REAL-TIME RENDERING DAN KOMPRESI VIDEO PARALEL DENGAN MENGGUNAKAN ALGORITMA *RUN LENGTH ENCODING (RLE)*

REAL TIME RENDERING AND PARALLEL VIDEO COMPRESSION USING RUN LENGTH ENCODING (RLE)

Yunasz Praditya[1], Fitriyani[2], Izzatul Ummah[3]

[1,2,3] Prodi S1 Ilmu Komputasi, Fakultas Informatika, Universitas Telkom
[1]yunaszpraditya@telkomuniversity.ac.id, [2]fitriyani@telkomuniversity.ac.id,
[3]izzatulummah@telkomuniversity.ac.id

Abstrak

Pengolahan citra semakin dibutuhkan untuk pengoptimalan waktu dan ukuran sebuah citra *digital* dalam dunia teknologi saat ini. Dengan adanya sebuah algoritma kompresi dapat memperkecil ukuran penyimpanan suatu citra *digital*. Namun, beberapa algoritma yang digunakan memakan banyak waktu untuk menyelesaikan proses kompresi pada sebuah semakin mengoptimalkan waktu kompresi. Penggunaan kompresi pada *real-time rendering and compression* ini menggunakan algoritma *Run Length Encoding* (RLE) karena algoritma ini sangat mudah, dengan prinsip kerjanya yaitu mencari piksel citra *digital* yang bertetanggaan dan menghitung jumlah panjang piksel sama yang bertetanggan. Hasil pengujian yang diharapkan yaitu *real-time rendering and compression* menggunakan algoritma kompresi RLE melalui *parallel computing* akan mengoptimalkan waktu dan ukuran data pada hasil kompresi.

Kata Kunci : Kompresi, Run Length Encoding (*RLE*), Citra digital, Parallel Computing, Rendering

Abstract

Image processing are increasingly required to optimize timing and size of a digital image in nowadays technological era. Given a compression algorithm can reduce the storage size of a digital image. However, some of the used algorithms take a lot of time for completing process of compression in a digital image. Digital image processing by using parallel computing will further optimize the compression time. The use of compression in real-time rendering and compression uses algorithms Run Length Encoding (RLE) because the algorithm is very easy, with the working principle is to look for neighboring pixel digital image and count the number of pixels equal length which neighbored. Test results are expected, namely real-time rendering and compression using RLE compression algorithm through parallel computing will optimize the timing and sizing of data on the results of the compression.

Keywords : Compression, Run Length Encoding (RLE), Digital Image, Parallel Computing, Rendering

1. Pendahuluan

Para ahli mengembangkan proses *rendering* dan kompresi yang tidak memakan waktu lama dan hasil ukuran data yang tidak besar.

Ada dua sifat untuk algoritma data kompresi yang pertama adalah *lossless*, algoritma yang bersifat *lossless* ini dengan membuang informasi data yang berlebihan. Sifat algoritma data kompresi yang kedua adalah *lossy*, membuang data yang tidak sesuai dan membangun kembali sesuai data yang asli.

Algoritma yang bersifat *lossless* kompresi ini sendiri terdiri dari 2 kategori: *Dictionary-Based techniques* dan *statistical method*. Untuk *dictionary-based techniques* ini diantaranya yaitu: *Run-Length Encoding*, dan *LZW Encoding*. Untuk *statistical encoding methods* sendiri diantaranya yaitu: *The information content of a Message*, *Huffman Coding*, *Binary Image Compression Standards*.

Pada proses *real-time rendering and compression* ini sudah pasti akan memakan waktu lama karena proses *rendering* akan dilakukan bersamaan dengan kompresi. Dimana pada proses *rendering* saja akan memakan waktu yang lama. Maka untuk proses *rendering* dan kompresi ini akan digunakan *parallel computing* untuk mempercepat proses *rendering* dan hasil ukuran data yang kecil pada proses kompresi. Apabila proses kompresi algoritma *Run Length Encoding* (RLE) menggunakan *single processor* dibandingkan dengan *quad-core processor* pada *i7* proses yang lebih cepat waktu kinerjanya adalah yang menggunakan *single processor*[13].

Maka dari itu tujuan percobaan *real-time rendering and compression parallel video* ini yaitu untuk mengetahui implementasi *real-time rendering and compression* video dengan *parallel computing*, mengetahui perbandingan optimasi waktu dan hasil kompresi video dengan *parallel computing* dan *squential* pada *real-time rendering and compression*, dan mengetahui *speedup*, *improvement performancy*, *efficiency* kompresi RLE menggunakan parallel computing.

