Image Processing App Documentation

Overview:

Application is responsible for doing Operations, Processing on an Image as (Equalizing Image, Fourier Transform, Adding, Removing Noise, Detecting Edges)

Application GUI:

- Login Page to authorize login user.
- Menu Tab which contains button that links another tabs
- Different Tabs (each tab has different functionality)

Included Tabs:

1. **Histogram Tab:** User can Equalize Image and Display Equalized Image, and Histogram Equalization
2. **Filter Tab:** User Can apply sobel, or laplacian filter
3. **Fourier Tab:** User Can apply fourier transform and display transformed image.
4. **Salt & Pepper Noise Tab:** User Can add Salt & Pepper Noise, Remove it, and display Nosiy and filtered Images.
5. **Periodic Noise Filter Tab:** User Can add Periodic Noise (even in x, y, or both directions) and remove it by 3 different methods (Notch Filter, Band Reject, or Masking with zeros)

** Note:

All functions in this documentation it's just a demo for main (actual) function to clarify what is the function do.

Tab 1 - Login Tab:

- User must log in to enter the app (Username: Admin, Password: 1234)

Highlighted Methods:

Login Function

```
% get username input object to use it later
userObject = findobj(0, 'tag', 'usernameInput');
% get password input object to use it later
passwordObject = findobj(0, 'tag', 'passwordInput');
% ger login page object to use it later
loginPageObject = findobj(0, 'tag', 'loginTab');
% get text in the input that users enters
username = get(userObject, 'string');
% get text in the input that users enters
password = get(passwordObject, 'string');
user = 'Admin'; % valid username
passwd = '1234'; % valid password
% Compare user input with valid username
```

```
checkUser = strcmp(username, user);
% Compare user input with valid password
checkPasswd = strcmp(password, passwd);
% check that user enters correct username and password
if checkUser == 1 && checkPasswd == 1
    % open menu tab
    menu;
    % msg for success logged in
    msgbox('You Logged in Successfully');
    % clearing username, password values
    set(userObject, 'string', '');
    set(passwordObject, 'string', '');
    % closing login tab
    delete(loginPageObject);
    % Check if username is incorrect and password is correct
elseif checkUser ~=1 && checkPasswd == 1
    % msg to user
    msgbox('Wrong Username, Please Enter a valid Username!');
    % Check if username is correct and password is incorrect
elseif checkUser ==1 && checkPasswd ~= 1
    % msg to user
    msgbox('Wrong Password, Please Enter a valid Password!');

    % Check if both username and password are incorrect
else
    % msg to user
    msgbox('Wrong Username and Password, Please try again!');
end
```

Tab 2 - Menu Tab:

- User can navigate by this tab through different tabs in app (Histogram, Filter, Noise, Fourier tabs).

Highlighted Methods:

Go to Histogram Tab Button:

```
% It's a button to go to histogram tab

% opening histogram page
histTab;
% closing current page to be unseen for user
currentPage = findobj(0, 'tag', 'homePage');
delete(currentPage);
```

Go to Filter Tab Button:

```
% It's a button to go to filter tab

% opening filters page
```

```
filterTab;
% closing current page to be unseen for user
currentPage = findobj(0, 'tag', 'homePage');
delete(currentPage);
```

Go to Salt & Pepper Noise Tab Button:

```
% It's a button to go to S%P tab

% opening Salt & Pepper Noise page
spTab;
% closing current page to be unseen for user
currentPage = findobj(0, 'tag', 'homePage');
delete(currentPage)
```

Go to Fourier Tab Button:

```
% opening Fourier page
fourierTab;
% closing current page to be unseen for user
currentPage = findobj(0, 'tag', 'homePage');
delete(currentPage);
```

Go to Periodic Noise Tab Button:

```
% opening Periodic Noise page
periodicNoiseTab;
% closing current page to be unseen for user
currentPage = findobj(0, 'tag', 'homePage');
delete(currentPage);
```

Tab 3 - Histogram Tab

- User can Equalize Image and Display Equalized Image, and Histogram Equalization

Highlighted Methods:

Histogram

```
% It's a Demo not a main function
% -----

% check if browsed image is empty
% if user clicks on it before choosing image to avoid application crash
if isempty(choosenImg)
    msgbox('Please Choose image first');
    return
end
% read image
choosenImg = imread(choosenImg);
```

```
% choose axes to display image
axes(handles.histogramView);
% apply histogram to image
imhist(choosenImg);
% color x, y axes to white colors
set(handles.histogramView,'XColor','white');
set(handles.histogramView,'YColor','white');
```

Histogram Equalization

```
% It's a Demo not a main function
% -----
name = strcat(Pathname, Filename);
% read image
choosenImg = imread(name);
% apply histogram equlization
hq = histeq(choosenImg);
% choose axes to plot equlaized image
axes(handles.histogramEqImgView);
% display equlaized image
imshow(hq);
% choose axes to plot histogram equalization of an equlaized image
axes(handles.histogramEqView);
% display histogram equalization
imhist(hq);
% color x, y axes to white colors
set(handles.histogramEqView,'XColor','white');
set(handles.histogramEqView,'YColor','white');
```

Tab 4 - Fourier Transform Tab

- User Can apply fourier transform and display transformed image.

