BAB III
LANDASAN TEORI

3.1. *Image* Digital

Image digital yang dimaksud adalah sebuah gambar seperti foto, lukisan, dsb. yang tersedia dalam format atau bentuk digital atau elektronik. *Image* digital tersebut memiliki bagian terkecil yang disebut piksel. Piksel tersebut berisi sebuah data atau nilai yang memiliki ukuran sebesar 8 bit, yang artinya nilai yang tersimpan akan berkisar dari nilai nol (0) sampai 255. Oleh komputer nilai nol pada *image* akan terbaca sebagai warna hitam, sedangkan nilai 255 akan terbaca sebagai warna putih. Hal tersebut menunjukkan bahwa untuk *image* digital nilai nol (0) sampai 255 memiliki istilah warna *greyscale*, yang artinya memiliki nilai keabu-abuan dari warna hitam hingga putih. *Image* Digital berwarna memanfaatkan tiga lapisan yang sering disebut *Red, Green, Blue* (RGB). Artinya masing-masing dari ketiga lapisan atau sering disebut *channel* mewakili satu buah warna yaitu merah, hijau dan biru. Lalu untuk nilai atau informasi yang membawa intensitas warna untuk masing-masing lapisan juga diwakili dari nilai nol (0) sampai 255. Semakin kecil nilainya maka warna juga akan semakin gelap, sebaliknya semakin tinggi nilai mendekati 255 maka warna juga akan semakin jelas atau terang (Kairul & Susanto, 2013).

Image digital tersebut akan diolah oleh komputer dan memiliki dua dimensi. Dua dimensi yang dimaksud memiliki arti bahwa *image* digital akan mempunyai dua buah koordinat yaitu horizontal (x) dan koordinat vertikal (y). Sehingga, *image* yang akan diolah nantinya akan dikonversi nilainya ke dalam matriks yang juga memiliki dua dimensi baris dan kolom.

3.2. *Computer Vision*

Computer Vision merupakan sebuah kajian atau studi yang bertujuan untuk membuat komputer menjadi seolah-olah manusia yang mampu mengenali suatu hal. Kemampuan manusia yang sangat kompleks untuk mengenali suatu hal atau objek. Komputer akan dibuat mampu untuk menganalisis dan memahami informasi

yang berguna dari sebuah *image* atau serangkaian *image* dengan menggunakan berbagai macam algoritma. Adapun pengaplikasian dari *computer vision* digunakan untuk berbagai macam topik atau kajian, diantaranya: agrikultur, *augmented reality, biometrics, forensics, face recognition, robotics,* dan lainnya.

3.3. Face Recognition

Face recognition merupakan sebuah teknik yang menggunakan komputer untuk menganalisis *image* wajah yang akan menghasilkan informasi pengenalan yang berguna dari *image* wajah tersebut. *Face recognition* sendiri merupakan salah satu pengaplikasian dari kajian *computer vision.*

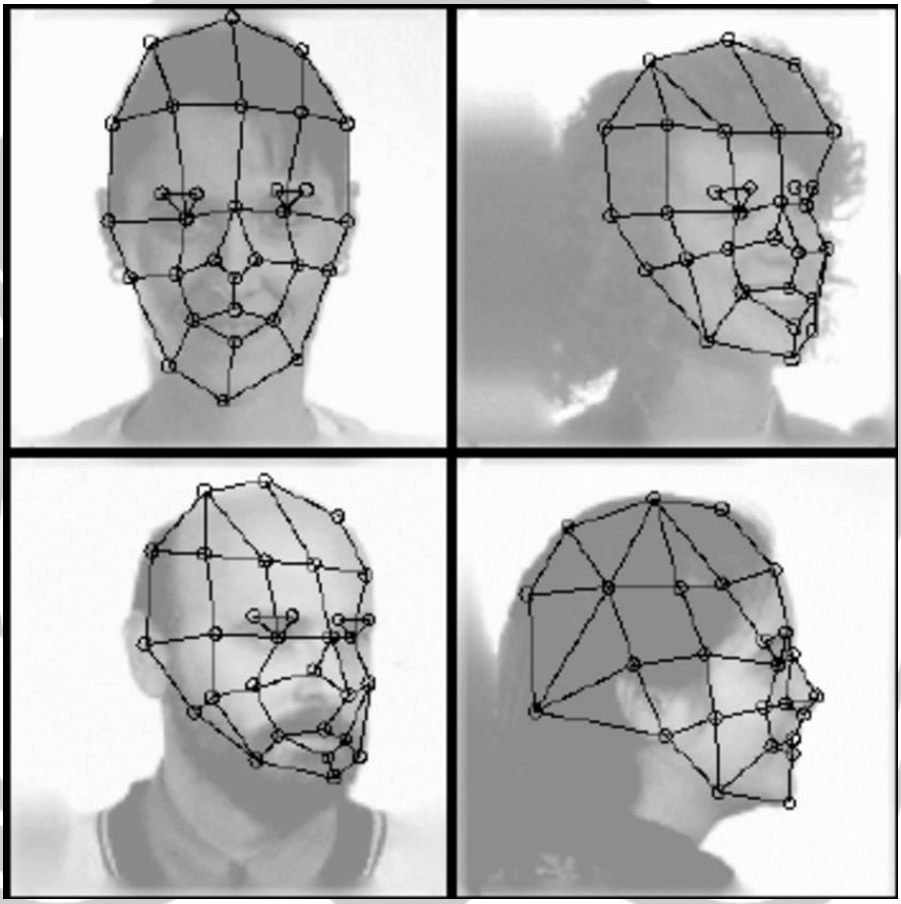
Gambar 3.1. Contoh Gambaran *Face Recognition*

