```
% feature extraction
image1 = getappdata(handles.figure1,'OriginalImage');
%%
%===== crop by [xmin ymin width height]
load coords;

rect = [ceil(min(xs)) ceil(min(ys)) ceil(min(xs))+100 ceil(min(ys))+100];
image2 = imcrop(image1, rect);

figure;
imshow(image2);axis tight;
title('Segmented image', 'fontSize', 14);

%% Histogram normalization
segImg = histeq(image2);
%segImg = adapthisteq(image2);
figure, imshow(segImg); axis tight;
%figure, imhist(segImg,64);  axis tight;
title('Histogram normalization', 'fontSize', 14);

%size(IMG_features)

%%

%try
    if isequal(get(handles.optTest,'value'),1)

        % Local Binary Pattern image
        imageLBP = pixelwiseLBP(segImg);

        figure, imshow(imageLBP); axis tight;
        title('LBP image', 'fontSize', 14);

        dataLBP = reshape(imageLBP,[],1);
        dataLBP = sort(dataLBP,'descend')
%ratio = ceil(0.5*length(dataLBP)); % 10% of features (biggest coefficient)
        dataLBP = dataLBP(1:4000);

        % Curvelet Transform (scale 3 decomposition)
        nbscales = 3;   %number of scales
        nbangles = 8;   %number of angles

        C = fdct_wrapping(double(imageLBP),[],[],nbscales,nbangles);

        % Arrange features

        IMG_features = [];

        % scale level 1
        datac = reshape(C{1}{1},[],1);
        datac = sort(datac,'descend');
      ratio = ceil(0.1*length(datac)); % 10% of features (biggest coefficient)
        IMG_features = [IMG_features; datac(1:ratio)];
```

```
    % scale level 2
    datac = reshape(C{2}{1},[],1);
    datac = sort(datac,'descend');
  ratio = ceil(0.1*length(datac)); % 10% of features (biggest coefficient)
    IMG_features = [IMG_features; datac(1:ratio)];

    % scale level 3
    datac = reshape(C{3}{1},[],1);
    datac = sort(datac,'descend');
  ratio = ceil(0.1*length(datac)); % 10% of features (biggest coefficient)
    IMG_features = [IMG_features; datac(1:ratio)];

    IMG_features=IMG_features(1:4000);

    load 'CTLBPclassifier';
    IMG_features=abs(IMG_features);
    predClass = predict(KNNmdl,IMG_features.'); % predicted class
    msgbox(predClass);

elseif isequal(get(handles.optTestC,'value'),1)
    %% Curvelet Transform (scale 3 decomposition)
    nbscales = 3;   %number of scales
    nbangles = 8;   %number of angles

    C = fdct_wrapping(double(segImg),[],[],nbscales,nbangles);

    IMG_features = [];

    % scale level 1
    datac = reshape(C{1}{1},[],1);
    datac = sort(datac,'descend');
  ratio = ceil(0.1*length(datac));% 10% of features (biggest coefficient)
    IMG_features = [IMG_features; datac(1:ratio)];

    % scale level 2
    datac = reshape(C{2}{1},[],1);
    datac = sort(datac,'descend');
  ratio = ceil(0.1*length(datac)); % 10% of features (biggest coefficient)
    IMG_features = [IMG_features; datac(1:ratio)];

    % scale level 3
    datac = reshape(C{3}{1},[],1);
    datac = sort(datac,'descend');
  ratio = ceil(0.1*length(datac)); % 10% of features (biggest coefficient)
    IMG_features = [IMG_features; datac(1:ratio)];

    IMG_features=IMG_features(1:4000);

    load 'CTclassifier';
    IMG_features=abs(IMG_features);
    predClass = predict(KNNmdl,IMG_features.'); % predicted class
    msgbox(predClass);
```

```
    else
        %% Local Binary Pattern image

        imageLBP = pixelwiseLBP(segImg);

        figure, imshow(imageLBP); axis tight;
        title('LBP image', 'fontSize', 14);

        dataLBP = reshape(imageLBP,[],1);
        dataLBP = sort(dataLBP,'descend');

%ratio = ceil(0.5*length(dataLBP)); % 10% of features (biggest coefficient)
        dataLBP = dataLBP(1:4000);
