semanTIK, Vol.1, No.2, Jul-Des 2015, pp. 1-10
ISSN: 2460-1446

APLIKASI KOMPRESI *FILE* CITRA MENGGUNAKAN ALGORITMA *ARITHMETIC CODING* BERBASIS JAVA

Aan Kurniawan Saputra *[1], Sutardi[2], Ika Purwanti Ningrum[3]
[*1,2,3]Jurusan Teknik Informatika, Fakultas Teknik, Universitas Halu Oleo, Kendari
e-mail : [*1]**aannerazzuri@gmail.com**, [2]sutardi_hapal@yahoo.com, [3]ika.purwanti.n@gmail.com

Abstrak

Kompresi data adalah proses mengubah suatu *input* data menjadi data dengan ukuran yang lebih kecil, atau proses pengkodean dari suatu data untuk mengurangi kebutuhan akan media penyimpanan. Kompresi data bertujuan untuk mengurangi ukuran data tanpa merusak tujuan dari data tersebut.Kompresi data menggunakan sedikit *disk space*. Kompresi data mempercepat akses, kompresi data dapat membuat lebih efisien.

Algoritma yang digunakan dalam penelitian ini adalah *Arithmetic Coding*. Algoritma *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*.

Hasil dari penelitian ini yaitu kompresi menggunakan algoritma *Arithmetic Coding* dapat menghasilkan citra dengan ukuran *file* yang lebih kecil, namun apabila citra uji yang digunakan mempunyai ukuran *file* yang besar, waktu yang dibutuhkan untuk melakukan proses kompresi maupun dekompresi membutuhkan waktu yang cukup lama. Dari hasil pengamatan, tingkat rasio kompresi tidak dipengaruhi oleh besarnya ukuran dimensi citra, tetapi tergantung komposisi warna citra yang bersangkutan. *File* citra hasil kompresi tidak dapat dibuka sebelum melewati proses dekompresi hal tersebut disebabkan terjadi perubahan struktur pada *file* citra saat proses dekompresi.

Kata kunci : Citra, *Arithmetic Coding*, Kompresi, Dekompresi dan *Lossles*

Abstract

Data compression is the process of converting a data input into the data with a smaller size, or the encoding of the data to reduce the need for storage media. Data compression aims to reduce the size of data without destroying the purpose of data. Data compression use less disk space. Speeding access to data compression, data compression can be made more efficient.

The algorithm used in this study is Arithmetic Coding. Arithmetic Coding algorithm is a part of the entropy encoding that converts the data into another form of data by using a little bit more often and rarely use more bits of character. This coding technique separates the message input into the symbol and exchange each symbol with a floating-point.

Results from this research that Arithmetic Coding compression algorithm can produce images with a smaller file size. However, if the test images used have a large file size, the time required to perform the compression and decompression process takes quite a long time. From the results, the level of compression ratio is not affected by the large size of the dimensions of the image, but depending on the composition of the color image is concerned. Image file compression can not be opened before passing through the decompression process that caused a change in the structure of the image file when the decompression process.

Keywords : *Image, Arithmetic Coding, Compression, Decompression and Lossles*

1. PENDAHULUAN

Perkembangan media penyimpan berkapasitas besar mengakibatkan orang tidak lagi menemui masalah jika mempunyai *file* dengan ukuran yang besar. Terlebih jika *file* yang tersebut merupakan *file image* atau gambar. Walaupun demikian, ada kalanya ukuran *file* yang besar tersebut terasa mengganggu jika harus mengatur media penyimpan untuk bermacam-macam data. Apalagi jika *file* tersebut akan dikirim secara elektronik, tentunya kapasitas *file* menjadi masalah tersendiri. Citra/gambar (*image*) merupakan hal yang vital dan menjadi bagian integral dari kehidupan sehari-hari. Pada kepentingan tertentu, citra (gambar) digunakan sebagai alat untuk mengungkapkan pertimbangan (*reason*), interpretasi, ilustrasi, penggambaran (*represent*), ingatan (*memorise*), pendidikan, komunikasi, evaluasi, navigasi, survey, hiburan, dan lain sebagainya.

