UTM

UNIVERSITI TEKNOLOGI MALAYSIA

SESSION 2019/2020 SEMESTER 1

ASSIGNMENT 2

NAME:

1) LIM BAO JING (A17CS0076)

2) LOW CHIA JING (A17CS0083)

3)TAN SEE JOU (A17CS0218)

LECTURER NAME: DR. MD. SAH BIN HJ. SALAM

SUBJECT NAME: FUNDAMENTAL OF IMAGE PROCESSING

SUBJECT CODE: SCSV 3213

SECTION: 01

Contents

A. The codes and documentation ... 3
Load Image ... 3
Darken Image ... 4
Blur Image ... 5
Sharpen Image ... 6
Brighten Image ... 7
Save Edited Image ... 8
B. Flow Chart ... 9
Procedure in solving the task ... 10
Create Blank GUI ... 10
Create a Panel ... 10
Create axes ... 11
Create push buttons ... 12
Create LoadImage_Callback function ... 12
Create DarkenImage_Callback function ... 12
Create BlurImage_Callback function ... 12
Create SharpenImage_Callback function ... 13
Create BrightenImage_Callback function ... 13
Create SaveImage_Callback function ... 13
Arrange GUI elements ... 13
Add user guidelines ... 14
C. Output ... 15
The final GUI ... 15
The User Process Chart - how they use this application ... 15
Load Image ... 16
Darken Image ... 17
Save darken image ... 18
Blur Image ... 19
Save Blur Image ... 19
Brighten Image ... 21
Save Brighten Image ... 21
Sharpen Image ... 23
Save Sharpen Image ... 23
Sample Output by mixing all of the Operation ... 25

A. The Codes and Documentation

Load Image

```
% --- Executes on button press in LoadImage.
function LoadImage_Callback(hObject, eventdata, handles)
% hObject    handle to LoadImage (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({'*.jpg';'*.png';'*.bmp';'*.tif' });
fullFileName = fullfile(path, file);
A = imread(fullFileName);
axes(handles.axes1);
imshow(A);
axes(handles.axes2);
imshow(A);
setappdata(handles.axes1, 'img', A);
setappdata(handles.axes2, 'img2', A);
```

Figure 1: Load image

This function enables users to load image from their computer. By using ***uigetfile('*. ')***, a modal dialog box will list the files with the specific extension available in the current folder. User is able to enter or select the file that desire, and if the file is valid, ***uigetfile()*** will return the path and the file name when the user click open. This dialog box aims to prevent the user from interaction with other Matlab windows. ***Fullfile()*** function is used to build a full filename from the directories and filename specified. After that, ***imread()*** function applied to read the image from graphics file. Before showing the image on first axes, ***axes(handles.axes1)*** is mention so that the current axes is set to axes1, so that any operation carry on will only apply to axes 1. After ***A*** display it with ***imshow(A)*** to the axes1, ***A*** store the variable ***A*** to the ***handles.axes1*** and with name 'img', which is ***setappdata(handles.axes1, 'img', A)*** and ***setappdata(handles.axes2, 'img', A)***. Then, same goes to axes2 which is for the edited image.

Darken Image

```
% --------------------DARKEN IMAGE--------------------------------
% --- Executes on button press in DarkenImage.
function DarkenImage_Callback(hObject, eventdata, handles)
% hObject    handle to DarkenImage (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%B = getimage(handles.axes2);
B = getappdata(handles.axes2, 'img2');
B_adj = imdivide(B, 3.0);       %to darken the image
                                %B_adj =immultiply(img,0.5);
bw = im2uint8(roipoly(B));
bw_cmp = bitcmp(bw);

roi = bitor(B_adj, bw_cmp);
not_roi = bitor(B, bw);
new_B = bitand(roi, not_roi);

axes(handles.axes2);
imshow(new_B);
setappdata(handles.axes2, 'img2', new_B);
```

Figure 2: Darken image

This function enables users to choose the area of the image to be darkened by connecting the points to form a region. Firstly, before processing the image, we have to obtain the data of the image. There are 2 approaches we used to obtain the image data. The first one is ***getimage()*** which returns the first image data contained in the graphics object h. In our case, the h is the axes2. The second approach is the ***getappdata()*** which is used to retrieve data stored using the setappdata() which we have used before. After we obtained the image data, we can start to process the image. To darken the image, we also have 2 approaches. The first one is using the ***imdivide()*** which can be used to divide the value of the element in the image with a certain value. In our case, we divide the value of all the elements of the image with 3. So, all the value of the elements will become smaller which means the image will become darker. The second one is using the ***immultiply()*** which can be used to multiply the value of the element in the image with a certain value. In our case, we multiply the value of all the elements of the image with 0.5. When the value that is going to be multiplied is lower than 1, it will cause the values multiplied with it become smaller which causes the image

to become darker. After that, to enable users to choose the region to be processed. We have to use the ***roipoly()*** which creates an interactive polygon tool that returns the mask as the binary image which sets pixels inside the ROI (Region Of Interest) to 1 and pixels outside the ROI to 0. After that, we converted the binary image to uint8 using ***im2uint8()*** and complement the mask with ***bitcmp()***. After that, use logic operator ***bitor()*** on darker image***(B_adj)*** and compliment polygon selected, ***bw_cmp***. Same goes to image(***B***) and ***bw***. Then, logic operator ***bitand()*** is applied to ***roi*** and ***not_roi***. Lastly, show the image on axes 2 and store it using ***setappdata()***.

Blur Image

```
% --------------------BLUR IMAGE--------------------------------
% --- Executes on button press in BlurImage.
function BlurImage_Callback(hObject, eventdata, handles)
% hObject    handle to BlurImage (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%B = getappdata(handles.DarkenImage, 'img2');
B = getappdata(handles.axes2, 'img2');
H =fspecial('disk', 20);
blurred = imfilter(B, H);       %to blur the image
                                %also can use blurred = imgaussfilt(B,20);

bw = im2uint8(roipoly(B));
bw_cmp = bitcmp(bw);

roi = bitor(blurred, bw_cmp);
not_roi = bitor(B, bw);
new_B = bitand(roi, not_roi);

axes(handles.axes2);
imshow(new_B);
setappdata(handles.axes2, 'img2', new_B);
```

Figure 3: Blur image

For the obtaining of image data, enabling users to choose ROI and storing the new data for the processed image, the codes are exactly the same as the others. The only thing different is the operation to process the image data. To blur the image, we also have 2 approaches. First one is using ***fspecial()*** which is used to create predefined 2-D filter. In our case, we used ***fspecial('disk', radius)***, and set the radius to 20. Then, applied ***imfilter()*** to the original image with the blur

operation. The other method that enable the blurring effect is applied ***imgaussfilt()***. This function is used to filter the image with a 2D Gaussian smoothing kernel with standard deviation specified with sigma. After that, the following operations is the same as the darken image function above.

Sharpen Image

```
% --------------------SHARPEN IMAGE--------------------------------
% --- Executes on button press in SharpenImage.
function SharpenImage_Callback(hObject, eventdata, handles)
% hObject    handle to SharpenImage (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%B = getappdata(handles.DarkenImage, 'img2');
B = getappdata(handles.axes2, 'img2');
sharpB = imsharpen(B);          %to sharp the image

bw = im2uint8(roipoly(B));
bw_cmp = bitcmp(bw);

roi = bitor(sharpB, bw_cmp);
not_roi = bitor(B, bw);
new_B = bitand(roi, not_roi);

axes(handles.axes2);
imshow(new_B);
setappdata(handles.axes2, 'img2', new_B);
```

Figure 4: Sharpen image

This sharpen image function enables user to choose the parts of the image to be sharpened. ***Imsharpen()*** function had been applied to increase the contrast along the edges of the color meet by using ***unsharp*** masking method. It can applied to grayscale or true color image. Unsharp masking is subtracting a blurred version of the image from itself. After that, the following operations carried on to the image is the same as the above function.

Brighten Image

```
% --------------------BRIGHTEN IMAGE-------------------------------
% --- Executes on button press in BrightenImage.
function BrightenImage_Callback(hObject, eventdata, handles)
% hObject    handle to BrightenImage (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%B = getappdata(handles.DarkenImage, 'img2');
B = getappdata(handles.axes2, 'img2');
brightB = immultiply(B,1.5);        %to brighten the image
                                    %also can use brightB = imdivide(B,0.5);
bw = im2uint8(roipoly(B));
bw_cmp = bitcmp(bw);

roi = bitor(brightB, bw_cmp);
not_roi = bitor(B, bw);
new_B = bitand(roi, not_roi);

axes(handles.axes2);
imshow(new_B);
setappdata(handles.axes2, 'img2', new_B);
```

Figure 5: Brighten image

For the obtaining of image data, enabling users to choose ROI and storing the new data for the processed image, the codes are exactly the same as the others. The only thing different is the operation to process the image data. To brighten the image, we also have 2 approaches. The first one is using the ***immultiply()*** which can be used to multiply the value of the element in the image with a certain value. In our case, we multiply the value of all the elements of the image with 1.5. So, all the value of the elements will become bigger which means the image will become brighter. The second one is using the ***imdivide()*** which can be used to divide the value of the element in the image with a certain value. In our case, we divide the value of all the elements of the image with 0.5. When the value that is going to be divided is lower than 1, it will cause the values divided with it become bigger which causes the image to become brighter.

