function [XD YD] = TrainIt(DataType , dirname , dirno,TrainType)

XD=[];YD=[];

cpath = strcat(pwd,'\');

cpath = strcat(cpath,dirname);

traindir = num2str(dirno);

cpath = strcat(cpath,'\');

cpath = strcat(cpath,traindir);

if (DataType==1)

cpath =strcat(cpath,'\Train\');

else

cpath =strcat(cpath,'\Test\');

end

%disp(cpath);

list = dir(sprintf('%s\\\*.png', cpath));

% Process for each image

mycounter=0;

for index = 1:length(list)

mycounter=mycounter+1;

disp(mycounter);

%load image

fpathn = cpath;

fpathname = strcat(fpathn,list(index).name);

image = imread(fpathname);

GIm = double(rgb2gray(image));

FeatureForGIm=[];

%======================================================================

switch TrainType

case 1

%The Proposed\_Feature\_Extractor (CLBP image+Wavelet+Gabor12+SVD300)

FeatureForGIm = New\_Feature\_Extractor(image);

%FeatureForGIm = Proposed\_Feature\_Extractor(image);

%FeatureForGIm = SVD\_GABOR\_Feature\_Extractor(image);

case 2

%====== SVD Singular Value Decomposition Features =======

Segma = (double(svds(GIm,25)))' ;

FeatureForGIm =Segma;

%=========================================================

case 3

mapping=getmapping(8,'ri');%(8,'u2')%(8,'riu2');

%CLBP histogram in (8,1) neighborhood

[CLBP\_SH,CLBP\_MH]=clbp(image,1,8,mapping,'h');

FeatureForGIm =[CLBP\_SH CLBP\_MH];

case 4

FeatureForGIm =LDP(GIm);

case 5

% Calculate the gray-level co-occurrence matrix (GLCM) for the grayscale image. By default, |graycomatrix| calculates the GLCM based on horizontal

glcm = graycomatrix(GIm,'Offset',[2 0]);

r1=glcm(1,:);

r2=glcm(2,:);

FeatureForGIm = [r1 r2];

case 6

%=================Wavelet moment Features Extractors========

WavFeature = waveletTransform(image);

%====================SVD Features===========================

%Segma1 = double(svds(GIm,200));

%Segma2 = double(svds(GIm,100,'smallest'));

%Segma =[Segma1' Segma2'];

%=================CLBP========================================

mapping=getmapping(8,'u2');%(8,'ri')%(8,'u2')%(8,'riu2');

[CLBP\_SH,CLBP\_MH]=clbp(GIm,1,8,mapping,'h');

%================= Combined Features =========================

FeatureForGIm =[CLBP\_SH CLBP\_MH WavFeature];

%FeatureForGIm =[Segma WavFeature];

case 7

% Extract LBP features from the images to encode their texture information.

FeatureForGIm = extractLBPFeatures(GIm,'Upright',false);

case 8

%% histogram.

%J = histeq(I);

%figure,imshow(A)

%BQW= edge(A,'canny');

% h1 = imhist(GIm,64);

% FeatureForGIm = h1;

case 9

NMFImage= ReNMFImage( GIm,50 );

mapping=getmapping(8,'u2');

[CLBP\_SH,CLBP\_MH]=clbp(NMFImage,1,8,mapping,'h');

FeatureForGIm =[CLBP\_SH CLBP\_MH];

%glcm = graycomatrix(NMFImage,'Offset',[2 0]);

%r1=glcm(1,:);

%r2=glcm(2,:);

%FeatureForGIm = [r1 r2];

case 10

Segma1 = double(svds(GIm,200));

Segma2 = double(svds(GIm,100,'smallest'));

Segma =[Segma1' Segma2'];

mapping=getmapping(8,'u2');%(8,'ri')%(8,'u2')%(8,'riu2');

[CLBP\_SH,CLBP\_MH]=clbp(GIm,1,8,mapping,'h');

FeatureForGIm =[CLBP\_SH CLBP\_MH Segma];

%FeatureForGIm =Segma;

case 11

[U S V] = svd(GIm,'econ');

S= diag(S);

Segma =S(:);

Segma1=Segma(1:150,:);

Segma2=Segma(250:300,:);

Segma =[Segma1' Segma2'];

mapping=getmapping(8,'u2');

[CLBP\_SH,CLBP\_MH]=clbp(GIm,1,8,mapping,'h');

