Bulletin of Data Science
ISSN 2745-7966 (Media Online)
Vol 3, No 1, October 2023
Hal 149-159
https://ejurnal.seminar-id.com/index.php/bulletinds

Kompresi File Video Menggunakan Algoritma Yamamoto's Recursive Code

Ronaldo P. Butar-Butar*

Fakultas Ilmu Komputer Dan Teknologi Informasi, Teknik Informatika, Universitas Budi Darma Medan Indonesia
Email: ronaldopartogibutarbutar@gmail.com

Abstrak−Pada saat ini sudah sangat lumrah digunakan disegala aspek, mulai dari iklan, film, bahkan pembelajaran saat masa pandemi covid-19 menggunakan media video sebagai media pembelajarannya. Ukuran file video yang besar dapat mengurangi ruang kosong *(space)* pada media penyimpanan seperti *hardisk*, *flasdisk* dan sebagainya. Semakin banyak video yang disimpan, maka semakin sedikit ruang kosong yang tersisa, atau bahkan tidak tersisa lagi ruang kosong pada media penyimpanan. Hal ini tidak akan menimbulkan masalah jika media penyimpanan yang digunakan memiliki kapasitas yang besar, namun tidak semua orang memiliki media penyimpanan yang besar. Salah satu cara untuk mengatasi masalah tersebut yaitu dengan mengunakkan teknik kompresi. Kompresi data adalah suatu proses pengubahan sekumpulan data menjadi bentuk suatu bentuk kode untuk menghemat kebutuhan tempat penyimpanan data. Atas dasar tersebut Penerapan *Algoritma Yamamoto's Recursive Code* Dalam Mengkompresi File Video dibuat agar dapat membantu pengguna mengkompresi file video yang awalnya berukuran besar menjadi ukuran terkecilnya. Kompresi video berhadapan dengan kompresi data video digital, Kompresi adalah sebuah konversi data ke sebuah format yang lebih kecil, biasanya dilakukan sehingga data dapat disimpan atau disalurkan lebih efisien. Proses pengembalian data yang sudah dikecilkan tersebut disebut dekompresi.

Kata Kunci: Kompresi video, Algoritma *Yamamoto's Recursive Code*

Abstract−At this time it is very common to use it in all aspects, from advertising, films, even learning during the COVID-19 pandemic using video media as a learning medium. Large video file sizes can reduce free space on storage media such as hard drives, flash drives and so on. The more videos that are stored, the less free space is left, or even no free space is left on the storage media. This will not cause a problem if the storage media used has a large capacity, but not everyone has a large storage media. One way to overcome this problem is to use compression techniques. Data compression is a process of converting a set of data into a form of code to save the need for data storage. On this basis, the application of Yamamoto's Recursive Code Algorithm in Compressing Video Files is made in order to help users compress video files that are initially large into their smallest size. Video compression is the opposite of digital video data compression. Compression is the conversion of data to a smaller format, usually done so that data can be stored or transmitted more efficiently. The process of recovering data that has been reduced is called decompression.

Keywords: Video compression, Yamamoto's Recursive Code Algorithm

1. PENDAHULUAN

Penggunaan media video pada saat ini sudah lumrah digunakan disegala aspek, mulai dari iklan, film, bahkan pembelajaran saat masa pandemi covid-19 menggunakan media video sebagai media pembelajarannya. Ukuran file video yang besar dapat mengurangi ruang kosong (*space*) pada media penyimpanan seperti hardisk, flasdisk dan sebagainya. Semakin banyak video yang disimpan, maka semakin sedikit ruang kosong yang tersisa, atau bahkan tidak tersisa lagi ruang kosong pada media penyimpanan. Hal ini tidak akan menimbulkan masalah jika media penyimpanan yang digunakan memiliki kapasitas yang besar, namun tidak semua orang memiliki media penyimpanan yang besar. Salah satu cara untuk mengatasi masalah tersebut yaitu dengan mengunakkan teknik kompresi.

Teknik kompresi dapat dijadikan solusi untuk masalah pengurangan ukuran file. Saat ini sudah banyak aplikasi untuk melakukan kompresi, seperti WinRAR, 7Zip, WinZip, dan lain sebagainya. Proses kompresi pada file video dilakukan agar ukuran file video menjadi lebih kecil sehingga banyak video bisa disimpan di media penyimpanan, atau bisa lebih menghemat ruang kosong pada media penyimpanan[1]. Teknik kompresi memiliki 2 (dua) jenis, yaitu *lossy compression* dan *lossless compression*. Lossy Compression merupakan jenis kompresi yang pada prosesnya terdapat beberapa bit yang hilang[2], sedangkan lossless compression merupakan jenis kompresi yang dalam proses kompresinya tidak ada kehilangan bit dan datanya bisa dikembalikan seperti semula[3].

Berdasarkan penelitian Hengki Tamando sihotang pada tahun 2018, untuk mengimplementasikan algoritma arithmetic coding untuk aplikasi kompresi data video dan audio, menyimpulkan ukuran file sebelum kompresi sama dengan ukuran file sesudah didekompresi yang berarti bahwa tidak ada data yang hilang selama proses kompresi file Mp4 lebih besar dibandingkan dengan kompresi file Mp3, algoritma Arithmetic Coding tidak optimal dalam melakukan kompresi audio dan video dapat dilihat pada rasio kompresi yang rendah dan grafik yang tidak stabil[4].

