BAB II
TINJAUAN PUSTAKA

A. Tinjauan Pustaka

1. Inpainting Citra Digital

Pada tahun 1976, Emile-Male, menulis buku yang menerangkan tentang teknik restorasi lukisan-lukisan kuno yang mengalami kerusakan(Emile-Male, 1976). Hal ini nantinya menjadi salah satu dasar dari berkembangnya *inpainting* citra digital. Setelah sembilan tahun berselang, tepatnya 1985, Sarah Walden membahas topik yang sama yaitu tentang cara mengembalikan lukisan ke kondisi yang menyerupai keadaan asalnya tanpa merusak kondisi lukisan (Walden, 1985).

Pada tahun 1984, Geman dan Geman, mulai melakukan teknik pengembalian citra digital dengan menggunakan metode statistik dari fungsi biaya yang dikombinasikan dengan *simulated annealing* untuk mencari nilai maksimum yang digunakan untuk mengisi citra yang rusak(Geman & Geman, 1984). Pada tahun 2000, digunakan model *inpainting* menggunakan non-linier PDP untuk melakukan restorasi citra digital. Pada tahun tersebut istilah *inpainting* mulai dikenalkan oleh Bertalmio (Bertalmio et al., 2000).

Chan dan Shen, 2001, setahun kemudian setelah *inpainting* dikenalkan, melakukan pendekatan yang berbeda untuk *inpainting*. Mereka mengenalkan variasi dari model *image denoising* dan segmentasi yang diadaptasi untuk melakukan tugas *inpainting*(Chan & Shen, 2001). Tahun 2002, Kang, Chan, dan Shen (Chan et al., 2002) mengenalkan variasi baru model *inpainting* citra yang berasal dari variasi dari Chan dan Shen(Chan & Shen, 2001). Hasil dari penelitian ini proses *inpainting* dapat diselesaikan sedikit lebih cepat. Namun

variasi ini mengalami kelemahan yaitu tidak dapat melakukan rekonstruksi citra bila daerah kerusakan *inpainting* cukup lebar.

Pada tahun yang sama, 2002, Esedoglu dan Shen(Esedoglu & Shen, 2002) mengenalkan model *inpainting* dengan mengembangkan kemampuan dari model *inpainting* Mumford-Shah(Mumford & Shah, 1989). Hal ini menghasilkan persamaan *gradient descent* orde keempat, PDP parabolik non-linear. Penemuan ini kemudian menjadi ide dari penerapan *inpainting* dengan persamaan Cahn-Hilliard. Pada tahun 2006, Allan Gillete mengenalkan image *inpainting* dengan persamaan Cahn-Hilliard yang dimodifikasi. Persamaan ini mengurangi waktu komputasi dengan metode *inpainting* lainnya. Kemampuan numerik yang cepat membuat persamaan Cahn-Hilliard dapat digunakan untuk perhitungan data yang besar dan mengurangi waktu komputasi(Gillette, 2006) (Bertozzi et al., 2007). Pada tahun 2009, Scho□nlieb, melakukan percobaan membandingkan algoritma Cahn-Hilliad dan TV-H^{-1} untuk proses *inpainting* citra. Dari perbandingan tersebut dikemukakan bahwa persamaan orde tinggi memberikan hasil yang lebih halus dan menyatu pada daerah kerusakan yang besar. Salah satu hasil dari percobaan itu beliau menekankan pentingnya dilakukan percepatan *inpainting* secara numerik(Scho□nlieb, 2009).

2. Pemrosesan Paralel pada GPU

Tahun 1965, Gordon E. Moore, mengatakan bahwa pertumbuhan kecepatan perhitungan mikroprosesor mengikuti rumusan eksponensial(Gelsinger, 2006). Hal ini mendorong desain *single processor* untuk selalu meningkatkan *clock speed* dari tahun ke tahun. Akibatnya prosesor membutuhkan power yang lebih banyak, akan tetapi peningkatan power menimbulkan permasalahan *overheating*. Karena keinginan untuk meningkatkan kecepatan selalu diharapkan, maka desainer chip mulai mencoba menerapkan *multicore processor* dan pemrograman secara paralel. Model tersebut membutuhkan power yang lebih

sedikit, sehingga mengurangi potensi *overheating*. Hal ini menjadi dasar dari paralel processing pada GPU (Blake et al., 2009).

Pada awal 2001, NVIDIA mengenalkan GPU (GeForce3) yang dapat di program untuk pertama kalinya. Dua tahun kemudian, 2003 di San Diego, diselenggarakan *workshop* Siggraph (Eurographics Graphics Hardware) yang menunjukkan perpindahan data dari grafik ke data yang tidak menggunakan grafik pada GPU. Hal ini kemudian memunculkan konsep GPGPU (*General-Purpose computing on GPU*). Pada awalnya digunakan Brook sebagai bahasa pemrograman pada GPU(Diaz et al., 2012).

