

Analisa Perbandingan Algoritma Stout Code Dan Algoritma Rice Code Pada Kompresi File Gambar

Karazisokhi Laia

Universitas Budi Darma, Indonesia, e-mail: sokhi@gmail.com

Info Artikel	Abstrak
Diajukan: 20-09-2024 Diterima: 23-09-2025 Diterbitkan: 30-09-2025 Kata Kunci: Perbandingan; Kompresi; Stout Code; Rice Code; Gambar. Keywords: Comparison; Compression; Stout Code; Rice Code; Image.  Lisensi: cc-by-sa Copyright © 2025 by Author. Published by Faatuatua Media Karya	Citra diartikan sebagai gambaran mengenai objek yang diamati. Penggunaan file gambar untuk proses pertukaran data kerap dilakukan, hal ini karena suatu gambar dapat mempresentasikan keadaan atau informasi tertentu. Gambar terdiri dari kumpulan matriks pixel yang memiliki tiga nilai warna pada setiap pixelnya, antara lain Red Green dan Blue atau RGB. Masalah yang dihadapi adalah semakin besar ukuran pixel suatu gambar maka semakin besar pula ukuran file yang harus disimpan di ruang penyimpanan. Keterbatasan media penyimpanan menjadi kendala untuk mengelola suatu gambar atau citra. Untuk membuat banyak ruang kosong dan memiliki ukuran data yang tidak besar pada media penyimpanan diperlukan teknik kompresi yang memperkecil ukuran data. Penelitian ini melakukan kompresi data gambar menggunakan perbandingan algoritma Stout Code dan Rice Code. Berdasarkan hasil pengujian disimpulkan bahwa algoritma yang lebih bagus untuk mengkompresi file citra adalah algoritma Rice Code dibandingkan dengan algoritma Stout Code. Hasil kompresi menggunakan algoritma Rice Code lebih besar dibandingkan hasil kompresi menggunakan algoritma Stout Cod, dimana Rice Code mendapatkan presentasi space saving sebesar 70.37% sedangkan Stout Code sebesar 66.66%. Abstract Image is defined as a description of the object being observed. The use of image files for the data exchange process is often done, this is because an image can represent certain situations or information. Images consist of a collection of pixel matrices that have three color values for each pixel, including Red Green and Blue or RGB. The problem faced is that the larger the pixel size of an image, the larger the file size that must be stored in the storage space. Limited storage media is an obstacle to managing images. To create a lot of free space and have a data size that is not large on the storage media, compression techniques are needed that reduce the size of the data. This research compresses image data using a comparison of the Stout Code and Rice Code algorithms. Based on the test results, it was concluded that a better algorithm for compressing image files is the Rice Code algorithm compared to the Stout Code algorithm. The compression results using the Rice Code algorithm are greater than the compression results using the Stout Cod algorithm, where Rice Code gets a space saving presentation of 70.37% while Stout Code is 66.66%.

1. PENDAHULUAN

Teknologi pemampatan data atau dikenal dengan kompresi data adalah sebuah cara untuk memadatkan data sehingga hanya memerlukan ruangan penyimpanan lebih kecil. Salah satu kegunaan kompresi adalah agar data yang disimpan tidak terlalu banyak dalam penggunaan media penyimpanan. Data adalah kenyataan dan kejadian yang berisikan fakta. Hasil dari pengolahan data adalah informasi yang dapat disajikan dalam bentuk gambar, teks, suara, maupun video. Tingginya kebutuhan informasi yang dapat diakses, terpercaya, dan cepat, membuat ukuran data sangat mempengaruhi. Semakin besar data, semakin lama pula transfer data dan mengurangi efisiensi waktu memperoleh informasi. Salah satunya adalah file gambar.

Gambar atau citra adalah kombinasi antara titik, garis, bidang, dan warna untuk menciptakan suatu imitasi dari suatu objek—biasanya objek fisik atau manusia. Gambar bisa berwujud dua dimensi,

seperti lukisan, foto, dan berwujud tiga dimensi, seperti patung [1][2]. Citra diartikan sebagai gambaran mengenai objek yang diamati. Penggunaan file gambar untuk proses pertukaran data kerap dilakukan, hal ini karena suatu gambar dapat mempresentasikan keadaan atau informasi tertentu. Gambar terdiri dari kumpulan matriks pixel yang memiliki tiga nilai warna pada setiap pixelnya, antara lain Red Green dan Blue atau RGB [3].