Penilitian yang dilakukan oleh I Gede Rasagama pada tahun 2020, untuk pengembangan model pembelajaran getaran berbasis video youtube untuk meningkatkan pemahaman konsep mahasiswa politeknik, menyimpulkan Model pembelajaran memuat konsep yang komprehensif, mudah diakses jika ada pengulangan, hemat tenaga bagi dosen dalam menjelaskan, dan resource untuk 1 sub pokok bahasan dapat dari beberapa video beragam dengan kreator berbeda, Video-video yang digunakan dalam pembelajaran 1 sub pokok bahasan masih berupa kompilasi beberapa file, resource yang berbahasa Indonesia terbatas, memuat konten dengan tujuan pembelajaran yang bersifat mengikat sehingga perlu penyesuaian dengan tuntutan kurikulum yang berlaku[5].

Penelitian lainnya juga telah dilakukan oleh Meri Sri Wahyuni pada tahun2019, untuk analisa membandingkan hasil kompresi file mpeg-4 dan flv menggunakan algoritma huffman dan lz77, menyimpulkan metode Huffman dilakukan

dengan mengubah nilai bit video menjadi kode Huffman dan Kompresi file video MPEG-4 dan FLV dilakukan dengan menggabungkan kode Huffman yang sama dan membentuk kode baru, Kompresi file video MPEG-4 dan FLV dilakukan dengan menggabungkan kode Huffman yang sama dan membentukkode baru Proses kompresi yang dihasilkan adalah lossles kompresion yang dapat mengembalikan file video ke ukuran semula tanpa ada kerusakan pada file video[6].

Pada tahun 2016 dilakukan penelitian Sudaryanto dkk, untuk pengaruh load balancing pada pemrosesan paralel untuk kompresi video, menyimpulkan dengan adanya ORDG_EDODQFLQJ pada pemrosesan paralel pembagian beban komputasi pada mesin server kompresor menjadi labih bijaksana/adil sesuai dengan kesibukan (CPU Usage) dan Konsep pemrosesan paralel load balancing yang diimplementasikan ke sistem kompresi video berhasil mempersingkat waktu kompresi bila dibandingkan dengan pemrosesan paralel non load balancing, dengan rata-rata nilai speed up sebesar 8,07% lebih cepat dari pemrosesan paralel QRQ_ORDG_EDODQFLQJ dengan dua kompresor; 37,57% dengan tiga kompresordan 41,24% kali dengan empat kompresor. Adapun tingkat efisiensi prosesor sebesar 61,36% dengan dua kompresor, 60,46% dengan tiga kompresor, dan 56,34% dengan empat kompresor[7].

Ada pun penelitian yang dilakukan oleh Haruno Sajati dkk pada tahun 2018, untuk the audio video of web-based compression with ffmpeg, menyimpulkan bahwa Hasil pengujian dan perbandingan antara kompresi video tanpa melakukan kompresi terhadapa audio dan kompresi video dan audio dengan web kompresi ini ternyata didapat hasil dengan melakukan kompresi audio didalam sebuah file video dapat menurunkan ukuran sekitar 1-2Mb dan Pengujian dengan membandingkan ukuran video asli dari Youtube dan video hasil web kompresi video menunjukan ada perbedaan ukuran yang cukup signifikan mulai dari 5- 20Mb, Perbandingan antara kualitas video asli dan video hasil kompresi dari web kompresi video ini menunjukan kualitas yang relatif lebih rendah dan tidak memenuhi standar kualitas yaitu 30dB[8].

Berdasarkan penelitian-penelitian yang terdahulu tersebut, maka dapat disimpulkan bahwa kompresi *file* video sangat membutuhkan penyimpanan yang sangat besar yang dimana semakin bagus kualitas video yang dihasilkan maka ukuran yang dihasilkan juga semakin besar. Ada beberapa dalam media penyimpanan yaitu *Online Storage* dan *Hard disk Drive* yang dimana penyimpanan ini tidak menjamin ketersediaan tempat dalam menyimpan sebuah file jika file yang dihasilkan semakin besar atau bisa dibilang semakin bertambah banyak. Dalam pengiriman file video akan gagal jika ukuran file video lebih besar daripada kapasitas memori penyimpanan yang tersedia. Salah satu format file video yang sering digunakan adalah format MP4 MPEG-4 dan FLV yang dimana dikembangkan oleh *Microsoft* yang terdiri dari susunan titik (pixel) dan tersimpan di memori komputer[9]. Format MP4 MPEG-4 dan FLV adalah format yang minim kompresi sehingga ukurannya lebih besar dan sangat efektif dalam menyimpan suatu *file*[10].

Banyak algoritma kompresi yang ada, salah satunya adalah yamamoto's recursive code. Algoritma yamamoto's recursive code termasuk dalam jenis lossless compression, dimana tidak terdapat kehilangan bit dalam proses kompresinya. Penelitian terkait tentang algoritma yamamoto's recursive code ini masih sedikit, dimana algoritma ini hanya dibahas pada buku kompresi, dan belum terdapat artikel yang mempublikasikannya[9]. Yamamoto Recursive Code adalah kode recursif untuk suatu angka positif dan setiap urutan yang dimasukkan dapat digunakan sebagai delimiter, berbeda dengan universal code lainnya yang menggunakan bit "0" sebagai delimiter seperti Elias omega code, Even-Rodeh code, Stout code, dan lain-lain[10]. Delimiter yang dipakai pada Algoritma Yamamoto Recursive Code lebih pendek dari $log2 * n$ dalam hampir dari semua angka bulat positif dibandingkan algoritma sebelumnya [11]. Berdasarkan uraian di atas, tujuan dari penelitian ini adalah untuk menyelesaikan studi dengan judul Kompresi File Video dengan Menerapkan Algoritma Yamamoto Recursive Code.

