semanTIK, Vol.2, No.1, Jan-Jun 2016, pp. 29-38
ISSN : 2502-8928 (Online)

APLIKASI KOMPRESI *FILE* AUDIO MENGGUNAKAN ALGORITMA *ARITHMETIC CODING*

Annisa Diah Mutiara[*1], Sutardi[2], Rahmat Ramadhan[3]
[*1,2,3]Jurusan Teknik Informatika, Fakultas Teknik, Universitas Halu Oleo, Kendari
e-mail : ***[1]nisaadm@gmail.com**, [2]sutardi_hapal@yahoo.com, [3]rahmat.ramadhan@innov-center.org

Abstrak

Proses kompresi data didasarkan pada kenyataan bahwa pada hampir semua jenis data selalu terdapat pengulangan pada komponen data yang dimilikinya, misalnya dalam suatu teks kalimat akan terdapat redudansi huruf alfabet dari huruf *a* sampai dengan huruf *z*. Kompresi data melalui proses *encoding* berusaha untuk menghilangkan unsur pengulangan ini dengan mengubahnya sedemikian rupa sehingga ukuran data menjadi lebih kecil. Dalam penelitian ini, dibangun sebuah aplikasi kompresi *file* audio dengan menggunakan algoritma *Arithmetic Coding*. *Tool* pengembangan sistem yang digunakan adalah *Unified Modelling Language* (UML). Dalam pembuatan aplikasi ini ditekankan untuk melakukan kompresi pada *file* audio berekstensi *.wav .

Pengujian aplikasi kompresi ini dilakukan dengan menguji *file* yang mempunyai ukuran berbeda-beda dengan nilai sampel *rate* dan *bitrate* yang sama. Disimpulkan bahwa *file* audio dengan ukuran yang besar membutuhkan waktu lebih lama dalam melakukan proses kompresi. *File* audio hasil kompresi tidak dapat dijalankan sebelum melewati proses dekompresi hal tersebut disebabkan terjadi perubahan struktur pada *file* audio saat proses dekompresi.

Kata kunci—Kompresi, *File* Audio, *Arithmetic Coding*

Abstract

*The process of data compression based on the fact that in almost all types of data are always there repetition in the data component has, for example, in a text sentence there will be redundancies letter alphabet of letters a through z. Compression of data through the encoding process seeks to eliminate the element of repetition by changing it such that the data size becomes smaller. In this study, an audio file compression applications is making by using Arithmetic Coding algorithm. Unified Modelling Language (UML) is used as system development tools. In making this application is emphasized to compress the audio file extension * .wav.*

Compression application testing is done by examining the files that have different sizes with a sample rate and bitrate values are the same. It was concluded that the audio file with a large size takes longer to perform the compression process. The compressed audio files can not be executed before passing the decompression process that caused a change in the structure of the current audio file decompression process.

Keywords—*Compression, File Audio, Arithmetic Coding*

1. PENDAHULUAN

Proses kompresi data didasarkan pada kenyataan bahwa pada hampir semua jenis data selalu terdapat pengulangan pada komponen data yang dimilikinya, misalnya didalam suatu teks kalimat akan terdapat redudansi huruf alfabet dari huruf *a* sampai dengan huruf *z*. Kompresi data melalui proses *encoding* berusaha untuk menghilangkan unsur pengulangan ini dengan mengubahnya sedemikian rupa sehingga ukuran data menjadi lebih kecil [1].

Format Waveform Audio atau WAV adalah format *file* audio yang tidak terkompresi sehingga seluruh sampel audio

disimpan semuanya di dalam media penyimpanan dalam bentuk digital. Ukuranya yang besar menjadi salah satu penyebab *file* WAV jarang digunakan sebagai *file* audio internet.

Pada umumnya, algoritma kompresi data didasarkan pada pemilihan cara melakukan penggantian satu atau lebih elemen-elemen yang sama dengan kode tertentu. Berbeda dengan cara tersebut, *Arithmetic Coding* menggantikan suatu deret simbol *input* dengan suatu *file* data dengan sebuah bilangan menggunakan proses *Arithmetic Coding*. Semakin panjang dan semakin kompleks pesan yang dikodekan, semakin banyak bit yang diperlukan untuk proses kompresi dan dekompresi data.

Dalam penelitian ini akan diimplementasikan algoritma kompresi, yaitu algoritma *Arithmetic Coding*. Algoritma ini diujikankan untuk mengkompresi *file* audio dengan ekstensi WAV, karena ekstensi tersebut merupakan bentuk audio yang belum terkompresi sebelumnya.

