BAB II

TINJAUAN PUSTAKA DAN DASAR TEORI

2.1 Tinjauan Pustaka

Pengenalan wajah akan lebih efektif dan efisien dengan metode yang tepat. Metode dan teknologi yang akan di pakai dan di bahas pada aplikasi adalah *Viola Jones* dan *OpenCV*. Sistem dan aplikasi pengenalan wajah sebelumya sudah banyak di buat, tetapi metode dan program aplikasinya yang di gunakan berbeda-beda.

Sigit wasista, dkk (2011) dengan judul *Sistem Pengenalan Wajah Pada Mesin Absensi Mahasiswa Menggunakan Metode PCA Dan DTW* membuat sistim pengenalan wajah dengan metode PCA dan DTW. Metode PCA di gunakan untuk ekstraksi fitur, dan metode DTW digunakan untuk pengambilan keputusan.

Sepritahara (2009) dengan judul "Sistem Pengenalan Wajah (*Face Recognition*) Menggunakan Metode *Hidden Markov Model* (HMM)". Hasil penelitiannya menunjukan Metode *Hidden Markov Model* (HMM) mampu menangani perubahan statistik dari gambar, dengan memodelkan eleman-eleman menggunakan probabilitas. Salah satu aplikasinya adalah pada image processing.

Ratna nur azizah (2008) dengan judul *Pengenalan Wajah Dengan Metode Subspace Lda.* Dalam penelitiannya membat sistem pengenalan wajah menggunakan metode linear discriminant analysis (LDA). Tujuan metode LDA adalah mencari proyeksi *linier* (yang biasa disebut dengan '*fisherimage'*) untuk memaksimumkan

matriks kovarian antar kelas (*between-class covariance matrix*) sekaligus meminimumkan matriks kovarian dalam kelas (*within-class covariance matrix*), agar anggota di dalam kelas lebih terkumpul penyebarannya dan pada akhirnya dapat meningkatkan keberhasilan pengenalan.

Berdasarkan beberapa tinjauan pustaka tersebut peneliti ingin membuat sebuah program dengan menggunakan metode *Viola Jones* sebagai pendeteksi wajah, *Eigenface* untuk sistem pengenalan wajah dan *OpenCV* sebagai platform untuk penyajian data dari ke dua metode tersebut untuk membuat aplikasi pengenalan wajah. Tinjauan pustaka di atas dapat di simpulkan dengan lebih jelas pada tabel 2.1

Tabel 2.1 tinjauan pustaka

No	Nama	Objek	Metode 1	Metode 2	hasil
1	Sepritahara (2009)	manusia	*Hidden markov model* (HMM)	Tidak ada	Sistem pengenalan wajah (*face recognition*) dengan menggunakan metode *Hidden Markov Model* (HMM) dapat mengenali gambar sesuai dengan label (nama) yang diberikan pada *database* dan tidak dapat mengenali gambar(namanya tidak sesuai dengan nama yang diberikan pada database).

Tabel 2.1 (Lanjutan)

2	Sigit wasista, Bima sena bayu D, Sandra agstyan putra (2011)	Mahasiswa	PCA	DTW	hasil pengujian, tingkat keberhasilan pengenalan menggunakan DTW sebesar 20% dan 40% hingga 82% dan PCA didapatkan tingkat keakurasian pengenalan sebesar 90.833%
3	Ratna nur azizah (2008)	Manusia	*Subspace* LDA	Tidak ada	Semakin ekstrim pencahayan pada maka semakin turun persentase laju pengenalan dan semakin banyak citra training dalam suatu kelas maka semakin naik persentase laju pengenalan
4	Fika Tiara Puri	Karyawan pengajaran dosen	*Eigenface*	Tidak ada	Metode *Eigenface* berhasil mencocokan wajah yang berada di dalam *database* dengan *capture* wajah secara *real time*
5	Arezky Willyadi T (diusulkan)	Mahasiswa	*Viola Jones*	*Eigenface*	

2.2 Dasar Teori

Dasar teori digunakan untuk memahami definisi, pengertian dasar dan istilah yang digunakan dalam penelitian ini. Berikut dasar teori yang digunakan.

1 Pengenalan Wajah

Pengenalan wajah adalah suatu metoda pengenalan yang berorientasi pada wajah. Pengenalan ini dapat dibagi menjadi dua bagian yaitu di kenali atau tidak dikenali, setelah dilakukan perbandingan dengan pola yang sebelumnya disimpan di dalam database. Secara umum, sistem pengenalan citra wajah dibagi menjadi 2 jenis, yaitu sistem *feature based* dan *image based*. Pada sistem pertama digunakan fitur yang diekstraksi dari komponen citra wajah (mata, hidung, mulut, dll) yang kemudian hubungan antara fitur-fitur tersebut dimodelkan secara geometris. Sedangkan sistem kedua menggunakan informasi mentah dari piksel citra yang kemudian direpresentasikan dalam metode tertentu, yang kemudian digunakan untuk klasifikasi identitas citra (Fatta, 2009).

