The IJICS (International Journal of Informatics and Computer Science)
Vol 9 No 2, July 2025, Page 51-59
ISSN 2548-8384 (online), ISSN 2548-8449 (print)
Available Online at https://ejurnal.stmik-budidarma.ac.id/index.php/ijics/index
DOI 10.30865/ijics.v9i2 .8921

Comparison of Elias Delta and Interpolative Coding Algorithms in Video File Compression

Nani Khairani*, Soeb Aripin, Meryance Viorentina Siagian
Faculty of Computer Science and Information Technology, Budi Darma University,Medan,Indonesia
Email: nanikhairani83@gmail.com, suefarifin@gmail.com, meryance1993@gmail.com
(* : nanikhairani83@gmail.com)
Submitted: **06/07/2025**; Accepted: **15/07/2024**; Published: **23/07/2025**

Abstrak- Large video file sizes can burden storage capacity and slow down data transmission, making effective compression methods essential. WebM, a commonly used video format supported by platforms such as YouTube and Skype, often results in large file sizes that demand storage efficiency. This study compares two lossless compression algorithms—Elias Delta Code and Interpolative Coding—for compressing WebM video files. Interpolative Coding applies a non-linear approach based on the entire content of the message, while Elias Delta Code is an integer-based algorithm efficient for encoding positive numbers. This research is motivated by the lack of direct comparative studies between these two algorithms in the context of video compression. The objective is to evaluate their performance based on compression ratio, processing time, and storage efficiency. The results are expected to provide recommendations on the most suitable compression algorithm for high-complexity video files.
Keywords: data compression; WebM video; Elias Delta Code; Interpolative Coding; compression ratio

1. INTRODUCTION

Data compression is a technique used to reduce the size of data in order to improve efficiency in storage and transmission. In the context of video files, large file sizes can burden storage capacity and slow down data transmission processes. Therefore, an effective compression method is needed to address these issues. WebM files are commonly found in video content. WebM is supported by several web browsers because it is sometimes used in HTML5 websites for video streaming. For example, YouTube uses the WebM video file format for all of its videos, ranging from 360p to higher resolutions. Similarly, Wikimedia and Skype also utilize the WebM format. Large video file sizes can negatively impact storage capacity and slow down data delivery. As a result, video compression is essential for reducing file sizes. Each compression algorithm has its own advantages and disadvantages, making it important to compare Interpolative Coding and Elias Delta Coding to determine which method is more efficient for compressing video files.

The Interpolative Coding algorithm is an unconventional method for assigning dynamic codes to data symbols. Unlike traditional compression techniques, the code assigned to each individual symbol is not static; instead, it depends on the entire message rather than just the symbol and its frequency. The entire message to be encoded must be available to the encoder in advance. The encoding process involves scanning the message in a specific order—rather than linearly from beginning to end—and assigning codewords to symbols as they are encountered. On the other hand, Elias Delta Coding is one of the three universal coding methods developed by Peter Elias. Elias Delta adds a layer of encoding to binary representations. It is typically used for encoding positive integers and is effective in scenarios where smaller values occur more frequently, allowing for compact representation. [1].

Previous research conducted by Jamiatul Sisca, titled “Implementation of the Elias Delta Code Algorithm for Video File Compression in a Video Downloader Application”, concluded that the Elias Delta Code algorithm can be effectively applied to compress video file sizes, resulting in smaller file sizes after compression [2].

In another study conducted by Desvika Riyansyah, titled "Design of a Video File Compression Application Using the Interpolative Coding Algorithm", it was concluded that the implementation of the Interpolative Coding algorithm achieved a compression ratio of 47%[3]. Research conducted by Michael Simangunsong, titled "Comparison of Elias Delta Code and Unary Coding Algorithms in Forensic Image Compression", concluded that both Elias Delta Code and Unary Coding are algorithms capable of compressing forensic images [4]. Another study by Riyo Oktavianty Finola, titled "Implementation of the Interpolative Coding Algorithm for Audio File Compression", found that the application of the Interpolative Coding algorithm to audio files can significantly reduce file size and save storage memory [5].

This study aims to compare two compression algorithms to determine which one is more accurate and efficient in the process of video file compression. The algorithms under comparison are Elias Delta Code and Interpolative Coding, both of which are designed for data compression tasks.

