Kode Huffman untuk Kompresi Citra pada Navigasi Visual Robot Jelajah

Dion Saputra, 13516045[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13516045@std.stei.itb.ac.id

***Abstract*—Robot technology is one of technology field that has quite influential lately. That is because robots are designed to simplify human tasks. One of the extreme task that given to robot is exploration tasks. Commonly, this type of robot is navigated by recognizing environtment using visual sensor. However, visual image processing requires big memory and long execution time. It a bad thing, because explorer robots have a fast respons to their movement changes. Therefore, the image needs to be compressed. In this paperr, author will discuss about implementation of Huffman code as aone of popular data compression to be implemented in visual navigation system of explorer robots.**

***Keywords*— binary tree, Huffman code, image compression, robot visual navigation.**

I. PENDAHULUAN

Robot merupakan perangkat yang dibuat dengan tujuan untuk memudahkan pekerjaan manusia. Secara sederhana, robot bekerja dengan cara menerima input eksternal, mengolahnya, lalu memberikan output yang berkorespondensi dengan input tersebut. Input yang diterima oleh robot dapat berupa input langsung dari manusia dengan menggunakan perangkat pengontrol robot atau *event* yang diperoleh robot dari lingkungan luar. Input yang diterima robot umumnya dalam bentuk input digital atau analog. Misalnya, melalui perangkat pengontrol, input diberikan dengan memanfaatkan perubahan arus dari perangkat pengontrol berkabel atau melalui transmisi gelombang radio yang diterjemahkan menjadi input digital. Sedangkan untuk meneriman input yang diperoleh dari lingkungan luar, digunakan sensor yang dapat mendeteksi perubahan keadaan lingkungan, lalu mengubahnya menjadi perbedaan tegangan listrik.

Perkembangan teknologi robot dewasa ini cukup pesat. Perkembangan tersebut dimanfaatkan di berbagai bidang, mulai dari automasi di industri, simulasi, riset, edukasi, dan untuk tugas penjelajahan. Robot-robot tersebut memiliki fitur-fitur yang dikembangkan sesuai dengan tujuan pembuatannya. Akibatnya tiap robot memiliki karakteristik masing-masing, baik dari segi ukuran, sumber daya yang digunakan, sistem penerimaan dan pengolahan input, dan kecerdasan buatan yang ditanamkan padanya.

Pada makalah ini, penulis akan membahas mengenai robot jelajah yang difokuskan pada pemanfaatan kompresi citra yang ditangkapnya dengan kode Huffman pada robot tersebut. Robot jelajah merupakan robot yang dibuat dengan tujuan eksplorasi medan-medan khusus di alam. Misalnya robot yang ditujukan untuk melakukan pemetaan suatu wilayah melalui udara, robot yang mempunyai tugas eksplorasi bawah laut, atau bahkan robot NASA yang didesain untuk menjelajah permukaan bulan.

Gambar 1.1 Robot Penjelajah Laut

Sumber : https://daruamanda.wordpress.com/2009/11/16/robot-penjelajah-laut/

Salah satu tantangan yang dihadapi oleh robot jelajah adalah mengenai pengolahan datanya, dimulai dari proses menerima data, mengolahnya, memberikan output, atau mengirimkannya. Robot jelajah yang diterjunkan langsung ke medan-medan alam dilengkapi dengan sensor-sensor dan perangkat elektronik yang presisi, terutama perangkat yang mendukung pengolahan citra digital yang baik. Namun, pengolahan data citra digital tanpa dikompresi sangat menghabiskan daya dan pengolahan datanya tidak dapat diselesaikan dalam waktu yang singkat. Selain itu, jenis robot jelajah tertentu memiliki misi untuk mengirimkan data-data yang diperolehnya ke pusat data menggunakan saluran yang memiliki *bandwidth* kecil [1]. Hal ini tentu saja tidak bagus untuk sebuah robot jelajah yang harus dapat berinteraksi cepat dengan perubahan medan di sekitarnya.

