IMPLEMENTASI METODE ALPHA TRIMMED MEAN FILTER MENGGUNAKAN OPENCL UNTUK PENGHAPUSAN NOISE PADA CITRA BERWARNA

SKRIPSI

Oleh :

KURNIA RIZKY AMALIA

12650078

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM MALANG
2016

HALAMAN PENGAJUAN

IMPLEMENTASI METODE ALPHA TRIMMED MEAN FILTER MENGGUNAKAN OPENCL UNTUK PENGHAPUSAN NOISE PADA CITRA BERWARNA

SKRIPSI

Diajukan Kepada:
Fakultas Sains dan Teknologi
Universitas Islam Negeri
Maulana Malik Ibrahim Malang
Untuk Memenuhi Salah Satu Persyaratan Dalam
Memperoleh Gelar Sarjana Komputer (S.Kom)

Oleh :
KURNIA RIZKY AMALIA
NIM. 12650078

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM MALANG
2016

LEMBAR PERSETUJUAN

IMPLEMENTASI METODE ALPHA TRIMMED MEAN FILTER MENGGUNAKAN OPENCL UNTUK PENGHAPUSAN NOISE PADA CITRA BERWARNA

SKRIPSI

Oleh :

KURNIA RIZKY AMALIA

NIM. 12650078

Telah Diperiksa dan Disetujui untuk Diuji :

Tanggal : September 2016

Pembimbing I	Pembimbing II
Fatchurrochman, M.Kom	Dr. Cahyo Crysdian
NIP. 19700731 200501 1 002	NIP. 19740424 200901 1 008

Mengetahui,

Ketua Jurusan Teknik Informatika

Fakultas Sains dan Teknologi

Universitas Islam Negeri Maulana Malik Ibrahim Malang

Dr. Cahyo Crysdian

NIP. 19740424 200901 1 008

HALAMAN PENGESAHAN

IMPLEMENTASI METODE ALPHA TRIMMED MEAN FILTER MENGGUNAKAN OPENCL UNTUK PENGHAPUSAN NOISE PADA CITRA BERWARNA

SKRIPSI

Oleh :

Kurnia Rizky Amalia

NIM : 12650078

Telah Dipertahankan Di Depan Dewan Penguji Skripsi

Dan Dinyatakan Diterima Sebagai Salah Satu Persyaratan

Untuk Memperoleh Gelar Sarjana Komputer (S.Kom)

Tanggal : September 2016

Susunan Dewan Penguji:			**Tanda Tangan**
1. **Penguji Utama**	:	**A'la Syauqi, M.Kom**	( )
		NIP. 19771201 200801 1 007	
2. **Ketua Penguji**	:	**Dr. M. Amin Hariyadi**	( )
		NIP. 19670118 200501 1 001	
3. **Sekretaris**	:	**Fatchurrochman, M.Kom**	( )
		NIP. 19700731 200501 1 002	
4. **Anggota**	:	**Dr. Cahyo Crysdian**	( )
		NIP. 19740424 200901 1 008	

Mengetahui,

Kepala Jurusan Teknik Informatika

Dr. Cahyo Crysdian

NIP. 19740424 200901 1 008

HALAMAN PERNYATAAN
ORISINALITAS PENELITIAN

Nama : Kurnia Rizky Amalia

NIM : 12650078

Jurusan : Teknik Informatika

Fakultas : Sains dan Teknologi

Judul Skripsi : **IMPLEMENTASI ALPHA TRIMMED MEAN FILTER MENGGUNAKAN OPENCL UNTUK PENGHAPUSAN NOISE PADA CITRA BERWARNA**

Menyatakan dengan sebenarnya bahwa skripsi yang saya tulis ini benar-benar merupakan hasil karya saya sendiri, bukan merupakan pengambil alihan data, tulisan, atau pikiran orang lain yang saya akui sebagai hasil tulisan atau pikiran saya sendiri, kecuali dengan mencantumkan sumber cuplikan pada daftar pustaka. Apabila dikemudian hari terbukti atau dapat dibuktikan skripsi ini hasil jiplakan, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Malang, 14 September 2016

Yang membuat pernyataan,

Kurnia Rizky Amalia

NIM : 12650078

KATAPENGANTAR

Assalamu`alaikum wr wb

Segala puji bagi Allah SWT yang maha pengasih dan penyayang, bahwa atas taufiq dan hidayah-Nya maka penulis dapat menyelesaikan penyusunan skripsi ini.

Shalawat dan salam semoga tetap tercurahkan kepada junjungan kita Nabi Muhammad saw kekasih Allah sang pemberi syafa'at beserta seluruh keluarga, sahabat dan para pengikutnya.

Skripsi yang berjudul "**Implementasi metode alpha trimmed mean menggunakan OpenCL untuk penghapusan noise pada citra berwarna",** ini disusun untuk memenuhi salah satu syarat guna memperoleh gelar Sarjana Strata Satu (S.1) Fakultas Sain dan Teknologi Universitas Maulana Malik Ibrahim Malang. Dalam penyusunan skripsi ini penulis menyadari bahwa penulisan ini tidak mungkin terlaksana tanpa adanya bantuan baik moral maupun spiritual dari berbagai pihak. Maka patut rasanya penulis menyampaikan terimakasih yang sedalamnya terutama kepada :.

1. Fatchurrochman ,M.Kom, selaku dosen pembimbing I dan Zainal Abidin M,Kom selaku dosen pembimbing penelitian yang telahbersedia meluangkan waktu, tenaga dan pikiran untuk memberikan bimbingan , pengarahan dalam penyusunan skripsi ini dan secara tidak langsung selalu menjadi sosok panutan penulis .
2. Dr. Cahyo Cyrsdian selaku Pembimbing II dan ketua jurusan Teknik Informatika , telah membimbing serta memberikan motivasi dan nasihat untuk semangat dalam menuntut ilmu dan berkarya .

3. Anwar ,Ummatin Choiriyah, Anita Fitriana dan Rachmad Wahyu Hidayat,selaku orang yang tidak pernah pamrih dalam memberi dukungan baik secara moril dan spiritual sehingga penulis sampai pada tahap saat ini ,terimaksih keluargaku.
4. Zuliatul Afifa dan Sakinah Amirah NR ,sebagai teman sahabat bahkan saudara yang saling mendukung dalam segala keadaan dan berjuang bersama dalam memperjuangkan gelar sarjana computer.
5. M.Ali Makrus S.Kep,Ners,atas segala kesabaran dan kegigihanmu dalam menemani perjalananku meraih gelar sarjana.
6. .Angkatan 2012 Teknik Informatika dan Sahabat-sahabatku (Nur Mutammima S.Si) ,sebagai penyemangat dan mood boaster penulis dalam menyelesaikan skripsi ini.

Harapan dan do'a penulis semoga semua amal kebaikan dan jasa-jasa dari semua pihak yang telah membantu hingga terselesaikannya skripsi ini diterima Allah SWT. serta mendapatkan balasan yang lebih baik dan berlipat ganda.

Penulis juga menyadari bahwa skripsi ini masih jauh dari kesempurnaan yang disebabkan keterbatasan kemampuan penulis. Oleh karena itu penulis mengharap saran dan kritik konstruktif dari pembaca demi sempurnanya skripsi ini. Akhirnya penulis berharap semoga skripsi ini dapat memberikan manfaat nyata bagi penulis khususnya dan para pembaca umumnya.

Malang, September 2016

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LISTING KODE

ABSTRAK

Kurnia Rizky Amalia.2016. Implementasi Alpha Trimmed Mean Menggunakan OpenCL Untuk Penghapusan Noise Pada Citra Berwarna**.** Jurusan Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang.
Pembimbing : Fatchurrochman, M.Kom, Dr. Cahyo Crysdian.

Kata Kunci: Komputasi Pararel , OpenCL, Penghapusan Noise, *Alpha Trimmed Mean*

Beberapa tahun lalu Kinerja tinggi dari komputasi pararel adalah suatu hal istimewa yang hanya diperuntukkan bagi perangkat khusus dengan harga mahal. Namun pada saat ini hal tersebut bisa dilakukan pada berbagai macam processor menggulalui OpenCL (Open Computing Language), kerangka kerja untuk pemrograman paralel dengan sistem heterogen yang terdiri dari prosesor dari CPU dan GPU.OpenCL dapat memberikan efisiensi dalam menyelesaikan berbagai masalah komputasi .Masalah yang di implementasikan pada komputasi pararel disini adalah tentang penghapusan noise pada citra. menggunakan. metode *Alpha Trimmed Mean*, metode menghitung nilai rata-rata dengan menghilangkan persentase tertentu dari nilai terbesar dan terkecil sebelum menghitung nilai mean. Adanya beberapa tahapan proses dalam menerapkan metode *Alpha Trimmed Mean*akan membutuhkan waktu yang cukup banyak jika dikerjakan secara skuensial. Maka perlu adanya penerapan komputasi pararel sebagai solusi. Pada ujicoba penelitian ini dilakukan pada gambar yang telah ditambahkan *Salt And Pepper Noise*dengan variasi kepadatan noise sebesar 30% untuk rendah, 50% untuk sedang, dan 80% untuk tinggi.

Hasil uji coba membuktikan bahwa implementasi alpha trimmed mean pada komputasi pararel memberikan nilai efisiensi yang lebih baik dibandingkan dengan dilakukan pada komputasi serial.Nilai terbaik dari rata-rata waktu yang dihasilkan dari pemrosesan pada CPU dan GPU adalah*62088.0912 ms* dan *284.4992 ms* pada gambar dengan *noise salt and pepper* kepadatannoisesebesar *30%*.

ABSTRACT

Kurnia Rizky Amalia.2016.*Implementation Alpha Trimmed Mean Using OpenCL For Noise Removal In Color Image. Department of Informatics Faculty of Science and Technology State Islamic University of Maulana Malik Ibrahim Malang.*
Supervisors: Fatchurrochman, M.Kom, Dr. Cahyo Crysdian.

Keywords: *Parallel Computing, OpenCL, Noise Removal, Alpha Trimmed Mean*

High performance parallel computing was something exclusive for expensive specialized hardware some years ago. but now, it can be done on wide range processor using OpenCL (Open Computing Language), which is a framework for parallel programming of heterogeneous systems consisting of CPU`s and GPU`s processors.Problem which are implemented on parallel computing here was about noise removal in image using Alpha Trimmed Mean method. It was the method of calculating the average value by eliminating certain percentage of the largest and smallest values before calculating the mean value. The existence of several stages of Alpha trim mean Implementationrequird a lot of time quite if was done using skuential. So that parallel computing become the solution. In this research trial researcher did in image which added by salt and pepper noise in variance noise density 30 % for low, 50% for medium , and 80 % for high.

The trial results proved that the implementation of the alpha trimmed mean on parallel computing using OpenCL gave better efficiency than the computation performed on serial. The best average value of the difference between the GPU and CPU time consuming is 62088.0912 ms and 284.4992 db in case image with salt and pepper noise density 30%.

الملخص

كورنيا رزقي أماليا، المشذبة تنفيذ ألفا يعني استخداماوفن-ج ل لإزالة الضوضاء في اللون الصور
المشرف الأول: فتح الرحمان، .الماجتير .كوم.
المشرف الثاني: د.جهيو جرسديان
كلمات البحث: الموازي الحاسبات، اوفن-جل(فتح حساب اللغة)، إزالة الضوضاء، ألفا المشذبة متوسط

وقبل بضع سنوات من الأداء العالي الحوسبة المتوازية هو حالة خاصة والتي هي مخصصة للأجهزة خاصة مع ارتفاع الأسعار .ولكن في هذه اللحظة يمكن أن يتم ذلك على مجموعة واسعة من المعالج باستخدام ، اوفن-ج ل (فتح حساب اللغة)، إطارا للبرمجة المتوازية من الأنظمة غير المتجانسة تتكون من المعالجات وحدة المعالجة المركزية والجرافيك .، اوفن-ج ل يمكن أن توفر الكفاءة في حل المشكلات الحسابية تنفذتماما على الحوسبة المتوازية هنا هو للقضاء على الضوضاء في الصورة .باستخدام ألفا المشذبة المتوسط، وطريقة حساب متوسط قيمة من خلال القضاء على نسبة معينة من أكبر وأصغر القيم قبل احتساب القيمة المتوسطة .كما أن وجود عدة مراحل في عملية تطبيق ألفا المشذبة طريقة المتوسط تتطلب الكثير جدا في حال القيام به بشكل متسلسل .ومن هنا جاءت الحاجة لتطبيق الحوسبة المتوازية كحل .في محاكمة أجريت الدراسة على الصورة التي تم إضافتها الملح والفلفل الضوضاء مع اختلاف كثافة الضوضاء من ٣٠٪ للانخفاض، و ٥٠٪ متوسطة، و٨٠٪ إلى الأعلى.

تثبت نتائج التجارب أن تنفيذ ألفا قلص يعني على الحوسبة المتوازية يعطي كفاءة أفضل من حساب أجريت على أفضل قيمة متوسط الوقت الناجمة عن تجهيز وحدة المعالجة المركزية والجرافيك هي ٦٢٠٨٨.٠٩١٢ .المسلسل ٢٨٤.٤٩٩٢مللي مللي ثانية وعلى الصورة مع الملح والفلفل ضجيج كثافة الضوضاء من ٣٠٪.

BAB I

PENDAHULUAN

1.1. Latar Belakang

Kemajuan dalam perkembangan *hardware* dalam tahun terakhir memungkinkan pengembangan aplikasi yang sangat kompleks dengan sumber daya komputasi yang tinggi. Selain itu, orang membutuhkan hasil secepat mungkin. Perkembangan Komputasi pararel menjadi hal yang gencar dikembangkan saat ini (*Marcelo Arroyo*,2013). Komputasi paralel merupakan teknik melakukan komputasi sebuah program secara serentak. Model komputasi paralel terutama dari dua jenis. Pertama adalah memori bersama dan yang kedua komputasi terdistribusi. Untuk menjalankan proses komputasi sebuah prosesor memiliki batasan maximum. Jika sebuah prosesor tetap menjalankan proses yang komplek dengan data yang besar maka akan memakan waktu yang sangat lama. Selain itu, banyak masalah aplikasi yang membutuhkan lebih banyak proses komputasi dan sumber daya. Waktu yang dibutuhkan untuk menyelesaikan kasus tersebut pada prosesor tunggal dimukinkan usang untuk sebuah komputasi *real time.*

وآتِ ذَا الْقُرْبَى حَقَّهُ وَالْمِسْكِينَ وَابْنَ السَّبِيلِ وَلا تُبَذِّرْ تَبْذِيرًا(27) إِنَّ الْمُبَذِّرِينَ كَانُوا إِخْوَانَ الشَّيَاطِينِ وَكَانَ الشَّيْطَانُ لِرَبِّهِ كَفُورًا

"*Dan berikanlah kepada keluarga-keluarga yang dekat akan haknya, kepada orang miskin dan orang yang dalam perjalanan dan janganlah kamu enghambur -hamburkan (hartamu) secara boros.(27)Sesungguhnya pemboros-pemboros itu adalah saudara-saudara syaitan dan syaitan itu adalah sangat ingkar kepada Tuhannya."*(QS.al-Baqoroh/2:27-28).

Dari kedua ayat tersebut dijelaskan mengenai harta benda yang dilarang untuk keluarkan secara sia-sia dan tidak pada jalan alloh. Harta benda tidak selalu berupa materi atau kekayaan namun jiga bias berupa kesempatan dan waktu. Dalam bidang komputasi hal yang paling di kejar adalah efisiensi.maka usaha terus dikembangkan untuk mencapai hal tersebut.dengan komputasi serial suatu masalah terpeecahkan dalam waktu yang relative lama sesuai dengan kompleksitas masalah yang dikerjakan.

Solusi yang jelas dari masalah yang telah diuraikan adalah komputasi paralel yang menghubungkan lebih banyak jumlah prosesor (*Zaman Rafiqul & Firoj Ali*,2012). Produsen *software* dan pt erangkat keras mulai menciptakan *software* untuk mendukung teknologi tersebut.Platform pemrograman paralel yang paling umum digunakaan adalah CUDA(*Compute Unified Device Architecture*) yang dikembangka oleh Nvidia. CUDA hadir pertama kali sebagai platform yang tidak hanya menangani tentang grafika komputer tetapi juga untuk segala jenis apikasi. Pada tahun 2008 OpenCL(*Open Computing Language*) dihadirkan oleh Khronos™ yang kemudian di adopsi untuk dipasangkan pada produk perusahaan AMD . Dengan berjalannya waktu dikembangkan device untuk pemrograman pararel yang saling berintegrasi tidak hanya berjalan pada GPU, namun juga CPU. AMD Radeon™ menghadirkan CPU dengan GPU terintergrasi dalam satu paket yang sempurna.Teknologi baru ini memberikan kinerja dan efisiensi dengan HSA (*Heterogeneous System Architecture*) yang memungkinkan CPU dan GPU untuk bekerja bersama pada aplikasi yang didukung dengan membagi dan mengarahkan tugas secara cepat dan otomatis ke *core* yang tepat. Dengan AMD dapat dilakukan

komputasi paralel menggunakan bahasa pemrograman OpenCL. OpenCL merupakan *framework* untuk memenuhi tujuan dari semua pemrograman paralel dan *portable* dengan GPU maupun CPU(*Kazuya Matsumoto et al*,2012). OpenCL dirancang untuk memenuhi hal tersebut. Dikembangkan oleh Khronos,OpenCL merupakan gabungan konsep CUDA, CAL,CTM, dan mendukung berbagai tingkat paralelisme dan efisien untuk homogen atau heterogen(*Dr. Avi endelson*,2013).OpenCL merupakan teknologi yang akan berjalan pada hampir semua platform.Usaha dalam peningkatan kecepatan ini dapat diterapkan untuk berbagai aplikasi seperti audio,pengenalan suara,video,database gambar,pengolahan citra ,dan masih banyak lainnya *(J. A. Fraire et al*,2013). Luasnya lingkup penerapan OpenCL peneliti akan coba terapkan pada citra untuk penghapusan noise.

Citra merupakan kombinasi antara titik, garis, bidang, dan warna untuk menciptakan suatu imitasi dari suatu obyek. Citra bisa berwujud gambar dua dimensi, seperti lukisan, foto, dan berwujud tiga dimensi, seperti patung. seiring dengan perkembangan peradaban manusia dan teknologi . Media citra juga tidak terbatas pada canvas ataupun kertas,namun sudah dalam grafika komputer. Citra terbagi menjadi 2 yaitu ada citra yang bersifat analog dan ada citra yang bersifat digital. Citra analog adalah citra yang bersifat kontinu seperti gambar pada monitor televisi, foto sinar X, hasil CT Scan. Sedangkan pada citra digital adalah citra yang dapat diolah oleh komputer(*T.Sutoyo et al* ,2009). Sebuah citra digital dapat diwakili oleh sebuah matriks yang terdiri dari M kolom N baris, dimana perpotongan antara kolom dan baris disebut piksel (*picture element*), yaitu elemen

terkecil dari sebuah citra. Citra digital dapat mengalami penurunan kualitas atau gangguan yang terjadi, seperti lensa tidak fokus, muncul bintik-bintik yang disebabkan oleh proses capture yang tidak sempurna, pencahayaan yang tidak merata yang mengakibatkan intensitas tidak seragam, kontras citra terlalu rendah sehingga objek sulit dipisahkan dari latar belakangnya, atau gangguan yang disebabkan oleh kotoran-kotoran yang menempel pada citra dan lain sebagainya. gangguan pada citra yang paling umum terjadi berupa variasi intensitas suatu pixel yang tidak berkolerasi dengan pixel tetanggannya, setiap gangguan tersebut dinamakan noise dansetiap noise harus dihilangkan untuk menghasilkan gambar yang lebih baik saat diterima mata.

Jenis Noise yang paling sering merusak citra adalah *Salt and pepper noise* atau *noise impluse* dan *gaussian noise*(*Lipeng Wang*,2011).Dari dua jenis noise tersebut yang paling umum muncul adalah *salt and pepper noise*,yang mana pixel yang mengalami gangguan akan memiliki intensitas tinggi yang bernilai 255 atau intensitas rendah yang bernilai 0(*Sanchez G et al*,2014). Pada nilai 0 diibaratkan sebagai *pepper*(lada) yang ditunjukkan dengan titik hitam dan nilai 255 sebagai *salt*(garam) yang ditunjukkan dengan titik putih, hal tersebut yang mendasari nama *salt and pepper noise*. Citra yang mengandung noise memerlukan langkah-langkah perbaikan untuk meningkatkan kualitas citra. Tujuan utama dari peningkatan kualitas citra adalah untuk memproses citra, sehingga citra yang dihasilkan lebih baik dari kualitas sebelumnya.Banyak algoritma dikembangkan untuk mengatasi noise tersebut sampai sekarang.

Teknik filter Linear sampai saat ini masih dikembangkan untuk penghilangan noise, dengan model matematis yang sederhana teknik ini sering di terapkan.Kekurangan untuk teknik linear kurang bisa mempertahankan garis tepi dari suatu citra, sehingga teknik non-linear menjadi solusi untuk masalah ini . Salah satu pengembangan dari teknik non-linear yaitu algorithma *alpha trimmed mean* dengan parameter *alfa* yang nantinya menjadi tolak ukur dilakukannya pembuangan nilai dari masing-masing poin tertingga dan terendah setelah proses pengurutan. Dari algorithma ini output dan waktu yang dihailkan lebih baik dibandingksn dengan algorithma *Taguchi's filter* dan *the Jaeckel's filter*(*Remzi Öten et al* ,2004). Diperkenalkan juga algorithma baru untuk meningkatkan efsiensi kinerja *mean filter* sehingga mampu menangani gambar dan kernel dengan ukuran besar pada proses penghilangannoise. Algorithma ini dikenalkan dengan nama *highly efficient mean filter algorithm*(EFMF) . hasil dari penelitian ini jauh lebih efisien dibandingkan dengan *Alpha Trimmed Mean Filter(*AMF) dan *fast mean filter* (FMF) (*Fu Bin et al* ,2011).Selain yang sudah dijelaskan. Dikenalkan pula algorithma pegembangan dari mean filter umtuk menangani *salt and pepper noise* berintensitas tinggi. Algorithma ini dikenal dengan *First Order Neighborhood Mean Filter* (FONMF).Prinsip kerja FONM melakukan pengecekan kondisi setiap pixel untuk mendapatkan pixel yang bernoise untuk selanjutnya dilakukan proses penghilangannoise dengan nilai baru dari *mean filter* (*Priyanka Shrivastaval et al* ,2014).

Metode yang diusulkan pada penelitian ini adalah *Alpha trimmed mean* yang mana merupakan filer jenis non-linear, dengan mempertahankan tepi dan

mengurangi kepadatan noise. *Alpha trimmed mean* adalah metode menghitung nilai rata-rata yang menghilangkan persentase tertentu dari nilai terbesar dan terkecil sebelum menghitung mean. Adanya beberapa tahapan proses pencarian nilai *alpha trimmed mean* dan *preprocessing* gambar masukan akan membutuhkan waktu yang banyak terutama untuk gambar masukan dengan ukuran besar atau memiliki kernel konvolusi besar (*Fu Bin et al* ,2011). Perubahan besar telah dilakukan pada dekade terakhir dalam pengembangan *hardware* yang terkait dengan komputasi Pararel. Penggunaan komputasi paralel akan meninggkatkan kinerja komputer dari segi efisiensi waktu. Dari besarnya ukuran citra atau besarnya kernel konvolusi akan di bagi menjadi banyak tugas untuk dilakukan komputasi dalam waktu yang sama.

Pada skripsi ini diterapan pemrograman pararel pada metode *alphaTrimmed mean* untuk menghilangkan noise jenis *salt* dan *papper* sehingga diharapkan memperoleh hasil yang memuaskan pada efisiensi waktu pemrosesan. Skripsi ini diharapkan mampu berkontribusi untuk menghadirkan teknik penghilangan noise dengan efisiensi waktu yang tinggi.

1.2. Identifikasi Masalah

Identifikasi masalah yang muncul dari latar belakang yang telah diuraikan adalah sebagai berikut :

1. Bagaimana menerapkan komputasi paralel pada metode Alpha trimmed mean untuk menghilangkan noise pada citra digital ?
2. Seberapa efisiens waktu yang di peroleh dalam penerapan metode Alpha trimmed mean menggunakan komputasi pararel

1.3. Tujuan Penelitian

Tujuan dari penelitian ini adalah sebagai berikut.

1. Untuk membuktikan penerapan komputasi paralel pada metode Alpha trimmed mean untuk menghilangkan noise pada citra digital.
2. Untuk mengetahui efisiensi waktu yang dibutuhkan dalam penerapan metode Alpha trimmed mean menggunakan komputasi paralel.

1.4. Batasan Masalah

Untuk memperoleh hasil penelitian yang terarah dan sesuai dengan pokok permasalahan, peneliti memberi batasan sebagai berikut:

1. Bahasa pemrograman yang digunakan adalah OpenCL
2. Spesifikasi laptop yang digunakan

Platform : Laptop Lenovo G485

Memori : 2 GB

Harddisk : 250 GB HDD

Processor : AMD E-300 APU with Radeon(tm) HD Graphics 1.30 Ghz

3. Jenis noise yang digunakan adalah salt and pepper .
4. Smatphone yang digunakan huawei honor 4c tipe CHM-U01

1.5. Manfaat

Penelitian ini diharapkan dapat memberikan kontribusi dalam penelitian dibidang komputasi pararel. hasil yang diperoleh dari implementasi metode alpha trimmed mean pada komputasi pararel diharapkan mampu memberi gambaran

tentang efisiensi dan akurasi dalam penggunaannya.selain itu diharapkan memberikan sedikit wawasan mengenai komputasi pararel kepada pembaca.

BAB II

STUDI LITERATUR

2.1. Penelitian Terkait

Dalam sebuah jurnal karya *Remzy oten* menjelaskan dalam penelitiannya yang berjudul "*Adaptive Alpha-Trimmed Mean Filter Deviation From Assumed Noise Model*", bahwa teknik penghapusan noise menggunakan model non linear lebih efisien digunakan. Dalam teknik non linear bebrapa metode penghapusan noise memiliki kelbihan dan kekurangannya masing-masing,namun diantaranya yang paling umum di terapkan adalah median filter dimana teknik ini mampu menjaga tepi dari sebuah gambar. Selain median filter terdapat filter alpha trimmed mean yang merupakan pengembangan dari median filter dan mean filter. Teknik ini melakukan penghapusan elemen dari terkecil dan terendah setelah dilakukan pengurutan. Banyak elemen yang akan dihapus tergantung dari nilai alpha. Jika nilai alpha mendekati setengah dari jumlah elemen teknil ini menyerupai median filter.jika nilai alpha menedekati nol ,teknik ini menyerupai mean filter. Dalam penelitian ini dilakukan pengembangan-pengambangan dari model konvensional alpha trimmed mean dengan memaksimalkan kinerja dalam penghapusan noise dengan merubah nilai alpha sesuai dengan kriteria gambar yang akan diproses . Pada hasil uji coba diperoleh kinerja alpha trimmed mean dengan nilai alpha yang tepat bekerja lebih baik dalam penghapusan noise dibandingkan dengan median filter untuk tetap mempertahankan garis tepi pada objek gambar. Selain itu pada jenis-jenis noise yang diterapkan pada penelitian ini

membutuhkan teknik yang bisa menyesuaikan melihat kondisi gambar. Dalam uji coba pengukuran kinerja dan pengujian terhadap macam-macam noise dari model gausian , diperoleh hasil bahwa teknik *alpha trimmed mean* masih kurang bagus dalam beberapa kasus model *mixed noise*. Untuk meningkatkan kinerja alpha trimmed mean dibutuhkan estimator MAD (*median of absolute deviations from median*),namun rumus matematis dan kompleksitas proses komputasi menjadi semakin kompleks. Kinerja alpha *alpha trimmed mean* yang diusulkan ini tergantung pada nialai *alpha* yang di berikan .Nilai *alpha* tersebut tergantung dari parameter-parameter yang ada pada gambar-masukan yang bernoise. Kinerja *alpha trimmed mean* semakin bagus jika kemungkinan range dari alpha semakin kecil.

Oleh *faruk ahmed* dan *swagatam das* pada jurnalnya yang berjudul *"removal of high-density salt-and-pepper noise in images with an iterative adaptive fuzzy filter using alpha-trimmed mean"* menjelaskan bahwa seringnya sebuah citra mengalami penurunan kualitas disebabkan oleh gangguan noise.Noise yang sering muncul jenis impuls noise yaitu salt and pepper noisedanGaussian noise. Penelitian terus dilakukan untuk mencapai hasil yang memuaskan dalam penghapusan noise . Teknik yang telah ada sebelumnya yaitu teknik linear.yang paling sering diterapkan sebelumnya adalah mean filter,dimana mencari nilai rata-rata dari sekian elemen yang dijumlahkan. Perkembangan selanjutnya yaitu teknik non-linear filter dimana memiliki kinerja yang lebih baik daripada teknik linear filter .Kelebihan teknik non-linear filter mampu mempertahankan garis tepi objek pada gambar yang sedang diproses.Teknik Median filter adalah teknik non-linear

yang paling umum diterapkan namun kekurangannya median filter ini belum mampu untuk menangan gambar dengan noise berintensitas tinggi .maka dilakukan usaha untuk mengatasi hal tersebut dengan meningkatkan proses dari one-stageed menjadi two-staged. Dikembangkanlah teknik *fuzzy* dengan 2 tahapan proses dimana aturan-aturan dibuat dalam perintah-perintah else yang disediakan. Teknik filter *switching median* ini yang kemudian diusulkan untuk diterapkan dengan model *fuzzy*. *Fuzzy filter* pada dasarnya sederhana dan cukup efisien .maka ada ketertarikan dalam penggunaanya untuk dilakukakn penghapusan noise yang berintensitas tinggi dengan menerapkan teknik *fuzzy filter*tersebut. Metode yang diusulkan peneliti memiliki 2 tahap *iterasi adaptive fuzzy* untuk penghapusan impuls noise.Untuk memahami teknik yang diusulkan dijelaskan diawal tentang alpha trimmed mean dimana teknik ini lebih baik secara kinerja dibandingkan dengan mean dan median filter.*Alpha trimmed mea*n menghitung nilai rata-rata dari sekelompok elemen setelah dilakukan pemangkasan dari nilai tertinggi dan terendah sebesar α/2 elemen. Pada penerapannya algoritma yang diusulkan akan melakukan pengecekan terhadap piksel yang korup ,jika didapati piksel korup maka dilakukan proses penghapusan noise. Dalam *adaptive iterative fuzzy* ada dua parameter yang penting yaitu K1 dan K2 untuk menunjukkan anggota dari aturan fuzzy yang diterapakn. Dalam percobaan untuk level noise yang diterapkan 10%,50% dan 90%. dari hasil percobaan diperoleh kesimpulam bahwa jika untuk noise berintensitas tinggi nilai k1 =h dan untuk noise berintensitas rendah k1=1. Dari seluruh pengujian nilai k2 tidak memberi banyak pengaruh.jadi untuk penggunaan secara umum atau terhadap noise berintensitas

tinggi ,sedang,maupun rendah nilai k1 dan k2 lebih baik di isi nilai 3. Output yang dihasilkan cukup memuaskan untuk gambar dengan noise berintensitas tinggi.

Pada jurnal *Hai Chi, Fuchun Sun, Qing Li* dan *Xiaoxiao Wu* yang berjudul " *A New Adaptive Alpha-Trimmed Mean Filtering Approach**" menjelaskan bahwa pada awal munculnya bidang pengolahan citra ,teknik linear filter menjadi solusi utama dalam penghapusan noise dengan model matematis yang sederhana. Dalam teknik linear filter, yang utama adalah mean filter dimana cocok digunakan pada jenis noise *Gaussian* atau *uniform*, namun semakin lama perkembangan teknologi juga menuntut teknik lain yang mampu menangani noise jenis non-linear atau non-gaussian , sehingga dikembangkan teknik non-linear *Median filter* menjadi teknik filter yang paling umum digunakan karena model matematisnya cukup sederhana. *Median filter* bekerja baik pada jenis noise non-gaussian ,namun untuk noise gabungan dari *Gaussian* dan non-gaussian diusulkan teknik filter alpha trimmed mean yangmana merupakan gabungan dari teknik filter mean dan *median* .*Adaptive alpha trimmed mean* adalah pengembangan dari *alpha trimmed mean* yang bekerja lebih baik karena pemrosesan pengahpusan noise dilakukan sesuai hasil pengecekan kondisi gambar dengan memliki banyak kondisi-kondisi yang berguna pada keadaan gambar-gambar tertentu sesuai keinginan programmer.dalam penelitian ini menerapkan *adaptive alpha trimmed mean* dengan rumus dimana rumus ini disebut *asymptotic variance*. Metode ini sebelumnya dikenalkan oleh remzi dengan penelitiannya berjudul *adaptive alpha-trimmed mean filter based on asymptotic variance minimization*. Untuk setiap n nilai data sampel diamati dalam a window yang di lambangkan dengan W. pada

penelitian ini membandingkan Filter I dan Filter II. Dimana filter 1 tentang *adaptive alpha trimmed mean* dan yang ke dua tentang *alpha adaptive alpha trimmed mean based on asymptotic variance minimization* . Dilakukan empat percobaan, tiga pertama membandingkan mereka kinerja dan kompleksitas waktu, dan yang terakhir membandingkan ketahanan mereka. Parameter A dari dua filter diambil sebagai 0,0006. Kedua filter digunakan 3 x 3 ukuran window. Ukuran gambar diproses adalah 448 × 464. Di mana A adalah konstanta yang nilainya ditentukan oleh toleransi terhadap penyimpangan dari distribusi simetris. Peneliti mengharapkan dari research ini menemukan nilai alpha terbaik yang akan menghapus impulsive noise dengan pemangkasan beberapa sampel marjinal saat penghapusan *Gaussian noise*. Dari ujicoba kedua filter tersebut , dengan menganalisis karaktistik dari simetris distribusi model matematika dari noise, diusulkan *adaptif alpha-trimmed mean filter* baru yang lebih sederhana dan lebih mudah untuk penerapannya. Dengan empat percobaan, filter ini menunjukkan kinerja yang sebanding dengan Filter saya selain itu kecepatan menghitung adalah sebagai dua kali lipat . secara khusus, filter tersebut juga memiliki kemampuan yang baik untuk menangani jenis noise campuran.

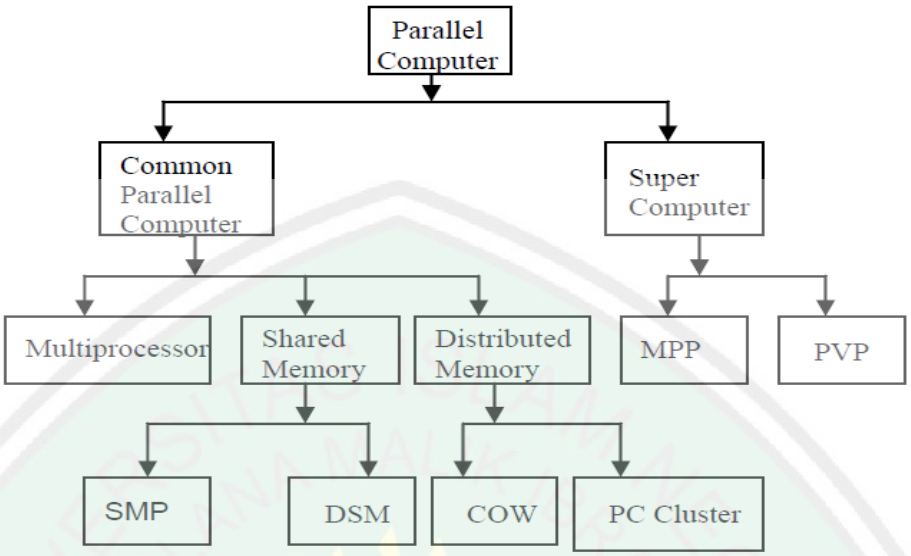
Dalam jurnal karya *Mahdi Shaneh*, dan *Arash Golibagh Mahyari* yang berjudul "*Image Enhancement using α-Trimmed Mean ε-Filters*" menjelaskan dalam akuisisi dan transmisi pada gambar sering kali noise menjadi masalah yang menjadikan suatu gambar turun kualitas.Terdapat banyak metode yang dikembangkan peneliti untuk menangani masaslah ini. Secara umum metode penghapusan noise dibagi menjadi dua jenis , yaitu linear dan nonlinear. Filter

mean adalah contoh metode linear sedangkan untuk nonlinear beberapa contohnya yaitu filter *median, min, max*. dalam menangani impulse noise metode nonlinear yang digunakan. Paling banyak diterapkan adalah filter median karena berdasarkan penerapannya hasilnya efektif dalam menangani *impulse noise*. Namun tetap saja median filter memiliki kekurangan untuk kernel image yang berukuran besar memberikan efek kabur. Pada sisi lain dikenal teknik nonlinear , metode ε-filter yang awalnya digunakan pada pengolahan suara. Metode *ε-filter* mampu menurunkan noise berdasarkan bobot dari *mean*. Filter tersebut sederhana untuk di desin dan memiliki efisiensi yang baik. Untuk hasil kerjanya noise ini menurut peneliti lebih baik dibandingkan dengan median filter. Jika dikaji lagi mean filter memiliki kinerja yang lebih baik disbanding median filter dalam menangani gausian noise.namun juga median filter bekerja lebih baik pada mempertahankan tepi-tepi gambar dan menangani noise *impulse*. Dari analisi tersebut maka dikembangkan metode untuk menyatukan mean filter dan median filter yaitu *α-trimmed mean filter*. *α-trimmed meanfilter* bekerja dengan menghapus nilai tertinggi dan terendah sebesar nilai α. Dalam penelitian ini ,peneliti menerapkan keuntungan dari kedua filter *meanfilter* dan *median filter* untuk perhitungan dari *ε-filter*. Metode yang diusulkan ini disebut *α-Trimmed Mean ε-Filter* .pada ujicobanya dibandingkan dengan metode *median filter, median ε-filter* dan *α-trimmed mean filter* pada gambar lena demgan kernel 9 x9. Berdasarkan nilai PSNR dari ke empat metode yang digunakan *α-trimmed meanε-Filter* memberikan nilai terbaik diantaranya sebesar 27.12 sedangkan *median ε-filter sebesar* 26,92 , *α-Trimmed Mean* 25,20 dan *Median filter* 25,20.