Interpolative Coding has demonstrated promising compression ratios in previous experiments, particularly in audio and video compression, achieving up to 47% compression in certain cases. However, most of these studies have been

The IJICS (International Journal of Informatics and Computer Science)
Vol 9 No 2, July 2025, Page 51-59
ISSN 2548-8384 (online), ISSN 2548-8449 (print)
Available Online at https://ejurnal.stmik-budidarma.ac.id/index.php/ijics/index
DOI 10.30865/ijics.v9i2 .8921

conducted independently and focused on specific types of data, such as images or audio, without performing a direct comparative analysis between the two algorithms in the context of video files.

So far, no study has been found that directly compares the effectiveness of Elias Delta Coding and Interpolative Coding in compressing WebM-formatted video files, in terms of compression ratio, compression time, or storage efficiency. Furthermore, there has been no evaluation of the performance of these algorithms under complex video scenarios involving varying resolutions and extended durations. This gap in the literature serves as the foundation for the present study.

2. RESEARCH METHODOLOGY

In the research framework, the stages carried out in the study are outlined. The research framework consists of several systematically related phases. These stages are essential to facilitate the execution of the research in a structured and efficient manner.

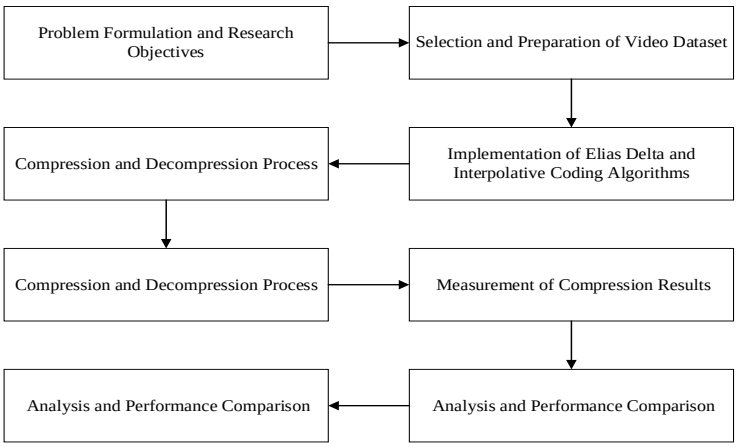

Figure 1. Research Stages

2.1 Data Compression

Data compression is the process of reducing data size by eliminating redundancy without losing essential information. In the context of video, compression is crucial because video files are typically large and require efficient storage and transmission[6][7].

Compression can be classified into two types:

1. Lossless Compression: No information is lost, and the original data can be fully restored.
2. Lossy Compression: Some information is discarded to achieve higher efficiency. This method is commonly used for image and video compression.

2.2 Moving Picture Exprort Gruop (MPEG)-4

Initially, MPEG Video Layer-4 was widely used by computer users. Files encoded with MPEG Video Layer-4 are typically saved with the .mp4 file extension. Subsequently, MPEG Audio Layer formats also became popularly associated with the MP4 file type [8]. An MPEG file consists of small units called frames. Typically, each frame is self-contained and includes a header that holds metadata about that particular frame. Since MPEG files do not contain a global file header, the file can be split at almost any point, as long as the cuts occur at frame boundaries [9]. n contrast, MP3 files (based on MPEG-1 Audio Layer III) may contain frames that are interdependent, where certain frames rely on data from previous ones—making arbitrary splitting less feasible. [10].

2.3 Elias Delta Code Algorithm

Elias Delta Code is one of the three Elias coding schemes introduced by Peter Elias. This algorithm extends binary encoding (β) by incorporating a length-prefix mechanism, making it more efficient for encoding larger numbers. Elias Delta Code is

The IJICS (International Journal of Informatics and Computer Science)
Vol 9 No 2, July 2025, Page 51-59
ISSN 2548-8384 (online), ISSN 2548-8449 (print)
Available Online at https://ejurnal.stmik-budidarma.ac.id/index.php/ijics/index
DOI 10.30865/ijics.v9i2 .8921

primarily used for encoding positive integers, particularly in applications where the values vary significantly in size and compact representation is important [2][11].
Encoding Rules Using Elias Delta Code The rules for encoding a number using Elias Delta Code are as follows:

1. Write the number n in binary. The most significant bit (MSB) will always be 1.
2. Count the number of bits in n, remove the leftmost bit (MSB), and prefix the remaining bits with the binary representation of the bit length of n.
3. Subtract 1 from the bit length obtained in step 2, and prepend that many zeros to the code.