Untuk mengatasi permasalahan tersebut, data-data yang akan diproses oleh robot dan data yang akan dikirimkan di kompresi terlebih dahulu. Salah satu teknik kompresi yang digunakan adalah kompresi data (terutama citra) dengan menggunakan kode Huffman yang memang populer digunakan di berbagai bidang keilmuan saat ini [1].

II. TEORI DASAR

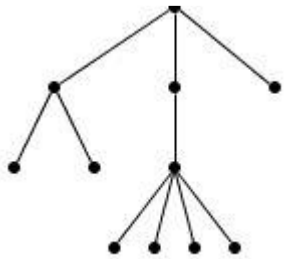
2.1 Pengertian Pohon dan Pohon Biner

Pohon adalah graf tak-berarah terhubung yang tidak mengandung sirkuit. Dalam teori graf, konsep pohon dapat dianggap konsep yang paling penting, karena memiliki terapan yang luas dalam berbagai bidang ilmu pengetahuan [2].

Sifat-sifat berikut dari graf T juga menyebabkan T dikategorikan sebagai pohon. Sifat-sifat tersebut yaitu :

1. T adalah pohon.
2. T tidak memiliki sirkuit dan terdiri atas $n-1$ sisi.
3. T terhubung dan terdiri atas $n-1$ sisi.
4. T terhubung dan setiap sisinya merupakan *cut-set*.
5. Setiap dua titik pada T terhubung oleh tepat satu lintasan.
6. T tidak mengandung sirkuit, namun penambahan satu sisi pada T menghasilkan tepat satu sirkuit [3].

Suatu simpul pada pohon dapat dijadikan simpul khusus yang disebut simpul akar. Pohon yang memiliki simpul akar disebut dengan pohon berakar. Pada pohon berakar semua simpul pada pohon tersebut dapat dicapai dari simpul akar dengan memberikan arah pada sisi dari pohon yang mengikuti akar. Akibatnya, akar memiliki derajat masuk 0 dan simpul selain akar memiliki derajat masuk 1. Sedangkan simpul yang memiliki derajat keluar sama dengan 0 disebut sebagai daun [2].



Gambar 2.1.1 Ilustrasi Pohon Berakar

Sumber : https://i.stack.imgur.com/43Nxe.png

Pada pohon terdapat beberapa terminologi yang biasa digunakan, yaitu :

1. Anak (*child* atau *children*) dan orang tua (*Parent*)
   Simpul y disebut anak dari simpul x dan x disebut orang tua dari y jika terdapat simpul dari x ke y.
2. Lintasan (*path*)
   Lintasan dari simpul v_i ke simpul v_j adalah urutan sisi-sisi $v_i, v_{i+1}, \ldots, v_j$ sedemikian sehingga v_i adalah orang tua dari v_{i+1}.
3. Keturunan (*descendant*) dan leluhur (*ancestor*)
   Simpul y disebut keturunan dari simpul x dan x disebut leluhur dari y jika terdapat lintasan dari x ke y.
4. Saudara kandung (*sibling*)
   Simpul a dan simpul b disebut bersaudara kandung satu sama lain jika orang tua dari simpul a sama dengan orang tua dari simpul b.
5. Upapohon (*subtree*)
   Jika x adalah simpul dalam pohon T, maka upapohon dengan akar x merupakan upagraf $T' = (V', E')$ dimana V' meliputi x dan semua keturunan x dan E' meliputi sisi dalam semua lintasan yang berasal dari x.
6. Derajat (*degree*)
   Derajat dari simpul x merupakan banyaknya anak dari simpul x.
7. Daun (*leaf*)
   Simpul x disebut daun jika derajat dari simpul x adalah 0.
8. Simpul dalam (*internal node*)
   Simpul dalam merupakan simpul yang memiliki anak (derajat simpul > 0).
9. Aras (*level*) atau tingkat
   Aras simpul x didefinisikan sebagai