        load 'LBPclassifier';
        predClass = predict(KNNmdl,dataLBP.'); % predicted class
        msgbox(predClass);
    end

%catch ME
 %   msgbox('Unknown feature!');
%end
```

TWO-STAGE FEATURE EXTRACTION

```
clear all;
close all;
clc;

%% Load training images

imagefile1 = 'mdb006'; % normal, malignant, benign

image1 = imread(imagefile1,'jpg');

if ndims(image1) == 3;
    image1 = rgb2gray(image1);
end  % Color Images

%%

%===== crop by 1021x616
%([xmin ymin width height]);
rect = [206 374 (206+128) (374+128)];
image2 = imcrop(image1, rect);

imshow(image2);axis tight;
title('Original image', 'fontSize', 14);


%% Histogram normalization
segImg = histeq(image2);
%segImg = adapthisteq(image2);
figure, imshow(segImg); axis tight;
%figure, imhist(segImg,64);  axis tight;

%%
%W=0.4;
%segImg=fftenhance(image2,W);

%figure, imshow(segImg); axis tight;

%% Local Binary Pattern image

imageLBP = pixelwiseLBP(segImg);

figure, imshow(imageLBP); axis tight;

dataLBP = reshape(imageLBP,[],1);
dataLBP = sort(dataLBP,'descend');
ratio = ceil(0.5*length(dataLBP)); % 10% of features (biggest coefficient)
dataLBP = dataLBP(1:ratio);

%% Curvelet Transform (scale 3 decomposition)

C = fdct_wrapping(double(imageLBP),[],1,3);
```

```
%% Arrange features

IMG_features = [];

% scale level 1
datac = reshape(C{1}{1},[],1);
datac = sort(datac,'descend');
ratio = ceil(0.1*length(datac)); % 10% of features (biggest coefficient)
IMG_features = [IMG_features; datac(1:ratio)];

% scale level 2
datac = reshape(C{2}{1},[],1);
datac = sort(datac,'descend');
ratio = ceil(0.1*length(datac)); % 10% of features (biggest coefficient)
IMG_features = [IMG_features; datac(1:ratio)];

% scale level 3
datac = reshape(C{3}{1},[],1);
datac = sort(datac,'descend');
ratio = ceil(0.1*length(datac)); % 10% of features (biggest coefficient)
IMG_features = [IMG_features; datac(1:ratio)];
```

```
disp('Testing with CT+LBP');
disp(' ');

%% Separate features for

%TestFeats = double(ctlbpFeats(:,TestId)).'; % features of testing samples
%TestLabels = Indx(TestId); % get testing feature's class

%TrainFeats = double(ctlbpFeats(:,TrainId)).'; % features of training samples
%TrainLabels = Indx(TrainId); % get training feature's class

TestId = reshape(TestId,[],1); TestId=[TestId; TestId];
TrainId = reshape(TrainId,[],1);
TestFeats = double(ctlbpFeats(:,TestId)).'; % features of testing samples
TestLabels = Indx(TestId); % get testing feature's class
TrainFeats = double(ctlbpFeats(:,TrainId)).'; % features of training samples
TrainLabels = Indx(TrainId); % get training feature's class


%% Construct a KNN classifier for the Curvelet Transform
K = 10; % K nearest neighbors

X = TrainFeats; % training set

Y=cell(length(TrainId),1);
for i=1:length(TrainId)
    Y{i} = Cancer{TrainId(i)}; % class
end

% create ExhaustiveSearcher KNN model object ('cosine','hamming','jaccard')
%KNNmdl =
fitcknn(X,Y,'NumNeighbors',K,'NSMethod','exhaustive','Distance','hamming');

%create KDTreeSearcher KNN model object ('euclidean' (default), 'cityblock',
'minkowski', 'chebychev')
KNNmdl =
fitcknn(X,Y,'NumNeighbors',K,'NSMethod','kdtree','Distance','minkowski');

trnTime=cputime-trnTime;

save 'CTLBPclassifier' KNNmdl

%% Classify Using k-Nearest Neighbours

load 'CTLBPclassifier';

%acanc = mean(X); % an average cancer

CompareP=cell(length(TestLabels),1);
CompareA=cell(length(TestLabels),2);

tstTime=cputime;

for i=1:length(TestLabels)

    predClass = predict(KNNmdl,TestFeats(i,:)); % predicted class
    CompareP{i} = predClass;
```

```
end
tstTime=cputime-tstTime;

gNormal=0;
