Jurnal Coding, Sistem Komputer Untan
Volume 03, No. 1 (2015). Hal 41-50 ISSN : 2338-493X

IMPLEMENTASI PENGENALAN WAJAH DENGAN METODE *EIGENFACE* PADA SISTEM ABSENSI

[1]Muhammad Rizki Muliawan, [2]Beni Irawan, [3]Yulrio Brianorman
[1] [2] [3]Jurusan Sistem Komputer, Fakultas MIPA Universitas Tanjungpura
Jalan Prof. Dr. H. Hadari Nawawi, Pontianak
Telp./Fax.: (0561) 577963
e-mail:
[1]rizki5991@gmail.com, [2]benicsc@yahoo.com, [3]yulrio.brianorman@siskom.untan.ac.id

Abstrak

Sistem pengenalan wajah yang termasuk dalam bidang pengolahan citra dapat dipadukan dengan sistem absensi sehingga menjadi salah satu hal yang menarik untuk dilakukan, dimana nantinya sistem absensi juga dapat dilakukan dengan wajah. Proses pembuatan aplikasi absensi dengan pengenalan wajah ini menggunakan algoritma Eigenface yang terdapat pada OpenCv. Secara umum proses absensi menggunakan pengenalan wajah ini dilakukan dengan memasukkan data wajah terlebih dahulu beserta password dari masing-masing orang, setelah itu dilakukan proses pemindaian untuk proses absensi. Metode eigenface dari opencv ini mencari data wajah yang mendekati dengan data wajah yang ada di database. Pada pengujian penelitian ini hasil yang didapat berbeda-beda antara wajah satu dengan wajah yang lainnya, pada saat database berisi 10 data wajah, hasil rata-rata persentase kecocokan mencapai 88%, sedangkan pada saat database berjumlah 20 data wajah, hasil rata-rata persentase kecocokan mencapai 52%. Penyebab dari perbedaan hasil tersebut adalah karena faktor pencahayaan, jarak, bentuk wajah, serta jumlah data yang tersedia.

Kata Kunci : *Eigenface*, *OpenCv*, Pengenalan Wajah.

1. PENDAHULUAN

Sistem absensi dengan wajah merupa-kan sistem absensi yang dipadukan dengan algoritma pengenalan wajah, dimana proses absensi dapat dilakukan dengan wajah seseorang. Namun, permasalahan yang di-hadapi adalah bagaimana menerapkan algoritma pengenalan wajah ke dalam sistem absensi. Oleh karena itu dibutuhkan suatu algoritma dimana algoritma tersebut dapat mengenali wajah dan dapat di-padukan dengan sistem absensi. Algoritma pengenalan wajah yang di-gunakan adalah algoritma *eigenface* yang berasal dari OpenCv, dimana algoritma tersebut merupakan *library* dari OpenCv yang sudah dapat digunakan dan dapat mengenali wajah seseorang. Dengan mem-buat aplikasi sistem absensi dengan wajah, diharapkan nantinya perkembangan sistem absensi tidak hanya menggunakan sidik jari atau kartu, tetapi absensi juga dapat dilaku-kan dengan memindai wajah seseorang.

2. TINJAUAN PUSTAKA

2.1 Pengenalan Pola

Pola adalah suatu bentuk dimana masing-masing pola memiliki ciri-cirinya. Ciri-ciri tersebut digunakan utuk mem-bedakan suatu pola dengan pola yang lainnya. Ciri yang baik adalah ciri yang memiliki daya pembeda yang tinggi, se-hingga pengelompokkan pola berdasarkan ciri yang dimiliki dapat dilakukan dengan keakuratan yang tinggi.[6]

2.2 Pengenalan Wajah (*Face Recogni-tion*)

Pengenalan wajah adalah salah satu teknologi biometrik yang telah banyak diaplikasikan dalam sistem keamanan selain pengenalan retina mata, pengenalan sidik jari dan iris mata. Dalam aplikasinya sendiri pengenalan wajah menggunakan sebuah kamera untuk menangkap wajah seseorang kemudian dibandingkan dengan wajah yang

sebelumnya telah disimpan di dalam *database* tertentu.[12]

Pengenalan wajah melibatkan banyak variabel, misalnya citra sumber, cira hasil pengolahan citra, citra hasil ekstraksi dan data profil seseorang. Dibutuhkan juga alat pengindra berupa sensor kamera dan metode untuk menentukan apakah citra yang ditangkap oleh webcam tergolong wajah manusia atau bukan, sekaligus untuk menentukan informasi profil yang sesuai dengan citra wajah yang dimaksud.[14]

2.2.1 Algoritma *Eigenface*

Kata *eigenface* sebenarnya berasal dari bahasa Jerman *"eigenwert"* dimana *"eigen"* artinya karakteristik dan *"wert"* artinya nilai. Menurut Hanif (Al-Fatta.H, 2006) *Eigenface* adalah salah satu *algoritma pengenalan pola wajah* yang berdasarkan pada *Principle Component Analysis* (PCA). Prinsip dasar dari penge-nalan wajah adalah dengan mengutip informasi unik wajah tersebut kemudian di-*encode* dan dibandingkan dengan hasil *de-code* yang sebelumnya dilakukan.[9] Dalam metode *eigenface, decoding* dilakukan dengan menghitung *eigenvector* kemudian direpresentasikan dalam sebuah matrikss yang berukuran besar. *Eigenvector* juga dinyatakan sebagai karakteristik wajah oleh karena itu metode ini disebut dengan *eigenface*. Setiap wajah direpresentasikan dalam kombinasi linear *eigenface*. Metode *eigenface* pertama kali dikembangkan oleh Matthew Turk dan Alex Pentland dari Vision and Modeling Group, The Media Laboratory, Massachusetts Institute of Technology pada tahun 1987. Metode ini disempurnakan lagi oleh Turk dan Pentland pada tahun 1991.[9]