Save Edited Image

```
% --------------------SAVE IMAGE--------------------------------
% --- Executes on button press in SaveImage.
function SaveImage_Callback(hObject, eventdata, handles)
% hObject    handle to SaveImage (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
imsave(handles.axes2);
```

Figure 6: Save edited image

imsave() can be used to create a Save image tool. The Save Image tool displays an interactive file chooser dialog box in which the user can specify a path and filename. When the user clicks Save, the Save Image tool writes the target image to a file using the image file format user selected in the Files of Type menu. In our case, only the image in axes2 will be saved since we specified it.

B. Flow Chart

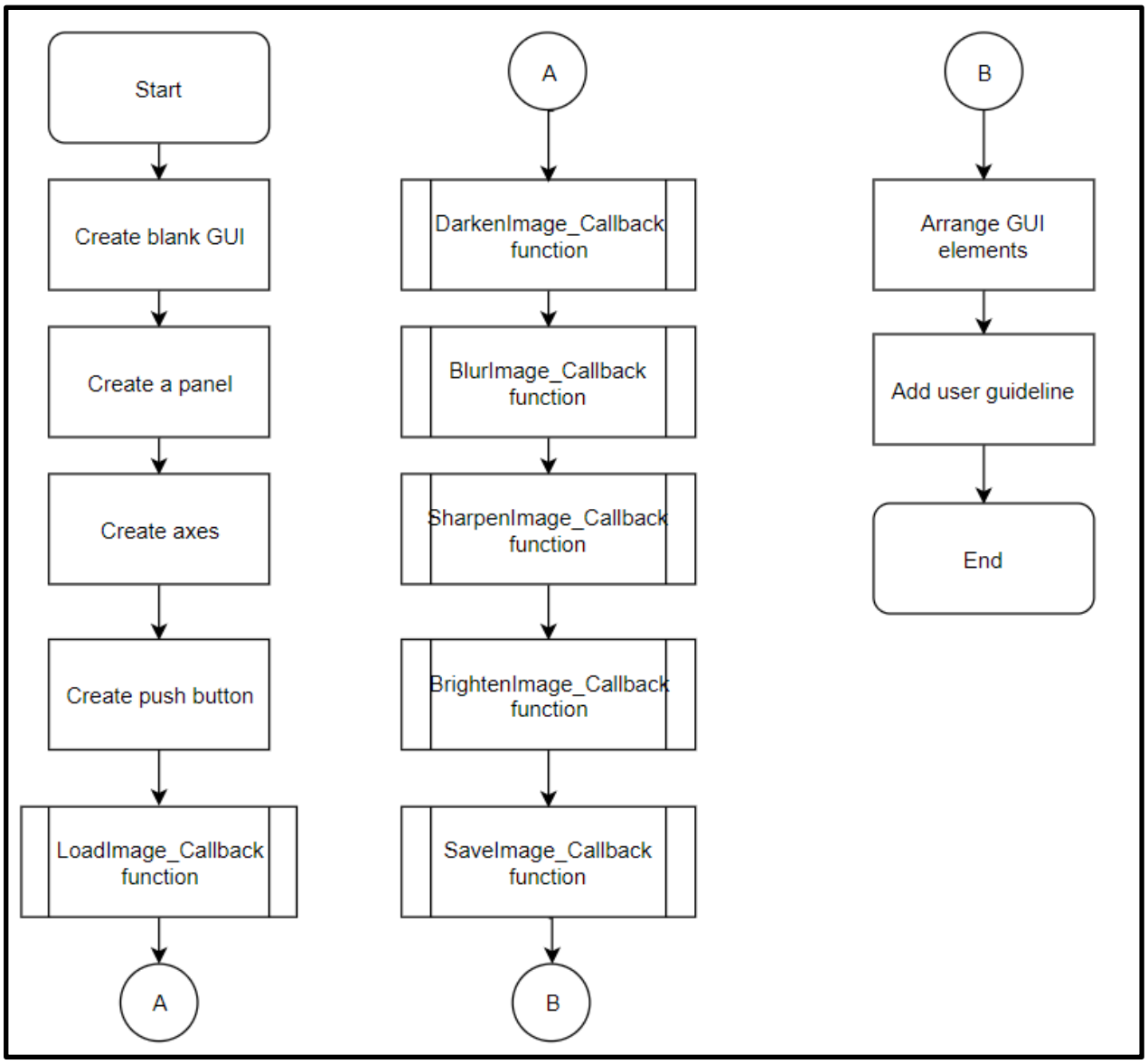

Figure 7: Flowchart of the process

Procedure in solving the task

Create Blank GUI

First, we use the comment ‘guide’ and create a blank GUI.

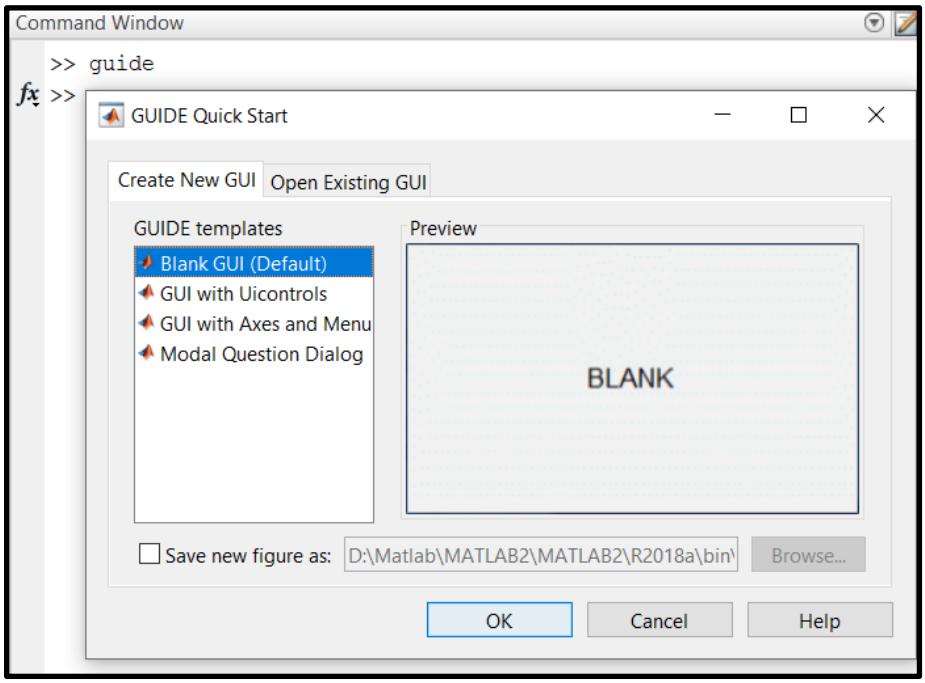

Figure 8: Command window for creating blank GUI

Create a Panel

We drag a panel which is suitable for the size of the GUI. The panel then will become the background which contains all of the GUI elements.

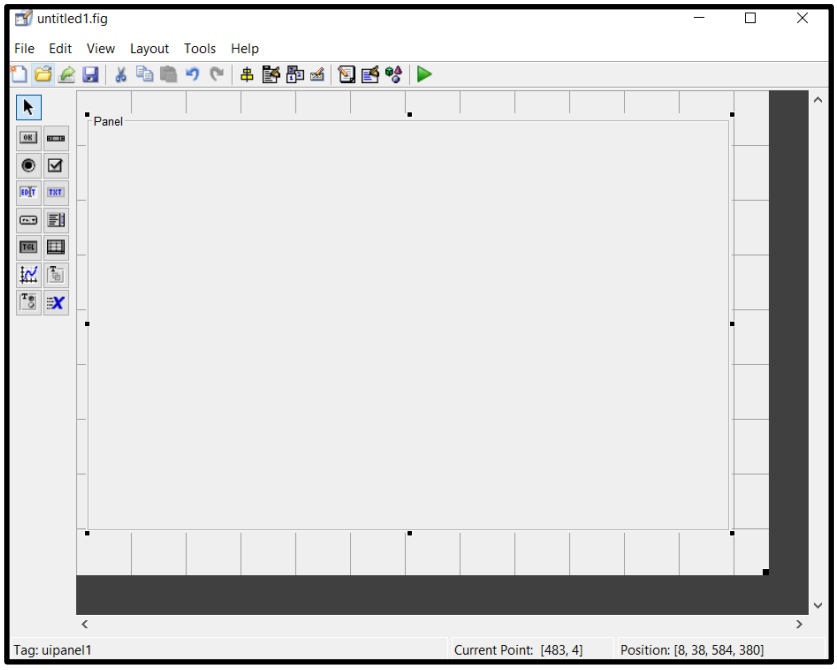

Figure 9: Creating Panel

Create axes

We create two axes and these two axes will then become the container for image loading.

