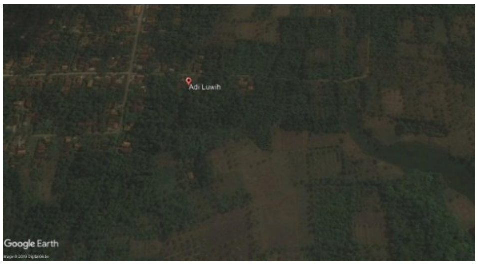
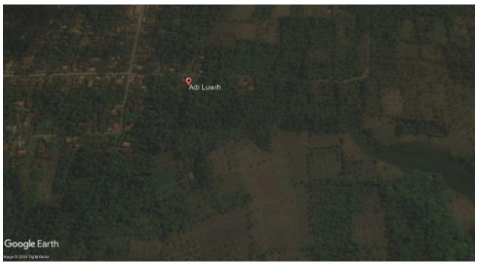
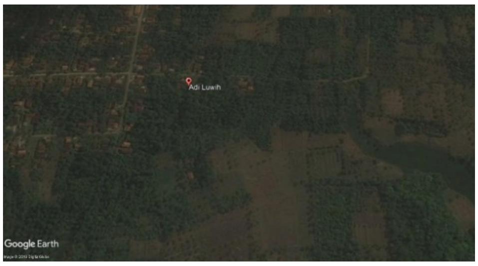
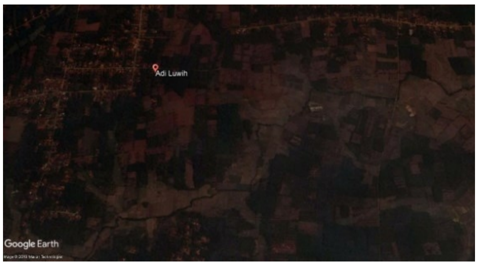
LAMPIRAN

