Analisis Perbandingan Komputasi GPU dengan CUDA dan Komputasi CPU untuk *Image* dan *Video Processing*

Bagus Kurniawan
Jurusan Teknik Elektro dan Teknologi Informasi
Fakultas Teknik, Universitas Gadjah Mada
Yogyakarta, Indonesia
bagusbekam@gmail.com

Teguh Bharata Adji
Jurusan Teknik Elektro dan Teknologi Informasi
Fakultas Teknik, Universitas Gadjah Mada
Yogyakarta, Indonesia
teguh_tba@mail.ugm.ac.id

Noor Akhmad Setiawan
Jurusan Teknik Elektro dan Teknologi Informasi
Fakultas Teknik, Universitas Gadjah Mada
Yogyakarta, Indonesia
noorwewe@mail.ugm.ac.id

***Intisari*— Teknologi yang semakin maju memicu berbagai paradigma komputasi untuk terus berkembang, tidak terkecuali mengenai teknik *image processing* maupun *video processing* yang dibutuhkan masyarakat untuk memanipulasi gambar guna kebutuhan informasi. Komputasi *Graphics Processing Unit* (GPU) menjadi salah satu alternatif komputasi paralel yang menawarkan kinerja komputer yang lebih cepat daripada komputasi *Central Processing Unit* (CPU) dengan memanfaatkan kartu grafis. Penelitian ini menganalisis teknik komputasi paralel GPU dengan *Compute Unified Device Architecture* (CUDA) dan membandingkan hasil kinerja dari komputasi sekuensial CPU dengan OpenCV yang dianalisis menggunakan metode eksperimen. Eksperimen dilakukan dengan implementasi *image* dan *video processing* untuk operasi *grayscale*, negatif, dan deteksi tepi. Penelitian ini menunjukkan sebuah hasil bahwa *image processing* untuk operasi grayscale dan negatif, komputasi paralel GPU lebih unggul antara 0.2 hingga 2 detik. Sedangkan untuk operasi deteksi tepi, komputasi GPU unggul hingga 14 detik. Atau 2.8 kali lipat lebih cepat daripada komputasi CPU. Untuk *video processing*, komputasi CPU lebih unggul dari komputasi GPU selisih antara 1-2 *frame per second*.**

Kata kunci—image processing, video processing, komputasi paralel, CUDA

I. PENDAHULUAN

A. Latar Belakang

Di era modern pada bidang multimedia yang sarat akan teknologi, khususnya mengenai *image processing a*tau teknik pengolahan citra yang merupakan teknik yang penting bagi kehidupan masyarakat. Teknik pengolahan citra digunakan untuk memanipulasi data gambar sesuai yang diinginkan dalam suatu kebutuhan informasi [1]. Teknik tersebut dapat dilakukan menggunakan komputasi paralel yang menjadi suatu terobosan baru untuk melakukan proses komputasi dan telah berkembang mengikuti perkembangan teknologi komputer di seluruh dunia. *Graphics Processing Unit* (GPU) menjadi salah satu komputasi paralel yang memanfaatkan kartu grafis [2].Hal ini bertujuan agar kinerja komputer jauh lebih cepat dibandingkan proses yang sepenuhnya hanya dilakukan oleh *Central Processing Unit* (CPU).

Ketertarikanpun timbul ketika mengetahui bahwa di dalam *video graphic card* (VGA) terdapat ratusan *core* dari GPU yang digunakan untuk mengolah gambar grafis untuk keperluan multimedia (pemutaran video dan *game*). Ratusan *core* tersebut dapat digunakan secara paralel untuk mengolah gambar. *Compute Unified Device Architecture* (CUDA) menjadi salah satu arsitektur bagi pengguna (*user*) untuk melakukan *image processing* yang memanfaatkan GPU secara terbuka [3]. *Image processing* mampu diolah secara paralel menggunakan GPU yang pada awalnya masing-masing *pixel* pada citra(*image*) diolah secara sekuensial yakni berurutan satu persatu dapat diolah secara parallel sekaligus untuk beberapa *pixel* dalam satu kali proses dengan membagi proses tersebut ke dalam *core-core* pada GPU.

Selain *image processing*, *video processing* juga menjadi salah satu grafik yang dapat dijalankan secara parallel. *Processor* pada CPU telah makin berkembang dengan meningkatkan kinerja secara optimal. Sehingga perlu dibandingkan antara komputasi paralel GPU dengan CUDA dengan komputasi CPU dengan melakukan *video processing* baik mengunakan file video ataupun memanfaatkan webcam untuk menjalankan video secara *realtime*.