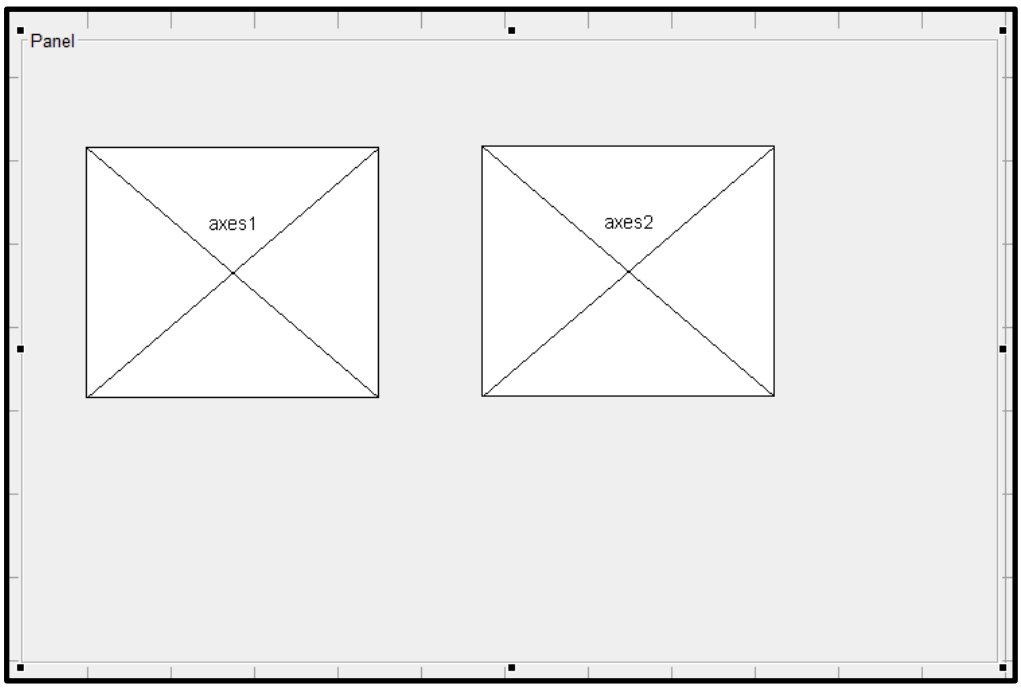

Figure 10: Creating axes

Create push buttons

We create several push buttons, change their name which match their function.

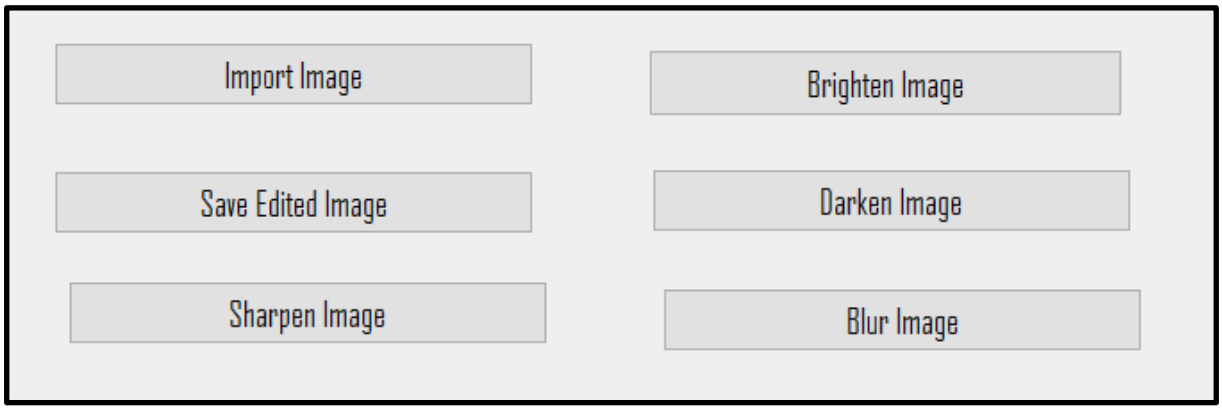

Figure 11: Creating push buttons

Create LoadImage_Callback function

We create a LoadImage_Callback function for the 'Import Image' button which can let user to load an image into both axes from their computer (refer Figure 1).

Create DarkenImage_Callback function

We create DarkenImage_Callback function for the 'Darken Image' button which can let user to darken the selected region (refer Figure 2).

Create BlurImage_Callback function

We create BlurImage_Callback function for the 'Blur Image' button which can let user to blur the selected region (refer Figure 3).

Create SharpenImage_Callback function

We create SharpenImage_Callback function for the 'Sharpen Image' button which can let user to sharpen the selected region (refer Figure 4).

Create BrightenImage_Callback function

We create BrightenImage_Callback function for the 'Brighten Image' button which can let user to brighten the selected region (refer Figure 5).

Create SaveImage_Callback function

We create SaveImage_Callback function for the 'Brighten Image' button which can let user to save the edited image in their computer (refer Figure 6).

Arrange GUI elements

We arrange all the GUI elements such as grouping all of the push button in a container, assigning title to both the image and also editing the panel name. This can make a better GUI which is more clean and easy to use.

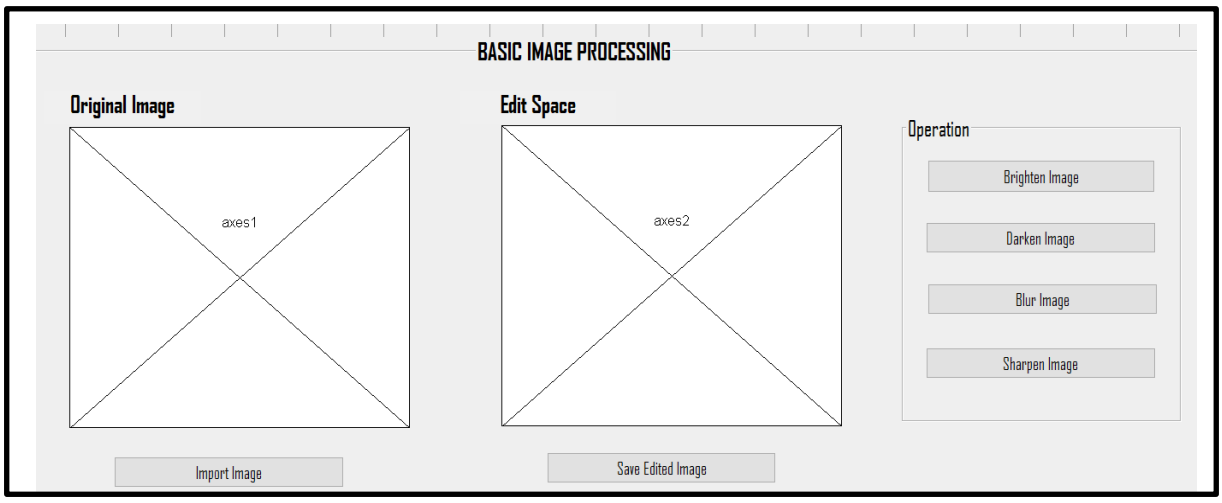

Figure 12: Arranged GUI elements

Add user guidelines

To make the GUI more user-friendly, we added step-by-step user guidelines to help user to understand the procedure of using this application.

