JURNAL INFOTEL
Informatika - Telekomunikasi - Elektronika

Website Jurnal : http://ejournal.st3telkom.ac.id/index.php/infotel
ISSN : 2085-3688; e-ISSN : 2460-0997

Perbandingan Kemampuan *Embedded Computer* dengan *General Purpose Computer* Untuk Pengolahan Citra

Herryawan Pujiharsono
Program Studi S1 Teknik Telekomunikasi, ST3 Telkom Purwokerto
Jl. D.I. Panjaitan No. 128 Purwokerto 53147, Jawa Tengah, Indonesia
Email korespondensi: herryawan@st3telkom.ac.id

Dikirim 01 Juli 2017, Direvisi 01 Agustus 2017, Diterima 13 Agustus 2017

Abstrak – Perkembangan teknologi komputer membuat pengolahan citra saat ini banyak dikembangkan untuk dapat membantu manusia di berbagai bidang pekerjaan. Namun, tidak semua bidang pekerjaan dapat dikembangkan dengan pengolahan citra karena tidak mendukung penggunaan komputer sehingga mendorong pengembangan pengolahan citra dengan mikrokontroler atau mikroprosesor khusus. Perkembangan mikrokontroler dan mikroprosesor memungkinkan pengolahan citra saat ini dapat dikembangkan dengan *embedded computer* atau *single board computer* (SBC). Penelitian ini bertujuan untuk menguji kemampuan *embedded computer* dalam mengolah citra dan membandingkan hasilnya dengan komputer pada umumnya (*general purpose computer*). Pengujian dilakukan dengan mengukur waktu eksekusi dari empat operasi pengolahan citra yang diberikan pada sepuluh ukuran citra. Hasil yang diperoleh pada penelitian ini menunjukkan bahwa optimasi waktu eksekusi *embedded computer* lebih baik jika dibandingkan dengan *general purpose computer* dengan waktu eksekusi rata-rata *embedded computer* adalah 4-5 kali waktu eksekusi *general purpose computer* dan ukuran citra maksimal yang tidak membebani CPU terlalu besar untuk *embedded computer* adalah 256x256 piksel dan untuk *general purpose computer* adalah 400x300 piksel.

Kata kunci – waktu eksekusi, bitmap, linux, *flip*, binerisasi, inversi, *mean filter*, *single board computer* (SBC)

Abstract - The improvements in computer technology make image processing developed widely to help people in various areas of work. However, some work can't be developed with image processing because it doesn't support the use of computers and it encourages the development of image processing with a microcontroller or special microprocessor. The improvement of microcontrollers and microprocessors features currently allows image processing can be developed with embedded computers or single board computer (SBC). This study aims to test the performance of embedded computers for image processing and then compare the results with the performance of general purpose computer. The result shows that the optimized execution time of embedded computer is better than general purpose computer with the comparison of average execution time of embedded computer is 4-5 times slower than the general purpose computer and the maximal image size that does not make the CPU overload for embedded computer is 256x256 pixel and for general purpose computer is 400x300 pixels.

Keywords - execution time, bitmap, linux, flip, binarization, inversion, mean filter, single board computer (SBC)

I. PENDAHULUAN

Perkembangan teknologi komputer membuat pengolahan citra tidak dapat dipisahkan dengan bidang *computer vision.* Hal tersebut dapat terjadi karena seiring dengan semakin meningkatnya kapasitas dan kecepatan komputasi pada komputer, pengolahan citra yang pada awalnya hanya digunakan untuk memperbaiki kualitas citra telah berkembang menjadi salah satu bidang ilmu yang dapat mengambil informasi dari suatu citra. Kemampuan tersebut membuat pengolahan citra saat ini dapat digunakan untuk membantu manusia dalam mempercepat proses pekerjaannya atau sebagai *second opinion* sehingga dapat meminimalisir kesalahan dalam pengambilan keputusan akibat *human error*.

Bidang pekerjaan yang saat ini banyak dikembangkan dengan pengolahan citra adalah bidang kedokteran atau biomedis. Salah satu contohnya adalah

untuk membantu dokter dalam menentukan tingkatan dari suatu kanker seperti kanker payudara [1], kanker mulut [2], dan kanker hati [3]. Selain itu, bidang lain seperti pertanian dan keamanan juga sudah dikembangkan dengan pengolahan citra. Pada bidang pertanian, pengolahan citra digunakan untuk membantu petani dalam menentukan kualitas buah melalui warnanya [4], sedangkan pada bidang keamanan, pengolahan citra membantu dalam mengidentifikasi atau memverifikasi seseorang melalui tanda tangannya [5] atau ciri biometrika seperti wajah [6], iris mata [7], dan sidik jari [8].

Namun, terdapat juga beberapa bidang pekerjaan yang sulit dikembangkan dengan pengolahan citra karena tidak memungkinkan penggunaan komputer pada bidang tersebut, padahal bidang tersebut sebenarnya dapat dioptimalkan dengan menggunakan citra. Salah satu contoh profesi pada bidang pekerjaan tersebut adalah petugas baca meter air atau listrik pascabayar. Saat ini petugas tersebut harus berkeliling setiap bulannya dari rumah ke rumah untuk mencatat angka meter dan mengambil citra angka meter sebagai buktinya [9]. Adanya pengolahan citra sebenarnya dapat membuat proses pencatatan menjadi lebih efisien dari sisi waktu. Beberapa metode pengolahan citra untuk proses pengambilan citra angka meter tersebut juga sudah dikembangkan [10][11].

