

Penerapan Algoritma J-Bit Encoding Untuk Kompresi File Video Pada Aplikasi Rekam Layar

Juita Hutagaol

Fakultas Ilmu Komputer dan Teknologi Informasi, Program Studi Teknik Informatika, Universitas Budi Darma, Medan, Indonesia

Email: witahutagaol19@gmail.com

Abstrak—Perekam layar adalah suatu aplikasi yang berguna untuk merekam aktivitas pada layar monitor saat kita memerlukannya untuk kebutuhan tertentu. Ketika pengguna melakukan aktivitas perekaman layar, dan menyimpan *file* video hasil rekam layar tersebut, maka akan menghasilkan ukuran *file* yang besar. Karena semakin panjang durasi perekamannya maka akan semakin besar pula ukuran *file* tersebut. Maka ukuran *file* yang besar itu akan menimbulkan masalah karena dibatasi dengan kapasitas penyimpanan. Untuk mengatasi masalah tersebut maka didapatkan solusi dengan melakukan kompresi. Algoritma *J-Bit Encoding* merupakan teknik kompresi *lossless*, yang tidak menghilangkan informasi sebelumnya, dimana hasil dekompresi dari video yang terkompresi sama dengan *file* video aslinya. Dengan menggunakan algoritma *J-Bit Encoding* menghasilkan kinerja *Compression Ratio* sebesar 36%. Hasil tersebut membuktikan bahwa kompresi *file* video hasil rekam layar dengan menerapkan algoritma *J-Bit Encoding* dapat memperkecil ukuran *file* video yang besar menjadi lebih kecil dari ukuran aslinya.

Kata Kunci: Kompresi; File Video; Rekam Layar; J-Bit Encoding.

Abstract—Screen recorder is an application that allows us to record activities on the monitor screen when we need it for certain needs. When the user performs screen recording activities, and saves the video file recorded on the screen, it will produce a large file size. Because the longer the recording duration, the larger the file size will be. Then the large file size will cause problems because it is limited by storage capacity. To solve this problem, a solution is obtained by compression. The J-Bit Encoding algorithm is a lossless compression technique, which does not remove previous information, where the decompression results of the compressed video are the same as the original video files. Using the J-Bit Encoding algorithm, the compression ratio is 36%. These results prove that the compression of the video files recorded on the screen by applying the J-Bit Encoding algorithm can reduce the size of large video files to be smaller than the original size.

Keywords: Compression, Video Files, Screen Record, J-Bit Encoding.

1. PENDAHULUAN

Perkembangan teknologi yang begitu cepat seperti sekarang ini sangat bermanfaat guna bertukar informasi secara global baik itu dalam bentuk teks, gambar, audio ataupun video. Sehingga secara tidak langsung menimbulkan masalah yaitu membuat kebutuhan akan penyimpanan semakin meningkat. Dan bila mengirimkan data melalui media transmisi, maka akan memerlukan waktu yang semakin lama untuk proses pengiriman data tersebut. Untuk itu diperlukan suatu teknik yang dapat mereduksi besarnya ukuran suatu *file* yang disebut kompresi data.

Perekam layar merupakan suatu aplikasi yang berguna untuk merekam aktivitas pada layar monitor saat kita memerlukannya untuk kebutuhan tertentu. Biasanya pengguna melakukan aktivitas perekaman layar itu pada PC. Aplikasi yang digunakan dalam proses rekam layar adalah aplikasi camtasia. Misalnya untuk membuat video panduan atau video tutorial. Ada pula para *gamers* yang merekam hasil permainan mereka untuk dibuat dalam video blog, merekam proses *editing*, dan masih banyak lagi. Merekam layar sebenarnya bisa dilakukan secara manual dengan kamera. Akan tetapi, merekam dengan menggunakan kamera hasilnya tidak terlalu jelas. Oleh karena itu digunakan perekam layar. Aplikasi rekam layar yang digunakan untuk merekam layar video adalah camtasia.

Ketika pengguna melakukan aktivitas perekaman layar, dan menyimpan *file* video hasil rekam layar tersebut, maka akan menghasilkan ukuran *file* yang besar. Karena pada umumnya pengguna melakukan aktivitas perekaman layar dengan durasi yang cukup panjang. Semakin panjang durasi perekamannya maka akan semakin besar pula ukuran *Video* tersebut. Maka ukuran *file* yang besar itu akan menimbulkan masalah karena dibatasi dengan kapasitas penyimpanan. Dan ketika kita ingin mengirimkannya maka kita akan memerlukan waktu yang lama. Maka dari itu, kompresi data menjadi sangat penting untuk mengatasi keterbatasan tersebut.

Kompresi data merupakan proses yang dapat mengubah sebuah aliran data masukan (sumber atau data asli) kedalam aliran data yang lain (keluaran atau data yang dimampatkan) yang memiliki ukuran lebih kecil [1]. Kompresi data menjadi sangat penting karena memperkecil kebutuhan penyimpanan data, dan mempercepat pengiriman data.