2. METODOLOGI PENELITIAN

2.1 Kompresi File Video

Kompresi merupakan salah satu pengubahan data yang berupa gabungan karakter yang menjadi suatu bentuk kode dengan tujuan agar mengirit kebutuhan dalam penyimpanan data, kompresi data sangat penting untuk memperkecil pada penyimpanan data, memperkecil kebutuhan handwith, dan mempercepat dalam pengiriman data[8]. Terdapat beberapa metedo dalam kompresi data dan metode-metode ini juga mempunyai ide-ide yang berbeda yang dimana sangat cocok untuk berbagai tipe data, dan juga menghasilkan output yang berbeda. Namun yang menjadi tujuan dasar tiap-tiap metode tetaplah sama yangb dimana mengkompresi data dengan menghilangkan redundancy dari data yang asli[9]. Teknik yang digunakan dalam proses kompresi data ada beberapa faktor dan variabel yang sering digunakan untuk menganalisa kualitas dari teknik kompresi data[10], yaitu:

1. *Ratio of Compression* (Rc)
   Ratio of Compression (Rc) yaitu nilai perbandingan antara ukuran bit pada data sebelum dikompresi dengan ukuran pada data bit yang telah dikompresi.
   Rumus:
   $$CR = \frac{Jumlah\ bit\ sebelum\ dikompresi}{Jumlah\ bit\ Sesudah\ dekompresi} x\ 100\% \quad (1)$$
2. *Compression Ratio* (Cg)
   Compression Ratio (Cg) yaitu persentase perbandingan antara data yang sudah dikompresi dengan data yang sudah dikompresi.
   Rumus:

Bulletin of Data Science
ISSN 2745-7966 (Media Online)
Vol 3, No 1, October 2023
Hal 149-159
https://ejurnal.seminar-id.com/index.php/bulletinds

$$CR = \frac{Jumlah\ bit\ sebelum\ dikompresi}{Jumlah\ bit\ Sesudah\ dekompresi} x\ 100\% \quad (2)$$

3. *Space Svasing* (Ss)
   Space saving yaitu selisih antara data yang belum dikompresi dengan data besar yang dikompresi.
   Rumus:
   $Ss = 100\% - Cr$ (3)
4. *Time Compression*
   Time Compression yaitu waktu yang diperlukan untuk melakukan proses kompresi dan dekompresi yang dimana semakin kecil waktu yang dihasilkan makan akan lebih mudah metode yang digunakan dalam proses kompresi dan dekompresi data.

Dalam kompresi ada dua teknik yang dilakukan[8]:

a. *Lossless compression*
   Lossless compression adalah teknik kompresi citra menghilangkan informasi sebelumnya, yang dimana hasil dari dekompresi citra yang terkompresi sama dengan citra lainnya.
b. *Lossy compression*
   Lossy compression adalah teknik kompresi citra yang dimana menghasikan beberapa informasi, dan hasil dari dekompresi citra yang sudah terkompresi tidak sama dengan citra aslinya, namun masih bisa dimaklumi oleh para penggunanya.

2.2 *Yamamoto's Recursive Code*

Yamamoto's Recursive Code adalah kode *recursif* untuk angka positif yang dimana setiap urutanyang diberikan dapat digunakan sebagai delimiter, berbeda dengan universal code lainnya yang menggunakan bit "0" sebagai delimiter seperti *Elias Omega Code*, *Even-Rodeh Code*, dan lain-lain[11]. Delimiter yang dipakai pada Algoritma *Yamamoto's Recursive Code* lebih pendek dari log*$_2$ n dalam hampir dari semua angka bulat positif dibandingkan algoritma sebelumnya[12]. Langkah–langkah untuk membangun Yamamoto's Recursive Code adalah sebagai berikut[13]:

1. Menentukan delimiter f-bit (dimana f adalah integer positif), misal f = 2, dan delimiter 00.
2. Menetukan tabel Bα,f (n). dengan mengurutkan bilangan biner yang tidak dimulai dengan delimiter, dalam hal ini kita contohkan 00.
3. Setelah didapatkan tabel Bα,f (n). selanjutnya tentukan tabel Bα,f (n) dengan mengeliminasi semua nilai Bα,f (n) yang jika digabung membentuk delimiter, dalam hal ini kita dapat menentukan Code Yamamoto dengan melakukan perulangan dengan grup Bα,f (n) hingga mencapai Bα,f (n).

Tabel 1. Ba.f (n) yamamoto codes

N	Bα,f (n)		Bα,f (n)	
	a=00	a=100	a=00	a=100
1	0	0	1	0
2	1	1	01	00
3	01	00	10	01
4	10	01	11	11
5	11	10	010	000
6	010	11	011	001
7	011	000	100	010
8	100	001	101	011
9	101	010	110	101
10	110	011	111	110
11	111	101	0100	111
12	0100	110	0101	0000
13	0101	111	0110	0001
14	0110	0000	0111	0010
15	0111	0001	1000	0011
16	1000	0010	1001	0100
17	1001	0011	1010	0101
18	1010	0100	1011	0110
19	1011	0101	1100	0111
20	1100	0110	1101	1010
21	1101	0111	1110	1011
22	1110	1010	1111	1100
23	1111	1011	01000	1101

Bulletin of Data Science
ISSN 2745-7966 (Media Online)
Vol 3, No 1, October 2023
Hal 149-159
https://ejurnal.seminar-id.com/index.php/bulletinds