2. Tinjauan Pustaka

2.1 Teori Dasar Citra

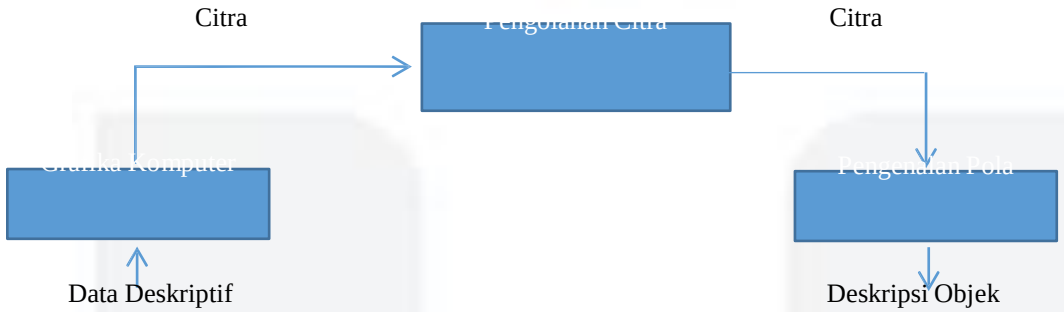
Citra adalah suatu gambar pada bidang dua dimensi. Citra merupakan fungsi *continue* dari intensitas cahaya pada bidang dua dimensi apabila dilihat dari sudut pandang matematis. Citra yang keluar dari suatu sistem perekaman bisa bersifat [2]:

a) *Optic* berupa foto
b) *Analog* berupa sinyal *video* seperti gambar pada monitor
c) *Digital* yang dapat langsung disimpan pada suatu pita *magnetic*

Data citra memiliki tiga bidang studi pada bidang komputer, namun ketiganya memiliki tujuan yang berbeda, yaitu:

a) Grafika komputer (komputer grafik)
b) Pengolahan citra (*image processing*)
c) Pengenalan pola (*pattern recognition*)

Dari tiga bidang studi data citra tersebut bisa dibuat suatu pola keterikatan yaitu:



Gambar 2.1 Pola Keterikatan Data Citra

Sudah banyak metode-metode kompresi citra yang ditemuka saat ini. Terdapat kriteria dalam pengukuran metode kompresi citra, yaitu [1]:

a) Waktu kompresi dan dekompresi
b) Kebutuhan memori
c) Kualitas kompresi (*fidelity*)
d) Format keluaran

Piksel memiliki nilai dalam rentang tertentu dengan memiliki nilai minimum dan nilai maksimum. Pada umum nya nilainya adalah 0-255. Citra dengan penggambaran seperti ini digolongkan ke dalam citra integer. Berikut adalah jenis-jenis citra berdasarkan nilai pikselnya.

a) Citra biner
   Citra biner ini hanya memiliki 2 kemungkinan nilai piksel yaitu hitam dan putih.
b) Citra *grayscale*
   Citra *grayscale* ini yang nilai bagian *Red = Green = Blue*.
c) Citra warna (8 bit)Citra warna (8 bit) ini hanya diwakili oleh 8 bit dengan jumlah warna maksimum adalah 256 warna.
d) Citra warna (16 bit)
   Citra warna 16 bit ini biasa disebut juga sebaga citra *highcolor* dengan setiap pikselnya diwakili dengan 2 byte memori (16 bit).
e) Citra warna (24 bit)
   Setiap piksel dari citra warna 24 bit diwakili dengan 24 bit sehingga total variasi warnanya adalah 16.777.216.

2.2 Format file citra

Ada dua jenis format file citra yang sering digunakan dalam pengolahan citra, yaitu bitmap dan citra *vector*.

a) Format *file* citra bitmap
   Pada citra bitmap menyimpan data kode citra secara *digital* dan lengkap (cara penyimpanannya adalah perpiksel). Beberapa format umum yang digunakan dalam pemrograman pengolahan citra adalah GIF

(*Graphic Interchange Format*), PNG (*Portable Network Graphics*), JPEG (*Joint Photographic Experts Group*)

b) Format file citra *vector*

Citra *vector* tidak berdasarkan piksel tapi dari perhitungan matematis, yaitu data yang tersimpan dalam bentuk vector posisi, di mana yang tersimpan hanya informasi *vector* posisi dengan bentuk sebuah fungsi. Yang termasuk dalam format ini adalah *AutoCAD Drawing Format (DWG), AutoCAD Drawing Exchange format (DXF), Microstation Drawing Format (DGN), dan Scalable Vector (SVG)*.