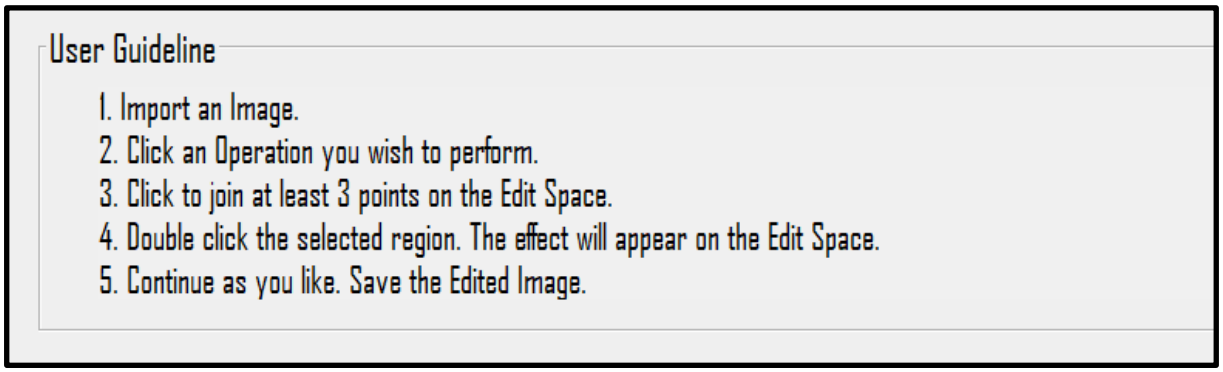

Figure 13: User guidelines in the GUI

C. Output

The final GUI

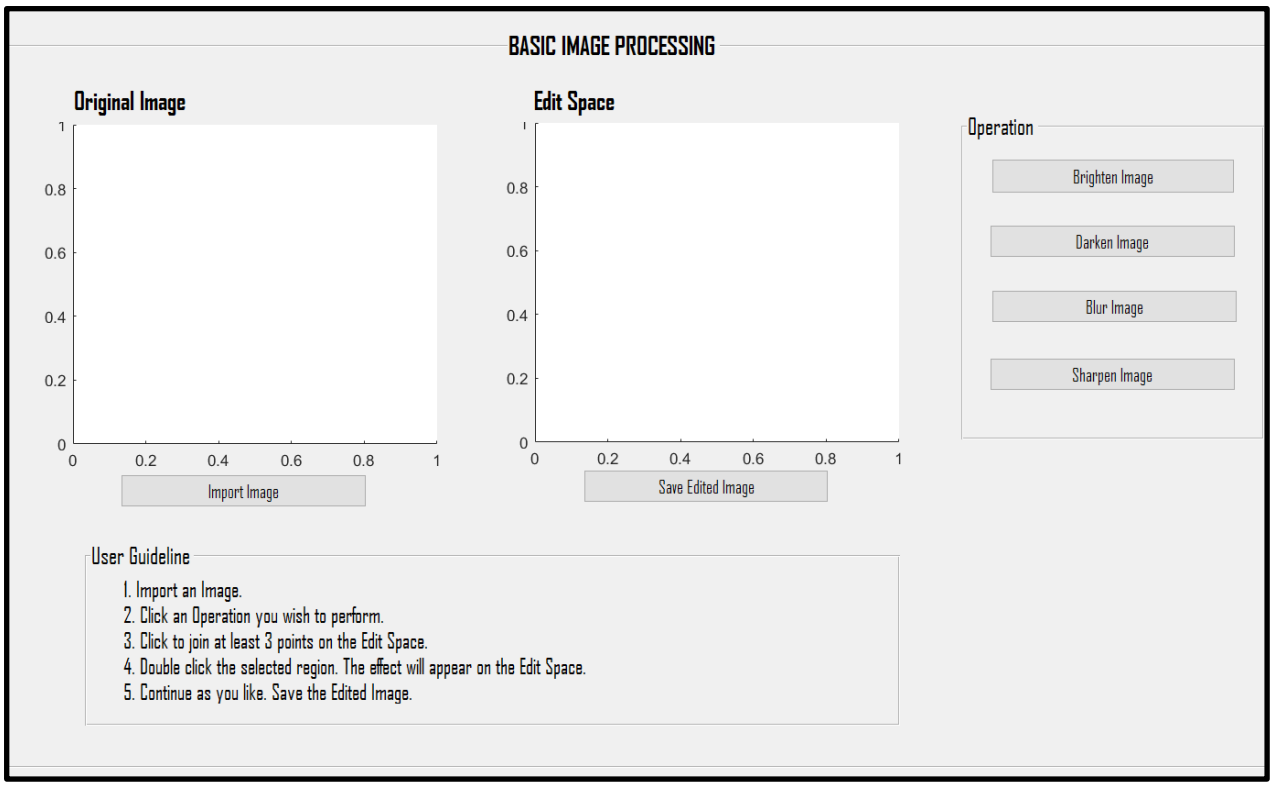

Figure 14: The final GUI

The User Process Chart - how they use this application

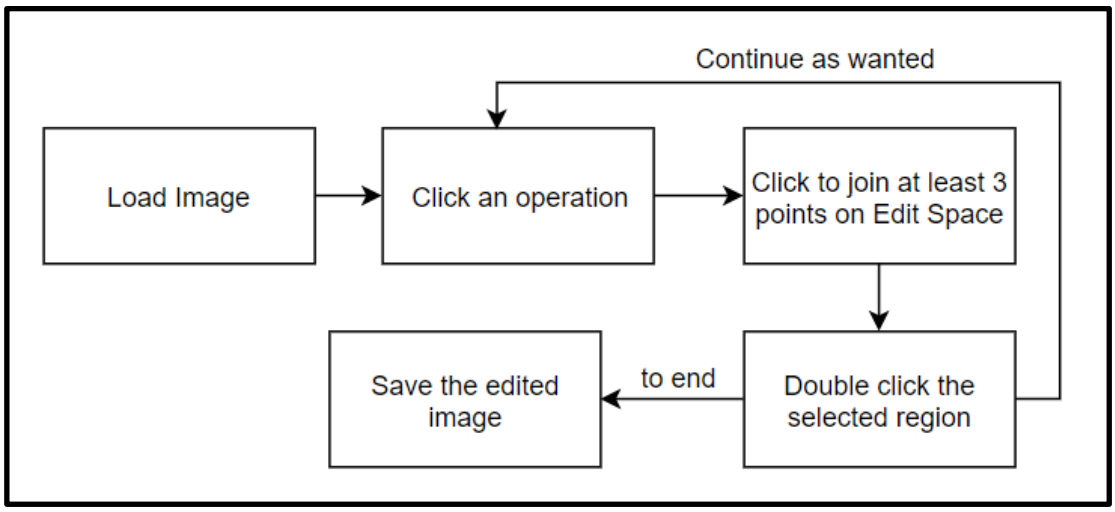

Figure 15: The user process chart

Load Image

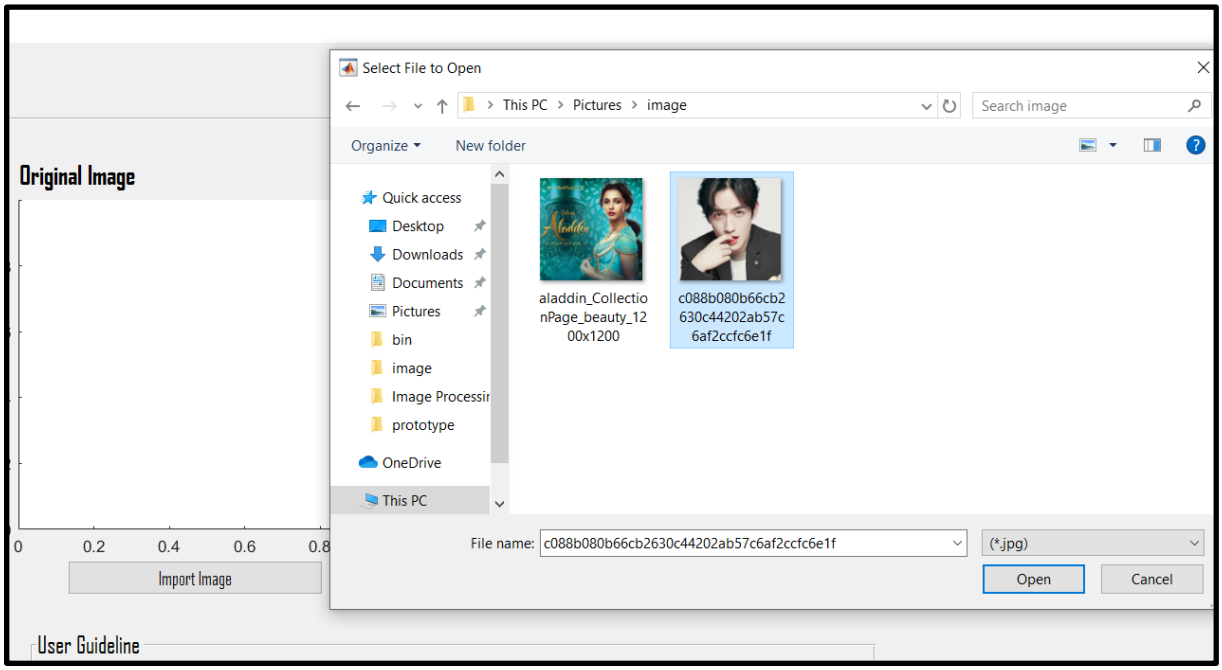

Figure 16: User is able to browse any image from their computer once they clicked 'Import Image' button

