USER MANUAL
IMAGE PROCESSING PROJECT

Group Leader: Bintang Sonda Sitorus Pane
001202000030
Group Members: Muhammad Satria Budhi Pratama
001202000136

PRESIDENT UNIVERSITY
2021

APPLICATION DESCRIPTION

The BS Image Processing by Matlab 2019 to modify images by predetermined conditions in 3 scope of processing

Application Capabilities

a. Purposes and Scope

The purpose of this program is to modify images by implementing all subjects of Image Processing and Understanding to learn how the process of modifying Image using code.

The scope of this program is :
Basic Operation
Image Transformation
Image Advancement

b. General Descriptions

Our Program not only learns how the image processing runs, but also could help to modify the original image to become what the user wants to be and save it by their own device. Our feature for the program are:

- Scaling
- Rotation
- Flip
- Distance measurement
- Black and white
- Complement of image (negative color)
- Create noise
- Noise reduction
- Greyscale
- Histogram
- Intensity light
- RGB
- Edge detection
- Image sharpening
- Image restoration
- Reset
- Exit
- Save

Platform Requirements

Our program uses Matlab 2019 Platform to run the program so the requirements to use our program is the requirements to run the Matlab 2019 Software, the requirements to run Matlab are :

Operating Systems
• Windows 10
• Windows 7 Service Pack 1
• Windows Server 2019
• Windows Server 2016

Note:
• Windows Server 2019 is supported as of R2019a.
• Windows Server 2012 and Windows Server 2012 R2 are not supported as of R2019a.

Processors
Minimum: Any Intel or AMD x86-64 processor
Recommended: Any Intel or AMD x86-64 processor with four logical cores and AVX2 instruction set support

Disk
Minimum: 2.9 GB of HDD space for MATLAB only, 5-8 GB for a typical installation
Recommended: An SSD is recommended
A full installation of all MathWorks products may take up to 29 GB of disk space

RAM
Minimum: 4 GB
Recommended: 8 GB
For Polyspace, 4 GB per core is recommended
Graphics
No specific graphics card is required.
Hardware accelerated graphics card supporting OpenGL 3.3 with 1GB GPU memory is recommended.
GPU acceleration using the Parallel Computing Toolbox requires a CUDA GPU

GETTING STARTED (MATLAB 2019)

A. Installation Steps

Write the steps for the installation (if any) here.

Download Process:

Step 1: Go to your Mathworks account homepage.

Step 2: Locate the License you would like to download products for in the list.

Step 3: Click the downwards-pointing blue arrow on the same row as the license in question.

Step 4: Click the blue button on the left to download the latest release of MATLAB you have access to, or select an older license in the menu on the right.

Step 5: Choose the platform you need the installer for.

Step 6: If prompted by your browser to Run or Save the installer choose to save.

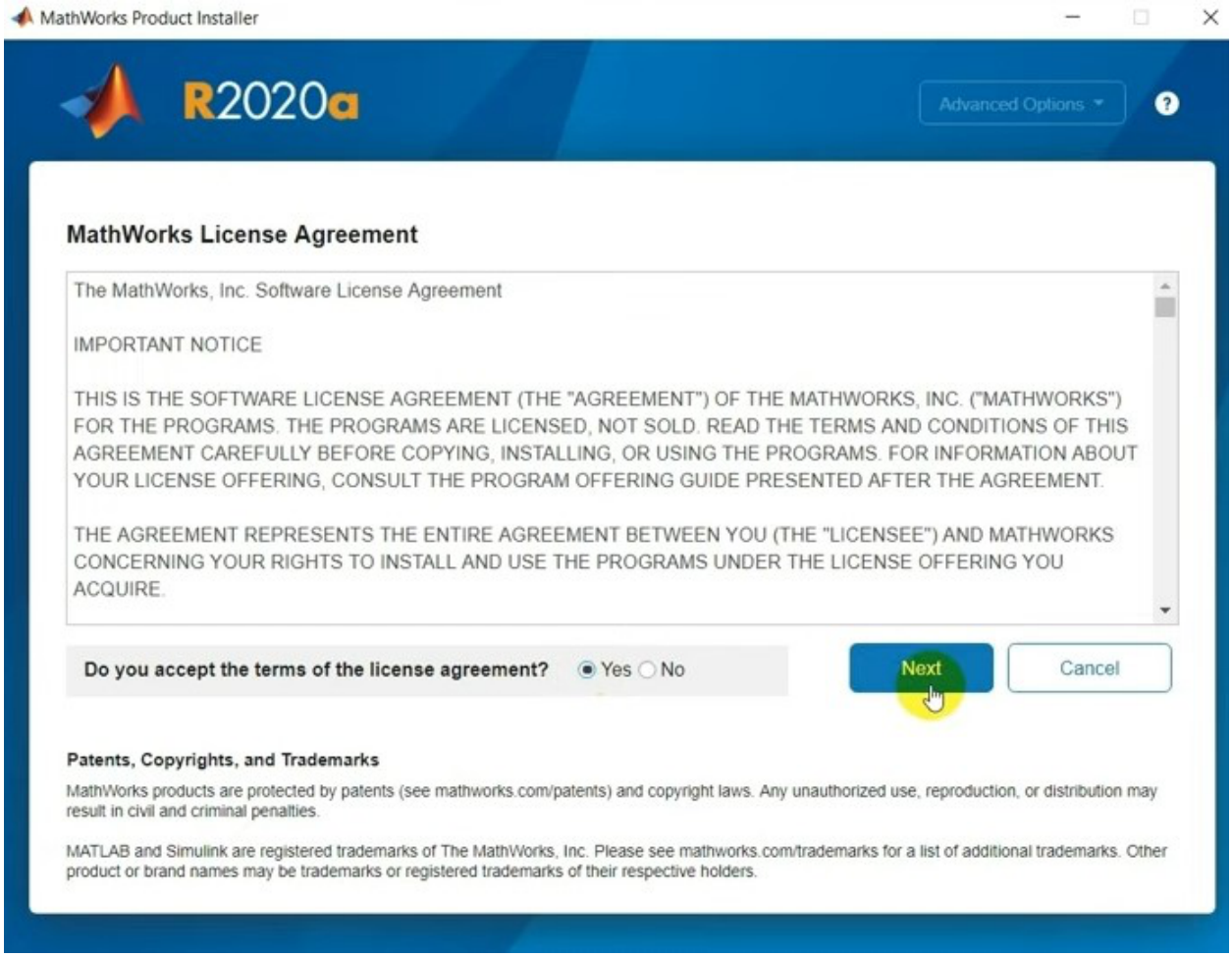
Step 7: Locate the installer in a file browser. It should be located in the default download location unless you specified another location. The installer will be named:

Windows 64 bit: matlab_R20XXx_win64.exe
Windows 32 bit: matlab_R20XXx_win32.exe
Mac: matlab_R20XXx_maci64.zip
Linux: matlab_R20XXx_glnxa64.zip

Installation Process:

To begin with the installation process you must have downloaded the MATLAB setup.

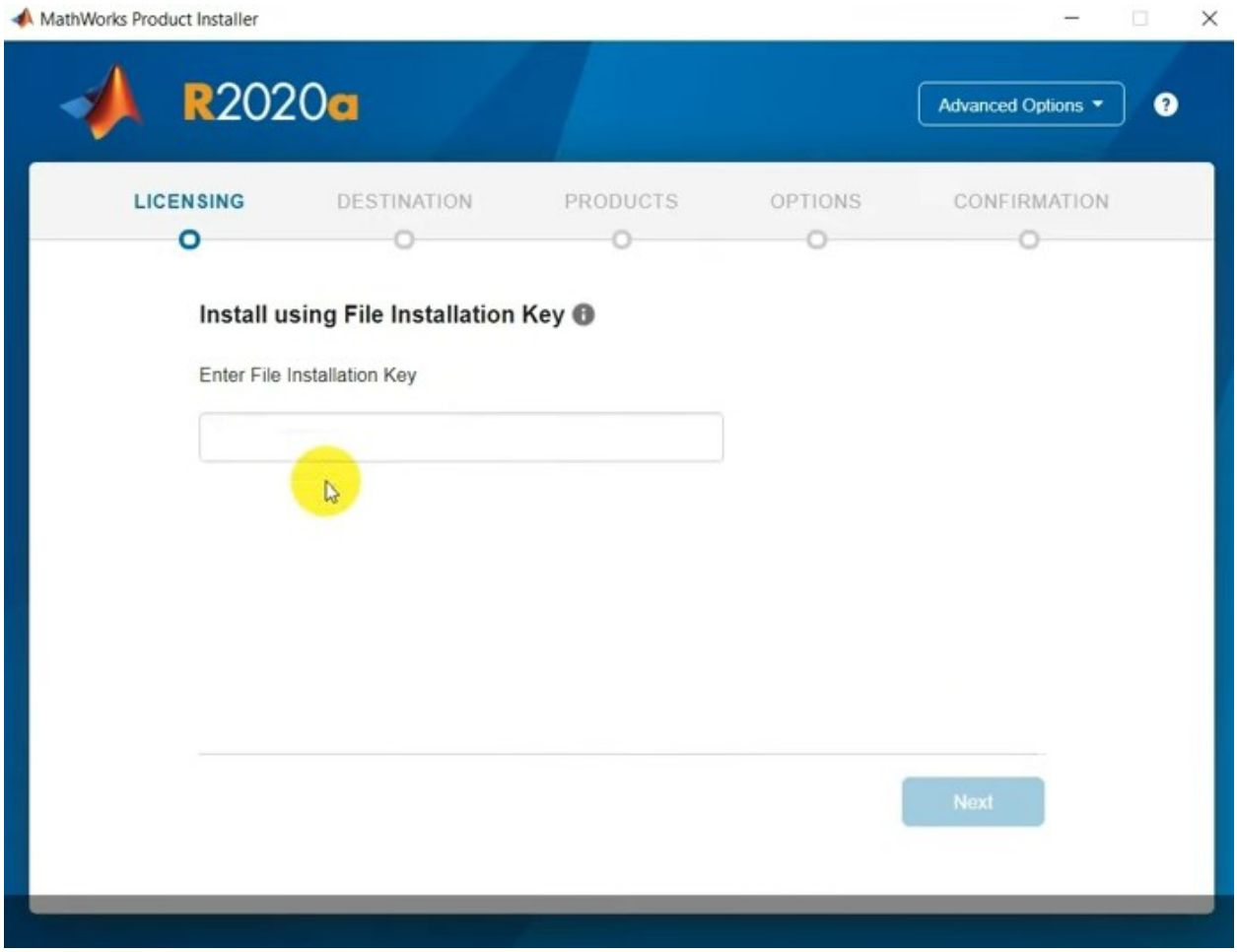
Step 1: After the extraction of the setup, an application named ‘setup’ with MATLAB icon will appear. Click on that application the following window will appear:



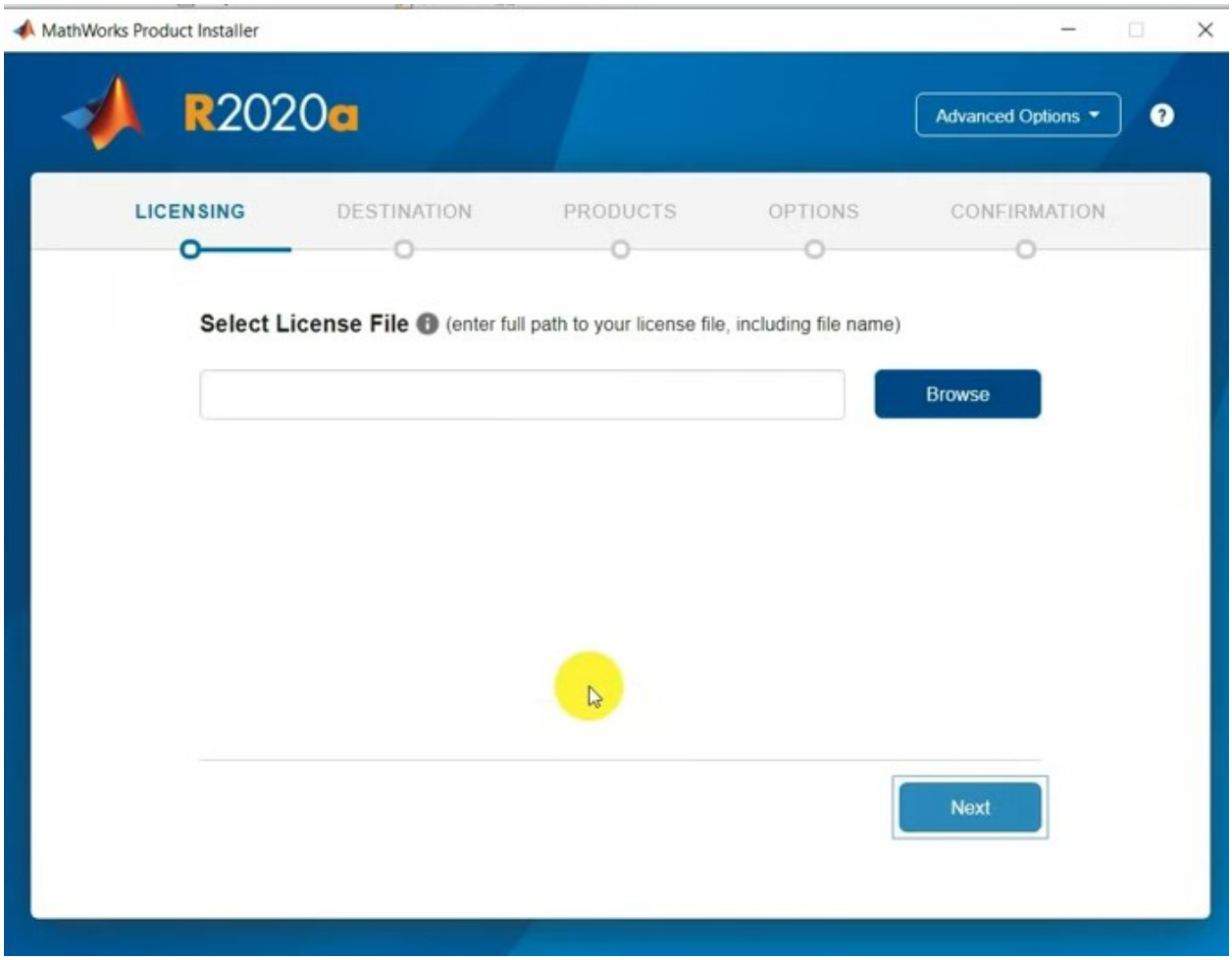
Step 2: After reading the terms and conditions click on “Yes” and then press Next.

Step 3: Then a window asking for the installation key and License file will appear key them ready beforehand so that you can complete the setup in one go without arranging the elements.

Note: Format of Installation key is a combination of 30 digits as xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx

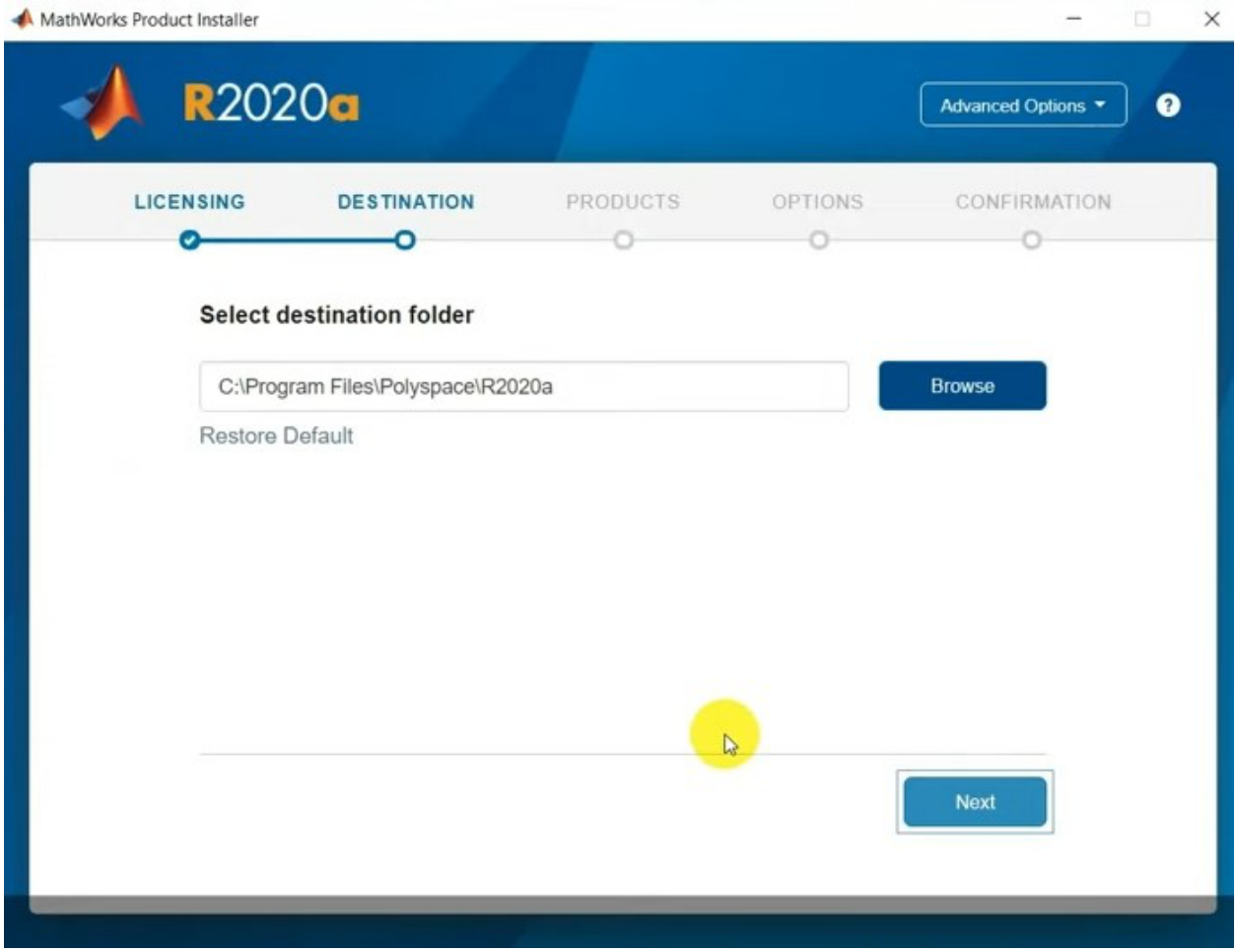


Step 4: After you have entered the Installation key, it will require a license file that will be extracted with the setup if you have purchased the software so give the location of that file to continue the installation.

