KOMIK (Konferensi Nasional Teknologi Informasi dan Komputer)
Volume 5, Nomor 1, Oktober 2021
DOI: 10.30865/komik.v5i1.3671

ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 193-202

Perancangan Aplikasi Kompresi File Gambar Menggunakan Algoritma Additive Code

Elvia Hariska

Fakultas Ilmu Komputer Dan Teknologi Informasi, Program Studi Teknik Informatika, Universitas Budi Darma, Medan, Indonesia
Email : hariskae@gmail.com

Abstrak- *File* gambar merupakan sebuah *file* yang memiliki *size* cukup besar, apalagi *file* gambar tersebut berformat *bitmap*. Dimana format *bitmap* merupakan format yang minim kompresi.Tentunya hal tersebut menjadi kendala dalam proses pengiriman *file* gambar serta memerlukan ruang penyimpanan yang besar. Masalah ini dapat diatasi dengan cara mengkompresi data tersebut. Kompresi data dapat mengurangi ukuran penyimpanan data pada memori internal maupun eksternal, mempercepat pengiriman data dikarenakan ukuran penyimpanan menjadi lebih kecil. Kompresi data memiliki beberapa algoritma yang dapat digunakan. Dalam penelitian ini penulis menggunakan algoritma *additive code* yang dimana merupakan salah satu teknik kompresi *lossless* sehingga hasil dekompresi dari data terkompresi sama dengan data aslinya. Perancangan sistem ini terdiri dari 2 proses utama yaitu proses kompresi dan proses dekompresi serta proses menghitung kinerja pembanding *Ratio of Compression (RC), Compression Ratio (CR),* dan *Redudancy*. Pada proses kompresi *file* gambar berwarna yang berekstensi *.bmp dengan menggunakan algoritma *Additive Code* menghasilkan *Ratio of Compression* (RC) sebesar 1,739 , *Compression Ratio* (CR) sebesar 57,5 %, dan *Redudancy* sebesar 42,5 %. Pada penelitian kompresi *file* gambar menggunakan algoritma *additive code* ini diharapkan dapat membantu *user* dalam mengkompresi *file* gambar yang awalnya berukuran besar menjadi lebih kecil.

Kata Kunci : *File* Gambar, Format *Bitmap*, Kompresi, Algoritma *Additive Code*, Perancangan

Abstract- *Image file is a file that has a size is quite large, let alone the image file is formatically bitmap. Where the bitmap format is a format that is minim compression. It is the constraints in the process of sending the image file and require a large storage space. This problem can be overcome by compressing the data. Compression data can reduce the size of data storage in internal and external memory, accelerate data delivery due to storage size smaller. Data compression has some algorithms that can be used. In this study the authors use the Additive Code Algorithm which is one of the lossless compression techniques so that the decompression results of the compressed data are equal to the original data. The design of this system consists of 2 main processes of compression and decompression process and process counting the performances of Ratio of Compression (RC), Compression Ratio (CR),* dan *Redudancy. In the process of the compression of the colored image file that is extensive *.bmp by using the Additive Code Algorithm produces the Ratio of Compression (RC) of 1,739 , Compression Ratio (CR) of 57,5 %, and Redudancy of 42,5 %. In the compression of the image file using the Additive Code Algorithm is expected to help the user in the compressing the large file image that is initially largeer.*

Keywords : *Image File, Bitmap Format, Compression, Additive Code Algorithm, Design*

1. PENDAHULUAN

Kemajuan teknologi informasi saat ini ikut memberikan efek dalam perubahan kebiasaan manusia dalam bertukar data dan informasi, yang menimbulkan meningkatnya kebutuhan terhadap data digital. Hal tersebut menjadi tantangan yang cukup besar akan terbatasnya ruang penyimpanan (*storage*). Dalam proses bertukar data sangat dipengaruhi oleh ukuran suatu data itu sendiri. Semakin besar kapasitas suatu data maka akan semakin lama proses dalam bertukar data tersebut. Seperti *file* gambar, yang dimana semakin besar ukuran *pixel* suatu gambar maka semakin besar pula ukuran *file* yang harus disimpan diruang penyimpanan, apalagi jika *file* gambar yang digunakan adalah *file* gambar berformat *Bitmap* (.bmp). Dimana format .bmp termasuk format yang sedikit kompresi sehingga ukurannya lumayan besar. Tetapi, walaupun demikian *bitmap* (.bmp) memiliki kelebihan dari segi mutu, dimana *bitmap* memiliki mutu lebih bagus dari format *file* gambar lainnya (.jpg dan .png) yang dimana format *bitmap* umumnya tidak terkompresi sehingga tidak terdapat data yang hilang[1].