Algoritma pengenalan wajah di-mulai dengan membuat matrikss kolom dari wajah yang dimasukkan ke dalam *database*. Rata-rata *vector* citra (*mean*) dari matrikss kolom dihitung dengan cara membaginya dengan jumlah banyaknya citra yang disimpan di dalam *database*.[9]

Algoritma *eigenface* secara keselu-ruhan cukup sederhana. Image matrikss (Γ) direpresentasikan ke dalam sebuah him-punan matrikss (Γ_1, Γ_2, …, Γ_M). Cari nilai rata-rata (Ψ) dan gunakan untuk meng-ekstraksi *eigenvector* (*v*) dan *eigenvalue* (λ) dari himpunan matrikss. Gunakan nilai *eigenvector* untuk mendapatkan nilai *eigenface* dari image. Apabila ada sebuah image baru atau test face (Γ_{new}) yang ingin dikenali, proses yang sama juga diberlaku-kan untuk image (Γ_{new}), untuk meng-ekstraksi *eigenvector* (*v*) dan *eigenvalue* (λ), kemudian cari nilai eigen-face dari image test face (Γ_{new}). Setelah itu barulah image baru (Γ_{new}) memasuki tahapan pengenalan dengan menggunakan metode-*euclidean distance*. Tahap perhitungan se-lengkapnya dapat dilihat seperti berikut ini.

Tahapan Perhitungan Eigenface :[15]

1. Langkah pertama adalah menyiapkan data dengan membuat suatu himpunan S yang terdiri dari seluruh training image, (Γ1, Γ2, …, ΓM)

$$S = (\Gamma 1, \Gamma 2, \ldots, \Gamma M) \quad (1)$$

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

$$\Psi = \frac{1}{M}\sum_{n=1}^{M} \Gamma \quad (2)$$

3. Langkah ketiga kemudian carı selisih (Φ) antara nilai training image (Γi) dengan nilai tengah (Ψ)

$$\phi_i = \Gamma_i - \Psi \quad (3)$$

4. Langkah keempat adalah menghitung nilai matriks kovarian (C)

$$C = \frac{1}{M}\sum_{n=1}^{M} \phi_n \phi_n^T = AA^T$$

$$L = A^T A \qquad L = \phi_m^T \phi_n \quad (4)$$

5. Langkah kelima menghitung eigenvalue (λ) dan eigenvector (v) dari matriks kovarian (C)

$$C \text{ x } v_i = \lambda_i \text{ x } v_i \quad (5)$$

6. Langkah keenam, setelah eigenvector (v) diperoleh, maka eigenface (μ) dapat dicari dengan:

$$\mu_i = \sum_{k=1}^{M} v_{ik} \phi_{\text{ı}}$$

$$l = 1, \ldots, M \quad (6)$$

Tahapan Pengenalan wajah :

1. Sebuah image wajah baru atau test face (Γnew) akan dicoba untuk dikenali, pertama terapkan cara pada tahapan pertama per-hitungan eigenface untuk mendapatkan nilai eigen dari image tersebut.

$$\mu_{new} = v \text{ x } (\Gamma_{new} - \Psi) \quad (7)$$

$$\Omega = [\mu_1, \mu_2, \ldots, \mu_M]$$

2.Gunakan metode euclidean distance untuk mencari jarak *(distance)* terpendek antara nilai *eigen* dari *training image* dalam *data-base* dengan nilai *eigen* dari *image testface.*

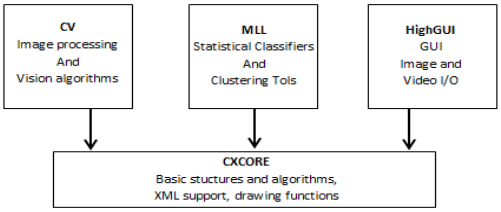
$$\varepsilon_k = \|\Omega - \Omega_k\| \quad (8)$$

2.2 OpenCv

OpenCv adalah *library* yang didalam-nya terdapat berbagai fungsi yang diper-gunakan dalam *computer vision*. OpenCv ditulis dengan bahasa C dan C++ dan dapat berjalan di Linux, Windows dan MacOS.[7] OpenCv ini adalah *library* yang sudah sangat terkenal pada pengolahan citra *computer vision*. OpenCv didesain untuk aplikasi *real-time*, memiliki fungsi-fungsi yang baik untuk *image*/video. OpenCv sendiri terdiri dari 5 *library*, yaitu :

1. CV : untuk algoritma *image processing* dan *vision*.
2. MLL : untuk machine learning *library*.
3. HighGUI : untuk GUI, *Image* dan Video I/O.
4. CvAux
5. CXCORE : untuk struktur data, support XML dan fungsi-fungsi grafis.

Struktur dan konten OpenCv[7] terlihat seperti gambar dibawah ini.



Gambar 1. Struktur dan konten OpenCv

2.3 EmguCv

EmguCv adalah jembatan untuk me-nghubungkan antara C# dan OpenCv.[7] 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 apapun seperti layaknya menggunakan OpenCv.

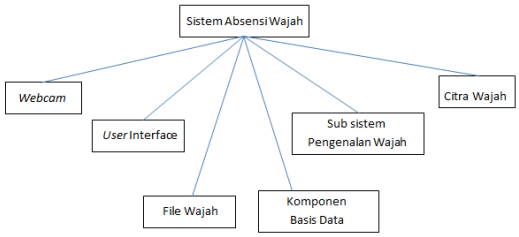
3. METODOLOGI

Pada penelitian skripsi ini menggu-nakan dua metode, yaitu metode studi literatur dan metode eksperimental. Metode studi literatur adalah tahap pencarian data mengenai pengenalan wajah, algoritma *eigenface*, OpenCv, serta penerapan pada sistem absensi. Sedangkan pada metode eksperimental adalah tahap untuk me-lakukan rancangan dan implementasi metode *eigenface* pada sistem absensi serta dilakukan pengujian dan pengambilan data dari pengujian tersebut.

4. PERANCANGAN DAN IMPLE-MENTASI

4.1 Komponen Sistem

Sistem absensi dengan wajah ini terdiri dari beberapa komponen penyusun, kom-ponen penyusun tersebut seperti terlihat pada gambar 2.