Kinerja dari komputasi paralel menggunakan GPU dapat menjadi pertimbangan bagi pengguna untuk melakukan *image processing* maupun *video processing* tersebut dengan membandingkan terhadap komputasi CPU. Teknik komputasi dapat mempengaruhi kecepatan proses terhadap suatu pengolahan.

B. Rumusan Masalah

Kinerja komputasi paralel GPU dengan CUDA yang belum teruji baik untuk *image processing* maupun *video processing*. Dengan demikian perlu dilakukan analisis kinerja mengenai hal tersebut sehingga akan tampak perbandingan komputasi GPU dengan komputasi CPU.

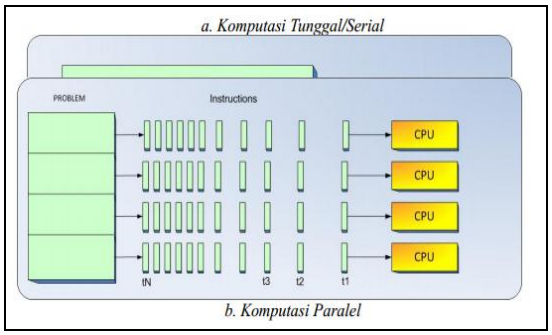
C. Tujuan Penelitian

Penelitian ini bertujuan untuk menerapkan teknik komputasi dengan teknologi CUDA dan memaparkan hasil kinerja dari perbandingan antara komputasi paralel GPU menggunakan CUDA dengan komputasi sekuensial CPU menggunakan OpenCV.

II. LANDASAN TEORI

A. Komputasi Paralel

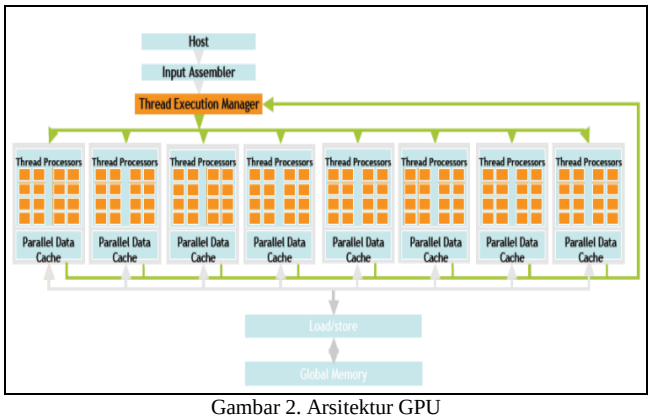
Komputasi paralel adalah penggunaan beberapa sumber daya komputasi secara simultan untuk menyelesaikan suatu permasalahan komputasi [4]. Sumber daya komputasi dapat berupa sebuah komputer dengan beberapa *processor*, beberapa komputer yang terhubunh melalui jaringan atau kombinasi dari keduanya. Perbandingan antara komputasi serial dengan komputasi paralel dapat dilihat pada Gambar 1, bahwa pada komputasi serial penyelesaian sebuah tugas komputasi/problem dibagi menjadi beberapa instruksi tugas (t), namun semua instruksi tersebut dijalankan oleh sebuah CPU saja. Lain halnya dengan komputasi paralel yang pada penyelesaian sebuah tugas komputasi/problem, setelah tugas komputasi tadi dibagi menjadi beberapa instruksi, masing-masing instruksi itu dibagi lagi bebannya ke beberapa CPU.



Gambar 1. Perbandingan komputasi serial dan komputasi paralel

Salah satu teknologi komputasi paralel adalah *Graphic Processing Unit*(GPU). GPU memiliki arsitektur tertentu, hal ini disebabkan karena GPU merupakan prosesor *multithread* yang mampu mendukung jutaan pemrosesan data pada satu waktu [5].

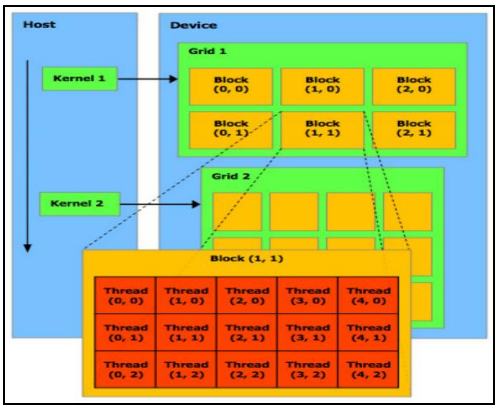
Arsitektur tersebut dapat digambarkan seperti Gambar 2 [6].