Penelitian yang dilakukan oleh [2], mengenai kompresi data dengan judul Perancangan Aplikasi Kompresi *File* Audio dengan Algoritma *Arithmetic Coding* menyimpulkan bahwa pada sistem terdapat tahap kompresi dan dekompresi. Tahap kompresi bertujuan untuk memampatkan ukuran *file* audio, sedangkan tahap dekompresi bertujuan untuk mengemblikan ukuran *file* audio ke ukuran semula.

Selain itu, penelitian yang dilakukan oleh [3], dengan judul penelitian Pengkodean Aritmatika untuk Kompresi Data Teks menyatakan bahwa rasio kompresi yang tertinggi diperoleh pada saat suatu teks hanya memiliki satu karakter saja, adapun pengaruh variasi karakter terhdap rasio kompresi adalah semakin banyak variasi karakter yang dimiliki oleh suatu teks maka rasio kompresi akan semakin rendah.

Serta penelitian oleh [4], mengenai Analisis Perbandingan Kompresi Data dengan Teknik *Arithmetic Coding* dan *Run Length Encoding* menyimpulkan bahwa teknik kompresi dan dekompresi *file* dokumen menggunakan teknik *Arithmetic Coding* memiliki kelebihan dalam hal rasio kompresi tetapi membutuhkan waktu lebih lama, sedangkan dengan menggunakan teknik *Run Length Encoding* membutuhkan waktu yang lebih cepat untuk proses kompresi dan dekompresi dibandingkan teknik *Arithmetic Coding*.

Penelitian selanjutnya oleh [1], mengenai Analisis dan Implementasi Kompresi *File Audio* dengan Menggunakan Algoritma *Run Length Encoding* (RLE) menyimpulkan dalam proses pemampatan data terdiri dari 2 tahap yaitu tahap kompresi dan dekompresi. Tahap kompresi bertujuan untuk memampatkan ukuran *file audio*, sedangkan tahap dekompresi bertujuan untuk mengembalikan ukuran *file* audio ke ukuran semula.

2. METODE PENELITIAN

2.1 Audio

Audio adalah fenomena fisik yang dihasilkan oleh getaran suatu benda yang berupa sinyal analog dengan amplitudo yang berubah secara kontinyu terhadap waktu yang disebut frekuensi. 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 [2].

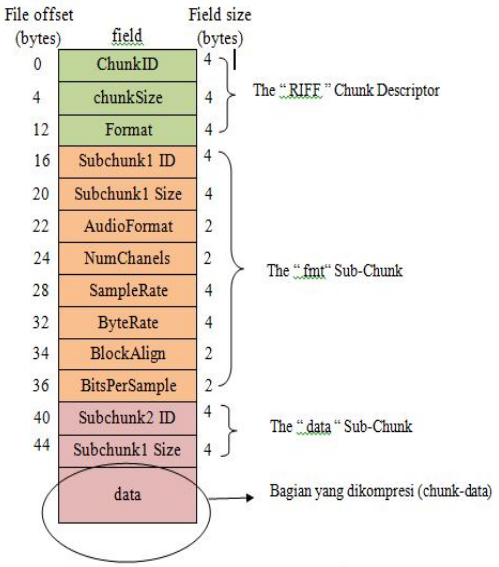
Sebuah format *file* audio adalah format *file* untuk menyimpan data audio digital pada sistem komputer. Data ini dapat disimpan tidak dikompresi, atau kompresi untuk mengurangi ukuran *file*.

2.2 WAVeform audio (*.wav)

WAV adalah singkatan dari istilah dalam bahasa Inggris *WAVeform audio* format merupakan standar format *file* audio yang dikembangkan oleh Microsoft dan IBM. WAV merupakan standar format *container file* yang digunakan oleh windows. WAV umumnya digunakan untuk menyimpan audio tak termampatkan (terkompresi), *file* suara berkualitas CD, yang berukuran besar (sekitar 10 MB per menit). *File* WAV juga dapat berisi data terkodekan dengan beraneka ragam *codec* untuk mengurangi ukuran *file*.

