

Perancangan Aplikasi Kompresi File Audio dengan Algoritma Aritmetic Coding

Nurasyiah

STMIK Budi Darma Medan, Jl. Sisimangaraja No.338 Simpang Limun, Medan, Indonesia
nurasyiah1211@gmail.com

Abstrak. *Pertukaran informasi saat ini membutuhkan kecepatan dalam pengiriman informasi. Kecepatan pengiriman ini sangat bergantung kepada ukuran dari informasi tersebut. Salah satu solusi untuk masalah di atas adalah dengan melakukan pemampatan (kompresi). Ada banyak sekali metode kompresi data yang ada saat ini, namun pada skripsi ini akan dibahas prinsip kerja algoritma Arithmetic Coding dengan implementasi menggunakan Visual Basic 6.0. Analisis kinerja algoritma ini bertujuan untuk mengetahui performansi algoritma ini pada file audio berformat *.MP3 dan *.WAV. Pada sistem ini terdapat tahap kompresi dan dekompresi. Tahap kompresi bertujuan untuk memampatkan ukuran file audio, sedangkan tahap dekompresi bertujuan untuk mengembalikan ukuran file audio ke ukuran semula.*

Kata Kunci : *Perancangan, Kompresi, Dekompresi, WAV, MP3, arithmetic coding.*

Abstract. *Information exchange nowadays requires speed in sending information. The speed of this transmission depends on the size of the information. One solution to the above problem is compression. There are lots of data compression methods available today, but in this thesis we will discuss the working principles of the Arithmetic Coding algorithm with an implementation using Visual Basic 6.0. This algorithm performance analysis aims to determine the performance of this algorithm in *.MP3 and *.WAV audio files. In this system there are compression and decompression stages. The compression stage aims to compress the audio file size, while the decompression stage aims to restore the audio file size to its original size.*

Keyword : *Designing, Compression, Decompression, WAV, MP3, arithmetic coding*

PENDAHULUAN

Kecepatan pengiriman informasi dalam bentuk perpaduan teks, suara dan gambar secara nyata akan menjadi bagian utama dalam pertukaran informasi. Kecepatan pengiriman ini sangat bergantung kepada ukuran dari informasi tersebut. Salah satu solusi untuk masalah di atas adalah dengan melakukan pemampatan (kompresi) data teks, suara, dan citra sebelum ditransmisikan dan kemudian penerima akan merekonstruksinya kembali menjadi data aslinya (dekompresi)[1], [2].

Kompresi data merupakan suatu hal yang esensial. Teknik kompresi ini esensial karena ukuran dari data semakin lama semakin besar, tetapi belum optimal karena tidak didukung oleh perkembangan dari teknologi bandwidth (untuk kecepatan download data dari internet) yang seimbang. Sementara orang-orang pun menginginkan data dengan kualitas terbaik dan kuantitas (ukuran) yang minimum. Melihat masalah-masalah tadi, maka pemecahannya adalah maksimalisasi kompresi, yaitu mengurangi tempat yang digunakan oleh data yang dimampatkan[3], [4].

Dengan latar belakang ini terlihat masih diperlukannya perbandingan-perbandingan untuk menentukan algoritma mana yang sebaiknya dipakai dalam melakukan kompresi data khususnya untuk File audio yang umumnya berformat) *.mp3, dan *.wav, midi yang sangat sensitif akan kehilangan data.

DASAR TEORI

2.1 Teori Dasar Kompresi

Pada dasarnya data apapun sebenarnya adalah merupakan rangkaian bit 0 dan 1. Yang membedakan antara suatu data tertentu dengan data yang lain adalah ukuran dari rangkaian bit dan bagaimana 0 dan 1 itu ditempatkan dalam rangkaian bit tersebut. Misalnya data berupa audio dan video, dalam data audio suatu rangkaian bit tertentu mewakili satu nada, sedangkan dalam data video



suatu rangkaian bit mewakili suatu stream video, dimana dalam stream video terdapat image dalam satu pixel. Semakin kompleks suatu data, ukuran rangkaian bit yang diperlukan semakin panjang, dengan demikian ukuran keseluruhan data juga semakin besar.[5]