Dalam jurnal penelitian *Rafiqul Zaman Khan* dan *Md Firoj Ali* yang berjudul ”*Current Trends in Parallel Computing* “ menjelaskan bahwa Sebuah prosesor memiliki batas-batas fisik sendiri dalam kecepatan maksimum pemrosesan. Untuk mengatasi keterbatasan ini beberapa prosesor yang bekerja sama dengan satu sama lain untuk memecahkan masalah besar tantangan. Komputasi paralel mengacu pada pengolahan beberapa pekerjaan secara bersamaan pada beberapa prosesor. Masalah besar dapat dibagi menjadi beberapa tugas independen dari ukuran hampir sama dengan menerapkan teknik tugas partisi yang tepat dan setiap tugas akan dijalankan pada prosesor yang berbeda secara bersamaan. Pada saat ini banyak masalah aplikasi yang membutuhkan daya komputasi lebih dari komputer sekuensial biasa. Masalah besar tersebut bisa berupa kompleksitas komputasi yang membutuhkan waktu lama atau waktu yang tidak terprediksi. Waktu yang dibutuhkan untuk menyelesaikan masalahnya jika ditangani dengan komputer skuensial memungkinkan usang pada real time. Jadi solusi yang jelas dari masalah di atas adalah komputasi paralel yang menjamin solusi dengan menghubungkan lebih banyak jumlah prosesor melalui media komunikasi kecepatan tinggi. Model komputasi paralel terutama dari dua jenis. Yang pertama adalah memori bersama dan yang kedua komputasi terdidistribusi. Dalam arsitektur memori bersama ,sejumlah prosesor yang terhubung ke memori bersama digunakan bersama. Data atau instruksi yang dibagi melalui lock dan Semaphore dalam segi program mudah tapi kadang-kadang hasil salah. Jadi tidak semua aplikasi cocok di implementasikan pada komputasi pararel. Ada model lain yang dikenal dengaan Hybrid dimana mengkolaborasikan konsep memori

bersama dan komputasi terdistribusi. Dalam segi hadware mengalami revolusi besar hampir pada semua vendor. Perkembangan hardware mengarah pada komputasi pararel. Berdasarkan harga dan tipe arsitektur hardware terbagi menjadi arsitektur computer pararel pada umumnya dan arsitektur super computer. arsitektur computer pararel tidak membutuhkan uasaha dan biaya yang besar disbanding pada super computer untuk mencapai tujuan dari komputasi pararel. Kekuatan pemrosesan gabungan dan kapasitas penyimpanan memecahkan banyak masalah besar secara paralel dengan mudah . Dalam arsitektur multiprosesor lebih dari satu CPU yang tergabung ke satu komputer. compiler bertanggung jawab untuk *parallelizing kode* secara otomatis. Jenis arsitektur ini tidak begitu efisien tapi lebih baik daripada komputer dengan CPU tunggal. uper computing memiliki tingkat eksekusi yang sangat tinggi dan I / O throughput yang sangat tinggi. Perlu memori primer dan sekunder yang sangat besar. Sehingga biaya dan waktu adalah dua faktor penting untuk memproduksi komputer super. ada tiga pendekatan utama untuk merancang algoritma paralel. Yang pertama adalah paralelisasi dari masalah yang memungkinkan dijadikan sekuensial. Yang kedua dengan cara paralelisasi pada saat start. Yang ketiga adalah bahwa menerapkan algoritma yang sudah ada dan memecahkan masalah yang sesuai dengan kebutuhan. Ada beberapa cara untuk merancang sebuah algoritma paralel. teknik yang paling banyak digunakan adalah *partitioning*, *divide* dan *conquer*, *pipelining* dan masih banyak lainnya. Pada *partitioning*, masalah dibagi menjadi sub-masalah dengan ukuran hampir sama sehingga tidak akan tumpang tindih saat dijalankan programnya. Sub-masalah akan diselesaikan kemudian secara

bersamaan. Dalam *divide* dan *conquer* pertama masalah akan dirubah menjadi sub-masalah. Sub-masalah akan diselesaikan secara *rekursif* dan hasil dari sub-masalah digabungkan pada akhir proses. Pipelining adalah teknik sederhana namun bagus untuk menerapkan algoritma paralel. Masalahnya akan dibagi menjadi segmen dan output dari satu segmen adalah masukan dari segmen berikutnya dan mereka menghasilkan hasil pada tingkat yang sama. *Dekomposisi*, menugaskan, pemetaan dan penjadwalan adalah cara umum untuk menerapkan algoritma paralel. Sebuah pembagian yang tepat dari sebuah masalah besar menjadi beberapa sub-masalah untuk dapat dijalankan dengan algoritma paralel. Ada dua teknik utama untuk dekomposisi masalah. Pertama adalah domain dekomposisi dan kedua adalah dekomposisi fungsional. Dalam dekomposisi domain data yang terkait dengan masalah ini dibagi menjadi potongan-potongan kecil dari ukuran data yang hampir sama. Sekarang algoritma dibagi sedemikian rupa yang beroperasi pada tugas masing-masing dan pada data yang berbeda. Dalam domain tugas dekomposisi mulai secara bersamaan. Dekomposisi fungsional membagi algoritma ke dalam tugas-tugas independen yang dapat diproses secara bersamaan. Jika data yang diperlukan untuk tugas-tugas juga independen, namun tetap dihindari perulangan data. peneliti mengamati bahwa model memori bersama lebih cepat tetapi mengalami perebutan dalam penggunaan memori yang diatasi dengan model memori terdistribusi. komputasi paralel membutuhkan software khusus untuk melakukan pemrosesan paralel. MPI, OpenMP dan PVM dan masih banyak lainnya merupakan *library* software paralel untuk mengeksekusi aplikasi paralel. Berikut pembagian dalam komputasi pararel.



Gambar 2.1. Pembagian dalam computer pararel.
(Sumber :Rafiqul zaman khan & md firoj ali,2014)

Pada jurnal dari *Jie Shen, Jianbin Fang, Henk Sips,* dan *Ana Lucia Varbanescu* yang berjudul *" Performance Traps in OpenCL for CPUs"* Sebagai multi dan many-core prosesor keduanya berkembang pada dua hal,kompleksitas dan keanekaragaman, vendor hardware dan software telah mengusulkan sebuah model pemrograman standar terbuka untuk komputasi paralel, yang disebut OpenCL (Open Computing Language). Dirancang sebagai komputasi virtual Platform yang mampu berjalan di CPU, GPU, dan prosesor lainnya. Dalam OpenCL, programmer mampu mengelola OpenCL inisialisasi, transfer data dan eksekusi kernel oleh satu set ekstensi bahasa dan API runtime Yang diadopsi dari GPGPU, OpenCL memiliki banyak kemiripan dengan CUDA (Compute Unified Device Architecture). Sehingga awal kehadirannya di tahun 2009 sudah menyedot perhatian .CPU tetap menjadi processor yang umum digunakan sehingga vendor CPU (misalnya, Intel, AMD, ARM) telah merilis SDK untuk OpenCL produk mereka. Disebutkan juga oleh peneliti bahwa OpenCL boleh jadi memberikan

kinerja yang sangat baik terhadap aplikasi tertentu begitupun sebaliknya bisa jadi aplikasi tersebut tidak sesui untuk diimplementasikan dengan model pararel pada OpenCL. Sehingga cara programmer dalam mengimplementasikan sebuah aplikasi pada OpenCL menjadi hal utama dalam menentukan kesuksesan hasil . jurnal ini bertujuan untuk menjawab satu pertanyaan penting: apa faktor-faktor yang programmer perlu memperhatikan saat menulis OpenCL kode untuk CPU?.penulis menjawab dengan memulai menganalisis ketidaksesuaian dari OpenCL dengan CPU selanjutnya tentang segi kode OpenCL. Peneliti menggunakan Intel Xeon E5620 2.40GHz hiper-threaded Dual quad-core (8 core, 16 thread hardware) sebagai utama kami platform perangkat keras. Masing-masing inti memiliki cache yang 32KB L1 Data dan 256 KB L2 cache yang dibagi antara dua benang hardware inti. Cache 12MB L3 dibagi oleh semua core. Itu OpenCL kernel dikompilasi dengan dua kompiler OpenCL: (1) Intel OpenCL (OCl) SDK 1.5 dan (2) AMD Accelerated Paralel Processing (APP) SDK 2.6. diperoleh beberapa cacatan bahwa spesialisasi datang dalam bentuk parameter tunning (misalnya, mengubah granularity, beralih antara baris dan kolom order, mengaktifkan / menonaktifkan menyalin data), dan tidak mengubah struktur aplikasi dan paralelisasi. Selain itu diperoleh hasil tentang kode OpenCL setara atau sedikit lebih baik dari kode OpenMP. Yang harus dilakukan seorang programmer terangkum sebagai berikut:

1) Periksa metode transfer data dan menerapkan copy nol teknik.

2) Tentukan masalah dekomposisi granularity, dan meningkatkan beban kerja per pekerjaan-item dan / atau menerapkan ubin.

3) Jika aplikasi proses dataset 2D, periksa pola akses memori, dan pastikan agar baris-besar

digunakan di semua tempat.

4) Terapkan vectorization eksplisit atau mencoba untuk mengaktifkan implisit vektorisasi. Bila menggunakan vektorisasi implisit, datadependent cabang harus dihapus, jika mungkin, untuk mengurangi hukuman.

5) Periksa eksperimen apakah ukuran kerja-kelompok ini optimal untuk memaksimalkan kinerja.

6) Periksa penggunaan memori lokal, sebagai manfaat dari menggunakan memori lokal tidak dijanjikan. Umumnya, memori lokal tidak dianjurkan pada CPU.

Penulis menganalisis kinerja kode OpenCL dari perspektif portabilitas. Ternyata implementasi OpenCL sesuia dengan kerja CPU selain itu dapat membawa perbaikan kinerja hingga 10 × lebih baik. Transformasi ini menargetkan tiga aspek utama: (1) Data lokalitas dan pemanfaatan Cache, (2) penghapusan menyalin yang tidak perlu, dan (3) Unit pemanfaatan SIMD. Hasil dari penelitian ini menunjukkan bahwa peningkatan efisiensi pada kinerja CPU. Namun hasil juga menunjukkan programmer untuk tidak mengubah struktur kode OpenCL asli.

Dalam sebuah jurnal dari *Robert Dietrich* dan *Ronny Tsch¨uter* yang berjudul "*A Generic Infrastructure for OpenCL Performance Analysis*" menjelaskan hasil analisa tentang sebuah framework OpenCL (Open Computing language),dengan versi terbarunya yang rilis yaitu versi 2.0. OpenCL adalah standart pemrograman untuk komputasi pararel pada platform heterogeneous.OpenCL bias berjalan pada hampir semua jenis compute device

seperti prosesor multi-core,GPUs,DSPs,FPGAs ,dan masih banyak lainnya. Pada penelitian ini mendiskusikan peneliti mengevaluasi analisis kinerja OpenCL. Pada OpenCL,setiap vendor yang produknya support memiliki tool untuk menganalisis kinerja OpenCL yang bekerja pada hanya perangkatnya saja. AMD dengan AMD CodeXL untuk menganalis program OpenCL yang berjalan pada GPU AMD. NVIDIA meneyediakan CUPTI untuk menganalisis kinerja OpenCL .selain contoh dari dua vendor itu masih banyak lagi tool untuk mengukur kinerja OpenCL seperti pada penelitian ini menggunakan Score-P. Score-P mendukung pada banyak model pemrograman,seperti MPI,SHMEM,OpenMP,Pthreads,dan CUDA. Kinerja OpenCL yang terekam pada runtime program dimuat dengan format yang sudah terstandarisasi ssehingga bias digunakan sebagai data analisis pada tool lainnya.Dalam OpenCL kinerja dapat diukur melalui beberapa model ,model eksekusi pada OpenCL terjadi pada 2 bagian yaitu Kernel dan Host Program diaman host sebagai pengontrol kernel . Dalam host harus membuat Contex untuk bisa mengisikan perintah-perintah didalamnya dengan mendefinisikan juga *work item, index space, dan work group*. Model eksekusi ada yang *In-Order-Execution* dan *Out-Order-Execution*. Fokus utama pada model eksekusi adalah pada pemetaan data secara pararel pada kernel dalam sebuah fungsi pararel yang ditulis dengan bahasa OpenCL Cdan dieksekusi pada sebuah compute device. Jalannya kernel terekam dalam runtime. Selanjutnya pada profiling / riwayat dri OpenCL,dimana berfungsi untuk mengukur waktu yang dibutuhkan untuk menjalankan program. Periwayatan termuat dalam 64-bit yang berisi informasi waktu saat itu pada compute device dalam format nanosecond

untuk satuan wkatu yang dihitung . Peneliti melakukan percobaan pada jurnal ini untuk menggali informasi yang untuk kemudian dilakukan analisis terhadap hasil eksekusi yang dijalankan dengan menuliskan kode program seperti listing kode pada listing code 2.1.Pada OpenCL dapat memanggil fungsi *clGetEventProfilingInfo*untuk mendapatkan informasi terkait pada riwayat event-event yang dijalankan pada OpenCL. Informasi lain yang bisa diperoleh dari API OpenCL adalah detail informasi dari *context, compute device,command queue, kernel* atau *memory object*. Informasi-informasi tersebut perlu diketahui programmer untuk menjadi acuan dalam memodelkan masalah yang akan diselsaikan. Selanjutnya analisis pada hal kinerja *interface*,dimana inti dari kinerja openCl adalah bagaimana tentang penggunaan *buffer*, waktu untuk sinkronisasi pada host dan device GPU. Peneliti melakukan percobaan pengukuran kinerja OpenCL pada beberapa vendor yaitu ARM SoC dengan Mali-T604 GPU dan NVIDIA Tesla K20Xm GPU yang keduanya masih support untuk OpenCL versi 1.1. OpenCL versi 1.2 diujicobakan pada platforms Intel Xeon E5-2690, Intel Phi 5110p, and AMD Tahiti HD7950 GPU. Hasil dari kasus scenario yang dibuat untuk pengujian. Perhitungan dilakukan pada waktu yang dibutuhkan untuk menjalankan program. Hasil yang diperoleh dari data input berupa runtime OpenCL dari masing-masing processor diperoleh hasil overhead yang sama. Untuk melakukan analisis tersebut dilakukan uji coba dengan mengetikkan source code seperti pada listing 2.1.

```
{
  CL_PROFILING_RECORD_NONE = 0,
  CL_PROFILING_RECORD_KERNEL = (1 << 0),
  CL_PROFILING_RECORD_MEMCPY = (1 << 1)
} cl_profiling_record_type;
{
  CL_PROFILING_DOMAIN_NONE = 0,
  CL_PROFILING_DOMAIN_API = 1,
  CL_PROFILING_DOMAIN_RESOURCE = 2,
  CL_PROFILING_DOMAIN_SYNCHRONIZE = 3,
} cl_profiling_domain_id;
{
  CL_PROFILING_ALL = 0,
  CL_PROFILING_clCreateContext = 1,
  CL_PROFILING_clReleaseContext = 2,
  CL_PROFILING_clReleaseDevice = 3,
  ...
  CL_PROFILING_clFinish = 42;
} cl_profiling_callback_id;
{
  CL_PROFILING_API_EVENT_START = 0,
  CL_PROFILING_API_EVENT_STOP = 1
} cl_profiling_api_event_type;
{
  cl_profiling_api_event_type event_type;
  cl_profiling_callback_id callback_id;
  cl_profiling_domain_id domain_id;
} cl_profiling_api_callback_info;
{
  cl_profiling_record_type type;
  uint64_t queued;
  uint64_t submit;
  uint64_t start;
  uint64_t end;
} cl_device_record;
void clProfilingRegisterBufferCallbacks(
  uint64_t (*get_host_time_stamp( void ) ),
  void (*buffer_request_callback( cl_device_record* buffer,
  size_t bytes ) ),
  void (*buffer_complete_callback( cl_device_record* buffer,
  cl_command_queue queue,
  size_t valid_bytes) ) );
void clProfilingDeviceEnable( cl_profiling_record_type record_types );
void clProfilingDeviceDisable( cl_profiling_record_type record_types );
void clProfilingRegisterApiCallback( cl_profiling_domain_id domain_id,
  cl_profiling_callback_id callback_id,
  void (*apiCallback( cl_profiling_api_callback_info* nfo ) ) );
void clProfilingApiEnable( cl_profiling_domain_id domain_id,
  cl_profiling_callback_id callback_id );
void clProfilingApiDisable( cl_profiling_domain_id domain_id,
  cl_profiling_callback_id callback_id );
```

Listing 2.1. *OpenCL Tools Interface Proposal*

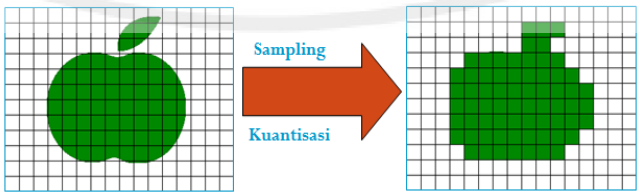
Sumber *:* Robert dietrich & ronny tsch¨uter ,2015

2.2. Pengolahan Citra

Pengolahan citra digital adalah pemrosesan citra menjadi citra yang lain dengan kualitas yang lebih baik. Citra yang telah menjadi gambar lain menggunakan algoritma atau teknik tertentu. Pengolahan citra mempunyai tujuan untuk proses memperbaiki kualitas citra agar mudah diinterpretasikan oleh manusia atau komputer , teknik pengolahan citra dengan mentrasformasikan citra menjadi citra lain, dan pengolahan citra merupakan proses awal dari komputer visi

Pada umumnya pengolahan citra berhubungan dengan citra-citra digital. Dalam hal ini, citra f(x,y) diperoleh secara diskrit dan kemudian dikuantisasi. Maka akan diperoleh suatu citra baru, : (m,n) → I dengan m,n ∈ I; di mana I adalah himpunan bilangan bulat (*integer*). Namun demikian, secara umum sistem pengolahan citra mengandaikankan citra asal yang bernilai riil dan menghasilkan bilangan riil juga, meskipun secara teknis pada akhirnya citra ini didigitalkan sebelum disimpan.

Ada beberapa hal yang penting didalam pengolahan citra digital, antara lain teknik-teknik pengambilan citra, model citra digital, sampling dan kuantitasi, histogram, proses filtering, perbaikan citra sampai pada pengolahan citra digital yang lebih lanjut seperti segmentasi, *image clustering* dan ekstrasi ciri.

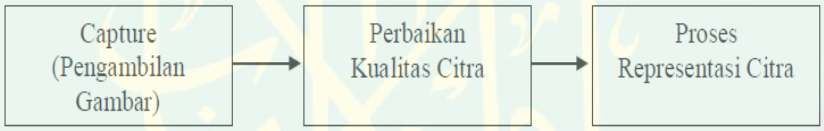


a b

Gambar 2.2 (a) citra Analog dan(b) Citra Digital .
Sumber :Mahmud Yunus ,2013.

Citra digital merupakan representatif dari citra yang diambil oleh mesin dengan bentuk pendekatan berdasarkan sampling dan kuantitasi. Sampling menyatakan besarnya kota-kotak yang disusun dalam baris dan kolom. Dengan kata lain sampling pada citra menyatakan besar kecilnya ukuran *pixel* (titik) pada citra, jika kuantitasi menyatakan besarnya nilai tingkat kecerahan yang dinyatakan dalam nilai tingkat keabuan (*gray scale*) sesuai dengan jumlah bit biner yang digunakan oleh mesin, maka dapat dikatakan bahwa kuantitasi pada citra menyatakan jumlah warna yang ada pada citra.

Proses pengolahan citra secara diagram yaitu proses dimulai dari pengambilan citra, perbaikan citra sampai dengan pernyataan representatif citra.



Gambar 2.3 Proses Pengolahan Citra

Secara umum teknik pengolahan citra digital dibagi menjadi 3 tingkat pengolahan:

1. Tahap 1 yang dinamakan dengan *Low-Level Processing* (pengolahan tingkat rendah). Pengolahan ini operasional-operasional dasar dalam pengolahan citra, seperti pengurangan *noise* (*noise redution*), perbaikan citra (*image enhancement*) dan restorasi citra (*image restoration*).

2. Tahap 2 yang dinamakan dengan *Mid-Level Processing* (pengolahan tingkat menengah). Pengolahan ini meliputi segmentasi pada citra, deskripsi objek dan klasifikasi objek secara terpisah.

3. Tahap 3 yang dinamakan dengan *High-Level Processing* (pengolahan tingkat tinggi), yang meliputi analisis citra.

Operasi – operasi pengolahan citra meliputi perbaikan kualitas citra, yakni perbaikan kualitas citra ini bertujuan memperbaiki kualitas citra dengan memanipulasi parameter-parameter citra. Melalui operasi ini, ciri-ciri khusus yang terdapat dalam didalam citra dapat lebih ditonjolkan. Dalam perbaikan kualitas citra dapat dilakukan operasi – operasi citra, seperti yang tertulis dalam buku Pengolahan Citra Digital yaitu

a. Perbaikan kontras gelap dan terang

b. Perbaikan tepian objek

c. Penajaman

d. Pemberian warna semu

e. Penapisan derau .

Jadi pengolahan citra merupakan usaha manusia untuk mencari kejelasan informasi dari suatu gambar yang dianggap perlu adanya perbaikan kualitas sehingga informasi yang didapatkan jelas.Tradisi tabayyun merupakan tradisi ajaran Islam yang dapat menjadi solusi dari zaman ke zaman. Terutama bagi informasi-informasi yang berpotensial memunculkan konflik dalam masyarakat. Metode tabayyun merupakan proses klarifikasi sekaligus analisis atas informasi dan situasi serta problem yang dialami umat. Harapannya akan mendapatkan hasil kesimpulan yang lebih bijak, arif dan lebih tepat sesuai keadaan masyarakat sekitarnya. Allah SWT memberikan pelajaran bagi kita tentang betapa pentingnya tabayyun bagi hal-hal yang belum jelas informasinya, dalam firmanNya yang berbunyi :

يَأَيُّهَاالَّذِيْن آمنُوْا إِنْ جآءَكمْ فَاسقٌ بِنَبإٍ فتبيّنُوْا أنْ تُصِبوْا قوْمًا بِجَهَالةٍ فتُصْبِحُوْا علَى مَا فعَلْتُمْ ندمين

"Hai orang-orang yang beriman, jika datang kepadamu orang fasik membawa suatu berita, maka periksalah dengan teliti agar kamu tidak menimpakan suatu musibah kepada suatu kaum tanpa mengetahui keadaannya yang menyebabkan kamu menyesal atas perbuatanmu itu."(QS. Al-Hujurat: 6).

Pada ayat lain Allah SWT berfirman :

وَلَا تَمْشِ فِي الْأَرْضِ مَرَحًا ۖ إِنَّكَلَنْتَخْرِقَالْأَرْضَوَلَنْتَبْلُغَ الْجِبَالَ طُولًا

"Janganlah kamu mengikuti apa yang kamu tidak ketahui. Sesungguhnya pendengaran, penglihatan, dan hati, semuanya itu akan dimintai pertanggungjawabannya. (QS al-Isrâ': 36).

Ayat tersebut, mengandung makna yang selaras dan saling melengkapi dengan ayat yang telah tersebutkan sebelumnya. Ayat pertama menyebutkan keharusan bertabayyun terhadap adanya suatu berita atau informasi ataupun datangnya suatu pemahaman dan cara berpikir keberagamaan yang baru. Sedangkan pada ayat kedua disiratkan tidak diperkenankannya mengikuti sesuatu yang belum diketahui secara jelas. Menyiratkan pula adanya proses tindak lanjut terhadap sesuatu yang belum diketahui, agar dapat diketahui secara benar dan jelas. Aktivitas pendengaran, aktivitas penglihatan dan aktivitas hati akan dimintakan pertanggungjawabannya oleh Allah SWT.

Tafsir dari ayat tersebut secara lengkap untuk bagian per kata dijelaskan bahwa يَـأَيُّهَاالَّذِيْن آمنُوْا(wahai orang- orang yang beriman). Ayat ini diawali dengan seruan kepada ahlul-îmân. Disamping kasus ini terjadi di antara kaum beriman seperti yang kami paparkan di atas, juga karena berkaitan dengan perintah yang tidak sah dilaksanakan kecuali oleh orang yang beriman. Ayat ini, sekaligus

menunjukkan bahwa penyelewengan terhadap perintah ini dapat mengurangi kadar keimanan seseorang. Oleh karena itu, mari kita mempersiapkan telinga dan hati, seraya memohon kepada Allah agar melapangkan dada kita dengan nasihat ayat ini.

إِنْ جَآءَكُمْ فَاسِقٌ بِنَبَإٍ (jika ada orang fâsiq yang datang kepadamu dengan membawa berita penting). An-Naba`, artinya isu (kabar) penting. Adapun orang faasiq, ialah pelaku fusuuq, yaitu orang yang keluar dari ketaatan kepada Allah. Setiap kemaksiatan adalah fusuuq. Karena itu, faasiq diklasifikasikan menjadi dua macam, yaitu fâsiq besar dan fâsiq kecil. Fâsiq besar, identik dengan kufur besar, yang mengeluarkan pelakunya dari agama Islam. Dinyatakan oleh Allaah Ta'ala dalam banyak ayat al-Qur`ân:

إِنَّ الْمُنَافِقِينَ هُمُ الْفَاسِقُون

"Sesungguhnya orang-orang munaafik itulah orang-orang yang fâsiq"(at-Taubah:67).

Kita juga mengetahui, kemunafikan kaum munafikin pada zaman Nabi Shallallahu 'alaihi wa sallam yang sering disebutkan dalam Al-Qur`ân ialah kemunafikan i'tiqâdi (besar). Begitu pula tentang Fir'aun dan para pengikutnya.

إِنَّهُمْ كَانُوا قَوْمًا فَاسِقِينَ

"Sesungguhnya mereka adalah kaum yang fâsiq". [al-Qashash:32].

Kefâsikan kecil, identik dengan dosa besar yang tidak mengeluarkan pelakunya dari agama Islam. Seperti berbohong, mengadu domba, memutuskan perkara tanpa melakukan tabayyun (penelitian terhadap kebenaran beritanya) terlebih dahulu. Hal ini banyak pula disebutkan Allah, di antaranya pada ayat-ayat berikut.

وَلَا يُضَارَّ كَاتِبٌ وَلَا شَهِيدٌ ۚ وَإِنْتَفْعَلُوافَإِنَّهُفُسُوقٌبِكُمْ ۗ وَاتَّقُوااللَّهَ ۖ وَيُعَلِّمُكُمُاللَّهُ ۗ وَاللَّهُبِكُلِّشَيْءٍعَلِيمٌ

"dan janganlah penulis dan saksi saling sulit menyulitkan. Jika kamu lakukan (yang demikian), maka sesungguhnya hal itu adalah suatu kefasikan pada dirimu. Dan bertakwalah kepada Allah; Allah mengajarmu; dan Allah Maha Mengetahui segala sesuatu." [al-Baqarah:282].

الْحَجُّ أَشْهُرُ مَّعْلُومَاتٌ فَمَنْ فَرَضَ فِيهِنَّ الْحَجَّ فَلاَ رَفَثَ وَلاَ فُسُوقَ وَلاَ جِدَالَ فِي الْحَجِّ

"Maka barang siapa yang telah menentukan pada bulan- bulan tersebut untuk berhajji, maka janganlah rafats, jangan pula melakukan fusûq, jangan pula berdebat pada saat berhaji." (al-Baqarah:197).

Dalam menafsirkan kata (fusûq) dalam ayat *al-Baqarah:197*, para ulama mengatakan, yaitu perbuatan maksiat . Dan kefasikan yang dilakukan oleh shahâbi (sahabat) dalam sababun-nuzûl ayat ini, yaitu kebohongannya dalam menyampaikan berita. Imam Al-Qurthubi berkata: "Al-Walîd dinyatakan fâsiq, artinya berbohong".[Sehingga, dampak dari indikasi fâsiq menunjukkan bahwa apabila kebohongan saja yang merupakan kefasikan kecil sudah mengharuskan kita mewaspadai serta perlu untuk tabayyun, maka apalagi jika perbuatan itu merupakan fâsiq besar.

فتبّينوا (maka telitilah dulu). Ada dua qirâ`ah pada kalimat ini. Jumhûr al-Qurrâ membacanya "fatabayyanû", sedangkan al-Kissâ`I dan para qurrâ` Madinah membacanya "fatatsabbatû". Keduanya benar dan memiliki makna yang sama. Tentang kalimat ini, ath-Thabari memaknainya: "Endapkanlah dulu sampai kalian mengetahui kebenarannya, jangan terburu-buru menerimanya" Syaikh al-Jazâ`iri mengatakan, artinya, telitilah kembali sebelum kalian berkata, berbuat atau memvonis. أنْ تُصِبوْا قوْمًا بِجَهَالةٍ (agar jangan sampai kalian menimpakan suatu bahaya pada suatu kaum atas dasar kebodohan). Keterkaitan makna antara

ketidaktahuan dengan kesalahan sangat erat, sehingga kata “jahâlah” dimaknai kesalahan. Imam Al-Qurthubi mengatakan, “bi jahâlah,” maksudnya ialah secara salah. Adapun kesalahan yang terus dibela serta dicari-cari pembenarannya dengan berbagai dalih, maka demikian ini merupakan sifat dan kebiasaan kaum Nashara, sehingga Allah Ta’ala menyebut mereka dengan azh-zhâllîn. Yaitu orangorang yang tersesat sebagaimana disebutkan dalam suurat al-Fâtihah. Penjelasan dari satu pihak yang mengadu tanpa tabayyun kepada yang diadukan, dapat menyebabkan keruhnya pandangan kita terhadap seseorang yang asalnya bersih, sehingga kita berburuk sangka kepadanya, enggan bertemu dan bahkan memboikotnya, dan akibat yang ditimbulkannyapun meluas. Jika dalam perdagangan bisa menurunkan omzet, dalam pergaulan menurunkan simpati, dalam dakwah menjadikan ummat tidak mau menerima nasihat dan pelajaran yang disampaikannya, bahkan bisa sampai pada anggapan bahwa semua yang diajarkannya dianggap tidak benar. Jika demikian, maka yang mendapat kerugian ialah ummat.فتُصْبِحُوْا علَى مَا فعَلْتمْ نـدميـن (kemudian kalian menyesal atas perlakuan kalian). Allah Ta’ala menyebutkan penyesalan ini akan menimpa seseorang yang salah dalam menjatuhkan keputusan karena memandang suatu masalah (perkara) tanpa tabayyun, dan bukan dari orang yang diisukan negatif. Karena yang memvonis ini telah berbuat zhalim. Sedangkan yang tertuduh tanpa bukti, ia berarti mazhlûm (*terzhalimi*) (almanhaj.or.id). Padahal Rasûlillâh Shallallahu ‘alaihi wa sallam pernah bersabda kepada Mu’adz bin Jabal Radhiyallahu anhu :

بْيَنَها وَبْيَنَ اللهِ حَجابٌلَيسفإِنَّهاٌلَمْظْلُومواَّتقِ دْعَوَة

"Dan hindarilah doa orang yang terzhalimi. Sesungguhnya tidak ada tabir penghalang antara doa orang yang terzhalimi dengan Allah "(Shahîh al-Bukhâri, al-Mazhâlim (9, 3/99)).

2.3. Noise

Noise adalah suatu gangguan yang disebabkan oleh penyimpanan data digital yang diterima oleh alat penerima data gambar yang dapat mengganggu kualitas citra. Noise dapat disebabkan oleh gangguan fisik (optik) pada alat penangkap citra misalnya kotoran debu yang menempel pada lensa foto maupun akibat proses pengolahan yang tidak sesuai. Ada tiga jenis noise yang umum dikenal , yaitu *Gaussian, Speckle,* dan *Salt Pepper.*

1.*Gaussian Noise*merupakan model noise yang mengikut i distribusi normal standar dengan rata-rata nol dan standar deviasi 1. Efek dari GaussianNoiseini pada gambaradalah munculnya tit ik-titik berwarna yang jumlahnya sama dengan persentase noise .

2.*Speckle Noise* merupakan model noise yang memberikan warna hitam pada titik yang terkena noise.

3.*Salt and Pepper Noise* adalah bentuk noise yang biasanya terlihat titik-titik hitam dan putih pada citra seperti tebaran salt dan pepper. Penerapan dari noise ini tergantung nilai pembangkit yang di pakai . Nilai yang dibangkitkan adalah nilai 0 dan 255 yang menunjukkan warna hitam dan putih

(a) (b) (c)

Gambar 2.4 Macam-macam noise (a) *Noise Gaussian* , (b) *Noise Speckle*, dan (c)*Noise Salt and Pepper*

Sumber : Basuki A,2005.

Pada suatu pengolahan citra terkadang untuk menguji suatu algoritma untuk dapat mereduksi noise, maka noise dihasilkan dari proses pembangkitan noise. Untuk Membangkitkan noise digunakan suatu bilangan acak sebagai pengganti noise yang dihasilkan (Basuki ,2005).

2.3.1. Jenis Noise Additive

Noise Additive adalah noise yang ditambahkan untuk meciptakan tiruan. Noise *additive* diciptakan berdasarkan *Probabilty Density Function* (PDF) kemudian diaplikasikan ke piksel yang bersangkutan . Adapun beberapa macam noise diantaranya ,noise Gaussian adalah model Noise yang memiliki fungsi kerapatan probabilitas (probability density function / FORMULA) yang diberikan oleh *kurva Gaussian*. Noise *salt* dan *pepper* biasa dinamakan sebagai Noise *impulspositif dan negatif, Noise tembakan,* atau *Noise biner*. Noise ini biasa disebabkan oleh gangguan yang tiba-tiba dan tajam pada proses perolehan isyarat citra. Bentuknya berupa bintik-bintik hitam atau putih di dalam citra.. Noise

*salt*dan *pepper*, sering muncul pada citra yang diperoleh melalui kamera. Noise eksponensial (terkadang dinamakan Noise eksponensial negatif) merupakan jenis Noise yang dihasilkan oleh laser yang koheren ketika citra diperoleh. Oleh karena. *Noise gamma* (atau Erlang) merupakan efek penapisan lolos-rendah terhadap citra yang mengandung *Noise eksponensial* sebagai hasil pengambilan citra yang teriluminasi oleh laser yang *koheren* (*Myler dan Weeks*, 1993. Noise periodis biasa terjadi karena interferensi listrik maupun elektromekanis selama citra diakuisisi. Noise ini biasanya berbentuk *sinusoidal*. Sifat periodis dapat berbentuk statisioner yaitu memiliki *amplitudo, frekuensi*, dan fase yang tetap, tetapi dapat juga *nonstasioner* dengan nilai *amplitudo, frekuensi*, dan fase berubah di sepanjang area citra. Sehingga pixel keseluruhan akan di proses dengan rumus untuk jenis noise additive yang di terapkan sesuai kebutuhan.

2.4. Alpha-Trimmed Mean

Sebuah *Alpha-Trimmed mean* adalah ukuran statistik dari tendensi sentral, seperti mean dan median. Ini melibatkan perhitungan mean setelah membuang diberi bagian dari distribusi probabilitas atau sampel pada tinggi dan low end, dan biasanya membuang jumlah yang sama dari keduanya. Jumlah ini poin yang harus dibuang biasanya diberikan sebagai persentase dari jumlah total poin, tetapi juga dapat diberikan sebagai jumlah poin tetap.

Untuk aplikasi statistik yang paling, 5 sampai 25 persen dari ujung dibuang; 25% dipangkas berarti (ketika terendah 25% dan tertinggi 25% yang dibuang) dikenal sebagai mean interkuartil. Misalnya, diberikan satu set 8 poin, pemangkasan

12,5% akan membuang minimum dan nilai maksimum dalam sampel: nilai terkecil dan terbesar, dan akan menghitung rata-rata dari sisa 6 poin.

alpha-trimmed mean filter menjelaskan , di mana {x(i),x(i-1),….,x(i-n+1)} yang mana n=2N+1 menjadi seperangkat nilai-nilai sinyal n sampel yang sedang diamati pada sebuah window,Wi. Jika nilai-nilai ini diatur dalam urutan rendah ke tinggi , menghasilkan:

$$x(1)(i) \leq x(2)(i) \leq \ldots \leq x(n)(i) \quad (2.1)$$

Yang mana x(1)(i) adalah minimum, x(n)(i) adalah maksimal, dan X (N + 1) (i) adalah median dari himpunan tersebut (*Remzi* ,2004). output dari alpha-trimmed mean filter, y (i, α) ditunjukkan pada rumus berikut :

$$y_n(i;\alpha) = \frac{1}{n - 2[\alpha n]} \sum_{j=[\alpha n]+1}^{n-[\alpha n]} x_{(j)}(i) \quad (2.2)$$

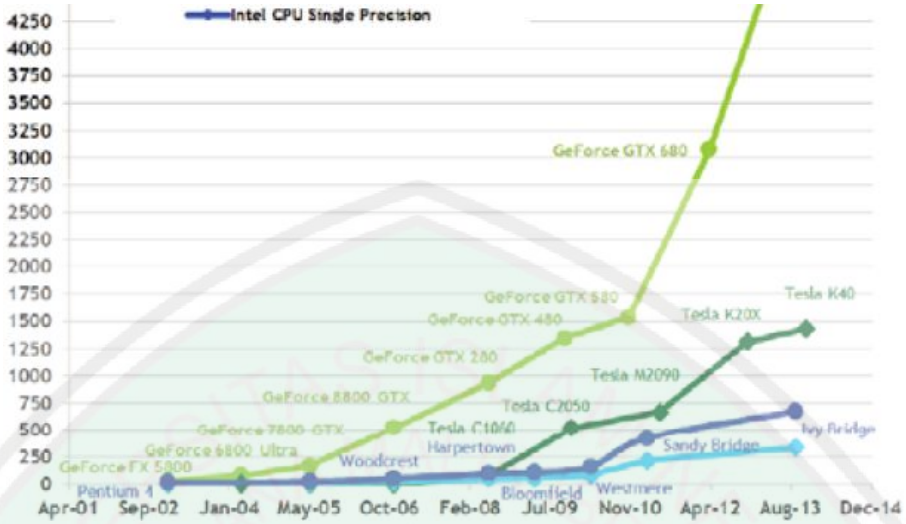
Seperti terlihat pada rumus , α menunjukkan persentase untuk pemangkasan . Oleh karena itu, alpha-trimmed mean filter sama halnya melakukan median filter ketika nilai α mendekati 0,5, dan sama halnya dengan mean filter ketika nilai α dekat dengan 0. Setelah proses pemangkasan sisa himpunan tersebut digunakan untuk penghitungan mean aritmetik.

2.5. Komputasi

Komputasi paralel adalah ide yang sederhana. Otak manusia benar-benar bekerja secara paralel. Konsep dasar dari komputasi paralel adalah mendistribusikan beban kerja diantara individual prosesor yang bekerja bersama-

sama dalam melakukan komputasi. Bagaimanapun, sebuah *single computer* hanya dapat melakukan satu hal pada satu waktu secara sekuensial. Sebuah *parallel computer*, dimana komputer tersebut dapat melakukan banyak komputasi pada waktu yang bersamaan, dan dapat menyelesaikan masalah sederhana dalam beberapa menitsaja yang seharusnya diselesaikan dengan membutuhkan waktu jam atau hari pada *single computer* (*Jackson*,2009).

