IMPLEMENTASI PEMROGRAMAN PARALEL MENGGUNAKAN *PLATFORM* OPENMP PADA CITRA DIGITAL DENGAN METODE *LOW-PASS FILTER* DAN *HISTOGRAM EQUALIZATION*

Ade Bastian[1], **Dadan Zaliluddin**[2],, **Muhammad Syifa Al Maroghi**[3]
[12,3]*Program Studi Informatika, Fakultas Teknik, Universitas Majalengka*
Email: [1]adebastian@unma.ac.id, [2]dadanzuu@gmail.com, [3]almaroghi13@gmail.com

ABSTRAK
Teknik untuk mengurangi noise pada citra digital agar citra digital terlihat lebih halus dapat menggunakan metode Low-Pass Filter, karena metode ini dapat menghaluskan transisi tajam pada citra digital serta mengurangi noise yang ada. Teknik pengolahan citra digital yang memperbaiki kualitas cahaya pada citra digital sehingga mudah diinterpretasikan oleh mata manusia dapat menggunakan metode Histogram Equalization, karena metode ini memungkinkan area kontras yang rendah menjadi lebihtinggi, sehingga nilai derajat keabuan pada suatu citra digital menjadi rata, serta menjadikan perbaikan kontras yang lebih efektif. Pada tulisan ini, diterapkan teknik paralel sehingga pemrosesan citra digital berjalan lebih cepat. Teknik pemrograman paralel menggunakan platform OpenMP yang diterapkan mampu mempercepat proses pengolahan citra digital dengan rata-rata sebesar 1.7x lebih cepat sehingga proses pengolahan citra digital yang dilakukan lebih menghemat waktu.

Kata Kunci: Pemrograman Paralel, OpenMP, Pengolahan Citra Digital, Low-Pass Filter, Histogram Equalization

1. PENDAHULUAN

1.1. Latar Belakang

Citra sebagai komponen multimedia mempunyai peran penting sebagai bentuk informasi visual. Dibandingkan dengan teks, informasi yang dimiliki citra lebih kaya dan kompleks. Meskipun demikian, citra seringkali mengalami penurunan kualitas (degradasi) karena *noise*, warna terlalu kontras, *blurring*, kurang tajam, dan sebagainya. (Ricky Aprias Sholihin, 2014). Citra adalah biner dari citra dua dimensi berupa elemen berbentuk *array* yang disebut piksel yang tentunya memiliki nilai numerik. Citra histogram merupakan distribusi probabilitas berupa nilai kecerahan warna yang diukur dari nilai 0 untuk warna gelap hingga 255 untuk warna putih. Ekualisasi histogram atau *Histogram Equalization* adalah proses merubah nilai dalam sumbu horizontal dan vertical untuk mendapatkan bentuk keluaran yang berbeda dari aslinya, proses meratakan seluruh nilai keabuan citra ke dalam warna cerah. (Kim, M, 2008). Metode *Histogram Equalization* (HE) bisa meningkatkan kontras citra dengan menggunakan pendeteksian nilai piksel minimal dan maksimal dari histogram citra, menyesuaikan kontras citra dengan merata. (Hidayat J, 2019)

Penggunaan citra digital semakin meningkat karena kelebihan yang dimiliki antara lain kemudahan dalam mendapatkan gambar, memperbanyak gambar, pengolahan gambar dan lain-lain. Tetapi tidak semua citra digital memiliki tampilan visual yang memuaskan mata manusia. Ketidakpuasan itu dapat timbul karena adanya *noise*, kualitas pencahayaan pada citra digital yang terlalu gelap atau terlalu terang. Sehingga diperlukan metode untuk dapat memperbaiki kualitas citra digital tersebut. Untuk meminimalisir *noise (noise reduction)* yang disebabkan karena kekurangan kualitas pencahayaan dapat dilakukan dengan menggunakan metode *Low-Pass Filter* (LPF) yaitu proses penghalusan transisi tajam pada sebuah citra yang mengambil data pada frekuensi rendah dan melemahkan frekuensi tinggi. Untuk meningkatkan kualitas citra dari sisi kontras warna (*contrast enhancement*) maka kita bisa memberikan perlakuan pada histogramnya. Perlakuan yang dimaksud yaitu dengan metode *Histogram Equalization* (HE) pada citra dalam level keabuan (*grayscale*) dimana distribusi nilai derajat keabuan pada suatu citra dibuat rata. Hal ini memungkinkan area kontras yang rendah menjadi lebih tinggi, sehingga perbaikan kontras akan lebih efektif. Histogram citra dikatakan baik bila mampu melibatkan semua level atau aras yang mungkin pada level keabuan. Tentu saja tujuannya agar mampu menampilkan detil pada citra sehingga mudah untuk diamati. Seiring dengan perkembangannya teknologi, resolusi piksel pada citra digital semakin tinggi demi memanjakan mata manusia, sehingga semakin banyak piksel yang perlu diolah pada saat mengolah citra digital maka akan semakin banyak waktu yang diperlukan. Oleh karena itu, untuk mengatasi permasalahan tersebut maka teknik pemrograman paralel dengan menggunakan *platform* OpenMP akan diuraikan sehingga pengolahan citra digital dapat dilakukan lebih cepat.

