Analisis Perbandingan Kompresi File Wav Menggunakan Metode Huffman dan Run Length Encoding

Fatmawaty
Teknik Informatika
Univeristas Budi Luhur
Jakarta, Indonesia
fatmawaty.2009@gmail.com

Mufty
Teknik Informatika
Univeristas Budi Luhur
Jakarta, Indonesia
muftyhayat@gmail.com

***Abstract*— Audio format is the medium of storing audio and music data in physical form and recording Format of an audio content. Nowadays there are a variety of audio formats such as Wave, MP3, WMA and many other formats, but in the Windows operating system used are wave type files are widely used for multimedia and games in the Windows operating system. But if we want to record sound audio CD quality then it takes a huge storage media so that it makes wasteful of data storage or hard disk so as to reduce the storage capacity and also for sending the file takes a long time so we have to compress the file so that storage is more efficient and delivery is faster. The Huffman method and the Run Length Encoding are used to compress the file so that the size becomes efesien, the research results indicate that the Huffman method can minimize the size of a WAV file with an average compression ratio of 28,954% while the Run Length Encoding algorithm can minimize the size of a WAV file with an average compression ratio of-46.77%**

Keywords— audio; wave; storage; multimedia; sound

***Abstrak*— Format Audio (Audio Format) adalah medium penyimpanan data audio dan musik dalam bentuk fisik dan format rekaman sebuah konten audio. Saat ini terdapat berbagai macam format audio contohnya adalah wave,Mp3, wma dan masih banyak format lainnya, namun dalam sistem operasi windows yang digunakan adalah file bertipe wave tipe ini banyak sekali digunakan untuk keperluan multimedia dan game di dalam sistem operasi windows. Namun jika kita ingin merekam suara sekualitas cd audio maka diperlukan suatu media penyimpanan yang sangat besar sehingga membuat boros penyimpanan data atau hard disk sehingga untuk mengurangi kapasitas penyimpanan tersebut dan juga untuk pengiriman file tersebut membutuhkan waktu yang lama sehingga kita harus mengkompres file tersebut sehingga penyimpanan lebih efesien dan pengiriman menjadi lebih cepat. Metode huffman dan Run Length Encoding digunakan untuk mengkompres file agar ukuran menjadi efesien, hasil penelitian menunjukan bahwa metode huffman dapat memperkecil ukuran sebuah file wav dengan Rata-rata ratio kompresi sebesar 28.954 % sedangkan algoritma Run Length Encoding dapat memperkecil ukuran sebuah file wav dengan Rata-rata ratio kompresi sebesar -46.77%.**

Keywords— audio; gelombang; penyimpanan; multimedia; suara

PENDAHULUAN

perekaman suara dengan kualitas cd audio diperlukan untuk keperluan multimedia dan permainan dalam sistem operasi windows. akan tetapi memerlukan tempat penyimpanan yang sangat besar, sehingga membuat boros tempat penyimpanan data sehingga kapasitas harddisk menjadi sangat berkurang. dan permasalahan yang lain adalah lamanya proses pengiriman file ketika file suara tersebut dipanggil. sehingga untuk mengatasi permasalahan tersebut maka penelitian ini akan menggunakan metode huffman dalam proses kompresi file audio menjadi format wave sehingga file tersebut menjadi lebih efesien dan tidak mengurangi kualitas suara hasil kompresi. Penelitian serupa juga telah dilakukan oleh Eka Prayoga dan Kristien Margi Suryaningrum yang berjudul Implementasi Algoritma Huffman Dan Run Length Encoding Pada Aplikasi Kompresi Berbasis Web. pada penelitian tersebut kompresi data menggunakan algoritma huffman dan dipadukan dengan Run Length Encoding file yang dikompresi adalah file dengan jenis txt sedangkan pada penelitian ini file yang akan di compress adalah .wav penelitian tersebut menggunakan 5 buah data uji dan hasil penelitian menunjukan ukuran file hasil dekompres tidak seperti semula karena proses kompresi bersifat lossy [1] penelitian yang lain juga dilakukan oleh Andysah yang berjudul Implementasi Teknik Kompresi Teks Huffman. penelitian tersebut mengkompres sebuah teks dengan panjang 33 karakter dan hasil setelah dikompresi menjadi 12 karakter sehingga menghemat 21 karakter. sehingga tingkat kompresi mencapai 63% dari pesan asli [2]. Dan penelitian yang membahas kompres data huffman audio adalah Supiyandi dan Okta Frida yang berjudul Analisi perbandingan pemampatan data teks dengan menggunanakan metode huffman dan half-byte. pada penelitiannya tersebut membadningkan 2 buah jenis algoritma pemampatan atau kompresi yaitu algoritma huffaman dengan Half-Byte dengan jenis data teks. hasilnya pemampatan atau kompresi yang lebih baik dan mendapatkan nisbah dengan nilai yang lebih tinggi adalah huffman [3]. Sedangkan pada penelitian ini menggunakan metode huffman dan metode Run Length

