Face Recognition Menggunakan Metode PCA

Dian Parikesit
JURUSAN MAGISTER ILMU KOMPUTER UNIVERSITAS BUDI LUHUR

Abstrak

Pelacakan dan pengenalan wajah manusia merupakan salah satu bidang yang cukup berkembang dewasa ini, dimana aplikasi dapat diterapkan dalam bidang keamanan (security system) seperti ijin akses masuk ruangan, pengawasan lokasi (surveillance), maupun pencarian identitas individu pada database kepolisian. Pada tugas akhir ini, pembuatan system pengenalan wajah dilakukan dengan menggunakan algoritma PCA (Principal Component Analysis). PCA adalah salah satu algoritma yang digunakan untuk pengenalan berdasarkan appearance based. PCA ini juga merupakan algoritma reduksi dimensi yang mampu menghasilkan komponen-komponen wajah yaitu eigenface. Pemilihan atau seleksi eigenvector dilakukan untuk mengetahui eigenvector mana yang sesuai dengan kandungan informasi yang lebih tinggi. Hasil yang diperoleh pada tugas akhir ini antara lain jumlah feature yang sedikit PCA memberikan hasil yang lebih baik bila dibangdingkan dengan penggunaan PCA, bahwa nilai minimal terletak pada data ke 12, data ke 12 merupakan orang ke 6 pose ke 2. Sehingga data yang diuji tersebut dikenali sebagai orang yang ke 6 pose dan jumlah ciri ke 10 adalah yang paling minimal.

I. PENDAHULUAN

1.1 Latar Belakang

Sistem pengenalan wajah banyak dimanfaatkan pada biometrics yang digunakan untuk identifikasi personal pada penggunaan mesin absensi, akses kontrol dan lain-lain. Secara umum sistem pengenalan image tidak menggunakan bitmap pixel secara langsung melainkan ia bekerja pada domain feature. Image direpresentasikan kedalam bentuk feature yang lebih kompak yang kemudian digunakan untuk pengenalan, dengan demikian dapat menghemat komputasi. Berbagai metode ekstrasi feature telah dimanfaatkan seperti metode moment, feature filter Gabor, Wavelet, dan lain-lain. Metode PCA dikenal juga dengan nama Karhunen-Loeve transformation (KLT), yang telah dikenal sejak 30 tahun dalam dunia pengenalan pola. PCA memberikan transformasi ortogonal yang disebut dengan 'eigenimage' yang mana sebuah image direpresentasikan kedalam bentuk proyeksi linier searah dengan eigenimage yang bersesuaian dengan nilai eigen terbesar dari matrix covariance (atau scatter matrix). Secara praktis matrix covariance ini dibangun dari sekumpulan image training yang diambil dari berbagai obyek/kelas.

1.2 Tujuan Penelitian

Untuk membuat software yang dapat mengidentifikasi wajah seseorang berdasarkan eigen face yang dimiliki suatu image, dan dibandingkan dengan image yang menjadi training pada database. Untuk metode pengenalannya, digunakan metode
PCA (Prinsipal Component Analysis)

1.3 Batasan Masalah

Dalam makalah ini beberapa hal terkait yang akan di batasi dalam pembahasan antara lain :

1. System warna RGB adalah kombinasi aditif dari ketiga warna utama merah, hijau dan

biru. Ketiga warna ini di campur bersama maka akan diperoleh warna putih atau warna hitam tergantung nilai RGB dari pixel-pixel tersebut. System warna RGB merupakan system warna standard yang dipakai. Warna RGB terdiri dari 24 bit.masing- masing 8 bit untuk Red, Green dan Blue. Bila masing-masing 8 bit itu memiliki nilai yang sama. Maka akan didapatkan image greyscale. Dalam makalah ini, di gunakan image-image grayscale.