Ada beberapa algoritma untuk mengkompresi data yang didasarkan pada beberapa ide yang cocok untuk berbagai jenis data dan menghasilkan hasil yang berbeda pula, tetapi semuanya didasari pada prinsip yang sama yaitu mengompres data dengan menghilangkan redundansi dari sumber data *file*. Saat ini algoritma yang digunakan untuk mengompresi *file* video salah satunya ialah algoritma J-Bit Encoding (JBE), algoritma ini merupakan salah satu teknik *lossy compression*. Sebelumnya pada tahun 2017, hasil penelitian yang dilakukan oleh Tedi Mizwar telah berhasil membuktikan bahwa algoritma J-Bit Encoding mampu mengompresi *File* teks dengan cukup baik, kompresi yang dilakukan dari ukuran asli 1231 byte menjadi 940 byte dengan *ratio* kompresi 24% [2].

Pada penelitian sebelumnya mengenai kompresi *file* video yang berjudul “Kompresi *File* Video Mp4 Dengan Menggunakan Metode Discrete Cosine Transform” oleh Rizky Syahputra pada tahun 2016 dengan dengan ISSN: 2407-389X. Pada penelitian tersebut memiliki kekurangan yaitu metode discrete cosine transform tidak tahan terhadap

perubahan suatu objek dikarenakan pesan mudah dihapus karena lokasi penyisipan data dan pembuatan data dengan metode DCT diketahui[3].

Ada pula penelitian yang berjudul “Perancangan Dan Implementasi Algoritma *Arithmetic Coding* Untuk Aplikasi Kompresi Data Video Dan Audio”, oleh Hengki Tomando Sihotang tahun 2018 dengan ISSN 2580-9741. Pada penelitian tersebut menyatakan bahwa ukuran *file* sebelum kompresi sama dengan ukuran *file* sesudah didekompresi yang berarti bahwa tidak ada data yang hilang selama proses kompresi dan waktu yang diperlukan untuk kompresi *file* MP4 lebih besar dibandingkan dengan waktu kompresi *file* MP3. Dan algoritma *Arithmetic Coding* tidak optimal dalam melakukan kompresi audio dan video dapat dilihat pada rasio kompresi yang rendah dan grafik yang tidak stabil[4].

2. METODOLOGI PENELITIAN

2.1 Kompresi

Kompresi merupakan proses untuk menghilangkan berbagai kerumitan yang tidak penting (redundansi) dari suatu informasi dengan cara memadatkan isi file sehingga ukurannya menjadi lebih kecil dengan memaksimalkan kesederhanaannya dan tetap menjaga kualitas penggambaran dari informasi tersebut[6]. Kompresi data menjadi sangat penting karena memperkecil kebutuhan penyimpanan data, dan mempercepat pengiriman data. Kompresi data mengacu pada proses mengurangi jumlah data yang memerlukan untuk menghadirkan kuantitas informasi ditentukan. Data kompresi mengacu pada proses mengurangi jumlah data memerlukan untuk menghadirkan kuantitas informasi yang ditentukan. Sebuah data yang dikompres tentunya harus dapat di kembalikan lagi kebentuk aslinya, prinsip ini dinamakan dekompresi.

2.2 Algoritma *J-Bit Encoding*

J-Bit Encoding (JBE) bekerja dengan memanipulasi bit data untuk mengurangi ukuran tanpa kehilangan data apapun setelah proses decoding. Ide utama dari algoritma ini adalah untuk membagi data masukan menjadi dua data dimana data pertama akan berisi *byte* nol asli dan data kedua akan berisi nilai bit posisi nol dan nol *byte*[2]. Langkah-langkah dari proses kompresi dapat menjelaskan sebagai berikut:

1. Membaca masukan per *byte* dari *file*.
2. Tentukan *byte* dibaca sebagai *nonzero byte* atau *zero byte*.
3. Menulis *nonzero byte* menjadi data I dan menulis bit '1' menjadi data *temporary byte*, atau hanya menulis bit '0' menjadi data *temporary byte* untuk nilai *zero byte*.
4. Isi data *temporary byte* dengan 8 bit data.
5. Jika *temporary byte* data diisi dengan 8 bit kemudian menulis nilai *temporary byte* data menjadi data II.
6. Bersihkan *temporary byte* data.
7. Ulangi langkah 1-6 sampai akhir *file* mencapai.
8. Menulis data *output* gabungan
 - a. Menulis panjang input asli.
 - b. Menulis data I.
 - c. Menulis data II.

Adapun langkah-langkah dari proses dekompresi adalah sebagai berikut:

1. Baca panjang input asli.
2. Membaca data II per bit.
3. Tentukan apakah membaca bit '0' atau '1'.
5. Menulis ke *output*, jika membaca sedikit adalah '1' kemudian membaca dan menulis data I ke *output*, jika membaca sedikit adalah '0' kemudian menulis *nol byte* untuk *output*.
6. Ulangi langkah 2-5 sampai panjang input asli mencapai.

3. HASIL DAN PEMBAHASAN

3.1 Analisa Masalah

Analisa yang dilakukan dalam penelitian ini yaitu mengkompresi *file* video pada aplikasi rekam layar dengan menerapkan algoritma *j-bit encoding*. Rekam layar adalah suatu aplikasi untuk merekam aktivitas *user* pada layar komputer. *File* video yang akan di kompresi adalah *file* video hasil rekam layar yang mempunyai format MP4.