Encoding Penelitian lainnya dilakukan oleh Ulfa Lu'luilmaknun, Nilza Humaira Salsabila yang berjudul Penggunaan Metode Run Length Encoding Untuk Kompresi Data. dalam peneliannya tersebut peneliti mencoba mengkompresi sebuah citra grafis sebanyak 28 Citra RGB (Red, Green, Blue) dan 28 Citra Grayscale dengan masing-masing format jpg, png, bmp dan tiff. hasil kompresi pada 28 citra RGB yang diuji. dihasilkan 1 buah citra yang berhasil dan efektif. 27 lainnya tidak efektif. sedangkan 28 citra grayscale yang diujicoba dihasilkan 6 buah citra yang efektif dan berhasil dan 22 citra tidak berhasil [4]. Penelitian lainnya juga dilakukan Darno Willfrid Midukta Simamora, Garuda Ginting dan Yasir Hasan yang berjudul Implementasi Algoritma Run Length Encoding pada kompresi file MP3. pada penelitiannya menggunakan file tipe hanya MP3 dengan ukuran awal sebuah file sebesar 4,382 kilobytes dan setelah berhasil dikompres menjadi 4,321 kilobytes dengan ratio kompres 99%. dan pada saat di dekompresi ukuran awal file dekompres sebesar 4,321 kilobytes dan setelah didekompresi ke ukuran awal sebesar 4,321 kilobytes. sehingga file tidak mengalami perubahan ketika dikembalikan kekondisi semula [5] penelitian lainnya yang membahas metode huffman dilakukan oleh Yetti Rahayu Nasution,Asahar Johar dan Funny Farady Coastera yang berjudul Aplikasi Penyembunyian Multimedia Menggunakan Metode End Of File (Eof) Dan Huffman Coding. pada penelitiannya menggunakan 2 buah metode yaitu End Of File dan metode huffman. pada metode End Of File diperoleh hasil kompresi untuk teks sebesar 55,076 %, Gambar sebesar 0656 %, Audio 4,788 % dan Video 4,04 %. sedangkan pada metode huffman untuk pengujian teks mendapatkan ratio 55,076 %, file gambar sebesar 11.62% dan audio 4,788. % pada penelitian tersebut mengambil kesimpulan bahwa algoritma huffman coding kurang efektif jika digunakan untuk mengkompresi media gambar, audio dan video. dan algoritma End Of File efektif digunakan sebagai algoritma penyisipan untuk data dengan ukuran yang sangat besar [6].

A. Algoritma Huffman

Algoritma Huffman adalah salah satu algoritma kompresi tertua yang disusun oleh David Huffman pada tahun 1952. Algoritma tersebut digunakan untuk membuat kompresi jenis lossy compression, yaitu pemampatan data dimana tidak satu byte pun hilang sehingga data tersebut utuh dan disimpan sesuai dengan aslinya [7].Algoritma Huffman menggunakan prinsip pengkodean yang mirip dengan kode Morse, yaitu tiap karakter (simbol) dikodekan hanya dengan rangkaian beberapa bit, dimana karakter yang sering muncul dikodekan dengan rangkaian bit yang pendek dan karakter yang jarang muncul dikodekan dengan rangkaian bit yang lebih panjang [8].

