LAMPIRAN A
KODE PROGRAM

```
  function recog
% Program utama

%================================%
% LAPLACIANFACE                  %
%================================%

clear
close all
clc

fprintf('\n1. Pilih Folder Train Database > ');
Train.Path = uigetdir('C:\');
if Train.Path==0
   return
end
fprintf('%s',Train.Path);

fprintf('\n2. Pilih Folder Test Database > ');
[file path]= uigetfile({'*.jpg','JPEG files (*.jpg)';'*.gif','GIF files
(*.gif)'},'*.pgm','PGM files (*.pgm)'; ... 'Input Image','C:\');

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

fprintf('\n3. Input Image ');
im = imread(TestImage);

T = CreateDatabase(TrainDatabasePath);
[m, A, Laplacianface] = Laplacianface(T);
OutputName = Recognition(TestImage, m, A, Laplacianface);

SelectedImage = strcat(TrainDatabasePath,'\',OutputName);
SelectedImage = imread(SelectedImage);

imshow(im)
title('Test Image');
figure,imshow(SelectedImage);
title('Equivalent Image');

str = strcat('Matched image is :  ',OutputName);
disp(str)
```

```
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 threshold 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 [m,A,PCA] = PCA(T)

% menghitung PCA
%================================================


[Wpca,train] = PCA(data,options);
X = double(reshape(train.Image, [irow*icol]))./255; % 1 kolom per wajah
m = mean(T,2)
A = [];
for i = 1 : Train_Number
   temp = double(T(:,i)) - m; % Menjumlahkan perbedaan image dari masing-masing training set  Ai = Ti - m
   A = [A temp]; % Menyatukan semua nilai tengah gambar
End

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)),irow*icol1);
Wpca = Wpca(:,1:Mp);
```

```
function [eigvector,eigvalue] = LPP(X, W, options)

% menghitung Laplacianfaces
%================================

[eigvector, eigvalue] = LPP(X, W, options);

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


D=diag(sum(W));
% L= D-W;
L=W;

DPrime=transpose(new_tou)*D*new_tou;
DPrime=(DPrime+transpose(DPrime))/2;
LPrime=transpose(new_tou)*L*new_tou;
LPrime=(LPrime+transpose(LPrime))/2;

% mencari eigen vektor dan nilai eigen
 [eigvector, LPPeigvalue] = eig(LPrime,DPrime);

% mengurutkan vektor eigen dengan nilai eigen terbesar
LPPeigvalue = diag(LPPeigvalue);
[junk, index] = sort(-LPPeigvalue);
LPPeigvalue = LPPeigvalue(index);
eigvector = eigvector(:,index);
LPPeigvalue = ones(length(LPPeigvalue),1) - LPPeigvalue;

LPPeigvector = PCAeigvector*eigvector;
Y = fea * LPPeigvector;                  % output Laplacianfaces
```

```
function [recog] = eudist(recog,train,W,thresh)

% Klasifikasi
%================================================

% mencari ukuran gambar, [eigenvector eigenvalue], dan banyaknya training
images, M

[eigvector, eigvalue] = LPP(X, W, options);

% Inisialisasi input face
Mp = length(Wp(1,:));
X2= double(reshape(recog.Image, [irow*icol] 1))./255; %input image dalam
matrik kolom
A2 = X2 - train.Mean;
% Proyeksi input face ke seluruh laplacianface, mencari bobot masing2
% laplacianface
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;
```