2.3 *Rendering*

Rendering dimulai dengan memberikan bayangan dan tekstur objek dan pencahayaan, dan proses rendering berakhir ketika material, cahaya, dan gerakan telah di proses menjadi citra akhir atau rangkaian citra.

2.4 Data Kompresi

Data kompresi ini memanfaatkan data yang berlebihan dan tidak sesuai tersebut dengan mengubah data file menjadi lebih kecil dari data aslinya lalu data tersebut di bangun ulang dengan menghasilkan sesuai dengan data aslinya kembali.

Ada 2 sifat untuk algoritma data kompresi yang pertama adalah *lossless*, algoritma yang bersifat *lossless* ini dengan membuang informasi data yang berlebihan, lalu satu informasi data yang ada bisa di kembalikan pada saat de*compression*. Sifat algoritma data kompresi yang kedua adalah *lossy*, algoritma yang bersifat *lossy* ini dengan membuang data yang tidak sesuai dan membangun kembali sesuai data yang asli.

2.5 *RLE (Run Length Encoding)*

RLE adalah kompresi yang umum digunakan pada citra. Kompresi RLE ini didasarkan dengan melihat pengulangan simbol dan melihat panjang simbol yang memiliki pengulangan. Bila suatu citra biner memiliki 30 piksel warna putih, lalu ada 20 piksel warna hitam, kemudian diikuti lagi 10 piksel dengan warna putih. Maka hanya nilai 30, 20, dan 10 yang mennjadi output untuk mewakili setiap pikselnya. Contoh kompresi algoritma RLE **[1]**.

Tinjau citra 10x10 piksel dengan 8 bit *grayscale* yang dinyatakan sebagai matriks

0	0	0	0	0	2	2	2	2	2
0	0	0	1	1	1	1	2	2	2
1	1	1	1	1	1	1	1	1	1
4	4	4	4	3	3	3	3	2	2
3	3	3	5	5	7	7	7	7	6
2	2	6	0	0	0	0	1	1	0
3	3	4	4	3	2	2	2	1	1
0	0	0	0	0	0	0	0	1	1
1	1	1	1	0	0	0	2	2	2
3	3	3	2	2	2	1	1	1	1

Semuanya ada 100 buah piksel. Hasil kompresinya akan menjadi (*p*,*q*) dimana *p* adalah sebagai nilai intensitas pikselnya dan *q* adalah sebagai jumlah intensitas piksel yang bertetanggaan.Pasangan nilai setiap baris *run* yang dihasilkan pada contoh di atas setelah dikompresi menggunakan metode RLE.

Semuanya ada 31 pasangan nilai atau nilai. Ukuran citra sebelum kompresi (1 piksel = 3 bit) adalah bit, sedangkan ukuran citra setelah kompresi adalah (1 derajat keabuan =3 bit, *run length* = 4 bit):

( ) ( )

Maka bisa dihitung citra yang berhasil dikompresi dengan rumus:

$$\left(100\% - \frac{217}{300}\right) \times 100\% = 27{,}67\%$$

Contoh lain kompresi menggunakan algoritma RLE suatu citra *grayscale* dengan 8-bit tanpa menggunakan matriks:

12,12,12,12,12,35,76,76,76,112,112,112,87,5,1,1,1,1

Maka hasil kompresinya menjadi:

5,12,35,3,76,3,112,87,5,4,1

Catatan: Untuk yang digarisbawah menyatakan banyak piksel dengan nilai piksel adalah nilai selanjutnya.
Pada kompresi RLE ini semakin *heterogen* (detail) citra yang di kompres semakin jelek hasil kompresinya dan apabila semakin *homogen* citra yang dikompres semakin baik pula hasil kompresinya.

Permasalahan yang biasa dihadapi pada algoritma RLE ini adalah pembeda angka yang menyatakan banyaknya nilai piksel yang ada dengan piksel itu sendiri.

Beberapa langkah yang dilakukan untuk menyelesaikan permasalahan yang ada di atas, yaitu:

a) Jika citra *grayscale* memiliki nilai maksimum 256 bit, maka nilai tersebut dapat dikurangi menjadi 255 dengan 1 nilai digunakan untuk penanda.
b) Solusi selanjutnya adalah 1 bit dari setiap byte disiapkan sebagai penunjuk byte tersebut merupakan merupakan nilai intensitas atau nilai banyak nya piksel. Pada solusi ini 1 bit ekstra tersebut disisipkan ke dalam deret 8 bit. Deret bit ekstra ini kemudian diikuti atau didahului oleh nilai yang menyatakan intensitas dan banyaknya piksel. Untuk hasil kompresi ini akan naik 12.5% karena ukuran bit ekstra adalah 1/8 dari total outputnya.
c) Menggunakan nilai negative dari selisih antara intensitas piksel dengan banyak nya jumlah piksel yang ditempatkan deretan intensitas piksel dan banyak pikselnya.