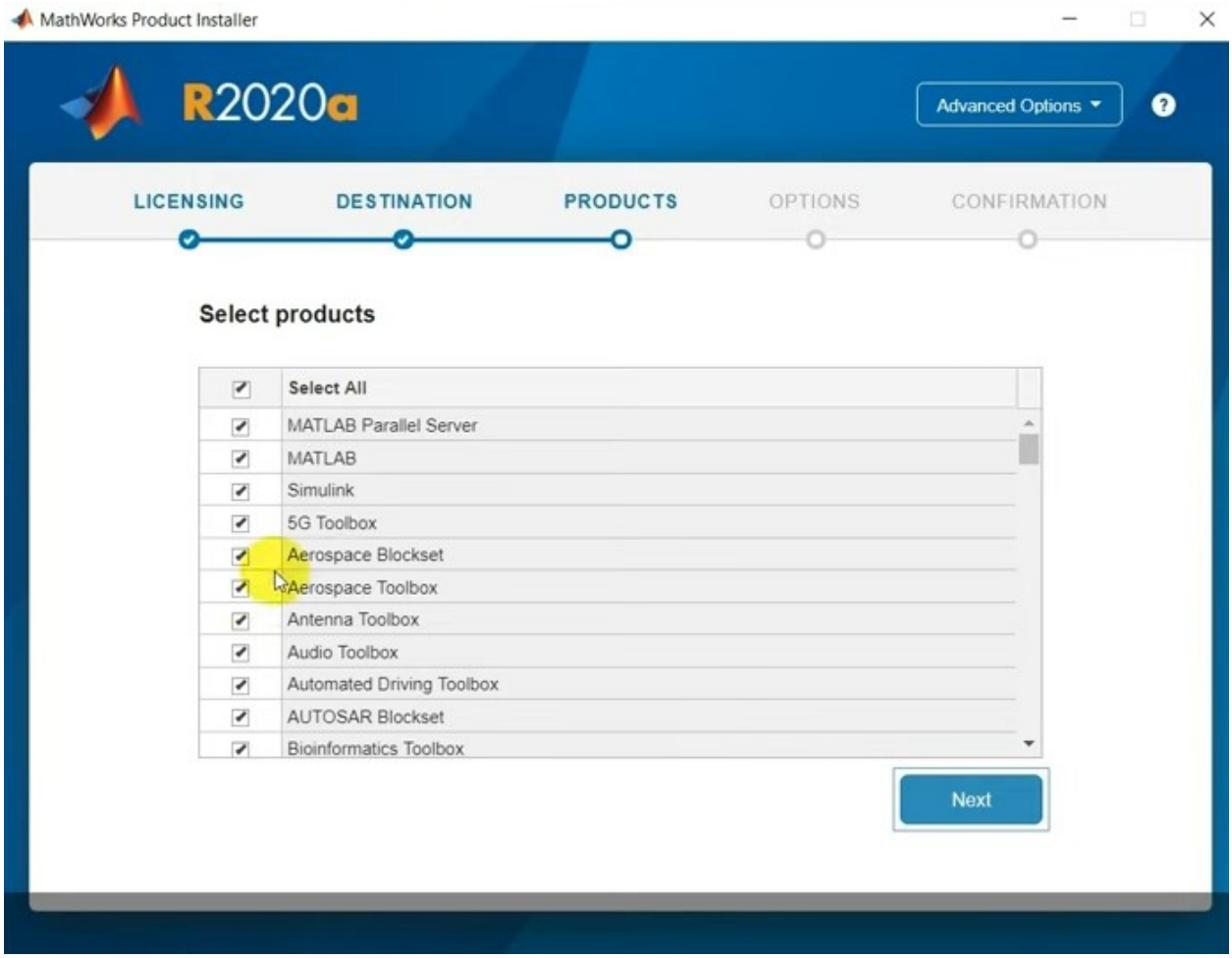


Step 5: When the Installation key and the License file are verified it will ask you for the location you want to install it on.

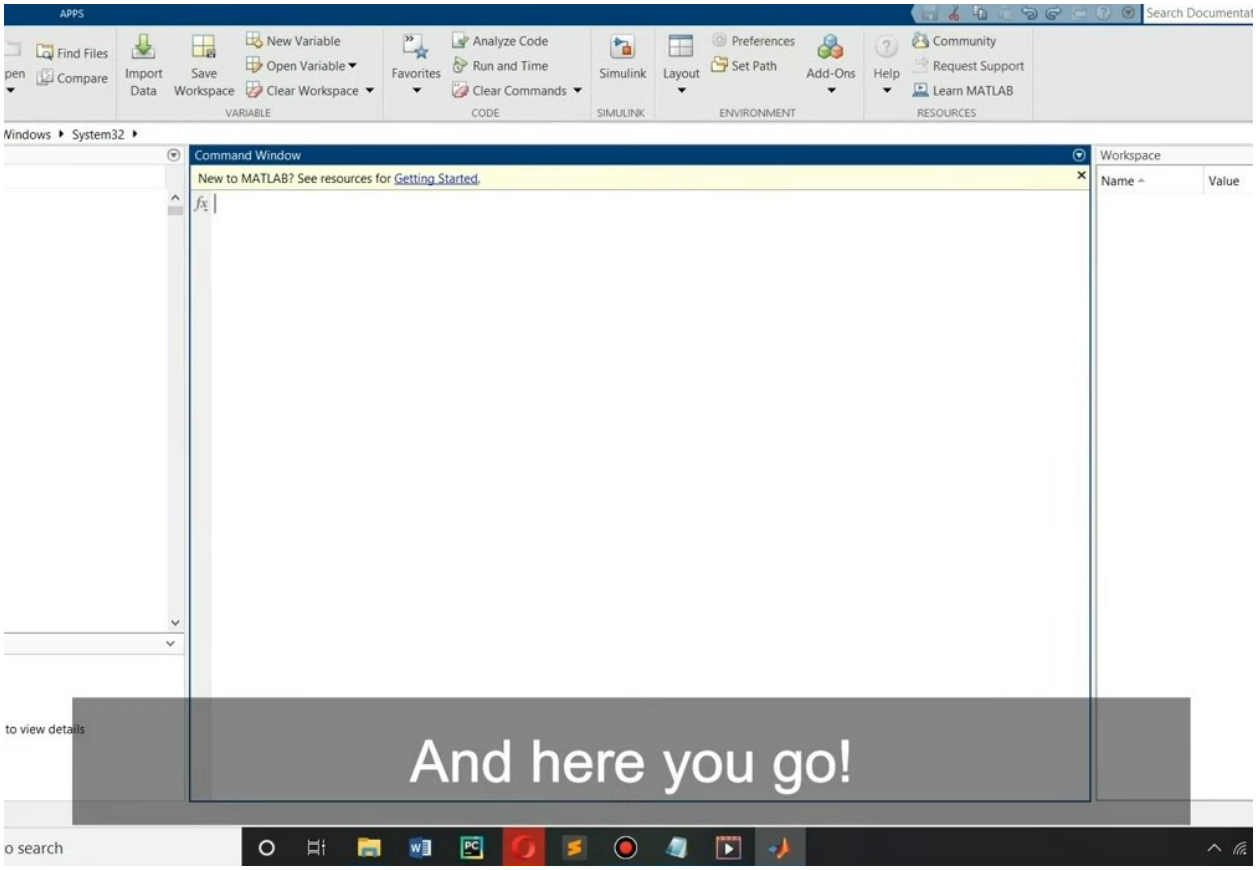
Note: Install it in the C:/ directory to avoid any problem in future



Step 6: For the final step before installation selects the products you want to use in your MATLAB. You should probably select all services in case you want to use some service in the future.



Step 7: After that wait for the installation process to be finished and then open the MATLAB application installed to confirm the proper functioning for the first time.