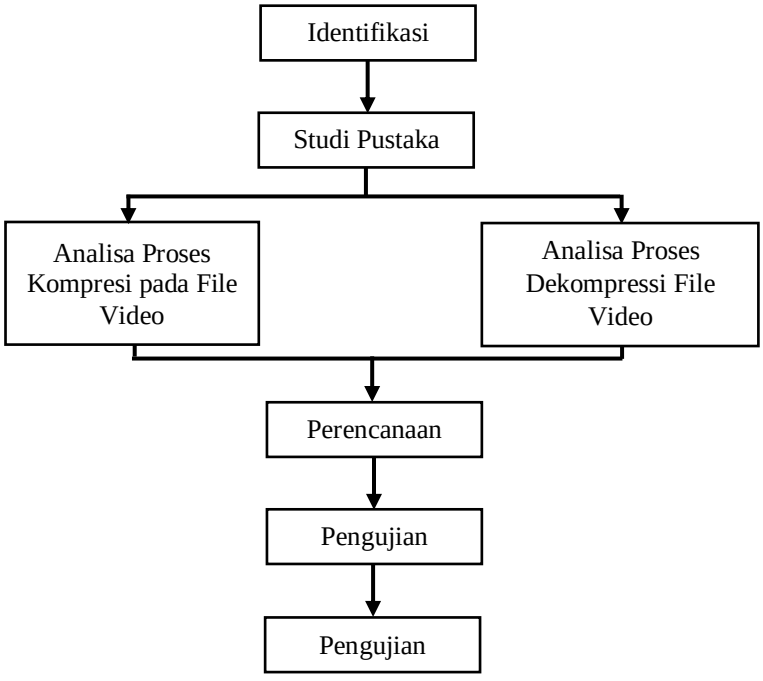
Codeword yang digunakan oleh yamamoto's recursive code dilengkapi dengan menambahkan pembatas a ke ujung kanan. Untuk melihat codeword dari algoritma yamamoto's recursive code dengan nilai a=00 dan a=100 dapat dilihat pada tabel berikut.

Tabel 2. Codeword dalam Yamamoto's Recursive Code

N	a=00	a=100
1	1 00	0 100
2	1 01 00	0 00 100
3	1 10 00	0 01 100
4	1 11 00	0 11 100
5	1 01 010 00	0 00 000 100
6	1 01 011 00	0 00 001 100
7	1 01 100 00	0 00 010 100
8	1 01 101 00	0 00 011 100
9	1 01 110 00	0 00 101 100
10	1 01 111 00	0 00 110 100
11	1 10 0100 00	0 00 111 100
12	1 10 0101 00	0 01 0000 100
13	1 10 0110 00	0 01 0001 100
14	1 10 0111 00	0 01 0010 100
15	1 10 1000 00	0 01 0011 100
16	1 10 1001 00	0 01 0100 100
17	1 10 1010 00	0 01 0101 100
18	1 10 1011 00	0 01 0110 100
19	1 10 1100 00	0 01 0111 100
20	1 10 1101 00	0 01 1010 100
21	1 10 1110 00	0 01 1011 100
22	1 10 1111 00	0 01 1100 100
23	1 10 01000 00	0 01 1101 100
24	1 10 01001 00	0 01 1110 100
25	1 10 01010 00	0 01 1111 100
26	1 10 01011 00	0 11 00000 100

2.3 Tahapan Penelitian

Pada penelitian ini akan dilakukan analisa dan perancangan perangkat lunak pengkompresian file video dengan menggunakan algoritma Yamamoto Recursive Code. Tahapan penelitian yang akan dilakukan dapat dilihat dalam kerangka kerja yang ada pada gambar 1.



Bulletin of Data Science
ISSN 2745-7966 (Media Online)
Vol 3, No 1, October 2023
Hal 149-159
https://ejurnal.seminar-id.com/index.php/bulletinds

Gambar 1. Kerangka Kerja Penelitian

3. ANALISA DAN PEMBAHASAN

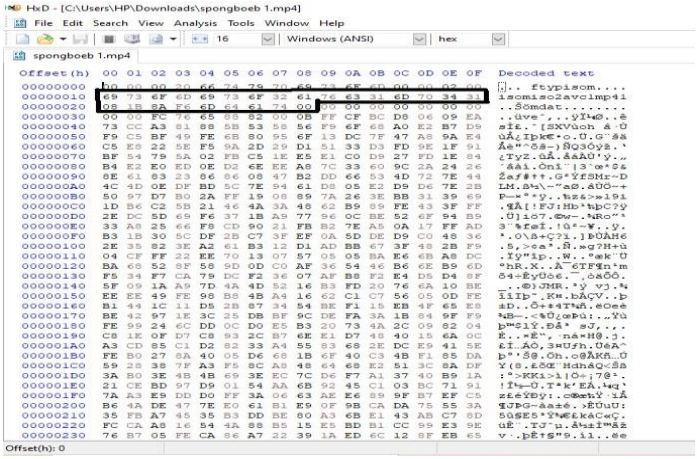
3.1 Sampel Data

Sampel data yang digunakan dalam penelitian ini adalah file video yang berekstensi MP4. Sampel data yang digunakan sebanyak 5 (lima) file video. Sampel data yang akan digunakan dapat dilihat pada tabel 1.

Tabel 1. Sampel Data

No	Nama file	Ukuran
1	Spongboeb 1.mp4	130 Mb
2	Spongboeb 2.mp4	137 Mb
3	Spongboeb 3.mp4	134 Mb
4	Spongboeb 4.mp4	123 Mb
5	Spongboeb 5.mp4	464 Mb

Dari 5 (lima) sampel data, untuk proses analisa diambil hanya 1 sampel saja, kemudian dari sampel tersebut diambil 25 nilai heksa. Untuk mengambil 25 nilai heksa, digunakan software HxD. Untuk lebih jelasnya dapat dilihat pada gambar 2 berikut.

