UNIVERSITI TEKNOLOGI MALAYSIA

FACULTY OF ENGINEERING, SCHOOL OF COMPUTING

SKUDAI, 81310 JOHOR BAHRU, JOHOR DARUL TAKZIM

SEMESTER I SESSION 2020/2021

SCSV3213 - 01 FUNDAMENTAL OF IMAGE PROCESSING

ASSIGNMENT 2: IMAGE ENHANCEMENT IN SPATIAL DOMAIN

LECTURER:

DR. MD SAH HJ SALAM

PREPARED BY:

Aimi Binti Rusdi (B19EC0001)

Mirhanieza Binti Matharuzaman (A18CS0106)

Nuramyra Natasha Binti Ismalludin (B19EC0035)

Faculty of Engineering | Bachelor of Computer Science

Table of Contents

1. Introduction

For subject Fundamental Image Processing SCSV3213 section 01, we were assigned a second assignment to explore more about image enhancement in spatial domains using MatLab software. This assignment is a group task consisting of three members of each team and the task requires us to create a simple application of image processing to make the image blur or dark on the part chosen by the user.

To complete this task, there are several functions required. Among the functions that must be available in this application is that users are allowed to choose, import and export pictures from the library on their own computer. Moreover, this application needs to provide two functions that can blur and darken the image of the chosen part by using the roipoly function that was taught in the tutorial class previously under the topic of image enhancement in the spatial domain in neighborhood processing. In addition, this assignment ensures an attractive GUI or interface that enables users to communicate with computers to execute tasks offered by our application so that they can perform blurry or dark features on selected parts of the image.

2. Codes and Documentation

A. Upload Button

```
% --- Executes on button press in upload.
function upload_Callback(hObject, eventdata, handles)
% hObject    handle to upload (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[a b] = uigetfile('*.*','All Files');
img = imread([b a]);
handles.currentData = img;
imshow(handles.currentData,'Parent',handles.axes1);
imshow(handles.currentData,'Parent',handles.axes2);
% Update handles structure
guidata(hObject, handles);
```

Figure 1. Shows the code for upload button

The upload button function is used to allow users to insert their own photos to blur or darken in each selection part. We use **uigetfile** which provides a dialog box containing a list of files in the current folder. Users can select the photo file here and open the photo in our application. The **imread** function reads an image from a file entered by the user while the **imshow** displays the image using the image's data and properties.

B. Select Button

```
% --- Executes on button press in select.
function select_Callback(hObject, eventdata, handles)
% hObject    handle to select (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global bw;
global pic;
global w;

      w = 0;
      pic=handles.currentData;
      bw = im2uint8(roipoly(pic));
% Update handles structure
guidata(hObject, handles);
```

Figure 2. Shows the code for select part of image button

For filtering operations carried out which blur and darken only selected parts of the image, Region of Interest (ROI) is used. **bw = im2unit8(roipoly(pic))** shows an image selected by the user from their computer into the image window to generate an interactive polygon selection tool related to the image.

C. Select All Button

```
% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global w;

    w = 1;

% Update handles structure
guidata(hObject, handles);
```

Figure 3. Shows the code for select all part of image button

The function button is to give users the option so that they can blur or darken all parts of the picture.

D. Save Button

```
% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

    [filename, foldername] = uiputfile('*.png', 'Where do you want the file saved?');
    fname = fullfile(foldername, filename);
    imwrite(handles.currentData,fname);
% Update handles structure
guidata(hObject, handles);
```

Figure 4. Show the code for save button

The function of the saved button is to save the edited photo either using blurry, dark or both functions by the user. In this code, **[filename, foldername] = uiputfile** is used because **uiputfile** provides a dialog box to store the image in a specific folder while **[filename, foldername]** to name the file in .png and location to save the image. **imwrite(handle.currentData, fname);** creates a new file in the current folder, it will write the current image to a particular file with the name of the file and then enters the file format that is already set as png. Output of the image depends on the type and file format of current image data.