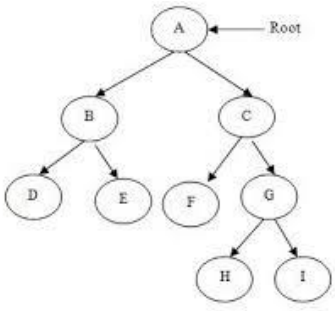
$$\text{Aras}(x) = \begin{cases} 0, & x \text{ adalah akar} \\ 1 + \text{Aras}(y), & y \text{ adalah orang tua } x \end{cases}$$

10. Tinggi (*Height*) atau kedalaman
    Tinggi pohon T merupakan nilai maksimum yang mungkin dari aras x untuk semua x yang merupakan simpul dari pohon T. [2]

Konsep pohon juga digunakan dalam struktur data. Struktur data yang memanfaatkan pohon termasuk struktur data nonlinear. Struktur data pohon ini dapat dimanfaatkan dalam berbagai kasus, diantaranya :

1. Merepresentasikan pohon keluarga
2. Menggambarkan struktur dari algoritma pengambilan keputusan
3. Merepresentasikan struktur data *priority-queue* (*heap*)
4. Penggunaan struktur data pohon di basis data dapat mempercepat akses informasi [4].

Salah satu jenis pohon yang khusus adalah pohon biner. Pohon biner B didefinisikan sebagai pohon berakar yang setiap simpulnya maksimum memiliki 2 anak. Kedua anak pada setiap simpul dari pohon biner dapat dibedakan menjadi anak kiri (*left*) dan anak kanan (*right*). Terminologi yang digunakan untuk pohon biner tidak jauh berbeda dengan terminologi pada pohon berakar.



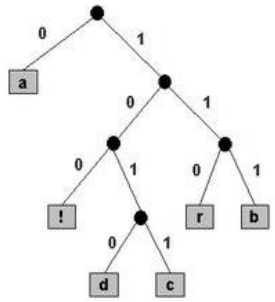
Gambar 2.1.2 Ilustrasi Pohon Biner

Sumber : https://stackoverflow.com/questions/12359660/difference-between-complete-binary-tree-strict-binary-tree-full-binary-tre

2.2 Kode Awalan

Kode awalan (*prefix code*) merupakan himpunan kode, misal kode biner, sedemikian sehingga tidak terdapat anggota himpunan yang merupakan awalan dari anggota lain di himpunan tersebut [2]. Kode awalan umumnya direpresentasikan dengan menggunakan pohon biner dimana

setiap daun dilabeli dengan karakter yang merupakan karakter penyusun dari pesan yang ingin dienkripsikan. Setiap sisi dari pohon biner yang digunakan dilabeli dengan angka 0 atau 1. Pelabelan dapat menggunakan angka 0 untuk sisi ke anak kiri dan 1 untuk sisi ke anak kanan atau sebaliknya, tetapi harus konsisten. Kode awalan terhadap karakter pada daun diperoleh dengan menelusuri lintasan yang dari akar pohon ke daun tersebut dan mengakuisisi kode 0 atau 1 yang ditemui di sepanjang lintasan.[5]



character	encoding
a	0
b	111
c	1011
d	1010
r	110
!	100

Gambar 2.2.1 Penerjemahan Karakter pada Pohon Biner ke Kode Awalan

Sumber : http://www.cs.princeton.edu/courses/archive/spr01/cs126/assignments/prefix.html

Properti dasar yang pertama dari konsep kode awalan ini adalah setiap karakter dapat dienkripsikan dalam jumlah bit yang berbeda. Prinsip dasar yang kedua adalah pesan dapat diterjemahkan langsung menjadi string kode awalan yang bersesuaian. Sebagai contoh, pesan "baca" akan diterjemahkan dengan cara

$$b \mid a \mid c \mid a \;=\; 111 \mid 0 \mid 1011 \mid 0 \;=\; 111010110$$

Prinsip ini dimungkinkan sebab setiap kode dari himpunan tersebut bukan merupakan awalan dari kode lainnya. Hal itu menyebabkan proses dekripsi dari suatu kode awalan menjadi lebih mudah.[5]