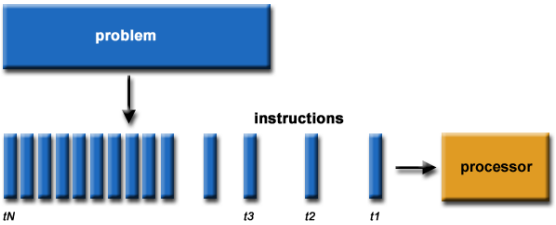
1.2. Tinjuan Pustaka

Pemrograman paralel adalah pemrograman untuk banyak komputer, atau komputer-komputer dengan banyak prosesor internal, untuk memecahkan sebuah masalah pada kecepatan komputasi yang lebih tinggi daripada yang bisa dilakukan dengan sebuah komputer tunggal (Wilkinson & allen, 2004). Pemrograman paralel juga menawarkan peluang untuk menangani masalah-masalah yang lebih besar, yaitu masalah dengan lebih banyak langkah komputasi atau yang membutuhkan memori lebih besar. Pemrosesan paralel merupakan teknik komputasi menggunakan dua atau lebih komputer untuk menyelesaikan suatu tugas dalam waktu yang simultan dengan cara mengoptimalkan resource pada sistem komputer yang ada untuk dapat mencapai tujuan yang sama (martins, ribeiro, & rodriguez, 2001).

Proses kerja dari paralel adalah dengan membagi beban kerja dan mendistribusikannnya pada komputer-komputer lain yang terdapat dalam sistem untuk menyelesaikan suatu masalah (ayuningtyas, 2016).

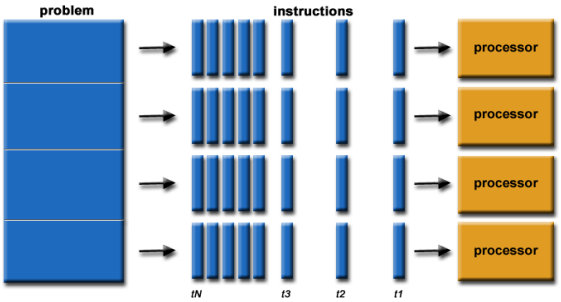
Pemrograman paralel adalah komputasi dua atau lebih task pada waktu bersamaan dengan tujuan untuk mempersingkat waktu penyelesaian tasks tersebut dengan cara mengoptimalkan resource pada sistem komputer yang ada. Pemrosesan paralel dapat mempersingkat waktu ekseskusi suatu program dengan cara membagi suatu program menjadi bagian-bagian yang lebih kecil yang dapat dikerjakan pada masing-masing prosesor secara bersamaan (hidayat, 2006). Suatu program yang yang dieksekusi oleh n prosesor diharapkan dapat mempersingkat waktu eksekusi n kali lebihcepat.