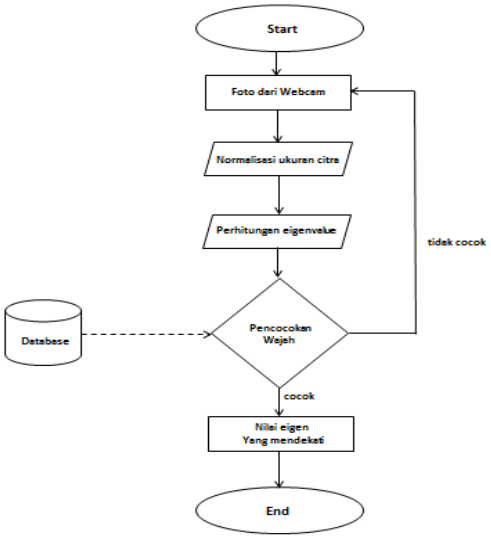
Gambar 2. Komponen penyusun sistem

Masing-masing komponen me-miliki fungsi berbeda-beda, fungsi dari setiap komponen akan dijelaskan sebagai berikut.

1. Komponen Webcam, adalah perangkat keras yang berfungsi sebagai alat untuk mengambil citra wajah *user*, selain untuk mengambil citra wajah *user*, webcam ini nantinya juga berfungsi sebagai alat *scan* saat melakukan absensi.
2. Komponen Citra Wajah, komponen ini berfungsi untuk melakukan mekanisme pengambilan cira wajah dengan bantuan webcam, pengambilan citra ini bergna untuk sebagai penyimpanan di *database* dan sebagai input pada saat absensi berlangsung.
3. Komponen *user* interface, berfungsi sebgai perantara antara *user* dengan system, perancangan diusahakan sederhana agra dapat dimudahkan oleh *user*.
4. Subsistem Pengenalan wajah, berfungsi untuk proses pengenalan wajah yaitu mencocokkan citra wajah yang di tangkap oleh webcam pada saat absensi dengan data wajah yang ada di *database*.
5. File wajah, berfungsi sebagai citra wajah *user* yang digunakan sebagai data untuk pencocokan pada saat proses absensi.

4.2 Langkah Pengenalan Wajah

Proses pengenalan wajah *user* di-perlukan beberapa tahap, pada metode *eigenface* proses pengenalan wajah terlihat seperti pada gambar 3.



Gambar 3. Proses pengenalan wajah

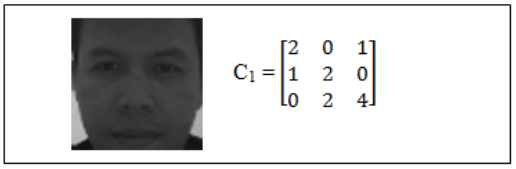
Dari gambar diatas, langkah penge-nalan wajah dapat dijelaskan sebagai berikut :
1. Mulai dari start, input dimulai dengan memasukkan foto dari webcam
2. Setelah melakukan input foto, langkah selanjutnya adalah proses normalisasi foto. Normalisasi tersebut meliputi perubahan format gambar dari RGB ke grayscale, kemudian dari grayscale tersebut diubah lagi menjadi bentuk matrikss.
3.Setelah proses normalisasi foto, selanjut-nya adalah proses perhitungan *eigenface*.
4. Tahap selanjutnya adalah pencocokan nilai *eigenface* input dengan *eigenface* di *database*, jika tidak cocok kembali ke proses foto dari webcam. Sedangkan jika cocok, akan berlanjut.
5. Selanjutnya jika cocok, akan dicari nilai eigen yang mendekati.
6. Proses selesai.

4.3 Perhitungan *Eigenface*

Algoritma pengenalan wajah dilakukan melalui beberapa tahapan, tahap pertama yaitu menyiapkan data dengan membuat suatu himpunan matriks yang ada di database, ambil nilai tengah atau mean, cari selisih antara *training image* dengan nilai tengah, hitung nilai matriks kovarian, menghitung *eigenvalue* dan *eigenvector*, tentukan nilai eigenface, dan terakhir adalah identifikasi. Untuk lebih jelasnya akan dijelaskan sebagai berikut.

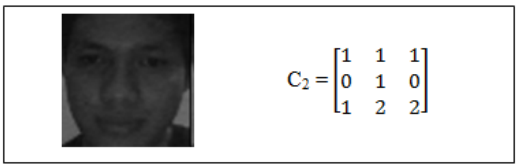
4.3.1 Penyusunan Flatvector Matriks Citra

Langkah pertama adalah menyusun suatu himpunan S matriks yang terdiri dari seluruh *training image* $(\Gamma_1, \Gamma_2, \ldots \Gamma_m)$. Misalnya, *training image* terdapat dua data wajah seperti terlihat pada gambar 4.4 dan 4.5, yang masing-masing memiliki nilai matriks.



$$C_1 = \begin{bmatrix} 2 & 0 & 1 \\ 1 & 2 & 0 \\ 0 & 2 & 4 \end{bmatrix}$$

Gambar 4. Citra wajah 1



$$C_2 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 2 & 2 \end{bmatrix}$$

Gambar 5. Citra wajah 2

4.3.2 Hitung Nilai Tengah atau Mean (Ψ)

Dari himpunan matriks yang telah diperoleh, langkah selanjutnya adalah mencari nilai tengah atau mean (Ψ). Jumlahkan nilai matrik wajah 1 dan wajah 2 kemudian bagi dengan jumlah data wajah yang ada di database, pada penelitian ini jumlah data wajah yang ada di database adalah dua data wajah.