2 Pendeteksi Wajah

Proses pendeteksi wajah ini bekerja dengan cara memeriksa citra yang dimasukan, apakah memiliki citra wajah atau tidak, jika memiliki, maka akan dilakukan pemisahan dengan cara memotong citra wajah dari latar belakang citra yang dimasukan. Jika masukan berbentuk video, proses yang dilakukan adalah proses *face tracking*. Secara umum, proses *face tracking* dan proses pendeteksian wajah

mempunyai fungsi yang sama. Perbedaannya terletak pada proses pendeteksiannya saja, jika pada masukan berbentuk citra, sistem berjalan offline sehingga dapat menggunakan proses pendeteksian wajah, sedangkan pada masukan video, sistem berjalan secara online atau realtime yang membutuhkan pendeteksian secara langsung maka proses yang digunakan adalah proses face tracking (Salamun, 2016).

3 Penyelarasan Wajah

Pada proses pendeteksian wajah, citra wajah yang didapatkan masih berupa perkiraan kasar atau masih memiliki kualitas yang cukup buruk seperti ukuran yang berbeda dengan ukuran normal, faktor pencahayaan yang kurang atau lebih, kejelasan citra yang buruk dan sebagainya. Maka perlu diadakannya proses penyelarasan. Proses penyelarasan wajah merupakan proses yang bertujuan untuk menormalisasi wajah dari citra wajah yang didapatkan dari proses pendeteksian wajah. Proses ini terdiri dari tahapan-tahapan sebagai berikut:

1. *Grayscaling*

   Grayscaling citra merupakan tahapan pertama dari proses penyelarasan, pada tahap ini terjadi pengkonversian citra warna RGB menjadi citra berwarna abu. Citra warna RGB terdiri dari 3 parameter warna yaitu merah (*red*), hijau (*green*) dan biru (*blue*), jika citra warna RGB ini dimasukan ke dalam proses ekstraksi, maka proses tersebut akan sulit untuk dilakukan karena citra RGB terdiri dari 3 parameter, oleh karena itu diperlukan penyamaan parameter yaitu dengan melakukan tahap *grayscaling* ini.

2. Pemotongan

   Pada tahapan ini terjadi pemotongan citra yang memisahkan citra wajah dengan citra masukannya, tujuannya untuk mengambil citra yang hanya diperlukan untuk proses ekstraksi, dalam hal ini adalah citra wajah dan membuang citra lain yang tidak diperlukan. Dimensi citra yang dipotong disesuaikan dengan dimensi dari proses segmentasi atau pengkotakan objek wajah yang dilakukan pada proses pendeteksian wajah.

3. *Resizing* (tahap normalisasi dimensi citra)

   Pada tahap *resizing* citra, terjadi proses normalisasi dimensi citra wajah, yaitu proses pembesaran atau pengecilan dimensi citra wajah menjadi dimensi yang telah ditentukan. Tujuannya, untuk menyamakan dimensi wajah dari tiap citra yang dimasukan, sehingga pada proses ekstraksi citra nanti tidak ada perbedaaan dimensi dari matriks data citra wajah.

4. *Equalizing* (tahap koreksi tingkat kecerahan citra)

   Tahap ini adalah tahapan terakhir dari proses penyelarasan, yang tujuannya untuk memperjelas nilai *histogram* dari citra wajah hasil tahapan-tahapan sebelumnya (Salamun, 2016).

4 Ekstraksi Wajah

Ekstraksi fitur adalah proses untuk mendapatkan ciri-ciri pembeda yang membedakan suatu sampel wajah dari sampel wajah yang lain. Ekstraksi fitur juga merupakan proses yang berfungsi untuk mendapatkan informasi yang efektif dan

berguna untuk membedakan wajah-wajah orang yang berbeda dari citra wajah yang telah diselaraskan. Proses ini dilakukan menggunakan algoritma-algoritma ekstraksi seperti *Principal Component Analysis* (PCA), *Linear Discriminant Analysis* (LDA), *Independent Component Analysis* (ICA) dan sebagainya. Informasi yang didapatkan dari ekstraksi fitur disebut vektor fitur, yaitu bentuk dasar pencarian citra berbasis konten, yang menangkap properti citra seperti warna dan tekstur (Salamun, 2016)

5 Penyimpanan Fitur Wajah

Proses penyimpanan fitur merupakan tahapan terakhir dari proses pelatihan citra wajah. Proses ini berfungsi untuk menyimpan fitur hasil ekstraksi citra wajah yang ada di dalam database ke dalam sebuah file berekstensi *.xml. File inilah yang nantinya akan digunakan untuk proses pencocokan antara citra wajah yang diuji dengan hasil ekstraksi fitur yang terdapat pada file ini (Salamun, 2016).

6 Pencocokan Fitur Wajah

Pencocokan fitur merupakan proses perbandingan fitur yang telah diekstrak dari citra uji dengan fitur citra wajah dari database, yang sebelumnya telah melalui proses pelatihan citra.