Untuk citra berwarna (RGB) kompresi RLE ini dilakukan terpisah pada masing-masing komponen warna (R,G, dan B). Algoritma RLE ini cocok untuk citra biner, seringkali untuk citra *grayscale*, citra warna hasil kompresinya akan lebih besar dibandingkan sebelum terkompresi.

2.6 Parallel Computing Toolbox

Parallel Computing Toolbox merupakan metode yang dikembangkan pada program MATLAB dan sangat efisien dalam melakukan *parallel task processing* atau biasa disebut *parfor*. Meskipun metode *parallel task processing* merupakan metode yang lebih mengarah kepada *speed-ing Central Processing Unit* (CPU) daripada *Graphic Processing Unit* (GPU), tetapi apabila dikombinasikan degan GPU maka waktu komputasi yang didapat akan lebih optimal.

2.7 Evaluasi Performansi

Evaluasi dari performansi menggunakan *speedup* dan *efficiency* berdasarkan waktu komputasi dari kompresi sekuensial dan paralel.

Speedup mengukur seberapa cepat waktu komputasi algoritma paralel dibandingkan dengan algoritma sekuensial.

$$Speedup = \frac{(\quad)}{(\quad)} \qquad (1)$$

Efficiency digunakan untuk mengestimasi seberapa baik proses pemanfaatan *processors* dalam menyelesaikan masalah dibandingkan dengan usaha yang digunakan dalam berkomunikasi dan sinkronisasi.

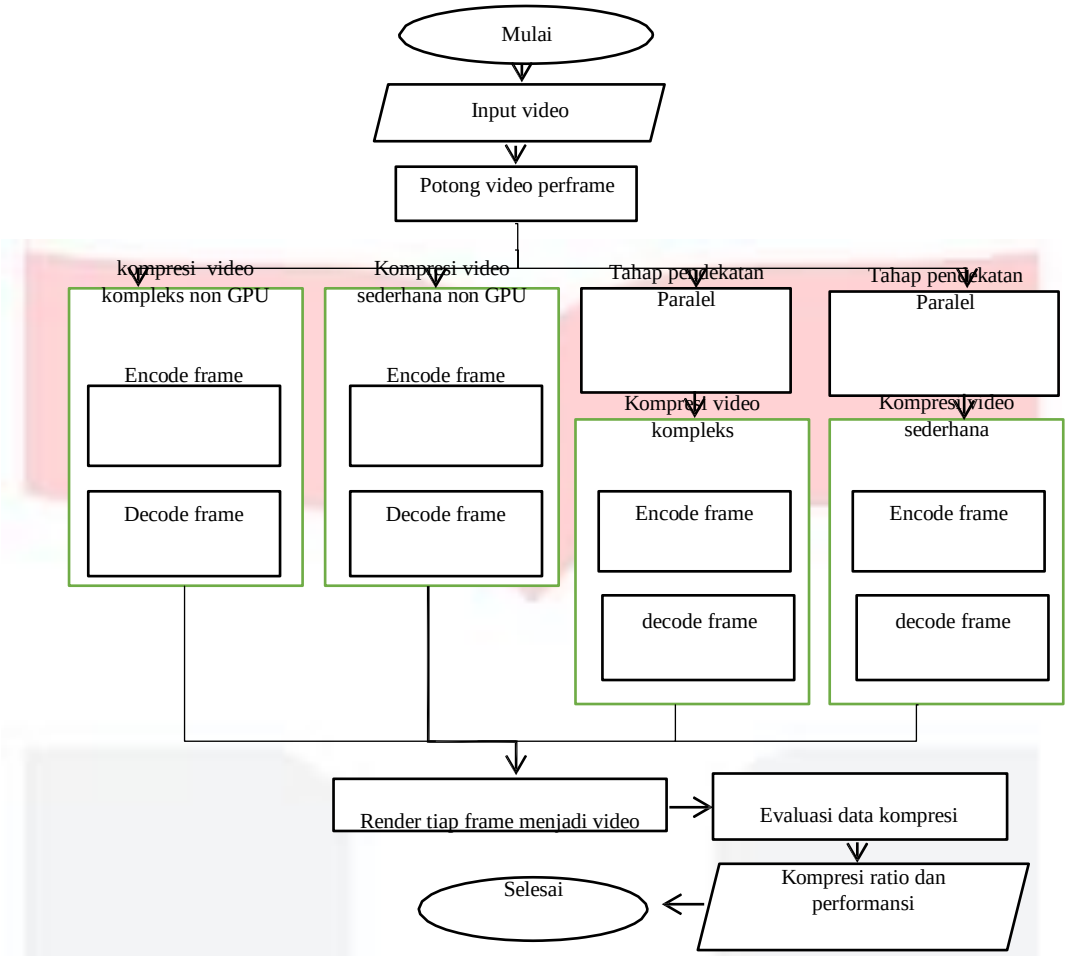
$$Efficiency = \frac{(\quad)}{(\quad)} \qquad (2)$$