Sejak tahun 2003 Terdapat 2 arsitektur dalam mengembangan mikroprosesor, yaitu *multicore* dan *manycore* (*Kirk&Hwu*, 2010). Arsitektur *multicore* bermula dari prosesor berinti dua dan terus bertambah hingga saat ini sudah mencapai 8 inti pada satu mikroprosesor. Arsitektur *multicore* banyak digunakan pada *central processing unit* (CPU). Arsitektur ini berfokus untuk memberikan performa yang baik pada program yang didesain untuk berjalan pada banyak inti, dan juga tetap menjaga performa kecepatan eksekusi program yang didesain secara sekuensial. Arsitektur *many-core* berfokus untuk memaksimalkan kinerja programyang benar-benar didesain untuk bekerja secara paralel.Arsitektur *many-core* memiliki lebih banyak unit pemroses daripada arsitektur *multicore* dan arsitektur ini biasanya digunakan pada *graphic processing unit* (GPU).Perbedaan kecepatan antara CPU dan GPU disebabkan karena GPU dikhususkan untuk perhitungan yang intensif dan paralel (NVIDIA, 2014), tetapi GPU tidak dapat menggantikan CPU sepenuhnya karena GPU hanya didesain untuk melakukan kalkulasi numerik dan untuk memproses secara paralel saja, bukan sekuensial seperti gambar berikut.

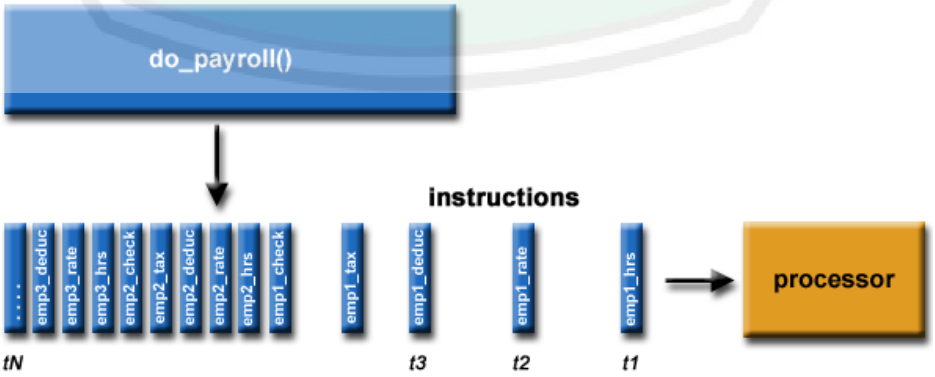


Gambar 2.5Grafik Perbandingan Performa CPU dan GPU

Sumber :*Goycola Raul*,2012.

. Dewasa ini model komputasi yang digunakan umumnya menerupakan komputasi serial kemudian disusul teknologi baru yaitu komputasi pararel

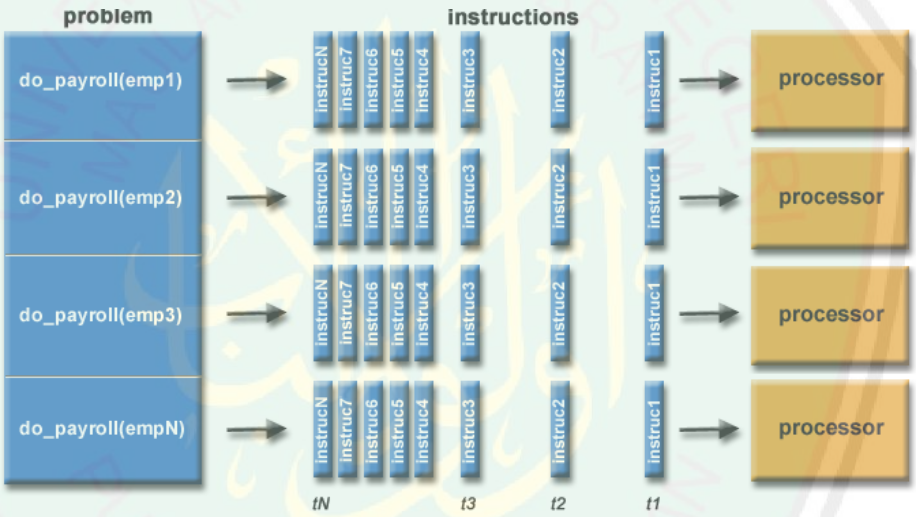
Secara tradisional komputasi serial, perangkat lunak yang telah ditulis untuk perhitungan serial dengan ciri Suatu masalah dipecah menjadi serangkaian diskrit instruksi,instruksi dieksekusi secara berurutan satu demi satu, dieksekusi pada prosesor tunggal, dan hanya satu instruksi dapat mengeksekusi setiap saat dalam waktu. Ilustarsi kerja komputasi serial sebagai berikut (*Goycola Raul*,2012.) .



Gambar 2.6 Arsitektur komputasi serial

Sumber*: Goycola Raul*,2012.

Komputasi Paraleldalam arti yang paling sederhana, adalah penggunaan simultan dari beberapa sumber komputasi untuk memecahkan masalah komputasi dengan ciri suatu masalah dipecah menjadi bagian-bagian diskrit yang dapat diselesaikan secara bersamaan, setiap bagian selanjutnya dipecah menjadi serangkaian instruksi Instruksi dari setiap bagian mengeksekusi secara bersamaan pada prosesor yang berbeda, mekanisme kontrol / koordinasi secara keseluruhan digunakan. Ilustarsi kerja komputasi pararel sebagai berikut



Gambar 2.7. Arsitektur komputasi paralel

Sumber : *Goycola Raul,2012.*

2.6. GPGPU

GPGPU singkatan *General-Purpose computation on Graphics Processing Units*, yang berarti, tujuan umum komputasi pada *Graphics Processing Unit* (GPU). GPGPUs digunakan untuk tugas-tugas yang sebelumnya domain dari CPU-daya tinggi, seperti perhitungan fisika, enkripsi / dekripsi, perhitungan ilmiah dan generasi mata uang cypto seperti Bitcoin. Karena kartu grafis yang dibangun untuk paralelisme besar, mereka dapat mengerdilkan tingkat perhitungan bahkan CPU yang paling kuat untuk banyak tugas pemrosesan

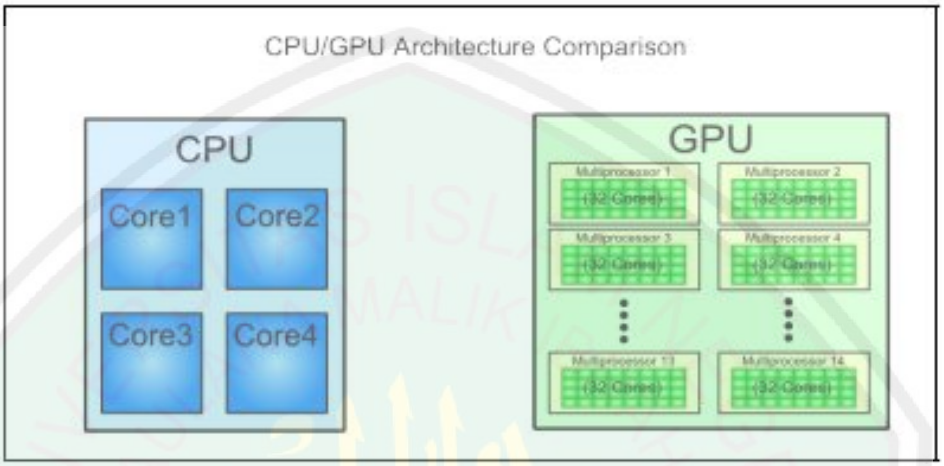
paralel. *Core shader* yang sama yang memungkinkan beberapa piksel yang akan diberikan secara bersamaan dapat pula memproses beberapa aliran data pada saat yang sama. Meskipun inti shader hampir tidak serumit CPU, GPU *high-end* mungkin memiliki ribuan *core shader*. Sebaliknya, CPU multicore mungkin memiliki delapan atau dua belas core.

Telah ada peningkatan fokus pada GPGPUs sejak DirectX 10 termasuk shader terpadu dalam spesifikasi shader inti untuk Windows Vista. Bahasa tingkat tinggi sedang dikembangkan sepanjang waktu untuk mempermudah pemrograman untuk perhitungan pada GPU. Kedua AMD / ATI dan Nvidia memiliki pendekatan untuk GPGPU dengan API produk mereka sendiri (OpenCL dan CUDA).

Nvidia GeForce 3 adalah GPU pertama yang menampilkan shader diprogram. Pada saat itu, tujuannya adalah membuat grafis 3D raster lebih realistis; kemampuan GPU baru diaktifkan 3D transform, benjolan pemetaan, pemetaan specular dan perhitungan pencahayaan. ATI 9700 GPU, yang pertama DirectX kartu 9-mampu mendekati fleksibilitas pemrograman CPU, meskipun beberapa perhitungan tujuan umum dilakukan pada saat itu. Dengan diperkenalkannya Windows Vista, dibundel dengan DirectX 10, core shader terpadu yang ditentukan sebagai bagian dari standar. Potensi kinerja baru ditemukan GPU menunjukkan meningkatkan beberapa kali lipat lebih perhitungan berbasis CPU.

GPU yang awalnya dikembangkan untuk mempercepat raster 3D (seperti raytracing terlalu mahal perhitungan-bijaksana) telah melampaui kinerja CPU untuk raytraced pra-diberikan grafis. Meskipun raytracing belum digunakan dalam game, telah ada demonstrasi real-time. Kemajuan GPGPUs berarti bahwa

dalam waktu yang tidak terlalu lama, komputer grafis harus mampu dari jenis yang sama geometri intensif dan pencahayaan sebagai film 3D.



Gambar 2. 8 Perbandingan Arsitektur CPU/GPU .
Sumber: Lu mian,OmniDB,2009.

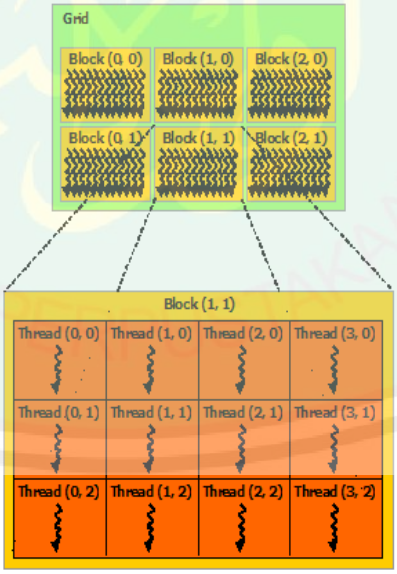
2.7. Nvidia CUDA

CUDA (*Compute Unified Device Architecture*) merupakan API yang dikembangkan oleh NVIDIA. CUDA adalah model pemrograman untuk komputasi paralel yang memungkinkan peningkatan dalam kinerja komputasi dengan memanfaatkan kelebihan dari GPU (NVIDIA, 2014). Pada tahun 2006 , NVIDIA meluncurkan bahasa pemrograman yang bernama CUDA, dan bahasa pemrograman tersebut secara khusus untuk melakukan komputasi yang umum atau sering disebut dengan GPGPU(*General Purpose Graphic Processing Unit*). CUDA adalah ekstensi dari bahasa pemrograman C yang ditambahkan dengan beberapa sintax untuk bekerja dengan GPU (*Jackson*, 2009).

CUDA pada dasarnya terbagi menjadi dua bagian utama , yaitu *Thread* dan *Block(Grid)*. *Thread* memiliki *memory* sendiri dan akan mengerjakan unit

pekerjaan yang kecil dan akan berjalan secara bersamaan dengan *thread* lain, sehingga waktu yang dibutuhkan untuk mengerjakan pekerjaan seluruhnya akan menjadi lebih singkat. *Thread-thread* tersebut dikelompokkan menjadi *block*, yaitu kumpulan *thread* yang memiliki satu *memory* yang dapat digunakan oleh setiap *thread* dalam *block* tersebut secara bersama-sama untuk media komunikasi antar thread tersebut yang dinamakan dengan *shared memory*. Setiap *block* tersebut akan dikelompokkan lagi menjadi sebuah *grid* yang merupakan kumpulan dari semua *block* yang digunakan dalam suatu komputasi.

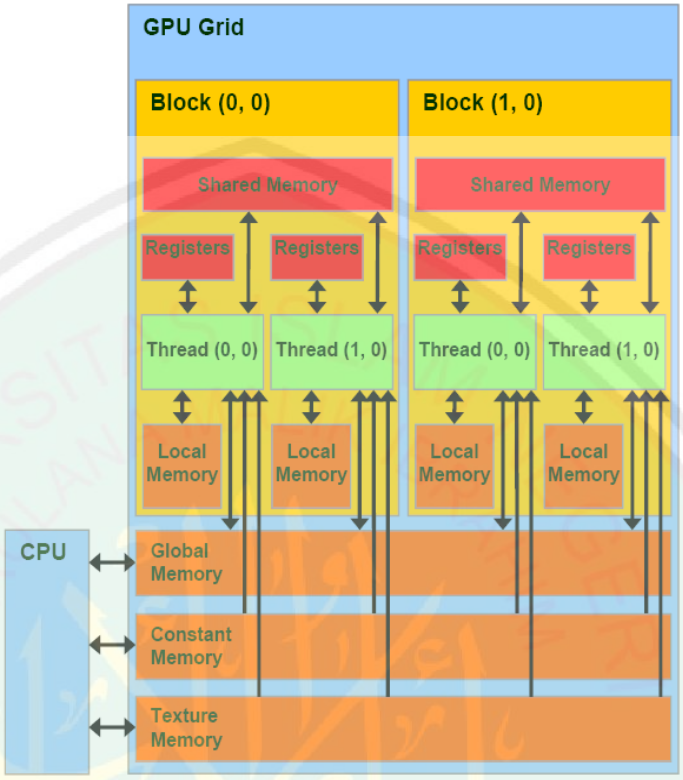
Dalam proses komputasim terdapat proses antrian data yang disebut sebagai *warp*.*Warp* memiliki efek pada kinerja GPU, sehingga jumlah *block* dan *thread* yang digunakan memiliki efek pada kecepatan proses komputasi



Gambar 2.9 Struktur Unit Pemroses pada CUDA**.**

Sumber :Doc.nvidia.com,2016.

CUDA memiliki beberapa macam *memory* yang dapat digunakan untuk proses komputasi yaitu; *Global Memory*, *Shared Memory*, *Texture Memory*, *Register*, *Local Memory*, dan *Constant Memory*.



Gambar 2.10 Struktur Memori pada CUDA

Sumber :Doc.nvidia.com,2016.

Cuda memiliki wilayah memory yang terbagi shared memory,local memory, global memory dan constant memory dengan aliran data sesuai pada mekanisme memory cuda seperti yang terlihat pada gambar(doc.nvidia.com).

2.8. OpenCL

OpenCL adalah kerangka kerja standar terbuka untuk tujuan umum pemrograman paralel pada platform heterogen. kerangka OpenCL termasuk bahasa berbasis C++ yang disediakan untuk menulis fungsi paralel disebut kernel, dan API runtime (antar muka pemrograman aplikasi) untuk mengendalikan platform dan device OpenCL. pemrograman OpenCL memungkinkan program untuk dijalankan pada CPU, GPU, prosesor Sel, DSP dan banyak Devices lainnya.

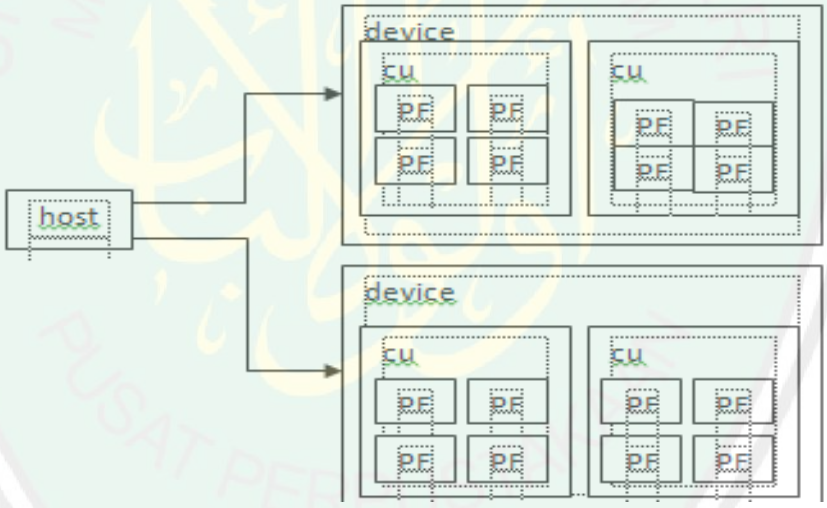
OpenCl terlahir pada tahun 2008 yang dikembang kan oleh Khronos kemudian selanjutnya dikembangkan oleh AMD.OpenCl yang merupakan cross-platform ,mampu berjalan pada hamper semua jenis perangkat berikut :

1. Intel
2. NVIDIA Corporation
3. Apple, Inc.
4. ARM Limited
5. QUALCOMM
6. Marvell
7. Vivante Corporation
8. MediaTek Inc
9. AMD
10. Texas Instruments
11. Altera Corporation
12. Xilinx, Inc.
13. Imagination Technologies
14. STMicroelectronics International NV
15. IBM Corporation
16. Creative Labs
17. Samsung Electronics

Di sini kita akan melihat dasar-dasar OpenCL dan penulis beberapa program sederhana untuk melakukan perhitungan pada GPU tersebut.OpenCL didasarkan pada model umum berikut:

1. Platform model
2. Memory model
3. Execution model
4. Programming model

Dalam platform model, setiap platform OpenCL (memberikan oleh beberapa vendor) terdiri dari sebuah host (CPU) yang terhubung ke satu atau lebih device komputasi.seperti pada gambar 5.Setiap device komputasi terdiri dari satu atau lebih unit-unit komputasi ,dan setiap unit komputasi terdiri dari beberapa elemen komputasi.Dan semua komputasi dikerjakan di elemen pemrosesan.



Gambar 2.11 OpenCL platform model .

Sumber*: Boreskov and Shikin,2014*

Untuk mengetahui semua platform yang tersedia gunakan perintah clGetPlatformIDs ,seperti source code berikut;

```
cl_intclGetPlatformIDs(cl_uint num_entries,
                       cl_platform_id *platforms,
                       cl_uint *num_platforms)
```

Listing 2.2. Perintah untuk mengetahui semua *platform* yang tersedia.

dan untuk mengetahui informasi platform gunakan perintah clGetPlatformInfo, seperti source code berikut;

```
cl_intclGetPlatformInfo(                cl_platform_id platform,
                                        cl_platform_info param_name,
                                        size_t param_value_size,
                                        void *param_value,
                                        size_t *param_value_size_ret)
```

Listing 2.3 Perintah untuk mengetahui informasi *platform*.

selain itu untuk mendapatkan device yang tersedia pada sebuah platform, gunakan perintah clGetDevicesID,untuk mengetahui info seputar device tersebut gunakan perintah clGetDevicesInfo seperti source code berikut;

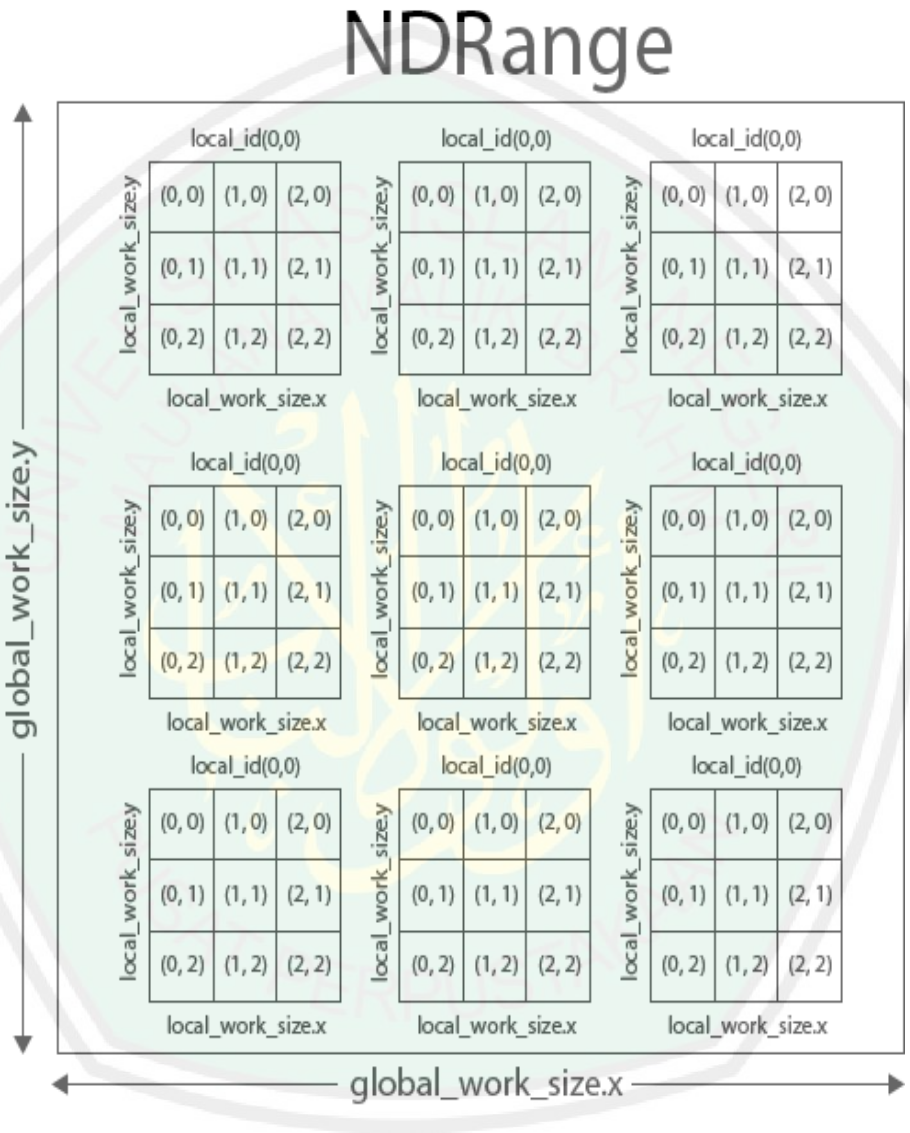
```
cl_intclGetDeviceIDs(                   cl_platform_id platform,
                                        cl_device_type device_type,
                                        cl_uint num_entries,
                                        cl_device_id *devices,
                                        cl_uint *num_devices

cl_intclGetDeviceInfo(                  cl_device_id device,
                                        cl_device_info param_name,
                                        size_t param_value_size,
                                        void *param_value,
                                        size_t *param_value_size_ret)
```

Listing 2.4 Perintah untuk mengetahui *device* yang tersedia dan informasi tentang *device* tersebut.

OpenCL menerapkan sebuah stream processing model.semua aplikasi OpenCL memuat kernel yang mana dijalankan pada device dan host yang menangani jalannya kernel pada device.

Kernel tersedia di setiap elemen dari domain komputasi yang berdimensi N (ND-range)(N=1,2,3).Setiap elemen pada domain tersebut dikenal dengan istilah

Work-item.setiap work-item menjalankan kernel yang sama.*work-item* dapat dianggap sebagai paralel thread atau multhi thread yang mengeksekusi kernel.



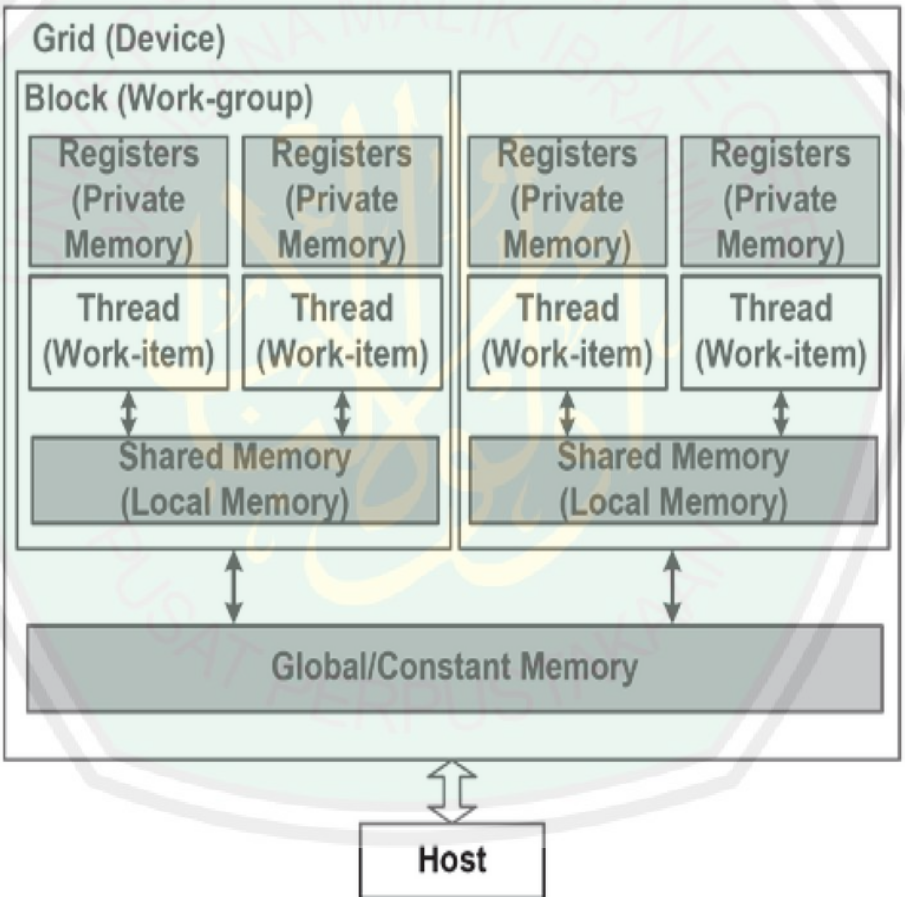
Gambar 2.12 Struktur ND-Range.

Sumber: Boreskov and Shikin,2014

Dari kumpulan *work-item* dikelompokkan menjadi *work-group.work-group* sendiri dapat berbentuk array 1/2/3 dimensi.hubungan dari beberapa work-item

yang termasuk dalam work-group yang sama terjadi melalui memori lokal dan sinkronisasi.

Dalam sebuah kernel OpenCL, empat daerah memori yang berbeda dapat diakses untuk *work-item*.arsitektur memori dan aliran data pada aplikasi OpenCL tergambarkan pada gambar berikut:



Gambar 2.13 Hierarki dari memory OpenCL.

Sumber : Doc.Khronos,2009.

1) memori global merupakan wilayah memori di mana data dapat dibaca / ditulis oleh semua *work-item*. Tidak mungkin untuk menyinkronkan *work-item* selama eksekusi kernel dalam memori ini.

2) memori Konstan adalah wilayah read-only memori global. Data di wilayah ini tidak berubah selama eksekusi.

3) memori lokal merupakan wilayah memori khusus untuk pekerjaan-kelompok. Kerja-item dalam kerja kelompok dapat berbagi data dalam memori lokal.

4) memori pribadi merupakan wilayah memori khusus untuk karya-item. Data dalam memori pribadi dari *work-item* tidak terlihat *work-item* lainnya. Pada sebagian besar

Program yang berjalan pada host hendaknya memiliki konteks . Konteks OpenCL dibuat dengan satu atau lebih *device*. Konteks digunakan oleh *runtime* OpenCL untuk mengelola objek seperti perintah-antrian, memori, program, dan benda-benda kernel dan untuk mengeksekusi kernel pada satu atau lebih perangkat yang ditentukan dalam konteks.adapun konteks terdiri dari sumber devices,kernel,program object,memory object, dan command queue.

OpenCL telah menyediakan perintah untuk menbuat konteks dengan memanggil fungsi clCreateContext. *Source code* yang ditulis untuk memangiil fungsi tersebut adalah memalui perintah berikut:

```
cl_context clCreateContext(        cl_context_properties *properties,
                                   cl_uint num_devices,
                                   const cl_device_id *devices,
                                   void *pfn_notify (

                                   const char *errinfo,
                                   const void *private_info,
                                   size_t cb,
                                   void *user_data),
                                   void *user_data,
                                   cl_int *errcode_ret)
```

Listing 2.5. Perintah untuk menbuat konteks

Perintah tersebut bertujuan untuk menciptakan context bagi satu atau lebih device yang ditunjukkan dari nilai berdasarkan numDevice dan parameternya. Parameter *notify* mengijinkan fungsi *callback* yang mana bertugas memanggil laporan error. Selain itucallback disini dapat dipanggil *asynchronously* (tidak serempak).

Host menggunakan *command-queue* untuk mengkontrol device.dan setiap device membutuhkan masing-masing *command-queue.* Beberapa perintah yang terdapat dalam *command-queue* ;

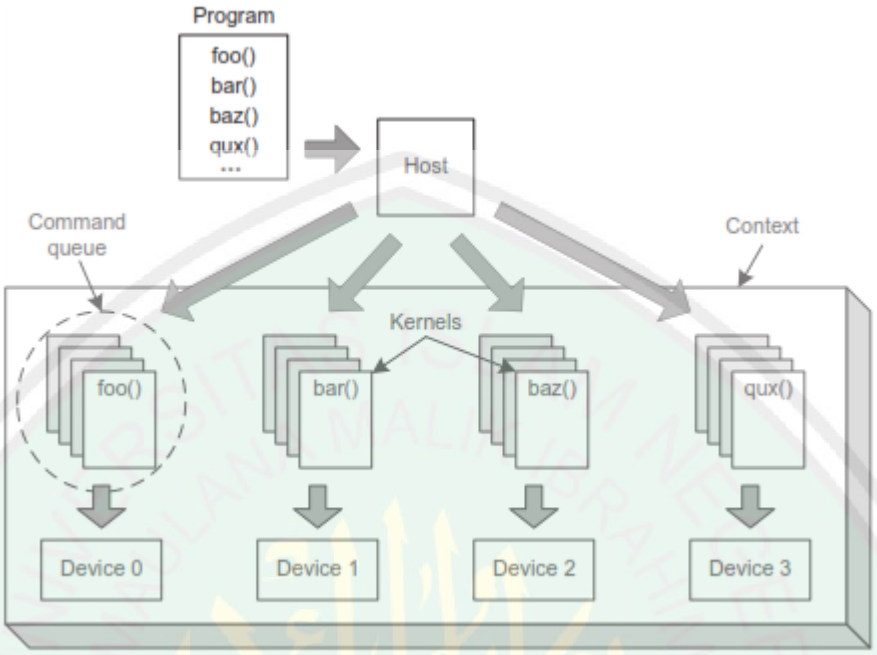
- Perintah untuk mengeksekusi kernel
- Perintah untuk memory
- Perintah untuk sinkronisasi

Adapun perintah untuk memanggil fungsi tersebut,tuliskan *source code* berikut:

```
cl_command_queueclCreateCommandQeue(
                                    cl_context context,
                                    cl_device_id device,
                                    cl_command_queue_properties properties,
                                    cl_int *errcode_ret)
```

Listing 2.6. Perintah untuk membuat *command queue.*

Fungsi OpenCL yang disampaikan kepada command-queue yang enqueued dalam urutan panggilan yang dibuat tetapi dapat dikonfigurasi untuk mengeksekusi *in-order* atau *out-of-order*. Sifat argumen dalam CreateCommandQueue dapat digunakan untuk menentukan urutan eksekusi..



Gambar 2.14 distribusi kernel pada saat OpenCL menjalankann program.

Sumber: Mathew scarpino,2012.

Seperti terlihat pada gambar 2.14 bahwa perintah di management dalam command queueuntuk dijadwalkan dieksekusi. Satuhost boleh memiliki beberapa *computation resource.*

BAB III

PERANCANGAN DAN IMPLEMENTASI

Pada bab ini akan diuraikan mengenai metode penelitian dan implementasi untuk membangunkomputasi sekuensial dan paralel pada metode *Alpha Trimmed Mean Filter*. Perancangan dilakukan untuk mengetahui tahap-tahap yang perlu ada untuk membuat sebuah program yang dibagung pada dua prosesor yang berbeda yaitu CPU dan GPU. Kemudian rancangan tahap tahap tersebut di terapkan pada model pemrograman serial menggunakan bahasa Java dan pararel menggunakan Bahasa OpenCL. Bab ini akan membahas mengenai deskripsi penelitian, data, simulasi metode, dan implementasi. Pembahasan tersebut diuraikan sebagai berikut:

3.1. Prosedur Penelitian

Prosedur penelitian dideklarasikan agar tahapan pengerjaan dilakukan secara berurutan. Langkah-langkah untuk implementasi metode *Alpha Trimmed Mean Filter* dengan komputasi paralel OpenCL dimulai dari pengumpulan data citra, pembuatan data citra uji dengan memberi *salt and pepper noise*,penerapan metode *Alpha Trimmed Mean Filter*, pencatatan waktu proses, hingga penyajian hasil perbandingan dari masing-masing gambar yang telah di tambahkan noise . Berikut ini rincian langkah prosedur penelitian.

3.1.1. Pengumpulan data citra

Data yang digunakan dalam penelitian ini berupa citra digital. Data yang dikumpulkan meliputi data citra RGB dengan format .jpg sejumlah 50gambar. Data citra didapatkan dari kamera *smartphoneHuawei honor 4c*. Data citra tersebut berukuran 3264×1836.Pengambilan data citra dibedakan menjadi dua lokasi yaitu dari foto luar ruangan(*outdoor*) dan foto dalam ruangan (*indoor*). Data *outdoor* yang digunakan adalah foto pemandangan pada jarak minimal 5 meter dan jarak maksimal mencapai 90 kilometer. Sedangkan yang digunakan sebagai data *indoor* adalah foto dalamruangan seperti di dalam aula, maupun ruang perkantoran. Sehingga diperkirakan jarak maksimal mencapai 50 meter.

3.1.2. Data Citra Uji

Data Citra Uji pada penelitian ini merupakan olah hasil dari data citra asli hasil kamera smarthphne huawi honor 4c. Data citra ujidiperoleh denganmemberi *noise* pada data citra asli. Adapun noise yang digunakan dalam penelitian ini yaitu *salt and pepper noise*.

Pemberian *noise*dilakukan menggunakan perangkat lunak Matlab R2007a. *Noise* yang ditambahkan meliputi *salt and peppernoise*dengan *noise density* sebesar 30% , 50% dan 80%. Sehingga didapatkan citra untuk disimpan sebagai data uji berjumlah 150 gambar. Rincian data citra uji adalah sebagai berikut :

1. 25 data citra indoor dengan noise density sebesar 30%
2. 25 data citra indoor dengan noise density sebesar 50%
3. 25 data citra indoor dengan noise density sebesar 80%

4. 25 data citra outdoor dengan noise density sebesar 30%
5. 25 data citra outdoor dengan noise density sebesar 50%
6. 25 data citra outdoor dengan noise density sebesar 80%

3.1.3. Perancangan metode *Alpha Trimmed Mean Filter*

Data yang digunakan adalah data kuantifikasi dengan ukuran data kuantitatif 20x20 element berikut tahapan proses simulasi alpha-trimmed mean.

Tabel 3.1 Hasil baca nilai pixel gambar uji

196	194	193	194	195	191	191	192	193	195	194	195	198	196	196	198	198	203	201	201
192	187	192	190	191	191	192	194	194	194	191	255	194	196	197	195	199	199	255	195
194	191	186	191	255	194	194	194	194	255	195	199	194	0	0	194	200	196	197	198
192	186	186	194	197	194	193	190	196	196	196	196	197	197	197	197	199	196	201	198
0	188	192	194	191	191	191	193	193	193	192	196	195	0	195	198	198	200	200	199
190	196	255	192	192	197	193	193	194	195	195	196	198	198	197	197	197	197	201	204
187	255	189	191	192	194	195	0	255	197	193	195	200	201	200	200	198	198	198	201
188	190	192	193	191	191	194	194	193	195	194	195	195	199	255	199	198	194	192	182
193	189	193	189	190	195	195	193	196	197	197	255	199	200	202	198	193	176	142	84
192	189	193	194	192	195	193	194	197	199	201	255	201	0	200	186	154	96	55	255
190	188	0	195	195	0	193	197	197	198	199	199	0	192	160	92	92	68	46	47
194	190	190	195	255	194	195	199	196	193	193	193	179	122	65	55	80	97	62	51
187	255	255	195	195	195	194	194	193	192	190	0	0	48	50	62	88	103	92	63
187	192	195	255	196	195	195	190	192	187	151	69	45	35	42	71	95	255	111	91
255	188	191	195	198	193	193	186	169	128	88	55	48	255	46	57	95	108	114	103
190	190	190	196	194	182	182	255	113	86	77	71	70	57	50	55	72	106	108	0
191	190	186	189	183	177	166	152	0	90	82	72	80	92	86	62	69	83	103	111
189	187	182	0	185	179	179	165	145	119	93	76	73	92	104	95	81	62	71	105
183	177	255	192	194	191	255	0	168	134	102	81	75	74	88	110	108	82	76	255
182	178	193	194	197	194	0	178	172	148	120	95	84	76	69	74	112	117	99	74

Perhitungan trim mean akan lebih mudah dengan membagi proses menjadi 2 tahap , Proses pemangkasan dan selanjutnya proses perhitungan nilai mean.seperti terilustrasikan pada contoh berikut:

196	194	193
192	187	192
194	191	186

Trimmed Mean Percent = 50/100 = 0.2 Sample Size=9

Untuk mendapat nilai T38rimmed Mean Menginisialkan nilai hitung pakai (g) , dimana g mengacu jumlah nilai yang akan dipangkas dari seri yang diberikan

g = Floor(Trimmed Mean Percent x Sample Size)

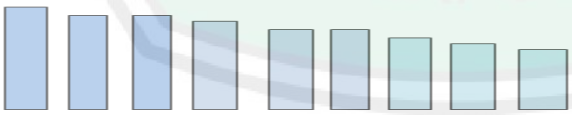
g = Floor(0.5 x 9)

g = Floor(4.5)

Trimmed count (g) = 4

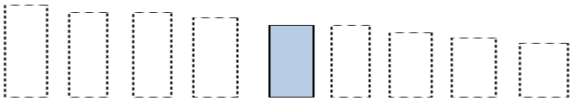
Kemudian pada kernel image dilakukan pengurutan untuk mendapat nilai tertingi sampai terendah ,hasil pengurutan nilai pixel terkecil hingga terbesar

186 187 191 192 192 193 194 194 196



hasil pemangkasan 50% didapatkan pemotongan element sebanyak 4 element dari nilai tertinggi dan. Jadi akan terlihat seperti berikut :

192



Sisa element pixel dari kernel image selanjutnya dilakukan penghitungan nilai rata-rata.setelah proses perhitungan selesai nilai pixel tengah dari kernel diganti dengan nilai trim mean yang diperoleh.selanjutnya , menghitung nilai mean dilambangkan sebagai μ.