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

Cari nilai tengah atau mean(Ψ).

$$\Psi = \frac{1}{2}\sum_{n=1}^{2} \Gamma_n = \frac{1}{2}\left[\begin{bmatrix} 2 & 0 & 1 \\ 1 & 2 & 0 \\ 0 & 2 & 4 \end{bmatrix} + \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 2 & 2 \end{bmatrix} \right]$$

$$\Psi = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 2 & 3 \end{bmatrix}$$

4.3.3 Hitung Selisih antara *Training Image* dengan Nilai Tengah

Dengan memakai nilai tengah citra di atas, langkah selanjutnya adalah mencari selisih (ϕ) antara *training image* (Γ) de-ngan nilai tengah (Ψ), dengan mengurangi *training image* (Γ) dengan nilai tengah (Ψ).

$$\phi_i = \Gamma_i - \Psi$$

$$\phi_1 = \Gamma_1 - \Psi = \begin{bmatrix} 2 & 0 & 1 \\ 1 & 2 & 0 \\ 0 & 2 & 4 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 2 & 3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\phi_2 = \Gamma_2 - \Psi = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 2 & 2 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 2 & 3 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

4.3.4 Hitung Nilai Matriks Kovarian

Nilai matriks kovarian (C) diguna-kan untuk menghitung *eigenvalue* (λ) dan *eigenvector* (v).

$$C = \frac{1}{M}\sum_{n=1}^{M} \phi_n \phi_n^T = AA^T$$
$$L = A^T A \qquad L = \phi_m^T \phi_n$$

Hitung nilai matriks kovarian (C)

$$L = \left[\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} x \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right]$$

$$L = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix} \rightarrow matriks\ kovarian$$

4.3.5 Hitung Nilai *Eigenvalue* dan *Eigen-vector*

Langkah selanjutnya adalah meng-hitung nilai *eigenvalue* (λ) dan *eigenvector* (v) dari matriks kovarian (C).

$$C \text{ x } v_i = \lambda_i \text{ x } v_i$$

Cari nilai nilai *eigenvalue* (λ) dan *eigen-vector* (v).

$$L \text{ x } v = \lambda \text{ x } v$$
$$L \text{ x } v = \lambda I \text{ x } v$$
$$(L - \lambda I) = 0\ atau\ (\lambda I - L) = 0$$

$$0 = \lambda \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}$$

$$0 = det \begin{bmatrix} \lambda - 2 & -1 & 0 \\ -1 & \lambda - 2 & 0 \\ 0 & 0 & \lambda - 3 \end{bmatrix}$$

Maka *eigenvalue* yang dihasilkan adalah $\lambda_1 = 3,\ \lambda_2 = 1,\ \lambda_3 = 3$

$$v = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

Eigenvector (v) dihasilkan dengan men-substitusikan nilai *eigenvalue* (λ) kedalam persamaan $(\lambda I - L)\, v = 0$. *Eigenvector* dari masing-masing *eigenvalue* didapat berdasarkan masing-masing kolom *eigen-value* dan kemudian dihimpun kembali menjadi satu matriks.

a. Untuk λ_1 = 3, maka :

$$\begin{bmatrix} \lambda - 2 & -1 & 0 \\ -1 & \lambda - 2 & 0 \\ 0 & 0 & \lambda - 3 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Dihasilkan *eigenvector* v_1 adalah $\begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}$,

b. Untuk λ_2 = 1, maka :

$$\begin{bmatrix} \lambda - 2 & -1 & 0 \\ -1 & \lambda - 2 & 0 \\ 0 & 0 & \lambda - 3 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Dihasilkan *eigenvector* v_2 adalah $\begin{bmatrix} -1 \\ -1 \\ 0 \end{bmatrix}$,

c. Untuk λ_3 = 3, maka :

$$\begin{bmatrix} \lambda - 2 & -1 & 0 \\ -1 & \lambda - 2 & 0 \\ 0 & 0 & \lambda - 3 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Dihasilkan *eigenvector* v_3 adalah $\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$

Setelah didapat *eigenvector* $v_1, v_2, dan\ v_3$, maka *eigenvector* yang dihasilkan dari matriks *L adalah* $\begin{bmatrix} 1 & -1 & 0 \\ -1 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

4.3.6 Nilai Eigenface

Langkah selanjutnya, setelah *eigen-vector* (v) diperoleh, maka nilai *eigenface* (μ) dapat dicari dengan:

$$\mu_i = \sum_{k=1}^{M} v_{ik} \phi_k$$
$$l = 1, \ldots, M$$

Cari nilai eigenface (μ) :

$$\mu_i = \sum_{k=1}^{M} v_{ik} \phi_k$$

$$\mu_1 = v \text{ x } \phi_1 = \begin{bmatrix} 1 & -1 & 0 \\ -1 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \text{ x } \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mu_1 = \begin{bmatrix} 0 & -1 & 0 \\ -2 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\mu_2 = v \text{ x } \phi_2 = \begin{bmatrix} 1 & -1 & 0 \\ -1 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \text{ x } \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

$$\mu_2 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

4.3.7 Proses Identifikasi

Proses identifikasi adalah proses dimana wajah baru masuk dan akan dicocokkan dengan wajah yang ada di dalam database untuk mengetahui wajah mana yang cocok antara wajah yang baru masuk dengan wajah yang ada di dalam database. Untuk mengenali wajah baru yang masuk (test face), langkah yang dilakukan sama dengan data wajah yang ada di database, untuk mendapatkan nilai eigenface dari wajah baru.

$$\mu_{new} = v \text{ x } (\Gamma_{new} - \Psi)$$
$$\Omega = [\mu_1, \mu_2, \dots, \mu_M]$$

Pertama cari selisih (Φ) antara test face dengan nilai tengah (Ψ). nilai matriks test face dari koordinat tersebut terlihat pada gambar 6.

$$\Phi_{new} = \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 0 & 2 & 4 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 2 & 3 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 0 & 2 & 4 \end{bmatrix}$$

Gambar 6. Citra wajah baru (*testface*)

Selanjutnya, setelah selisih (Φ) antara *tesface* dengan nilai tengah (Ψ) di-dapat, maka nilai eigenface dapat dicari.

