ANALISIS PERBANDINGAN ALGORITMA RLE DAN HUFFMAN PADA KOMPRESI CITRA

Gilang Rizki Akbar[1], Rosa Andrie Asmara[2], Arief Prasetyo[3]

[1,2,3]Program Studi Teknik Informatika, Jurusan Teknologi Informasi, Politeknik Negeri Malang
[1]gilangxpango@gmail.com, [2]rosa_andrie@polinema.ac.id, [3]arief.prasetyo@polinema.ac.id

***Abstrak*— Dengan seiring meningkatnya kebutuhan akan media penyimpanan citra, penggunaan teknik kompresi menjadi sangat penting. Terdapat berbagai algoritma yang bisa digunakan dalam melakukan kompresi citra diantaranya algoritma RLE dan Huffman. Pada penelitian ini akan dilakukan analisis perbandingan antara kedua algoritma tersebut pada kompresi citra. Analisis yang dilakukan akan membandingkan rasio dan waktu kompresi tiap algoritma pada citra. Hasil analisis yang didapatkan pada penelitian ini dari data set citra bitmap yaitu algoritma huffman memiliki rasio kompresi terbaik dengan nilai rasio kompresi rata – rata sebesar 35,36% dibandingkan dengan algoritma RLE dengan rasio kompresi rata – rata sebesar -18,71 %. Dan hasil analisis waktu kompresi tercepat didapatkan pada kompresi dengan algoritma RLE dengan waktu kompresi rata – rata 730,41 ms jika dibandinkan dengan kompresi menggunakan algoritma Huffman dengan waktu kompresi rata – rata 4656,24 ms**

Kata kunci*— *kompresi citra, run length encoding, RLE, huffman

I. PENDAHULUAN (STYLE: *HEADING 1*)

Perkembangan teknologi informasi semakin cepat. Pada bidang multimedia perkembangan dapat dilihat pada teknologi citra dimana sebuah citra dapat memiliki resolusi yang tinggi. Akibatnya ruang penyimpanan dan transmisi data yang dibutuhkan semakin besar. Sehingga dibutuhkan sebuah metode untuk mengatasi permasalahan tersebut.

Kompresi citra adalah aplikasi kompresi data yang mengkodekan citra asli dengan beberapa bit. Kompresi citra bertujuan meminimalka kebutuhan memori untuk merepresentasikan citra digital. Prinsip umum yang digunakan dalam kompresi citra adalah mengurangi duplikasi data di dalam citra. Sehingga memori atau penyimpanan yang dibutuhkan untuk merepresentasikan citra menjadi lebih sedikit dari pada representasi citra semula[1].

Kompresi citra dibagi menjadi 2 macam yaitu kompresi lossy dan lossless. Kompresi lossy berarti citra yang terkompresi ketika didekompresi kembali hasilnya tidak sama dengan citra awal. Semakin banyak kompresi yang dilakukan berarti semakin banyak pengurangan pada besar citra dan semakin banyak penurunan pada kualitas citra. Sedangkan kompresi lossless tidak kehilangan data citra manapun ketika citra yang dikompresi didekompresi kembali. Namun hasil kompresi yang dicapai oleh kompresi lossless tidak sebaik kompresi lossy[2].

Ada beberapa macam algoritma yang bersifat lossless, di antaranya adalah algoritma Run Length Encoding (RLE) dan Huffman. Algoritma RLE melakukan kompresi citra dengan melakukan pengelompokan banyaknya kemunculan nilai dalam satu baris. Sedangkan algoritma Huffman melakukan kompresi dengan cara tiap nilai dikodekan dengan rangkaian beberapa bit, dimana nilai yang sering muncul dikodekan dengan bit yang pendek dan nilai yang jarang muncul dikodekan dengan rangkaian bit yang panjang. kedua algoritma tersebut memiliki kecepatan waktu dan besar rasio kompresi citra yang berbeda.

Pada penelitiane ini dirancang sebuah aplikasi kompresi citra untuk membandingkan algoritma RLE dan Huffman. Diharapkan dari penlitian ini akan diketahui perbandingan estimasi waktu dan rasio kompresi file antara algoritma RLE dan Huffman.

II. LANDASAN TEORI

A. *Kompresi Citra*

Kompresi mengacu pada pengurangan jumlah data yang digunakan untuk mewakili konten file, gambar atau video tanpa secara berlebihan mengurangi kualitas data asli. Kompresi citra adalah aplikasi kompresi data pada citra digital. Tujuan utama dari kompresi gambar adalah untuk mengurangi redundansi dan ketidakrelevenan yang ada dalam gambar, sehingga dapat disimpan dan ditransfer secara efisien[6]. Sehingga ukuran penyimpanan yang diperlukan untuk menyimpan citra akan berkurang, akibatnya banyaknya jumlah citra yang dapat disimpan akan bertambah dan waktu transfer data citra dapat semakin cepat sehingga menghemat waktu.