Jika terdapat suatu string str yang merupakan hasil enkripsi menggunakan kode awalan dan diberikan sebuah pohon biner B yang merupakan representasi dari kode awalan dengan kode 0 pada anak kiri dan kode 1 pada anak kanan, maka algoritma berikut dapat digunakan untuk mendekripsikan str menjadi pesan semula. Langkah-langkahnya yaitu

1. Mulai dari akar pohon B
2. Lakukan aksi berikut secara berulang hingga tercapai salah satu daun dari pohon B. Proses berulang yang dilakukan adalah
    a. Terima satu bit dari str
    b. Jika bit bernilai 0, lakukan pencarian pada anak kiri. Jika bit bernilai 1, lakukan pencarian pada anak kanan.
3. Jika dicapai daun, tulis/simpan karakter yang ada di daun tersebut.
4. Ulangi algoritma hingga akhir string. [5]

Contohnya, diberikan kode awalan 111010110 dan pohon biner yang ada pada gambar 2.2.1. Cara mendekripsikannya adalah sebagai berikut :

1. Lakukan langkah 1 pada algoritma, penunjuk berada pada akar pohon biner
2. Dengan langkah 2.1 diperoleh bit 1. Dengan langkah 2.2, penunjuk berpindah ke anak kanan dari akar pohon.
3. Aksi 2 diulangi hingga diperoleh karakter "b" pada daun dengan lintasan yang telah ditempuh adalah 1-1-1.
4. Tulis atau simpan karakter "b". Pesan yang diperoleh saat ini adalah "b".
5. Lakukan kembali aksi 1,2,3,dan 4 hingga akhir string. Setiap aksi 1,2,3,dan 4 dilakukan, maka tambahkan karakter yang diakuisisi sebagai karakter terakhir pesan. Proses ini akan membentuk pesan "b" – "ba" – "bac – "baca".

2.3 Kode Huffman

Kode Huffman merupakan salah satu contoh kode awalan yang sangat populer digunakan dalam teknik kompresi data. Metode kompresi data dengan kode Huffman ini adalah dengan mengonversikan simbol-simbol dari data asli yang ingin dikompresi menjadi rangkaian kode bit yang berbeda panjangnya dengan memperhatikan jumlah kemunculan setiap simbol pada data. Proses konversinya menggunakan konsep pohon biner seperti pada kode awalan.

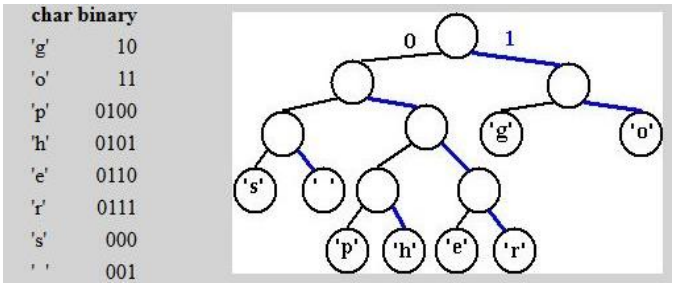
Data yang tidak dikompresi umumnya menggunakan format ASCII yang direpresentasikan oleh string 8 bit. Dengan format seperti ini, sebuah pesan yang terdiri dari 10 karakter memerlukan 80 bit memori. Hal ini berlaku untuk semua karakter yang ada pada pesan tersebut. Ruang memori yang dipakai oleh karakter yang paling sering muncul pada pesan adalah sama dengan ruang memori yang sangat jarang muncul dalam pesan. Dengan menggunakan kode Huffman, karakter yang sangat sering muncul dalam pesan dikodekan dengan bit yang lebih sedikit dibandingkan dengan karakter yang sangat jarang muncul. Hal ini membuat total panjang string bit yang digunakan menjadi jauh lebih sedikit. Sebagai contoh, pesan 10 karakter terdiri atas 9 karakter c1 dan 1 karakter c2. Dengan kode Huffman pesan ini dapat dikompresikan denga $9 \times 1 + 1 \times 1 = 10$ bit string yang 8 kali lebih singkat dan hemat memori dibandingkan metode ASCII.