Masalah yang dihadapi adalah semakin besar ukuran pixel suatu gambar maka semakin besar pula ukuran file yang harus disimpan diruang penyimpanan. Keterbatasan media penyimpanan menjadi kendala untuk mengelola suatu gambar atau citra. Data dengan ukuran file gambar yang besar akan membutuhkan waktu penyimpanan lebih lama dibandingkan dengan file gambar yang memiliki ukuran lebih kecil, terkadang ada resiko tidak dapat tertampung pada media penyimpanan dan tidak tersampainya, sehingga akan memperkecil kapasitas kosong dalam memori media penyimpanan, masalah lainnya penggunaan didalam ruangpenyimpananya tidak hanya menyimpan data berupa file gambar, tentu ada juga beberapa file yang dapat berukuran yang lebih besar, sehingga beberapa file tersebut harus dapat saling berbagi ruang penyimpanan tanpa harus menghapus file penting lainnya.

Berdasarkan dari masalah tersebut, maka untuk membuat banyak ruang kosong dan memiliki ukuran data yang tidak besar pada media penyimpanan diperlukan teknik kompresi yang memperkecil ukuran data. Teknik kompresi membutuhkan suatu langkah-langkah penyelesaian yang disebut dengan algoritma [4][5][6]. Saat ini banyak algoritma kompresi yang telah tercipta, sehingga membingungkan peneliti untuk memilih algoritma kompresi yang tepat. Oleh sebab itu, penelitian ini akan membandingkan salah satu algoritma seperti algoritma Stout Code dan Rice Code untuk mengkompresi file gambar.

Algoritma Stout Code adalah suatu prosedur penyelesaian masalah dalam kalimat yang tersusun secara logis dan sistematis untuk transformasi data input menjadi output berupa informasi. Algoritma yang efisien dimana prosesnya dapat diselesaikan secepat mungkin dengan frekuensi perhitungan yang pendek [7]. Sedangkan algoritma Rice Code adalah salah satu algoritma yang dapat digunakan untuk melakukan kompresi data sehingga ukuran data yang dihasilkan menjadi lebih rendah dari ukuran sebenarnya. Dalam algoritma Rice Codes, ada sebuah nilai k yang artinya adalah banyaknya angka 1 pada suffix dari kode terkompresi [8].

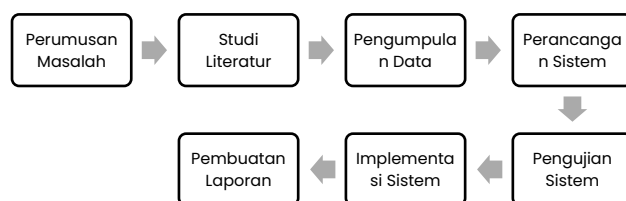
Penelitian yang dilakukan oleh Lamsah dan Dito Putro Utomo pada tahun 2020 dengan judul "Penerapan Algoritma Stout Codes Untuk Kompresi Record Pada Database Di Aplikasi Kumpulan Novel" berhasil menarik kesimpulan bahwa algoritma Stout Codes dengan kompresi file record database berhasil mendapatkan kompresi ukuran file lebih kecil dari ukuran semula dan dapat digunakan sebagai alternatif penghematan media penyimpanan [7]. Penelitian lain yang dilakukan oleh Rosi Tamala Aini Sitanggang pada tahun 2022 dengan judul "Penerapan Algoritma Stout Codes Untuk Kompresi Databases Pada Pinjaman Online" berhasil menarik kesimpulan bahwa algoritma Stout Codes mampu mengecilkan ukuran database yang dikompresi hingga 50% dari ukuran semula [9].

Sedangkan penelitian yang dilakukan oleh Kiki Ramayani pada tahun 2021 dengan judul "Penerapan Algoritma Rice Codes Untuk Mengkompresi File Video" berhasil menarik kesimpulan bahwa algoritma Rice Codes mampu menkompresi sebuah file video dari ukuran semula menjadi ukuran yang lebih kecil hingga 64% [10]. Berdasarkan dari beberapa keberhasilan penelitian terdahulu dengan menerapkan Algoritma Stout Codes dan Rice Codes, maka pada penelitian ini akan melakukan kompresi file gambar dengan membandingkan dua algoritma kompresi. Hal ini bertujuan untuk mendapatkan salah satu algoritma yang optimal dalam melakukan pengecilan ukuran terhadap suatu gambar. Parameter yang digunakan untuk membandingkan kedua algoritma adalah rasio kompresi (RC dan CR) Space saving (SS) serta waktu kompresi.