Dari proses perbandingan fitur tersebut akan menghasilkan nilai jarak terdekat yang menandakan nilai fitur citra uji hampir menyamai dengan fitur citra latih. Nilai jarak ini akan menjadi nilai masukan untuk nilai kemiripan citra (Salamun, 2016).

Nilai kemiripan citra merupakan nilai tingkat kemiripan citra uji dengan citra latih, semakin besar nilainya menandakan bahwa orang yang sedang diamati adalah orang yang sama dengan orang yang citra wajahnya telah disimpan dalam database. Nilai kemiripan ini sebelum mengeluarkan output hasil pengenalan wajah akan melalui proses threshold terlebih dahulu.

Proses threshold pengenalan wajah adalah proses penyaringan nilai kemiripan citra, dimana jika nilai kemiripan berada dibawah threshold yang telah ditentukan maka output hasil pengenalan tidak akan ditampilkan. Jika berada diatas output hasil pengenalan akan ditampilkan. Tujuannya agar output hasil pengenalan yang ditampilkan bernilai benar, karena memiliki nilai kemiripan yang tinggi. Setelah, proses-proses tersebut dilakukan proses terakhir adalah proses pencarian tingkat keakuratan. Keakuratan adalah kondisi yang menunjukan kebenaran atau ketepatan suatu objek yang diamati, dalam sistem pengenalan ini adalah kebenaran bahwa citra wajah yang diujikan benar-benar citra wajah orang yang dimaksud (Salamun, 2016).

2.2.1 *Viola Jones*

Metode *Viola-Jones* merupakan metode pendeteksian obyek yang memiliki tingkat keakuratan yang cukup tinggi yaitu sekitar 93,7 % dengan kecepatan 15 kali lebih cepat daripada detektor Rowley Baluja-Kanade dan kurang lebih 600 kali lebih cepat daripada detektor Schneiderman Kanade. Metode ini, diusulkan oleh Paul Viola dan Michael Jones pada tahun 2001 (Viola, 2003).

Metode *Viola Jones* adalah metode yang memiliki kapabilitas untuk mendeteksi wajah pada suatu citra gambar dengan tingkat keakuratan yang cukup tinggi. Pada standarnya pengolahan citra pada gambar akan dilakukan dengan merubah skala ukuran gambar kedalam ukuran yang ditentukan dan melakukan pendeteksian dengan skala tetap pada gambar tersebut. Dengan hal tersebut simpulkan bahwa waktu untuk pendeteksian akan lebih cepat dibandingkan dengan melakukan pendeteksian terhadap citra gambar yang berukuran berbedabeda.

Berbeda halnya dengan metode *Viola Jones* yang terus merubah ukuran skala pendeteksian dibandingkan harus merubah ukuran dari skala gambar dan menjalankan banyak pendeteksian terhadap gambar dengan ukuran skala pendektesian yang berbeda tiap waktunya. Hal ini diperkirakan akan membutuhkan lebih banyak waktu, tapi Metode Viola Jones memiliki fitur pendeteksian yang mengkonstruksi gambar kedalam gambar integral dan beberapa fitur persegi yang simple yang disebut *Haar Wavelets*. Sehingga hasil kalkulasi dalam skala pendeteksian akan sama berapapun ukuran dari gambar yang diproses (M. Yogi, 2014).

Tahapan Metode *Viola Jones*

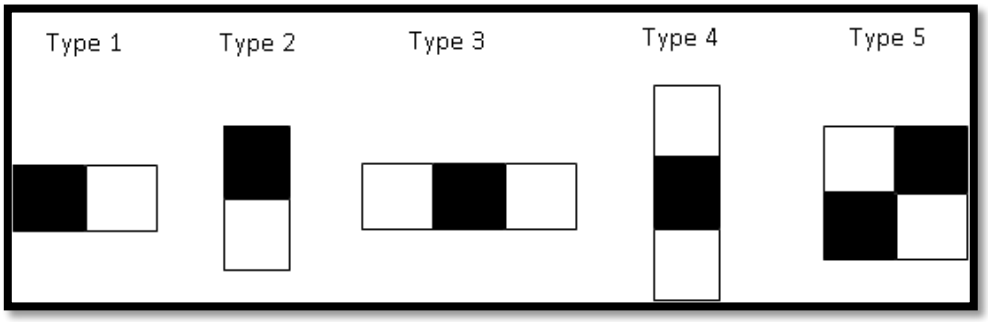
Adapun tahapan metode *viola jones* dapat di jelaskan sebagai berikut :

1. Proses *Haar-Like Feature*

Haar-Like Feature digunakan dalam mendeteksi objek pada *image digital*. *Haar-like feature* memproses gambar dalam kotak-kotak, dimana dalam satu kotak terdapat beberapa pixel. Per kotak itu pun kemudian di-proses dan didapatkan perbedaan nilai

(*threshold*) yang menandakan daerah gelap dan terang. Nilai– nilai inilah yang nantinya dijadikan dasar dalam *image processing*.