E. Blur Button

```
% --- Executes on button press in blur.
function blur_Callback(hObject, eventdata, handles)
% hObject    handle to blur (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global bw;
global pic;
global w;

try
    pic=handles.currentData;
    fn_gau = fspecial('gaussian',9,10);
    filter = imfilter(pic,fn_gau);

    if (all(bw(:)==0)   || w == 1)
        handles.currentData = filter;
        imshow(filter,'Parent',handles.axes2);

    else

        bw_cmp = bitcmp(bw);
        roi = bitor(filter,bw_cmp);
        not_roi = bitor(pic,bw);
        handles.currentData = filter;
        edited_img = bitand(roi,not_roi);
        handles.currentData = edited_img;
        imshow(edited_img,'Parent',handles.axes2);

   end
```

Figure 5. Shows the code for blur button

Gaussian low pass filters have been used for this application for blurred operation. Gaussian filters can blur images, reduce noise and decrease the contrast in photos. **fn_gau = fspecial ('gaussian',9,10);** for the calculation of low-pass symmetrical Gaussin rotation sampling, the value of 9 is the size of the filter and 10 is the standard deviation sigma value. In the code we use if..else operation, if the user chooses all parts of the image that are the same as **w = 1**, they will blur the whole parts of the image, otherwise else will be execute where **w != 1** then results in where the user selects the part they choose to blur, then a roipoly operation will be performed and chosen parts of the image will be blurred.

F. Darken Button

```
% --- Executes on button press in darken.
function darken_Callback(hObject, eventdata, handles)
  global bw;
  global w;
  global pic;
%TF = isempty(bw); %check if roi is selected

    try
    pic=handles.currentData;
    darken = imsubtract(pic,10);

    if( all(bw(:)==0) || w == 1)
        handles.currentData = darken;
        imshow(darken,'Parent',handles.axes2);
    else
        bw_cmp = bitcmp(bw);
        roi = bitor(darken,bw_cmp);
        not_roi = bitor(pic,bw);
        handles.currentData = darken;
        new_img = bitand(roi,not_roi);
        handles.currentData = new_img;
        imshow(new_img,'Parent',handles.axes2);

    ends
```

Figure 6. Shows the code for darken button

We use Arithmetic operation, subtraction to darken the picture. To perform dark operation in our application, this code fragment will reduce the pic value to 10, **darken = imsubtract(pic,10);** Therefore, when we click the dark button each time, the order of the image values will be reduced by 10 each time we click. We also use the operation if..else for darken, if the user selects all parts of the picture equal to **w = 1**, they will darken the whole part of the picture, otherwise it will be done where **w != 1**, then the result where the user selects the part they selected becomes dark, then the roipoly operation will be performed and the selected part of the images will be dark.

G. If else

```
bw_cmp = bitcmp(bw);
roi = bitor(filter,bw_cmp);
not_roi = bitor(pic,bw);
handles.currentData = filter;
edited_img = bitand(roi,not_roi);
handles.currentData = edited_img;
imshow(edited_img,'Parent',handles.axes2);
```

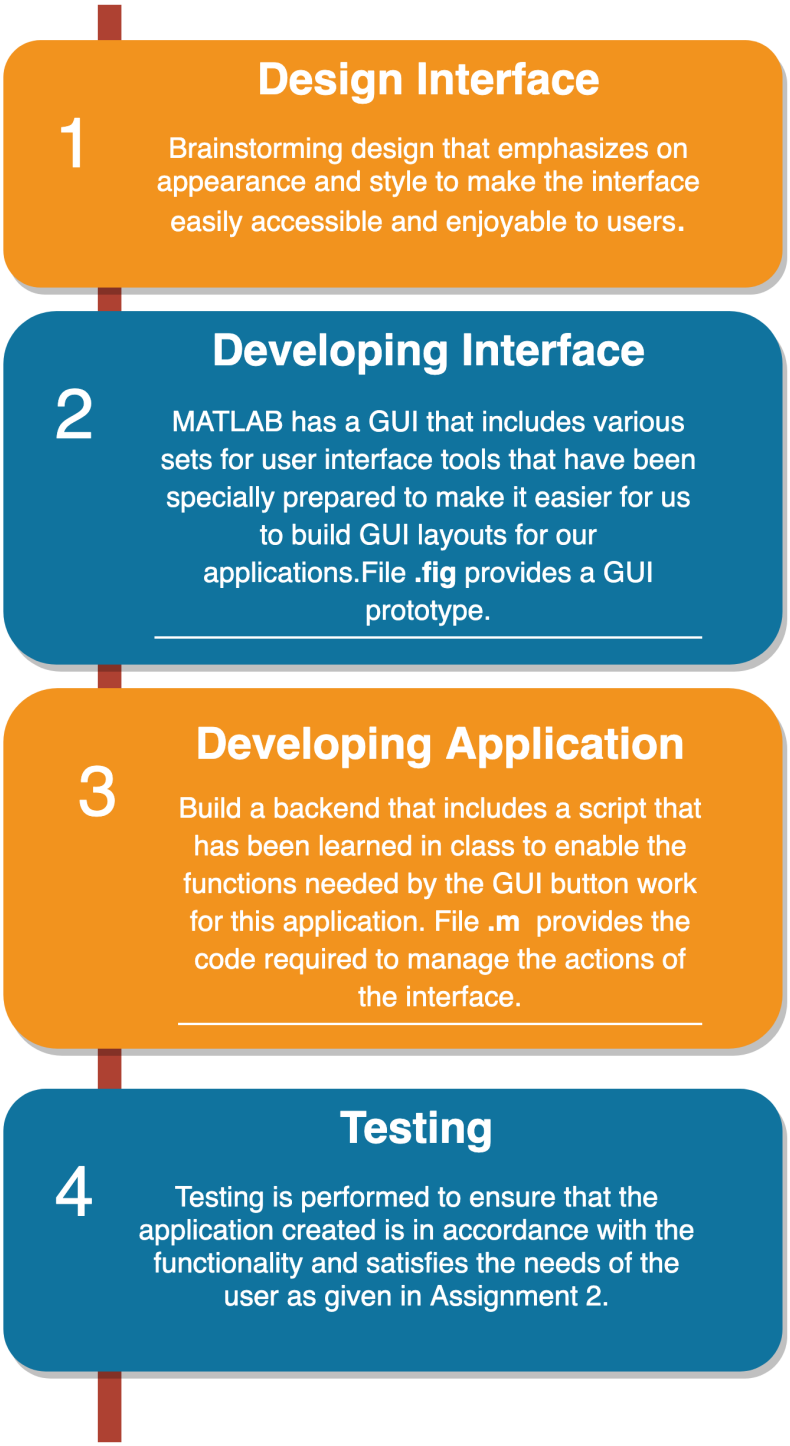
Figure 7. Shows the code for selected part operation

