PERBANDINGAN PERFORMA PADA *GAMMA CORRECTION* DENGAN MENERAPKAN *PARALLELISM* DAN *SEQUENTIAL PROGRAMMING*

Muhammad Fikri Alfaqih, Adriel Silaban, Farid Khoirur Rizal,Sandy Fitra Yanuar

Jurusan Ilmu Komputer/ Informatika, Fakultas Sains dan Matematika,

Universitas Diponegoro, Semarang

Email: fikrialf2019@gmail.com, sandyfitra036@gmail.com, faridkhoirur34@gmail.com, amshstk18@gmail.com

Abstrak

Ada dua macam proses komputasi dalam pemrograman, yaitu komputasi paralel dan komputasi sekuensial. Komputasi paralel sendiri telah terbukti meningkatkan kinerja komputasi dalam hal waktu dan oleh karena itu semakin banyak digunakan. Artikel ini akan membahas perbandingan kinerja waktu dari program Koreksi Gamma dengan menggunakan dua jenis komputasi, yaitu pemrograman paralel dan pemrograman sekuensial. Dari hasil pengujian dapat disimpulkan bahwa kinerja waktu eksekusi terbukti meningkat dengan menggunakan komputasi paralel. Hal ini dikarenakan dengan menggunakan komputasi paralel, proses-proses akan berjalan secara bersamaan. Sedangkan jika digunakan perhitungan berurutan, maka proses akan dilakukan secara berurutan.

Kata kunci: Performa waktu, *Gamma Correction*, *Sequential*, *Parallel*

Abstract

There are two kinds of computing processes in programming, namely parallel computing and sequential computing. Parallel computing itself has been shown to increase computational performance in terms of time and is therefore increasingly being used. This article will discuss the time performance comparison of Gamma Correction programs using two types of computation, namely parallel programming and sequential programming. From the test results it can be concluded that the execution time performance is proven to increase by using parallel computing. This is because by using parallel computing, processes will run concurrently. Whereas if sequential calculations are used, the process will be carried out sequentially.

Keyword: *Time performance, Gamma Correction, Sequential, Parallel*

1. Pendahuluan

Di era digital ini, manusia sudah terbiasa menggunakan teknologi untuk menyelesaikan berbagai masalah komputasi, karena solusi dengan menggunakan teknologi dianggap lebih cepat dan akurat daripada menyelesaikan masalah secara manual. Karena itu, tuntutan kinerja komputasi telah meningkat dari waktu ke waktu.

Komputasi paralel dapat dibangun dari komputer pribadi menggunakan program yang dapat meniru tingkat paralelisme. Metode yang digunakan untuk komputasi paralel adalah dengan menggunakan CUDA. Sebagai alat pemrograman paralel pada GPU, sumber daya dapat diambil dari semua core yang tersedia. Sedangkan proses komputasi sekuensial terjadi dalam urutan yang ketat, dan tidak mungkin untuk melanjutkan ke langkah berikutnya hingga proses saat ini selesai.

Artikel ini akan membahas tentang perbandingan performa waktu pada program uji *Gamma Correction* yang menerapkan dua macam komputasi yaitu paralel dan sekuensial pada contoh gambar dengan resolusi yang berbeda-beda..

Gamma Correction atau yang juga dikenal sebagai *Power Law Transform* bertujuan untuk meningkatkan kecerahan gambar namun tetap menjaga agar gambar tersebut memiliki nilai kontras yang normal.

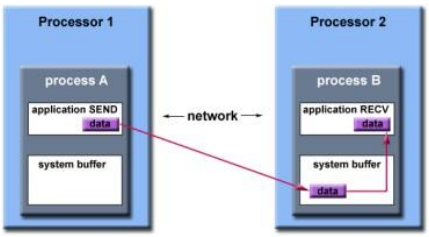
2. Dasar Teori

2.1.Komputasi Paralel

Komputasi paralel adalah metode komputasi untuk memecahkan masalah komputasi, dengan membagi beban komputasi menjadi beberapa sub-proses komputasi kecil, sub-komputasi berjalan pada prosesor yang berbeda pada waktu yang sama dan berinteraksi satu sama lain. Komputasi paralel diperlukan ketika daya komputasi yang dibutuhkan sangat besar, baik karena Anda harus mengolah data, atau karena banyak proses komputasi yang diperlukan.Tujuan dari komputasi paralel adalah meningkatkan kinerja komputer dalam menyelesaikan berbagai masalah dengan membagi sebuah masalah besar ke dalam