Algoritma pembentukan kode Huffman dengan memanfaatkan pohon biner adalah sebagai berikut:

1. Daftarkan semua simbol yang ada pada pesan dan hitung peluang kemunculannya. Peluang kemunculan dihitung dengan
$$P(c) = \frac{jumlah\ kemunculan\ c\ pada\ string}{jumlah\ semua\ karakter\ pada\ string}$$
2. Ambil dua buah karakter dengan peluang kemunculan paling sedikit, gabungkan kedua karakter membentuk suatu simbol baru. Tentukan jumlah kemunculan simbol baru ini dengan
$$P(c_1 c_2) = P(c_1) + P(c_2)$$
3. Jadikan simbol baru sebagai orang tua dari karakter penyusunnya pada pohon biner, dengan karakter dengan peluang lebih kecil sebagai anak kiri dan karakter dengan peluang lebih besar sebagai anak kanan, tambahkan simbol baru pada daftar simbol, dan hapus dari daftar simbol karakter pembentuk dari simbol baru tersebut.
4. Ulangi langkah 2 dan 3 hingga diperoleh satu simbol

dengan peluang kemunculan 1 dan sebuah pohon biner (disebut juga pohon Huffman) yang mengakuisisi semua karakter sebagai daun-daunnya.

5. Dimulai dari akar, beri label 0 atau 1 pada sisi-sisi yang menghubungkan orang tua dengan anaknya. Pelabelan harus konsisten seperti halnya pada kode awalan.
6. Kode Huffman diperoleh dengan menelusuri lintasan dari akar pohon biner ke daun pohon biner yang memuat info karakter yang dikodekan. [2]



Gambar 2.3.1 Pohon Huffman dari pesan go go gopher

Sumber : https://www.cs.duke.edu/csed/poop/huff/info/

III. KOMPRESI CITRA DENGAN KODE HUFFMAN

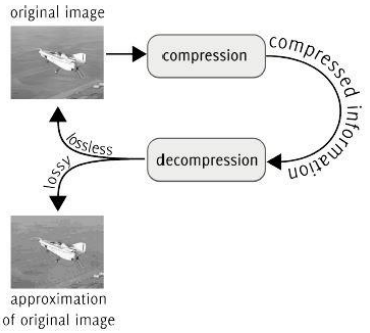
Dalam navigasi robot yang menggunakan citra visual, informasi yang diperoleh untuk melakukan aksi berasal dari citra digital yang ditangkap melalui kamera dan sejenisnya. Informasi ini disimpan dalam bentuk data. Dalam hal ini data dan informasi merupakan dua hal yang berbeda. Data merupakan cara merepresentasikan informasi yang diterima yang kemudian dapat diolah ole prosesor. Data dapat berukuran lebih besar dibandingkan informasi, hal ini dapat disebabkan karena cukup banyaknya data yang mubazir. Semakin banyak data yang mubazir, tidak menjamin bertambahnya informasi yang didapatkan. Oleh sebab itu, data yang mubazir lebih baik tidak disimpan agar dapat mengefisienkan ruang dan waktu pemrosesan data. Teknik yang dapat digunakan untuk membuang data-data yang mubazir tersebut adalah dengan mengompresinya. [6]

Dalam kompresi data, terdapat rasio kompresi yang didefinisikan sebagai berikut. Misalkan n_1 ukuran data dan n_2 ukuran data setelah dikompresi. Maka rasio kompresi dihitung dengan rumus

$$CR = \frac{n_1}{n_2}$$

Untuk gambar, terdapat dua jenis kompresi, yaitu :