File citra atau gambar yang lebih besar dapat memakan waktu yang lama dalam proses pengiriman data apalagi jika *file* gambar yang digunakan adalah *file* berformat *Bitmap* (.bmp), terkadang sering terjadi resiko tidak dapat tertampung pada media penyimpanan dan terkadang tidak tersampaikan dalam proses pengiriman data. Salah satu teknik yang diperlukan dalam permasalah ini adalah kompresi. Ada beberapa aplikasi kompresi yang dapat digunakan dengan hasil kinerja yang berbeda, beberapa memberikan hasil yang terbaik dan dapat mengurangi 50-75% kapasitas dari data asli. Hal tersebut terjadi karena dipengaruhi oleh algoritma kompresi yang digunakan, dengan mengurangi jumlah bit yang sama dari data asli. Dalam kompresi data terdapat banyak algoritma didalamnya yang dapat mengkompres *file* gambar dan memiliki hasil yang berbeda pula, tetapi semuanya didasari pada prinsip yang sama yaitu mengkompres *file* gambar. Salah satu algoritma diantarnya adalah algoritma *Additive Code*[2].

Algoritma *additive code* merupakan algoritma yang mencakup dua algoritma kompresi yaitu algoritma *golbach code* dan algoritma *elias gamma code* yang dimana memungkinkan lebih efisien dalam mengkompres sebuah *file*. Pada saat ini, penelitian tentang pengkompresian *file* dengan menggunakan algoritma *additive code* masih sangat sedikit padahal algoritma *additive code* dimungkinkan dapat lebih efisien dalam mengkompres sebuah *file*.

Pada tahun 2019, Agus Adi Pramadi dkk telah berhasil mengkompresi file gambar dengan menggunakan algoritma *even-rodeh*. Bahwasanya dengan menggunakan algoritma *even-rodeh* ini dapat memperkecil ukuran data

ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 193-202

suatu *file* gambar sehingga penyimpanan menjadi lebih kecil sehingga dapat menghemat ruang penyimpanan memori lebih sedikit[3].

Berdasarkan penelitian yang dilakukan oleh Rizki Yannur Tanjung pada tahun 2020 tentang "Perancangan Aplikasi Kompresi File Dokumen Menggunakan Algoritma *Additive Code*" disimpulkan bahwa hasil dari kompresi *file* dokumen dengan menggunakan algoritma *Additive Code* memiliki rasio sampai 64% [4]. Berdasarkan penelitian yang dilakukan oleh Michael Simangunsong pada tahun 2020 tentang "Perbandingan Algoritma *Elias Delta Code* Dan *Unary Coding* Dalam Kompresi Citra Forensik" dapat disimpulkan bahwa dari perbandingan antara algoritma *elias delta code* dan *unary coding* diperoleh rasio 82% dan 1,29%, sehingga algoritma *elias delta code* yang paling efektif dalam proses kompresi citra forensik[5].

Berdasarkan penelitian yang dilakukan oleh Raras Krasmala dkk pada tahun 2017 tentang "Kompresi Citra Dengan Menggabungkan Metode *Discrete Cosine Transform* (DCT) dan Algoritma *Huffman*" disimpulkan bahwa dengan menggabungkan kedua algoritma tersebut dapat mengkompres gambar dengan maksimal sehingga dapat mengurangi pemakaian ruang penyimpanan[6]. Berdasarkan penelitian yang dilakukan oleh Muhammad Yogie pada tahun 2018 tentang "Penerapan Algoritma *Goldbach Codes* Pada Kompresi File Gambar Terenskripsi *Vigenere Cihper* " disimpulkan bahwa algoritma *goldbach codes* merupakan algoritma pengkompresian file yang cukup handal dalam kompresi *file* gambar[7].

Berdasarkan uraian latar belakang masalah di atas, penulis bermaksud untuk menguraikan tahap-tahap pemberian solusi yang dituangkan dalam penelitian dengan judul **"Perancangan Aplikasi Kompresi File Gambar Menggunakan Algoritma *Additive Code*"**.

2. METODOLOGI PENELITIAN

2.1 Kompresi

Kompresi merupakan pengubahan data menjadi data yang baru dengan ukuran yang lebih kecil. Kompresi data bertujuan untuk mempercepat proses pengiriman data dan menghemat pemakaian ruang penyimpanan komputer. 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[8].
Pada suatu teknik yang digunakan untuk mengkompesi data terdapat beberapa faktor yang dapat digunakan untuk menentukan kualitas data yang dikompresi, antara lain:

a. *Ratio of Compression* (RC)
Ratio of Compression (RC) ialah nilai perbandingan antara ukuran bit data sebelum dikompresi dengan ukuran bit data yang telah dikompresi. Secara sistematis dapat dituliskan sebagai berikut :
$$RC = \frac{Ukuran\ Data\ Sebelum\ Dikompresi}{Ukuran\ Data\ Setelah\ Dikompresi} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (1)$$
b. *Compression Ratio* (CR)
Compression Ratio (CR) ialah persentase perbandingan antara data yang sudah dikompresi dengan data yang belum dikompresi. Secara sistematis dapat dituliskan sebagai berikut :
$$CR = \frac{Ukuran\ Data\ Setelah\ Dikompresi}{Ukuran\ Data\ Sebelum\ Dikompresi} \times 100\% \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (2)$$
c. *Redudancy* (Rd)
Redudancy ialah hasil penilaian nilai rasio dengan hasil nilai rasio kompresi. Secara sistematis dapat dituliskan sebagai berikut :
$$Rd = \frac{file\ sebelum\ dikompresi - file\ setelah\ dikompresi}{Ukuran\ file\ sebelum\ dikompresi}\ x\ 100\% \dots\dots\dots\dots\dots\dots (3)$$
d. *Space Saving* (*Ss*)
Space Saving (*Ss*) adalah selisih antara data yang belum dikompresi dengan besar data yang sudah dikompresi
$$S_s = \left(1 - \left(\frac{file\ setelah\ dikompresi}{file\ sebelum\ dikompresi}\right)\right) x\ 100\% \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (4)$$