File WAV menggunakan struktur standar RIFF dengan mengelompokan isi *file* ke dalam bagian-bagian seperti format WAV dan data digital audio. Setiap bagian memiliki *header*-nya sendiri-sendiri beserta dengan ukurannya. Struktur RIFF (*Resource Interchange File Format*) ini merupakan struktur yang biasa digunakan untuk data

multimedia dalam *Windows*. Struktur ini mengatur data dalam *file* ke dalam bagian-bagian yang masing-masing memiliki *header* dan ukurannya sendiri dan disebut sebagai *chunk*. Struktur ini memungkinkan bagi program bila tidak mengenali bagian tertentu untuk melompati bagian tersebut dan terus memproses bagian yang dikenal. Data dari suatu bagian bisa memiliki sub-bagian dan seluruh data dalam *file* berstruktur RIFF selalu merupakan sub bagian dari suatu bagian yang memiliki *header* "RIFF". Gambar 1 menunjukan struktur *file* audio wav [5].



Gambar 1 Strukur *file* *.wav [5]

The "RIFF" Chunk descriptor menunjukan *file* format wav yang terdiri dari 2 *subChunk* yaitu "fmt" dan "data". *ChunkID* menunjukan terdiri dari kata "RIFF" dalam bentuk ASCII, Chunk *size* menunjukan besarnya ukuran *file* dalam byte dikurangi 8 *byte* untuk 2 *field* yang tidAk termasuK dalam hitungan, yaitu *ChunkID* dan *Chunk size*, serta format terdiri dari kata "wav" dalam bentuk ASCII.

The "fmt" sub-Chunk menggambarkan format informasi suara di dalam *subChunk*. *SubCunk1ID* terdiri dari kata "fmt", *subChunk1size* ukuranya sebesar data format wav "16 *byte*" ditambah dengan ekstra format byte yang diperlukan untuk format wav khusus, *Audio Format* bernilai pcm = 1(*linear quantitation*) jika nilai lebih dari 1 mengindikasikan *file* wav kompresi, *NumChanel* menunjukan jumlah chanel yang digunakan dalam *file*, Sampel *rate* menunjukan *jumlah* sampel dalam *file* per detik. *Bit rate* mengindikasikan berapa besar *bit* data wav harus di-*stream* ke converter digital audio tiap detik sewaktu *file* dimainkan, *Block align* menunjukan jumlah *byte* tiap potong sampel dan *Bit per* sampel menunjukan jumlah bit yang digunakan untuk mendefinisikan tiap sampel.

The "data" SubChunk2ID berisi ukuran dari informasi suara dan terdiri dari data suara mentah. *SubChunk2ID* terdiri dari kata "data", *SubChunk2Size* menunjukan besar ukuran data dan data berisi data audio sebenarnya [5].

2.3 Kompresi data

Kompresi adalah pengubahan data ke dalam bentuk yang memerlukan *bit* yang Íebih sedikit, biasanya dilakukan agar data dapat disimpan atau dikirimkan dengan lebih efisien. Kebalikan dan proses kompresi, yaitu dekompresi. Dekompresi merupakan proses untuk mengembalikan data baru yang telah dihasilkan oleh proses kompresi menjadi data awal. Dekompresi yang menghasilkan data sama persis dengan data aslinya sebelum dikompresi, maka kompresi tersebut disebut *lossless compression*. Sebaliknya, jika hasil dekompresi tersebut tidak sama dengan data aslinya sebelum dikompresi, karena ada data yang dihilangkan karena dirasa tidak terlalu penting tetapi tidak mengubah informasi yang dikandungnya, maka disebut *loosy compression* [6].

Teknik kompresi data terbagi menjadi dua, yaitu :

1. *Lossless* data kompresi adalah kelas dari algoritma data kompresi yang memungkinkan data yang asli dapat disusun kembali dari data kompresi. Lossless data kompresi digunakan dalam berbagai aplikasi seperti format ZIP dan GZIP. *Lossless* juga sering digunakan sebagai komponen dalam teknologi kompresi data *lossy*. Kompresi *lossless* digunakan ketika sesuatu yang penting pada kondisi asli [1].
2. *Lossy* kompresi adalah suatu metode untuk mengkompresi data dan men-dekompresinya, data yang diperoleh mungkin berbeda dari yang aslinya tetapi cukup dekat perbedaaanya. *Lossy* kompresi ini paling sering digunakan untuk kompres data multimedia (suara

atau gambar diam). Sebaliknya, kompresi *lossless* diperlukan untuk data teks dan *file*, seperti catatan bank, artikel teks dan lainnya.