2. Dalam proses training digunakan PCA. PCA digunakan untuk mereduksi dimensi dari image-image yang diproses.

II. ALGORITMA PRINCIPAL COMPONENTS ANALYSIS (PCA)

2.1 Principal Components Analysis (PCA)

Pengenalan wajah adalah suatu masalah pada pengenalan pola visual. Dimana dalam suatu wajah yang direpresentasikan menjadi suatu citra tiga dimensi (3D) terdapat didalamnya variasi tingkat pecerahan, pencahayaan, pose, ekspresi dan lain-lain yang kemudian dilakukan proses identifikasi berdasarkan informasi citra dua dimensinya (2D). Suatu parameter terdekat yang digunakan untuk proses pengenalan wajah ini salah satunya yaitu melalui pencarian lokasi fitur khusus (Local fitur-based) dari citra, seperti mata, hidung dan mulut, yang kemudian dilakukan perhitungan jarak antar fiturnya. Metode lain untuk pengenalan wajah dapat dilakukan dengan membandingkan citra yang telah diproyeksikan menjadi level grayscale menjadi citra yang memiliki dimensi rendah, metode ini biasa disebut dengan metode eigenfaces (Holistic fitur-based).

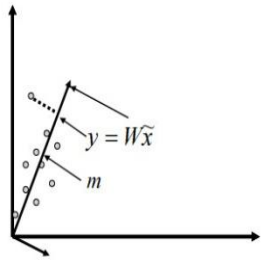
- Untuk reduksi dimensi data (Dimensional Reduction)
- Ekstraksi struktur data dari dataset high dimenson dimenson.
- Mencari basis signal berdasarkan data statistik objek objek. .

PCA

- Let x_i = data lives in M-dimensional space
- x_i is a column vector
- Let N = number of data
- Matrix A = [x_1 x_2 x_3 . . . x_N]

PCA algorithms
- Correlation matrix
- Singular value decomposition

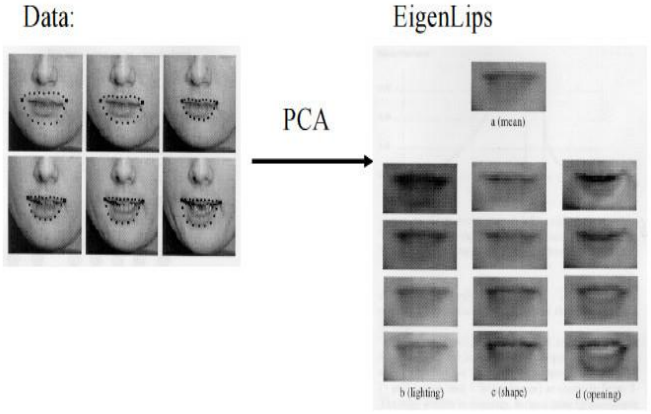
Principal Components Analysis:



$$s_T^2 = \sum_{n=1}^{N} (y[n] - m)^2$$

$$S_T = \sum_{n=1}^{N} (\tilde{x} - \mu)(\tilde{x} - \mu)^T$$

$$s_T^2 = W S_T W^T$$

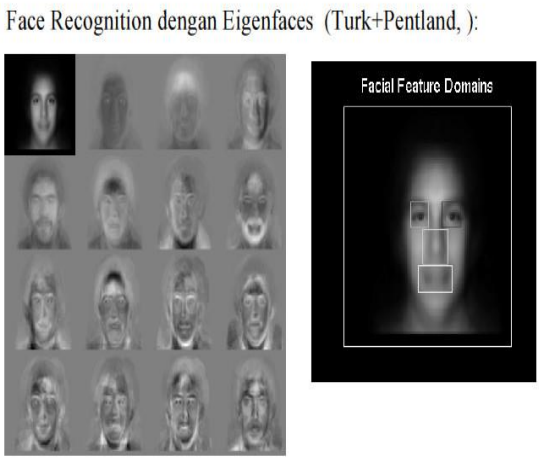


Matriks covariance dengan dimensi m x m, dimana m adalah jumlah image training. Dengan dekomposisi eigen berlaku :

$$\boldsymbol{V\,V\,C} \times = \times \lambda$$

Hitungi eigen value (λ) dan eigen vector (V) dari matriks C menggunakan metode Jacobi. Eigen value dan eigen vector yang bersesuaian diurutkan secara descending . Eigen faces yang

