2D encoding. Line is first encoded one dimensionally, then the original coded line is interpreted as the reference line for the next line which is to be encoded two-dimensionally. It is more widely known as the Modified READ Coding technique, where READ is the acronym for "Relative Element Address Designate", it exploits the correlation between successive lines.

B. Encoding Methodology

1) K parameter and Interpretation of the Image: Before understanding the two-dimensional encoding techniques, we must understand some keywords like K parameter, Coding line, and Reference Line, Changing Elements (a0, a1, a2, b1, b2).

2) What is the K parameter?: Since in the Two-Dimensional encoding technique the current scanline is encoded with reference to the previous line, we have to make sure that there are no errors in the coding of any line. If we make an error in coding then the next line will also have an error as the previous line was coded incorrectly. In case of a transmission error, the whole document is prone to error. To avoid this, after a line is encoded one-dimensionally, only subsequent K-1 lines should follow the 2-D Encoding algorithm. The value of K is based on Vertical Resolution
– Std. vertical resolution: K = 2. – Optional higher vertical resolution: 200 Lines per 25.4 mm - K Value: 4
300 Lines per 25.4 mm - K Value: 6
400 Lines per 25.4 mm - K Value: 8
600 Lines per 25.4 mm - K Value: 12
800 Lines per 25.4 mm - K Value: 16
1200 Lines per 25.4 mm - K Value: 24

3) Concept of the Reference line and Coding line.: As we know in the Two-Dimensional encoding scheme a scanline ("Coding Line") is coded with the help of the previous line

that is called the “Reference Line “. Once the current scanline has been encoded completely it is then used as the Reference line for the next scan line.

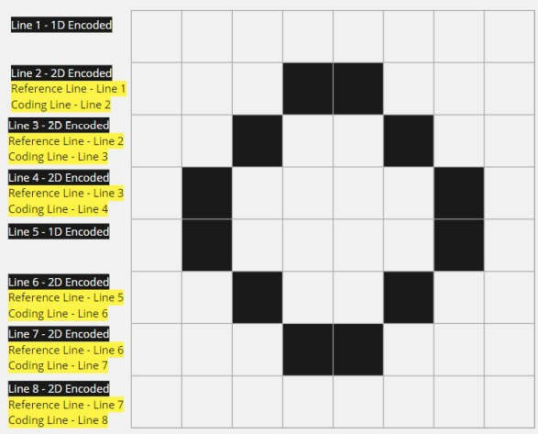

Fig. 5. Interpretation of Image based on K parameter.

4) Concept of the Changing Elements: In order to encode a scan line, we need some markings on the current line to be encoded and the previous line , so that we can compare them and use them as a reference based on the algorithm. The concept of the Changing elements gives us these valuable makings at different positions on both the coding line and the reference line. A **Changing element** is described as follows: An element belonging to a ”color” (i.e.. black / white) that is opposing from that of the preceding element on the current scan line. Using this concept, we can give names to the valuable markings that we can use for 2D encoding. a0 – The starting/Reference changing element for each iteration of the 2D encoding scheme. This pixel lies within the current scan line that is to be encoded and values a1, a2, b1, and b2 are set based on a0. For the first iteration, a0 is fixed as an Imaginary white pixel right before the first picture element in the scan line.

• **a1** - The next element of the opposite color after a0 in the scan line.

• **a2** - The next element of the opposite color after a1 in the scan line.

• **b1** - The changing element which is of the opposite color to the reference element a0 but on the previous/Reference line. b1 should be to the right of a0.

• **b2** - The following element of the opposite color after b1 in the reference line.

5) Coding Modes: As discussed in the previous section, based on the changing elements (a0, a1, a2, b1, b2) ,one of the three following modes can be chosen to perform the 2D Encoding.

• **Pass Mode:** If the point of b2 is situated to the left of a1, coding mode, Pass mode is identified. Pass mode is encoded with the codeword 0001, the new position of a0 is then switched to the element of the coding line which is below b2 in preparation for the next coding iteration.

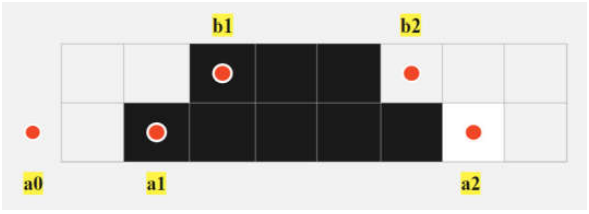

Fig. 6. Concept of Changing Elements.

