KOMIK (Konferensi Nasional Teknologi Informasi dan Komputer)
Volume 6, Nomor 1, November 2022
DOI: 10.30865/komik.v6i1.5716
ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 322-332

Analisa Perbandingan Algoritma Run Length Encoding Dengan Burrows-Wheeler Transform Dalam Kompresi File Video

Muhammad Rizky Ramadhan

Fakultas Ilmu Komputer dan Teknologi Informasi, Prodi Teknik Informatika, Universitas Budidarma, Medan, Indonesia
Email: m.rizky.ramadhan621@gmail.com

Abstrak−Perkembangan teknologi terkini mempengaruhi ukuran file, terutama file berformat video. File video cukup besar untuk menggunakan banyak ruang penyimpanan, semakin panjang durasi video dan kualitas video semakin bagus maka semakin besar pula ukuran file video tersebut. Data dalam jumlah besar juga membutuhkan banyak ruang penyimpanan. Untuk melakukan ini, maka diperlukan solusi seperti pembuatan media penyimpanan yang berkapasitas besar. Namun ke depannya, ukuran file video akan bertambah dan kualitas video akan meningkat, sehingga solusi ini menjadi kurang efektif. Oleh karena itu, diperlukan solusi tambahan untuk menghemat ruang media penyimpanan dan mengurangi ukuran data yang disimpan. Mengurangi penggunaan memori dan ukuran file tanpa mempengaruhi informasi dalam file dapat diatasi dengan kompresi. Kompresi video adalah proses kompresi data yang dilakukan pada file video dengan tujuan memperkecil ukuran file video agar dapat disimpan atau dikirim dengan cepat dan efisien. Pada perbandingan kedua algoritma Run Length Encoding dengan algoritma Burrows-Wheeler Transform sebagai berikut, pada algoritma Run Length Encoding ini bekerja dengan baik pada kompresi video, dan dapat mencapai rasio kompresi yang sangat tinggi, sedangkan algoritma Burrows-Wheeler Transform disimpulkan bahwa ukuran data tidak berkurang sama sekali, tetapi mengalami perubahan atau transformasi data..

Kata Kunci: Run Length Encoding (RLE); Burrows-Wheeler Transform (BWT); Kompresi Video

Abstract−Recent technological developments affect file sizes, especially video format files. The video file is large enough to use a lot of storage space, the longer the video duration and the better the video quality, the larger the video file size. Large amounts of data also require a lot of storage space. To do this, we need solutions such as the manufacture of large-capacity storage media. But in the future, the video file size will increase and the video quality will increase, so this solution becomes less effective. Therefore, additional solutions are needed to save storage media space and reduce the size of the data stored. Reducing memory usage and file size without affecting the information in the file can be overcome by compression. Video compression is a data compression process that is carried out on video files to reduce the size of the video file so that it can be stored or sent quickly and efficiently. In the comparison of the two Run Length Encoding algorithms with the Burrows-Wheeler Transform algorithm as follows, the Run Length Encoding algorithm works well on video compression and can achieve very high compression ratios, while the Burrows-Wheeler Transform algorithm concludes that the data size is not reduced. at all, but undergoes a data change or transformation.

Keywords: Run Length Encoding (RLE); Burrows-Wheeler Transform (BWT); Video Compression

1. PENDAHULUAN

Di zaman yang semakin maju saat ini, penggunaan *gadget* semakin menarik peminat dari yang muda hingga yang tua, dan banyak juga pengguna *gadget* yang menonton video dari *gadget* tersebut melalui aplikasi seperti *youtube*, *facebook*, *instagram*, dan yang lainnya. Perkembangan teknologi yang semakin canggih di saat ini berkembang sangat cepat dan dapat mempengaruhi media penyimpanan dan dalam transmisi data, *file* video merupakan ukuran data yang lebih besar dibandingkan dengan ukuran data pada gambar atau suara maupun pada teks, salah satu format video yang banyak digunakan oleh pengguna adalah format *MP4* atau *MP4 Player*. *MP4* merupakan format video yang umum digunakan karena data yang disimpan terlihat seperti data asli saat direkam dan ukurannya tidak terlalu besar dibandingkan dengan format lain[1].

Ukuran *file* video diukur dari durasi dan kualitas video yang dihasilkan, semakin panjang durasi video dan kualitas video semakin bagus maka semakin besar pula ukuran *file* video tersebut, dikarena kapasitas *file* video yang besar, sehingga akan memakan lebih banyak ruang memori, menggunakan ruang memori yang cukup besar akan membatasi *file* yang dapat disimpan[2], [3].

Agar banyak ruang kosong dan memiliki ukuran data yang kecil pada media penyimpanan, maka harus memperkecil ukuran *file* video saat disimpan, salah satunya dengan melakukan kompresi. Ada banyak contoh algoritma dalam mengkompresi *file* video diantaranya algoritma *Run Length Encoding,* algoritma *Burrows Wheeler Transform,* algoritma *huffman,* dan masih banyak lagi algoritma lainnya, dalam penelitian ini akan membandingkan antara algoritma *Run Length Encoding* dengan algoritma *Burrows-Wheeler Transform* untuk mengetahui rasio kompresi *file* video dengan format *MP4* terbaik agar dapat digunakan dan dapat diaplikasikan oleh banyak orang, pada kompresi data menggunakan rasio kompresi yaitu untuk mengikut persentase data yang telah berhasil dimampatkan. Pengukuran rasio kompresi dimaksudkan untuk melihat seberapa besar pengurangan yang terjadi pada *file* hasil proses kompresi bila dibandingkan dengan *file* aslinya menggunakan kedua algoritma tersebut. Algoritma *Burrows-Wheeler Transform* merupakan algoritma yang bekerja dalam mode blok, di mana *input stream* dibaca blok demi blok dan setiap blok dikodekan secara terpisah sebagai satu string, metode ini disebut sebagai blok penyortiran, sedangkan algoritma *Run Length Encoding* merupakan algoritma pengkodean yang mengkodekan semua data yang memiliki nilai yang sama dan disatukan menjadi nilai data beserta jumlah kemuculan nilai tersebut, oleh karena itu penulis membandingkan algoritma tersebut dalam penelitian ini untuk mengetahui algoritma mana yang memiliki rasio kompresi *file* video terbaik antara algoritma *Run Length Encoding* dengan algoritma *Burrows Wheeler Transform*.