Berdasarkan Owens et al, penggunaan GPU non grafik untuk pertama kalinya dilakukan untuk menyelesaikan set persamaan diffrensial yang besar, hal ini dilakukan oleh Kruger et al. pada tahun 2005 untuk melakukan pemodelan partikel (Owens et al., 2008)(Kruger et al., 2005).

Pada bulan November 2006, NVIDIA mengenalkan CUDA untuk melakukan *general purpose paralel computing* dan sebagai model bahasa pemrograman paralel yang dapat melakukan komputasi pada GPU untuk menyelesaikan permasalahan komputasi dengan cara yang lebih efisien dibandingkan CPU (NVIDIA, 2014).

B. Landasan Teori

1. Citra digital

Citra digital adalah suatu citra yang didapat dari citra analog (citra yang bersifat kontinyu) dengan cara pengambilan sampel dan kuantisasi secara digital(Scho□nlieb, 2009). Sebuah citra digital dapat mewakili oleh sebuah matriks yang terdiri dari M kolom N baris, dimana perpotongan antara kolom dan baris disebut piksel (*pixel = picture element*), yaitu elemen terkecil dari sebuah citra. Piksel mempunyai dua parameter, yaitu koordinat dan intensitas atau warna. Nilai yang terdapat pada koordinat (x,y) adalah f(x,y), yaitu besar

intensitas atau warna dari piksel di titik itu. Oleh sebab itu, sebuah citra digital dapat ditulis dalam bentuk matriks berikut:

$$f(x,y) = \begin{matrix} f(0,0) & \dots & f(0,M-1) \\ \dots & \dots & f(1,M-1) \\ f(N-1,0) & f(N-1,1) & f(N-1,M-1) \end{matrix} \quad (2)$$

Berdasarkan gambaran tersebut, secara matematis citra digital dapat dituliskan sebagai fungsi intensitas f(x,y) adalah nilai fungsi pada setiap titik (x,y) yang menyatakan besar intensitas citra atau tingkat keabuan atau warna dari piksel di titik tersebut. Pada proses digitalisasi (sampling dan kuantitas) diperoleh besar baris M dan kolom N hingga citra membentuk matriks M × N dan jumlah tingkat keabuan piksel G (Gonzalez & Woods, 2008).

Citra digital dapat dipandang sebagai fungsi matematika, oleh karena itu kita dapat memberikan operasi matematika kepadanya. Beberapa operasi matematika yang dapat dilakukan pada citra digital adalah metode statistika, operasi *morphological*, dan persamaan diferensial parsial (Scho□nlieb, 2009). Pada tesis ini akan secara spesifik membahas pengolahan citra digital dengan menggunakan operasi persamaan diferensial parsial.

2. PDP (Persamaan Diferensial Parsial)

PDP adalah persamaan yang memuat satu atau lebih turunan parsial dengan dua atau lebih variabel bebas. PDP digunakan untuk melakukan formulasi dan menyelesaikan permasalahan yang melibatkan fungsi-fungsi yang tidak diketahui, yang merupakan bentuk dari beberapa variabel(Evans, 2002). Persamaan ini digunakan untuk merepresentasikan fenomena-fenomena yang terjadi di kehidupan sehari-hari pada interval waktu kontinyu dalam suatu model matematika. Hal ini biasanya digunakan untuk meninjau topik-topik dinamika fluida, pemodelan pada biologi, ilmu bahan, elektromagnetik, pengolahan citra, grafika komputer, desain geometrik dan pengenalan pola(Greer et al., 2006). Bentuk paling sederhana dari persamaan diferensial adalah:

$$F(x, y, \frac{\partial y}{\partial x}, \frac{\partial^2 y}{\partial x^2}, \dots, \frac{\partial^n y}{\partial x^n}) = 0 \quad (3)$$

Orde dari persamaan diferensial adalah orde tertinggi dari turunan dalam persamaan.

PDP menempati bagian utama fisika komputasi karena berbagai gejala penting dalam fisika dapat dinyatakan dalam bentuk tersebut. Bentuk umum persamaan diferensial parsial yang sering ditemukan dalam problema fisika adalah PDP orde dua, yaitu:

$$a_{11} \frac{\partial^2 u}{\partial x^2} + 2a_{12} \frac{\partial^2 u}{\partial x \partial y} + a_{22} \frac{\partial^2 u}{\partial y^2} + f(x, y, u, \frac{\partial y}{\partial x}, \frac{\partial^2 y}{\partial x^2}) = 0 \quad (4)$$

dimana,

a_{11}, a_{12}, a_{22} : koefisien

u : variabel tak bebas

x, y : variabel bebas berupa koordinat ruang dalam sistem koordinat Cartesian

Berdasarkan nilai koefisien-koefisiennya, bentuk umum ini dapat dibedakan atas beberapa bentuk khusus, yang kemudian dikenal sebagai bentuk PDP parabolik, hiperbolik dan eliptik. Persamaan-persamaan ini banyak ditemui pada persamaan transport polutan. Pembagian persamaan diferensial menjadi tiga jenis di atas harus memenuhi syarat-syarat berikut:

1. Jika $a_{11}a_{22} - {a_{12}}^2 > 0$ maka persamaan disebut PDP eliptik. Contohnya adalah persamaan *Laplace*:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \quad (5)$$

Dengan x,y menyatakan koordinat dan waktu t.

