PERCEPATAN *MOTION ESTIMATION* BERBASIS *PHASE ONLY CORRELATION* DENGAN TEKNIK *FULL SEARCH* MENGGUNAKAN *PARALEL THREADING* PADA GPU

Rosa A. Asmara[1], Cahya Rahmad[1], dan Anik N. Handayani[2]

[1] Laboratorium Multimedia, Politeknik Negeri Malang, Jalan Semarang 5 Malang, Jawa Timur, 65145, Indonesia
[2] Laboratorium Multimedia, Universitas Negeri Malang, Jalan Semarang 5 Malang, Jawa Timur, 65145, Indonesia

E-mail: rosa_andrie@poltek-malang.ac.id

Abstrak

Penelitian ini menyajikan penggunaan metode *Phase Only Correlation* (POC) pada *motion estimation* dengan teknik *full search* menggunakan *Graphical Processing Unit (GPU)*. Dengan fungsi POC, seseorang dapat melakukan estimasi translasi *motion* antara dua blok citra referensi dan citra yang diproses. *Full Search* berbasis POC adalah algoritma yang membutuhkan waktu proses lama. Hal ini menyebabkan sistem yang dicoba pada penelitian ini memproses fungsi POC pada *Graphical Processing Unit* (GPU) yang memiliki kelebihan dalam menyelesaikan perhitungan bilangan *floating point* dibandingkan CPU. Evaluasi dilakukan dengan menghitung kecepatan waktu proses menggunakan GPU pada video resolusi tinggi dengan resolusi hingga 1280x720 *pixel*. Hasil pengujian menunjukkan bahwa metode yang diselesaikan menggunakan GPU memiliki percepatan hingga hampir dua kali lipat pada ukuran blok *POC* 256 x 256 daripada menggunakan CPU.

Kata Kunci: *graphic processing unit, motion estimation, phase only correlation*

Abstract

This research presents a method using Phase Only Correlation (POC) on the motion estimation with full search technique using the Graphical Processing Unit (GPU). With POC function, someone can estimate the translational motion between two blocks of the reference image and the processed image. POC based Full Search is an algorithm that takes long time process. This leads the system that is used in this research to process the POC function on Graphical Processing Unit (GPU) which has advantages in solving the floating point calculations than the CPU. Evaluation is conducted by calculating the speed of processing time using a GPU on a high-resolution video with resolutions up to 1280x720 pixels. The test results show that the method that is solved using GPU has an acceleration up to nearly twice the size of the POC block 256 x 256 instead of using the CPU.

Keywords: *graphic processing unit, motion estimation, phase only correlation*

1. Pendahuluan

Motion Estimation adalah suatu proses untuk menentukan pergerakan suatu objek pada sequences video. Umumnya motion tersebut diwujudkan dengan vektor motion pada titik yang dipilih di dalam frame saat ini dihubungkan dengan frame lain yang disebut dengan frame referensi. Vektor motion adalah representasi perpindahan dari suatu titik antara frame saat ini dengan frame referensi [1].

Di antara berbagai macam metode estimasi motion, algoritma pencocokan blok adalah yang paling banyak digunakan dan populer karena kesederhanaan dalam aplikasinya. Pada algoritma penyocokan blok, suatu blok citra yang berpusat pada satu titik di tengahnya di frame saat ini dibandingkan dengan blok kandidat di frame referensi berdasarkan perbedaan atau persamaan tertentu untuk menemukan blok cocok yang terbaik pada area yang dicari. Salah satu contoh pengukuran dari perbedaan adalah Sum of Absolute Differences (SAD). Vektor motion dari titik didapat dari perpindahan blok.

Metode untuk mencari block match terbaik diklasifikasikan dalam dua tipe, yakni metode full search dan hierarchical search. Full search lebih cocok untuk mendeteksi motion lokal dari objek-objek individual. Sementara hierarchical search lebih cocok dipakai untuk mendeteksi motion global dari suatu scene.