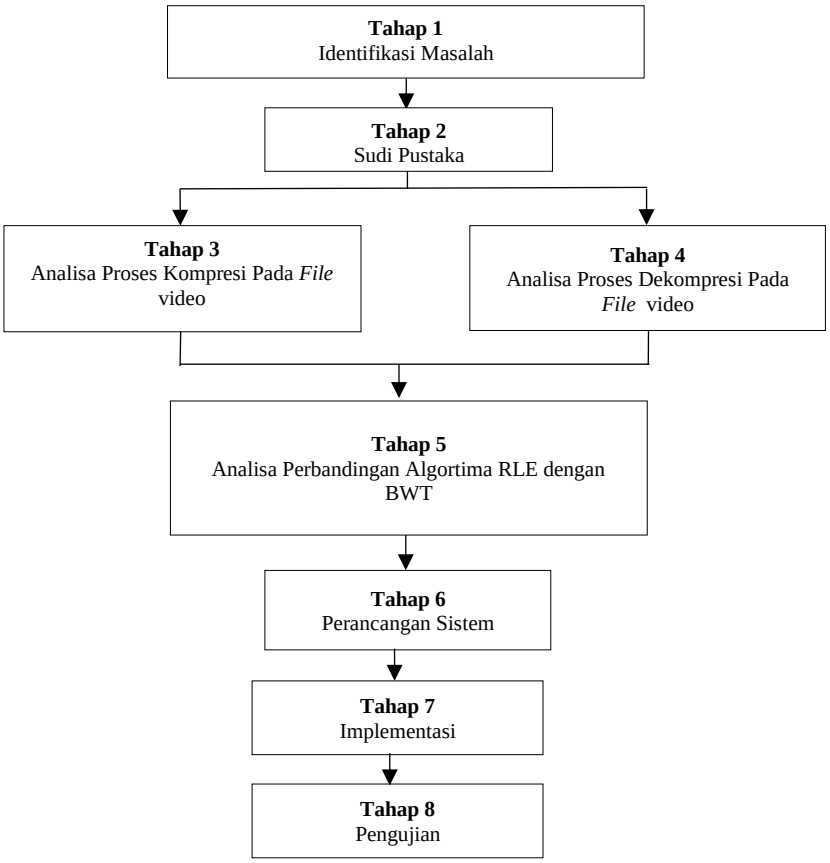
ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 322-332

Berdasarkan penelitian yang dilakukan oleh Sarmiyana Purba yang membahas tentang penerapan algoritma *Burrows Wheeler Transform (BWT)* untuk kompresi *file* citra, maka dapat disimpulkan bahwa algoritma *Burrows Wheeler Transform(BWT)* merupakan algoritma kompresi *file* yang cukup handal dalam kompresi gambar [4]. Pada penelitian lainnya yang dilakukan oleh Asrul Soleh Harahap yang membahas tentang penerapan algoritma *Burrows Wheeler Transform* dalam kompresi *file* citra, menyimpulkan bahwa algoritma *Burrows-Wheeler Transform* memproses data input menjadi antrian, dan secara langsung memproses blok data teks sebagai satu unit, dan karakter yang sama dapat ditemukan secara berurutan. Hasil kompresi *file* gambar sebelumnya terdiri dari 72 bit, tapi setelah dikompres menjadi 22 bit jadi 69,5% [5]. Pada penelitian yang telah dilakukan oleh Cut Try Utari yang membahas tentang algoritma *Run Length Encoding* dalam kompresi dan dekompresi *file* citra, maka dapat disimpulkan bahwa kompresi atau dekompresi gambar dengan menggunakan algoritma *Run Length Encoding(RLE)* dapat mengompresi *file* gambar dalam format *bmp* dan *jpg*, yang dapat memperkecil ukuran *file*, sehingga menghemat ruang penyimpanan [6]. Pada penelitian lainnya yang dilakukan oleh Pujianto, Mujito, Basuki Hari Prasetyo, Danang Prabowo yang membahas tentang perbandingan algoritma *Huffman* dengan algoritma *Run Length Encoding* untuk kompresi dokumen, menyimpulkan bahwa pada algoritma *Huffman* yang mengkompresi semua dokumen ukuran *file* diperkecil, sedangkan pada metode *Run Length Encoding* terdapat 1 *file PDF* yang mengalami peningkatan ukuran sehingga menyebabkan 402237 *byte* menjadi 402813 *byte* [7]. Pada penelitian lainnya yang dilakukan oleh Fatmawati dan Mufty yang membahas mengenai perbandingan algoritma *huffman* dan *Run Length Encoding* dalam kompresi *file Wav*, menyimpulkan bahwa kompresi dengan algoritma *Huffman* memberikan hasil 28,95 %, sedangkan algoritma *Run Length Encoding* menghasilkan kompresi sebanyak 6,77%. Oleh karena itu, algoritma *Huffman* lebih efisien dalam mengompresi jenis *file .wav* daripada algortima *Run Length Encoding* [8].

2. METODOLOGI PENELITIAN

2.1 Tahapan Penelitian

Kerangka kerja penelitian merupakan jenis atau bentuk kerangka kerja yang dapat digunakan sebagai pendekatan untuk pemecahan suatu masalah yang terdiri dari tahapan penilitian yang sistematis. Adapun kerangka kerja penelitian yang akan digunakan adalah sebagai berikut:



Gambar 1. Tahapan Penelitian

Beberapa metode dan teknik yang digunakan dalam pengumpulan dan pengolahan data penelitian ini antara lain:

1. Identifikasi Masalah

ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 322-332

Langkah ini merupakan cara penulis untuk memprediksi, memperkirakan, dan mengkarakterisasi apa yang menjadi sumber masalah pada penyimpanan berkapasitas tinggi sehingga tidak dapat lagi ditampung pada area penyimpanan yang ada.

2. Studi Pustaka
   Pada tahap ini penulis mencari referensi dan literatur tentang teori-teori yang berkaitan dengan permasalahan yang akan diangkat dalam penelitian ini. Referensi akan digunakan dalam bentuk buku, jurnal, artikel, tesis, dan sumber lain yang terkait dengan masalah yang akan penulis bahas dalam studi lebih lanjut.
3. Tahap Analisa Proses Kompresi Pada File Video
   Pada tahap ini dilakukan kompresi file video dengan format MP4 yaitu mambandingkan algoritma *Burrows-Wheeler Transform* dengan algoritma *Run Length Encoding* serta mempelajari teknik dasar pemrograman menggunakan bahasa pemrograman Microsoft Visual Studio 2010.
4. Tahap Analisa Proses Dekompresi Pada File Video
   Proses pengembalian file gambar yang telah dikompresi menjadi kebentuk file asli sebelum dikompresi menggunakan algoritma *Burrows-Wheeler Transform* dan algoritma *Run Length Encoding.*
5. Analisa Perbandingan Algortima RLE Dengan BWT
   Pada proses ini dilakukan perbandingan antara algoritma *Run Length Encoding* dengan algoritma *Burrows-Wheeler Transform*, untuk mengetahui algoritma yang terbaik menurut tiga parameter yaitu *ratio compression*, *compression of ratio* dan *space saving*.
6. Perancangan Sistem
   Pada tahap ini dilakukan pembuatan desain dan perancangan sistem aplikasi yang dapat memodelkan data yang diperoleh.
7. Implementasi
   Pada tahap ini, implementasi desain aplikasi data file MP4 dilakukan dengan menggunakan bahasa pemrograman komputer.
8. Pengujian
   Langkah ini dilakukan untuk memeriksa hasil kompresi file video format .MP4. Proses ini bertujuan untuk menentukan hasil akhir dari pencarian yang diimplementasikan.

