FUNDAMENTAL OF IMAGE PROCESSING

S C S V 3 2 1 3

ASSIGNMENT 3
IMAGE RESTORATION & ENHANCEMENT

PREPARED BY: TAN SEE JOU A17CS0218

TASK

Restore and enhance old images with general problems like noise, not clear and so on, by applying enhancement methods with Matlab.

There are a few sets of problem images have been prepared for testing, in *png and *jpg extension.

IMAGES WITH NOISE

- Different level of noise (up to 30%)

Parthenon 5%

Parthenon 10%

Parthenon 15%

- Different images with noise

Penguin 5%

Polar Bear 5%

BLURRED IMAGE

GUI

Zoom in and out of the image

Move the image

Browse the image from folder

Apply median filter to reduce noise

Apply mean filter to reduce noise

Sharpen blurred image

Add contrast

Clear both images

Original problem image

Restored Image
(all methods applied at here)

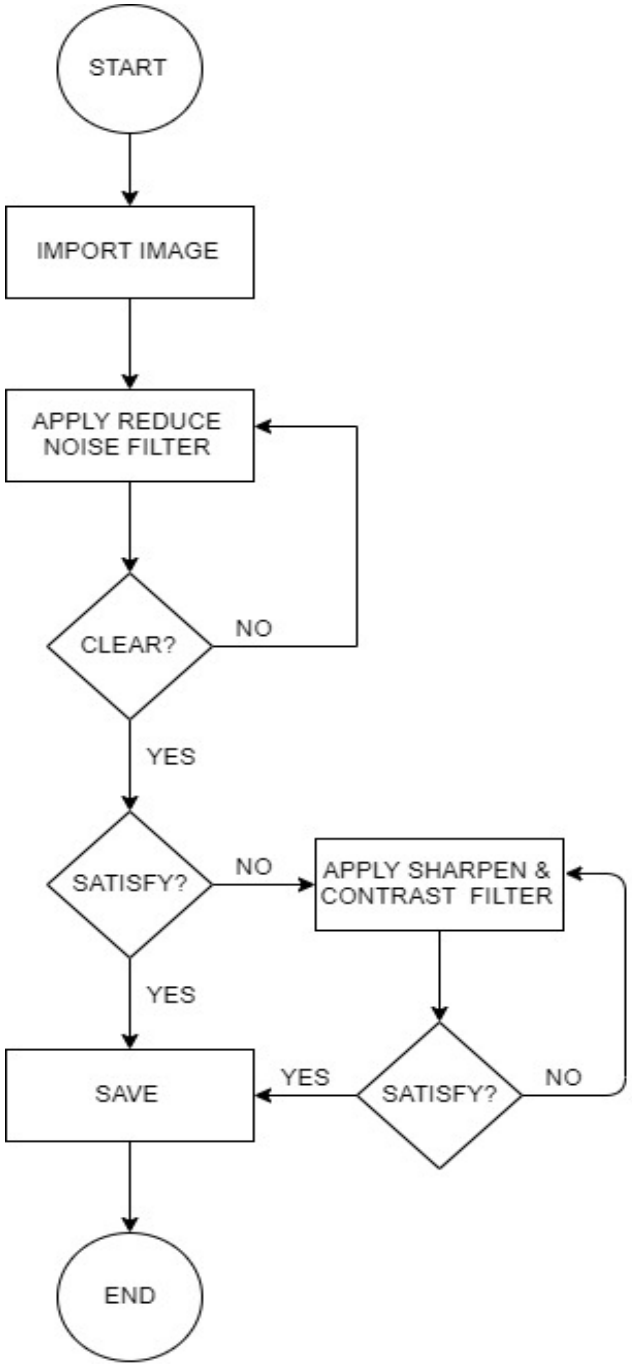
USER MANUAL

I. Reducing noise

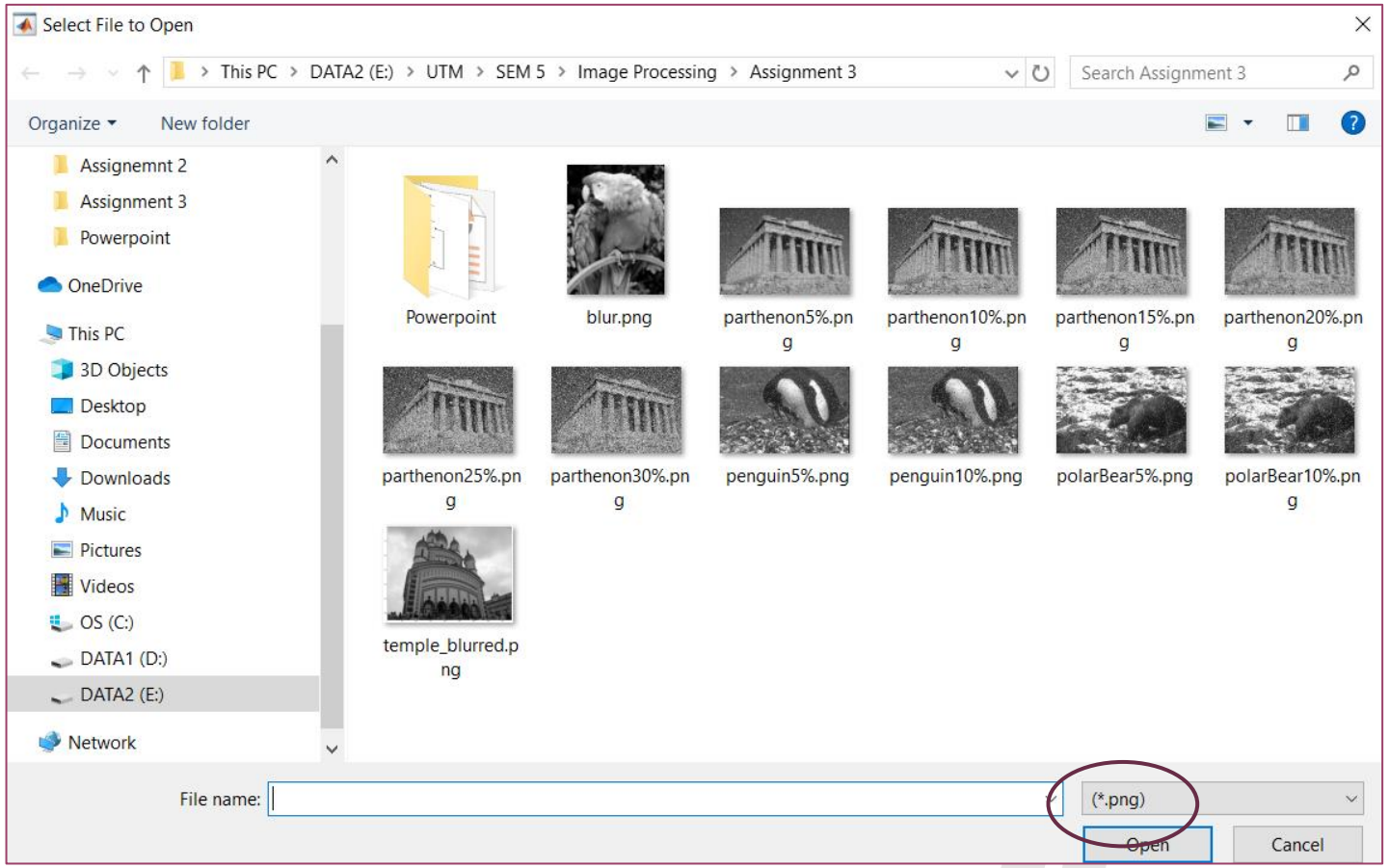
1. Import image from file **(*png)**
2. Choose the type of noise reduce filter *Recommend **median filter.** (can apply several times)
3. a. If the result image is clear, restoration done, else, apply sharpen filter and contrast filter to get a clear image.

I. Deblur image

1. Import image from file **(*jpg)**
2. Apply sharpen filter to get a clear image. (can apply several times)
3. Applied contrast filter to get more contrast image.