FeatureForGIm =[CLBP\_SH CLBP\_MH Segma];

case 12

myfeatures = (double(svds(GIm,400)))' ;

mapping=getmapping(8,'u2');

[CLBP\_SH,CLBP\_MH]=clbp(GIm,1,8,mapping,'h');

FeatureForGIm =[CLBP\_SH CLBP\_MH myfeatures];

case 13

% Generates the Gabor filter bank

gaborArray = gaborFilterBank(5,8,39,39);

% Extracts Gabor feature vector, 'featureVector', from the image

featureVector = gaborFeatures(GIm,gaborArray,4,4);

ndim=300;

[residuals,reconstructed] = pcares(featureVector ,ndim);

%The 'reconstructed' will have the reduced dimensions data based on

%ndims input. Note that 'reconstructed' still the original dimension.

%You can choose the first ndims if you'd like.

%If you want the reduced dimensions in the new basis then

%just take the first ndims of the SCORE variable

%SCORE(:,1:ndims);

%[COEFF,SCORE] = princomp(reconstructed);

%SCORE

FeatureForGIm= reconstructed(:,1:ndim)

case 14

[U S V] = svd(GIm,'econ');

S= diag(S);

Segma =S(:);

Segma=Segma(1:200,:);

Segma =Segma';

% Apply Gabor filter to image.

wavelength = 4;

orientation = 90;

[mag,phase] = imgaborfilt(GIm,wavelength,orientation);

mapping=getmapping(8,'u2');

[CLBP\_SH,CLBP\_MH]=clbp(GIm,1,8,mapping,'h');

FeatureForGIm =[CLBP\_SH CLBP\_MH];

mapping=getmapping(8,'u2');

[CLBP\_SH1,CLBP\_MH1]=clbp(phase,1,8,mapping,'h');

mapping=getmapping(8,'u2');

[CLBP\_SH2,CLBP\_MH2]=clbp(mag,1,8,mapping,'h');

FeatureForGIm =[FeatureForGIm CLBP\_SH1 CLBP\_MH1 CLBP\_SH2

CLBP\_MH2 Segma];

case 15

%=============================New GABOR===================

mySegma = (double(svds(GIm,200)))' ;

% Create array of Gabor filters, called a \_filter bank\_.

%This filter bank contains two orientations and two wavelengths.

gaborArray = gabor([2],[0 30 60 90 120 150]);

% Apply filters to input image.

gaborMag = imgaborfilt(GIm,gaborArray);

[r c]= size(gaborMag(:,:,1));

allgaborMag=zeros(r,c);

for p = 1:6

%theta = gaborArray(p).Orientation;

%lambda = gaborArray(p).Wavelength %title(sprintf('Orientation=%d,Wavelength=%d',theta,lambda));

allgaborMag= allgaborMag+ gaborMag(:,:,p);

%Resultant edges of the Image

end

mapping=getmapping(8,'u2');

[CLBP\_SHGE,CLBP\_ME]=clbp(allgaborMag,1,8,mapping,'h');

FeatureForGIm=[CLBP\_SHGE,CLBP\_ME,mySegma];

case 16

FeatureForGIm = extractLBPFeatures(GIm,'Upright',true);

FeatureForGIm = detectFASTFeatures(GIm,'MinContrast',0.1);

detectBRISKFeatures(GIm)

FeatureForGIm=double(FeatureForGIm);

level = graythresh(GIm);

BW = im2bw(GIm,level);

FeatureForGIm=mean(BW,1);

otherwise

warning('Unexpected function type.');

end

%===================== END CASE =========================

%%

XD=[XD;FeatureForGIm];

YD=[YD;dirno];

end

end

%================= END of the Trainit Function ===============

%%

%================= SVD Compression ===================

%a=imread(query\_fullpath);

%imshow(a)

%a=rgb2gray(a);

%hostimage = im2double(a);

%figure,imshow(hostimage)

% determine size of the host image

%[Mc Nc]=size(hostimage); %image dimension Width Height

% Compression Factor # of Sigma from dig matrix Sh?????????????

%k=50;

%%SVD Matrix Factorization

%[Uh,Sh,Vh]= svd(double(hostimage(1:Mc,1:Nc)));

%%Compression process

%VhT=transpose(Vh(1:Nc,1:k));

%compressed\_image= Uh(1:Mc,1:k)\*Sh(1:k,1:k)\*VhT;

%======================================================================