College of Computer and Information Sciences
Computer Science Department

CSC 478
Image Processing

HW2: MATLAB 2

Prepared by:

Name: Sarah Alajlan

Student ID: 442202314

Dr. Seetah Alsalamah

Task 1:

MATLAB Code:

```
function varargout = GUI(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name', mfilename, ...
'gui_Singleton', gui_Singleton, ...
'gui_OpeningFcn', @GUI_OpeningFcn, ...
'gui_OutputFcn', @GUI_OutputFcn, ...
'gui_LayoutFcn', [] , ...
'gui_Callback', []);
if nargin && ischar(varargin{1})
gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
[varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
gui_mainfcn(gui_State, varargin{:});
end
function GUI_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);
function varargout = GUI_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
function Histogram_Callback(hObject, eventdata, handles)
image = getappdata(0,'image');
imagehist = image;
imagehist = rgb2gray(imagehist);
axes(handles.axes1);
imhist(imagehist);
function ComplementImage_Callback(hObject, eventdata, handles)
image = getappdata(0,'image');
imagecomp = image;
imagecomp = imcomplement(imagecomp);
axes(handles.axes1);
imshow(imagecomp);
function EdgeDetection_Callback(hObject, eventdata, handles)
image = getappdata(0,'image');
imagedge = image;
imagedge = rgb2gray(imagedge);
imagedge = edge(imagedge,'prewitt');
axes(handles.axes1);
imshow(imagedge);
function Uploadimage_Callback(hObject, eventdata, handles)
image = uigetfile('.jpg')
```

```
image = imread(image);
axes(handles.axes1);
imshow(image);
setappdata(0,'image',image)
function rgb2gray_Callback(hObject, eventdata, handles)
image = getappdata(0,'image');
imagegray = rgb2gray(image);
axes(handles.axes1);
imshow(imagegray);
function Convert2image_Callback(hObject, eventdata, handles)
image = getappdata(0,'image');
imagetoBinary = im2bw(image);
axes(handles.axes1);
imshow(imagetoBinary);
function reset_Callback(hObject, eventdata, handles)
image = getappdata(0,'image');
axes(handles.axes1);
imshow(image);
```

Output:

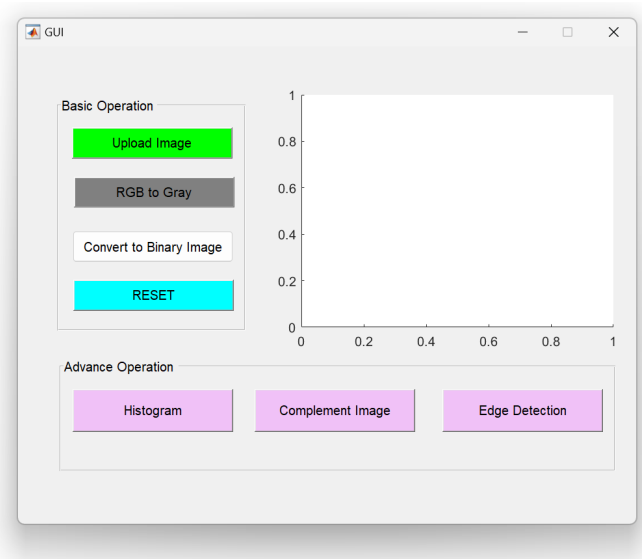

Figure 1

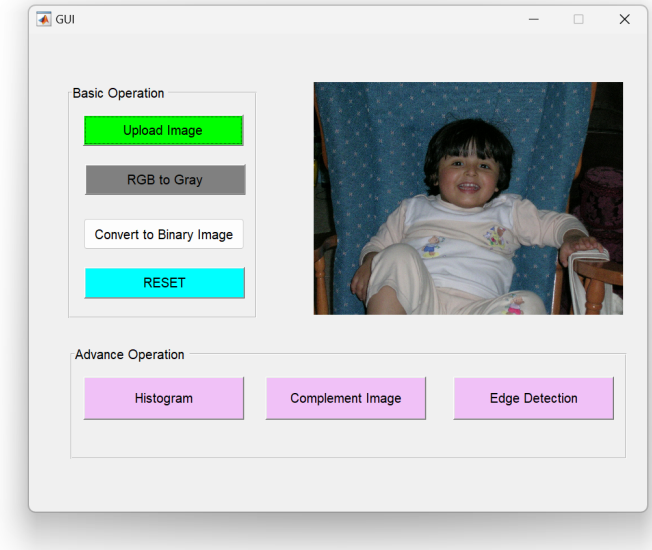

Figure 2: after uploading image

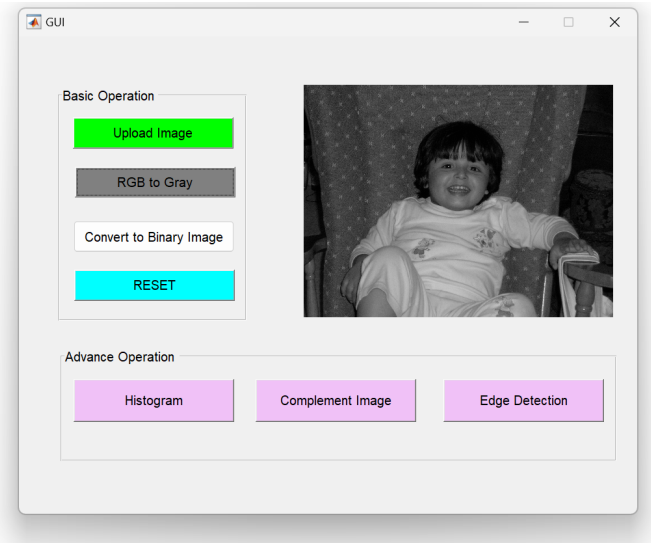

Figure 3: after RGB to Gray

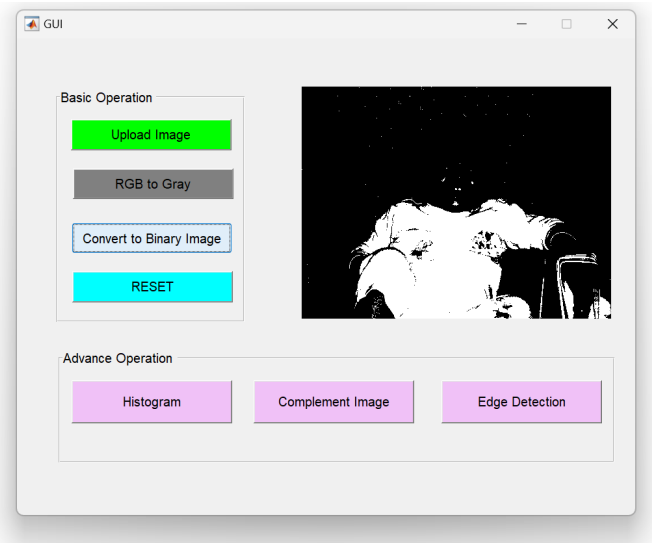

Figure 4: after converting to binary