Ada dua teknik yang dapat dilakukan dalam melakukan kompresi[9].

a. *Lossless Compression*
Lossless Compression merupakan teknik kompresi data yang tidak menghilangkan informasi sebelumnya, dimana hasil dekompresi dari data yang terkompresi sama dengan data aslinya.
b. *Lossy Compression*
Lossy Compression merupakan teknik kompresi data yang akan menghilangkan beberapa informasi, dimana hasil data yang dikompresi tidak sama dengan data aslinya, meskipun perbedaan itu cukup dekat dan masih bisa ditorerir oleh persepsi mata.

2.2 Algoritma *Additive Code*

ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 193-202

Algoritma *additive code* merupakan salah satu teknik kompresi yang dapat memperkecil suatu data berdasarkan dengan karakter pada objek yang akan dilakukan proses kompresi. Penggunaan algoritma *additive code* akan dilakukan berdasarkan karakter yang sering muncul dan akan memiliki jumlah bit terkecil berdasarkan kode *additive code*, sedangkan karakter yang paling sedikit muncul akan memiliki jumlah bit terpanjang[6].

Algoritma *additive code* mencakup dua algoritma kompresi sekaligus didalamnya yaitu Algoritma *Goldbach* untuk menetukan nilai *index* dan Algoritma *Elias Gamma Code* untuk mentukan *codeword*, sehingga algoritma *additive code* lebih efisien dalam mengkompres suatu *file* data.

Langkah-langkah untuk membangun *codeword* dari *Additive Code* adalah sebagai berikut :

a. Menentukan nilai *n*, nilai yang berasal dari *sequence* Algoritma *Goldbach*
b. Menentukan nilai *sum*, penjumlahan 2 *sequence* Algoritma *Goldbach* yang menghasilkan nilai n ($a^i + a^j = n$)
c. *Indexes* = index (posisi urutan) dari a^i, a^j
d. *Pair* = selisih index dari a^i dan a^j
   $a^i, (a^j - a^i + 1)$
e. *Codeword* adalah hasil Algoritma *Elias Gamma Code* dari a^i dan a^j dengan ketentuan :
   1. Tentukan bilangan bulat N terbesar sehingga
      $2^N \leq n < 2^{N+1}$ dan tulis $n = 2^N + L$
   2. Ubah nilai n menjadi bilangan biner, lalu hilangkan 1 bit paling kiri
   3. Kodekan N dalam bentuk *unary* sebagai N nol diikuti oleh 1 atau N 1 diikuti oleh 0
   4. Tambahkan sisa digit biner n dibelakang kode *unary* yang telah dihasilkan
f. *Lenght*, jumlah digit biner yang didapat dari *codeword*[10].

Langkah-langkah diatas merupakan rumus dalam algoritma *additive code* yang digunakan untuk memperoleh tabel kebenaran dimana nilai *heksadecimal file* gambar akan dikompres.

Tabel 1 Kode *Additive Code*

n	Sum	Indexes	Pair	Codeword	Len.
10	3+7	3,5	3,3	011:011	6
20	9+11	6,7	6,2	00110:010	8
30	5+25	4,9	4,6	00100:00110	10
40	11+29	7,11	7,5	00111:00101	10
50	25+25	9,9	9,1	0001001:1	8
60	29+31	11,12	11,2	0001011:010	10
70	35+35	14,14	14,1	0001110:1	8
80	0+80	1,20	1,20	1:000010100	10
90	29+61	11,17	11,7	0001011:00111	14
100	3+97	3,23	3,21	011:000010101	14

Sumber : David Salomon, Giovanni Motta, 2007[11].

2.3 *File* Gambar

Gambar merupakan suatu gambaran dengan objek yang sama seperti objek aslinya yang dapat diseimpan secara digital dan tersimpan di ruang penyimpanan[5]. Gambar dan *file* gambar merupakan suatu objek yang berbeda. Perbedaan gambar dengan *file* gambar yaitu jika gambar maka hanya terdapat keterangan *pixel* dan jika *file* gambar terdapat keterangan tentang dimensi, *size*, kapan terakhir kalinya gambar tersebut diedit dan lainnya[12].