PROCESS FLOW: REDUCE NOISE



1. Import noise image

- Import an old image with noise from the folder (*png).
- In this example, parthenon15% will be import.

PROCESS FLOW: REDUCE NOISE

2. Choose median filter to have a clear result.

PROCESS FLOW: REDUCE NOISE

2. Resulted image[with median filter].

- Most of the noise had been remove as compare to the imported image.

PROCESS FLOW: REDUCE NOISE

3. Can zoom in and out with the magnifying tool.

- There are tool for the user the zoom in and out to check the edited image easily.

PROCESS FLOW: REDUCE NOISE

4. Move the edited image with '*hand*' tool.

- There is tool for the user to move and check the edited image easily.

PROCESS FLOW: REDUCE NOISE

*. Resulted image[with **mean** filter].

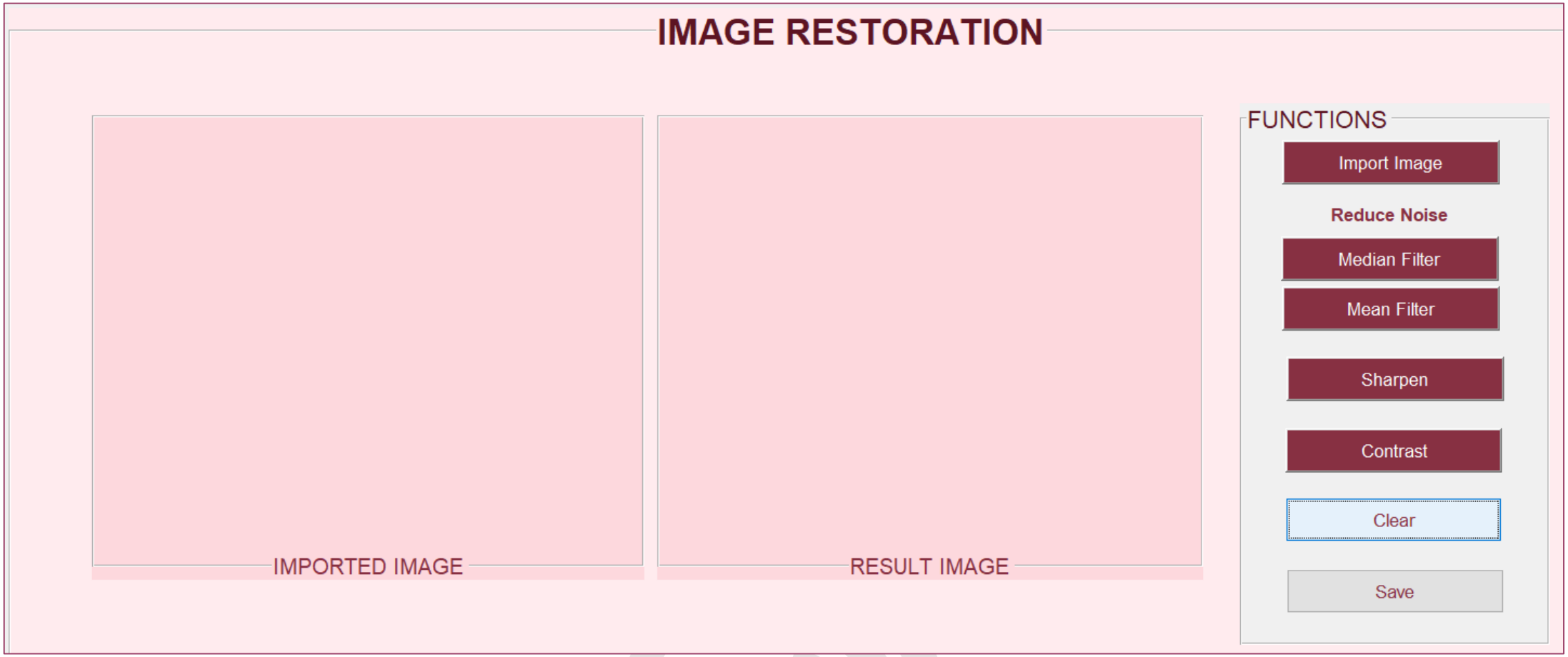
- The noise cannot be reduce effectively when applied the mean filter on the same image(parthenon15%).