Adanya fitur *Haar* ditentukan dengan cara mengurangi rata-rata piksel pada daerah gelap dari rata-rata piksel pada daerah terang. Jika nilai perbedaannya itu di atas nilai ambang atau treshold, maka dapat dikatakan bahwa fitur tersebut ada. Contoh *Haar-Like Fiture* disajikan dalam gambar 2.1 (M. Yogi, 2014).

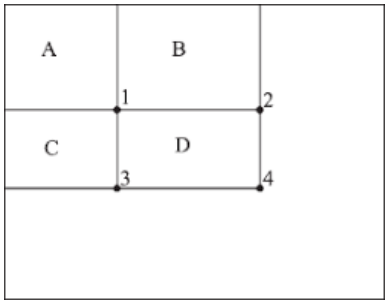


Gambar 2.1 *Haar Like Feature*

2. Proses *Integral Image*

Integral Image berfungsi untuk menentukan ada atau tidaknya dari ratusan fitur *Haar* pada sebuah gambar dan pada skala yang berbeda secara efisien digunakan. Pada umumnya, pengintegrasian tersebut menambahkan unit-unit kecil secara bersamaan. Dalam hal ini unit-unit kecil tersebut adalah nilai-nilai piksel. Nilai *integral* untuk masing-masing piksel adalah jumlah dari semua piksel-piksel dari atas sampai bawah.

Di mulai dari kiri atas sampai kanan bawah, keseluruhan gambar itu dapat di jumlahkan dengan beberapa operasi bilangan bulat perpiksel.



Gambar 2.2 perhitungan nilai fitur

Dengan menggunakan integral image dapat mengetahui nilai piksel untuk beberapa segiempat misalkan, seperti segiempat D pada Gambar 2.2 dapat dilakukan dengan cara menggabungkan jumlah piksel pada area segiempat A+B+C+D, dikurangi jumlah dalam segiempat A+B dan A+C, ditambah jumlah piksel di dalam A. Dengan A+B+C+D adalah nilai dari integral image pada lokasi 4, A+B adalah nilai pada lokasi 2, A+C adalah nilai pada lokasi 3, dan A pada lokasi 1. Sehingga hasil dari D dapat dikomputasikan D = (A+B+C+D)- (A+B)- (A+C)+A (M. Yogi, 2014).

3. Proses *AdaBoost Machine Learning*

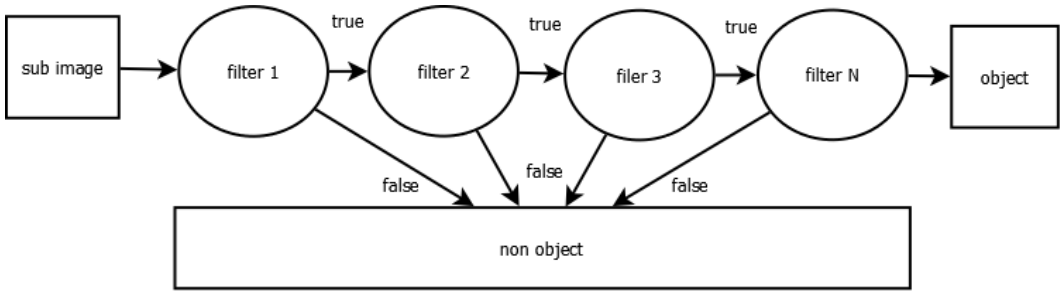
Boosting merupakan meta-algoritma dalam *machine learning* untuk melakukan *supervised learning*. *AdaBoost* merupakan salah satu algoritma *boosting* yang berfungsi untuk melakukan pemilihan fitur-fitur dalam jumlah banyak dengan hanya memilih fitur-fitur tertentu. Hal ini dilakukan dengan mengevaluasi setiap fitur

terhadap data latih dengan menggunakan nilai dari fitur tersebut, fitur yang memiliki Batasan terbesar antara objek dan non-objek dianggap sebagai fitur terbaik

AdaBoost Learning juga dapar digunakan untuk meningkatkan kinerja klasifikasi dengan pembelajaran sederhana untuk menggabungkan banyak *classifier* lemah menjadi satu *classifier* kuat. *Classifier* lemarh adalah suatu jawaban benar dengan tingkat kebenaran yang kurang akurat (M. Yogi, 2014).

4. Proses *cascade classifier*

Cascade classifier adalah sebuah metode untuk mengkombinasikan *classifier* yang kompleks dalam sebuah struktur bertingkat yang dapat meningkatkan kecepatan pendeteksian obyek dengan memfokuskan pada daerah citra yang berpeluang saja seperti pada gambar 2.3 (M. Yogi, 2014).