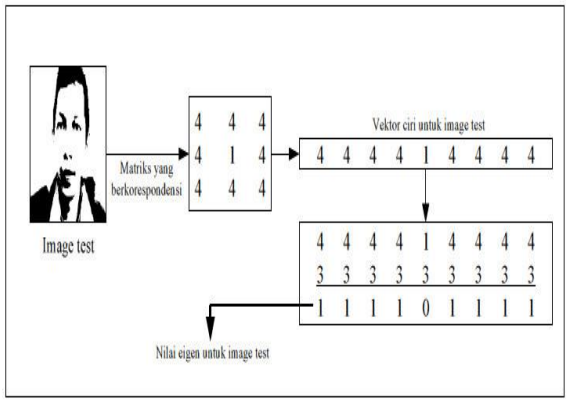
didapatkan ini dapat dilihat dalam ilustrasi di bawah ini :



III. PERANCANGAN SISTEM

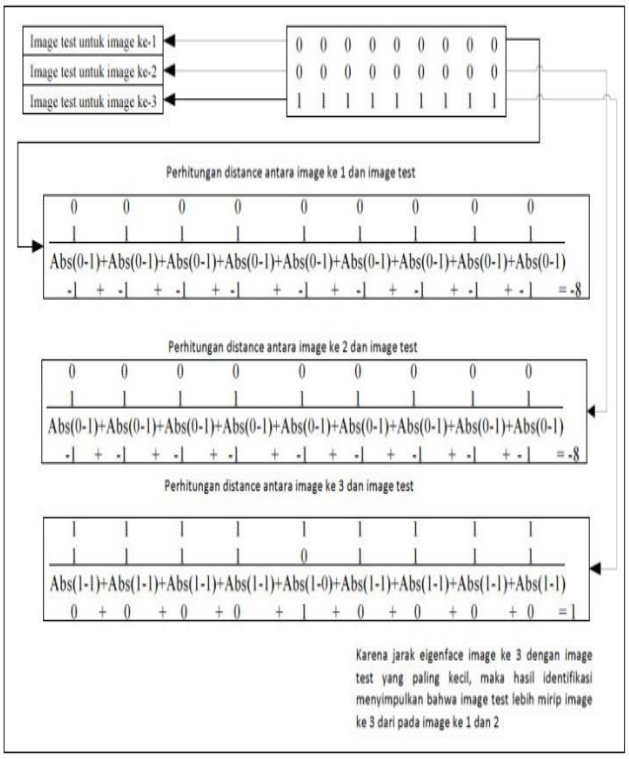
Ektraksi PCA *image test*

Hasil proyeksi tersebut diektraksi dengan perhitungan PCA untuk mendapatkan *feature* dari image. *Feature* adalah komponen–komponen penting dari *image-image training* yang didapatkan dari proses *training. Feature* inilah yang nanti akan digunakan untuk mengidentifikasikan *image* yang akan dikenali. Kalkulasi nilai *eigenface* untuk matriks *testface,* dengan cara yang sama dengan penentuan *eigenface* untuk *vector* ciri.