$$\mu_{new} = v \text{ x } \phi_{new}$$

$$\mu_{new} = \begin{bmatrix} 1 & -1 & 0 \\ -1 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \text{ x } \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mu_{new} = \begin{bmatrix} 0 & 0 & -1 \\ -2 & -2 & -1 \\ 0 & 0 & 0 \end{bmatrix}$$

Gunakan *euclidean distance* untuk mencari selisih terkecil antara *eigenface training image* (Γ_i) dalam *database* dengan *eigenface test face* (Γ_{new}), setelah itu jumlahkan matriks dari masing-masing *euclidean distance*.

$$\varepsilon_k = \|\Omega - \Omega_k\|$$
$$\varepsilon_k = \|\Omega - \Omega_{new}\|$$

$$\varepsilon_1 = \|\Omega - \Omega_{new}\|$$

$$= \left\| \begin{bmatrix} 0 & 0 & -1 \\ -2 & -2 & -1 \\ 0 & 0 & 0 \end{bmatrix} - \begin{bmatrix} 0 & -1 & 0 \\ -2 & -2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \right\|$$

$$= \sqrt{\begin{matrix}(0)^2 + (1)^2 + (1)^2 + (0)^2 + (1)^2 + \\ (1)^2 + (0)^2 + (0)^2 + (0)^2\end{matrix}}$$

$$= \sqrt{4} = 2$$

$$\varepsilon_2 = \|\Omega - \Omega_{new}\|$$

$$= \left\| \begin{bmatrix} 0 & 0 & -1 \\ -2 & -2 & -1 \\ 0 & 0 & 0 \end{bmatrix} - \begin{bmatrix} 0 & 1 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \right\|$$

$$= \sqrt{\begin{matrix}(0)^2 + (1)^2 + (1)^2 + (4)^2 + (1)^2 + \\ (1)^2 + (0)^2 + (0)^2 + (0)^2\end{matrix}}$$

$$= \sqrt{8} = 2{,}828$$

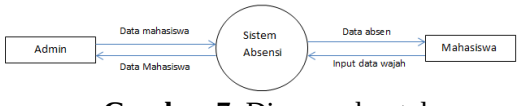
Dalam hasil perhitungan, diperoleh jarak eigenface citra wajah satu memiliki jarak yang terkecil. Karena jarak eigenface wajah satu dengan testface yang paling kecil, maka hasil identifikasi menyimpul-kan bahwa testface lebih mirip dengan wajah satu dibandingkan dengan wajah dua.

4.4 Perancangan Sistem

Perancangan sistem yang digunakan pada penelitian ini adalah diagram konteks, data flow diagram (DFD), rancangan antar muka serta rancangan *database*.

4.4.1. Diagram Konteks

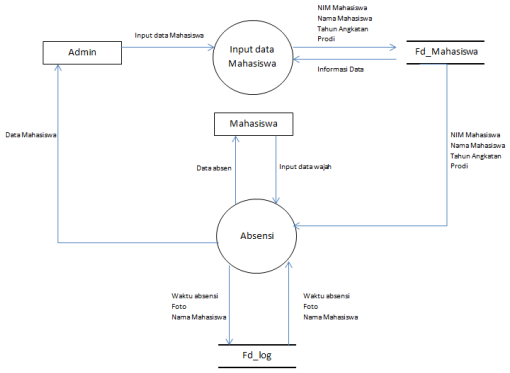
Rancangan diagram konteks sistem absensi wajah ini digambarkan dengan diagram konteks yang menggambarkan *Admin* mengaktifkan *webcam* untuk me-lakukan pengambilan foto wajah dalam proses pemasukan data peserta absensi ke-dalam *database* serta dalam proses absensi. Diagram konteks terlihat seperti pada gambar 7.



Gambar 7. Diagram konteks

4.4.2. Data Flow Diagram

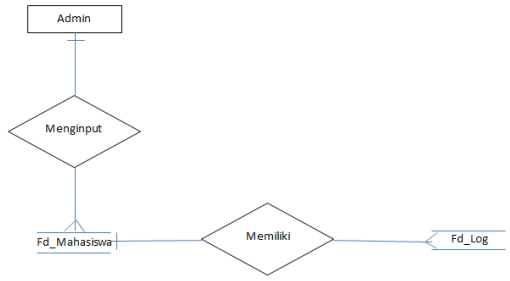
Data Flow Diagram level 1 (DFD level 1) berfungsi untuk mengetahui ke-butuhan sistem secara lebih detil lagi. DFD level 1 terlihat seperti pada gambar 8.



Gambar 8. Data flow diagram level 1

4.4.3 Entity Relationship Diagram

Entity Relationship Diagram (ERD) merupakan salah satu model yang diguna-kan untuk merancang *database* dengan tujuan menggambarkan data yang berelasi pada *database*. Pada sistem absensi dengan wajah ini, rancangan *database* terlihat seperti pada gambar 9.



Gambar 9. Entity relationship diagram

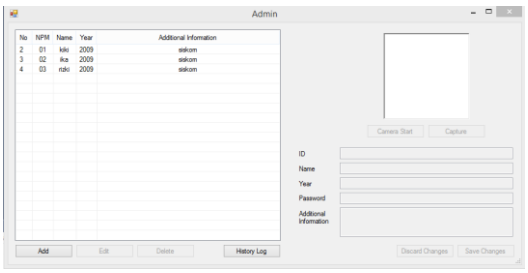
5. PENGUJIAN DAN ANALISA HASIL

5.1 Pengujian Aplikasi

Setelah perangkat lunak absensi dengan wajah ini dibangun, tahap selanjut-nya adalah tahap uji coba tampilan. Tahap uji coba ini meliputi pengujian dari awal pemasukan data *user* sampai proses iden-tifikasi pada saat absensi. Selanjtnya akan dijelaskan sebagai berikut.