Gambar 2.3 *cascade classifier*

Penerapan Viola Jones Pada Aplikasi

Algoritma *Viola Jones* di implementasikan pada penelitian ini karena merupakan metode pendeteksi wajah yang didukung oleh *OpenCV / EmguCV* dan memiliki hasil yang baik. Dalam metode ini, *OpenCV* (*EmguCV*) menggunakan jenis detektor wajah yang disebut fitur *Haar Cascade*, yang diperkenalkan oleh *Viola* dan *Jones*. *Haar Cascade* adalah *classifier* (detektor) yang dilatih pada ribuan wajah manusia. Data pelatihan ini disimpan dalam file XML, dan kemudian digunakan oleh *classifier* selama deteksi.

classifier dapat diartikan dengan 'detektor' gambar, yang harus dideteksi, dapat berasal dari hard drive lokal atau langsung dari kamera. Ketika objek (wajah) *Viola Jones classifier /* detektor berjalan di atas gambar atau kamera secara *real time*, proses tersebut menggunakan data terlatih yang disimpan dalam file XML. Dengan bantuan data ini, classifier (detektor) memutuskan apakah objek yang diidentifikasi adalah objek yang dipermasalahkan atau tidak. dalam hal deteksi wajah, ia memutuskan apakah itu "Wajah" atau "Bukan Wajah".

Viola Jones Haar Classifier adalah detektor generik, yang klasifikasinya tergantung pada data terlatih yang disimpan dalam file XML yang digunakan. jadi objek apa pun yang digunakan untuk melatih file XML ini, *classifier* (detektor) akan mendeteksi objek itu dalam gambar. Misal jika file XML menyimpan data yang terlatih untuk wajah manusia, maka *Viola Jones Haar Classifier* menjadi "*face detector*". tetapi jika file XML memiliki data yang disimpan tentang mobil, maka *classifier* (detektor) berfungsi sebagai detektor mobil (Mahvish Nasir, 2011).

Berikut adalah tahapan penerapan metode *Viola Jones* ke dalam aplikasi :

1. Deklarasi Variabel *Haarcascade*

Klasifikasi / detektor *Viola Jones* yang digunakan dalam *EmguCV* untuk mendeteksi wajah dalam sebuah gambar disebut *HaarCascade*. Mendeklarasikan objek global kelas *HaarCascade* dan menyebutnya *'face'*.

```
HaarCascade face;
```

2. Load File XML

Klasifikasi menggunakan data yang disimpan dalam file XML untuk memutuskan bagaimana mengklasifikasikan setiap objek pada gambar. Jadi, fitur *haar* memerlukan beberapa file XML untuk memuat data yang terlatih, yang berupa data wajah. pada penelitian ini, akan digunakan file XML yang disebut "haarcascade_frontalface_default.xml".

```
face = new HaarCascade("haarcascade_frontalface_default.xml");
```

3. Tahap Pendeteksi Wajah

Pada tahap pendeteksi setelah gambar di tangkap secara *real time* dimulai dengan konversikan gambar ke dalam bentuk *Grayscale* (abu-abu) karena detektor pendeteksi wajah hanya dapat berjalan pada skala *Grayscale.*

```
gray = currentFrame.Convert<Gray, Byte>();
```

Setelah mengkonversi gambar ke skala *Grayscale* (abu-abu), perintah berikut adalah panggilan untuk menjalankan detektor wajah pada gambar skala *Grayscale* ini (disebut grayframe di sini) (Mahvish Nasir, 2011).

```
MCvAvgComp[][] facesDetected = gray.DetectHaarCascade(
face,
1.2,
10,
Emgu.CV.CvEnum.HAAR_DETECTION_TYPE.DO_CANNY_PRUNING,
new Size(20, 20));
```

2.2.2 *Eigenface*

Metode *Eigenface* dianggap sebagai teknologi pengenalan wajah otomatis pertama yang pernah diciptakan. Teori ini dikembangkan oleh Turk dan Petland. Teori ini dikembangkan dengan membagi sebuah citra wajah menjadi data set fitur karakteristik yang disebut *Eigenface*. Fitur karakteristik ini merupakan komponen utama (*principal component*) dari *training* set awal dari citra wajah. Penelitian yang dilakukan oleh Carey dan Diamond menunjukkan bahwa fitur wajah sebuah individu dan hubungan langsung antar fitur tersebut tidak dapat menyamai kemampuan manusia dalam memperhatikan dan mengenal wajah (Indra, 2012)

Untuk menghasilkan *Eigenface*, sekumpulan citra digital dari wajah manusia diambil pada kondisi pencahayaan yang sama kemudian dinormalisasikan dan diproses pada resolusi yang sama (misal m x n), kemudian citra tadi diperlakukan sebagai vektor dimensi m x n dimana komponennya diambil dari nilai piksel citra.

Prinsip dasar dari pengenalan wajah adalah dengan mengutip informasi unik wajah tersebut kemudian di-*encode* dan dibandingkan dengan hasil *decode* yang

sebelumnya dilakukan. Dalam metode *Eigenface*, *decoding* dilakukan dengan menghitung *eigenvector* kemudian direpresentasikan dalam sebuah matriks yang berukuran besar. *Eigenvector* juga dinyatakan sebagai karakteristik wajah oleh karena itu metode ini disebut dengan *Eigenface*. Setiap wajah direpresentasikan dalam kombinasi *linear Eigenface*.