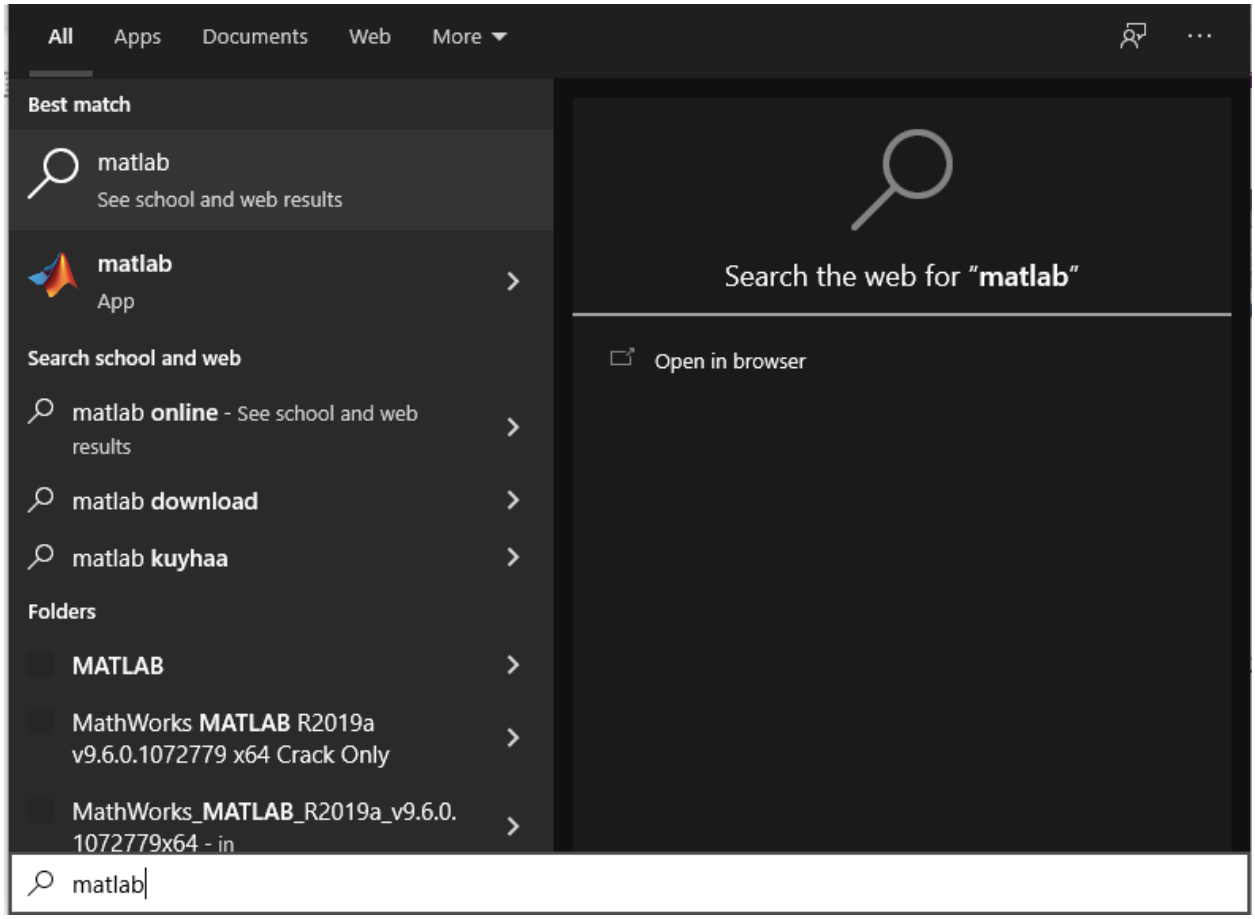


Now the MATLAB is successfully installed on your windows, and you can use it to its fullest.

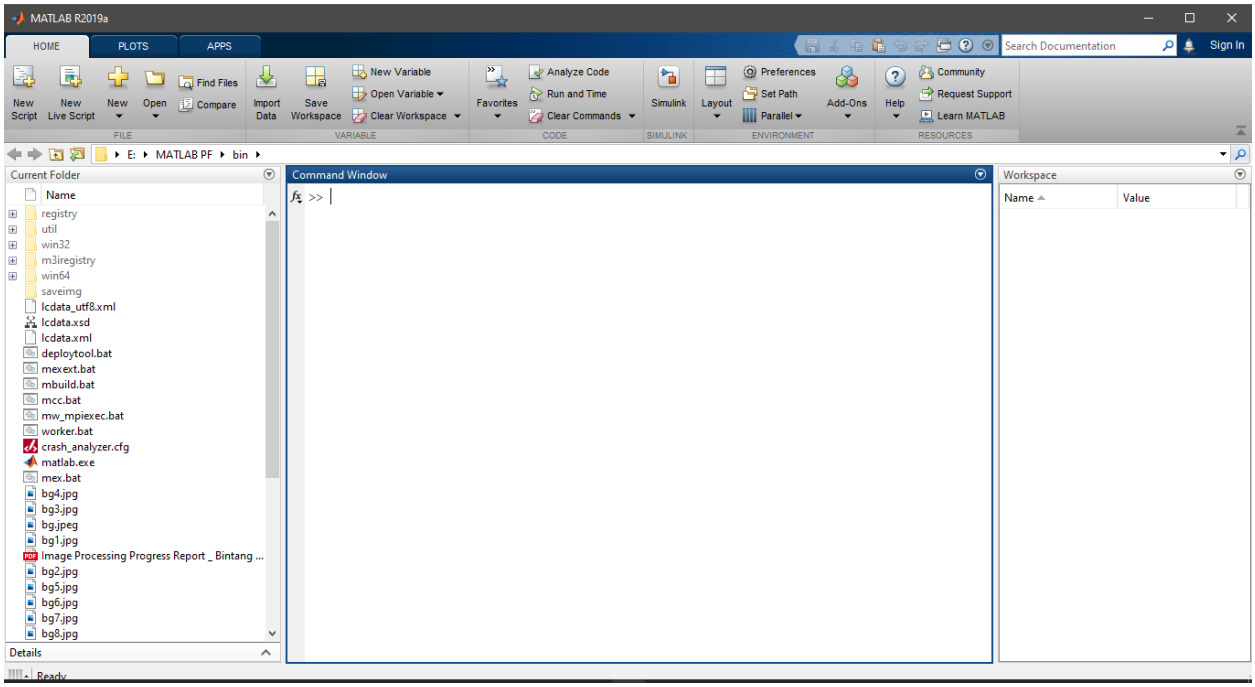
B. Application Menu

Provide the user with the application menu in starting the application.