Full search berbasis POC adalah algoritma dengan komputasi tinggi sehingga membutuhkan waktu proses lama. Oleh karena itu pada

penelitian ini sistem yang dibangun membuat fungsi POC agar dapat diproses pada Graphical Processing Unit menggunakan teknologi parallel threading.

2. Metodologi

Pada bagian *Phase Only Correlation* (POC), peneliti menjelaskan jika dua buah citra resolusi $N_1 \times N_2$, $f(n_1,n_2)$ dan $g(n_1,n_2)$, yang diasumsikan *range index*-nya adalah $n_1 = -M_1,...,M_1$ dan $n_2 = -M_2,...,M_2$ agar lebih sederhana dalam perhitungan, sehingga $N_1 = 2M_1 + 1$ dan $N_2 = 2M_2 + 1$. Jika $F(k_1,k_2)$ dan $G(k_1,k_2)$ adalah Diskrit Transformasi Fourier 2D (2D DFTs) dari dua buah citra, $F(k_1,k_2)$ dan $G(k_1,k_2)$ didapat dengan menggunakan persamaan 1 dan persamaan 2.

$$F(k_1,k_2) = \sum_{n_1 n_2} f(n_1,n_2) W_{N_1}^{k_1 n_1} W_{N_2}^{k_2 n_2} = A_F(k_1,k_2) e^{j\theta_F(k_1,k_2)} \quad (1)$$

$$G(k_1,k_2) = \sum_{n_1 n_2} g(n_1,n_2) W_{N_1}^{k_1 n_1} W_{N_2}^{k_2 n_2} = A_G(k_1,k_2) e^{j\theta_G(k_1,k_2)} \quad (2)$$

$$k_1 = -M_1,...,M_1 \quad (3)$$

$$k_2 = -M_2,...,M_2 \quad (4)$$

$$W_{N_1} = e^{-j\frac{2\pi}{N_1}} \quad (5)$$

$$W_{N_2} = e^{-j\frac{2\pi}{N_2}} \quad (6)$$

$$\sum n_1 n_2 = \sum_{n_1=-M_1}^{M_1} \sum_{n_2=-M_2}^{M_2} \quad (7)$$

Pada persamaan 1 dan 2, $AF(k_1,k_2)$ dan $AG(k_1,k_2)$ adalah komponen amplitudo, sementara $e^{j\theta_{F(k_1,k_2)}}$ dan $e^{j\theta_{G(k_1,k_2)}}$ adalah komponen fase. *Spektrum Cross-phase* (*cross spectrum* ternormalisasi) $\hat{R}(k_1,k_2)$ didapat dari persamaan 8.

$$\hat{R}(k_1,k_2) = \frac{F(k_1,k_2)\overline{G(k_1,k_2)}}{\left|F(k_1,k_2)\overline{G(k_1,k_2)}\right|} = e^{j\theta(k_1,k_2)} \quad (8)$$

Ekspresi $\overline{G(k_1,k_2)}$ menyatakan konjugasi kompleks dari $G(k_1,k_2)$ dan $\theta(k_1,k_2) = \theta F(k_1,k_2) - \theta G(k_1,k_2)$. Fungsi *Phase Only Correlation* $\hat{r}(n_1,n_2)$ adalah *inverse* Diskrit Transformasi Fourier 2D (2D IDFT) dari $\hat{R}(k_1,k_2)$ dan didapat dari persamaan 9.