Salah satu hal yang menyebabkan penggunaan komputer tidak mudah diimplementasikan untuk pengembangan pengolahan citra pada profesi seperti tersebut adalah rendahnya tingkat portabilitas komputer karena komputer membutuhkan daya listrik yang tinggi untuk dapat beroperasi (*high-power*), sedangkan profesi seperti tersebut membutuhkan peralatan dengan tingkat portabilitas yang tinggi agar mudah dibawa dan digunakan di lapangan. Hal tersebut mendorong beberapa penelitian untuk merancang sistem pengolahan citra portabel dengan memanfaatkan mikrokontroler seperti mikrokontroler 8-bit ATmega32 [12], mikrokontroler 8-bit SX28AC [13], dan mikrokontroler 16-bit PIC24/dsPIC [14]. Salah satu kelebihan dari mikrokontroler adalah *low-power* sehingga dapat beroperasi hanya dengan baterai dan membuat tingkat portabilitasnya tinggi. Sayangnya, *resources* pada mikrokontroler seperti kapasitas memori dan *clock* yang sangat rendah membuat pengolahan citra tidak optimal dan sangat terbatas sehingga perlu ditambahkan komponen seperti RAM eksternal [12] atau co-prosesor SSD 1928 [14].

Seiring dengan perkembangan teknologi mikrokontroler dan mikroprosesor, keterbatasan mikrokontroler tersebut saat ini dapat ditingkatkan dengan penggunaan *embedded computer* atau *single board computer* (SBC), yaitu sebuah *board* dengan spefisikasi mendekati komputer pada umumnya (*general purpose computer*) tetapi penggunaannya lebih dikhususkan untuk pengembangan *embedded system* [15]. Beberapa jenis perangkat *embedded computer* yang cukup terkenal adalah Raspberry pi, Beaglebone, dan Cubieboard [15] dimana ketiganya memiliki spefisikasi yang hampir sama, yaitu berisi mikroprosesor 32-bit dengan *clock* yang mampu mencapai 1 GHz dan kapasitas memori mencapai 1 GB serta sudah mendukung kernel linux sehingga dapat diintegrasikan dengan sistem operasi layaknya komputer pada umumnya [16]. *Resources* pada *board* tersebut membuat *embedded computer* memungkinkan untuk menggantikan peran *general purpose computer* untuk mengolah citra sehingga pekerjaan-pekerjaan yang membutuhkan pengolahan citra tetapi tidak memungkinkan terpasangnya komputer dapat digantikan dengan *embedded computer* tersebut.

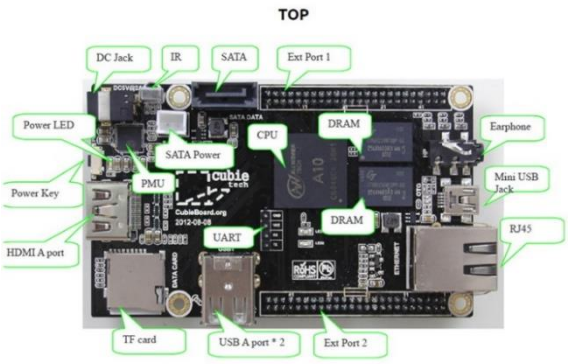
Namun, penggunaan *embedded computer* sebagai pengolah citra perlu diuji dan dibandingkan dengan *general purpose computer* untuk memberi gambaran kemampuan *embedded computer* dalam melakukan beberapa operasi pengolahan citra sehingga batasan kemampuan dari *embedded computer* sebagai pengolah citra dapat diketahui.

II. METODE PENELITIAN

A. Spesifikasi Perangkat

Perangkat yang digunakan untuk pengujian pada penelitian ini adalah Cubieboard, seperti ditunjukkan pada Gambar 1, yang merupakan produk dari Cubietech dengan spesifikasi sebagai berikut [17].

- Mikroprosesor A10 dari AllWinnerTech SOC yang berisi ARM Cortex-A8
- GPU ARM Mali 400 MP1 yang mendukung OpenGL ES 2.0/1.1
- 1GB DDR3 @480MHz
- 4GB internal NAND flash, up to 32GB on SD slot, up to 2T on 2.5 SATA disk
- 5VDC input 2A or USB OTG input
- 1x 10/100 ethernet, mendukung usb wifi
- 2x USB 2.0 HOST, 1x mini USB 2.0 OTG, 1x micro SD
- 1x HDMI 1080P display output
- 1x IR, 1x line in, 1x line out
- 96 pin eksternal yang dapat digunakan untuk I2C, SPI, RGB/LVDS, CSI/TS, FM-IN, ADC, CVBS, VGA, SPDIF-OUT, R-TP



Gambar 1. Cubieboard Tampak Atas [17]

Sistem operasi linux untuk Cubieboard pada penelitian ini adalah Cubieez [18] dari Distro Debian yang memang dikembangkan khusus untuk Cubieboard.