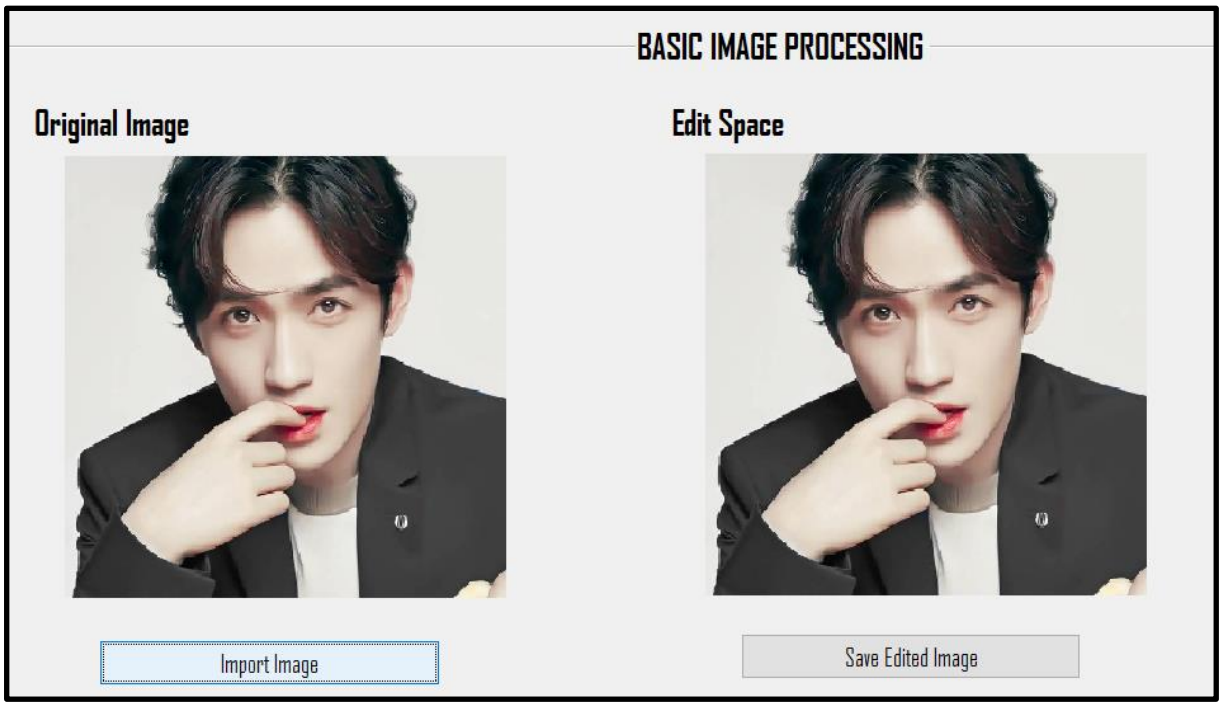

Figure 17: The selected image will be placed in both axes

Choose part in the Edit Space

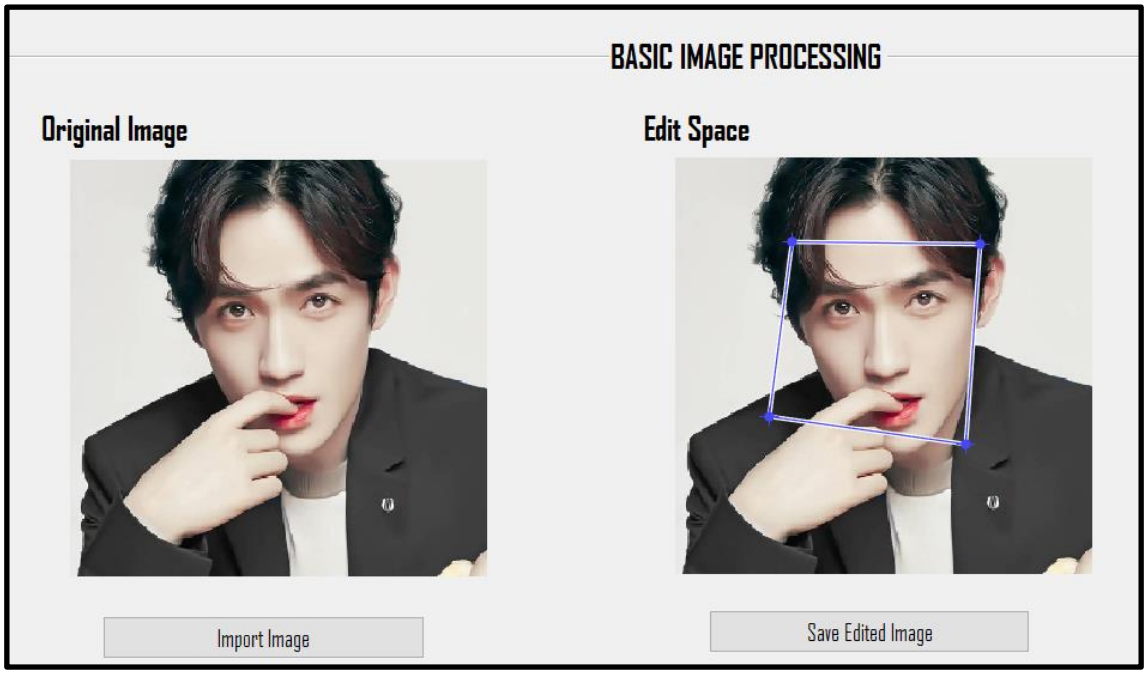

Figure 18: The operation enables users to choose the area of the image to be edited by connecting the points to form a region.

Darken Image

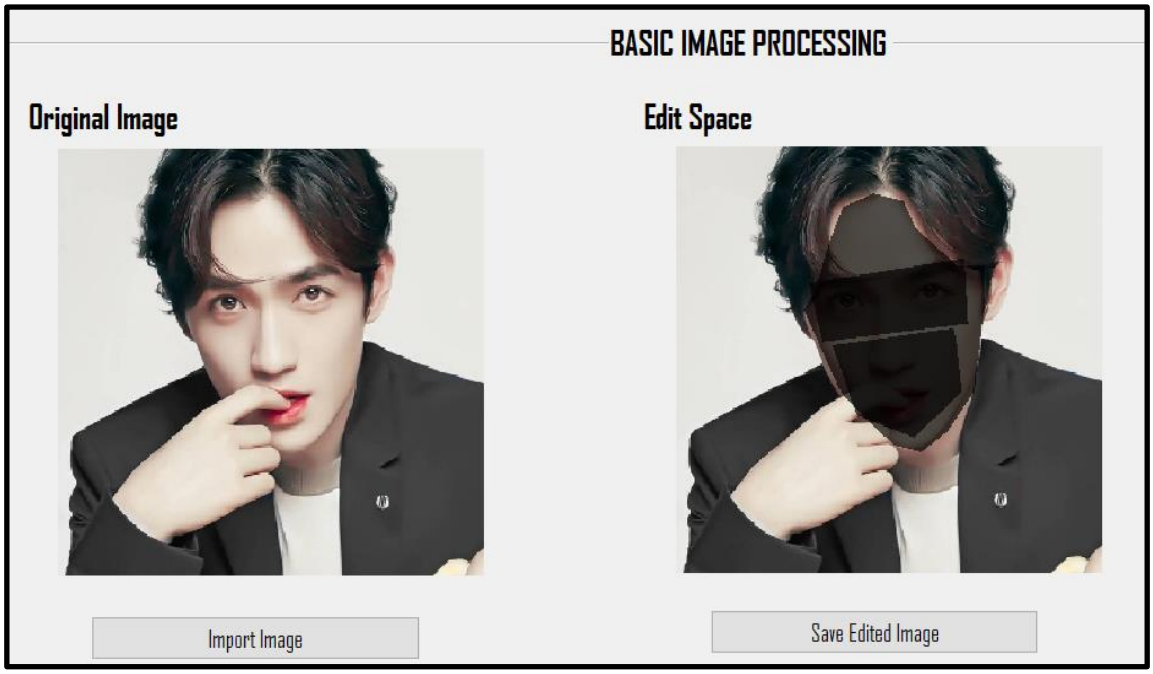

Figure 19: The 'Darken Image' effect - user can apply more than 1 times

Save darken image

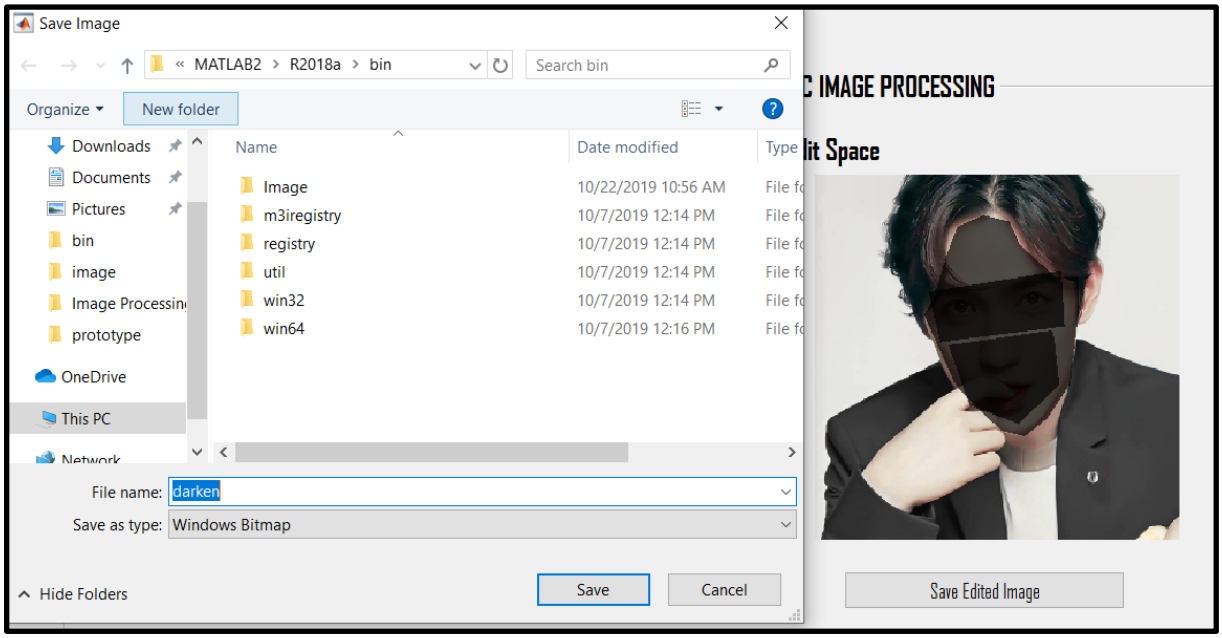

Figure 20: User can save the edited image in computer when they pressed 'Save Edited Image' button