$$\hat{r}(n_1,n_2) = \frac{1}{N_1 N_2} \sum_{k_1 k_2} \hat{R}(k_1,k_2) W_{N_1}^{-k_1 n_1} W_{N_2}^{-k_2 n_2} \quad (9)$$

$$\sum k_1 k_2 = \sum_{k_1=-M_1}^{M_1} \sum_{k_2=-M_2}^{M_2} \quad (10)$$

Asumsikan $f_c(x_1,x_2)$ adalah citra 2D dalam ruang temporal dengan bilangan real disimbolkan sebagai x_1 dan x_2. Simbol δ_1 dan δ_2 mewakili perpindahan *subpixel* $fc(x_1,x_2)$ pada arah x_1 dan x_2. Maka citra yang berpindah dapat disimbolkan $fc(x_1-\delta_1,x_2-\delta_2)$. Anggap bahwa $f(n_1,n_2)$ dan $g(n_1,n_2)$ adalah citra yang ter-*sampling* secara spasial dari $fc(x_1,x_2)$ dan $fc(x_1-\delta_1,x_2-\delta_2)$, dan didefinisikan dengan persamaan 11 dan 12.

$$F(n_1,n_2) = f_c(x_1,x_2)|_{x_1=n_1T_1,x_2=n_2T_2} \quad (11)$$

$$F(n_1,n_2) = f_c(x_1-\delta_1,x_2-\delta_2)|_{x_1=n_1T_1,x_2=n_2T_2} \quad (12)$$

T_1 dan T_2 adalah interval *sampling* secara spasial. *Range index* didapat dari persamaan 13.

$$n_1 = -M_1,...,M_1 \text{ dan } n_2 = -M_2,...,M_2. \quad (13)$$

Fungsi POC $\hat{r}(n_1,n_2)$ antara $f(n_1,n_2)$ dan $g(n_1,n_2)$ didapat dari persamaan 14.

$$\hat{r}(n_1,n_2) \cong \frac{\alpha}{N_1 N_2} \frac{\sin\{\pi(n_1+\delta_1)\}}{\sin\left\{\frac{\pi}{N_1}(n_1+\delta_1)\right\}} \frac{\sin\{\pi(n_2+\delta_2\}}{\sin\left\{\frac{\pi}{N_2}(n_2+\delta_2)\right\}}$$

Dengan $\alpha < 1$ (14)

Posisi puncak dari fungsi POC menyatakan perpindahan antara dua citra dan nilai puncak α menyatakan derajat kemiripan antara dua citra. Gambar 1 menunjukkan contoh fungsi *fitting* untuk melakukan estimasi posisi yang benar dan tinggi dari puncak korelasi.

Sementara pada bagian Metode *Full Search*, peneliti menjelaskan metode *full search motion estimation* berbasis POC (atau dapat juga disebut POC-FS) yang membutuhkan lebih sedikit kalkulasi daripada *full search* biasa tetapi memiliki hasil pencocokan yang lebih akurat.

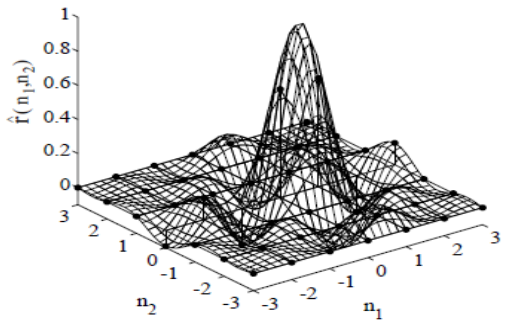
Dikatakan dua blok citra dengan ukuran W x W dengan menerapkan fungsi *Hanning Window* pada ukuran yang sama. Dikarenakan setengah lebar dari fungsi *Hanning Window* adalah $\frac{\mathbf{W}}{\mathbf{2}}$, dapat dikatakan pergeseran maksimum yang diestimasi antara dua blok citra adalah ± $\frac{\mathbf{W}}{\mathbf{4}}$ baik horizontal maupun vertikal, sehingga tidak perlu memeriksa setiap blok kandidat, melainkan hanya memeriksa blok kandidat pada interval $\frac{\mathbf{W}}{\mathbf{4}}$ piksel pada area yang sama.