1. *Lossless compression*
   Gambar yang telah dikompresi dapat dikembalikan ke gambar semula. Rasio kompresi 2 hingga 10 kali. Kode Huffman termasuk dalam kompresi jenis ini.
2. *Lossy compression*
   Gambar yang telah dikompresi tidak dapat dikembalika ke gambar semula. Rasio kompresi 10 hingga 30 kali. [6]



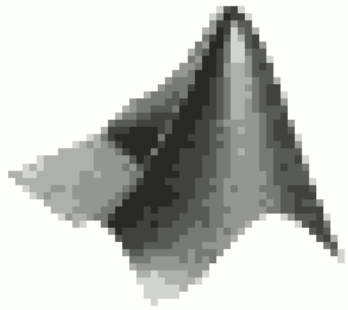
Gambar 3.1 Ilustrasi Lossless Compression dan Lossy Compression

Sumber : Center for Image Analysis, Swedish University of Agricultural Upsala University

Metode kompresi gambar dengan kode Huffman adalah sebagai berikut :

1. Nyatakan tiap sel gambar sesuai dengan level warna pada sel tersebut.
2. Untuk setiap level warna, hitung probabibilitas kemunculan level warna tersebut. Daftarkan dalam sebuah list.
3. Urutkan probabilitas level warna tersebut dari probabilitas terbesar hingga terkecil
4. Untuk dua probabilitas terkecil, digabungkan dan membentuk level baru dengan probabilitas kemunculannya merupakan jumlah dari probabilitas level warna pembentuknya.
5. Dalam list, tambahkan level baru yang diperoleh, dan hapus level warna pembentuknya. Ulangi proses 4 hingga diperoleh dua level warna.
6. Nyatakan level warna dengan probabilitas terbesar dengan bit 0 dan probabilitas terkecil dengan bit 1.
7. Kemudian untuk komponen pembentuk level warna tersebut tambahkan bit 0 atau bit 1 dari kode level warna orang tuanya. [6]

Berikut adalah ilustrasi kompresi gambar menggunakan kode Huffman.



Gambar 3.2 Contoh Gambar yang Akan Dikompresi

Sumber : Center for Image Analysis, Swedish University of Agricultural Upsala University

Tabel 3.1 Data Level Warna Gambar 3.2
Sumber : Center for Image Analysis, Swedish University of Agricultural Upsala University

Level warna	Jumlah muncul	Probabilitas	Bit
0	113	0.051	000
1	139	0.063	001
2	142	0.064	010
3	145	0.066	011
4	181	0.082	100
5	105	0.047	101
6	52	0.023	110
7	1323	0.061	111

Berdasarkan data pada tabel, ruang memori yang dibutuhkan jika citra tidak dikompresi adalah

$$n_1 = \sum_{i=0}^{7} \text{Kemunculan}(i) \times \text{Bit}(i) = 6600 \text{ bit} = 825 \text{ Byte}$$

Jika data dikompresi menggunakan kode Huffman, diperoleh kodefikasi gambar sebagai berikut :

Tabel 3.2 Hasil Kode Huffman
Sumber : Center for Image Analysis, Swedish University of Agricultural Upsala University

Level warna	Jumlah muncul	Probabilitas	Kode Huffman
0	113	0.051	1011
1	139	0.063	1010
2	142	0.064	1001
3	145	0.066	1000
4	181	0.082	110
5	105	0.047	1110
6	52	0.023	1111
7	1323	0.061	0

Ruang memori yang dibutuhkan jika citra dikompresi dengan kode Huffman adalah :

$$n_1 = \sum_{i=0}^{7} \text{Kemunculan}(i) \times \text{Bit}(i) = 4650 \text{ bit} = 581.25 \text{ Byte}$$

dengan

$$CR = \frac{n_1}{n_2} = \frac{825.00 \text{ Byte}}{581.25 \text{ Byte}} = 1.42 \text{ kali}$$

IV. Efek Kompresi Citra pada Pemrosesan Citra Digital Robot

Kompresi citra pada navigasi visual robot memberikan efek yang lebih baik pada sisi konsumsi daya dan waktu pemrosesan citra pada robot jelajah. Hasil ini dibuktikan melalui eksperimen yang dilakukan oleh Yanming Fan, Licheng Wu, dan Xiaer Li dari Minzu University of China.