Dalam penyimpanan dan pengiriman data komputer, selain isi dari data tersebut parameter yang tidak kalah pentingnya adalah ukurannya. Kerap kali data yang disimpan dalam suatu media penyimpanan berukuran sangat besar sehingga memerlukan tempat yang lebih banyak dan tidak efisien. Apalagi bila data tersebut akan dikirim, semakin besar ukurannya, waktu yang diperlukan untuk pengiriman akan lebih lama. Untuk itu, diperlukan kompresi data (data compression) untuk memperkecil ukuran suatu data tanpa merubah isi atau informasi yang terkandung dalam data tersebut.[6]

2.2 Kompresi Data

Kompresi data adalah proses mengkodekan informasi menggunakan bit atau information-bearing unit yang lain yang lebih rendah dari pada representasi data yang tidak terkodekan dengan suatu sistem encoding tertentu. Menurut Ida Kompresi data adalah ilmu atau seni merepresentasikan informasi dalam bentuk yang lebih compact[7].

Data kompresi adalah proses mengkompersikan sebuah input data stream (stream sumber, atau data mentah asli) menjadi data stream lainnya (bitstream hasil, atau stream yang telah terkompresi) yang berukuran lebih kecil.

Dalam ilmu komputer dan teori informasi, kompresi data atau source coding adalah proses meng-encode informasi dengan menggunakan lebih sedikit bit dari suatu sumber yang belum di-encode melalui penggunaan skema pengkodean yang spesifik .

Tujuan dari kompresi data adalah untuk merepresentasikan suatu data digital dengan sesedikit mungkin bit, tetapi tetap mempertahankan kebutuhan minimum untuk membentuk kembali data aslinya. Data digital ini dapat berupa text, gambar, suara, dan kombinasi dari ketiganya, seperti video. Contoh kompresi sederhana yang biasa kita lakukan misalnya adalah menyingkat kata-kata yang sering digunakan tapi sudah memiliki konvensi umum. Misalnya: kata “yang” dikompres menjadi kata “yg”.

Data digital yang telah dikompresi dapat dikembalikan ke bentuk data digital semula (dekompresi) dimana hal ini tergantung pada aplikasi software yang mendukung kompresi tersebut. Ketika suatu aplikasi mampu ‘menghilangkan’ atau mengkompresi data yang tidak dibutuhkan maka aplikasi tersebut juga mampu mengembalikan data yang dihilangkan tersebut sehingga menjadi data digital semula (dekompresi) namun terdapat juga suatu aplikasi yang dapat mengkompresi namun ketika dekompresi dapat menggunakan aplikasi lain contohnya aplikasi winzip dengan aplikasi winrar.

Kompresi data sangat populer sekarang ini karena dua alasan yaitu

1. Orang-orang lebih suka mengumpulkan data. Tidak peduli seberapa besar media penyimpanan yang dimilikinya. Akan tetapi cepat atau lambat akan terjadi overflow.
2. Orang-orang benci menunggu waktu yang lama untuk memindahkan data. Misalnya ketika duduk di depan komputer untuk menunggu halaman Web terbuka atau men-download sebuah file .

Rasio kompresi data adalah ukuran persentase data yang telah berhasil dimampatkan. Secara matematis rasio pemampatan data ditulis sebagai berikut :

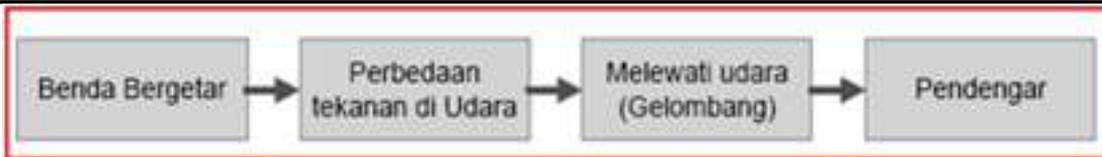
$$\text{Rasio kompresi} = \left(\frac{\text{ukuran file asli} - \text{ukuran file terkompresi}}{\text{ukuran file asli}} \times 100 \% \right)$$

Misalkan rasio kompresi adalah 25%, artinya 25 % data semula telah berhasil dimampatkan

2.3 File Audio

Audio (suara) adalah fenomena fisik yang dihasilkan oleh getaran suatu benda yang berupa sinyal analog dengan amplitudo yang berubah secara kontinyu terhadap waktu yang disebut frekuensi[8].