Hasil informasi yang diambil dari *image* baru atau *image* yang dimasukkan ke dalam sistem akan dibandingkan dengan *image* wajah yang sudah ada pada basis data sesuai dengan individual yang tersedia. Jika ada yang sesuai maka *image* wajah baru tersebut akan teridentifikasi sesuai dengan individu yang tersimpan pada basis data.

3.4. Elastic Bunch Graph Mathcing

EBGM merupakan sebuah algoritma yang digunakan untuk *face recognition*. *Image* wajah pada algoritma ini direpresentasikan sebagai grafis dengan titik-titik pada bagian-bagian wajah seperti hidung, mata, dsb. Setiap titik tersebut

mengandung sekumpulan nilai koefisien gabor wavelet pada ukuran dan orientasi yang berbeda yang disebut dengan *jets*. Pengenalan dilakukan berdasarkan grafis yang sudah diketahui, grafis yang sudah diketahui pada basis data akan dibandingkan dengan grafis dari *image* wajah yang dimasukkan.



Gambar 3.2. Contoh Pengambilan *Feature Point* EBGM (Sumber: www.researchgate.net)

Ada tiga tahap yang dilakukan pada algoritma ini, yaitu sebagai berikut.

a. Pemodelan

Pada tahap ini *image* wajah yang dimasukkan akan dilakukan normalisasi terlebih dahulu seperti mengubah, memotong dan mengubah warna *image* menjadi *grayscale*. Setelah itu akan dibuat model grafis dari wajah yang disebut *jets*. Jet menggambarkan sebuah potongan kecil dari *gray values* pada

sebuah *image* di sekitar piksel yang diberikan. Ekstraksi jets atau dapat disebut fitur vektor tersebut dapat dilakukan dengan menggunakan metode *Gabor Wavelet Transform.* Kemudian langkah berikutnya adalah dengan memberikan tanda pada bagian-bagian tertentu wajah seperti mata, hidung, dll. dengan menggunakan *jets* yang sudah terbuat sehingga menjadi model yang bernama *face bunch graph* (FBG).

b. Pelatihan

Seluruh hal yang dilakukan pada tahap pemodelan dilakukan untuk *image* yang akan diuji dan juga *image* yang ada pada *training* set atau basis data. Proses pelatihan terjadi untuk mendapatkan model dari seluruh *image* yang ada pada basis data. Semua *image* pada dataset *training* yang telah diubah menjadi *jets* atau fitur vektor akan disimpan ke dalam sebuah *file* yang nanti akan berisi seluruh nilai transformasi fitur ekstraksi.

c. Pengenalan

Proses pengenalan dilakukan dengan membandingkan gambar yang dimasukkan dengan hasil pelatihan. Pada algoritma EBGM, pengenalan ini meliputi perhitungan *similarity measurement* antara *image* yang di uji dengan hasil pelatihan. Untuk satu *image* yang di uji, akan dihitung *similarity* dengan masing-masing fitur vektor yang ada, dan memilih nilai terbesar sebagai hasilnya. Rumus dari similarity yang digunakan adalah sebagai berikut (Wiskott, et al., 1997).

$$S_a(J, J') = \frac{\sum_j a_j a'_j}{\sqrt{\sum_j a_j^2 \sum_j a'^2_j}} \qquad (3.1)$$