When these steps are applied to n = 17, the result is as follows: $17 = 2^4 + 1 \rightarrow$ binary: 10001, which has 5 bits. The gamma code of N + 1 (i.e., 5) is: 00101 Removing the leading 1 from 10001 gives: 0001 Therefore, the Elias Delta Code for 17 is: 00101 | 0001
A table showing the Elias Delta Codes for the first 18 positive integers is presented below.

Table 1. Elias Delta Code

1=20	10=2
2=21	11=2
3=21	12=2
4=22	13=2
5=22	14=2
6=22	15=2
7=2	16=2
8=2	17=2
9=2	18=2

2.3 Algoritma Interpolative Code

Interpolative coding is an unconventional method for assigning dynamic codes to data symbols. It differs from traditional approaches because the code assigned to a symbol is not static and depends on the entire message rather than on the symbol itself or its probability. The entire message must be available beforehand in order to perform encoding [12]. The encoding is carried out by scanning the message in a specific, non-sequential order, rather than linearly from beginning to end, and assigning codewords when symbols are encountered. As a result, the codeword assigned to a symbol depends on the order of symbols within the message [13].

3. RESULT AND DISCUSSION

This stage of the research plays a crucial role in determining the subsequent steps. The study involves a comparative analysis of video file compression, specifically using the Elias Delta Code and Interpolative Coding algorithms. The comparison is intended to identify which algorithm is more accurate and efficient in reducing the file size of WebM-format video files. The primary goal of video file compression is to optimize storage space.

Before performing the compression, the video file is first examined in hexadecimal format using the HxD application, to understand how the data is structured at the byte level. Below is an example of a WebM video file that undergoes both compression and decompression.
1. Analysis of the Video File Compression Process Using Elias Delta Code
a. Importing the File

Table 2. Sample Data

Nama File	Video
Extension File	Webm
Size	12,1 MB

The IJICS (International Journal of Informatics and Computer Science)
Vol 9 No 2, July 2025, Page 51-59
ISSN 2548-8384 (online), ISSN 2548-8449 (print)
Available Online at https://ejurnal.stmik-budidarma.ac.id/index.php/ijics/index
DOI 10.30865/ijics.v9i2 .8921

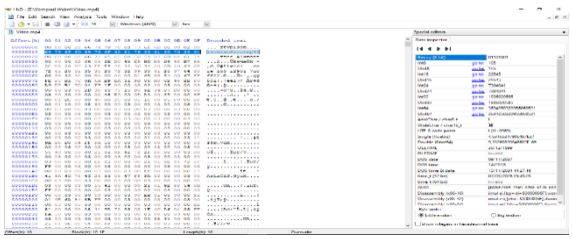

Figure 2. Contents of the Video File from HxD Output

As a result, the hexadecimal values of the sample video file were obtained. In the manual calculation, 16 hexadecimal characters from the sample video file were used. These hexadecimal values were taken from left to right.

a. Reading the File Contents

The hexadecimal values extracted from the video file are as follows:
69, 73, 6F, 6D, 69, 73, 6F, 32, 61, 76, 63, 31, 6D, 70, 34, 31.
These data values are entered into a table for frequency analysis. The frequency analysis is performed by counting the number of times each unique value appears in the dataset. The results of the frequency reading can be seen in the table below:

Table 3. File Values

Values	Frekuensi
69	2
73	2
6F	2
6D	2
32	1
61	1
76	1
63	1
31	2
70	1
34	1
Total Value	16

Sorting the characters from highest to lowest frequency. The order of the values can be seen in the table below:

Table 4. Sample Video Bit Values

Nilai		Bit	Frek	Bit x Frek
Hexa	Biner			
69	01101001	8	2	16
73	01110011	8	2	16
6F	01101111	8	2	16
6D	01101101	8	2	16
31	00110001	8	2	16
32	00110010	8	1	8
61	01100001	8	1	8
76	01110110	8	1	8
63	01100011	8	1	8
70	01110000	8	1	8
34	00110111	8	1	8
Total Bit				128