B. *Teknik Kompresi Citra*

Terdapat dua teknik yang digunakan dalam melakukan kompresi citra yaitu teknik Lossy compression dan teknik Lossless compression.

1. Lossy Compression Dalam teknik kompresi Lossy, ia mengurangi bit dengan mengenali informasi yang tidak diperlukan dan dengan menghilangkannya[7]. Lossy Compression merupakan kompresi citra di mana hasil dekompresi citra yang terkompresi tidak sama dengan citra aslinya karena ada informasi yang hilang, tetapi masih bisa ditolerir oleh persepsi mata. Mata tidak dapat membedakan perubahan kecil pada gambar. Metode ini menghasilkan rasio kompresi yang lebih tinggi daripada metode lossless. Contoh dari metode lossy compression

diantaranya adalah color redution, chroma subsampling, dan transform coding, seperti transformasi Fourier, Wavelet, dan lain-lain

2. Lossless Compression Dalam teknik kompresi Lossless dengan mengompresi data yaitu ketika melakukan dekompresi, akan menjadi replika data aktual yang sama[7]. Lossless Compression merupakan kompresi citra dimana hasil dekompresi citra yang terkompresi sama dengan citra aslinya, tidak ada informasi yang hilang. Banyak aplikasi yang memerlukan kompresi tanpa cacat, seperti pada radiografi, kompresi citra hasil diagnosa medis atau gambar satelit, dimana kehilangan gambar sekecil apapun menyebabkan hasil yang tak diharapkan. Contoh dari metode lossless compression diantaranya adalah run length encoding, Huffman, Lempel Ziv Welch dan lain – lain.

C. Algoritma RLE

Algoritma RLE atau Run Length Encoding merupakan pengkodean yang mengkodekan sederetan data yang memiliki nilai sama menjadi satu nilai data beserta dengan jumlah kemunculannya. Banyaknya atau panjangnya jumlah piksel secara berurutan yang memiliki nilai data yang sama disebut run length, sedangkan pengkodean untuk kebutuhan kompresinya disebut run length encoding[13].

Algoritma RLE menggunakan pendekatan ruang. Algoritma ini digunakan untuk memapatkan citra yang memiliki kelompok – kelompok piksel yang berderajat keabuaan yang sama. Metode ini menyatakan seluruh baris citra menjadi sebuah baris run, lalu menghitung run-length untuk setiap derajat keabuaan yang berurutan.[8]

Secara umum, algoritma RLE secara optimal digunakan pada file yang memiliki karakter yang cenderung homogen. Karena itu, jika algoritma digunakan secara universal maka perlu dilakukan pengelompokan atau transformasi karakter / simbol yang serupa[9]. Langkah-langkah mengompresi data menggunakan Run Length Encoding adalah sebagai berikut:

1. Mengurangi ukuran data yang mengandung simbol berulang.
2. Simbol berulang ditandai sebagai run
3. 1 run biasanya dikodekan menjadi data 2-byte.
4. Byte kedua mewakili simbol berulang, yang disebut run value.
5. Byte pertama mewakili jumlah simbol, yang disebut run count.
6. RLE cocok untuk mengompresi data teks yang berisi banyak pengulangan simbol
7. Contoh dari penggunaan algoritma ini adalah dimisalkan terdapat teks dengan isinya berupa “AACBBBCCBBBDDDDEEE”. Teks tersebut kemudain di encode dengan algoritma RLE maka akan didapatkan hasil sebagai berikut :
8. Kode = (A, 2) (C, 1) (B, 3) (A, 1) (C, 2) (B, 3) (D, 4) (E, 3).

Dari hasil kompresi file teks yang semula berukuran 18 byte atau menjadi 16 byte. Hasil ini menunjukkan bahwa kompresi pada teks tersebut menjadi 18 / 16 = 1,125 kali lebih kecil.