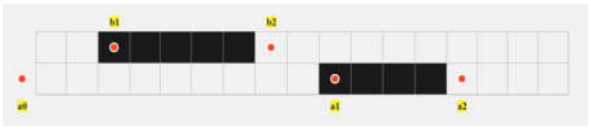

Fig. 7. Pass Mode Example.

• **Vertical Mode:** Vertical mode is identified when the relative distance between a1 and b1 is within 3 pixels to the left or right. The relative distance of a1b1 is represented by one of the values of V(0), VR(1), VR(2), VR(3), VL(1), VL(2), and VL(3), each of which has a unique codeword. The R and L subscripts tell us about the relative direction, R indicates that a1 is to the right of b1 and L indicates that point a1 is to the left of b1. The value within the brackets indicate the relative distance between a1 and b1. Once Vertical mode coding is complete, **the new value of a0 is switched to the current value a1.**

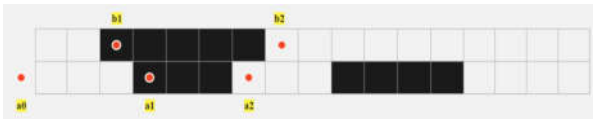

Fig. 8. Vertical Mode Example.

• **Horizontal Mode:** When we don’t detect Pass mode or Vertical mode, we can conclude it’s the coding mode of Horizontal type. In this mode, the relative number of pixels between a0-a1 and the pixels between a1-a2 are calculated are encoded using the 1D Huffman code words based on the respective colors of the runs. The 2D encoding is finally of the following format: Concatenation of H, M(a0a1), M(a1a2, where H is the H-mode/flag codeword 001 as shown in the two-dimensional code table (Figure 5.6). Once Horizontal mode encoding is completed, the new a0 is set as position of a2.

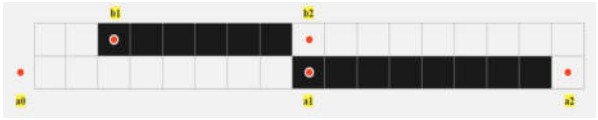

Fig. 9. Horizontal Mode Example.

Mode	Elements to be coded		Notation	Code word
Pass	b_1, b_2		P	0001
Horizontal	a_0a_1, a_1a_2		H	001 + M(a_0a_1) + M(a_1a_2) (Note 1)
Vertical	a_1 just under b_1	$a_1b_1 = 0$	V(0)	1
	a_1 to the right of b_1	$a_1b_1 = 1$	$V_R(1)$	011
		$a_1b_1 = 2$	$V_R(2)$	000011
		$a_1b_1 = 3$	$V_R(3)$	0000011
	a_1 to the left of b_1	$a_1b_1 = 1$	$V_L(1)$	010
		$a_1b_1 = 2$	$V_L(2)$	000010
		$a_1b_1 = 3$	$V_L(3)$	0000010

Fig. 10. 2-D Codeword Table

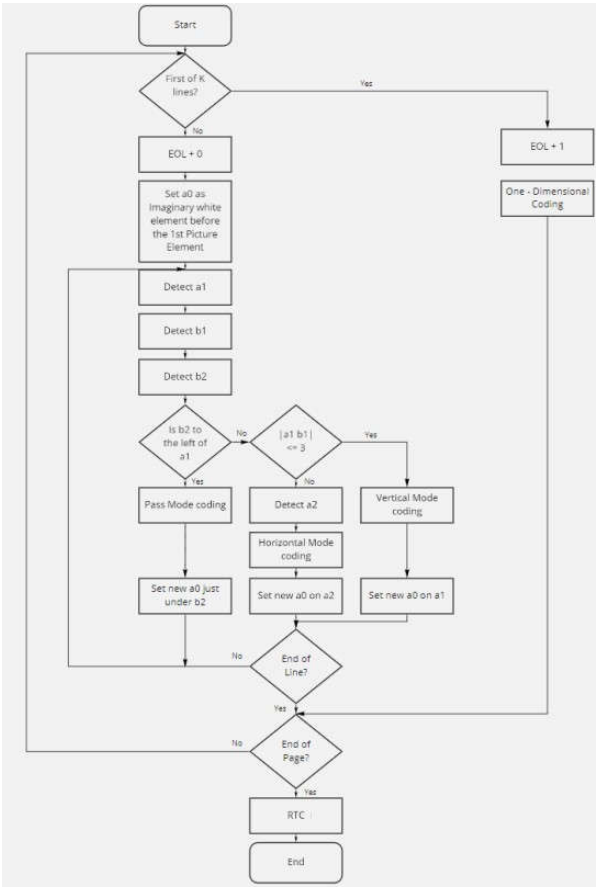

Fig. 11. Flowchart of T.4 2-D Coding

C. Procedure

Once the values of a0, a1, a2, b1, b2 are found, We can find an appropriate coding mode to start coding.

