EE 440 Autumn 2018

Final Project: Image Artistic Stylization

Professor:Ming-Ting Sun

December 8th. 2018

He Feng 1427841

Introduction

This is the final project report of the EE 440 in University of Washington. I aimed to develop two artistics effects from real photos. I also developed a simple graphic user interface to load the image, apply artistic effects and display the input and output separately. The artistic effects I developed are Oil Painting Effect and Emboss Effect. The oil painting effect can cause the input color image turn into the oil painting like image, and the emboss effect can make the input color image into the lithograph format. There are three scripts in my project, the first one demonstrate the functions of the the simple graphic user interface, the second one gives a function to generate a oil painting image, and the third one is also a function which can generate a litho format image.

Details of the Algorithm

To develop the graphic user interface, I used the GUI operation introduction and the sample script on canvas to build it. I also add a function to generate the pop menu in the interface. The link will be shown in the reference part, and by using this graphic user interface I can load a RGB image as input, and output an oil painting image or emboss image by choosing the option in the pop menu bar. In the process of adding the pop menu function into the graphic user interface, I need to set the options in the callback function, and then I use the command

```
set(hObject,'String',{'Oil Painting Effect';'Emboss Effect'});
```

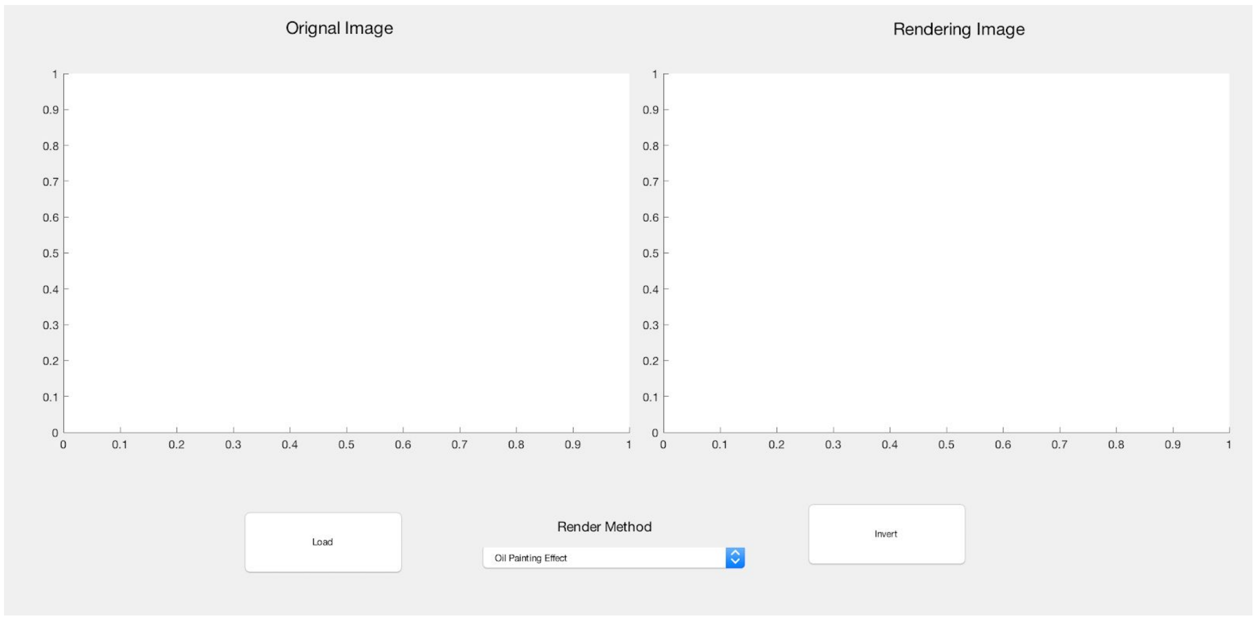
In this case we can set up the pop menu bar in the graphic user interface, and we can start the process by choosing the options in the pop menu bar.

To develop the oil painting function, initially I determine the size of each dimension for the image by using the size function, then I used im2double function to change the original image to type double. To generate the output variable im, I set all its elements as one for the future operations. The general thinking for the oil painting effect is to generate each pixel one by one, and then I need to consider about the pixels around this specific pixel during the processing. In this case, I set the radius two, which means we need to consider about the pixels within this area. Then I set the intensity level as ten which will be used in the comparing procedure in the following procedures. To generate the pixels one by one, I used two for loops to go through each pixel individually. For each trail, I initially define the target area for this pixel, which is a squared area with 5*5 pixel size and the specific pixel stays at the center place. Then I use the mean of the target area output with different dimension to time the intensity level and get the intensity area. The intensity area has the same size as the previous target area. Later I used uint8 function to transfer the element in that matrix into uint8 type.After this step, I used the mode function to find the intensity that appear the maximum time. Then I determine each pixel in the single target range one by one, if the intensity value of the element in the previous matrix is equal to the intensity which appears the most frequently, I add the maximum count number by one, and I change the each dimension of the new output image by the target area values. No matter which layer of the RGB image is, I generate the output image from the previous step by times to 255 and divided by maximum counting number to get the final output value for this specific pixel. The last step is to convert the image into the uint8 type and this will be the final output image in the graphic user interface.