Sebagai pembanding pengujian kemampuan pengolahan citra, *general purpose computer* yang digunakan adalah komputer laptop Toshiba C640 dengan spesifikasi utama sebagai berikut.

- Prosesor Intel Pentium Dual-Core P6100 (2Ghz, Cache 3 MB)
- 3 GB DDR3 SDRAM
- GPU Intel Graphics Media Accelerator

Sistem operasi yang digunakan pada komputer tersebut adalah sistem operasi linux dari Distro Debian. Penggunaan distro linux yang sama dimaksudkan agar proses yang dilalui sama sehingga tidak ada perbedaan perlakukan dalam pengujian.

B. Operasi Pengolahan Citra

Citra yang diujikan adalah citra bitmap (BMP) *grayscale* 8-bit yang tidak terkompres. Pengujian ini menggunakan citra seperti ditunjukkan pada Gambar 2 dengan sepuluh ukuran citra yang berbeda-beda, yaitu 50 x 50 piksel, 128 x 128 piksel, 240 x 210 piksel, 249 x 224 piksel, 256 x 256 piksel, 332 x 330 piksel, 400 x 300 piksel, 512 x 512 piksel, 800 x 600 piksel, dan 1024 x 768 piksel.

Gambar 2. Citra Asli

Operasi pengolahan citra yang diujikan terdiri dari operasi sederhana berupa *load dan save* citra, dan operasi yang lebih kompleks seperti *flip* citra, binerisasi dan inversi citra, dan *mean filter*.

a) Operasi *Load* dan *Save*

Operasi *load* digunakan untuk mengambil data piksel citra pada *file* yang berekstensi bitmap. Urutan proses dari operasi *load* tersebut pada penelitian ini dapat dilihat pada Gambar 3. Proses operasi *load* dimulai dengan membuka *file* berekstensi bitmap dalam bentuk biner. Selanjutnya, data sebesar 54 byte pertama diambil dan disimpan karena data tersebut berisi *header* dari bitmap yang memuat informasi terkait data piksel citra seperti ditunjukkan pada Tabel 1. Setelah data *header* tersimpan, dilakukan pengecekan untuk *bitmap signature* pada dua byte pertama dari *header* tersebut. Jika *signature* sesuai dengan *bitmap signature*, yaitu karakter BM, proses selanjutnya adalah mengambil nilai lebar citra (*L*) pada byte ke-18 sampai byte ke-21 dari *header* tersebut dan tinggi citra (*T*) pada byte ke-22 sampai byte ke-25 untuk kemudian dilakukan perhitungan alokasi memori untuk menampung data piksel citra berdasarkan Persamaan (1).

$$Memori = L\ (px) \times T\ (px) \quad (1)$$

dimana:

L (px): Lebar citra (piksel)

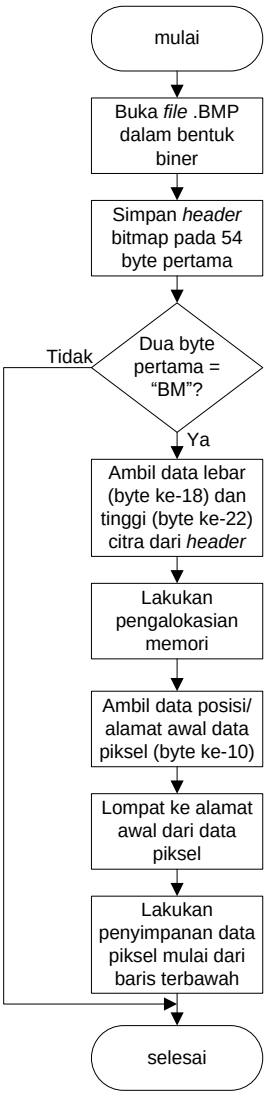
T (px): Tinggi citra (piksel)

Tabel 1. *Bitmap Header* [19]

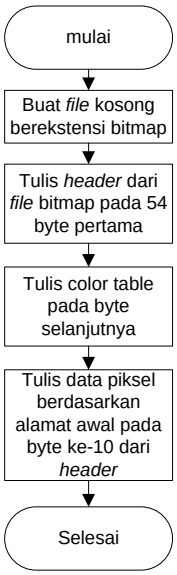
Informasi	Ukuran (byte)
Bitmap signature ("BM" atau 0x4D42)	2
Ukuran *bitmap file*	4
Reserved (bernilai 0)	4
Posisi atau alamat awal dari *image data*	4
Ukuran lebar citra sampai jumlah warna penting	4
Lebar citra (piksel)	4
Tinggi citra (piksel)	4
Jumlah *planes* warna (bernilai 1)	2
Jumlah bit per piksel	2
Tipe kompresi	4
Ukuran *image data* dalam byte	4
Resolusi horisontal (piksel/meter)	4
Resolusi vertikal (piksel/meter)	4
Jumlah warna terpakai	4
Jumlah warna yang penting	4

Setelah itu, data pada byte ke-10 sampai ke-14 diambil untuk menunjukkan posisi atau alamat awal dari data piksel citra berada. Proses pengambilan data piksel citra kemudian dilakukan berdasarkan alamat awal tersebut untuk selanjutnya disimpan pada memori yang telah dialokasikan. Hal yang perlu diperhatikan dalam mengambil dan menyimpan data piksel citra adalah urutan piksel per baris dimana urutan penyimpanan per barisnya terbalik, yaitu dimulai dari baris bawah ke atas, agar hasilnya tidak terbalik.