$\mu = \Sigma Xi / n$

μ = (Sum of your Trimmed Set / Total Numbers in Trimmed set)

μ = 192/1

μ = 192

96	194	193
192	187	192
194	191	186

→

196	194	193
192	192	192
194	191	186

Penerapan metode yang dilakukan adalah penulisan *source code*ke dalam komputasi CPU dan GPU dengan metode *Alpha Trimmed Mean Filter*.Pada komputasi CPU, *source code* yang ditulis adalah menggunakan bahasa java.Sehingga proses yang dilakukan menunjukkan proses pemrograman sekuensial.

Sedangkan komputasi GPU, menggunakan*framework* yang mengeksekusi dengan *heterogenousplatform* (CPU dan GPU) yang dinamakan Opencl. Dalam Opencl tersebut, source code yang ditulis berekstensi .cl. Namun tidak hanya dengan file Opencl (.cl) tersebut, namun juga ada proses penulisan *source code* pada host. Dalam penelitian ini pemrograman pada host menggunakan bahasa java dengan library yang menawarkan *bindings* untuk OpenCL.Komputasi GPU

dapat membagi piksel menjadi beberapa kelompok untuk diproses secara bersamaan. Hal inilah yang menunjukkan proses pemrograman paralel.Pemrograman pararel pada OpenCL terbagi menjadi dua yaitu OpenCL kode kernel dan OpenCL kode Host.

Kernel dibangun oleh OpenCL runtime compiler . kernel ini menghitung blok histogram dari 256 * 16 nilai-nilai pixel,masing-masing untuk komponen R, G, dan B. Setiap work-item pertama menghitung dari 256 elemen ke dalam sharedArray memori lokal. Setelah setiap work-item melakukan komputasi yang dikonfirmasi oleh lokal barrier dalam kode kernel, loop terakhir di kernel terakumulasi semua nilai histogram menjadi tingkat blok histogram dari 256 * 16nilai piksel dari masing-masing komponen warna yang digunakan untuk proses alpha-trim mean filter.

UntukOpenCL Kode Host melibatkan langkah-langkah berikut:

1. membaca Gambar ke dalam buffer pixel baku: Dalam gambar BMP pixelnilai-nilai yang disimpan sebagai nilai-nilai disisipkan RGB pixel atau sebagai referensi untuktabel palet. pertama kali piksel gambar dibaca dikirim ke memori sistem. Untuk tujuan ini, dibuatlah objek gambar sederhana, yang menyimpan buffer danukuran gambar menggunakan fungsi utilitas.

void ReadBMPImage(string filename, Image **image)

2. Setup OpenCL Platform: Setelah kita mendapatkan nilai gambar baris pixel di memori system pada buffer yang bersebelahan, kami mendirikan OpenCL platform dan kemudian mengirim buffer ini ke device OpenCL. Menyiapkan device OpenCL melibatkan pememilih platform yang tersedia, memilih device

untuk menjalankan kernel,menciptakan konteks eksekusi dan antrian perintah yang terkait.

3. Membuat OpenCL Buffer: Untuk histogram perhitungan kita buat sebanyak empat OpenCL Buffer menggunakan API clCreateBuffer. Salah satunya adalah buffer inputyang memiliki nilai-nilai pixel baku. buffer ini perlu ditulis ke devicememori menggunakan clEnqueueWriteBuffer. Tiga sisanya adalah output buffer yang setelah perhitungan histogram perlu dibaca kembalike memori host menggunakan clEnqueueReadBuffer OpenCL runtime API. Setelah buffer dibuat selanjutnya menciptakan aplikasi perintah antrian untuk menulis data buffer yang ditunjuk oleh Image-> piksel ke dalam device yang terkait menggunakan*command queue*. Setiap perintah pada antrian dikaitkan dengan event handle.Hasil operasi dituliskan dalam *even writeEvt*untuk dikembalikan olehaplikasi. Aplikasi ini bisa menunggu event ini untuk menyelesaikan dan melanjutkandengan pengolahan lebih lanjut. Untuk menyelesaikan semua tugas dalam antrian menggunakan fungsi clFinish.

Ada cara lain untuk membuat buffer, yang tidak membutuhkan transferdata antara device. Buffer dibuat dengan CL_MEM_USE_HOST_PTR flag. Keduanya bisa digunakan sesuai kebutuhan .Buffer ditandai dengan flag CL_MEM_WRITE_ONLY. Input dan outputbuffer dicirikan oleh nilai-nilai masing–masing cl_mem_flags CL_MEM_READ_ONLYdan CL_MEM_WRITE_ONLY.

4. Membangun kernel: Kernel ditampilkan dalam potongan kode yang dikompilasi dan menentukan parameter kernel sesuai kebutuhan.

5. Pengaturan argumen kernel: Setelah kernel dibangun selanjutnya merealisasikan hubungan / korespondensi antara sisi host dari OpenCL objek buffer dan sisi kernel dari pointer memori global.

6. membaca Buffer ke memori host: Setelah setiap thread telah dihitung kemudian 256 elemen di kirimpada shared memori final dari perhitungan masing-masing proses dandisimpan ke dalam memori global. Selesainya eksekusi kernel, hasilnya kemudian dibaca kembali kememori host.

3.1.4. Pencatatan dan perhitungan waktu proses

Setelah proses penerapan metode *Alpha Trimmed Mean Filter*dan uji coba setiap gambar selesai, maka akan dinilai performapemrosesannya per uji coba.Adapun pencatatan waktu yang diambil untuk menilai komputasi sekuensial ataukah paralel yang paling cepat.Data waktu proses dicatat ke dalam sebuah tabel menurut macamnya. Tabel yang dibuat meliputi tabel data uji *indoor* dengan *salt and pepper noise*30%, 50%, dan 80% . Kemudiantabel data uji *outdoor* dengan *salt and pepper noise*30%,50%,80% .

3.1.5. Penyajian hasil

Untuk mempermudah dalam membaca data yang telah didapat, maka data harus dibuat diagram. Diagram yang disajikan digunakan untuk melihat perbandingan proses komputasi sekuensial dan paralel. Data tabel yang telah dibuat pada langkah sebelumnya digunakan sebagai acuan untuk membuat diagram.

3.2. Implementasi

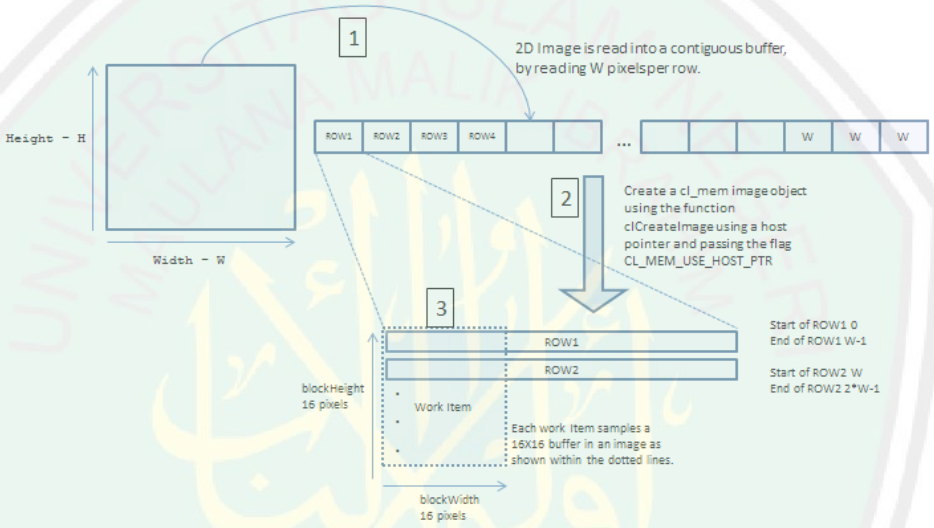
Pada subbab ini akan dijelaskan tentang implementasi metode *Alpha Trimmed Mean Filter* dengan komputasi sekuensial dan komputasi paralel. Sehingga dalam skripsi ini dibangun dua skenario dalam pemrosesannya, yaitu Skenario atau alur yang menerangkan tentang pemrosesan secara paralel (skenario 1) dan Skenario atau alur yang menerangkan tentang pemrosesan secara sekuensial (skenario 2). Rancangan uji coba data citra dengan *salt and pepper noise* pada penelitian kedua skenario tersebut dibangun agar diketahui perbedaan waktu proses (*consuming time*) pada masing-masing skenario.

3.2.1. *Management Thread* dalam OpenCL

Gambar dipetakan melalui *host pointer*. Gambar yang dikomputasi pada device harus tersedia di space alamat host, yang digunakan untuk menulis gambar kembali ke sistem file. Fungsi clEnqueueMapImage akan memetakan citra dari memori device ke space alamat host. Hal tersebut termasuk tugas yang diantrikan pada command_queue device

Gambar berwarna mengandung komponen RGB,sehingga proses dilakukan serentak pada setiap komponen RGB untuk penghapusan noise. Gambar input dibaca ke dalam buffer yang berdekatan dan objek gambar dibuat menggunakan fungsi clCreateImage. Di sisi kernel nilai piksel dapat ditampilkan menggunakan read_image OpenCL built-in. Input image dari sistem file dibaca ke dalam buffer yang berdekatan, caranya ditunjukkan seperti Langkah 1 dalam diagram. Gambar masukan dapat dari format apapun BMP, PNG, atau JPEG. *Buffer* baris dari pixel

gambar selanjutnya digunakan untuk membuat objek gambar OpenCL menggunakan fungsi clCreateImage. The CL_MEM_USE_HOST_PTR flag dilewatkan dulu karna ditampilkan pada langkah 2 dalam diagram. Akhirnya setiap kernel mengeksekusi pada gambar buffer seperti yang ditunjukkan oleh langkah 3.

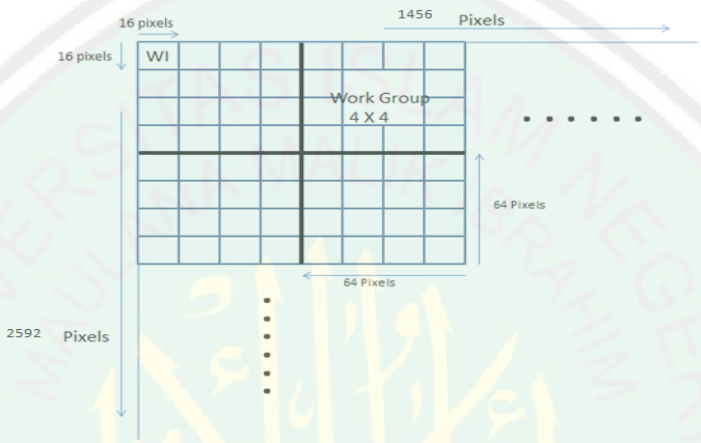


Gambar 3.1 Komputasi histogram RGB.
Sumber :Banger revishekhar,2013.

Pada kernel OpenCL kernel tersebut memproses 16 X 16 ukuran pixel gambar. Mari kita pertimbangkan gambar ukuran (2592,1456), maka NDRange dari kernel ini secara global (64,64), dan dimensi lokal work-group adalah (4,4). Alasan menggunakan (4,4) karena diharapkan mampu menjaga penggunaan memori lokal dalam batas yang diizinkan. Batas yang diperbolehkan untuk memori device lokal dapat diperoleh menggunakan fungsi clGetDeviceInfo dengan PARAM_NAME,CL_DEVICE_LOCAL_MEM_SIZE. histogram_kernel ini menggunakan memori lokal dengan ukuran :

3 * BIN_SIZE * groupSize * sizeof (cl_uchar);

Untuk objek groupSize 16 dan BIN_SIZE dari 256 sehingga dikalikan 3 untuk tiap komponen RGB menjadi total 12.288 bytes (12 K) dari memori lokal yang digunakan. Diagram berikut menunjukkan work-item dan pengolahannya pada gambar .

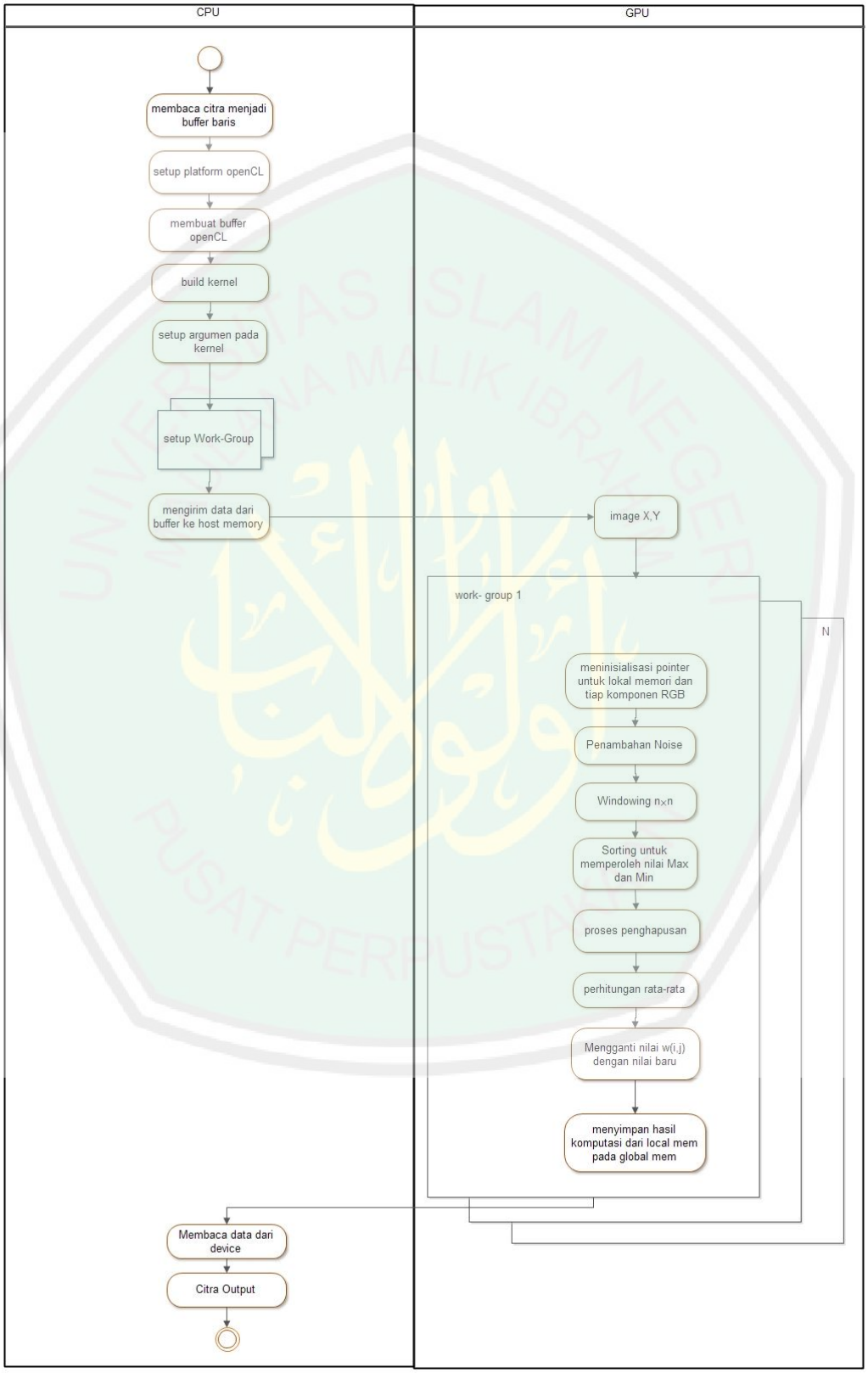


Gambar 3.2 Model eksekusi pada pemetaan data gambar.
Sumber : Banger revishekhar,2013.

Masing-masing work-item akan mengatur 256 elemen pada memori sharedArrayR, sharedArrayG, dan sharedArrayB ke nol. Kemudian kernel bisa melakukan proses komputasi mulai dari histogram RGB hingga penghapusan noise. Gambar dibaca menggunakanFungsi read_imageui. Fungsi ini mengambil sebagai masukan objek sampler_t,pixelkoordinat, dan gambar itu sendiri dari mana nilai-nilai pixel yang harus dibaca. Karena gambar adalah format CL_RGBA 4-channel, nilai kembali dari fungsi read_imageui adalah vektor unit4, yang berisi nilai-nilai RGB pixel(*Bangerrevishekhar*,2013).

3.2.2. Skenario 1 (GPU)

Skenario ini dijalankan pada GPU menggunakan framework OpenCL .



Gambar 3.3 Activity Diagram program secara umum yang berjalan

Berikut ini adalah tahapan proses skenario 1:

a. Ambil data gambar yang akan diolah. Dari gambar tersebut, diambil nilai pixelnya untuk dialokasikan ke dalam memori host buffer (CPU) dan diinisialisasi.Listing program untuk implementasinya dapat dilihat pada (listing 3.1).

```
BufferedImage input = new BufferedImage(sizeX, sizeY,
        BufferedImage.TYPE_INT_RGB);
```

Listing 3. 4 kode mengambil nilai gambar

b. Ambil informasi platform dan perangkat (device) yang tersedia. Perintah untuk mendapatkan informasi tersebut adalah dengan `clGetPlatformIDs`dan `clGetDeviceIDs`. Dari daftar perangkat tersebut diatur platform (OpenCL) kemudian dipilih tipe perangkat yang akan dijalankan.Listing program untuk implementasinya dapat dilihat pada (listing 3.2).

```
int jumlahDevicesArray[] = new int[1];
clGetDeviceIDs(platform, tipeDevice, 0, null, jumlahDevicesArray);
int jumlahDevices = jumlahDevicesArray[0];
cl_device_id devices[] = new cl_device_id[jumlahDevices];
clGetDeviceIDs(platform, tipeDevice, jumlahDevices, devices, null);
cl_device_id device = devices[indexDevice];
```

Listing 3. 5 Setting platform dan perangkat

c. Setelah penentuan perangkat, dibuatlah OpenCL context untuk perangkat tersebut. Sebuah *context* mendefinisikan seluruh lingkungan OpenCL, termasuk perangkat, obyek program, kernel OpenCL, hingga *memory object*. Sebuah konteks dapat dikaitkan dengan beberapa perangkat atau

dengan hanya satu perangkat. Antrian perintah dan kernel harus dari konteks OpenCL yang sama, tidak bisa dari konteks yang berbeda. Sebelum dapat membuat suatu konteks, dilakukan*queryruntime* OpenCL untuk menentukan vendor platform yang tersedia dalam sistem. Setelah vendor platform dipilih, inisialisasi pelaksanaan OpenCL untuk membuat konteks. Sebuah konteks dapat memiliki sejumlah perangkat terkait, yang dapat berupa CPU atau GPU atau keduanya. Konteks di OpenCL diinisialisasi menggunakan `clCreateContext`.Listing program untuk implementasinya dapat dilihat pada (listing 3.3).

```
cl_context context = clCreateContext(contextProperties, 1, new
                cl_device_id[]{device}, null, null, null);
```

Listing 3. 6 Setting context

d. Selanjutnya adalah membuat antrian perintah (*command queue*). Antrian perintah dibuat untuk setiap penggunaan perangkat OpenCL untuk eksekusi kernel. Sebuah Opencl kernel dapat memiliki beberapa antrian perintah untuk tugas yang berbeda dalam aplikasi. Sehingga dapat menjalankan tugas-tugas secara independen pada antrian perintah yang berbeda.Antrian perintah dapat dibuat dengan inisialisasi `cl_command_queue`.Untuk implementasinya dapat dilihat pada (listing3.4).

```
cl_command_queue cQueue = clCreateCommandQueue(context, device, 0,
null);
```

Listing 3. 7 Setting *command-queue*

Sebuah perintah merupakan sebuah transfer data, atau perintah eksekusi kernel atau hambatan (barrier) dalam antrian perintah. Host akan

mengantrikan perintah tersebut pada antrian perintah. Setiap perintah atau tugas dihubungkan dengan sebuah OpenCL event. Event ini dapat digunakan sebagai mekanisme sinkronisasi untuk mengkoordinasikan eksekusi antara host dan perangkat.

e. Membuat*memoryobject* pada perangkat. Adapun objek memori yang akan dibuat adalah objek *buffer (buffer object)*. Data uji citra sebagai *memoryobject* berukuran 512×512 px. Memory object yang dibuat meliputi input dan output. Objek input berisi objek gambar yang akan diproses dan objek output disediakan untuk menampung output yang sudah terproses. Untuk membuat memori objek digunakan fungsi `clCreateBuffer` dengan argumen meliputi *context* yang disediakan ; `cl_mem_flags` sebagai variabel untuk spesifikasi tipe dari memori ; ukuran buffer yang dialokasikan dalam bytes ; pointer untuk menempatkan data dari/ke host (pointer ini ukurannya harus sama atau lebih besar dari ukuran yang dialokasikan) ; kode error jika ada. Listing program untuk implementasinya dapat dilihat pada (listing 3.5).

```
//membuat memory object untuk gambar input dan output
DataBufferInt        dataBufferSrc        =        (DataBufferInt)
        src.getRaster().getDataBuffer();
int dataSrc[] = dataBufferSrc.getData();
inputImageMem    =    clCreateBuffer(context,    CL_MEM_READ_ONLY    |
        CL_MEM_USE_HOST_PTR,   dataSrc.length   *   Sizeof.cl_uint,
        Pointer.to(dataSrc), null);
outputImageMem    =    clCreateBuffer(context,    CL_MEM_WRITE_ONLY,
        imageSizeX * imageSizeY * Sizeof.cl_uint, null, null);
```

Listing 3. 8 membuat memory object (input & output)

f. Selanjutnya yaitu membuat perintah-perintah OpenCL (Opencl kode kernel). Pada kode kernel berisi perintah untuk menjalankan filter *alpha trimmed mean*, Kemudian dilakukan pengurutkan element dari data window

```
int totalNumber = windowSize*windowSize;
    for(int i=0; i< totalNumber;++i)
    {
        for(int j=0; j< totalNumber-1 ; ++j) {
            if(oldPixels[j] > oldPixels[j+1]){
            tmp = oldPixels[j];
            oldPixels[j] = oldPixels[j+1];
            oldPixels[j+1] = tmp;
            }}
```

Listing 3.6 Source code untuk mengurutkan element dari data window gambar

Sleanjutnya membuang elements terdepan dan terbelakang dari data window yang telah diurutkan.Source code membuang elements terdepan dan terbelakang dari data window yang telah diurutkan.

```
if(numb==0)
          res=sum;
        for(int u=0;u<(alpha/2);u++)
            oldPixels[u]=-1;
        for(int v=totalNumber-1;v>totalNumber-1-
(alpha/2);
            oldPixels[v]=-1;
        for(int l=0;l<totalNumber;l++)
        {
            if(oldPixels[l]!=-1)
          sum=sum+oldPixels[l]; }
```

Listing 3.7 Source code untuk ptoses trim(pemotongan)

Selanjutnya melakukan perhitungan mean dari sisa elemnt yang telah di lakukan pembuangan.Source code menlakukan perhitungan mean dari sisa elemnt yang telah di lakukan pembuangan seperti pada listing 3.8

```
res=sum/(totalNumber-alpha);
```

Listing 3.8 Source code untuk ptoses perhitungan mean.

g. Source kernel yang sudah dibuat dijadikan objek program dengan menggunakan fungsi `clCreateProgramWithSource`. Objek program dibuat untuk device yang berkaitan dengan *context* Opencl. Setelah objek program baru diciptakan untuk sebuah *context*,baik menggunakan biner atau *source* OpenCL, langkah berikutnya adalah membangun program. Program kernel perlu untuk dibangun dan dihubungkan pada saat runtime. Untuk membangun program, fungsi yang digunakan yaitu `clBuildProgram`. Tahap pembangunan (*build*) ini melibatkan kompilasi source code karena program dibuat menggunakan fungsi `clCreateProgramWithSource`. Jika program dibuat menggunakan fungsi `clCreateProgramWithBinary` maka hanya langkah menghubungkan runtime yang dijalankan. listing program untuk implementasinya dapat dilihat pada (listing 3.9).

```
cl_program program = clCreateProgramWithSource(context, 1, new
        String[]{sumber}, null, null);
String compileOption = "-cl-mad-enable";
clBuildProgram(program, 0, null, compileOption, null, null);
```

Listing 3. 9Source code untuk *build*

h. Setelah proses *building* program, tahap selanjutnya adalah membuat objek kernel dengan fungsi `clCreateKernel`.Setiap program adalah kumpulan kernel. Objek program dapat dikatakan sebagai *library* kernel-kernel. Sebuah kernel ketika diantrekan pada antrian perintah, *runtime* OpenCL menghasilkan biner untuk eksekusi pada perangkat. Jika kernel lebih dari satu, maka setiap kernel dijalankan pada perangkat yang berbeda. Dalam hal ini kernel yang dibuat hanya satu kernel. Sebuah objek kernel adalah enkapsulasi untuk satu kesatuan yang dieksekusi secara paralel. Objek kernel digunakan sebagai jalan untuk melewatkan argumen menggunakan API `clSetKernelArg`, sebelum *running* kernel menggunakan API `clEnqueueNDRangeKernel`.

```
clKernel = clCreateKernel(program, "MedianFilter", null);
```

Listing 3. 10 Source code membuat ojek kernel

i. Setelah membuat objek kernel, selanjutnya adalah mengatur argumen kernel sebelum pengeksekusian kernel. Argumen kernel yang diatur meliputi objek gambar input dan output serta ukuran gambar diberi pointer ke *device* dari host dengan menggunakan fungsi `clSetKernelArg`. Adapun argumen yang ada pada fungsi `clSetKernelArg` meliputi objek kernel yang argumennya akan diatur ; index dari argumen dimulai dari indeks ke 0 hingga akhir argumen ; spesifikasi ukuran dari `arg_value` ; `arg_value`sebagai pointer untuk data. Listing program untuk implementasinya dapat dilihat pada (listing 3.10)

```
//menentukan work group dan argumen pada opencl kernel code
long localWorkSize[] = new long[2];
localWorkSize[0] = 5;
localWorkSize[1] = 5;
long globalWorkSize[] = new long[2];
globalWorkSize[0] = round(localWorkSize[0], imageSizeX);
globalWorkSize[1] = round(localWorkSize[1], imageSizeY);
int imageSize[] = new int[]{imageSizeX, imageSizeY};
clSetKernelArg(clKernel, 0, Sizeof.cl_mem,
        Pointer.to(inputImageMem));
clSetKernelArg(clKernel, 1, Sizeof.cl_mem,
        Pointer.to(outputImageMem));
clSetKernelArg(clKernel, 2, Sizeof.cl_int2,
        Pointer.to(imageSize));
```

Listing 3. 10 Source code mengatur argumen kernel

j. Objek cl_kernel yang telah dibuat dan telah diatur argumen, kemudian dilakukaneksekusi pada perangkat. Untuk eksekusi objek kernel, digunakan fungsi `clEnqueueNDRangeKernel`dimana fungsi tersebut untuk menyebarkan opencl kernel ke perangkat dan mendefinisikan berapa banyak work item yang dibuat untuk mengeksekusi kernel (global_work_size) dan jumlah work item dalam setiap work group (local_work_size). Adapun fungsi `clEnqueueNDRangeKernel`meliputi `command_queue` ; objek Opencl Kernel ; dimensi dari NDRange ; `global_work_offset` yang juga digunakan untuk menghitung global_id dari work item, jika argumen ini bernilai null, maka nilai default adalah 0 ; `global_work_size` yang mendefinisikan global work

item dalam setiap dimensi ; `local_work_size` untuk mndefinisikan local work item dalam setiap dimensi ; `event_wait_list` ; `event`.

Selanjutnya proses yang dilakukan adalah membaca kembali memori dari device (GPU) ke host buffer (CPU) menggunakan fungsi `clEnqueueReadBuffer` dengan argumen yang berisi `command_queue` ; objek buffer `cl_mem` yang akan dibaca dan akan ditulis pada pointer ; `blocking_read; offset` ; total bytes yang dibaca dari device yang dipointerkan oleh buffer ; host memory pointer dari dimana data dibaca ; `event_wait_list` ; `event`. Buffer yang dibaca dari perangkat ke host adalah memori buffer untuk output dari memori input yang sudah diproses pada kernel code.

```
clEnqueueNDRangeKernel(queue, clKernel, 2, null, globalWorkSize,
        localWorkSize, 0, null, null);
DataBufferInt dataBufferDest = (DataBufferInt)
        dest.getRaster().getDataBuffer();
int dataDest[] = dataBufferDest.getData();
clEnqueueReadBuffer(queue, outputImageMem, CL_TRUE, 0,
        dataDest.length * Sizeof.cl_uint, Pointer.to(dataDest), 0,
        null, null);
```

Listing 3. 9 Source codemembaca memori dari device ke

k. Kemudian bersihkan objek-objek seperti memori, perintah, dan context pada OpenCL dan host buffer yang sudah dialokasikan di awal.Setiap objek program seharusnya dilepas (*released*) dari penyimpanan Opencl setelah digunakan. Untuk melepas objek program, fungsi yang digunakan adalah

`clRetainProgram`.Listing program untuk implementasinya dapat dilihat pada (listing 3.11).

```
clRetainProgram(program);
```

Listing 3. 10Source code membersihkan memori objek program

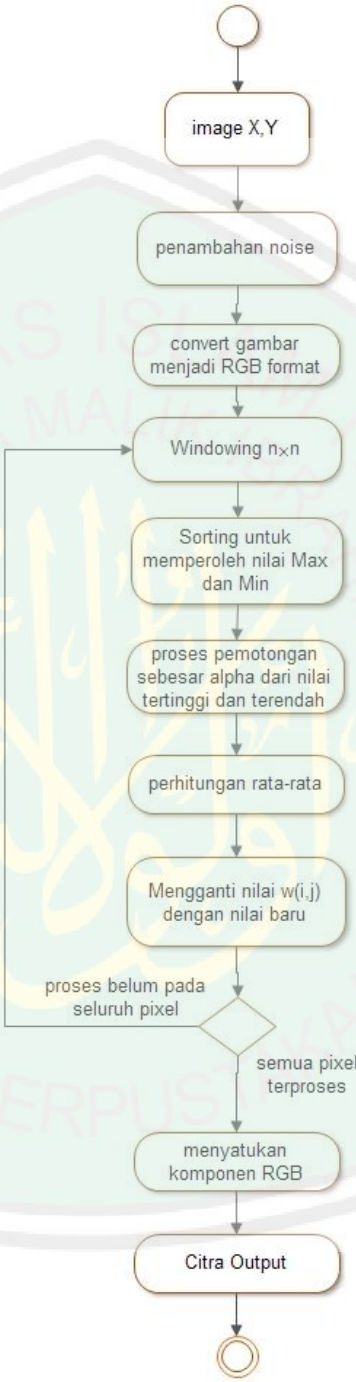
Sama halnya dengan objek program, objek kernel juga seharusnya dilepas (*released*) setelah proses selesai. Berbeda dengan objek program, objek kernel menggunakan fungsi `clReleaseKernel`. Adapun untuk *release command queue* menggunakan fungsi `clReleaseCommandQueue`dan untuk *release context* menggunakan fungsi `clReleaseContext`.

```
void shutdown() {
    clReleaseKernel(clKernel);
    clReleaseCommandQueue(queue);
    clReleaseContext(context);
}
```

Listing 3. 11Source code membersihkan memori untuk objek kernel, command-queue, dan context

3.2.3. Skenario 2 (CPU)

Pada skenario yang kedua terlihat pada Gambar 3.20, seluruh proses filtering dilakukan pada CPU. Proses ini diawali dengan mendapatkan nilai RGB pada citra masukan. Selanjutnya dilakukan proses windowing dan proses *Alpha Trimmed Mean Filter*. Dari hasil filtering, didapatkan citra keluaran dari proses pada CPU.Berikut ini adalah tahap-tahap implementasi metode *Alpha Trimmed Mean Filter*dengan pemrograman sekuensial menggunakan CPU:



Gambar 3.4 Activity Diagram program secara umum yang berjalan pada CPU

Cara kerja metode alpha trimmed mean filter terangkum menjadi 5 tahapan proses untuk pertama dilakukan proses windowing sebesar n x n.Proses windowing

dilakukan sebanyak pixel dalam gambar dengan merubah index melalui pergeseran.Selanjutnya pengambilan nilai dari windowing yang sudah dilakukan.Langkah ke tiga yaitu mengurutkan element dalam window berukuran n x n tersebut.karena nilai tengah dari element tersebut akan dijadikan acuan selanjutnya untuk penghapusan atau proses trim sebesar nilai alpha.Untuk melakukan proses trim atau pembuangan element dari elemen tertinggi dan Setelah semua proses selesai maka sisa hasil pembuangan tersebut dilakukan perhitungan nilai rata-rata. Berikut listing proses utama dalam Alpha trimmed mean

```
Arrays.sort(supp);
        for(int u=0;u<(alpha/2);u++)//0,1
            supp[u]=-1;
        for(int v=number-1;v>number-1-(alpha/2);v--)
            supp[v]=-1;
        for(int l=0;l<number;l++)
        {
            if(supp[l]!=-1)
                sum=sum+supp[l];
        }
        if(sum==0)
            return 0;
        else
return (float)(sum/(number-alpha));
```

Listing 3.13 Source code proses alpha trimmed mean

BAB IV

UJI COBA DAN PEMBAHASAN

Pada bab ini akan diuraikan mengenai hasil uji coba penelitian untuk dalam implementasi *Alpha Trimmed Mean Filter* pada komputasi paralel menggunakan OpenCL .

4.1. Data Uji

Data uji pada penelitian ini yaitu data citra digital yang berukuran 3264×1836. Macam data yang diujikan dibedakan menjadi data citra indoor yang diberi *salt and peppernoise*dengan *noise density* sebesar 30% , 50% dan 80%. berjumlah 25 gambar; citra outdoor dengan*salt and peppernoise*dengan *noise density* sebesar 30% , 50% dan 80%. berjumlah 25 gambar. Data citra uji dapat dilihat pada lampiran.

4.2. Langkah Pengujian

Implementasi metode *Alpha Trimmed Mean Filter*yang sudah dilakukan, diuji coba untuk didapatkan data efisiensinya. Data efisiensi didapatkan karena adanya perbandingan antara pemrosesan sekuensial dan paralel. Sehingga pengujian dilakukan pada CPU dan GPU. Pengujian dilakukan pada setiap data uji seperti yang dijelaskan pada (subbab 4.1). Adapun langkah pengujian implementasi metode *Alpha Trimmed Mean Filter* menggunakan komputasi

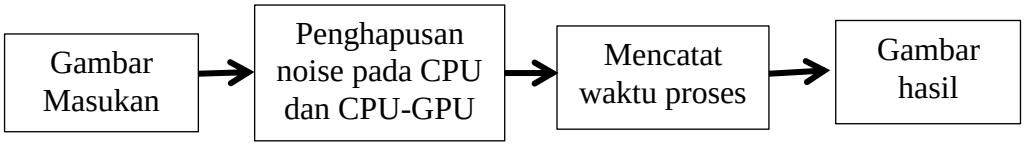
sekuensial dan paralel untuk penghapusan *noise*. Langkah pengujian secara ringkas dapat dilihat pada (gambar 4.1).

Alur pengujian untuk pemrosesan sekuensial dan pararel sama. Pada masing-masing proses dimulai dari mengambil citra masukan berupa data uji indoor dan outdoor masing-masing sebanyak 25 dengan penambahan noise salt and pepper sebanyak 30%,50%,dan 80%. Kemudian selanjutnya adalah proses penghapusan noise (*denoise*) dengan metode *alpha trimmed mean*pada dua model pemrograman yaitu sekuensial dan pararel yang menggunakan platform java dan OpenCL. Selama pemrosesanpenghapusan noise waktu dijalankan untuk menghasilkan data waktu. Data waktu dicatat dalam satuan *millisecond* (ms).

```
before = System.nanoTime();
bufferGbrOutputCL = jop.filter(bufferGbrInput,
bufferGbrOutputCL);
after = System.nanoTime();
 //diambil durasi waktu selama filter dengan milisecond
durationMSgpu = (after - before) / 1e6;
pesan = "GPU: " + String.format("%.2f", durationMSgpu)+" ms";
```

Listing 4.1 Pencatatan waktu yang dibutuhkan untuk filterSetelah proses

Penghapusan*noise* akan diperoleh data citra yang sudah terproses yang menjadi data hasil atau data output dari proses filter *Alpha trimmed mean*. Langkah pengujiuan dilakukan pada data keseluruhan sebanyak 150 data gambar.



Gambar 4. 1. Langkah pengujian

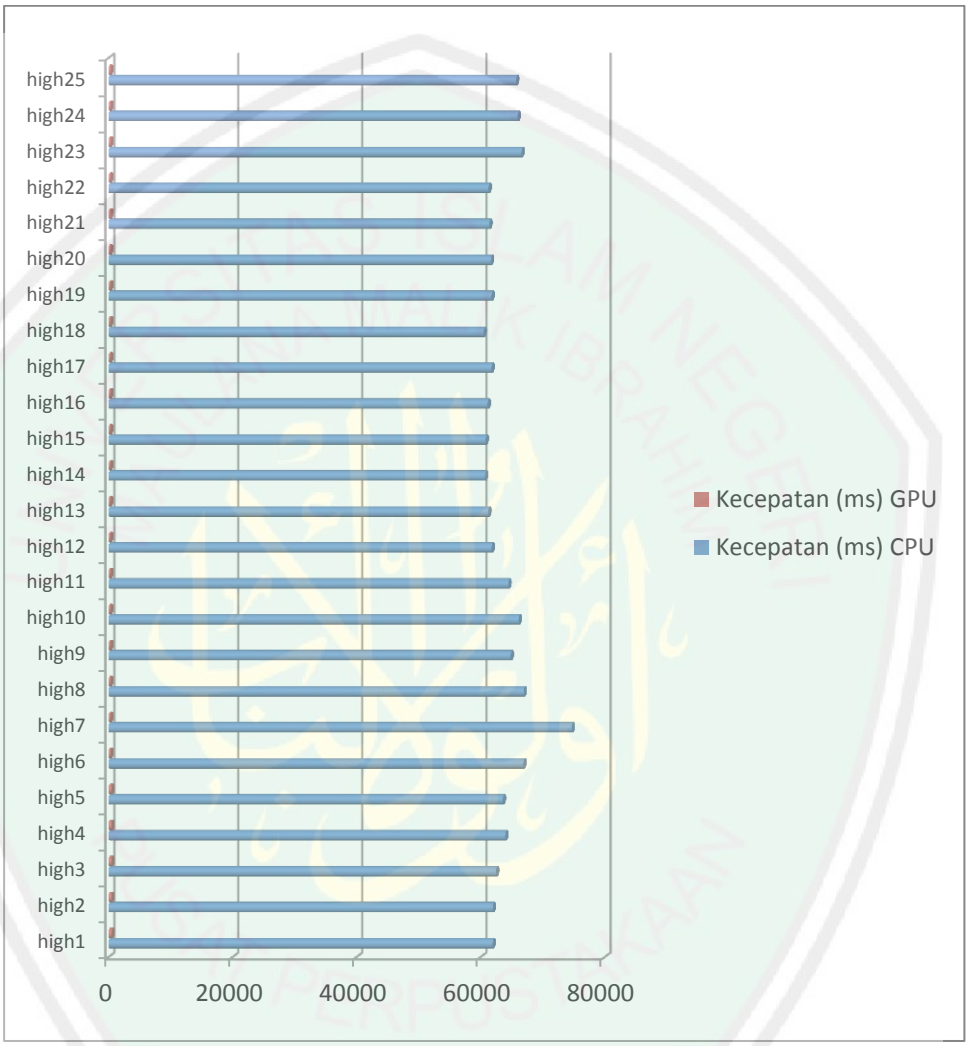
4.3. Hasil Uji Cobadan Analisa

Berdasarkan pengujian yang telah dilakukan pada data yang sudah disiapkan. Data yang disiapkan untuk diujikan yaitu data gambar indoor dan data gambar outdoor dengan noise salt and pepper sebesar 30%,50%, dan 80% didapatkan hasil pemrosesan komputasi sekuensial dan paralel sebagai berikut.