Gambar 2. Arsitektur GPU

B. Compute Unified Device Architecture(CUDA)

Arsitektur yang digunakan untuk memanfaatkan GPU dalam komputasi paralel adalah *Compute Unified Device Architecture*(CUDA). Arsitektur CUDA ini memungkinkan pengembang perangkat lunak untuk membuat program yang berjalan pada GPU buatan NVIDIA dengan *syntax* yang mirip dengan *syntax* C yang sudah banyak dikenal. Arsitektur CUDA terdiri dari tiga bagian dasar yang membantu pengembang sistem untuk memanfaatkan kemampuan komputasi secara efisien dari kartu grafis tersebut. CUDA membagi *device* ke *grid*, *block*, dan *thread* dalam struktur hirarki seperti Gambar 3. Karena ada sejumlah *thread* dalam satu blok dan jumlah *block* dalam satu *grid* dan sejumlah *grid* dalam satu GPU. Sehingga proses paralel tersebut dicapai dengan menggunakan suatu arsitektur hirarkis yang sangat besar.



Gambar 3. Arsitektur CUDA

CUDA biasa dipakai untuk pemrograman grafis seperti image dan *video processing* secara digital seperti *image segmentation*, perubahan kualitas citra, pengenalan pola yang dapat diaplikasikan untuk berbagai kebutuhan dalam dunia nyata [7].

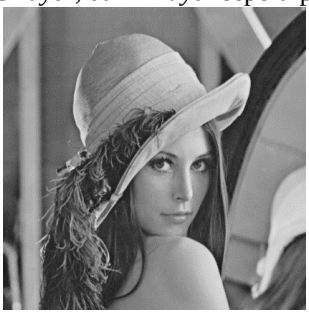
C. Pengolahan Citra

citra (*image*) adalah representasi dari dua dimensi untuk bentuk nyata tiga dimensi. Citra mempunyai karakteristik

yang tidak dimiliki oleh data teks, yaitu kaya dengan informasi. Pengolahan citra digital atau *digital image processing* merupakan bidang komputer yang berkembang sangat pesat sejalan dengan kemajuan teknologi industri saat ini [8]. Banyak operasi yang dapat dilakukan untuk mengolah suatu gambar maupun video antara lain :

a. Grayscale

Proses awal yang banyak dilakukan dalam pengolahan citra adalah mengubah citra berwarna menjadi citra *grayscale*, hal ini digunakan untuk menyederhanakan model citra. Citra berwarna terdiri dari tiga *layer* matrik yaitu R-*layer*, G-*layer*, dan B-*layer* seperti pada Gambar 4.



Gambar 4. Citra *Grayscale*

b. Negatif

Citra negatif suatu citra seperti halnya meniru film negatif pada fotografi, yaitu titik berwarna putih pada citra warna hitam pada film negatifnya. Demikian juga sebaliknya dengan cara mengurangi nilai intensitas *pixel* dari nilai keabuan maksimum. Misalnya citra dengan 256 derajat keabuan (8 bit). Contoh citra negatif dapat dilihat pada Gambar 5.

Gambar 5. Citra Negatif

c. Deteksi Tepi

Deteksi tepi (*Edge Detection*) pada suatu citra adalah suatu proses yang menghasilkan tepi-tepi dari obyek-obyek citra, tujuannya adalah untuk menandai bagian yang menjadi detail citra, untuk meperbaiki detail citra yang kabur, yang terjadi karena *error* atau adanya efek dari proses akuisisi citra. Suatu titik(x,y) dikatakan sebagai tepi(*edge*) dari suatu citra bila titik tersebut mempunyai perbedaan yang tinggi dengan tetangganya seperti pada Gambar 6.

Gambar 6. Citra Deteksi Tepi

Pengolahan gambar maupun video untuk komputasi CPU menggunakan OpenCV. OpenCV sendiri merupakan program *open source* khusus untuk program *computer vision* seperti mengolah baik gambar maupun video untuk komputasi CPU. *Library* untuk masing-masing operasi sudah optimal dengan waktu proses yang cepat. *library* ini mampu menciptakan aplikasi yang handal, kuat dibidang *digital vision*, dan mempunyai kemampuan yang mirip dengan cara pengolahan pada manusia.

III. PERANCANGAN SISTEM

Penelitian ini terbagi menjadi dua bagian, yaitu membandingkan antara komputasi CPU dengan komputasi GPU untuk *image processing* dan *video processing*. Operasi yang dilakukan adalah *grayscale*, negatif, dan deteksi tepi dengan metode sobel filter. Fungsi dalam algoritme yang digunakan sama baik dari CPU maupun GPU. Hanya saja cara mengolah operasi tersebut yang berbeda. Pemograman yang digunakan untuk komputasi CPU adalah OpenCV, sedangkan untuk komputasi GPU menggunakan CUDA.

