JOINT

Journal of Informatics and Technology, Vol 2, No 1, Tahun 2013, p 102-110
http://ejournal-s1.undip.ac.id/index.php/joint

PENGENALAN WAJAH MENGGUNAKAN ALGORITMA *EIGENFACE* DAN *EUCLIDEAN DISTANCE*

Widodo Muda Saputra, Helmie Arif Wibawa, S.Si, M.Cs, dan Nurdin Bahtiar, S.Si, M.T
Fakultas Sains dan Matematika, Jurusan Ilmu Komputer / Informatika
Universitas Diponegoro Semarang
Email: *widodomudasaputra@gmail.com*

ABSTRAK

Pengenalan wajah manusia merupakan salah satu bidang penelitian yang penting, dan dewasa ini banyak aplikasi yang dapat menerapkannya. Melalui pengembangan suatu teknik seperti *Eigenface*, komputer sekarang dapat melebihi kemampuan otak manusia dalam berbagai tugas pengenalan wajah, terutama tugas-tugas yang membutuhkan pencarian pada *database* wajah yang besar. *Eigenface* menggunakan metode *Principal Component Analysis* (PCA) dan dapat digunakan untuk mereduksi dimensi gambar wajah sehingga menghasilkan variabel yang lebih sedikit yang lebih mudah untuk diobservasi dan ditangani. Hasil yang diperoleh kemudian akan dimasukkan ke suatu *pattern classifier* untuk menentukan identitas pemilik wajah. Hasil pengujian sistem menunjukkan bahwa penggunaan *Eigenface* dengan *Euclidean Distance* sebagai *classifier* dapat memberikan tingkat akurasi yang cukup tinggi. Untuk gambar wajah yang diikutsertakankan dalam pelatihan, dapat diperoleh 91% identifikasi benar.

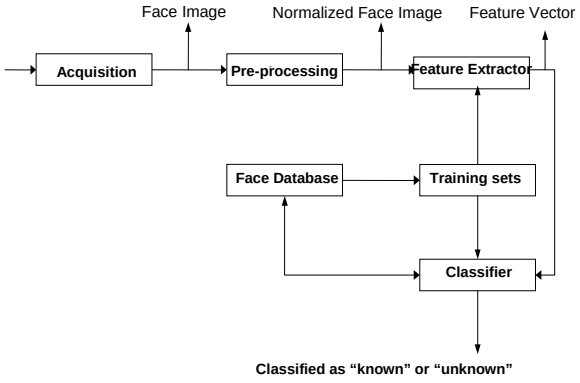
Kata Kunci: Pengenalan Wajah, *Principal Components Analysis, Eigenface*

1. Pendahuluan

Pengenalan wajah manusia adalah salah satu bidang penelitian penting dengan banyak aplikasi yang dapat menerapkannya. Penelitian terhadap pengenalan wajah manusia sudah banyak dilakukan dengan kelebihan dan kekurangan tertentu. Hal ini disebabkan karena wajah manusia mempresentasikan sesuatu yang kompleks, sehingga untuk mengembangkan model komputasi yang ideal untuk pengenalan wajah manusia adalah sesuatu hal yang sulit. Pengenalan wajah manusia mendapatkan banyak perhatian beberapa tahun terakhir ini, hal ini karena banyak aplikasi yang menerapkannya, antara lain dalam pengamanan gedung, alat identifikasi, ATM, *Tele-Conference*, alat bantu dalam pelacakan pelaku kriminal dan lain-lain. Ada banyak teknik pengenalan wajah yang dapat dipakai dan salah satunya adalah *Eigenface*

2. Pengenalan Wajah

Pengenalan wajah dapat diartikan sebagai pengelompokan sebuah wajah sebagai "wajah dikenali" atau "wajah tidak dikenali", setelah membandingkannya ke dalam wajah individu dikenali yang tersimpan sebelumnya (*database* wajah). [1]



Gambar 1. Komponen Sistem Pengenalan Wajah [1]