PROCESS FLOW: REDUCE NOISE

5. Applied sharpen filter

- The result image is more sharp as compare to before, more details can be seen.

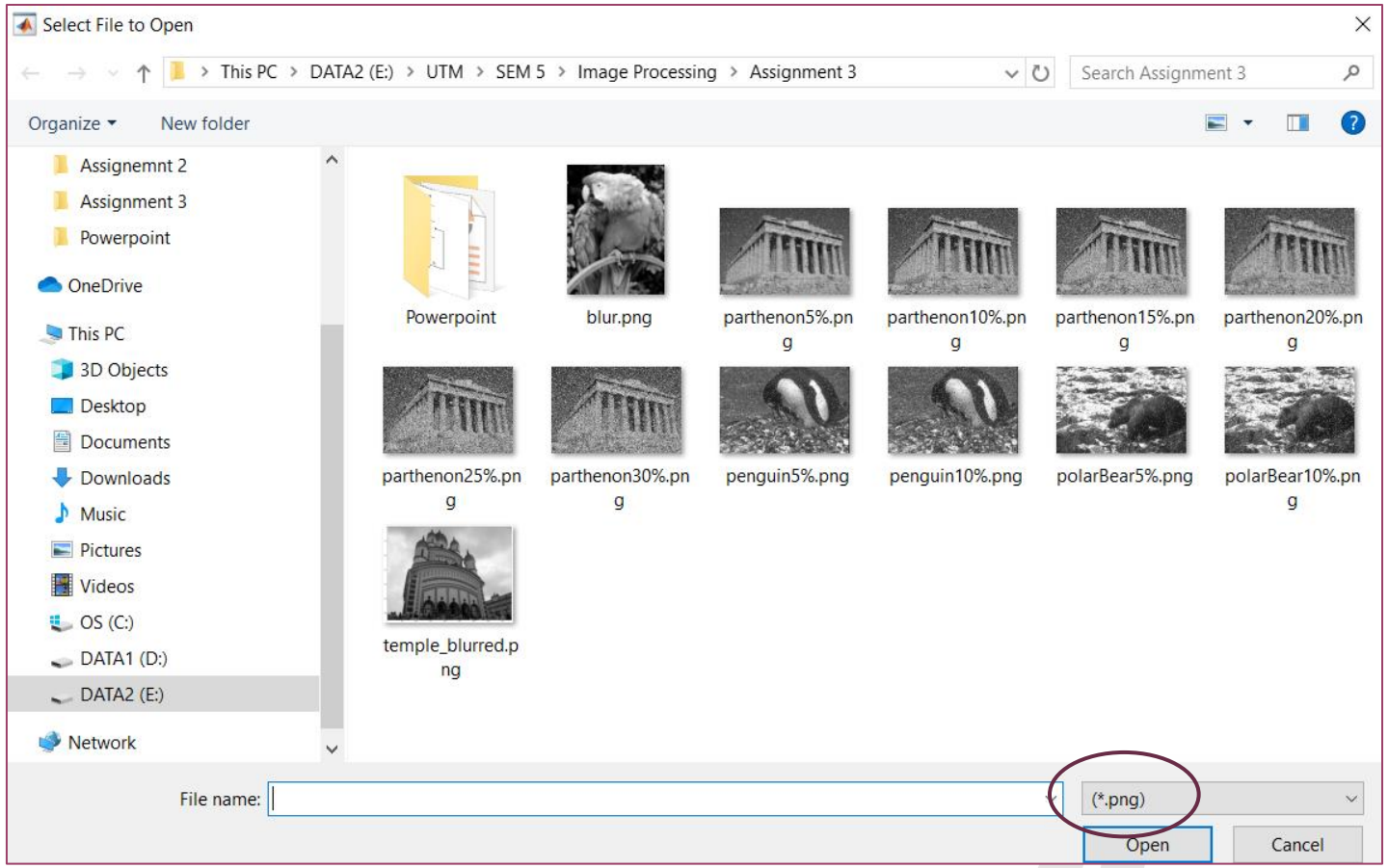
PROCESS FLOW: REDUCE NOISE



6. Click clear button

- Both images had been removed, and ready for next image.