This is coding where the operation of the logic operator occurs to show a blurry and dark image only on the selected part which is the region of interest (ROI), handles.axes1 will show the original picture while handles.axes2 will show the edited picture.

bw_cmp = bitcmp(bw); this is where the bitwise complement is used to return the complement of the bitwise image. Next we have **roi = bitor(filter, bw_cmp);** applies the dark filter to the user-selected region of interest (ROI) while **not_roi = bitor(image,bw);** is where not part of the selected region. The **edited_img = bitand(roi, not_roi);** generates the newly edited image by combining the **roi** and **not_roi**. Using this code, **handles.currentData = edited_img;** executes the newly edited image that was edited and shows the image in **imshow(edited_img,'Parent',handles.axes2);**

3. Flow and Process Chart

Here is the explanation of the processing in achieving the task.



4. Example of the Image and Output

Here is an image that we used to demonstrate our application.

Once we have performed the operation found in our application that is blurry and dark on the selected part, here is the outcome.

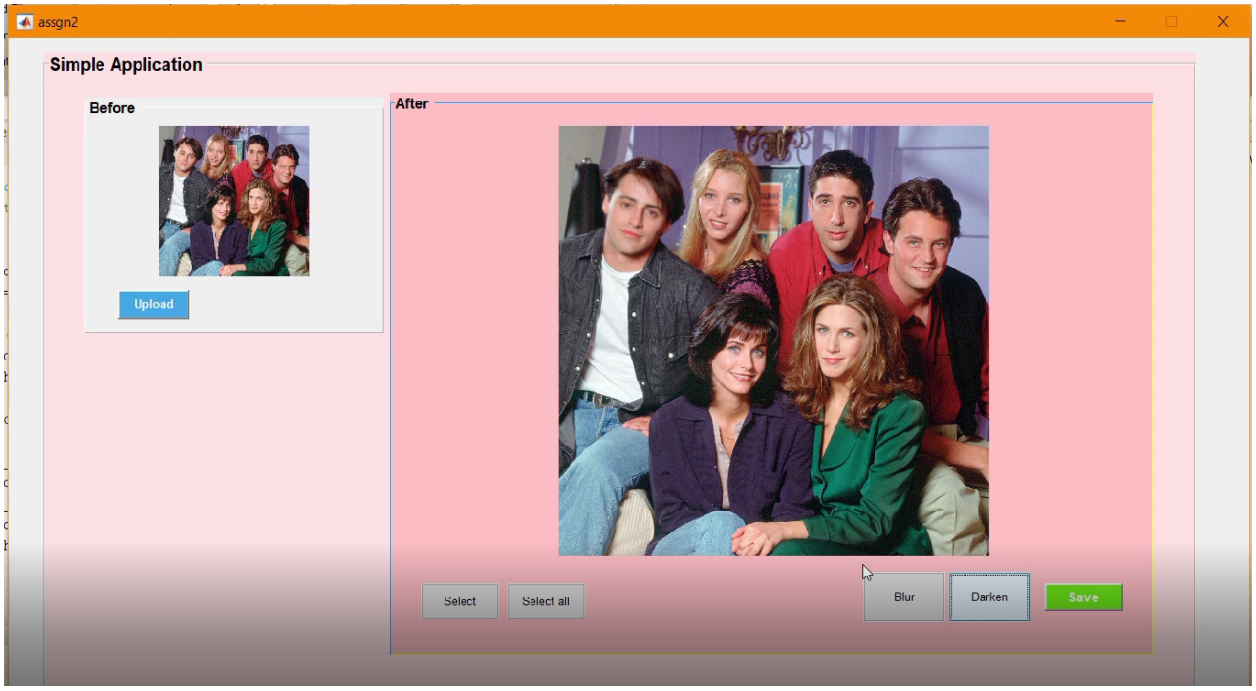


Figure 8. Shows the original image.

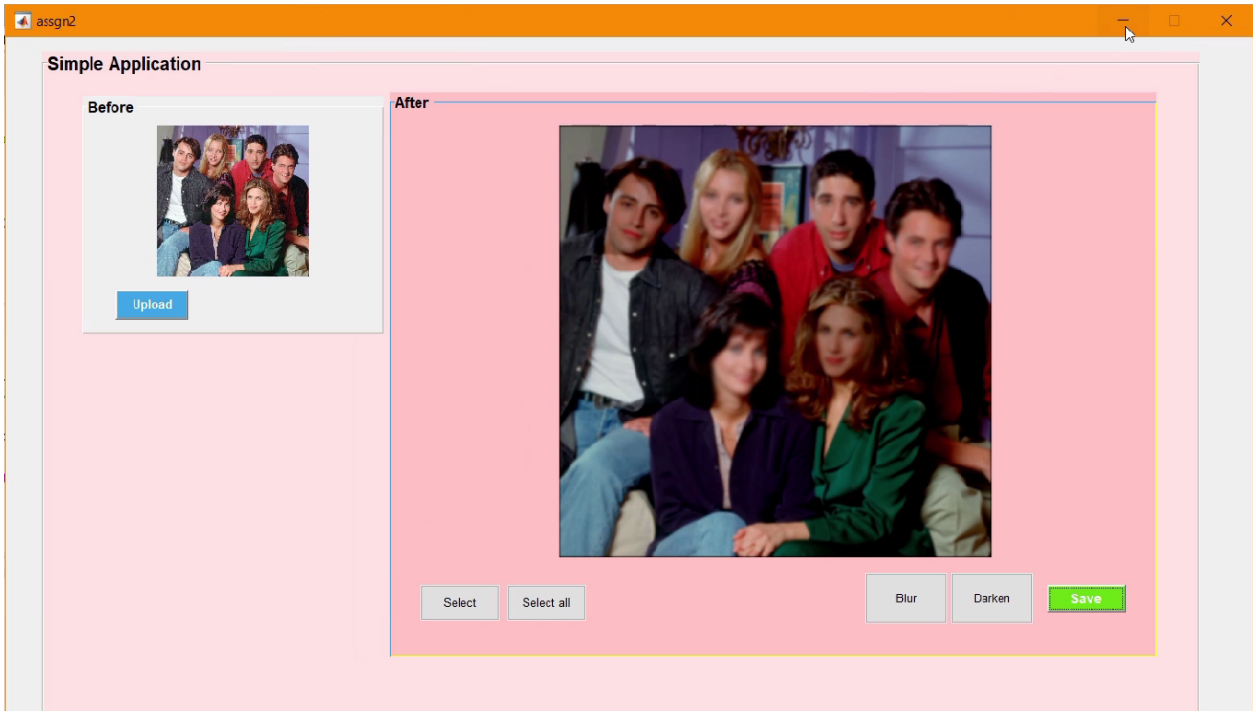


Figure 9. Shows the image after blurring and darken.

5. Video Explanation Link

Here is the attachment of our video link in Youtube

➔ **https://www.youtube.com/watch?v=zFPpZ-0abec&feature=youtu.be**