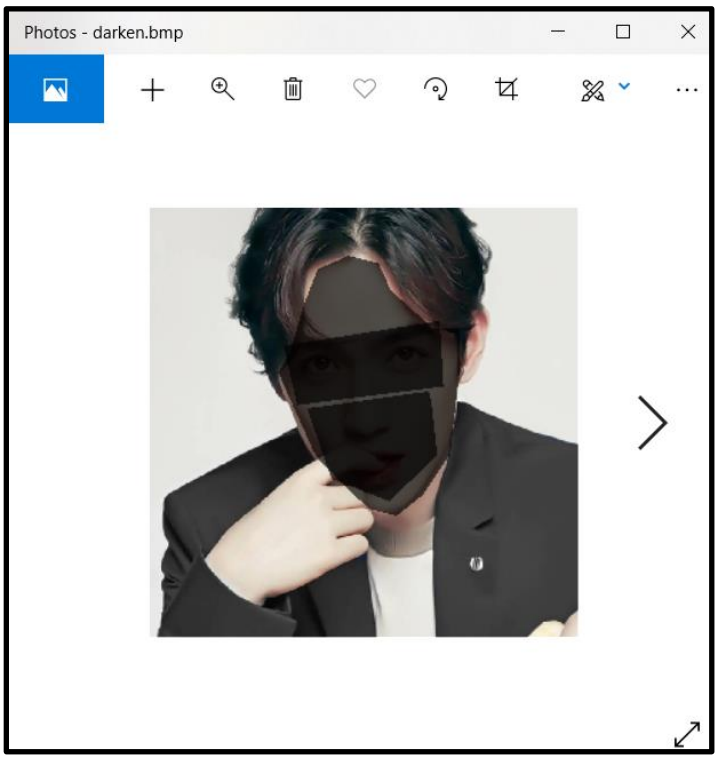

Figure 21: The saved image

Blur Image

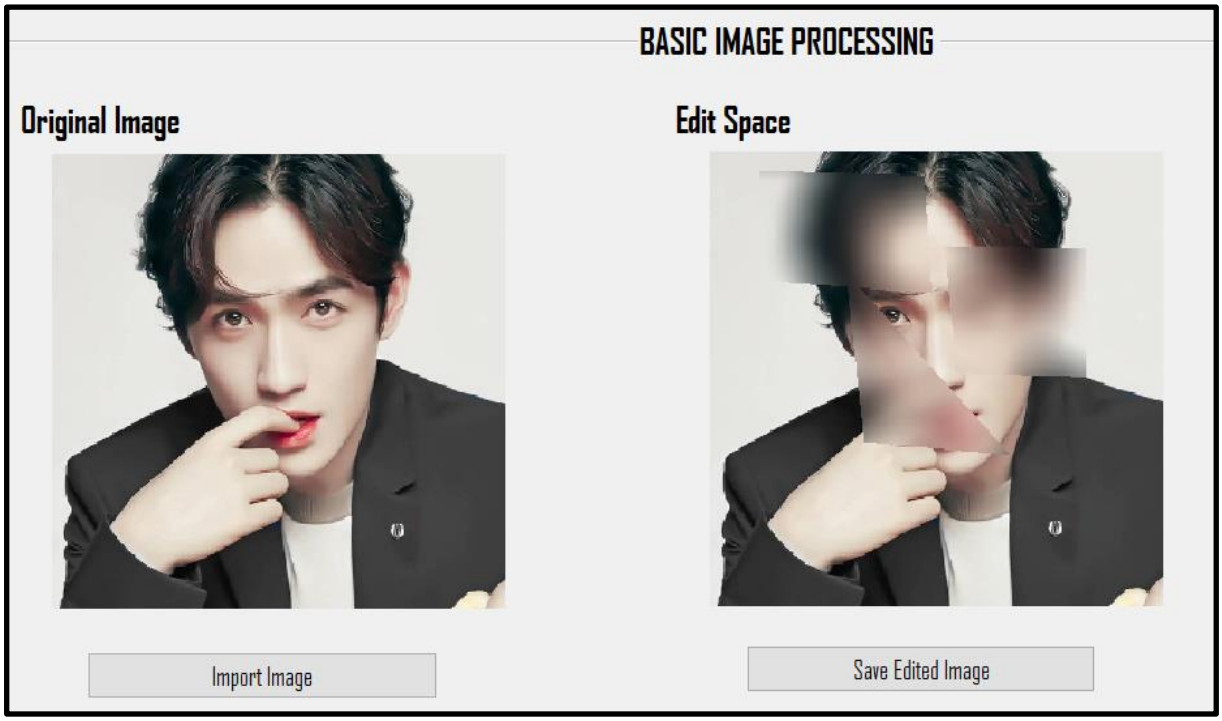

Figure 22: The 'Blur Image' effect

Save Blur Image

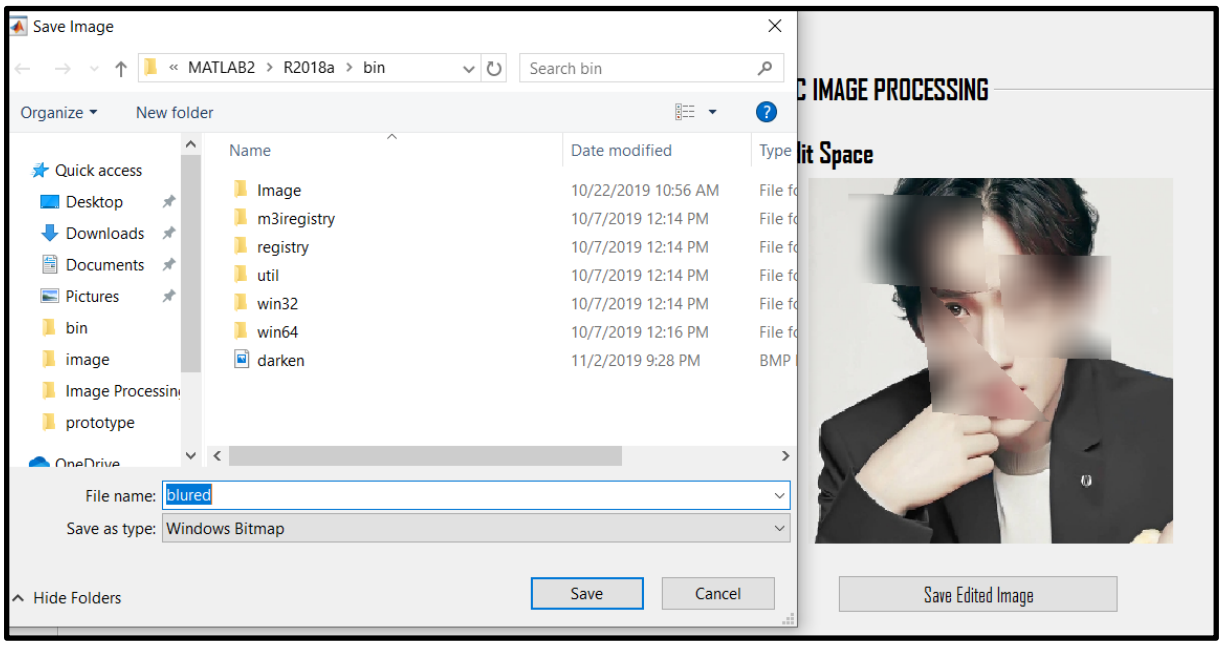

Figure 23: User can save the edited image

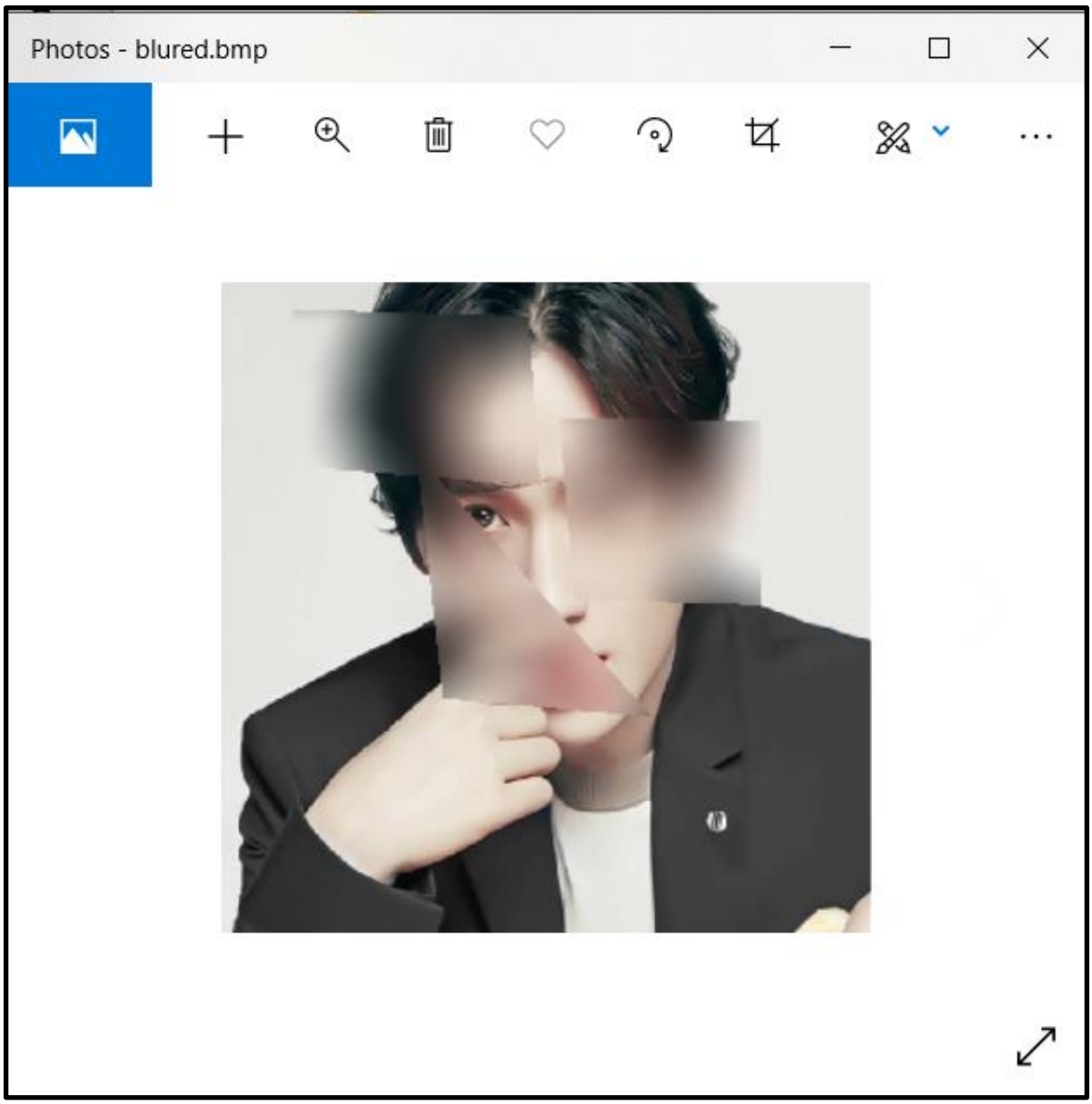

Figure 24: The saved image