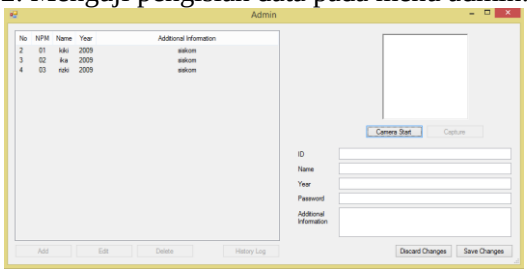
1. Menguji Jendela masuk aplikasi

Gambar 10. Tampilan menu awal aplikasi

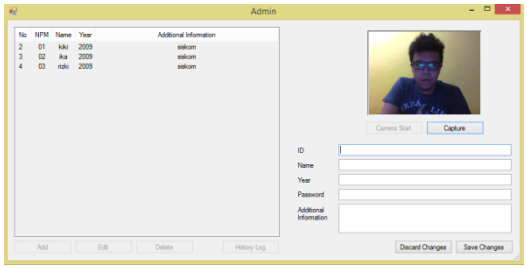


Gambar 11. Tampilan menu admin

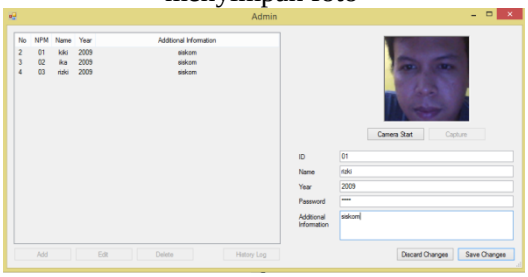
2. Menguji pengisian data pada menu admin.



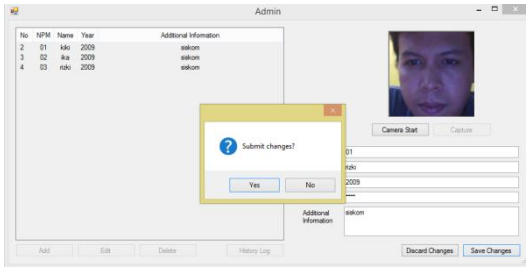
Gambar 12. Tombol kamera aktif



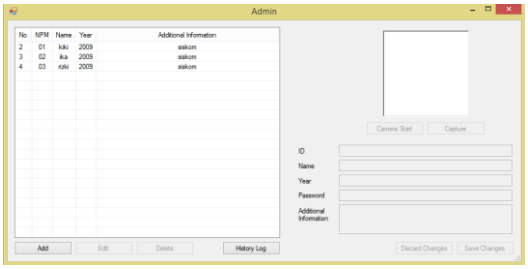
Gambar 13. Tombol *capture* aktif untuk menyimpan foto



Gambar 14. Foto *user* disimpan



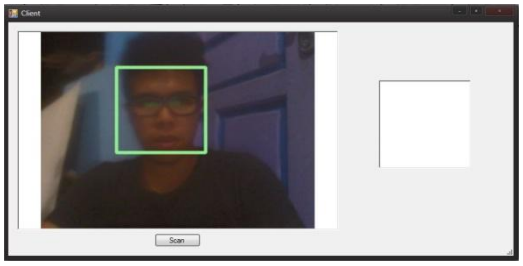
Gambar 15. Pemberitahuan saat menyimpan data



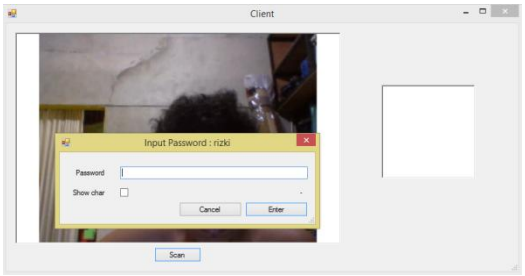
Gambar 16. Data berhasil disimpan

3. Menguji Absensi pada menu *user*

Pada menu ini, *user* akan melaku-kan absensi dengan menggunakan wajah. Langkah yang dilakukan adalah *user* duduk di depan aplikasi yang berjarak kurang lebih 20cm dari *user*, kemudian selanjutnya tunggu sampai kotak hijau mendeteksi wajah, seperti gambar di bawah ini.



Gambar 17. Tampilan pada saat melakukan absensi



Gambar 18. *Form* untuk mengisi *password* untuk absensi

Gambar 19. Absensi berhasil dilakukan

5.2 Tingkat Persentase Absensi Penge-nalan Wajah

Untuk tingkat persentase hasil ab-sensi pada pengenalan wajah ini dilakukan dengan uji coba absensi sebanyak 10 kali yang dilakukan oleh 5 orang dengan *data-base* berjumlah 20 data dan 10 data. Untuk posisi wajah, yang sangat baik adalah posisi wajah menghadap ke depan. Hasil dari uji coba tersebut dapat dilihat pada tabel di bawah ini.

Tabel 1. Hasil *scan* citra wajah 1 dengan *database* sebanyak 20 data wajah

Citra Wajah yang diuji	Posisi	Citra Wajah Pencarian	Persentase Keberhasilan
 Citra Wajah 1	Tampak Depan	Terdeteksi	6/10 x 100% = 60%
	Tampak Depan	Tidak Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Tidak Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Tidak Terdeteksi	
	Tampak Depan	Tidak Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Terdeteksi	

Tabel 1 berisi data hasil *scan* untuk absensi, proses *scan* berlangsung sebanyak 10 kali dengan posisi wajah menghadap kedepan dan jumlah data wajah yang ada di database sebanyak 20 data wajah, persen-tase keberhasilan sebesar 60%.

Tabel 2. Hasil *scan* citra wajah 2 dengan *database* sebanyak 20 data wajah

Citra Wajah yang diuji	Posisi	Citra Wajah Pencarian	Persentase Keberhasilan
 Citra Wajah 2	Tampak Depan	Tidak Terdeteksi	5/10 x 100% = 50%
	Tampak Depan	Tidak Terdeteksi	
	Tampak Depan	Tidak Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Tidak Terdeteksi	
	Tampak Depan	Tidak Terdeteksi	

Tabel 2 berisi data hasil *scan* untuk absensi, proses *scan* berlangsung sebanyak 10 kali dengan posisi wajah menghadap kedepan dan jumlah data wajah yang ada di database sebanyak 20 data wajah, persen-tase keberhasilan sebesar 50%.