3.5. Gabor Wavelet Transform

Gabor Wavelet Transform merupakan sebuah metode atau algoritma yang digunakan untuk merepresentasikan fitur lokal pada *image* yang nantinya akan digunakan untuk ekstraksi fitur (Wiskott, et al., 1997). *Gabor wavelet* memiliki visualisasi karakteristik yang baik dan memiliki kemampuan untuk melakukan

transformasi untuk *image* dengan berbagai macam arah dari *image* wajah (Yan, et al., 2015). *Gabor wavelet* akan membuat sebuah kumpulan nilai koefisien untuk *kernel* yang diukur berdasarkan ukuran dan orientasi yang dikenal dengan sebutan *jets*. Rumus yang digunakan adalah sebagai berikut.

$$G_{u,v}(z) = \frac{\|k_{u,v}\|}{\partial^2} e^{-\|k_{u,v}\|^2} \left[e^{ik_{u,v}z} - e^{-\frac{\partial^2}{2}} \right]$$

Dimana $k_{u,v} = \begin{Bmatrix} K_v cos\phi_u \\ K_v sin\phi_u \end{Bmatrix}, K_v = \frac{k_{max}}{f^v}, \phi_u = \frac{u\pi}{N}, z = (x, y)$. (3.2)

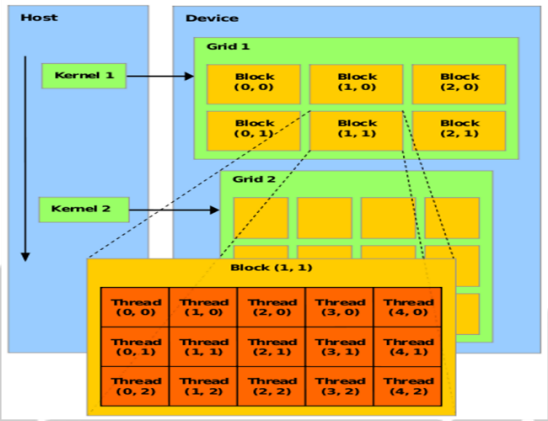
Simbol u dan v merupakan nilai dari orientasi dan skala dari *gabor filter*. k_{max} merupakan maksimal frekuensi, dan f adalah *spacing factor* diantara *kernel* dengan nilai $\sqrt{2}$. (x, y) merepresentasikan koordinat piksel. Kemudian, nilai $\partial = 2\pi, k_{max} = \frac{\pi}{2}$. Lalu untuk ekstraksi fitur pada *image* wajah biasanya digunakan lima dimensi (skala) dan delapan orientasi (arah).

Untuk mendapatkan nilai *gabor* yang diinginkan maka *image* wajah yang sedang digunakan I(x, y) akan dilakukan proses *filter* secara konvolusi dengan rumus sebagai berikut.

$$H(u, v, x, y) = G_{u,v}(z) \otimes I(x, y) \quad (3.3)$$

3.6. Paralelisasi Dengan GPU

Pemrograman paralel adalah sebuah model pemrograman dimana proses komputasi atau algoritma pada program sebagian besar dilakukan tidak lagi berurutan melainkan paralel atau secara bersamaan. Untuk mewujudkan hal tersebut, saat ini sudah ada teknologi yang mampu menangani hal tersebut yaitu dengan memanfaatkan *Graphical Processing Unit* (GPU).



Gambar 3.3. Pemetaan Grid, Block dan Thread Pada GPU CUDA (Sumber: http://gpucuda.ece.uprm.edu/wp-content/uploads/2008/11/41.png)

Langkah-langkah atau algoritma yang dilakukan untuk melakukan paralelisasi menggunakan GPU:

a. Memasukkan *image* lalu di ubah warna *image* menjadi abu-abu.
b. Menggandakan gambar ke dalam *array* satu dimensi ke GPU.
c. Menghitung integral gambar.
d. Untuk semua kemungkinan potongan jendela, lakukan: buat semua *sub-window*, lakukan *sub-window* ke blok lainnya.
e. Untuk setiap *sub-window*: lakukan perhitungan sesuai dengan algoritma yang digunakan.
f. Kirim hasil perhitungan kembali ke CPU.

Setiap proses komputasi atau algoritma yang akan dijalankan, pada GPU nantinya akan dijalankan secara bersamaan di tiap-tiap *core* pada GPU. Pada pengolahan *image*, akan dilakukan proses perhitungan atau komputasi pada tiap-tiap piksel yang artinya akan ada banyak sekali proses komputasi yang diulang-ulang namun data pikselnya saja yang berbeda. Maka dari itu dengan memanfaatkan GPU, komputasi di tiap-tiap piksel dari *image* secara bersamaan. Untuk mewujudkan hal tersebut maka GPU yang digunakan menggunakan teknologi dari produsen GPU yaitu Nvidia dengan produk GPU yang bernama *Compute Unified Device Architecture* (CUDA) yang berbasis bahasa pemrograman C/C++.