2. METODE PENELITIAN

2.1 Kerangka Kerja Penelitian

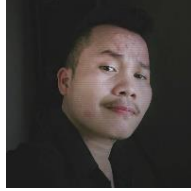
Penyusunan penelitian ini, maka perlu adanya susunan kerangka kerja (frame work) yang jelas tahapan-tahapannya. Kerangka kerja ini merupakan langkah-langkah yang akan dilakukan dalam penyelesaian masalah yang akan dibahas. Adapun kerangka kerja penelitian yang digunakan adalah sebagai berikut:



Gambar 1. Kerangka Kerja Penelitian

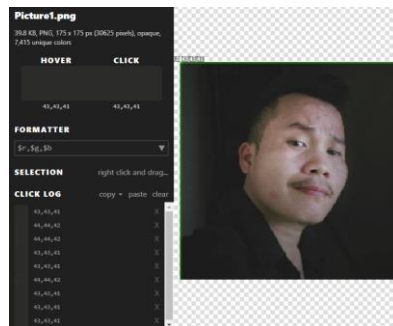
3. HASIL DAN ANALISIS

Analisa dilakukan dengan input data citra yang dijadikan sampel untuk proses perbandingan kompresi menggunakan algoritma stout code dan rice code. Data yang digunakan untuk keperluan hitungan manual perbandingan kompresi menggunakan algoritma *stout code* dan *rice code* adalah sampel citra yang berukuran 175 x 175 pixel. Berikut adalah sampel citra yang digunakan dalam penelitian seperti pada gambar 4.1:



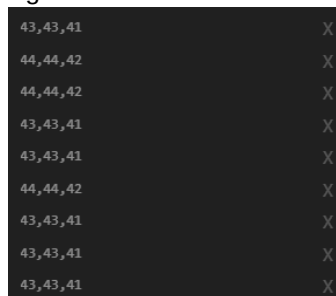
Gambar 2. Citra Sampel

Berdasarkan pada gambar 2, dari ukuran 175 x 175 pixel, diambil nilai pixel citra sebanyak 3x3 pixel untuk diproses menjadi dataset citra kompresi. Adapun nilai pixel sampel 3x3 diambil dari indeks [0] hingga indeks [2] dari citra sampel seperti pada gambar 3:



Gambar 3. Nilai Pixel 3x3 Citra

Berdasarkan pada gambar 3, pengambilan nilai RGB citra sampel 3x3 dilakukan dengan aplikasi pixspys.com dengan hasil seperti pada gambar 4:



Gambar 4. Nilai Citra

Berdasarkan pada gambar 4, berikut adalah tabel nilai pixel citra 3x3 sampel seperti pada tabel 1:

Tabel 1. Nilai Pixel RGB Citra 3x3			
x/y	0	1	2
0	43	44	44
	43	44	44
	41	42	42
1	43	43	44
	43	43	44
	41	41	42
2	43	43	43
	43	43	43
	41	41	41

Pada proses hitungan manual, sebelumnya string nilai pixel dikelompokkan kedalam tabel dengan banyaknya jumlah nilai RGB yang muncul seperti pada tabel 2:

Tabel 2. Urutan Sampel Nilai Citra

<i>Nilai</i>	<i>Frekuensi</i>
43	12
41	6
44	6
42	3
Total	27 Nilai

Berdasarkan pada tabel 2, terdapat 27 nilai citra RB yang akan dikompresi menggunakan algoritma *Stout Code* dan *Rice Code*. Pada dasarnya 1 nilai adalah 8 bit.

3.1. Hasil Kompresi dan Dekompresi Stout Code

Sebelum melakukan kompresi menggunakan algoritma *Stout Code*, nilai pixel yang sudah disusun (tabel 2) terlebih dahulu disusun berdasarkan frekuensi terbesar dan ditentukan banyaknya jumlah bit pada sampel pixel RGB tersebut. Adapun prosesnya dapat dilihat seperti pada tabel 3:

Tabel 3. Urutan Pixel RGB Sesuai Banyaknya Frekuensi

<i>Nilai Pixel RGB</i>	<i>Binner Nilai Pixel RGB</i>	<i>Bit</i>	<i>Frekuensi</i>	<i>Bit * Frek</i>
43	00101011	8	12	96
41	00101001	8	6	48
44	00101100	8	6	48
42	00101010	8	3	24
Total				216 Bit