Format kompresi *lossy* mengalami *generation loss* yaitu jika melakukan berulang kali kompresi dan dekompresi *file* akan menyebabkan kehilangan kualitas secara progresif. hal ini berbeda dengan kompresi data *lossless*. Pengguna yang menerima *file* terkompresi secara *lossy* (misalnya untuk mengurangi waktu *download*) *file* yang diambil dapat sedikit berbeda dari yang asli di-*level bit* ketika tidak dapat dibedakan oleh mata dan telinga manusia untuk tujuan paling praktis [1].

2.4 Algoritma 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 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 $(0.0 = n < 1.0)$.

Arithmetic Coding menggantikan satu deretan simbol *input* dengan sebuah bilangan *floating point*. Semakin panjang dan semakin kompleks pesan yang dikodekan, semakin banyak *bit* yang diperlukan untuk keperluan tersebut. *Output* dari *Arithmetic Coding* ini adalah satu angka yang lebih kecil dari 1 dan lebih besar atau sama dengan 0. Angka ini secara unik dapat di-*decode* sehingga menghasilkan deretan simbol yang dipakai untuk menghasilkan angka tersebut. Untuk menghasilkan angka *output* tersebut, tiap simbol yang akan di-*encode* diberi satu set nilai probabilitas [2]. Untuk menjelaskan proses *encoding* dipakai algoritma pada Gambar 2, *'Low'* adalah *output* dari proses *Arithmetic Coding*.

a. Kompresi

Struktur data pada *file* audio berbeda-beda tergantung format audionya, pada contoh berikut akan dikompresi *file* audio dengan ukuran 12 byte untuk nilai sampel1(010010100001000001001010000000000000011100111101000000000000000111000001110100101001001010000000000) dengan sampel *rate* 22050 dan *bit rate* 88200 yang kemudian diubah menjadi bilangan hexadecimal 4a, 10, 4a, 0, 07, 3d, 0, 4a, 0, 07, 07, 4a, 4a, 3d dan 0. Dari sample audio asumsikan nilai chunk audio dari nilai sampel 1 audio (#1) di atas yang datanya akan di-*encoding*.

```
- set low = 0.0
    - set high = 1.0
    - while (simbol input masih ada) do
        - ambil simbol input
        - cr = high – low
        - high = low + cr*high_range (simbol)
        - low = low + cr*low_range (simbol)
        - end while
    - cetak low
```

Gambar 2 Algoritma *Arithmetic Coding*

Langkah awal dari proses tersebut yaitu mencari probabilitas masing-masing data dengan cara membagi jumlah nilai 4a dengan jumlah keselurahan data yang ada yaitu 15. Maka akan dapat dibuat sebuah tabel probabilitas ditunjukan pada Tabel 1 dan persamaan probabilitas oleh Persamaan (1) [2].

$$P(x) = \frac{n(x)}{N} \quad (1)$$

Diketahui $P(x)$ adalah probabilitas data x, $n(x)$ adalah jumlah data x dan N adalah jumlah total data.

Tabel 1 Probabilitas frekuensi

No	Nilai	Frekuensi	Probabilitas
1	4a	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 dengan menjumlahkan nilai *range*

terendah (*low*) yang ditunjukkan oleh Persamaan (2) yaitu 0 ditambah 0,33 dilanjutan dengan 0,33 ditambah 0,06. Tabel 2 menunjukkan tabel *Range* Probabilitas [2].

$$I(x_i) = Low(x_i) + P(x_i)\ ;\ i = 1,2,..n \qquad (2)$$

Diketahui $I(x_i)$adalah *range* probabilitas data x_i, $Low(x_i)$ adalah *range* terendah data x_i dan $P(x_i)$ adalah probabilitas data x_i.

Tabel 2 *Range* probabilitas data sampel

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

Ilustrasi proses partisi pada garis peluang untuk nilai sampel audio 1 (#1) yaitu diawali dengan mencari nilai $cr(x_i)$ atau nilai jarak interval karakter kemudian menghitung nilai $high(x_{i+1})$ dan $low(x_{i+1})$. Data yang diproses adalah data yang *range* awalnya adalah 0. Data yang pertama diolah adalah -1 dengan nilai $low(x_1)$ = 0,0 dan $high(x_1)$ = 0,33. Setelah menentukan $cr(x_2) = high(x_1) - low(x_1) = 0{,}33$ kemudian menentukan nilai $high(x_2)$ dan $low(x_2)$, yaitu

$$low(x_2) = low(x_1) + cr(x_2) * low(x_2)$$
$$= 0{,}0 + 0{,}33 * 0{,}33 = 0{,}1089$$

$$high(x_2) = low(x_1) + cr(x_2) * high(x_2)$$
$$= 0{,}0 + 0{,}33 * 0{,}39 = 0{,}1287$$