Table 4.1 Hasil ujicoba data outdoor yang telah ditambahkan noise salt and pepper sebesar 80% .

No.	Nama Data	Kecepatan (ms)	
		CPU	GPU
1	high1	61970.1	497.12
2	high2	61996.8	470.63
3	high3	62531.5	469.04
4	high4	64007	470.27
5	high5	63627.8	471
6	high6	66927.9	308.35
7	high7	74720.5	296.75
8	high8	66970.2	289.95
9	high9	64886.4	284.05
10	high10	66166.2	287.7
11	high11	64521.5	289.73
12	high12	61814.9	285.93
13	high13	61289.7	282.41
14	high14	60749.8	278.06
15	high15	60849.8	279.12
16	high16	61179.6	282.66
17	high17	61794.2	279.07
18	high18	60520.4	287.3
19	high19	61796.6	286.55
20	high20	61692.5	283.03
21	high21	61450.8	284.69
22	high22	61330.4	281.49
23	high23	66624.1	298.59
24	high24	65997.9	294.27
25	high25	65772.2	285.12

Untuk hasil ujicoba yang berupa waktu pemrosesan ditampilkan dalam bentuk diagram colom seperti berikut.



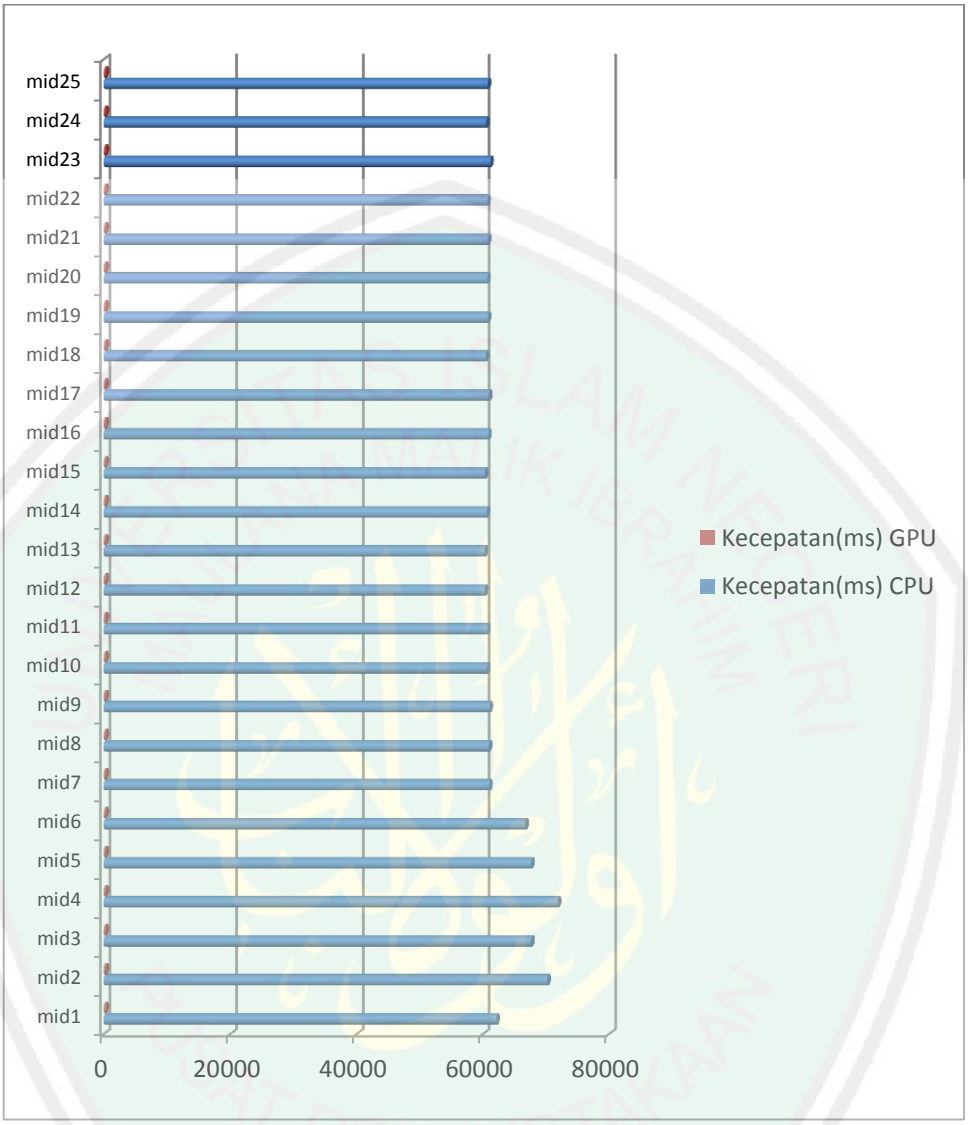
Gambar 4.2. Grafik waktu pemrosesan yang dibutuhkan untuk penghapusan noise salt & pepper sebesar 80% dengan metode *alpha trimmed mean* pada gambar outdoor.

Dari Informasi pada table 4.1 dan Grafik 4.2 , hasil ujicoba gambar indoor yang mengandung noise salt and pepper sebesar 80% diperoleh rata-rata waktu pemrosesan pada komputasi yang dijalankan murni pada CPU menggunakan platform java sebesar 63647.55 ms dan yang berjalan pada CPU-GPU menggunakan platform OpenCL sebesar 324.9152 ms.

Table 4.2 Hasil ujicoba data outdoor yang telah ditambahkan noise salt and pepper sebesar 50%.

No.	Nama Data	Kecepatan(ms)	
		CPU	GPU
1	mid1	62136.8	286.73
2	mid2	70220.8	293.19
3	mid3	67667.6	288.15
4	mid4	71890	291.9
5	mid5	67653.3	288.73
6	mid6	66748.8	288.8
7	mid7	61015.1	283.12
8	mid8	60954.7	287.25
9	mid9	61051.4	286.5
10	mid10	60683.6	279.27
11	mid11	60702.4	277.83
12	mid12	60294.6	287.53
13	mid13	60278.3	285.62
14	mid14	60591.2	286.95
15	mid15	60304.2	278.69
16	mid16	60856.5	296.02
17	mid17	60974.2	294.22
18	mid18	60486.6	282.92
19	mid19	60779.8	286.04
20	mid20	60707.1	282.05
21	mid21	60806.7	286.88
22	mid22	60718.7	287.63
23	mid23	61177.9	281.57
24	mid24	60526.7	285.51
25	mid25	60779.6	289.54

Untuk hasil ujicoba yang berupa waktu pemrosesan tidak hanya ditampilkan dalam bentuk table tapi juga ditampilkan dalam bentuk diagram colom untuk lebih mudah dipahami.



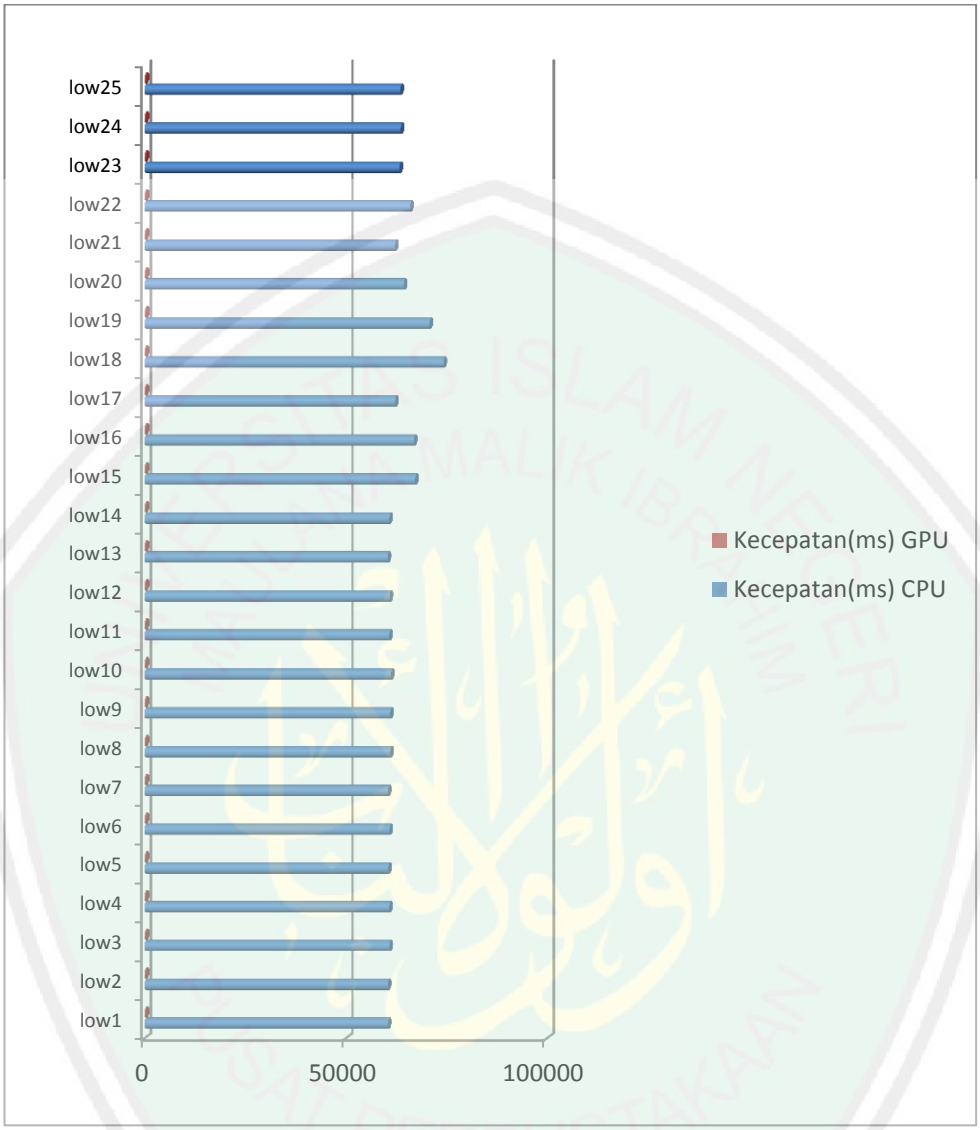
Gambar 4.3. Grafik waktu pemrosesan yang dibutuhkan untuk penghapusan noise salt & pepper sebesar 50% dengan metode *alpha trimmed mean* pada gambar outdoor.

` Dari Informasi pada table 4.2 dan Grafik 4.3, hasil ujicoba gambar indoor yang mengandung noise salt and pepper sebesar 50% diperoleh rata-rata waktu pemrosesan pada komputasi yang dijalankan murni pada CPU menggunakan platform java sebesar 62400.26 ms dan yang berjalan pada CPU-GPU menggunakan platform OpenCL sebesar 286.5056 ms. Waktu yang diperlukan lebih sedikit dibandingkan pada noise 80%.

Table 4.3 Hasil ujicoba data outdoor yang telah ditambahkan noise salt and pepper sebesar 30% .

No.	Nama Data	Kecepatan(ms)	
		CPU	GPU
1	low1	60434.3	290.53
2	low2	60445	285.87
3	low3	60833.7	296.14
4	low4	60771.5	285.16
5	low5	60543.7	285.98
6	low6	60830	278.56
7	low7	60453.5	282.62
8	low8	61046.8	288.99
9	low9	61018.5	282.63
10	low10	61218.8	282.01
11	low11	60781.7	283.76
12	low12	60882.1	282.43
13	low13	60397.4	283.57
14	low14	60784	278.67
15	low15	67210.7	302.7
16	low16	66917	285.92
17	low17	62249.5	279.49
18	low18	74266.3	288.02
19	low19	70795.4	360.46
20	low20	64413.7	288.67
21	low21	62218	288.78
22	low22	65979.6	293.29
23	low23	63353.2	280.39
24	low24	63663.4	287.89
25	low25	63565.3	291.89

Untuk hasil ujicoba yang berupa waktu pemrosesan tidak hanya ditampilkan dalam bentuk table tapi juga ditampilkan dalam bentuk diagram colom untuk lebih mudah dipahami.



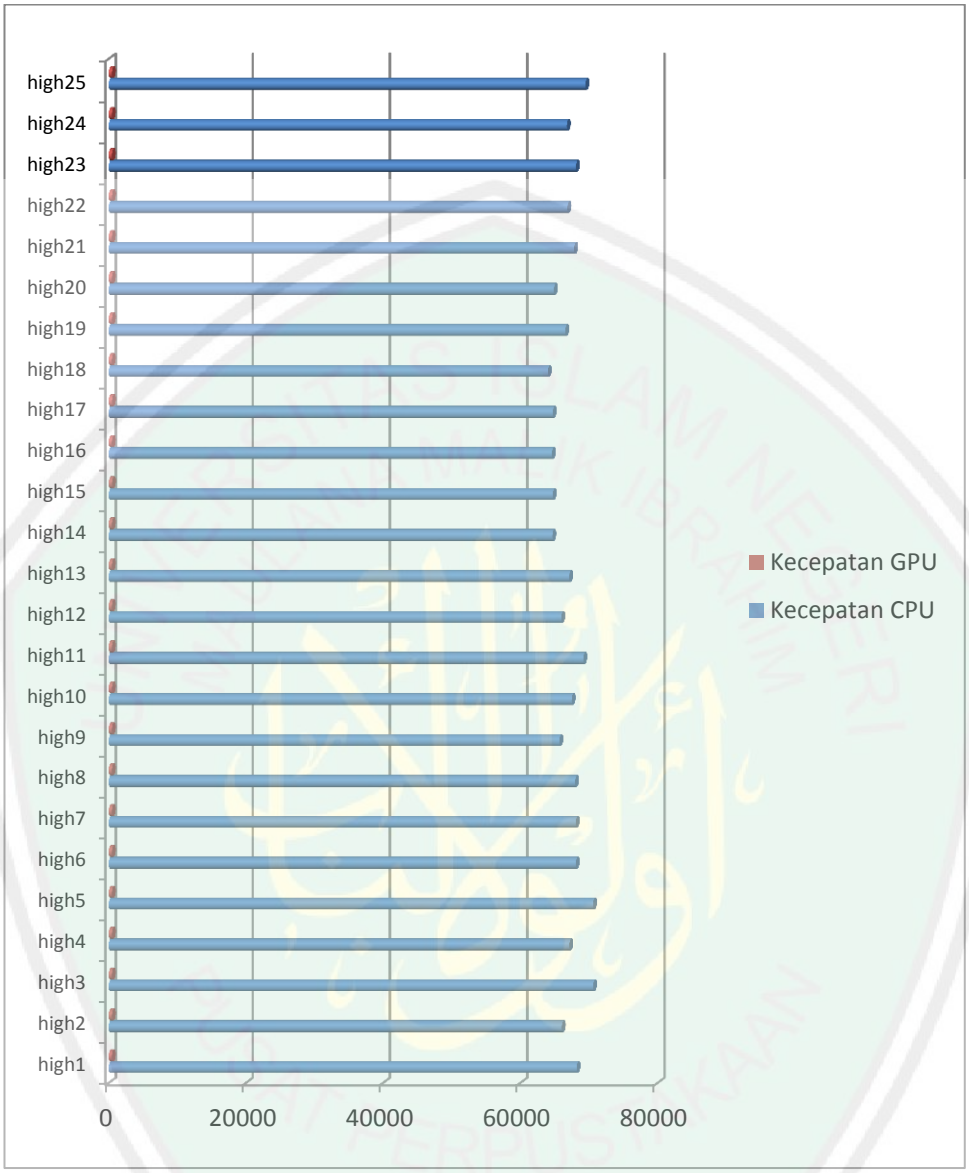
Gambar 4.4 Grafik waktu pemrosesan yang dibutuhkan untuk penghapusan noise salt & pepper sebesar 30% dengan metode *alpha trimmed mean* pada gambar outdoor.

Dari Informasi pada table 4.3 dan Grafik 4.4 , hasil ujicoba gambar indoor yang mengandung noise salt and pepper sebesar 30% diperoleh rata-rata waktu pemrosesan pada komputasi yang dijalankan murni pada CPU menggunakan platform java sebesar 63002.92ms dan yang berjalan pada CPU-GPU menggunakan platform OpenCL sebesar 289.3768 ms. Waktu yang diperlukan lebih sedikit dibandingkan pada noise 80% dan 50%.

Table 4.4 Hasil ujicoba data indoor yang telah ditambahkan noise salt and pepper sebesar 80% .

No.	Nama Data	Kecepatan	
		CPU	GPU
1	high1	68212.9	469.94
2	high2	66066.75	472.32
3	high3	70621.5	469.15
4	high4	67166.7	475.62
5	high5	70621.53	469.15
6	high6	68094.5	470.88
7	high7	68097.2	468.82
8	high8	68006.9	468.93
9	high9	65709.2	469.52
10	high10	67502.4	471.67
11	high11	69198.9	469.85
12	high12	65995.7	468.91
13	high13	67148.8	469.47
14	high14	64687	469.09
15	high15	64742.8	483.72
16	high16	64622.6	463.52
17	high17	64742.8	483.72
18	high18	64021.8	466.69
19	high19	66572.9	475.35
20	high20	64913.8	469.84
21	high21	67854.7	468.38
22	high22	66894.4	485.91
23	high23	68124.5	468.52
24	high24	66790.9	468.79
25	high25	69502.6	471.58

Untuk hasil ujicoba yang berupa waktu pemrosesan ditampilkan dalam bentuk diagram colom seperti berikut.



Gambar 4.5 Grafik waktu pemrosesan yang dibutuhkan untuk penghapusan noise salt & pepper sebesar 80% dengan metode *alpha trimmed mean* pada gambar Indoor.

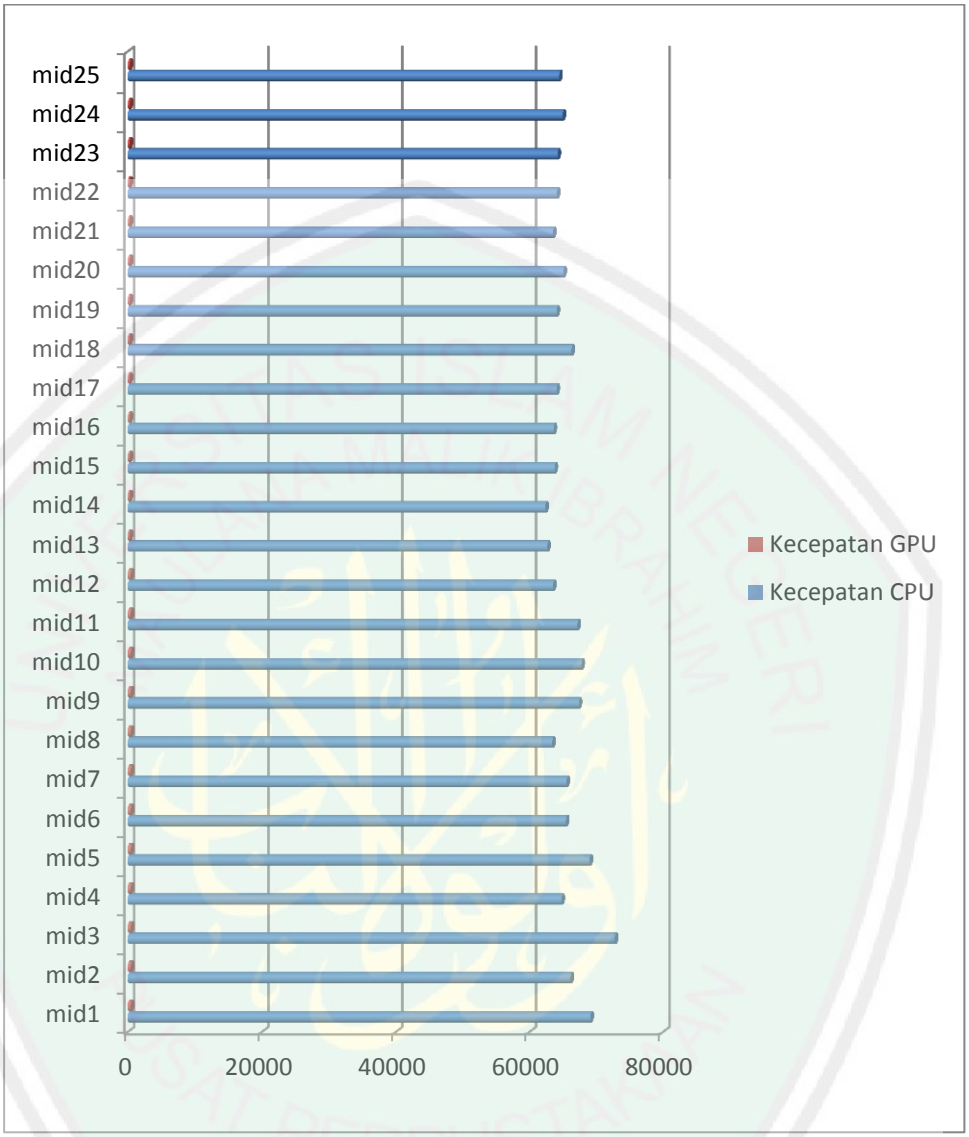
Dari Informasi pada table 4.4 dan Grafik 4.5 , hasil ujicoba gambar outdoor yang mengandung noise salt and pepper sebesar 30% diperoleh rata-rata waktu pemrosesan pada komputasi yang dijalankan murni pada CPU menggunakan platform java sebesar 67036.54 ms dan yang berjalan pada CPU-GPU

menggunakan platform OpenCL sebesar 471.5736 ms. Waktu yang diperlukan lebih sedikit dibandingkan pada noise 80%.

Table 4.5 Hasil ujicoba data outdoor yang telah ditambahkan noise salt and pepper sebesar 50% .

No.	Nama Data	Kecepatan	
		CPU	GPU
1	mid1	69142.78	473.38
2	mid2	66170.59	469.15
3	mid3	72800.68	468.67
4	mid4	64809.71	468.44
5	mid5	69029.91	469.65
6	mid6	65474.55	469.07
7	mid7	65563.13	470.42
8	mid8	63425.26	470.35
9	mid9	67399.35	470.16
10	mid10	67812.41	471.78
11	mid11	67186.42	467.36
12	mid12	63539.26	471.68
13	mid13	62670.64	289.72
14	mid14	62398.51	279.15
15	mid15	63779.14	288.83
16	mid16	63636.94	284.31
17	mid17	64071.32	281.26
18	mid18	66316.26	287.04
19	mid19	64086.86	292.75
20	mid20	65108.03	319.81
21	mid21	63540.76	279.85
22	mid22	64124.12	285
23	mid23	64224.32	286.11
24	mid24	64986.46	292.08
25	mid25	64471.53	299.72

Untuk hasil ujicoba yang berupa waktu pemrosesan ditampilkan dalam bentuk diagram colom seperti berikut.



Gambar 4.6 Grafik waktu pemrosesan yang dibutuhkan untuk penghapusan noise salt & pepper sebesar 50% dengan metode *alpha trimmed mean* pada gambar Indoor.

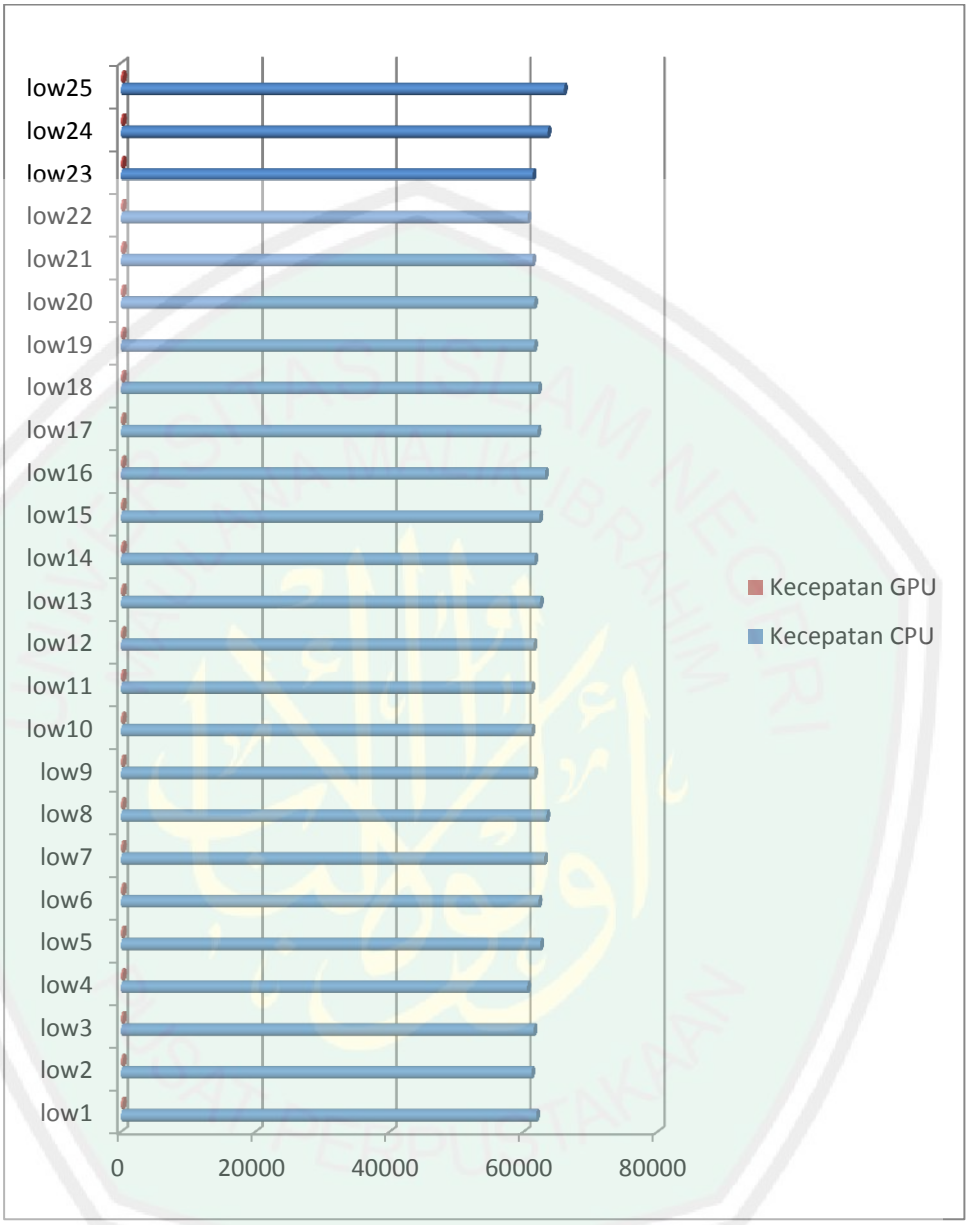
Dari Informasi pada table 4.5 dan Grafik 4.6, hasil ujicoba gambar outdoor yang mengandung noise salt and pepper sebesar 30% diperoleh rata-rata waktu pemrosesan pada komputasi yang dijalankan murni pada CPU menggunakan platform java sebesar 65430.7576 ms dan yang berjalan pada CPU-GPU

menggunakan platform OpenCL sebesar 376.2296 ms. Waktu yang diperlukan lebih sedikit dibandingkan pada noise 80%.

Table 4.6 Hasil ujicoba data outdoor yang telah ditambahkan noise salt and pepper sebesar 30% .

No.	Nama Data	Kecepatan	
		CPU	GPU
1	low1	61905.07	283.61
2	low2	61180.27	282.14
3	low3	61530.5	279.41
4	low4	60559.71	280.91
5	low5	62512.88	280.81
6	low6	62272.22	282.87
7	low7	63110.71	280.95
8	low8	63439.34	281.24
9	low9	61603.91	277.71
10	low10	61220.56	280.13
11	low11	61180.27	282.14
12	low12	61529.32	282.17
13	low13	62459.5	289.07
14	low14	61624.87	295.96
15	low15	62400	286.06
16	low16	63210.3	289.49
17	low17	62124.22	277.8
18	low18	62196.67	281.96
19	low19	61584.62	281.59
20	low20	61588.24	282.65
21	low21	61323.88	309.08
22	low22	60613.98	281.4
23	low23	61388.94	278.34
24	low24	63605.97	286.81
25	low25	66036.33	298.18

Untuk hasil ujicoba yang berupa waktu pemrosesan ditampilkan dalam bentuk diagram colom seperti berikut.

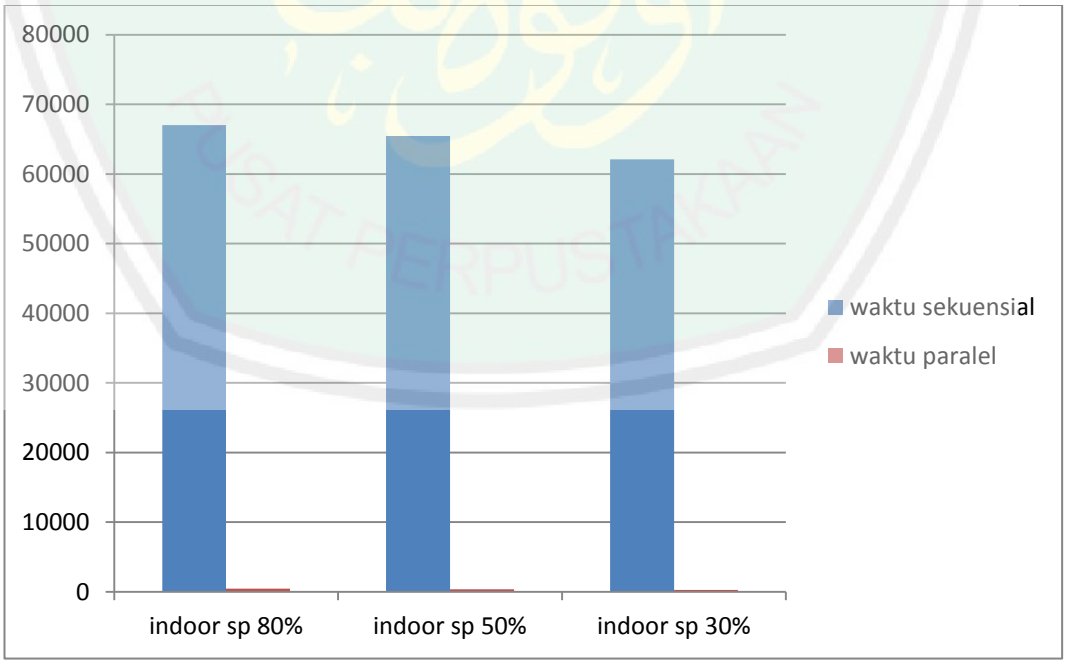


Gambar 4.7 Grafik waktu pemrosesan yang dibutuhkan untuk penghapusan noise salt & pepper sebesar 30% dengan metode *alpha trimmed mean* pada gambar Indoor.

Dari Informasi pada table 4.6 dan Grafik 4.7 , hasil ujicoba gambar outdoor yang mengandung noise salt and pepper sebesar 30% diperoleh rata-rata waktu pemrosesan pada komputasi yang dijalankan murni pada CPU menggunakan platform java sebesar 62088.09 ms dan yang berjalan pada CPU-GPU

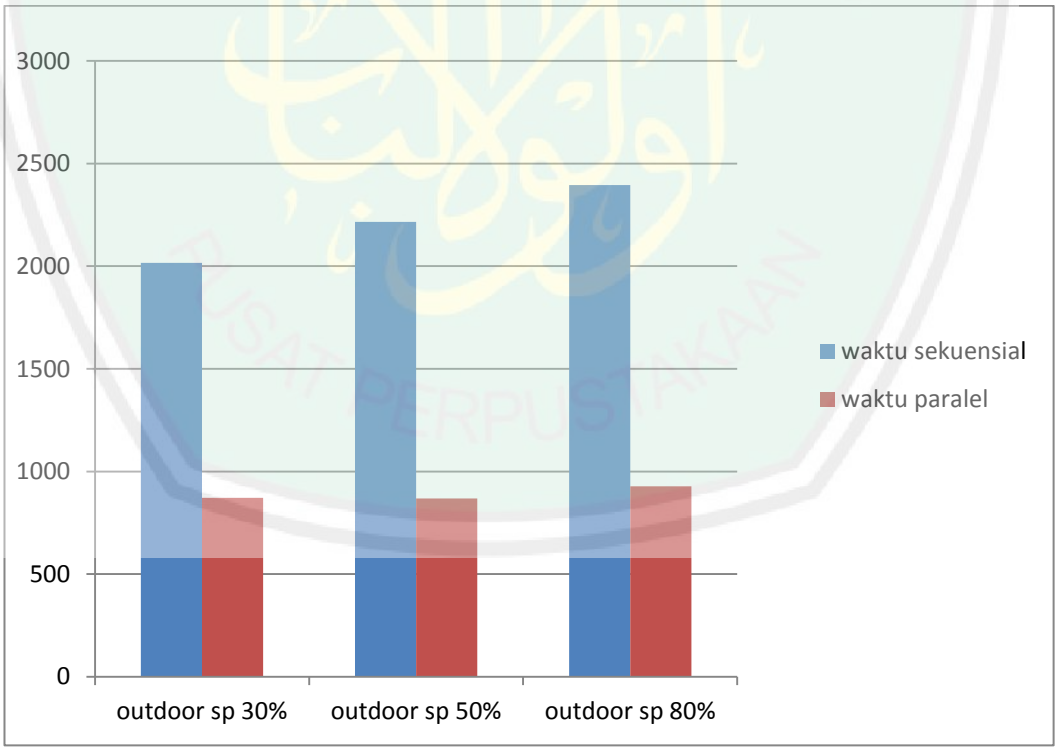
menggunakan platform OpenCL sebesar 284.4992 ms. Waktu yang diperlukan lebih sedikit dibandingkan pada noise 80% dan 50%.

Dari data keseluruhan table 2 diambil rata-rata dari hasil setiap file gambar uji coba. Untuk gambar indoor berprosentase noise salt & pepper sebesar 80% memiliki nilai rerata pada pemrograman skuensial pada platform java 67036.5428 ms untuk waktu pemrosesan dan 471.5736 ms untuk waktu pemrosesan. Untuk gambar indoor berprosentase noise salt & pepper sebesar 50% memiliki nilai rerata pada pemrograman skuensial pada platform java 65430.7576 ms untuk waktu pemrosesan dan 376.2296 ms untuk waktu pemrosesan. Untuk gambar indoor berprosentase noise salt & pepper sebesar 30% memiliki nilai rerata pada pemrograman skuensial pada platform java 62088.0912 ms untuk waktu pemrosesan dan 284.4992 ms untuk waktu pemrosesan pada platform OpenCL .



Gambar 4.8 Grafik waktu pemrosesan rata-rata dari masing-masing gambar yang bernoise salt and pepper 80%,50%, dan 30%.

Dari data table diambil rata-rata dari hasil setiap file gambar uji coba. untuk gambar outdoor berprosentase noise salt & pepper sebesar 80% memiliki nilai rerata pada pemrograman skuensial pada platform java 2395.994 ms untuk waktu pemrosesan dan 927.4158 ms untuk waktu pemrosesan padaOpenCL . Untuk gambar outdoor berprosentase noise salt & pepper sebesar 50% memiliki nilai rerata pada pemrograman skuensial pada platform java 2215.577 ms untuk waktu pemrosesan dan 868.9413 ms untuk waktu pemrosesan. Untuk gambar outdoor berprosentase noise salt & pepper sebesar 30% memiliki nilai rerata pada pemrograman skuensial pada platform java 2016.11ms untuk waktu pemrosesan dan pada GPU871.845ms untuk waktu pemrosesan .



Gambar 4.9 Grafik waktu pemrosesan rata-rata dari masing-masing gambar outdoor yang bernoise salt and pepper 80%,50%, dan 30%.

.

4.4. Pembahasan

Dari serangkaian uji coba yang dilakukan diperoleh hasil output berupa gambar yang sudah dilakukan penghilangan noise dengan metode *alpha trimmed mean filter*. Output ini selanjutnya dihitung dengan gambar referensi untuk mengetahui tingkat keberhasilan dari penerapan metode alpha trimmed mean filter pada komputasi pararel dengan cara mengurangkan gambar input dengan gambar output dengan menyimpan gambar referensi sebagai ground truth. Untuk detail hasil perhitungan nilai yang dihasilkan sebagai berikut.

Cara memperoleh nilai selisih adalah dengan menjumlah tiap layer gambar referensi kemudian dibagi dengan banyaknya pixel pada gambar tersebut yang selanjutnya dikurangkan dengan jumlah tiap layer gambar hasil yang dibagi dengan banyaknya pixel gambar hasil.

Layer Red= (743470177: 5992704)-(675862190: 5992704)=

124.0626- 112.7808=11.2818

Layer Green= (750687255: 5992704)-(723306548: 5992704)=

125.2669-120.6979=4.569

Layer Blue= (729515640: 5992704)-(613889709: 5992704)=

121.7340- 102.4395=19.2945

Nilai selisih harusnya 0 ,yang berarti gmbar hasil sama halnya dengan gambar referensi. Artinya metodde alpha trimmed mean bekerja 100% dalam penghapusan noise. Selain perhitungan selisih pixel di hitung juga kualitas citra hasil menggunakan PSNR (*Peak Signal Noise Ratio*) adalah perbandingan antara nilai maksimum dari sinyal yang diukur dengan besarnya derau yang

berpengaruh pada sinyal tersebut. PSNR biasanya diukur dalam satuan decibel (db). PSNR digunakan untuk mengetahui perbandingan kualitas citra cover sebelum dan sesudah disisipkan pesan. Untuk menentukan PSNR, terlebih dahulu harus ditentukan nilai MSE (Mean Square Error). MSE adalah nilai error kuadrat rata-rata antara citra asli dengan citra manipulasi (dalam kasus steganografi ; MSE adalah nilai error kuadrat rata-rata antara citra asli (cover-image) dengan citra hasil pemfilteran (stego-image).