3. HASIL DAN PEMBAHASAN

3.1 Analisa Masalah

Analisa dalam penelitian ini merupakan perhitungan dan perancangan perangkat lunak pengkompresian *file* gambar yang dalam prosesnya penulis menggunakan algoritma *additive code*. Algoritma *additive code* merupakan salah satu teknik kompresi *lossless*, dimana dapat memperkecil ukuran kapasitas suatu data berdasarkan karakter yang terdapat pada objek yang akan dikompres. Penelitian ini dilakukan pengkompresian dengan menerapkan algoritma *additive code* terhadap *file* gambar berformat .bmp yang pada umumnya memiliki ukuran yang relatif besar. Sehingga hasil dari penelitian ini diharapkan akan bermanfaat dalam pengurangan ukuran data suatu *file* gambar.

ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 193-202

Gambar 1 Sampel *File* Gambar

3.1.1 Penerapan Algoritma *Additive Code*

Pada tahap ini, *file* gambar diubah menjadi nilai *hexadecimal* sehingga dapat dilakukannya proses kompresi *file*. Apabila *file* gambar telah menjadi nilai *hexadecimal* maka dapat diambil 20 sampel dari nilai *hexadecimal* tersebut, seperti tabel dibawah ini :

Tabel 2 Nilai *Hexadecimal*

42	4D	38	00	4B	00	00	00	00	00
36	00	00	00	28	00	00	00	00	05

Dari tabel diatas, maka menghasilkan bilangan *hexadecimal* file .bmp yang akan dihitung secara manual. Berikut nilai *hexadecimal* diurutkan berdasarkan frekuensinya, nilai frekuensi paling banyak maka akan berada diurutan pertama.

Tabel 3 Nilai *Hexadecimal* Yang Belum Dikompresi

n	Nilai Hexadecimal	Nilai Biner	Bit	Freq	Bit x Freq
1	00	00000000	8	13	104
2	42	01000010	8	1	8
3	4D	01001101	8	1	8
4	38	00111000	8	1	8
5	4B	01001011	8	1	8
6	36	00110110	8	1	8
7	28	00101000	8	1	8
8	05	00000101	8	1	8
		Total Bit			160

Setelah bilangan *hexadecimal* diurutkan berasarkan frekuensi kemunculan dan didapatkan nilai *biner*, maka selanjutnya menghitung *bit* dan pengurutan kode *Additive Code* serta memperoleh *bit file* terkompresi. Adapun proses kompresi *file* gambar dapat dilihat pada tebel dibawah ini.

Tabel 4 Pengurutan Kode *Additive Code*

n	Hex	Codeword	Bit	Freq	Bit x Freq
1	00	1010	4	13	52
2	42	0101	4	1	4
3	4D	1011	4	1	4
4	38	010010	6	1	6
5	4B	100100	6	1	6
6	36	010011	6	1	6
7	28	100101	6	1	6
8	05	01000100	8	1	8
		Total Bit x Freq			92

Setelah kode diurutkan berdasarkan perhitungan *additive code*, maka tahap selanjutnya adalah menyusun kembali *string bit* yang telah dihasilkan dari proses kompresi sesuai dengan posisi pada bilangan *hexadecimal*.

Tabel 5 *String Bit* Hasil Kompresi Menggunakan Algoritma *Additive Code*

42	**4D**	**38**	**00**	**4B**
0101	1011	010010	1010	100100
00	**00**	**00**	**00**	**00**
1010	1010	1010	1010	1010
36	**00**	**00**	**00**	**28**
010011	1010	1010	1010	100101
00	**00**	**00**	**00**	**05**
1010	1010	1010	1010	01000100

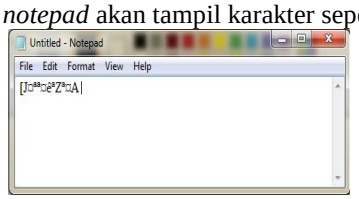
ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 193-202

String bit yang dihasilkan adalah :
"01011011010010101010010010101010101010101001001110101010101010010111010101010101010010001000100" dengan total bit yaitu 92 bit, maka perlu dilakukan penambahan jumlah *bit* karena 92 tidak habis dibagi 8. Apabila jumlah *bit* habis dibagi 8 atau kelipatan 8, maka tidak perlu dilakukan *padding*, tetapi tetap harus menambahan *flagging*.

Tabel 6 Penambahan *Padding* Dan *Flagging*

Padding	Flagging
$92\ mod\ 8 = 4 = n$ $7 - n + "1"$ $7 - 4 + "1" = 0001$	$9 - n$ $9 - 4 = 5 = 00000101$

Jadi *string bit* yang terbentuk setelah penambahan *padding* dan *flagging* yaitu :
"01011011010010101010010010101010101010101001001110101010101010010111010101010101010010001000100**00 0100000101**". Sehingga total panjang *bit* adalah 104.
File gambar yang telah dikompresi dengan menggunakan algoritma *additive code*, tipe *file*nya akan diubah menjadi tipe yang baru sesuai dengan format yang telah ditentukan sebagai pengenal bahwa *file* tersebut telah dikompres dengan aplikasi kompresi yang menerapkan algoritma *additive code*. Format *file* yang telah ditentukan adalah (*.adv), dan jika *file* tersebut dibuka menggunakan aplikasi *notepad* akan tampil karakter seperti dibawah ini :



Gambar 2 Karakter Hasil Kompresi

Untuk mengetahui tingkat kinerja algoritma kompresi, maka penulis akan mengukur hasil kompresi *file* gambar tersebut sesuai dengan parameter yang telah ditentukan. Semakin kecil hasil *file* kompresinya maka semakin berkualitas kinerja kompresinya, begitu sebaliknya. Berikut merupakan parameter untuk mengukur kinerja algoritma *additive code*.