Secara umum misalkan $i = 1,2,\dots,n$ maka Persamaan (3), (4) dan (5) berturut-turut menunjukkan penentuan $cr(x_i)$, $Low(x_i)$ dan $high(x_i)$.

$$cr(x_{i+1}) = High(x_i) - Low(x_i) \qquad (3)$$

$$Low(x_{i+1}) = Low(x_i) + cr(x_{i+1}) * Low(x_{i+1}) \qquad (4)$$

$$High(x_{i+1}) = Low(x_i) + cr(x_{i+1}) * high(x_{i+1}) \qquad (5)$$

Tabel 3 menunjukkan hasil *encoding* untuk data sampel.

Tabel 3 Hasil *encoding* data sampel

No Awal	Nilai	*Low*	*High*	Cr
		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	0,119329060	0,119470319	0,0005148
1	-1	0	0,33	1

Tabel 3 menunjukkan bahwa nilai *low* untuk data terakhir adalah nilai batas bawah = 0,119329060 dengan nilai biner 0,000111101 yang akan digunakan untuk menggantikan sampel audio yang telah di-*encoding* yaitu nilai sampel audio 4a, 10, 4a, 0, 07, 3d, 0, 4a, 0,07, 07, 4a, 4a, 3d dan 0. Selanjutnya sampel 1 adalah #1 4a 10 4a 0 07 3d 04a 0 07 07 4a 4a 3d 0 berubah menjadi #1 0,119329056 yang ketika dikonversi ke biner maka menjadi 0,000111101. Untuk sampel 2 gantikan dengan nilai batas bawah menjadi #2 0,xxxxxxxxx dan selanjutnya.

Pada proses kompresi terdapat perhitungan rasio kompresi. Rasio kompresi menunjukan presentase besarnya kompresi yang dilakukan terhadap *file* asli. Misalkan d = selisih ukuran *file* kompresi dan *file* asli dan u = ukuran *file* asli, maka rasio kompresi R ditunjukkan oleh Persamaan (6).

$$R = \frac{d}{u} \qquad (6)$$

b. Dekompresi

Proses kompresi *file* audio tidak bisa terlepas dari proses dekompresi, dimana *file* audio yang telah terkompresiharus dapat direkonstruksi kembali agar dapat dijalankan. Pada algoritma *Arithmetic Coding file* audio yang dikompresi harus dapat dikembalikan seperti *file* audio aslinya tanpa kehilangan informasi.

Proses dekompresi *file* audio pada algoritma *Arithmetic Coding* akan dilakukan proses *decoding* dengan cara :

1. Baca nilai *coding* terakhir.
2. Ambil *Encoded-Symbol*.

3. Cari *range* dari *symbol* yang melingkupi *ES*.
4. Cetak *symbol* hingga selesai.
5. Simpan *file* hasil dekompresi.

Proses dekompresi diawali dengan mengolah nilai *ES* = 0,119329060 yang dikurangi nilai *low* yaitu 0 kemudian dibagi dengan nilai 0,33 menghasilkan 0,3616032. Untuk melakukan *decoding* digunakan Persamaan (7).

$$Es_{i+1} = \frac{Es_i - Low_i}{Range} \; ; i = 1,2\,..\,n \qquad (7)$$

Tabel 4 menunjukan hasil *decoding* sampel audio.

Tabel 4 Hasil *decoding* sampel audio

No	*Encode Symbol* (ES)	Nilai	Batas bawah (*Low*)	Batas atas (*High*)	Jarak interval karakter (*cr*)
1.	0,3616032	-1	0	0,33	0,33
2.	0,5672	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	-	-	-