Salah satu masalah yang terus berkembang sejalan dengan berkembangnya dunia komputer adalah terkait dengan penanganan data yang berukuran besar. Masalah ini muncul karena *hardware* yang digunakan terkait dengan penanganan data kurang mampu mengikuti perkembangan ukuran data yang demikian besar. Salah satu kajian yang terkait dengan penanganan data adalah kompresi data.

Kompresi data adalah proses mengubah suatu *input* data menjadi data dengan ukuran yang lebih kecil, atau proses pengkodean dari suatu data untuk mengurangi kebutuhan akan media penyimpanan [1].

Berdasarkan fungsinya, metode kompresi dibagi menjadi dua yaitu metode *Lossy* dan metode *Lossless*. Metode *Lossless* adalah suatu metode pemampatan data tanpa menghilangkan data yang dimilikinya. Proses kompresi dapat dilakukan dengan mengganti suatu data dengan kode yang berukuran lebih ringkas. Data yang telah diganti dapat dikembalikan dengan ukuran dan struktur yang sama persis dengan data asli melalui proses dekompresi. Teknik ini bersifat dua arah, karena untuk membaca data yang telah terkompres diperlukan proses dekompresi. Berbeda dengan algoritma *Lossy*, *file* hasil kompresi dengan metode *Lossless* tak dapat dibaca tanpa melalui proses dekompresi.

Teknik dekompresi ini bersifat spesial untuk satu teknik kompresi, sehingga tidak mungkin *file* yang terkompres oleh oleh algoritma kompresi A dapat didekompresi dengan algoritma dekompresi B. Tujuan utama dari metode *Lossless* adalah menghasilkan *file* yang kecil namun tanpa sedikitpun mengurangi kualitas data tersebut. Hal ini sangat penting untuk data-data yang memerlukan keakuratan yang tinggi.

Salah satu jenis berkas yang paling banyak membutuhkan proses kompresi adalah berkas citra. Citra asli umumnya disimpan dalam format Bitmap (*.bmp). Format ini menghasilkan ukuran berkas yang besar dan tidak efektif untuk disimpan atau ditransfer. Oleh karena itu penulis merancang suatu program untuk kompresi gambar dengan menggunakan algoritma *Arithmetic Coding* berbasis *java.*

2. METODE PENELITIAN

2.1 Citra Digital

Semua citra digital yang ditampilkan di layar komputer adalah sederetan atau sekumpulan piksel (*picture element*). Citra tersebut dikatakan sebagai citra digital karena bentuk representasinya yang berupa bilangan (*numbers*). Oleh komputer akan dikenal dalam urutan ‘0’ dan ‘1’ [2].

Ada beberapa format citra digital, antara lain: *.bmp, *.png, *.jpg, *.gif, *.pcx, dan sebagainya. Masing-masing format mempunyai perbedaan satu dengan yang lain terutama pada *header file*, namun ada beberapa yang mempunyai kesamaan, yaitu penggunaan *palette* untuk penentuan warna piksel. Sebagai studi kasus dalam tugas akhir ini akan digunakan format citra *.bmp yang dikeluarkan oleh Microsoft.

2.2 Citra *.bmp

Format *.bmp, disebut dengan bitmap atau format DIB (*Device Independent Bitmap*) adalah sebuah format citra yang digunakan untuk menyimpan citra bitmap digital terutama pada sistem operasi Microsoft Windows. Pada citra berformat *.bmp (bitmap) yang tidak terkompresi, piksel citra disimpan dengan kedalaman warna 1, 4, 8, 16, 24, atau 32 bit per piksel [2].

Pada umumnya citra bitmap terdiri dari 4 blok data yaitu: *.bmp *header*, *Bit*

Information (DIB *header*), *Color Palette*, dan *Bitmap Data*. BMP *header* berisi informasi umum dari citra *bitmap*. Blok ini berada pada bagian awal *file* citra dan digunakan untuk mengidentifikasi citra. Beberapa aplikasi pengolah citra akan membaca blok ini untuk memastikan bahwa citra tersebut berformat *bitmap* dan tidak dalam kondisi rusak. *Bit information* berisi informasi detail dari citra bitmap, yang akan digunakan untuk menampilkan citra pada layar. *Color palette* berisi informasi warna yang digunakan untuk indeks warna bitmap, dan bitmap data berisi data citra yang sebenarnya, piksel per piksel.