Kinerja prosesor pada dasarnya dilakukan dengan cara menghitung waktu sebelum dilakukan proses pekerjaan dimulai dan diakhiri saat proses pekerjaan selesai. Proses ini pada dasarnya membagi sejumlah array ke dalam sub array dimana setiap sub array dikerjakan oleh satu buah prosesor, secara berurutan dalam kurun waktu tertentu, seperti terlihat pada Gambar 2.1. Tujuan dari penggunaan prosesor parallel adalah untuk mengatasi kendala kecepatan dan kapasitas memori, dengan asumsi bahwa sumber daya parallel sudah tersedia. Seperti komputasi pada prosesor tunggal, kinerja komputasi parallel dipengaruhi oleh Teknik pemrogaman, arsitektur, atau keduanya (Arnanda, 2014).



Gambar 1. Penyelesaian sebuah masalah pada komputasi tunggal

Proses paralel adalah proses yang dilakukan pada p buah prosesor, sehingga data yang dipecah beberapa bagian, dimana setiap bagian data tersebut diserahkan ke prosesor masing-masing untuk diolah, seperti terlihat pada Gambar 2.2



Gambar 2. Penyelesaian sebuah masalah pada komputasi paralel

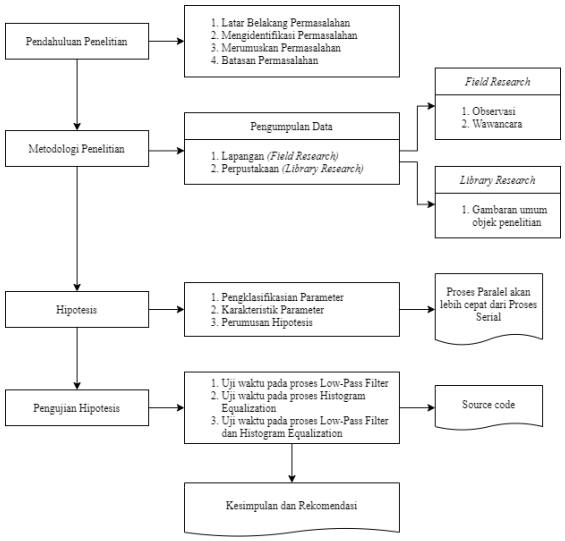
Dari kedua gambar di atas, dapat disimpulkan bahwa kinerja komputasi paralel lebih efektif dan dapat menghemat waktu untuk pemrosesan data yang banyak daripada komputasi tunggal. Banyak parameter yang dapat digunakan untuk mengukur kinerja sistem paralel, diantaranya adalah waktu komputasi dan speedup komputasi paralel (Arnanda, 2014).

Dalam suatu sistem paralel biasanya *programmer* perlu mengukur kinerja dengan membandingkan waktu proses algoritma paralel dengan waktu proses sekuensial. Pemrograman paralel bertujuan untuk meningkatkan performa komputasi. Semakin banyak hal yang bisa dilakukan secara bersamaan, semakin banyak pekerjaan yang bisa diselesaikan. Batas bawah *speed up* adalah 1 (satu) dan batas atasnya adalah jumlah prosesor yang digunakan (p). Performa dalam pemrograman paralel diukur dari berapa banyak peningkatan kecepatan ($SpeedUp$) yang diperoleh dalam menggunakan teknik paralel. Dengan mendefinisikan T_s dan T_p sebagai waktu proses pemrograman serial dan pemrograman paralel, maka *speed up* dapat dihitung dengan persamaan (Wilkinson & Allen, 2004):

$$SpeedUp = \frac{T_s}{T\mathrm{p}} \qquad \textbf{(2.1)}$$

Speed up pada satu prosesor adalah sama dengan 1 (satu) dan *speed up* pada (p) prosesor bernilai $1 \leq S_p \leq p$. Secara ideal *speed up* meningkat sebanding dengan bertambahnya jumlah prosesor. Jadi jika digunakan p prosesor, *speed up* idealnya adalah p (Wilkinson & Allen, 2004).

1.3. Metodologi Penelitian



Gambar 3. Kerangka Penelitian

Citra digital yang akan diproses dikonversikan terlebih dahulu kedalam matriks dua dimensi dengan menggunakan bantuan aplikasi Octave. Kemudian menggunakan program Bahasa C yang menerapkan teknik pemrograman parallel menggunakan platform OpenMP, semua data matriks citra dibaca dan dilakukan 2 (dua) tahap pemrosesan.