Terdapat empat prosedur untuk POC-FS. Dimana masing-masing *input* adalah citra saat ini $I(n_1, n_2)$, citra referensi $J(n_1, n_2)$, dan titik p $I(n_1, n_2)$. Kemudian akan menghasilkan dua *output*, yakni titik korespondensi q dari titik p pada $J(n_1, n_2)$ dan *Motion vector* v_p^{FS} dari titik p. Langkah pertama adalah mengekstrak citra per blok dengan ukuran blok *W x W* dengan titik p berada di bagian tengah pada *frame* sekarang $I(n_1, n_2)$. Langkah kedua mengkalkulasikan fungsi POC antara blok citra dari *frame* sekarang $I(n_1, n_2)$ dan blok kandidat dari *frame* referensi $J(n_1, n_2)$ tiap $\frac{\mathrm{W}}{4}$ piksel pada area pencarian. Peneliti menggunakan versi sederhana dari fungsi POC untuk mendapatkan nilai korelasi puncak (didefinisikan sebagai nilai maksimum dari fungsi POC $\hat{r}(n_1, n_2)$ pada persamaan 4 dan posisi puncak yang akan memberikan nilai pergeseran antara dua citra dengan akurasi tinggi. Langkah ketiga adalah mengidentifikasikan 3 blok cocok terbaik pada langkah 2 dan geser blok-blok tersebut menurut nilai pergeserannya sehingga akan didapatkan puncak korelasi pada bagian tengah dari blok. Kalkulasi ulang fungsi POC untuk 3 kandidat blok baru. Blok paling cocok adalah blok dengan korelasi puncak tertinggi diantara tiga blok. Fungsi POC yang digunakan lagi adalah fungsi POC versi sederhana. Titik korespondensi q diposisikan di bagian tengah dari blok yang paling cocok.

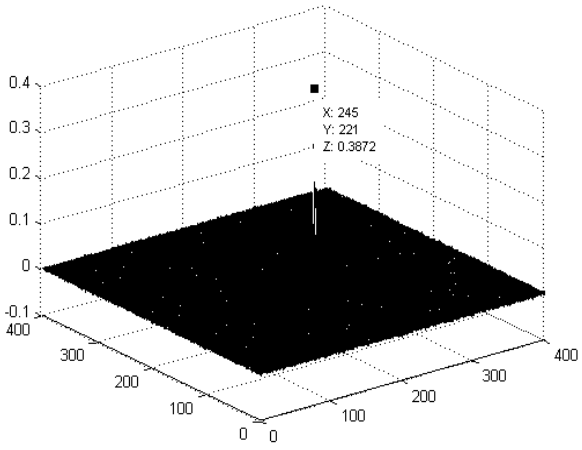
Langkah keempat adalah menggambarkan *motion* vektor $v_p^{FS} = q - p$. Pada percobaan ini, peneliti menggunakan ukuran *window* 128x128 dan area pencarian 64 horizontal dan vertikal.

GPU dipabrikasi dan lebih cocok dipakai untuk menyelesaikan permasalahan yang dapat diekspresikan ke dalam komputasi *parallel-data* (program yang sama dieksekusi pada banyak elemen data dalam paralel) dengan intensitas aritmatika tinggi (rasio dari operasi aritmatika dibandingkan operasi memori). Dikarenakan program yang sama dieksekusi untuk tiap elemen data, dibutuhkan spesifikasi yang lebih rendah untuk *flow control* canggih. Kemudian karena dijalankan pada banyak elemen data dan memiliki intensitas aritmatika tinggi, akses memori *latency* dapat disembunyikan dengan perhitungan tanpa *cache* data yang besar.

