BAB II

TINJAUAN PUSTAKA

A. Tinjauan Pustaka

Pengolahan citra adalah sebuah bentuk pengolahan sinyal dimana masukannya berupa sebuah citra, dan keluarannya dapat berupa citra kembali atau apapun yang melalui beberapa pengolahan berarti (Tse, 2012). Mencerahkan citra, mendeteksi bagian tepi, memperbaiki citra adalah contoh pengolahan citra.

Image inpainting merupakan sebuah proses pengolahan citra yang digunakan untuk mengembalikan area yang hilang atau rusak dari sebuah citra (Guillemot & Le Meur, 2014). Teknik digital mulai dikenal secara luas sebagai sebuah cara untuk melakukan image inpainting, mulai dari upaya deteksi yang sepenuhnya otomatis dan penghapusan goresan pada film, sampai pada *software* dengan proses canggih tetapi masih diperlukan keterlibatan *user* (Bertalmio, Sapiro, Caselles, & Ballester, 2000).

Salah satu pencetus *digital image inpainting* adalah Bertalmio et al. Mereka memanfaatkan proses difusi persamaan *heat* dengan *partial differential equation* (PDE) bentuk parabolik untuk menghantarkan informasi intensitas citra pada sekitar daerah yang rusak kedalam daerah yang rusak. Terdapat dua macam skema dalam teknik ini, skema eksplisit dan implicit. Pertama diprakarsai oleh Bertalmio et al, kemudian disusul Panca dalam skripsinya dengan mengganti metode

Laplacian dengan metode *Perona-Malik* (Panca, 2015). Penggunaan metode ini memiliki kelebihan yaitu implementasinya yang mudah dan hasil *inpainting* yang halus. Kelemahan metode ini adalah hasil *inpainting* pada citra berbentuk geometri dan langkah waktu yang cenderung kecil sehingga diperlukan banyak iterasi. Zhang et al kemudian melakukan modifikasi pada metode diatas dengan melindungi bagian tepi citra (terutama untuk citra berbentuk geometri), karena menurutnya mata manusia memiliki ketertarikan terhadap informasi pada bagian tepi sebuah citra (Zhang, et al., 2015).

Skema implicit memiliki kelebihan yaitu menghasilkan solusi yang stabil dan konvergen meskipun langkah waktu yang digunakan cukup besar. Bagaimanapun juga, pemecahan solusi skema implicit pada ruang dua dimensi sangat memakan waktu. Salah satu cara untuk mengatasi kekurangan dan ketidakefisiensian metode tersebut adalah menggunakan metode pemecahan. Metode ini dikenal dengan metode *Alternating-Direction Implicit* (ADI). Algoritmanya menghasilkan dua buah persamaan tridiagonal yang diselesaikan secara berurutan (Hoffmann & Chiang, 2000).

Teknik lain dalam *image inpainting* menggunakan *partial differential equation* (PDE) adalah dengan implementasi model *phase-field*. Kelebihan teknik ini adalah melindungi bagian tepi sekaligus mengurangi iterasi karena dapat menggunakan langkah waktu yang tidak terlalu kecil. Dua persamaan model phase-field yang sering dipakai adalah persamaan *Cahn-Hilliard* dan persamaan *Allen-Cahn*. Bertozzi et al memprakarsai implementasi model *phase-field* untuk

image inpainting, yang kemudian diperbarui oleh Darae et al yang metode *multigrid* untuk penyelesaian solusinya. Persamaan *Cahn-Hilliard* membutuhkan pemahaman matematis yang baik untuk mengimplementasinya, karena persamaan ini melibatkan operator diferensial ordo keempat. Alasan tersebut mendorong Li et al menggunakan persamaan *Allen-Cahn* dalam *image inpainting*.

Sering kali, proses pengolahan citra terjadi pada hampir keseluruhan citra. Hal ini berarti banyak pengulangan pada suatu pekerjaan yang sama. Teknologi terbaru memungkinkan kualitas citra yang semakin baik, yang berarti ukuran *file* yang lebih besar dan waktu proses yang lebih lama. Dengan dimunculkannya CUDA, pemrograman dengan GPU dipermudah. Teknologi ini sudah siap untuk digunakan sebagai pemecah masalah pada bidang pengolahan citra (Tse, 2012).