Contoh kita akan mengkompres sebuah karakter AABCCEDABABEDCDCC dengan ukuran 17 byte. Menggunakan metode huffman, langkah-langkahnya adalah :

1. Menghitung frekwesni jumlah karakter yang muncul yaitu A sebanyak 4 kali , B sebanyak 3 kali, C sebanyak 5 kali, D sebanyak 3 kali dan E sebanyak 2 kali.
2. Langkah berikutnya adalah mengurutkan jumlah karakter yang muncul paling sedikit ke karakter yang muncul dengan jumlah besar sehingga didapatkan urutan yaitu E, B, D, A, C
3. Membuat sebuah pohon biner dengan dasar jumlah yang paling kecil ke aarah jumlah yang paling besar, seperti pada Fig. 1.

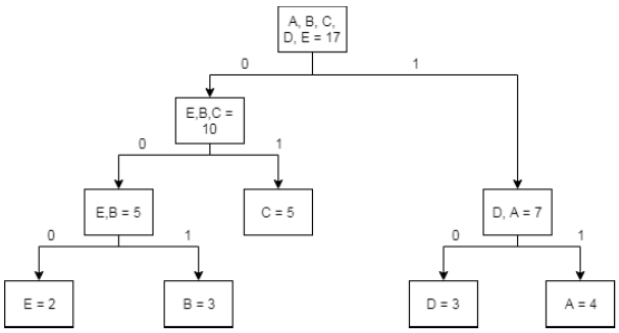


Fig. 1. Pohon biner

4. Mengganti sejumlah data karakter menggunakan kode bit yang didapatkan pada hasil pohon biner. Proses penggantian karakter dapat dilihat pada posisi root node sehingga pergantian karakter yaitu A diganti dengan 11, B diganti dengan 001, C diganti dengan 01, D diganti dengan 10, E diganti dengan 000.

5. Langkah terakhir adalah mengganti semua karakter yaitu AABCCEDABABEDCDCC diganti dengan kode biner yang didapat sehingga menjadi 11110010101000101100111001000 1001100101.

Hasil akhir dari kompresi karakter tersebut, ukuran file yang semula adalah 17 byte dan jika dikonversi ke ukuran bit adalah 17 x 8 = 136 bit. setelah di kompresi menggunakan metode huffman menjadi 39 bit. Sehingga efesiensi ukuran menjadi 3,49 kali menjadi lebih kecil.

B. Algoritma Run Length Encoding.

Proses kompresi data didasarkan pada kenyataan bahwa pada hampir semua jenis data selalu terdapat pengulangan pada komponen data yang dimilikinya, misalnya di dalam suatu data audio akan terdapat pengulangan penggunaan angka 0 sampai 9 atau huruf a sampai huruf z. Kompresi data melalui proses encoding bertujuan untuk menghilangkan unsur pengulangan ini dengan mengubahnya sedemikian rupa sehingga ukuran data menjadi lebih kecil. Proses pengurangan unsur pengulangan ini dapat dilakukan dengan memakai beberapa teknik kompresi. Misalnya

DOI: https://doi.org/10.25047/jtit.v7i1.139

jika suatu komponen muncul berulang kali dalam suatu data, maka komponen tersebut tidak harus di kodekan berulang kali, tapi dapat dikodekan dengan menulis frekuensi muncul komponennya dan dimana komponen tersebut muncul. Teknik kompresi data Run Length Encoding adalah suatu algoritma kompresi data yang bersifat lossless [9] Algoritma ini mungkin merupakan algoritma yang mudah untuk dipahami dan diterapkan untuk kompresi. Metode kompresi ini sangat sederhana, yaitu hanya memindahkan pengulangan byte yang sama berturut-turut secara terus-menerus dan teknik run length encoding ini bekerja berdasarkan sederetan karakter yang berurutan [10] Adapun cara kerja dari Algoritma Run Length Encoding yaitu langkah pertama pembacaan sampel audio kemudian kompresi data nilai jika ditemukan data nilai sampel yang sama secara berurutan lebih dari dua dengan cara: Berikan bit penanda # pada data kompresi kemudian Tambahkan deretan bit untuk menyatakan jumlah nilai data yang sama berurutan. Setelah itu Tambahkan deretan bit yang menyatakan karakter yang berulang. kemudian Konversikan semua data kompresi ke dalam hexa. kemudian Simpan file audio dan terakhir Hitung rasio kompresi [11]