Penjelasan penghitungan PSNR adalah sebagai berikut. Tersedia suatu citra asli dan citra hasil Untuk menghitung nilai PSNR, Jangkahpertama yang perlu dilakukan adalah menghitung nilai dari MSE (mean squared error Setelah Setelah mendapatkan semua koefisien yang dibutuhkan, maka penghi tungan PSNR dapat dilakukan.

$$PSNR = 10\log\left(\frac{255^2}{\frac{1}{M\times N}\sum_{i=1}^{M}\sum_{j=1}^{N}\left(X(i,j)-\hat{X}(i,j)\right)^2}\right) \quad (4.1)$$

Dimana X adalah gambar referensi atauground truth dan Ẋadalah hasil dari metode penghapusan noise menggunakan *alpha trimmed mean*.Dari keseluruhan data yang sudah disiapkan dilakukan perhitungan selisih dan perhitungan PSNR untuk selanjutnya dijadikan bahan acuan untuk menilai kinerja alpha trimmed mean filter untuk penghapusan noise. Berikut data yang diperoleh dari perhitungan selisih dan PNSR.

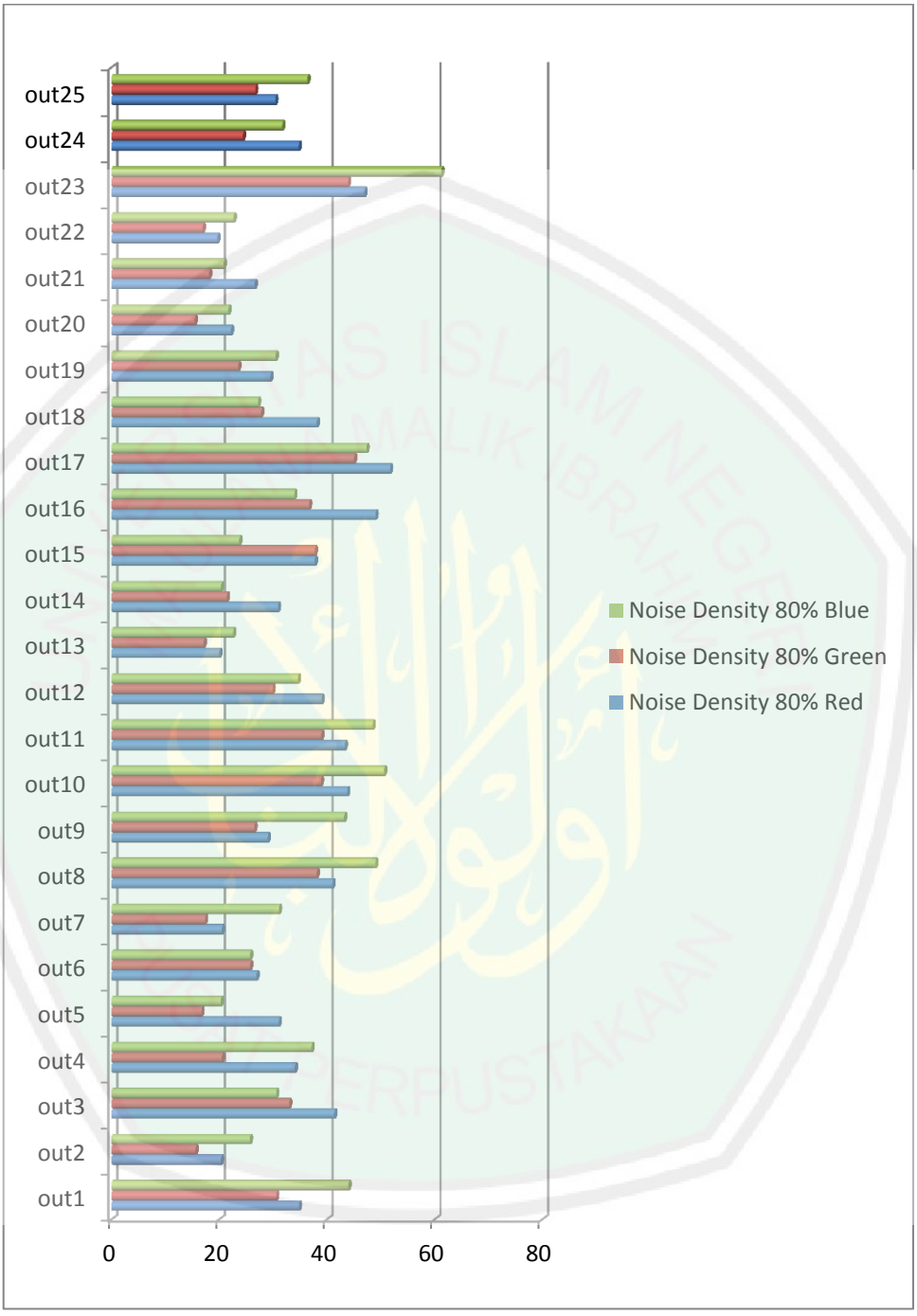
4.4.2. Data Outdoor

Pembahasan untuk data outdoor mengenai kinerja metode alpha trimed mean di jelaskan pada bagian ini.

Tabel 4.3 Hasil pengurangan gambar input outdoor dengan gambar hasil pada noise *salt and pepper* 80%.

No	Nama	Noise Density 80%		
		Red	Green	Blue
1	out1	34.8336	30.6054	44.0303
2	out2	20.3423	15.6406	25.7233
3	out3	41.3895	33.0139	30.6029
4	out4	34.0829	20.5418	37.1638
5	out5	31.079	16.7191	20.3258
6	out6	26.9522	25.776	25.776
7	out7	20.5316	17.3884	31.13
8	out8	41.0642	38.1388	49.0246
9	out9	29.0449	26.5675	43.2308
10	out10	43.7912	38.9216	50.662
11	out11	43.3487	38.9983	48.5228
12	out12	39.0592	29.9224	34.6421
13	out13	20.0317	17.1482	22.6212
14	out14	30.9414	21.4426	20.3745
15	out15	37.7797	37.7797	23.764
16	out16	49.0517	36.7223	33.9739
17	out17	51.7524	45.082	47.3803
18	out18	38.149	27.8254	27.2435
19	out19	29.5391	23.5618	30.5167
20	out20	22.2091	15.4765	21.7349
21	out21	26.6073	18.151	20.8633
22	out22	19.686	16.9598	22.7073
23	out23	46.9853	43.9514	61.2857
24	out24	34.7976	24.4266	31.7575
25	out25	30.4445	26.7169	36.4083

Untuk memudahkan melihat perbedaan hasil selisih antar satu data dengan data lain disajikan data table dalam data grafik .



Gambar 4.10 Grafik Hasil pengurangan gambar input outdoor dengan gambar hasil pada noise *salt and pepper* 80%.

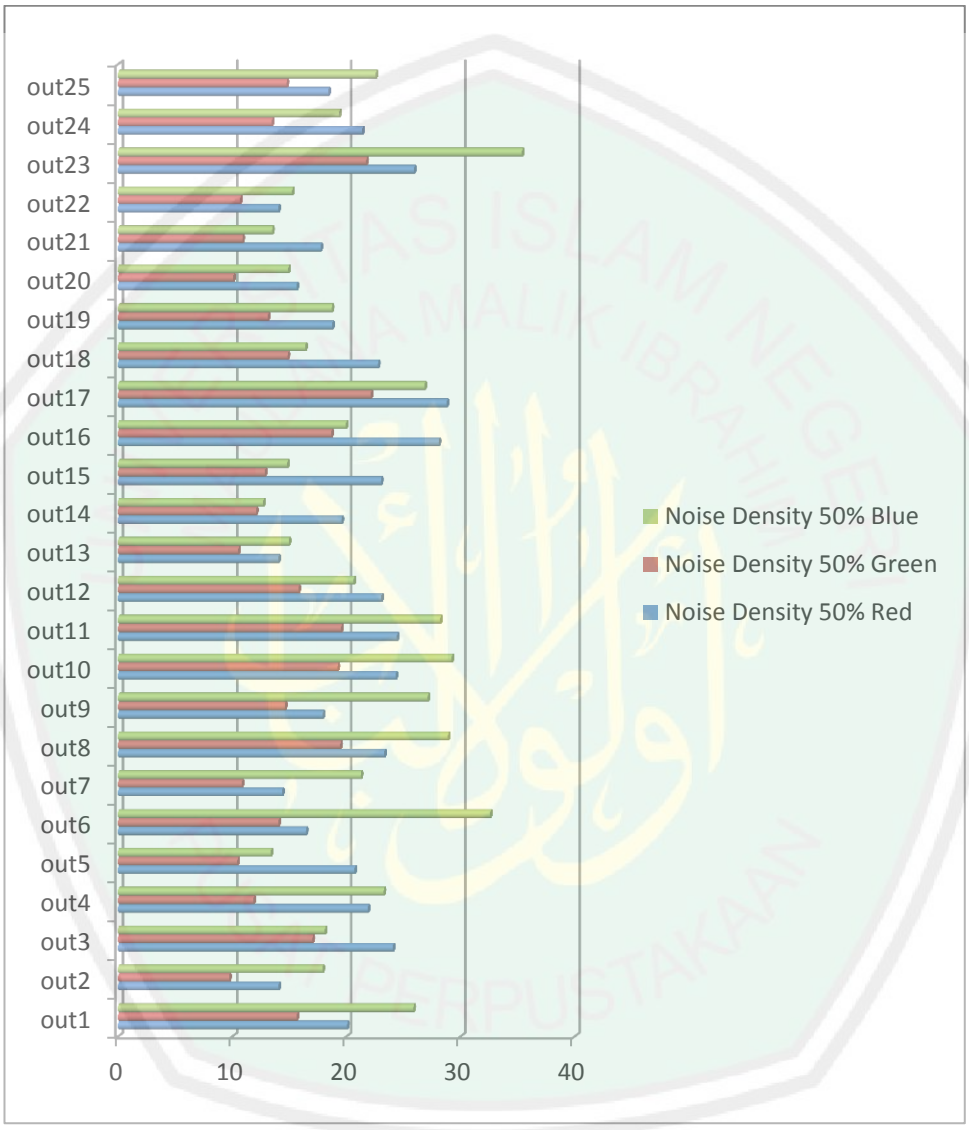
Table 4.3 dan Grafik 4.10 menunjukkan secara keseluruhan nilai selisih dari setiap gambar data dengan gambar input. Nilai memuaskan jika selisih diperoleh semakin kecil dan mendekati nol. Diperoleh selisih rerata dari

keseluruhan data outddor dengan noise 80% sebesar 33.73976 pada layer merah, 27.49912 pada layer hijau, dan pada 33.65862 layer biru. Layer biru menyumbangkan nilai selisis paling besar .Dari ketiga nilai selisih tersebut memiliki rata-rata 31.6325 . Niali tersebut masih terpaut cukup besar dari nilai 0 yang artinya metode alpha trimmed mean belum bekerja dengan baik dalam penghapusan noise salt and pepper pada gambar yang bernoise tinggi 80%.

Tabel 4.4 Hasil pengurangan gambar input outdoor dengan gambar hasil pada noise *salt and pepper* 50%

No	Nama	Noise Density 50%		
		Red	Green	Blue
1	out1	20.0738	15.655	25.887
2	out2	14.0633	9.7307	17.9082
3	out3	24.0814	17.0397	18.1274
4	out4	21.8917	11.8729	23.2785
5	out5	20.7212	10.4567	13.3782
6	out6	16.4647	14.0541	32.608
7	out7	14.3761	10.8437	21.2696
8	out8	23.3267	19.4788	28.9142
9	out9	17.9473	14.6261	27.1455
10	out10	24.3278	19.2369	29.2409
11	out11	24.4441	19.554	28.2384
12	out12	23.0727	15.8137	20.6501
13	out13	14.0524	10.537	14.9581
14	out14	19.5831	12.0907	12.7145
15	out15	23.038	12.8689	14.8378
16	out16	28.0977	18.6886	19.968
17	out17	28.8039	22.155	26.8616
18	out18	22.7783	14.8596	16.4041
19	out19	18.7665	13.134	18.7333
20	out20	15.6444	10.0934	14.9222
21	out21	17.7487	10.8996	13.5082
22	out22	14.0424	10.6951	15.262
23	out23	25.9372	21.7542	35.4005
24	out24	21.3946	13.4781	19.3785
25	out25	18.4271	14.7743	22.5846

Untuk memudahkan melihat perbedaan hasil selisih antar satu data dengan data lain disajikan data table dalam data grafik .



Gambar 4.11 Grafik Hasil pengurangan gambar input outdoor dengan gambar hasil pada noise *salt and pepper* 50%.

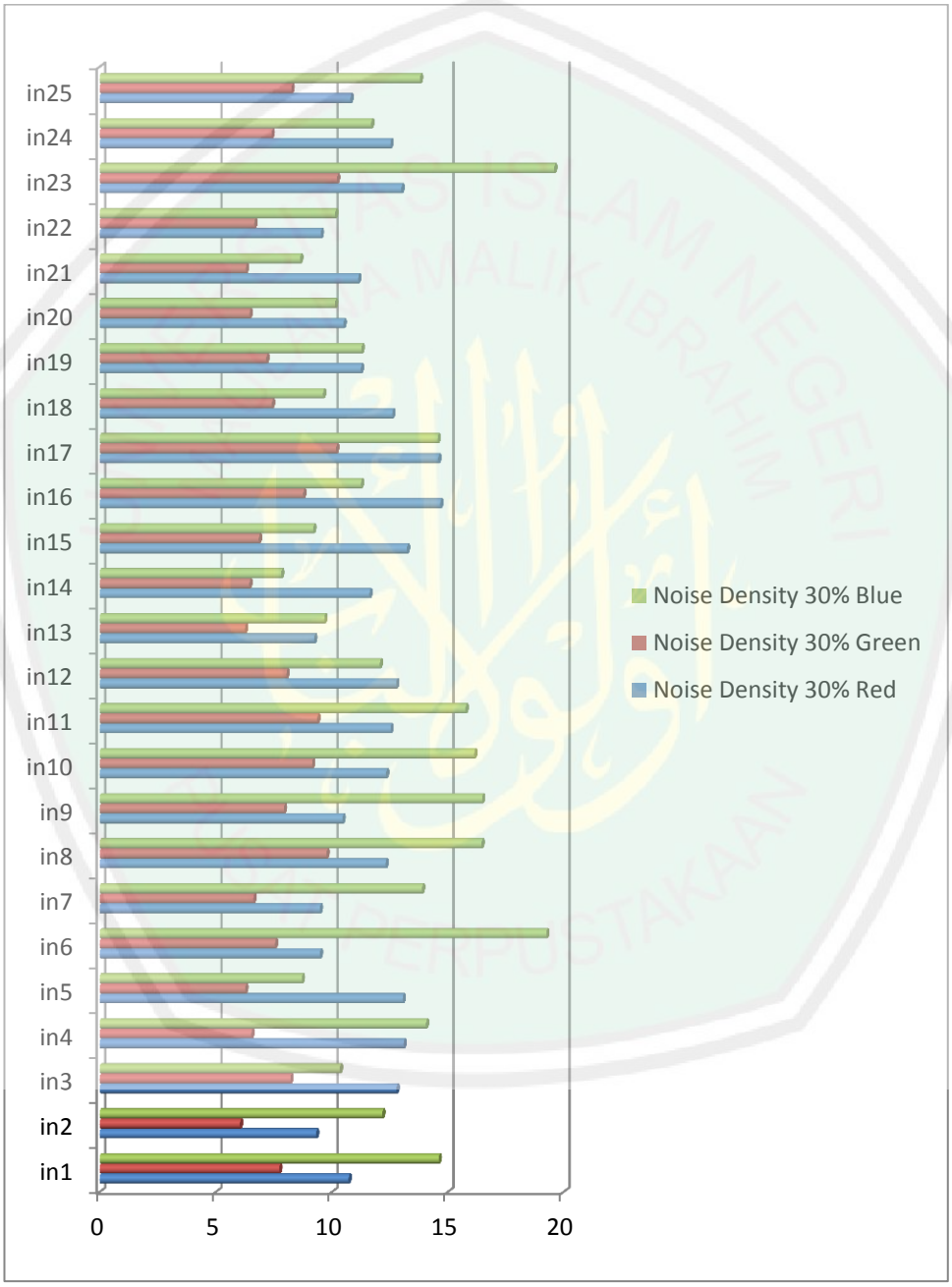
Table 4.4dan Grafik 4.11 menunjukkan secara keseluruhan nilai selisih dari setiap gambar data dengan gambar input. Nilai memuaskan jika selisih diperoleh semakin kecil dan mendekati nol. Diperoleh selisih rerata dari keseluruhan data outddor dengan noise 50% sebesar 20.5242 pada layer

merah,14.57563 pada layer hijau, dan pada 21.28718 layer biru. Layer biru menyumbangkan nilai selisis paling besar .Dari ketiga nilai selisih tersebut memiliki rata-rata 18.79567. Niali tersebut masih terpaut cukup besar dari nilai 0 yang artinya metode alpha trimmed meanbekerja lebih baik dalam penghapusan noise salt and pepper pada gambar yang bernoise tinggi 50% karena selisih yang terpaut semakin kecil disbanding pada gambar dengan noise 80%.

Tabel 4.5 Hasil pengurangan gambar input outdoor dengan gambar hasil pada noise *salt and pepper* 30%

No	Nama	Noise Density 50%		
		Red	Green	Blue
1	in1	10.7235	7.7377	14.6164
2	in2	9.3267	6.0459	12.1802
3	in3	12.7945	8.2139	10.3485
4	in4	13.0964	6.5361	14.0657
5	in5	13.0576	6.275	8.7098
6	in6	9.4891	7.5536	19.2588
7	in7	9.4866	6.6215	13.8959
8	in8	12.3189	9.7742	16.4625
9	in9	10.4622	7.9369	16.4857
10	in10	12.3475	9.148	16.1496
11	in11	12.5328	9.3772	15.7742
12	in12	12.7789	8.0461	12.0725
13	in13	9.241	6.2581	9.6765
14	in14	11.6231	6.4525	7.8216
15	in15	13.2417	6.86	9.2031
16	in16	14.6822	8.7705	11.2567
17	in17	14.596	10.1871	14.5649
18	in18	12.608	7.4248	9.6251
19	in19	11.2538	7.1817	11.2821
20	in20	10.5076	6.4587	10.1386
21	in21	11.1367	6.2956	8.6382
22	in22	9.5265	6.6633	10.1459
23	in23	13.0041	10.2248	19.6014
24	in24	12.5285	7.407	11.6996
25	in25	10.8047	8.263	13.8088

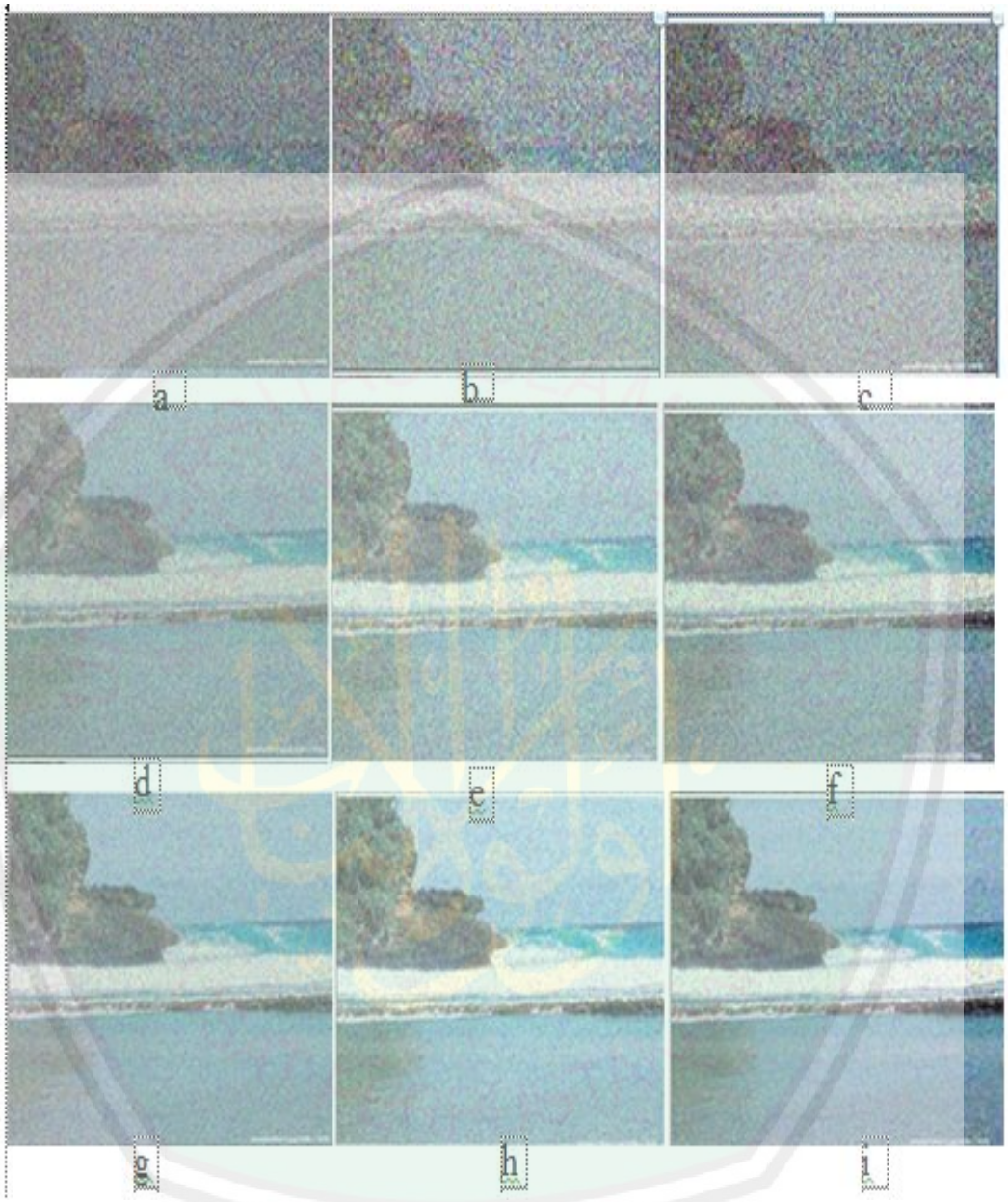
Untuk memudahkan melihat perbedaan hasil selisih antar satu data dengan data lain disajikan data table dalam data grafik .



Gambar 4.12 Grafik Hasil pengurangan gambar input outdoor dengan gambar hasil pada noise *salt and pepper* 30%.

Table 4.5 dan Grafik 4.12 menunjukkan secara keseluruhan nilai selisih dari setiap gambar data dengan gambar input. Nilai memuaskan jika selisih diperoleh semakin kecil dan mendekati nol. Diperoleh selisih rerata dari keseluruhan data outddor dengan noise 50% sebesar 11.72674 pada layer merah, 7.668528 pada layer hijau, dan pada 12.69929 layer biru. Layer biru menyumbangkan nilai selisis paling besar .Dari ketiga nilai selisih tersebut memiliki rata-rata 10.69819. Niali tersebut masih terpaut cukup besar dari nilai 0 yang artinya metode alpha trimmed mean bekerja lebih baik dalam penghapusan noise salt and pepper pada gambar yang bernoise tinggi 50% karena selisih yang terpaut semakin kecil dibanding pada gambar dengan noise 50% dan 80%.

Selain data grafik hasil bisa dilihat dari Pada gambar 4.10. Setiap gambar menunjukkan hasil perpotongan dari gambar yang telah diproses. Kolom pertama menunjukkan gambar input dari yang paling atas ke bawah bernoise 80%,50%, dan 30%. Kolom kedua menunjukkan hasilmetode *Alpha Trimmed Mean* yang dikerjakan di CPU. Kolom ketiga menunjukkan hasil metode Alpha Trimmed Mean yang dikerjakan di GPU. Dari hasil metode alpha trimmed mean filter men yang ditampilkan pada gambar telihat hnilai akurasi yang semakin baik jika kondisi noise semakin direndahkan .semakin besar noise yang terkandung gambar yang dihasilkan juga semakin rendah kualitasnyaatau masih banyak mengandung noise.



Gambar 4.13. Hasil dari penghapusan noise salt & pepper pada gambar outdoor (a)Gambar masukan bernoise 80% ,(b) Output CPU dari input gambar a, (c) Output CPU dari input gambar a, (d) Gambar masukan bernoise 50% ,(e) Output CPU dari input gambar d, (f) Output CPU dari input gambar d, (g)Gambar masukan bernoise 30% ,(b) Output CPU dari input gambar g, (c) Output CPU dari input gambar g.

Nilai selisih yang mencapai angka 30 menunjukkan metode a*lpha trimmed mean* pada penelitian ini tidak bekerja baik untuk melakukan penghapusan noise dengan inrtensitas tinggi . Terlihat pada gambar 4.13 dan data yang ditampilkan pada table 4.22 menunjukkan semakin kecil noise yang ditangani semakin baik kinerja metode alpha trimmed mean untuk penghapusan noise.

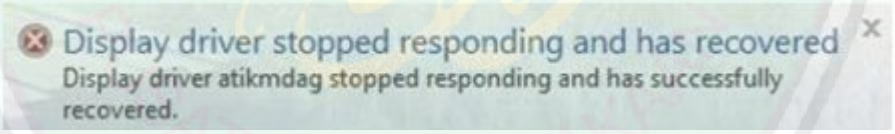
Tabel 4.6 Nilai PNSR dari hasil gambar outdoor bernoise 30%, 50%, 80%.

No	PSNR(db)		
	High	Mid	Low
1	10.7105	15.0132	19.7611
2	12.2202	16.1426	20.2773
3	11.621	15.815	20.5712
4	11.5871	15.5644	19.8867
5	12.3198	16.0914	20.0617
6	11.4726	15.4885	19.8513
7	13.3192	16.8852	20.6676
8	10.7185	14.8245	19.1513
9	12.3122	16.1417	20.2094
10	10.5292	14.795	19.4137
11	10.9001	15.1323	19.7896
12	11.2021	15.3466	19.8364
13	12.8484	16.6912	20.8096
14	12.3033	16.3222	20.7408
15	11.6278	15.679	20.1709
16	11.2203	15.4466	20.2452
17	10.4661	14.8735	19.8267
18	11.8394	15.9614	20.5773
19	11.7214	15.7708	20.1365
20	12.7701	16.4529	20.2526
21	12.5465	16.4048	20.5086
22	12.8428	16.5528	20.4059
23	9.8745	14.2049	18.9475

Table 4.6 menerangkan tentang nilai PSNR yang diperoleh dari tiap-tiap

Hasil ujicoba pada masing-masing data outdoor dengan gambar acuan (Ground Truth). Dari setiap kelompok data dengan no kepadatan noise yang berbeda diperoleh nilai rata-rata. Nilai PSNR untuk gambar outdoor dengan

kepadatan noise 80% sebesar 11.66587 db, pada kepadatan noise 50% sebessar15.69571 db, pada kepadatan noise 30% sebessar20.06161 db.PSNR sering dinyatakan dalam skala logaritmik dalam decibel (dB). Nilai PSNR jatuh dibawah 30 dB mengindikasikan qualitas yang relative rendah, dimana distorsi yang dikarenakan penyisipan terlihat jelas. Akan tetapi kualitas stego-image yang tinggi berada pada nilai 40dB dan diatasnya (Cheddad, 2010). Pada penelitian ini penerapan *alpha trimmed mean* ini untuk kasus penghapusan noise bekerja tidak maksimal dengan kualitas relative rendah untuk gambar output yang dihasilkan. Hal ini terjadi karena window kernel gambar yang di pakai hanya terbatas pada ukuran 3x3. Dalam penelitian ini ukuran window tidak dapat diperbesar karena keterbatasan sumberdaya computer yang digunakan. Ketika di perbesar ukuran window kernel ,maka computer tidak menjalankan program dan memunculkan jendela dengan pesan " *Display driver stopped responding and has recovered*".



Hal tersebut disebabkan oleh Windows service bernama Timeout Detection And Recovery (TDR) TDR diperkenalkan sejak perilisan Windows Vista dan hadir pula dalam Windows 7 dan Windows 8. Fungsi TDR adalah untuk mendeteksi momen hang. Begitu momen itu terdeteksi, TDR akan otomatis berusaha untuk menyelesaikan masalah itu tanpa memaksa pengguna untuk reboot. Dalam kasus VGA card, saat display hang, TDR akan memanggil fungsi driver miniport bernama DxgkDdiResetFromTimeout untuk mereboot driver dan mereset Graphic Processing Unit (GPU).

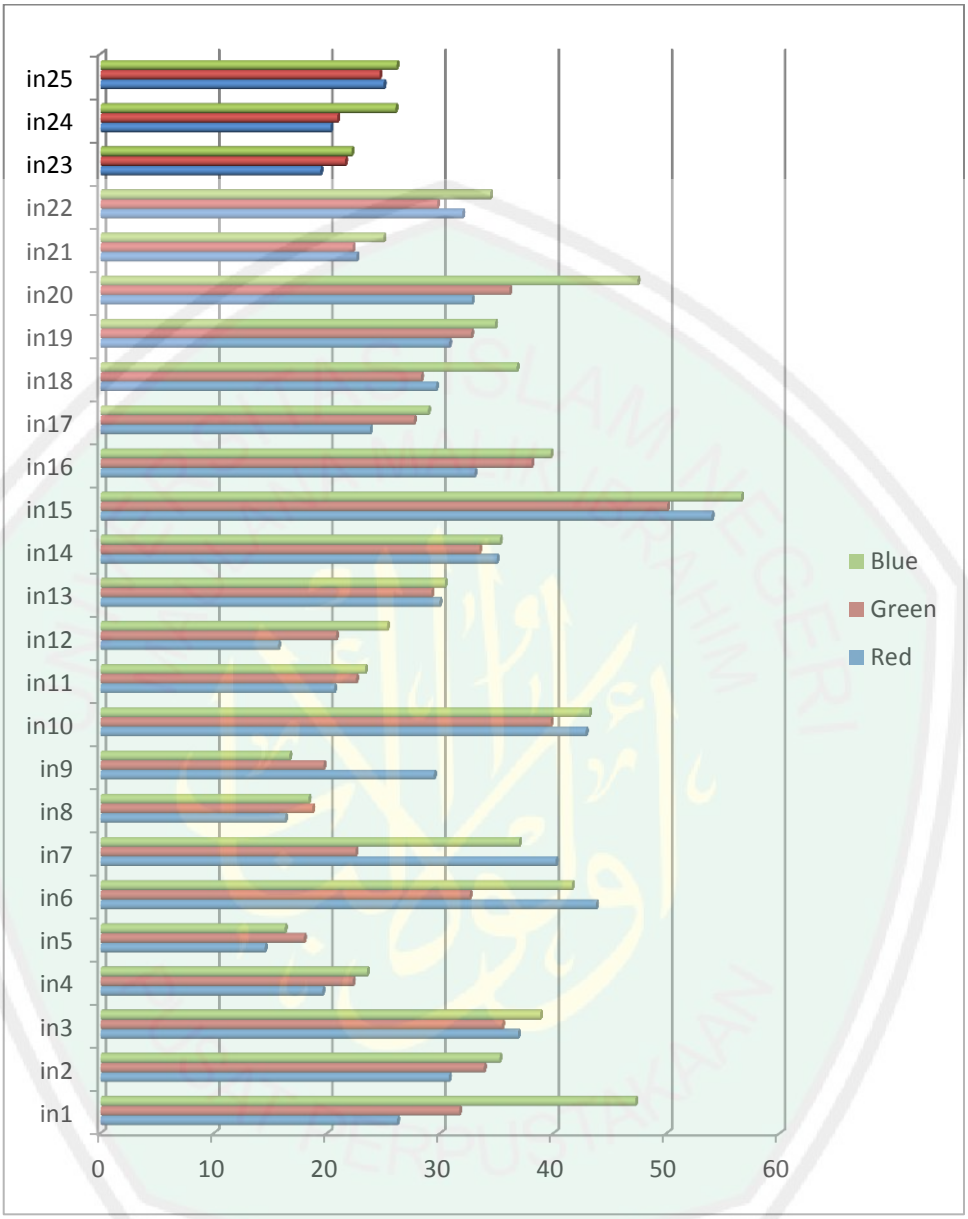
4.4.2. Data Indoor

Pembahasan untuk data indoor mengenai kinerja metode *alpha trimed mean filter* di jelaskan pada bagian ini.

Tabel 4.7 Hasil pengurangan gambar input indoor dengan gambar hasil pada noise *salt and pepper* 80%

No	Nama	Noise Density 80%		
		Red	Green	Blue
1	in1	26.2174	31.6831	47.2476
2	in2	30.7871	33.8721	35.2427
3	in3	36.87	35.4838	38.8433
4	in4	19.6066	22.2937	23.533
5	in5	14.5131	17.9787	16.2852
6	in6	43.7809	32.6124	41.6382
7	in7	40.2252	22.5295	36.9627
8	in8	16.3053	18.7226	18.3911
9	in9	29.4568	19.7021	16.6992
10	in10	42.8698	39.7708	43.1596
11	in11	20.6383	22.5794	23.3662
12	in12	15.699	20.8046	25.3112
13	in13	29.9575	29.2554	30.4167
14	in14	34.9952	33.464	35.292
15	in15	53.9924	50.0636	56.5998
16	in16	33.0367	38.0658	39.7536
17	in17	23.8111	27.6837	28.9417
18	in18	29.6355	28.3314	36.7672
19	in19	30.8064	32.7579	34.8617
20	in20	32.7988	36.124	47.4459
21	in21	22.6233	22.2883	25.0031
22	in22	31.976	29.7306	34.4316
23	in23	19.4349	21.6217	22.1739
24	in24	20.3247	20.9212	26.062
25	in25	25.0235	24.6405	26.1548

Untuk memudahkan melihat perbedaan hasil selisih antar satu data dengan data lain disajikan data table dalam data grafik .



Gambar 4.14Grafik Hasil pengurangan gambar input indoor dengan gambar hasil pada noise *salt and pepper* 80%.

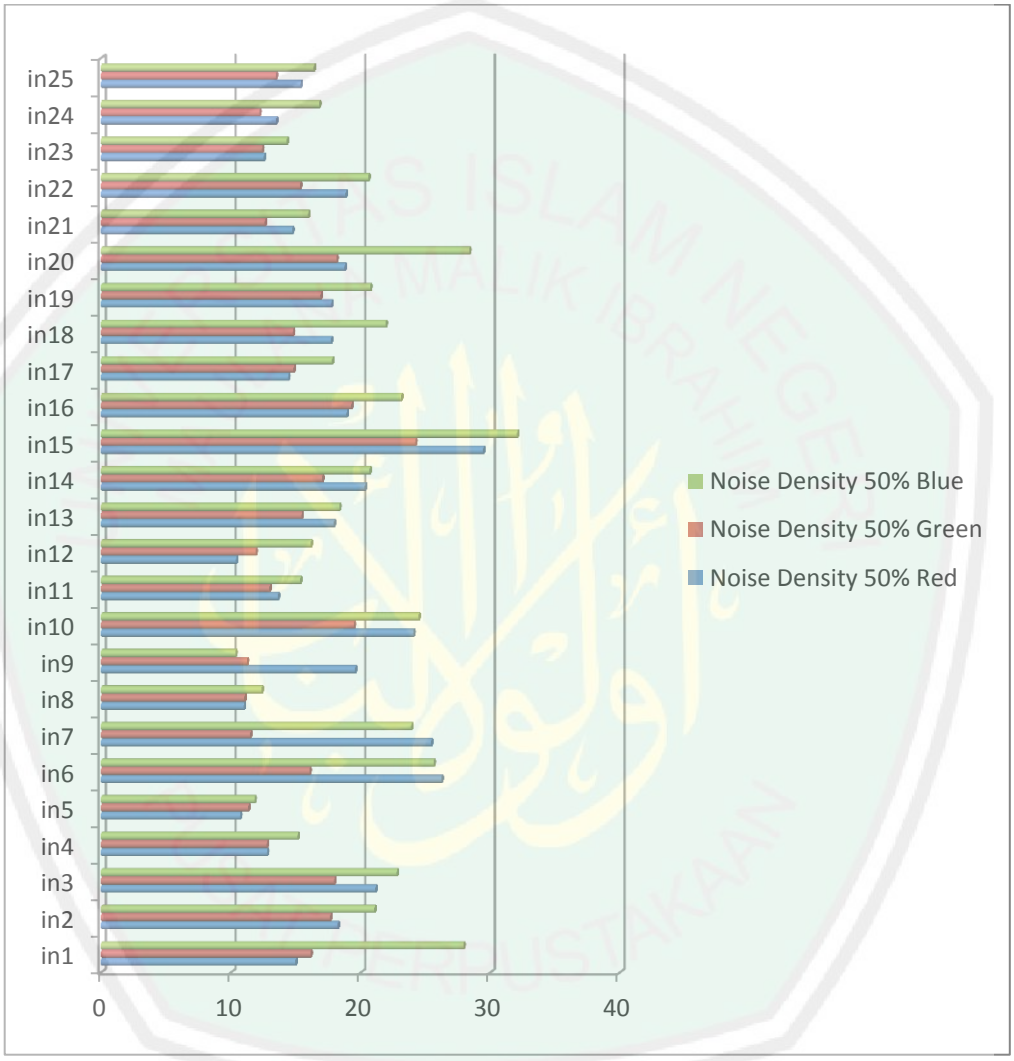
Table 4.7dan Grafik 4.14 menunjukkan secara keseluruhan nilai selisih dari setiap gambar data dengan gambar input. Nilai memuaskan jika selisih diperoleh semakin kecil dan mendekati nol. Diperoleh selisih rerata dari keseluruhan data indoor dengan noise 80% sebesar 29.01542 pada layer merah,

28.51924 pada layer hijau, dan pada 32.42336 layer biru. Layer biru menyumbangkan nilai selisis paling besar .Dari ketiga nilai selisih tersebut memiliki rata-rata 29.98601. Niali tersebut masih terpaut cukup besar dari nilai 0 yang artinya metode *alpha trimmed mean* bekerja lebih baik dalam penghapusan noise salt and pepper pada gambar yang bernoise tinggi 80% karena selisih yang terpaut masih jauh dari nilai 0.

