LAMPIRAN A

KODE PROGRAM

```
function recog
%program utama

clear
close all

fprintf('\n1. Masukan lokasi folder training set > ');
Train.Path = uigetdir('C:\');
if Train.Path==0
   return
end
fprintf('%s',Train.Path);

fprintf('\n2. Masukan lokasi test image > ');
[file path]= uigetfile({'*.jpg','JPEG files (*.jpg)';'*.gif','GIF files (*.gif)'},...
   'Input Image','C:\');

if isequal(file,0) | isequal(path,0)
   return
else
   Recog.Path = fullfile(path,file);
end
fprintf('%s',Recog.Path);

fprintf('\n3. Tentukan metode klasifikasi ');
klas = input('\n   (Euclidian Distance = 1, Cosine Distance = 2) > ');
dBase = input('4. Training Set (Data Base) BARU ? (yes = 1, no = 0) > ');

%==================================
%membaca input
%==================================

 [Train,Nxt,Nyt,M,map] = BacaFile(Train);

if M==0 %cek apakah file tidak ada
   fprintf('Error: tidak ada training image pada direktori: %s \n\n', Train.Path);
   clear
   return
end

%baca kelas dari training faces (database).
TrainClass = BacaKelas(Train,M,Nxt,Nyt);

%membaca gambar input
Recog = deteksiwajah(Recog,Nxt,Nyt);     %Face Detecting
```

```
%==================================
% menghitung PCA & SVM
%==================================

svdFile = fullfile(Train.Path,'saveVar.mat');
foundCache = exist(svdFile,'file');
MpFish = M-TrainClass.bnykclass;

if foundCache & ~dBase
   load(svdFile)
else
   [Wpca,Train] = Hitung_PCA(Train,MpSvm);

   [Wsvm,Train] = SVM(Train,TrainClass,Wpca);

   save(svdFile,'Wpca','Wsvm','Train');
end

%==================================
% Proses Klasifikasi Wajah
%==================================

if klas == 1
   % klasifikasi dengan euclidian distance
   [Recog] = eudist(Recog, Train, Wsvm, 10.5);
elseif klas == 2
   % klasifikasi dengan cosine distance
   [Recog] = cosinedist(Recog,Train,Wsvm,-0.6);
else
   fprintf('\n\nTentukan Metode Klasifikasi!!');
   clear;
   return;
end


fprintf('\n')
fprintf('-------------------------HASIL--------------------------\n');
fprintf('Banyak gambar dalam training set adalah = %d \n', M);
fprintf('Banyak kelas dalam training set adalah = %d \n', TrainClass.bnykclass);
fprintf('Jarak Minimum = %f \n', Recog.minDis);
fprintf('indeks training set ke - %d\n', Recog.classEst);
fprintf('Input face dikenali sebagai = %s \n\n', Recog.classNameEst);

%==================================
% plot image
%==================================
```

```
iptsetpref('ImshowAxesVisible','on')
rgb = imread(Recog.Path);
figure,imshow(rgb),h=rectangle('position',Recog.Posisi);
set(h,'EdgeColor','white'),title('Gambar Masukan');

figure,imshow(Recog.Image,map),title('Input Face');
figure,montage(reshape(Train.Image,[Nxt Nyt 1 M]),map),title('Training Set');
figure,imshow(reshape(Train.Mean,[Nxt Nyt])),title('Average Image');

% plot pricipal component analisys
I=zeros(Nxt, Nyt, 1, MpSvm);
I = reshape(Wpca,[Nxt Nyt 1 MpSVM]);
for i = 1:MpSVM % mengubah skala menjadi 0 - 1
   mx = max(Wpca(:,i));
   mi = min(Wpca(:,i));
   I(:,:,1,i) = (I(:,:,1,i)-mi)./(mx-mi);
end
figure,montage(I),title('Principal Component Analisys'); % PCA

% plot svm
I=zeros(Nxt, Nyt, 1, TrainClass.bnykclass - 1);
I = reshape(Wsvm,[Nxt Nyt 1 TrainClass.bnykclass - 1]);
for i = 1:(TrainClass.bnykclass - 1) % skala untuk diplot
   mx = max(Wsvm(:,i));
   mi = min(Wsvm(:,i));
   I(:,:,1,i) = (I(:,:,1,i)-mi)./(mx-mi);
end
figure,montage(I),title('SVM'); % svm

% Plot rekonstruksi images
I=zeros(Nxt, Nyt, 1, M);