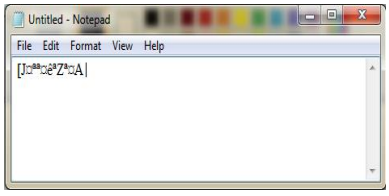
a. *Ratio of Compression* (RC)
$$RC = \frac{Ukuran\ Data\ Sebelum\ Dikompresi}{Ukuran\ Data\ Setelah\ Dikompresi}$$
$$RC = \frac{160}{92} = 1{,}739$$
b. *Compression Ratio* (CR)
$$CR = \frac{Ukuran\ Data\ Setelah\ Dikompresi}{Ukuran\ Data\ Sebelum\ Dikompresi} \times 100\%$$
$$CR = \frac{92}{160} \times 100\%$$
$$CR = 57{,}5\%$$
c. *Redudancy* (Rd)
$$Rd = \frac{160 - 92}{160}\ x\ 100\%$$
$$Rd = 42{,}5\%$$
d. *Space Saving* (*Ss*)
$$S_s = \left(1 - \left(\frac{file\ setelah\ dikompresi}{file\ sebelum\ dikompresi}\right)\right) x\ 100\%$$
$$S_s = \left(1 - \left(\frac{92}{160}\right)\right) x\ 100\% = (1 - 0{,}575)\ x\ 100\% = 0{,}425\ x\ 100\%$$
$$S_s = 42{,}5\%$$

Dari perhitungan diatas dapat disimpulkan bahwa persentase hasil kompresi *file* gambar dengan menggunakan algoritma *additive code* sebesar 57,5%.
Adapun proses dekompresi untuk *file* gambar yang telah dikompres dapat dilakukan dengan mengembalikan karakter menjadi *hexadecimal* kemudian diubah menjadi *string bit* semula.

a. *File* Hasil Kompresi

File hasil kompresi yang apabila dibuka menggunakan aplikasi *notepad* akan terlihat seperti dibawah ini.

ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 193-202



Gambar 3 Karakter Hasil Kompresi

Selanjutnya analisa karakter tersebut kedalam bentuk biner dan *decimal* sehingga menghasilkan *bit* hasil kompresi sebelumnya. Adapun *bit* hasil kompresi sebelumnya dapat dilihat pada tabel dibawah ini :

Tabel 7 Nilai Decimal Dan Biner Hasil Kompresi

No.	Karakter	*Biner*	*Decimal*
1	[	01011011	91
2	J	01001010	74
3	¤	10100100	164
4	ª	10101010	170
5	ª	10101010	170
6	¤	10100100	164
7	ê	11101010	234
8	ª	10101010	170
9	Z	01011010	90
10	ª	10101010	170
11	¤	10100100	164
12	A	01000001	65
13		00000101	5

Berdasarkan pada tabel diatas, diabil keseluruhan nilai biner kemudian diganbungkan menjadi :
"0101101101001010101001001010101010101010101001001110101010101010010110101010101010100100010000
0100000101".

b. Pengurangan *bit* menjadi *string bit* semula

Pada proses pengurangan *bit* menjadi *string bit* semula dapat dilakukan dengan menghilangkan biner *padding* dan *flagging*. Untuk mengembalikan *bit* menjadi *string bit* awal dapat dilakukan dengan pembacaan 8 *bit* terakhir kemudian ubah nilai biner menjadi *decimal*. Nyatakan hasil pembacaan dengan n, kemudian gunakan rumus $7 + n$ untuk mengembalikan *string bit* kebentuk semula.
"0101101101001010101001001010101010101010101001001110101010101010010110101010101010100100010000
01**00000101**"
8 *bit* terakhir = 00000101 = 5 = n
7 + n = 7 + 5 = 12
Hilangkan dari *string bit* sebanyak 12 *bit* terakhir, sehingga menjadi :
"0101101101001010101001001010101010101010101001001110101010101010010110101010101010100100010"
Berdasarkan perhitungan diatas, *string bit* berjumlah 92 bit seperti semula sehingga dapat dilakukan pembacaan *string bit*.

c. Pembacaan string bit

Pada pembacaan *string bit* dengan algoritma *additive code* dilakukan dengan mengecek bit pertama dengan kode bit *additive code* pada tabel 4. Apabila didapati *string bit* yang sesuai dengan *string bit additive code* maka dirubah dari nilai biner menjadi nilai *hexadecimal*. Adapun tabel hasil pengecekan *bit* yang telah didekompresi sebagai berikut :