Teknik yang digunakan algoritma *J-Bit encoding* ini ialah teknik *Lossless Compression* yang memperkecil suatu data berdasarkan dengan karakter pada objek yang akan dilakukan kompresi. Untuk melakukan proses kompresi dengan menggunakan metode ini terlebih dahulu diambil nilai sampelnya atau nilai hexadecimal yang dibutuhkan. Setelah dilakukan pembacaan terhadap nilai tersebut, kemudian nilai ini disimpan secara sementara pada variabel yang telah ditentukan.

Tahapan analisa terhadap suatu sistem dilakukan sebelum tahapan perancangan dilakukan. Langkah-langkah prosedur dalam melakukan kompresi dan dekompresi *file* video pada aplikasi layar yaitu melakukan proses rekam layar.

Lalu video hasil rekam layar tersebut dikompresi. Kemudian video hasil rekam layar yang sudah terkompresi tersebut didekompresi lalu kembali ke video hasil rekam layar.

3.1.1 Penerapan Algoritma J-Bit Encoding

Dengan melakukan kompresi data, data yang berukuran besar akan dikompresi menjadi ukuran yang kecil dan akan mengurangi alokasi penyimpanan. Dalam menganalisa *file* video harus dilakukan pengambilan *sample file* untuk mendapatkan nilai dari data pada sebuah *file* video yang berupa nilai *hexadecimal*. Berikut adalah langkah untuk mengkompresi dan mendekompresi *file* video.

1. Analisa proses kompresi *file* video dengan menggunakan *j-bit encoding*

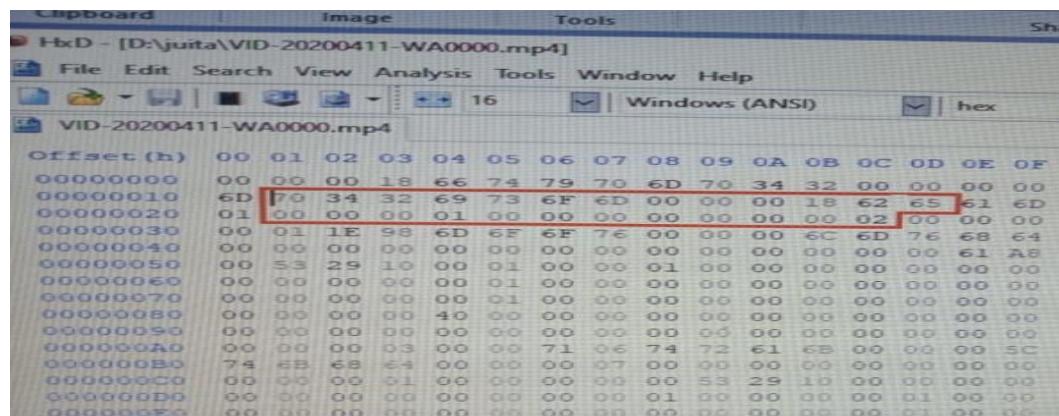
a. Memasukkan *File*

File video yang digunakan adalah *file* video yang dihasilkan oleh aplikasi rekam layar filmora. Berikut adalah informasi objek *file* video yang akan diambil sampelnya sebelum dilakukan kompresi:

Tabel 1. Informasi *File* Video *Sample*

Keterangan	
Format	.MP4
Nama <i>File</i>	VID-20200411
Ukuran	8.42 MB
Durasi	3.38 Menit

Dari sample diatas didapatkan nilai *hexadecimal* menggunakan bantuan Hexcadesimal(HxD) seperti pada gambar dibawah ini :



Gambar 1. Nilai Hexcadesimal *sample grayscale*

Berdasarkan pada gambar diatas maka didapatkan nilai *hexadecimal file* video sample. Untuk keperluan hitungan manual, maka hanya akan diambil *sample* nilai sebanyak 25 byte pixel dua baris dari sisi kiri ke sisi kanan. Dalam penerapan algoritma *J-Bit Encoding* terdapat dua proses penyelesaian yaitu proses kompresi dan proses dekompresi. Berdasarkan pada gambar 1. di atas maka didapatkan nilai *pixel* karakter sebagai berikut :70, 34, 32, 69, 73, 6F, 6D, 00, 00, 00, 18, 62, 65, 00, 00, 00, 01, 00, 00, 00, 00, 00, 02. Selanjutnya nilai byte *pixel* yang terdapat pada *file* video tersebut dimasukkan kedalam sebuah tabel dan diubah kenilai biner. Tabel biner nilai *pixel* tersebut dapat dilihat pada Gambar 1.