Pemrosesan data paralel memetakan elemen data pada *thread parallel processing*. Banyak aplikasi yang memproses set data berukuran besar dapat menggunakan model pemrograman data-paralel untuk mempercepat perhitungan. Pada *rendering* 3D, set piksel dan titik berukuran besar dipetakan pada *thread* paralel. Dengan cara yang sama, aplikasi pemrosesan media dan citra seperti *post-processing* dari citra di-*render*, *video encoding* dan *decoding*, penskalaan citra, *stereo vision*, dan pengenalan pola dapat memetakan blok citra dan piksel pada *thread parallel processing*. Kenyataannya banyak algoritma selain bidang *rendering* dan pemrosesan citra dapat dipercepat menggunakan pemrosesan data-paralel, dari pemrosesan sinyal umum ataupun simulasi fisika hingga komputansi *financial* atau komputasi biologi. Tipe data dari GPU yang peneliti gunakan hanya dapat memproses data *single precision* sehingga peneliti harus mengonversi tipe datanya terlebih dahulu jika data pada *double precision*.



Gambar 1. Fungsi *fitting* untuk estimasi posisi puncak dan koordinat pergeseran.



(a)

(b)

(c)

Gambar 2. (a) Translasi pergeseran dari puncak korelasi, (b) dan (c) dengan *phase only correlation* menggunakan GPU.

Terdapat delapan langkah dalam prosedur POC GPU. Langkah pertama adalah mengonversi tipe data *input* dari *double precision* ke *single precision*. Langkah kedua adalah meminindahkan data *single precision* dari *RAM CPU* ke *RAM GPU*. Langkah ketiga adalah melakukan komputasi paralel FFT paralel GPU. Langkah keempat adalah membuat kernel GPU untuk *Cross Correlation* untuk melakukan estimasi pergeseran (gambar 2). Pada percobaan, peneliti menggunakan 256 *thread per block*. Langkah kelima adalah melakukan komputasi paralel IFFT pada GPU. Langkah keenam adalah memindahkan kembali hasil pergeseran dari GPU ke CPU. Langkah ketujuh adalah melakukan konversi kembali tipe data dari *single precision* ke *double precision*. Langkah terakhir adalah menggambarkan *motion vector*.

TABEL I
SPESIFIKASI FISIK GPU YANG DIGUNAKAN

Threads / Warp	32
Warps / Multiprocessor	24
Threads / Multiprocessor	768
Thread Blocks / Multiprocessor	8
Total # of 32-bit registers / Multiprocessor	8192
Register allocation unit size	256
Shared Memory / Multiprocessor (bytes)	16384
Warp allocation granularity (untuk alokasi register)	2

3. Hasil dan Pembahasan

Untuk melakukan manajemen terhadap ribuan *thread* yang berjalan bersamaan, multiprosesor menggunakan arsitektur yang dinamakan SIMT (*single instruction, multiple thread*). Multiprosesor memetakan tiap *thread* pada satu *core* prosesor skalar, dan tiap *thread* skalar dieksekusi tersendiri sesuai dengan alamat instruksinya dan *register* pada multiprosesor yang bersangkutan. Unit SIMT dari multiprosesor membuat, mengelola, menjadwal, dan mengeksekusi *thread* ke dalam *group* 32 *thread* paralel dan dinamakan *Warp*. *Maximum Occupancy* dari multiprosesor adalah rasio dari jumlah *Warp* aktif terhadap jumlah maksimum *Warp* yang dapat ditangani oleh satu multiprosesor GPU. Tiap multiprosesor memiliki sejumlah *N register* sebagai *resource* yang di-*sharing* dan dialokasikan pada *thread* di dalam *block* yang dieksekusi. *Compiler* berusaha untuk meminimalisasi penggunaan *register* untuk memaksimalkan jumlah *thread* yang aktif secara simultan. Jika program mencoba untuk memproses *kernel* dengan *register* yang digunakan tiap *thread* dikalikan dengan *thread block* lebih besar dari *N*, proses akan gagal. Ukuran *N* dari GPU yang digunakan pada penelitian ini adalah 8192 32-bit *register* tiap *multiprosesor*.