Implementasi CUDA diterapkan oleh Panca (2015) yang dalam penelitiannya (penyelesaian persamaan *heat*), CUDA dapat mempercepat proses komputasi hingga 56 kali, daripada menggunakan CPU. Kemudian Prananta (2015) juga menggunakan CUDA untuk mempercepat penyelesaian persamaan *Cahn-Hilliard* yang mengimplementasi metode spektral, mampu mempercepat proses komputasi hingga 48 kali.

B. Landasan Teori

1. Image Inpainting

Bertalmio et al (2000) menggunakan istilah *inpainting* untuk melakukan restorasi maupun menghilangkan objek pada suatu citra. Mereka

menggunakan persamaan *heat* yang diselesaikan menggunakan *Partial Differential Equation* (PDE).

Persamaan algoritma *inpainting* secara umum menurut Bertalmio et al. adalah sebagai berikut (katakanlah I adalah intensitas sebuah citra):

$$I^{n+1}_{(i,j)} = I^{n}_{(i,j)} + \Delta t . I^{n}_{t\,(i,j)}, \forall (i,j) \in \Omega \qquad (1)$$

Dimana n menunjukkan indeks iterasi, (i,j) adalah piksel koordinat, Δt merupakan kadar perbaikan, dan $I^{n}_{t\,(i,j)}$ merupakan perbaruan dari citra $I^{n}_{(i,j)}$. Perlu dicatat bahwa perhitungan hanya dilakukan pada Ω, yaitu daerah yang akan dilakukan proses *inpainting*. (Bertalmio et al, 2000)

2. *Partial Differential Equation* (PDE)

Partial Differential Equation (PDE) berbentuk parabolik dapat didiskretisasi menggunakan metode *Finite Difference* ordo kedua (*second-order*). Berikut penjabaran persamaan *inpainting* secara eksplisit :

$$\frac{\partial I}{\partial t} = \alpha \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

$$\frac{\partial I}{\partial t} = \frac{I^{n+1}_{(i,j)} - I^{n}_{(i,j)}}{\Delta t}$$

$$\frac{\partial^2 I}{\partial x^2} = \frac{I^{n}_{(i+1,j)} - 2I^{n}_{(i,j)} + I^{n}_{(i-1,j)}}{\Delta x^2}$$

$$\frac{\partial^2 I}{\partial y^2} = \frac{I^{n}_{(i,j+1)} - 2I^{n}_{(i,j)} + I^{n}_{(i,j-1)}}{\Delta y^2}$$

$$\frac{I_{(i,j)}^{n+1} - I_{(i,j)}^{n}}{\Delta t} = \alpha \frac{I_{(i+1,j)}^{n} - 2I_{(i,j)}^{n} + I_{(i-1,j)}^{n}}{\Delta x^2} + \frac{I_{(i,j+1)}^{n} - 2I_{(i,j)}^{n} + I_{(i,j-1)}^{n}}{\Delta y^2} \quad (2)$$

Δx dan Δy merupakan jarak antar piksel pada sumbu x dan y. Pada citra diketahui bahwa jarak antar piksel adalah 1, sehingga jika disubstitusikan pada persamaan (2), maka akan didapati sebagai berikut :

$$I_{(i,j)}^{n+1} - I_{(i,j)}^{n} = \alpha\Delta t \; I_{(i+1,j)}^{n} + I_{(i-1,j)}^{n} + I_{(i,j+1)}^{n} + I_{(i,j-1)}^{n} - 4I_{(i,j)}^{n} \quad (3)$$

Kelompokkan untuk nilai yang sudah diketahui dengan nilai yang tidak diketahui, maka persamaannya akan menjadi sebagai berikut :

$$I_{(i,j)}^{n+1} = I_{(i,j)}^{n} + \alpha\Delta t \; I_{(i+1,j)}^{n} + I_{(i-1,j)}^{n} + I_{(i,j+1)}^{n} + I_{(i,j-1)}^{n} - 4I_{(i,j)}^{n} \quad (4)$$