beberapa masalah kecil, sehingga hal tersebut membuat kinerja menjadi cepat**.**

Komputasi paralel berjalan dengan menggunakan pemrograman Paralel. Pemrograman paralel merupakan teknik pemrograman komputer yang memungkinkan eksekusi perintah/operasi secara simultan (komputasi paralel), baik dalam komputer dengan satu (prosesor tunggal) ataupun banyak (prosesor ganda dengan mesin paralel) CPU. Komunikasi data pada sistem pemrograman parallel ditunjukkan pada gambar 1 dibawah ini



Path of a message buffered at the receiving process

2.2. Komputasi *Sequential*

Komputasi berurutan adalah komputasi di mana proses terjadi dalam urutan yang ketat, dan tidak mungkin untuk melanjutkan ke langkah berikutnya sampai proses saat ini selesai. Komputasi berurutan biasanya dijalankan hanya menggunakan satu core/inti pada CPU.

2.3. CUDA

CUDA singkatan dari *Compute Unified Device Architecture* merupakan arsitektur komputer paralel yang dikembangkan oleh NVIDIA. CUDA memiliki kemampuan untuk melakukan komputasi pada CPU dan GPU secara bersamaan, memungkinkan komputasi berjalan lebih cepat. Sebagai contoh, perhitungan yang dilakukan oleh CPU dapat dilakukan dalam hitungan detik, namun dengan bantuan GPU, proses ini dapat dilakukan dalam hitungan *millisecond*. GPU yang dilengkapi CUDA mampu melakukan simulasi grafis dengan baik daripada hanya menggunakan CPU sehingga GPU yang dilengkapi CUDA mampu melakukan simulasi grafis dengan baik daripada hanya menggunakan CPU.

CUDA merupakan sebuah framework yang bertugas untuk menjembatani proses di CPU dan GPU sehingga keduanya dapat bekerja secara bersamaan (parallel). Seri pertama dari CUDA dikeluarkan oleh NVIDIA pada tahun 2006, yang terus melakukan pengemabnagan pada CUDA sehingga saat ini sudah mencapai

versi ke-4. CUDA merupakan sistem properti buatan NVIDIA.

3. Metode Penelitian

Studi kasus yang diamati adalah penerapan *Gamma Correction,* yang mengubah gamma pada gambar. Gamma dapat digambarkan sebagai hubungan antara input dan output yang dihasilkan. Untuk ruang lingkup percobaan ini inputnya adalah nilai intensitas RGB darisuatu gambar.

Pertama, intensitas piksel gambar harus diskalakan dari *range* [0, 255]. Kemudian untuk mendapat gambar *output gamma correction*, harus melalui formula berikut:

$$O = 255 \times (I/255)^{1/\gamma}$$

I adalah gambar input dan γ adalah nilai gamma. O adalah gambar output dari hasil operasi.

Algoritma *Gamma Correction*

Input: *I* gambar
Output : O hasil gambar

1. Muat gambar *I*.
2. Transformasikan gambar *I* ke dalam matriks *integer* dengan tiap indeks matriks berisi 3 *channel* warna (*Red, Green,Blue*).
3. Lakukan operasi untuk setiap channel warna pada tiap-tiap *pixel*.
4. Simpan hasil perhitungan ke dalam matriks *O.* Kemudian transformasikan kembali matriks *O* ke dalam bentuk gambar.

Untuk fokus pada penelitian kali ini yaitu perbandingan *runtime gamma correction* yang dilakukan dengan CPU dan GPU. Gambar inputan yang digunakan memiliki 4 resolusi yang berbeda yaitu 256px, 512px, 1024px, 2048px, 4096px, 8192px. Kemudian catat runtime untuk setiap resolusi. Untuk proses *gamma correction* menggunakan CPU atau secara serial yaitu dilakukan satu waktu dan dijalankan oleh prosesor secara berurutan. Untuk proses gamma correction menggunakan GPU yaitu tugas dikerjakan oleh beberapa prosesor dalam waktu yang sama menggunakan CUDA.

3.1. Proses Pengujian

Setelah model dihasilkan dari tahap pelatihan, maka saatnya untuk menguji model tersebut untuk mengubah data berupa gambar yang kabur, menjadi gambar yang jelas.