Step 1: Check is If pass mode is identified, if yes then it is coded using the codeword 0001 (Figure 5.6). After Pass mode coding is done the new value of a0 is set as the picture element right under b2 on the coding line. If a pass mode is not identified, then proceed to step 2.

Step 2: i. Find the absolute value of the relative distance between a1 and b1.

ii. If —a1b1— ¡= 3(Eg: Refer to the image shown in Figure 5.6), a1b1 is coded by the vertical mode. After vertical mode coding is done the new value of a0 is set as a1.

iii. If —a1b1— ¿ 3, as shown in Figure 5.6, In this case, the encoding is done using the Horizontal mode codeword ”001” followed by Huffman 1D codewords for the run lengths a0a1 and a1a2 respectively. After Horizontal mode coding is done the new value of a0 is set as a2.

This process is repeated until a0 reaches the end of the scanline and the whole scanline is encoded. For the 2D encoded next scan line the whole procedure is repeated again with the previous line as the reference line.

D. Processing Image Edges

Processing the First element: The first picture element a0, on every coding line, is imaginarily fixed at a position preceding the first picture element and is considered as a white picture element. The first run length on a line, a0a1 is replaced by a0a1 – 1. As a result, if the first actual picture element in the line is a black pixel, then it is concluded that horizontal coding mode is used. Hence the M(a0a1) will represent a white run of 0 length.

Processing the Last element: The 2D encoding keeps iterating in each line till it reaches the end of the line. Hence the coding of the line continues until a0 has been set to the last actual picture element of the coding line. We also consider an imaginary picture element after the last actual picture element of the current scan line. In case after setting the value of a0 we are not able to find another changing element on the scan line before the last picture element then a1 and a2 are set on the imaginary picture element just after the last picture element on the scan line. Also, if b1 and/or b2 are not detected at any time during the coding of the line, they are positioned on the imaginary changing element situated just after the last actual picture element on the reference line. In a case where a0 is set on the imaginary picture element we stop the iteration for that scan line and append the fill bits and the EOL codewords. After this the coding for the next scan line can begin.

E. EOL, Fill bits and RTC

• **EOL:** After encoding a coding line we must have a codeword that will indicate the receiver that whole line is completed and the following data belongs to the next line. Hence, we use the EOL (End of line) codeword which is represented as 000000000001 to perform the above task. Now to indicate if the next line is encoded One Dimensionally or Two dimensionally a tag bit is introduced after the EOL codeword.

Format: EOL + 1: When the succeeding line is coded One – Dimensionally.

EOL + 0: When the succeeding line is coded Two – Dimensionally.

• **Fill Bits:** Fill bits are inserted between an encoded Data and the end of line synchronization signal codeword, but is not inserted in Data. Fill must be added to ensure that each encoded line of data must start from a new byte boundary. Format: variable length string of 0s.

• **Return To Control (RTC):** This is represented by six consecutive line synchronization code words, i.e., 6 × (EOL +

1). It is used to indicate that the transmission of the encoded data is completed.

V. Group 4: 2-Dimensional Encoding

It is a modified version of the Group 3 2D encoding technique. Most of the 2D encoding techniques in Group 3 2D encoding are followed in Group 4 2D Encoding except for a few differences. The differences are explained below.

A. Difference between Group3-2D and Group4-2D

• The K parameter does not exist and every line is encoded 2 dimensionally with respect to the previous line. (Unbounded Vertical Resolution)

• In the case of T.6 2D encoding, the first line is encoded two-dimensionally, with an imaginary white line as a reference line.

• The concept of end of line (EOL) and Fill bits does not exist Instead, there exists an End of Facsimile Block (EOFB)

B. The End-of-facsimile block

The end-of-facsimile block (EOFB) code is added to the end of every coded facsimile block. The format of EOFB is as follows:

Format: 000000000001000000000001
24 bits

C. Pad Bits

Pad bits may be used after the end-of-facsimile block code if it is necessary to align on octet boundaries or to a fixed block size. The format used is as follows.

Format: Variable length string of 0s.