Tabel 4.8 Hasil pengurangan gambar input indoor dengan gambar hasil pada noise *salt and pepper* 50%

No	Nama	Noise Density 50%		
		Red	Green	Blue
1	in1	14.9732	16.1356	27.9328
2	in2	18.2343	17.6527	21.0655
3	in3	21.1419	17.9627	22.7873
4	in4	12.784	12.775	15.1429
5	in5	10.7111	11.3467	11.8235
6	in6	26.2369	16.0569	25.6307
7	in7	25.4371	11.4902	23.9001
8	in8	10.9988	11.0553	12.3684
9	in9	19.5679	11.2329	10.3483
10	in10	24.0633	19.4624	24.4659
11	in11	13.6248	12.9742	15.3425
12	in12	10.4024	11.9036	16.1547
13	in13	17.9377	15.4457	18.3494
14	in14	20.3263	17.0334	20.6855
15	in15	29.4572	24.2119	32.0719
16	in16	18.9222	19.2886	23.1234
17	in17	14.4145	14.8394	17.8006
18	in18	17.7078	14.7829	21.9223
19	in19	17.7374	16.9181	20.7307
20	in20	18.7708	18.1315	28.3757
21	in21	14.7295	12.6253	15.9602
22	in22	18.8438	15.3284	20.5922
23	in23	12.549	12.4096	14.3098
24	in24	13.4855	12.195	16.8024
25	in25	15.3693	13.4645	16.3953

Untuk memudahkan melihat perbedaan hasil selisih antar satu data dengan data lain disajikan data table dalam data grafik .



Gambar 4.15 Grafik Hasil pengurangan gambar input indoor dengan gambar hasil pada noise *salt and pepper* 50%.

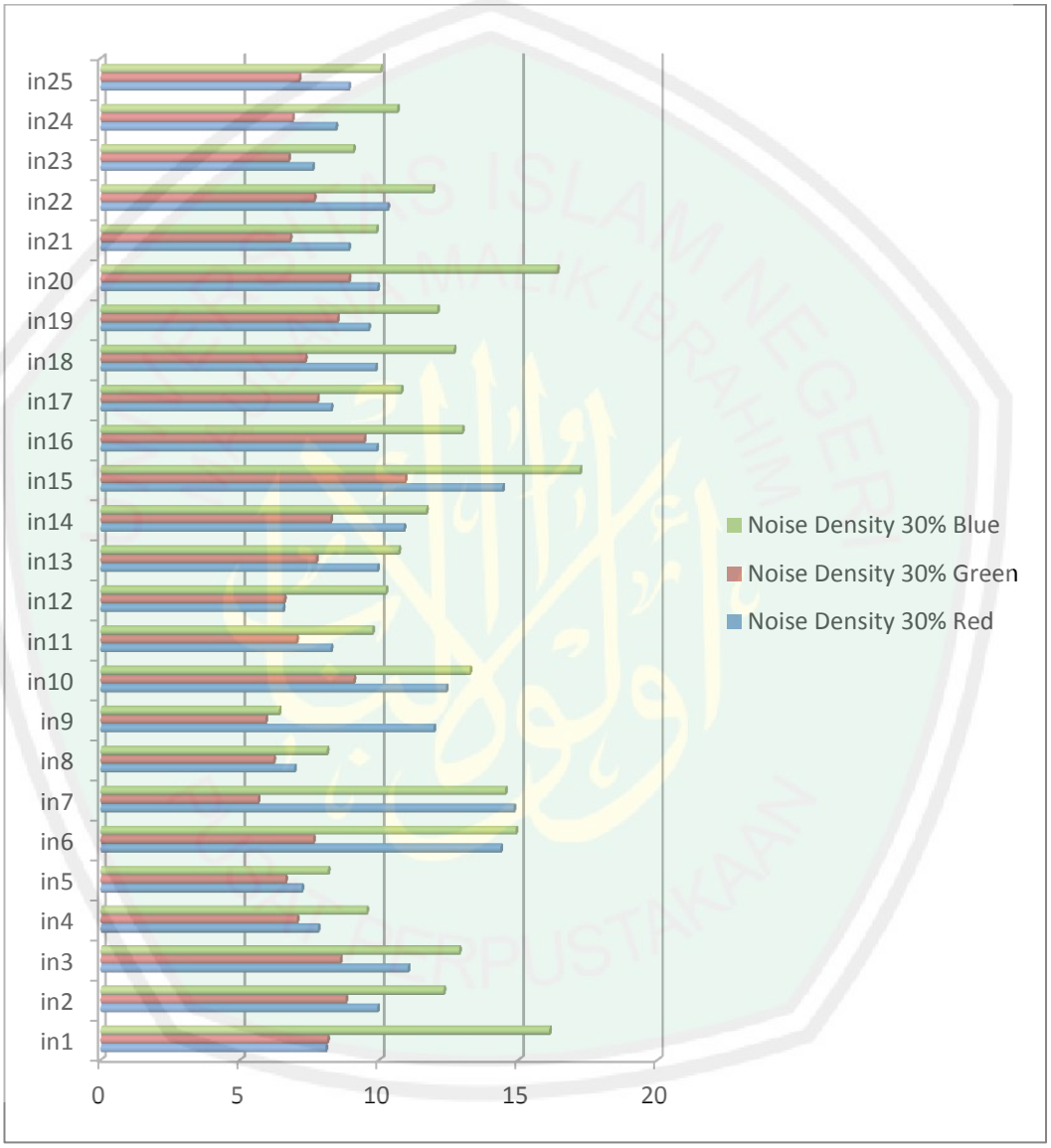
Table 4.8 dan Grafik 4.15 menunjukkan secara keseluruhan nilai selisih dari setiap gambar data dengan gambar input. Nilai memuaskan jika selisih diperoleh semakin kecil dan mendekati nol. Diperoleh selisih rerata dari keseluruhan data indoor dengan noise 50% sebesar 17.53707 pada layer merah,

15.0689 pada layer hijau, dan pada 19.76328 layer biru. Layer biru menyumbangkan nilai selisis paling besar .Dari ketiga nilai selisih tersebut memiliki rata-rata 17.45642. Niali tersebut masih terpaut cukup besar dari nilai 0 yang artinya metode *alpha trimmed mean* bekerja lebih baik dalam penghapusan noise salt and pepper pada gambar yang bernoise sedang 50% jika dibandingkan pada hasil dari gambar bernoise tinggi 80%

Tabel 4.9Hasil pengurangan gambar input indoor dengan gambar hasil pada noise *salt and pepper* 30%

No	Nama	Noise Density 30%		
		Red	Green	Blue
1	in1	8.0722	8.1349	16.1149
2	in2	9.9408	8.7958	12.3163
3	in3	11.0366	8.6039	12.8699
4	in4	7.8002	7.0441	9.5433
5	in5	7.2139	6.6271	8.1646
6	in6	14.3626	7.6326	14.9015
7	in7	14.831	5.6447	14.5286
8	in8	6.9533	6.2095	8.128
9	in9	11.9554	5.9168	6.4082
10	in10	12.4008	9.0889	13.2519
11	in11	8.266	7.0315	9.7553
12	in12	6.5484	6.5875	10.237
13	in13	9.9298	7.7377	10.7017
14	in14	10.8781	8.2527	11.6949
15	in15	14.4317	10.9206	17.2109
16	in16	9.9079	9.4586	12.9778
17	in17	8.2682	7.7785	10.7847
18	in18	9.8691	7.3417	12.6868
19	in19	9.6105	8.5001	12.0902
20	in20	9.9437	8.9147	16.3962
21	in21	8.8944	6.7942	9.8982
22	in22	10.2952	7.655	11.9258
23	in23	7.6042	6.7383	9.0638
24	in24	8.439	6.8773	10.6508
25	in25	8.8953	7.1193	10.0492

Untuk memudahkan melihat perbedaan hasil selisih antar satu data dengan data lain disajikan data table dalam data grafik .



Gambar 4.16Grafik Hasil pengurangan gambar input indoor dengan gambar hasil pada noise *salt and pepper* 30%.

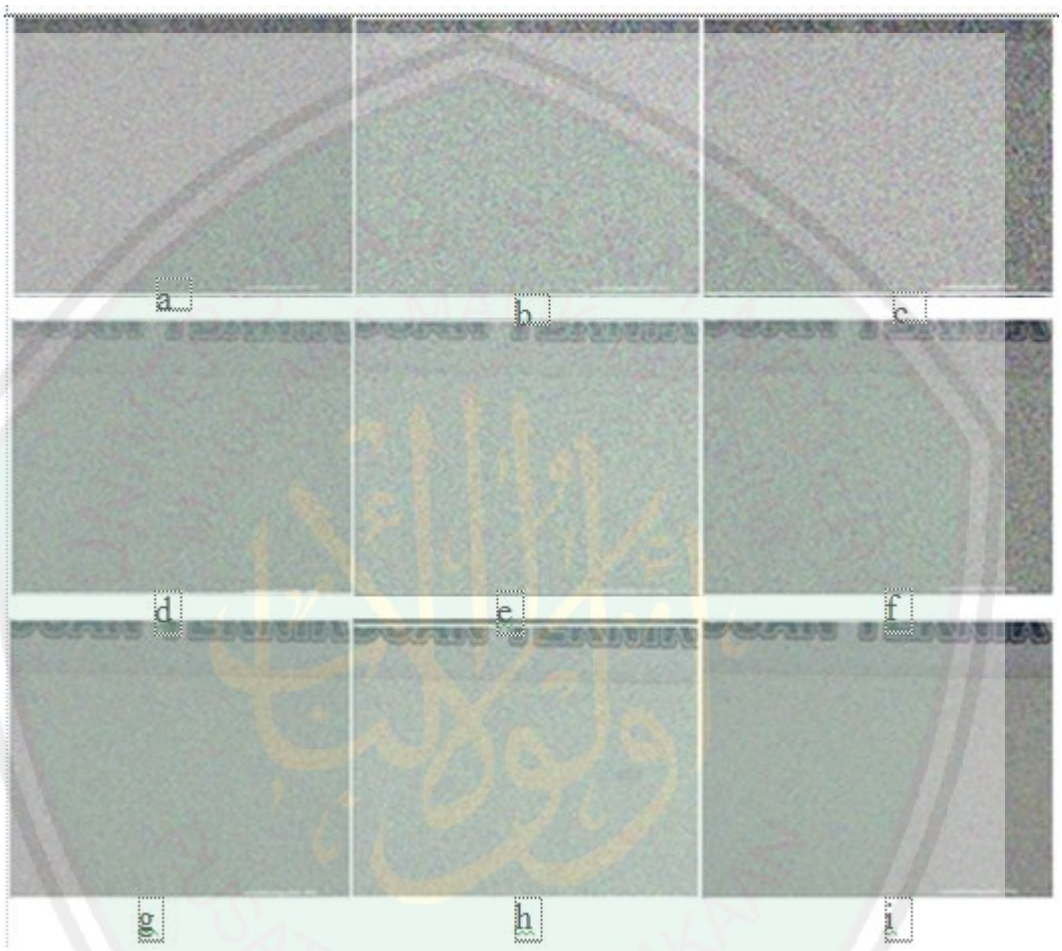
Table 4.9 dan Grafik 4.16 menunjukkan secara keseluruhan nilai selisih dari setiap gambar data dengan gambar input. Nilai memuaskan jika selisih diperoleh semakin kecil dan mendekati nol. Diperoleh selisih rerata dari

keseluruhan data indoor dengan noise 30% sebesar 9.853932 pada layer merah, 7.65624 pada layer hijau, dan pada 11.69402 layer biru. Layer biru menyumbangkan nilai selisis paling besar .Dari ketiga nilai selisih tersebut memiliki rata-rata 9.734731. Niali tersebut masih terpaut cukup besar dari nilai 0 yang artinya metode *alpha trimmed mean* bekerja paling baik dalam penghapusan noise salt and pepper pada gambar yang bernoise rendah 30% diantara hasil dari gambar bernoise sedang 50%.dan bernoise tinggi 80%

Selain data grafik hasil bisa dilihat dari Pada gambar 4.10. Setiap gambar menunjukkan hasil perpotongan dari gambar yang telah diproses. Kolom pertama menunjukkan gambar input dari yang paling atas ke bawah bernoise 80%,50%, dan 30%. Kolom kedua menunjukkan hasilmetode *alpha trimmed mean*yang dikerjakan di CPU. Kolom ketiga menunjukkan hasil metode Alpha Trimmed Mean yang dikerjakan di GPU. Dari hasil metode *alpha trimmed mean* filter men yang ditampilkan pada gambar telihat hnilai akurasi yang semakin baik jika kondisi noise semakin direndahkan .semakin besar noise yang terkandung gambar yang dihasilkan juga semakin rendah kualitasnya atau gambar tesebut masih banyak mengandung noise. Untuk gambar dari uraian tersebut bisa dilihat pada gambar 4.18.

Gambar 4.18 tersebut hasil dari penghapusan noise salt & pepper. Setiap gambar menunjukkan hasil perpotongan dari gambar yang telah diproses. Kolom pertama menunjukkan gambar input dari yang paling atas ke bawah bernoise 80%, 50%, dan 30%. Kolom kedua menunjukkan hasilmetode *Alpha Trimmed*

*Mean*yang dikerjakan di CPU. Kolom ketiga menunjukkan hasil metode *Alpha Trimmed Mean* yang dikerjakan di GPU.



Gambar 4.17.Hasil dari penghapusan noise salt & pepper pada gambar indoor (a)Gambar masukan bernoise 80% ,(b) Output CPU dari input gambar a, (c) Output CPU dari input gambar a,(d)gambar masukan bernoise 50% ,(e) Output CPU dari input gambar d, (f) Output CPU dari input gambar d,(g)Gambar masukan bernoise 30% ,(b) Output CPU dari input gambar g, (c) Output CPU dari input gambar g.

Nilai selisih yang mencapai angka 30 menunjukkan metode a*lpha trimmed mean* pada penelitian ini tidak bekerja baik untuk melakukan penghapusan noise dengan inrtensitas tinggi . Terlihat pada gambar 4.17dan data yang ditampilkan

pada table 4.10 menunjukkan semakin kecil noise yang ditangani semakin baik kinerja metode alpha trimmed mean untuk penghapusan noise.

Tabel 4.10 Nilai PNSR dari hasil gambar outdoor bernoise 30%, 50%, 80%.

No	PSNR(db)		
	High	Mid	Low
1	10.6901	15.0348	19.8509
2	12.3065	16.4619	21.0931
3	11.5896	15.8565	20.6785
4	13.247	17.1475	21.5445
5	15.2171	18.4955	22.3562
6	11.5918	15.8064	20.435
7	11.3072	15.5179	20.165
8	13.099	17.0613	21.5496
9	12.6446	16.705	21.176
10	11.0091	15.4005	20.3044
11	14.2687	17.9062	22.0703
12	13.1474	17.1043	21.5418
13	12.5204	16.6353	21.2551
14	12.3664	16.5362	21.2187
15	10.7912	15.183	20.1876
16	11.5988	15.825	20.5822
17	12.0999	16.2974	21.0013
18	11.9548	16.1899	20.9205
19	11.3516	15.6696	20.5303
20	11.8379	15.9376	20.4435
21	13.2985	17.153	21.4674
22	11.7323	15.9447	20.6043
23	12.8601	16.8782	21.4507
24	13.3927	17.2	21.4173
25	12.7731	16.8296	21.4128

Hasil ujicoba pada masing-masing data outdoor dengan gambar acuan (Ground Truth). Dari setiap kelompok data dengan no kepadatan noise yang berbeda diperoleh nilai rata-rata. Nilai PSNR untuk gambar indoor dengan kepadatan noise 80% sebesar 12.34783db, pada kepadatan noise 50% sebesar

16.43109db, pada kepadatan noise 30% sebesar 21.01028db. Nilai PSNR jatuh dibawah 30 dB mengindikasikan qualitas yang relative rendah, dimana distorsi yang dikarenakan penyisipan terlihat jelas. Akan tetapi kualitas stego-image yang tinggi berada pada nilai 40dB dan diatasnya (Cheddad, 2010). Pada penelitian ini penerapan *alpha trimmed mean* ini untuk kasus penghapusan noise bekerja tidak maksimal dengan kualitas relative rendah untuk gambar output yang dihasilkan. Hal ini terjadi karena window kernel gambar yang di pakai hanya terbatas pada ukuran 3x3. Dalam penelitian ini ukuran window tidak dapat diperbesar karena keterbatasan sumberdaya computer yang digunakan. Ketika di perbesar ukuran window kernel ,maka computer tidak menjalankan program dan memunculkan jendela dengan pesan " *Display driver stopped responding and has recovered*".Hal tersebut disebabkan oleh Windows service bernama Timeout Detection And Recovery (TDR) TDR diperkenalkan sejak perilisan Windows Vista dan hadir pula dalam Windows 7 dan Windows 8. Fungsi TDR adalah untuk mendeteksi momen hang. Begitu momen itu terdeteksi, TDR akan otomatis berusaha untuk menyelesaikan masalah itu tanpa memaksa pengguna untuk reboot. Dalam kasus VGA card, saat display hang, TDR akan memanggil fungsi driver miniport bernama DxgkDdiResetFromTimeout untuk mereboot driver dan mereset Graphic Processing Unit (GPU).

4.5. Integrasi Penelitian dengan Al-Qur'an

Pada penelitian ini,diterapkan teknik komputasi pararel dimana penggunaan simultan dari beberapa sumber komputasi untuk memecahkan

masalah komputasi. Metode yang di gunakan yaitu *alpha trimmed mean* untuk penghilangan *noise* sebuah citra digital yang dikerjakan secara pararel dengan menerapkan komputasi pararel.

Teknik Linear filter maupun non linear filter telah digunakan beberapa tahun kebelakang untuk penghilangan noise. Kedua teknik ini terus dikembangkan untuk menangani keadaan kepadatan noise yang tinggi dan besarnya ukuran window. Dari abu Hurairah radhiyallahu ‘anhu, dia bercerita bahwa Rasulullah bersabda :

مأأنزلاللهداءإلاأنزللهدواء

“*Tidaklah Allah menurunkan suatu penyakit, melainkan Dia turunkan obat untuknya.” (HR. Ibn Majah & dishahihkan al-Albani*)

Dari hadis diatas jelas bahwasana dalam image untuk segala jenis noise pasti ada teknik penghapusan noise untuk menaganinya.Algorithma yang diusulkan pada penelitian ini adalah Alpha Trimmed mean yang mana merupakan filer jenis non-linear, *yang* menghitung nilai rata-rata yang menghilangkan persentase kecil dari nilai terbesar dan terkecil sebelum menghitung mean .selain itu metode yang digunakan diimplementasikan pada pemrograman pararel. Alloh berfirman :

سُبْحٰنَالَّذِيْخَلَقَالْاَزْوَاجَكُلَّهَامِمَّاتُنْۢبِتُالْاَرْضُوَمِنْاَنْفُسِهِمْوَمِمَّالَايَعْلَمُوْنَ

“Maha suci Tuhan yang telah menciptakan pasangan –pasangan semuanya ,baik dari apa yang ditumbuhkan oleh bumi dan dari diri mereka ataupun dari dari apa yang tidak mereka ketahui “(QS.Yasin :36) .

Dalam kitab tafsir Al-Misbah ,Quraish shihab berpendapat:

> *Mahasuci Allah yang telah menciptakan segala sesuatu secara berpasangan--jantan dan betina--baik dalam dunia tumbuh-tumbuhan,*

> *diri mereka sendiri dan hal-hal yang tidak diketahui oleh manusia(1). (1) Kata "min" dalam ayat ini berfungsi sebagai penjelas. Yakni, bahwa Allah telah menciptakan pejantan dan betina pada semua makhluk ciptaan-Nya, baik berupa tumbuh-tumuhan, hewan, manusia dan makhluk hidup lainnya yang tak kasat mata dan belum diketahui manusia.*

Dalam islam dikenal istilah amal jama`i yaitu amalan yang dilakukan bersama orang lain.hal mendasar pada amal jama`i adalah berpasang-pasangannya mahluk sebagaimana dijelaskan dalam QS.Yasin 36.Mahluk di ciptakan sebagai mahluk sosial yang hidup berdampingan dengan mahluk lain dan mereka saling membutuhkan. Dengan amal jama`i ini bisa diterapkan pada segala hal untuk memperoleh kekuatan yang dahsyat .Contoh pada science, jika proton dipertemukan dengan electron akan menghasilkan kekuatan listrik yang besar.sperti itu juga bahwa pada teknologi ,otak dari computer CPU jika dipertemukan dengan jenis lain GPU untuk melaksanakn tugas komputasi dalam memecahkan masalah yang besar akan meningkatkan efisiensi waktu dilihat dari bidang komputasi.Teknologi baru ini dikenal dengan komputasi pararel.

Komputasi pararel membuat program maupun proses berjalan lebih cepat karena memiliki lebih banyak sumber komputasi yang digunakan dalam mengeksekusi perintah dibandingkan pada komputasi skuensial atau serial . komputasi pararel yang diterapkan mengintegrasikan antara kerja CPU dan GPU yang dapat meningkatkan efisiensi dalam memecahkan masalah noise pada citra. Dalam ajaran islam, efisien diartikan sebagai sesuatu yang bermanfaat, sesuatu yang tidak berlebihan, serta tidak mubadzir, sebagaimana Allah swt. Berfirman dalam kitab suci AL-Qur`an seperti berikut:

وَهُوَ الَّذِي أَنشَأَ جَنَّاتٍ مَّعْرُوشَاتٍ وَغَيْرَ مَعْرُوشَاتٍ وَالنَّخْلَ وَالزَّرْعَ مُخْتَلِفًا أُكُلُهُ وَالزَّيْتُونَ وَالرُّمَّانَ مُتَشَابِهًا وَغَيْرَ مُتَشَابِهٍ كُلُوا مِن ثَمَرِهِ إِذَا أَثْمَرَ وَآتُوا حَقَّهُ يَوْمَ حَصَادِهِ وَلَا تُسْرِفُوا إِنَّهُ لَا يُحِبُّ الْمُسْرِفِينَ

"Dan Dialah yang menjadikan kebun-kebun yang berjunjung dan yang tidak berjunjung, pohon korma, tanam-tanaman yang bermacam-macam buahnya, zaitun dan delima yang serupa (bentuk dan warnanya) dan tidak sama (rasanya). Makanlah dari buahnya (yang bermacam-macam itu) bila dia berbuah, dan tunaikanlah haknya di hari memetik hasilnya (dengan disedekahkan kepada fakir miskin); dan janganlah kamu berlebih-lebihan.Sesungguhnya Allah tidak menyukai orang yang berlebih-lebihan."(Q.S. Al An'aam: 141).

Dalam kitab tafsir Al-Misbah ,Quraish shihab berpendapat:

> *Hanya Allahlah yang menciptakan berbagai kebun. Ada yang ditanam dan disanggah tiang, ada pula yang tidak. Allah menciptakan pula pohon korma dan tanaman-tanaman lain yang menghasilkan buah- buahan dengan berbagai warna, rasa, bentuk dan aroma yang berbeda-beda. Juga, Allah menciptakan buah zaitun dan delima yang serupa dalam beberapa segi, tetapi berbeda dari beberapa segi lain. Padahal, itu semua tumbuh di atas tanah yang sama dan disiram dengan air yang sama pula. Makanlah buahnya yang baik dan keluarkan zakatnya saat buah-buah itu masak. Namun, janganlah kalian berlebih-lebihan dalam memakan buah-buahan itu, sebab hal itu akan membahayakan diri sendiri dan akan mengurangi hak orang miskin. Allah tidak akan memberi perkenan atas perbuatan orang-orang yang berlebih-lebihan.*

Dari ayat tersebut dijelaskan bahwa Allah menciptakan alam semesta dan lingkungannya memiliki arti dan fungsi namun sejauh mana manusia mampu mencari tahu tentang kebesaran Allah tersebut. Dalam perancangan ini mencoba memberikan suatu wadah kepada manusia/masyarakat untuk mengetahui lebih dalam tentang komputasi pararel yang sedang menjadi obyek yang dikembangkan dalam dunia teknologi saat ini .diharapkan umat manusia bisa mengambil manfaat, menggali kemudian mengelolanya untuk kesejahteraan manusia serta sebagai tindakan kita beramal sholeh.

BAB V

PENUTUP

5.1. Kesimpulan

Berdasarkan program yang telah dibuat dan hasil uji coba yang telah dilakukan, maka dapat ditarik kesimpulan bahwa:

1. Pemrograman pararel yang di implementasikan menggunakan platform OpenCL mempekerjakan prosesor dari CPU dan GPU,sehingga mampu berjalan lebih cepat dalam menyelesaikan masalah penghapusan noise menguunakan *alpha trimmed mean*dibandingkan dikerjakan pada CPU saja menggunakan platform JAVA.
2. Implementasi metode *alpha trimmed mean* dalam penghapusan noise memberikan hasil cukup baikdalam hal efisiensi pada gambar yang memiliki noise density rendah sebesar 30% dengan waktu yg dibutuhkan sebesar 284.4992 ms gambar indoor bernoise 30% dan memberikan hasil relative rendah dalam hal efisiensi dilihat dari nilai PNSR tertinggi yang diperoleh sebesar 21.01028 db pada gambar indoor bernoise 30%.

5.2. Saran

Peneniti mengharapkan ada pengembangan selanjutnya untuk teknologi bidangkomputasi pararel dengan melakukan eksperimen lebih jauh pada bidang bidang lain menggunakan sumber daya computer yang lebih tinggi.Komputasi pararel masih menjadi hal awam dan belum menyeluruh digunakan karena

sedikitnya orang yang mau mencoba hal tersebut sehingga mengharapkan banyak penelitian dilakukan untuk komputasi pararel.

DAFTAR PUSTAKA

Arroyo marcelo.2013 *.teaching parallel and distributed computing to undergraduate computer science students*. Ieee 27th international symposium on parallel & distributed processing workshops and phd forum
.

Boreskov alexey,shikin avgeny.2014.Computer Graphics: From Pixel To Programmable grapichs hardware.London new york:crc press.

Cheddad, A., Condell, J., Curran, K., Kevitt, P.Mc., 2010. Digital Image Steganography : Survey and Analysis of Current Methods. Signal Processing, Elsevier. Northern Ireland, UK.

Dash arabinda,kumar sujaya.2015.*High Density Noise Removing by Using Cascading Algorithms*.IEEE.

Faruk ahmed and swagatam das.2014. *Removal Of High-Density Salt-And-Pepper Noise In Images With An Iterative Adaptive Fuzzy Filter Using Alpha-Trimmed Mean.* Ieee transactions on fuzzy systems, vol. 22, no. 5.

Fu bin ,et al ,.2011. *An Efficient Mean Filter Algorithm*.IEEE

G sanchez maria,and friends .2014.*Image Noise Removal On Heterogeneous Cpu-Gpu Configurations*. Volume 29 iccs .elsevier b.v.

Goycoolea raul.*Parallel Multi Processing Architecture & Programming Techniques*. Www.slideshare.net.diunggah 16 februari 2012. Diakses 8 mei 2016.

Hai chi, fuchun sun, qing li and xiaoxiao wu . 2006. *A New Adaptive Alpha-Trimmed Mean Filtering Approach**.IEEE.Proceedings of the 6th world congress on intelligent control and automation.

Hestiningsih, idhawati. 2008. Pengolahan citra digital, elex media komputindo, jakarta.

Https://computing.llnl.gov/tutorials/parallel_com.diakses tanggal 16 desember 2015.

Homepages.cae.wisc.edu/~ece533/images.Diakses tanggal 11 januari 2016

Homepages.inf.ed.ac.uk/rbf/hipr2/mean.html.Diakses tanggal 22 desember 2015

Jie shen, jianbin fang, henk sips, and ana lucia varbanescu .2013. *Performance Traps In Opencl For Cpus.* 21st euromicro international conference on parallel, distributed, and network-based processing. 2012 ieee.

Li jie,huang shitan .2008.*Adaptive Salt And Pepper Noise Removal:A Function Level Evolution Based Approach*.IEEE.

J. A. Fraire, p. Ferreyra and c. Marques.2013.*Opencl Overview, Implementation, And Performance Comparison*. IEEE latin america transactions, vol. 11.

Kazuya matsumoto, naohito nakasato, and stanislav g. Sedukhin.2012.*performance tuning of matrix multiplication in opencl on different gpus and cpus*.ieee computer society

Khronos.org/registry/cl/sdk/1.0/docs.Diakses tanggal 22 desember 2015.

Dr. Mendelson .avi .2013. *Foreword to The First Edition Of Heterogeneous Computing With Opencl*. Usa: elsevier inc.

Mahdi shaneh, and arash golibagh mahyari.2011.*Image Enhancement Using A-Trimmed Mean E-Filters. I*nternational science index, electrical and computer engineering vol:5, no:11.

Pattnaik ashutosh,and friends.2012.*A New And Eficient Mthod For Removal On High Density Salt And Pepper Noise Throught Cascade Decision Based Filtering Alghorithm*.Icccs-2012.elsevier ltd.

Priyanka shrivastaval et al.2014. *Noise Removal Using First Order Neighborhood Mean Filter* . IEEE

Rafiqul zaman khan and md firoj ali.2014.*Current Trends In Parallel Computing.* IEEE transactions on fuzzy systems, vol. 22, no. 5..

Robert dietrich and ronny tsch¨uter .2015. *A Generic Infrastructure For Opencl Performance Analysis.* The 8th ieee international conference on intelligent data acquisition and advanced computing systems: technology and application.

Remzy oten.2004.*Adaptive Alpha-Trimmed Mean Filter Deviation From Assumed Noise Model.* IEEE transactions on image processing, vol. 13, no. 5.

Sulistyo, wiwin. Bech, yos richard. Frans, filipus y. 2009. *Analisis Penerapan Metode Median Filter Untuk Mengurangi Noise Pada Citra Digital*. Konferensi nasional sistem dan informatika; bali, november 14, 2009 kns&i09-035: hal. 189-195.

Sutoyo. T. Et al. 2009. Teori pengolahan citra digital, yogyakarta: penerbit andi..

Vasanth k,and friends.2015. *A Decision Based Unsymmetrical Trimmed Modified Winsorized Mean Filter For The Removal Of High Density Salt And Pepper Noise In Image And Video*.Imcip-2015.elsevier ltd.

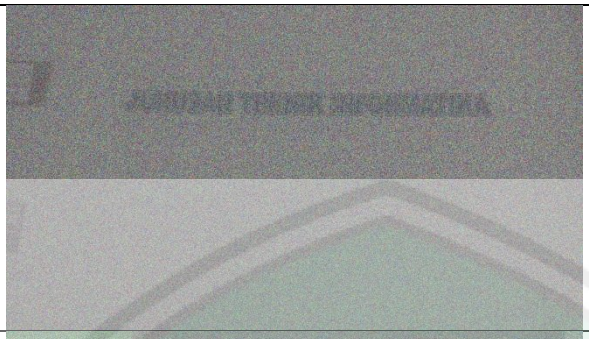
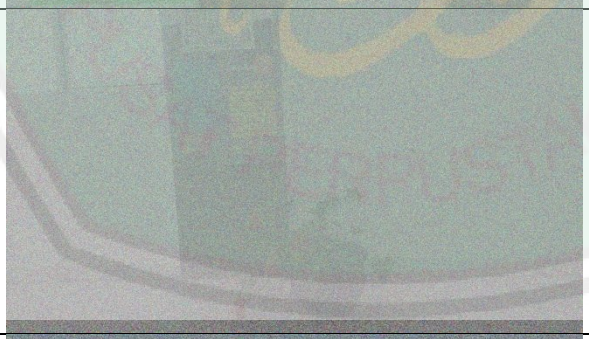
Wang lipeng,wang hanbin .2011.*Implementation Of A Soft Morphological Filer Based On Gpu Framework*.IEEE

Wiwin sulistyo ,dkk.2009.*Analisis Penerapan Metode Median Filter Untuk Mengurangi Noise Pada Citra Digital*. Konferensi nasional sistem dan informatika 2009; bali.

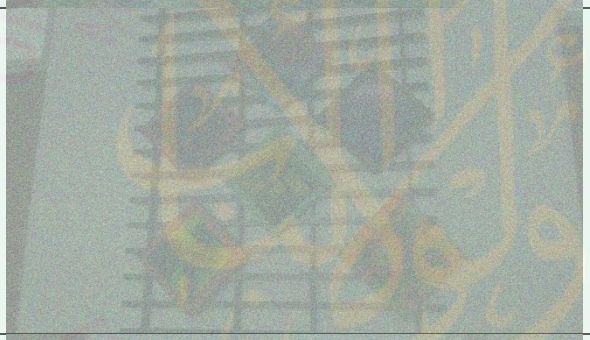
Zaman rafiqul khan and md firoj ali.2012.*Current Trends in Parallel Computing. International Journal Of Computer Applications*. Volume 59–no.2.

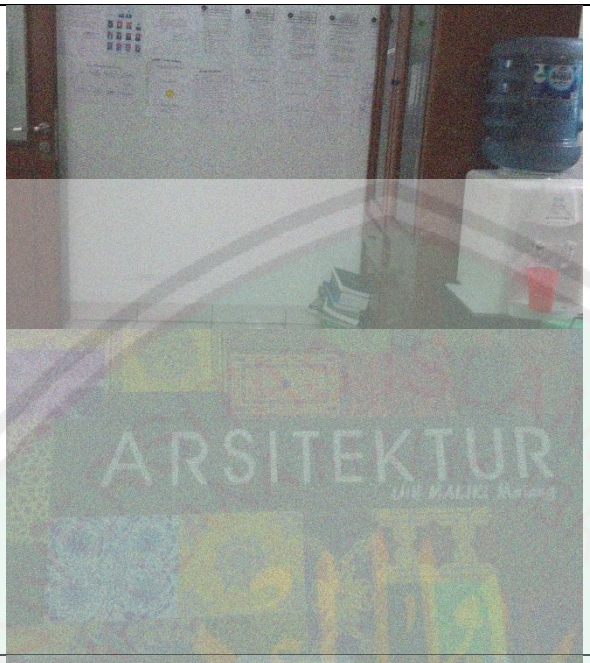
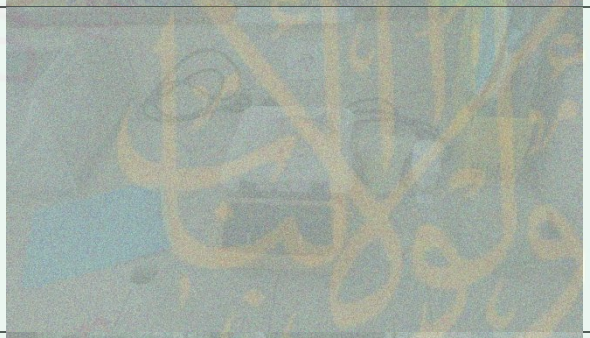
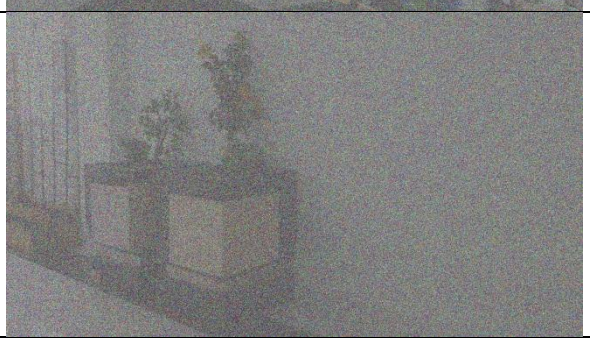
DAFTAR LAMPIRAN DATA UJI COBA

No	Data	Nama Data	Keterangan
1		high1	gambar indoor dengan noise density rendahsebesar 30%
2		high2	
3		high3	
4		high4	

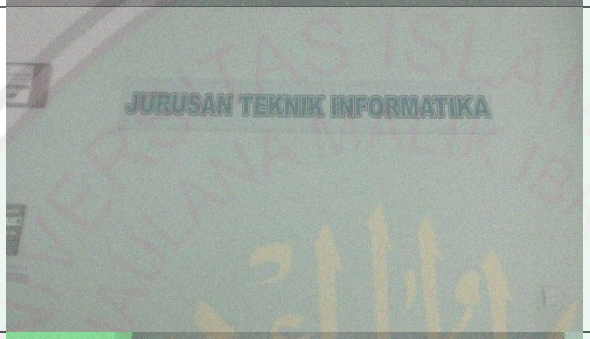
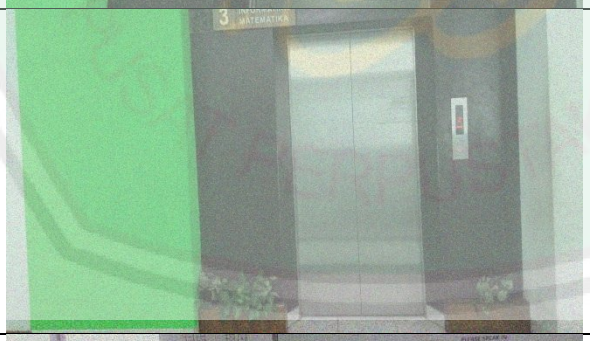
5		high5	
6		high6	
7		high7	
8		high8	
9		high9	

10		high10	
11		high11	
12		high12	
13		High13	
14		high14	

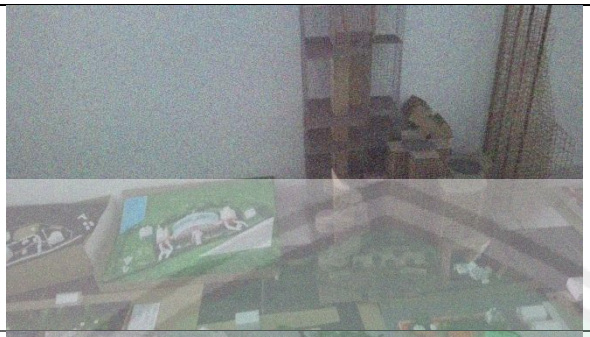
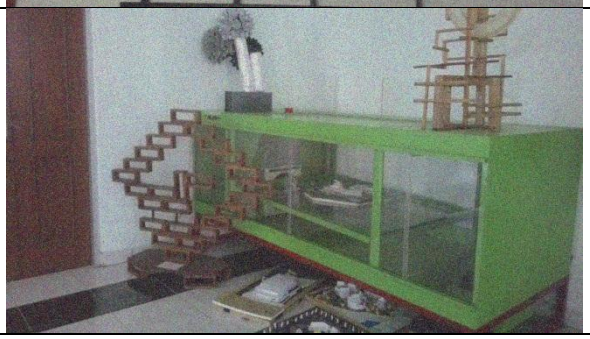
			