Tabel 2. Biner Niai *Pixel*

No.	Hexcadesimal	Biner
1	70	1110000
2	34	110100
3	32	110010
4	69	1101001
5	73	1110011
6	6F	1101111
7	6D	1101101
8	0	0
9	0	0
10	0	0
11	18	11000
12	62	1100010
13	65	1100101
14	0	0

No.	Hexcadesimal	Biner
15	0	0
16	0	0
17	1	1
18	0	0
19	0	0
20	0	0
21	0	0
22	0	0
23	0	0
24	0	0
25	2	10
		200 Bit

Berdasarkan tabel diatas, satu nilai *decimal* bernilai 8 bit bilangan biner. Sehingga 25 bilangan *decimal* mempunyai nilai biner sebanyak 200 bit. Untuk mengubah satuan menjadi *byte* maka jumlah keseluruhan bit dibagi 8, maka dihasilkan $200/8 = 25$ *byte*. Proses berikutnya adalah menentukan *byte* dibaca sebagai *non zero byte* atau *zero byte* dari nilai *Hexcadesimalnya*. Untuk proses penentuannya dapat dilakukan dengan cara membaca bit pertama terlebih dahulu secara berurutan dimulai dari atas hingga kebawah (vertikal) kemudian menuliskan ulang bit yang telah dibaca tersebut pada Data I. Untuk proses mencari nilai proses DataI dapat dilihat pada Tabel 3.

Tabel 3. Proses Data I

Biner	Data I
01110000	
00110100	
00110010	
01101001	
01110011	
01101111	
01101101	
00000000	
00000000	
00000000	000000000000000000000000
00011000	100111100001100000000000
01100010	111111100001100000000000
01100101	111010000010000000000000
00000000	000101100010000000000000
00000000	010001100000100000000000
00000000	001011000001000000000001
00000001	000111100000100010000000
00000000	
00000000	
00000000	
00000000	
00000000	
00000000	
00000000	
00000000	
00000010	

Langkah berikutnya adalah menentukan apakah nilai bit “0” atau bit “1” yang dibaca untuk dimasukkan kedalam *temporary byte data* dan pada kasus ini penulis memilih bit “1”, maka proses berikutnya adalah menulis semua bit 1 berdasarkan per *byte* kemudian dimasukkan kedalam *non zero byte(temporary byte data)* dan hasilnya sebagai berikut :
Data I:

00000000 00000000 00000000 01001111 00001100 00000000
00111111 10000110 00000000 00011101 00000100 00000000
00000001 01100010 00000000 00000010 00110000 01000000
00000000 10110000 01000000 00000010 00111100 00010001
00000000 = **200 Bit**

Temporary ByteData :

0100111100001100 0011111110000110 00011101 00000100
00000001 01100010 00000010 00110000 01000000 10110000
01000000 00000010 00111100 00010001 = **128 Bit**

Total panjang bit keseluruhan setelah ditentukan *Temporary Byte Data* adalah 144 Bit. Kemudian *Temporary Byte Data* dibagi kedalam beberapa kelompok yg terdiri dari 8bit untuk dimasukkan dan diproses kedalam Data II, dan hasilnya dapat dilihat pada Tabel 4.

Tabel 4. Pembagian Kelompok BitData II

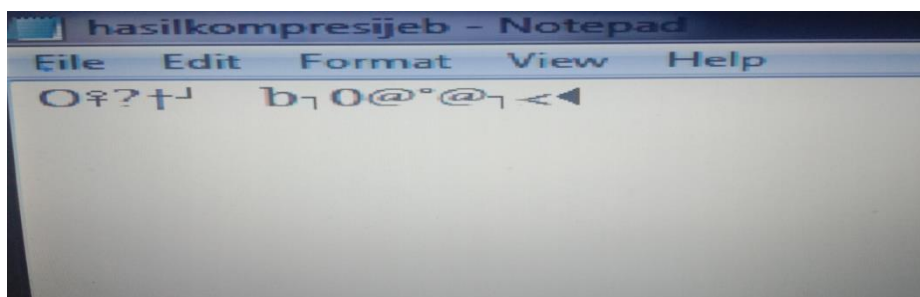
<i>Temporary Byte Data</i>	<i>Data II</i>
	01001111
	00001100
	00111111
	10000110
	00011101
010011110000110000111111	00000100
0000110000111010000010000	00000001
0000010110001000000010001	01100010
1000001000000101100000100	00000010
0000000000100011110000010	00110000
001	01000000
	10110000
	01000000
	00000010
	00111100
	00010001
	128 BIT

Berdasarkan pada pembagian kelompok nilai biner, didapatkan 16 kelompok nilai biner baru atau sama dengan 16 *byte* hasil yang sudah terkompresi. Setelah pembagian dilakukan, maka biner yang sudah dibagi dirubah kedalam suatu karakter dengan terlebih dahulu mencari nilai desimal dari *string bit* tersebut menggunakan kode ASCII untuk mengetahui nilai dari *pixel* yang sudah terkompresi. Adapun nilai *pixel* yang sudah terkompresi dapat dilihat pada Tabel 5. berikut.