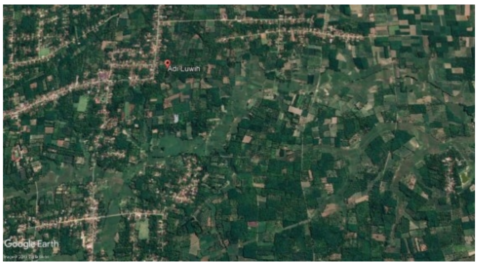
➢ Kecamatan Adiluwih

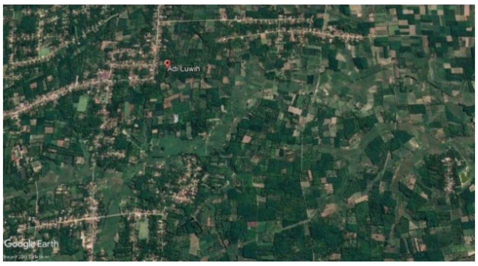




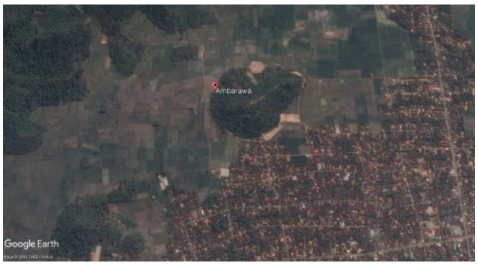


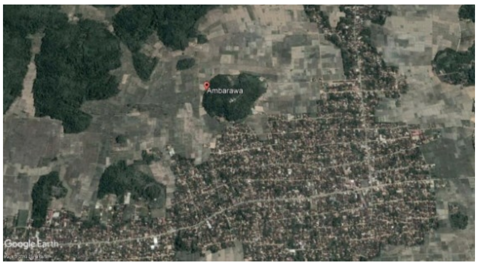


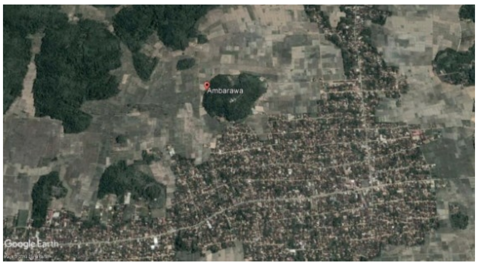




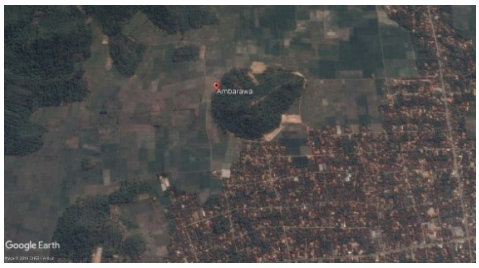
- **Kecamatan Ambarawa**

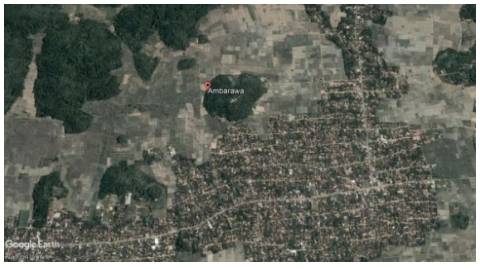


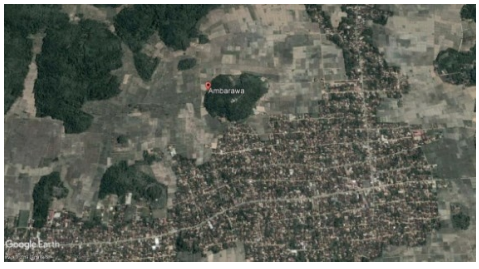




➢ Kecamatan Banyumas



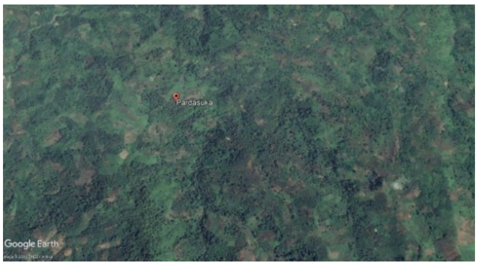


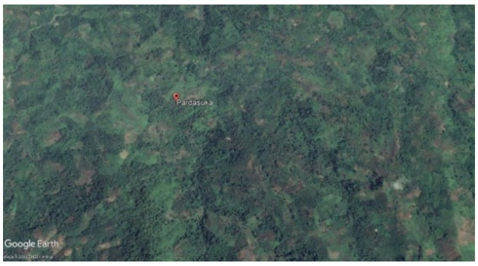


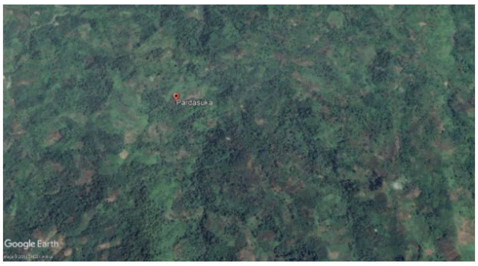
➢ Kecamatan Gading Rejo

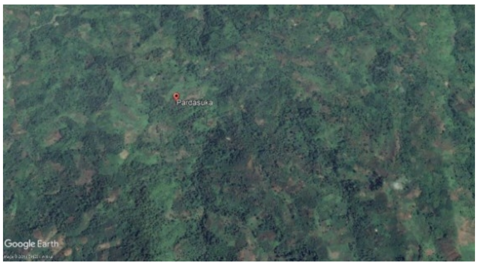
- **Kecamatan Pagelaran**

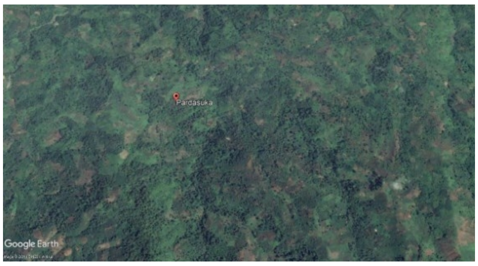
- **Kecamatan Pardasuka**

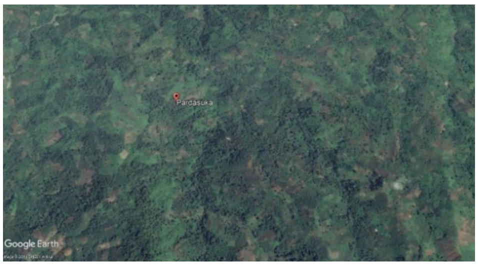






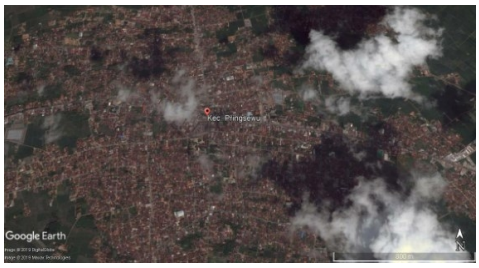






- **Kecamatan Pringsewu**

2014 **2015**



➢ Kecamatan Sukoharjo

Listing Program

```
      function varargout = cbires(varargin)
% CBIRES MATLAB code for cbires.fig
%      CBIRES, by itself, creates a new CBIRES or raises the
existing
%      singleton*.
%
%      H = CBIRES returns the handle to a new CBIRES or the handle
to
%      the existing singleton*.
%
%      CBIRES('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in CBIRES.M with the given input
arguments.
%
%      CBIRES('Property','Value',...) creates a new CBIRES or
raises the
%      existing singleton*.  Starting from the left, property
value pairs are
%      applied to the GUI before cbires_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to cbires_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help cbires

% Last Modified by GUIDE v2.5 23-May-2013 22:01:15

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
    'gui_Singleton',  gui_Singleton, ...
    'gui_OpeningFcn', @cbires_OpeningFcn, ...
    'gui_OutputFcn',  @cbires_OutputFcn, ...
    'gui_LayoutFcn',  [] , ...
    'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
```

```
end
% End initialization code - DO NOT EDIT


% --- Executes just before cbires is made visible.
function cbires_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to cbires (see VARARGIN)

% Choose default command line output for cbires
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes cbires wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = cbires_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in btn_BrowseImage.
function btn_BrowseImage_Callback(hObject, eventdata, handles)
% hObject    handle to btn_BrowseImage (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

[query_fname, query_pathname] = uigetfile('*.jpg; *.png; *.bmp',
'Select query image');

if (query_fname ~= 0)
    query_fullpath = strcat(query_pathname, query_fname);
    [pathstr, name, ext] = fileparts(query_fullpath); % fiparts
returns char type

    if ( strcmp(lower(ext), '.jpg') == 1 || strcmp(lower(ext),
'.png') == 1 ...
            || strcmp(lower(ext), '.bmp') == 1 )
```

```
        queryImage = imread( fullfile( pathstr, strcat(name,
ext) ) );
%         handles.queryImage = queryImage;
%         guidata(hObject, handles);

        % extract query image features
        queryImage = imresize(queryImage, [384 256]);
        hsvHist = hsvHistogram(queryImage);
        autoCorrelogram = colorAutoCorrelogram(queryImage);