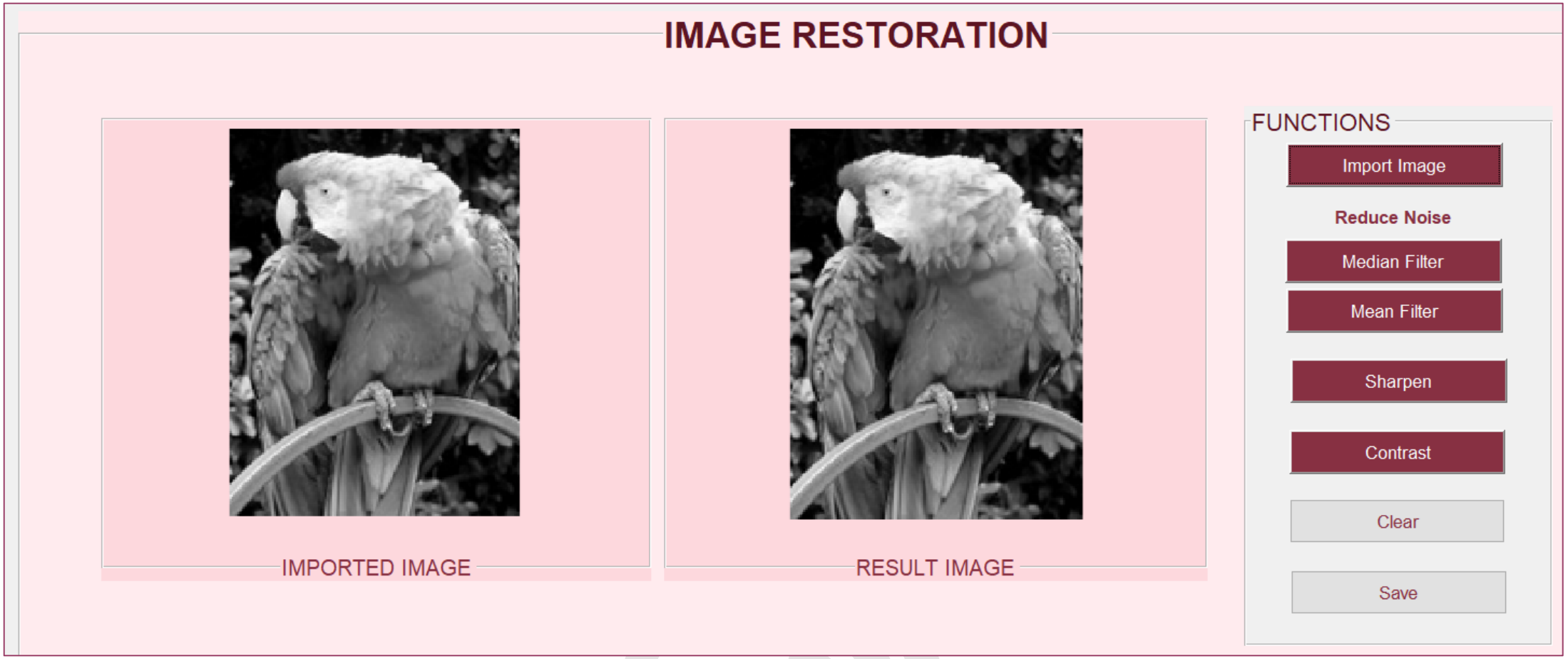
PROCESS FLOW: DEBLUR IMAGE



1. Import not cleaar image

- Import an old not clear from the folder (*jpg or *png).
- In this example, blur.png (which is the parrot image) will be import.