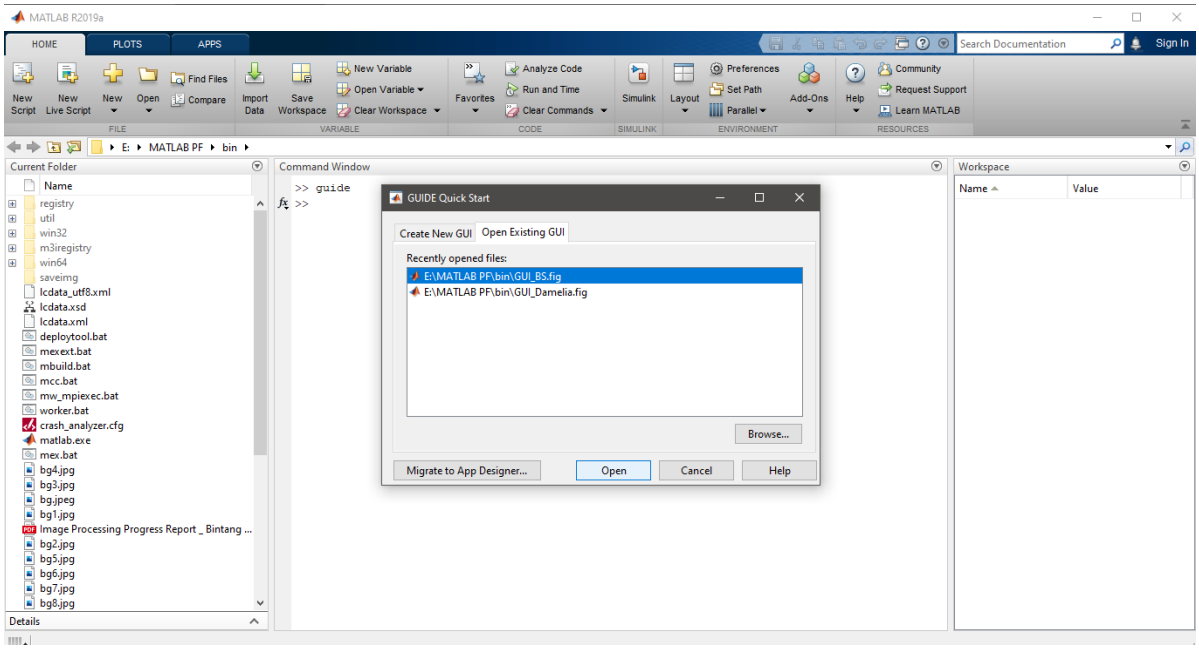
OPEN THE MATLAB SOFTWARE



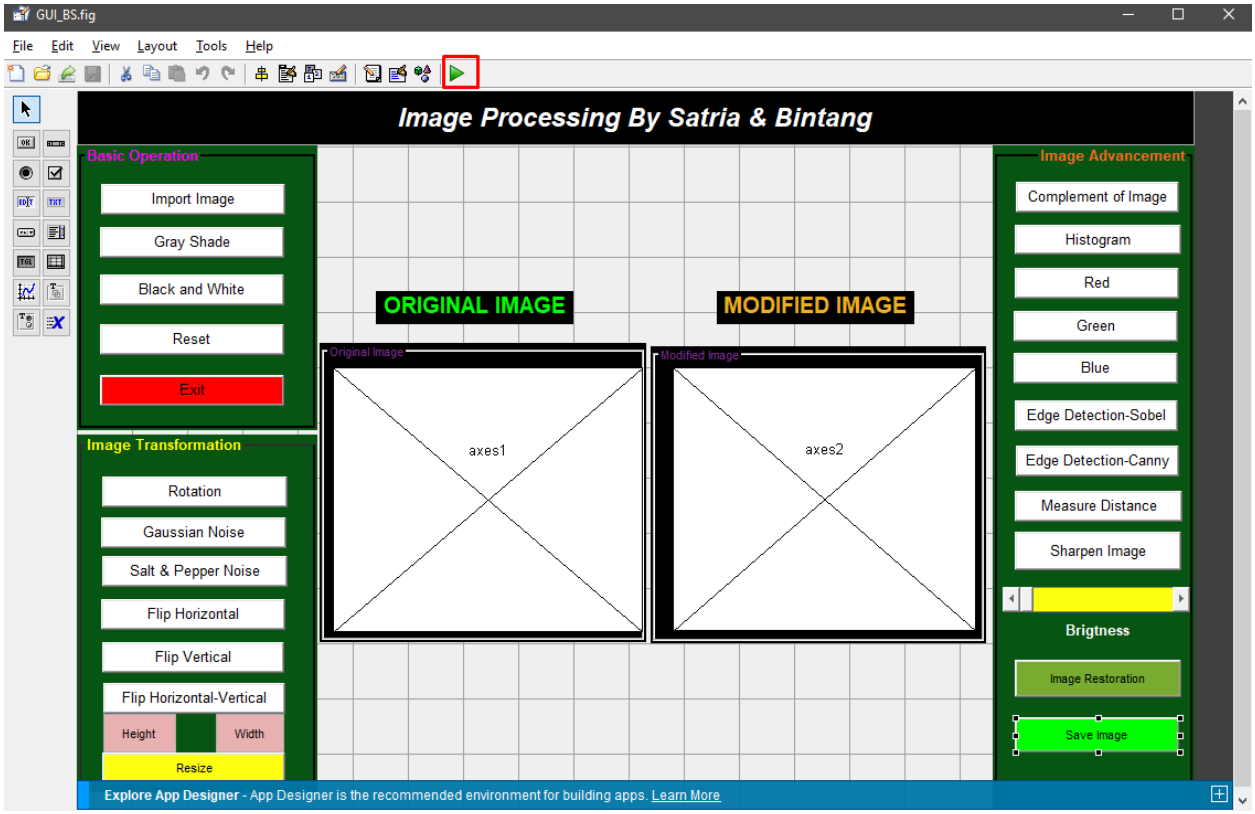
This is the menu shown when you open the Matlab application



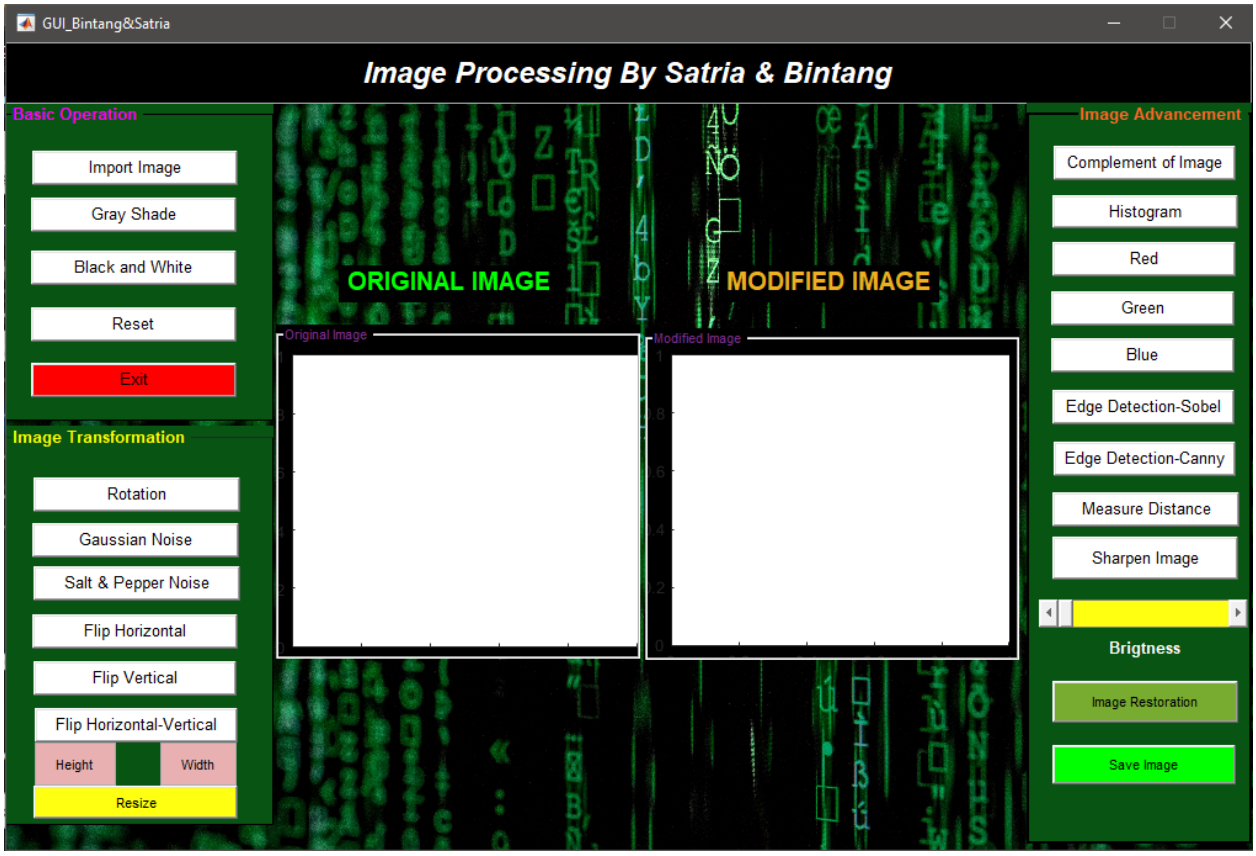
Type “guide” in the command window and press enter >> choose to Create New GUI if you don’t have GUI, but if you have our GUI just choose open existing GUI and click our File “GUI_BS.fig” and then OPEN



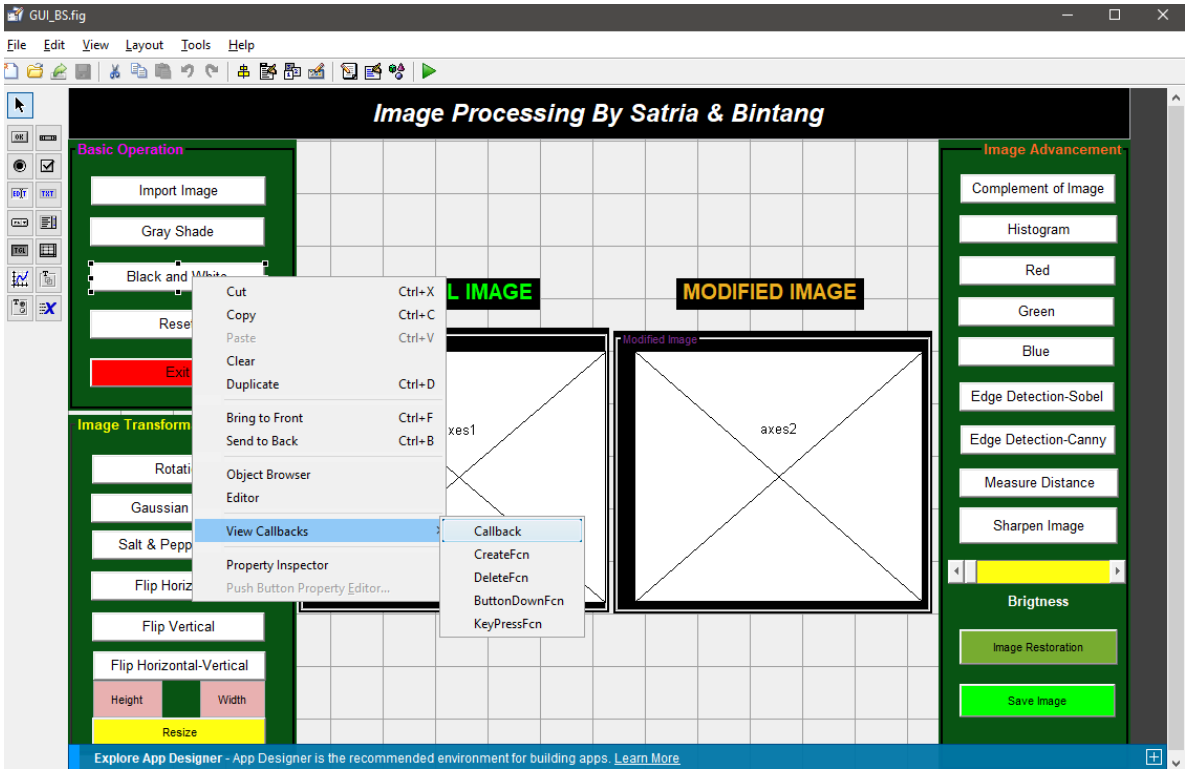
After the fig has shown you can just click play/run (green triangle in the menu bar above)