VI. Experiment and Results

In this section we present the results of compression achieved with each of the mentioned algorithms.Sample test images with distinct properties have been used to understand the impact of compression for different algorithms.Compression ratio is calculated and compared and important inferences have been made. Compression ratio is calculated as :

(1)

A. G3 1-D Encoding

Test Image 1: Compression Ratio = 84.39% (Fig. 12)
Test Image 2: Compression Ratio = 87.08% (Fig. 13)

B. G3 2-D Encoding

Test Image 1: (Fig. 14)
K = 2 Compression Ratio = 85.10%
K = 4 Compression Ratio = 85.47 %
K = 6 Compression Ratio = 85.56 %
K = 8 Compression Ratio = 85.75%
K = 12 Compression Ratio = 85.84 %
K = 16 Compression Ratio = 85.84 %
K = 24 Compression Ratio = 85.93%
Test Image 2: (Fig. 14)
K = 2 Compression Ratio = 90.31 %

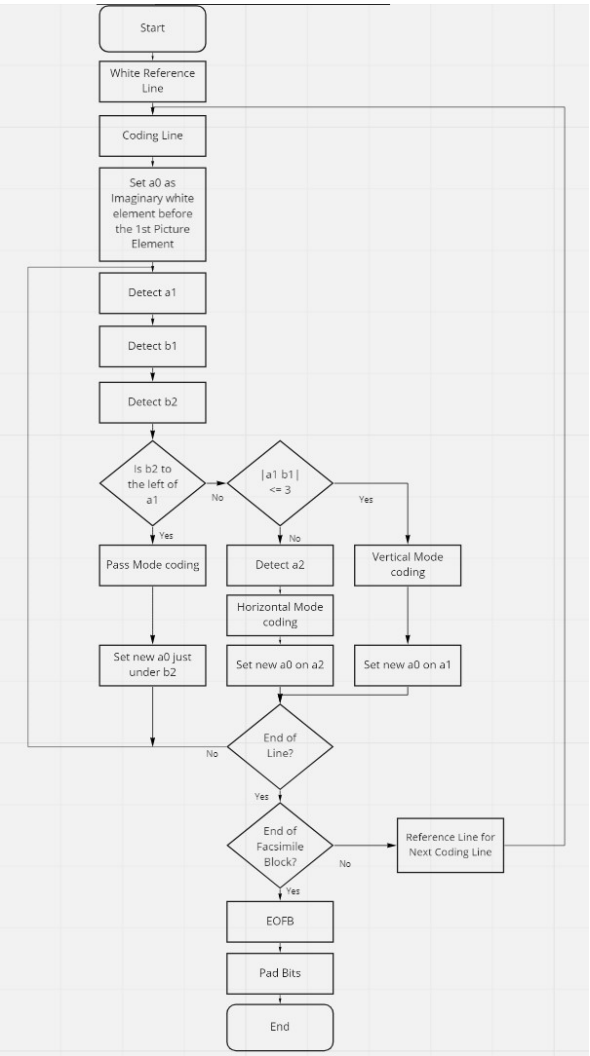

Fig. 12. Group 4 2D Encoding Flowchart

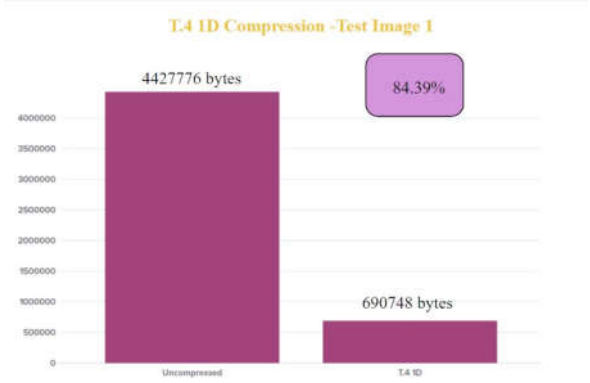

Fig. 13. Uncompressed vs Compressed file size - Test Image 1

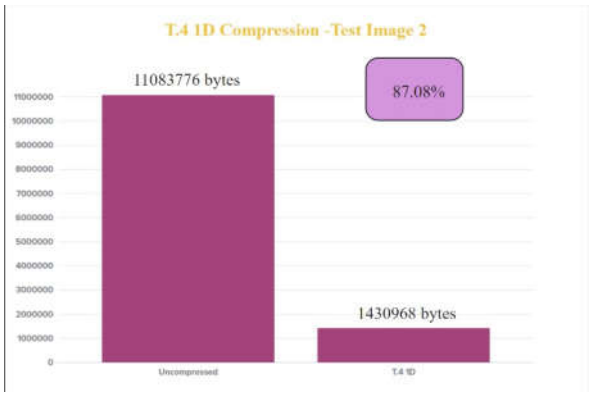

Fig. 14. Uncompressed vs Compressed file size - Test Image 2

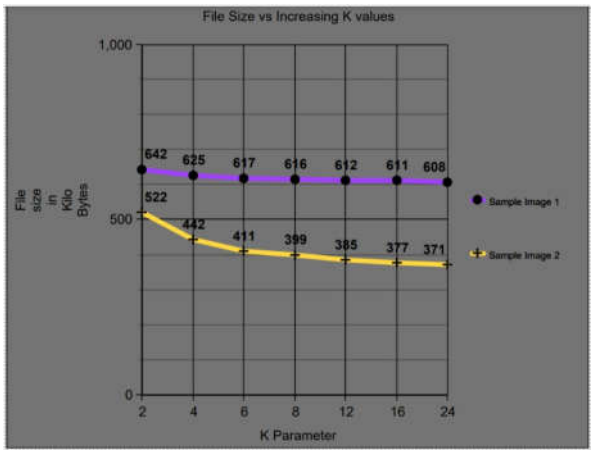

Fig. 15. Uncompressed vs Compressed file size

K = 4 Compression Ratio = 91.79 %
K = 6 Compression Ratio = 92.38 %
K = 8 Compression Ratio = 92.60 %
K = 12 Compression Ratio = 92.83 %