2. Jika $a_{11}a_{22} - {a_{12}}^2 = 0$ maka disebut PDP parabolik. Contohnya adalah persamaan *Heat-Equation.*

$$\frac{\partial u}{\partial t} - \alpha \nabla^2 u = 0 \quad (6)$$

Dengan t adalah waktu, α adalah koefisien difusi panas dan ∇^2 adalah operator Laplace.

3. Jika$a_{11}a_{22} - {a_{12}}^2 < 0$ maka persamaan disebut PDP hiperbolik. Contohnya adalah persamaan gelombang:

$$\frac{\partial^2 A}{\partial x^2} - C\frac{\partial^2 A}{\partial t^2} = 0 \quad (7)$$

Dengan A menyatakan amplitudo gelombang dan C adalah laju gelombang

3. Metode Beda Hingga

Penyelesaian PDP dapat dilakukan dengan metode analisis dan numerik. Penyelesaian analitis model matematika adalah penyelesaian yang didapat dari manipulasi aljabar terhadap persamaan dasar sehingga didapat suatu penyelesaian yang berlaku untuk setiap titik dalam domain yang menjadi perhatian.

Namun, tidak semua masalah fisika dalam model matematis dapat diselesaikan secara analistis. Untuk menyelesaikan permasalahan ini biasanya digunakan penyelesaian numeris, di mana persamaan dasar diubah menjadi persamaan yang hanya berlaku pada titik-titik tertentu di dalam domain penyelesaian. Pengubahan persamaan tersebut dapat menggunakan metode elemen hingga ataupun metode beda hingga. Untuk permasalahan satu dimensi, metode yang umum digunakan adalah metode beda hingga karena mudah digunakan dan lebih dahulu dikenal sehingga sifat-sifatnya sudah dipahami(Chapra & Canale, 1998).

Finite difference atau metode beda hingga merupakan metode penyelesaian numerik dengan menggunakan persamaan turunan yang dibatasi pada suatu orde. Dengan menggunakan pendekatan fungsi limit dari selisih nilai fungsi-fungsi pada sekitarnya, nilai titik tertentu dapat diketahui turunannya(Chapra & Canale, 2013).

Forward difference atau beda maju digunakan untuk untuk mencari variabel yang nilainya digeser ke depan. Persamaan ini dituliskan sebagai berikut:

$$\frac{\partial f}{\partial x} \approx \frac{f\ x + \Delta x\ - f(x)}{\Delta x} \quad (8)$$

Backward difference atau beda mundur digunakan untuk untuk mencari variabel yang nilainya digeser ke belakang. Persamaan ini dituliskan sebagai berikut:

$$\frac{\partial f}{\partial x} \approx \frac{f\ x\ - f\ x + \Delta x}{\Delta x} \quad (9)$$

Central difference atau beda tengah digunakan untuk untuk mencari variabel yang nilainya digeser ke depan dan belakang. Persamaan ini dituliskan sebagai berikut:

$$\frac{\partial f}{\partial x} \approx \frac{f\ x + \Delta x\ - f\ x - \Delta x}{2\Delta x} \quad (10)$$

4. Metode Spektral

Penyelesaian PDP dengan menggunakan metode *finite difference* berdasarkan dari solusi fungsi lokal. Hal ini berbeda dengan penyelesaian metode spektral berdasarkan solusi fungsi global. Model penyelesaian global akan memberikan tingkat akurasi yang lebih presisi dibandingkan dengan metode beda hingga. Untuk menyelesaikan permasalahan spektral ini dengan menggunakan Transformasi Fourier. Fungsi Transformasi Fourier $u\ x\ , x \in \mathbb{R}$ adalah fungsi $u\ k$ yang didefinisikan sebagai:

$$u\ k\ = \int_{-\infty}^{\infty} e^{-ikx}\, u\ x\ dx,\ \ k \in \mathbb{R}. \quad (11)$$

Sedangkan *inverse* dari Transformasi Fourier:

$$u\ x\ = \frac{1}{2} \int_{-\infty}^{\infty} e^{-ikx}\, u\ k\ dk,\ \ x \in \mathbb{R}. \quad (12)$$

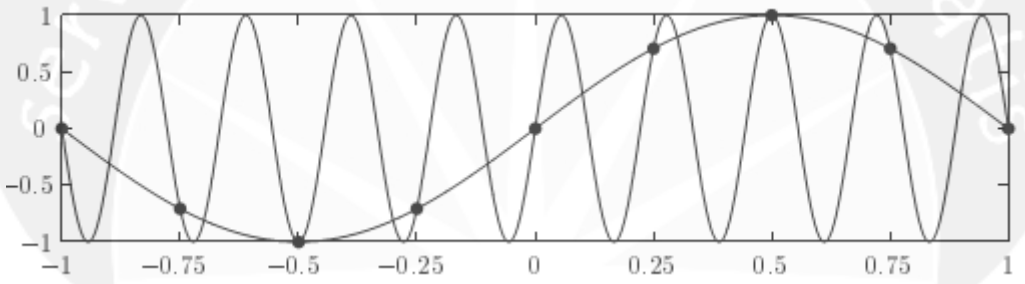
Variabel x adalah variabel fisik dan variabel k adalah variabel Fourier atau wavenumber.