Pada riset Bertalmio et al. Δt dinyatakan sebagai sebuah konstanta yaitu 0.1. Bagaimanapun juga, skema eksplisit hanya stabil pada kondisi tertentu: jika langkah waktu (Δt) terlalu besar, maka tidak dapat diperoleh solusi dengan kesalahan yang terkontrol (Lin, Zhang, & Tang, 2009). Sehingga mereka menggunakan *2D standard heat equation* $f_t = K(f_{xx} + f_{yy})$:

$$\Delta t \leq \frac{1}{4K} \min((\Delta x)^2, (\Delta y)^2) \quad (5)$$

3. Metode *Crank-Nicolson*

Skema eksplisit memiliki kelemahan yaitu ketidakstabilan dan juga langkah waktu yang cenderung kecil sehingga memerlukan banyak iterasi dalam prosesnya. Oleh karena itu, dimunculkanlah skema implisit untuk

mengatasi kelemahan skema eksplisit tersebut. Salah satu persamaan dalam skema implicit yang terkenal adalah persamaan *Crank-Nicolson*. Persamaan *Crank-Nicolson* dapat ditulis sebagai berikut (Hoffmann & Chiang, 2000):

$$\frac{I_{(i,j)}^{n+1} - I_{(i,j)}^{n}}{\Delta t} = \frac{\alpha}{2} \begin{array}{l} \dfrac{I_{i-1,j}^{n+1} - 2I_{i,j}^{n+1} + I_{i+1,j}^{n+1}}{\Delta x^2} + \\ \dfrac{I_{i,j-1}^{n+1} - 2I_{i,j}^{n+1} + I_{i,j+1}^{n+1}}{\Delta y^2} + \\ \dfrac{I_{i-1,j}^{n} - 2I_{i,j}^{n} + I_{i+1,j}^{n}}{\Delta x^2} + \\ \dfrac{I_{(i,j-1)}^{n} - 2I_{(i,j)}^{n} + I_{(i,j+1)}^{n}}{\Delta y^2} \end{array} \quad (6)$$

Sangat sulit untuk mendapat solusi persamaan ini dan juga tidak efisien secara komputasi. Sehingga, penggunaan metode *Alternating-Direction Implicit* (ADI) terkenal dapat mengatasi hal tersebut. Metode ADI terkenal karena performanya yang lebih baik daripada metode implicit lain (Laszlo, 2016).

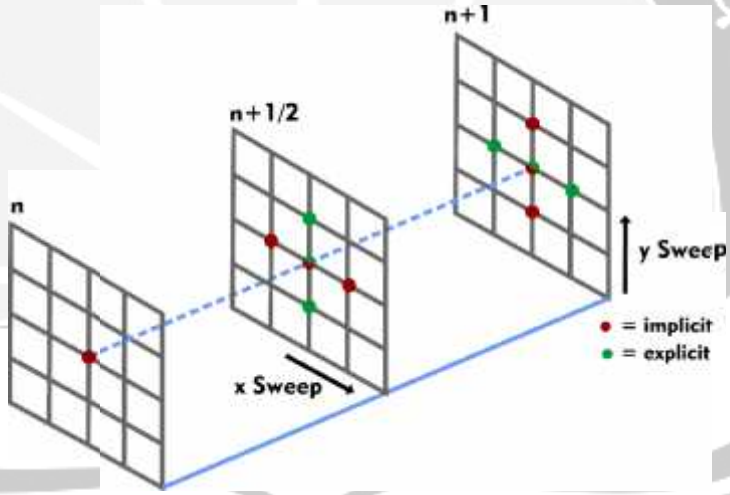
4. *Alternating-Direction Implicit* (ADI)

Metode *Alternating-Direction Implicit* (ADI) membagi persamaan *Crank-Nicolson* diatas menjadi dua persamaan (Hoffmann & Chiang, 2000):

$$\frac{I_{(i,j)}^{n+\frac{1}{2}} - I_{(i,j)}^{n}}{\Delta t} = \frac{\alpha}{2} \begin{array}{l} \dfrac{I_{i-1,j}^{n+\frac{1}{2}} - 2I_{i,j}^{n+\frac{1}{2}} + I_{i+1,j}^{n+\frac{1}{2}}}{\Delta x^2} + \\ \dfrac{I_{(i,j-1)}^{n} - 2I_{(i,j)}^{n} + I_{(i,j+1)}^{n}}{\Delta y^2} \end{array} \quad (7)$$

$$\frac{I_{(i,j)}^{n+1} - I_{(i,j)}^{n+\frac{1}{2}}}{\Delta t} = \frac{\alpha}{2}\left[\frac{I_{(i-1,j)}^{n+\frac{1}{2}} - 2I_{(i,j)}^{n+\frac{1}{2}} + I_{(i+1,j)}^{n+\frac{1}{2}}}{\Delta x^2} + \frac{I_{(i,j-1)}^{n+1} - 2I_{(i,j)}^{n+1} + I_{(i,j+1)}^{n+1}}{\Delta y^2}\right] \quad (8)$$

Dari dua persamaan diatas dapat dicermati bahwa persamaan (6) tersebut implicit pada arah sumbu x, sedangkan persamaan (7) implicit pada arah sumbu y. kedua persamaan dikerjakan secara simultan. Berikut penggambaran ilustrasi metode ADI:



Gambar 3 Ilustrasi Metode ADI

Solusi pada tiap arah nantinya akan menghasilkan sebuah system matriks berbentuk tridiagonal. Penyelesaian system matriks tersebut dapat dilakukan dengan berbagai cara, seperti eliminasi Gauss, Gauss-Jordan, Algoritma Thomas, dan lain sebagainya.