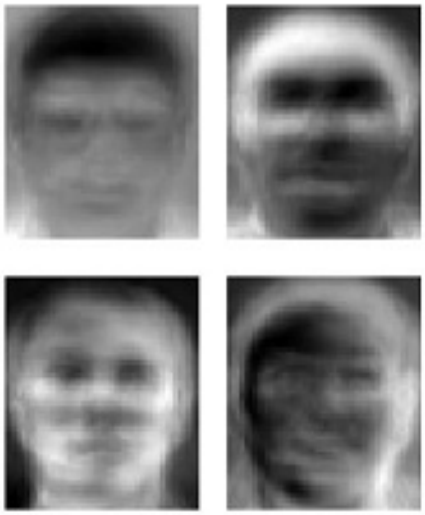
Terdapat 6 komponen utama dalam pengenalan wajah [1]:

1. *Acquisition Module*
   Citra wajah dihadirkan ke sistem di modul ini melalui media input atau dari *file* di penyimpanan.
2. *Pre-processing Module*
   hasil dari modul akuisisi akan dinormalisasi dan apabila diperlukan akan dilakukan pengolahan lanjutan agar didapatkan hasil pengenalan yang lebih baik.
3. *Feature Extraction Module*
   Modul ini bertanggung jawab terhadap penyusunan sebuah *feature vector* yang terbaik untuk merepresentasikan citra wajah.
4. *Classification Module*
   fitur terekstraksi dari sebuah wajah dapat dibandingkan dengan yang lainnya yang tersimpan di perpustakaan wajah (*database* wajah) dengan bantuan *pattern classifier*.
5. *Training set*
   Modul ekstraksi fitur dan modul klasifikasi menyesuaikan parameter mereka agar dapat mencapai performa pengenalan optimum dengan penggunaan training set.
6. *Face library or face database*
   Modul ini menyimpan fitur vektor dari citra wajah di himpunan data pelatihan (*training set*)

3. Algoritma *Eigenface*

Eigenface merupakan suatu algoritma yang menggunakan *Principal Component Analysis* (PCA) untuk mengurangi dimensionalitas serta untuk mencari vektor terbaik guna mendistribusikan citra wajah ke dalam ruang wajah yang ada. Tujuan utama dari PCA adalah untuk mencari vektor yang paling cocok yang dapat menggambarkan distribusi citra wajah di dalam ruang citra ke dalam ruang wajah. Untuk membentuk *principal component* sejumlah m *eigenvector* digunakan sesuai dengan distribusi *eigenvalue*. *Eigenvector* dan *eigenvalue* didapatkan dari matriks kovarian yang di-*generate* dari citra wajah yang dilatihkan. *Eigenvector* diurutkan berdasarkan *eigenvalue*-nya (tinggi ke rendah) dan dipilih sebanyak m *eigenvector* pertama untuk membentuk *principal component* [3].

Eigenface pada dasarnya adalah himpunan *eigenvector*. *Eigenvector* diturunkan dari sebuah matriks kovarian dari ruang vektor yang mungkin dari wajah-wajah manusia. Untuk mendapatkan *Eigenface*, sejumlah himpunan gambar citra wajah diambil di bawah kondisi pencahayaan yang sama kemudian dilakukan normalisasi ke ukuran resolusi pixel yang sama. *Eigenface* kemudian diekstrak dari data citra dengan menggunakan PCA. PCA dikenal sebagai teknik untuk mereduksi dimensi dengan mentransformasi sejumlah variabel yang berkorelasi ke dalam sejumlah kecil variabel yang tidak berkorelasi. *Eigenface* akan tampak seperti area gelap dan terang yang disusun ke dalam pola tertentu.