Kita harus mempertimbangkan range nilai x. karena pada transformasi furier akan merubah range data.

$$\begin{array}{lcccl} \text{Physical space} & : & \text{discrete}, & \text{unbounded} & : \quad x \in h\mathbb{Z} \\ & & \updownarrow & \updownarrow & \\ \text{Fourier space} & : & \text{bounded}, & \text{continuous} & : \quad k \in [-\pi/h, \pi/h] \end{array}$$

Untuk alasan simetris maka akan digunakan interval $[-\pi/h\,, \pi/h]$. Untuk fungsi v yang didefinisikan pada $h\mathbb{Z}$ dengan nilai v_j pada x_j, maka *Semidiscrete Fourier Transform* akan didefinisikan dengan:

$$v\ k\ = h \sum_{j=-\infty}^{\infty} e^{-ikx_j}\, vj, \qquad k\ \in [-\pi/h\,, \pi/h], \tag{13}$$



Gambar 2.1. Fungsi identik sin πx dan sin 9πx

Dan inverse dari *Semidiscrete Foruier Transform* adalah

$$v_{j=}\frac{1}{2\pi}\int_{-\pi/h}^{\pi/h} e^{-ikx_j}\, v(k)dk, \quad j \in \mathbb{Z}. \tag{14}$$

Untuk difrensial spektral maka dibutuhkan interpolan, pada inverse transform akan menghasilkan data tersebut. Yang perlu kita lakukan adalah menentukan v dan kita dapat mendefinisikan interpolan p dengan:

$$p\ x\ = \frac{1}{2\pi}\int_{-\pi/h}^{\pi/h} e^{-ikx}\, v(k)dk, \quad x \in \mathbb{R}. \tag{15}$$

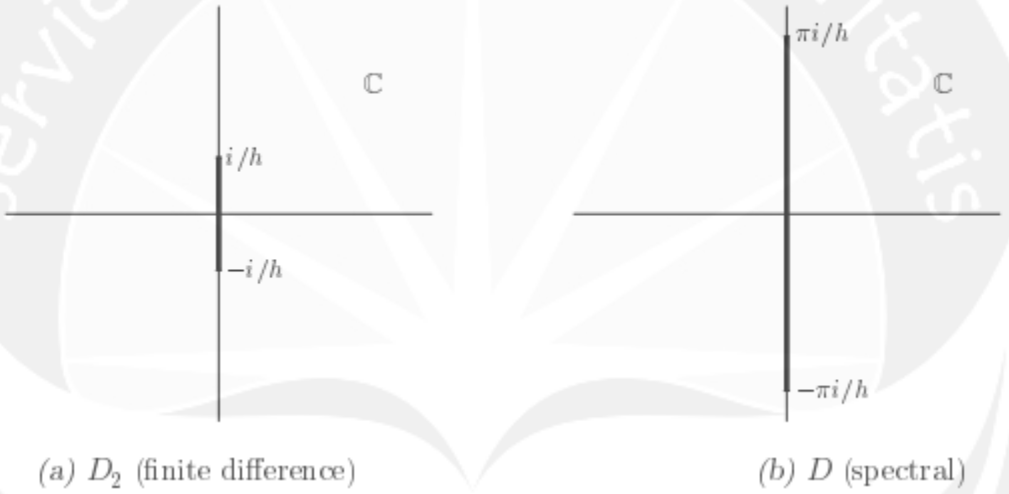
Transformasi Fourier p didefinisikan dari Persamaan 11 adalah:

$$p = \begin{matrix} v\ k\ , & k \in [-\pi/h\ , \pi/h] \\ 0 & nilai\ k\ \ yang\ lain. \end{matrix}$$

Jika u adalah fungsi *differentiable* dengan Transformasi Fourier u , maka Transformasi Fourier dari u'adalah $iku\ \ k$:

$$u'(k) = iku(k) \quad \textbf{(16)}$$

Pada dimensi frekuensi bilangan riil akan berubah menjadi blangan kompleks. Namun dengan transformasi tersebut akan lebih meningkatkan akurasi. Contoh ilustrasi :



Gambar 2.2. Perbandingan metode beda hingga dan spektral

5. Inpainting

Inpainting adalah proses untuk melakukan rekonstruksi area yang rusak(Bertalmio et al., 2000). Proses *inpainting* citra digital terinspirasi dari seniman lukis dalam melakukan restorasi lukisan dari zaman Meditrania dan Renaissance. Algoritma *inpainting* dapat dikategorikan menjadi beberapa macam yaitu(Ravi et al., 2013):

1. *Texture synthesis based image inpainting*

   Model ini mengisi area yang rusak dengan menggunakan sampel copy dari piksel yang berada di sebelahnya.