Berdasarkan nilai tabel 3 di atas, satu nilai terdiri dari 8 bit. Sehingga dengan jumlah nilai sebanyak 27 dikalikan dengan 8 bit, maka total keseluruhan ukuran nilai adalah 216 bit. Setelah nilai bit didapatkan dari semua nilai, selanjutnya adalah melakukan kompresi pertama menggunakan algoritma *Stout Code*. Kompresi dilakukan dengan mengalikan nilai frekuensi dari nilai yang muncul dengan nilai frekuensi bit dari algoritma *Stout Code*. Adapun kode nilai *Stout Code* sesuai dengan jumlah nilai pixel RGB yang akan dikompresi seperti pada tabel 4:

Tabel 4 Kode Nilai Algoritma *Stout Code*

<i>N</i>	<i>Stout Codes S₂ (n)</i>
1	01
2	10
3	11
4	00100

Selanjutnya dilakukan kompresi nilai citra yang sudah disusun berdasarkan pada tabel 3 di atas dengan mengalikan banyaknya frekuensi nilai yang muncul dengan banyaknya nilai bit dari kode nilai algoritma *Stout Code*. Adapun proses keseluruhannya dapat dilihat seperti pada tabel 5:

Tabel 5. Kompresi Berdasarkan Algoritma *Stout Code*

<i>N</i>	<i>Nilia Pixel RGB</i>	<i>Stout Code S₂ (n)</i>	<i>Jumlah Bit Stout Codes</i>	<i>Frekuensi Nilai RGB</i>	<i>Bit * Frek</i>
1	43	01	2	12	24
2	41	10	2	6	12
3	44	11	2	6	12
4	42	00100	5	3	15
Total					63 bit

Berdasarkan pada proses kompresi menggunakan algoritma *Stout Code* tabel 5 di atas, maka untuk mendapatkan nilai baru hasil kompresi, digabungkan nilai kode *Stout Code* yang berurutan dengan nilai pixel sampel awal. Contohnya nilai RGB "43" mendapatkan nilai kode

Stout Code pada tabel 5 yaitu "01". Proses selanjutnya dilakukan dengan cara yang seperti pada tabel 6:

Tabel 6 Susunan Bit Baru Kompresi *Stout Code*

Pixel	Nilai RBG	<i>Stout Code</i>
1	43	01
	43	01
	41	10
2	44	11
	44	11
	42	00100
3	44	11
	44	11
	42	00100
4	43	01
	43	01
	41	10
5	43	01
	43	01
	44	11
6	44	11
	44	11
	42	00100
7	43	01
	43	01
	41	10
8	43	01
	43	01
	41	10
9	43	01
	43	01
	41	10
Total		63 Bit

Kemudian gabungan setiap nilai bit kode *Stout Code* dari atas ke bawah menjadi:
"010110111100100111100100010110010110111100100010110010110010110"

Jumlah bit yang dihasilkan adalah 63 bit. Proses selanjutnya adalah melakukan penambahan *padding* dan *flag bits*. Hal ini dilakukan karena jumlah bit hasil kompresi tidak habis dibagi 8 atau memiliki sisa. Sedangkan *flags bit* adalah nilai biner dari nilai angka *padding*. Karena jumlah bit hasil kompresi adalah 63 bit tidak habis dibagi 8 dan memiliki sisa 7 maka ditambahkan *padding* sebanyak kurangnya sisa yaitu 1 bit nilai biner "0". Dengan demikian, *flags bit*nya adalah nilai biner dari angka 1 (*padding*) yaitu "00000001". Sehingga terbentuk *string* baru hasil kompresi keseluruhannya adalah :

"01011011 11001001 11100100 01011001 01101111 00100010 11001011 00101100
00000001"

Adapun total keseluruhan bit setelah ditambahkan *padding* dan *flag bits* adalah 63 bit + 1 bit (*padding*) + 8 bit (*flag bits*) = 72 bit. Nilai Binner citra hasil kompresi dirubah kembali kedalam bentuk desimal:

01011011 = 91
11001001 = 201
11100100 = 228
01011001 = 89
01101111 = 111
00100010 = 34
11001011 = 203
00101100 = 44
00000001 = 1

Berikut adalah tabel nilai pixel citra 3x3 hasil kompresi *Stout Code* seperti pada tabel 7:

Tabel 7 Nilai Pixel RGB Citra 3x3 Hasil Kompresi *Stout Code*

x/y	0	1	2
0	91	89	203
	201	111	44
	228	34	1
1	0	0	0
	0	0	0
	0	0	0
2	0	0	0
	0	0	0
	0	0	0

Berdasarkan hasil kompresi dengan *Stout Code* pada proses sebelumnya, adapun dapat dihitung proses kinerja kompresinya berupa parameter yaitu :