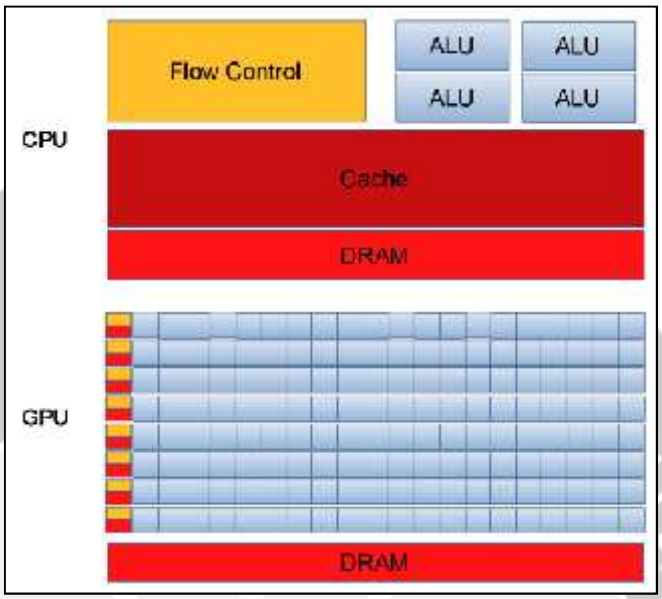
5. *Tridiagonal Matrix Algorithm* (TDMA)

Tridiagonal Matrix Algorithm (TDMA) dikembangkan oleh Llevellyn Thomas, dan merupakan sebuah cara untuk memecahkan permasalahan system persamaan tridiagonal (Quarteroni, Sacco, & Saleri, 2000). Berikut contoh system tridiagonal, dimana u^n adalah diketahui dan u^{n+1} adalah tidak diketahui.

$$\begin{matrix} b_l & c_l & 0 & 0 \\ a_i & b_i & c_i & 0 \\ 0 & \ddots & \ddots & \ddots \\ 0 & 0 & a_0 & b_0 \end{matrix} \begin{matrix} u_l^{n+1} \\ u_i^{n+1} \\ \vdots \\ u_0^{n+1} \end{matrix} = \begin{matrix} u_l^n \\ u_i^n \\ \square \\ u_0^n \end{matrix}$$

6. Komputasi Paralel pada GPU

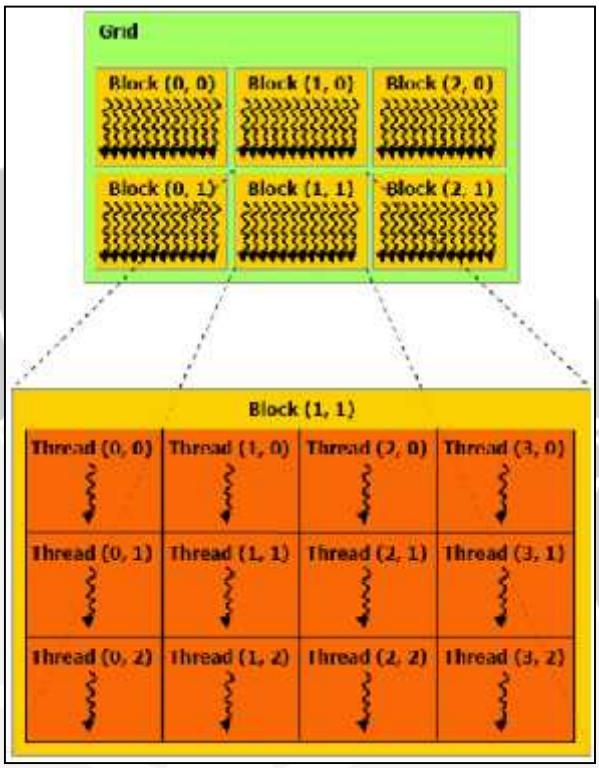
Pada tahun 2003, sebuah tim peneliti yang dipimpin oleh Ian Buck meluncurkan Brook, model pemrograman pertama yang diadopsi secara luas untuk mengembangkan C dengan kemampuan konstruksi data paralel. Menggunakan konsep seperti arus, kernel, reduksi operator, *Brook compiler* dan *runtime system* memperkenalkan GPU sebagai prosesor untuk tujuan umum didalam bahasa tingkat tinggi. Yang terpenting, program Brook tidak hanya mudah ditulis daripada kode GPU *hand-tuned*, tetapi juga tujuh kali lebih cepat dari kode lain yang sejenis. Berawal dari situlah pada tahun 2006, NVIDIA meluncurkan CUDA.