Gambar 1. Perhitungan *eigenface* untuk *image test*

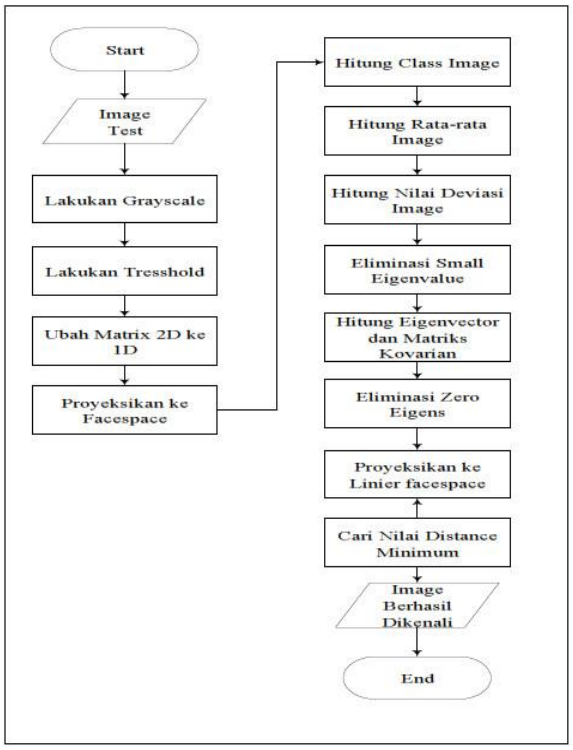
Cari *distance* minimum antara *image test* dan *image* hasil *training.* Bandingkan nilai *euclidean distance minimum* dari *image* yang di-*capture* dengan image yang sudah ada di *database.* Setelah nilai *eigenface* untuk *image test* diperoleh maka kita bisa melakukan identifikasi dengan menentukan jarak (*distance)* terpendek dengan *eigenface* dari *eigenvector training image.* Caranya dengan menentukan nilai *absolute* dari pengurangan baris *i* pada matriks *eigenface training image* dengan *eigenface* dari *testface,* kemudian jumlahkan elemen-elemen penyusun vektor yang dihasilkan dari pengurangan tadi dan ditemukan jarak *d* indeks *i.* Lakukan untuk semua baris. Cari nilai *d* yang paling kecil.



Gambar 2. Proses Indentifikasi dengan *Input Image test*

Langkah-langkah dari proses pengenalan wajah *(recognition)* dari sebuah citra dengan

metode *fisherface* terlihat pada *flowchart* (Gambar 3) dibawah ini :



Gambar 3. *Flowchart* Tahapan Proses Pengenalan Wajah Metode *Fisherface*

Contoh Contoh: : Eigenfaces vs Fisherfaces

Konstruksi *fisherface* adalah pembuatan suatu set *fisherface* dari suatu set gambar *training* dengan menggunakan perhitungan PCA dan FLD. Perhitungan PCA dilakukan dengan langkah sebagai berikut :

1. Buat *MakeFlatVectors* (*ImageList, N, M):* Image List adalah kumpulan dari *N training image,* dimana setiap image adalah *W* x *H* piksel. *M* adalah ukuran dari *vector flat* yang harus dibuat.
2. Gabungkan setiap *image* dalam *WH* elemen *vector* dengan menggabungkan semua baris. Buat *image* matriks sebagai matriks *N* x *WH* berisi semua gambar yang digabung.
3. Jumlahkan semua baris pada *image* matriks dan bagi dengan N untuk mendapatkan rataan vektor gabungan. Namakan vektor elemen *WH* dengan *R.*
4. Kurangi *image* matriks dengan *average image R.* Namakan matriks baru ukuran *N* x *WH* sebagai *R'.* Kemudian identifikasi dilakukan dengan menggunakan perhitungan FLD sebagai berikut:

1. Buat ProjectToFaceSpace(test_image): Image berukuran *W* x *H* Piksel.
2. Transformasikan *training set* ke dalam vektor kolom.
3. Bentuk *average face* dari *facespace*
4. Hitung nilai deviasi *image.*
5. Eliminasi nilai terkecil *eigenvalue.*
6. Hitung *eigenvector* dan matriks kovarian.
7. Hitung matriks proyeksi *fisher* dengan mengurutkan vektor eigen berdasarkan besarnya nilai *eigen* masing-masing vektor *eigen* dan mengambil komponen vektor *eigen* yang memiliki nilai *eigen* tidak nol.
8. Hitung bobot tiap *fisherface* terhadap masing-masing gambar wajah pada *training set* dengan memproyeksikan nilai deviasi *facespace* terhadap *average face* kedalam matriks proyeksi optimal. Proses terakir adalah identifikasi yaitu dengan memproyeksikan *test image* ke *face space* dan menghitung *score.*