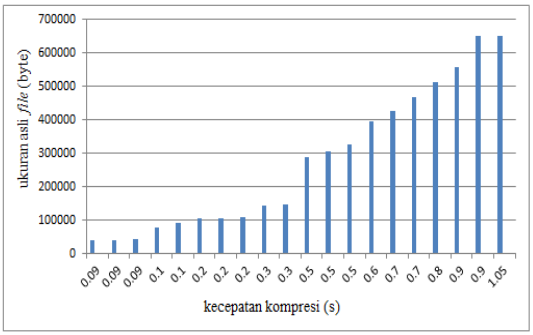
3. HASIL DAN PEMBAHASAN

Data yang digunakan sebagai bahan dalam penelitian ini merupakan 20 *file* audio yang berekstensi *.wav berklasifikasi stereo dengan nilai *sampling rate* dan *bit rate* 8000 Hz – 128 Kbps, 44.100 Hz - 1141 Kbps, 48.000 Hz – 1536 Kbps, 96.000 Hz – 3072 Kbps, yang untuk setiap pengujian yang digambarkan dalam bentuk grafik dengan membandingkan ukuran *file* asli dan kecepatan kompresi serta antara ukuran *file* asli dan rasio kompresi.

3.1 Perbandingan Ukuran *File* Asli dan Kecepatan kompresi

a. Sampling Rate 8.000 Hz dan Bit Rate 128 Kbps

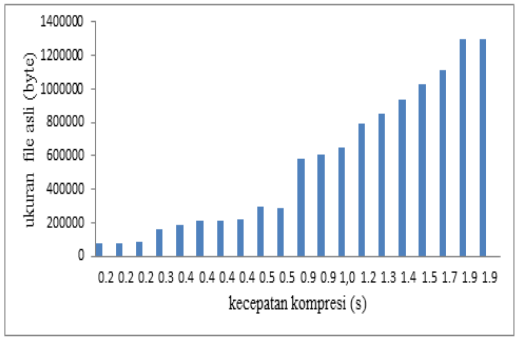
Gambar 3 menunjukan perbandingan antara ukuran *file* dan kecepatan kompresi. Tampak waktu yang dibutuhkan untuk proses kompresi semakin meningkat sesuai dengan semakin meningkatnya ukuran *file* yang di kompresi.



Gambar 3 Perbandingan ukuran *file* dan kecepatan kompresi untuk *sampling rate* 8.000 Hz dan *bit rate* 128 Kbps

b. Sampling Rate 8.000 Hz dan Bit Rate 256 Kbps

Gambar 4 menunjukan perbandingan antara ukuran *file* dan kecepatan kompresi. Sama seperti Gambar 3, proses kompresi semakin meningkat sesuai dengan semakin meningkatnya ukuran *file* yang dikompresi.



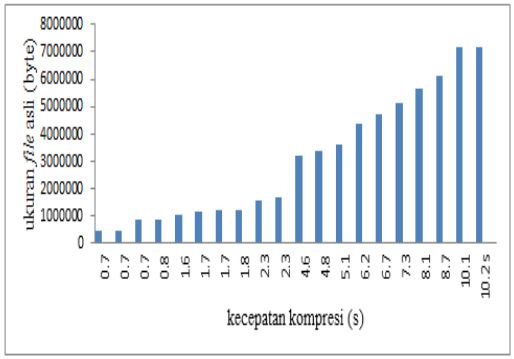
Gambar 4 Perbandingan ukuran *file* dan kecepatan kompresi untuk *sampling rate* 8.000 Hz dan *bit rate* 256 Kbps

c. Sampling Rate 44.100 Hz dan Bit Rate 1.141 Kbps

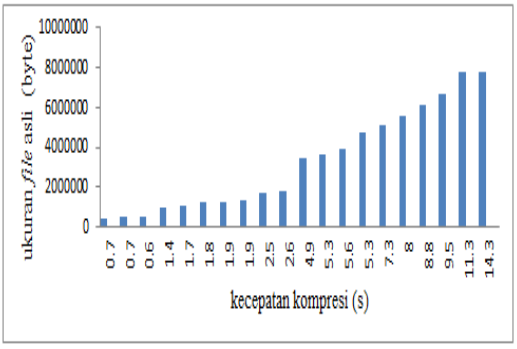
Gambar 5 menunjukan perbandingan antara ukuran *file* dan kecepatan kompresi. Tampak waktu yang dibutuhkan untuk proses kompresi semakin meningkat sesuai dengan semakin meningkatnya ukuran *file* yang di kompresi.

d. Sampling Rate 48.000 Hz dan Bit Rate 1.536 Kbps

Gambar 6 menunjukan perbandingan antara ukuran *file* dan kecepatan kompresi. Dapat dilihat pada Gambar 6 bahwa waktu yang dibutuhkan dalam proses kompresi semakin meningkat.