LAMPIRAN B
DATABASE WAJAH

abuy_1 abuy_2 abuy_3 abuy_4 abuy_5

adi_1 adi_2 adi_3 adi_4 adi_5

andris_1 andris_2 andris_3 andris_4 andris_5

cath_1 cath_2 cath_3 cath_4 cath_5

ari_1 ari_2 ari_3 ari_4 ari_5

david_1 david_2 david_3 david_4 david_5

dude_1 dude_2 dude_3 dude_4 dude_5

ferry_1 ferry_2 ferry_3 ferry_4 ferry_5

hesron_1 hesron_2 hesron_3 hesron_4 hesron_5

mike_1 mike_2 mike_3 mike_4 mike_5

heri_1 heri_2 heri_3 heri_4 heri_5


edi_1
edi_2
edi_3
edi_4
edi_5

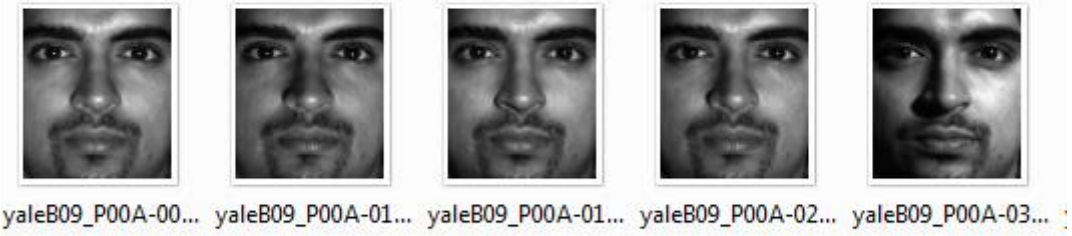

lutfi_1
lutfi_2
lutfi_3
lutfi_4
lutfi_5


widi_1
widi_2
widi_3
widi_4
widi_5


oki_1
oki_2
oki_3
oki_4
oki_5


yaleB03_P00A-01...
yaleB03_P00A-02...
yaleB03_P00A-05...
yaleB03_P00A-07...
yaleB03_P00A-11...


yaleB07_P00A-01...
yaleB07_P00A-05...
yaleB07_P00A-06...
yaleB07_P00A-07...
yaleB07_P00A-11...


yaleB09_P00A-00...
yaleB09_P00A-01...
yaleB09_P00A-01...
yaleB09_P00A-02...
yaleB09_P00A-03...


yaleB16_P00A-00...
yaleB16_P00A-01...
yaleB16_P00A-02...
yaleB16_P00A-02...
yaleB16_P00A-03...


yaleB19_P00A-00...
yaleB19_P00A-00...
yaleB19_P00A-01...
yaleB19_P00A-01...
yaleB19_P00A-01...


yaleB21_P00A-00...
yaleB21_P00A-01...
yaleB21_P00A-01...
yaleB21_P00A-01...
yaleB21_P00A-02...


yaleB04_P00A-01...
yaleB04_P00A-01...
yaleB04_P00A-02...
yaleB04_P00A-02...
yaleB04_P00A-03...
yaleB08_P00A-01...
yaleB08_P00A-03...
yaleB08_P00A-05...
yaleB08_P00A-11...
yaleB08_P00A+0...
yaleB10_P00A-00...
yaleB10_P00A-00...
yaleB10_P00A-01...
yaleB10_P00A-01...
yaleB10_P00A-01...
yaleB18_P00A-00...
yaleB18_P00A-02...
yaleB18_P00A-02...
yaleB18_P00A-02...
yaleB18_P00A-03...
yaleB20_P00A-00...
yaleB20_P00A-01...
yaleB20_P00A-01...
yaleB20_P00A-01...
yaleB20_P00A-02...
yaleB23_P00A-00...
yaleB23_P00A-01...
yaleB23_P00A-01...
yaleB23_P00A-01...
yaleB23_P00A-02...


yaleB25_P00A-00...
yaleB25_P00A-00...
yaleB25_P00A-01...
yaleB25_P00A-01...
yaleB25_P00A-01...
yaleB31_P00A-00...
yaleB31_P00A-08...
yaleB31_P00A-09...
yaleB31_P00A-11...
yaleB31_P00A-11...
/aleB26_P00A-00...
yaleB26_P00A-01...
yaleB26_P00A-01...
yaleB26_P00A-01...
yaleB26_P00A-02...