3. Pembahasan

3.1 Deskripsi Sistem

Sistem yang dibangun adalah sebuah sistem yang digunakan untuk melakukan *real-time rendering and compression* pada teknik kompresi menggunakan algoritma *Run Length Encoding* (RLE), dimana sistem yang dibangun akan dijalankan pada *GPU* (*multi core*) dan *single processor*. Dalam sistem yang dibangun, akan dilakukan analisis mengenai perbandingan *ratio* kompresi dan waktu komputasi. Untuk ratio kompresi akan dibandingkan video yang memiliki gambar kompleks dengan video yang memiliki gambar sederhana. Sedangkan untuk waktu komputasi akan dibandingkan video kompleks dengan menggunakan GPU dan tidak menggunakan GPU, lalu video sederhana dengan mengunakan GPU dan tidak menggunakan GPU.

Secara umum rancangan sistem pada *real-time rendering and compression* pada yang akan dibangun digambarkan dalam diagram blok berikut:



Gambar 3.1 Blok Diagram Sistem Real-Time Rendering and Compression

3.1.1 *Encode, Decode,* dan Kompresi

Sebelum dilakukan kompresi, terlebih dahulu encode gambar yang sudah di pisahkan dari video, mengubah matriks hasil encode gambar menjadi sebuah vector, lalu gunakan kompresi dengan algoritma RLE, setelah mendapatkan hasil kompresinya lalu dapat dikembalikan lagi dengan proses dekompres. Hasil dekompres yang masih berupa vector tersebut lalu di ubah menjadi matriks kembali setelah itu dapat diubah kembali menjadi gambar dengan proses decode.

3.1.2 *Parallel Computing Toolbox* pada Algoritma RLE

Tahapan dalam mengimplementasikan MATLAB *Parallel Computing Toolbox* pada *Run Length Encoding* (RLE) adalah dengan menentukan jumlah *worker* yang digunakan untuk mengerjakan perintah, kemudian mengubah algoritma pada bagian *for-loop* menggunakan *parfor* seperti pada tabel di bawah ini.

Tabel 3.1 *Konsep MATLAB* Parallel Computing Toolbox

<table>
<tr><td colspan="5">for i = 1:100
perintah
…
End</td></tr>
<tr><td>matlabpool

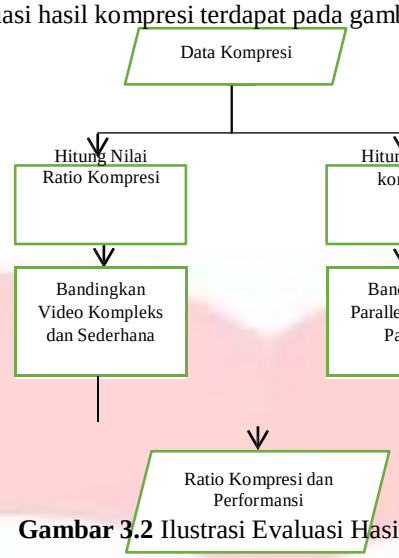
parfor i = 1:100
perintah
…
end</td><td colspan="4">Jika worker berjumlah 4, maka pembagian perintah for akan dieksekusi secara paralel seperti yang digambarkan pada tabel.</td></tr>
<tr><td></td><td>for i = 1:25
perintah
…
End</td><td>for i = 26:50
perintah
…
end</td><td>for i = 51:75
perintah
…
end</td><td>for i = 76:100
perintah
…
End</td></tr>
</table>

3.1.3 *Rendering*

Pada tahap ini gambar yang sudah di encode lalu dikompresi hingga di dekompresi dan dikembalikan lagi dengan proses decode digabungkan menjadi video kembali.

3.1.4 Evaluasi Data Kompresi

Ilustrasi untuk tahap evaluasi hasil kompresi terdapat pada gambar 3.2 berikut:



Gambar 3.2 Ilustrasi Evaluasi Hasil Kompresi

3.2 Hasil Pengujian

Data yang digunakan untuk percobaan ini diambil dari rekaman video secara langsung menggunakan webcam yang tersambung ke PC. Video yang diambil untuk percobaan ini sebanyak 2 video dengan resolusi 320x240, dengan menggunakan 8 fps dan citra berwarna (8 bit). Video yang digunakan untuk percobaan ini tidak menggunakan suara.