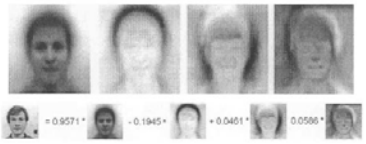


Gambar 2. Contoh *Eigenface* [4]

Secara umum langkah pada algoritma *Eigenface* adalah sebagai berikut[5] :

1. Membuat himpunan citra wajah untuk pelatihan. Citra wajah harus mempunyai resolusi serta *align* yang sama. Citra ini pada dasarnya adalah sebuah matriks dimana dimisalkan Γ_i adalah matriks yang merepresentasikan citra wajah.
2. Menghitung matriks rata-rata ψ dengan persamaan berikut
$$\psi = \frac{1}{M}\sum_{n=1}^{M} \Gamma_i \quad \text{……(1)}$$
M adalah jumlah citra wajah untuk pelatihan. Kemudian kurangkan nilai rata-rata dari tiap wajah Γ.
$$\Phi_i = \Gamma_i - \psi \quad \text{…….....(2)}$$
3. Menghitung matriks kovarian
$$C = \frac{1}{M}\sum_{n=1}^{M} \Phi_n \Phi_n^T = AA^T \quad \text{.....(3)}$$
dimana A=[$\Phi_1\ \Phi_2 \dots \Phi_m$].
4. Menghitung *eigenvector* dan *eigenvalue* dari matriks kovarian.
C adalah matrik kovarian. Skalar λ disebut sebagai suatu nilai eigen atau nilai karakteristik (*characteristic value*) jika terdapat suatu vektor tak nol *v*, sehingga
$$Cv = \lambda v \quad \text{……(4)}$$
Vektor *v* disebut *eigenvector* (vektor eigen) dari λ .
5. Pilih *principal component*, ambil *k* *eigenvector* dengan nilai *eigenvalue* terbesar dari M citra. Semakin tinggi nilai *eigenvalue* maka semakin banyak karakteristik fitur yang berkorespondensi dengan *eigenvalue* dijelaskan.
6. Mengubah citra ke komponen *Eigenface*. Tiap wajah (dikurangi citra rata-rata) dapat direpresentasikan sebagai kombinasi linier dari M *eigenvector* terbaik. Jika jumlah data di ruang citra lebih kecil dari dimensinya (M < N^2) maka hanya akan ada M-1 *eigenvector* yang berarti. Sisanya akan berasosiasi dengan *eigenvalue* nol. Pada kasus ini *eigenvector* berdimensi N^2 dapat diselesaikan dengan menyelesaikan *eigenvector* dari matriks M x M.
Dari persamaan 4 didapatkan
$$AA^T v = \lambda v \quad \text{……(5)}$$
Dengan mengalikan kedua ruas dengan A didapatkan
$$AA^T Av = \lambda Av \quad \text{……...(6)}$$
Hal ini berarti bahwa A*v* adalah eigenvector dari C = AA^T .

Dari analisis sebelumya dibentuk M x M matriks L = $A^T A$ dengan
$$L_{mn} = \Phi_m^T \Phi_n \quad \text{……..(7)}$$
Eigenvector dari matriks L merupakan kombinasi linier dari M citra wajah untuk membentuk *Eigenface* u_i.
$$u_i = \sum_{k=1}^{M} v_{ik} \Phi_k \ , l = 1\text{…M……(8)}$$
Citra wajah akan diproyeksikan ke ruang wajah dengan persamaan
$$W_k = u_k^T(\Gamma - \psi) \ , k = 1\text{… M'….(9)}$$
M' adalah jumlah *eigenvector* yang digunakan. Hasil dari proyeksi digunakan untuk membentuk vektor fitur Ω.



Gambar 3. Kombinasi Linier *Eigenface*
Setiap citra wajah pelatihan dapat direpresentasikan dalam basis ini oleh sebuah vektor

$$\Omega = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \dots \\ w_M \end{bmatrix} \quad \text{.....(10)}$$

4. Metode Jarak *Euclidean*

Secara umum, yang dimaksud dengan jarak antara dua titik adalah garis terpendek di antara semua garis yang menghubungkan kedua titik tersebut. Dalam ruang *Euclidean* berdimensi m, R^m , jarak antara titik x dan y dapat dirumuskan sebagai berikut [2]:

$$D_{x,y} = \sqrt{\sum_{i=0}^{m} |x_i - y_i|^2} \quad \text{.....(11)}$$

Pemilihan jarak *Euclidean* dikarenakan metode ini cocok digunakan untuk menghitung jarak antara titik-titik *pixel* dari dua citra yang berbeda. Jarak *Euclidean* dihitung di antara vektor bobot ini dengan citra yang dibentuk oleh citra pelatihan. Jarak ini merupakan sebuah ukuran kesamaan di antara citra yang dites dengan citra wajah yang ada di *database*.