Tabel 8 Pengecekan Bit Dekompresi

Nilai Biner	Nilai Hexadecimal
0101	42
1011	4D
010010	38
1010	00
100100	4B
1010	00
1010	00
1010	00
1010	00

ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 193-202

1010	00
010011	36
1010	00
1010	00
1010	00
100101	28
1010	00
1010	00
1010	00
1010	00
01000100	05

Dari penjabaran diatas dapat disimpulkan bahwa hasil dekompresi dari algoritma *additive code* dengan melakukan pembacaan ulang keseluran sampel bilangan biner hasil kompresi menggunakan algoritma *additive code*. Pembacaan bilangan biner dimulai dari awal sampai akhir hingga membentuk sebuah nilai biner yang benar sesuai dengan tabel kode kebenaran algoritma *additive code*. Hasil dari dekompresi sesuai dengan *string bit* semula yang terdapat pada *file* gambar. Berikut adalah hasil dekompresi yang sesuai dengan *string bit* semula yaitu dengan bilangan *hexadecimal* "42 4D 38 00 4B 00 00 00 00 00 36 00 00 00 28 00 00 00 00 05".

3.2 Implementasi

Implementasi program merupakan tahapan uji coba dari suatu sistem yang telah dirancang. Pada bagian ini membahas tentang spesifikasi perangkat keras (*hardware*) dan perangkat lunak (*software*), serta tampilan sistem ketika sedang berjalan.

Pada spesifikasi perangkat keras (*hardware*) yang digunakan dalam membangun sistem dan menjalankannya agar berjalan dengan baik, maka diperlukan spesifikasi minimum perang karas (*hardware*) sebagai berikut :

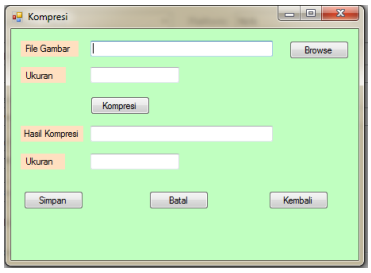
a. *Processor* : Intel(R) Core(TM) i3-2348 CPU @ 2.30GHz 4(CPUs) ~2.3GHz
b. Memori : RAM 2 GB
c. *Harddisk* : 500 GB
d. Monitor : 1366 x 768 pixel
e. *Keyboard* dan *mouse*

Spesifikasi perangkat lunak (*software*) yang digunakan dalam membangun sistem dan menjalankannya agar berjalan dengan baik, maka diperlukan spesifikasi minimum perang lunak (*software*) sebagai berikut :

a. Sistem Operasi : *Windows 7 Ultimate* atau versi setelahnya
b. Aplikasi : Microsoft Visual Studio 2008

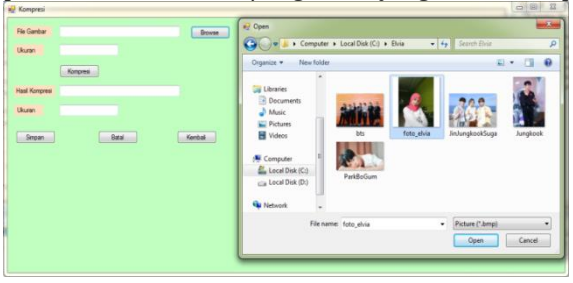
Tampilan sistem merupakan penampakan akhir dari antarmuka sistem yang dirancang.

a. *Form* kompresi merupakan tampilan *form* dimana sistem dapat melakukan proses pengkompresian *file* gambar yang telah di*input*. Setelah *file* gambar dikompres maka menghasilkan *file* gambar terkompresi dengan ukuran yang lebih kecil.



Gambar 4 *Form* Kompresi

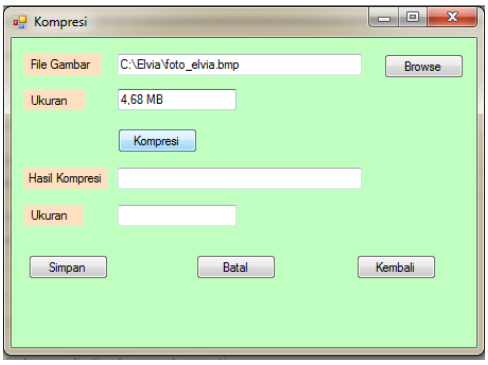
Pada *form* ini akan menampilkan proses pengkompresian *file* gambar menggunakan algoritma *additive code*, yang dimana *user* menginputkan terlebih dahulu *file* gambar yang akan dikompresi.