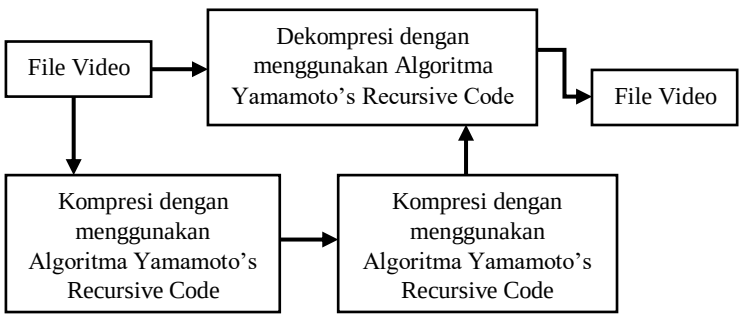


Gambar 2. Sampel Data Dari HxD

Dari gambar 3.2 diatas maka sampel nilai heksa yang diambil yaitu : "69, 73, 6F, 6D, 69, 73, 6F, 32, 61, 76, 63, 31, 6D, 70, 34, 31, 08, 1B, 8A, F6, 6D, 64, 61, 74 ,00."

3.2 Analisa dan Penerapan Metode

Analisa yang dilakukan dalam penelitian ini meliputi kompresi *file* video dengan menerapkan *file* algoritma *yamamoto's recursive code*. *File* video sangat memerlukan penyimpanan yang sangat besar, sehingga memerlukan kapasitas yang sangat besar oleh karna itu dilakukan pemngkompresan pada video tersebut. *File* video yang akan dikompres adalah *file* video yang mempunyai format MP4. Algoritma *Yamamoto's Recursive Code* merupakan teknik kompresi *lossles* dan kode *recursif* untuk angka positif yang dimana setiap urutan yang diberikan dapat digunakan sebagai *delimeter* , berbeda dengan *universal code* lainnya yang menggunakan bit "0" sebagai *delimiter* seperti *Elias omega code* , *Even-Rodeh, Stout code*, dll. *Delimiter* yang dipakai pada Algoritma *Yamamoto's Recursive Code* lebih pendek dari log^*_2 dalam hampir semua angka bulat dibandikan algoritma sebelumnya. Untuk menganalisa terhadap suatu sistem yang dilakukan sebelum tahapan perancangan dilakukan. Langkah-langkah prosedur dalam melakukan kompresi dan dekompresi *file* video. Kemudian *file* video yang sudah dikompresi tersebut didekompresi lalu kembali ke hasil *file* video. Untuk mengetahui prosedur kompresi dan dekompresi suatu *file* video.



Bulletin of Data Science
ISSN 2745-7966 (Media Online)
Vol 3, No 1, October 2023
Hal 149-159
https://ejurnal.seminar-id.com/index.php/bulletinds

Gambar 3. Prosedur Kompresi dan Dekompresi

3.2.1 Contoh Kasus

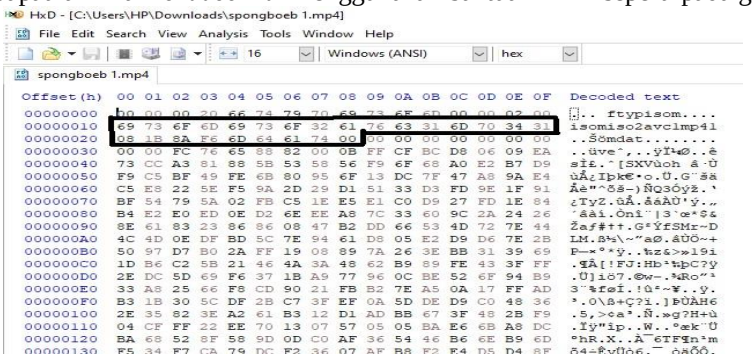
Dengan melakukan kompresi data, data yang berukuran besar akan dikompresi menjadi ukuran yang kecil sehingga akan mengurangi alokasi penyimpanan. Untuk menganalisa *file* video harus dilakukan pengembalian sampel *file* untuk dapat nilai dari data pada sebuah *file* video berupa nilai hexadecimal. Berikut langkah untuk mengkompresi dan mendekompresi *file* video.

1. Analisa proses kompresi *file* gambar dengan menggunakan *Yamamoto's Recursive Code.*
   a. Memasukkan *File* video
   File video yang digunakan adalah *file* video yang ada pada tabel 1 sampel data atau bisa dilihat pada gambar berikut ini:

Tabel 2. Informasi *File* Dokumen Sampel

Kriteria	Keterangan
Format	MP4
Nama File	Spongboeb 1.mp4
Ukuran	130 Mb

Dari sampel diatas didapatkan nilai *hexadecimal* menggunakan bantuan HXD seperti pada gambar dibawah ini:



Gambar 4. Nilai *Hexadecimal File* Dokumen *Sample*

Berdasarkan gambar diatas maka didapatlah nilai *hexadecimal file* video *sample* nilai sebanyak 25 karakter nilai *hexadecimal file* video *sample*. Nilai *hexadecimal* diambil dari sisi kiri sampai bilangan ke 25.

   b. Melakukan Pembacaan Isi *File*.
   Adapun bilangan *hexadecimal* dari *file* video tersebut adalah Nilai 69, 73, 6F, 6D, 69, 73, 6F, 32, 61, 76, 63, 31, 6D, 70, 34, 31, 08, 1B, 8A, F6, 6D, 64, 61, 74 ,00. *hexadecimal* tersebut dimasukkan kedalam tabel untuk dilakukan pembacaan frekuensi. Pembacaan frekuensi dilakukan dengan menghitung jumlah nilai yang sama di setiap nilai data yang muncul. Adapun pembacaan frekuensi dapat dilihat pada tabel di bawah ini :

Tabel 3. Nilai *File* Video

Nilai	Frekuensi
69	2
73	2
6F	2
6D	3
32	1
61	2
76	1
63	1
31	2
70	1
34	1
08	1
1B	1
8A	1
F6	1

