LAMPIRAN

PROGRAM MATLAB

PERCOBAAN 1 *FACE CLUSTERING* MENGGUNAKAN ALGORITMA *LOCAL SUBSPACE AFFINITY – SPECTRAL CLUSTERING*

```
%
%Program Utama 'Pengelompokan Citra Wajah dengan
Menerapkan Algoritma LSA - SC
%(LOCAL SUBSPACE AFFINITY - SPECTRAL CLUSTERING)'
%
% PERCOBAAN 1
%


%% Tutup semua jendela, hapus semua variabel, dan
bersihkan command windows
close all;
clear all;
clc;
```

DETEKSI WAJAH

```
jum_gbr = input('Total Gambar dalam database = ');

for m=1:jum_gbr
    % Proses membaca input
    indeks=num2str(m);
    ind=[indeks,'.jpg'];
    x=imread(ind);%figure;imshow(x);title('Gambar masukan
(RGB)');
    gbr_input{m}=x;

    %Proses face detection
    y=deteksi_face(x);
    gbr_det{m}=y;

    %Proses samakan ukuran
    z=samakan_ukuran(y);
    gbr_sama{m}=z;
end;
```

KONSTRUKSI *EIGENFACE*

```
%% Konstruksi Eigenface
[eigenface lamda vektor A
rata]=konstruksi_eigenface(gbr_sama);

ProjectedImage = [];
Train_Number = jum_gbr;
for i = 1 : Train_Number
    temp = eigenface'*A(:,i); % Projection of centered
images into facespace
    temp=temp./max(temp);   % normalisasi nilai
projection image
    ProjectedImage = [ProjectedImage temp];
end;
```

Function : *konstruksi_eigenface*

```
function [keluar1 keluar2 keluar3 keluar4
keluar5]=konstruksi_eigenface(masuk)
%
%
%Fungsi ini melakukan konstruksi eigenface
%
%variabel masukan : masuk = citra wajah yang sama
ukurannya
%
%variabel keluaran : keluar1 = eigenface
%                    keluar2 = nilai eigen
%                    keluar3 = vektor eigen matriks
kovarians
%                    keluar4 = matriks A
%                    keluar5 = vektor_rata
%

%Proses ubah citra menjadi vektor
for m=1:length(masuk)
    temp=masuk{m};
    temp=double(temp);
    temp=reshape(temp,size(temp,1)*size(temp,2),1);
    vektor_citra(:,m)=temp;
end
```

```
%Proses mencari rata-rata citra
vektor_rata=mean(vektor_citra,2);

for m=1:size(vektor_citra,2)
    citra_selisih(:,m)=vektor_citra(:,m)-vektor_rata;
end

A=citra_selisih;
mat_kov=A'*A;
[vektor lamda]=eig(mat_kov);

Eigenfaces = A * vektor;

keluar1=Eigenfaces;
keluar2=lamda;
keluar3=vektor;
keluar4=A;
keluar5=vektor_rata;
```

ALGORITMA *LOCAL SUBSPACE AFFINITY – SPECTRAL CLUSTERING*

```
%% Algoritma Local Subspace Affinity (LSA)

n=size(ProjectedImage,2)./3;
kneigh=6;
d=4;

[group,ranks]=lsa(ProjectedImage,n,kneigh,d);
```

Function : local_subspace_affinity

```
function [group,ranks]=local_subspace_affinity(x,n,k,d)
%
% Fungsi ini akan mencari local normal subspace dan
affinity matrix untuk
% proses clustering
%
%   Local Subspace Affinity (LSA)
%   Input :    x = data points
%              n = jumlah subspace (jumlah identitas
citra wajah)
%              d = dimensi subspace (default, atau
dimensi dari x - 1)
```

```
%              k = neighbour yang digunakan (default,
atau if k==0, k=d)
%   Output :   group = cluster yang terbentuk
%              ranks = rank dari vektor distance (dimensi
subspace)

% Mengetahui ukuran dari data points
[K,N]=size(x);

if(exist('d','var')==0)
    d=K-1;
end

if(exist('k','var')==0 || k==0)
    k=d;
end

% Melakukan Normalisasi terhadap data points
x = cnormalize(x);

% Menemukan neighbors
angles=acos(min(max(x'*x,-1),1));
[sorted,neighbours]=sort(angles);

% Memperkirakan local normal subspace
bases=zeros(size(x,1),d,N);
ranks=zeros(N,1);
for i=1:N
    [U,S,V]=svd(x(:,neighbours(1:k+1,i)));
    bases(:,:,i)=U(:,1:d);
    ranks(i)=d;
end

% Menghitung sudut (angle) antara dua subspace =>
similarity matrix
for j=1:N
     for i=j:N
         distance(j,i) =
subspaceaffinity(bases(:,:,j),bases(:,:,i));
         distance(i,j) = distance(j,i);
     end
end

% Melakukan spectral clustering untuk proses clustering
```

```
[diagMat,LMat,X,Y,group,errorsum]=spectralcluster(distanc
e,n,n);
```