Gambar 5 Pengimputan *File* Gambar

ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 193-202

Setelah memilih *file* gambar yang akan dikompres, maka aplikasi akan menampilkan *file* gambar tersebut seperti gambar dibawah ini :



Gambar 6 Tampilan *File* Gambar Yang Dipilih

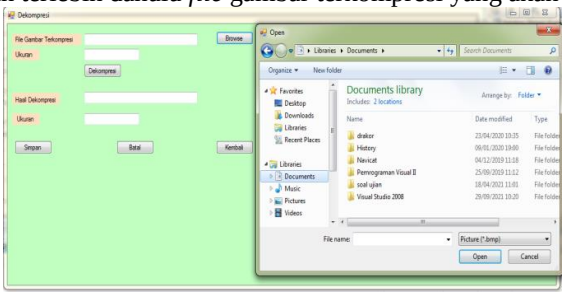
Apabila *file* gambar telah diinput maka *user* dapat melakukan proses kompresi pada *file* tersebut. Adapun proses kompresi *file* gambar tersebut dapat dilihat pada gambar berikut :

Gambar 7 Kompresi *File* Gambar

Dari gambar diatas, *file* gambar telah berhasil dikompres yang dimana hasil *file* gambar terkompresi memiliki ukuran yang lebih kecil dari *file* aslinya. Kemudian, *user* dapat memilih tombol simpan, tombol batal dan tombol kembali. Dimana tombol simpan untuk menyimpan *file* gambar terkompresi, tombol batal untuk mengulang kembali proses pemilihan *file* gambar yang akan dikompres, dan tombol kembali untuk kembali ke *form* menu utama.

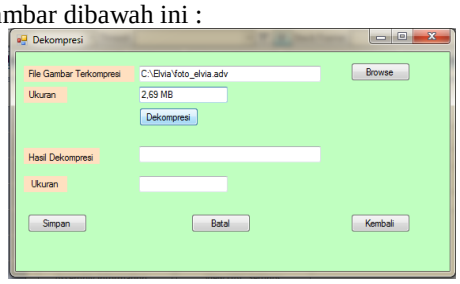
b. Form Dekompresi

Pada *form* ini akan menampilkan proses dekompresi *file* gambar menggunakan algoritma *additive code*, yang dimana *user* menginputkan terlebih dahulu *file* gambar terkompresi yang akan didekompresi.



Gambar 8 Pengimputan *File* Terkompresi

Setelah memilih *file* gambar terkompresi yang akan didekompres, maka aplikasi akan menampilkan *file* gambar terkompresi tersebut seperti gambar dibawah ini :



ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 193-202

Gambar 9 Tampilan *File* Gambar Terkompresi Yang Dipilih

Apabila *file* gambar terkompresi telah diinput maka *user* dapat melakukan proses dekompresi pada *file* tersebut. Adapun proses dekompresi *file* gambar terkompresi tersebut dapat dilihat pada gambar berikut :

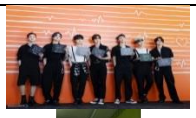
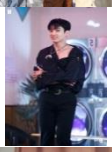
Gambar 10 Proses Dekompresi *File* Terkompresi

Dari gambar diatas, *file* gambar terkompresi telah berhasil didekompresi yang dimana hasil *file* gambar hasil dekompresi memiliki ukuran yang sama dengan *file* aslinya. Kemudian, *user* dapat memilih tombol simpan, tombol batal dan tombol kembali. Dimana tombol simpan untuk menyimpan *file* gambar hasil dekompresi, tombol batal untuk mengulang kembali proses pemilihan *file* gambar yang akan didekompresi, dan tombol kembali untuk kembali ke *form* menu utama.

3.2.1 Hasil Pengujian

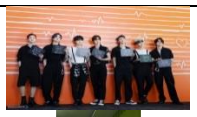
Adapun perbedaan *file* gambar sebelum dikompresi dan sesudah dikompresi dapat dilihat dari hasil pengujian dibawah ini :

Tabel 9 Hasil Pengujian Kompresi *File* Gambar

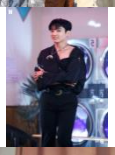
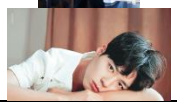
No	Sebelum Dikompresi			Sesudah Dikompresi		
	Gambar	Nama *File* Gambar	Ukuran	Nama *File* Gambar Terkompresi	Ukuran	Kinerja
1		bts.bmp	7,62 MB	bts.adv	4,572 MB	RC 1,667 CR 60% Rd 40%
2		foto_elvia.bmp	4,68 MB	foto_elvia.adv	2,69 MB	RC 1,739 CR 57,5% Rd 42,5%
3		JinJungkookSuga.bmp	7,62 MB	JinJungkookSuga.adv	4,572 MB	RC 1,667 CR 60% Rd 40%
4		Jungkook.bmp	8,00 MB	Jungkook.adv	4,696 MB	RC 1,7 CR 58,7% Rd 41,2%
5		ParkBoGum.bmp	2,75 MB	ParkBoGum.adv	1,61 MB	RC 1,7 CR 58,7% Rd 41,2%

Dari hasil pengujian diatas dapat disimpulkan bahwa *file* gambar terkompresi memiliki ukuran yang lebih kecil dibandingkan dengan *file* gambar yang belum dikompres. Sedangkan hasil pengujian dekompresi terhadap *file* terkompresi dapat dilihat pada tabel berikut :