        color_moments = colorMoments(queryImage);
        % for gabor filters we need gary scale image
        img = double(rgb2gray(queryImage))/255;
        [meanAmplitude, msEnergy] = gaborWavelet(img, 4, 6); % 4 =
number of scales, 6 = number of orientations
        wavelet_moments = waveletTransform(queryImage);
        % construct the queryImage feature vector
        queryImageFeature = [hsvHist autoCorrelogram color_moments
meanAmplitude msEnergy wavelet_moments str2num(name)];

        % update handles
        handles.queryImageFeature = queryImageFeature;
        guidata(hObject, handles);
        helpdlg('Proceed with the query by executing the green
button!');

        % Clear workspace
        clear('query_fname', 'query_pathname', 'query_fullpath',
'pathstr', ...
            'name', 'ext', 'queryImage', 'hsvHist',
'autoCorrelogram', ...
            'color_moments', 'img', 'meanAmplitude', 'msEnergy',
...
            'wavelet_moments', 'queryImageFeature');
    else
        errordlg('You have not selected the correct file type');
    end
else
    return;
end


% --- Executes on selection change in popupmenu_DistanceFunctions.
function popupmenu_DistanceFunctions_Callback(hObject, eventdata,
handles)
% hObject    handle to popupmenu_DistanceFunctions (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns
popupmenu_DistanceFunctions contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
popupmenu_DistanceFunctions
```

```
handles.DistanceFunctions =
get(handles.popupmenu_DistanceFunctions, 'Value');
guidata(hObject, handles);


% --- Executes during object creation, after setting all
properties.
function popupmenu_DistanceFunctions_CreateFcn(hObject, eventdata,
handles)
% hObject    handle to popupmenu_DistanceFunctions (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: popupmenu controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in
popupmenu_NumOfReturnedImages.
function popupmenu_NumOfReturnedImages_Callback(hObject,
eventdata, handles)
% hObject    handle to popupmenu_NumOfReturnedImages (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns
popupmenu_NumOfReturnedImages contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
popupmenu_NumOfReturnedImages

handles.numOfReturnedImages =
get(handles.popupmenu_NumOfReturnedImages, 'Value');
guidata(hObject, handles);


% --- Executes during object creation, after setting all
properties.
function popupmenu_NumOfReturnedImages_CreateFcn(hObject,
eventdata, handles)
% hObject    handle to popupmenu_NumOfReturnedImages (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called
```

```
% Hint: popupmenu controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in btnExecuteQuery.
function btnExecuteQuery_Callback(hObject, eventdata, handles)
% hObject    handle to btnExecuteQuery (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% check for image query
if (~isfield(handles, 'queryImageFeature'))
    errordlg('Please select an image first, then choose your
similarity metric and num of returned images!');
    return;
end

% check for dataset existence
if (~isfield(handles, 'imageDataset'))
    errordlg('Please load a dataset first. If you dont have one
then you should consider creating one!');