I = reshape(Train.reconst,[Nxt Nyt 1 M]);
for i = 1:M % skala untuk diplot
   mx = max(Train.reconst(:,i));
   mi = min(Train.reconst(:,i));
   I(:,:,1,i) = (I(:,:,1,i)-mi)./(mx-mi);
end
figure,montage(I),title('Rekonstruksi Training Set');

I = reshape(Recog.reConstructed,[Nxt Nyt]);
% skala untuk plot
mx = max(Recog.reConstructed);
mi = min(Recog.reConstructed);
I = (I-mi)./(mx-mi);
figure,subplot(1,2,1),imshow(Recog.Image),title('Input Face');
subplot(1,2,2),imshow(I),title('Rekonstruksi Input Face');
```

```
figure,subplot(1,2,1),stem(j,Recog.Wt),title({'Bobot'; 'Wajah Masukan'});grid
ylabel('Bobot (Weight)'),xlabel('Index SVM');
subplot(1,2,2),stem(i,Recog.Dist),
if klas ==1
   title('Jarak Euclidian'),ylabel('Euclidian Distance');
   xlabel('Index Training Set'),grid;
elseif klas ==2
   title('Jarak Cosinus'),ylabel('Cosine Distance');
   xlabel('Index Training Set'),grid;
end

if Recog.classEst~=0
   I = Train.Image(:,:,Recog.classEst);
   figure,subplot(1,2,1),imshow(Recog.Image),title('Gambar Input');
   subplot(1,2,2),imshow(I),title({'Dikenali'; 'Sebagai'});
end
```

```
function [I,Nx,Ny,M,map] = BacaFile(I)

di = dir(fullfile(I.Path,'*.gif'));   % membaca image pd direktori
if isempty(di)
   M = 0;Nx = [];Ny = [];map = [];  % tidak ada file
   return
end
Info = imfinfo(fullfile(I.Path,di(1).name));
Nx = Info.Height;
Ny = Info.Width;
M = length(di);   % banyaknya images pd direktori
I.Image = uint8(zeros(Nx,Ny,M));  % membuat inisialisasi, biar proses lebih cepat

for i = 1:M
   [Im map] = imread(fullfile(I.Path,di(i).name),'gif');
   I.Image(:,:,i) = Im;
   I.NamaFile{i} = di(i).name(1:end-4); % nama gambar
end
```

```
function class = BacaKelas(train,M,Nx,Ny)
%mengelompokan/membagi2kan image_training kedalam kelas2nya berdasarkan
%3huruf pertama dari nama file image_training.

di = dir(fullfile(train.Path,'*.gif'));   % membaca image pd direktor
indeks = 1;
bataskelas = 0;
bnykclass = 0;

for i = 1:M
   if i>1
      if ~strcmp(di(i-1).name(1:3),di(i).name(1:3))
      j = i;
      bataskelas = 1;
      end
   end

   if i==M % force class bndry for last image
         j = i+1;
         bataskelas = 1;
   end

   if bataskelas
      bnykclass = bnykclass + 1;
      class.NamaClass{bnykclass} = ['class_',di(j-1).name(1:3)];
      class.bnykimage(bnykclass) = j-indeks;
      I = double(...
         reshape(train.Image(:,:,indeks:(j-1)),[Nx*Ny (j-indeks)])./255;
      class.Mean{bnykclass} = mean(I,2);
      class.mulaiClass(bnykclass) = indeks;
      class.akhirClass(bnykclass) = j-1;
      indeks = j;
      bataskelas = 0;
  end
end
class.bnykclass = bnykclass;
```

```
function recog = deteksiwajah(recog,Nxt,Nyt)
%program untuk mendeteksi posisi wajah dari gambar
%masukan dan memisahkan bagian wajah.

rgb = imread(recog.Path);
gray = rgb2gray(rgb);

[Nx Ny M] = size(rgb);
if M ~=3
   error(['masukan harus berupa matriks MxNx3,'...
       ' atau matriks dari gambar berwarna'])
end

yuv = rgb2ycbcr(rgb);
y = yuv(:,:,1);
cb = yuv(:,:,2);
cr = yuv(:,:,3);

%menerapkan thresshold pada Cb & Cr.
cbb = zeros([Nx Ny]);
crr = zeros([Nx Ny]);

i1 = find(cb>105 & cb<125);
i2 = find(cr>135 & cr<160);

cbb(i1) = 1;
crr(i2) = 1;

bwimage = immultiply(cbb,crr);

%menerapkan filter pada gambar biner untuk menghilangkan noise pd gambar
%yaitu operasi erosi dan dilasi
filt = strel('disk',2);
filt2 = strel('disk',3);

bwimage = imerode(bwimage,filt);
bwimage = imdilate(bwimage,filt2);

%memeriksa nilai euler pada bagian2 gambar.
