REAL-TIME RENDERING DAN KOMPRESI VIDEO PARALEL MENGGUNAKAN ALGORITMA HUFFMAN

Hanif Wicaksono[1], Fitriyani[2],Izzatul ummah[3]
[1.2.3] Prodi S1 Ilmu Komputasi, Fakultas Informatika, Universitas Telkom
Jalan Telekomunikasi No.1, Dayeuh Kolot, Bandung 40257
hanippon@gmail.com[1],fitriyani@gmail.com[2],izzatul.ummah@gmail.com[3]

Abstrak

Video sebagai salah satu komponen multimedia memegang peranan penting sebagai bentuk informasi visual. Namun kendala terbesar adalah besarnya tempat penyimpanan file. Masalah kedua yang sering dihadapi adalah saat proses kompresi memakan waktu yang cukup lama karena itu digunakan *Parallel computing* untuk menyelesaikannya.

Di dalam tugas akhir ini dibahas penggunaan metode *Huffman Code* untuk kompresi dengan jumlah data yang besar, yaitu pada sebuah *Frame Video* dengan menggunakan nilai intensitas warna pada setiap *pixel* sebagai data dan letak *pixel* sebagai dimensi matriks. Data yang diamati adalah kecepatan proses kompresi, rasio antara jumlah bit sebelum dan sesudah kompresi, dan perbedaannya menggunakan serial (sekuensial) dan *Parallel computing* (paralel).

Hasil menunjukkan bahwa kecepatan proses kompresi untuk 1 *frame* membutuhkan waktu sekitar 8 menit sedangkan proses kompresi untuk 1 *frame* dengan *Parallel computing* membutuhkan waktu sekitar 4 menit. Sedangkan rasio kompresi antara sekuensial dan paralel sama sekitar 89,82% dan 95,71%.

Kata kunci:* *Pemampatan (kompresi), kode huffman, rendering, Paralel, Skuensial

Abstract

Video as one multimedia component plays an important role as a form of visual information. But the biggest obstacle is the large file storage. The second problem is during the compression process takes a long time because it is used parallel computing to solve it.

In this thesis discussed the use of Huffman Code for compression method with a large amount of data, ie on a Frame Video using color intensity values for each pixel as pixel data and layout as the dimensions of the matrix.The data observed is the speed of the compression process, the ratio between the number of bits before and after compression, and the difference using a serial (sequential) and Parallel computing (parallel).

Results showed that the rate of 1 frame compression process takes about 8 minutes while the compression process for 1 frame with Parallel computing takes about 4 minutes. The compression ratio between sequential and parallel at approximately 89,82% and 95,71%.

***Keywords:* *compression, Huffman code, rendering, Parallel, Skuensial*.**

1. Pendahuluan

Video sebagai salah satu komponen multimedia memegang peranan penting sebagai bentuk informasi visual. *Video* mempunyai karakteristik yang tidak dimiliki oleh data teks, yaitu citra kaya akan informasi. Pada umumnya representasi *video* membutuhkan memori yang besar. Semakin besar ukuran citra tentu semakin besar pula memori yang dibutuhkannya. Pada sisi lain, kebanyakan video mengandung duplikasi data. Masalah kedua yang sering dijumpai adalah lamanya proses kompresi.

Saat ini, kebanyakan aplikasi menginginkan representasi video dengan kebutuhan memori yang sesedikit mungkin. Kompresi citra *(image compression)* bertujuan meminimalkan kebutuhan memori untuk merepresentasikan citra digital. Prinsip umum yang digunakan dalam kompresi citra adalah mengurangi duplikasi data di dalam citra sehingga memori yang dibutuhkan untuk merepresentasikan citra menjadi lebih sedikit dari pada representasi citra semula. Kompresi citra memberikan manfaat yang sangat besar dalam industri multimedia saat ini. Salah satu contoh dari kompresi tipe *lossless* adalah metode *Huffman code.*

Oleh karena itu, dengan kebutuhan yang ada saat ini yaitu bagaimana memperkecil ukuran video tanpa menghilangkan informasi di dalamnya, dan untuk mempercepat proses kompresi digunakan *Parallel computing.*

2. Kompresi

2.1 Kompresi Data

Kompresi data merupakan proses untuk mengkodekan informasi dalam bentuk jumlah bit yang lebih rendah daripada representasi data yang tidak terkodekan menggunakan suatu sistem *encoding* untuk mencapai *bitrate* tertentu. Tujuan

dari kompresi data adalah untuk merepresentasikan nilai informasi dalam data digital dengan jumlah bit yang sesedikit mungkin, tetapi tetap mempertahankan nilai informasi di dalamnya.