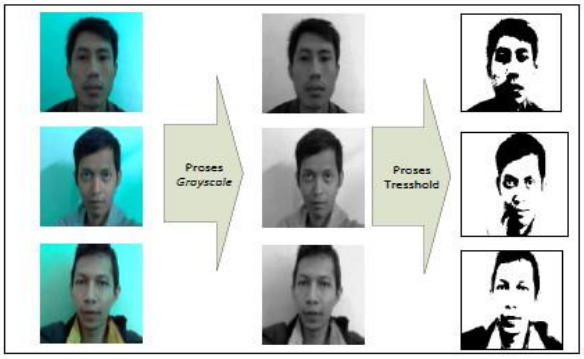
1. Load semua wajah yang sudah diproyeksikan ke database
2. Proj=projectToFaceSpace(test_image).
3. Lakukan operasi pengurangan, proj dengan semua wajah yang telah diproyeksikan. Ambil nilai absolutnya dan jumlahkan, hasil adalah *distance.*

4. Hitung jarak *distance* antara bobot *image test* dan bobot *training set.*
5. Ambil *distance* terkecil sebagai hasil dari wajah yang telah diproyeksikan. Wajah ini menjadi hasil identifikasi.

Untuk lebih jelasnya algoritma *fisherface* dapat diuraikan dalam ilustrasi sebagai berikut :

1. Proses *Image Test*

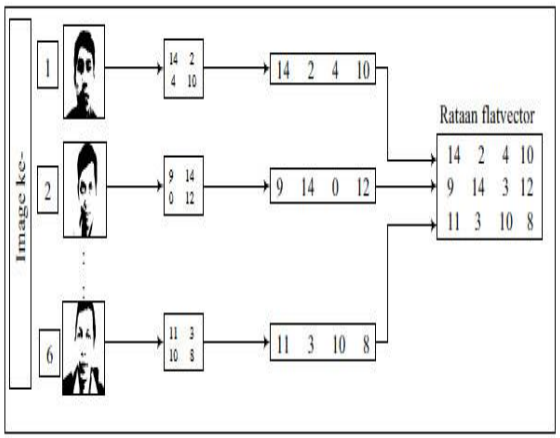
Langkah pertama setelah didapatkan citra wajah hasil *capture,* citra wajah dirubah ke dari bentuk RGB ke dalam bentuk *grayscale,* setelah didapatkan citra keabuan ubah menjadi citra hitam putih dengan melakukan *tresshold* agar kompleksitas citra lebih sederhana.



Gambar 3.12 Proses Mengubah Citra RGB ke *grayscale* dan *tresshold*

Penyusunan *flatvector*

Langkah pertama adalah menyusun seluruh *training image* menjadi suatu matriks tunggal. Misalnya *image* yang disimpan berukuran *H* x *W* piksel dan jumlahnya *N* buah, maka akan dimiliki *vector* ciri dengan dimensi *N* x (*W* x *H).* Misalnya dalam *training image* terdapat 3 *image* dengan ukuran 3 x 4 piksel maka kita akan mempunyai *eigenvector* ukuran 3 x 9. Ilustrasinya sebagai berikut :



Gambar 4. Penyusunan *flatvector*

IV PENGUJIAN SISTEM

PerhitunganTingkat Kemiripan

Dari *vector* ciri yang telah diperoleh, jumlahkan seluruh barisnya sehingga diperoleh matriks berukuran 1 x (*W* x *H).* Setelah itu bagi matriks tersebut dengan jumlah *image N* untuk mendapatkan nilai rata-rata *vector* ciri, ilustrasinya sebagai berikut :

Tabel 5. Ciri dari 7 Orang, Masing-masing 2 pose

Orang ke	j	$f_{j,1}$	$f_{j,2}$	$f_{j,3}$	$f_{j,4}$
1	1	14	4	4	10
	2	3	2	2	5
2	3	9	0	0	12
	4	7	11	11	7
3	5	13	6	6	10
	6	11	13	13	6
4	7	6	6	6	4
	8	0	6	6	2
5	9	12	12	12	2
	10	6	7	7	10
6	11	9	3	3	4
	12	11	10	10	8
7	13	13	12	12	2
	14	11	0	0	10

Misalkan citra yang diuji mempunyai ciri ke 1, 2, 3 dan 4 dan masing-masing adalah 7, 6, 10, 9. Maka beda kuadrat dari masing-masing ciri pelatihan terhadap citra uji dapat dilihat pada tabel 5. Dengan persamaan

$$(f_{j,i} - f_{uji,i})^2 \qquad (3.2)$$

Maka didapatkan nilai beda kuadrat dari masing-masing ciri pelatihan terhadap citra uji.