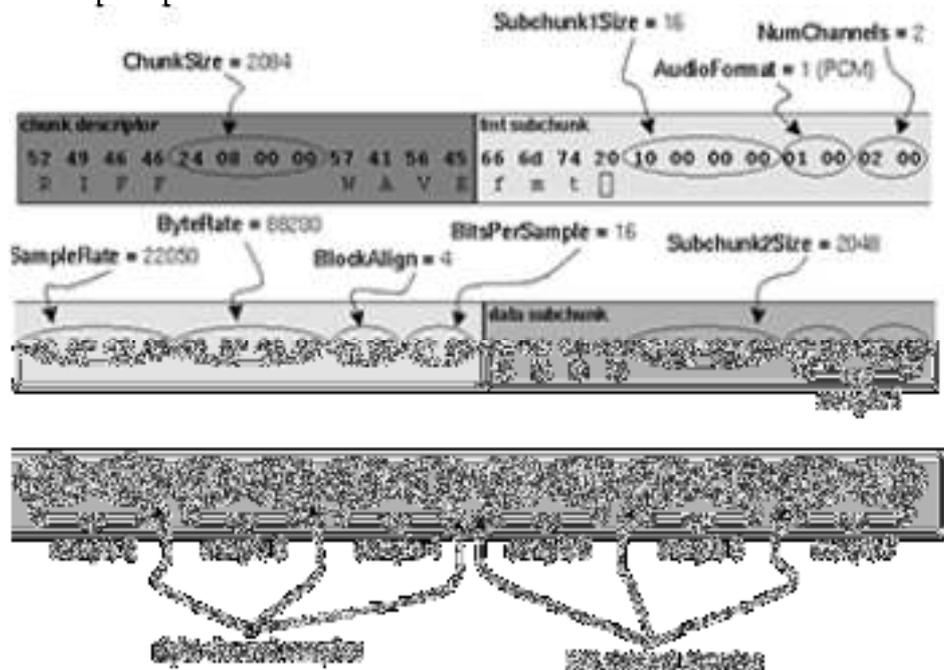


Gambar 1. Fenomena Fisik Audio[8]

Selama bergetar, perbedaan tekanan terjadi di udara sekitarnya. Pola osilasi yang terjadi dinamakan sebagai gelombang. Gelombang mempunyai pola sama yang berulang pada interval tertentu, yang disebut sebagai periode. Contoh suara periodik adalah instrumen musik, nyanyian burung sedangkan contoh suara non periodik adalah batuk, percikan ombak dan lain-lain.

2.4. Struktur Data pada FileAudio

Struktur data pada file audio berbeda-beda tergantung format audio-nya. Misalnya file Wav memiliki struktur seperti pada Gambar 2.



Gambar 2. Struktur File WAVE Dalam Bentuk Hexawavw[9]

Pada struktur file Wav di atas terdiri dari:

1. Chunk Descriptor yang terdiri dari data:
52 49 46 46 28 08 00 00 57 41 56 45.
2. Fmt subChunk yang terdiri data subChunk1size, audioFormat, numChannel, sampleRate, byteRate dan BlockAlign yaitu:
66 6d 74 20 10 00 00 00 01 00 02 00 22 56 00 00 ## 50 01 00 04 00 10 00
3. Data subChunk yang terdiri dari data subChunk2size serta sample-sample yaitu:
64 61 74 61 00 00 #1 00 00 00 00 #2 24 17 1e f3 #3 3c 13 3c 14 #4 16 f9 18 f9 34 e7 23 a6 3c f2 24 f2 24 f2 11 ce 1a 0d

2.5 File WAV

WAV adalah format audio standar Microsoft dan IBM untuk personal computer (PC), biasanya menggunakan coding PCM (Pulse Code Modulation). WAV adalah data tidak terkompres sehingga seluruh sampel audio disimpan semuanya di harddisk. Software yang dapat menciptakan WAV dari analog sound misalnya adalah Windows Sound Recorder[8]. File audio ini jarang sekali digunakan

di internet karena ukurannya yang relatif besar dengan batasan maksimal untuk file WAV adalah 2GB.