Tabel 3. Hasil *scan* citra wajah 3 dengan *database* sebanyak 20 data wajah

Citra Wajah yang diuji	Posisi	Citra Wajah Pencarian	Persentase Keberhasilan
 Citra Wajah 3	Tampak Depan	Tidak Terdeteksi	4/10 x 100% = 40%
	Tampak Depan	Tidak Terdeteksi	
	Tampak Depan	Tidak Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Tidak Terdeteksi	
	Tampak Depan	Tidak Terdeteksi	
	Tampak Depan	Tidak Terdeteksi	

Tabel 3 berisi data hasil *scan* untuk absensi, proses *scan* berlangsung sebanyak 10 kali dengan posisi wajah menghadap kedepan dan jumlah data wajah yang ada di database sebanyak 20 data wajah, persen-tase keberhasilan sebesar 40%.

Tabel 4. Hasil *scan* citra wajah 4 dengan *database* sebanyak 20 data wajah

Citra Wajah yang diuji	Posisi	Citra Wajah Pencarian	Persentase Keberhasilan
 Citra Wajah 4	Tampak Depan	Terdeteksi	6/10 x 100% = 60%
	Tampak Depan	Terdeteksi	
	Tampak Depan	Tidak Terdeteksi	
	Tampak Depan	Tidak Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Tidak Terdeteksi	
	Tampak Depan	Tidak Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Terdeteksi	

Tabel 4 berisi data hasil *scan* untuk absensi, proses *scan* berlangsung sebanyak 10 kali dengan posisi wajah menghadap kedepan dan jumlah data wajah yang ada di database sebanyak 20 data wajah, persen-tase keberhasilan sebesar 60%.

Tabel 5. Hasil *scan* citra wajah 5 dengan *database* sebanyak 20 data wajah

Citra Wajah yang diuji	Posisi	Citra Wajah Pencarian	Persentase Keberhasilan
 Citra Wajah 5	Tampak Depan	Terdeteksi	5/10 x 100% = 50%
	Tampak Depan	Terdeteksi	
	Tampak Depan	Tidak Terdeteksi	
	Tampak Depan	Tidak Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Tidak Terdeteksi	
	Tampak Depan	Tidak Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Tidak Terdeteksi	

Tabel 5 berisi data hasil *scan* untuk absensi, proses *scan* berlangsung sebanyak 10 kali dengan posisi wajah menghadap kedepan dan jumlah data wajah yang ada di database sebanyak 20 data wajah, persen-tase keberhasilan sebesar 50%.

Berikut adalah hasil *scan* dengan *database* sebanyak 10 data :

Tabel 6. Hasil *scan* citra wajah 1 dengan *database* sebanyak 10 data wajah

Citra Wajah yang diuji	Posisi	Citra Wajah Pencarian	Persentase Keberhasilan
 Citra Wajah 1	Tampak Depan	Terdeteksi	10/10 x 100% = 100%
	Tampak Depan	Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Terdeteksi	

Tabel 6 berisi data hasil *scan* untuk absensi, proses *scan* berlangsung sebanyak 10 kali dengan posisi wajah menghadap kedepan dan jumlah data wajah yang ada di database sebanyak 10 data wajah, persen-tase keberhasilan sebesar 100%.

Tabel 7. Hasil *scan* citra wajah 2 dengan *database* sebanyak 10 data wajah

Citra Wajah yang diuji	Posisi	Citra Wajah Pencarian	Persentase Keberhasilan
 Citra Wajah 2	Tampak Depan	Terdeteksi	9/10 x 100% = 90%
	Tampak Depan	Terdeteksi	
	Tampak Depan	Tidak Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Terdeteksi	

Tabel 7 berisi data hasil *scan* untuk absensi, proses *scan* berlangsung sebanyak 10 kali dengan posisi wajah menghadap kedepan dan jumlah data wajah yang ada di database sebanyak 10 data wajah, persen-tase keberhasilan sebesar 90%.

Tabel 8. Hasil *scan* citra wajah 3 dengan *database* sebanyak 10 data

Citra Wajah yang diuji	Posisi	Citra Wajah Pencarian	Persentase Keberhasilan
 Citra Wajah 3	Tampak Depan	Tidak Terdeteksi	8/10 x 100% = 80%
	Tampak Depan	Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Tidak Terdeteksi	

Tabel 8 berisi data hasil *scan* untuk absensi, proses *scan* berlangsung sebanyak 10 kali dengan posisi wajah menghadap kedepan dan jumlah data wajah yang ada di database sebanyak 10 data wajah, persen-tase keberhasilan sebesar 80%.

Tabel 9. Hasil *scan* citra wajah 4 dengan *database* sebanyak 10 data wajah

Citra Wajah yang diuji	Posisi	Citra Wajah Pencarian	Persentase Keberhasilan
 Citra Wajah 4	Tampak Depan	Terdeteksi	8/10 x 100% = 80%
	Tampak Depan	Terdeteksi	
	Tampak Depan	Tidak Terdeteksi	
	Tampak Depan	Tidak Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Terdeteksi	

Tabel 9 berisi data hasil *scan* untuk absensi, proses *scan* berlangsung sebanyak 10 kali dengan posisi wajah menghadap kedepan dan jumlah data wajah yang ada di database sebanyak 10 data wajah, persen-tase keberhasilan sebesar 80%.

Tabel 10. Hasil *scan* citra wajah 5 dengan *database* sebanyak 10 data wajah

Citra Wajah yang diuji	Posisi	Citra Wajah Pencarian	Persentase Keberhasilan
 Citra Wajah 5	Tampak Depan	Terdeteksi	9/10 x 100% = 90%
	Tampak Depan	Terdeteksi	
	Tampak Depan	Tidak Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Terdeteksi	
	Tampak Depan	Terdeteksi	

Tabel 10 berisi data hasil *scan* untuk absensi, proses *scan* berlangsung sebanyak 10 kali dengan posisi wajah menghadap kedepan dan jumlah data wajah yang ada di database sebanyak 10 data wajah, persen-tase keberhasilan sebesar 90%.