Pada proses ini, metode *Low-Pass Filter* pada pengolahan citra digital mampu menghaluskan transisi tajam pada citra sehingga dapat mengurangi *noise* bahkan menghapusnya. Data matriks citra diproses dengan metode *Low-Pass Filter* untuk mengurangi *noise* pada citra tersebut. Persamaan matematikanya dapat dinotasikan sebagai berikut (Phillips, 2000).

$$D_{i,j} = \frac{1}{k} W_{3\times3} \oplus C_{i,j} \; ; \quad i = 1,2,\dots,N \; ; \quad j = 1,2,\dots,M \; ; \quad k = 16$$

$$W_{3\times3} = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Dengan $C_{i,j}$ adalah matriks citra masukan, i dan j adalah indeks baris dan kolom pada matriks, N dan M adalah jumlah baris dan kolom pada matriks, $W_{3\times3}$ adalah *weighted matrix* dari *Low-Pass Filter*, k adalah faktor skalar dari $W_{3\times3}$ dan $D_{i,j}$ adalah matriks citra keluaran (hasil).

Pada proses ini, metode *Histogram Equalization* pada pengolahan citra digital mampu mendistribusikan nilai derajat keabuan pada suatu citra sehingga perbaikan kontras lebih efektif.

Kemudian dilakukan perbaikan kontras pada data matriks citra denganmenggunakan metode *Histogram Equalization*. Persamaan matematikanya dapat dinotasikan sebagai berikut (Phillips, 2000).

$$E_{i,j} = \frac{GrayLevel}{Area} Hc_k \; ; \quad k = D_{i,j} \; ; \quad i = 1,2,\dots,N \; ; \; j = 1,2,\dots,M$$

$$GrayLevel = 256 \; ; \quad Area = N \times M$$

Dengan $D_{i,j}$ adalah matriks citra masukan, i dan j adalah indeks baris dan kolom pada matriks, N dan M adalah jumlah baris dan kolom pada matriks, Hc_k adalah nilai kumulatif histogram, $GrayLevel$ adalah tingkat keabiuan (biasanya bernilai 256), $Area$ adalah luas matriks citra masukan dan $E_{i,j}$ adalah citra keluaran (hasil). Kemudian nilai kumulatif histogram didefinisikan sebagai berikut (Phillips, 2000).

$$Hc_k = \sum_{k=0}^{GrayLevel} H_k \; ; \quad GrayLevel = 256$$

$$H_k = H_k + 1 \; ; \quad k = D_{i,j} \; ; \quad i = 1,2,\dots,N \; ; \quad j = 1,2,\dots,M$$

Dengan Hc_k adalah nilai kumulatif histogram, dan H_k adalah nilai histogram dari matriks citra masukan $D_{i,j}$. Perangkat lunak yang digunakan untuk menuliskan kode program bahasa C adalah CodeBlocks sedangkan untuk mengkonversikan gambar serta menampilkan gambar adalah Octave yang dijalankan pada sistem operasi Windows 10 dengan spesifikasi prosesor Intel i5-2410M @ 2.30 GHz (4CPUs) dan memori 8192MB RAM.

2. PEMBAHASAN

Hasil dari pengolahan citra digital dengan metode Low-Pass Filter dan Histogram Equalization dapat dilihat pada gambar-gambar berikut ini. Dalam penelitian ini, digunakan 3 (tiga) sampel citra digital dengan dimensi berbeda yang dapat dilihat pada Tabel 4.1.

Tabel 1. Tabel dimensi sampel citra digital

Sampel	Dimensi (piksel)
Gambar 1	1152 x 864
Gambar 2	1957 x 1468
Gambar 3	645 x 483

Langkah pertama yang dilakukan dalam penelitian ini yaitu mengkonversikan terlebih dahulu sampel citra digital kedalam matriks dua dimensi dengan menggunakan bantuan aplikasi Octave. Citra digital dengan ekstensi jpg dapat dibaca dan dimasukan kedalam suatu variabel dengan cara:

```
input = imread('Gambar1.jpg');
```

Kemudian citra digital tersebut dipecah berdasarkan kanal warnanya yaitu *red*, *green* dan *blue* dengan cara:

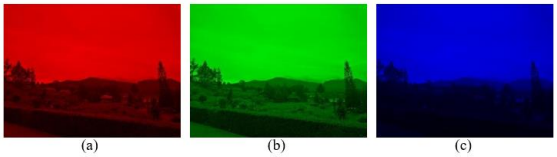
```
inputR = input(:, :, 1);
inputG = input(:, :, 2);
inputB = input(:, :, 3);
```

Lalu disimpan kedalam file berekstensi txt untuk

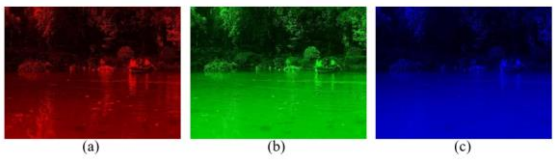
diproses di tahap selanjutnya menggunakan program bahasa C dengan cara :

```
dlmwrite('Gambar1-red.txt', inputR,
'delimiter', ' ');

dlmwrite('Gambar1-green.txt',
inputG, 'delimiter', ' ');

dlmwrite('Gambar1-blue.txt',
inputB, 'delimiter', ' ');
```

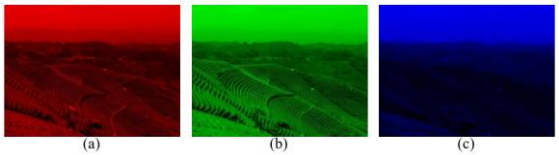
Dalam tahap ini, setelah citra digital dipecah berdasarkan kanal warnanya, kondisi sampel citra tersebut dapat dilihat pada Gambar …. Terlihat warna yang dihasilkan dari citra digital berdasarkan kanal warnanya.



Gambar 4. Sampel citra digital pertama setelah dilakukan pemecahan kanal warna



Gambar 5. Sampel citra digital kedua setelah dilakukan pemecahan kanal warna



Gambar 6. Sampel citra digital ketiga setelah dilakukan pemecahan kanal warna

pada tahap pemroresan menggunakan metode *Low-Pass Filter*, citra dikuragi noise-nya berdasarkan persamaan. Berikut ini notasi algoritma metode *Low-Pass Filter* :

```
#pragma omp parallel
for i=0 to N-1
for j=0 to M-1
if i=0 OR j=0 OR i=N-1 OR j=M-1 then
outputR(i,j) <- inputR(i,j)
outputG(i,j) <- inputG(i,j)
outputB(i,j) <- inputB(i,j)
    else
    outputR(i,j) <- (1/k) * (W(0,0) *
inputR(i-1,j-1)
    +W(0,1)  * inputR(i-1,j)
    +W(0,2)  * inputR(i-1,j+1)
    +W(1,0)  * inputR(i,j-1)
    +W(1,1)  * inputR(i,j)
    +W(1,2)  * inputR(i,j+1)
    +W(2,0)  * inputR(i+1,j-1)
    +W(2,1)  * inputR(i+1,j)
    +W(2,2)  * inputR(i+1,j+1))
outputG(i,j) <- (1/k) * (W(0,0) *
inputG(i-1,j-1)
     +W(0,1)  * inputG(i-1,j)
     +W(0,2)  * inputG(i-1,j+1)
     +W(1,0)  * inputG(i,j-1)
     +W(1,1)  * inputG(i,j)
     +W(1,2)  * inputG(i,j+1)
     +W(2,0)  * inputG(i+1,j-1)
     +W(2,1)  * inputG(i+1,j)
     +W(2,2)  * inputG(i+1,j+1))
outputB(i,j) <- (1/k) * (W(0,0) *
inputB(i-1,j-1)
     +W(0,1)  * inputB(i-1,j)
     +W(0,2)  * inputB(i-1,j+1)
     +W(1,0)  * inputB(i,j-1)
     +W(1,1)  * inputB(i,j)
     +W(1,2)  * inputB(i,j+1)
     +W(2,0)  * inputB(i+1,j-1)
     +W(2,1)  * inputB(i+1,j)
     +W(2,2)  * inputB(i+1,j+1))
     endif
   endfor
endfor
```