Dalam proses pengujian juga terdapat pilihan sumber daya

komputasi yang ingin digunakan. Sehingga kita dapat dengan jelas melihat perbedaan performa ketika menggunakan sekuensial dan paralel.

4. Hasil dan Pembahasan

4.1 Pengujian

Uji dilakukan dengan menggunakan dua pemrosesan, yaitu pemrosesan secara sekuensial dan paralel. Mengunakan gambar dengan beberapa resolusi yang berbeda yaitu 256x256, 512x512, 1024x1024, 2048x2048, 4096x4096 dan 8192x8192.

4.1.2 Pengujian Secara Sekuensial

Pengujian secara sekuensial ini dilakukan dengan menggunakan code berikut ini.

```
def gamma_correction_seq
(image, gamma):
  init_image =
np.zeros(image.shape,
np.uint8)
  for i in
range(image.shape[0]):
    for j in
range(image.shape[1]):
      init_image[i][j] =
255.0 * (image[i][j] /
255.0)**(1 / gamma)
  return init_image
```

```
255.0)**(1 / gamma)
  return init_image
```

Disini kita mengisi tiap index matrix di init_image dengan formula gamma correction. Untuk mengakses matriksnya disini masih sekuensial, ditunjukkan dengan penggunaan *for loop*.

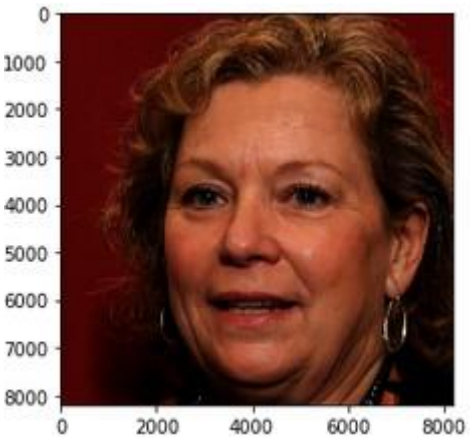
4.1.2 Pengujian Secara Paralel

Pengujian secara parallel ini dilakukan dengan menggunakan code berikut ini.

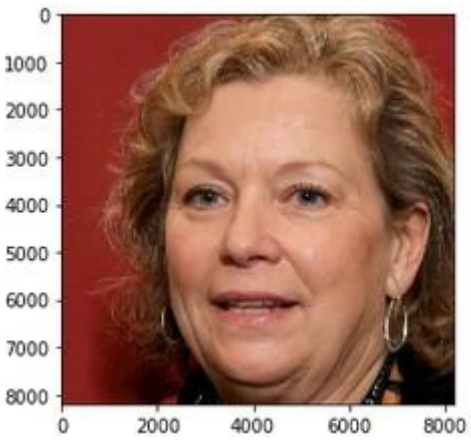
```
@numba.vectorize('uint8(uint
8,float64)',target='cuda')
def
gamma_correction_par(image,g
amma):
  return 255.0 * (image /
255.0)**(1 / gamma)
```

Disini kita menggunakan numba vectorize dengan target GPU CUDA. Fungsi dari gamma correctionnya juga tidak menggunakan *for loop* dan sudah di otomasi menggunakan numba vectorize tadi.