Tabel 3.2 Data Video

No	Video	Resolusi	Jumlah Frame	Ukuran File
1	Sederhana	320x240	45	11,22 Mb
2	Kompleks	320x240	45	9,89 Mb

Komputer yang digunakan dalam penelitian ini adalah computer dengan tambahan GPU. *Hardware* yang digunakan dalam penelitian ini adalah sebagai berikut:

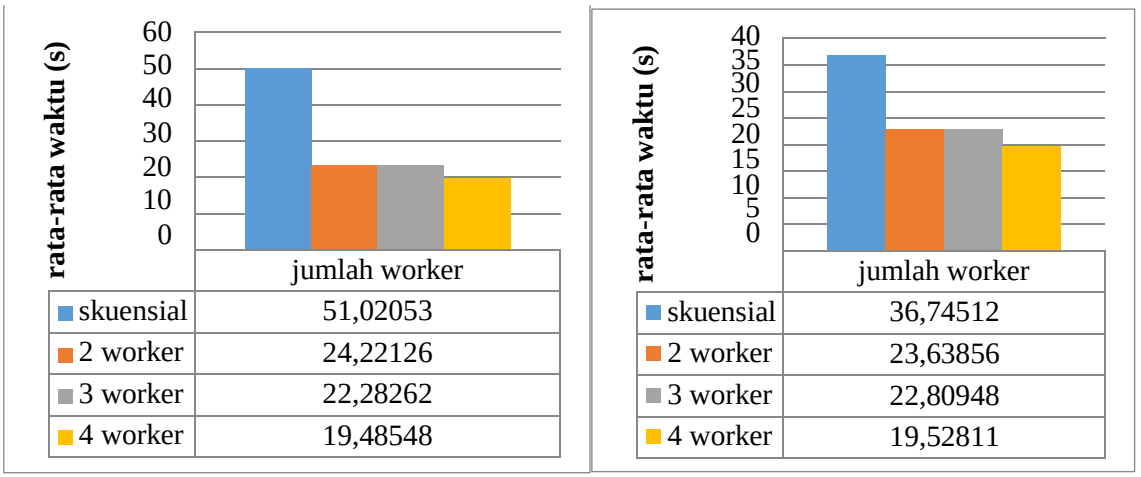
1. Core i3 3220 3.3Ghz
2. GTX 660 Ti 2 Gb
3. RAM 8 GB

Dalam penelitian ini dibuatkan GUI untuk mempermudah melakukan proses kompresi GUI yang dapat di akses. Yang tersedia pada GUI dalam penelitian ini adalah live video, pemilihan resolusi, pemilihan jumlah frame, pemilihan methode, *parallel worker*, kompres, dan 2 layar untuk input dan output.

Hasil pengujian yang didapat dari penelitian ini adalah analisis waktu dalam segi performansi apabila menggunakan parallel computing. Yang nanti nya akan di dapat *speedup* dan *efficiency* berdasarkan waktu. Selain itu perbandingan antara video warna yang sederhana dengan warna yang kompleks.

3.2.1 Analisis Waktu

Dari hasil percobaan yang dilakukan pada video sederhana menggunakan parallel dengan 4 worker didapatkan waktu rata-rata paling cepat pada proses kompresi pada video *complex* juga memiliki waktu rata-rata paling cepat pada saat 4 worker yang digunakan. Perbandingan mengenai waktu komputasi antara skuensial dan parallel dapat dilihat pada gambar 3.3.

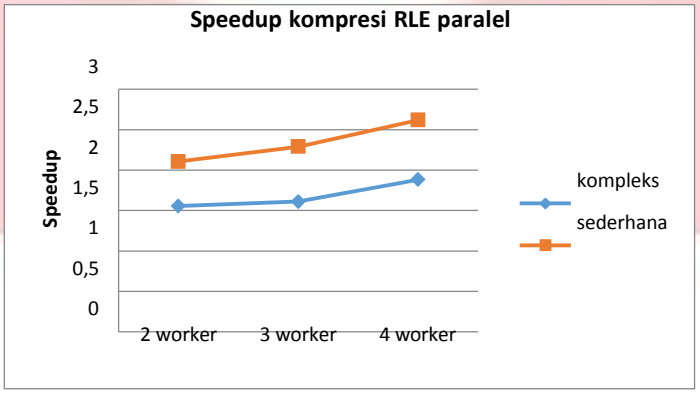


Gambar 3.3 Perbandingan Waktu Komputasi Antara Skuensial Dan Parallel Pada Video Sederhama dan Video Kompleks

Dari gambar 3.3 dapat dilihat peningkatan waktu komputasi yang besar antara proses skuensial dengan 2 worker sama halnya seperti video sederhana peningkatan waktu untuk penambahan worker tidak terlalu meningkat jauh dari yang sebelumnya dari dua gambar di atas bisa dilihat bahwa penggunaan parallel computing sangat berfungsi untuk mepercepat waktu kerja pada kompresi alghoritma RLE ini.