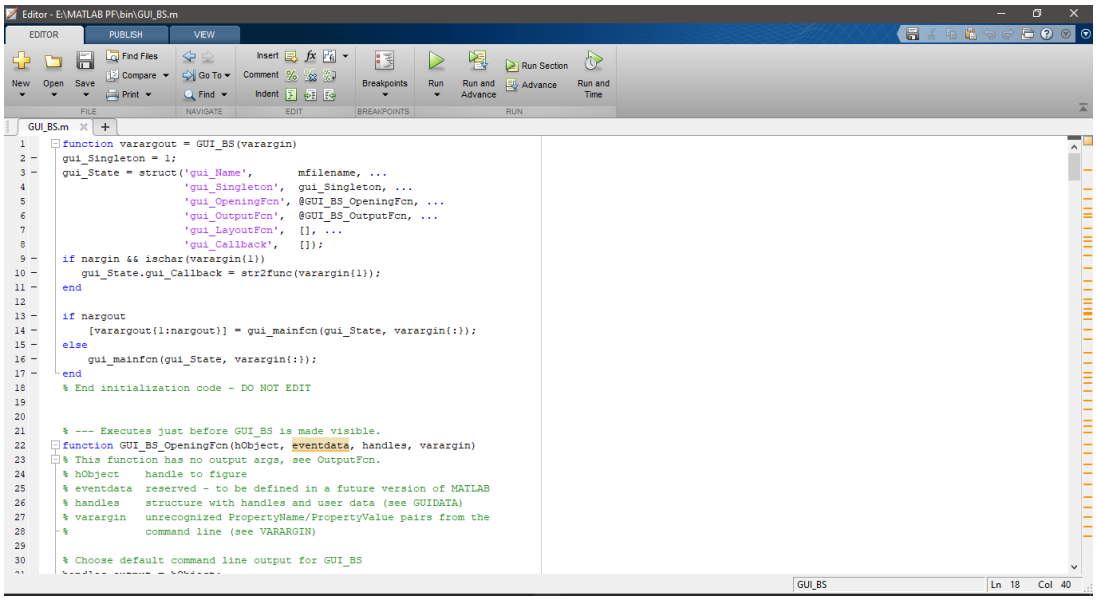
If the program work then this user interface will be shown and now you can just modify your image to whatever button feature in the program below, by insert an image from “Import Image” button



But if you want to edit/see the source code, just right-click one of the Button>>View Callbacks>>Callback



This is the source code from our code, you can modify it to edit the code become what your needs



So that's all the steps to run our program by using Matlab 2019 software Platform.

USING THE APPLICATION

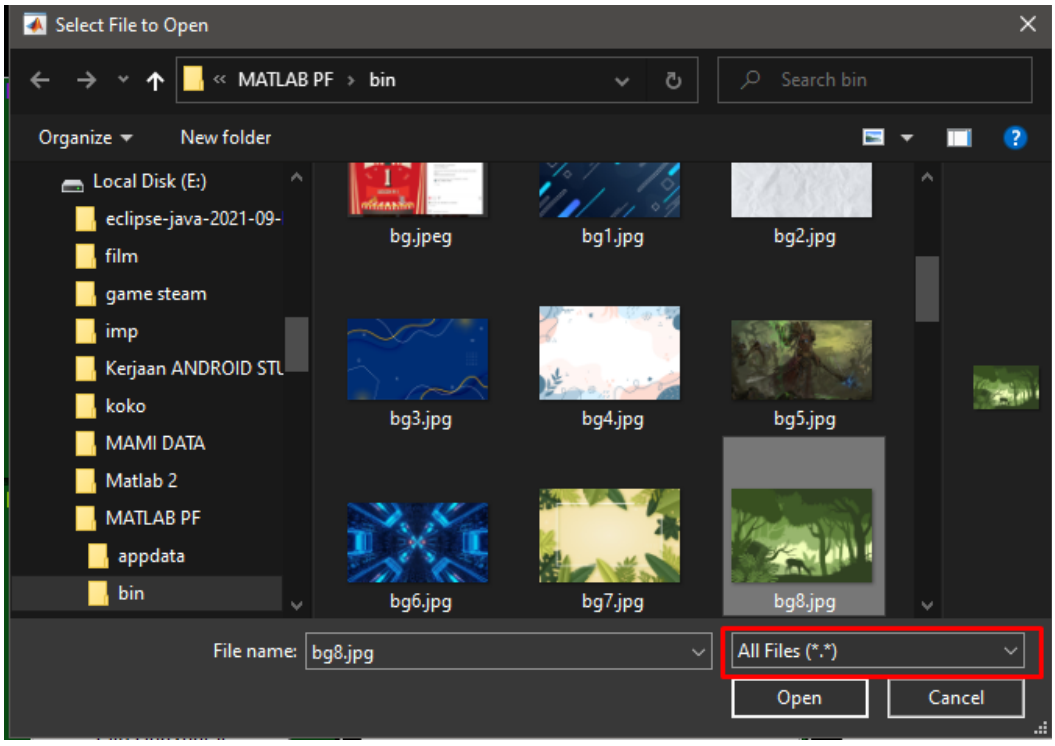
Write detailed descriptions of features including screenshots of features here. Each feature should be explained in **Concise, unambiguous, and clear words.** The information provided is accurate and **supported** with the screenshots of the results.

1. Import Image Button

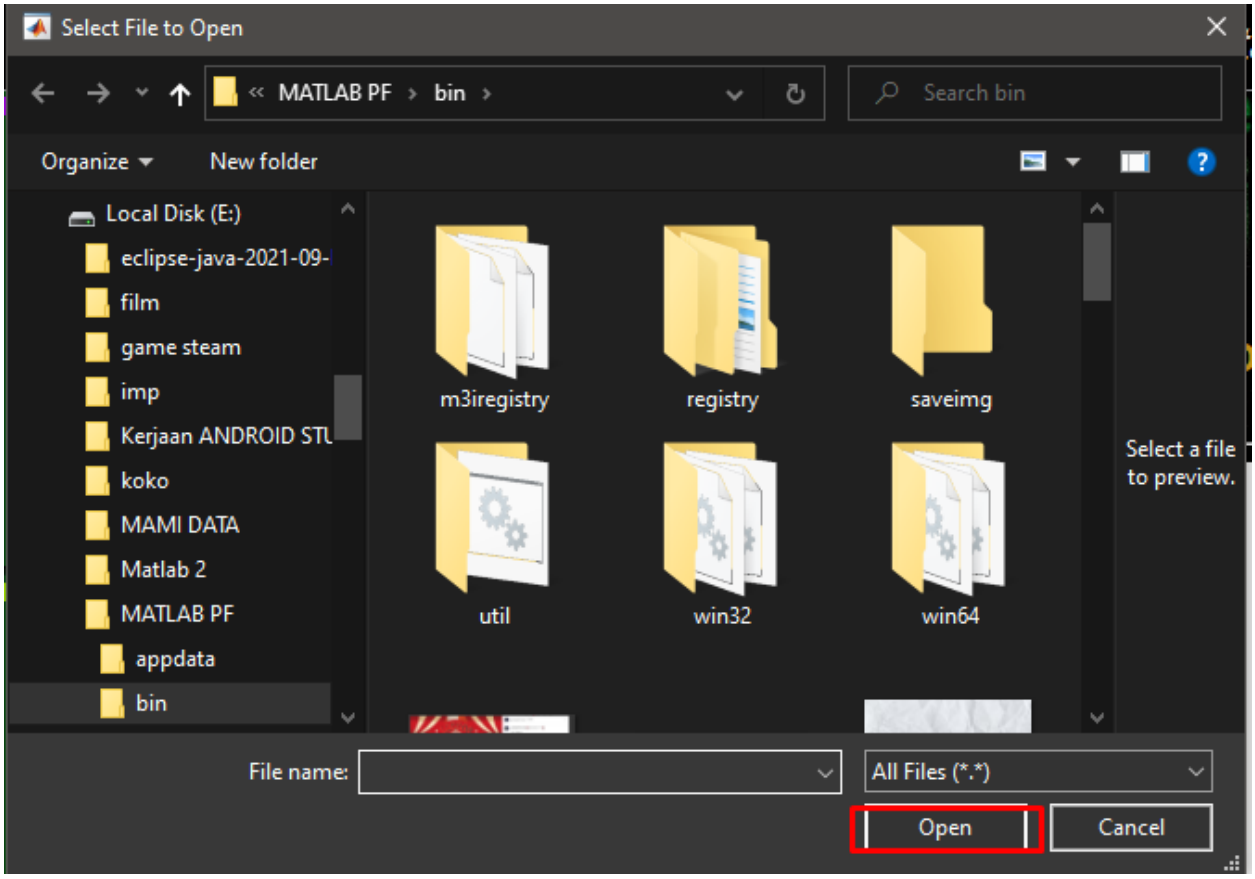
This button is used to import images from our file explorer into the original image section in the program.