Eksperimen dilakukan pada robot jelajah bawah air dengan menggunakan algoritma kompresi citra *lossless* LZW dan RLE. Berikut adalah hasil eksperimen yang diperoleh :

Tabel 4.1 : Hasil eksperimen kompresi cirra pada robot jejalah bawah air
Sumber : [7]

Citra	Konsumsi Waktu (s)			Konsumsi Energi (mJ)		
	Non-kompresi	LZW	RLE	Non-kompresi	LZW	RLE
1	8.80	23.50	6.65	336.16	897.70	254.30
2	9.10	19.80	5.45	347.62	756.36	208.20
3	9.00	21.70	6.20	343.80	828.94	236.84
$\overline{x}$	8.97	21.67	6.10	342.53	827.67	233.11

Menurut Yanming Fan, Licheng Wu, dan Xiaer Li konsumsi energi robot berbanding lurus dengan konsumsi energi. Pada kompresi dengan algoritma LZW yang merupakan *dictionary compression algorithm* membutuhkan waktu implementasi yang lama, sehingga konsumsi waktu dan konsumi energi yang dihasilkan lebih boros dibanding citra yang tidak dikompresi. Namun, dengan menggunakan kompresi dengan algoritma RLE, diperoleh konsumsi waktu dan energi waktu yang lebih sedikit. [7]

Dari data eksperimen tersebut dapat dihitung perbandingan waktu yang dikonsumsi antara citra yang dikompresi dengan RLE dan citra yang tidak dikompresi adalah

$$\frac{t(RLE)}{t(Nonkompresi)} = \frac{6.90\ s}{8.97\ s} = 0.77$$

atau proses kompresi RLE lebih cepat sebesar

$$\frac{1}{0.77} = 1.30 \text{ kali}$$

dibandingkan jika tidak dikompresi.

V. Perbandingan Kompresi RLE dan Kode Huffman

Kompresi data RLE dan kode Huffman merupakan algoritma kompresi data yang bersifat *lossless compretion*. Namun, secara umum terdapat keunggulan kompresi data menggunakan Kode Huffman dibandingkan RLE. Hasil ini dibuktikan melalui eksperimen yang dilakukan oleh Dr.Amin Mubark Alamin Ibrahim dan Dr. Mustafa Elgili Mustafa dari Shaqra University, Saudi Arabia.

Eksperimen dilakukan dengan mengimplementasikan algoritma RLE dan Kode Huffman pada bahasa pemrograman C++. Berikut adalah hasil eksperime yang diperoleh :

Tabel 5.1 : Hasil Eksperimen Perbandingan Kompresi RLE dan Kode Huffman
Sumber : [8]

Ukuran file (bit)	RLE (bit)	Kode Huffman (bit)	Ukuran file (bit)	RLE (bit)	Kode Huffman (bit)
9	5	2	54	40	29
12	5	2	57	45	34
15	10	5	60	45	34
18	10	5	63	50	39
21	15	8	66	50	39
24	15	8	69	55	44
27	20	12	72	55	44
30	20	16	75	60	49
33	25	16	78	60	49
36	25	21	81	65	54

39	30	21	84	65	54
42	30	25	87	70	59
45	35	25	90	70	59
48	35	24	93	75	63
51	40	29	96	75	68

Dengan hasil eksperimen tersebut, Dr.Amin Mubark Alamin Ibrahim dan Dr. Mustafa Elgili Mustafa menyimpulkan bahwa algoritma kode Huffman lebih efisien dibandingkan dengan algoritma RLE dimana kode Huffman dapat mengompresi hingga 40% sementara algoritma RLE hanya dapat mengompresi hingga 23%.