2. *Exemplar and search based image inpainting*

   Model ini menggunakan mekanisme prioritas untuk mengisi area yang kosong. Hal ini dapat bekerja baik untuk tekstur dan replikasi struktur. Model ini juga dapat melakukan pengisian area rusak dengan cara pencarian piksel yang sesuai dari database citra.

3. *PDE based inpainting*

   Menggunakan PDP untuk melakukan pengisian area citra yang rusak. Model ini menggunakan konsep isotrop dan proses difusi.

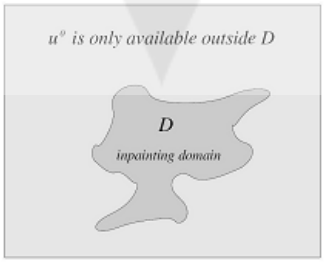
4. *Fast semiautomatic inpainting*

   Model ini melakukan *inpainting* dengan menggunakan *iterative convolving* dengan menggunakan kernel difusi.

5. *Hybrid inpainting*

   Model ini melakukan kombinasi PDP dan *texture synthesis* dalam melakukan *inpainting*.

Pendekatan yang diambil untuk melakukan penelitian ini dengan menggunakan pendekatan PDP. Melalui pendekatan PDP ini proses *inpainting* sebenarnya merupakan proses interpolasi suatu citra digital. Jika f merupakan representasi dari citra digital pada domain citra Ω, Maka bisa dikatakan bahwa permasalahan rekontruksi citra asli u pada domain citra rusak $D \subset \Omega$, yang disebut sebagai domain *inpainting* atau *hole* (Scho□nlieb, 2009).



Gambar 2.3. Domain inpainting

Hal tersebut menghasilkan skema dari model diskrit yang berasal dari *non-linier* PDP:

$$U_t = \nabla^{\perp} u . \nabla u, \quad (17)$$

Persamaan tersebut harus diselesaikan dalam *inpainting* domain D menggunakan informasi dari tepi citra disekitar D. Operator menggambarkan gradien tegak lurus $(-\partial_y, \partial_x)$. Pada tulisan-tulisan sebelumnya variasi model ini biasanya digunakan untuk *denoising* citra, *debluring* citra yang kemudian diadopsi untuk proses *inpainting*(Burger et al., 2009).

6. Persamaan PDP Orde keempat untuk *inpainting*

Penelitian penggunaan PDP orde keempat untuk permasalahan *inpainting* sudah dilakukan oleh Bertozzi, Esedoglu dan Gillette (Bertozzi et al., 2007). Hal tersebut menghasilkan versi *inpainting u* dari $f \in L^2(\Omega)$ yang membentuk algoritma sebagai berikut:

$$u_t = \Delta\left(-\epsilon\Delta u + \frac{1}{\epsilon}F'(u)\right) + \lambda(f-u), di\ \Omega, \quad (18)$$

Dimana $F(u)$ disebut sebagai potensial *double-well*. Contohnya $F(u) = u^2(u-1)^2$, dan

$$\lambda(x) = \begin{matrix} \lambda_0 & \Omega \backslash D \\ 0 & D \end{matrix}$$

Persamaan 18 tersebut identik dengan persamaan Cahn-Hilliard. Persamaan tersebut kemudian berkembang dalam dunia *inpainting* oleh Schönlieb dan Bertozzi kemudian mengalami evolusi (Schönlieb & Bertozzi, 2011). Persamaan tersebut menjadi:

$$\frac{u_{k+1}-u_k}{\Delta t} + C_1\Delta\Delta u_{k+1} + C_2 u_{k+1} = C_1\Delta\Delta u_{k+1} - \Delta\left(\nabla . \frac{\Delta u_k}{\Delta u_k}\right) + C_2 u_k + \lambda(f-u) \quad (19)$$

Untuk meyakinkan E_{11} , E_{12} dan E_{21} , E_{22} adalah benar-benar convex, maka konstan dari C_1 dan C_2 harus dipilih nilai yang $C_1 > \frac{1}{\epsilon}, C_2 > \lambda_0$.