    return;
end

% set variables
if (~isfield(handles, 'DistanceFunctions') && ~isfield(handles,
'numOfReturnedImages'))
    metric = get(handles.popupmenu_DistanceFunctions, 'Value');
    numOfReturnedImgs = get(handles.popupmenu_NumOfReturnedImages,
'Value');
elseif (~isfield(handles, 'DistanceFunctions') ||
~isfield(handles, 'numOfReturnedImages'))
    if (~isfield(handles, 'DistanceFunctions'))
        metric = get(handles.popupmenu_DistanceFunctions,
'Value');
        numOfReturnedImgs = handles.numOfReturnedImages;
    else
        metric = handles.DistanceFunctions;
        numOfReturnedImgs =
get(handles.popupmenu_NumOfReturnedImages, 'Value');
    end
else
    metric = handles.DistanceFunctions;
    numOfReturnedImgs = handles.numOfReturnedImages;
end

if (metric == 1)
    L1(numOfReturnedImgs, handles.queryImageFeature,
handles.imageDataset.dataset);
```

```
elseif (metric == 2 || metric == 3 || metric == 4 || metric == 5
|| metric == 6  || metric == 7 || metric == 8 || metric == 9 ||
metric == 10 || metric == 11)
    L2(numOfReturnedImgs, handles.queryImageFeature,
handles.imageDataset.dataset, metric);
else
    relativeDeviation(numOfReturnedImgs,
handles.queryImageFeature, handles.imageDataset.dataset);
end


% --- Executes on button press in btnExecuteSVM.
function btnExecuteSVM_Callback(hObject, eventdata, handles)
% hObject    handle to btnExecuteSVM (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% check for image query
if (~isfield(handles, 'queryImageFeature'))
    errordlg('Please select an image first!');
    return;
end

% check for dataset existence
if (~isfield(handles, 'imageDataset'))
    errordlg('Please load a dataset first. If you dont have one
then you should consider creating one!');
    return;
end

numOfReturnedImgs = get(handles.popupmenu_NumOfReturnedImages,
'Value');
metric = get(handles.popupmenu_DistanceFunctions, 'Value');

% call svm function passing as parameters the numOfReturnedImgs,
queryImage and the dataset
[~, ~, cmat] = svm(numOfReturnedImgs,
handles.imageDataset.dataset, handles.queryImageFeature, metric);

% plot confusion matrix
opt = confMatPlot('defaultOpt');
opt.className = {
    'Africa', 'Beach', 'Monuments', ...
    'Buses', 'Dinosaurs', 'Elephants', ...
    'Flowers', 'Horses', 'Mountains', ...
    'Food'
    };
opt.mode = 'both';
figure('Name', 'Confusion Matrix');
confMatPlot(cmat, opt);
xlabel('Confusion Matrix');
```

```
% --- Executes on button press in btnPlotPrecisionRecall.
function btnPlotPrecisionRecall_Callback(hObject, eventdata,
handles)
% hObject    handle to btnPlotPrecisionRecall (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

if (~isfield(handles, 'imageDataset'))
    errordlg('Please select a dataset first!');
    return;
end

% set variables
numOfReturnedImgs = 20;
database = handles.imageDataset.dataset;
metric =  get(handles.popupmenu_DistanceFunctions, 'Value');

precAndRecall = zeros(2, 10);

for k = 1:15
    randImgName = randi([0 999], 1);
    randStrName = int2str(randImgName);
    randStrName = strcat('images\', randStrName, '.jpg');
    randQueryImg = imread(randStrName);

    % extract query image features
    queryImage = imresize(randQueryImg, [384 256]);
    hsvHist = hsvHistogram(queryImage);