Kemudian tahap selanjutnya adalah perbaikan kontras pada matriks citra digital dengan mengunakan metode *Histogram Equalization* berdasarkan persamaan. Berikut notasi algoritmanya :

```
#pragma omp parallel
for i=0 to N-1
   for j=0 to M-1
  kR <- inputR(i,j)
  kG <- inputG(i,j)
  kB <- inputB(i,j)
  histogramR(kR) <- histogramR(kR) + 1
histogramG(kG) <- histogramG(kG) + 1
histogramB(kB) <- histogramB(kB) + 1
   endfor
endfor
#pragma omp parallel
for i=0 to GrayLevel-1
       sumR <- sumR + histogramR(i)
       sumG <- sumG + histogramG(i)
       sumB <- sumB + histogramB(i)
       kumulatifHistogramR(i) <- sumR
       kumulatifHistogramG(i) <- sumG
       kumulatifHistogramB(i) <- sumB
endfor
#pragma omp parallel
for i=0 to N-1
   for j=0 to M-1
       kR <- inputR(i,j)
       kG <- inputG(i,j)
       kB <- inputB(i,j)
       outputR(i,j)<-
(GrayLevel/Area)*kumulatifHistogramR(kR)
       outputG(i,j)<-
```

```
(GrayLevel/Area)*kumulatifHistogramG(kR)
       outputB(i,j)< -
(GrayLevel/Area)*kumulatifHistogramB(kR)
    endfor
endfor
```

Penelitian ini menuliskan program dalam dua versi yaitu serial dan parallel. Masing-masing program mempunyai masukan dan keluaran yang sama. Berdasarkan hasil penelitian, waktu pengolahan citra digital dapat dilihat pada Tabel :

Tabel 2. Perbandingan waktu pengolahan citra digital pada proses serial dan paralel

Sampel	Serial (Detik)			Parallel (detik)		
	LPF	HE	LPFHE	LPF	HE	LPFHE
Gambar 1	0.237	0.062	0.279	0.188	0.029	0.207
Gambar 2	0.568	0.218	0.692	0.556	0.087	0.826
Gambar 3	0.090	0.031	0.095	0.057	0.011	0.067

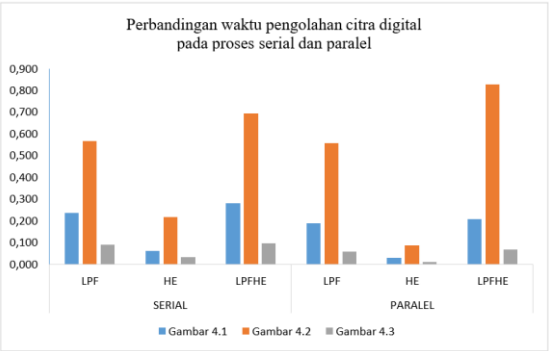
Keterangan :

LPF adalah proses *Low-Pass Filter*

HE adalah proses *Histogram Equalization*

LPFHE adalah proses *Low-Pass Filter* dan *Histogram Equalization*

Pada Tabel 4.2 dapat diketahui bahwa dimensi dari citra digital mempengaruhi waktu pemrosesan, baik dalam proses serial atau proses paralel. Secara berurutan,sampel yang memproses dengan waktu tercepat adalah sampel Gambar 4.3 dengan total waktu 0.216 detik pada proses serial dan 0.135 detik pada proses paralel, Gambar 4.1 dengan total waktu 0.578 detik pada proses serial dan 0.424 detik pada proses paralel, serta Gambar 4.2 dengan total waktu 1.478 detik pada proses serial dan 1.469 detik pada proses paralel. Perbandingan waktu pengolahan citra digital pada proses serial dan paralel dapat juga dilihat dalam bentuk grafik pada Gambar 4.13.



Gambar 7. Grafik perbandingan waktu pengolahan citra digital pada proses serial dan paralel

Kemudian peningkatan kecepatan proses serial terhadap proses parallel berdasarkan persamaan …. Dapat dilihat pada tabel ….