Secara umum data audio digital memiliki karakteristik yang dapat dinyatakan dengan parameter-parameter berikut:

1. Laju sampel (sampling rate) dalam sampel/detik, misalnya 22050 atau 44100 sampel/detik.
2. Jumlah bit tiap sampel, misalnya 8 atau 16 bit.
3. Jumlah kanal, yaitu 1 untuk mono dan 2 untuk stereo.

Parameter-parameter tersebut menyatakan setting yang digunakan oleh ADC (Analog-to-Digital Converter) pada saat data audio direkam. Biasanya laju sampel juga dinyatakan dengan satuan Hz atau kHz. Sebagai gambaran, data audio digital yang tersimpan dalam CD audio memiliki karakteristik laju sampel 44100 Hz, 16 bit per sampel, dan 2 kanal (stereo), yang berarti setiap satu detik suara tersusun dari 44100 sampel, dan setiap sampel tersimpan dalam data sebesar 16-bit atau 2 byte. Laju sampel selalu dinyatakan untuk setiap satu kanal. Jadi misalkan suatu data audio digital memiliki 2 kanal dengan laju sampel 8000 sampel/detik, maka sesungguhnya di dalam setiap detiknya akan terdapat 16000 sampel.

Sebagaimana telah dijelaskan sebelumnya bahwa untuk stream data audio menggunakan header berupa struktur PCMWAVEFORMAT. PCM merupakan singkatan dari Pulse Coded Modulation, yaitu suatu metode yang digunakan untuk mengkonversikan sinyal audio dari bentuk analog ke bentuk digital.

Adapun struktur dari PCMWAVEFORMAT adalah sebagai berikut: `typedef struct {WAVEFORMAT wf;WORD wBitsPerSample;} PCMWAVEFORMAT;`

Field wf merupakan struktur WAVEFORMAT yang berisi keterangan umum mengenai format data audio. Field wBitsPerSample menunjukkan banyaknya bit per sample.

Adapun struktur data dari WAVEFORMAT adalah sebagai berikut:

```
typedef struct {WORD wFormatTag;
    WORD nChannels;
    DWORD nSamplesPerSec;
    DWORD nAvgBytesPerSec;
    WORD nBlockAlign;}
WAVEFORMAT;
```

WAVEFORMAT;

Keterangan mengenai field-field dari struktur WAVEFORMAT ini dapat dilihat pada Tabel 1.

Tabel 1. Field-field pada struktur data. WAVEFORMAT

wFomatTag	Menunjukkan type format data dan memiliki nilai WAVE_FORMAT_PCM.
nChannels	Menunjukkan banyaknya kanal yang ada di dalam data waveform audio. Untuk mono menggunakan satu kanal sedangkan untuk stereo menggunakan dua kanal.
nSamplesPerSec	Menunjukkan besarnya sample rate dalam sample per detik.
nAvgBytesPerSec	Menunjukkan rata-rata laju transfer data, dalam byte per detik. Misalnya untuk PCM 16-bit stereo pada 44.1 kHz, nAvgBytesPerSec bernilai 176400 (2 kanal * 2 byte per sample * 44100).
nBlockAlign	Menunjukkan banyaknya byte yang digunakan untuk satu buah sample. Misalnya untuk PCM 16-bit mono, nBlockAlign akan bernilai 2.

2.6 Arithmetic Coding

Arithmetic coding memiliki sejarah yang sangat penting karena pada saat itu algoritma ini sukses menggantikan Huffman coding selama 25 tahun. Arithmetic coding adalah suatu bagian dari entropy encoding yang mengkonversi suatu data ke dalam bentuk data yang lain dengan lebih sering [10],



[11] menggunakan sedikit bit dan jarang menggunakan lebih banyak bit karakter. Teknik pengkodean ini memisahkan pesan masukan ke dalam simbol dan menukar masing-masing simbol dengan suatu floating-point. arithmetic coding mengkodekan seluruh pesan ke dalam suatu bilangan pecahan n di mana ($0.0 = n < 1.0$).

Algoritma arithmetic coding melakukan penggantian satu deretan simbol input dengan sebuah bilangan floating point. Semakin panjang dan semakin kompleks pesan yang dikodekan, akan semakin banyak bit yang diperlukan untuk keperluan tersebut. suatu urutan simbol ke dalam sebuah bilangan yang unik dengan interval (0,1) notasi (0,1) memiliki arti bahwa semua bilangan real dari 0 sampai 1, termasuk 0 tetapi tidak termasuk 1 atau ditulis dengan $0 = x < 1$. Angka ini secara unik dapat dapat didekoding sehingga menghasilkan deretan simbol yang dipakai untuk menghasilkan angka tersebut.