Function : *cnormalize*

```
function [xn,normx] = cnormalize(x)
% Fungsi ini digunakan untuk melakukan normalisasi
terhadap input (dalam
% Tugas Akhir ini input = Projected Image hasil
konstruksi eigenface)
%
%   NORMALIZATION
%   Input :     x = data points (projected image)
%   Output :    xn    = vektor normal
%               normx = panjang vektor
%

% Menentukan ukuran dari data points
[K,N] = size(x);

% Mencari panjang vektor
normx = sqrt(sum(conj(x).*x,1));

% Normalisasi
xn =  x ./ (ones(K,1)*normx);
```

Function : *subspaceaffinity*

```
function d=subspaceaffinity(base1, base2)
%
% Fungsi ini akan menghitung pengukuran affinity
(affinity matrix) antara
% dua subspace yang direpresentasikan dengan basis
ORTHONORMAL (base1 dan
% base2)
%
%   Subspace Affinity
%   Input :  base1 = vektor normal subspace pertama
%            base2 = vektor normal subspace kedua
%   Output : d = jarak antara dua subspace ("distance")
%
```

```
% Menghitung sudut (angle) dari dua subspace
theta=subspaceangle(base1,base2);

% Menghitung jarak ("distance")
d=exp(-sum((sin(theta).^2)));
```

Function : *subspaceangle*

```
function [theta]=subspaceangle(base1, base2)
%
% Fungsi ini akan menghitung sudut (angle) principal
antara dua subspace
%
%   SUBSPACE ANGLE
%   Input :   base1 = basis ORTHONORMAL subspace pertama
%             base2 = basis ORTHONORMAL subspace kedua
%   Output :  theta = sudut (angle) dari kedua subspace
%
%   Contoh :
%   subspaceangle([1 0 0 0; 0 1 0 0]',orth([0 1 1 0; 0 0
0 1]'))
%
%   ans =
%
%       1.5708 ----->(90 degrees)
%       0.7854 ----->(45 degrees)
%

% Mencari SVD dari kedua basis ORTHONORMAL
[U,S,V]=svd(base1'*base2,'econ');

% Mencari cosinus sudut (angle) kedua basis ORTHONORMAL
costheta=flipud(svd(base2'*base1));

% Mencari sudut (angle)
theta=acos(min(1,costheta));
```

Function : *spectralcluster*

```
function [diagMat,LMat,X,Y,IDX,errorsum]=
spectralcluster(affMat,k,num_class)
%
```

```
% Fungsi ini akan melakukan proses clustering dengan
menggunakan algoritma
% spectral clustering
%
%   SPECTRAL CLUSTERING
%   Input :     affMat = affinity matrix
%                    k = jumlah eigenvektor terbesar
(pemotongan eigenvektor)
%            num_class = jumlah kelas yang terbentuk
%   Output :  diagmat = matrix diagonal
%             Lmat    = matrix L
%             X & Y   = matrix yang terbentuk dari
eigenvektor matrix L
%             IDX     = hasil clustering (group)
%            errorsum = jarak dari k - means
%

% Menentukan ukuran dari affinity matrix
numDT=size(affMat,2);

% Menghitung matrix diagonal dengan mencari vektor normal
diagMat= diag((sum(affMat,2)).^(-1./2));

% Mencari matrix L
LMat=diagMat*affMat*diagMat;

% Mencari eigenvektor dari matrix L
[v,d]=eigs(LMat,k);

% Normalisasi eigenvektor
X=v(:,1:k);
normX=(sum(X.^2,2)).^(1./2);
Y=zeros(size(X));
for index=1:numDT
    Y(index,:)=X(index,:)./normX(index,1);
end

% Proses clustering menggunakan algoritma k - means
[IDX,C,sumd] =
kmeans(Y,num_class,'EmptyAction','drop','Replicates',10);

errorsum=1;
```