%pada dasarnya sebuah wajah terdiri minimal atas 3 hole (lubang)
%jadi sebuah wajah memiliki nilai euler < -1.

[l,n] = bwlabel(bwimage);   %memberikan label pada bagian2 gambar
```

```
i = 0;
while i~=n

   i = i+1;
  % mencari koordinat daerah (region).
  [x,y] = find(l == i);

  % mengambil daerah (region) yang dipilih saja
  bw = bwselect(bwimage,y,x);

  % mencari nilai euler dari region.
  eul = bweuler(bw);

  % memeriksa nilai euler, sebuah wajah minimal memiliki lebih
  % dari 2 lubang (hole)

  if (eul < -1)
     i = n;
  end;
end

%mencari batas wajah
bw = bwfill(bw,'holes');

bawah = 0;
atas = Ny;
kanan = 0;
kiri = Nx;

for i = 1:Nx
   for j = 1:Ny
      if bw(i,j) == 1
         if bawah < i     %mencari titik atas
            bawah = i;
         end
         if atas > i    %mencari titik bawah
            atas = i;
         end
         if kanan < j     %mencari titik kiri
            kanan = j;
         end
         if kiri > j    %mencari titik kanan
            kiri = j;
         end
      end
   end
end
```

```
panjang = (bawah-atas);
lebar = (kanan-kiri);
ratio = panjang / lebar;

if ratio > 1.5
   panjang = floor(1.5 * lebar);
   bw(atas+panjang:Nx,:) = 0;
end

%menghitung posisi tengah wajah
[cx, cy] = center(bw);

startx = cx - floor(panjang/2);
starty = cy - floor(panjang/2);

% memisahkan wajah dari gambar
recog.Posisi = [startx, starty, panjang, panjang];
keluaran = imcrop(gray,recog.Posisi);
recog.Image = imresize(keluaran,[Nxt Nyt]);
```

```
function [xmean, ymean] = center(bw)
% menghitung posisi tengah wajah pada topeng

area = bwarea(bw);
[m n] =size(bw);

xmean =0; ymean = 0;
for i=1:m,
  for j=1:n,
     xmean = xmean + j*bw(i,j);
     ymean = ymean + i*bw(i,j);
  end;
end;

xmean = xmean/area;
ymean = ymean/area;

xmean = round(xmean);
ymean = round(ymean);
```

```
function [Wpca,train] = Hitung_PCA(train,Mp)

[Nx Ny M] = size(train.Image);
X = double(reshape(train.Image,[Nx*Ny M]))./255; % 1 kolom per wajah
train.Mean = mean(X,2);
A = X - repmat(train.Mean,[1 M]);

C = A'*A;
[V D] = eig(C);

% mengurutkan eigen vektor berdasarkan besar nilai eigen
% dari besar ke kecil untuk mendapatkan vektor dominan

[eigVal ndx] = sort(diag(D),'descend');
V = V(:,ndx);

% mengambil N-C komponen dengan nilai eigen terbesar
% mencari eigen vektor

eigVec = A * V;

% normalisasi eigen vektor --> PCA(eigenface)

Wpca = eigVec./repmat(sqrt(sum(eigVec.^2)),Nx*Ny,1);
Wpca = Wpca(:,1:Mp);
```

```
function [P,train] = SVM(train,TrainClass,Wpca)

[Nx Ny M] = size(train.Image);

X = double(reshape(train.Image,[Nx*Ny M]))./255; % 1 kolom per wajah
me = mean(X,2);
A = X - repmat(me,[1 M]);

[Sb,Sw]=ScatterMatriks(TrainClass,me,X,Nx,Ny);    %scatter matrik Sb & Sw

Sbb = Wpca.'*Sb*Wpca;
Sww = Wpca.'*Sw*Wpca;
clear Sb Sw

% mencari vektor dan nilai eigen dari Sww dan Sbb, eig(A,B)
[V,D] = eig(Sbb,Sww);

% mengurutkan vektor eigen dengan nilai eigen terbesar
Ds = diag(D);
[eigVals,ndx] = sort(abs(Ds),'descend');
V = V(:,ndx);
% mencari matrik proyeksi maksimal
eigVecs = Wpca*V;
clear D Ds V

% mengambil C-1 komponen dengan nilai eigen terbesar, dan