Prinsip dasar dari metode *Eigenface* adalah bagaimana caranya untuk mengekstrak informasi yang *relevan* dari sebuah citra wajah lalu mengubahnya ke dalam satu set kode yang paling efisien, dan membandingkan kode wajah ini dengan *database* berisi beragam wajah yang telah dikodekan secara serupa.

Algoritma selengkapnya dapat di lihat pada langkah-langkah berikut :

1. Langkah pertama adalah menyiapkan data dengan membuat suatu himpunan *S* yang terdiri dari seluruh *training image* ($\Gamma_1, \Gamma_2, \ldots\ldots \Gamma_M$).

   $$S = \{\Gamma_1, \Gamma_2, \ldots\ldots \Gamma_M\}$$

2. Langkah kedua adalah ambil nilai tengah atau *mean* (Ψ)

   $$\Psi = \frac{1}{M} \sum_{n=1}^{M} \Gamma n$$

3. Langkah ketiga kemudian cari selisih (Φ) antara *Training image* (Γ_i) dengan nilai tengah (Ψ), apabila ditemukan nilainya dibawah nol ganti nilainya dengan nol.

   $$\Phi_i = \Gamma_i - \Psi$$

Tahapan Pengenalan

1. Sebuah *image* wajah baru atau *test face (Γ_{new})*, akan dicoba untuk dikenali, pertama terapkan cara pada tahapan pertama perhitungan *Eigenface* untuk mendapatkan nilai *Eigenface* dari *image* tersebut.

$$\mu_{new} = v.(\Gamma_{new} - \Psi)$$

$$\Omega = [\mu_1, \mu_2, \mu_3, \dots , \mu_M]$$

2. Gunakan metode *Euclidean Distance* untuk mencari jarak *(distance)* terpendek antara nilai *Eigenface* dari *training image* dalam *database* dengan *eigenface image test face*.

$$\varepsilon_k = || \Omega - \Omega_k ||$$

(Rizky, 2015).

Impementasi Algoritma

Penyusunan *FlatVector* Matrix

Langkah pertama adalah menyusun seluruh *training image* menjadi 1 matrix tunggal. Misalnya image yang disimpan berukuran H x W piksel dan jumlahnya N buah, maka memiliki *flatvector* dengan dimensi N x (H x W). representasikan semua matriks *training* menjadi matriks dengan bentuk N×1 atau matriks *linier* seperti yang ditunjukkan berikut ini:

$$\begin{bmatrix} a & b & c \\ x & y & z \end{bmatrix} > [a \quad b \quad c \quad x \quad y \quad z]$$

Misalnya didalam *training image* terdapat tiga *image* ukuran 3x3 piksel maka akan mempunyai *eigenvector* ukuran 2x9. Contoh dibawah ini mengunakan empat wajah citra yang telah diubah menjadi matrix, lalu matrix tersebut diubah kedalam bentuk rataan *FlatVector*.

$$C_1 \rightarrow \begin{bmatrix} 10 & 10 & 10 \\ 10 & 10 & 10 \\ 10 & 10 & 10 \end{bmatrix} \rightarrow [10 \quad 10 \quad 10 \quad 10 \quad 10 \quad 10 \quad 10 \quad 10 \quad 10]$$

$$C_2 \rightarrow \begin{bmatrix} 2 & 2 & 2 \\ 2 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix} \rightarrow [2 \quad 2 \quad 2 \quad 2 \quad 2 \quad 2 \quad 2 \quad 2 \quad 2]$$

$$C_3 \rightarrow \begin{bmatrix} 11 & 11 & 11 \\ 11 & 11 & 11 \\ 11 & 11 & 11 \end{bmatrix} \rightarrow [11 \quad 11 \quad 11 \quad 11 \quad 11 \quad 11 \quad 11 \quad 11 \quad 11]$$

$$C_4 \rightarrow \begin{bmatrix} 9 & 9 & 9 \\ 9 & 9 & 9 \\ 9 & 9 & 9 \end{bmatrix} \rightarrow [9 \quad 9 \quad 9 \quad 9 \quad 9 \quad 9 \quad 9 \quad 9 \quad 9]$$

Hitung Rataan FlatVector

Dari *FlatVector* yang diperoleh, jumlahkan seluruh barisnya sehingga diperoleh matrix berukuran 1 x (H x W).

$$C_1 + C_2 + C_3 + C_4 = \begin{bmatrix} 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 11 & 11 & 11 & 11 & 11 & 11 & 11 & 11 & 11 \\ 9 & 9 & 9 & 9 & 9 & 9 & 9 & 9 & 9 \end{bmatrix}$$

$$C_1 + C_2 + C_3 + C_4 = [32 \quad 32 \quad 32 \quad 32 \quad 32 \quad 32 \quad 32 \quad 32 \quad 32]$$

Setelah itu bagi matriks tadi dibagi dengan jumlah image N untuk mendapatkan Rataan *FlatVector*.

$$\frac{[32 \quad 32 \quad 32 \quad 32 \quad 32 \quad 32 \quad 32 \quad 32 \quad 32]}{4}$$

Rataan Flatvector = [8 8 8 8 8 8 8 8 8]

Nilai *flatvector* citra akan digunakan untuk menghitung nilai *Eigenface* citra wajah untuk *training image*.