Model ruang warna yang digunakan pada citra bitmap adalah RGB (*red, green, dan blue*). Sebuah ruang warna RGB dapat diartikan sebagai semua kemungkinan warna yang dapat dibuat dari tiga warna dasar *red*, *green*, dan *blue*. RGB sering digunakan di dalam sebagian besar aplikasi komputer karena dengan ruang warna ini tidak diperlukan transformasi untuk menampilkan informasi di layar monitor.

2.3 Kompresi Data

Kompresi data adalah proses mengubah suatu *input* data menjadi data dengan ukuran yang lebih kecil, atau proses pengkodean dari suatu data untuk mengurangi kebutuhan akan media penyimpanan [1].

Kompresi data bertujuan untuk mengurangi ukuran data tanpa merusak tujuan dari data tersebut. Kompresi data menggunakan sedikit *disk space*. Kompresi data mempercepat akses, kompresi data dapat membuat lebih efisien.

Untuk mengkompres data, ada beberapa format *file* yang digunakan seperti bzip2 (*.bz2), gzip (*.gz), lzma (*.lzma), lzo (*.lzo), pack (*.z), *compress* (*.Z). Perbedaan masing-masing format kompresi ini adalah algoritma yang digunakan. Seperti bzip2 yang menggunakan *Burrows-Wheeler Transform* diikuti dengan *move-to-front Transform* dan terakhir *Huffman Coding*.

Format *.gzip yang menggunakan algoritma DEFLATE untuk kompresi data, lzma menggunakan algoritma 7-zip, lzo menggunakan algoritma LZO. Beberapa dari format kompresi data ini digunakan bersama-sama ketika meng-*archive file*. Seperti *Tape Archiver* (*.tar) yang digunakan bersama bzip2 (ekstensi file menjadi *.tar.bz2), gzip (ekstensi file menjadi *.tar.gz) atau *compress* (ekstensi file menjadi *.tar.Z).

Teknik kompresi data terbagi menjadi dua yaitu :

a. *Lossy Compression*

Teknik kompresi dimana data hasil dekompresi tidak sama dengan data sebelum kompresi namun sudah "cukup" untuk digunakan. Contoh: Mp3, *streaming media*, JPEG, MPEG, dan WMA. Kelebihan *Lossy Compression* adalah ukuran *file* lebih kecil dibanding *loseless* namun masih tetap memenuhi syarat untuk digunakan.

Biasanya teknik ini membuang bagian-bagian data yang sebenarnya tidak begitu berguna, tidak begitu dirasakan, tidak begitu dilihat oleh manusia sehingga manusia masih beranggapan bahwa data tersebut masih bisa digunakan walaupun sudah dikompresi.

b. *Loseless*

Teknik kompresi dimana data hasil kompresi dapat didekompres lagi dan hasilnya tepat sama seperti data sebelum proses kompresi. Kompresi *lossless* utamanya digunakan untuk pengarsipan, dan penyuntingan. Untuk keperluan pengarsipan, tentu kualitas yang diinginkan adalah kualitas terbaik. Begitu juga dengan penyuntingan. Menyunting data yang terkompresi secara *lossy* menyebabkan turunnya kualitas suara pada setiap penyimpanan.

Berdasarkan tipe peta kode yang digunakan untuk mengubah pesan awal (isi berkas masuk) menjadi sekumpulan *codeword*, metode kompresi terbagi menjadi dua kelompok, yaitu :

a. Metode Statik

Menggunakan peta kode yang selalu sama. Metode ini membutuhkan dua fase (*two-pass*): fase pertama untuk menghitung probabilitas kemunculan tiap simbol/karakter dan menentukan peta kodenya, dan fase kedua untuk mengubah pesan menjadi kumpulan kode yang akan ditransmisikan. Contoh: algoritma Huffman Statik.

b. Metode Dinamik (Adaptif)

Menggunakan peta kode yang dapat berubah dari waktu ke waktu. Metode ini disebut adaptif karena peta kode mampu beradaptasi terhadap perubahan karakteristik

isi berkas selama proses kompresi berlangsung. Metode ini bersifat *onepass*, karena hanya diperlukan satu kali pembacaan terhadap isi berkas. Contoh: algoritma LZW dan DMC.