% normalisasi eigen vektor
Mp = TrainClass.bnykclass-1;
lamda = eigVals(1:Mp); % bobot masing2 svm
P = eigVecs(:,1:Mp);    % ouput svm
P = P./repmat(sum(P.^2).^0.5,Nx*Ny,1);

% proyeksi setiap gambar pada training set kedalam setiap svm
% mengambil bobot svm terhadap setiap gambar pd training set
train.Wt = P.'*A;

% Rekonstruksi training set
train.reconst = P*train.Wt + repmat(train.Mean,[1 M]);

train.svm = eigVecs./repmat(sum(eigVecs.^2).^0.5,Nx*Ny,1);
train.eigVals = lamda;
```

```
function [Sb,Sw]=ScatterMatriks(trainClass,rata2,wajah,Nx,Ny)
% between-class scatter matrix, Sb
% within-class scatter matrix,  Sw

prod = zeros(Nx*Ny);
Sb = zeros(Nx*Ny);
for i = 1:trainClass.bnykclass
   row = trainClass.Mean{i} - rata2;
   prod = row * row';  %ada masalah dalam hal besar memori
   Sb = Sb + prod;
end

Sw = zeros(Nx*Ny);
for i = 1:trainClass.bnykclass
   for j = (trainClass.mulaiClass(i)):(trainClass.akhirClass(i))
      row = wajah(:,j) - trainClass.Mean{i};
      prod = row * row';  %ada masalah dalam hal besar memori
      Sw = Sw + prod;
   end
end
clear prod row
```

```
function [recog] = eudist(recog,train,Wp,thresh)
% klasifikasi dengan metode Euclidian distance

% mencari ukuran gambar, [Nx Ny], dan banyaknya training images, M
[Nx Ny M] = size(train.Image);

% Inisialisasi input face
Mp = length(Wp(1,:));
X2 = double(reshape(recog.Image,[Nx*Ny 1]))./255; %input image dalam matrik
kolom
A2 = X2 - train.Mean;
% Proyeksi input face ke seluruh svm, mencari bobot masing2
% svm
recog.Wt = Wp'*A2;
% rekonstruksi input face
recog.reConstructed = Wp*recog.Wt + train.Mean;

% mencari jarak Euclidian dari input face terhadap masing2 training images
recog.Dist = zeros(M,1);
for i = 1:M % banyaknya training images
     x = recog.Wt - train.Wt(:,i);
     recog.Dist(i) = x'*x;
end

% Klasifikasi menggunakan Nearest-Neighbor dengan thresholds:
%   thresh  => batas jarak Euclidian, untuk gambar input dapat dinyatakan
%              terdapat pada training images

[minDis ndx] = min(recog.Dist);

if minDis > thresh
   recog.classNameEst = 'Tidak Dikenali';
   recog.classEst = ndx;
else
   recog.classNameEst = train.NamaFile{ndx};
   recog.classEst = ndx;
end

recog.minDis = minDis;
```

```
function recog = cosinedist(recog, train, Wp, thress)
% klasifikasi dengan metode cosine similarity

[Nx Ny M] = size(train.Image);

% Inisialisasi input face
Mp = length(Wp(1,:));
X2 = double(reshape(recog.Image,[Nx*Ny 1]))./255; %input image dalam matrik
kolom
A2 = X2 - train.Mean;
% Proyeksi input face ke seluruh svm, mencari bobot masing2
% svm
recog.Wt = Wp'*A2;
% rekonstruksi input face
recog.reConstructed = Wp*recog.Wt + train.Mean;

Xinput = sqrt(recog.Wt'*recog.Wt);
Xtrain = sqrt(diag(train.Wt'*train.Wt));
penyebut = Xinput*Xtrain';
pembilang = recog.Wt'*train.Wt;

recog.Dist = -(pembilang./penyebut);

[min_dist, ndx] = min(recog.Dist);

if min_dist > thress
   recog.classNameEst = 'Tidak Dikenali';
   recog.classEst = ndx;
else
   recog.classNameEst = train.NamaFile{ndx};
   recog.classEst = ndx;
end

recog.minDis = min_dist;
```