Tentukan Nilai Eigenface

Dengan memakai rataan *flatvector* citra di atas nilai eigenface untuk matriks flatvector yang sudah disusun tersebut dapat dihitung nilai *eigenface*-nya. Caranya dengan mengurangi baris-baris pada matrix *flatvector* dengan rataan *flatvector*. Jika didapatkan nilai dibawah nol, maka nilainya diganti dengan nol. Tentukan Nilai *Eigenface*.

$$C_1 = \begin{array}{r} 10 \quad 10 \quad 10 \quad 10 \quad 10 \quad 10 \quad 10 \quad 10 \quad 10 \\ 8 \quad 8 \quad 8 \quad 8 \quad 8 \quad 8 \quad 8 \quad 8 \quad 8 \\ \hline 2 \quad 2 \quad 2 \quad 2 \quad 2 \quad 2 \quad 2 \quad 2 \quad 2 \end{array}$$

$$C_2 = \begin{array}{r} 2 \quad 2 \quad 2 \quad 2 \quad 2 \quad 2 \quad 2 \quad 2 \quad 2 \\ 8 \quad 8 \quad 8 \quad 8 \quad 8 \quad 8 \quad 8 \quad 8 \quad 8 \\ \hline 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \end{array}$$

$$C_3 = \begin{array}{r} 11 \quad 11 \quad 11 \quad 11 \quad 11 \quad 11 \quad 11 \quad 11 \quad 11 \\ 8 \quad 8 \quad 8 \quad 8 \quad 8 \quad 8 \quad 8 \quad 8 \quad 8 \\ \hline 3 \quad 3 \quad 3 \quad 3 \quad 3 \quad 3 \quad 3 \quad 3 \quad 3 \end{array}$$

$$C_4 = \begin{array}{r} 9 \quad 9 \quad 9 \quad 9 \quad 9 \quad 9 \quad 9 \quad 9 \quad 9 \\ 8 \quad 8 \quad 8 \quad 8 \quad 8 \quad 8 \quad 8 \quad 8 \quad 8 \\ \hline 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \end{array}$$

Proses Identifikasi

Untuk mengenali citra tes (*testface*), langkah identifikasinya adalah hitung nilai *Eigenface* untuk matriks *testface* dengan cara sebelumnya untuk penentuan nilai *Eigenface* dan *flatvector* citranya.

$$C_t \rightarrow \begin{bmatrix} 9 & 9 & 9 \\ 9 & 7 & 9 \\ 9 & 7 & 9 \end{bmatrix} \rightarrow [9 \quad 9 \quad 9 \quad 9 \quad 7 \quad 9 \quad 9 \quad 7 \quad 9]$$

$$\begin{array}{r} 9\ 9\ 9\ 9\ 7\ 9\ 9\ 7\ 9 \\ -\ 8\ 8\ 8\ 8\ 8\ 8\ 8\ 8\ 8 \\ \hline 1\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 1 \end{array}$$

Nilai *eigen* dari *test face* = [1 1 1 1 0 1 1 0 1]

Nilai *eigen* (*eigenvalue*) dari *testface* digunakan untuk identifikasi dengan menentukan jarak terpendek dengan *Eigenface* dari *eigenvector training* dengan cara menentukan nilai absolut dari pengurangan baris I pada matriks *Eigenface training* citra dengan *eigenface* dari *testface* dan jumlahkan dengan elemen penyusun *vector* yang dihasilkan dari pengurangan dan didapat jarak d indeks I dan cari nilai yang paling kecil.

Setelah nilai *Eigenface* untuk *testface* diperoleh, dapat dilakukan proses identifikasi dengan menentukan jarak terpendek dengan *Eigenface* dari *eigenvector training image*. Pertama tentukan nilai absolut dari pengurangan baris pada *matrix eigenface training image* dengan *Eigenface* dari *testface*, kemudian jumlahkan elemen-elemen penyusun vector yang dihasilkan dari pengurangan tadi dan ditemukan jarak indeks. Lakukan untuk semua baris, dan cari nilai yang paling kecil.

Nilai Eigenface C_1 = [2 2 2 2 2 2 2 2 2]

$$C_1 = \begin{array}{r} 2\ \ 2\ \ 2\ \ 2\ \ 2\ \ 2\ \ 2\ \ 2\ \ 2 \\ 1\ \ 1\ \ 1\ \ 1\ \ 0\ \ 1\ \ 1\ \ 0\ \ 1 \\ \hline 1\ \ 1\ \ 1\ \ 1\ \ 2\ \ 1\ \ 1\ \ 2\ \ 1 \end{array}$$

C_1 = 1+1+1+1+2+1+1+2+1=11

Nilai Eigenface C_2 = [0 0 0 0 0 0 0 0 0]

$$C_2 = \begin{array}{r} 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0 \\ 1\ \ 1\ \ 1\ \ 1\ \ 0\ \ 1\ \ 1\ \ 0\ \ 1 \\ \hline -1\ \ -1\ \ -1\ \ -1\ \ 0\ \ -1\ \ -1\ \ 0\ \ -1 \end{array}$$