2.4 Perangkat Lunak Kompresi Data

Dalam dunia komputer, kompresi sudah bukan menjadi hal yang baru. Hal terbukti denga telah banyaknya aplikasi konpresi yang dijual bebas dipasaran. Banyak perusahaan-perusahaan modern yang mengembangkan aplikasi kompresi menerapkan berbagai macam algoritma agar hasil kompresi dapat semaksimal mungkin. Aplikasi kompresi yang telah dikenal dipasaran diantaranya adalah WinZip dan WinRar.

2.5 Algoritma *Arithmetic Coding*

Pengkodean *Arithmetic Coding* adalah membuat suatu garis peluang dari 0 sampai 1 dan memberi interval pada setiap karakter dari teks *input* berdasarkan peluang kemunculannya. Semakin tinggi peluang yang dimiliki suatu karakter maka semakin besar interval yang akan diperoleh. Setelah semua karakter memiliki interval maka dilakukan pengkodean untuk menghasilkan satu bilangan *output.*

Pada umumnya, algoritma kompresi data melakukan penggantian satu atau lebih simbol *input* dengan kode tertentu. Berbeda dengan cara tersebut, *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 [3]. Untuk menjelaskan proses *encoding* dipakai algoritma pada Gambar 1, *'Low'* adalah *output* dari proses *Arithmetic Coding*.

2.6 Rancangan Sistem

a. Analisis Masalah

Pada dasarnya *file* apapun adalah rangkaian bit 0 dan 1 yang membedakan antara suatu *file* tertentu dengan *file* yang lain adalah ukuran dari rangkaian bit serta bagaimana 0 dan 1 itu ditempatkan dalam rangkaian bit tersebut. Semakin kompleks suatu *file*, ukuran rangkaian bit yang diperlukan semakin panjang, dengan demikian ukuran keseluruhan *file* juga semakin besar.

Dalam penyimpanan dan pengiriman *file* komputer, selain isi dari *file* tersebut parameter yang tidak kalah penting adalah ukuran *file*. *File* yang disimpan dalam suatu media penyimpanan berukuran sangat besar sehingga memerlukn tempat yang lebih banyak dan tidak efisien. Dalam penyimpanan dan pengiriman *file* komputer, selain isi dari *file* tersebut parameter yang tidak kalah penting adalah ukuran *file*. *File* yang disimpan dalam suatu media penyimpanan berukuran sangat besar sehingga memerlukan tempat yang lebih banyak dan tidak efisien.

b. Gambaran Umum Sistem

Sistem ini digunakan untuk melakukan proses kompresi. Kompresi merupakan metode untuk mengurangi kapasitas ukuran *file*.

Gambar 1 menunjukkan gambaran umum pemakaian aplikasi kompresi pada *file* citra.

Gambar 1 Gambaran umum sistem

Salah satu solusi untuk mengatasi masalah diatas adalah dengan melakukan kompresi *file*. Kompresi *file* terdiri dari dua proses utama yaitu kompresi dan dekompresi *file*. Jika *file* citra dikompresi maka *file* tersebut harus dapat dibaca kembali setelah *file* tersebut didekompresi. Gambar 1 menunjukan gambaran umum sistem yang dibangun.

Salah satu teknik pemampatan *file* citra yang digunakan yaitu menggunakan algoritma *Arithmetic Coding.* Ide dasar *Arithmetic Coding* adalah membuat suatu garis peluang dari 0 sampai 1 dan memberi interval pada setiap *symbol input* berdasarkan peluang kemunculanya. Semakin tinggi peluang yang dimiliki suatu karakter maka semakin besar interval yang akan diperoleh.