Tabel 5. Nilai Desimal *Pixel* Terkompresi

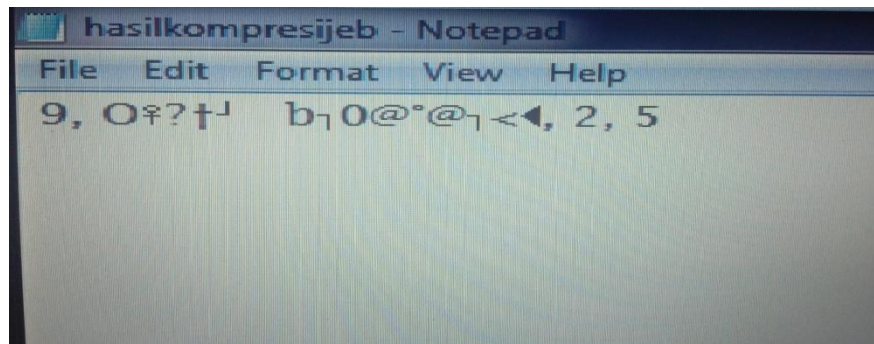
Urutan Pixel Terkompresi	Biner	Decimal
1	01001111	79
2	00001100	12
3	00111111	63
4	10000110	134
5	00011101	29
6	00000100	4
7	00000001	1
8	01100010	98
9	00000010	2
10	00110000	48
11	01000000	64
12	10110000	176
13	01000000	64
14	00000010	2
15	00111100	60
16	00010001	17

Setelah nilai desimal diketahui, maka mengubah nilai desimal kedalam suatu karakter. Karakter hasil dari proses kompresi yang dihasilkan tersimpan dalam suatu *file* dengan ekstensi “.jbe”, dan jika *file* tersebut dibuka dengan aplikasi notepad, maka akan tampil karakter seperti pada Gambar 3.3



Gambar 2. Hasil Karakter Kompresi

Agar proses dekompresi selanjutnya dapat diketahui bahwa telah terjadi penghilangan *zero byte*, maka akan ditambahkan *string* “9,” (tanpa tanda kutip) diawal karakter dan untuk mengetahui bahwa pada dimensi dari video awal adalah 25 *character*, maka akan ditambahkan *string* “,2,5” pada akhir karakter hasil kompresi. Karakter “9” menandakan jumlah *byte* pada *zero byte* yang telah dihilangkan. Karakter koma (,) pertama digunakan untuk memisahkan *zero byte* dengan karakter hasil kompresi. Karakter koma (,) yang kedua digunakan untuk memisahkan karakter hasil kompresi dengan Karakter “2” yang pertama digunakan untuk mengetahui nilai yang terdapat pada ukuran awal. Karakter koma (,) yang ketiga digunakan untuk memisahkan nilai yang terdapat pada pada ukuran awal. Karakter “2” yang kedua digunakan untuk mengetahui nilai dari karakter awal yang terdapat pada nilai *sample*. Sehingga didapatkan hasil kompresi video setelah penambahan *string bit* “9,” dan ukuran awal “,2,5” sebagai berikut :



Gambar 4. Hasil Kompresi Penambahan *StringZero Byte* Dan ukuran Awal

Berdasarkan hasil kompresi dengan *J-Bit Encoding* dapat dihitung *Ratio of Compression*(Rc) yaitu :

$$Rc = \frac{\text{Ukuran File Asli}}{\text{Ukuran File Terkompresi}}$$

$$= \frac{200}{128} = 1,5625$$

Jika dinyatakan dalam bentuk persentase yaitu *Space Savings* (Ss). maka dituliskan dalam rumus berikut:

$$Ss = \frac{(1 - \text{Ukuran File Terkompresi})}{\text{Ukuran File Asli}} \times 100\%$$

$$= \frac{(1 - (128/200))}{1} \times 100\%$$

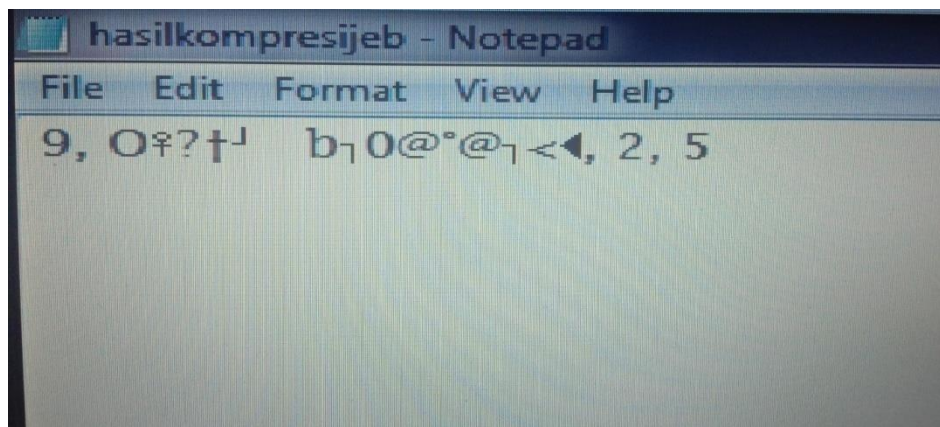
$$= (1 - 0,64) \times 100\%$$

$$= 0,36 \times 100\%$$

$$= 36\%$$

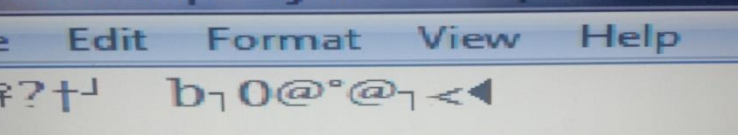
2. Analisa proses dekompresi *file* video dengan menggunakan *J-Bit Encoding*.