Operasi selanjutnya adalah operasi *save* yang berfungsi untuk untuk menyimpan data citra ke bentuk *file* dengan ekstensi bitmap. Gambar 4 menunjukkan proses dari operasi *save* yang dilakukan pada penelitian ini. Berdasarkan Gambar 4, proses dari operasi *save* dimulai dengan membuat *file* kosong, kemudian data *bitmap header* yang diperoleh dari hasil operasi *load* dituliskan pada 54 byte pertama dari *file* tersebut agar *file* yang dibuat menjadi *file* dengan ekstensi bitmap. Setelah itu, proses penulisan *color table* dilakukan pada byte setelah *bitmap header* untuk memetakan urutan semua warna yang digunakan pada data piksel citra [19]. Selanjutnya, penulisan data piksel citra dapat dilakukan dimulai dari alamat awal yang ditunjukkan oleh byte ke-10 pada *bitmap header* dengan memperhatikan urutan baris piksel.



Gambar 3. Operasi *Load* Citra Bitmap



Gambar 4. Operasi *Save* Citra Bitmap

b) Operasi *Flip* Citra

Operasi *flip* citra memiliki dua jenis, yaitu *flip* vertikal dan *flip* horisontal seperti ditunjukkan oada Gambar 5. Proses *flip* pada citra dilakukan dengan mengubah urutan piksel pada memori. *Flip* vertikal mengubah urutan piksel per kolom dari kiri ke kanan menjadi kanan ke kiri, sedangkan proses *flip* horisontal merubah urutan piksel per baris dari atas ke bawah menjadi bawah ke atas.

Gambar 5.(a). Flip Horisontal

Gambar 5.(b). Flip Vertikal

c) Operasi Binerisasi dan Inversi Citra

Operasi binerisasi citra dilakukan dengan membandingkan nilai setiap piksel dengan batas ambang (*threshold*) untuk menghasilkan citra hitam putih dari citra *grayscale* seperti ditunjukkan pada Gambar 6.(*a*). Operasi binerisasi citra dapat dinyatakan dengan Persamaan (2). Nilai *threshold* yang digunakan pada penelitian ini adalah 128 yang merupakan nilai tengah antara 0 (hitam) dan 255 (putih).

$$g(x,y) = \begin{cases} 0, f(x,y) < Threshold \\ 255, f(x,y) \geq Threshold \end{cases} \quad (2)$$

dimana:

$g(x,y)$ = intensitas *grayscale* pada piksel citra hasil binerisasi di koordinat (x,y)

$f(x,y)$ = intensitas *grayscale* pada piksel citra asli di koordinat (x,y)

x = baris ke-x

y = kolom ke-y

Citra yang telah terbinerisasi tersebut dapat dilakukan inversi dengan membalikkan nilai 0 menjadi 255 dan sebaliknya. Hasil dari proses inversi ini akan mengubah warna hitam menjadi putih dan sebaliknya pada citra seperti ditunjukkan pada Gambar 6.(*b*).

Gambar 6.(a) Binerisasi Citra

Gambar 6.(*b*). Inversi Citra

d) Operasi *Mean Filter*

Filter pada citra umumnya digunakan untuk menghilangkan *noise* yang terdapat citra. *Noise* pada citra berbentuk variasi intensitas yang bersifat acak yang mengganggu sehingga citra tidak dapat diolah langsung. *Noise* pada citra pun memiliki berbagai jenis sehingga terdapat banyak pula jenis *filter* yang dapat digunakan.

Jenis *filter* yang diuji pada penelitian ini adalah *mean filter* karena jenis *filter* ini sering digunakan

di beberapa pengolahan citra. *Filter* ini menggunakan teknik *windowing* pada citra, yaitu mengkotak-kotakkan citra berukuran M x N ke dalam ukuran lebih kecil *(kernel window*) dan setiap *kernel window* dihitung nilai rata-ratanya untuk menggantikan nilai pada piksel tengah *window* [20]. Urutan proses dari operasi *mean filter* adalah sebagai berikut.

Langkah 1. *Windowing* citra menjadi ukuran kecil, penelitian ini menggunakan *kernel window* dengan ukuran 5x5 mulai dari piksel ke (0,0) sampai (4,4)

Langkah 2. Hitung nilai rata-rata dari semua nilai piksel yang terdapat pada *window* tersebut untuk *mean filter*

Langkah 3. Ganti nilai pada piksel tengah *window*, yaitu piksel ke (2,2) dengan nilai nilai yang diperoleh pada langkah 2.

Langkah 4. Geser *window* 1 kolom dan ulangi langkah ke-2 dan ke-3 sampai kolom terakhir

Langkah 5. Geser *window* 1 baris ke bawah dan kembali ke kolom pertama kemudian ulangi langkah ke-2 sampai 4 sampai baris terakhir.