1. *Ratio of Compression* (R_C)

$$R_C = \frac{\text{Ukuran data sebelum dikompresi}}{\text{Ukuran data setelah dikompresi}} \dots\dots\dots 4.1$$

$$R_C = \frac{216}{72}$$

$$R_C = 3$$

2. *Compression Ratio* (C_R)

$$C_r = \frac{\text{Ukuran data setelah dikompresi}}{\text{Ukuran data sebelum dikompresi}} \times 100\% \dots\dots\dots 4.2$$

$$C_r = \frac{72}{216} \times 100$$

$$C_r = 33.33 \%$$

3. *Space Saving* (S_s)

$$S_s = \frac{\text{Ukuran data sebelum dikompresi} - \text{setelah kompresi}}{\text{Ukuran data sebelum dikompresi}} \times 100\% \dots\dots\dots 4.3$$

$$S_s = \frac{216 - 72}{216} \times 100$$

$$S_s = 66,66 \%$$

Sementara untuk proses dekompresi hal yang dilakukan adalah menganalisa keseluruhan bit hasil dari kompresi sebelumnya menggunakan algoritma *Stout Code*. Ambil keseluruhan bit hasil kompresi citra dengan algoritma *Stout Code* dan gabungkan seperti berikut :

"0101101111001001111001000101100101101111001000101100101100101100 00000001"

Adapun proses dekompresi menggunakan algoritma *Stout Code* sebagai berikut :

1. Tahap Awal

Proses awal adalah membaca *flags bit* dari keseluruhan bit yang sudah digabungkan. Fungsi dari membaca *flags bit* adalah agar dapat mengetahui berapa jumlah *padding* yang dilakukan. Adapun proses membaca *flags bit* seperti yaitu ambil 8 bit nilai akhir dan rubah kedalam desimal:

"010110111100100111100100010110010110111100100010110010110010110000000001".

Didapati 8 bit terakhir 00000001, didalam desimal adalah angka "1". Nilai 1 tersebut merupakan banyaknya *padding* yang dilakukan saat proses kompresi. Selanjutnya hapus *flags bit* sebanyak 8 bit dari urutan terakhir dan hapus 1 bit (*padding*) selanjutnya dari urutan terakhir.

"010110111100100111100100010110010110111100100010110010110010110"

2. Pengecekan Bit

Selanjutnya melakukan pengecekan setiap bit dari indek ke-1. Pembacaan bit dari indeks pertama lalu dicocokkan dengan kode *Stout Code*. Jika terdapat nilai yang sama maka tukar dengan nilai kompresi sebelumnya pada tabel 4.5. Contoh pada indeks ke-1 = 1, nilai 0 tidak ada didalam tabel *Stout Code*, kemudian sambung indeks ke-2 = 01, nilai 01 terdapat pada tabel 4.5 *Stout Code* dengan nilai RGB "43". Kemudian mulai awal dengan indeks ke-3 = 0 dan seterusnya. Proses dilakukan dengan cara yang sama sehingga didapatkan keseluruhan dekompresi hasil pengecekan bit menggunakan nilai kode *Stout Code* seperti pada tabel 8:

Tabel 8 Nilai Pixel RGB Citra 3x3 Hasil Dekompresi *Stout Code*

x/y	0	1	2
0	43	44	44

	43	44	44
	41	42	42
1	43	43	44
	43	43	44
	41	41	42
2	43	43	43
	43	43	43
	41	41	41

3.2. Hasil Kompresi dan Dekompresi Rice Code

Kompresi dilakukan dengan mengalikan nilai frekuensi dari nilai yang muncul dengan nilai frekuensi bit dari algoritma *Rice Code*. Adapun kode nilai *Rice Code* sesuai dengan jumlah nilai pixel RGB yang akan dikompresi adalah seperti pada tabel 9:

Tabel 9 Kode Nilai Algoritma *Rice Code*

<i>N</i>	<i>Rice Codes K0</i>
1	0
2	10
3	110
4	110 1

Selanjutnya dilakukan kompresi nilai citra yang sudah disusun berdasarkan pada tabel 3 di atas dengan mengalikan banyaknya frekuensi nilai yang muncul dengan banyaknya nilai bit dari kode nilai algoritma *Rice Code*. Adapun proses keseluruhannya dapat dilihat seperti pada tabel 10:

Tabel 10 Kompresi Berdasarkan Algoritma *Rice Code*

<i>N</i>	<i>Pixel RGB</i>	<i>Rice Code K0</i>	<i>Jumlah Bit Rice Code</i>	<i>Frekuensi Nilai RGB</i>	<i>Bit * Frek</i>
1	43	0	1	12	12
2	41	10	2	6	12
3	44	110	3	6	18
4	42	110 1	4	3	12
Total					54 bit

Berdasarkan pada proses kompresi menggunakan algoritma *Rice Code* tabel 10 di atas, maka untuk mendapatkan nilai baru hasil kompresi, digabungkan nilai kode *Rice Code* yang berurutan dengan nilai pixel sampel awal. Contohnya nilai RGB “43” mendapatkan nilai kode *Rice Code* pada tabel 10 yaitu “0”. Proses selanjutnya dilakukan dengan cara yang seperti pada tabel 11:

Tabel 11 Susunan Bit Baru Kompresi *Rice Code*

<i>Pixel</i>	<i>Nilai RGB</i>	<i>Rice Code</i>
1	43	0
	43	0
	41	10
2	44	110
	44	110
	42	1101
3	44	110
	44	110
	42	1101
4	43	0
	43	0
	41	10
5	43	0
	43	0

	41	10
	44	110
6	44	110
	42	1101
	43	0
7	43	0
	41	10
	43	0
8	43	0
	41	10
	43	0
9	43	0
	41	10
Total		54 Bit

Kemudian gabungan setiap nilai bit kode *Rice Code* dari atas ke bawah menjadi:
“001011011011011101101101001000101101101101001000100010”
Jumlah bit yang dihasilkan adalah 54 bit. Proses selanjutnya adalah melakukan penambahan *padding* dan *flag bits*. Hal ini dilakukan karena jumlah bit hasil kompresi tidak habis dibagi 8 atau memiliki sisa. Sedangkan *flags bit* adalah nilai biner dari nilai angka *padding*. Karena jumlah bit hasil kompresi adalah 54 bit tidak habis dibagi 8 dan memiliki sisa 6 maka ditambahkan *padding* sebanyak kurangnya sisa yaitu 2 bit nilai biner “00”. Dengan demikian, *flags bitnya* adalah nilai biner dari angka 2 (*padding*) yaitu “00000010”. Sehingga terbentuk *string* baru hasil kompresi keseluruhannya adalah :
“0010110110110111011011010010001011011011010010001000100000000010”

Adapun total keseluruhan bit setelah ditambahkan *padding* dan *flag bits* adalah 54 bit + 2 bit (*padding*) + 8 bit (*flag bits*) = 64 bit. Nilai Binner citra hasil kompresi dirubah kembali kedalam bentuk desimal untuk membuat matriks citra hasil kompresi seperti berikut:

00101101 = 45
10110111 = 183
01101101 = 109
00100010 = 34
11011011 = 219
01001000 = 72
10001000 = 136
00000010 = 2

Berdasarkan pada hasil konversi nilai biner ke desimal citra 3x3 hasil kompresi, nilai pixel citra 3x3 hasil kompresi *Rice Code* seperti pada tabel 12:

Tabel 12 Nilai Pixel RGB Citra 3x3 Hasil Kompresi *Rice Code*

x/y	0	1	2
	45	34	136
0	183	219	2
	109	72	0
	0	0	0
1	0	0	0
	0	0	0
	0	0	0
2	0	0	0
	0	0	0

Berdasarkan hasil kompresi dengan *Rice Code* pada proses sebelumnya, adapun dapat dihitung proses kinerja kompresinya berupa parameter yaitu :

1. *Ratio of Compression* (R_c)

$$R_c = \frac{\text{Ukuran data sebelum dikompresi}}{\text{Ukuran data setelah dikompresi}} \dots\dots\dots 4.4$$

$$R_c = \frac{216}{64}$$

$$R_c = 3.375$$
2. *Compression Ratio* (C_R)

$$Cr = \frac{\text{Ukuran data setelah dikompresi}}{\text{Ukuran data sebelum dikompresi}} \times 100\% \dots\dots\dots 4.5$$

$$Cr = \frac{64}{216} \times 100$$

$$Cr = 29.62 \%$$

3. Space Saving (S_s)

$$Ss = \frac{\text{Ukuran data sebelum dikompresi} - \text{setelah kompresi}}{\text{Ukuran data sebelum dikompresi}} \times 100\% \dots\dots\dots 4.6$$

$$Ss = \frac{216 - 64}{216} \times 100$$

$$Ss = 70,37 \%$$

Pada proses dekompresi hal yang dilakukan adalah menganalisa keseluruhan bit hasil dari kompresi sebelumnya menggunakan algoritma *Rice Code*. Ambil keseluruhan bit hasil kompresi citra dengan algoritma *Rice Code* dan gabungkan seperti berikut :