D. Algoritma Huffman

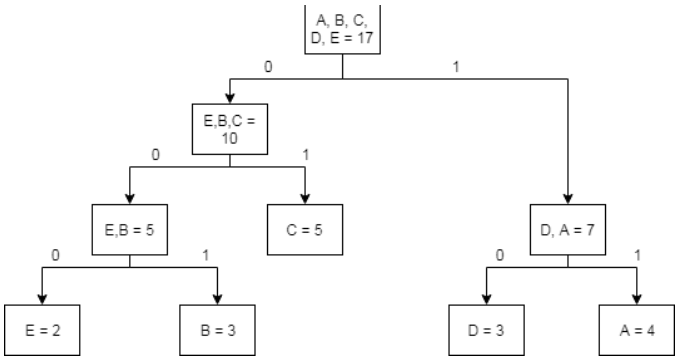
Metode Huffman merupakan salah satu metode dalam teori persandian yang dimanfaatkan dalam proses kompresi data. Kode Huffman diusulkan oleh Dr. David A. Huffman pada tahun 1952 sebagai metode yang ditujukan untuk mengkonstruksi kode redundansi minimum [11]. Konsep dasar dari metode algoritma Huffman adalah dengan membangun skema atau tabel yang berisikan frekuensi kemunculan masing – masing simbol. Dari tabel tersebut kemudian dibangun suatu kode – kode unik untuk mengidentifikasikan masing – masing simbol tersebut[10]. Kode – kode unik direpresentasikan dalam bentuk kode bit dimana kode bit yang lebih pendek digunakan oleh simbol dengan frekuensi terbanyak dan kode bit yang lebih panjang untuk simbol dengan frekuensi terkecil.

Algoritma Huffman memiliki beberapa tahapan dalam proses kompresinya. Cara kerja algoritma Huffman ini adalah sebagai berikut [12]:

1. Menghitung banyaknya jenis karakter dan jumlah dari masing-masing karakter yang terdapat dalam sebuah file.
2. Menyusun setiap jenis karakter dengan urutan jenis karakter yang jumlahnya paling sedikit ke yang jumlahnya paling banyak.
3. Membuat pohon biner berdasarkan urutan karakter dari yang jumlahnya terkecil ke yang terbesar, dan memberi kode untuk tiap karakter.
4. Mengganti data yang ada dengan kode bit berdasarkan pohon biner.
5. Menyimpan jumlah bit untuk kode bit yang terbesar, jenis karakter yang diurutkan dari frekuensi keluarnya terbesar ke terkecil beserta data yang sudah berubah menjadi kode bit sebagai data hasil kompresi.

Contoh kompresi dengan algoritma huffman dengan menggunakan teks. Misalkan sebuah file teks yang isinya ““AABCCEDABABEDCDCC”. File ini memiliki ukuran 17 byte dimana tiap satu karakter sama dengan 1 byte. Berdasarkan dengan cara kerja algoritma Huffman, dapat dilakukan kompresi sebagai berikut :

1. Menghitung banyaknya jumlah tiap karakter yaitu karakter A = 4, B = 3, C = 5, D = 3, E = 2.
2. Mengurutkan karakter dengan jumlah paling sedikit ke jumlah paling besar sehingga didapatkan urutan karakter yaitu E, B, D, A, C
3. Membuat pohon biner berdasarkan urutan karakter dari jumlah kecil ke yang terbesar. Pohon biner dapat dilihat pada gambar 1



Gambar 1. Pohon Biner

4. Mengganti data karakter dengan kode bit berdasarkan pohon biner yang dibuat. Penggantian karakter menjadi kode biner dilihat dari node paling atas atau disebut dengan node akar. Karakter akan diganti menjadi : A = 11, B = 001, C = 01, D = 10, E = 000.
5. Selanjutnya berdasarkan pada kode biner maka semua karakter dalam file dapat diganti menjadi "11110010101000101100111001000 1001100101".

Dari hasil kompresi file teks yang semula berukuran 17 byte atau dalam bit berukuran 17 x 8 = 136 bit menjadi 39 bit. Hasil ini menunjukkan bahwa kompresi file tersebut menjadi 136 / 39 = 3,49 kali lebih kecil.

E. Rasio Kompresi

Rasio kompresi citra adalah rumus yang digunakan untuk menghitung besar hasil kompresi citra dibandingkan dengan besar citra. Semakin besar nilai dari rasio kompresi citra maka semakin kecil hasil dari kompresi yang dihasilkan.

Rumus yang digunakan untuk melakukan perhitungan rasio kompresi citra adalah seperti pada persamaan (1).

$$\boldsymbol{Rasio} = 100\% - \left(\frac{Hasil\ Kompresi}{Citra\ Asli}\ x\ 100\%\right) \quad (1)$$

Keterangan :

- Hasil kompresi : merupakan besarnya ukuran penyimpanan citra setelah terkompresi.
- Citra asli : merupakan besarnya ukuran penyimpanan file citra.