K = 16 Compression Ratio = 92.97 %
K = 24 Compression Ratio = 93.12 %

C. G4 2-D Encoding

Test Image 1: Compression Ratio = 86.86 % (Fig . 15)
Test Image 2: Compression Ratio = 94.30 % (Fig. 16)

VII. CONCLUSION

Images of type 1 have a higher compression ratio for 1D encoding when compared to 2D encoding.This is because 2D compression is not suitable for highly uncorrelated data. However, for Test Image 2, 2D performs a better encoding when compared to 1D encoding. Although, in a majority of the cases, T.6 2D encoding gives the best results as test images perform 2D encoding with an unbounded vertical parameter and do not have any EOL or fill bits present in it.

REFERENCES

[1] ITU-T, T.4, T-Terminals for Telematic Services, "Standardization of Group 3 Facsimile terminals for document transmission".

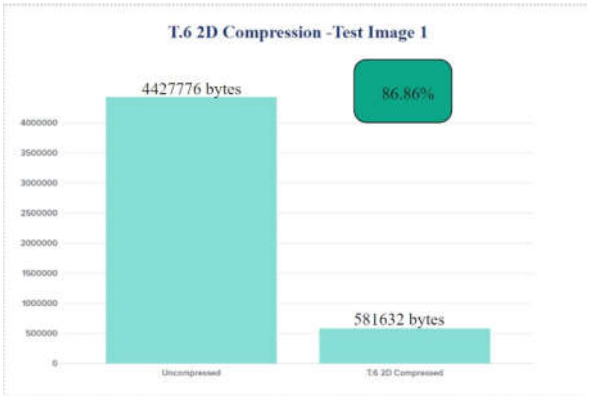

Fig. 16. Uncompressed vs Compressed file size - Test Image

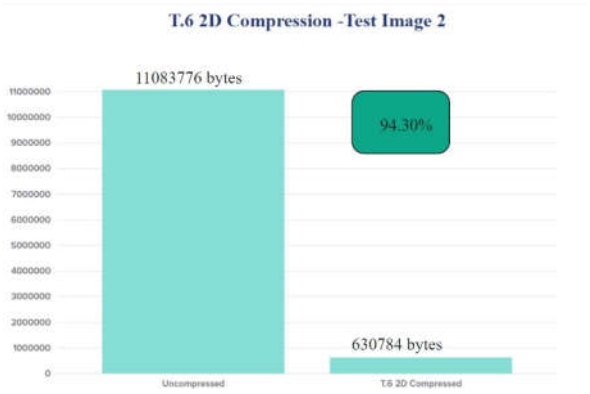

Fig. 17. Uncompressed vs Compressed file size - Test Image 2

[2] ITU-T, T.6, Terminal Equipment and Protocols for Telematic Services, "Facsimile coding schemes and coding control functions for Group 4 Facsimile Apparatus".
[3] Adobe Developers Association, "TIFF Revision 6.0", June 3, 1992.
[4]Encyclopedia of Graphics File Formats, 2nd Edition, Murray, James D., Van Ryper, William .
[5]AsTiffTagViewer , Aware Systems.