5. Pembahasan

Proses pengenalan wajah dibagi menjadi tiga bagian, yaitu praproses, ekstraksi fitur dan klasifikasi. Proses praproses merupakan proses membentuk citra wajah ternormalisasi baik dari segi jenis, ukuran, dan tingkat kecerahan. Citra hasil dari praproses akan berupa citra wajah *grey* berukuran 128 x 128 piksel dengan tingkat kecerahan yang relatif sama. Proses ekstraksi fitur akan mencari vektor fitur dari citra wajah yang akan dikenali. Proses ini pada dasarnya adalah mencari bobot dari masing-masing *Eigenface* yang digunakan untuk merepresentasikan citra wajah.

Contoh

Terdapat lima citra wajah standar yang mempunyai ukuran 5 x 5 *pixel*.

Langkah 1. Proses Linieriasasi

Proses linierisasi dilakukan dengan mengubah citra yang berukuran 5 x 5 *pixel* menjadi 5 buah vektor berdimensi 25 x 1 yang diberi nama L1, L2, L3, L4, dan L5. Matriks citra hasil linierisasi dapat dinyatakan sebagai :

L = [L1 L2 ... Li] .....(12)

Keterangan :
L : Matriks citra hasil linierisasi
i : Jumlah citra yang digunakan

Hasil proses linierisasi dapat dilihat pada tabel 1.

Tabel 1. Nilai Linierisasi dan Nilai Rataan

L1	L2	L3	L4	L5	U (Nilai Rataan)
132	113	120	109	114	117.6
143	122	104	131	66	113.2
114	152	59	53	49	85.4
118	145	89	43	56	90.2
163	153	109	143	119	137.4
143	126	113	126	107	123
6	10	121	62	38	47.4
26	24	230	238	32	110
38	39	156	230	38	100.2
65	53	89	72	67	69.2
139	122	101	97	98	111.4
63	85	81	103	45	75.4
169	199	119	68	81	127.2
147	164	45	95	83	106.8
41	49	32	82	57	52.2
124	123	97	120	122	117.2
121	107	150	114	86	115.6
148	203	152	129	130	152.4
157	170	118	113	125	136.6
125	104	85	81	88	96.6
99	135	91	112	114	110.2
123	104	89	79	68	92.6
80	129	84	131	67	98.2
94	117	52	48	49	72
93	96	62	72	54	75.4

Langkah 2. Proses Rataan Citra

Proses rataan citra adalah proses untuk menghitung nilai rata-rata citra. Matriks rataan citra yang diberi nama U merupakan matriks yang dihasilkan dari rata-rata nilai L1, L2, L3, L4, dan L5. Nilai rataan citra U dapat dilihat pada tabel 1. Matriks U dapat dinyatakan sebagai :

$$U = \frac{1}{M}\sum_{n=1}^{M} L_n \quad \text{.....(13)}$$

Keterangan:
U : Matriks rata-rata
M : Jumlah citra yang digunakan
n : bernilai 1 sampai dengan jumlah citra yang digunakan

Langkah 3. Proses Selisih Antarcitra

Proses selisih antarcitra adalah proses menghitung selisih antara citra wajah yang ada di *database* dengan citra rata-rata membentuk matriks A. Matriks A dihasilkan dari pengurangan matriks L dengan matriks U. Matriks A dapat dinyatakan sebagai:

Ai = Li – U .....(14)

Keterangan:
A : Matriks Selisih Antarcitra
L : Matriks hasil linierisasi
U : Matriks rata-rata
i : bernilai 1 sampai dengan jumlah data yang digunakan Hasil perhitungan matriks selisih antarcitra A dapat dilihat pada tabel 2.