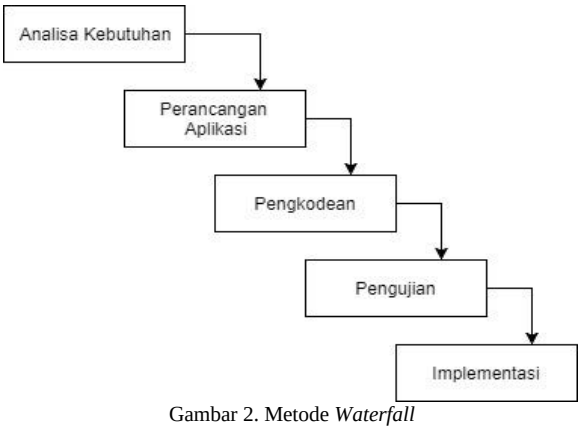
III. METODOLOGI PENELITIAN

A. Metode Pengambilan Data

Tahap pengumpulan data yang digunakan dalam pembuatan aplikasi ini dilakukan melalui studi pustaka yaitu dengan mengumpulkan dan mempelajari sejumlah referensi yang berkaitan dengan judul penelitian.

B. Metde Perancangan Sistem

Metode yang digunakan dalam perancangan aplikasi ini menggunakan metode Waterfall atau metode Air Terjun. Adapun tahap-tahap dalam metode Waterfall seperti yang ditampilkan oleh gambar 2.



Gambar 2. Metode *Waterfall*

IV. ANALISA DAN PERANCANGAN

A. Analisa Kebutuhan

Analisia kebutuhan merupakan suatu penjabaran mengenai komponen-komponen penyusun aplikasi dalam penelitian ini baik perangkat lunak maupun perangkat keras. Untuk mempermudah dalam menentukan kebutuhan aplikasi, maka dalam bagian ini akan ditunjukkan desain model perancangan aplikasi dan struktur implementasi aplikasi.

1) Kebutuhan Perangkat Lunak yang dibutuhkan dalam pembuatan aplikasi ini yaitu seperti pada tabel I.

TABEL I. KEBUTUHAN PERANGKAT LUNAK

Perangkat Lunak	Keterangan
Windows 7	Sistem operasi yang digunakan
Microsoft Visual Studio 2010	Digunakan untuk merancang aplikasi
Microsoft .NET Framework 4.5	Pustaka/library pemrograman

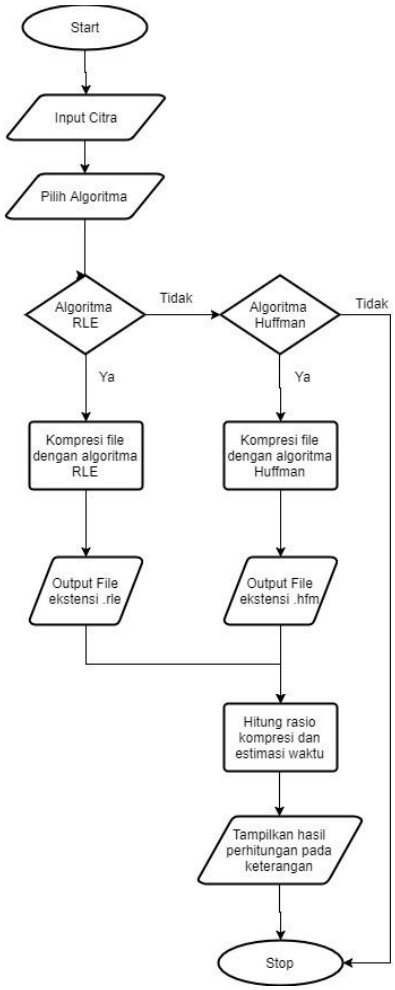
2) Kebutuhan Perangkat Keras yang dibutuhkan untuk penelitian ini yaitu : Laptop dengan spesifikasi seperti pada tabel II.

TABEL II. KEBUTUHAN PERANGKAT KERAS

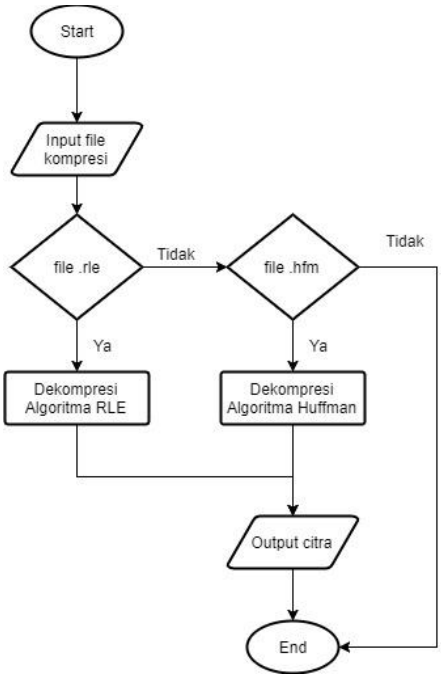
Jenis Perangkat	Spefikasi
Prosesor	Core i3
RAM	4 GB
Hardisk	500 GB

B. Flowchart

Flowchart Merupakan serangkaian bagian-bagian yang berfungsi untuk menerangkan alur dari jalannya aplikasi yang akan dibuat. Pada *flowchart* aplikasi ini ada dua proses yaitu proses kompresi dan proses dekompresi citra. *Flowchart* dari proses kompresi ditampilkan pada gambar 3. Sedangkan *flowchart* proses dekompresi ditampilkan pada gambar 4.