PROCESS FLOW: DEBLUR IMAGE



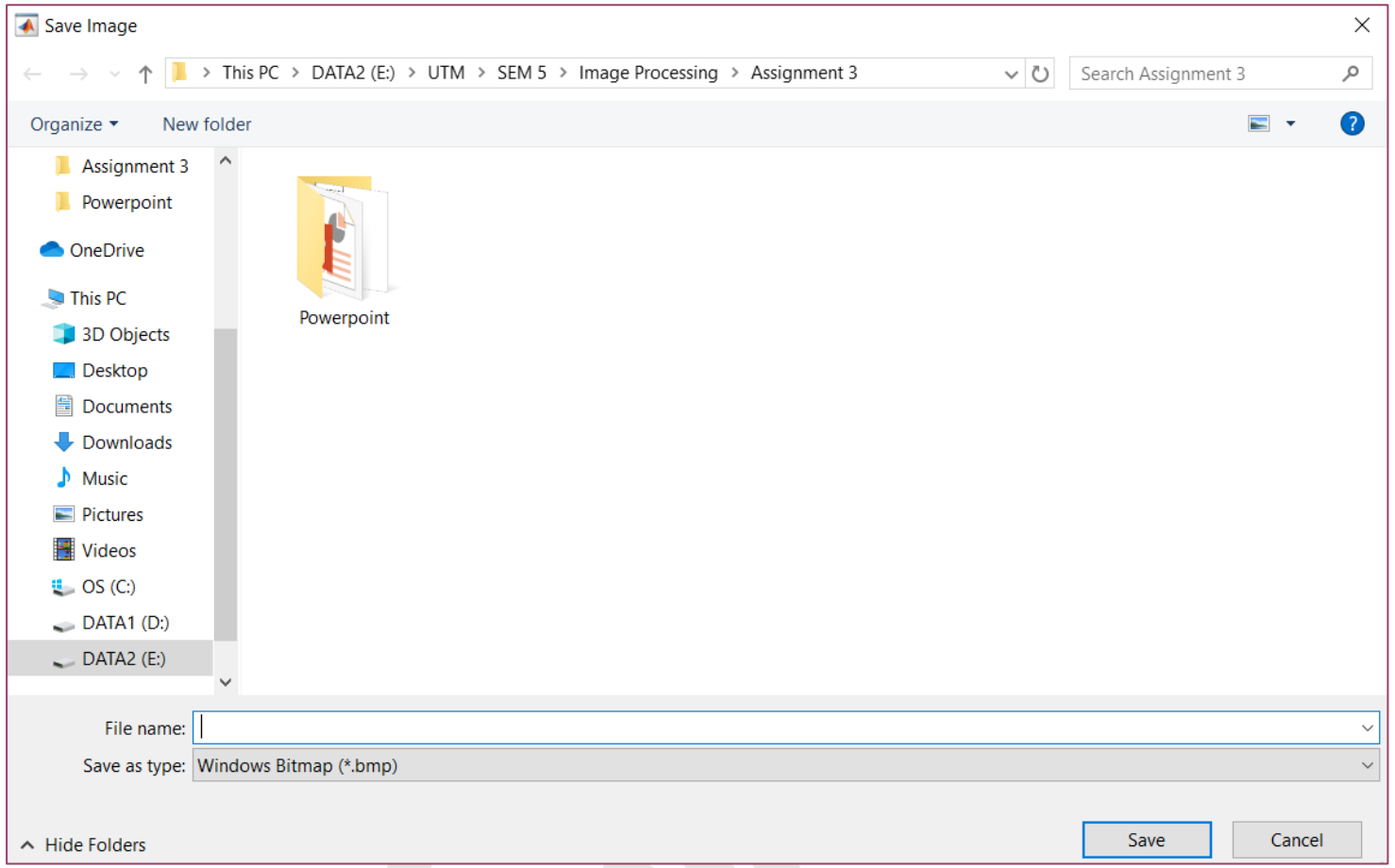
2. Choose sharpen filter.

PROCESS FLOW: DEBLUR IMAGE

2. Resulted Image.

- The result image is more sharp and clear as compare to before, more details can be seen. [sharpen filter had been applied 3 times]

PROCESS FLOW: DEBLUR IMAGE



3. Save Image.

- User can save the edited image.