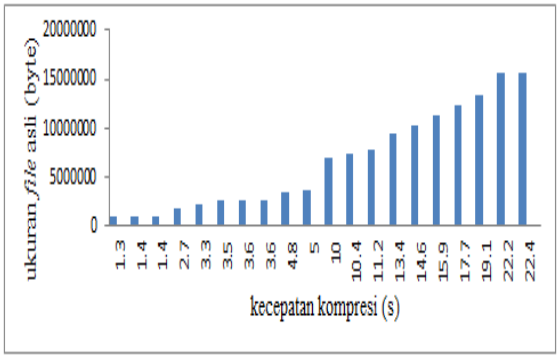
Gambar 5 Perbandingan ukuran *file* dan kecepatan kompresi untuk *sampling rate* 44.100 Hz dan *bit rate* 1.141 Kbps



Gambar 6 Perbandingan ukuran *file* dan kecepatan kompresi untuk *sampling rate* 48.000 Hz dan *bit rate* 1.536 Kbps

e. Sampling Rate 96.000 Hz dan Bit Rate 3.072 Kbps

Gambar 7 menunjukan perbandingan antara ukuran *file* dan kecepatan kompresi. Dapat dilihat pada Gambar 7 bahwa waktu untuk proses kompresi semakin meningkat pada ukuran *file* yang lebih besar.

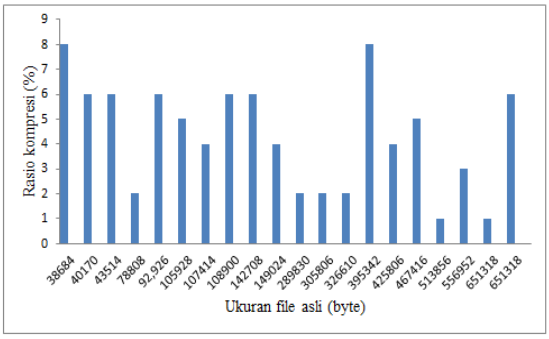


Gambar 7 Perbandingan ukuran *file* dan kecepatan kompresi untuk *sampling rate* 96.000 Hz dan *bitrate* 3.072 Kbps

3.2 Grafik Perbandingan Ukuran *File* Asli dan Rasio Kompresi

a. Sampling Rate 8.000 Hz dan Bit Rate 128 Kbps

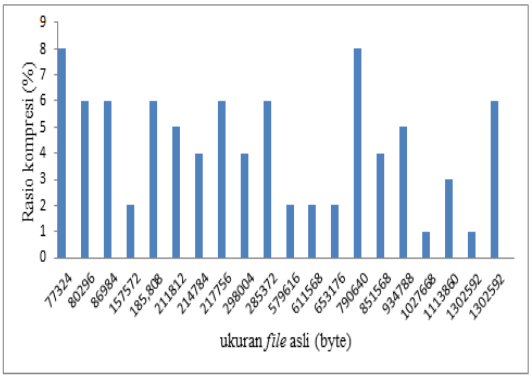
Gambar 8 menunjukan perbandingan antara ukuran *file* asli dan rasio kompresi *file* yang di-*input* dengan *sampling rate* 8.000 Hz dan *bit rate* 128 Kbps.



Gambar 8 Perbandingan ukuran *file* dan rasio kompresi untuk *sampling rate* 8.000 Hz dan *bit rate* 128 Kbps

b. Sampling Rate 8.000 Hz dan Bit rate 256 Kbps

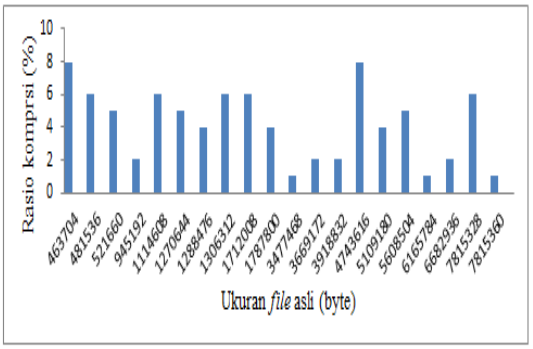
Gambar 9 menunjukan perbandingan antara ukuran *file* asli dan rasio kompresi *file* yang di-*input* dengan *sampling rate* 8.000 Hz dan *bit rate* 256 Kbps.