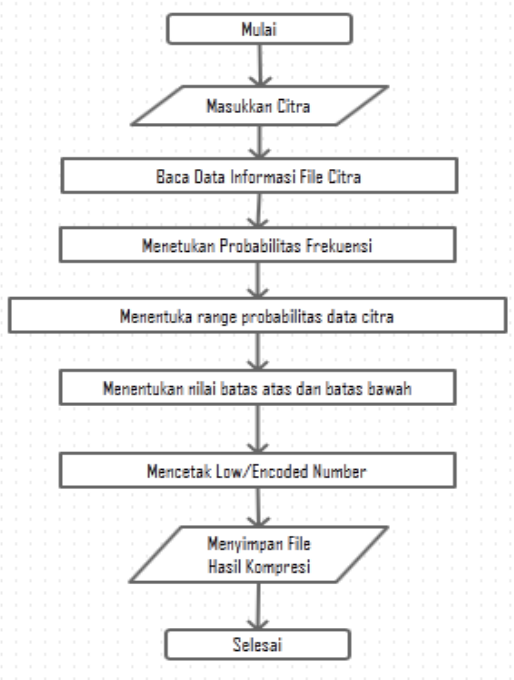
c. Proses Kompresi-Dekompresi *File* Citra

Proses kompresi *file* citra, *file* citra yang dikompresi dikodekan dengan representasi yang meminimumkan kebutuhan memori dan pada proses dekompresi, *file* citra yang sudah mengalami kompresi harus bisa dikembalikan lagi menjadi representasi semula/tidak terkompresi.

1. Proses Kompresi *File* Citra

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

Algoritma yang digunakan untuk aplikasi ini adalah algoritma *Arithmetic Coding*, dimana prosesnya berjalan seperti pada Gambar 2.



Gambar 2 *Flowchart* proses kompresi algoritma *Arithmetic Coding*

Sebagai contoh, misalnya terdapat citra digital dalam bentuk matriks seperti pada Gambar 3.

Nilai probabilitas untuk setiap nilai *Gray Level* kemudian dapat diperoleh menggunakan persamaan (1).

$$probabilitas = \frac{frekuensi}{jumlah\ data} \quad (1)$$

255	50	50	255	25	25
25	50	25	120	180	50
25	75	25	50	75	120
120	180	180	255	255	25
255	255	120	25	255	255
255	255	50	25	255	255

Gambar 3 Potongan *pixel* citra digital

Setelah probabilitas tiap nilai *Gray Level* telah diketahui, tiap nilai akan diberikan *range* tertentu yang nilainya diantara 0 dan 1, sesuai dengan probabilitas yang ada. Hal ini ditunjukkan oleh Tabel 1. Selanjutnya dilakukan proses *encoding* menggunakan algoritma *encoding*.

Tabel 1 Probabilitas frekuensi

Gray Level	**Frekuensi**	**Probabilitas**	***Range***
255	12	12/36 (1/3)	0,00 - 0,33
50	6	6/36 (1/6)	0.33 – 0,5
25	9	9/36 (1/4)	0,5 – 0,75
120	4	4/36 (1/9)	0,75 – 0,86
180	3	3/36 (1/12)	0,86 – 0,94
75	2	2/36 (1/18)	0,94 – 1,00
Total	36	1	0,00 – 1,00

Gambar 3 mempunyai 36 nilai *Gray Level*, sehingga dimisalkan 4 nilai pixel yang diambil seperti pada Gambar 4.

255	**50**	50	255	25	25
25	**50**	25	120	180	50
25	75	25	50	75	120
120	180	180	255	255	25
255	255	120	25	255	255
255	255	50	25	255	255

Gambar 4 Potongan nilai *gray level* citra digital

Pertama diambil nilai *Gray Level* '255'. Nilai *Code Range* (kondisi awal) adalah 1,0 – 0,0 = 1,

$high\_range\ (S) = 0{,}33,$

$low\_range\ (S) = 0{,}00$

Kemudian diperoleh nilai :

$High = low + CR * high\ range$

$= 0{,}00 + 1{,}0 * 0{,}33 = 0{,}33$

$Low = low + CR * low\ range$

$= 0{,}00 + 1{,}0 * 0.00 = 0{,}00$

Kemudian diambil nilai *Gray Level* '50'. Nilai *Code Range* adalah 0,33 – 0,00 = 0,33,