Satu hal yang perlu dicatat dari tabel ini adalah tiap karakter melingkupi range yang disebutkan, kecuali bilangan yang tinggi. Jadi, simbol "3e" sesungguhnya mempunyai range mulai dari 0,8 sampai dengan 0,9999... Selanjutnya untuk melakukan proses encoding algoritma berikut dipakai[12].

1. Set low = 0,0 (kondisi awal)
2. Set high = 1.0 (kondisi awal)
3. While (simbol input masih ada) do
4. Ambil simbol input.
5. CR = high – low.
6. High = low + CR*high_range(simbol)
7. Low = low + CR*low_range(simbol)
8. End while
9. Cetak low

Pada algoritma di atas low, high dan CR berturut-turut menyatakan batas bawah, batas atas dan jangkauan interval dari setiap simbol (character). Pada tahap awal low dan high di inialisasi dengan nilai 0 dan 1 kemudian pada proses selanjutnya kedua nilai tersebut diperbaharui dengan rumus :

1. High = low + CR*high_range(simbol)
2. Low = low + CR*low_range(simbol)

HASIL DAN PEMBAHASAN

3.1 Analisa File Audio

Sebelum *file audio* dikompresi, terlebih dahulu dilakukan pembacaan *file audio* untuk mendapatkan data berupa *header* dalam ukuran *byte* (8 bit) bentuk pasangan bilangan heksa desimal.

3.2 Langkah-Langkah Kompresi Arithmetic Coding

Secara umum langkah-langkah yang dilakukan untuk kompresi *file audio* dengan metode *Arithmetic Coding* adalah sebagai berikut :

1. Buka *file audio* untuk membaca *header-header* dan *sample audio*.
2. Baca *file audio* untuk mendapatkan data *sample*.
3. Ambil nilai *sample audio* ke 1 sampai ke n .
4. Susun nilai *sample audio* pada senarai berantai dengan karakter khusus pembatas antara data *sample audio* (#).

Dari data hasil pembacaan *sample audio* diperoleh data mulai dari *sample* ke 1 dengan tanda #1 sampai *sample* ke 12 dengan tanda #12 pada blok terakhir data *sample audio*. Asumsikan nilai *chunk audio* nilai *sample 1 audio* (#1) di atas yang akan di-*encoding* adalah :-1, 10, -1, 0, 07, 3d, 0, -1, 0, 07, 07, -1, -1, 3d dan 0.

Dari data yang akan di-*encoding* di atas, maka akan dapat dibuat sebuah tabel probabilitas seperti Tabel .

Tabel 2. Tabel Probabilitas



No	Nilai	Frekuensi	Probabilitas
1	-1	5	5/15=0,33
2	10	1	1/15=0,06
3	07	3	3/15=0,20
4	3d	2	2/15=0,13
5	0	4	4/15=0,26

Selanjutnya akan diperoleh tabel *range* probabilitas seperti Tabel 3.2.

Tabel 3. Tabel *Range* Probabilitas

No	Nilai	Frekuensi	Probabilitas	Range
1	-1	5	5/15=0,33	$0,0 \leq -1 < 0,33$
2	10	1	1/15=0,06	$0,33 \leq 10 < 0,39$
3	07	3	3/15=0,20	$0,39 \leq 07 < 0,59$
4	3d	2	2/15=0,13	$0,59 \leq 3d < 0,72$
5	0	4	4/15=0,26	$0,72 \leq 0 < 0,98$

Keterangan :

1. $0,0 = -1 < 0,33$: Nilai “-1” memiliki *range* dari 0,0 sampai dengan 0,33
2. $0,33 = 10 < 0,39$: Nilai “10” memiliki *range* dari 0,33 sampai dengan 0,39
3. $0,39 = 07 < 0,59$: Nilai “07” memiliki *range* dari 0,39 sampai dengan 0,59
4. $0,59 = 3d < 0,72$: Nilai “3d” memiliki *range* dari 0,59 sampai dengan 0,72
5. $0,72 = 0 < 0,98$: Nilai “0” memiliki *range* dari 0,72 sampai dengan 0,98

Dalam bentuk tabel data hasil *encoding* dapat dilihat seperti pada Tabel .