Tabel 10 Hasil Pengujian Dekompresi *File* Gambar Terkompresi

No	Nama *File* Gambar Terkompresi	Ukuran	Gambar	Nama *File* Gambar Sesudah Dekompresi	Ukuran
1	bts.adv	4,572 MB		bts.bmp	7,62 MB
2	foto_elvia.adv	2,69 MB		foto_elvia.bmp	4, 68 MB

ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 193-202

3	JinJungkookSuga.adv	4,572 MB		JinJungkookSuga.bmp	7,62 MB
4	Jungkook.adv	4,696 MB		Jungkook.bmp	8,00 MB
5	ParkBoGum.adv	1.61 MB		ParkBoGum.bmp	2,75 MB

Dari hasil pengujian dekompresi diatas, dapat disimpulkan bahwa hasil dekompresi *file* gambar memiliki ukuran yang sama dengan *file* gambar sebelum terjadinya kompresi.

4. KESIMPULAN

Berdasarkan dari pembahasan penelitian yang telah dilakukan, maka penulis mendapatkan hasil akhir dari penelitian tersebut yang disimpulkan menjadi beberapa kesimpulan sebagai berikut :

a. Setelah mengikuti prosedur kompresi dengan menggunakan algoritma *additive code* menghasilkan bahwa suatu *file* gambar yang memiliki ukuran yang cukup besar dapat dikompres menjadi *file* gambar yang baru dengan ukuran yang lebih kecil.
b. Setelah menerapkan algoritma *additive code* untuk mengkompresi *file* gambar telah membuktikan bahwa *file* gambar tersebut telah berhasil dikompresi, dengan hasil *Ratio of Compression* (RC) = 1,739, *Compression Ratio* (CR) = 57,5%, dan *Redudancy* (Rd) = 42,5%.
c. Aplikasi kompresi *file* gambar dengan menerapkan algoritma *additive code* yaitu menggunakan Microsoft Visual Studio 2008 dapat berjalan dengan baik.

REFERENCES

[1] A. Satyapratama, M. Yunus, P. Studi, and T. Informatika, "KOMPRESI FILE GAMBAR BMP DAN PNG," *J. Teknol. Inf.*, vol. 6, pp. 69–81.
[2] G. A. NABILA, "ANALISIS PERBANDINGAN ALGORITMA BOLDI VIGNA ζ1 CODE DAN ALGORITMA EVEN-RODEH CODE PADA KOMPRESI FILE TEKS," UNIVERSITAS SUMATERA UTARA MEDAN, 2019.
[3] A. A. Pramadi, S. D. Nasution, B. Purba, A. Event, and R. Code, "PENERAPAN ALGORITMA EVEN-RODEH PADA APLIKASI KOMPRESI FILE," *KOMIK (Konferensi Nas. Teknol. Inf. dan Komputer)*, vol. 3, pp. 73–84, 2019, doi: 10.30865/komik.v3i1.1570.
[4] R. Y. TANJUNG, "Perancangan aplikasi kompresi file dokumen menggunakan algoritma additive code," UNIVERSITAS BUDIDARMA MEDAN, 2020.
[5] M. Simangunsong, "Perbandingan Algoritma Elias Delta Code Dan Unary Coding Dalam Kompresi Citra Forensik," vol. 12, no. 1, pp. 18–26, 2020.
[6] R. Krasmala, A. B. Purba, U. T. Lenggana, and T. Informatika, "Kompresi Citra Dengan Menggabungkan Metode Discrete Cosine Transform (DCT) dan Algoritma Huffman," *ISSN 2527-9165*, vol. 2, no. 1, pp. 1–9, 2017.
[7] M. Yogie, "PENERAPAN ALGORITMA GOLDBACH CODES PADA KOMPRESI FILE GAMBAR TERENKRIPSI VIGENERE CIHPER," *J. Pelita Inform.*, vol. 7, pp. 81–85, 2018.
[8] N. F. Rizky and S. D. Nasution, "Penerapan Algoritma Elias Delta Codes Dalam Kompresi File Teks," *Build. Informatics, Technol. Sci.*, vol. 2, no. 2, pp. 109–114, 2020.
[9] M. H. D. A. L. I. Subada, "ANALISIS PERBANDINGAN ALGORITMA EVEN-RODEH CODE DAN ALGORITMA FIBONACCI CODE UNTUK KOMPRESI FILE TEKS," UNIVERSITAS SUMATERA UTARA, 2018.
[10] A. PRATIWI, "PERANCANGAN APLIKSI KOMPRESI FILE AUDIO DENGAN MENERAPKAN ALGORITMA ADDITIVE CODE SKRIPSI Oleh :," UNIVERSITAS BUDI DARMA, 2020.
[11] D. Salomon and G. Motta, *Handbook of Data Compression*. New York.
[12] E. Hariska, E. Yuliani, and S. D. Nasution, "Performance Comparison Analysis of the Elias Delta Code Algorithm with the Even Rodeh Code Algorithm for Compressing Image Files," vol. 5, no. 1, pp. 29–36, 2021, doi: 10.30865/ijics.v5i1.2888.