Brighten Image

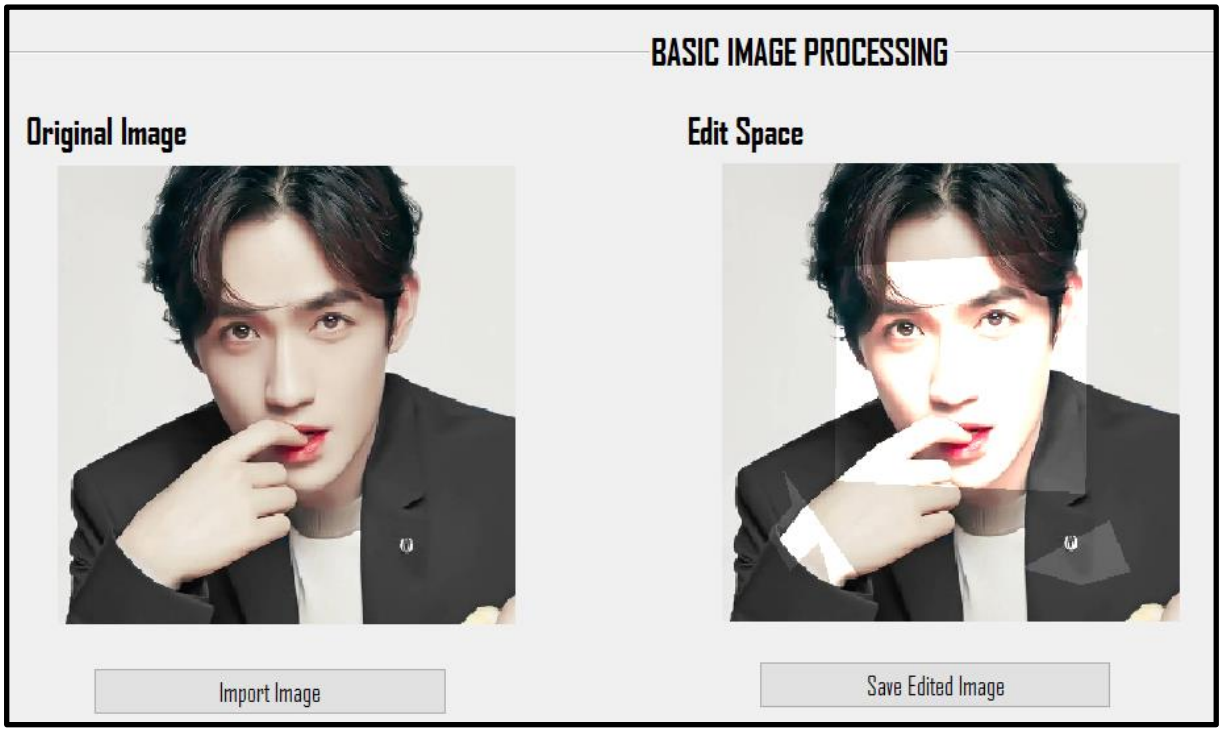

Figure 25: The 'Brighten Image' effect

Save Brighten Image

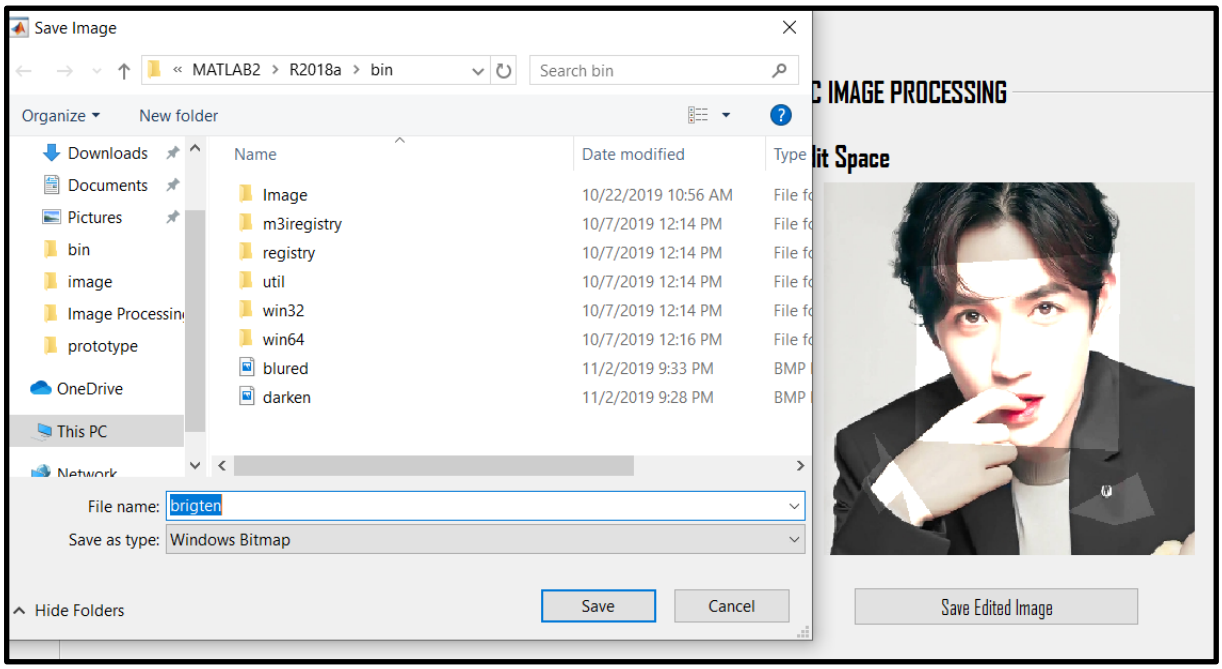

Figure 26: User can save the edited image in computer

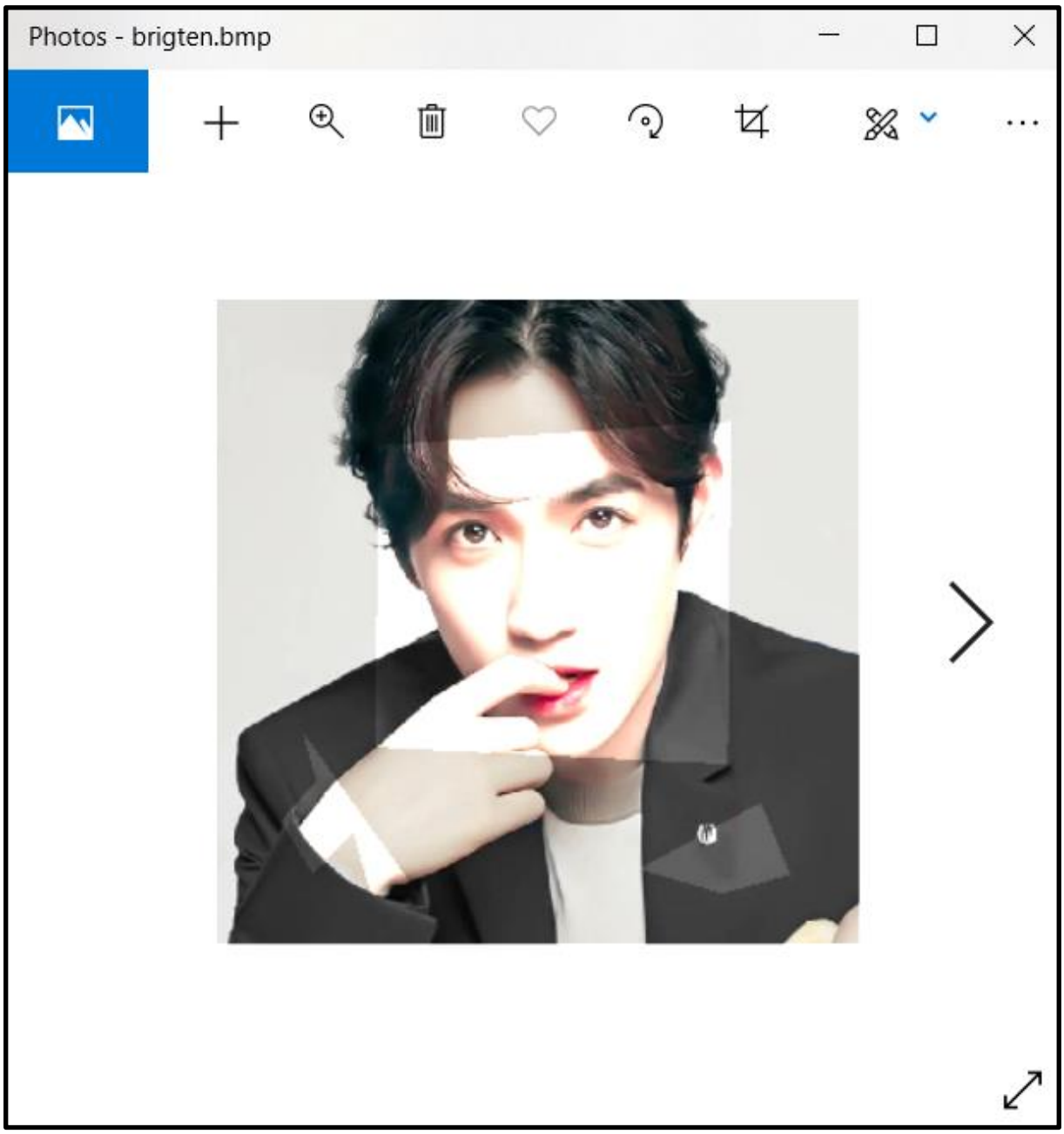

Figure 27: The saved image

Sharpen Image

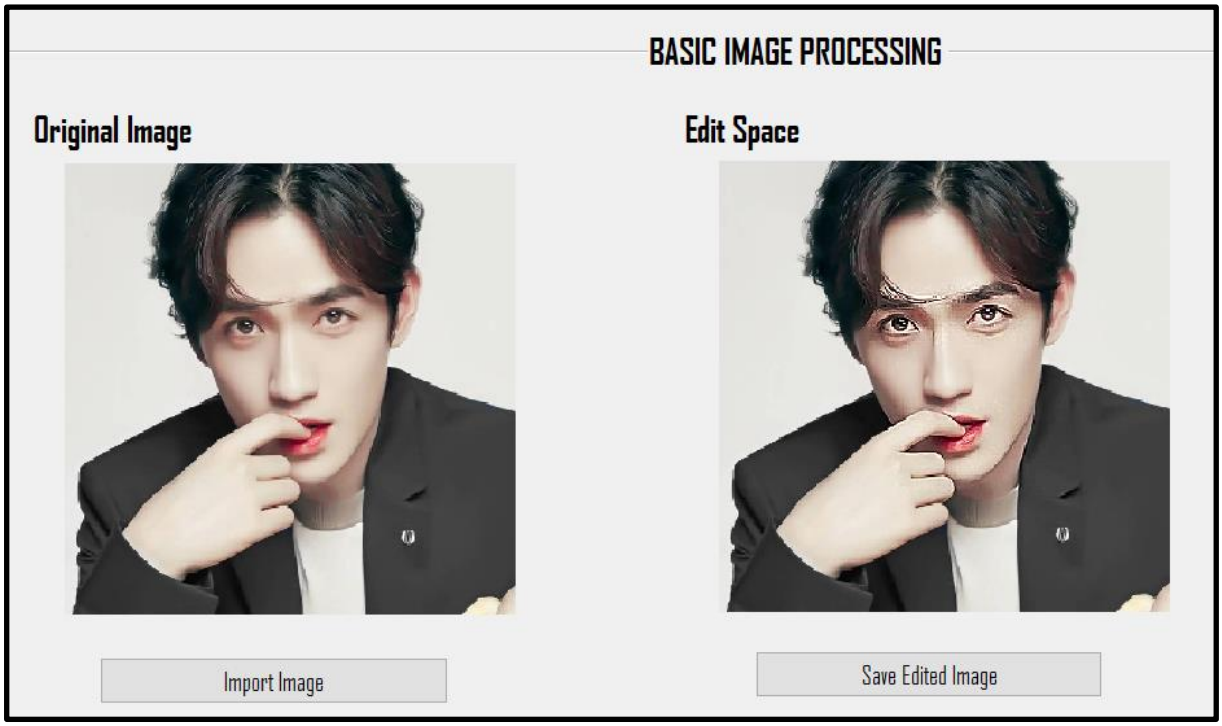