Contoh kita akan mengkompres karakter AACBBBACCBBBDDDDEEE. Menggunakan metode Run Length Encoding maka didapatkan hasil sebagai berikut : (A, 2) karakter A muncul sebanyak 2 kali, (C, 1) karakter C muncul sebanyak 1 kali, (B, 3) karakter B muncul sebanyak 3 kali, (A, 1) karakter A muncul sebanyak 1 kali, (C, 2) karakter C muncul sebanyak 2 kali, (B, 3) karakter B muncul sebanyak 3 kali, (D, 4) karakter D muncul sebanyak 4 kali dan (E, 3) karakter E muncul sebanyak 3 kali

Hasil akhir kompresi karakter yang awalnya berjumlah 18 byte menjadi 16 byte. atau 18 / 16 = 1,125 kali menjadi lebih kecil.

Proses Konversi Analog Ke Digital

Di dalam sebuah komputer untuk mengenal sebuah suara haruslah dalam bentuk digital yaitu sebuah tegangan yang diterjemahkan dengan angka “0” dan “1” atau sering disebut dengan istilah bit. [12] komputer akan membaca bit – bit tersebut sehingga menghasilkan suatu data atau informasi sesuai yang kita inginkan. Untuk mengubah sinyal analog kedalam sinyal digital diperlukan sebuah alah bantu yaitu *transducer* yaitu sebuah peralatan yang dapat mengubah tekanan udara (suara) kedalam tegangan elektrik yang dapat dimengerti oleh peralatan elektronik. Contoh transducer yaitu Mikrofon dan speaker. Adapun fungsi mikrofon mengubah tegangan udara menjadi tegangan elektrik dan speaker mengubah menjadi tegangan elektrik menjadi suara. Adapun didalam computer tegangan elektrik diproses menjadi sinyal digital oleh *sound card.* contoh : kita ingin menyanyikan sebuah lagu dan kita rekam ke dalam computer maka prosesnya adalah tegangan elektrik diproses menjadi sinyal digital oleh *sound card* kemudian sound card akan mengubah menjadi gelombang suara menjadi digital dan ketika lagu yang sudah kita rekam akan kita mainkan maka prosesnya adalah sound card akan mengubah sinyal digital menjadi sinyal analog melalui media speaker [13]Berikut akan ditampilkan proses konversi secara terperinci di dalam komputer

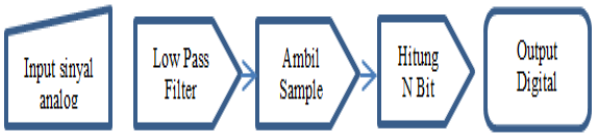


Fig. 2. Proses analog ke digital

Keterangan :
Pada Fig. 2, proses analog ke digital dimulai dengan Input sinyal analog (misal kita merekam sebuah suara melalui media mikrofon dan kita simpan) kemudian Membatasi Frekwensi dengan dengan low pass filter, low pass filter dipilih karena frekwensi suara yang kita inginkan tidak terlalu tinggi sehingga ukurannya tidak menjadi besar. setelah itu Kita ambil sampel sinyal analog menjadi beberapa potongan waktu dan langkah terakhir adalah masing-masing sampel yang telah kita ambil kita berikan nilai dalam bentuk digital.
Dan untuk mengubah dari sinyal digital ke menjadi suara (sinyal analog) proses nya adalah kebalikan dari proses pertama yaitu :

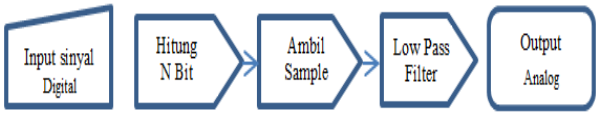


Fig. 3. Proses digital ke analog

Keterangan :
Untuk proses digital ke analog seperti pada Fig. 3, Langkah pertama untuk proses digital ke analog adalah Ambil sebuah sinyal digital (misal suara yang telah disimpan) kemudian hitung nilai digitalnya setelah itu baca sample yang telah dipotong-potong perbagian dan langkah terakhir adalah Membaca sampel yang frekwensi nya telah di batasi dan mainkan suara (sinyal analog).