64	1
74	1
00	1
Total	25

c. Mengurutkan karakter dari yang memiliki frekuensi terbesar (banyak nilai yang sama) ke frekuensi terkecil. Urutan nilai dapat dilihat pada tabel berikut ini:

Tabel 4. Nilai Bit *File* Video

Nilai Hexa	Biner	Bit	Frek	Bit x Frek
69	01101001	8	2	16
73	01100011	8	2	16
6F	01101111	8	2	16
6D	01101101	8	3	24
32	00110010	8	1	8
61	01100001	8	2	16
76	01110110	8	1	8
63	01100011	8	1	8
31	00110001	8	2	16
70	01110000	8	1	8
34	00110100	8	1	8
08	00001000	8	1	8
1B	00011011	8	1	8
8A	10001010	8	1	8
F6	11110110	8	1	8
64	01100100	8	1	8
74	01110100	8	1	8
00	00000000	8	1	8
	Total			200

Berdasarkan tabel diatas, rata-rata memiliki nilai *hexadecimal* (karakter) bernilai 8bit bilangan biner. Sehingga 25 bilangan *hexadecimal* mempunyai nilai biner sebanyak 200 bit. Untuk mengubah satuan menjadi *byte* maka jumlah keseluruhan bit dibagikan 8. Maka dihasilkan 200/8 = 25 *byte*. Dari perhitungan tabel setelah dikompresi dengan menggunakan *yamamoto's recursive code* adalah 175 bit. Untuk diubah menjadi satuan *byte* maka dibagi menjadi 8 yaitu 175/8 = 21,875 byte

d. Melakukan hasil *string bit yamamoto's recursive code* menjadi nilai *file*
Sebelum melakukan hasil *string bit yamamoto's recursive code* menjadi nilai *file* terlebih dahulu dilakukan pemeriksaan terhadap panjang *string bit*. Pembentukan nilai bit baru hasil kompresi dari susunan nilai *hexadecimal* sebelum dikompresi yaitu "69, 73, 6F, 6D, 69, 73, 6F, 32, 61, 76, 63, 31, 6D, 70, 34, 31, 08, 1B, 8A, F6, 6D, 64, 61, 74 ,00" seperti yang disajikan pada tabel 4.6 Dibawah ini:

Tabel 5. *String bit* Hasil Kompresi dengan Algoritma *Yamamoto's Recursive Code*

69	73	6F	6D	69	73
10100	11000	11100	100	10100	11000
6F	32	61	76	63	31
11100	10110000	10101000	10110100	10111000	10101100
6D	70	34	31	08	1B
100	10111100	110010000	10101100	1100101001100011000	
8A	F6	6D	64	61	74
110011100	11010000	100	110100100	10101000	110101000
00					
110101100					

Berdasarkan pada Tabel 6 *string bit* yang dihasilkan dalam kompresi dengan algoritma *yamamoto's recursive code* dapat dituliskan pada tabel berikut:

Tabel 6. nilai terkompresi dari codeword

10100110001110010010100110001110010110000101010001011010010111
00101011001001011110011001000010101100110010100110011000110011
00110100000100110100100101010001101010001101011 00

Tabel 7. Perhitungan Penambahan Bit

Padding	Flagging
7 – n + "1"	9 –n
7 – 7+"1" = 1	9 – 7 = 2 = 00000010 (biner)

Tabel 8. *String Bit* Yang Telah Dilakukan Penambahan

10100110 00111001 00101001 10001110 01011000 01010100
01011010 01011100 01010110 01001011 11001100 10000101
01100110 01010011 00110001 10011100 11010000 01001101
00100101 01000110 10100011 01011001 00000010

Total panjang bit keseluruhan setelah ada penambahan bit adalah 175 + 1 + 8 = 184. Selanjutnya lakukan pemisahan bit menjadi beberapa kelompok. Setiap kelompok terdiri dari 8 bit seperti gambar dibawah ini :

Tabel 9. pemisahan bit menjadi beberapa kelompok

10100110	00111001	00101001	10001110	01011000	01010100
01011010	01011100	01010110	01001011	11001100	10000101
01100110	01010011	00110001	10011100	11010000	01001101
00100101	01000110	10100011	01011001	00000010	

Berdasarkan pada pembagian kelompok nilai *biner* diatas, maka didapatkan nilai *biner* baru dan sudah terkompresi beserta nilai *biner* penambahan yang terdiri dari 22 *byte*. Setelah melakukan pembagian, makan nilai *biner* yang sudah dibagi diubah kedalam suatu karakter dengan terlebih dahulu mencari nilai *desimal* dari *string* bit tersebut dengan menggunakan kode ASCII untuk mengetahui nilai *biner* yang sudah terkompresi. Adapun nilai *biner* yang sudah terkompresi dapat dilihat pada tabel yang dibawah ini :

Tabel 10. Nilai Desimal Terkompresi

No	Biner	Nilai Desimal	Karakter
1	10100110	166	¦
2	00111001	57	9
3	00101001	41	)
4	10001110	142	A
5	01011000	88	X
6	01010100	84	T
7	01011010	90	Z
8	01011100	92	\
9	01010110	86	V
10	01001011	75	K
11	11001100	204	“
12	10000101	133	…
13	01100110	102	f
14	01010011	83	S
15	00110001	49	!
16	10011100	156	ST
17	11010000	208	D
18	01001101	77	M
19	00100101	37	%
20	01000110	70	F
21	10100011	163	£
22	01011001	89	Y
23	00000010	2	-

e. Menghitung Kinerja Kompresi
Ukuran *file* awal sebelum dikompresi adalah 200 bit, sehingga rasio kompresinya adalah :
RC = (Ukuran *File* Asli) / (Ukuran *File* Terkompresi)
RC = 200 /175 = 1,14

Jika dinyatakan dalam bentuk persen maka dapat dituliskan dalam rumus sebagai berikut:
SS = (1 – Ukuran *File* Terkompresi) / (Ukuran *File* Asli) x 100%
SS = (1 – (175 / 200)) x 100%
SS = (1 – 0,875) x 100%
SS = 0,125 x 100%
SS = 12,5%
Setelah melakukan perhitungan diatas, maka dapat disimpulkan bahwa dengan menggunakan algoritma *yamamoto's recursive code* dengan sampel data diatas maka hasil kompresinya tidak berkurang dari data sebelum dikompresi.