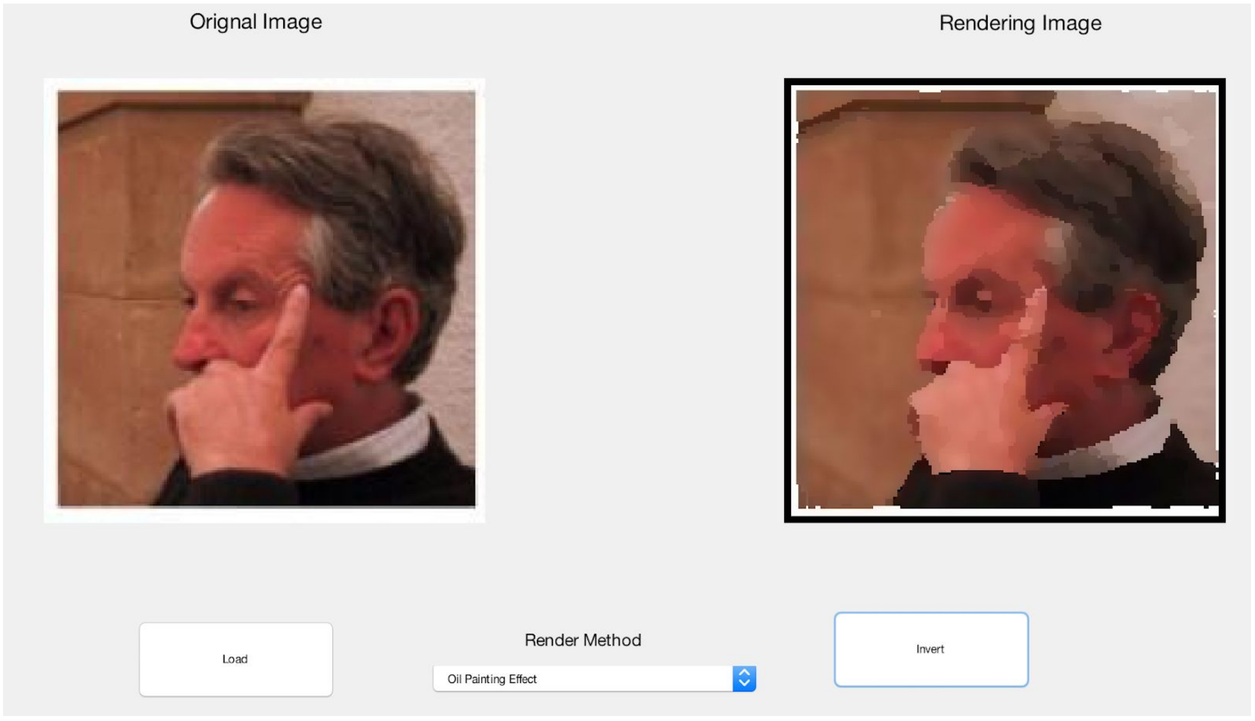
To develop the emboss effect, this one is much easier than the previous effect. Initially I determine the input image size and the input image dimension. If the dimension is three, then I convert the input image into the gray scale image. Later I add multiplicative noise with variance 0.01 to the gray scaled input image, and I transfer the add noise image to type double. Then I created a predefined 2-D filter by using the fspecial function, and I applied the generated impulse response with the double type add noise image by using the filter2 function. Finally In convert the matrix to grayscale, which will be the output of the original image.

Instruction about the Functionalities

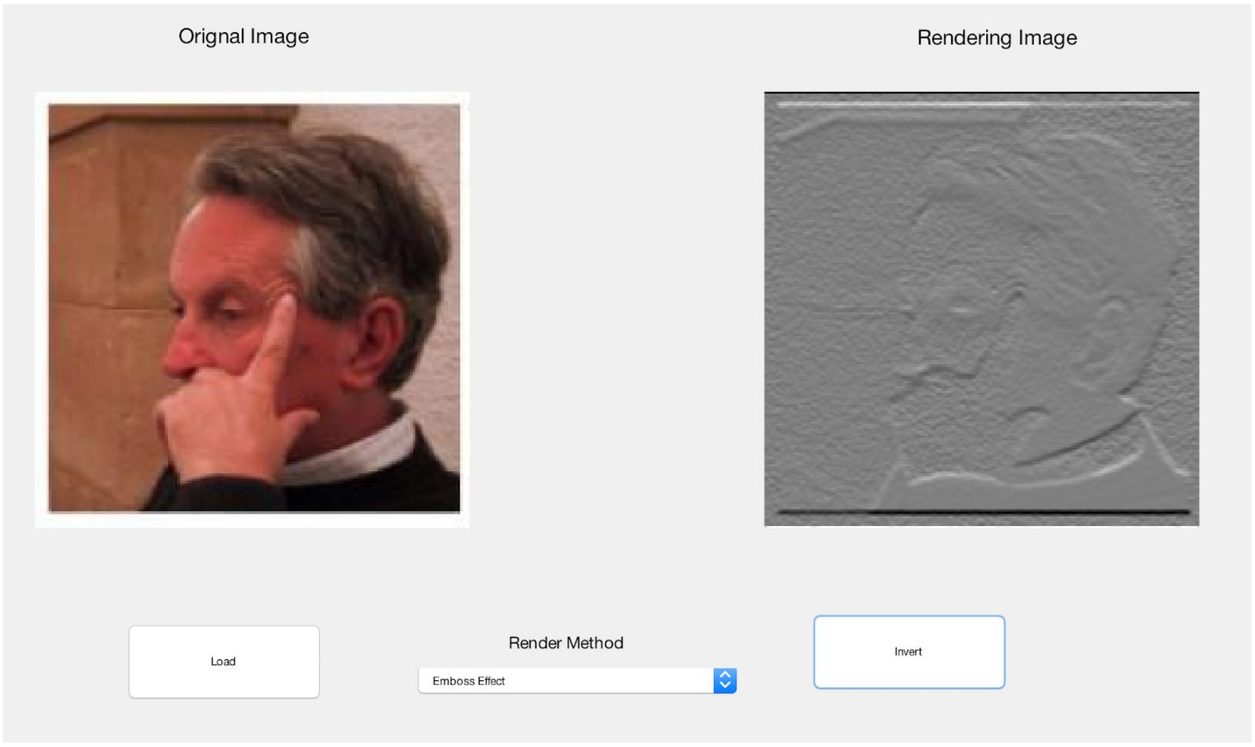
To run the code, we need to run in the demo script, and the graphic user interface will come out:



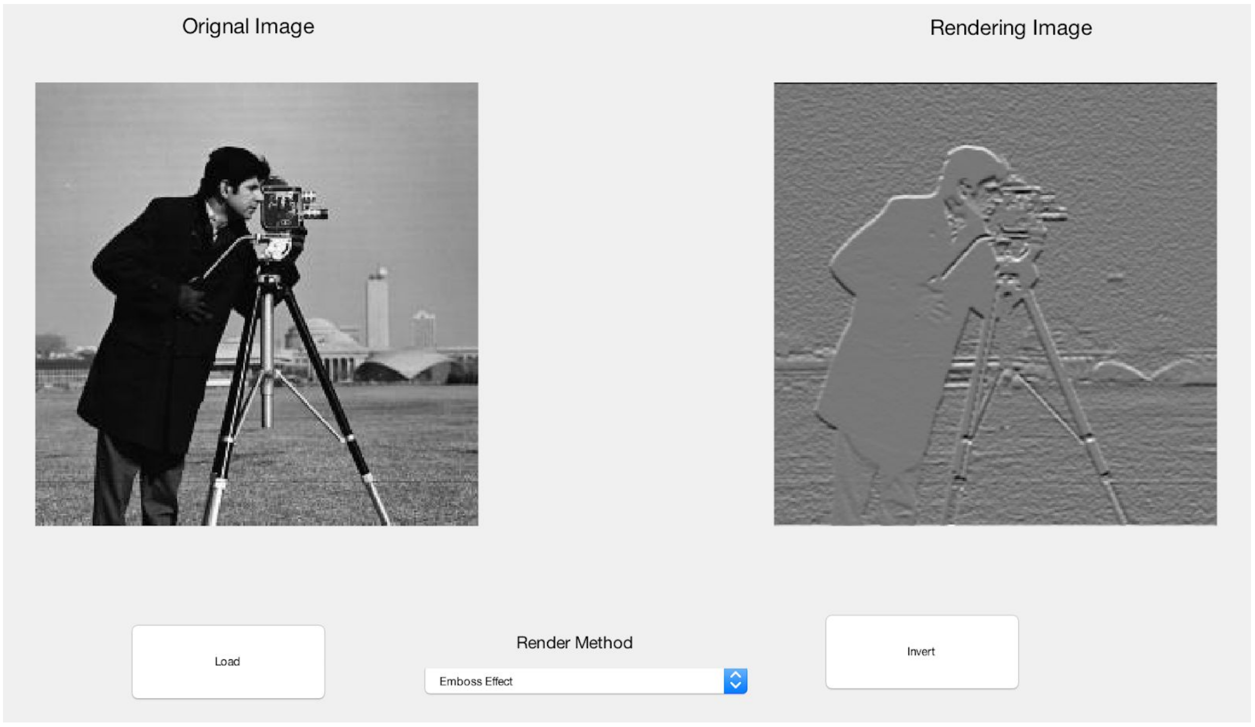
As the graph shown, we need to choose the effect from the render method, and load the image by clicking the load button on the left bottom corner. To generate the oil painting effect, the output is:

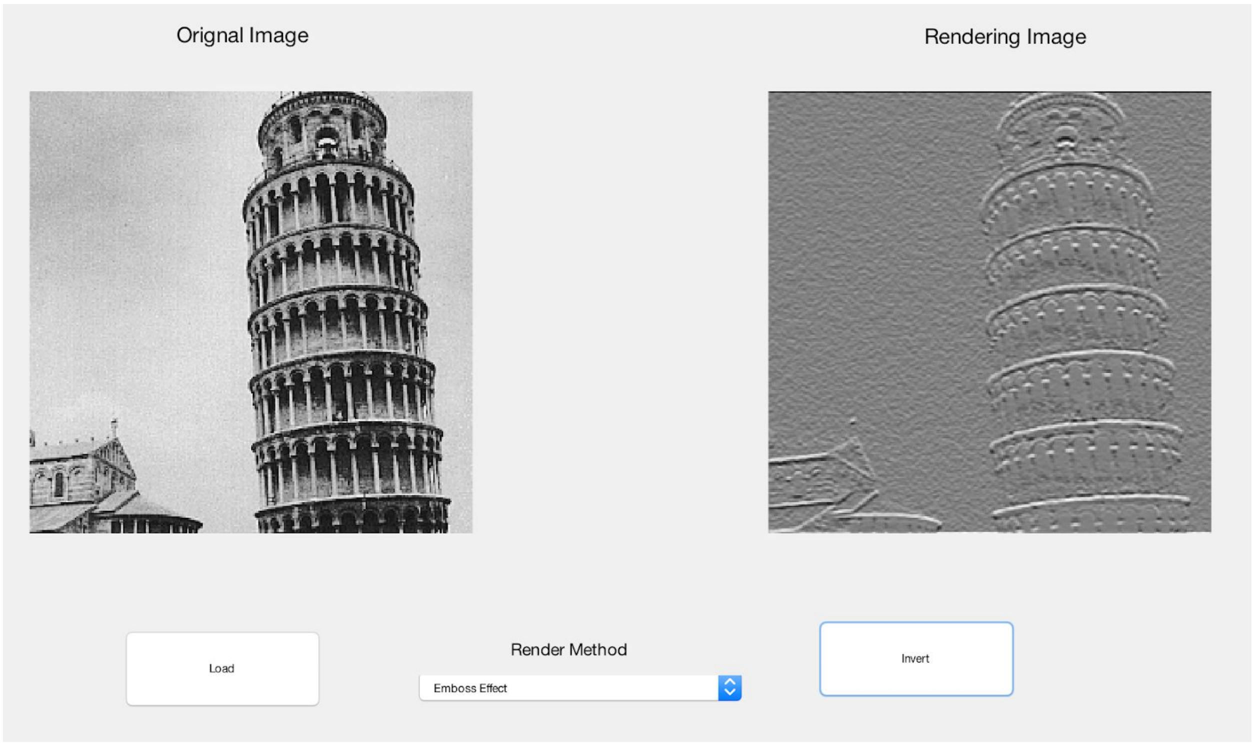


To generate the emboss effect, I used the same input image, and the output is:

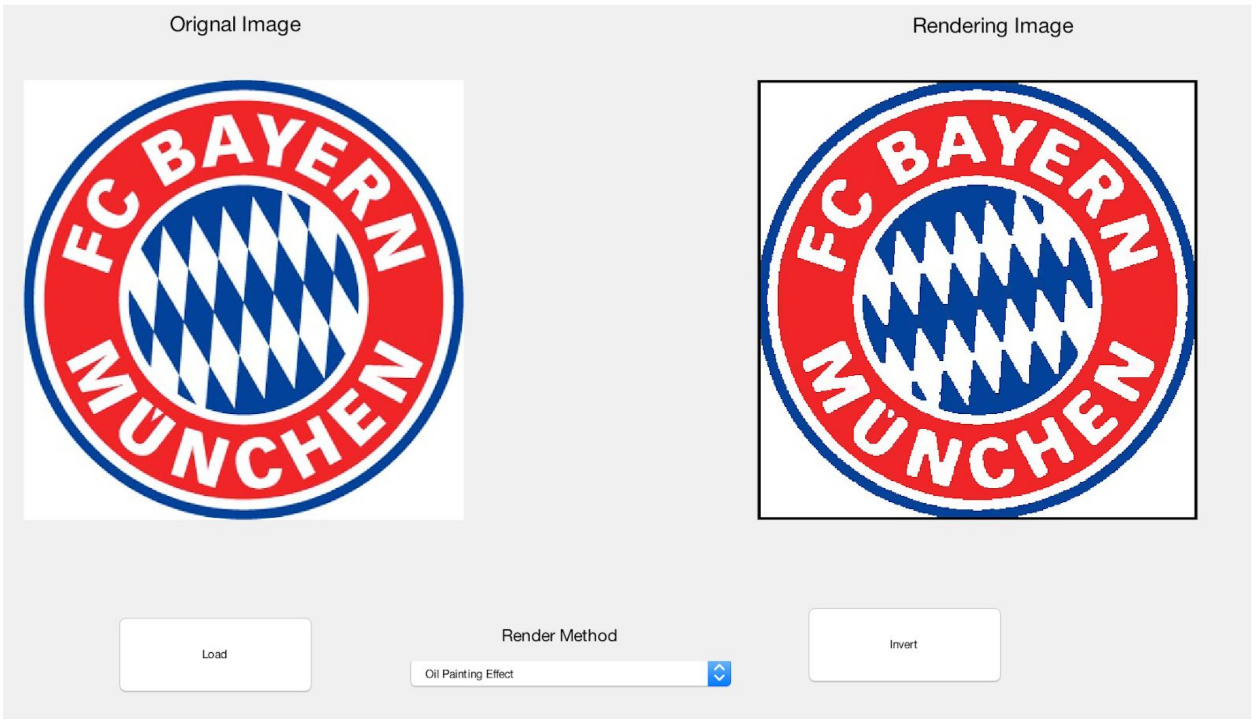


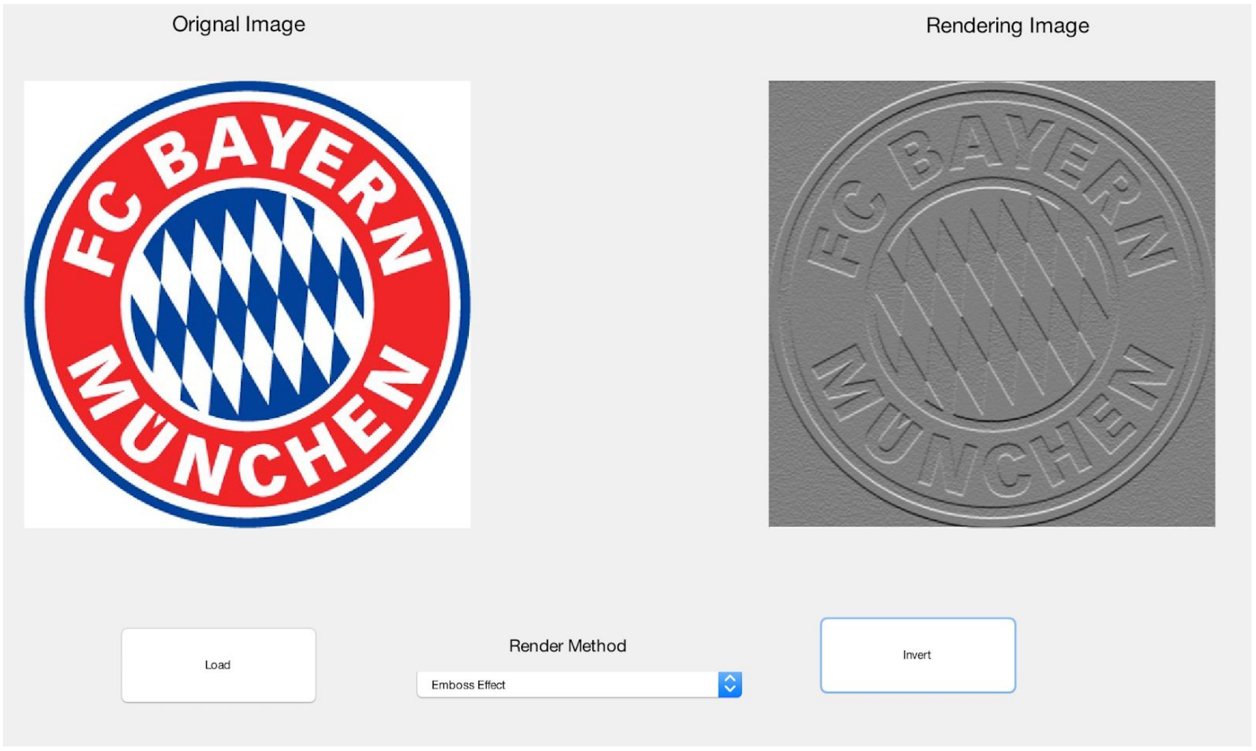
The oil painting effect can only applied by the color image, but we can input a gray scale image to the emboss effect. The output is:

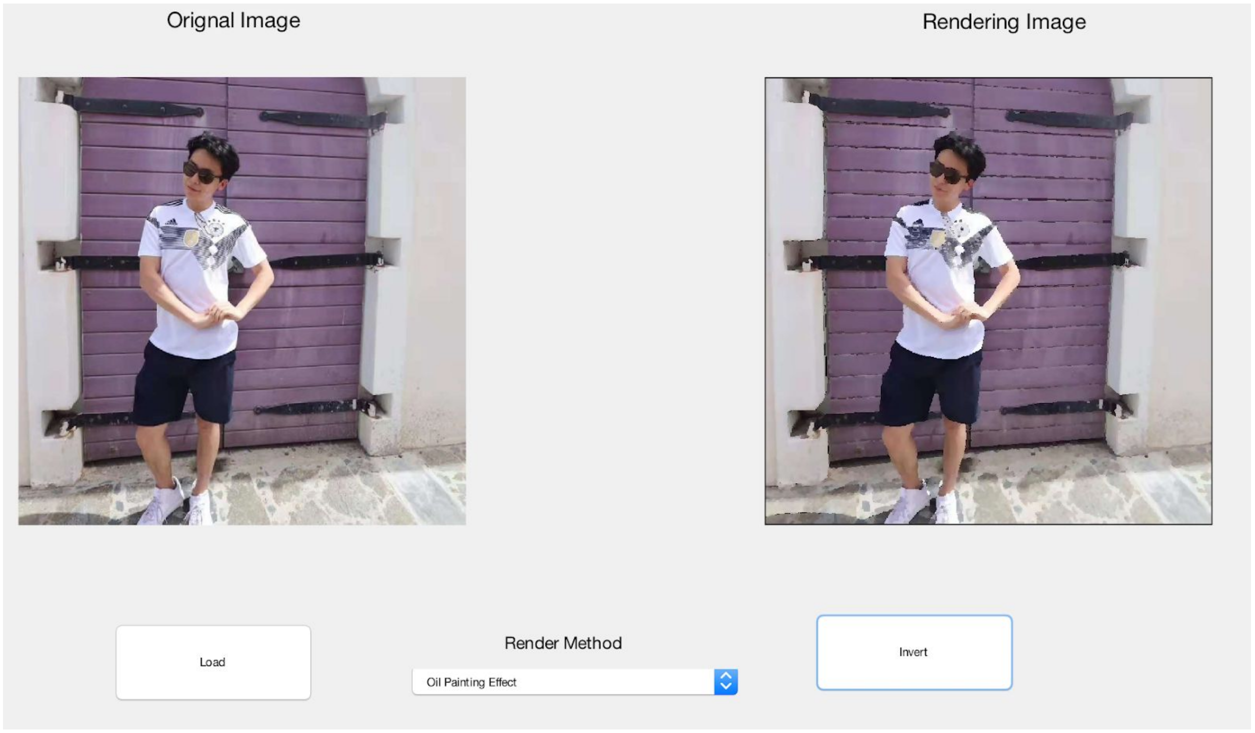




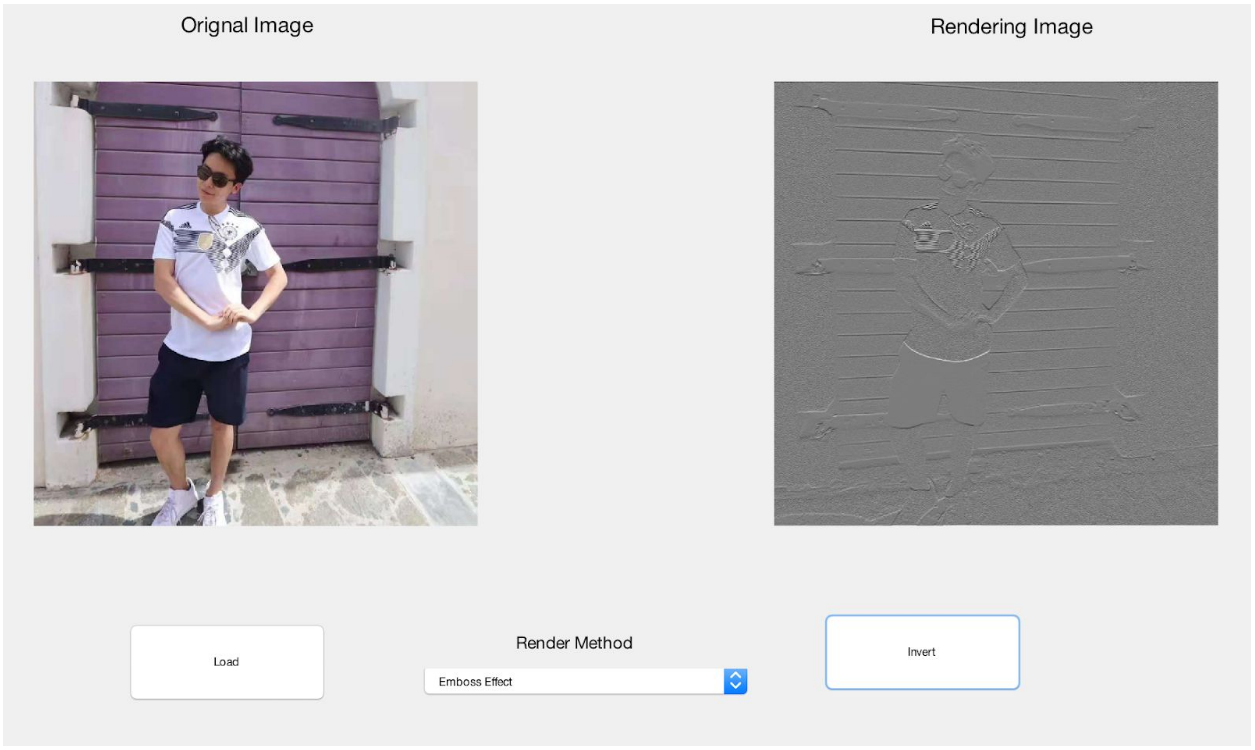
The output of other images are shown below:







Render Method: Oil Painting Effect



Render Method: Emboss Effect

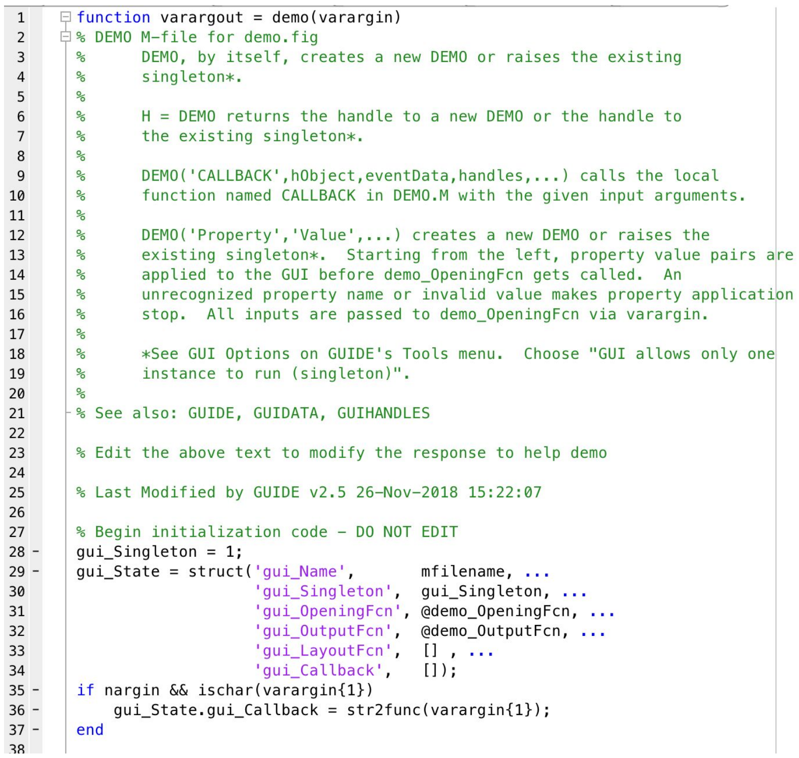
Conclusion

From the output images above, we can see this interface can generate a color image by using oil painting effects and emboss effect. The interface can also apply the emboss effect to a grayscale image. When running the program, the image with oil painting effect need to some time to generate because its program is a little bit complicated. Also, if you close the interface and run again, you might meet the situation that the output is from the previous operation. If you meet this situation, please change the effect from the render method bar and re-run it.

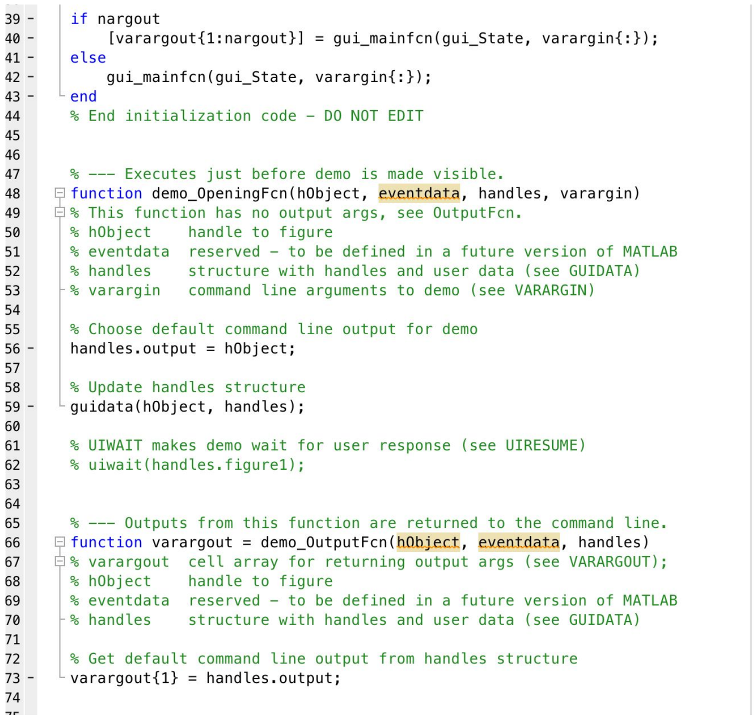
I have learned a lot from this class, and this project is the stage to prove the thing I learned. I learned how to implement the graphic user interface, and I know how to add the oil painting effect and emboss effect to an image. The most important to be is this project improve my critical thinking, and I can use the knowledge I learned before to generate some new artistic even though the algorithm of the new artistic is not that hard. I learned a lot from this class. In the future, I will try to implement more effects, and I want to combine some effect together to build some innovative and interesting algorithm. For example, I can combine the oil painting effect and emboss effect to make an oil painting image with emboss effect. The graphic user interface will be very useful in the future learning. I can add more functions or artistics in it to make it comprehensive.