Eksperimen ini menggunakan satu buah *notebook* dengan processor Intel Core i7-3520M 2.90Ghz, RAM 4GB, sistem Operasi Microsoft Windows 7 32 bit, dan OpenCV versi 2.4.3. Untuk Pemograman CUDA menggunakan VGA Nvidia NVS 5400M 2GB RAM DDR3.

Untuk *image processing*, gambar yang dibandingkan juga sama dengan resolusi(ukuran) gambar yang bertahap, dari yang beresolusi kecil hingga besar. Masing-masing resolusi tesebut diwakilkan oleh masing-masing 10 gambar. Resolusi tersebut antara lain 1280x768, 1900x1200, 4000x2500, 7500x5000 *pixel*. Dari proses tersebut akan dihasilkan waktu proses tiap gambar dengan masing-masing resolusi. Sehingga dapat dianalisis tren waktu proses yang dilakukan masing-masing komputasi.

Untuk *video processing*, dengan memanfaatkan media webcam untuk merekam video secara *realtime* sambil menjalankan proses pengolahan gambar dengan tiga operasi tersebut. Masing-masing komputasi dilihat dan dianalisis video tersebut dengan *Frame Per Second*(FPS)sebagai alat ukurnya.

A. Tahapan Algoritme pada CUDA

Komputasi paralel dengan pemograman CUDA melakukan beberapa tahap algoritme untuk mengolah gambar secara paralel antara lain :

1) Memuat gambar (*load an image*),
2) mengalokasikan CPU *memory*,
3) mengalokasikan GPU *memory* dengan jumlah memory yang sama dengan CPU menggunakan fungsi *library* CudaMalloc,
4) mengambil input data (data gambar/*image*) dari CPU *memory* yang disebut dengan *host memory*,
5) menyalin data ke dalam GPU *memory* menggunakan fungsi *library* CudaMemCpy dengan parameter CudaMemcpyHostToDevice, sehingga data dari *host memory* akan diolah ke dalam *device memory* (GPU),
6) melakukan pemrosesan dalam GPU *memory* di dalam *kernel* yakni untuk menjalankan komputasi paralel sesuai dengan *grid*, *block*, dan *thread* yang dibutuhkan untuk proses paralel,
7) menyalin data hasil (*result data*) dalam CPU *memory* menggunakan fungsi *library* CudaMemCpy dengan parameter CudaMemcpyDeviceToHost. Dengan kata lain, data hasil dari *device memory* (GPU) dikembalikan ke *host memory* (CPU), dan
8) membebaskan GPU *memory* atau *thread* lain menggunakan fungsi *library* CudaFree .

Dari tahapan algoritme maka disusun dalam suatu fungsi yang akan dijalankan pada GPU dengan menentukan berapa *grid* dan *blok* untuk menjalan pengolahan gambar secara paralel seperti pada Gambar 7.

```
1. extern "C" void cuda_parallel( unsigned
   char* h_in,  unsigned char* h_out, int
   width, int height, int widthStep, int
   widthStepOutput, int channels){
2. unsigned char* d_in;
3. unsigned char* d_out;
4. cudaMalloc((void**) &d_in,
   width*height*channels);
5. cudaMalloc((void**) &d_out,
   width*height);
6. cudaMemcpy(d_in, h_in,
   width*height*channels*sizeof( unsigned
   char), cudaMemcpyHostToDevice);
7. dim3 block (16,16);
8. dim3 grid (width/16, height/16);
9. kernel_grayscale<<<grid,block>>>(d_in,
   d_out, width, height, widthStep,
   widthStepOutput, channels);
10. cudaMemcpy(h_out, d_out,
   width*height*sizeof( unsigned char),
   cudaMemcpyDeviceToHost);
11. cudaFree(d_in); cudaFree(d_out);}
```

Gambar 7. Algoritme fungsi CUDA

Setelah itu maka hasil gambar yang diolah di dalam *device memory*(GPU) dan dikembalikan ke *host memory* (CPU) akan ditampilkan ke dalam canvas. Sehingga untuk *load* dan *save* gambar dilakukan oleh *host memory* di dalam CPU.

B. Algoritme

Operasi yang diuji dalam proses pengolahan adalah *grayscale*, negatif, dan deteksi tepi dengan metode *sobel filter*. Masing-masing memiliki fungsi dan algoritme yang berbeda, terutama pada deteksi tepi yang menggunakan algoritme yang lebih panjang. Fungsi dari masing-masing algoritme tersebut berlaku untuk pengujian baik *image processing* maupun *video processing*. Pengujian hanya dibedakan bahannya yakni gambar dengan video.

Operasi *grayscale* yang mengubah gambar menjadi gambar keabu-abuan menggunakan rumus dari gambar RGB menjadi *grayscale* seperti pada Persamaan 1.

$$Grayscale = 0.299R + 0.587G + 0.114B \quad (1)$$

Persamaan tersebut merupakan penjumlahan dari RGB dengan pengali masing-masing warna sehingga didapat warna keabu-abuan.