FUNCTION EXPLAINATION-I

function importimage_callback(hobject, eventdata, handles)

```
function importImage_Callback(hObject, eventdata, handles)
% hObject    handle to importImage (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[file, path] = uigetfile({'*.png';'*.jpg';'*.bmp';'*.tif';'*.jpeg' });
fullFile = fullfile(path, file);    % returns a character vector contain
I=imread(fullFile);
axes(handles.axes1);
imshow(I);
axes(handles.axes2);
imshow(I);
setappdata(handles.axes1, 'img', I);
setappdata(handles.axes2, 'img2', I);
```

- Image had been imported, and *imread()* use to read the image file.
- Select axes and displayed the image on both axes.
- Store the image, *I*, into the handle: *handle.axes1*, as name: *img*.

```
function medianFilter_Callback(hObject, eventdata, handles)
% hObject    handle to medianFilter (see GCBO)
% eventdata  reserved - to be defined in a future version o
% handles    structure with handles and user data (see GUID
J=getappdata(handles.axes2, 'img2');
J_median=medfilt2(J);
axes(handles.axes2);
imshow(J_median);
setappdata(handles.axes2, 'img2', J_median);
```

- If the user click 'median filter', *medianFilter_Callback function()* will be called.
- Last value stored at handles.axes2 is get by using *getappdata()* and store it in *J*.
- Applied *medfilt2()* to *J* and stored it in *J_median*.
- Showed the image and update the value stored in the handle.axes2, so that any further enhancement can be add on.

function medianfilter_callback(hobject, eventdata, handles)

FUNCTION EXPLAINATION-II

function meanfilter_callback(hobject, eventdata, handles)

```
function meanFilter_Callback(hObject, eventdata, handles)
% hObject    handle to meanFilter (see GCBO)
% eventdata  reserved - to be defined in a future version
% handles    structure with handles and user data (see GUI
J=getappdata(handles.axes2, 'img2');
meanFilter=fspecial('average');
J_mean=imfilter(J,meanFilter);
axes(handles.axes2);
imshow(J_mean);
setappdata(handles.axes2, 'img2', J_mean);
```

- If the use choose *'mean filter'*, *meanFilter_Callback function* will be called.
- *fspecial('average')* applied to the image, and stored in *J_mean*.
- Show the image and update the value as same as the last step of median filter.

```
function sharpen_Callback(hObject, eventdata, handles)
% hObject    handle to sharpen (see GCBO)
% eventdata  reserved - to be defined in a future vers
% handles    structure with handles and user data (see
J=getappdata(handles.axes2, 'img2');
J_sharp=imsharpen(J);
axes(handles.axes2);
imshow(J_sharp);
setappdata(handles.axes2, 'img2', J_sharp);
```

- If the user choose 'sharpen filter', *sharpen_Callback function()* will be called.
- Last value stored at handles.axes2 is get by using *getappdata()* and store it in *J*.
- Applied *imsharpen()* to *J* and stored it in *J_sharp*.
- Showed the image and update the value stored in the handle.axes2, so that any further enhancement can be add on.

function sharpen_callback(hobject, eventdata, handles)

FUNCTION EXPLAINATION-III

function contrast_callback(hobject, eventdata, handles)

```
function contrast_Callback(hObject, eventdata, handles)
% hObject    handle to contrast (see GCBO)
% eventdata  reserved - to be defined in a future versi
% handles    structure with handles and user data (see
J=getappdata(handles.axes2, 'img2');
J_contrast=imadjust(J, stretchlim(J),[]);
axes(handles.axes2);
imshow(J_contrast);
setappdata(handles.axes2, 'img2', J_contrast);
```

- If the use choose *'contrast filter'*, *contrast_Callback function* will be called.
- *Imadjust(J, stretchlim(J), [])* applied to the image, and stored in *J_contrast*.
- Show the image and update the value as same as the last step of median filter.

```
function clear_Callback(hObject, eventdata, handles)
% hObject    handle to clear (see GCBO)
% eventdata  reserved - to be defined in a future ve
% handles    structure with handles and user data (s
% cla reset;
arrayfun(@cla,findall(0,'type','axes')) % clear all
```