Tabel 3. Peningkatan kecepatan proses serial terhadap proses paralel

Sampel	*Speeed Up*		
	LPF	HE	LPFHE
Gambar 1	1.3 x	2.1 x	1.3 x
Gambar 2	1.0 x	2.5 x	0.8 x
Gambar 3	1.6 x	2.8 x	1.4 x
Rata-rata	1.3 x	2.5 x	1.2 x

Keterangan :

LPF adalah proses *Low-Pass Filter*

HE adalah proses *Histogram Equalization*

LPFHE adalah proses *Low-Pass Filter* dan *Histogram Equalization*

Tabel 4.3 menunjukan bahwa peningkatan kecepatan pada proses paralel lebih cepat dibandingkan proses serial. Rata-rata peningkatan kecepatan pada proses *Low- Pass Filter* sebesar 1.3x lebih cepat, proses *Histogram Equalization* sebesar 2.5x lebih cepat serta pada proses *Low-Pass Filter* dan *Histogram Equalization* sebesar 1.2x lebih cepat. Pada setiap sampel gambar, peningkatan kecepatan tercepat yaitu pada proses *Histogram Equalization* dengan peningkatan lebih besar dari 2x. Peningkatan kecepatan terkecil yaitu saat mengolah sampel Gambar 4.2 pada proses *Low-Pass Filter* dan *Histogram Equalization* (LPFHE) sebesar 0.8x yang bisa dikatakan bahwa proses serial lebih cepat dibandingkan dengan proses paralel. Kemudian peningkatan kecepatan terbesar yaitu saat mengolah sampel Gambar 4.3 pada proses *Histogram Equalization* dengan peningkatan sebesar 2.8x yang artinya proses paralel 2.8x lebih cepat dibandingkan dengan proses serial.

3. KESIMPULAN

Berdasarkan hasil penelitian, dapat disimpulkan bahwa :

1. Citra digital dapat memiliki tampilan visual yang memuaskan dengan dilakukan pengolahan terhadap citra digital diantaranya menerapkan metode *Low-Pass Filter* untuk mengurangi *noise* dan *Histogram Equalization* untuk perbaikan kontras.
2. Tingkat *noise* pada citra digital dapat dikurangi/diminimalisir dengan dilakukan penerapan metode *Low-Pass Filter* yang

mengolah citra digital dengan menghaluskan transisi tajam pada citra digital namun citra digital terlihat sedikit lebih buram (*blurry*).

3. Kualitas pencahayaan pada citra digital dapat diatur sehingga memiliki pencahayaan yang ideal dengan menerapkan metode *Histogram Equalization* yang mengolah citra digital dengan meningkatkan nilai kontras pada area kontras yang rendah menjadi lebih tinggi sehingga nilai derajat keabuan pada citra digital menjadi rata yang menunjukan bahwa pencahayaan menjadi lebih ideal.
4. Teknik pemrograman paralel menggunakan *platform* OpenMP yang diterapkan mampu mempercepat proses pengolahan citra digital dengan rata-rata sebesar 1.7x lebih cepat sehingga proses pengolahan citra digital yang dilakukan lebih menghemat waktu.

PUSTAKA

Ayuningtyas, A. (2016). Pemrosesan Paralel Pada Low Pass Filtering Menggunakan Transform Cosinus Di MPI (Message Passing Interface). *Seminar Nasional Teknologi Informasi dan Kedirgantaraan (SENATIK), II*, pp. 115-120. Yogyakarta.

Arnanda, H. (2014). *Implementasi Pemrograman Paralel Dalam Deteksi Tepi Menggunakan Metode Operator Sobel.* Universitas Islam Negeri Sultan Syarif Kasim Riau.

Hidayat J, Usman U, Faisal A, Syafriwel S. Perbandingan Metode Perbaikan Kualitas Citra Berbasis Histogram Equalization Pada Citra Satelit. Journal of Electrical Technology. 2019 ; 4 (3) : 111-115.

Hidayat, S. (2006). Pemrosesan Paralel Menggunakan Komputer Heterogen. *Seminar Nasional Aplikasi Teknologi Informasi 2006 (SNATI 2006)* (pp. 13-18). Yogyakarta: Universitas Islam Indonesia.

Martins, S. d., Ribeiro, C. C., & Rodriguez, N. (2001). Parallel ComputingEnvironments. *Handbook of Combinatorial Optimization*.

Phillips, D. (2000). *Image Processing in C Second Edition.* Kansas: R & DPublications.

Wilkinson, B., & Allen, M. (2004). *Parallel Programming.* Pearson India.