2.2 Algoritma *Run Length Encoding* (RLE)

Algoritma RLE (*Run Length Encoding*) merupakan salah satu algoritma yang digunakan untuk mengkompres data menjadi ukuran yang lebih kecil dari ukuran sebelumnya, algoritma ini berguna pada data yang memiliki beberapa nilai yang sama secara berurutan, seperti *file* gambar, dan animasi, dan lainnya namun algoritma ini kurang cocok diterapkan pada data normal karena akan menghasilkan ukuran data yang lebih besar dibandingkan dengan data aslinya [9].
Pada penelitian ini akan membahas dua proses utama dalam pengkompresian *file* video yaitu proses kompresi dan dekompresi dengan menerapkan algoritma *Run Length Encoding* (*RLE*) yang merupakan salah satu teknik kompresi *lossless***,** dengan menjalankan data (yaitu urutan di mana nilai data yang sama terjadi di banyak elemen data yang berurutan) disimpan sebagai satu nilai data dan hitungan, bukan seperti data yang asli. Simbol yang berulang berurutan biasanya disebut berjalan dalam deretan simbol. Oleh karena itu, sumber data yang menarik adalah urutan simbol alfabet. Tujuan dari algoritma *Run-Length Encoding (RLE)* adalah untuk mengidentifikasi jalur dan mencatat panjangnya dari masing-masing *run* dan simbol di jalankan.
Berikut contoh kompresi dengan algoritma *Run Length Encoding* :
12, 12, 12, 12, 12, 12, 12, 12, 35, 76, 112, 87, 87, 87, 87, 5, 5, 5, 5, 1
Hasil kompresinya menjadi :
(8 12), (35), (76), (112), (**4** 87), (**4** 5), (1)
Catatan : nomor dengan garis bawah dan tebal menyatakan banyaknya nilai *pixel* dengan nilai *pixel* ditunjukan oleh nomor berikutnya [10].
Langkah-langkah pada kompresi dengan menggunakan algoritma *Run Length Encoding* adalah:

1. Pembacaan data sampel pada video
2. Kompres data nilai data sampel jika ditemukan data nilai sampel yang sama secara berurutan lebih dari dua dengan cara:
   a. Tambahkan urutan bit untuk mewakili sejumlah nilai data sampel yang sama secara berurutan.
   b. Tambahkan urutan bit yang mewakili karakter yang berulang.
   c. Konversikan seluruh data sampel kompresi ke dalam bentuk hexa.
3. Simpan *file* data video.
4. Hitung rasio kompresi[11].

2.3 Algoritma *Burrows-Wheeler Transform* (BWT)

Algoritma Burrows-Wheeler Transform merupakan algoritma yang mentransformasikan blok data teks menjadi format baru yang berisi karakter yang sama, Algoritma ini mengambil catatan dan mengurutkannya menggunakan algoritma pengurutan[12]. *Burrows and Wheeler* menerbitkan makalah penelitian pada tahun 1994 yang berjudul "*A Block Sorting Lossless Data Compression Algorithm*" yang menyajikan algoritma kompresi data berdasarkan algoritma pengurutan.

ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 322-332

Algoritma Burrows Wheeler mengambil blok data dan memprosesnya menggunakan diagram pengurutan. Algoritma *Burrows Wheeler* dapat dibagi menjadi tiga tahapan, yaitu:

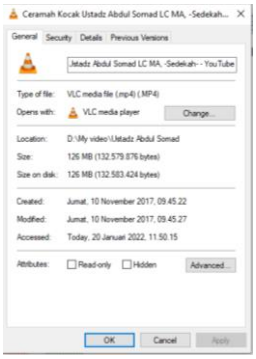
1. *Burrows-Wheeler Transform* artinya mengubah posisi simbol input data sehingga simbol yang sama saling berdekatan..
2. *Global Structure Transform* (*GST*) artinya yang mentransformasi redundansi local menjadi global menggunakan *List of Updated Table*(*LUT*), sehingga menghasilkan sederetan simbol nol,
3. Pengkodean (*Entropy Coding-EC*) merupakan tahap terakhir dari teknik kompresi untuk mentransformasikan data [13].

Pada penelitian ini akan membahas 2 proses utama dalam pengkompresian *file* video yaitu proses kompresi dan dekompresi dengan menerapkan algoritma *Burrows Wheeler Transform (*BWT*)* yang merupakan salah satu teknik kompresi *lossless*. Metode *Burrows-Wheeler Transform* di mulai dengan *string* S dari N simbol dan berlanjut ke *string* L dan string lain yang memenuhi dua kondisi:

a. Setiap wilayah L akan cenderung hanya berisi beberapa simbol. Artinya, jika simbol S terjadi pada posisi tertentu di L, maka kemunculan S lainnya kemungkinan akan ditemukan di dekatnya.
b. Hal ini dimungkinkan untuk merekonstruksi *string* S asli dari L (data yang lebih sedikit dibutuhkan untuk rekonstruksi, selain L, tapi tidak banyak)[14].

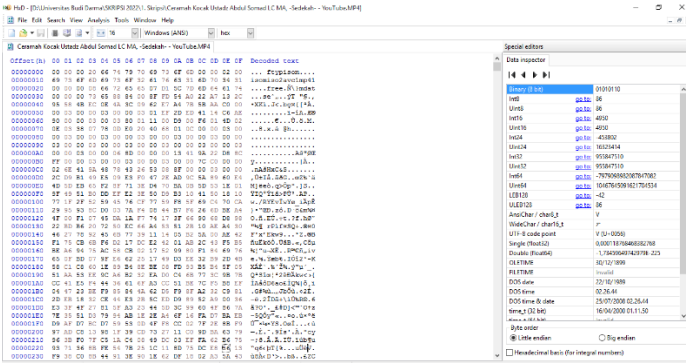
2.4 Sampel Data

Sebagai penerapan algoritma, akan digunakan *file* video dengan judul Ceramah Kocak Ustadz Abdul Somad LC MA, -Sedekah.*MP4*.