Tiap metode kompresi memiliki algoritma yang berbeda-beda. Terdapat kriteria dalam algoritma dan aplikasi kompresi data :

❖ Kualitas data hasil *encoding*

Kualitas berhubungan dengan ukuran dari data yang telah terkompresi, semakin kecil semakin baik. Namun hal terpenting adalah data tidak rusak dan nilai informasi di dalamnya tetap terjaga walaupun ada yang dikurangi. Parameter ini biasanya digunakan untuk kompresi dengan metode *lossy*.

❖ Kecepatan *encoding-decoding*

Kecepatan berhubungan dengan waktu yang dibutuhkan untuk mengkodekan informasi (*encoding*) dan mengembalikan informasi ke bentuk awalnya *{decoding).*

❖ Rasio dan efisiensi

Rasio berhubungan dengan perbandingan antara data terkompresi dengan data asli dan seberapa efisien penggunaan metode pengkodean tersebut.

❖ Ketepatan data hasil *decoding*

Ketepatan berhubungan dengan ketepatan hasil dekompresi dengan data aslinya. Parameter ini umumnya digunakan untuk metode kompresi *lossless,* dimana data hasil dekompresi tetap sama dengan data sebelum dikompresi.

Berdasarkan teknik pengkodean/ pengubahan simbol yang digunakan, metode kompresi dapat dibagi ke dalam tiga kategori :

❖ Metode *Symbolwise*

Metode ini menggunakan peluang probabilitas kemunculan suatu karakter (simbol) dari suatu data kemudian memberikan kode untuk tiap-tiap karakter sesuai dengan probabilitas kemunculannya. Simbol yang lebih sering muncul memiliki panjang kode yang lebih singkat. Algoritma *Huffman*menggunakan metode ini

❖ Metode *Dictionary*

Metode ini mengambil karakter (simbol) dari suatu data untuk dikodekan kemudian hasil pengkodean tersebut dimasukkan ke dalam suatu kamus (*Dictionary)* yang berisikan daftar kode untuk masing-masing karakter. Selanjutnya tiap-tiap karakter dikodekan sesuai dengan kode yang ada pada kamus (*Dictionary*) tersebut sesuai indeks lokasinya. Pada pemrograman dengan *tools* Matlab, pengkodean *Huffman* j uga memanfaatkan metode *Dictionary* ini.

❖ Metode *Predictive*

Metode ini menggunakan model *finite-context* atau *finite-state* untuk memprediksi distribusi probabilitas dari simbol-simbol selanjutnya.

Berdasarkan metode algoritma yang digunakan, kompresi diklasifikasikan menjadi dua jenis, yaitu:

- Kompresi *Lossless*

Kompresi jenis ini menghasilkan data yang sudah terkompresi dan dapat dikembalikan ke dalam bentuk semula yang tepat sama dengan data aslinya. Oleh karena itu kompresi jenis ini disebut *reversible,* dua arah (*encoding-decoding*).

- Kompresi *Lossy*

Kompresi jenis ini menghasilkan data yang sudah terkompresi dan tidak dapat dikembalikan lagi menjadi tepat sama dengan data aslinya. Oleh karena itu kompresi*lossy* bersifat *irreversible,* hanya bersifat satu arah dan tidak dapat dikembalikan menjadi tepat sama seperti semula.

2.2 Metode Huffman

Metode pengkodean Huffman merupakan salah satu jenis entropy *encoding*. Metode ini menggunakan prinsip pengkodean yang serupa dengan kode morse, yaitu tiap simbol masukan dikodekan hanya dengan rangkaian beberapa *bit*, dimana karakter yang sering muncul dikodekan dengan rangkaian *bit* yang lebih pendek dibandingkan karakter dengan frekuensi kemunculan lebih sedikit.