PERCOBAAN 2 *FACE CLUSTERING* MENGGUNAKAN ALGORITMA *LOCAL SUBSPACE AFFINITY – SPECTRAL CLUSTERING*

```
%
%Program Utama 'Pengelompokan Citra Wajah dengan
Menerapkan Algoritma LSA - SC
%(LOCAL SUBSPACE AFFINITY - SPECTRAL CLUSTERING)'
%
% PERCOBAAN 2
%

%% Tutup semua jendela, hapus semua variabel, dan
bersihkan command windows
close all;
clear all;
clc;
```

DETEKSI WAJAH

```
jum_gbr = input('Total Gambar dalam database = ');

for m=1:jum_gbr
    % Proses membaca input
    indeks=num2str(m);
    ind=[indeks,'.jpg'];
    x=imread(ind);%figure;imshow(x);title('Gambar masukan
(RGB)');
    gbr_input{m}=x;

    %Proses face detection
    y=deteksi_face(x);
    gbr_det{m}=y;

    %Proses samakan ukuran
    z=samakan_ukuran(y);
    gbr_sama{m}=z;
end;
```

KONSTRUKSI *EIGENFACE*

```
%% Konstruksi Eigenface
[eigenface lamda vektor A
rata]=konstruksi_eigenface(gbr_sama);

ProjectedImage = [];
Train_Number = jum_gbr;
for i = 1 : Train_Number
    temp = eigenface'*A(:,i); % Projection of centered
images into facespace
    temp=temp./max(temp);   % normalisasi nilai
projection image
    ProjectedImage = [ProjectedImage temp];
end;
```

Function : *konstruksi_eigenface*

```
function [keluar1 keluar2 keluar3 keluar4
keluar5]=konstruksi_eigenface(masuk)
%
%
%Fungsi ini melakukan konstruksi eigenface
%
%variabel masukan : masuk = citra wajah yang sama
ukurannya
%
%variabel keluaran : keluar1 = eigenface
%                    keluar2 = nilai eigen
%                    keluar3 = vektor eigen matriks
kovarians
%                    keluar4 = matriks A
%                    keluar5 = vektor_rata
%

%Proses ubah citra menjadi vektor
for m=1:length(masuk)
    temp=masuk{m};
    temp=double(temp);
    temp=reshape(temp,size(temp,1)*size(temp,2),1);
    vektor_citra(:,m)=temp;
end
```

```
%Proses mencari rata-rata citra
vektor_rata=mean(vektor_citra,2);

for m=1:size(vektor_citra,2)
    citra_selisih(:,m)=vektor_citra(:,m)-vektor_rata;
end

A=citra_selisih;
mat_kov=A'*A;
[vektor lamda]=eig(mat_kov);

Eigenfaces = A * vektor;

keluar1=Eigenfaces;
keluar2=lamda;
keluar3=vektor;
keluar4=A;
keluar5=vektor_rata;
```

ALGORITMA *LOCAL SUBSPACE AFFINITY – SPECTRAL CLUSTERING*

```
%% Algoritma Local Subspace Affinity (LSA)

jum_identitas=input('Jumlah identitas wajah = ');

switch(jum_identitas)
    case 3
        n=3;
        kneigh=6;
        d=4;

[group,ranks]=local_subspace_affinity(ProjectedImage,n,kn
eigh,d);
    case 4
        n=4;
        kneigh=6;
        d=4;

[group,ranks]=local_subspace_affinity(ProjectedImage,n,kn
eigh,d);
    case 5
        n=5;
        kneigh=6;
        d=4;
```

```
[group,ranks]=local_subspace_affinity(ProjectedImage,n,kn
eigh,d);
end
```

Function : local_subspace_affinity

```
function [group,ranks]=local_subspace_affinity(x,n,k,d)
%
% Fungsi ini akan mencari local normal subspace dan
affinity matrix untuk
% proses clustering
%
%   Local Subspace Affinity (LSA)
%   Input :    x = data points
%              n = jumlah subspace (jumlah identitas
citra wajah)
%              d = dimensi subspace (default, atau
dimensi dari x - 1)
%              k = neighbour yang digunakan (default,
atau if k==0, k=d)
%   Output :   group = cluster yang terbentuk
%              ranks = rank dari vektor distance (dimensi
subspace)

% Mengetahui ukuran dari data points
[K,N]=size(x);

if(exist('d','var')==0)
    d=K-1;
end

if(exist('k','var')==0 || k==0)
    k=d;
end

% Melakukan Normalisasi terhadap data points
x = cnormalize(x);

% Menemukan neighbors
angles=acos(min(max(x'*x,-1),1));
[sorted,neighbours]=sort(angles);

% Memperkirakan local normal subspace
```

```
bases=zeros(size(x,1),d,N);
ranks=zeros(N,1);
for i=1:N
    [U,S,V]=svd(x(:,neighbours(1:k+1,i)));
    bases(:,:,i)=U(:,1:d);
    ranks(i)=d;
end

% Menghitung sudut (angle) antara dua subspace =>
similarity matrix
for j=1:N
     for i=j:N
         distance(j,i) =
subspaceaffinity(bases(:,:,j),bases(:,:,i));
         distance(i,j) = distance(j,i);
     end
end

% Melakukan spectral clustering untuk proses clustering
[diagMat,LMat,X,Y,group,errorsum]=spectralcluster(distanc
e,n,n);
```