Highlighted Methods:

Apply Fourier

```
% It's a Demo not a main function
% -----
browsedImg = strcat(Pathname, Filename);
browsedImg = imread(browsedImg);
% apply fourier and fourier shift to display the transformed image
browsedImgFourier = fftshift(fft2(browsedImg));
fl = log(1+abs(browsedImgFourier));
fm = max(fl(:));
% choosing axes to display
axes(handles.TransformedImgView);
% display the transformed image
imshow(im2uint8(fl/fm));
```

Tab 5 - Filters Tab

- User Can apply sobel with Threshold, and Direction of edges (vertical, horizontal, or both), or laplacian filter with alpha, and shape of image (same, valid, or full).

Highlighted Methods:

Apply Filters

```
% It's a Demo not a main function
% -----
originalImg = strcat(Pathname, Filename);
originalImg = imread(originalImg);
% Convert rgb image to gray image
if size(originalImg, 3) == 3
    originalImg = rgb2gray(originalImg);
end
% If user chose laplacian method
if popChoice == "laplacian"
    % ------ Handling Errors to avoid app craching ----- %
    % check if input value1(alpha) is within range
    % check if input value2(shape) is a right method (same, valid, or full)
    if (inputValue1 >= 0 && inputValue1 <= 1) && (inputValue2 == "same" || inputValue2 == "vali
        % set output text object by value1(alpha)
        set(alphaResult, 'string', inputValue1);
        % set output text object by value2(shape)
        set(shapeResult, 'string', inputValue2);
        % apply laplacian filter
        fltr = fspecial('laplacian', inputValue1);
        laplaceFltr = filter2(fltr, originalImg, inputValue2);
        % choose axes to display filterd image
        axes(handles.filterdImgView);
        % display image
        imshow(laplaceFltr);
    end

% If user chose sobel method
elseif popChoice == "sobel"
    % ------ Handling Errors to avoid app craching ----- %
    % check if input value1(threshold) is within range [0, 1]
    % check if input value2(direction) is a right method (vetical, horizontal, or both)
    if (inputValue1 >= 1 && inputValue1 <= 255)&& (inputValue2 == "vertical" || inputValue2 ==
        % set output text object by value1(threshold)
        set(alphaResult, 'string', inputValue1);
        % set output text object by value2(direction)
        set(shapeResult, 'string', inputValue2);
        % apply sobel filter
        fltr = edge(originalImg, 'sobel', inputValue1/255, inputValue2);
        % choose axes to display filterd image
        axes(handles.filterdImgView);
        % display image
        imshow(fltr);
    end
end
```

Tab 6 - Salt & Pepper Noise Tab

- User Can add Salt & Pepper Noise, Remove it, and display Nosiy and filtered Images

Highlighted Methods:

Add Salt & Pepper Noise

```
% check if user click button without choosing image
if isempty(origImg)
    msgbox('Please Choose image first');
    return
end
origImg = strcat(Pathname, Filename);
origImg = imread(origImg);
% convert rgb image to gray image
if size(origImg, 3) == 3
    origImg = rgb2gray(origImg);
end
% density input
densityInputObject = findobj(0, 'tag', 'densityInput');
densityValue = get(densityInputObject, 'string');
temp = densityValue;
densityValue = str2double(char(densityValue));

% check if user enter value not within range
if (densityValue >= 0 && densityValue <= 1)
    set(densityResultObject, 'string', densityValue);
    noisyImg = imnoise(origImg, 'salt & pepper', densityValue);
    axes(handles.noisyImgView);
    imshow(noisyImg);
end
```

Remove Salt & Pepper Noise

```
% check if noise added to an image or not
if isempty(noisyImg)
    msgbox('Please Choose image first');
    return
end
% Mask input
maskInputObject = findobj(0, 'tag', 'maskInput1');
maskInput2Object = findobj(0, 'tag', 'maskInput2');
maskValue1 = get(maskInputObject, 'string');
maskValue2 = get(maskInput2Object, 'string');
mskVal1 = maskValue1;
```

```
mskVal2 = maskValue2;
maskValue1 = str2double(char(maskValue1));
maskValue2 = str2double(char(maskValue2));
mskVal1 = isempty(mskVal1);
mskVal2 = isempty(mskVal2);

% check that mask value not negative number
if (maskValue1 > 0 && maskValue2 > 0)
    x = num2str([maskValue1 maskValue2]);
    x = ['[', x, ']'];
    set(maskResultObject, 'string', x);
    % apply median filter
    filtredImg = medfilt2(noisyImg, [maskValue1 maskValue2]);
    axes(handles.clearImgView);
    imshow(filtredImg);
end
```