    autoCorrelogram = colorAutoCorrelogram(queryImage);
    color_moments = colorMoments(queryImage);
    % for gabor filters we need gary scale image
    img = double(rgb2gray(queryImage))/255;
    [meanAmplitude, msEnergy] = gaborWavelet(img, 4, 6); % 4 =
number of scales, 6 = number of orientations
    wavelet_moments = waveletTransform(queryImage);
    % construct the queryImage feature vector
    queryImageFeature = [hsvHist autoCorrelogram color_moments
meanAmplitude msEnergy wavelet_moments randImgName];

    disp(['Random Image = ', num2str(randImgName), '.jpg']);
    [precision, recall] = svm(numOfReturnedImgs, database,
queryImageFeature, metric);
    precAndRecall(1, k) = precision;
    precAndRecall(2, k) = recall;
end

figure;
plot(precAndRecall(2, :), precAndRecall(1, :), '--mo');
xlabel('Recall'), ylabel('Precision');
title('Precision and Recall');
legend('Recall & Precision', 'Location', 'NorthWest');
```

```
% --- Executes on button press in btnSelectImageDirectory.
function btnSelectImageDirectory_Callback(hObject, eventdata,
handles)
% hObject    handle to btnSelectImageDirectory (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% select image directory
folder_name = uigetdir(pwd, 'Select the directory of images');
if ( folder_name ~= 0 )
    handles.folder_name = folder_name;
    guidata(hObject, handles);
else
    return;
end


% --- Executes on button press in btnCreateDB.
function btnCreateDB_Callback(hObject, eventdata, handles)
% hObject    handle to btnCreateDB (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

if (~isfield(handles, 'folder_name'))
    errordlg('Please select an image directory first!');
    return;
end

% construct folder name foreach image type
pngImagesDir = fullfile(handles.folder_name, '*.png');
jpgImagesDir = fullfile(handles.folder_name, '*.jpg');
bmpImagesDir = fullfile(handles.folder_name, '*.bmp');

% calculate total number of images
num_of_png_images = numel( dir(pngImagesDir) );
num_of_jpg_images = numel( dir(jpgImagesDir) );
num_of_bmp_images = numel( dir(bmpImagesDir) );
totalImages = num_of_png_images + num_of_jpg_images +
num_of_bmp_images;

jpg_files = dir(jpgImagesDir);
png_files = dir(pngImagesDir);
bmp_files = dir(bmpImagesDir);

if ( ~isempty( jpg_files ) || ~isempty( png_files ) ||
~isempty( bmp_files ) )
    % read jpg images from stored folder name
    % directory and construct the feature dataset
    jpg_counter = 0;
    png_counter = 0;
    bmp_counter = 0;
    for k = 1:totalImages
```

```
        if ( (num_of_jpg_images - jpg_counter) > 0)
            imgInfoJPG = imfinfo( fullfile( handles.folder_name,
jpg_files(jpg_counter+1).name ) );
            if ( strcmp( lower(imgInfoJPG.Format), 'jpg') == 1 )
                % read images
                sprintf('%s \n', jpg_files(jpg_counter+1).name)
                % extract features
                image = imread( fullfile( handles.folder_name,
jpg_files(jpg_counter+1).name ) );
                [pathstr, name, ext] =
fileparts( fullfile( handles.folder_name,
jpg_files(jpg_counter+1).name ) );
                image = imresize(image, [384 256]);
            end

            jpg_counter = jpg_counter + 1;

        elseif ( (num_of_png_images - png_counter) > 0)