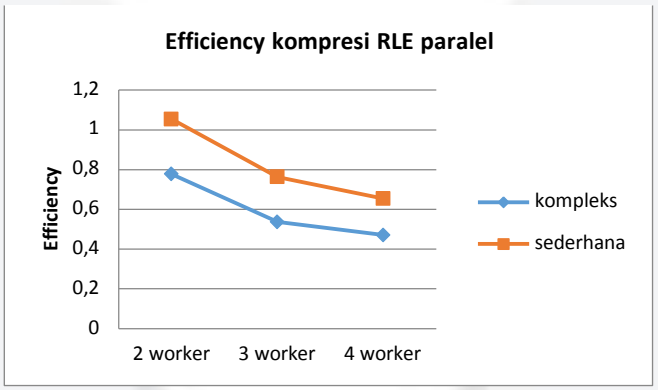
3.2.2 Analisis Performansi Paralel

Penjelasan mengenai *speedup* dan *efficiency* pada percobaan kompresi menggunakan algoritma RLE dapat dilihat pada Gambar 3.4 dan 3.5.



Gambar 3.4 *Speedup* pada Kompresi RLE

Berdasarkan penelitian menggunakan video sederhana dan video kompleks yang disediakan dengan jumlah worker yang berbeda, dihasilkan bahwa rata-rata speedup yang tertinggi diperoleh saat menggunakan 4 worker.



Gambar 3.5 *Efficiency* Pada Kompresi RLE

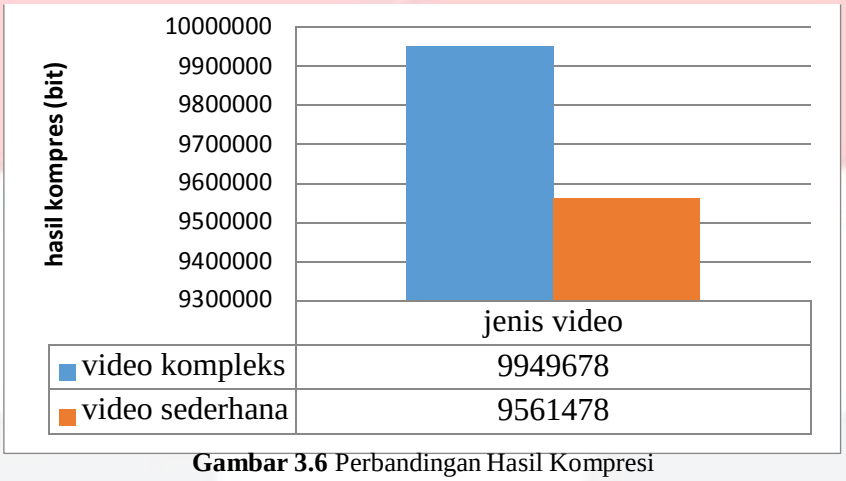
Dijelaskan pada Gambar 3.5, bahwa jumlah *worker* yang sama dengan 2, rata-rata menghasilkan nilai efisiensi tertinggi. Hal ini dikarenakan hanya dengan menggunakan 2 worker, rata-rata waktu komputasi paralel telah lebih baik dari rata-rata waktu komputasi sekuensial.

3.2.3 Analisis Ratio Kompresi

Ratio kompresi adalah dengan mencari total vector bit terkompresi tiap gambar di bagi dengan total vector bit awal tiap gambar sebelum terkompresi. Ratio kompresi video sederhana:

Hasil dari ratio kompresi video sederhana yaitu hasil percobaan yang dilakukan untuk kompresi video sederhana menggunakan algoritma RLE didapatkan ratio kompresinya adalah 7,79416% dengan rata-rata tiap frame terkompresi adalah 143.662 bit. Sedangkan, Dari hasil percobaan yang dilakukan untuk mengkompresi video kompleks menggunakan algoritma RLE didapatkan ratio kompresinya adalah 4,034742% dengan rata-rata tiap frame terkompresi adalah 74.368 bit.