Dari spesifikasi fisik GPU yang ditunjukkan pada tabel I, peneliti dapat mengkalkulasi *maximum occupancy multiprocessor* sebagai berikut:

$$\lceil X_1 \rceil = \min\left\{ n \in Z \mid n \geq \frac{Thread\ \ per\ \ block}{thread\ \ per\ \ warp} \right\} \tag{15}$$

dengan x_1 = 8, Z = nilai *integers*, dan $x_1 \approx$ *Warp* tiap *thread block.*

$$\lceil X_2 \rceil = \min\left\{ n \in Z \mid n \geq x_1, warp\ \ allocation\ \ granullarity \right\}$$

maka

$$\lceil X_3 \rceil = \min\left\{ n \in Z \mid n \geq x_2 \times r_2 \times 32, y_2 \right\}$$

$$x_3 = 2304 \tag{16}$$

dengan r_1= *register per thread*, y_2= *ukuran unit register allocation*, $x_3 \approx$ *register tiap thread block.*

$$\lceil X_4 \rceil = \min\left\{ n \in Z \mid n \geq sharedmem.perthread, 512 \right\}$$

$$x_4 = 512 \tag{17}$$

dengan $x_4 \approx$ *shared memory tiap thread block*.

$$\lceil X_5 \rceil = \min\left\{ n \in Z \mid n \geq \frac{Limit\ \ warp\ \ per\ \ multiptocessor}{warp\ \ per\ \ block} \right\}$$

$$x_5 = 3 \tag{18}$$

dengan $x_5 \approx$ *Max. Warp tiap multiprocessor.*

$$\lceil X_6 \rceil = \min\left\{ n \in Z \mid n \geq \frac{Total\ \ register\ \ per\ \ multiptocessor}{register\ \ per\ \ thread\ \ block} \right\}$$

$$x_6 = 3 \tag{19}$$

dengan $x_6 \approx$ *Max. Register tiap multiprocessor*

$$\left\lceil X_7 \right\rceil = \max\left\{ n \in Z \mid n \le \frac{shared\ \ memory\ \ per\ \ multiptocessor}{shared\ \ memory\ \ per\ \ thread\ \ block} \right\}$$

$x_7 = 32$ (20)

dengan $x_7 \approx$ *Max. Shared memory per multiprocessor.*

Dengan mengambil nilai minimum dari persamaan 8, 9, dan 10, yang nilainya sama dengan *thread block* aktif per *multiprocessor*.

Active Warp per multiprocessor = Active thread block per multiprocessor X Warp per thread block = 24 (21)

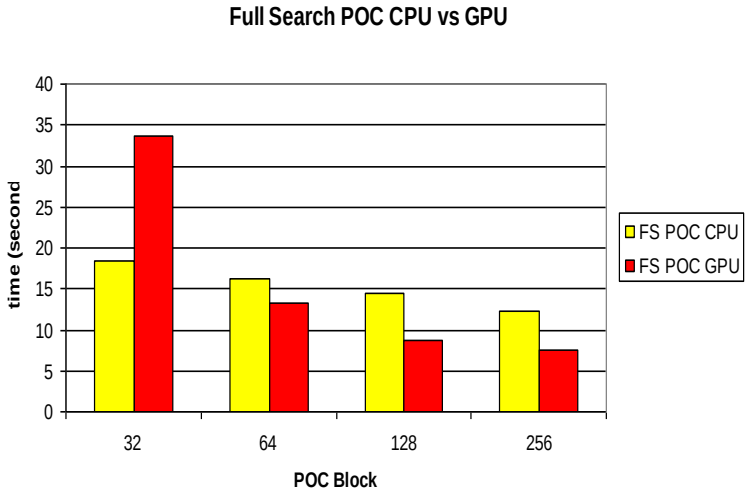
Active Threads per Multiprocessor = Active thread block per multiprocessor X thread per block = 768 (22)

Multiprocessor Maximum Occupancy = (Active Warp per multiprocessor / Warp per multiprocessor) X 100% = 100% (23)