C_2 = 1+1+1+1+0+1+1+0+1=7

Nilai Eigenface C_3 = [3 3 3 3 3 3 3 3 3]

$$C_3 = \begin{array}{r} 3\ \ 3\ \ 3\ \ 3\ \ 3\ \ 3\ \ 3\ \ 3\ \ 3 \\ 1\ \ 1\ \ 1\ \ 1\ \ 0\ \ 1\ \ 1\ \ 0\ \ 1 \\ \hline 2\ \ 2\ \ 2\ \ 2\ \ 3\ \ 2\ \ 2\ \ 3\ \ 2 \end{array}$$

C_3 = 2+2+2+2+3+2+2+3+2=20

Nilai Eigenface C_4 = [1 1 1 1 1 1 1 1 1]

$$C_4 = \begin{array}{r} 1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1 \\ 1\ \ 1\ \ 1\ \ 1\ \ 0\ \ 1\ \ 1\ \ 0\ \ 1 \\ \hline 0\ \ 0\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0\ \ 1\ \ 0 \end{array}$$

C_4 = 0+0+0+0+1+0+0+1+0=2

Dari hasil perhitungan, diperoleh jarak citra wajah empat memiliki nilai yang terkecil yaitu dua. Karena jarak *Eigenface face* satu dengan *testface* yang paling kecil, maka hasil identifikasi menyimpulkan bahwa *testface* lebih mirip dengan *face* empat dari pada *face* satu, *face* dua, dan *face* tiga (Indra, 2012).

2.2.3 *OpenCV*

OpenCV merupakan singkatan dari *Intel Open Source Computer Vision Library* yang sekurang-kurangnya terdiri dari 300 fungsi-fungsi C, bahkan bisa lebih. *Software* ini gratis, dapat digunakan dalam rangka komersil maupun non komersil, tanpa harus membayar lisensi ke intel. *OpenCV* dapat beroperasi pada komputer berbasis Windows ataupun Linux. *Library OpenCV* adalah suatu cara penerapan bagi komunitas *open source vision* yang sangat membantu dalam kesempatan meng-*update* penerapan *computer vision* sejalan dengan pertumbuhan PC (*personal computer*) yang terus berkembang. *Software* ini menyediakan sejumlah fungsi-fungsi *image processing*, seperti halnya dengan fungsi-fungsi analisis gambar dan pola (Kurniawan, 2009).

Beberapa contoh aplikasi dari *OpenCV* adalah pada *Human-Computer Interaction* (interaksi manusia komputer); *Object Indentification* (Identifikasi Objek), Segmentation (segmentasi) dan *Recognition* (pengenalan); *Face Recognition* (pengenalan wajah); *Gesture Recognition* (pengenalan gerak isyarat), *Motion Tracking* (penjajakan gerakan), *Ego Motion* (gerakan ego), dan *Motion Understanding* (pemahaman gerakan); *Structure From Motion* (gerakan dari struktur); dan *Mobile Robotics* (robot-robot yang bergerak) (Kurniawan, 2009).

2.2.4 *EmguCV*

EmguCv adalah jembatan untuk menghubungkan antara C# dan *OpenCv*. Dengan *EmguCv,* fungsi-fungsi dalam *OpenCv* bisa dipanggil melalui bahasa pemrograman yang compatible dengan .Net seperti C#, VB, dan VC++. Keuntungan

menggunakan *EmguCv* yang paling utama adalah *library* ini sepenuhnya ditulis dengan bahasa pemrograman C# yang tentunya lebih aman. Dengan *EmguCv* dapat dibuat aplikasi seperti layaknya menggunakan *OpenCv* (Rizki, 2015).

EmguCV terdiri dari 2 layer, yaitu *basic* layer dan *second* layer. *Basic* layer mengandung fungsi, struktur, dan enumerasi yang secara langsung merefleksikan apa yang ada di *OpenCV*. Dengan adanya layer inilah kita bisa memanggil fungsi-fungsi pada *OpenCV* dengan bahasa pemrograman C#. Sedangkan *second* layer mengandung kelas-kelas yang memanfaatkan keunggulan teknologi .NET.

Untuk memulai EMGU diperlukan beberapa tambahan referensi file DLL pada *project* yang dibuat untuk dapat mengakses *library* EMGU dan melakukan proses pengolahan citra wajah. file tersebut terdapat di dalam instalasi *library* EMGU.

- Emgu.CV.dll
- Emgu.CV.UI.dll
- Emgu.Util.dll

Sekarang untuk *library* c ++ yang lebih rumit, untuk memuat, menampilkan, mengakses data gambar dan melakukan banyak fungsi yang lebih sederhana, dibutuhkan beberapa file *library* dari OpenCV.

- opencv_core220.dll
- opencv_imgproc220.dll

(C Johnson, 2011).