Gambar 3. *Flowchart* proses kompresi

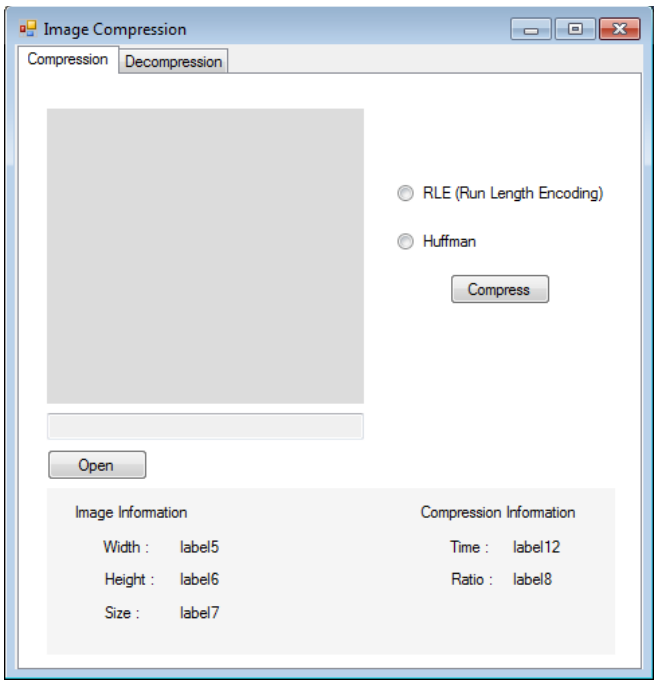


Gambar 4. *Flowchart* proses dekompresi

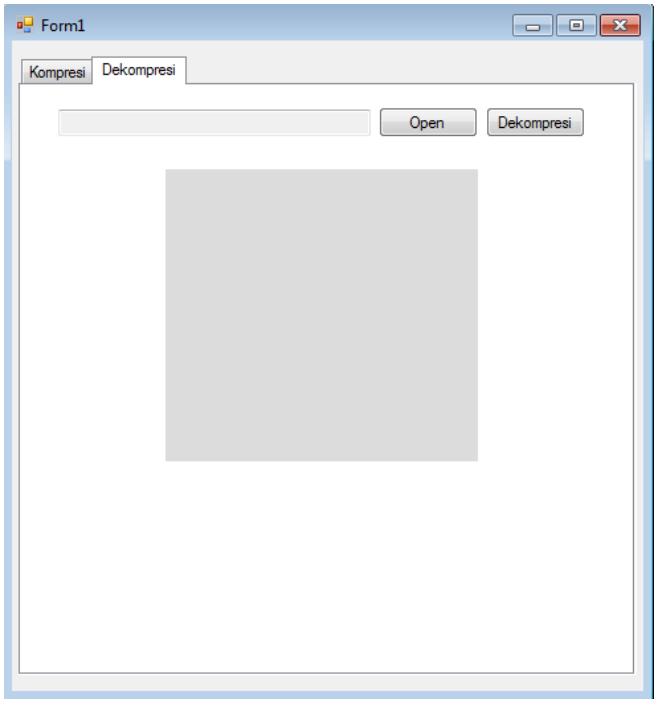
V. IMPLEMENTASI DAN PENGUJIAN

A. Implementasi Antarmuka

Desain antarmuka terdiri dari dua menu yaitu menu kompresi dan menu dekompresi. Antarmuka menu kompresi ditampilkan pada gambar 5. Dan antarmuka menu dekompresi ditampilkan pada gambar.