Code

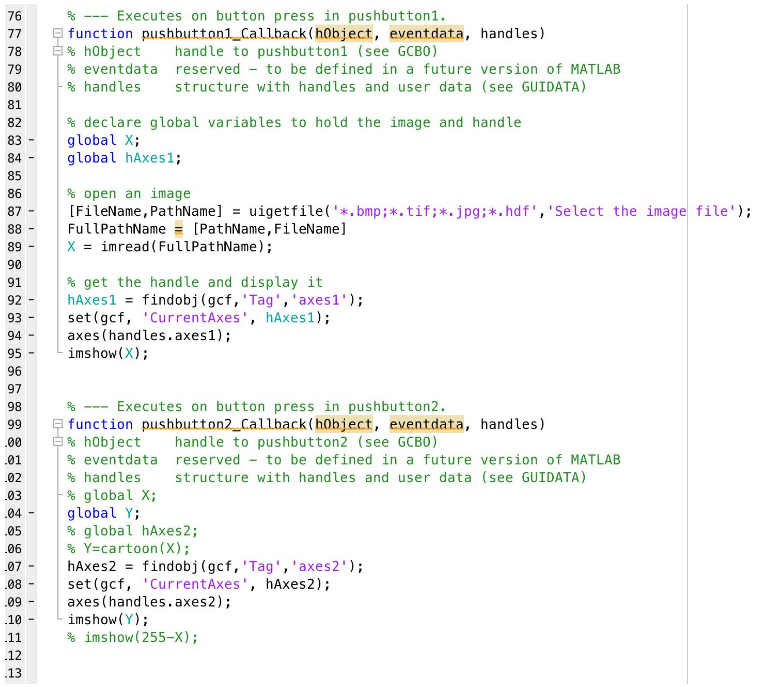
__Demo.m:__



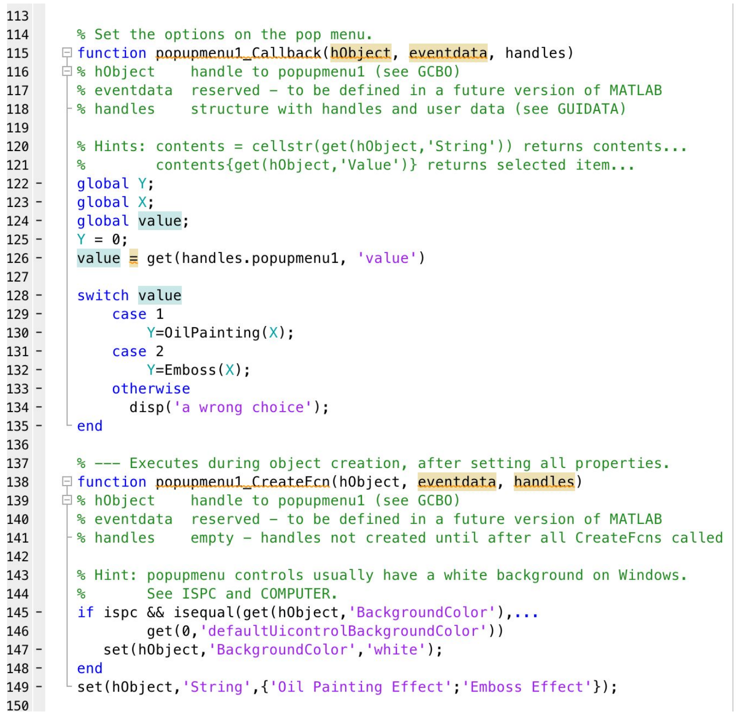
```matlab
function varargout = demo(varargin)
% DEMO M-file for demo.fig
%      DEMO, by itself, creates a new DEMO or raises the existing
%      singleton*.
%
%      H = DEMO returns the handle to a new DEMO or the handle to
%      the existing singleton*.
%
%      DEMO('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in DEMO.M with the given input arguments.
%
%      DEMO('Property','Value',...) creates a new DEMO or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before demo_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to demo_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 demo

% Last Modified by GUIDE v2.5 26-Nov-2018 15:22:07

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

```



```matlab
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 demo is made visible.
function demo_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 demo (see VARARGIN)

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

% Update handles structure
guidata(hObject, handles);

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


% --- Outputs from this function are returned to the command line.
function varargout = demo_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;

```

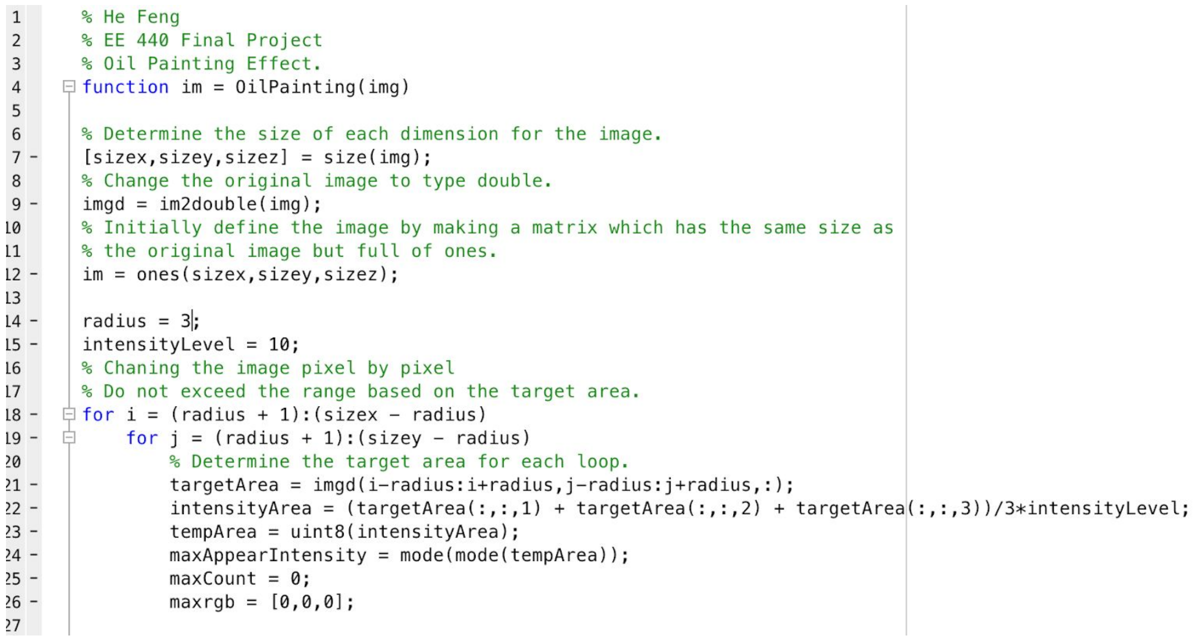


```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% declare global variables to hold the image and handle
global X;
global hAxes1;

% open an image
[FileName,PathName] = uigetfile('*.bmp;*.tif;*.jpg;*.hdf','Select the image file');
FullPathName = [PathName,FileName]
X = imread(FullPathName);

% get the handle and display it
hAxes1 = findobj(gcf,'Tag','axes1');
set(gcf, 'CurrentAxes', hAxes1);
axes(handles.axes1);
imshow(X);


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% global X;
global Y;
% global hAxes2;
% Y=cartoon(X);
hAxes2 = findobj(gcf,'Tag','axes2');
set(gcf, 'CurrentAxes', hAxes2);
axes(handles.axes2);
imshow(Y);
% imshow(255-X);
```

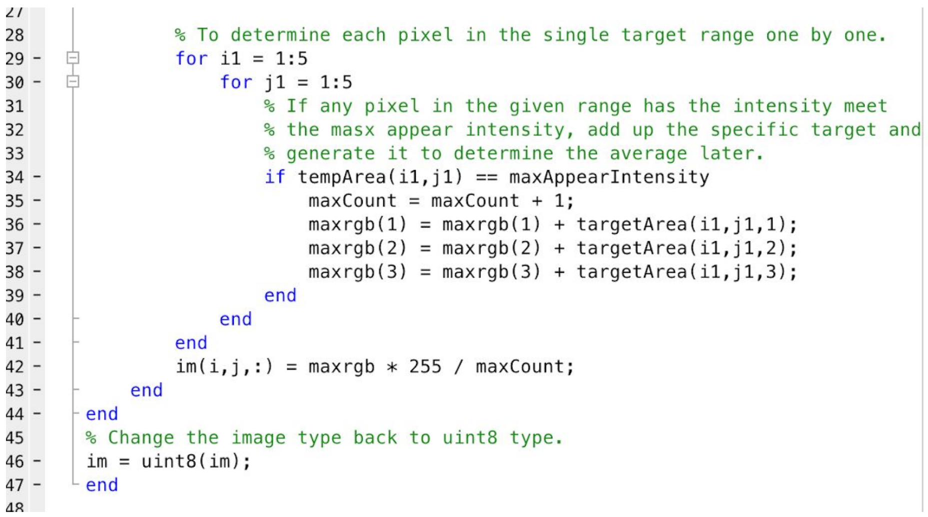


```
% Set the options on the pop menu.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns contents...
%        contents{get(hObject,'Value')} returns selected item...
global Y;
global X;
global value;
Y = 0;
value = get(handles.popupmenu1, 'value')

switch value
    case 1
        Y=OilPainting(X);
    case 2
        Y=Emboss(X);
    otherwise
      disp('a wrong choice');
end

% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (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
set(hObject,'String',{'Oil Painting Effect';'Emboss Effect'});
```