Operasi negatif merupakan suatu proses pada pengolahan citra yang dilakukan dengan cara mengurangi nilai intensitas *pixel* dari nilai keabuan maksimum. Rumus untuk membentuk gambar negatif menggunakan 256 derajat keabuan (8 bit) seperti pada Persamaan 2.

$$\begin{aligned} x &= (R + G + B)/3 \\ \text{Negatif} &= 255 - x \end{aligned} \quad (2)$$

Pada persamaan negatif tersebut, mula-mula variabeL x menghitung jumlah RGB dibagi 3 yang hasilnya adalah warna derajat keabu-abuan. Kemudian diinversi warnanya dengan mengurangi warna putih(RGB=255) dengan variabeL x, sehingga didapat warna negatif.

Operasi deteksi tepi merupakan suatu proses yang menghasilkan tepi-tepi dari obyek-obyek citra. Metode yang digunakan untuk deteksi tepi bermacam-macam. Salah satu metode yang digunakan untuk penelitian ini adalah metode *sobel*. Perbedaan antara metode *sobel* dengan metode deteksi tepi yang lain terdapat pada rumusnya sehingga hasil deteksi tepi tiap metode berbeda-beda. Untuk *sobel filter* menggunakan Persamaan 3.

$$Sx = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$$
$$Sy = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} \quad (3)$$

Pada persamaan deteksi tepi menggunakan metode sobel yang terdiri dari matriks konvolusi 3×3. Matriks-matriks ini didesain untuk dapat merespon dengan maksimal terhadap tepi yang berjalan secara vertikal dan horizontal relatif terhadap batas *pixel*(satu matriks untuk dua orientasi sudut 90 derajat). Matriks ini dapat digunakan secara terpisah pada gambar masukan, untuk menghasilkan pengukuran yang terpisah dari komponen gradien pada setiap orientasi (Sx dan Sy). Matriks-matriks ini dapat selanjutnya dikombinasikan untuk mencari skala absolut dari gradien pada setiap titik dan orientasi dari gradien tersebut.

IV. HASIL DAN PEMBAHASAN

Operasi *grayscale*, negatif, dan deteksi tepi ini akan diuji menggunakan gambar maupun video webcam yang memiliki paduan warna (RGB) sesuai dengan rumus tiap operasi. Masing-masing operasi akan diuji dengan melakukan dua komputasi dengan dua proses yang berbeda yakni gambar dan video. Hasil pengujian akan disajikan dalam bentuk tabel dan grafik untuk masing-masing komputasi.

Hasil perbandingan pada operasi *grayscale* yang mengacu pada Persamaan 1 dengan menguji 10 gambar dapat dilihat pada Tabel 1.

Tabel 1. Perbandingan waktu proses *image processing* operasi *grayscale*

Ukuran : 1024 x 768										
Image	1	2	3	4	5	6	7	8	9	10
CUDA	0.131	0.112	0.165	0.168	0.185	0.127	0.126	0.178	0.124	0.116
OpenCV	0.152	0.118	0.172	0.194	0.13	0.126	0.126	0.182	0.13	0.115
Ukuran : 1900x1200										
Image	1	2	3	4	5	6	7	8	9	10
CUDA	0.462	0.38	0.365	0.343	0.407	0.364	0.407	0.487	0.467	0.414
OpenCV	0.467	0.307	0.394	0.352	0.425	0.384	0.468	0.485	0.473	0.446
Ukuran : 4000x2500										
Image	1	2	3	4	5	6	7	8	9	10
CUDA	2.061	2.019	1.496	1.242	1.555	1.238	1.841	1.776	1.439	1.667
OpenCV	2.051	2.077	1.549	1.92	1.611	1.301	1.905	1.83	1.493	1.719
Ukuran : 7500x5000										
Image	1	2	3	4	5	6	7	8	9	10
CUDA	6.746	5.462	6.759	8.849	6.537	6.55	5.411	4.571	6.767	5.812
OpenCV	6.35	5.627	6.692	7.701	6.804	6.927	5.679	4.848	7.004	6.062

Pada perbandingan *grayscale* terlihat bahwa waktu proses antara CUDA dan OpenCV berkisar 0.001 hingga 0.006 detik untuk ukuran 1024x768 serta untuk ukuran 7000x5000 waktu proses maksimal hanya berselisih 0.3 detik untuk keunggulan CUDA.

Selanjutnya adalah perbandingan waktu proses untuk operasi negatif sesuai dengan Persamaan 2 dapat dilihat pada Tabel 2.