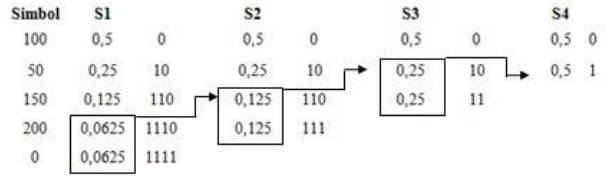
Setiap *pixel* pada *frame* video menyimpan informasi tentang warna (kedalaman dan model warnanya). Bila posisi dari *pixel* (berupa koordinat) dan nilai intensitas warna dari tiap *pixel* dikumpulkan, maka akan diperoleh sebuah matriks yang berisi informasi posisi *pixel* dan intensitas warna pada *pixel* tersebut.

50	100	100	50
100	50	100	0
50	100	150	10
200	100	100	50

Gambar 2.1 Matriks gambar *grayscale* berukuran 4x4 *pixel*

Data *input* yang berupa matriks tersebut diubah terlebih dahulu ke dalam bentuk vektor (matriks 1 baris) kemudian disusun daftar probabilitasnya dan diurutkan dari probabilitas tertinggi ke terendah. Bentuk matriks yang sudah diubah ke dalam bentuk vektor adalah : [50 100 100 50 100 50 100 0 50 100 150 100 200 100 100 50],

Proses pengkodean *Huffman* untuk menentukan label kode *Huffman* dari masing-masing simbol dapat dilakukan setelah diperoleh tabel 2.1. Susunan lengkap pengkodean *Huffman* untuk tiap-tiap simbol dapat dilihat pada gambar 2.4.



Gambar 2.2 Proses pembuatan *Huffman Tree*

Tabel 2.1 *Perubahan simbol menjadi kode Huffman*

Simbol	Frekuensi kemunculan	Probabilitas	Kode Huffman
100	8	8/16 = 0.5	0
50	4	4/16 = 0.25	10
150	2	2/16 = 0.125	110
200	1	1/16 = 0.0625	1110
0	1	1/16 = 0.0625	1111

Vektor dari matriks data *input* kemudian disusun kembali menggunakan kode *Huffman* untuk setiap karakter. Data yang sudah diganti dengan kode *Huffman* kemudian dihitung jumlah bitnya dan dibandingkan dengan jumlah bit pada data *input.* Rasio didapatkan dengan menggunakan persamaan : Rasio = Jumlah bit terkompresi / Jumlah bit asli .

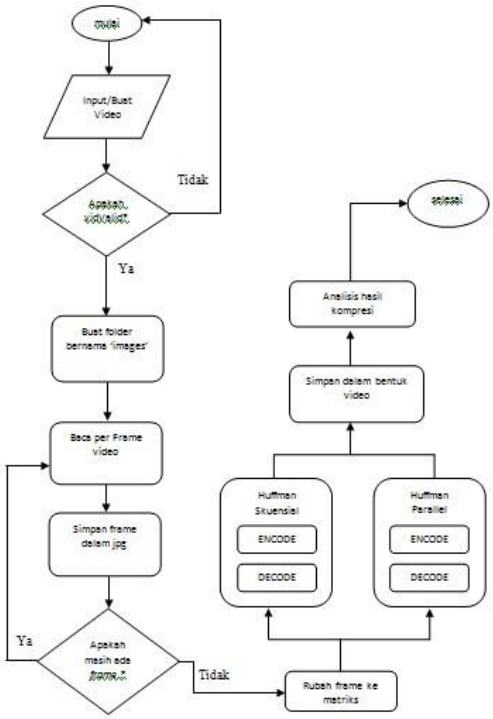
Data *input:*
[50 100 100 50 100 50 100 0 50 100 150 100 200 100 100 50]
Total jumlah elemen dalam vektor *input* : 16.
Kedalamanwarna8-bit (256 wama).
Total jumlah bit pada vektor *input*1 6 x 8 = 128-bit.
Data output:
[1 0 0 0 1 0 0 1 0 0 1 1 1 1 1 0 0 1 1 0 0 1 1 1 0 0 0 1 0]
Total jumlah bit pada vektor *output* : 29-bit. Rasio :29/128= 0,2265625. Persentase Rasio:0,2265625*100%=22,65%. Dekompresi *output*: [10|0|0| 10|0| 10|0| 1111110|0| 110|0| 1110|0|0|10]

Dekompresi menggunakan tabel 2.1 hasilnya sama dengan *input*:
[50 100 100 50 100 50 100 0 50 100 150 100 200 100 100 50],