Berdasarkan hasil kompresi dengan penambahan *stringzero byte* dan karakter awal didapati seperti pada Gambar 5.



Gambar 5. Hasil Kompresi video

Pada proses dekompresi hal yang pertama dilakukan mencari informasi *zero byte* dengan cara mengambil karakter yang terdapat pada *file* hasil kompresi dimulai dari karakter yang diawal sampai ketemu karakter koma (,). Karakter “9” menandakan jumlah *byte* pada *zero byte* yang telah dihilangkan. Karakter koma (,) pertama digunakan untuk memisahkan *zero byte* dengan karakter hasil kompresi. Karena sudah terdapat karakter koma (,) maka informasi mengenai *zero byte* sudah diketahui yaitu 9 *byte*. Setelah dikenali *zero byte* dari proses pengenalan nilai biner di atas maka nilai karakter biner *zero byte* tersebut dihilangkan. sedangkan ukuran awal dengan cara mengambil karakter yang terdapat pada *file* hasil kompresi dimulai dari karakter yang diakhir sampai ketemu karakter koma (,) sebanyak 2 (dua) kali. Karakter pertama yang terdapat diakhir isi *file* hasil kompresi yaitu “2” menandakan itu adalah karakter yang terdapat pada ukuran awal. Karakter kedua yang terdapat diakhir isi *file* hasil kompresi yaitu koma (,) yang pertama dimana karakter tersebut merupakan karakter pemisah antara ukuran awal dengan ukuran akhir. Karakter ketiga yang terdapat diakhir isi *file* hasil kompresi yaitu “5” yang menandakan itu adalah ukuran akhirnya. Dan Karakter yang keempat adalah karakter koma (,) yang kedua dimana karakter tersebut merupakan pemisah antara *string* hasil kompresi. Karena sudah terdapat 2 (dua)



hasilkompresijeb - Notepad

File Edit Format View Help

O?+ b 0@°@ <<

No.	Char	Decimal	Biner
1	O	79	01001111
2		12	00001100
3	?	63	00111111
4	†	134	10000110
5		29	00011101
6	7	4	00000100
7		1	00000001
8	b	98	01100010
9		2	00000010
10	0	48	00110000
11	@	64	01000000
12	°	176	10110000
13	@	64	01000000
14		2	00000010
15	<	60	00111100
16		17	00010001

Data I	Biner
00000000000000000000000000000000	01110000
10011110000110000000000000000000	00110100
11111110000110000000000000000000	00110010

Data I	Biner
11101000001000000000000000	01101001
00010110001000000000000000	01110011
01000110000010000000000000	01101111
00101100000100000000000001	01101101
00011110000010001000000000	00000000
	00000000
	00000000
	00011000
	01100010
	01100101
	00000000
	00000000
	00000000
	00000001
	00000000
	00000000
	00000000
	00000000
	00000000
	00000000
	00000010

Berdasarkan pada tabel di atas maka diambil seluruh nilai biner dan digabungkan menjadi:

“00000000 00000000 00000000 01001111 00001100 00000000
00111111 10000110 00000000 00011101 00000100 00000000
00000001 01100010 00000000 00000010 00110000 01000000
00000000 10110000 01000000 00000010 00111100 00010001
00000000”

Berdasarkan perhitungan dengan algoritma *J-Bit Encodin, string bit* pada diatas berjumlah 200 bit seperti diawal sehingga dilakukan pembacaan *string bit* awal. Adapun hasil perhitungan *string bit* (pengecekan bit) adalah sebagai berikut:

Tabel 8. Pengecekan Bit

Indeks	Nilai	Keterangan
1	0	Tidak Ada
2	01	Tidak Ada
3	011	Tidak Ada
4	0111	Tidak Ada
5	01110	Tidak Ada
6	011100	Tidak Ada
7	0111000	Tidak Ada
8	01110000	Ada Pada Tabel
9	0	Tidak Ada
10	00	Tidak Ada
11	001	Tidak Ada
12	0011	Tidak Ada
13	00110	Tidak Ada
14	001101	Tidak Ada
15	0011010	Tidak Ada
16	00110100	Ada Pada Tabel
17	0	Tidak Ada
18	00	Tidak Ada
19	001	Tidak Ada
20	0011	Tidak Ada
21	00110	Tidak Ada
22	001100	Tidak Ada
23	0011001	Tidak Ada
24	00110010	Ada Pada Tabel
25	0	Tidak Ada
26	01	Tidak Ada
27	011	Tidak Ada