Urutan proses tersebut menghasilkan citra seperti ditunjukkan pada Gambar 7. Berdasarkan prosesnya tersebut, operasi *mean filter* membutuhkan komputasi yang lebih tinggi jika dibandingkan dengan operasi *flip* atau binerisasi dan inversi citra sehingga pada penelitian ini dijadikan sebagai operasi yang mempunyai tingkat kompleksitas yang paling tinggi.

Gambar 7. Citra Hasil Operasi *Mean Filter*

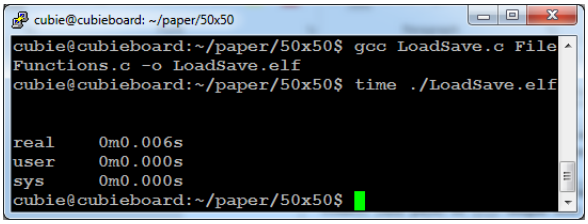
C. *Parameter Pengujian*

Parameter yang diuji untuk mengukur kemampuan dari *embedded system* dalam mengolah citra adalah waktu eksekusi. Pengukuran waktu eksekusi dapat dilakukan dengan beberapa metode seperti ditunjukkan pada Tabel 2.

Tabel 2. Metode Pengukuran Waktu Eksekusi [21]

Metode	Tingkat Resolusi	Akurasi	*Granularity*	Tingkat kesulitan penggunaan
Stop-watch	0,01 s	0,5 s	Program	Mudah
Date	0,02 s	0,2 s	Program	Mudah
Time	0,02 s	0,2 s	Program	Mudah
Prof and gprof	10 ms	20 ms	Subrutin	Sedang
Clock ()	15-30 ms	15-30 ms	*Statement*	Sedang
Software analysers	10 µs	20 µs	Subrutin	Sedang
Timer/ counter chips	0,5 – 4 µs	1-8 µs	*Statement*	Sangat sulit
Logic or bus analyzer	50 ns	0,5 µs	*statement*	sulit

Metode yang dipilih pada penelitian ini adalah *time*. Metode ini memberikan estimasi waktu kasar dari jalannya sebuah kode program (*coarse-grain*) [21]. Proses pengukuran waktu eksekusi dengan menggunakan metode *time* dapat dilihat pada Gambar 8.



Gambar 8. Proses Menggunakan Metode *Time*

Proses pertama adalah proses kompilasi kode program untuk menghasilkan *file* program yang dapat dijalankan (*executable file*). Operasi pengolahan citra ini ditulis menggunakan Bahasa C sehingga kompiler yang digunakan adalah GCC yang merupakan kompiler bawaan dari sistem operasi linux. Selanjutnya, proses pengukuran dilakukan dengan menambahkan perintah *time* di depan *executable file* yang akan dijalankan pada *command line* tersebut. Hasil yang diperoleh dari proses pengukuran tersebut terdiri dari tiga informasi, yaitu *real*, *user*, dan *sys*. *Real* menunjukkan waktu total yang dibutuhkan program untuk berjalan dari awal dari akhir (termasuk ketika program harus menunggu antrian untuk dieksekusi oleh CPU atau oleh bagian Input-Output), sedangkan *user* dan *sys* menunjukkan waktu total yang hanya dibutuhkan oleh CPU dalam mengeksekusi program [21][22].

Berdasarkan hal tersebut, terdapat dua parameter waktu eksekusi yang dapat digunakan pada penelitian ini, yaitu waktu CPU (*user* + *sys*) untuk mengukur seberapa berat proses komputasi program dan waktu *real* untuk mengukur waktu total yang dibutuhkan program secara keseluruhan. Contoh pada Gambar 8 menunjukkan bahwa waktu *real* adalah 6 ms, sedangkan waktu CPU adalah 0 ms. Nilai 0 ms yang diperoleh pada Tabel 3 menunjukkan bahwa waktu eksekusi yang dibutuhkan mempunyai nilai di bawah 1 ms. Hal tersebut dikarenakan oleh resolusi untuk perintah *time* berdasarkan pada Tabel 2 hanya terbatas dalam satuan ms sehingga nilai di bawah 1 ms akan terbaca 0 ms.

III. HASIL DAN PEMBAHASAN

Operasi yang pertama diujikan adalah operasi *load* dan *save* citra. Pengujian dilakukan dengan mengambil data pada citra bitmap dan kemudian menyimpannya kembali dengan nama *file* yang berbeda tanpa mengubah informasi dan nilai piksel pada citra bitmap tersebut. Hasil pengukuran waktu eksekusi dari pengujian pertama ini pada sepuluh citra dengan ukuran berbeda ditunjukkan pada Tabel 3.