3. Pemrograman Matlab

Penggunaan bantuan *software* Matlab menjadi penting karena dalam prakteknya, proses *Huffman encoding* pada sebuah *frame* video memiliki kompleksitas (tingkat kerumitan) yang tinggi dan terlalu rumit untuk dilakukan perhitungan secara *manual.*

- Ukuran *frame* dari video mempengaruhi jumlah *pixel* yang dibutuhkan untuk menampilkan *frame* tersebut.
- Kedalaman warna juga mempengaruhi tingkat kerumitan pada proses pengkodean *Huffman.* Tingkat kedalaman warna 8-bit memiliki 256 variasi intensitas warna dan intensitas warna merupakan simbol (karakter) yang akan dikodekan.
- Model warna juga mempengamhi tingkat kerumitan pada proses *Huffman encoding.* Model warna RGB menggunakan 3 matriks yang mewakili setiap elemen warna (Merah, Hijau, Biru).
- Nama Program.
  Program ini bernama "*Real-Time rendering* dan kompresi *video* paralel dengan menggunakan algoritma *Huffman*".
- Fungsi Program.
  Program ini berfungsi untuk membuat pengkodean *Huffman* dari setiap simbol pada sebuah matriks *frame* video dengan *software* matlab.
- Tujuan Program.
  Program ini bertujuan mengurangi jumlah duplikasi data dalam sebuah *frame* dengan metode *Huffman*. Selain itu digunakan juga untuk mengukur



Gambar 3.3 Flowchart diagram sistem

Dalam sistem yang dibangun, akan dilakukan percobaan data dan dianalisis mengenai perbandingan rasio kompresi dan waktu komputasi. Untuk ratio kompresi akan dibandingkan video yang memiliki gambar kompleks (tingkat kerumitan warnanya tinggi) dengan video yang memiliki gambar sederhana(tingkat kerumitan warnanya rendah).

Sedangkan untuk analisis waktu komputasi akan dianalisis perbedaan waktu kompresi saat menggunakan komputasi skuensial dan komputasi paralel.

4. Impelementasi Huffman

Data yang digunakan untuk percobaan ini diambil dari rekaman *video* secara langsung menggunakan alat *webcam* yang tersambung ke PC. *Video* yang diambil untuk percobaan ini sebanyak 2 *video* dengan resolusi 320x240, dengan menggunakan 8 fps(default) dan citra berwarna (8 bit)*RGB (Red,Green,Blue)*. *Video* yang digunakan untuk percobaan ini tanpa suara*(Audio).*

Tabel 4.1 Data Video *yang digunakan*

No	Video	Resolusi	Jumlah frame	Ukuran file
1	Sederhana.avi	320x240	45	11,22 Mb
2	Complex.avi	320x240	45	9,89 Mb

Data yang diambil berupa video *Complex.avi* dan video *sederhana.avi*, video sederhana ini berupa video yang hanya memiliki warna sedikit dan tidak memiliki pergerakan yang banyak. Ilustrasi potongan gambar video „sederhana.avi" terdapat pada gambar di bawah ini.

Gambar 4.1 video sederhana.avi

Untuk video Complex.avi memiliki gambar yang lebih banyak warnanya dan memliki pergerkan yang banyak didalamnya. Ilustrasi potongan gambar video Complex.avi terdapat pada gambar di bawah ini.

Gambar 4.2 Video Complex.avi

4.1 Analisis Waktu

Pada penelitian ini, untuk menentukan waktu komputasi adalah dengan melihat berapa lama waktu dari proses *encode* sampai *decode* dengan membandingkan antara skuensial dan paralel (2, 3, 4 worker) dimana video sederhana.avi dan video Complex.avi di beri 45 frame.

Berikut perbandingan waktu komputasi kompresi *huffman* dari video sederhana dan video complex dengan skuensial dan paralel.