Tabel 6 Beda Kuadrat dari masing-masing ciri pelatihan Terhadap Citra Uji

No	Ciri ke-1	Ciri ke-2	Ciri ke-3	Ciri ke-4
1	$(14-7)^2$	$(2-6)^2$	$(4-10)^2$	$(10-9)^2$
2	$(3-7)^2$	$(6-6)^2$	$(2-10)^2$	$(5-9)^2$
3	$(9-7)^2$	$(14-6)^2$	$(0-10)^2$	$(12-9)^2$
4	$(7-7)^2$	$(13-6)^2$	$(11-10)^2$	$(7-9)^2$
5	$(13-7)^2$	$(6-6)^2$	$(6-10)^2$	$(10-9)^2$
6	$(11-7)^2$	$(13-6)^2$	$(13-10)^2$	$(6-9)^2$
7	$(6-7)^2$	$(0-6)^2$	$(6-10)^2$	$(4-9)^2$
8	$(0-7)^2$	$(5-6)^2$	$(6-10)^2$	$(2-9)^2$
9	$(12-7)^2$	$(12-6)^2$	$(12-10)^2$	$(2-9)^2$
10	$(6-7)^2$	$(0-6)^2$	$(7-10)^2$	$(10-9)^2$
11	$(9-7)^2$	$(2-6)^2$	$(3-10)^2$	$(4-9)^2$
12	$(11-7)^2$	$(3-6)^2$	$(10-10)^2$	$(8-9)^2$
13	$(13-7)^2$	$(2-6)^2$	$(12-10)^2$	$(2-9)^2$
14	$(11-7)^2$	$(9-6)^2$	$(0-10)^2$	$(10-9)^2$

Tabel 7. Hasil Penjumlahan Beda Kuadrat Dari Masing-Masing Ciri Pelatihan Terhadap Citra Uji

No	Ciri ke-1	Ciri ke-2	Ciri ke-3	Ciri ke-4	Jumlah
1	49	16	36	1	102
2	16	0	64	16	96
3	4	64	100	9	177
4	0	49	1	4	54
5	36	0	16	1	53
6	16	49	9	9	83
7	1	36	16	25	78
8	49	1	16	49	115
9	25	36	4	49	114
10	1	36	9	1	47
11	4	16	49	25	94
12	16	9	0	1	26
13	36	16	4	49	105
14	16	9	100	1	126

Cari *distance* minimum antara *image test* dan *image* hasil *training.* Bandingkan nilai *euclidean distance minimum* dari *image* yang di-*capture* dengan image yang sudah ada di *database.* Setelah nilai *eigenface* untuk *image test* diperoleh maka kita bisa melakukan identifikasi dengan menentukan jarak (*distance)* minimum dengan *eigenface* dari *eigenvector training image.* Dengan persamaan :

$$d_v = \sum_{j-1}^{m} \| F_{trainke\ j,w} - F_{uji,w} \|$$

Maka akan didapatkan nilai distance minimum dari setiap pose *image.* Seperti terlihat pada Tabel 3.6 dibawah ini :

No	Jumlah Ciri	Nilai *Eucledean Distance*	Hasil
1	102	10.1	
2	96	9.798	
3	177	13.304	
4	54	7.3485	
5	53	9.1104	
6	83	8.8318	
7	78	10.724	
8	115	6.8557	
9	114	9.9654	
10	47	**5.099**	**Minimal**
11	94	10.247	
12	26	11.225	

Tabel 8. Hasil Akhir Perhitungan Tingkat Kemiripan Menggunakan *Eucledean Distance*

V KESIMPULAN

Dari hasil percobaan ternyata secara umum bila variasi image training cukup tinggi (misalnya: iluminasi dan ekspresi) maka penggunaan PCA akan memberikan kontribusi yang tinggi. Bahkan dengan jumlah feature yang sedikit PCA memberikan hasil yang lebih baik bila dibangdingkan dengan penggunaan PCA saja. Pengambilan jumlah *feature* yang dihitung harus dipertimbangkan. Bila terlalu sedikit atau terlalu banyak akan menurunkan *recognition percentage.* Jumlah *feature* yang benar-benar optimal bisa didapatkan dengan melakukan eksperimen berulang-ulang. Dari eksperimen yang telah dilakukan, bahwa nilai minimal terletak pada data ke 12, data ke 12 merupakan orang ke 6 pose ke 2. Sehingga

data yang diuji tersebut dikenali sebagai orang yang ke 6 pose dan jumlah ciri ke 10 adalah yang paling minimal.