Based on the table above, one hexadecimal value (character) is worth 8 bits of a binary number. Up to 16 hexadecimal numbers have a binary value of 128 bits. To convert a unit to a byte then the total number of bits is shared 8. So 182/8 = 16.

b. Forming *the Elias Delta Code Table*

The IJICS (International Journal of Informatics and Computer Science)
Vol 9 No 2, July 2025, Page 51-59
ISSN 2548-8384 (online), ISSN 2548-8449 (print)
Available Online at https://ejurnal.stmik-budidarma.ac.id/index.php/ijics/index
DOI 10.30865/ijics.v9i2 .8921

The rules in the formation of number codes using *elias delta code can be seen in the previous chapter. The elias delta code* can be seen in the table below.

Table 5. Elias Delta *Code*

N	Elias Delta Code
1	1
2	0100
3	0101
4	01100
5	01101
6	01110
7	01111
8	00100000
9	00100001
10	00100010
11	00100011
12	00100100
13	00100101

The next process is to compress the value from the sample with *the elias delta code* obtained from the table above. The process of compressing the sample video file can be seen in the following table:

Table 6. Compression of *FileVideo Sample* Values with Elias Delta Code

N	Hexa values	Elias Delta Code	Bit	Frek	Bit x Frek
1	69	1	1	2	2
2	73	0100	4	2	8
3	6F	0101	4	2	8
4	6D	01100	5	2	10
5	31	01101	5	2	10
6	32	01110	5	1	5
7	61	01111	5	1	5
8	76	00100000	8	1	8
9	63	00100001	8	1	8
10	70	00100010	8	1	8
11	34	00100011	8	1	8
		Total Bit			80

From calculations table in top after compressed with Using *Elias delta code.*80 bit. To be changed become unit byte Mark divided 8 i.e 80/8 = Sec. 10.

1. Perform the result of *the elias delta code bit string*
   the hexadecimal values before compression i.e. 69, 73, 6F, 6D, 69, 73, 6F, 32, 61, 76, 63, 31, 6D, 70, 34, 31 (without commas and spaces) become binary bit values :
   "10100010101100101000101011100111001000000100001011010110000100**0100010001101101**".

The total overall bit length after the addition of bits is 80+8+8=96. Next, separate the bits into groups. Each group consists of 8 bits:

10100010 10110010 10001010 11100111 10010000 00010000 10110101 10000100 01000100 01101101 **0000000100001001**

The compressed values can be seen in the table below.

Table 7. Compressed *Hexadecimal* Values

Decimal Value	Character
162	¢
178	²
138	Š
231	Ç

The IJICS (International Journal of Informatics and Computer Science)
Vol 9 No 2, July 2025, Page 51-59
ISSN 2548-8384 (online), ISSN 2548-8449 (print)
Available Online at https://ejurnal.stmik-budidarma.ac.id/index.php/ijics/index
DOI 10.30865/ijics.v9i2 .8921

144	90
16	10
181	µ
132	„
68	D
109	M
1	2
9	

7 – n + "1"
7 – 0 + "1" = **00000001**
Final Bit 9 – n
Final Bit = 9 – 0 = 9 = **00001001**
Data size before compression = 128/8 = 16 bytes
Compressed data size = 96/8 = 12 bytes
Based on this data, the compression performance can be calculated, namely:

- *Compression Ratio* (Cr)
  $Cr=\frac{Ukuran\ data\ Sesudah\ Dikompresi}{Ukuran\ data\ Sebelum\ \ Dikompresi}x\ 100\%$
  $Cr=\frac{12}{16}x100\%$
  $Cr = 75\%$
- *Space Saving*
  SS = Data size before compression – Data size after compression
  SS = 128- 92
  SS = 36

1. Analyze the decompression process of *video files* using *Elias Delta Code*

Analysis of all bits from the previous compression results is carried out to carry out the decompression process. As for the overall bits and combined compression results, namely :
"10100010 10110010 10001010 11100111 10010000 00010000 10110101 10000100 01000100 01101101 00000001 00001001".