Penyelesaian sebuah masalah pengiriman file suara yang bertipe .wav tersebut adalah dengan memanfaatkan metode kompresi Huffman dan Run Length Encoding, file suara tersebut nantinya akan dikompres dan hasil kompres file suara dapat dimainkan kembali tanpa mengurangi kualitas dari suara asal. *File* merupakan data digital yang berupa representasi atas bit ‘0’ dan ‘1’. Seringkali dalam sebuah *file* terjadi perulangan data atau *redundancy*. Semua metode kompresi melakukan pemadatan terhadap data berulang tersebut. Seperti diketahui jenis algoritma kompresi terbagi atas *lossless*

compression dan *lossy compression*. Pada *lossy compression* ada data yang hilang tetapi tidak banyak setelah data dikompresi. Contoh standar yang menggunakan jenis *lossy compression* adalah JPEG (*Joint Picture Experts Group*) sebagai standar *image* gambar atau *still image*, MPEG (*Motion Picture Experts Group*) untuk *audio video* sepeti Video CD, MP3 (*MPEG-1 Layer 3*) untuk *audio*. Data hasil kompresi dengan lossy compression jika dikembalikan maka hasilnya tidak akan sama persis lagi dengan data orisinil. Berbeda dengan *lossy compression*, pada *lossless compression* tidak ada data yang hilang setelah proses kompresi dan data dapat dikembalikan seperti data semula. Contoh standar yang menggunakan jenis ini adalah Gzip, Unix Compress, WinZip, GIF (*Graphic Interchange Forma*t) untuk *still image*, dan Morse Code.

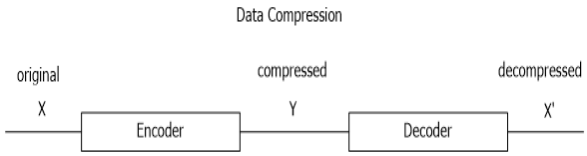


Fig. 4. Skema Proses Kompresi dan Dekompresi

Proses kompresi dan dekompresi seperti pada Fig. 4, dimulai dari sebuah file X original kemudian di encoder menjadi file Y yaitu file sesudah di kompresi. Untuk proses Dekompresi dimulai dengan memilih file Y yaitu file yang telah dikompres kemudian di decoder menjadi file X^1 yaitu file original yang telah didekompres

METODE PENELITIAN

Metode penelitian digunakan untuk menggambarkan proses penelitian yang berlangsung, dimulai dari collecting data sampai dengan tahap pengujian sistem. Berikut adalah metode penelitian secara keseluruhan

1. Collecting Data
   Penelitian ini dilakukan dengan langkah pertama adalah mengumpulkan data-data audio yang berjenis .wav pada tempat penelitian.
2. Mengolah Data
   Setelah proses pengumpulan data seelsai maka tahapan berikutnya adalah mengolah data penelitian, file-file audio berjenis .wav akan diolah menggunakan metode Huffman dan metode Run Length Encoding.
3. Mengembangkan Sistem
   Pada tahapan ini akan membahas mengenai perancangan dari bentuk sistem yang terkait. Perancangan dilakukan untuk menggambarkan proses kerja dari metode Huffman dan Run Length Encoding.
4. Implementasi Sistem
   Setelah proses pengembangan sistem berhasil dilakukan maka langkah berikutnya adalah mengimplementasikan sistem yang telah berhasil dibuat.
5. Pengujian Sistem
   Langkah terakhir adalah menguji sistem yang telah berhasil dibangun. Pada tahapan ini sistem akan menguji file audio berjenis .wav menggunakan algoritma Huffman dan Run Length Encoding. Sehingga hasil akhir penelitian adalah bentuk perbandingan hasil kompresi file .wav menggunakan algoritma Huffman dan Run Length Encoding. Sehingga penelitian ini dapat melihat metode kompresi mana yang lebih baik terhadap file berjenis .wav.