DAFTAR PUSTAKA

[1] R. Chellappa, C. Wilson, and S. Sirohey, "Human and machine recognition: A survey," *Proceedings of the IEEE*, vol. 83, no. 5, pp. 705--740, 1995.

[2] M. Turk and A. Pentland, "Eigenfaces for recognition." *Journal of Cognitive Neuroscience,* Vol. 3, pp. 71-86, 1991.

[3] K. Etemad and R. Chellappa, "Discriminant Analysis for Recognition of Human Face Images," *Journal of Optical Society of America A,* pp. 1724-1733, Aug. 1997.

[4] P.N. Belhumeur, J.P. Hespanha, and D.J. Kriegman, "Eigenfaces vs. fisherfaces: Recognition using class specific linear projection", *IEEE Trans. PAMI,* Vol. 19, No. 7, July 1997.

[5] Press, William H., et. al., *Numerical Recipes In C, The Art Of Scientific Computing,* Second Edition, Cambridge : Cambridge University Press, 1995.

[6] Nayar, Shree K. and Tomaso Poggio, *Early Visual Learning.* Oxford : Oxford University Press, 1996.

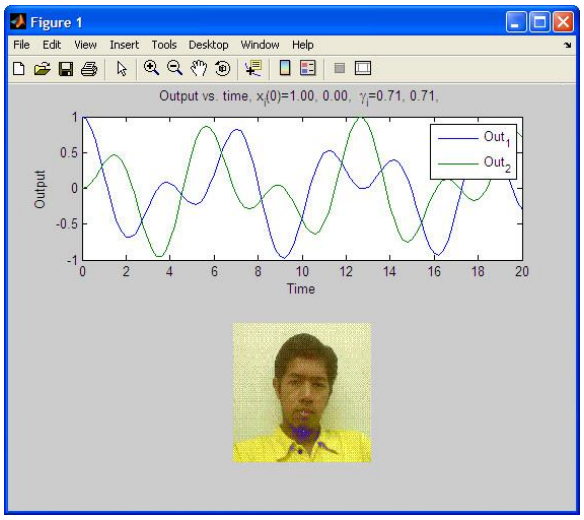
Lampiran :

Gambar Asli

```
clear all;close all;clc
[fly,map] = imread('E:\dian.gif');
fly=double(fly);
whos
image(fly)
colormap(map)
axis off, axis equal
[m n]=size(fly);
mn = mean(fly,2);
X = fly - repmat(mn,1,n);
Z=1/sqrt(n-1) * X';
covZ=Z' * Z;
[U,S,V] = svd(covZ);
variances=diag(S) .* diag(S);
bar(variances,'b')
xlim([0 20])
xlabel('eigenvector number')
ylabel('eigenvalue')
tot=sum(variances)
[[1:512]' cumsum(variances)/tot]
PCs=40;
VV=V(:,1:PCs);
Y=VV' * X;
ratio=512/(2 * PCs+1)
XX=VV * Y;
XX=XX+repmat(mn,1,n);
image(XX)
colormap(map)
axis off, axis equal
z=1;
for PCs=[2 6 10 14 20 30 40 60 90 120 150 180]
VV=V(:,1:PCs);
Y=VV' * X;
XX=VV * Y;
XX=XX+repmat(mn,1,n);
subplot(4,3,z)
z=z+1;
image(XX)
colormap(map)
axis off, axis equal
title({[num2str(round(10 * 512/(2 * PCs+1))/10) ':1 compression'];..
[int2str(PCs) ' principal components']})
end
```

Script Matlab PCA

Hasilnya :



Diperbesar hasil PCA :