Gambar 5. Antarmuka Menu Kompresi



Gambar 6. Antarmuka Menu Dekompresi

B. Hasil Pengujian pada Citra

Berikut merupakan hasil dari 17 percobaan proses kompresi file gambar berformat BMP dengan menggunakan algoritma RLE dan Huffman ditampilkan pada tabel III sebagai berikut :

TABEL III. HASIL PENGUJIAN PADA KOMPRESI CITRA

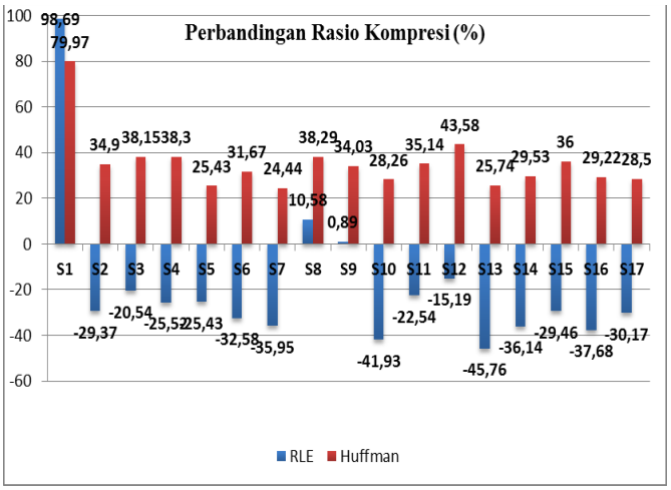
No	*Image* (Citra)	Ukuran	*Size* Awal	Algoritma	Waktu (rata-rata)	*Size* Kompresi	Rasio
1	Sample 1.bmp	256 x 256	257 KB	RLE	36 ms	3,34 KB	98,69%
				Huffman	167 ms	51,2 KB	79,97%
2	Sample 2.bmp	256 x 256	257 KB	RLE	349 ms	331 KB	-29,37 %
				Huffman	1324 ms	166 KB	34,9%
3	Sample 3.bmp	256 x 256	257 KB	RLE	334 ms	308 KB	-20,54 %
				Huffman	1143 ms	158 KB	38,15 %
4	Sample 4.bmp	256 x 256	257 KB	RLE	366 ms	321 KB	-25,52 %
				Huffman	1921 ms	157 KB	38,3%
5	Sample 5.bmp	256 x 256	257 KB	RLE	375 ms	321 KB	-25,43 %
				Huffman	2109 ms	190 KB	25,43%
6	Sample 6.bmp	256 x 256	257 KB	RLE	415 ms	339 KB	-32,58 %
				Huffman	2368 ms	147 KB	31,67%
7	Sample 7.bmp	256 x 256	257 KB	RLE	381 ms	348 KB	-35,95 %
				Huffman	2241 ms	193 KB	24,44%
8	Sample 8.bmp	256 x 256	257 KB	RLE	267 ms	228 KB	10,58%
				Huffman	2070 ms	158 KB	38,29%
9	Sample 9.bmp	256 x 256	257 KB	RLE	294 ms	253 KB	0,89%
				Huffman	1731 ms	168 KB	34,03%
10	Sample 10.bmp	512 x 512	1025 KB	RLE	1176 ms	1,41 MB	-41,93 %
				Huffman	9644ms	734 KB	28,26%
11	Sample 11.bmp	512 x 512	1025 KB	RLE	1334 ms	1,22 MB	-22,54 %
				Huffman	8759 ms	664 KB	35,14%
12	Sample 12.bmp	512 x 512	1025 KB	RLE	1334 ms	1,15 MB	-15,19 %
				Huffman	7563 ms	577 KB	43,58%
13	Sample 13.bmp	512 x 512	1025 KB	RLE	1312 ms	1,45 MB	-45,76 %
				Huffman	8680 ms	760 KB	25,74%
14	Sample 14.bmp	512 x 512	1025 KB	RLE	963 ms	1,36 MB	-36,14 %
				Huffman	8521 ms	721 KB	29,53%
15	Sample 15.bmp	512 x 512	1025 KB	RLE	994 ms	1,29 MB	-29,46 %
				Huffman	5595 ms	655 KB	36%
16	Sample 16.bmp	512 x 512	1025 KB	RLE	1418 ms	1,37 MB	-37,68 %
				Huffman	7513 ms	724 KB	29,22%
17	Sample 17.bmp	512 x 512	1025 KB	RLE	1069 ms	1,30 MB	-30,17 %
				Huffman	7789 ms	732 KB	28,5%

Keterangan tabel pengujian algoritma pada kompresi citra adalah sebagai berikut :

- Pada pengujian sample 1.bmp didapatkan rasio kompresi algoritma yang unggul merupakan algoritma RLE karena citra pada sample 1.bmp memiliki nilai piksel satu deret dengan homogenitas yang tinggi.
- Pada pengujian sample 2.bmp sampai pengujian sample 17.bmp didapatkan rasio kompresi yang unggul merupakan algoritma Huffman karena pada citra sample 2.bmp sampai sample 17.bmp memiliki piksel dengan nilai frekuensi yang tinggi meskipun tidak ada piksel dengan nilai deretan yang sama.
- Pada pengujian sample 1.bmp sampai dengan sample 17.bmp, waktu kompresi yang unggul merupakan algoritma RLE karena kompresi dengan menggunakan algoritma RLE hanya melakukan pencocokan nilai piksel pada deretan yang sama. Sedangkan algoritma Huffman perlu melakukan beberapa tahapan seperti menghitung frekuensi, pengurutan, pembuatan pohon huffman, pemberian kode biner, dan mencocokan piksel denga kode biner yang telah dibuat.