Tab 7 - Periodic Noise Tab

- User Can add Salt & Pepper Noise, Remove it, and display Nosiy and filtered Images

Highlighted Methods:

Add Periodic Noise

```
% function that used to add periodic noise to an image
function addingPeriodicNosie(value, direction, img, nxValue, nyValue, handles)
global periodicNoisyImg;
imgSize = size(img);
[x, y]= meshgrid(1:imgSize(2),1:imgSize(1));
Wx = max(max(x));
Wy = max(max(y));
% user chose x-direction
if direction == 'x'
    fx = nxValue/Wx;
    % if user chose sine
    if value == 1
        px = sin(2*pi*fx*x)+1;
    % if user chose cosine
    elseif value == 2
        px = cos(2*pi*fx*x)+1;
    end
    periodicNoisyImg = mat2gray((im2double(img)+px));
% user chose y-direction
elseif direction == 'y'
    fy = nyValue/Wy;
    % if user chose sine
    if value == 1
        py = sin(2*pi*fy*y)+1;
    % if user chose cosine
    elseif value == 2
```

```
        py = cos(2*pi*fy*y)+1;
    end
    periodicNoisyImg = mat2gray((im2double(img)+py));
% user chose both direction
elseif direction == 'b'
    fx = nxValue/Wx;
    fy = nyValue/Wy;
    if value == 1
        % if user chose sine
        pxy = sin(2*pi*(fx*x + fy*y))+1;
    elseif value == 2
        % if user chose cosine
        pxy = cos(2*pi*(fx*x + fy*y))+1;
    end
    periodicNoisyImg = mat2gray((im2double(img)+pxy));
end
% Display Nosiy Image
axes(handles.noisyImgView);
imshow(periodicNoisyImg);
noisyImgFrequency = fftshift(fft2(periodicNoisyImg));
% Display Fourier of Nosiy Image
axes(handles.fourierNoisyImgView);
fl = log(1+abs((noisyImgFrequency)));
fm = max(fl(:));
imshow(im2uint8(fl/fm))
end
```

Removing Periodic Noise

- There are three methods to remove periodic noise (Notch, Band , and Masking)

Notch Filter

```
% Demo from a main function
function removePeriodicNoise(noisyImg, method, handles)
% if method is notch
if method == "notch"
    % if noise in x, y directions
    if nxValue > 0 && nyValue > 0
        val1 = z((centerX + nxValue), (centerY + nyValue));
        val2 = z((centerX - nxValue), (centerY - nyValue));
        [xVal1, yVal1] = find(z == val1);
        [xVal2, yVal2] = find(z == val2);
        noisyImgFrequency(xVal1, :) = 0;
        noisyImgFrequency(xVal2, :) = 0;
        noisyImgFrequency(:, yVal1) = 0;
        noisyImgFrequency(:, yVal2) = 0;
        axes(handles.cleanImageView);
        fl = log(1+abs(ifft2(noisyImgFrequency)));
        fm = max(fl(:));
        imshow(im2uint8(fl/fm))
        return
    end
end
end
```

Band Reject

```
% Demo from a main function
function removePeriodicNoise(noisyImg, method, handles)
% if method is band
if method == "band"
    % if noise in x, y directions
    if nxValue > 0 && nyValue > 0
        val1 = z((centerX + nxValue), (centerY + nyValue));
        val2 = z((centerX - nxValue), (centerY - nyValue));
        br = (z < (val1 - 2) | z > (val1 + 2)); % reject band of frequencies around spikes
        imgFourierBinary = noisyImgFrequency.*br;
        axes(handles.cleanImageView);
        fl = log(1+abs(ifft2(imgFourierBinary)));
        fm = max(fl(:));
        imshow(im2uint8(fl/fm))
        return
    end
end
end
```