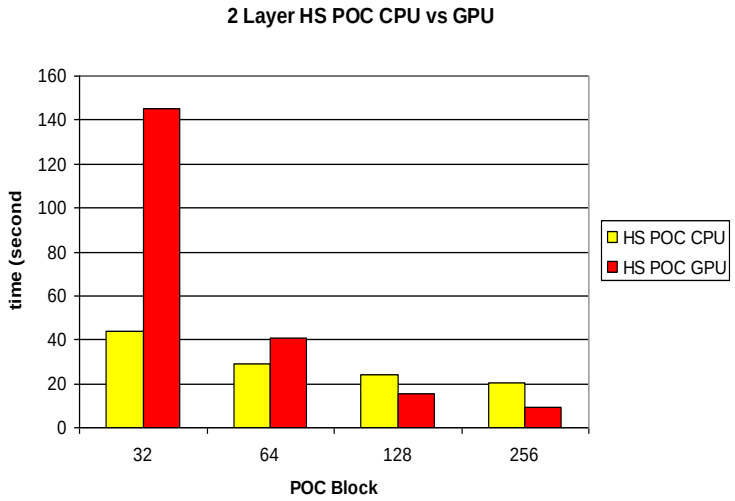
Gambar 4 dan 5 adalah hasil dari lamanya waktu komputasi dari *Full Search* dan *Hierarchical Search* 2 *layer* menggunakan blok POC berukuran 32x32 hingga 256x256. *Bar* kuning adalah waktu komputasi pada CPU dan *bar* berwarna merah adalah waktu komputasi pada GPU.

(a) (b) (c)

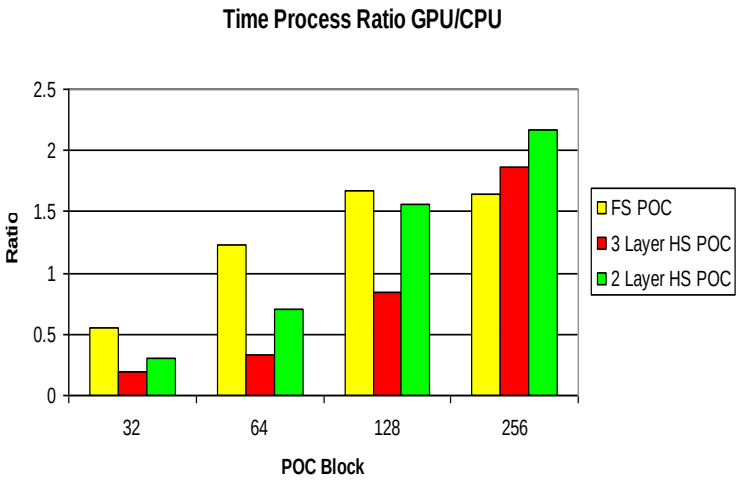
Gambar 3. (a) *Motion Vector* dari *Mobile Calendar* menggunakan blok berukuran 32x32 diambil dari *frame* referensi (b), dan *frame* saat ini (c).



Gambar 4. Waktu komputasi dari *Full Search* POC GPU dan CPU.



Gambar 5. *Processing time of hierarchical search* POC GPU dan CPU 2 *layer.*



Gambar 6. Rasio waktu komputasi antara *POC* full search, *hierarchical search* 3 layer, dan *hierarchical search 2 layer* pada GPU dan CPU.

Gambar 6 adalah hasil dari rasio waktu komputasi antara CPU dan GPU pada *hierarchical search* 2 *layer*, *hierarchical search 3 layer*, dan *full search* menggunakan blok POC berukuran 32x32 hingga 256x256. *Bar* kuning adalah rasio waktu komputasi pada *full search*, *bar* merah adalah rasio waktu komputasi pada *hierarchical* 3 *layer*, dan *bar* hijau adalah rasio waktu komputasi pada *hierarchical search* 2 *layer.*