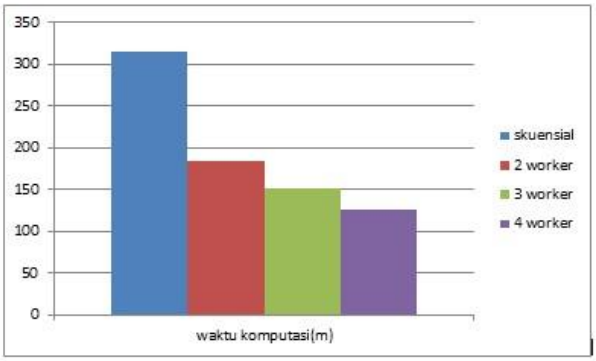
Tabel 4. 2 rata rata waktu komputasi video sederhana.avi

jenis video	bit terkompresi	worker	waktu komputasi(m)
sederhana	74507380	skuensial	310.2377
	74507380	skuensial	316.4584
	74507380	skuensial	319.4568
rata-rata			**315.3843**
sederhana	74507380	2 worker	184.4568
	74507380	2 worker	182.2354
	74507380	2 worker	185.1284
rata-rata			**183.9402**
sederhana	74507380	3 worker	153.6588
	74507380	3 worker	150.6538
	74507380	3 worker	150.1236
rata-rata			**151.4787**
sederhana	74507380	4 worker	128.1235
	74507380	4 worker	125.1236
	74507380	4 worker	126.2347
rata-rata			**126.4939**

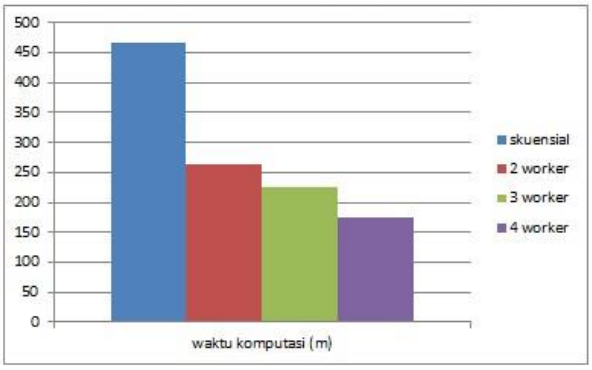
Tabel 4. 3 rata rata waktu komputasi video complex.avi

jenis video	bit terkompresi	worker	waktu komputasi(m)
Complex.avi	79389101	skuensial	465.1593
	79389101	skuensial	468.2356
	79389101	skuensial	463.9523
rata-rata			**465.7824**
Complex.avi	79389101	2 worker	264.1272
	79389101	2 worker	266.1287
	79389101	2 worker	260.7256
rata-rata			**263.6605**
Complex.avi	79389101	3 worker	220.0234
	79389101	3 worker	225.0954
	79389101	3 worker	229.3612
rata-rata			**224.8826**
Complex.avi	79389101	4 worker	172.1693
	79389101	4 worker	178.7632
	79389101	4 worker	170.3478
rata-rata			**173.7601**

Dari hasil percobaan diatas yang dilakukan pada video sederhana.avi menggunakan paralel dengan 2 worker, 3 worker, 4 worker didapatkan hasil waktu rata-rata paling cepat pada proses kompresi menggunakan 4 worker sedangkan pada video complex.avi juga memiliki hasil waktu rata-rata paling cepat pada saat 4 worker. Perbandingan mengenai waktu komputasi antara skuensial dan paralel dapat dilihat



Gambar 4.4 Hasil kompresi video sederhana.avi



Gambar 4.5 Hasil kompresi video complex.avi

4.2Rasio kompresi

Tabel 4.1 Hasil bit kompresi pada video sederhana

frame ke - n	bit asli	bit terkompresi
1	1843200	1594491
2	1843200	1667465
3	1843200	1699236
4	1843200	1697664
5	1843200	1697314
6	1843200	1656078
7	1843200	1657059
8	1843200	1658513
9	1843200	1654715
10	1843200	1655640
11	1843200	1655561
12	1843200	1653650
13	1843200	1655411
14	1843200	1657219
15	1843200	1657949
16	1843200	1658490
17	1843200	1654343
18	1843200	1652408
19	1843200	1653524
20	1843200	1654365
21	1843200	1654791
22	1843200	1655173
23	1843200	1656032
24	1843200	1650693
25	1843200	1649104
26	1843200	1649030

27	1843200	1643533
28	1843200	1649578
29	1843200	1655615
30	1843200	1664065
31	1843200	1665400
32	1843200	1653131
33	1843200	1649020
34	1843200	1652885
35	1843200	1658649
36	1843200	1653340
37	1843200	1647139
38	1843200	1640996
39	1843200	1644092
40	1843200	1650254
41	1843200	1651406
42	1843200	1652950
43	1843200	1657624
44	1843200	1659716
45	1843200	1652069
Jumlah	82944000	74507380