Ketika nilai konstan dari C_1 dan C_2 telah memenuhi persyaratan $C_1 > \frac{1}{\epsilon}, C_2 > \lambda_0$ maka akan terpenuhi kondisi berikut (Scho□nlieb, 2009):

a. Konsisten
b. Stabil tanpa syarat
c. Konvergen

7. Validasi Citra Digital

Semua proses yang dilakukan pada citra digital dapat mengakibatkan hilangnya informasi atau berkurangnya kualitas. Pengukuran kualitas citra digital dapat dilakukan metode subjektif dan metode objektif. Pengukuran subjektif berdasarkan penilaian manusia tanpa disertai dengan referensi kriteria secara eksplisit. Sedangkan pengukuran objektif menggunakan perbandingan eksplisit dari kriteria numerik dan disertai dengan referensi-referensi dari penelitian yang sebelumnya dalam parameter statistik untuk kepentingan tes(Hore & Ziou, 2010).

Citra yang rusak tentunya akan susah diukur kualitas perbaikan yang sudah dilakukan oleh algoritma, oleh karena itu untuk mengukur perbaikan citra digunakan bangun 2D untuk membantu proses pengukuran tesebut. Bangun yang digunakan yaitu persegi panjang dan lingkaran. Bangun 2D tersebut kemudian dirusak, setelah itu akan digukur kualitas perbaikan citra yang terjadi. Pengukuran kualitas digunakan validasi sebagai berikut:

a. *Peak Signal to Noise Ratio (PSNR) dan MSE (Mean Square Error)*

PSNR adalah istilah dari teknik yang menggambarkan rasio dari tenaga maksimum dari suatu sinyal dibandingkan dengan tenaga dari sinyal *noise*. PSNR merupakan pengukuran objektif yang umum digunakan dalam pengukuran kualitas citra digital(Ismail Avcıbas, 2002).

Jika suatu citra f dan suatu citra test g, yang masing-masing memiliki ukuran M x N, maka dapat dirumuskan PSNR antara f dan g:

$$PSNR\ f,g\ = 10\log_{10}\ \frac{255^2}{MSE\ f,g} \quad (20)$$

dimana,

$$MSE\ f,g\ = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N} f_{ij} - g_{ij}\ ^2 \quad (21)$$

Nilai dari PSNR akan menjadi tidak terhingga ketika MSE mendekati atau sama dengan nol. Hal ini menunjukkan jika semakin tinggi nilai PSNR akan menunjukkan tingginya kualitas suatu citra. Sebaliknya jika nilai PSNR tinggi menunjukkan besarnya perbedaan dari perbandingan suatu citra(Hore & Ziou, 2010).

b. *Structural Similarity Index Measure (SSIM)*

SSIM merupakan indek untuk melakukan pegukuran kepriripan antara dua buah citra digital. Indeks SSIM dapat dilihat sebagai pengukuran kualitas suatu gambar yang dibandingkan , asalkan sumber pengukuran dianggap sebagai gambar dengan kualitas yang sempurna. SSIM diperhitungkan mempunyai hubungan dengan penilaian kualitas presepsi dari *human visual system* (HVS) (Hore & Ziou, 2010).

$$SSIM\ x,y\ = \frac{2\mu_x\mu_y + C_1\ \ 2\sigma_{xy} + C_2}{\mu_x^2 + \mu_y^2 + C_1\ \ \sigma_x^2 + \sigma_y^2 + C_2} \quad (22)$$

dengan,

μ_x = nilai rata-rata dari x

μ_y = nilai rata-rata dari y

σ_x^2 = nilai varian dari x

σ_y^2 = nilai varian dari y

σ_{xy}= nilai kovarian dari x dan y

$C_1 = \ k_1 L\ ^2$, sedangkan nilai dari $C_2 = \ k_2 L\ ^2$, dimana nilai L adalah range dinamik dari pixel (biasanya bernilai 2)

k_1 dan k_2 merupakan nilai koefisien yang mempunyai nilai default ($k_1 = 0{,}01$ dan $k_2 = 0{,}03$)

8. OpenCV

OpenCV merupakan open source (lihat http://opensource.org) *library* pengindraan komputer yang tersedia di http://SourceForge.net/projects/opencvlibrary. *library* tersebut ditulis dalam bahasa pemrograman C dan C ++ yang berjalan di sistem operasi Linux, Windows dan Mac OS X. terdapat pengembangan aktif pada antarmuka untuk Python, Ruby, Matlab, dan bahasa lainnya (GaryBradski & Kaebler, 2008).

OpenCV tumbuh dari penelitian Intel pada aplikasi komputer *vision* tingkat lanjut. Untuk itu, Intel meluncurkan banyak proyek termasuk *real-time ray tracing* dan dinding layar 3D. Salah satu penulis bekerja untuk Intel pada saat itu mengunjungi universitas dan melihat bahwa beberapa kelompok universitas top, seperti MIT Media Lab, telah mempunyai struktur yang kuat dan pempunyai kode internal untuk melakukan komputer *vison*. Hal tersebut diserahkan untuk siswa dan siswa mulai menggembangkan aplikasinya sendiri. Alih-alih menciptakan kembali fungsi dasar dari awal, mahasiswa mulai membangun sistem sehingga mahasiswa yang baru akan meneliti bisa memulai dengan membangun di atas apa yang sudah dibuat oleh mahasiswa yang datang sebelumnya.