2. Analisa Proses dekompresi *file* dokumen dengan mengunakan *Yamamoto's Recursive Code*
Proses dekompresi dimulai dari merubah karakter hasil kompresi, menjadi nilai *biner*. Karakter hasil kompresi yang digunakan diambil dari hasil analisa proses kompresi diatas. Untuk merubah karakter hasil kompresi menjadi nilai *biner*, bisa dilihat pada tabel 11 berikut.

Tabel 11. Nilai Desimal Terkompresi

No	Karakter Hasil Kompresi	Nilai Desimal	Biner
1	¦	166	10100110
2	9	57	00111001
3	)	41	00101001
4	A	142	10001110
5	X	88	01011000
6	T	84	01010100
7	Z	90	01011010
8	\	92	01011100
9	V	86	01010110
10	K	75	01001011
11	“	204	11001100
12	…	133	10000101
13	f	102	01100110
14	S	83	01010011
15	!	49	00110001
16	ST	156	10011100
17	D	208	11010000
18	M	77	01001101
19	%	37	00100101
20	F	70	01000110
21	£	163	10100011
22	Y	89	01011001
23	-	2	00000010

Berdasarkan pada tabel yang ada diatas, maka didapatkan nilai *biner* “10100110 00111001 00101001 10001110 01011000 01010100 01011010 01011100 01010110 01001011 11001100 10000101 01100110 01010011 00110001 10011100 11010000 01001101 00100101 01000110 10100011 01011001 00000010” Langkah berikutnya yaitu dengan menghilangkan padding dan Flagging. Menghilangkan padding dan flagging dilakukan dengan cara mengambil 8 (delapan) bit terakhir lalu merubahnya ke desimal dan nyatakan dengan n, setelah itu hilangkan bit dari akhir sebanyak jumlah bit dari hasil rumus 7+n. Delapan bit terakhir “00000010” yang dirubah ke desimal menjadi 2 lalu dinyatakan dengan n. Hilangkan bit akhir sebanyak 7+n = 7+2 = 9, sehingga string bit menjadi “10100110 00111001 00101001 10001110 01011000 01010100 01011010 01011100 01010110 01001011 11001100 10000101 01100110 01010011 00110001 10011100 11010000 01001101 00100101 01000110 10100011 0101100”. Langkah terakhir yaitu melakukan pengecekan nilai bit sesuai dengan codeword yang digunakan. Dekompresi yang dilakukan dengan algoritma *yamamoto's recursive code* yang dilakukan dengan mengubah hasil seluruh *heksa* menjadi *string* bit semula yang terdapat pada *file* awal , seperti pada tabel 12 di bawah ini :

Tabel 12. Hasil Dekompresi

No	Yamamoto's Recursive Code	Nilai Heksa
1	10100	69
2	11000	73
3	11100	6F

No	Yamamoto's Recursive Code	Nilai Heksa
4	100	6D
5	10100	69
6	11000	73
7	11100	6F
8	10110000	32
9	10101000	61
10	10110100	76
11	10111000	63
12	10101100	31
13	100	6D
14	10111100	70
15	110010000	34
16	10101100	31

Setelah dilakukan pengecekan bit, maka didapatkan hasil nilai heksa "69, 73, 6F, 6D, 69, 73, 6F, 32, 61, 76, 63, 31, 6D, 70, 34, 31, 08, 1B, 8A, F6, 6D, 64, 61, 74 ,00". Dari hasil tersebut, maka dianggap telah berhasil didekompresi, karna nilai heksa yang dihasilkan sama seperti nilai heksa sebelum dikompresi.

4. KESIMPULAN

Berdasarkan dari penelitian-penelitian yang telah dilakukan , maka hasil akhir dari penelitian tersebut dapat diambil beberapa kesimpulan dari pembahasan sebelumnya. Adapun kesimpulan tersebut yaitu prosedur dalam mengkompresi file video dengan menggunakan algoritma yamamoto's recursive code telah berhasil dilakukan dengan file video yang berformat *.mp4 dan dalam proses pengkompresian berjalan dengan sesuai dengan teknik kompresi. Dalam penerapan algoritma Yamamoto's Recursive Code telah dilakukan dan terbukti bahwa suatu file video yang memiliki ukuran lebih besar dapat dikompres menjadi ukuran yang lebih kecil. Aplikasi kompresi file video telah berhasil dirancang dan dibangun dengan menggunakan aplikasi Microsoft Visual Studio 2008 dengan menerapkan algoritma Yamamoto's Recursive Code. Penelitian ini membuktikan bahwa penggunaan algoritma Yamamoto's Recursive Code dalam mengkompresi file video berformat MP4 memberikan hasil yang efektif dan efisien. Proses kompresi yang dilakukan menunjukkan penurunan ukuran file tanpa mengurangi kualitas video secara signifikan. Aplikasi yang dikembangkan juga menunjukkan kinerja yang baik dalam mengimplementasikan algoritma tersebut. Hasil ini memberikan kontribusi penting dalam bidang kompresi video, khususnya untuk aplikasi yang membutuhkan efisiensi penyimpanan dan transmisi data video. Dengan demikian, algoritma Yamamoto's Recursive Code dapat dijadikan sebagai solusi alternatif dalam kompresi file video yang berukuran besar, memberikan manfaat praktis bagi pengguna yang memerlukan pengurangan ukuran file tanpa kehilangan kualitas video yang berarti.