Dari hasil percobaan yang dilakukan diatas dapat dilihat untuk kompresi video sederhana.avi menggunakan metode *Huffman Code* memiliki ratio kompresi sekitar 89,82 % dan bit yang terkompresi 10,18%

Tabel 4.2 Hasil bit kompresi pada video kompleks

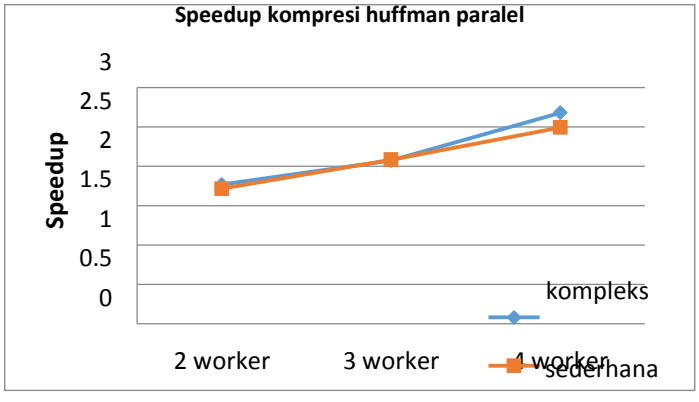
frame ke – n	**bit asli**	**bit terkompresi**
1	1843200	1676889
2	1843200	1748090
3	1843200	1731972
4	1843200	1731921
5	1843200	1724342
6	1843200	1759184
7	1843200	1760123
8	1843200	1764685
9	1843200	1774218
10	1843200	1769100
11	1843200	1756051
12	1843200	1758539
13	1843200	1739368
14	1843200	1730415
15	1843200	1712580
16	1843200	1717061
17	1843200	1714118
18	1843200	1708702
19	1843200	1709091
20	1843200	1711491
21	1843200	1785559
22	1843200	1786249
23	1843200	1789079
24	1843200	1789169
25	1843200	1788197
26	1843200	1786013
27	1843200	1787477
28	1843200	1786913
29	1843200	1788578
30	1843200	1790797
31	1843200	1788459
32	1843200	1789057
33	1843200	1786413
34	1843200	1786507
35	1843200	1787938
36	1843200	1788633
37	1843200	1787881
38	1843200	1784858
39	1843200	1783287
40	1843200	1787325
41	1843200	1785257
42	1843200	1787043
43	1843200	1789215
44	1843200	1790597
45	1843200	1790660
Jumlah	82944000	79389101

Dari hasil percobaan yang dilakukan diatas dapat dilihat untuk kompresi video Complex.avi menggunakan metode *Huffman Code* memiliki ratio kompresi 95,71% dan bit yang terkompresi 4,29%

4.3Evaluasi Performansi

Tabel 4.3hasil evaluasi performansi kompresi video paralel

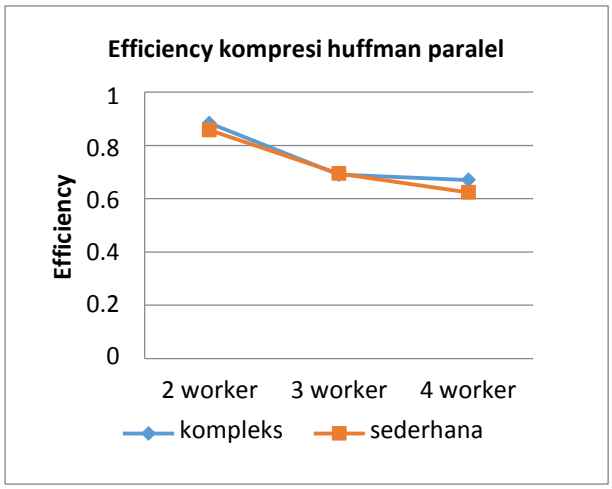
jenis video	rata-rata waktu kompresi skuensial(m)	worker	rata-rata waktu kompresi paralel (m)	speedup	Efficiency
kompleks	465.7824	2	263.6605	1.766599	0.8833
		3	224.8266	2.071741	0.69058
		4	173.7601	2.680606	0.670152
sederhana	315.3843	2	183.9402	1.714602	0.857301
		3	151.4784	2.082041	0.694014
		4	126.4939	2.493277	0.623319