OpenCV memiliki struktur modular, yang berarti bahwa terdapat paket atau beberapa *library* berbagi atau statis. OpenCV memudahkan programmer dalam melakukan input, output dan proses yang berhubungan dengan citra. Modul-modul dasar yang biasa digunakan:

a. **core** - modul kompak mendefinisikan struktur data dasar, termasuk fungsi padat multi-dimensi *array* Mat dan dasar yang digunakan oleh semua modul lainnya.

b. **imgproc** - modul pengolahan citra yang mencakup linear dan citra non-linear *filtering*, transformasi gambar geometri (mengubah ukuran, perspektif, warping), konversi ruang warna, histogram, dan sebagainya.

c. **video** - modul analisis video yang mencakup estimasi gerak, latar belakang pengurangan, dan algoritma pelacakan objek.

d. **calib3d** - dasar multiple-lihat algoritma geometri, kamera kalibrasi tunggal dan stereo, objek menimbulkan estimasi, algoritma stereo korespondensi, dan unsur-unsur rekonstruksi 3D.

e. **features2d** - detektor fitur yang menonjol, deskripsi, dan pencocokan deskripsi.

f. **objdetect** - deteksi objek dan contoh dari kelas yang telah ditetapkan (misalnya, wajah, mata, orang, mobil, dan sebagainya).

g. **highgui** - mudah digunakan antarmuka untuk merekam video, gambar dan video codec, serta kemampuan UI sederhana.

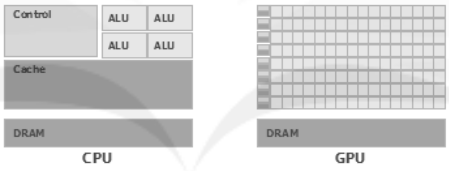
h. Beberapa modul pembantu lainnya, seperti Flann dan Google *test wrappers*, Python *bindings* dan lain-lain.

9. Paralel Computing GPU CUDA

Tahun 2001, NVIDIA mengenalkan GPU (GeForce3) yang dapat di program untuk pertama kalinya. Dua tahun kemudian, 2003 di San Diego, diselenggarakan workshop Siggraph (*Eurographics Graphics Hardware*) disana ditunjukan perpindahan komputasi grafik ke non-grafik pada GPU. Hal ini kemudian memunculkan konsep GPGPU (*General-Purpose computing on GPU*). Pada awalnya digunakan Brook sebagai bahasa pemrograman pada GPU (Diaz et al., 2012).

Pada bulan November 2006, NVIDIA mengenalkan CUDA(*Compute Unified Device Architecture*) untuk melakukan general purpose paralel computing dan sebagai model bahasa pemrograman paralel yang dapat melakukan komputasi pada GPU untuk menyelesaikan permasalahan komputasi dengan cara yang lebih efisien dibandingkan CPU (Nvidia, 2014).

GPU sangat cocok untuk mengatasi masalah yang melakukan perhitungan banyak data. GPU akan menjalankan program yang sama dan melakukan perhitungan dengan intensitas aritmatika yang tinggi pada banyak elemen data secara paralel dengan melakukan operasi memori pada GPU. GPU memiliki kendala memori yang terbatas. Namun GPU dapat lebih cepat dari CPU karena program yang sama dijalankan untuk setiap elemen data dan melakukan proses aritmatika dalam data yang banyak. Kendala perpindahan data yang memakan banyak waktu, tertutup dengan kecepatan komputasi yang dimiliki GPU (NVIDIA, 2014).



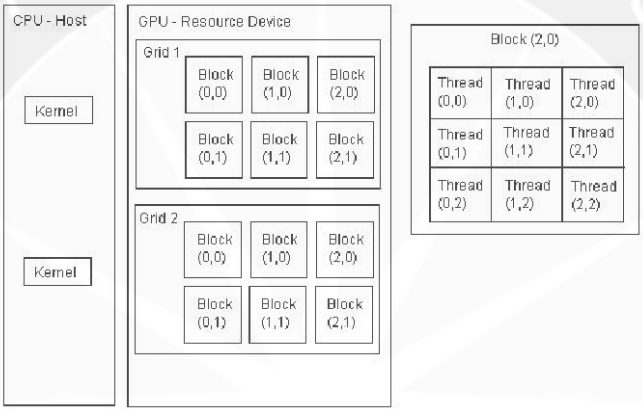
Gambar 2.4. Perbedaan GPU dan CPU

Sistem pemrograman secara paralel di dalam CUDA terdiri dari *host*(CPU) dan *device*(GPU). Proses komputasi terjadi pada thread pada GPU secara paralel. Arsitektur dari GPU threads terdiri dari dua *level* hirarki, yaitu: *block* dan *grid* (NVIDIA, 2014)(Diaz et al., 2012).