Maka bisa dibandingkan dari hasil video sederhana apabila video yang memiliki warna lebih sedikit hasil kompresi menggunakan algoritma RLE yang diberikan akan lebih baik, dibandingkan dengan video kompleks yang memiliki warna lebih banyak pada tiap gambarnya algoritma RLE akan menghasilkan data terkompresi lebih sedikit. Penjelasan mengenai perbandingan hasil kompresi antara video sederhana dan video kompleks bisa dilihat pada gambar 3.7.



Gambar 3.6 Perbandingan Hasil Kompresi

4. Kesimpulan

Berdasarkan hasil dan pembahasan pada sistem yang telah dibuat, maka dapat disimpulkan bahwa:

1. Cara mengimplementasikan algoritma *Run Length Encoding (RLE)* dengan menggunakan MATLAB *Parallel Computing Toolbox* adalah dengan mengganti algoritma *for-loop* dengan menggunakan *parfor* pada fungsi kernel. Cara ini terbukti dapat mempercepat proses komputasi dari algoritma RLE.
2. Algoritma *Run Length Encoding* (RLE) hasil kompresi untuk video sederhana 7,79416% dan hasil kompresi untuk video kompleks sebesar 4,034742%. Untuk algoritma parallel pada video sederhana lebih unggul 13-17 detik dibandingkan dengan skuensial, dan untuk video kompleks lebih unggul 26-31 detik.
3. Hasil kompresi yang diperoleh menggunakan Run Length Encoding (RLE) untuk kompresi video RGB kecil.
4. Setelah dilakukan evaluasi performansi pada algoritma *Run Length Encoding* (RLE), terlihat bahwa jumlah *worker* sama dengan 2 memiliki efisiensi paling baik. Sedangkan apabila menggunakan 4 worker memilki rata-rata nilai *speedup* yang lebih besar dibandingkan dengan jumlah *worker* lainnya.
5. Pada penelitian kompresi menggunakan algoritma Run length Encoding (RLE) jenis video yang digunakan berpengaruh pada lama proses kompresi semakin rumit warna pada tiap frame yang ada pada video semakin lama juga proses kompresi nya dan semakin kecil juga hasil kompresi yang didapat, begitu juga sebaliknya apabila warna yang ada di tiap frame video semakin sedikit semakin cepat juga proses kompresinya dan hasil kompresinya semakin besar.
6. Hasil video setelah dikompres sama seperti asli nya karena alghoritma Run Length Encoding (RLE) kompresi yang sifatnya lossless.

Daftar Pustaka:

[1] Low, Adrian. (1991). *Introductory Computer Vision and Image Processing*. New York: McGraw-Hill.

[2] Murni, Aniati. (1992). *Pengantar Pengolahan Citra.* Jakarta: Elex Media Komputindo.

[3] Murni, Rinaldi. (2004). *Pengolahan Citra Digital Dengan Pendekatan Algoritmik*. Bandung: Informatika.

[4] Putra, Darma. (2010). *Pengolahan Citra Digital*. Yogyakarta: Andi.

[5] Sid-Ahmed, A. A. (1995). *Image Processing, Theory, Algorithms, and Architectures*. New York: McGraw-Hill.

[6] Ballard, Dana H. (1982). *Computer Vision*. New Jersey: Prentice-Hall.

[7] Sutoyo, T., Edy Mulyanto, Vincent Suhartono, Oky Dwi Nurhayati, dan Wijanarto. (2009). *Teori Pengolahan Citra Digital*. Yogyakarta: Andi.

[8] W.Suh, Jung; Youngmin, Kim. (2014). "*Accelerating MATLAB with GPUs A Primer with Examples*". Elsevier Inc

[9] http://www.eem.anadolu.edu.tr/ongerek/EEM562/icerik/rle.pdf (23/10/2014 3:15PM). Turkey: Anadolu University.

[10] http://www.ijcse.com/docs/IJCSE10-01-04-23.pdf (23/10/2014 3:38PM). Indian Journal of Computer Science Engineering.

[11] https://courses.cs.washington.edu/courses/cse459/06wi/help/mayaguide/Complete/Rendering.pdf (16/11/2014 3:12AM)

[12] http://www.*file*format.info/mirror/egff/ch09_03.htm (2/11/2014 8:00 PM)

[13] Ahmed Adnan Aqrawi. “Effects of *Compression* on Data Intensive Algorithms” 2010. Norwegian University of Science and Technology.

[14] https://courses.cs.washington.edu/courses/cse459/06wi/help/mayaguide/Complete/Rendering.pdf (16/11/2014 3:12AM)

[15] http://pdf.aminer.org/000/236/452/grid_based_computer_animation_rendering.pdf (16/11/2014 8:44PM)