Tabel 2. Perbandingan waktu proses *image processing* operasi negatif

Ukuran : 1024 x 768										
Image	1	2	3	4	5	6	7	8	9	10
CUDA	0.18	0.16	0.222	0.222	0.256	0.176	0.171	0.239	0.175	0.165
OpenCV	0.2	0.186	0.248	0.255	0.256	0.202	0.193	0.27	0.193	0.185
Ukuran : 1900x1200										
Image	1	2	3	4	5	6	7	8	9	10
CUDA	0.622	0.546	0.521	0.485	0.558	0.528	0.607	0.62	0.599	0.516
OpenCV	0.742	0.57	0.608	0.576	0.645	0.581	0.683	0.695	0.706	0.758
Ukuran : 4000x2500										
Image	1	2	3	4	5	6	7	8	9	10
CUDA	2.772	2.747	2.11	1.824	2.189	1.855	2.473	2.396	2.077	2.318
OpenCV	3.06	3.023	2.514	2.189	2.572	2.296	2.968	2.926	2.362	2.627
Ukuran : 7500x5000										
Image	1	2	3	4	5	6	7	8	9	10
CUDA	8.7	8.003	10.617	10.187	9.205	9.616	7.861	6.911	9.322	8.335
OpenCV	10.878	9.22	10.448	11.612	10.557	10.635	9.357	8.823	10.759	9.714

Pada perbandingan negatif terlihat bahwa waktu proses antara CUDA dan OpenCV untuk ukuran 1024x760 maksimal hanya berselisih 0.03 detik. Seiring ukuran bertambah besar selisih waktu semakin meningkat yakni dari selisih 0.1 detik untuk ukuran 1900x2500, 0.5 detik untuk ukuran 4000x2500 hingga berselisih satu hingga dua detik untuk ukuran 7000x5000 dengan keunggulan CUDA.

Pada Tabel 3 merupakan perbandingan waktu proses untuk operasi deteksi tepi yang mengacu pada Persamaan 3.

Tabel 3. Perbandingan waktu proses *image processing* operasi *deteksi tepi*

Ukuran : 1024 x 768										
Image	1	2	3	4	5	6	7	8	9	10
CUDA	0.079	0.131	0.184	0.172	0.198	0.14	0.141	0.192	0.147	0.156
OpenCV	0.361	0.512	0.512	0.51	0.58	0.463	0.443	0.513	0.487	0.461
Ukuran : 1900x1200										
Image	1	2	3	4	5	6	7	8	9	10
CUDA	0.471	0.509	0.392	0.374	0.465	0.492	0.509	0.506	0.528	0.556
OpenCV	1.444	1.308	1.372	1.372	1.434	1.369	1.446	1.461	1.482	1.595
Ukuran : 4000x2500										
Image	1	2	3	4	5	6	7	8	9	10
CUDA	2.717	2.074	1.739	1.527	1.806	1.364	1.949	2.047	1.786	1.767
OpenCV	6.313	6.388	5.946	5.799	6.162	5.927	6.832	7.967	6.544	5.389
Ukuran : 7500x5000										
Image	1	2	3	4	5	6	7	8	9	10
CUDA	8.216	9.347	8.36	8.939	8.009	7.731	6.306	5.976	7.838	7.428
OpenCV	22.785	18.864	22.212	23.168	22.484	22.526	20.748	21.023	22.644	22.07

Pada perbandingan deteksi tepi tampak selisih waktu yakni untuk ukuran 1024x760 mencapai 0.4 detik, kemudian untuk ukuran 1900x1200 mencapai satu detik dan ukuran 4000x2500 mencapai lima detik serta untuk ukuran 7500x5000 mencapai 14 detik untuk keunggulan CUDA.

Dari pengujian tiap operasi dapat dihitung rata-rata waktu proses dari masing-masing komputasi. Pada perbandingan waktu rata-rata *grayscale* yang mengacu dari hasil pengujian pada Tabel 1 dapat dilihat pada Tabel 4.

Tabel 4. Perbandingan waktu rata-rata *image processing* operasi *grayscale*

Ukuran(Pixel)	OpenCV(detik)	CUDA(detik)
1280x768	0.1445	0.1432
1900x1200	0.8026	0.4096
4000x2500	1.7456	1.6334
7500x5000	6.3694	6.3464

Dari perbandingan waktu rata-rata didapat bahwa dari ukuran 1280x760 hingga ukuran 7500x2500 selisih waktu hampir merata dengan selisih paling besar hanya 0.023 detik.

Pada perbandingan waktu rata-rata negatif yang mengacu dari hasil pengujian pada Tabel 2 dapat dilihat pada Tabel 5.