Block adalah set dari *threads* yang tergabung erat, masing-masing teridentifikasi oleh *ThreadID*. Sedangkan *Grid* adalah set dari *block* yang tergabung erat. Tidak diperlukan sinkoronisasi sama sekali antara *block* dan *grid* yang dilakukan pada GPU tunggal. GPU dapat dikatakan sebagai kumpulan dari *multiprocessors*, dimana tiap *multiprocessors*

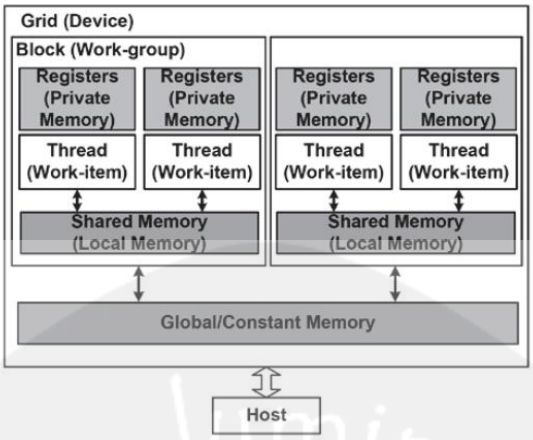
bertanggung jawab terhadap satu atau lebih *block* di dalam *grid*. *Threads* didalam *blok* dapat beroperasi dan melakukan sinkronisasi dalam mengeksekusi memori koordinat akses dengan cara berbagi data melalui suatu *shared memory*(Diaz et al., 2012).

Manajemen kerja pada CUDA dilakukan secara implisit.Hal ini berarti *programmer* tidak perlu mengelola pembuatan dan penghancuran *thread*. Namun cukup melakukan spesifikasi dimensi dari tiap *grid* dan *blok* yang diperlukan untuk melakukan suatu kegiatan(Diaz et al., 2012) (NVIDIA, 2014).



Gambar 2.5. Pemetaan Thread, Block dan Grid

Pemrograman paralel memetakan tiap elemen data secara paralel. Aplikasi yang melakukan pemrosesan dengan menggunakan *data set* yang besar sangat cocok menggunakan model pemrograman secara paralel untuk mempercepat proses komputasi. Kenyataan yang terjadi banyak dari algoritma yang dapat dipercepat dengan pemrograman secara paralel (NVIDIA, 2014).



Gambar 2.6. Arsitektur CUDA

Secara umum pemrograman GPU akan melakukan langkah sebagai berikut (Owens et al., 2008):

1. Programmer menentukan geometri yang mencakup wilayah domain. *Rasterizer* ini menghasilkan fragmen di setiap lokasi piksel ditutupi oleh yang geometri.
2. Setiap fragmen dinaungi oleh program fragmen.
3. Program fragmen menghitung nilai fragmen dengan kombinasi operasi matematika dan memori global
4. Hasil perhitungan yang dihasilkan kemudian dapat digunakan sebagai tekstur yang akan ditransfer dari kartu grafis.

Pemrograman pada GPU juga mempunyai berbagaimacam teknik pengaksesan data yang dapat digunakan untuk memetakan algoritma yang kompleks menjadi lebih efisien sehingga cocok dengan arsitektur pada GPU. Teknik tersebut antara lain (Owens et al., 2007):

1. *Map*

   Merupakan operasi yang paling mudah pada pemrosesan paralel. Hal ini berarti akan memetakan satu persatu data ke fungsi yang ada. Teknik ini akan sering digunakan dalam pemrosesan citra.

2. *Reduce*

   Seringali suatu perhitungan membutuhkan pengurangan data. Contohnya adalah mencari nilai maksimum/minimum dan penjumlahan total data(sum). Untuk melakukan hal ini maka dibutuhkan teknik *reduce*.

3. *Scatter and gather*

   Gather merupakan operasi yang melakukan beberapa operasi pengambilan data sekaligus dan hanya memetakan satu hasil saja. Operasi *gather* yang populer adalah *stencil*. Ini merupakan operasi yang melibatkan atas, bawah, kiri dan kanan piksel citra.

   Operasi *Scatter* memiliki kebalikan dari gather, yaitu suatu data diambil kemudian hasilnya akan dipetakan di beberapa lokasi.

4. dan lain- lain seperti teknik *Scan, Stream filtering, Short* dan *Search*

10. Pengukuran Pemrosesan secara Paralel

a. Latency

Pengukuran eksekusi waktu suatu proses pada CPU dari awal sampai dengan selesai dibandingkan dengan eksekusi waktu proses pada GPU dari awal sampai dengan selesai. Pengukuran ini menghasilkan ukuran percepatan waktu antara proses CPU dan GPU. Pengukuran ini menggunakan satuan detik.

b. Throughput

Throughput adalah pengukuran seberapa banyak unit informasi yang dapat diproses dalam suatu satuan waktu. Dengan membandingkan banyaknya pekerjaan dan waktu yang diperlukan maka throughput dapat juga disebut transactions per second (TPS). Jadi ketika nilai throughput tinggi, artinya jumlah data yang dapat diproses tiap satuan waktu juga banyak.