Gambar 4.3speedup pada kompresi Huffman paralel

Berdasarkan penelitian menggunakan video sederhana dan video kompleks yang

memiliki tingkat kerumitan warna yang berbeda didapatkan hasil rata rata tertinggi pada 4 worker di kedua video. Semakin banyak worker yang dipakai semakin cepat proses komputasi untuk kompresinya. Hal ini dikarenakan saat proses kompresi pembagian kerja dikerjakan oleh 4 worker (core) sehingga dapat mempercepat proses komputasinya.



Gambar 4.8 Efficiency pada kompresi Huffman

Dijelaskan pada Gambar 4.8, bahwa 2 worker menghasilkan nilai efisiensi tertinggi dibanding 3 worker dan 4 worker.Hal tersebut membuktikan bahwa semakin banyak worker yang dipakai belum tentu efisien.Karena semakin banyak worker berarti semakin banyak juga biaya yang dikeluarkan. Hal ini membuktikan bahwa untuk 2 worker saja sudah dapat mempercepat proses dengan efisiesn tanpa biaya yang lebih banyak.

5.Kesimpulan

1. Implementasi kompresi metode *Huffman Code* dengan menggunakan MATLAB *Paralel Computing Toolbox* adalah menggunakan core pada CPU.Sehingga didapat waktu kompresi lebih cepat menggunakan paralel daripada skuensial.
2. Dengan Algoritma *Huffman Code* diperoleh hasil kompresi untuk video sederhana.avi (memiliki kerumitan warna yang rendah) sebesar 10,18%dan hasil kompresi untuk video complex.avi (memiliki kerumitan warna yang tinggi) sebesar 4,29%.
3. Untuk proses Algoritma *Huffman Code* dengan cara komputasiparalel pada video sederhana.avi dan complex.avilebih cepat dua kali lipat jika menggunakan 4 worker (core) dibandingkan dengan komputasi skuensial.
4. Pada penelitian kompresi menggunakan metode *Huffman Code* isi video yang digunakan berpengaruh pada lama proses kompresi semakin tinggi kerumitan warna dan banyaknya pergerakan pada tiap frame yang ada pada video semakin lama juga proses kompresi dan semakin kecil bit yang terkompresi dikarenakan jumlah duplikasi datanya sedikit, begitu juga sebaliknya apabila warna yang ada di tiap frame video semakin sedikit semakin cepat juga proses kompresinya dan bit yang terkompresi juga
5. Proses kompresi dan dekompresi membutuhkan kompresi dan sekitar 6-8 menit pada dekompresi tiap *frame.*
6. Ketepatan data yang sudah didekompresi dengan data aslinya.

DAFTAR PUSTAKA

[1] Mauridhi Hery P, Arif Muntasa, "*Konsep pengolahan citra digital dan ekstraksi fitur*" (2010).

[2] T. Sutoyo, Edy Mulyanto, Dr. Vincent Suhartono, Oky Dwi Nurhayati dan Wijanarto. 2005. "*Teori Pengolahan Citra Digital"*. Andi Publisher, Yogyakarta.

[3] Citra *Digital*, ANDI Yogyakarta dan UDINUS Semarang, 2009

[4] Practical Huffman Coding, *http://www.compressconsult.com/huffman/***,** Tanggal akses: Senin, 27 Oktober 2014 pukul 05:50

[5] http://www.geforce.com/hardware/desktop-gpus/geforce-gtx-770/specifications, Tanggal akses: Senin, 27 Oktober 2014 pukul 18.30

[6] Munir Rinaldi 2004. *Pengolahan Citra Digital*. Bandung: Informatika

[7] *Prijono*, *Agus* dan *Marvin Ch*. *Wijaya*. Pengolahan Citra Digital Menggunakan. Matlab Image Processing Toolbox. Cetakan Pertama. 2007.

[8] Irmalia Suryani, Bara Firmana Budiono."Implementasi Metode HUFFMAN Sebagai Teknik Kompresi Citra." 2011.

[9] http://wenythepooh.wordpress.com/2011/02/22/proses-rendering-dan-animasi-serta-contoh-nyatanya, Tanggal akses: Senin, 10 November 2014 pukul 21.00