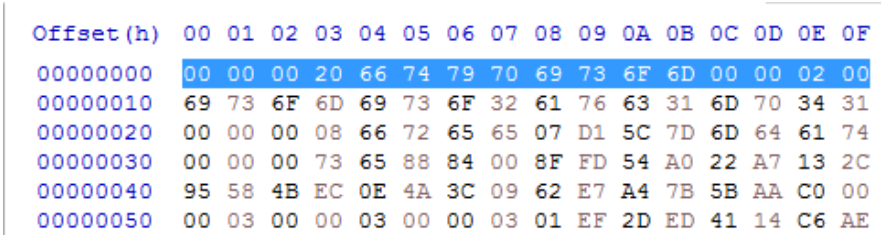


Gambar 2. Detail *File* Video

Untuk memulai kompresi maka kita perlu mencari nilai hexadesimal dari sebuah video yaitu dengan menggunakan bantuan aplikasi Hex editor.



Gambar 3. Detail Bit Nilai



Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00000000	00	00	00	20	66	74	79	70	69	73	6F	6D	00	00	02	00
00000010	69	73	6F	6D	69	73	6F	32	61	76	63	31	6D	70	34	31
00000020	00	00	00	08	66	72	65	65	07	D1	5C	7D	6D	64	61	74
00000030	00	00	00	73	65	88	84	00	8F	FD	54	A0	22	A7	13	2C
00000040	95	58	4B	EC	0E	4A	3C	09	62	E7	A4	7B	5B	AA	C0	00
00000050	00	03	00	00	03	00	00	03	01	EF	2D	ED	41	14	C6	AE

Gambar 4. Nilai *Hexa* Video

Sebagai sampel, maka *hexadecimal* yang digunakan sebanyak 16 yaitu: **00 00 00 20 66 74 79 70 69 73 6F 6D 00 00 02 00**. Dibawah ini adalah tabel ukuran awal dari data sampel.

ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 322-332

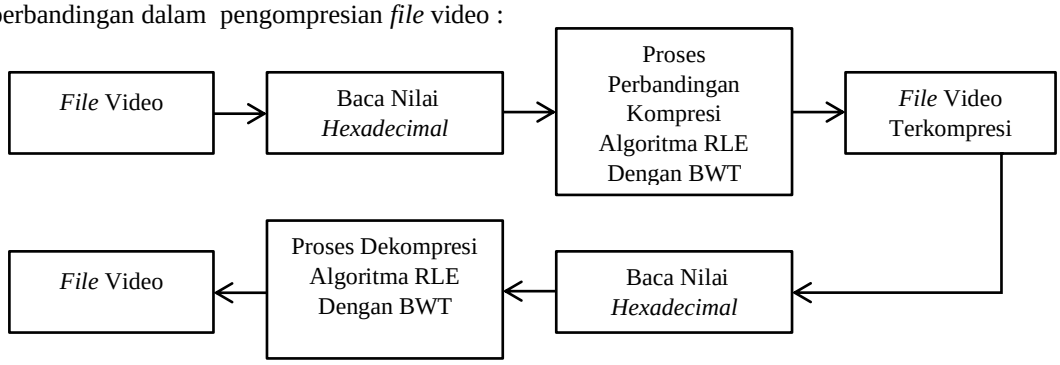
Tabel 1. Ukuran Sampel Sebelum Dikompresi

Hex	*Frequency*	*Binary*	Bit	Bit x *Frequency*
00	6	00000000	8	48
69	1	01101001	8	8
73	1	01110011	8	8
6F	1	01101111	8	8
6D	1	01101101	8	8
20	1	00100000	8	8
66	1	01100110	8	8
74	1	01110100	8	8
79	1	01111001	8	8
70	1	01110000	8	8
02	1	00000010	8	8
Ukuran sebelum dikompresi				128

3. HASIL DAN PEMBAHASAN

3.1 Analisa Masalah

Analisa dalam penelitian ini merupakan perhitungan dan perancangan perangkat lunak pengkompresian *file* video yang dalam prosesnya penulis membandingkan kedua algoritma yaitu algoritma *Run Length Encoding* dengan algoritma *Burrows Wheeler Transforms* dilakukan dengan tujuan agar mengetahui kualitas dari masing masing algoritma, jenis video yang akan dikompresi dengan format *MP4*. Algoritma *Run Length Encoding* dan algoritma *Burrows Wheeler Transforms* merupakan salah satu teknik kompresi *lossless*, dimana dapat memperkecil ukuran kapasitas suatu data berdasarkan karakter yang terdapat pada objek yang akan dikompres. Dalam melakukan kompresi *file MP4* sebelumnya harus dilakukan analisa terhadap *file MP4* yang akan dikompresi. *File MP4* merupakan *file* yang sangat dikenal dan paling populer diantara *file* yang lainnya. Dalam menganalisa *file MP4* yang harus dilakukan adalah mengambil sample *file MP4* dengan melakukan pembacaan *file MP4*. Pembacaan *file MP4* dilakukan untuk mendapatkan nilai dari data pada sebuah *file MP4* yang berupa bilangan *hexadecimal* melalui aplikasi *HxD*, Berikut merupakan alur sederhana proses perbandingan dalam pengompresian *file* video :



Gambar 5. Prosedur Kompresi dan Dekompresi File Video

3.1.1 Penerapan Algoritma *Run Length Encoding* (RLE)

Sebelum *file* video dikompresi dan didekompresi, maka terlebih dahulu mengambil sampel nilai *hexa* yang terdapat pada gambar 3.4, Setelah nilai *hexadesimal* dari *file* video tersebut didapat maka langkah selanjutnya adalah membuat data sampel diatas untuk dilakukan perhitungan sesuai dengan langkah-langkah RLE yaitu dengan menyajikan data diatas ke dalam bentuk tabel. Sebagai sampel, maka hexadecimal yang digunakan sebanyak 16 yaitu: **00 00 00 20 66 74 79 70 69 73 6F 6D 00 00 02 00**. Berikut nilai *hexadecimal* diurutkan berdasarkan frekuensinya, nilai frekuensi paling banyak maka akan berada diurutan pertama.