Indeks	Nilai	Keterangan
28	0110	Tidak Ada
29	01101	Tidak Ada
30	011010	Tidak Ada
31	0110100	Tidak Ada
32	01101001	Ada Pada Tabel
33	0	Tidak Ada
34	01	Tidak Ada
35	011	Tidak Ada
36	0111	Tidak Ada
37	01110	Tidak Ada
38	011100	Tidak Ada
39	0111001	Tidak Ada
40	01110011	Ada Pada Tabel
41	0	Tidak Ada
42	01	Tidak Ada
43	011	Tidak Ada
44	0110	Tidak Ada
45	01101	Tidak Ada
46	011011	Tidak Ada
47	0110111	Tidak Ada
48	01101111	Ada Pada Tabel
49	0	Tidak Ada
50	01	Tidak Ada
51	011	Tidak Ada
52	0110	Tidak Ada
53	01101	Tidak Ada
54	011011	Tidak Ada
55	0110110	Tidak Ada
56	01101101	Ada Pada Tabel
No.	Nilai	Keterangan
57	0	Tidak Ada
58	00	Tidak Ada
59	000	Tidak Ada
60	0000	Tidak Ada
61	00000	Tidak Ada
62	000000	Tidak Ada
63	0000000	Tidak Ada
64	00000000	Ada Pada Tabel
65	0	Tidak Ada
66	00	Tidak Ada
67	000	Tidak Ada
68	0000	Tidak Ada
69	00000	Tidak Ada
70	000000	Tidak Ada
71	0000000	Tidak Ada
72	00000000	Ada Pada Tabel
73	0	Tidak Ada
74	00	Tidak Ada
75	000	Tidak ada
76	0000	Tidak Ada
77	00000	Tidak Ada
78	000000	Tidak Ada
79	0000000	Tidak Ada
80	00000000	Ada Pada Tabel
81	0	Tidak Ada
82	00	Tidak Ada
83	000	Tidak Ada
84	0001	Tidak Ada
85	00011	Tidak Ada
86	000110	Tidak Ada
87	0001100	Tidak Ada
88	00011000	Ada Pada Tabel
89	0	Tidak Ada
90	01	Tidak Ada

Indeks	Nilai	Keterangan
91	011	Tidak Ada
92	0110	Tidak Ada
93	01100	Tidak Ada
Indeks	Nilai	Keterangan
94	011000	Tidak Ada
95	0110001	Tidak Ada
96	01100010	Ada Pada Tabel
97	0	Tidak Ada
98	01	Tidak Ada
99	011	Tidak Ada
100	0110	Tidak Ada
101	01100	Tidak Ada
102	011001	Tidak Ada
103	0110010	Tidak Ada
104	01100101	Ada Pada Tabel
105	0	Tidak Ada
106	00	Tidak Ada
107	000	Tidak Ada
108	0000	Tidak Ada
109	00000	Tidak Ada
110	000000	Tidak Ada
111	0000000	Tidak Ada
112	00000000	Ada Pada Tabel
113	0	Tidak Ada
114	00	Tidak Ada
115	000	Tidak Ada
116	0000	Tidak Ada
117	00000	Tidak Ada
118	000000	Tidak Ada
119	0000000	Tidak Ada
120	00000000	Ada Pada Tabel
121	0	Tidak Ada
122	00	Tidak Ada
123	000	Tidak Ada
124	0000	Tidak Ada
125	00000	Tidak Ada
126	000000	Tidak Ada
127	0000000	Tidak Ada
128	00000000	Ada Pada Tabel
129	0	Tidak Ada
130	00	Tidak Ada
Indeks	Nilai	Keterangan
131	000	Tidak Ada
132	0000	Tidak Ada
133	00000	Tidak Ada
134	000000	Tidak Ada
135	0000000	Tidak Ada
136	00000001	Ada Pada Tabel
137	0	Tidak Ada
138	00	Tidak Ada
139	000	Tidak Ada
140	0000	Tidak Ada
141	00000	Tidak Ada
142	000000	Tidak Ada
143	0000000	Tidak Ada
144	00000000	Ada Pada Tabel
145	0	Tidak Ada
146	00	Tidak Ada
147	000	Tidak Ada
148	0000	Tidak Ada
149	00000	Tidak Ada
150	000000	Tidak Ada
151	0000000	Tidak Ada
152	00000000	Ada Pada Table

Indeks	Nilai	Keterangan
153	0	Tidak Ada
154	00	Tidak Ada
155	000	Tidak Ada
156	0000	Tidak Ada
157	00000	Tidak Ada
158	000000	Tidak Ada
159	0000000	Tidak Ada
160	00000000	Ada Pada Tabel
161	0	Tidak Ada
162	00	Tidak Ada
163	000	Tidak Ada
164	0000	Tidak Ada
165	00000	Tidak Ada
166	000000	Tidak Ada
167	0000000	Tidak Ada
Indeks	Nilai	Keterangan
168	00000000	Ada Pada Tabel
169	0	Tidak Ada
170	00	Tidak Ada
171	000	Tidak Ada
172	0000	Tidak Ada
173	00000	Tidak Ada
174	000000	Tidak Ada
175	0000000	Tidak Ada
176	00000000	Ada Pada Tabel
177	0	Tidak Ada
178	00	Tidak Ada
179	000	Tidak Ada
180	0000	Tidak Ada
181	00000	Tidak Ada
182	000000	Tidak Ada
183	0000000	Tidak Ada
184	00000000	Ada Pada Tabel
185	0	Tidak Ada
186	00	Tidak Ada
187	000	Tidak Ada
188	0000	Tidak Ada
189	00000	Tidak Ada
190	000000	Tidak Ada
191	0000000	Tidak Ada
192	00000000	Ada Pada Tabel
193	0	Tidak Ada
194	00	Tidak Ada
195	000	Tidak Ada
196	0000	Tidak Ada
197	00000	Tidak Ada
198	000000	Tidak Ada
199	0000000	Tidak Ada
200	00000001	Ada Pada Tabel

Adapun Tabel hasil perhitungan diatas dapat dilihat pada Tabel 9..