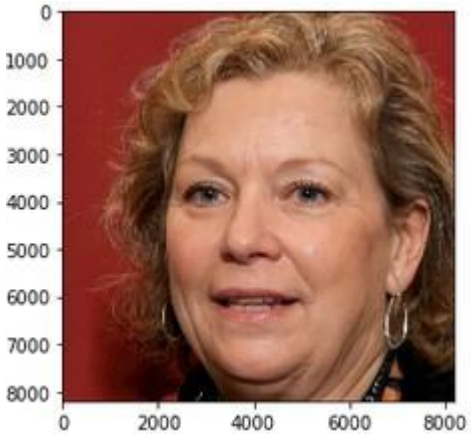
4.2 Hasil Pengujian



Gambar 4.2.1. Gambar Asli



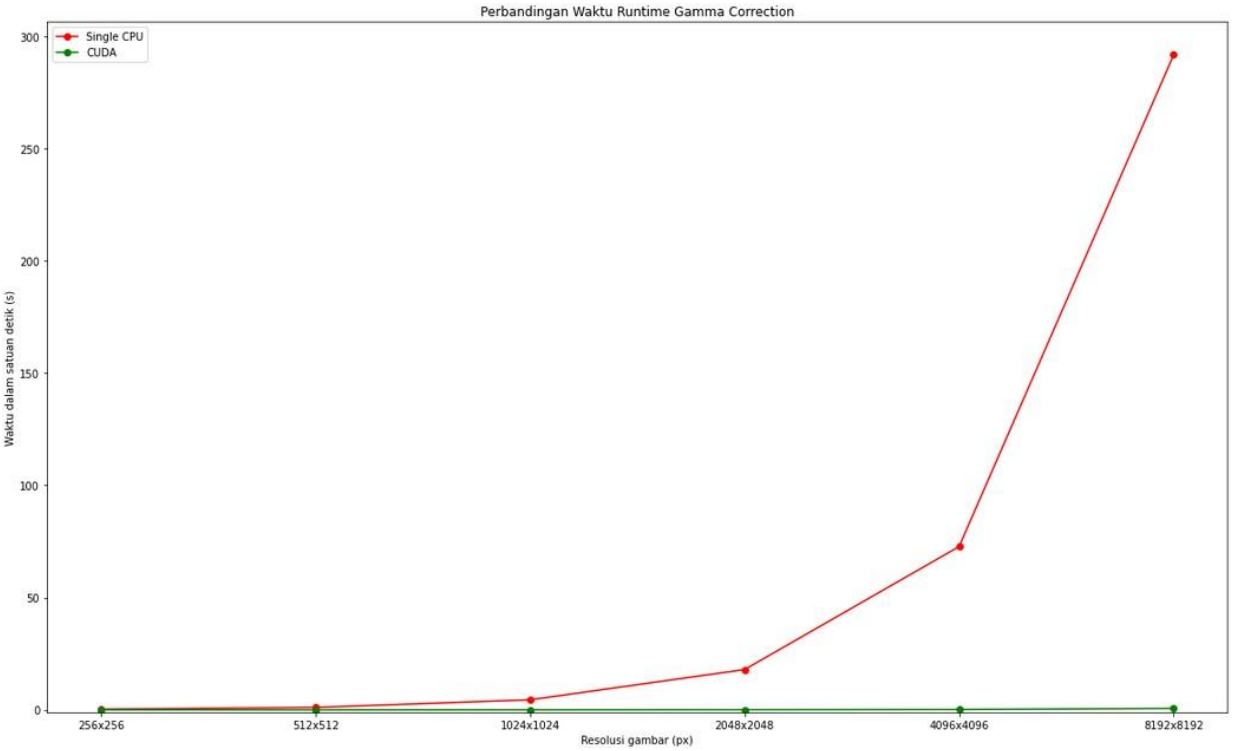
Gambar 4.2.2. Hasil Sequential



Gambar 4.2.3 Hasil Paralel

Gamma correction pada resolusi 8192x8192 dengan metode sekuensial berjalan selama 290.9415 detik. Sedangkan saat menggunakan metode parallel, kita hanya membutuhkan waktu 0.8221detik.

Dari dua metode pengujian yang telah dijelaskan diatas, didapatkan hasil dalam satuan waktu pada tiap resolusinya. Berikut bentuk grafik perbandingan antara Sekuensial dan Parallel.



5. Kesimpulan

Berdasarkan proses *Gamma Correction* dengan menggunakan sekuensial dan paralel maka dapat disimpulkan performa waktu terbaik pada *Gamma Correction* berhasildicapai oleh proses yang dilakukan dengan menggunakan pendekatan *parallelism*, dengan rincian sebagai berikut:

Resolusi	Waktu	
	CPU	GPU
256x256	0.2474s	0.0108s
512x512	0.9124s	0.0090s
1024x1024	4.3790s	0.0281s
2048x2048	14.8150s	0.1052s
4096x4096	64.8511s	0.3483s
8192x8192	256.2207s	1.4352s

6. Daftar Pustaka

[1] Singnoo, Jakkarin & Finlayson, Graham. (2010). Understanding the Gamma Adjustment of Images.

[2] Loch, F., 2021. *The Crypt Magazine*. [online] Thecryptmag.com. Available at: <https://thecryptmag.com/Online/57/imgproc_6.html> [Accessed 6 December 2021].

[3] Amiri, S. Asadi, and H. Hassanpour. "A preprocessing approach for image analysis using gamma correction." *International Journal of Computer Applications* 38.12 (2012): 38-46.