            imgInfoPNG = imfinfo( fullfile( handles.folder_name,
png_files(png_counter+1).name ) );
            if ( strcmp( lower(imgInfoPNG.Format), 'png') == 1 )
                % read images
                sprintf('%s \n', png_files(png_counter+1).name)
                % extract features
                image = imread( fullfile( handles.folder_name,
png_files(png_counter+1).name ) );
                [pathstr, name, ext] =
fileparts( fullfile( handles.folder_name,
png_files(png_counter+1).name ) );
                image = imresize(image, [384 256]);
            end

            png_counter = png_counter + 1;

        elseif ( (num_of_bmp_images - bmp_counter) > 0)
            imgInfoBMP = imfinfo( fullfile( handles.folder_name,
bmp_files(bmp_counter+1).name ) );
            if ( strcmp( lower(imgInfoBMP.Format), 'bmp') == 1 )
                % read images
                sprintf('%s \n', bmp_files(bmp_counter+1).name)
                % extract features
                image = imread( fullfile( handles.folder_name,
bmp_files(bmp_counter+1).name ) );
                [pathstr, name, ext] =
fileparts( fullfile( handles.folder_name,
bmp_files(bmp_counter+1).name ) );
                image = imresize(image, [384 256]);
            end

            bmp_counter = bmp_counter + 1;

        end

        hsvHist = hsvHistogram(image);
```

```
        autoCorrelogram = colorAutoCorrelogram(image);
        color_moments = colorMoments(image);
        % for gabor filters we need gary scale image
        img = double(rgb2gray(image))/255;
        [meanAmplitude, msEnergy] = gaborWavelet(img, 4, 6); % 4 =
number of scales, 6 = number of orientations
        wavelet_moments = waveletTransform(image);
        % construct the dataset
        set = [hsvHist autoCorrelogram color_moments meanAmplitude
msEnergy wavelet_moments];
        % add to the last column the name of image file we are
processing at
        % the moment
        dataset(k, :) = [set str2num(name)];

        % clear workspace
        clear('image', 'img', 'hsvHist', 'autoCorrelogram',
'color_moments', ...
            'gabor_wavelet', 'wavelet_moments', 'set',
'imgInfoJPG', 'imgInfoPNG', ...
            'imgInfoGIF');
    end

    % prompt to save dataset
    uisave('dataset', 'dataset1');
    % save('dataset.mat', 'dataset', '-mat');
    clear('dataset', 'jpg_counter', 'png_counter', 'bmp_counter');
end


% --- Executes on button press in btn_LoadDataset.
function btn_LoadDataset_Callback(hObject, eventdata, handles)
% hObject    handle to btn_LoadDataset (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
[fname, pthname] = uigetfile('*.mat', 'Select the Dataset');
if (fname ~= 0)
    dataset_fullpath = strcat(pthname, fname);
    [pathstr, name, ext] = fileparts(dataset_fullpath);
    if ( strcmp(lower(ext), '.mat') == 1)
        filename = fullfile( pathstr, strcat(name, ext) );
        handles.imageDataset = load(filename);
        guidata(hObject, handles);
        % make dataset visible from workspace
        % assignin('base', 'database',
handles.imageDataset.dataset);
        helpdlg('Dataset loaded successfuly!');
    else
        errordlg('You have not selected the correct file type');
    end
else
    return;
end
```