"001011011011011101101101001000101101101101001000100010000000010"

Adapun proses dekompresi menggunakan algoritma *Rice Code* sebagai berikut :

1. Tahap Awal

Proses awal adalah membaca *flags bit* dari keseluruhan bit yang sudah digabungkan. Fungsi dari membaca *flags bit* adalah agar dapat mengetahui berapa jumlah *padding* yang dilakukan. Adapun proses membaca *flags bit* seperti di bawah ini :

Ambil 8 bit nilai akhir dan rubah kedalam desimal

"001011011011011101101101001000101101101101001000100010000000010". Didapati 8 bit terakhir **00000010**, didalam desimal adalah angka "2". Nilai 2 tersebut merupakan banyaknya *padding* yang dilakukan saat proses kompresi. Selanjutnya hapus *flags bit* sebanyak 8 bit dari urutan terakhir dan hapus 2 bit (*padding*) selanjutnya dari urutan terakhir. sehingga didapatkan keseluruhan bit tanpa *padding* adalah sebagai berikut :

"001011011011011101101101001000101101101101001000100010"

2. Pengecekan Bit

Selanjutnya melakukan pengecekan setiap bit dari indek ke-1. Pembacaan bit dari indeks pertama lalu dicocokkan dengan kode *Rice Code*. Jika terdapat nilai yang sama maka tukar dengan nilai kompresi sebelumnya pada tabel 10. Contoh pada indeks ke-1 = 0, nilai 0 terdapat pada tabel 10 *Rice Code* dengan nilai RGB "43". Kemudian mulai awal dengan indeks ke-2 = 0 dan seterusnya. Proses dilakukan dengan cara yang sama sehingga didapatkan keseluruhan dekompresi hasil pengecekan bit menggunakan nilai kode *Rice Code* seperti pada tabel 13:

Tabel 13 Nilai Pixel RGB Citra 3x3 Hasil Dekompresi Algoritma *Rice Code*

x/y	0	1	2
0	43	44	44
	43	44	44
	41	42	42
1	43	43	44
	43	43	44
	41	41	42
2	43	43	43
	43	43	43
	41	41	41

3.3. Perbandingan Eksponensial

Adapun penerapan dari metode perbandingan yaitu metode *exponential* dapat dilihat sebagai berikut :

$$\text{Total nilai (TNi)} = \sum (V_{ij})B_j$$

$$\text{Total Nilai Stout Code} = \sum (V_{ij})B_j \text{ m } J=1$$

$$= ((V_1 * B_{Cr}) + (V_2 * B_{Cr}) + (V_3 * B_{Cr}))$$

$$= ((3 * 3) + (33.33\% * 3) + (66.66\% * 3))$$

$$= (9 + 99.99 + 199.98) = 308.97$$

$$\text{Total Nilai Rice Code} = \sum (V_{ij})B_j \text{ m } J=1$$

$$= ((V_1 * B_{Cr}) + (V_2 * B_{Cr}) + (V_3 * B_{Cr}))$$

$$= ((3.375 * 3.375) + (29.62\% * 3.375) + (70.37\% * 3.375))$$

$$= (11.39 + 99.96 + 237.49) = 348.84$$

Setelah didapatkan nilai *exponential*, berikut adalah hasil keputusan perbandingan metode *Stout Code* dengan *Rice Code* seperti pada tabel 14:

Tabel 14 Perbandingan Metode *Stout Code* dengan *Rice Code*

Alternatif	SB	SD	Rc	Cr	Ss	Eexponential	Rangking
<i>Stout Code</i>	216 bit	72 bit	3	33.33%	66.66%	308.97	2
<i>Rice Code</i>	216 Bit	64 bit	3.375	29.62%	70.37%	348.84	1

Keterangan :

RC = *Ratio Of Compression*

CR = *Compression Rasio*

SS = *Space Savings*

SB = Nilai bit sebelum dikompresi

SD = Nilai bit sesudah dikompresi

Setelah dilihat dari tabel 14, maka dapat disimpulkan bahwa algoritma yang lebih bagus untuk mengkompresi file citra adalah algoritma *Rice Code* dibandingkan dengan algoritma *Stout Code*. Hasil kompresi menggunakan algoritma *Rice Code* lebih besar dibandingkan hasil kompresi menggunakan algoritma *Stout Code*.