Tabel 5. Perbandingan waktu rata-rata *image processing* operasi negatif

Ukuran(Pixel)	OpenCV(detik)	CUDA(detik)
1280x768	0.2188	0.1966
1900x1200	0.6564	0.5602
4000x2500	2.6537	2.2761
7500x5000	10.2003	8.8757

Dari perbandingan waktu rata-rata operasi negatif didapat bahwa dari selisih waktu paling besar mencapai 1.325 detik pada ukuran 7500x5000 untuk keunggulan CUDA

Pada perbandingan waktu rata-rata deteksi tepi yang mengacu dari hasil pengujian pada Tabel 3 dapat dilihat pada Tabel 6.

Tabel 6. Perbandingan waktu rata-rata *image processing* operasi *deteksi tepi*

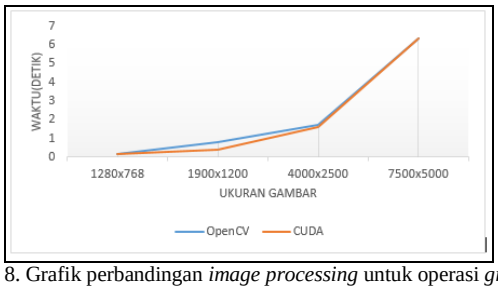
Ukuran(Pixel)	OpenCV(detik)	CUDA(detik)
1280x768	0.4842	0.154
1900x1200	1.4283	0.4802
4000x2500	6.3267	1.8776
7500x5000	21.8524	7.815

Dari perbandingan waktu rata-rata deteksi tepi didapat bahwa dari selisih waktu yang cukup signifikan mencapai 14.037 detik pada ukuran 7500x5000 untuk keunggulan CUDA

Berdasarkan perbandingan waktu rata-rata pada Tabel 4,6, dan 6, dapat dilihat pada Gambar 8, 9, dan 10 yang

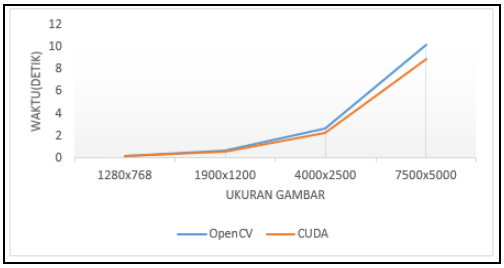
menampilkan graifk untuk perbandingan dari hasil pengujian yang didapat dari tabel perbandingan waktu rata-rata(detik) proses masing-masing komputasi.

Pada Gambar 8 tampak grafik untuk perbandingan waktu rata-rata untuk operasi *grayscale*.



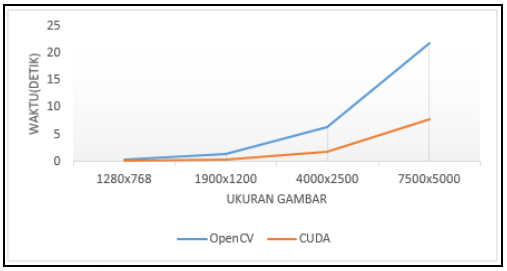
Gambar 8. Grafik perbandingan *image processing* untuk operasi *grayscale*

Pada grafik perbandingan *grayscale* tampak perbedaan waktu antara CUDA dan OpenCV berada dalam garis yang hampir sejajar. Lalu pada grafik perbandingan untuk operasi negatif dapat dilihat pada Gambar 9.



Gambar 9. Grafik perbandingan *image processing* untuk negatif

Pada grafik perbandingan negatif hampir mirip dengan operasi grayscale karena selisih waktu proses sangat sedikit yakni hingga satu detik yang terlihat pada ukuran 7500x2500. Kemudian untuk grafik perbandingan deteksi tepi dapat dilihat pada Gambar 10.



Gambar 10. Grafik perbandingan *image processing* untuk deteksi tepi

Pada grafik perbandingan deteksi tepi tampak selisih waktu cukup jauh yang terlihat mulai pada ukuran 1900x1200 hingga pada ukuran 7500x2500 yang mencapai selisih sekitar 14 detik atau mencapai 2.8 kali lipat untuk keunggulan CUDA.

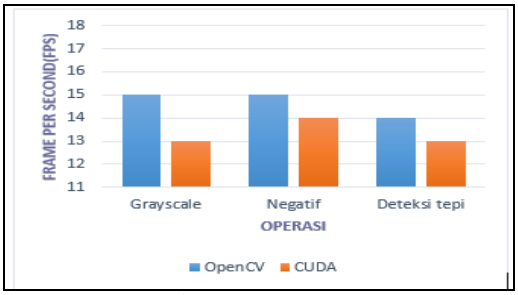
Dari hasil pengujian tiga operasi untuk komputasi CUDA dan OpenCV dapat dilihat bahwa secara keseluruhan CUDA lebih unggul daripada OpenCV.