Masking with zeros

```
% Demo from a main function
function removePeriodicNoise(noisyImg, method, handles)
if method == "mask"
    % Filtering in Fourier Domain
    % Mask 1 inputs
    xAxisMask1InputObject = findobj(0, 'tag', 'xAxisMask1Input');
    yAxisMask1InputObject = findobj(0, 'tag', 'yAxisMask1Input');
    % Mask 2 inputs
    xAxisMask2InputObject = findobj(0, 'tag', 'xAxisMask2Input');
    yAxisMask2InputObject = findobj(0, 'tag', 'yAxisMask2Input');
    % get objects from ui
    mask1ResultObject = findobj(0, 'tag', 'mask1Result');
    mask2ResultObject = findobj(0, 'tag', 'mask2Result');
    %%% Mask
    mask = ones(size(noisyImgMagnitude));

    % getting inputs from user by mouse click
    axes(handles.fourierNoisyImgView);
    [x, y] = ginput(2);
    xVal1 = round(y(1));
    yVal1 = round(x(1));
    xVal2 = round(y(2));
    yVal2 = round(x(2));
    if nxValue > 0 && nyValue > 0
        set(xAxisMask1InputObject, 'string', xVal1);
        set(yAxisMask1InputObject, 'string', yVal1);
        set(xAxisMask2InputObject, 'string', xVal2);
        set(yAxisMask2InputObject, 'string', yVal2);

        mask(xVal1, yVal1) = 0;
        mask(xVal2, yVal2) = 0;
        %
        msk = num2str([xVal1 yVal1]);
        msk = ['(', msk, ')'];
        set(mask1ResultObject, 'string', msk);
        %
        msk2 = num2str([xVal2 yVal2]);
        msk2 = ['(', msk2, ')'];
        set(mask2ResultObject, 'string', msk2);
        %
        filtered = mask.*noisyImgFrequency;
        filteredImage = ifft2(ifftshift(filtered));
        ampFilteredImage = abs(filteredImage);
        minValue = min(min(ampFilteredImage));
        maxValue = max(max(ampFilteredImage));
        axes(handles.cleanImageView);
        imshow(ampFilteredImage, [minValue maxValue]);
        return

    end
end
end
```

Global Functions

Browse Function:

```
% ----- Function to browse an image
% --- functions prevents user to choose any file except images to avoid
% crash
function [filename, Pathname, choosenImg] = BrowseImagefile(orginalImgeAxesObject, ...
    imagePathPreviewObject)
% get static text object to use it later
imgPathAxesObject = findobj(0, 'tag', 'imgPath');
% clear any text on it
set(imgPathAxesObject, 'string', '');
% To make user choose images only
[filename, Pathname] = uigetfile({'*.jpg;*.png;*.jpeg;*.JPG;*.PNG;*.JPEG;*.tif;*.TIF'}, ...
    'File Selector');
% check if user click browse image and didn't choose any file
if Pathname ~= 0
    % Choosen file
    choosenImg = strcat(Pathname, filename);
    [~,~,ext] = fileparts(filename);
    % Check if user choose one of these extensions (image)
    if ext == ".jpg" || ext == ".JPG" || ext == ".jpeg" || ext == ".JPEG" || ...
        ext == ".png" || ext == ".PNG" || ext == ".tif" || ext == ".TIF"
        % set background of browse filename text object by green color
        % "which means browsed file is image"
        set(imagePathPreviewObject, 'BackgroundColor', [0 1 0]);
        set(imagePathPreviewObject, 'string', choosenImg);
        % display original image on it's axes
        axes(orginalImgeAxesObject);
        imshow(choosenImg);
    else
        set(imagePathPreviewObject, 'string', choosenImg);
        % set background of browse filename text object by red color
        % "which means browsed file is not image"
        set(imagePathPreviewObject, 'BackgroundColor', [1 0 0]);
        % msg for user that there is an error
        promptMessage = sprintf('You did not choose image. try again?');
        titleBarCaption = 'ERROR!';
        % ask user if user wants to choose image again or not
        buttonText = questdlg(promptMessage, titleBarCaption, 'Ok', ...
            'Quit', 'Ok');
        % if user clicks quit
        if contains(buttonText, 'Quit')
            % return all values, object to their intial states
            choosenImg = [];
            set(imagePathPreviewObject, 'string', '---');
            set(imagePathPreviewObject, 'BackgroundColor', [1 1 1]);
            return
            % if user clicks ok
        elseif contains(buttonText, 'Ok')
            choosenImg = [];
            % recalling browse function
            BrowseImagefile(filename, Pathname, choosenImg, ...
                orginalImgeAxesObject, imagePathPreviewObject)
            return
        end
        return
    end
else
    % msg for user that there is an error
    promptMessage = sprintf('You did not choose any image. try again?');
    titleBarCaption = 'ERROR!';
    % ask user if user wants to choose image again or not
    buttonText = questdlg(promptMessage, titleBarCaption, 'Ok', 'Quit', 'Ok');
    if contains(buttonText, 'Quit')
        choosenImg = [];
        set(imagePathPreviewObject, 'string', '---');
        set(imagePathPreviewObject, 'BackgroundColor', [1 1 1]);
        return
    elseif contains(buttonText, 'Ok')
        choosenImg = [];
        BrowseImagefile(filename, Pathname, choosenImg, ...
            orginalImgeAxesObject, ...
            imagePathPreviewObject)
        return
    end
end
end
```