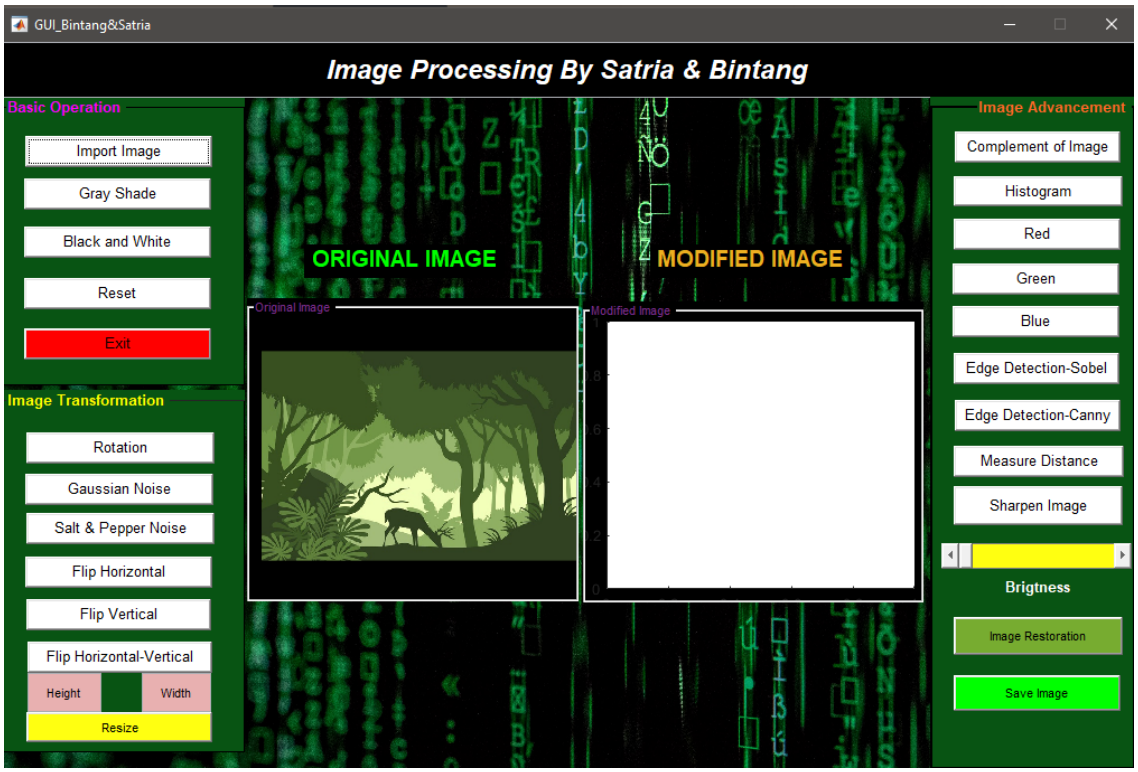
How to use :

- Click IMPORT IMAGE button, your file explorer must be automatically be opened
- Choose the image you want to process in the file explorer. Make sure the file is not of type gif or text(pdf, txt, doc, etc..) to avoid errors
- Make sure the extension is “All Files (“.”)



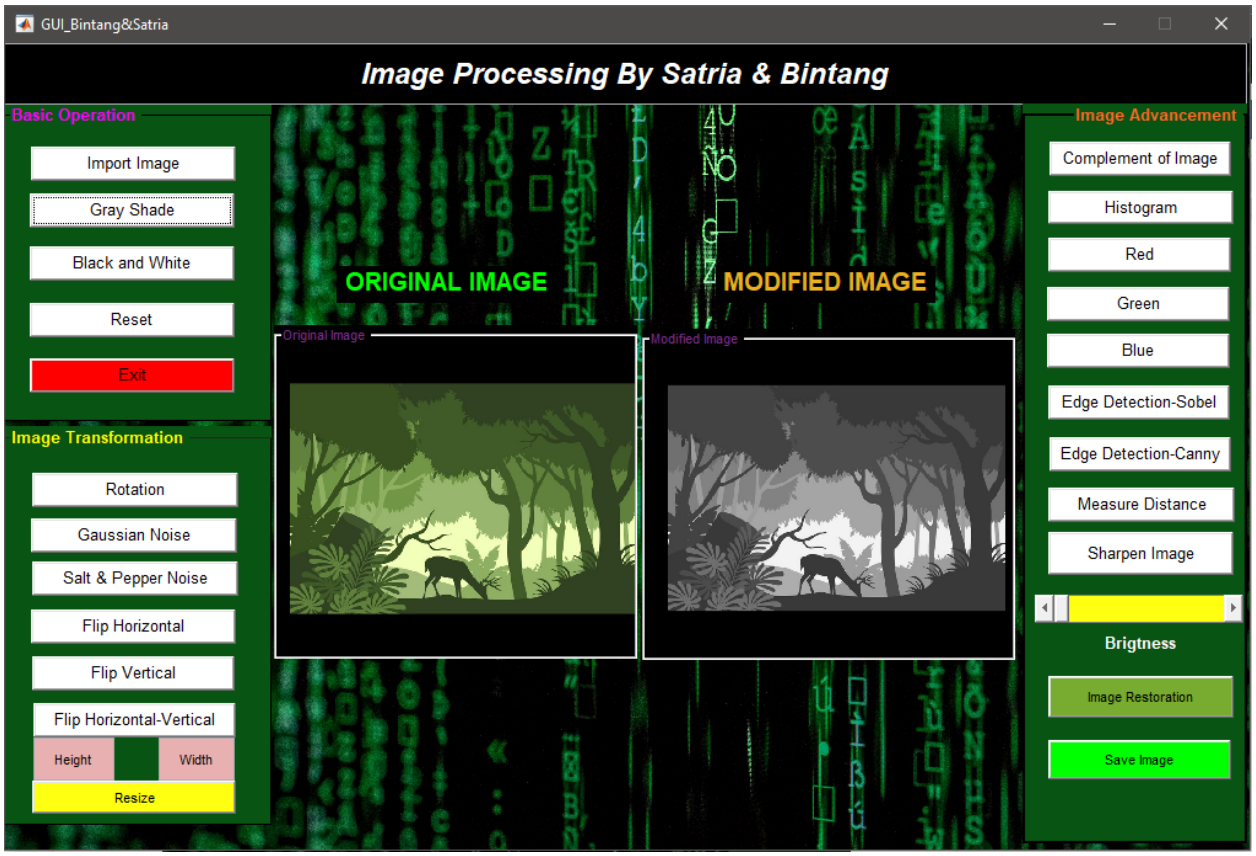
- Click the OPEN button
- Your Image is now in th original image section, ready to be processed :


Select File to Open
MATLAB PF › bin ›
Search bin
Organize
New folder
Local Disk (E:)
eclipse-java-2021-09-
film
game steam
imp
Kerjaan ANDROID STU
koko
MAMI DATA
Matlab 2
MATLAB PF
appdata
bin
m3iregistry
registry
saveimg
util
win32
win64
Select a file to preview.
File name:
All Files (*.*)
Open
Cancel


GUI_Bintang&Satria
Image Processing By Satria & Bintang
Basic Operation
Import Image
Gray Shade
Black and White
Reset
Exit
Image Transformation
Rotation
Gaussian Noise
Salt & Pepper Noise
Flip Horizontal
Flip Vertical
Flip Horizontal-Vertical
Height
Width
Resize
ORIGINAL IMAGE
MODIFIED IMAGE
Original Image
Modified Image
Image Advancement
Complement of Image
Histogram
Red
Green
Blue
Edge Detection-Sobel
Edge Detection-Canny
Measure Distance
Sharpen Image
Brigtness
Image Restoration
Save Image