$high\_range\ (A) = 0{,}5$

$low\_range\ (A) = 0{,}33,$

Kemudian diperoleh nilai:

$High = low + CR * high\ range$

$= 0{,}00 + 0{,}33 * 0{,}5 = 0{,}165$

$Low = low + CR * low\ range$

$= 0{,}00 + 0{,}33 * 0{,}33$

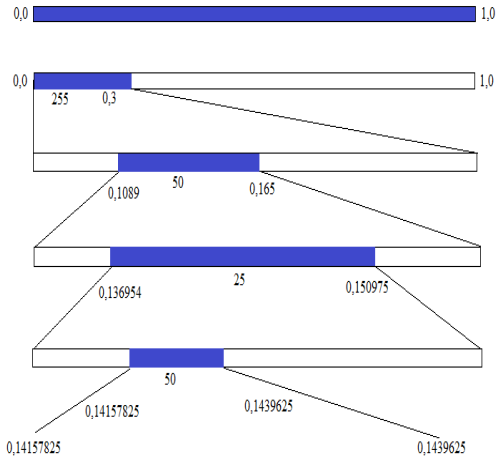
$= 0{,}1089$

Hasil *encoding* keseluruhan dapat ditunjukkan oleh Tabel 2.

Tabel 2 Proses *encoding*

Gray Level	*Low*	*High*	*CodeRange*
	0,00	1,00	
255	0,00	0,33	1,00
50	0,1089	0,165	0,33
25	0,13695	0,150975	0,0561
50	0,14157825	0,1439625	0,014025

Perulangan untuk setiap interval ditunjukkan oleh Gambar 5.



Gambar 5 Perulangan setiap interval

Dari proses tersebut diperoleh nilai *low* = 0,14157825. Nilai inilah yang direpresentasikan untuk membawa citra pada Gambar 4.

2. Proses Dekompresi *File* Citra

Selanjutnya apabila gambar tersebut akan di-*decode* maka digunakan algoritma *decoding*. Misalkan untuk Gambar 4 akan dilakukan *decoding*. Gambar 6 menunjukkan *flowchart* proses dekomporesi algoritma *Arithmetic Coding*.

Gambar 6 *Flowchart* proses dekompresi algoritma *Arithmetic Coding*

Pertama sekali diperoleh *encoded number* 0,14157825 dengan nilai *Gray Level* awal '255'. Maka proses *decoding* untuk nilai '255' yaitu :

$Low = 0{,}0$

$High = 0{,}33$

$CodeRange = 0{,}33 - 0.00 = 0{,}33$

$Encoded\ number = 0{,}14157825 - 0{,}00$

$= 0{,}14157825$

$Encoded\ number = \frac{0{,}141578}{0{,}33}$

$= 0{,}429025$

Proses *decoding* dilakukan seterusnya sampai tidak ada lagi simbol. Proses *decoding* diringkas pada Tabel 3.

Tabel 3 Proses *decoding*

Encoded number	Symbol	*Low*	*high*	*Code range*
0,429025	255	0,00	0,33	0,33
0,5825	50	0,33	0,5	0,17
0,33	25	0,5	0,75	0,25
0	50	0,33	0,5	0,27

Pada proses kompresi terdapat perhitungan rasio kompresi. Rasio kompresi menunjukan presentase besarnya kompresi yang dilakukan terhadap *file* asli [4]. Rasio kompresi diperoleh dari persamaan :

$$Rasio\ kompresi = \frac{selisih\ ukuran}{file\ asli} \times 100\% \quad (2)$$

3. HASIL DAN PEMBAHASAN

3.1 Kompresi

Hasil pengujian *file* citra diperoleh rasio kompresi yang dihasilkan tergantung dari komposisi warna gambar yang bersangkutan, Sedangkan kecepatan kompresi dipengaruhi oleh ukuran *file* asli. Semakin besar ukuran *file* maka waktu yang dibutuhkan untuk proses kompresi semakin banyak.

1. *File* citra dengan komposisi warna sama dan dimensi yang berbeda

Pada penelitian ini digunakan 5 *file* citra yang memiliki komposisi warna yang sama tetapi dengan dimensi yang berbeda. Tabel 4 menunjukan hasil kompresi citra uji.