Dengan hasil tersebut, dapat disimpulkan bahwa kode Huffman lebih efisien dibandingkan RLE dengan perbandingan

$$\frac{40\%}{23\%} = 1.74 \text{ kali}$$

VI. KESIMPULAN

Berdasarkan hasil eksperimen dari Yanming Fan, Licheng Wu, dan Xiaer Li diperoleh hasil bahwa waktu pemrosesan citra pada robot jelajah bawah air menjadi lebih cepat jika citra dikompresi dengan menggunakan algoritma RLE dengan waktu pemrosesan 1.30 kali lebih cepat dibandingkan jika tidak dikompresi. Selain itu dengan hasil eksperimen oleh Dr.Amin Mubark Alamin Ibrahim dan Dr. Mustafa Elgili Mustafa diperoleh bahwa kompresi dengan menggunakan kode Huffman memiliki efisiensi kompresi sebesar 1.74 kali dibanding algoritma RLE.

Mengingat bahwa kompresi kode Huffman tidak termasuk jenis kompresi data dengan jenis *dictionary compression algorithm* seperti halnya algoritma LZW, maka penulis menyimpulkan bahwa penerapan kompresi citra pada navigasi visual robot jelajah dengan kode Huffman sama atau bahkan lebih efisien dibandingkan kompresi citra dengan RLE. Dengan kesimpulan tersebut, maka pengaplikasian kode Huffman dalam mengompresi citra visual sangat membantu dalam proses navigasi robot jelajah.

VII. UCAPAN TERIMA KASIH

Pertama penulis mengucapkan syukur kepada Tuhan Yang Maha Esa, sehingga dapat menyelesaikan penulisan makalah matematika diksrit ini. Setela itu penulis mengucapkan terima kasih kepada Bapak Rinalid Munir, Ibu Harlili, dan Bapak Judhi Santoso sebagai dosen mata kuliah matematika diskrit di program studi Teknik Informatika Institut Teknologi Bandung yang telah memberikan pengajaran terhadap mata kuliah matematika diskrit di semester ini. Kemudian, penulis juga menyampaikan ucpan terima kasih kepada kedua orang tua penulis, keluarga, dan teman-teman yang telah membantu dari segi nonteknis.

REFERENSI

[1] R. Ponalagusamy, 2007, "A Huffman Decoding Algorithm in Mobile Robot Platform", Asian Network for Scientific Information.

[2] Munir, Rinaldi, 2010, "Matematika Diskrit, Edisi 3", Bandung : Penerbit Informatika.

[3] w3.marietta.edu/~mmm002/Math350Spr06/Lectures/Chapter4.pdf, diakses 2 Desember 2017 pukul 01.00 WIB

[4] http://pages.cs.wisc.edu/~vernon/cs367/notes/8.TREES.html, diakses pada 3 Desember 2017 pukul 10.14 WIB

[5] http://www.cs.princeton.edu/courses/archive/spr01/cs126/assignments/prefix.html, diakses pada 3 Desember 2017 pukul 12.36 WIB

[6] Center for Image Analysis, Swedish University of Agricultural Upsala University, lecture notes.

[7] Y. Fan, L. Wu, X. Li, 2015 *"Low Power Consumption Lossless Image Compression Algortihm and Its Application in Water Strider Robot Vision Systems"*, International Conference on Material, Mechanical and Manufacturing Engineering (IC3ME 2015), submitted for publication

[8] A.M.A. Ibrahim, M.E. Mustafa, *2015, Comparison Between (RLE And Huffman) Algorithms for Lossless Data Compression*, international journal of innovative technology and research, submitted for publication

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 3 Desember 2017

Dion Saputra
13516045