Berdasarkan hasil uji coba dan hasil pada tabel tersebut menunjukan hasil persentase yang berbeda pada setiap citra wajah dan pada saat jumlah *database* yang berbeda, hasil persentase juga menunjukkan perbedaan pada masing-masing citra wajah. Pada saat *database* yang terdiri dari 20 orang, hasil test menunjukkan hasil yang berbeda antara citra wajah 1 hingga citra wajah 5. Citra wajah 1 menghasilkan per-sentase sebesar 60%, citra wajah 2 meng-hasilkan persentase sebesar 50%, citra wajah 3 menghasilkan persentase sebesar 40%, citra wajah 4 menghasilkan persen-tase sebesar 60%, da citra wajah 5 meng-hasilkan persentase sebesar 50%.

Sedangkan pada saat *database* ber-jumlah 10 orang, hasil yang didapat juga berbeda-beda antara citra wajah 1 hingga citra wajah 5. Pada citra wajah 1 meng-hasilkan persentase sebesar 100%, citra wajah 2 menghasilkan persentase sebesar 90%, citra wajah 3 menghasilkan persen-tase sebesar 80%, citra wajah 4 mengasil-kan persentase sebesar 80% dan citra wajah 5 mengasilkan persentase sebesar 90%.

6. KESIMPULAN

Berdasarkan hasil analisa yang dilakukan dan aplikasi yang dikembangkan, maka dapat ditarik kesimpulan sebagai berikut :

1. Telah berhasil diterapkan algoritma *eigenface* untuk mendeteksi wajah dengan bantuan opencv, sehingga dapat diimplementasikan ke dalam sistem.
2. Pemrosesan pengenalan wajah pada sistem absensi dengan menggunakan metode *eigenface* pada opencv dapat dilakukan dengan cara input data pengguna dan data wajah beserta password dari masing-masing peng-guna. Selanjutnya *scan* untuk menge-tahui apakah wajah sudah sesuai dengan *database*, lalu masukkan pass-word, maka proses absensi berhasil.
3. Pemrosesan pengenalan wajah dengan menggunakan metode *eigenface* pada opencv ini dikatakan sensitif, karena bergantung pada intensitas cahaya, jarak, dan sudut pandang wajah.
4. Pemrosesan pengenalan pada sistem absensi dapat berjalan baik jika data yang ada di*database* berkisar 10 orang dengan rata-rata persentase keber-hasilan sebesar

88%, serta pada pen-cahayaan yang sama, sehingga tingkat pencarian wajah yang mendekati di *database* dapat lebih baik dibanding-kan dengan saat *database* berjumlah 20 orang.

7. SARAN

Penelitian yang dilakukan ini sangat sederhana dan masih banyak kekurangan-nya, oleh sebab itu penulis mengharapkan agar kedepannya dapat dibangun sebuah sistem yang dapat mengenali wajah dengan tingkat kemiripan yang lebih akurat, di-antaranya adalah sebagai berikut:

1. Diharapkan membangun suatu sistem yang dapat mengenali wajah tidak hanya dari tampak depan, tetapi juga dapat mengenali dari tampak samping.
2. Sebaiknya perlu dilakukan penyem-purnaan dalam intesitas cahaya, se-hingga di tempat yang agak redup pencahayaannya, proses pengenalan wajah dapat berjalan dengan baik.
3. Agar pengembangan sistem selanjut-nya, pengenalan wajah ini dapat mengenali berbagai macam ekspresi wajah pengguna, walaupun data orang yang ada di *database* sangat banyak.

DAFTAR PUSTAKA

[1] Al Fatta, Hanif. 2006. Sistem Presensi Karyawan Berbasis Pengenalan Wajah Dengan Algoritma *Eigenface*. Yogyakarta: STMIK AMIKOM

[2] Anonim. (2014, August 28). Emgu CV Main Page. Retrieved August 2014, from

[3] Emgu CV: OpenCV in .NET (C#, VB, C++ and more): http://www.emgu.com/wiki/index.php/Main_Page.

[4] Bradski, G and Kaehler, A. 2008. Learning OpenCV. United States of America: O'Reilly Media, Inc.

[5] Brigida. 2012. Pengenalan Pola. http://informatika.web.id/pengenalan-pola.htm.

[6] Brigida. 2012. Komponen Sistem Pengenalan Pola. http://informatika.web.id/komponen-sistem-pengenalan-pola.htm.

[7] Denny, M. 2012. *Pengenalan Computer Vision dengan EmguCV di C#.Net*. http://bisakomputer.com/pengenalan-computer-vision-dengan-emgucv-di-c-net/

[8] Hermawati, F. A. 2013. Pengolahan Citra Digital. Yogyakarta: Penerbit Andi.

[9] Indra. 2012. Sistem Pengenalan Wajah Dengan Metode *Eigenface* Untuk Absensi Pada PT. florindo lestari. Jakarta: Universitas Budi Luhur.

[10] Iqbal, Muhammad. 2009. Dasar Pengolahan Citra Menggunakan MATLAB. Bogor: Institut Pertanian Bogor.

[11] Lestya Dila Rahma. 2009. Pengolahan Wajah Berdasarkan Pengolahan Citra Digital Dengan Metode Gabor Wavelet. Universitas Sumatera Utara. Medan

[12] Rahim, M.A. 2013.Perancangan Aplikasi Pengenalan Wajah Menggunakan Metode Viola Jonnes.Medan: STMIK Budi Dharma.

[13] Sianipar, R. H. 2014. Pemrograman Visual Basic.NET. Bandung: Informatika Bandung.

[14] Suprianto, D. 2013. Sistem Pengenalan Wajah Secara *Real-Time*, dengan *Adaboost*, *Eigenface* PCA & MySql. Malang. Universitas Brawijaya Malang.

[15] Turk, M and Pentland, A. 1991. *Face Recognition Using Eigenfaces*. http://www.cs.ucsb.edu/~mturk/Papers/mturk-CVPR91.pdf