Tabel 4 Hasil kompresi *file* citra dengan komposisi warna sama dan dimensi berbeda

Nama File	Ukuran File (Byte)	Dimensi	Hasil Kompresi (Byte)	Rasio	Kecepatan (ms)
1	2764854	1280 x 720	661378	76,01399%	8841
2	1080054	800 x 450	266192	75,383%	3075
3	691254	640 x 360	163094	76,40607%	2080
4	338742	448 x 252	84282	75,11912%	1219
5	167142	314 x 177	41390	75,236626%	601

2. *File* citra dengan komposisi warna berbeda dan dimensi yang sama

Pada penelitian ini digunakan 16 *file* citra yang memiliki komposisi warna yang berbeda tetapi dengan dimensi yang sama. Tabel 5 menunjukan hasil kompresi citra uji.

Tabel 5 Hasil kompresi *file* citra dengan komposisi warna berbeda dan dimensi yang sama

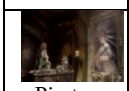
Nama File	Ukuran File (Byte)	Dimensi	Hasil Kompresi (Byte)	Rasio	Kecepatan (ms)
16 tc	786.486	512 x 512	422.570	46,271133%	2,505
15 tc	786.486	512 x 512	398.719	49,303738%	2,625
14 tc	786.486	512 x 512	374.863	52,33698%	2,500
13 tc	786.486	512 x 512	349.737	55,531693%	2,510
12 tc	786.486	512 x 512	323.937	58,807533%	2,403
11 tc	786.486	512 x 512	298.048	62,10384%	2,605
10 tc	786.486	512 x 512	282.662	64,060135%	2,460
9 tc	786.486	512 x 512	266.466	66,12196%	2,493
8 tc	786.486	512 x 512	241.254	69,32507%	2,341
7 tc	786.486	512 x 512	208.824	73,448488%	2,320

6 tc	786.486	512 x 512	177.060	77,487206%	2,468
5 tc	786.486	512 x 512	151.056	80,793556%	2,401
4tc	786.486	512 x 512	119.832	84,763622%	2,302
3 tc	786.486	512 x 512	98.596	87,463733%	2,555
2 tc	786.486	512 x 512	74.177	90,568555%	2,563
1 tc	786.486	512 x 512	37.333	95,253118%	2,350

3. *File* citra dengan komposisi warna berbeda dan dimensi yang berbeda

Pada penelitian ini digunakan 7 *file* citra yang memiliki komposisi warna yang berbeda dan dimensi yang berbeda. Tabel 6 menunjukan hasil kompresi citra uji

Tabel 6 Hasil Kompresi *File* Citra dengan komposisi warna berbeda dan dimensi yang berbeda

Nama File	Ukuran File (Byte)	Dimensi	Hasil Kompresi (Byte)	Rasio	Kecepatan (ms)
 Pirates	2.785.334	1280 x 544	2.025.853	27,267141%	9,356
 Gunung	1.440.054	800 x 600	1.387.829	3,6222253%	5,160
 Jin	1.049.906	1366 x 768	1.029.397	1,953413%	3,731
 Baboon	786.488	512 x 512	763.357	2,9410493%	2,599
 RGB2	786.488	512 x 512	754.226	4,102033%	2,583
 Butterfly	603.702	524 x 384	382.147	36,699398%	2,164
 Ayame	22.637	640 x 640	22.028	2,690286%	1,18

3.2 Dekompresi

Hasil uji coba untuk proses dekompresi ditunjukan pada Tabel 7, 8 dan 9, Proses dekompresi dari sistem terbukti dapat mengembalikan sebuah uji *file* *.arth menjadi seperti *file* aslinya dan dapat dibuka seperti semula.