Tabel 2. Nilai *Hexadecimal* Sebelum Dikompresi

Hex	*Frequency*	*Binary*	Bit	Bit x *Frequency*
00	6	00000000	8	48
69	1	01101001	8	8
73	1	01110011	8	8
6F	1	01101111	8	8
6D	1	01101101	8	8
20	1	00100000	8	8
66	1	01100110	8	8

ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 322-332

Hex	*Frequency*	*Binary*	Bit	Bit x *Frequency*
74	1	01110100	8	8
79	1	01111001	8	8
70	1	01110000	8	8
02	1	00000010	8	8
Ukuran sebelum dikompresi				128

Setelah mengetahui ukuran awal sebelum dikompresi, selanjutnya akan dilakukan proses kompresi dengan algoritma *Run Length Encoding*, setelah mendapatkan nilai sampel dari video yang akan dikompres maka tahap selanjutnya adalah mengompresi dengan mengunakan algoritma *Run Length Encoding*. Langkah pertama periksa nilai *hexa* sebagai string sampel yang ada disampingnya, apabila nilai *hexa* tersebut sama dengan nilai disampingnya maka gabungkan nilai tersebut menjadi satu dan tambahkan nilai *counter* untuk nilai tersebut, Maka ditemukan nilai yang sama pada nilai 00, lakukan hal yang sama pada nilai lainnya apabila nilai selanjutnya tidak ada yang sama, maka simpan nilai tersebut, setelah dicek kembali tidak ada nilai yang sama maka langkah pengompresian telah selesai maka hasilnya seperti berikut:
Hasil kompresi : (**3**,00) (20) (66) (74) (79) (70) (69) (73) (6F) (6D) (**2**,00) (02) (00)
Hasil pengkodean: 3 00 20 66 74 79 70 69 73 6F 6D 2 00 02 00
Setelah hasil kompresi didapat, maka langkah selanjutnya adalah membuat tabel hasil kompresi sebagai berikut:

Tabel 3. Nilai *Hexadecimal* Setelah Dikompresi

Hex	*Frequency*	*Binary*	Bit	Bit x *Frequency*
00	3	00000000	8	24
3	1	00000011	8	8
20	1	00100000	8	8
66	1	01100110	8	8
74	1	01110100	8	8
79	1	01111001	8	8
70	1	01110000	8	8
69	1	01101001	8	8
73	1	01110011	8	8
6F	1	01101111	8	8
6D	1	01101101	8	8
2	1	00000010	8	8
02	1	00000010	8	8
Ukuran setelah dikompres				112

Berdasarkan tabel, maka ukuran akhir dari string "**3 00 20 66 74 79 70 69 73 6F 6D 2 00 02 00**" adalah **112** bit. Setelah dilakukan perhitungan maka didapatkan rasio kompresi dan dapat dilihat berapa rasio pengurangan ukuran yang didapat. Untuk mengetahui tingkat kinerja algoritma kompresi, maka penulis akan mengukur hasil kompresi *file* video tersebut sesuai dengan parameter yang telah ditentukan. Semakin kecil hasil *file* kompresinya maka semakin berkualitas kinerja kompresinya, begitu sebaliknya, Ukuran sebelum dikompres : **128** bit, Ukuran setelah dikompres : **112** bit. Berikut merupakan parameter untuk mengukur kinerja algoritma *Run Length Encoding (*RLE*).*

1. *Ratio of Compression* (RC)
   $$RC = \frac{Ukuran\ Data\ Sebelum\ Dikompresi}{Ukuran\ Data\ Setelah\ Dikompresi}$$
   $$RC = \frac{128}{112} = 1{,}142$$
2. *Compression Ratio* (CR)
   $$CR = \frac{Ukuran\ Data\ Setelah\ Dikompresi}{Ukuran\ Data\ Sebelum\ Dikompresi} \times 100\%$$
   $$CR = \frac{112}{128} \times 100\%$$
   $$CR = 87{,}5\%$$
3. *Space Saving* (*Ss*)
   $$S_s = 100\% - C_R$$
   $$S_s = 100\% - 87{,}5\%$$
   $$S_s = 12{,}5\%$$

Dari perhitungan diatas dapat disimpulkan bahwa persentase hasil kompresi *file* video dengan menggunakan algoritma *Run Length Encoding* sebesar 12,5%.

ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 322-332

3.1.2 Proses Dekompresi *Run Length Encoding*

Proses dekompresi untuk *file* video yang telah dikompres dapat dilakukan dengan mengembalikan karakter menjadi *hexadecimal* seperti semula[15].

1. Nilai hasil kompresi
   (3,00) (20) (66) (74) (79) (70) (69) (73) (6F) (6D) (2,00) (02) (00)
2. Setelah nilai hasil kompresi didapat, maka langkah selanjutnya ialah cari nilai *counter.*
   (**3**,00) (20) (66) (74) (79) (70) (69) (73) (6F) (6D) (**2**,00) (02) (00)
3. Setelah didapatkan nilai *counter* nya, maka ubah nilai *counter* tersebut dengan nilai disampingnya sesuai dengan jumlah nilai *counter*.
   00 00 00 20 66 74 79 70 69 73 6F 6D 00 00 02 00.

Dari penjabaran diatas dapat disimpulkan bahwa hasil dekompresi dari algoritma *Run Length Encoding* dengan melakukan pembacaan ulang keseluran sampel string bilangan hasil kompresi menggunakan algoritma *Run Length Encoding*. Hasil dari dekompresi sesuai dengan *string* bit semula yang terdapat pada *file* video. Berikut adalah hasil dekompresi yang sesuai dengan *string* bit semula yaitu dengan bilangan *hexadecimal* "00 00 00 20 66 74 79 70 69 73 6F 6D 00 00 02 00".

3.1.3 Penerapan Algoritma *Burrows-Wheeler Transform* (BWT)

Sebelum *file* video dikompresi dan didekompresi, maka terlebih dahulu mengambil sampel nilai *hexa* yang terdapat pada gambar 3.4, setelah mendapatkan nilai *hexadesimal* dari *file* video tersebut didapat maka langkah selanjutnya adalah membuat data sampel diatas untuk dilakukan perhitungan sesuai dengan langkah-langkah BWT yaitu dengan menyajikan data diatas ke dalam bentuk tabel. Sebagai *sample*, maka *hexadecimal* yang digunakan sebanyak 16 yaitu: **00 00 00 20 66 74 79 70 69 73 6F 6D 00 00 02 00**.