Function : *cnormalize*

```
function [xn,normx] = cnormalize(x)
% Fungsi ini digunakan untuk melakukan normalisasi
terhadap input (dalam
% Tugas Akhir ini input = Projected Image hasil
konstruksi eigenface)
%
%   NORMALIZATION
%   Input :     x = data points (projected image)
%   Output :    xn    = vektor normal
%               normx = panjang vektor
%

% Menentukan ukuran dari data points
[K,N] = size(x);

% Mencari panjang vektor
normx = sqrt(sum(conj(x).*x,1));

% Normalisasi
xn =  x ./ (ones(K,1)*normx);
```

Function : *subspaceaffinity*

```
function d=subspaceaffinity(base1, base2)
%
% Fungsi ini akan menghitung pengukuran affinity
(affinity matrix) antara
% dua subspace yang direpresentasikan dengan basis
ORTHONORMAL (base1 dan
% base2)
%
%   Subspace Affinity
%   Input :  base1 = vektor normal subspace pertama
%            base2 = vektor normal subspace kedua
%   Output : d = jarak antara dua subspace ("distance")
%

% Menghitung sudut (angle) dari dua subspace
theta=subspaceangle(base1,base2);

% Menghitung jarak ("distance")
d=exp(-sum((sin(theta).^2)));
```

Function : *subspaceangle*

```
function [theta]=subspaceangle(base1, base2)
%
% Fungsi ini akan menghitung sudut (angle) principal
antara dua subspace
%
%   SUBSPACE ANGLE
%   Input :   base1 = basis ORTHONORMAL subspace pertama
%             base2 = basis ORTHONORMAL subspace kedua
%   Output :  theta = sudut (angle) dari kedua subspace
%
%   Contoh :
%   subspaceangle([1 0 0 0; 0 1 0 0]',orth([0 1 1 0; 0 0
0 1]'))
%
%   ans =
%
%       1.5708 ----->(90 degrees)
%       0.7854 ----->(45 degrees)
%
```

```
% Mencari SVD dari kedua basis ORTHONORMAL
[U,S,V]=svd(base1'*base2,'econ');

% Mencari cosinus sudut (angle) kedua basis ORTHONORMAL
costheta=flipud(svd(base2'*base1));

% Mencari sudut (angle)
theta=acos(min(1,costheta));
```

Function : *spectralcluster*

```
function [diagMat,LMat,X,Y,IDX,errorsum]=
spectralcluster(affMat,k,num_class)
%
% Fungsi ini akan melakukan proses clustering dengan
menggunakan algoritma
% spectral clustering
%
%   SPECTRAL CLUSTERING
%   Input :     affMat = affinity matrix
%                    k = jumlah eigenvektor terbesar
(pemotongan eigenvektor)
%            num_class = jumlah kelas yang terbentuk
%   Output :  diagmat = matrix diagonal
%             Lmat    = matrix L
%             X & Y   = matrix yang terbentuk dari
eigenvektor matrix L
%             IDX     = hasil clustering (group)
%            errorsum = jarak dari k - means
%

% Menentukan ukuran dari affinity matrix
numDT=size(affMat,2);

% Menghitung matrix diagonal dengan mencari vektor normal
diagMat= diag((sum(affMat,2)).^(-1./2));

% Mencari matrix L
LMat=diagMat*affMat*diagMat;

% Mencari eigenvektor dari matrix L
[v,d]=eigs(LMat,k);

% Normalisasi eigenvektor
```

```
X=v(:,1:k);
normX=(sum(X.^2,2)).^(1./2);
Y=zeros(size(X));
for index=1:numDT
    Y(index,:)=X(index,:)./normX(index,1);
end

% Proses clustering menggunakan algoritma k - means
[IDX,C,sumd] =
kmeans(Y,num_class,'EmptyAction','drop','Replicates',10);

errorsum=1;
```