Tabel 4. Hasil *Encoding* Sampel *Audio*

No	Nilai	Low	High	CR
Awal		0	1	1
1	-1	0	0,33	1
2	10	0,1089	0,1287	0,33
3	07	0,116622	0,120582	0,0198
4	3d	0,1189584	0,1194732	0,00396
5	0	0,119329056	0,119462904	0,0005148

Dari proses ini, nilai *low* untuk data terakhir adalah :

nilai *low* = 0,119329056 yang akan digunakan untuk menggantikan sample *audio* yang telah di-*encoding* yaitu nilai *sample audio* -1, 10, -1, 0, 07, 3d, 0, -1, 0, 07, 07, -1, -1, 3d dan 0. Selanjutnya *sample* 1 adalah #1 -1 10 -1 0 07 3d 0 -1 0 07 07 -1 -1 3d 0 berubah menjadi #1 0,119329056 Selanjutnya untuk *sample* 2 gantikan dengan nilai *low* menjadi #2 0,xxxxxxxxx dan selanjutnya.

3.3 Langkah-Langkah Dekompresi *Arithmetic Coding*

Untuk melakukan melakukan dekomposisi *file audio*, maka dilakukan proses *decoding*, dengan cara :

1. Ambil *encoded symbol* (ES).
2. Repeat.
3. Cari *range* dari simbol yang melingkupi *encoded symbol* (ES)
4. Cetak simbol.



5. $CodeRange \leftarrow high_range - low_range.$
6. $ES = ES - low_range.$
7. $ES = ES/CodeRange.$
8. *Until simbol habis.*

Dalam hal ini simbol habis dapat ditandai dengan dengan simbol khusus yang dalam penelitian ini digunakan tanda #. Untuk pesan yang telah di-encode, proses decoding berikut dilakukan.

$$ES = 0,119329056$$

Bandingkan nilai ini dengan range simbol berikut dengan Tabel Range Probabilitas berikut ini :

Tabel 5. Range Probabilitas

No	Nilai	Frekuensi	Probabilitas	Range
1	-1	5	5/15=0,33	$0,0 \leq -1 < 0,33$
2	10	1	1/15=0,06	$0,33 \leq 10 < 0,39$
3	07	3	3/15=0,20	$0,39 \leq 07 < 0,59$
4	3d	2	2/15=0,13	$0,59 \leq 3d < 0,72$
5	0	4	4/15=0,26	$0,72 \leq 0 < 0,98$

Sampai disini perhitungan dihentikan karena diperoleh nilai $ES = 0$. Hasil perhitungan di atas dapat dilihat pada tabel 3.5.

Tabel 6. Hasil Decoding Sampel Audio

No	ES	Nilai	Low	High	CR
1	0,3616032	-1	0	0,33	0,33
2	0,52672	10	0,33	0,39	0,06
3	0,6836	07	0,39	0,59	0,2
4	0,72	3d	0,59	0,72	0,13
5	0 (Finish)	0	-	-	-

Keterangan :

1. Maka diperoleh $ES = 0,3616032$
2. Yang sesuai dengan nilai *sample #1 audio* “-1, 10, 1, 0, 07, 3d, 0, -1, 0, 07, 07, -1, -1, 3d dan 0”

Langkah pertama yang dilakukan adalah dengan melakukan pemilihan mode Kompresi pada Menu Utama dan hasilnya dapat dilihat pada Gambar 3.





Gambar 3. Tampilan Menu Utama

Selanjutnya dilakukan penambahan *file* yang akan dikompresi dengan menekan tombol *Browse* untuk pemilihan *file* audio yang akan dikompresi serta pemilihan folder tujuan untuk *file* terkompresi. Bentuk tampilan dari pengujian ini seperti terlihat pada Gambar 4.



Gambar 4. Tampilan Menu Hasil Kompresi

Apabila file terkompresi akan didekompresi maka langkah yang dilakukan adalah dengan melakukan pemilihan mode dekompresi dan hasilnya dapat dilihat pada Gambar 5.