OilPainting.m:

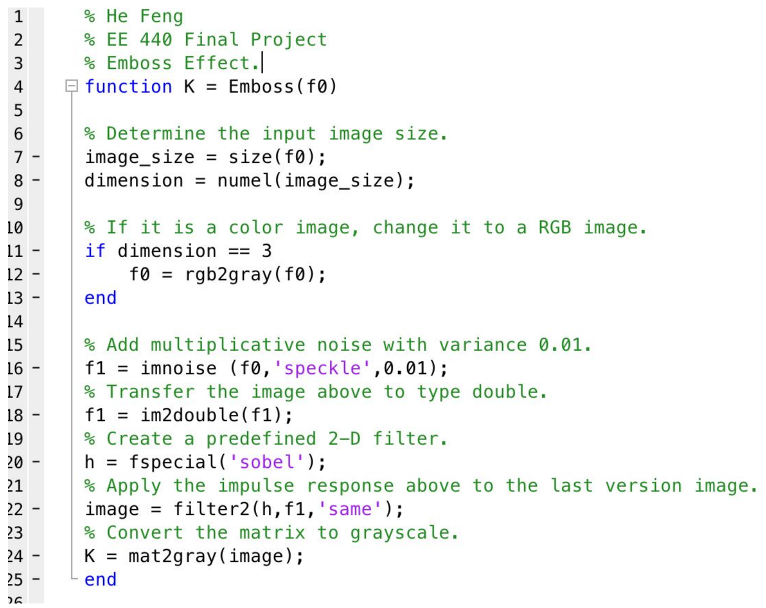


```
% He Feng
% EE 440 Final Project
% Oil Painting Effect.
function im = OilPainting(img)

% Determine the size of each dimension for the image.
[sizex,sizey,sizez] = size(img);
% Change the original image to type double.
imgd = im2double(img);
% Initially define the image by making a matrix which has the same size as
% the original image but full of ones.
im = ones(sizex,sizey,sizez);

radius = 3;
intensityLevel = 10;
% Chaning the image pixel by pixel
% Do not exceed the range based on the target area.
for i = (radius + 1):(sizex - radius)
    for j = (radius + 1):(sizey - radius)
        % Determine the target area for each loop.
        targetArea = imgd(i-radius:i+radius,j-radius:j+radius,:);
        intensityArea = (targetArea(:,:,1) + targetArea(:,:,2) + targetArea(:,:,3))/3*intensityLevel;
        tempArea = uint8(intensityArea);
        maxAppearIntensity = mode(mode(tempArea));
        maxCount = 0;
        maxrgb = [0,0,0];

```



```

        % To determine each pixel in the single target range one by one.
        for i1 = 1:5
            for j1 = 1:5
                % If any pixel in the given range has the intensity meet
                % the masx appear intensity, add up the specific target and
                % generate it to determine the average later.
                if tempArea(i1,j1) == maxAppearIntensity
                    maxCount = maxCount + 1;
                    maxrgb(1) = maxrgb(1) + targetArea(i1,j1,1);
                    maxrgb(2) = maxrgb(2) + targetArea(i1,j1,2);
                    maxrgb(3) = maxrgb(3) + targetArea(i1,j1,3);
                end
            end
        end
        im(i,j,:) = maxrgb * 255 / maxCount;
    end
end
% Change the image type back to uint8 type.
im = uint8(im);
end

```

Emboss.m:



```
% He Feng
% EE 440 Final Project
% Emboss Effect.
function K = Emboss(f0)

% Determine the input image size.
image_size = size(f0);
dimension = numel(image_size);

% If it is a color image, change it to a RGB image.
if dimension == 3
    f0 = rgb2gray(f0);
end

% Add multiplicative noise with variance 0.01.
f1 = imnoise (f0,'speckle',0.01);
% Transfer the image above to type double.
f1 = im2double(f1);
% Create a predefined 2-D filter.
h = fspecial('sobel');
% Apply the impulse response above to the last version image.
image = filter2(h,f1,'same');
% Convert the matrix to grayscale.
K = mat2gray(image);
end
```

Reference

1. Select a Web Site. (n.d.). Retrieved from https://www.mathworks.com/help/matlab/creating_guis/add-code-for-components-in-callbacks.html
2. Sermet (view profile). (n.d.). Retrieved from https://www.mathworks.com/matlabcentral/answers/140495-creating-callback-for-each-string-of-a-popupmenu-in-gui
3. Type. (n.d.). Retrieved from https://www.mathworks.com/help/images/ref/fspecial.html
4. Lin, J. (2012, November 15). Experience talks. Retrieved from http://j42lin.blogspot.com/2012/11/oil-painting-effect-in-matlab.html
5. Oil-Paint Filter. (2016, July 09). Retrieved from http://cbhushan.github.io/oil-paint/
6. Angel, A. (n.d.). IMAGE PROCESSING. Retrieved from https://www.imageeprocessing.com/2011/06/oil-painting-in-matlab.html
7. A New Embossing Method for Color Images - IJCSNS. (n.d.). Retrieved from http://paper.ijcsns.org/07_book/201002/20100223.pdf
8. Select a Web Site. (n.d.). Retrieved from https://www.mathworks.com/help/matlab/creating_guis/about-the-simple-guide-gui-example.html