HASIL DAN PEMBAHASAN

Setelah melakukan uji coba menggunakan 5 buah file audio maka didapatkan data sebagai berikut :

Tabel 1. ujicoba menggunakan metode huffman

Nama file	Ukuran file Asli	Ukuran file Kompres	Lama proses (ms)	Ratio
a.wav	21,8 Kb	6.8 Kb	47	63.85 %
b.wav	137 Kb	117 Kb	62	17.55 %
c.wav	106 Kb	71.9 Kb	63	31.41 %
d.wav	172 Kb	138 Kb	94	19.55 %
e.wav	131 Kb	116 Kb	59	10.28 %
Rata-rata ratio kompresi				28.954 %

Dari Tabel 1 ujicoba menggunakan metode huffman dengan nama file a.wav, b.wav, c.wav, d.wav dan e.wav diperoleh ratio 63.85%, 17.55 %, 31.41%, 19.55% dan 10.28 dengan rata-rata ratio kompresi 28.954%. dari data tersebut dapat terlihat bahwa file a.wav memiliki ukuran file 21,8 Kb setelah dikompress menjadi 6.8 Kb. Dan memiliki lama waktu kompress 47 mili second dengan ratio kompress sebesar 63.85 %. sedangkan file d.wav dengan ukuran paling besar yaitu 172 Kb setelah di kompres menjadi 138 Kb memerlukan waktu 94 mili second dengan ukuran ratio 19.55 % . sehingga file a.wav dengan ratio kompress 63.85% yang paling baik mendapatkan pengkompresan.

Tabel 2. ujicoba menggunakan metode Run Length Encoding

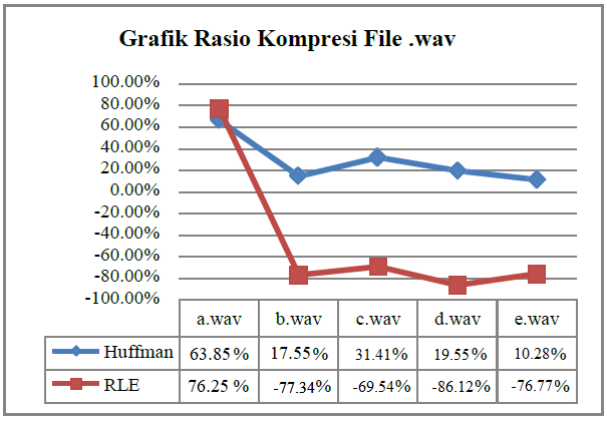
Nama file	Ukuran file Asli	Ukuran file Kompres	Lama proses (ms)	Ratio
a.wav	21,8 Kb	16.9 Kb	50	76.25 %
b.wav	137 Kb	242 Kb	97	-77.34 %
c.wav	106 Kb	179 Kb	78	-69.54 %
d.wav	172 Kb	320 Kb	110	-86.12 %
e.wav	131 Kb	230 Kb	96	-76.77 %
Rata-rata ratio kompresi				-46.77 %

Dari Tabel 2 ujicoba menggunakan metode Run Length Encoding dengan nama file a.wav, b.wav, c.wav, d.wav dan e.wav diperoleh ratio 76.25%, -77.34%, -69.54%, -86.12% dan -76.77 dengan rata-rata ratio kompresi -46.77%. dari data kompres menggunakan metode Run Length Encoding diperoleh hasil file a.wav ukuran asli file 21.8 Kb setelah dikompresi menjadi 16.9 Kb dengan lama waktu 50 mili second dengan ratio 76.25 %. Sedangkan file d.wav ukuran file semula yaitu 172 Kb dikompres menjadi 320 Kb dengan lama waktu 110 mili second dan mendapatkan ratio kompres -86.12%.

DOI: https://doi.org/10/25047/jtit.v7i1.139

sehingga file a.wav mendapatkan hasil yang paling baik didalam kompresi file.

Dari percobaan pada Tabel 1 dan Tabel 2 dapat dilihat rata-rata rasio kompresi algoritma Huffman pada file *.wav adalah 28.954 % sedangkan algoritma RLE adalah -46,77 %.Untuk mengetahui lebih jelas perbandingan kedua algoritma dalam rasio kompresi dapat dilihat pada Fig. 5 berikut.



	a.wav	b.wav	c.wav	d.wav	e.wav
Huffman	63.85 %	17.55%	31.41%	19.55%	10.28%
RLE	76.25 %	-77.34%	-69.54%	-86.12%	-76.77%

Fig. 5. Grafik Rasio Kompresi File .wav

Pada grafik perbandingan ratio kompresi file yang ditunjukan pada Fig. 5, Untuk kompresi menggunakan metode huffman yang diberi warna biru terlihat tidak menyentuh angka dibawah 0.00% namun untuk kompresi menggunakan metode Run Length Encoding yang diberi warna merah terlihat hanya 1 buah file yakni a.wav yang mendapatkan angka diatas 0.00% yakni 63.85% sedangkan yang lainnya mendapatkan penilaian dibawah 0.00%. sehingga metode huffman lebih baik didalam melakukan pengkompresan file dibandingkan metode Run Length Encoding.