Tabel 3. Hasil Pengukuran Waktu Eksekusi Pengujian Pertama

Ukuran Citra	Perangkat	Real (ms)	CPU (ms)
50x50	*General Purpose Computer*	2	0
	Embedded Computer	6	0
128x128	*General Purpose Computer*	2	0
	Embedded Computer	10	0
240x210	*General Purpose Computer*	3	0
	Embedded Computer	13	0
249x224	*General Purpose Computer*	3	0
	Embedded Computer	13	0
256x256	*General Purpose Computer*	3	0
	Embedded Computer	14	0
332x330	*General Purpose Computer*	4	0
	Embedded Computer	17	8
400x300	*General Purpose Computer*	5	0
	Embedded Computer	21	0
512x512	*General Purpose Computer*	7	4
	Embedded Computer	37	10
800x600	*General Purpose Computer*	10	8
	Embedded Computer	48	30
1024x768	*General Purpose Computer*	15	12
	Embedded Computer	71	30

Tabel 3 juga memberikan informasi bahwa rata-rata besarnya waktu *real* yang dibutuhkan *embedded computer* untuk menjalankan operasi *load* dan *save* adalah 4,5 kali waktu yang dibutuhkan oleh *general purpose computer*. Selain itu, Tabel 3 juga menunjukkan bahwa ukuran citra yang disarankan agar operasi *load* dan *save* optimal untuk kedua perangkat tersebut adalah 512x512 piksel. Hal tersebut dapat dilihat dari waktu CPU yang dibutuhkan baik untuk *embedded computer* maupun *general purpose computer* dimana waktu CPU mulai mengalami peningkatan secara signifikan ketika ukuran diubah dari 512x512 piksel menjadi 800 x 600 piksel, yang berarti proses komputasi yang dilakukan CPU sudah semakin berat.

Pengujian kedua dilakukan dengan menambahkan operasi *flip* vertikal dan horisontal setelah operasi *load* citra. Selanjutnya, hasil operasi *flip* tersebut masing-masing disimpan dengan nama yang berbeda dengan menggunakan menggunakan operasi *save*. Tabel 4 menunjukkan hasil pengukuran waktu eksekusi dengan penambahan operasi *flip* vertikal dan horisontal pada citra.

Tabel 4. Hasil Pengukuran Waktu Eksekusi Pengujian Kedua

Ukuran Citra	Perangkat	Real (ms)	CPU (ms)
50x50	*General Purpose Computer*	3	0
	Embedded Computer	7	0
128x128	*General Purpose Computer*	3	0
	Embedded Computer	8	4
240x210	*General Purpose Computer*	5	4
	Embedded Computer	14	10
249x224	*General Purpose Computer*	5	4
	Embedded Computer	14	10
256x256	*General Purpose Computer*	6	4
	Embedded Computer	14	10
332x330	*General Purpose Computer*	7	4
	Embedded Computer	20	10
400x300	*General Purpose Computer*	8	4
	Embedded Computer	22	10
512x512	*General Purpose Computer*	11	12
	Embedded Computer	36	30
800x600	*General Purpose Computer*	14	12
	Embedded Computer	115	60
1024x768	*General Purpose Computer*	23	20
	Embedded Computer	150	100

Berdasarkan Tabel 4, penambahan operasi *flip* membuat rata-rata rasio waktu *real* antara *embedded computer* dan *general purpose computer* menjadi lebih rendah jika dibandingkan dengan operasi *load* dan *save*, yaitu menjadi 3,7 kali. Hal ini terjadi karena rata-rata peningkatan waktu *real* untuk *general purpose computer* lebih besar jika dibandingkan dengan waktu *real* untuk *embedded computer*. Waktu *real* untuk *general purpose computer* rata-rata meningkat sebesar 1,6 kali jika dibandingkan dengan operasi *load* dan save, sedangkan untuk *embedded computer* hanya meningkat sebesar 1,2 kali lipat. Selain itu, ukuran citra yang optimal untuk kedua perangkat tersebut adalah 400x300 piksel karena peningkatan waktu CPU yang cukup signifikan terjadi saat ukuran diubah dari 400x300 piksel menjadi 512x512 piksel.

Pengujian selanjutnya adalah mengganti operasi *flip* dengan operasi binerisasi dan inversi citra. Operasi ini mempunyai tingkat komputasi yang lebih tinggi jika dibandingkan dengan operasi *flip* karena nilai setiap piksel perlu diolah terlebih dahulu sesuai dengan Persamaan (2) untuk menjadi citra hitam putih sebelum disimpan dengan operasi *save* dan diinversikan. Hasil pengukuran waktu eksekusi untuk pengujian ketiga ini ditunjukkan pada Tabel 5.

Tabel 5. Hasil Pengukuran Waktu Eksekusi Pengujian Ketiga

Ukuran Citra	Perangkat	Real (ms)	CPU (ms)
50x50	*General Purpose Computer*	3	0
	Embedded Computer	7	0
128x128	*General Purpose Computer*	3	0
	Embedded Computer	8	0
240x210	*General Purpose Computer*	5	4
	Embedded Computer	14	10
249x224	*General Purpose Computer*	5	4
	Embedded Computer	15	10
256x256	*General Purpose Computer*	6	4
	Embedded Computer	15	10
332x330	*General Purpose Computer*	7	4
	Embedded Computer	20	20
400x300	*General Purpose Computer*	8	4
	Embedded Computer	23	20
512x512	*General Purpose Computer*	17	12
	Embedded Computer	42	30
800x600	*General Purpose Computer*	19	20
	Embedded Computer	129	70
1024x768	*General Purpose Computer*	25	24
	Embedded Computer	153	100