15		high15	
16		high16	
17		high17	
18		high18	
19		high19	

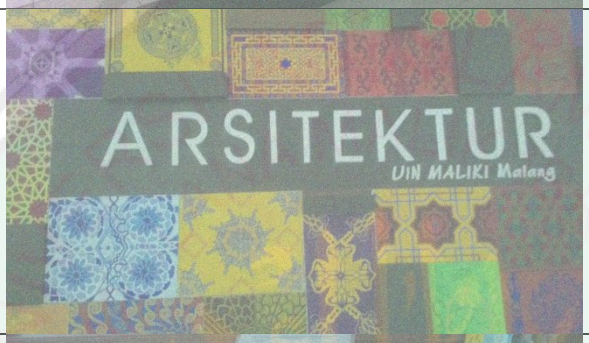
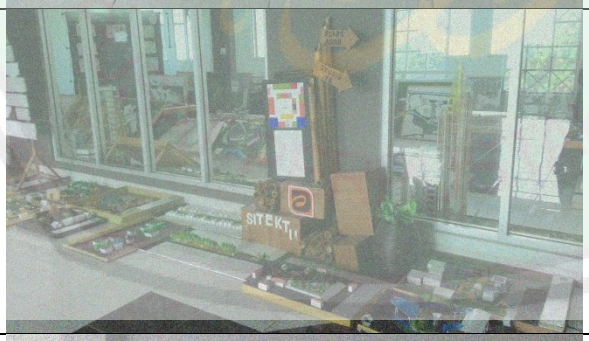
20		high20	
21		high21	
22		high22	
23		high23	

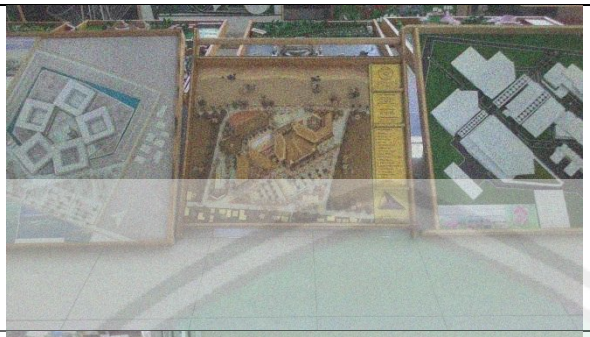
24		high24	
25		high25	
26		low1	gambar indoor dengan noise density sedangsebesar 30%
27		low2	
28		low3	

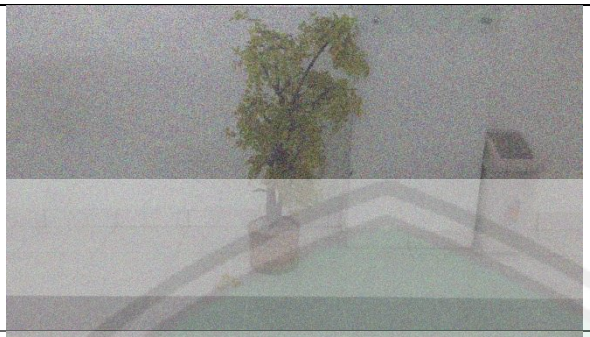
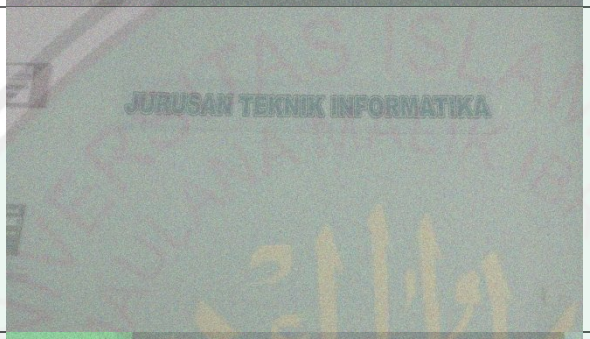
29		low4	
30		low5	
31		low6	
32		low7	
33		low8	

34		low9	
35		low10	
36		low11	
37		low12	
38		low13	

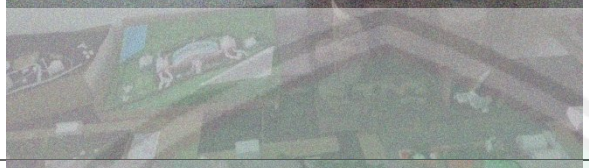
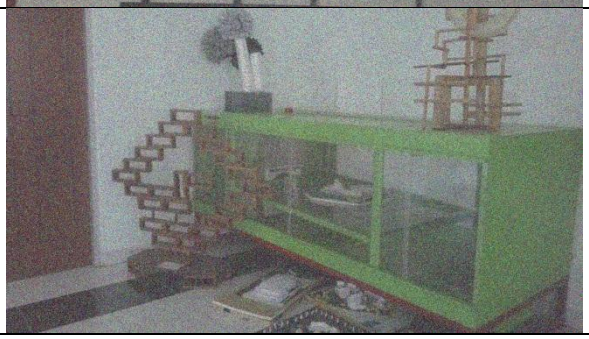
39		low14	
40		low15	
41		low16	
42		low17	
43		low18	

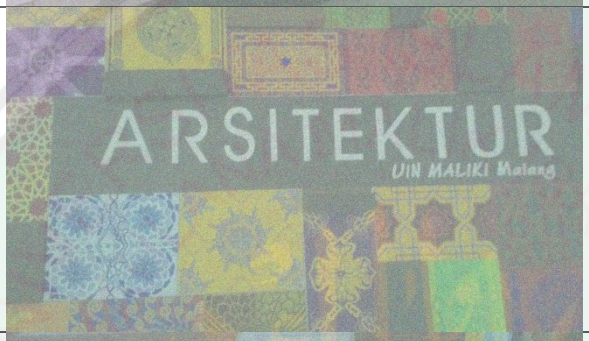
44		low19	
45		low20	
46		low21	
47		low22	
48		low23	

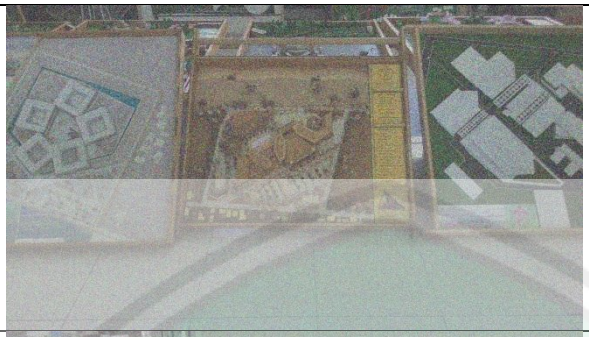
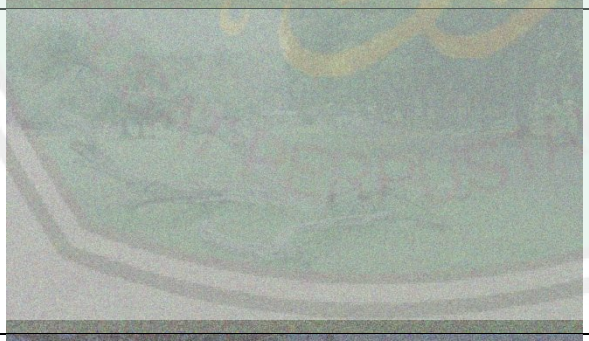
49		low24	
50		low25	
51		mid1	
52		mid2	
53		mid3	

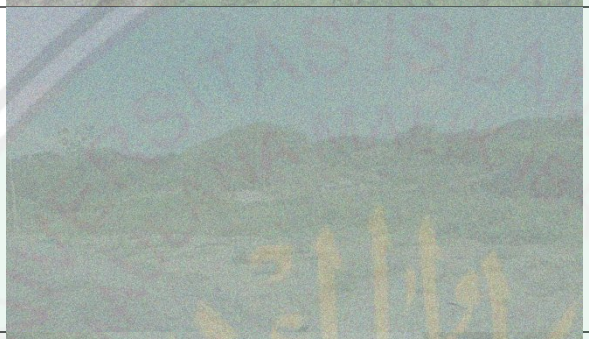
54		mid4	
55		mid5	
56		mid6	
57		mid7	
58		mid8	

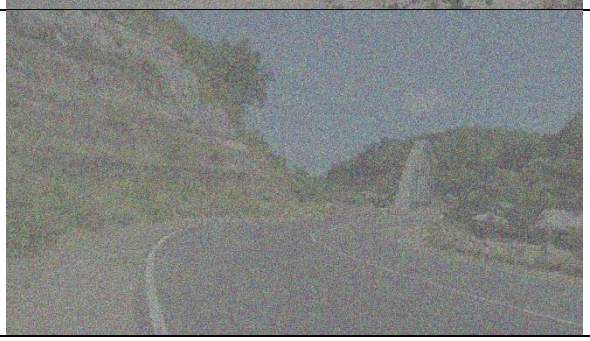
59		mid9	
60		mid10	
61		mid11	
62		mid12	
63		mid13	

			
64		mid14	
65		mid15	
66		mid16	
67		mid17	
68		mid18	

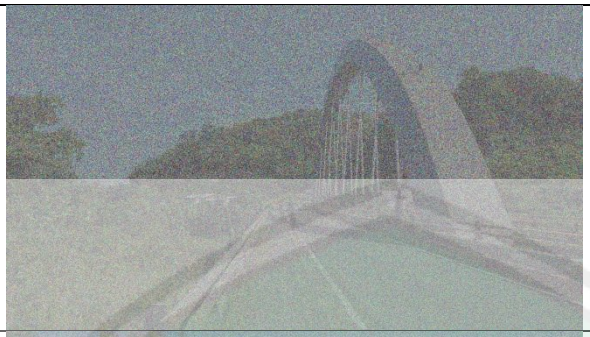
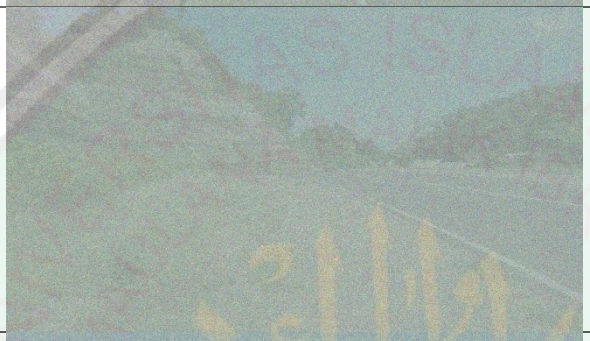
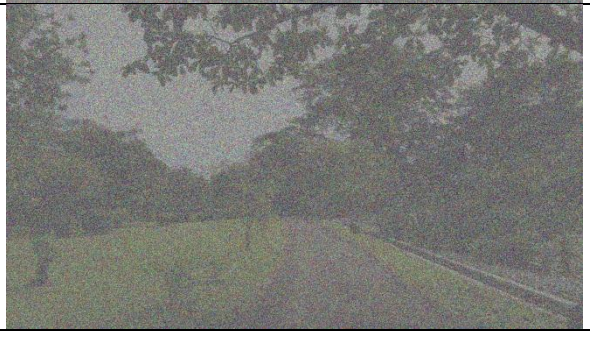
69		mid19	
70		mid20	
71		mid21	
72		mid22	
73		mid23	

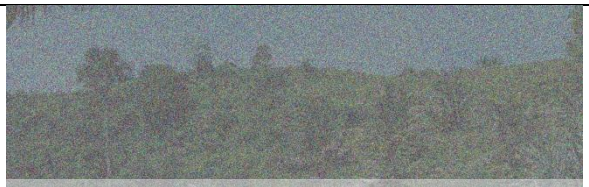
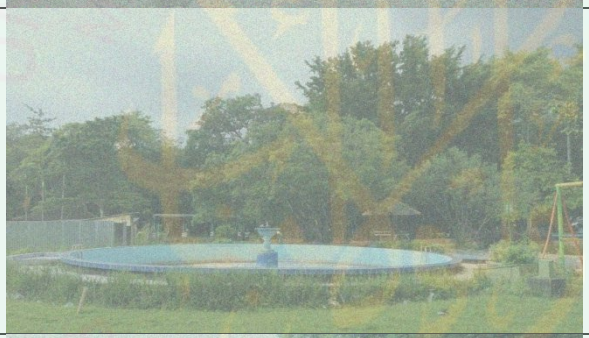
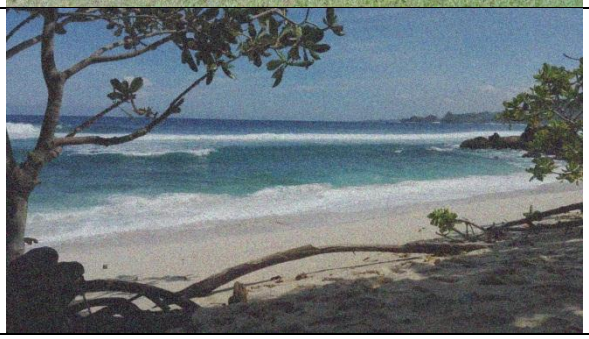
74		mid24	
75		mid25	
76		high1	
77		high2	
78		high3	gambar outdoor dengan noise density rendahsebesar 30%

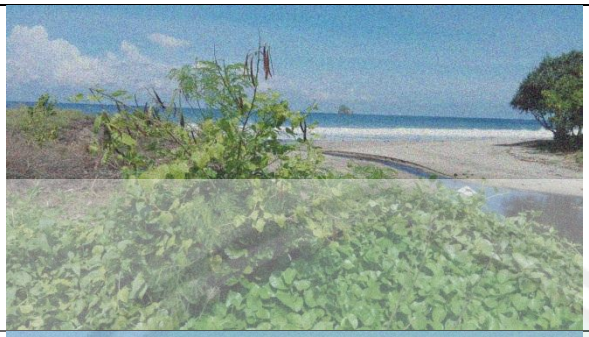
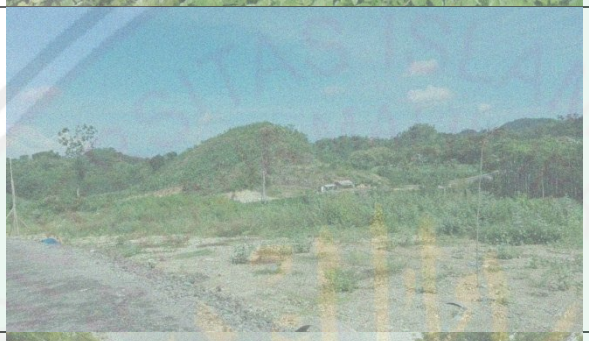
79		high4	
80		high5	
81		high6	
82		high7	
83		high8	

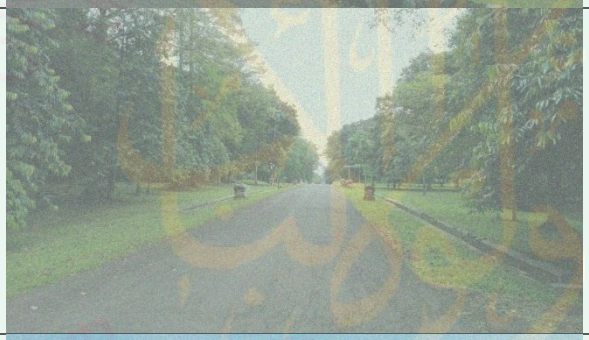
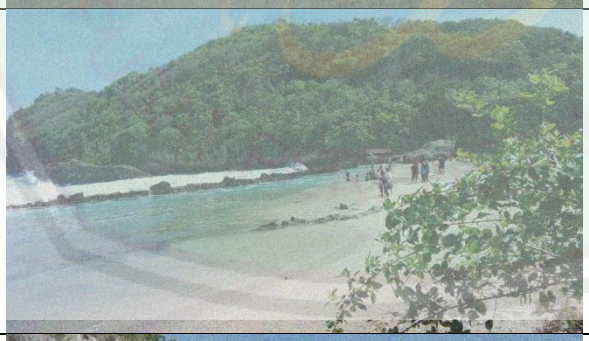
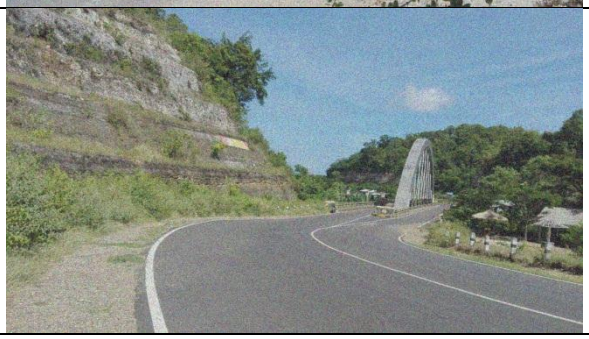
			
84		high9	
85		high10	
86		high11	
87		high12	
88		High13	

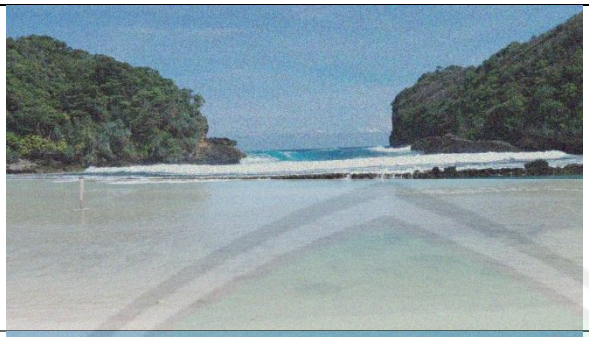
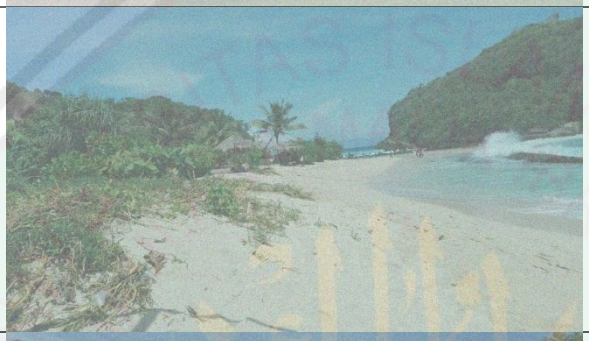
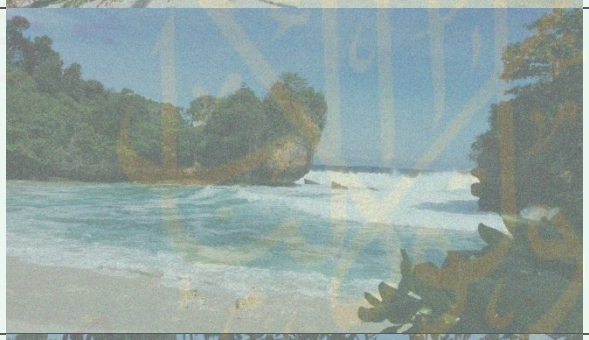
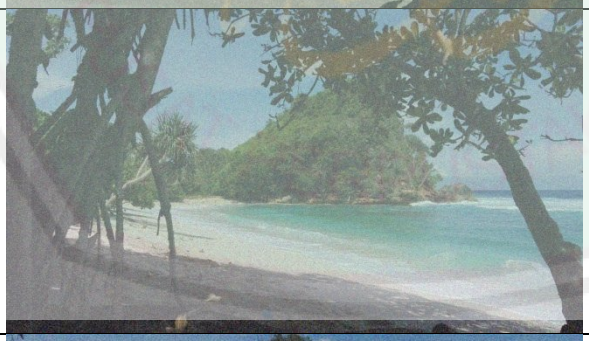
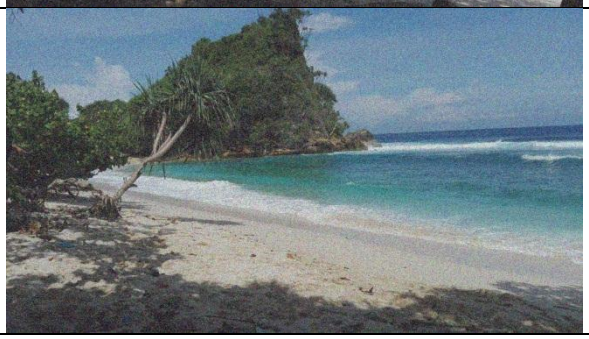
89		high14	
90		high15	
91		high16	
92		high17	
93		high18	

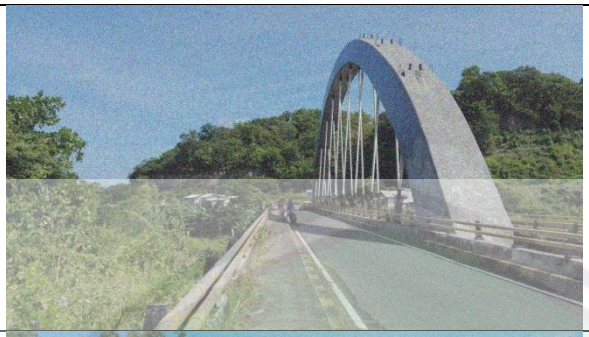
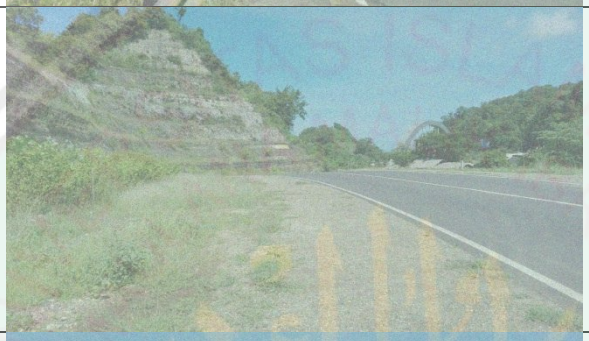
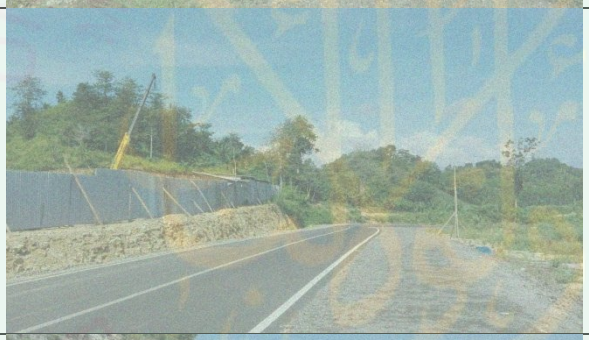
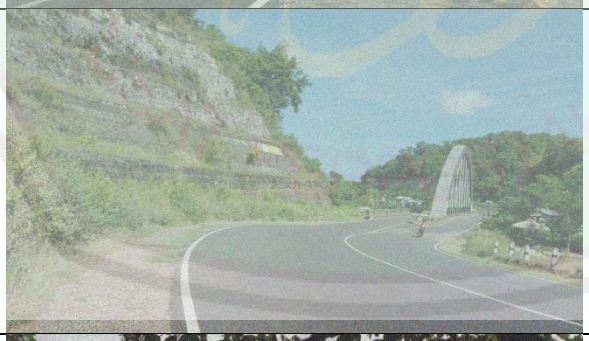
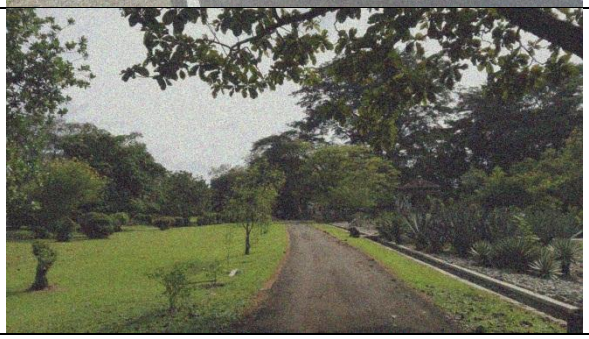
94		high19	
95		high20	
96		high21	
97		high22	
98		high23	

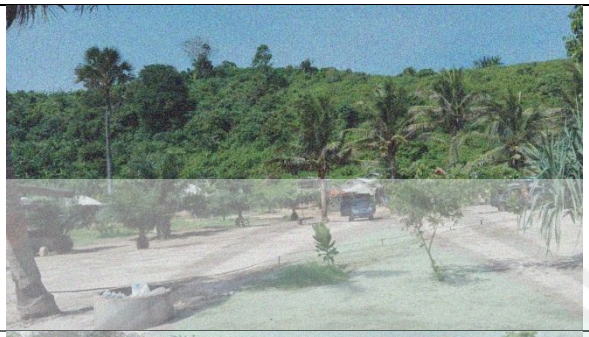
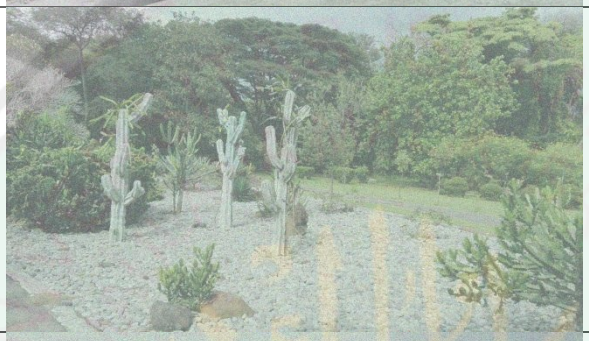
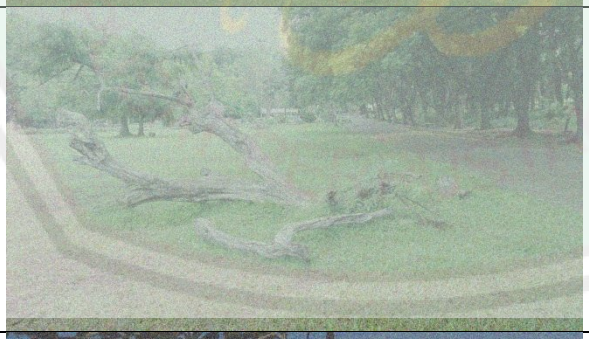
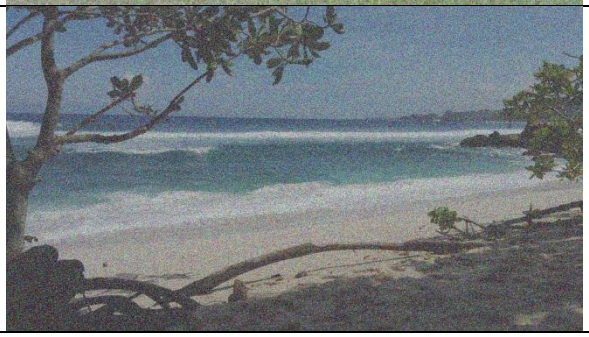
99		high24	
100		high25	
101		low1	
102		low2	
103		low3	gambar outdoor dengan noise density sedangsebesar 30%

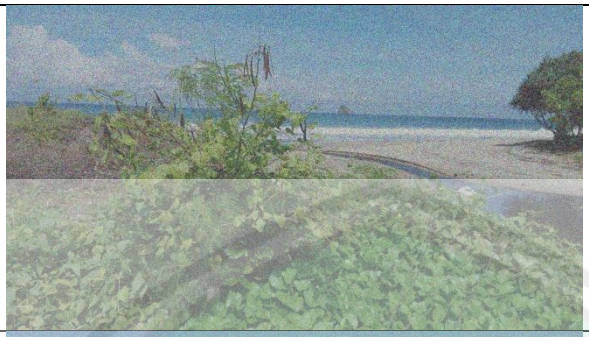
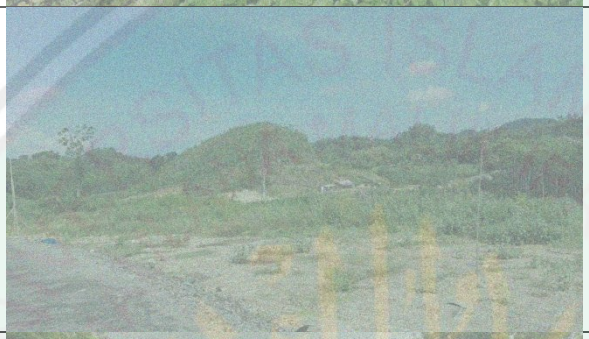
104		low4
105		low5
106		low6
107		low7
108		low8

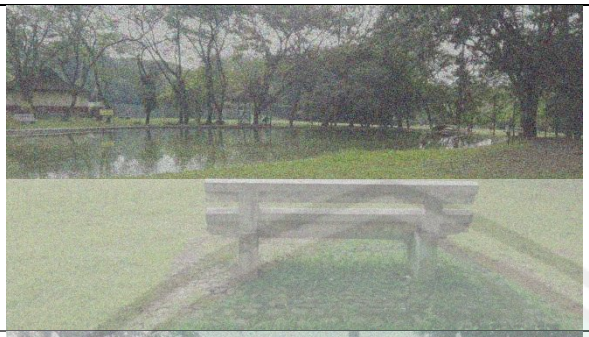
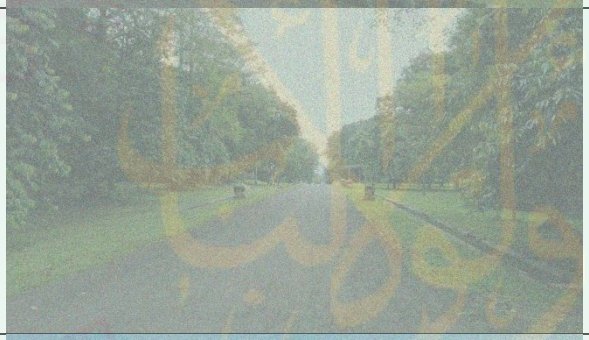
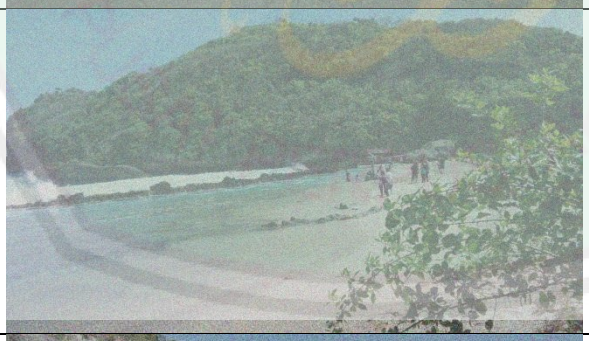
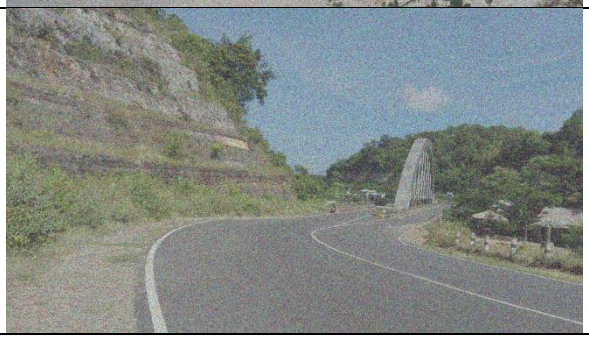
109		low9	
110		low10	
111		low11	
112		low12	
113		low13	

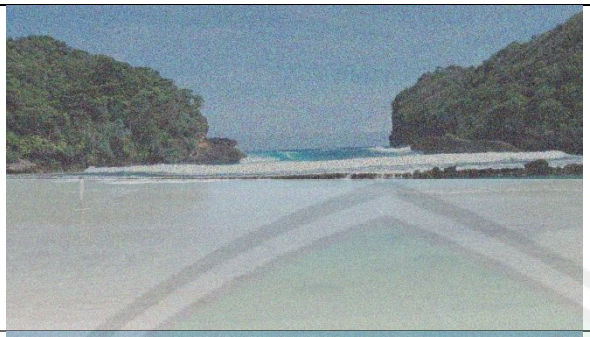
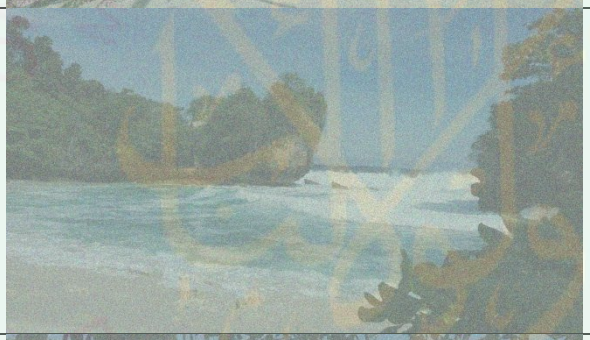
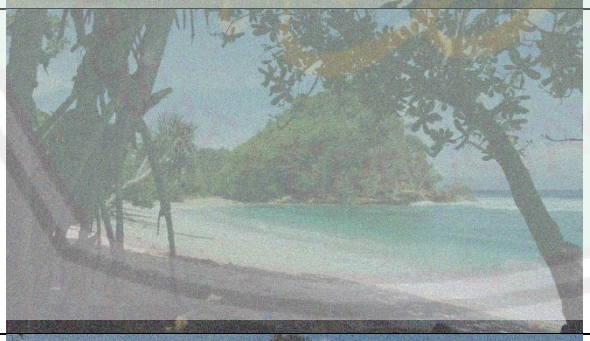
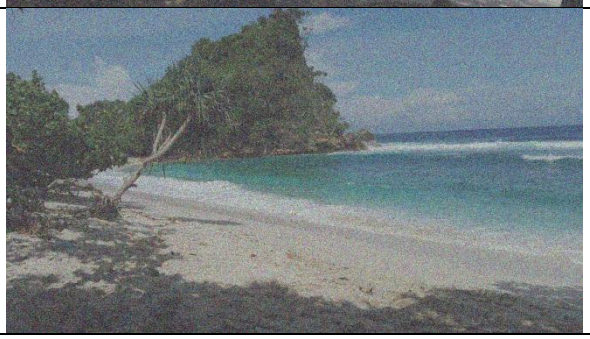
114		low14	
115		low15	
116		low16	
117		low17	
118		low18	

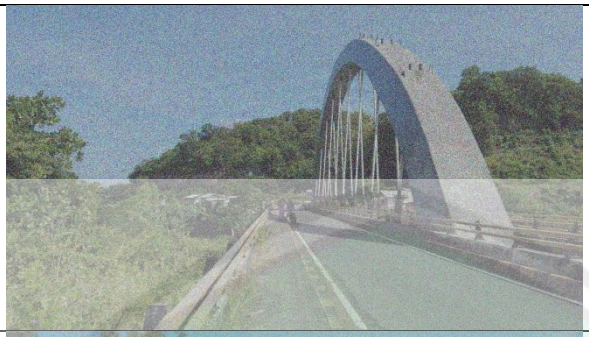
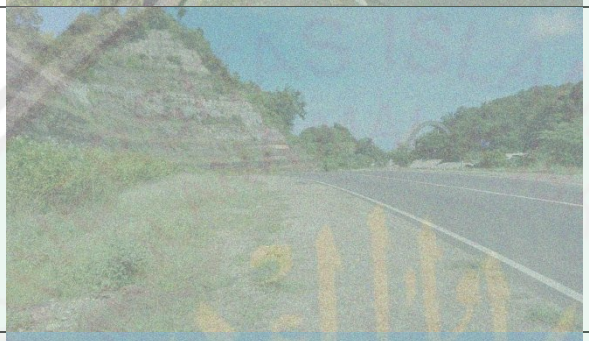
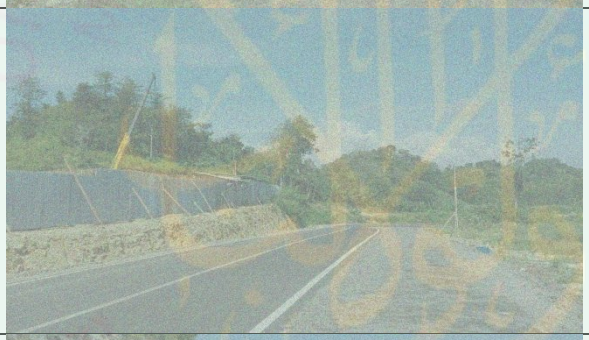
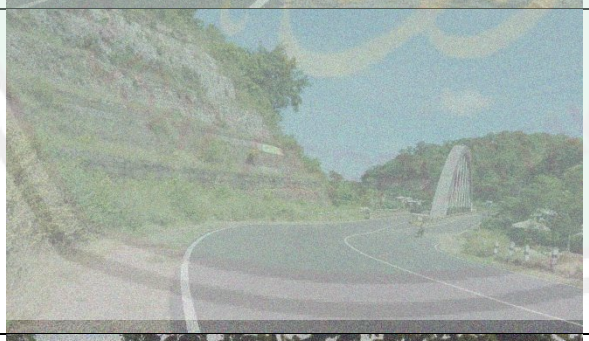
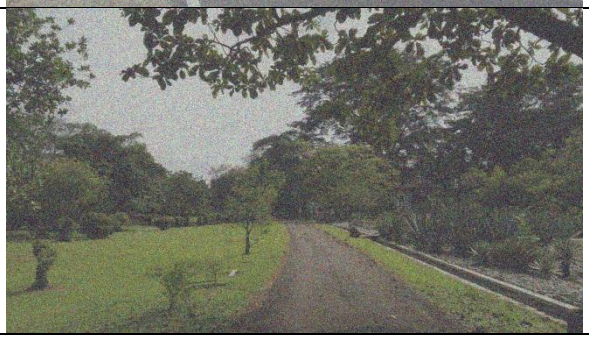
119		low19	
120		low20	
121		low21	
122		low22	
123		low23	

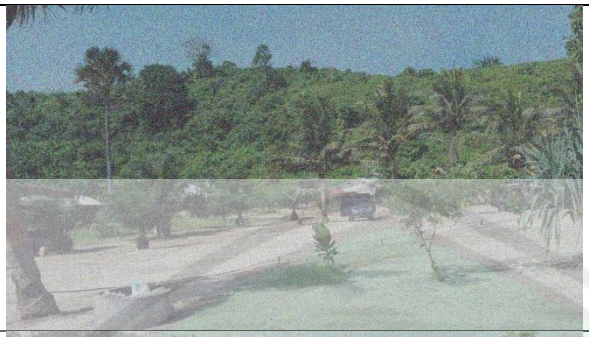
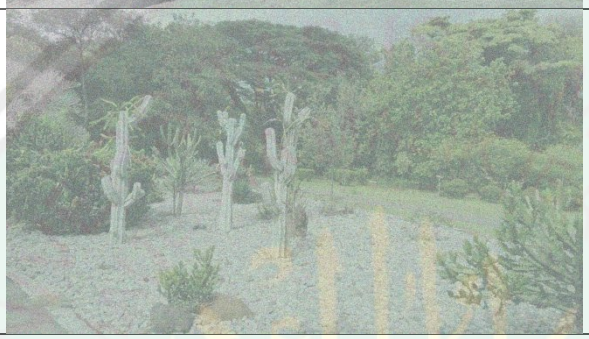
124		low24	
125		low25	
126		mid1	
127		mid2	
128		mid3	gambar outdoor dengan noise density tinggisebesar 80%

129		mid4	
130		mid5	
131		mid6	
132		mid7	
133		mid8	

134		mid9	
135		mid10	
136		mid11	
137		mid12	
138		mid13	

139		mid14	
140		mid15	
141		mid16	
142		mid17	
143		mid18	

144		mid19	
145		mid20	
146		mid21	
147		mid22	
148		mid23	

149		mid24	
150		mid25	