C. Analisa Pengujian Kompresi pada Citra

Dari hasil pengujian algoritma RLE dan Huffman pada citra yang telah dilakukan, maka perbandingan rasio kompresi citra dapat ditunjukkan dalam bentuk grafik seperti yang ditampilkan pada gambar 7.

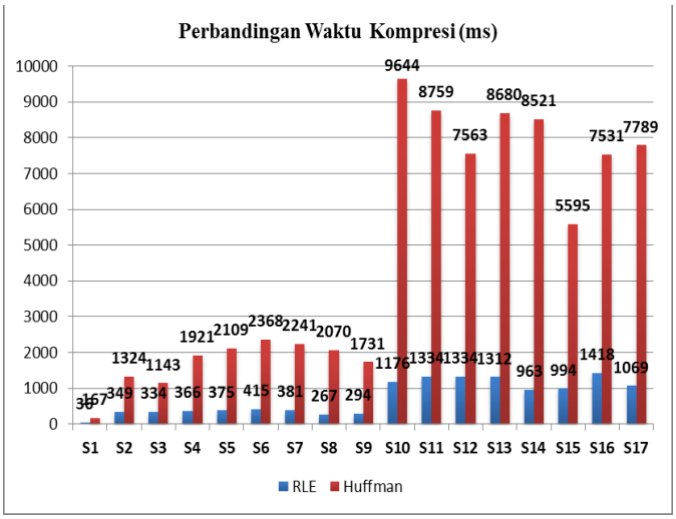


Gambar 7. Perbandingan rasio kompresi (%)

Pada grafik gambar 7 dapat dilihat perbandingan rasio kompresi algoritma RLE dan Huffman pada citra dengan hasil analisa sebagai berikut :

1. Sampel rasio perbandingan kompresi ditunjukkan oleh S1(sample 1.bmp) sampai S9 (sample 9.bmp) untuk sampel citra ukuran 256 x 256 dan S10 (sample 10.bmp) sampai S17 (sample 17.bmp) untuk sampel citra ukuran 512 x 512.
2. Dari hasil pengujian didapatkan algoritma RLE memiliki nilai rasio kompresi rata – rata -18,71 % dengan nilai rasio terbesar 98,69% pada S1 (sample 1.bmp) dan nilai rasio terkecil -45,93% pada S10 (sample 10.bmp).
3. Sedangkan hasil pengujian menggunakan algoritma Huffman didapatkan rasio kompresi rata – rata 35,36% dengan nilai rasio terbesar 79,97% pada S1 (sample 1.bmp) dan nilai rasio terkecil 24,44% pada S7 (sample 7.bmp).
4. Rasio kompresi pada sample 1.bmp dengan menggunakan algoritma RLE memperoleh nila rasio kompresi 98,69% sedangkan dengan menggunakan algoritma Huffman didapatkan rasio kompresi 78,97%. Hal ini karena sample 1.bmp memiliki nilai piksel homogenitas yang tinggi sehingga algoritma RLE lebih unggul.
5. Sedangkan kompresi pada sample 2.bmp sampai dengan sample 17.bmp kompresi dengan menggunakan algoritma Huffman lebih unggul. Hal ini dikarenakan algoritma huffman memiliki keuntungan kompresi pada citra dengan frekuensi piksel yang tinggi meskipun piksel yang sama tidak berdekatan.

Selain itu, dari hasil pengujian juga didapatkan perbandingan estimasi waktu kompresi pada citra antara kompresi dengan algoritma RLE dan algoritma huffman yang ditampilkan pada bentuk grafik seperti pada gambar 8.



Gambar 8. Perbandingan waktu kompresi (ms)

Pada grafik gambar 5.17 dapat dilihat perbandingan waktu kompresi algoritma RLE dan Huffman pada citra.