Tabel 2. Nilai Selisih Antarcitra

A1	A2	A3	A4	A5
14.4	-4.6	2.4	-8.6	-3.6
25.4	4.4	-13.6	13.4	-51.6
-3.6	34.4	-58.6	-64.6	-68.6
0.4	27.4	-28.6	-74.6	-61.6
45.4	35.4	-8.6	25.4	1.4
25.4	8.4	-4.6	8.4	-10.6
-111.6	-107.6	3.4	-55.6	-79.6
-91.6	-93.6	112.4	120.4	-85.6
-79.6	-78.6	38.4	112.4	-79.6
-52.6	-64.6	-28.6	-45.6	-50.6
21.4	4.4	-16.6	-20.6	-19.6
-54.6	-32.6	-36.6	-14.6	-72.6
51.4	81.4	1.4	-49.6	-36.6
29.4	46.4	-72.6	-22.6	-34.6
-76.6	-68.6	-85.6	-35.6	-60.6
6.4	5.4	-20.6	2.4	4.4
3.4	-10.6	32.4	-3.6	-31.6
30.4	85.4	34.4	11.4	12.4
39.4	52.4	0.4	-4.6	7.4
7.4	-13.6	-32.6	-36.6	-29.6
-18.6	17.4	-26.6	-5.6	-3.6
5.4	-13.6	-28.6	-38.6	-49.6
-37.6	11.4	-33.6	13.4	-50.6
-23.6	-0.6	-65.6	-69.6	-68.6
-24.6	-21.6	-55.6	-45.6	-63.6

Langkah 4. Proses Matriks Kovarian

Matriks kovarian dihitung dengan mengalikan matriks A dengan transpose-nya. Dimensi matriks yang besar dapat memperberat proses komputasi sehingga dapat dilakukan reduksi dimensi pada saat penghitungan matriks kovarian. Berdasarkan persamaan 6 dapat dihitung terlebih dahulu matriks C pada dimensi lebih rendah yaitu dengan mengalikan nilai transpose matriks A dengan matriks A. Hasil perkalian antara matriks transpose A yang berdimensi 5 x 25 dengan matriks A yang berdimensi 25 x 5 akan mempunyai dimensi 5 x 5. Nilai matriks C dapat dilihat pada tabel 3.

Tabel 3. Nilai Matriks Kovarian

C1	C2	C3	C4	C5
24294.3	23763.3	-22564	-27522	2028.72
23763.3	37215.3	-26584	-31211	-3183.3
-22564	-26584	32686.9	25780.3	-9319.5
-27522	-31211	25780.3	45872.7	-12919
2028.72	-3183.3	-9319.5	-12919	23393.1

Langkah 5. Proses Nilai *Eigen*

Proses nilai *eigen* akan menghitung nilai *eigen*, vektor *eigen*, dan *Eigenface* dari matriks kovarian C. Vektor *eigen* dapat dinyatakan sebagai:

E = [E1 E2 ... Ei].....(15)

Keterangan :
E : matriks vektor *eigen*
I : jumlah citra yang digunakan

Proses nilai eigen akan menghasilkan nilai *eigen* dan vektor *eigen* seperti terlihat pada tabel 4 dan tabel 5.

Tabel 4. Tabel Nilai *Eigen*

E1	E2	E3	E4	E5
116427.8	0	0	0	0
0	28994.45	0	0	0
0	0	0	0	0
0	0	0	5418.38	0
0	0	0	0	12621.8

Tabel 5. Tabel Vektor *Eigen*

V1	V2	V3	V4	V5
-0.42	-0.12	-0.447	0.78	-0.009
-0.506	-0.434	-0.447	-0.596	0.014
0.466	-0.072	-0.447	-0.025	-0.76
0.58	-0.232	-0.447	0.027	0.64
-0.119	0.859	-0.447	-0.187	0.115

Vektor *eigen* yang digunakan adalah vektor *eigen* yang nilai *eigen*-nya tidak sama dengan nol. Vektor *eigen* yang digunakan dapat dilihat pada tabel 6.