Tabel 9.. Nilai Biner *Pixel* Dekompresidan Nilai Desimal

No.	Biner	Desimal	Hexadecimal
1	01110000	112	70
2	00110100	52	34
3	00110010	50	32
4	01101001	105	69
5	01110011	115	73
6	01101111	111	6F
7	01101101	109	6D
8	00000000	00	00
9	00000000	00	00

No.	Biner	Desimal	Hexadecimal
10	00000000	00	00
11	00011000	24	18
12	01100010	98	62
13	01100101	101	65
14	00000000	00	00
15	00000000	00	00
16	00000000	00	00
17	00000001	01	01
18	00000000	00	00
19	00000000	00	00
20	00000000	00	00
21	00000000	00	00
22	00000000	00	00
23	00000000	00	00
24	00000000	00	00
25	00000010	02	02

Berdasarkan hasil dekompresi diatas, maka didapatkan nilai *hexadecimal* awal sebelum kompresi yaitu 70, 34, 32, 69, 73, 6F, 6D, 00, 00, 00, 18, 62, 65, 00, 00, 00, 01, 00, 00, 00, 00, 00, 00, 00, 02.

4. KESIMPULAN

Berdasarkan dari penelitian yang telah dilakukan, maka hasil akhir dari penelitian tersebut dapat diambil beberapa kesimpulan. Adapun kesimpulan Berdasarkan prosedur kompresi dengan menggunakan algoritma *J-Bit Encoding* telah berhasil melakukan proses kompresi *file* video hasil rekam layar berekstensi *.MP4 sehingga proses kompresi dapat berjalan sesuai dengan teknik kompresi. Berdasarkan penerapan algoritma *J-Bit Encoding* telah membuktikan bahwa suatu *file* video yang memiliki ukuran besar dapat dikompres menjadi ukuran yang lebih kecil. Berdasarkan dari hasil pengujian terhadap sistem bahwa ukuran *file* video lebih kecil setelah dilakukan kompresi.

REFERENCES

- [1] D. Salomon and G. Motta, *Handbook Of Data Compression*, Fifth Edit. London: Springer, 2010.
- [2] and D. S. T. Mizwar, G. L. Ginting, Mesran, A. Fau, S. Aripin, “Implementasi Algoritma J-Bit Encoding pada kompresi File Teks,” KOMIK (Konferensi Nas. Teknol. Inf dan komputer), vol. 1, pp. 232–236, 2017.
- [3] R. Syahputra, “KOMPRESI FILE VIDEO MP4 DENGAN MENGGUNAKAN METODE,” pp. 52–57, 2016.
- [4] H. T. Sihotang, “PERANCANGAN DAN IMPLEMENTASI ALGORITMA ARITHMETIC CODING UNTUK APLIKASI KOMPRESI DATA VIDEO DAN AUDIO,” vol. 2, no. 1, pp. 58–64, 2018.
- [5] “KAMUS BESAR BAHASA INDONESIA (KBBI).”
- [6] H. O. D. COMPRESSION, *David Salomon, Giovani Motta*, Fifth. Spinger London Dordrecht Heidelberg New York, 2010.
- [7] K. Y. and Y. Melita, “Aplikasi kompresi Citra digital menggunakan teknik Kompresi jpeg Dengan fungsi GUI pada Matlab,” vol. 3, no. no.2, pp. 269–278, 2011.
- [8] F. G. et Al, “Analisa perbandingan Kompresi dan Dekompresi Menggunakan Algoritma Shannon,” vol. 3, no. no.3, pp. 5197–5204, 2016.
- [9] I. Ozsyald, “the Screencasting Handbook Procast,” 2010.
- [10] A. Nugroho, “Rekayasa perangkat lunak Berorientasi Objek dengan metode USDP (Unfied Software Development Proses).”
- [11] R. A. S.-M. Salahuddin, *Rekayasa Perangkat Lunak Terstruktur dan Berorientasi Objek*, 4th ed. Bandung: Informatika Bandung, 2016.
- [12] R. Priyanto, “Langsung Bisa Visual Basic NET 2008,” 2010.
- [11] Ihsan and D. P. Utomo, “Analisis Perbandingan Algoritma Even-Rodeh Code Dan Algoritma Subexponential Code Untuk Kompresi File Teks,” KOMIK (Konferensi Nas. Teknol. Inf. dan Komputer), vol. 4, no. 1, 2020.
- [12] S. R. Saragih and D. P. Utomo, “Penerapan Algoritma Prefix Code Dalam Kompresi Data Teks,” KOMIK (Konferensi Nas. Teknol. Inf. dan Komputer), vol. 4, no. 1, 2020.
- [13] Lamsah and D. P. Utomo, “Penerapan Algoritma Stout Codes Untuk Kompresi Record Pada Databade Di Aplikasi Kumpulan Novel,” KOMIK (Konferensi Nas. Teknol. Inf. dan Komputer), vol. 4, no. 1, 2020.