Gambar 9 Perbandingan ukuran *file* dan rasio kompresi untuk *sampling rate* 8.000 Hz dan *bit rate* 256 Kbps

c. Sampling Rate 44.100 Hz dan Bit Rate 1.141 Kbps

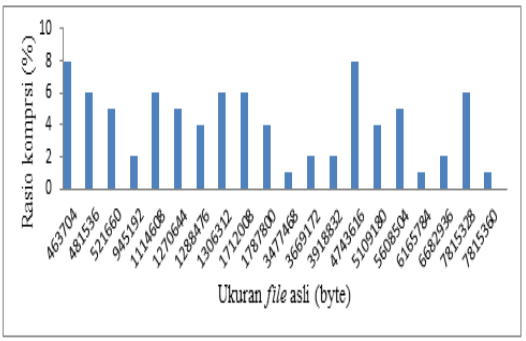
Gambar 10 menunjukan perbandingan antara ukuran *file* asli dan rasio kompresi *file* yang di-*input* dengan *sampling rate* 44.100 Hz dan *bitrate* 1.141 Kbps.



Gambar 10 Perbandingan ukuran *file* dan rasio kompresi untuk *sampling rate* 44.100 Hz dan *bit rate* 1.141 Kbps

d. Sampling Rate 48.000 Hz dan Bit Rate 1.536 Kbps

Gambar 11 menunjukan perbandingan antara ukuran *file* asli dan rasio kompresi *file* yang di-*input* dengan *sampling rate* 48.000 Hz dan *bit rate* 1.536 Kbps.



Gambar 11 Perbandingan ukuran *file* dan rasio kompresi untuk *sampling rate* 48.000 Hz dan *bit rate* 1.536 Kbps

4. KESIMPULAN

Berdasarkan pengujian dan hasil penelitian yang dilakukan terhadap aplikasi kompresi *file* audio dengan algoritma *Arithmetic Coding*, maka dapat disimpulkan :

1. Algoritma *Arithmetic Coding* dapat diimplementasikan pada proses kompresi dan dekompresi *file* audio berekstensi *.wav berklasifikasi stereo.
2. Proses kompresi membutuhkan waktu lebih lama pada *file input* berukuran besar dibanding *file* berukuran kecil.
3. Algoritma *Arithmetic Coding* tidak optimal dalam melakukan kompresi audio dapat dilihat pada rasio kompresi yang rendah dan grafik yang tidak stabil.

Aplikasi kompresi ini menghasilkan *file* audio dengan format *.arth yang dapat dijalankan setelah didekompresi terlebih dahulu. Hal tersebut dikarenakan terjadi perubahan struktur pada *file* setelah melewati proses kompresi.

5. SARAN

Berikut adalah saran yang dapat penulis berikan untuk pengembangan terhadap penelitian ini :

1. Dilakukan penelitian lebih lanjut membandingkan antara kompresi *Arithmetic Coding* dengan metode kompresi yang lain sehingga akan diperoleh sebuah metode kompresi yang benar-benar memiliki kinerja optimal dalam melakukan kompresi.
2. Agar algoritma *Arithmetic Coding* dapat diimplementasikan untuk proses kompresi *file* audio berekstensi lainya seperti *.mp3, *.m4a, dan pada *file* lainnya seperti *file* citra dan *file* teks maupun video.

DAFTAR PUSTAKA

[1] Rahandi. K., Rachmawati. D dan Sembiring. S., 2013, *Analisis dan Implementasi Kompresi File Audio dengan Menggunakan Algoritma Run Length Encoding*, Ilmu Komputer, Universitas Sumatra Utara, Medan.

[2] Nuraisyah, 2013, Perancangan Aplikasi Kompresi *File* Audio Dengan Algoritma Aritmetic Coding, Informatika, Medan.

[3] Bukhari,F, Nurhudayani,S dan Silalahi, B.P., 2003, Pengkodean aritmatika untuk kompresi data teks,*http://download.portalgaruda.org/article.php?article=85675&val=235&title=*, diakses: 26 September 2014.

[4] Gozali.F dan Mervyn, 2004, Analisa Perbandingan Kompresi Data dengan Teknik Arithmetic Coding dan Run Length Encoding, *JETri*, Nomor 1, Volume 4, Halaman 37-38.

[5] Gunawan,I dan Gunadi, K., 2005, Pembuatan Perangkat Lunak Wave Manipulator untuk Memanipulasi File

WAV, Nomor 1, Volume 6, Halaman 41-50.

[6] Prasetyo, B. G., Santoso, E dan Marji., *Kompresi File Audio Menggunakan Algoritma Huffman Shift Coding*, Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya, Malang.