1. Sampel rasio perbandingan kompresi ditunjukkan oleh S1(sample 1.bmp) sampai S9 (sample 9.bmp) untuk sampel citra ukuran 256 x 256 dan S10 (sample 10.bmp) sampai S17 (sample 17.bmp) untuk sampel citra ukuran 512 x 512.
2. Dari hasil pengujian didapatkan algoritma RLE melakukan kompresi citra dengan waktu rata – rata 730,41 ms dengan waktu tercepat 36 ms didapatkan pada S1 (sample 1.bmp) dan waktu terlambat 2368 ms pada S6 (sample 6.bmp).
3. Sedangkan hasil dari pengujian menggunakan algoritma huffman didapatkan waktu rata – rata kompresi adalah 4656,24 ms dengan waktu tercepat 167 ms yang didapatkan dari S1 (sample 1.bmp) dan waktu paling lambat 9644 ms yang didapatkan dari S10 (sample 10.bmp).

VI. KESIMPULAN

Dari hasil pembahasan Analisis Perbandingan Algoritma RLE dan Huffman, Pada Kompresi Citra, maka di peroleh kesimpulan :

- Kompresi dengan menggunakan Algoritma Huffman memiliki nilai rasio kompresi rata - rata yang lebih baik dibandingkan dengan kompresi yang menggunakan algoritma RLE.
- Algoritma RLE merupakan algoritma dengan waktu kompresi rata- rata tercepat dibandingkan dengan algoritma Huffman.
- Kompresi dengan menggunakan algoritma RLE lebih unggul pada citra dengan sederet piksel bernilai homogen tinggi. Sedangkan kompresi dengan menggunakan algoritma Huffman lebih unggul pada citra dengan piksel berfrekuensi tinggi.

DAFTAR PUSTAKA

[1] Faradisa, Irmalia Suryani dan Budiono, Bara Firmana, "Implementasi Metode Huffman Sebagai Teknik Kompresi Citra", Jurnal Elektro ELEKTEK Vol.2, Institut Teknologi Nasional Malang, 2011

[2] Madhu dan Dalal, Sunil, "Review Paper on Image Compression Using Lossless and Lossy Technique", International Journal of Advance Research, Ideas And Innovations In Technology, Volume 3, Issue 2, 2017

[3] Panduan Penulisan Laporan Akhir dan Skripsi Versi 2.5, Malang : Jurusan Teknologi Informasi Politeknik Negeri Malang, 2018.

[4] T. Sutoyo, Edy Mulyanto, Vincent Suhartono, Oky Dwi Nurhayati, dan Wijanarto, "Teori Pengolahan Citra Digital", Edisi I, Yogyakarta : ANDI, 2009

[5] Bhat, Muzamil, "Digital Image Processing", International Journal Of Scientific And Technology Research, Volume 3, Issue 1, 2014

[6] Sonal Chawla, Meenakshi Beri, Ritu Mudgil, "Image Compression Technique: A Review", International Journal of Computer Science and Mobile Computing, Vol. 3 Issue 8, 2014

[7] Manjari Singh, Sushil Kumar, Siddharth Singh Chouhan, dan Manish Shrivastava, "Various Image Compression Technique: Lossy And Lossless", International Journal of Computer Application, Volume 146 – No.6, 2016

[8] Utari, Cut Try, "Implementasi Algoritma Run Length Encoding Untuk Perancangan Aplikasi Kompresi Dan Dekompresi File Citra", Jurnal Times Vol. 5 No. 2, Universitas Sumetera Utara, 2016

[9] Kenang Eko Prasetyo, Tito Waluyo Purbo, dan Randy Erfa Saputra, "Comparisson of Text Data Compression Using Run Length Encoding, Arithmetic Encoding, Punctured Elias Code and Goldbach Code", International Journal of Engineering and Technology, Vol 9 No. 5, 2017

[10] Andika Satyapratama, Widjianto, Mahmud Yunus, "Analisis Perbandingan Algoritma LZW Dan Huffman Pada Kompresi File Gambar BMP Dan PNG", Jurnal Teknologi Informasi Vol. 6 No. 2. 69, STMIK Pradnya Paramita Malang, 2015

[11] S. A. S. Magi Utomo, L. S. Rahmawati, dan R. Sundari, "Meningkatkan Rasio Kompresi Citra Digital dengan Huffman Coding Pada Transfer Data", SMATIKA Jurnal, Edisi 1, Vol. 4, 2014

[12] Prayoga, Eka dan Suryaningrum, Kristien Margi. "Implementasi Algoritma Huffman Dan Run Length Encoding Pada Aplikasi Kompresi Berbasis Web". Jurnal Ilmiah Teknologi Terapan, Unversitas Widyatama, 2018

[13] Madenda, Sarifudin, "Pengolahan Citra & Video Digital", Jakarta : ERLANGGA, 2015

[14] Putra, Darma, "Pengolahan Citra Digital" , Yogyakarta : ANDI, 2010