4. KESIMPULAN

Berdasarkan hasil dari implementasi sistem yang telah dilakukan pada bab sebelumnya, dapat diambil kesimpulan bahwa algoritma *Rice Code* lebih unggul untuk mengkompresi citra dengan pengujian parameter yang lebih baik yaitu space saving 59.29% sedangkan *Stout Code* hanya 36.18%. Perbandingan algoritma *Stout Codes* dan *Rice Codes* dilakukan dengan *exponential* berdasarkan nilai dari parameter ratio of compresi *Stout Codes* 1.566 dan *Rice Code* 2.456, compresi ratio *Stout Codes* 63.81% dan *Rice Code* 40.70% sedangkan space saving *Stout Codes* 36.18% dan *Rice Code* 59.29%. Perancangan aplikasi berhasil dilakukan dengan 4 form yaitu form kompresi, form dekompresi, form cara pemakaian dan form tentang penulis.

UCAPAN TERIMAKASIH

Ucapan terima kasih kepada pembimbing dan pendamping skripsi yang telah membantu dalam penyusunan dan perbaikan lewat rekomendasi pada saat seminar. Terima kasih juga kepada sivitas universitas budi darma sehingga saya dapat menyelesaikan perkuliahan dengan baik dan menyanggah gelar sarjana.

REFERENSI

- [1] N. A. Hasibuan, "Analisa Perbandingan Algoritma Huffman Dengan Rice Code Dalam Kompresi File Video," vol. 6, no. November, pp. 159–166, 2022, doi: 10.30865/komik.v6i1.5751.
- [2] H. Edha, S. H. Sitorus, and U. Ristian, "PENERAPAN METODE TRANSFORMASI RUANG WARNA HUE SATURATION INTENSITY (HSI) UNTUK MENDETEKSI KEMATANGAN BUAH MANGGA HARUM MANIS Hendryanto," *J. Komput. dan Apl.*, vol. 8, no. 1, pp. 1–10, 2020.
- [3] W. Mertiana, T. Sardjono, and N. Hikmah, "Peningkatan Kontras Citra Mamografi Digital Dengan Menggunakan Clahe dan Contrast Stretching," *J. Tek. ITS*, vol. 9, no. 2, pp. 222–227, 2020.
- [4] R. Tari, S. D. Nasution, and T. Zebua, "Penerapan Algoritma Arithmetic Coding Untuk Mengkompresi Record Database Pada Aplikasi Ensiklopedia Flora Berbasis Android," *TIN Terap. Inform. Nusantara*, vol. 2, no. 7, pp. 433–444, 2021, [Online]. Available: <http://ejurnal.seminar-id.com/index.php/tin/article/view/950>.
- [5] I. Hasan, N. I. Syahputri, and U. H. Medan, "Analisis Parameter Kompresi Algoritma Elias Omega Code dan Fibonacci Code Pada File Digital," vol. 6341, no. April, pp. 8–22, 2021.
- [6] I. Ihsan and D. P. Utomo, "Analisis Perbandingan Algoritma Even-Rodeh Code Dan Algoritma Subexponential Code Untuk Kompresi File Teks," *KOMIK (Konferensi Nas. Teknol. Inf. dan Komputer)*, vol. 4, no. 1, pp. 223–227, 2020, doi: 10.30865/komik.v4i1.2684.
- [7] Lamsah and D. P. Utomo, "Penerapan Algoritma Stout Codes Untuk Kompresi Record Pada Database Di Aplikasi Kumpulan Novel," *KOMIK (Konferensi Nas. Teknol. Inf. dan Komputer)*, vol. 4, no. 1, pp. 311–314, 2020, doi: 10.30865/komik.v4i1.2710.

- [8] M. A. Latif, S. D. Nasution, and Pristiwanto, "Analisa Perbandingan Algoritma Rice Codes Dengan Algoritma Goldbach Codes Pada Kompresi File Text Menggunakan Metode Exponential," *Maj. Ilm. INTI (Informasi dan Teknol. Ilmiah)*, vol. 13, no. 1, pp. 28–33, 2018.
- [9] R. T. A. Sitanggang, "Penerapan Algoritma Stout Codes Untuk Kompresi Databases Pada Pinjaman Online," *Bull. Inf. Technol.*, vol. 3, no. 3, pp. 189–194, 2022, [Online]. Available: <https://journal.fkpt.org/index.php/BIT/article/view/334>.
- [10] K. Ramayani, "Penerapan Algoritma Rice Codes Untuk Mengkompresi File Video," vol. 5, pp. 186–192, 2021, doi: 10.30865/komik.v5i1.3670.