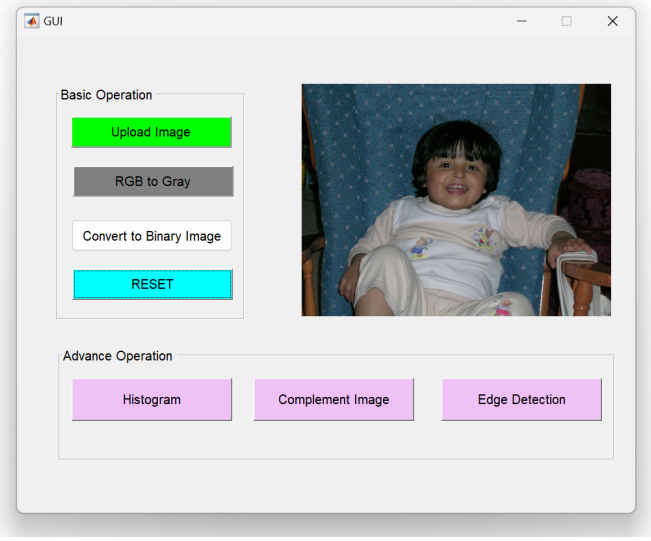

Figure 5: after Reset

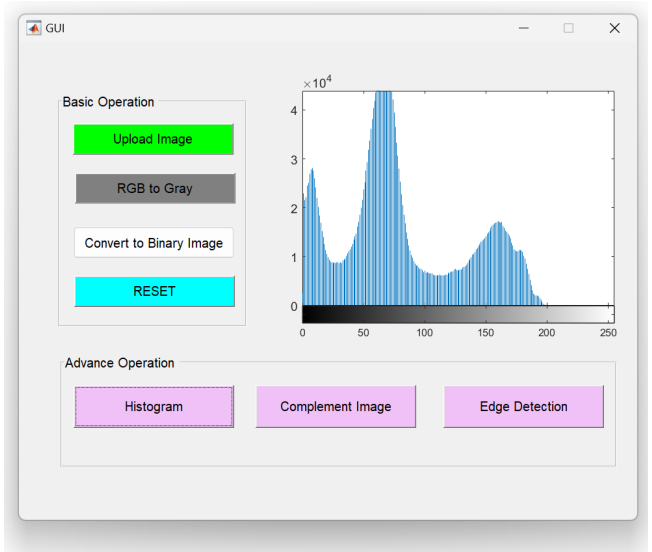

Figure 6: after Histogram

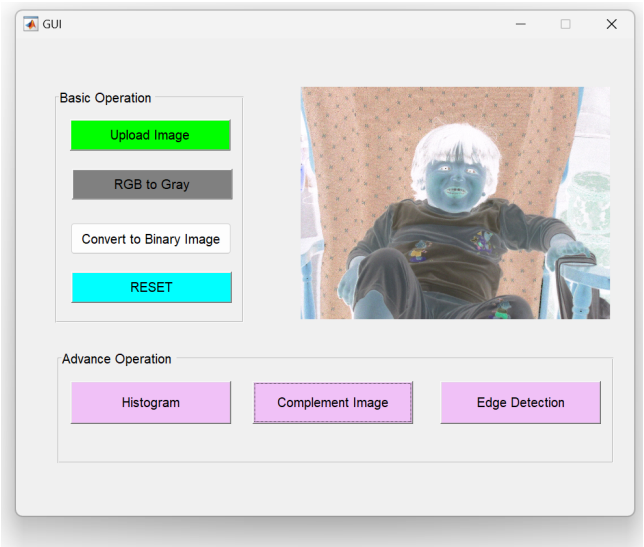

Figure 7: after Complement

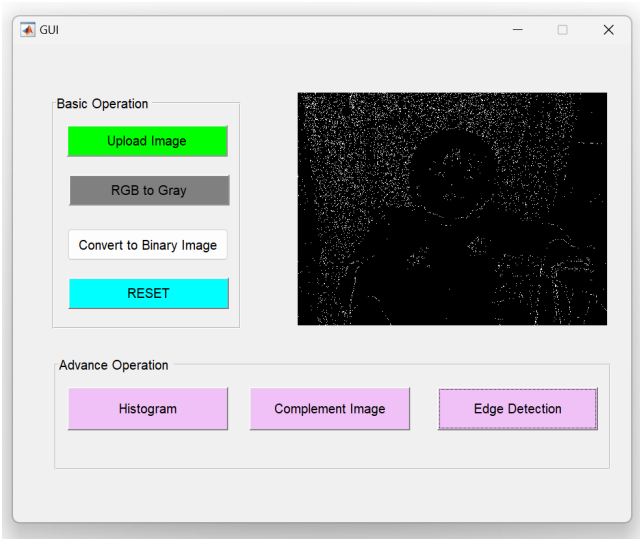

Figure 8: after Edge detection

Task 2:

MATLAB Code:

```
% Loading the image
image = imread('HW2.jpeg');
% Convert the image to grayscale if needed
image = rgb2gray(image);
% Perform 2D Fourier transform
imageFft = fft2(image);
% Create a filter in the frequency domain
% Can adjust the filter shape and size based on your requirements
filter = fspecial('gaussian', size(image), 30);
% Apply the filter in the frequency domain
FilteredImageFft = imageFft .* fftshift(filter);
% Perform the inverse Fourier transform to get the smoothed image
SmoothedImageUsingFrequency = ifft2(FilteredImageFft);
% Define the standard deviation for the Gaussian filter
sigma = 2;
% Apply Gaussian smoothing using spatial domain filtering
SmoothedImageUsingSpatial = imgaussfilt(image, sigma);
% Create a figure to display all four images
```

```
figure;
% Create a subplot for the image before smoothing
subplot(1, 3, 1);
% Display the original image
imshow(image);
% Setting a title for the subplot
title('Image before smoothing');
% Create a subplot for the smoothed image (frequency domain)
subplot(1, 3, 2);
% Display the smoothed image (frequency domain)
imshow(abs(SmoothedImageUsingFrequency), []);
% Setting a title for the subplot
title('Frequency domain');
% Create a subplot for the smoothed image (spatial domain)
subplot(1, 3, 3);
% Display the smoothed image (spatial domain)
imshow(SmoothedImageUsingSpatial);
% Setting a title for the subplot
title('Spatial domain');
```

Output:

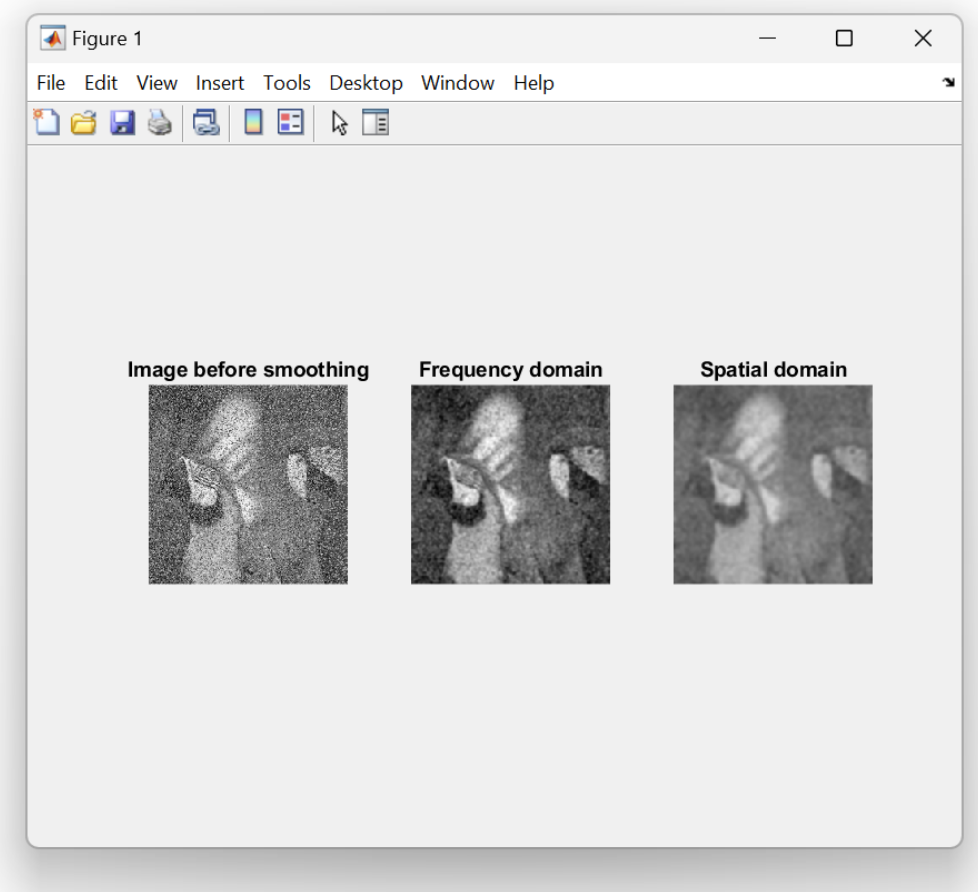

Figure 9

In my opinion, frequency domain smoothing is better because it provided a more efficient result than spatial domain smoothing. It effectively removed some of the noise while preserving the image's features, unlike spatial domain smoothing.