Gambar 5. Tampilan Menu Dekompresi

Selanjutnya dilakukan penambahan *file* yang akan didekompresi dengan memilih tombol *Browse* untuk pemilihan *load file* audio yang akan didekompresi serta pemilihan folder tujuan untuk *file* audio asli. Bentuk tampilan dari pengujian ini seperti terlihat pada Gambar 6.



Gambar 6. Tampilan Menu Hasil Dekompresi

KESIMPULAN

Setelah mendapatkan hasil tampilan perangkat lunak, tahap selanjutnya penulis melakukan pengujian terhadap sistem tersebut. Adapun metode pengujian sistem yang penulis lakukan adalah metode statis (*static technique*) di mana pengujian dibagi dalam beberapa tahapan.

1. Menetapkan Parameter Pengujian



Adapun parameter pengujian yang penulis gunakan dalam pengujian sistem ini adalah sebagai berikut:

a. Kestabilan Sistem

Parameter ini digunakan untuk menguji apakah sistem masih mengalami *error* pada saat dieksekusi atau pada saat melakukan penghapusan *file* dalam daftar *file* yang akan dihapus.

b. Ketepatan Hasil

Parameter ini digunakan untuk menguji apakah sistem telah dapat bekerja seperti apa yang diharapkan dalam perancangan.

2. Melakukan Pengujian Kompresi

Dalam tahap ini, penulis melakukan pengujian terhadap kemampuan sistem dalam melakukan kompresi pada *file* audio serta mengamati hasil kompresi yang dihasilkan.

DAFTAR PUSTAKA

- [1] Y. F. Amri, "Kompresi File Audio," *Universitas Sebelas Maret*, 2012.
- [2] W. Zarman and T. Pamungkas, "IMPLEMENTASI ALGORITMA KOMPRESI LZW PADA DATABASE SERVER," *Jurnal Ilmiah Komputer dan Informatika (KOMPUTA)*, vol. 7, no. 1, 2013.
- [3] S. I. Murpratiwi and I. M. O. Widyantara, "Pemilihan Algoritma Kompresi Optimal untuk Citra Digital Bitmap," *Majalah Ilmiah Teknologi Elektro*, vol. 17, no. 1, p. 94, 2018, doi: 10.24843/MITE.2018.v17i01.P13.
- [4] B. D. Raharja and P. Harsadi, "Implementasi Kompresi Citra Digital Dengan Mengatur Kualitas Citra Digital," *Jurnal Ilmiah SINUS*, vol. 16, no. 2, 2018, doi: 10.30646/sinus.v16i2.363.
- [5] N. J. Tuturoong, "PERBANDINGAN RASIO DAN KECEPATAN KOMPRESI MENGGUNAKAN ALGORITMA HUFFMAN, LZW DAN DMC," *TEKNO*, 2010. <https://ejournal.unsrat.ac.id/index.php/tekno/article/view/4320> (accessed Jan. 22, 2020).
- [6] V. Lusiana, "Teknik Kompresi Citra Digital untuk Penyimpanan File menggunakan Format Data XML," *Teknologi Informasi DINAMIK*, vol. 19, no. 2, pp. 112–119, 2014.
- [7] E. Kristian Gulo, "PERANCANGAN APLIKASI KOMPRESI AUDIO DENGAN MENERAPKAN ALGORITMA GOLOMB," 2017.
- [8] F. Fatmawaty and M. Mufty, "Analisis Perbandingan Kompresi File Wav Menggunakan Metode Huffman dan Run Length Encoding," *Jurnal Teknologi Informasi dan Terapan*, vol. 7, no. 1, pp. 61–65, Jun. 2020, doi: 10.25047/jtit.v7i1.139.
- [9] H. Santoso and M. Fakhriza, "PERANCANGAN APLIKASI KEAMANAN FILE AUDIO FORMAT WAV (WAVEFORM) MENGGUNAKAN ALGORITMA RSA," 2018.
- [10] P. Santoso, "Studi Kompresi Data dengan Metode Arithmetic Coding," *Jurnal Teknik Elektro*, Dec. 2004. <https://ojs.petra.ac.id/ojsnew/index.php/elk/article/view/16394> (accessed May 29, 2021).
- [11] J. T. Elektro and P. Santoso, "Studi Kompresi Data dengan Metode Arithmetic Coding," 2001. [Online]. Available: <http://puslit.petra.ac.id/journals/electrical/>
- [12] "An Introduction to Arithmetic Coding," 1984.