Tabel 4. Nilai *Hexadecimal* Sebelum Dikompresi

Hex	*Frequency*	*Binary*	Bit	Bit x *Frequency*
00	6	00000000	8	48
69	1	01101001	8	8
73	1	01110011	8	8
6F	1	01101111	8	8
6D	1	01101101	8	8
20	1	00100000	8	8
66	1	01100110	8	8
74	1	01110100	8	8
79	1	01111001	8	8
70	1	01110000	8	8
02	1	00000010	8	8
Ukuran sebelum dikompresi				128

Setelah mengetahui ukuran awal sebelum dikompresi, selanjutnya akan dilakukan proses kompresi dengan algoritma *Burrows Wheeler Transform*. Setelah mendapatkan string sample maka akan dilakukan proses kompresi dengan algoritma *BWT* terlebih dahulu, langkah kompresi dengan algoritma *BWT* adalah sebagai beriut:

1. Pindahkan urutan depan ke urutan belakang sampai yang paling belakang menjadi urutan pertama.

Tabel 5. Langkah-Langkah Kompresi Dengan Algoritma BWT

Index	Nilai															
1	00	00	00	20	66	74	79	70	69	73	6F	6D	00	00	02	00
2	00	00	20	66	74	79	70	69	73	6F	6D	00	00	02	00	00
3	00	20	66	74	79	70	69	73	6F	6D	00	00	02	00	00	00
4	20	66	74	79	70	69	73	6F	6D	00	00	02	00	00	00	00
5	66	74	79	70	69	73	6F	6D	00	00	02	00	00	00	00	20
6	74	79	70	69	73	6F	6D	00	00	02	00	00	00	00	20	66
7	79	70	69	73	6F	6D	00	00	02	00	00	00	00	20	66	74
8	70	69	73	6F	6D	00	00	02	00	00	00	00	20	66	74	79
9	69	73	6F	6D	00	00	02	00	00	00	00	20	66	74	79	70
10	73	6F	6D	00	00	02	00	00	00	00	20	66	74	79	70	69
11	6F	6D	00	00	02	00	00	00	00	20	66	74	79	70	69	73
12	6D	00	00	02	00	00	00	00	20	66	74	79	70	69	73	6F
13	00	00	02	00	00	00	00	20	66	74	79	70	69	73	6F	6D
14	00	02	00	00	00	00	20	66	74	79	70	69	73	6F	6D	00
15	02	00	00	00	00	20	66	74	79	70	69	73	6F	6D	00	00
16	00	00	00	00	20	66	74	79	70	69	73	6F	6D	00	00	02

ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 322-332

2. Urutkan nilai terkecil hingga terbesar

Tabel 6. Langkah-Langkah Pengurutan Nilai Dengan Algoritma BWT

Index	Nilai															
1	00	00	00	00	20	66	74	79	70	69	73	6F	6D	00	00	02
2	00	00	00	20	66	74	79	70	69	73	6F	6D	00	00	02	00
3	00	00	02	00	00	00	00	20	66	74	79	70	69	73	6F	6D
4	00	00	20	66	74	79	70	69	73	6F	6D	00	00	02	00	00
5	00	02	00	00	00	00	20	66	74	79	70	69	73	6F	6D	00
6	00	20	66	74	79	70	69	73	6F	6D	00	00	02	00	00	00
7	02	00	00	00	00	20	66	74	79	70	69	73	6F	6D	00	00
8	20	66	74	79	70	69	73	6F	6D	00	00	02	00	00	00	00
9	66	74	79	70	69	73	6F	6D	00	00	02	00	00	00	00	20
10	69	73	6F	6D	00	00	02	00	00	00	00	20	66	74	79	70
11	73	6F	6D	00	00	02	00	00	00	00	20	66	74	79	70	69
12	70	69	73	6F	6D	00	00	02	00	00	00	00	20	66	74	79
13	74	79	70	69	73	6F	6D	00	00	02	00	00	00	00	20	66
14	79	70	69	73	6F	6D	00	00	02	00	00	00	00	20	66	74
15	6D	00	00	02	00	00	00	00	20	66	74	79	70	69	73	6F
16	6F	6D	00	00	02	00	00	00	00	20	66	74	79	70	69	73

3. Setelah diurutkan nilai tersebut maka ambil dari nilai yang paling ujung pada hasil *BWT* langkah ke dua, maka didapat string baru yaitu:

Tabel 7. Langkah-Langkah Pengurutan Nilai Akhir Kompresi

02	00	6D	00	00	00	00	00	20	70	69	79	66	74	6F	73

Setelah hasil kompresi didapat, maka langkah selanjutnya adalah membuat tabel hasil kompresi sebagai berikut:

Tabel 8. Nilai *Hexadecimal* Setelah Dikompresi

Hex	***Frequency***	***Binary***	**Bit**	**Bit x *Frequency***
00	6	00000000	8	48
02	1	00000010	8	8
6D	1	01101101	8	8
20	1	00100000	8	8
70	1	01110000	8	8
69	1	01101001	8	8
79	1	01111001	8	8
66	1	01100110	8	8
74	1	01110100	8	8
6F	1	01101111	8	8
73	1	01110011	8	8
Ukuran setelah dikompresi				128

Berdasarkan tabel, maka ukuran akhir dari string "**02 00 6D 00 00 00 00 00 20 70 69 79 66 74 6F 73**" adalah **112** bit. Setelah dilakukan perhitungan maka didapatkan rasio kompresi dan dapat dilihat berapa rasio pengurangan ukuran yang didapat. Untuk mengetahui tingkat kinerja algoritma kompresi, maka penulis akan mengukur hasil kompresi *file* video tersebut sesuai dengan parameter yang telah ditentukan. Semakin kecil hasil *file* kompresinya maka semakin berkualitas kinerja kompresinya, begitu sebaliknya, ukuran sebelum dikompres : **128** bit, Ukuran setelah dikompres: **128** bit. Berikut merupakan parameter untuk mengukur kinerja algoritma.

1. *Ratio of Compression* (RC)

$$RC = \frac{Ukuran\ Data\ Sebelum\ Dikompresi}{Ukuran\ Data\ Setelah\ Dikompresi}$$

$$RC = \frac{128}{128} = 1$$

2. *Compression Ratio* (CR)

$$CR = \frac{Ukuran\ Data\ Setelah\ Dikompresi}{Ukuran\ Data\ Sebelum\ Dikompresi} \times 100\%$$

$$CR = \frac{128}{128} \times 100\%$$

$$CR = 100\%$$

3. *Space Saving* (*Ss*)

ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 322-332

$S_s = 100\% - C_R$
$S_s = 100\% - 100\%$
$S_s = 0\%$

Setelah dilakukan proses kompresi dengan menggunakan algoritma *Burrows-Wheeler Transform*, maka dapat disimpulkan bahwa ukuran data tidak berkurang sama sekali.

3.1.4 Penerapan Proses Dekompresi *Burrows-Wheeler Transform*

Proses dekompresi untuk *file* video yang telah dikompres dapat dilakukan dengan mengembalikan karakter menjadi *hexadecimal* seperti semula.