Tabel 6. Tabel Vektor *Eigen* yang Digunakan

V1	V2	V3	V4
-0.42	-0.12	0.78	-0.009
-0.506	-0.434	-0.596	0.014
0.466	-0.072	-0.025	-0.76
0.58	-0.232	0.027	0.64
-0.119	0.859	-0.187	0.115

Eigenface dihitung dengan mengalikan nilai selisih antarcitra A dengan nilai vektor *eigen* V. Matriks E dapat dinyatakan sebagai :

E = A V .....(16)

Keterangan :
E : Matriks *Eigenface*
A : Matriks Selisih Antarcitra
V : Matriks vektor *eigen*

Matriks A yang mempunyai dimensi 25 x 5 dikalikan dengan matriks vektor *eigen* yang berdimensi 5 x 4 akan menghasilkan matriks *Eigenface* yang berdimensi 25 x 4. Nilai *Eigenface* dapat dilihat pada tabel 7.

Tabel 7. Tabel *Eigenface*

E1	E2	E3	E4
-7.162	-1.002	14.355	-7.936
-5.315	-51.407	27.536	12.811
-72.474	-54.186	-10.794	-4.183
-63.27	-45.46	-5.826	32.712
-26.442	-24.902	14.972	23.04
-10.934	-17.422	17.135	7.542
80.197	4.439	-9.69	-47.824
218.238	-57.934	0.778	-18.698
165.78	-53.536	1.7	33.214
21.074	3.57	6.371	-13.698
-28.559	-15.333	17.587	-2.953
22.586	-35.598	-9.104	10.158
-86.546	-61.538	-2.943	-36.34
-78.638	-42.906	2.942	37.118
13.623	1.397	-6.417	35.032
-14.151	1.595	1.53	17.716
20.708	-24.448	13.97	-30.741
-34.848	-35.216	-30.023	-16.5
-46.444	-20.094	-1.997	-2.018
-29.103	-9.553	19.219	-2.309
-16.2	-5.19	-23.699	16.629
-25.175	-26.313	21.241	-8.911
8.179	-44.571	-25.478	28.791
-52.513	-34.919	-5.507	-2.373
-23.485	-27.681	5.695	5.677

Langkah 6. Proses Proyeksi

Proyeksi ke ruang wajah pada dasarnya adalah mencari bobot atau nilai kombinasi

linier terbaik untuk merepresentasikan wajah. Kombinasi linier ini dapat dihitung dengan mengalikan matriks selisih antarcitra dengan nilai *Eigenface*-nya. Matriks bobot dapat dinyatakan sebagai :

$$W = E' A \quad \text{.....(17)}$$

Keterangan :
W : Matriks bobot
E' : Transpose dari *Eigenface*
A : Matriks selisih antarcitra

Matriks E' mempunyai dimensi 4 x 25 dikalikan dengan matriks A yang mempunyai dimensi 25 x 5 akan menghasilkan matriks W yang berdimensi 4 x 5. Hasil dari kombinasi linier ini dapat dilihat pada tabel 8.

Tabel 8. Tabel Bobot Citra Wajah

W1	W2	W3	W4	W5
-48947.1	-58923.4	54221.99	67509.67	-13861.1
-3476.15	-12582.3	-2094.85	-6747.62	24900.95
4390.244	-2908.57	-141.195	-130.596	-1209.89
1700.289	3754.704	-9662	5002.532	-795.523

Nilai bobot citra wajah atau vektor fitur akan disimpan pada saat pelatihan sebagai *database* wajah terekstraksi serta merupakan hasil proyeksi citra wajah pada dimensi yang lebih rendah. Namun, nilai bobot citra tes akan langsung dibandingkan dengan nilai bobot dari *database* wajah terekstraksi saat aplikasi bekerja mengenali wajah yang dites.

Semua vektor fitur yang ada di dalam *database* akan dibandingkan dengan vektor fitur dari citra yang dicari identitasnya. Selanjutnya akan diseleksi apakah jarak *Euclidean* antara vektor tersebut lebih kecil dari nilai ambangnya (*treshold*). Nilai ambang yang dipakai adalah 0.1 atau dengan akurasi sebesar 90 %. Dari semua nilai yang memenuhi nilai ambang akan dicari nilai minimumnya. Hal ini dikarenakan semakin kecil jarak *Euclidean* suatu vektor fitur maka semakin mirip citra wajah dengan yang tersimpan di *database*. Identitas pegawai merupakan identitas citra di *database* dengan jarak *Euclidean* paling kecil dari citra yang dicari identitasnya. Sebagai contoh, ekstraksi fitur pada sebuah wajah tes menghasilkan matriks T yang merupakan matriks bobot citra wajah citra tes. Nilai T dan jarak *Euclidean* antara matriks T dengan bobot citra wajah pada tabel 8 dapat dilihat pada tabel 9 dan tabel 10.