Pada *video processing* dilakukan dengan menampilkan *video webcam* secara *realtime*, sehingga didapat hasil waktu dengan satuan fps. Pada Tabel 7 menunjukkan hasil dari masing-masing operasi untuk komputasi CPU dan GPU.

Tabel 7. Hasil perbandingan *frame per second* untuk *video processing*

Ukuran(pixel)	Grayscale	Negatif	Detektif Tepi
OpenCV	15 fps	15 fps	14 fps
CUDA	13 fps	14 fps	13 fps

Dari hasil perbandingan untuk *video processing* tersebut menunjukkan perbedaan grafik seperti pada Gambar 11.



Gambar 11. Grafik perbandingan *Video Processing*

Pada grafik perbandingan untuk *video processing* didapat bahwa OpenCV unggul tipis dengan CUDA antara satu hingga dua fps. Hal itu dikarenakan data yang di-*copy* ke GPU terlalu banyak dan berjalan secara perulangan sehingga memakan waktu lebih lama untuk ekseksusi.

Dari penelitian ini sudah menunjukkan secara keseluruhan bahwa komputasi paralel terbukti lebih cepat untuk melakukan pengolahan multimedia, bahkan sudah diterapkan proses secara paralel untuk *rendering*, *converting*, dan pemrosesan *editing* gambar maupun video dengan kompleksitas yang tinggi.

V. KESIMPULAN

Teknologi CUDA untuk komputasi paralel menggunakan GPU menunjukkan keunggulan terhadap komputasi sekuensial menggunakan CPU. Berdasarkan pengujian tersebut, diketahui bahwa selisih kedua komputasi untuk *image processing* tidak begitu besar terutama untuk operasi *grayscale* dan negatif. Namun untuk operasi deteksi tepi, makin besar ukuran gambar semakin terlihat kecepatan proses CUDA hingga 2.8 kali lipat lebih cepat daripada OpenCV yang didapat pada ukuran gambar 7500x5000. Hal ini disebabkan pada perbedaan algoritme pada tiap operasi. Pada operasi *grayscale* dan negatif memiliki algoritme yang hampir mirip yakni cukup menggunakan satu persamaan untuk mengubah warna tiap *pixel* gambar. Sedangkan algoritme untuk operasi deteksi tepi menggunakan matriks konvolusi 3x3 untuk matriks X dan Y. Hasil dari matriks tersebut baru dimasukkan pada tiap pixel gambarnya. Algoritme persamaan matriks tersebut lebih cepat dijalankan secara paralel.

Namun untuk *video processing*, OpenCV sedikit lebih unggul dari CUDA dengan perbedaan hingga dua *frame* meskipun tidak begitu terlihat di dalam video.

Dalam penelitian selanjutnya perlu dilakukan pengujian untuk membandingkan antara komputasi GPU(CUDA) dengan komputasi paralel untuk operasi dengan algoritme yang lebih kompleks. Untuk *video processing* bisa dibandingkan dengan memanfaatkan OpenGL untuk mengoptimakan GPU. Bahkan perlu dilakukan pengujian secara nyata untuk proses *rendering* gambar maupun video.

DAFTAR PUSTAKA

[1] Munir, R. 2004, "Pengolahan Citra Digital dengan Pendekatan Algoritmik". Bandung : Informatika, 2007

[2] Di Salvo, R dan Pino, C, "Image and Video Processing on CUDA: State of The Art and Future Directions", Catania : University of Catania, 2011.

[3] Yang, Z., Zhu, Y., Pu, Y., "Parallel image processing based on CUDA, China: Polytechnical University, 2008.

[4] Barney, B, "Introduction to Parallel Computing", https://computing.llnl.gov/tutorials/parallel_comp/ Diakses tanggal 20 Oktober 2014, 2009.

[5] Mukhlis, Y dan Harmanto, L. Metode Sorting Bitonic Pada GPU. http://openstorage.gunadarma.ac.id/mwiryana/KOMMIT/per-artikel/02-02-007- Metode, 02 Februari 2007.

[6] Lippert, A, "NVidia GPU Architecture for General Purpose Computing" , www.cs.wm.edu/~kemper/cs654/slides/nvidia.pdf/ Diakses tanggal 7 Oktober 2013, 2009.

[7] Basuki, A. 2005, "Pengolahan Citra Digital Menggunakan Visual Basic 6", Yogyakarta: Graha Ilmu, 2005

[8] Ramadijanti, N. 2005. Pengenalan Rambu-rambu Lalu Lintas menggunakan Pola Fitur Histogram Sudut. Surabaya: PENS ITS Surabaya.