1. The initial decompression process is to read the *value of the flag* bit of all the bits by changing the value of the last 8 bits to a decimal value such as 00001001 = 9. The value of 9 is 8 bit values that are at the *time of padding* before *adding flag* bits.
2. Next remove *the bit flag* and *padding* from the overall value of the bits so that menjadi "10100010 10110010 10001010 11100111 10010000 00010000 10110101 10000100 01000100 01101101".
1. Bit checking is by checking bits from the first bit with the *elias delta code table.* If a bit is found that matches the *delta elias code table* , then change the appropriate string value so that you will get the result of the table as below.

Table 8. Decompression results using elias delta code

Char	Character	Biner
¢	162	10100010
²	178	10110010
Š	138	10001010
Ç	231	11100111
90	144	10010000
10	16	00010000
µ	181	10000100
„	132	01000100
D	68	01101101

So the results are "69, 73, 6F, 6D, 69, 73, 6F, 32, 61, 76, 63, 31, 6D, 70, 34, 31".

Compression and Decompression Process with Interpolative Coding Algorithm Sorting to the smallest frequency.

Table 9. Nilai *Bit* Video Sample

Nilai		Bit	Frek	Bit x Frek
Hexa	Biner			

The IJICS (International Journal of Informatics and Computer Science)
Vol 9 No 2, July 2025, Page 51-59
ISSN 2548-8384 (online), ISSN 2548-8449 (print)
Available Online at https://ejurnal.stmik-budidarma.ac.id/index.php/ijics/index
DOI 10.30865/ijics.v9i2 .8921

69	01101001	8	2	16
73	01110011	8	2	16
6F	01101111	8	2	16
6D	01101101	8	2	16
31	00110001	8	2	16
32	00110010	8	1	8
61	01100001	8	1	8
76	01110110	8	1	8
63	01100011	8	1	8
70	01110000	8	1	8
34	00110111	8	1	8
	Total Bit			128

So 128/8 = 16.

1. Forming Interpolative Coding Tables

Table 10. Kode *Interolative Coding*

N	Kode Elias Delta
1	1001
2	1111
3	100
4	10111
5	100
6	1000
7	010
8	100011
9	010
10	100
11	010
12	010
13	1001

Interpoltive coding obtained from the table above. As for the *file compression process*

Table 11. Compression of Sample Video File Values with Interpolative Coding

N	Hexa Values	Kode Interpolative	Bit	Frek	Bit x Frek
1	69	1001	4	2	8
2	73	1111	4	2	8
3	6F	100	3	2	6
4	6D	10111	5	2	10
5	31	100	3	2	6
6	32	1000	4	1	4
7	61	010	3	1	3
8	76	100011	6	1	6
9	63	010	3	1	3
10	70	100	3	1	3
11	34	010	3	1	3
		Total Bit			60

From the calculation of the table above after being compressed using *elias delta codeis 60 bits. To convert it into a byte unit, it is divided by 8 i.e. 60/8 = 7.5.*

a. Performing the result of *the elias delta interpolative coding bit string*

7 – n + "1"
7 – 0 + "1" = **00000001**
Bit Akhir 9 – n
Bit Akhir = 9 – 0 = 9 = **00001001**

b. *String* bits that have been added

The IJICS (International Journal of Informatics and Computer Science)
Vol 9 No 2, July 2025, Page 51-59
ISSN 2548-8384 (online), ISSN 2548-8449 (print)
Available Online at https://ejurnal.stmik-budidarma.ac.id/index.php/ijics/index
DOI 10.30865/ijics.v9i2 .8921

The total overall bit length after the addition of bits is 60+4+8=96. Next, separate the bits into groups. Each group consists of 8 bits as shown below

10011111100101111001111110010000101000110101001011110001 010**0000100000101**

String Bit Divisions

The compressed values can be seen in the table below.

Table 12. Compressed Hexadecimal Values

Decimal Value	Character
159	Ÿ
151	—
159	Ÿ
144	?
163	£
82	R
241	Ñ
65	A
5	

Data size before compression = 128/8 = 16 bytes

Compressed data size = 72/8 = 9 bytes

Based on this data, the compression performance can be calculated, namely:

Compression Ratio (Cr)

$$Cr=\frac{Ukuran\ data\ Sesudah\ Dikompresi}{Ukuran\ data\ Sebelum\ \ Dikompresi}x\ 100\%$$

$$Cr=\frac{9}{16}x100\%$$

$$Cr = 56, 25\%$$

Space Saving

SS = data size before compression – data size after compression

SS = 128- 72

SS = 56

c. Comparison of Elias Delta Code Algorithm and Interpolative Coding

The comparison of the elias delta code algorithm and interpolative coding is the result of the compression ratio of the elias delta code algorithm of 75% and space saving of 36, while of the large interpolative coding algorithm the compression ratio is 56.25% and space saving is 56. So from the results obtained, the compression ratio of the elias delta code algorithm is compared to the interpolative coding algorithm.