Tabel 7 Hasil dekompresi *File* Citra dengan komposisi warna sama dan dimensi berbeda

Nama File	Hasil Kompresi (Byte)	Hasil Dekompresi (Byte)	Ukuran Asli (Byte)	Kecepatan (ms)	Ukuran Citra Kembali Seperti Semula
1	661.378	276.4854	276.4854	9193	Ya
2	266.192	108.0054	108.0054	4127	Ya
3	163.094	691.254	691.254	2867	Ya
4	84.282	338.742	338.742	1462	Ya
5	41.390	167.142	167.142	731	Ya

Tabel 8 Hasil dekompresi *File* Citra dengan komposisi warna berbeda dan dimensi sama

Nama File	Hasil Kompresi (Byte)	Hasil Dekompresi (Byte)	Ukuran Asli (Byte)	Kecepatan (ms)	Ukuran Citra Kembali Seperti Semula
16 tc	422.570	786.486	786.486	3.186	Ya
15 tc	398.719	786.486	786.486	3.444	Ya
14 tc	374863	786.486	786.486	3.014	Ya
13 tc	349.737	786.486	786.486	2.821	Ya
12 tc	323.937	786.486	786.486	2.794	Ya
11 tc	298.048	786.486	786.486	2.802	Ya
10 tc	282.662	786.486	786.486	2.764	Ya
9 tc	266.466	786.486	786.486	2.866	Ya
8 tc	241.254	786.486	786.486	2.809	Ya
7 tc	208.824	786.486	786.486	2.775	Ya
6 tc	177.060	786.486	786.486	2.808	Ya
5 tc	151.056	786.486	786.486	2.730	Ya
4 tc	119.832	786.486	786.486	2.775	Ya
3 tc	98.596	786.486	786.486	2.801	Ya
2 tc	74.177	786.486	786.486	2.811	Ya
1 tc	37.333	786.486	786.486	2.749	Ya

Tabel 9 Hasil dekompresi *file* citra dengan komposisi warna berbeda dan dimensi berbeda

Nama File	Hasil Kompresi (Byte)	Hasil Dekompresi (Byte)	Ukuran Asli (Byte)	Kecepatan (ms)	Ukuran Citra Kembali Seperti Semula
Pirates	2.025.853	2.785.334	2.785.334	9341	Ya
Gunung	1.387.829	1.440.054	1.440.054	5264	Ya
Jin	1.029.397	1.049.906	1.049.906	4203	Ya
Baboon	763.357	786.488	786.488	3332	Ya
RGB2	754.226	786.488	786.488	3325	Ya
Butterfly	382.147	603.702	603.702	2345	Ya
Ayame	22.028	22.637	22.637	128	Ya

4. KESIMPULAN

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

1. Algoritma *Arithmetic Coding* dapat diimplementasikan pada proses kompresi dan dekompresi *file* citra berekstensi *.bmp.
2. Proses kompresi membutuhkan waktu lebih lama pada *file input* berukuran besar dibanding *file* berukuran kecil.
3. Tingkat rasio kompresi sebuah citra tidak dipengaruhi oleh ukuran dimensi citra, melainkan tergantung pada banyak atau sedikitnya komposisi warna citra yang bersangkutan..
4. Ukuran *file* citra yang dihasilkan setelah proses dekompresi tepat seperti semula.

5. SARAN

Saran yang dapat diberikan untuk penelitian selanjutnya yaitu :

1. Dilakukan penelitian lebih lanjut untuk membandingkan antara metode kompresi menggunakan *Arithmetic Coding* dengan mentode kompresi yang lain sehingga diperoleh metode kompresi yang lebih baik.
2. Agar algoritma *Arithmetic Coding* dapat diimplementasikan untuk proses kompresi *file* lainnya seperti *file* suara dan video.

DAFTAR PUSTAKA

[1] Indah, 2013, Teknik Kompresi Data, *http://indah.blog.widyatama.ac.id/2013/01/Teknik-Kompresi-Data*, diakses 1 Juli 2015

[2] Elka, 2010, Pengolahan Citra digital, *http://elka2002,blogspot.com/2010/01apa-itu-pengolahan-citra-digital.html?=1*, diakses 1 Juli 2015

[3] Nuraisyah, 2013, *Perancangan Aplikasi Kompresi File audio dengan Algoritma Arithmetic Coding*, Informatika Medan.

[4] Bentley, Lonnie D, dan Whitten, J.L., 2007, *System Analysis and Design for the Global Seventh edition*, New York, McGraw-Hill