Tabel 5 menunjukkan bahwa rata-rata kenaikan waktu eksekusi untuk *general purpose computer* adalah 1,7 kali waktu eksekusi operasi *load* dan *save*, sedangkan untuk *embedded computer* adalah 1,4 kali. Kenaikan waktu eksekusi tersebut membuat rata-rata rasio waktu eksekusi antara *embedded computer* dengan *general purpose computer* menjadi 3,4 kali. Tabel 5 juga menunjukkan bahwa ukuran citra yang menyebabkan peningkatan waktu CPU yang signifikan untuk *embedded computer* dan *general purpose computer* tidak sama, dimana untuk *embedded computer* terjadi ketika perubahan ukuran dari 256x256 piksel ke 332x330 piksel, sedangkan untuk *general purpose computer* dari 400x300 piksel ke 512x512 piksel. Hal tersebut membuat ukuran yang disarankan untuk *embedded computer* adalah 256x256 piksel dan untuk *general purpose computer* adalah 400x300 piksel.

Pengujian yang terakhir dilakukan adalah operasi *load* dan *save* yang ditambahkan dengan operasi *mean filter*. Pada penelitian ini, operasi *mean filter* dijadikan sebagai operasi yang mempunyai tingkat kompleksitas yang paling tinggi karena berdasarkan urutan prosesnya, operasi *mean filter* membutuhkan komputasi yang lebih tinggi jika dibandingkan dengan operasi *flip* ataupun binerisasi dan inversi citra sehingga. Tabel 6 menunjukkan hasil pengukuran waktu eksekusi untuk pengujian terakhir ini.

Tabel 6. Hasil Pengukuran Waktu Eksekusi Pengujian Keempat

Ukuran Citra	Perangkat	Real (ms)	CPU (ms)
50x50	*General Purpose Computer*	3	0
	Embedded Computer	24	0
128x128	*General Purpose Computer*	6	4
	Embedded Computer	16	10
240x210	*General Purpose Computer*	11	8
	Embedded Computer	38	30
249x224	*General Purpose Computer*	12	12
	Embedded Computer	50	40
256x256	*General Purpose Computer*	15	12
	Embedded Computer	68	50
332x330	*General Purpose Computer*	18	20
	Embedded Computer	110	60
400x300	*General Purpose Computer*	24	20
	Embedded Computer	137	70
512x512	*General Purpose Computer*	35	30
	Embedded Computer	170	160
800x600	*General Purpose Computer*	56	56
	Embedded Computer	304	40
1024x768	*General Purpose Computer*	87	80
	Embedded Computer	469	450

Tabel 6 menunjukkan bahwa operasi *mean filter* membuat kenaikan waktu eksekusi yang cukup besar, yaitu 4,3 kali untuk *general purpose computer* dan 4,8 kali untuk *embedded computer* jika dibandingkan dengan operasi *load* dan *save* pada pengujian pertama. Kenaikan waktu eksekusi yang cukup besar tersebut berefek pada meningkatnya rata-rata rasio waktu eksekusi antara *embedded computer* dengan *general purpose computer*, yaitu menjadi 5 kali. Ukuran citra yang optimal untuk operasi *mean filter* ini adalah 128x128 piksel untuk kedua perangkat tersebut karena ketika ukuran diubah menjadi 240x210 piksel, waktu CPU untuk kedua perangkat tersebut mengalami peningkatan secara signifikan.

IV. PENUTUP

A. Kesimpulan

Berdasarkan hasil yang diperoleh, dapat diambil kesimpulan sebagai berikut. Optimasi waktu eksekusi *embedded computer* lebih baik jika dibandingkan dengan *general purpose computer* karena peningkatan waktu eksekusi *general purpose computer* pada setiap operasi pengolahan citra selalu lebih besar jika dibandingkan dengan *embedded computer*. Besarnya waktu eksekusi *embedded computer* rata-rata adalah 4-5 kali waktu eksekusi *general purpose computer*. Ukuran citra yang membebani CPU terlau besar untuk operasi sederhana adalah 256x256 piksel untuk *embedded computer* dan 400x300 piksel untuk *general purpose computer*, sedangkan untuk operasi yang

kompleks adalah 128x128 piksel untuk kedua perangkat tersebut.

B. Saran

Beberapa saran yang dapat dipertimbangkan untuk pengembangan penelitian ini antara lain. Metode yang digunakan untuk pengukuran waktu eksekusi sebaiknya mempunyai resolusi yang lebih tinggi lagi agar waktu eksekusi dapat diamati hingga satuan µs atau ns. Perbandingan kemampuan beberapa jenis *embedded computer* dalam mengolah citra sebaiknya perlu dilakukan untuk mendapatkan spesifikasi yang sesuai untuk pengolahan citra. Pengukuran konsumsi daya dan memori juga sebaiknya dilakukan untuk mendapatkan kemampuan *embedded computer* yang lebih detail dalam melakukan proses pengolahan citra

DAFTAR PUSTAKA

[1] J. Tang, R. M. Rangayyan, J. Xu, I. El Naqa, and Y. Yang, "Computer-Aided Detection and Diagnosis of Breast Cancer With Mammography: Recent Advances," *IEEE Trans. Inf. Technol. Biomed.*, vol. 13, no. 2, pp. 236–251, 2009.