Figure 28: The 'Sharpen Image' effect

Save Sharpen Image

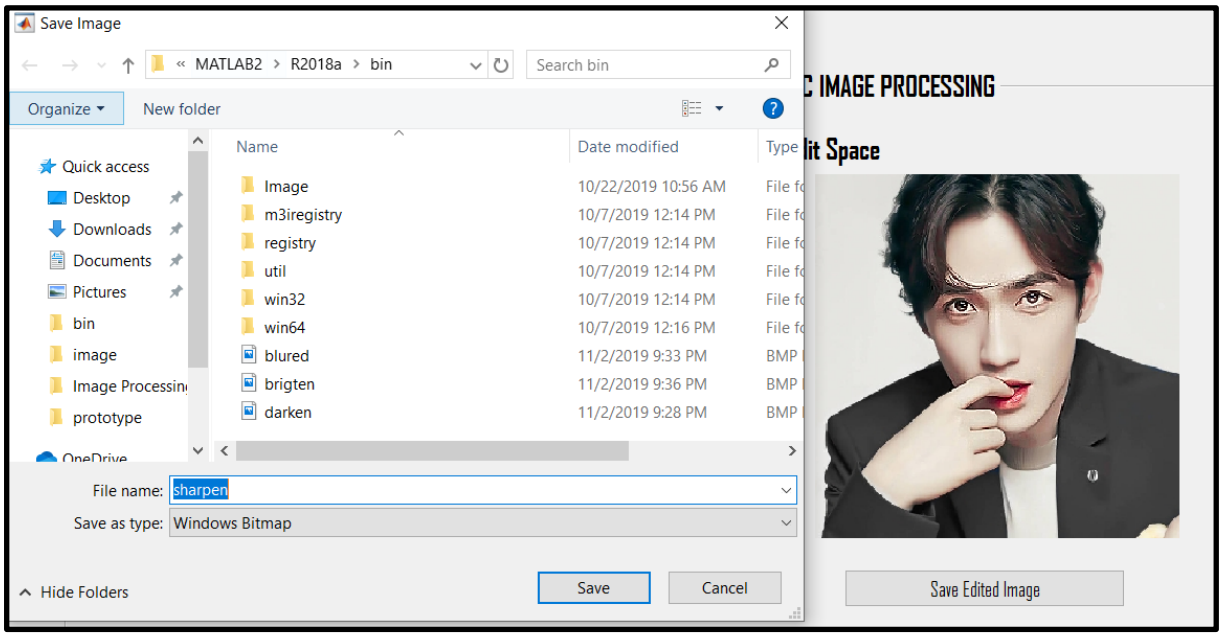

Figure 29: User can save the edited image in computer

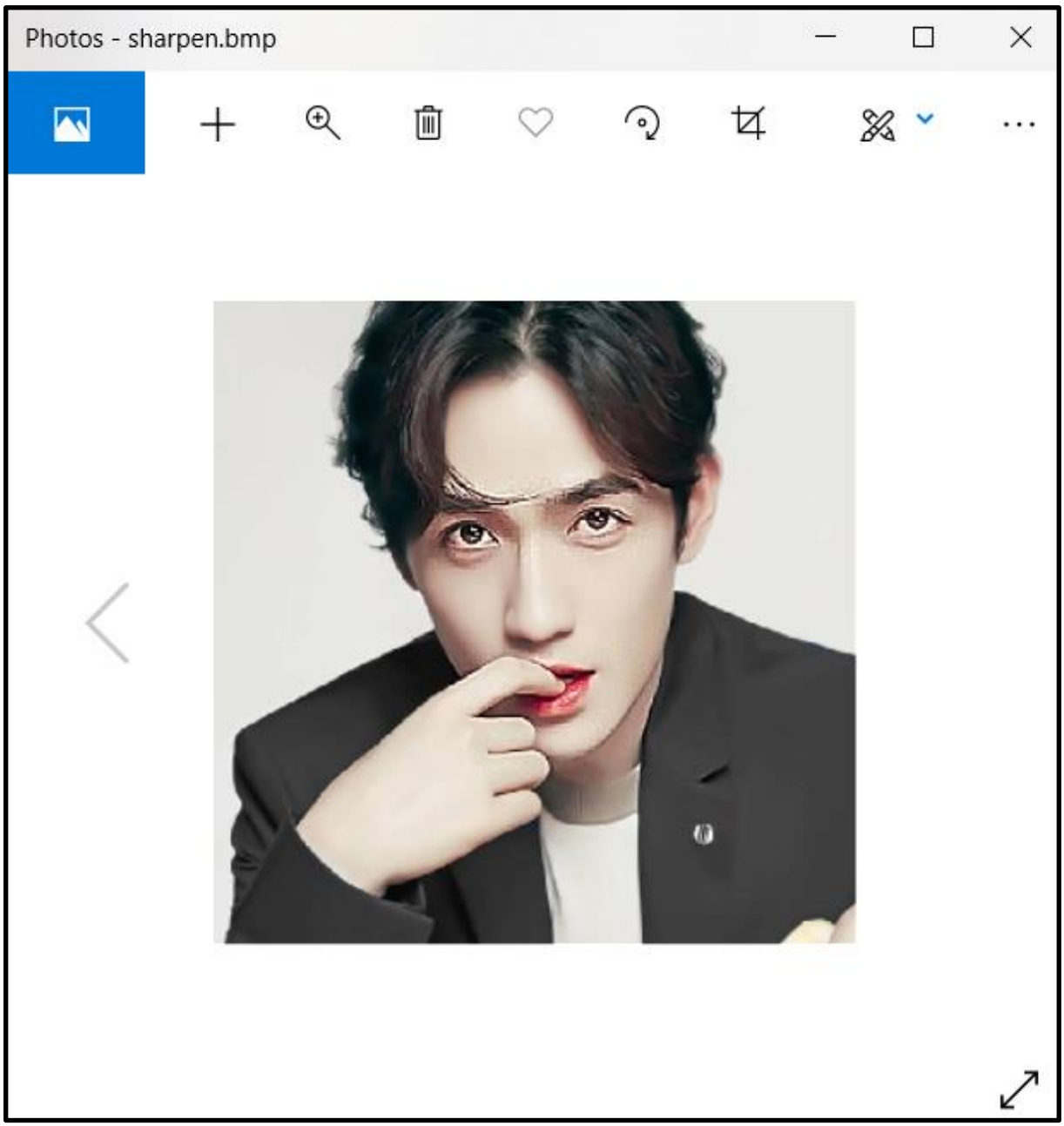

Figure 30: The saved image

Sample Output by mixing all of the Operation

Figure 31: The sample output -1

Figure 31: The sample output -2