Gambar 4 Perbedaan CPU dan GPU (Xu, 2011)

Pemanfaatan GPU untuk tujuan umum dikenal dengan sebutan *General Purpose Graphic Processing Unit* (GPGPU). Fitur penting dalam GPU adalah GPU mempunyai beberapa *thread*, yang dapat bekerja bersamaan secara serentak. Sejumlah *thread* dapat dikombinasikan menjadi sebuah *block*, yang memproses bagian tertentu dari sejumlah data (Xu, 2011).

Pada dasarnya, data disimpan secara satu dimensi. Bahkan ketika data disimpan didalam secara multi dimensi, data tetap dipetakan secara satu dimensi didalam fisik memori. CPU memproses data secara sekuensial dari awal hingga akhir data (selesai satu data untuk memproses data selanjutnya). Sedangkan GPU dapat mengalokasikan setiap *thread* yang dimiliki untuk memproses setiap satuan data pada satu waktu, tergantung pada kemampuan GPU itu sendiri.



Gambar 5 Grid, Block, dan Thread (Tse, 2012)

Menurut Owens et al, banyak peneliti sudah mendemonstrasikan aplikasi yang mengekspoitasi kekuatan GPU untuk berbagai tujuan, seperti : pengurutan, pencarian dan *database queries*, persamaan diferensial, aljabar linier. (Owens et al, 2008)

7. Pengukuran Pemrosesan secara Paralel

a. *Latency* (Waktu Pemrosesan)

Waktu proses komputasi pada CPU dari awal iterasi hingga akhir iterasi dibandingkan dengan waktu proses komputasi pada GPU dari awal iterasi hingga akhir iterasi. Pengukuran ini menghasilkan besaran

percepatan waktu (*speed-up*) antara proses komputasi pada CPU dan proses komputasi pada GPU. Pengukuran ini menggunakan satuan *millisecond* (ms).

b. Throughput

Pengukuran yang dilakukan dengan cara membandingkan banyaknya unit data yang dapat dieksekusi dalam suatu satuan waktu. *Throughput* seringkali disebut dengan *transactions per second* (TPS). Semakin tinggi nilai *throughput* maka jumlah data yang dapat dieksekusi tiap satuan waktu juga banyak.

8. *Mean Squared Error* (MSE) dan *Peak Signal to Noise Ratio* (PSNR)

MSE diukur dengan cara membandingkan dua buah sinyal dengan menyediakan kuantitas nilai yang mendeskripsikan derajat kesamaan atau sebaliknya, tingkat kesalahan/distorsi diantara keduanya. Biasanya, diasumsikan bahwa salah satunya adalah sinyal asli, sementara yang lain adalah sinyal terdistorsi atau mengandung kesalahan. (Wang & Bovik, 2009)

$$MSE\ x,y\ = \frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^2 \qquad (9)$$

Dalam literatur tentang pengolahan citra, MSE sering dikonversi menjadi pengukuran PSNR :

$$PSNR = 10\log_{10}\frac{L^2}{MSE} \qquad (10)$$

9. *Structural Similarity Index Measure* (SSIM)

Structural Similarity Index Measure (SSIM) merupakan pengujian untuk mengukur kemiripan antar dua buah citra. Penilaian SSIM berhubungan dengan kualitas persepsi dan *human visual system* (Hore & Ziou, 2010).

$$SSIM\ x, y\ = \frac{2\mu_x\mu_y + C_1\ \ 2\sigma_{xy} + C_2}{\mu_x^2 + \mu_y^2 + C_1\ \ \sigma_x^2 + \sigma_y^2 + C_2} \quad (11)$$

dengan,

μ_x = nilai rerata dari x

μ_y = nilai rerata dari y

σ_x^2 = nilai varian dari x

σ_y^2 = nilai varian dari y