KESIMPULAN

Setelah mengamati dan mempelajari penelitian tersebut maka dapat diambil kesimpulan kompresi menggunakan metode huffman diperoleh hasil 28.954% dan menggunakan Run Length Encoding diperoleh hasil -46.77%. Kompresi menggunakan metode Run Length Encoding terdapat 4 buah file yang mendapatkan ratio dibawah 0,00 %. Kompresi metode huffman lebih baik didalam melakukan kompresi jenis file .wav jika dibandingkan dengan metode Run Length Encoding.

PENGHARGAAN

Terima kasih kepada bapak Basuki Hari Prasetyo, S.Kom, M.Kom yang telah membantu penulis dalam melakukan penelitian

.

REFERENSI

[1] E. Prayoga and K. M. Suryaningrum, "Implementasi Algoritma Huffman Dan Run Length Encoding Pada Aplikasi Kompresi Berbasis Web," *J. Ilm. Teknol. Inf. Terap.*, vol. IV, no. 2, pp. 92–101, 2018.

[2] A. Putera and U. Siahaan, "Implementasi Teknik Kompresi Teks Huffman," *J. Inform.*, vol. 10, no. 2, pp. 1251–1261, 2016.

[3] O. Frida *et al.*, "Analisis Perbandingan Pemampatan Data Teks Dengan Menggunakan Metode Huffman Dan Half – Byte," *Algoritm. J. Ilmu Komput. dan Inform.*, vol. 6341, no. April, pp. 1–6, 2018.

[4] U. Lu and N. H. Salsabila, "Penggunaan Metode Run Length Encoding Untuk Kompresi Data," *Semin. Mat. DAN Pendidik. Mat. UNY 2017*, no. 1, pp. 273–280, 2017.

[5] D. Willfrid, M. Simamora, G. Ginting, and Y. Hasan, "Implementasi Algoritma Run Length Encoding Pada Kompresi File Mp3," vol. 3, no. 4, pp. 5–9, 2016.

[6] Y. Rahayu, A. Johar, and F. F. Coastera, "Aplikasi Penyembunyian Multimedia Menggunakan Metode End Of File (Eof) Dan Huffman Coding," *J. Rekursif*, vol. 5, no. 1, pp. 86–106, 2017.

[7] R. A. Suhermanto, "Time Optimization for Lossy Decompression of the LISA Sensor Data on LAPAN A3 Satellite Using a Grouping Method of HUFFMAN Code Bit Number," *J. Teknol. Dirgant.*, vol. 16, no. 1, p. 23, 2018.

[8] M. Aria and A. Sanjaya, "Teknik Kompresi pada Transmisi Data Citra Payload KOMURINDO," *Komputika J. Sist. Komput.*, vol. 7, no. 2, pp. 103–111, 2018.

[9] Y. A. Jaya, L. Chrisantyo, and W. S. Raharjo, "Pengembangan Dan Analisis Kombinasi Run Length Encoding Dan Relative Encoding Untuk Kompresi Citra," *J. Inform.*, vol. 12, no. 2, pp. 141–150, 2016.

[10] C. T. Utari, "Implementasi Algoritma Run Length Encoding Untuk Perancangan Aplikasi Kompresi Dan Dekompresi File Citra," *J. TIMES*, vol. V, no. 2, pp. 24–31, 2016.

[11] A. P. Gilang Rizki Akbar, Rosa Andrie Asmara, "Analisis perbandingan algoritma rle dan huffman pada kompresi citra," *Semin. Inform. Apl. 2019*, vol. 1, no. 1, pp. 323–329, 2019.

[12] Anna, "Modul Converter (Adc Dan Dac) Dengan Seven Segment Display," *J. Informanika*, vol. 5, no. 1, p. 27, 2019.

[13] Khairunnisa and Y. Indrasary, "Simulasi Akuisisi Data Sinyal Audio," *J. Simantec*, vol. 5, no. 2, pp. 76–84, 2016.