1. Nilai *hexa* Hasil Kompresi

Tabel 9. Nilai Hasil Kompresi

02	00	6D	00	00	00	00	00	20	70	69	79	66	74	6F	73

2. Setelah nilai hexa hasil kompresi didapatkan, maka letak diakhir pengerjaan sama seperti tabel 10.

Tabel 10. Peletakan Nilai Hexa

Index	**Nilai**															
1	00	00	00	00	20	66	74	79	70	69	73	6F	6D	00	00	02
2	00	00	00	20	66	74	79	70	69	73	6F	6D	00	00	02	00
3	00	00	02	00	00	00	00	20	66	74	79	70	69	73	6F	6D
4	00	00	20	66	74	79	70	69	73	6F	6D	00	00	02	00	00
5	00	02	00	00	00	00	20	66	74	79	70	69	73	6F	6D	00
6	00	20	66	74	79	70	69	73	6F	6D	00	00	02	00	00	00
7	02	00	00	00	00	20	66	74	79	70	69	73	6F	6D	00	00
9	66	74	79	70	69	73	6F	6D	00	00	02	00	00	00	00	20
10	69	73	6F	6D	00	00	02	00	00	00	00	20	66	74	79	70
11	73	6F	6D	00	00	02	00	00	00	00	20	66	74	79	70	69
12	70	69	73	6F	6D	00	00	02	00	00	00	00	20	66	74	79
13	74	79	70	69	73	6F	6D	00	00	02	00	00	00	00	20	66
14	79	70	69	73	6F	6D	00	00	02	00	00	00	00	20	66	74
15	6D	00	00	02	00	00	00	00	20	66	74	79	70	69	73	6F
16	6F	6D	00	00	02	00	00	00	00	20	66	74	79	70	69	73

3. Setelah nilai *hexa* didapatkan maka lakukan proses *right cycel permutation* seperti tabel dibawah ini

Tabel 11. Proses *Right Cycel Permutation*

Index	**Nilai**														
1	00	00	00	00	20	66	74	79	70	69	73	6F	6D	00	00
2	02	00	00	00	00	20	66	74	79	70	69	73	6F	6D	00
3	00	02	00	00	00	00	20	66	74	79	70	69	73	6F	6D
4	00	00	02	00	00	00	00	20	66	74	79	70	69	73	6F
5	6D	00	00	02	00	00	00	00	20	66	74	79	70	69	73
6	6F	6D	00	00	02	00	00	00	00	20	66	74	79	70	69
7	73	6F	6D	00	00	02	00	00	00	00	20	66	74	79	70
8	69	73	6F	6D	00	00	02	00	00	00	00	20	66	74	79
9	70	69	73	6F	6D	00	00	02	00	00	00	00	20	66	74
10	79	70	69	73	6F	6D	00	00	02	00	00	00	00	20	66
11	74	79	70	69	73	6F	6D	00	00	02	00	00	00	00	20
12	66	74	79	70	69	73	6F	6D	00	00	02	00	00	00	00
13	20	66	74	79	70	69	73	6F	6D	00	00	02	00	00	00
14	00	20	66	74	79	70	69	73	6F	6D	00	00	02	00	00
15	00	00	20	66	74	79	70	69	73	6F	6D	00	00	02	00
16	00	00	00	20	66	74	79	70	69	73	6F	6D	00	00	02

4. Ambil nilai paling bawah pada akhir *index*.

Tabel 12. Hasil Dekompresi

00	00	00	20	66	74	79	70	69	73	6F	6D	00	00	02

Dari penjabaran diatas dapat disimpulkan bahwa hasil dekompresi dari algoritma *Burrows-Wheeler Transform* dengan melakukan pembacaan ulang keseluran sampel string bilangan hasil kompresi menggunakan algoritma

ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 322-332

Burrows-Wheeler Transform. Hasil dari dekompresi sesuai dengan *string bit* semula yang terdapat pada *file* video. Berikut adalah hasil dekompresi yang sesuai dengan *string bit* semula yaitu dengan bilangan *hexadecimal* "00 00 00 20 66 74 79 70 69 73 6F 6D 00 00 02 00".

3.1.5 Metode Perbandingan *Eksponential*

Adapun penerapan dari metode perbandingan yaitu metode *exponential* dapat dilihat sebagai berikut :

Total nilai (TNi) = $\sum_{j=1}^{m}(Vij)Bj$

Keterangan :

TNi = Jumlah keseluruhan nilai ke –
Vij = Drajat kepentingan relatif kriteria ke-j pada keputusan ke- i yang dapat dinyatatakan dengan skala ordinal (1,2,3,4,5….)
Bj = Kriteria keputusan yang paling terpenting (dinyatakan bobot)
m = Banyaknya kriteria keputusan
n = Banyaknya keputusan yang dipilih
j = 1,2,3……..,m = Banyaknya kriteria
i = 1,2,3……...,n = Banyaknya nilai alternatif yang dipilih

Total Nilai RLE = $\sum_{j=1}^{m}(Vij)Bj$
= ((V1 * BCr)+ (V2 * BCr)+ (V3 * BCr)
= ((1,142 * 1,142) + (87,5% * 1,142) + (12,5% * 1,142) = (1,30 + 0,99 + 0,14)
= 2,43

Total Nilai BWT = $\sum_{j=1}^{m}(Vij)Bj$
= ((V1 * BCr)+ (V2 * BCr)+ (V3 * BCr)
= ((1 * 1) + (100% * 1) + (0% * 1))
= (1 + 1 + 0)
= 2

Dari perhitungan yang sudah dilakukan sebelumnya didapat hasil dari penerapan metode exponential pada algoritma *Run Length Encoding* sebanyak 2,43 sedangkan algoritma *Burrows-Wheeler Transform* sebanyak 2, adapun kriteria selanjutnya untuk membandingkan antara kedua algoritma *Run Length Encoding* dengan algoritma *Burrows-Wheeler Transform* pada saat menganalisis proses dan cara kerjanya yaitu dengan menggunakan parameter kinerja kompresi, untuk mengetahui parameter kinerja algoritma kompresi, maka parameter yang telah dibentuk harus diberikan nilai. Nilai tersebut diambil berdasarkan analisa dari perbandingan algoritma *Run Length Encoding* dan algoritma *Burrows-Wheeler Transform* sebelumnya, seperti pada tabel dibawah ini :

Tabel 14. Pemberian Nilai pada setiap Parameter

Alternatif	Parameter				
	RC		CR		SS
	SB	SD	SB	SD	
Algoritma BWT	128 bit	128 bit	128 bit	128 bit	1-CR
Algoritma RLE	128 bit	112 bit	128 bit	112 bit	1-CR

Keterangan :