4. CONCLUSION

Compression of video files can be done by converting them into hexadecimal shapes with the application of hxd. The hexadecimal numbers obtained are processed with *elias delta code algorithms and interpolative coding so as to produce new hexadecimal number values from the video file that has been calculated.* After the video file compression process, the compression ratio and space saving are obtained from elias delta code and interpolative coding. The compression and space saving ratios of the two algorithms are used as a comparison of the elias delta code algorithm and interpolative coding. The compression ratio of elias delta code is 75% and the compression ratio of interpolative coding algorithms is 56.25%. It can be said that the compression of video files with the elias delta code algorithm gets a greater compression ratio.

The IJICS (International Journal of Informatics and Computer Science)
Vol 9 No 2, July 2025, Page 51-59
ISSN 2548-8384 (online), ISSN 2548-8449 (print)
Available Online at https://ejurnal.stmik-budidarma.ac.id/index.php/ijics/index
DOI 10.30865/ijics.v9i2 .8921

REFERENCES

[1] N. F. Rizky, S. D. Nasution, and F. Fadlina, "Penerapan Algoritma Elias Delta Codes Dalam Kompresi File Teks," *Build. Informatics, Technol. Sci.*, vol. 2, no. 2, pp. 109–114, 2020, doi: 10.47065/bits.v2i2.138.
[2] J. Sisca, "Penerapan Algoritma Elias Delta Code Untuk Kompresi File Video Pada Aplikasi Video Downloader," vol. 1, no. 4, pp. 254–264, 2021.
[3] D. Riyansyah, "Perancangan Aplikasi Kompresi File Video Menggunakan Algoritma Interpolative Coding," *KOMIK (Konferensi Nas. Teknol. Inf. dan Komputer)*, vol. 3, no. 1, pp. 392–397, 2019, doi: 10.30865/komik.v3i1.1618.
[4] M. Simangunsong, "Perbandingan Algoritma Elias Delta Code Dan Unary Coding Dalam Kompresi Citra Forensik," vol. 12, no. 1, pp. 18–26, 2020.
[5] R. O. Finola, "Penerapan Algoritma Interpolative Coding Untuk Kompresi File Audio," *KOMIK (Konferensi Nas. Teknol. Inf. dan Komputer)*, vol. 3, no. 1, pp. 378–384, 2019, doi: 10.30865/komik.v3i1.1616.
[6] K. Sayood, *Introduction to Data Compression (5th ed.)*. Morgan Kaufmann Publishers, 2017.
[7] G. Salomon, David and Motta, *Handbook of Data Compression*. Springer, 2020. doi: 10.1007/978-3-030-38879-5.
[8] K. R. Rao and J. J. Hwang, *Techniques and Standards for Image, Video, and Audio Coding*. New York: Springer, 2011. doi: 10.1007/978-1-4419-1818-1.
[9] R. Iqbal, F. Doctor, A. Khelifi, and C. Maple, "A review on video streaming techniques, challenges and opportunities in 5G networks," *Multimed. Tools Appl.*, vol. 80, pp. 26203–26234, 2021, doi: 10.1007/s11042-021-11040-x.
[10] D. Iqbal, "Implementasi Algoritma Levenstein Untuk Kompresi File Video Pada Aplikasi Chatting Berbasis Android," *KOMIK (Konferensi Nas. Teknol. Inf. dan Komputer)*, vol. 3, no. 1, pp. 266–273, 2019, doi: 10.30865/komik.v3i1.1601.
[11] A. Moffat and A. Turpin, *Compression and Coding Algorithms*. Boston, MA: Springer, 2002.
[12] D. Salomon and G. Motta, *Handbook of data compression*. 2010. doi: 10.1007/978-1-84882-903-9.
[13] R. V. Giuseppe Ottaviano, "Partitioned Elias-Fano Indexes," *ACM Trans. Inf. Syst.*, 2020, doi: 10.1145/3386259.