[2] K. Anuradha, K. Sankaranarayanan, "A Review on Computer Aided Detection Techniques of Oral Cancer," *Int. J. Comput. Sci. Eng.*, vol. 2, no. 3, pp. 109–114, 2014.

[3] P. R. Anisha, C. K. K. Reddy, and L. V. N. Prasad, "A Pragmatic Approach for Detecting Liver Cancer using Image Processing and Data Mining Techniques," *2015 International Conference on Signal Processing and Communication Engineering Systems*. pp. 352–357, 2015.

[4] M. R. Satpute and S. M. Jagdale, "Automatic Fruit Quality Inspection System," *2016 International Conference on Inventive Computation Technologies (ICICT)*, vol. 1. pp. 1–4, 2016.

[5] J. Arifin and M. Z. Naf 'an, "Verifikasi Tanda Tangan Asli Atau Palsu Berdasarkan Sifat Keacakan (Entropi)," *J. Infotel*, vol. 9, no. 1, pp. 130–135, 2017.

[6] N. T. Deshpande and S. Ravishankar, "Face Detection and Recognition Using Viola-Jones Algorithm and Fusion of LDA and ANN," *Adv. Comput. Sci. Technol.*, vol. 10, no. 5, pp. 1173–1189, 2017.

[7] D. Kurnianto, I. Soesanti, and H. A. Nugroho, "Deteksi Iris Berdasarkan Metode Black Hole dan Circle Curve Fitting," *J. Infotel*, vol. 5, no. 2, pp. 10–16, 2013.

[8] Jyoti and J. Singh, "A Survey on Fingerprint Recognition Methods," *J. Netw. Commun. Emerg. Technol.*, vol. 7, no. 5, pp. 36–41, 2017.

[9] PT. PLN, "Term Of Reference (TOR) Pekerjaan Jasa Manajemen Billing PT. PLN Distribusi Jawa Tengah dan D.I. Yogyakarta Tahun 2013 s/d 2018." Semarang, 2013.

[10] A. Sudiarso and R. J. Merischaputri, "An Automation of Electricity Usage Reading on Postpaid kWh Meter using Kohonen-Type Artificial Neural Network," *Int. J. Mining, Metall. Mech. Eng.*, vol. 1, no. 4, pp. 238–240, 2013.

[11] H. Pujiharsono, H. A. Nugroho, and O. Wahyunggoro, "The Stand Meter Extraction of kWh-meter," *2015 International Conference on Science in Information Technology (ICSITech)*. pp. 202–206, 2015.

[12] A. Aggarwal, "Embedded Vision System (EVS)," *Mechtronic and Embedded Systems and Applications, 2008. MESA 2008. IEEE/ASME International Conference on*. pp. 618–621, 2008.

[13] E. Mozef, "Sistem Pengolahan Citra Stand-Alone Ekonomis Berbasis Mikrokontroler," *J. Tek. Elektro*, vol. 2, no. 1, pp. 32–38, 2002.

[14] P. Premaratne, S. Ajaz, R. Monaragala, N. Bandara, and M. Premaratne, "Design and Implementation of Edge Detection Algorithm in dsPIC Embedded Processor," *Information and Automation for Sustainability (ICIAFs), 2010 5th International Conference on*. pp. 8–13, 2010.

[15] U. Isikdag, "Internet of Things: Single-Board Computers," in *Enhanced Building Information Models: Using IoT Services and Integration Patterns*, Springer International Publishing, 2015, pp. 43–53.

[16] D. Y. Shi and W. S. Gan, "Comparison of Different Development Kits and Its Suitability in Signal Processing Education," *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. pp. 6280–6284, 2016.

[17] S. Chandrasekaran, "Cubieboard a $ 50 Single Board Computer," *EmbedJournal.com*, 2013. [Online]. Available: https://embedjournal.com/cubieboard/. [Accessed: 23-Mar-2017].

[18] Cubie Team, "Cubieez," *cubieboard.org*. [Online]. Available: http://dl.cubieboard.org/model/CubieBoard1/Image/cubieez/cubieez-hdmi-v2.0/. [Accessed: 27-Jun-2017].

[19] J. Miano, "Windows BMP," in *Compressed Image File Formats: JPEG, PNG, GIF, XBM, BMP*, Kanada: ACM Press, 1999, pp. 23–30.

[20] B. Fu, X. Xiong, and G. Sun, "An Efficient Mean Filter Algorithm," *The 2011 IEEE/ICME International Conference on Complex Medical Engineering*. pp. 466–470, 2011.

[21] D. B. Stewart, "Measuring Execution Time and Real-Time Performance," in *Embedded Systems Conference (ESC SF)*, 2002, no. September, pp. 1–15.

[22] S. Lakshman and S. Tushar, *Linux Shell Scripting Cookbook*, Second edi. Birmingham, UK: Packt Publishing Ltd., 2013.