REFERENCES

[1] A. Setiawan, "Kompresi Video Menggunakan Metode Hybrid DWT dan DCT untuk Pengurangan Ukuran File," Jurnal Teknologi Informasi dan Komputer, vol. 4, no. 2, pp. 123-130, Apr. 2016.
[2] S. Andriani, "Penerapan Algoritma Kompresi Lossy pada Citra Digital," Jurnal Teknologi Informasi dan Komunikasi, vol. 7, no. 1, pp. 23-30, 2017
[3] MusdalifaThamrin, "jurnal penelitian kompresi," vol. 148, hal. 148–162.
[4] H. T. Sihotang, "Perancangan Dan Implementasi Algoritma Arithmetic Coding Untuk Aplikasi Kompresi Data Video Dan Audio," *J. Mantik Penusa*, vol. 2, no. 1, hal. 58–64, 2018.
[5] I. G. Rasagama, "Pengembangan Model Pembelajaran Getaran Berbasis Video Youtube Untuk Meningkatkan Pemahaman Konsep Mahasiswa Politeknik," *J. Pendidik. Sains*, vol. 8, no. 2, hal. 91, 2020, doi: 10.26714/jps.8.2.2020.91-101.
[6] M. S. Wahyuni, "Analisa Membandingkan Hasil Kompresi File Mpeg-4 Dan Flv Menggunakan Algoritma Huffman Dan Lz77," *Saintek ITM*, vol. 32, no. 1, hal. 40–47, 2019, doi: 10.37369/si.v32i1.52.
[7] . S., T. B. Adji, dan H. A. Nugroho, "Pengaruh Load Balancing Pada Pemrosesan Paralel untuk Kompresi Video," *Conf. Senat. STT Adisutjipto Yogyakarta*, vol. 2, hal. 121, 2016, doi: 10.28989/senatik.v2i0.85.
[8] A. Triantoro, H. Sajati, dan A. Pujiastuti, "the Audio Video of Web-Based Compression With Ffmpeg," *Compiler*, vol. 7, no. 2, hal. 132, 2018, doi: 10.28989/compiler.v7i2.280.
[9] A. Pratama, "Analisis Teknik Kompresi Video pada Media Penyimpanan dan Pengiriman Data", Jurnal Teknologi Informasi dan Komunikasi, vol. 8, no. 2, pp. 100-110, 2016.
[10] D. G. Irianto, "Analisis Perbandingan Format Video MP4 dan FLV Berdasarkan Kualitas dan Ukuran File," Jurnal Teknologi Informasi dan Ilmu Komputer, vol. 4, no. 2, pp. 120-126, Mei 2016.
[11] Arianti Rhamadani, "Analisa Perbandingan Algoritma Taboo Codes dan Algoritma Yamamoto's Recursive Code untuk Kompresi File Teks Menggunakan Metode Exponential," *KOMIK (Konferensi Nasional Teknologi Informasi dan Komputer)*, vol. 6, no. 1, pp. 123-130, 2018. DOI: 10.30865/komik.v6i1.5728.
[12] M. Y. Alfian, "Implementasi Kode Recursif Yamamoto pada Pengkodean Data Positif," *Jurnal Teknologi Informasi dan Komunikasi*, vol. 7, no. 2, pp. 123-130, 2016.
[13] D. Salomon dan G. Motta, *Handbook of Data Compression 5th*, vol. 53, no. 9. 2019.

[14] H. Herdianto, "Perbandingan Metode RLE dan Huffman dalam Kompresi Data Teks," Jurnal Ilmiah Core IT : Community Research Information Technology, vol. 9, no. 3, pp. 1-10, 2017.
[15] M. L. Imani, R. R. Muhima, dan S. Agustini, "Penerapan Metode Huffman dalam Kompresi Data," Prosiding Seminar Nasional Sains dan Teknologi Terapan, 2016.
[16] Ilham Akhsanu Ridlo, "Pedoman Pembuatan Flowchart," *Academia.Edu*, hal. 14, 2017, [Daring]. Tersedia pada: https://www.academia.edu/34767055/Pedoman_Pembuatan_Flowchart.
[17] A. S. Sembiring, S. R. Siregar, "Analisa Perbandingan Algoritma Yamamoto's Recursive Code dan Algoritma Fibonacci Code dalam Mengkompresi File Pptx," KOMIK (Konferensi Nasional Teknologi Informasi dan Komputer), vol. 6, no. 1, pp. 1-10, 2018.
[18] S. B. Ginting, "Perbandingan Algoritma Yamamoto's Recursive Code dan Additive Code dalam Kompresi File Video," KOMIK (Konferensi Nasional Teknologi Informasi dan Komputer), vol. 5, no. 1, pp. 1-7, 2018.
[19] N. Aftikasyah, "Penerapan Algoritma Yamamoto's Recursive Code untuk Mengkompresi File Dokumen," KOMIK (Konferensi Nasional Teknologi Informasi dan Komputer), vol. 5, no. 1, pp. 1-8, 2018.
[20] T. A. Kurniawan, "Pemodelan Use Case (UML): Evaluasi Terhadap beberapa Kesalahan dalam Praktik," *J. Teknol. Inf. dan Ilmu Komput.*, vol. 5, no. 1, hal. 77, 2018, doi: 10.25126/jtiik.201851610.