for i=1:length(TestLabels)
    actClass = Cancer{TestId(i)}; % actual class
    CompareA{i} = actClass;
    if strcmp(actClass,'Normal')
       gNormal=gNormal+1;
   end
end

fprintf('%s  %s\n'  ,'   Predicted', '   Actual   ');
fprintf('%s  %s\n'  ,'   ---------', ' ---------- ');
for i=1:length(TestLabels)
    disp([CompareP{i} CompareA{i}]);

end

%% performance measures
%TN=>normal classified as normal
%FN=>normal classified as abnormal
%TP=>abnormal classified as abnormal
%FP=>abnormal classified as normal

TP=0; FP=0; TN=0; FN=0;

for i=1:length(TestLabels)

    if strcmp(CompareP{i},'Normal')&&strcmp(CompareA{i},'Normal')
        TN=TN+1;
    elseif
(strcmp(CompareP{i},'Malignant')||strcmp(CompareP{i},'Benign'))&&strcmp(Compar
eA{i},'Normal')
        FN=FN+1;
    elseif
(strcmp(CompareP{i},'Malignant')||strcmp(CompareP{i},'Benign'))&&(strcmp(Compa
reA{i},'Malignant')||strcmp(CompareA{i},'Benign'))
        TP=TP+1;
    elseif
strcmp(CompareP{i},'Normal')&&(strcmp(CompareA{i},'Malignant')||strcmp(Compare
A{i},'Benign'))
        FP=FP+1;
    end
end

sensitivity = (TP/(TP+FP))*100;

specificity = (TN/(TN+FP))*100;

Accuracy = ((TP+TN)/(TP+TN+FP+FN))*100;



FNMfB=0; FPBfM=0; TNMfB=0; TPBfM=0;

for i=1:length(TestLabels)
```

```
   if strcmp(CompareP{i},'Benign')&&strcmp(CompareA{i},'Malignant')
        FNMfB = FNMfB+1;
    elseif strcmp(CompareP{i},'Malignant')&&strcmp(CompareA{i},'Benign')
        FPBfM = FPBfM+1;
    elseif strcmp(CompareP{i},'Malignant')&&strcmp(CompareA{i},'Malignant')
        TNMfB = TNMfB+1;
    elseif strcmp(CompareP{i},'Benign')&&strcmp(CompareA{i},'Benign')
        TPBfM = TPBfM+1;
    end

end

sensitivity2 = (TPBfM/(TPBfM+FPBfM))*100;

specificity2 = (TNMfB/(TNMfB+FPBfM))*100;

Accuracy2 = ((TPBfM+TNMfB)/(TPBfM+TNMfB+FPBfM+FNMfB))*100;

%%
disp('========================================================= ');
fprintf('Feature extraction method           : %s\n', 'CT+LBP');
fprintf('Total trained breast images         : %d\n', length(TrainId));
fprintf('Total tested breast images          : %d\n', TP+TN+FP+FN);
fprintf('Total testing time (secs)           : %g\n', tstTime);
fprintf('Total training time (secs)          : %g\n', trnTime);
disp(' ');
disp('        Classification between Normal/Abnormal             ');
disp('========================================================= ');
fprintf('Normal classified as normal (TN)    : %d\n', TN);
fprintf('Normal classified as abnormal (FN)  : %d\n', FN);
fprintf('Abnormal classified as abnormal (TP): %d\n', TP);
fprintf('Abnormal classified as normal (FP)  : %d\n', FP);
fprintf('Sensitivity                         : %g %s\n',sensitivity,'%');
fprintf('specificity                         : %g %s\n',specificity,'%');
fprintf('Accuracy                            : %g %s\n',Accuracy,'%');
disp(' ');
disp('        Classification between Malignant/Benign            ');
disp('========================================================= ');
fprintf('Malignant classified as Malignant (TN): %d\n', TNMfB);
fprintf('Malignant classified as Benign (FN)   : %d\n', FNMfB);
fprintf('Benign classified as Benign (TP)      : %d\n', TPBfM);
fprintf('Benign classified as Malignant (FP)   : %d\n', FPBfM);
fprintf('Sensitivity                           : %g %s\n',sensitivity2,'%');
fprintf('specificity                           : %g %s\n',specificity2,'%');
fprintf('Accuracy                              : %g %s\n',Accuracy2,'%');

%%
save 'Result_CTLBP' TN FN TP FP TNMfB FNMfB TPBfM FPBfM sensitivity
specificity Accuracy tstTime trnTime gNormal sensitivity2 specificity2
Accuracy2
```