Tabel 9. Tabel Bobot Citra Tes

T	54280	-2160	-180	-9680

Tabel 10. Tabel Nilai *Euclidean*

D1	D2	D3	D4	D5
91.2	100.4	0.085	17.8	64.77

Berdasarkan tabel 10 dapat diketahui bahwa nilai *Euclidean* yang memenuhi nilai ambang dan mempunyai nilai minimal adalah D3 dengan nilai jarak *euclidean* 0.085. Hal ini berarti citra wajah ke 3 adalah citra yang paling mirip dengan citra tes.

6. Pengujian

Pengujian dilakukan dengan mengenali 11 orang yang sudah dilatihkan ke dalam *database*. Masing-masing orang melatihkan 20 citra wajahnya. Diambil 219 *Eigenface* untuk merepresentasikan citra wajah dalam pelatihan sehingga didapatkan table hasil pengenalan sebagai berikut :

Tabel 9. Tabel Pengujian Pengenalan Wajah

No	Wajah Tes	Wajah Dikenali Sebagai	Keterangan
1	Sidiq Al Amin	Sidiq Al Amin	Cocok
2	Widodo Muda S	Widodo Muda S	Cocok
3	Eko Budi W	Eko Budi W	Cocok

4	Fitri Ayu	Fitri Ayu	Cocok
5	Meina Riyani	Meina Riyani	Cocok
6	Muhammad K N	Muhammad K N	Cocok
7	Jainin	Jainin	Cocok
8	Seno Harmantyo	Seno Harmantyo	Cocok
9	Shinta K	Shinta K	Cocok
10	Wisnu Nur H	Wisnu Nur H	Cocok
11	Widodo Muda S	Tidak Dikenali	Tidak dikenali

Pengenalan dengan menggunakan metode *Euclidean Distance* pada proses klasifikasinya menunjukkan bahwa akurasi pengenalan wajah dengan algoritma *Eigenface* ini mendekati 91 %. Kesalahan pengenalan wajah terjadi karena perbedaan pencahayaan di tempat sistem bekerja.

7. Kesimpulan

Algoritma *Eigenface* dapat merepresentasikan wajah dengan baik ke dimensi yang lebih rendah dari dimensi citra. Penggunaan algoritma *Eigenface* dan *Euclidean Distance* pada proses klasifikasi memberikan tingkat akurasi sebesar 91 % pada pengenalan sejumlah data. Algoritma *Eigenface* dapat meringankan proses komputasi pada pengenalan sejumlah besar data wajah karena menggunakan metode yang dapat mengurangi dimensi citra.

Daftar Pustaka

[1] Atalay, I., 1996, "*Face Recognition Using Eigenfaces*", Istanbul Technical University Institute Of Science And Technology.

[2] Greenacre,M., 2008, "*Measures of distance between samples: Euclidean*", diakses dari http://www.econ.upf.edu/~michael/standford/maeb4.pdf, pada tanggal 1 September 2011, pukul 20.05 WIB.

[3] Hasanuzzaman dan Haruki Ueno, 2004, "*Face and Gesture Recognition for Human-Robot Interaction*", Bangladesh.

[4] Morshed, R., 2009, "*Face Recognition in the Real World: A Compressive Sensing Perspective*", Pomona College Department of Computer Science.

[5] Turk, M. dan Pentland A., 1991, "*Eigenfaces for recognition*", Journal of Cognitive Neuroscience.

Pembimbing Utama

Pembimbing Anggota

Helmie Arif Wibawa, S.Si, M.Cs
NIP. 19780516 200312 1 001

Nurdin Bahtiar, S.Si, M.T
NIP. 19790720 200312 1 002