- If the user choose *'clear filter'*, *clear_Callback function()* will be called.
- Both the images from both axes will deleted and ready to restart again.

function clear_callback(hobject, eventdata, handles)

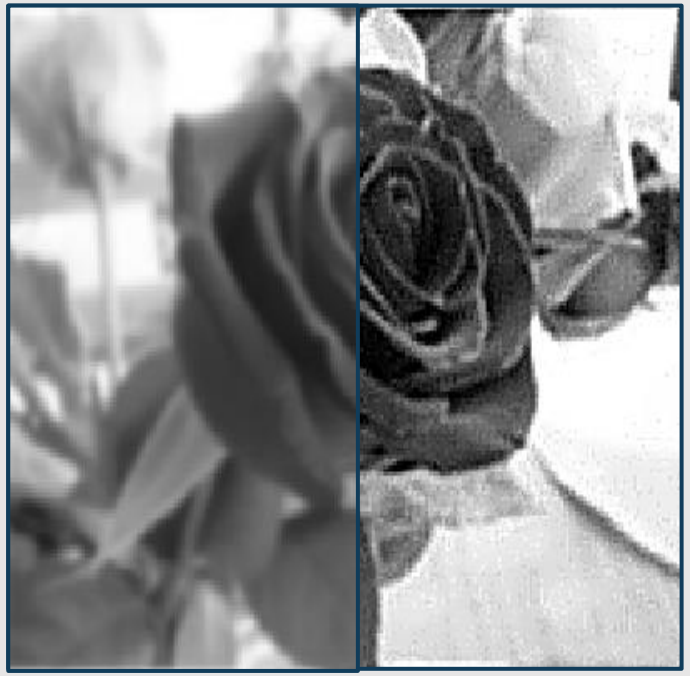
FUNCTION EXPLAINATION-IV

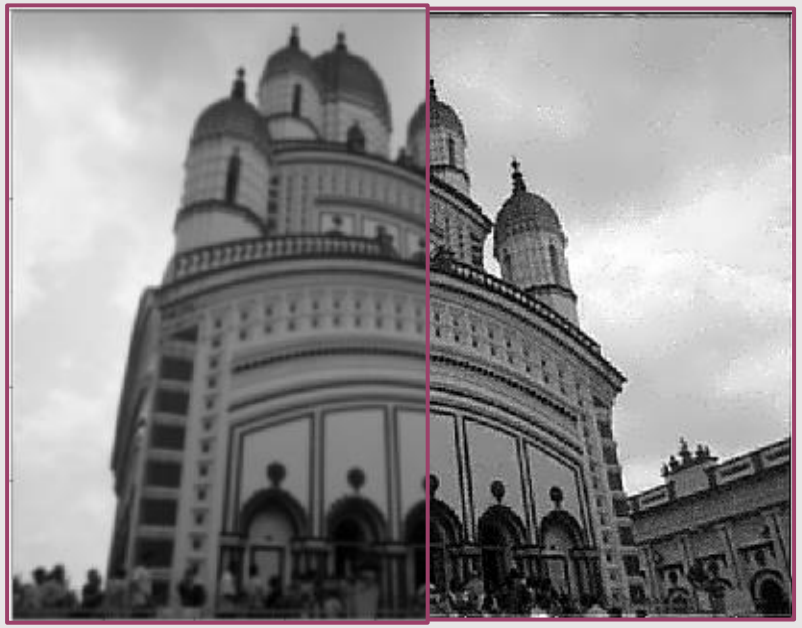
function saveImage_callback(hobject, eventdata, handles)

```
function saveImage_Callback(hObject, eventdata, handles)
% hObject    handle to saveImage (see GCBO)
% eventdata  reserved - to be defined in a future version
% handles    structure with handles and user data (see GU
imsave(handles.axes2);
```

- If the use choose *'save'*, *save_Callback function* will be called.
- User can save the edited image.

OTHERS EXAMPLE IMAGE RESTORATION





THANK YOU !