RC = *Ratio Of Compression*
CR = *Compression Rasio*
SS = *Space Savings*
SB = Nilai bit sebelum dikompresi
SD = Nilai bit sesudah dikompresi

Berdasarkan data diatas dapat dihitung parameter kinerja kompresinya seperti dibawah ini :

1. *Ratio Of Compression* (Rc)
   Run Length Encoding = $\frac{128}{112}$ = 1,142
   Burrows-Wheeler Transform = $\frac{128}{128}$ = 1
2. *Compression Rasio* (Cr)
   Run Length Encoding = $\frac{112}{128}$ * 100 % = 87,5%
   Burrows-Wheeler Transform = $\frac{128}{128}$ * 100 % = 100%
3. *Space Savings* (Ss)
   Run Length Encoding = 100% - 87,5% = 12,5%
   Burrows-Wheeler Transform = 100% - 100% = 0%

Setelah diperoleh nilai akhir atau total nilai dari setiap alternatif, langkah selanjutnya adalah menentukan prioritas keputusan berdasarkan nilai dari setiap alternatif. Hasil keputusan prioritas dapat dilihat pada tabel berikut :

ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 322-332

Tabel 15. Perbandingan algoritma *Run Length Encoding* dengan algoritma *Burrows-Wheeler Transform*

Alternatif	RC	CR	SS	*Exponential*	*Rangking*
Run Length Encoding	1,142	87,5%	12,5%	2,43	1
Burrows-Wheeler Transform	1	100%	0%	2	2

Setelah dilihat dari tabel diatas, maka dapat disimpulkan bahwa algoritma yang lebih bagus untuk mengkompresi *file* video adalah algoritma *Run Length Encoding* dibandingkan dengan algoritma *Burrows-Wheeler Transform.* Hasil kompresi menggunakan algoritma *Run Length Encoding* lebih besar dibandingkan hasil kompresi menggunakan algoritma *Burrows-Wheeler Transform.*

4. KESIMPULAN

Berdasarkan pembahasan dan evaluasi dari bab-bab sebelumnya dan analisis terhadap kompresi *file* video yang berekstensi MP4, maka dapat disimpulkan dalam penerapkan algoritma *Run Length Encoding* dan algoritma *Burrows Wheeler Transform,* dalam mengkompresi *file* video termasuk kategori baik, karena hasilnya tidak ada pengurangan pada isi *file* nya. Aplikasi yang dirancang melakukan kompresi *file* video dengan menggunakan algoritma *Run Length Encoding* dengan algoritma *Burrows Wheeler Transform,* dan melakukan dekompresi terhadap hasil kompresi dengan *Run Length Encoding* dan algoritma *Burrows Wheeler Transform,* menjadi *file* asli. Hasil perbandingan kompresi *file* video antara *Run Length Encoding* dengan algoritma *Burrows Wheeler Transform* sangat berbeda hasilnya, Algoritma RLE bisa mengompresi lebih baik dengan nilai RC=1,142 CR=87,5% SS=12,5% dan pada perbandingan metode eksponensial dengan nilai 2,43 daripada algoritma BWT dengan nilai RC=1 CR=100% SS=0% dan pada perbandingan metode eksponensial dengan nilai 2 dan ketika diambil sampel bahwa algoritma BWT ketika mengompresi data tidak mengurangi data tetapi hanya mentransformasikan data saja.

REFERENCES

[1] N. Rizka, S. D. Nasution, and K. Ulfa, "Penerapan Algoritma Elias Omega Code Untuk Kompresi File Video Pada Aplikasi Rekam Layar," *Pelita Inform. Inf. dan Inform.*, vol. 9, no. 4, pp. 257–265, 2021.
[2] S. B. Ginting, "Perbandingan Algoritma Yamamoto's Recursive Code Dan Additive Code Dalam Kompresi File Video," *KOMIK (Konferensi Nas. Teknol. Inf. dan Komputer)*, vol. 5, no. 1, 2021.
[3] D. Riyansyah, "Perancangan Aplikasi Kompresi File Video Menggunakan Algoritma Interpolative Coding," *KOMIK (Konferensi Nas. Teknol. Inf. dan Komputer)*, vol. 3, no. 1, 2019.
[4] S. Purba, "Penerapan Metode Kompresi Burrows Wheler Transform Pada Citra Hasil Interpolasi Bicubic," vol. 6, no. 5, pp. 517–522, 2019.
[5] A. S. Harahap, "Analisis Dan Implementasi Kompresi File Citra Menggunakan Algoritma Burrows Wheeler Transform," vol. 1, no. 1, pp. 6–12, 2021.
[6] C. T. Utari, P. Studi, M. Teknik, U. S. Utara, and K. Citra, "IMPLEMENTASI ALGORITMA RUN LENGTH ENCODING UNTUK PERANCANGANAPLIKASI KOMPRESI DAN DEKOMPRESI," vol. V, no. 2, pp. 24–31, 2016.
[7] H. Sinaga, P. Sihombing, and Handrizala, "Perbandingan Algoritma Huffman Dan Run Length Encoding Untuk Kompresi File Audio," vol. 1, no. 1, 2018.
[8] Fatmawati and Mufty, "Analisis Perbandingan Kompresi File Wav Menggunakan Metode Huffman dan Run Length Encoding," vol. 7, no. 1, pp. 61–65, 2020.
[9] E. Prayoga and K. M. Suryaningrum, "Implementasi Algoritma Huffman Dan Run Length Encoding Pada Aplikasi Kompresi Berbasis Web," *J. Ilm. Teknol. Infomasi Terap.*, vol. 4, no. 2, pp. 92–101, 2018, doi: 10.33197/jitter.vol4.iss2.2018.154.
[10] D. Putra, *Pengolahan Citra Digital*. Yogyakarta: C.V ANDI OFFSET, 2010.
[11] D. Willfrid, M. Simamora, G. Ginting, and Y. Hasan, "Implementasi Algoritma Run Length Encoding Pada Kompresi File Mp3," *JURIKOM (Jurnal Ris. Komputer)*, vol. 3, no. 4, pp. 5–9, 2016.
[12] G. Hasibuan, "Analisa Kombinasi Algoritma Burrows Wheeler Transform dan Adaptive Huffman Coding untuk Kompresi Citra," *Bull. Multi-Disciplinary Sci. Appl. Technol.*, vol. 1, no. 2, pp. 34–40, 2022.
[13] I. R. Lubis, "MENGGUNAKAN METODE EKSPONENSIAL," vol. 6, pp. 184–186, 2017.
[14] D. Salomon and G. Motta, *Handbook of Data Compression*, 5th ed. Springer, 2010. doi: 10.1007/10.1007/978-1-84882-903-9.
[15] 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.