4. Kesimpulan

Penelitian pada *paper* ini menyajikan percepatan *motion estimation* berbasis POC dengan teknik *hierarchical search* menggunakan GPU. Pada metode yang digunakan, hasil *motion vector* didapat dari proses hasil pergeseran blok citra pada GPU. Peneliti telah menunjukkan bahwa metode yang digunakan rata-rata dapat menyelesaikan komputasi 1,7 kali lebih cepat dibandingkan menggunakan metode yang sama pada CPU. Menggunakan GPU NVidia GeForce 9600GT, *kernel* dieksekusi dengan 256 *thread per block*, 9 32-bit *register per thread*, dan 36 *bytes memory shared* untuk tiap *thread block*, *maximum occupancy multiprocessor* adalah 100%, dengan 768 *threads active per multiprocessor*, 24 *Warps Active per multiprocessor*, dan 3 *thread blocks active per multiprocessor.*

Ucapan Terima Kasih

Peneliti mengucapkan terima kasih kepada Direktur Jenderal Pendidikan Tinggi, Direktorat Pendidikan Tinggi, Departemen Pendidikan Nasional Indonesia atas dukungan finansialnya sehingga penelitian ini dapat terselesaikan dengan baik.

Referensi

[1] L.H. Chien & Takafumi Aoki, "Robust Motion Estimation for Video Sequences Based on Phase-Only Correlation" *In Proceeding 6th IASTED International Conference Signal and Image Processing*, pp. 441-446, 2004.

[2] C.D. Kuglin & D.C. Hines, "The Phase Correlation Image Alignment Method" *In Proceeding of Cybernetics and Society*, pp. 163-165, 1975.

[3] K. Takita, M.A. Muquit, T. Aoki, & T. Higuchi, "High-Accuracy Subpixel Iimage Registration Based on Phase-only Correlation," *IEICE Transaction Fundamentals*, vol. E86-A, pp. 1925-1934, 2003.

[4] K. Takita, M. A. Muquit, T. Aoki, & T. Higuchi, "A Sub-pixel Correspondence Search Technique for Computer Vision Applications," *IEICE Transaction Fundamentals*, pp. 1913-1923, 2004.

[5] T. Klein, M. Strengert, S. Stegmaier, & T. Ertl, "Exploiting Frame-to-Frame Coherence for Accelerating High-Quality Volume Raycasting on Graphics Hardware," *IEEE Visualization 2005 (VIS'05)*, pp. 223-230, 2005.

[6] N. Vasconcelos, "Coarse-to-Fine Least Squares Motion Estimator," 1993.

[7] S.N. Sinha, J. Frahm, M. Pollefeys, & Y. Genc, "GPU-based Video Feature Tracking and Matching" *Workshop on Edge Comput*ing Using New C*ommodity Architectures*, pp. 695-699, 2006.

[8] E.P. Simoncelli, "Coarse-to-Fine Estimation of Visual Motion" *In Proceeding 8th Workshop on Image and Multidimensional Signal Processing in Cannes France*, pp. 128-129, 1993.

[9] J. Watkinson, *The Engineer's Guide to Motion Compensation*, Snell and Wilcox Handbook Series, Hampshire, 1994.

[10] NVidia Cuda Team, SC-07 Cuda Tutorial, 2007.

[11] GPGPU VISCOURSE05, International conference on GPGPU, Minneapolis USA, 2005.

[12] Matlab R2007a Documentation, 2007.

[13] Nvidia Developer Team, NVIDIA CUDA Programming Guide Version 2.0, 2008.

[14] Nvidia Developer Team, CUDA Reference Manual Version 2.0, 2008.

[15] Nvidia Developer Team, CUDA CUFFT Library Manual Version 2.0, 2008.

[16] Nvidia Developer Team, CUDA nvcc Manual Version 2.0, 2008.

[17] Nvidia Developer Team, Accelerating MATLAB with CUDA using MEX Files, In Nvidia White Paper, September 2007.

[18] Nvidia CUDA Forums, http://forums.nvidia.com/index.php?showforum=62, 2009, retrieved January 4, 2010.