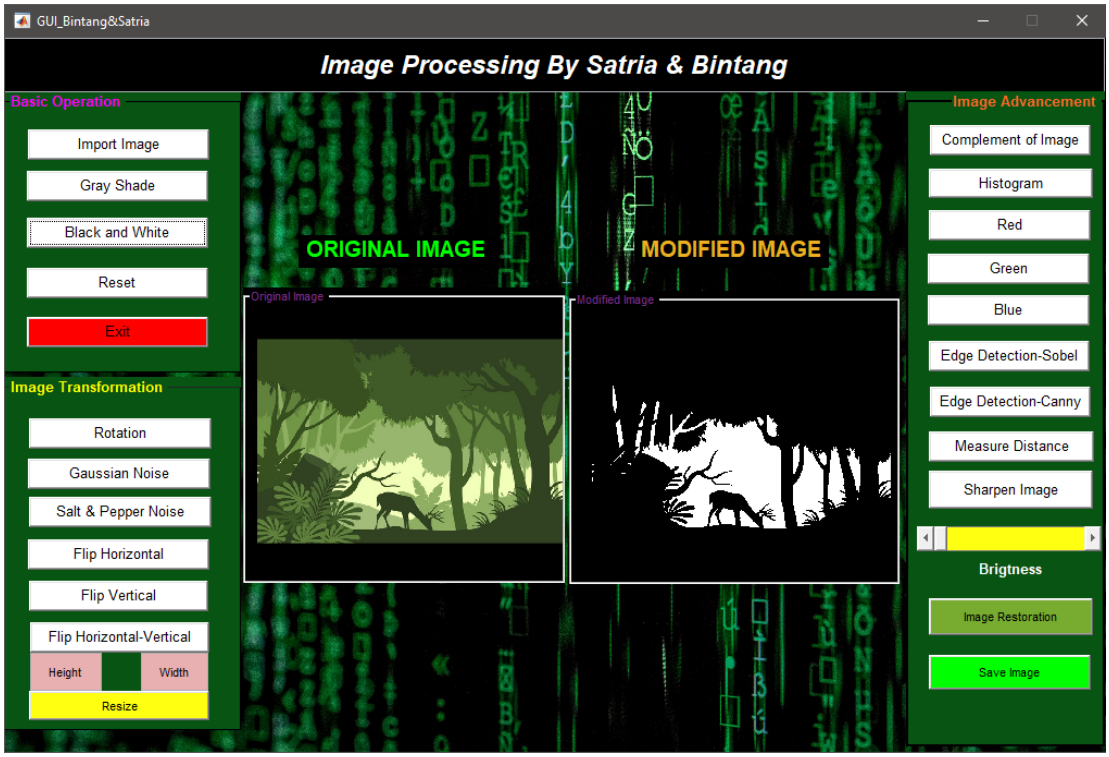
2. Gray Scale Button

This button is used to convert the image color from RGB into Gray Scale.
In Grayscale function, the code for this feature has its own function called “rgb2gray(x)”. So we only implement the code and add the “x” become the value of original image.



3. Black and White Button

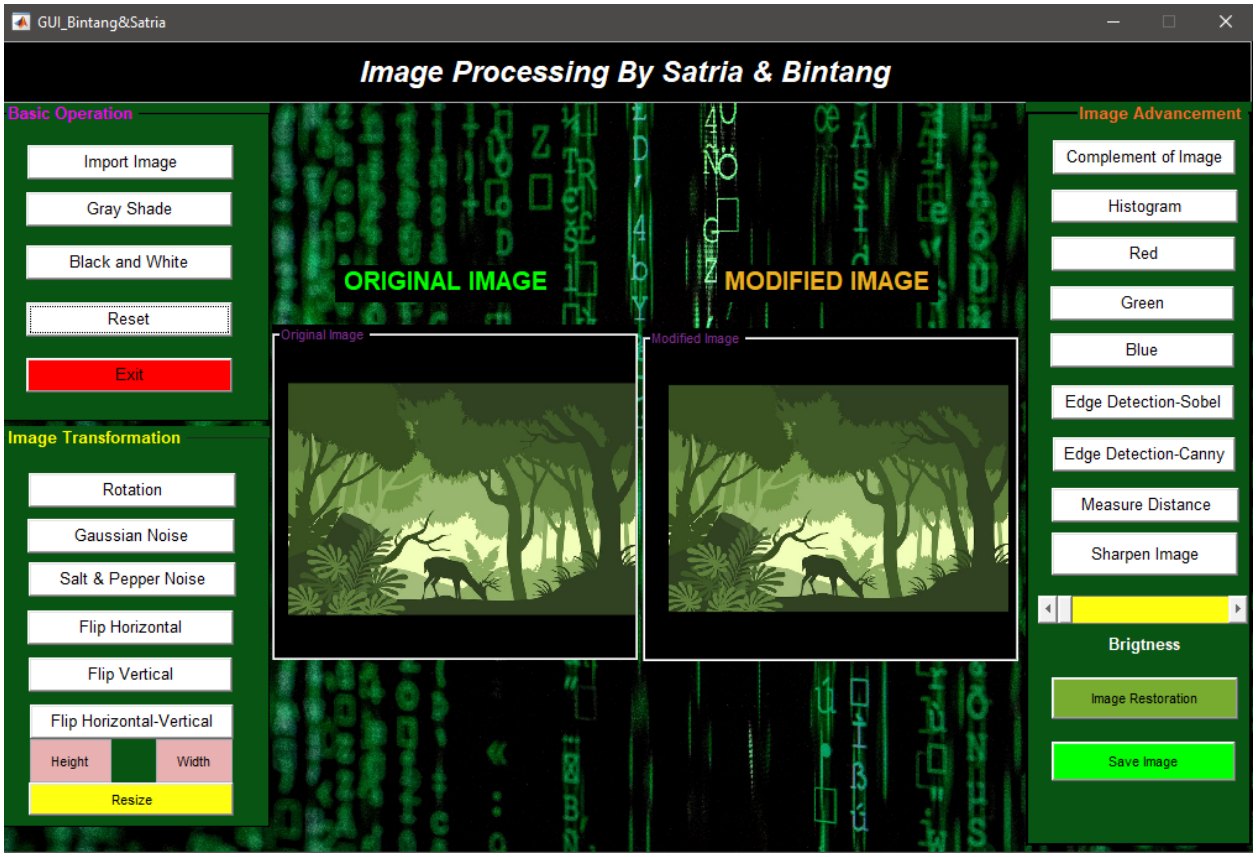
This button is used to convert the image from RGB into Black and White. The code for this is “rgb2bw(x)”. So we only insert the image into the parameter “x”, and it will be converted to black and white version.



4. Reset Button

This button is used to undo all the changes that are made into its original state.

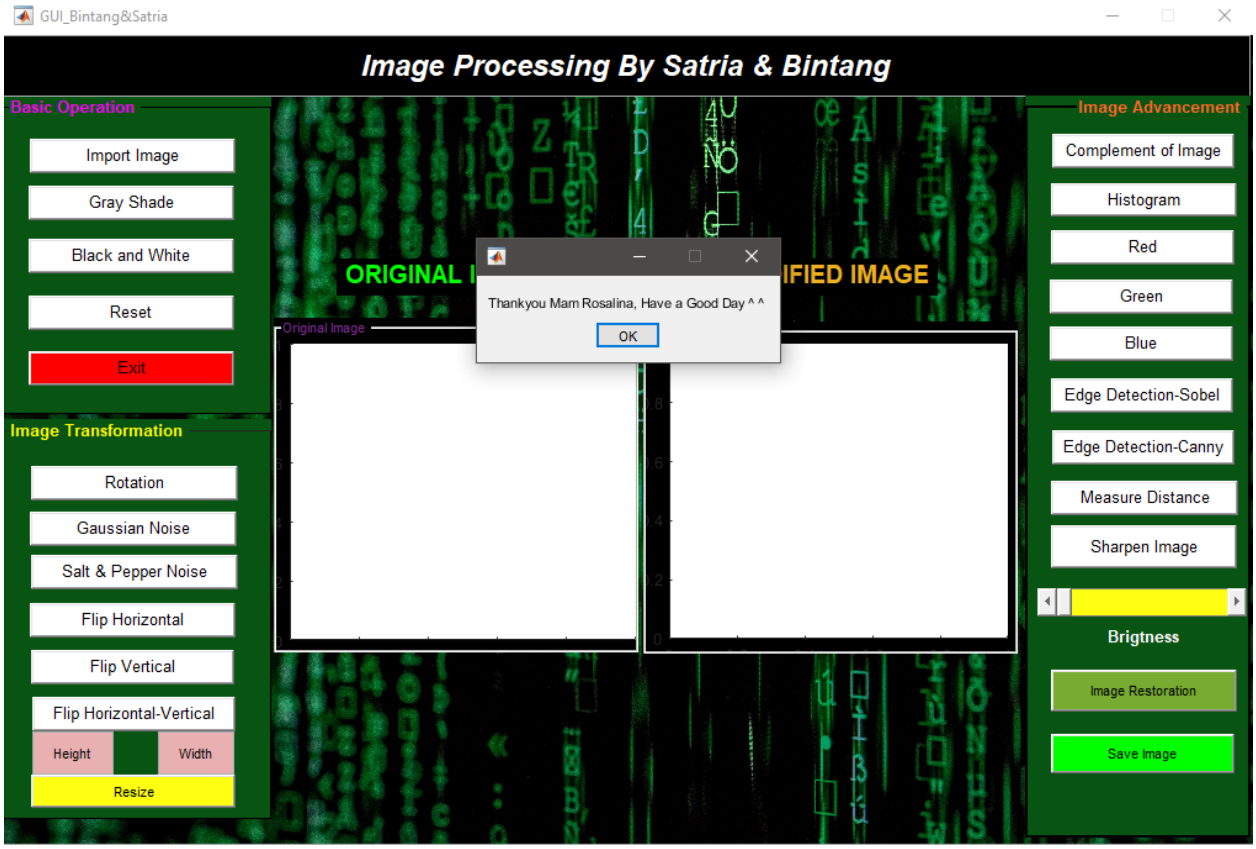
The function code for reset button actually didn't need any function name, we just called/show the original image to the modified image so it looks like already reset but actually just showing the original image again.



5. Exit Button

This button is used to stop the program

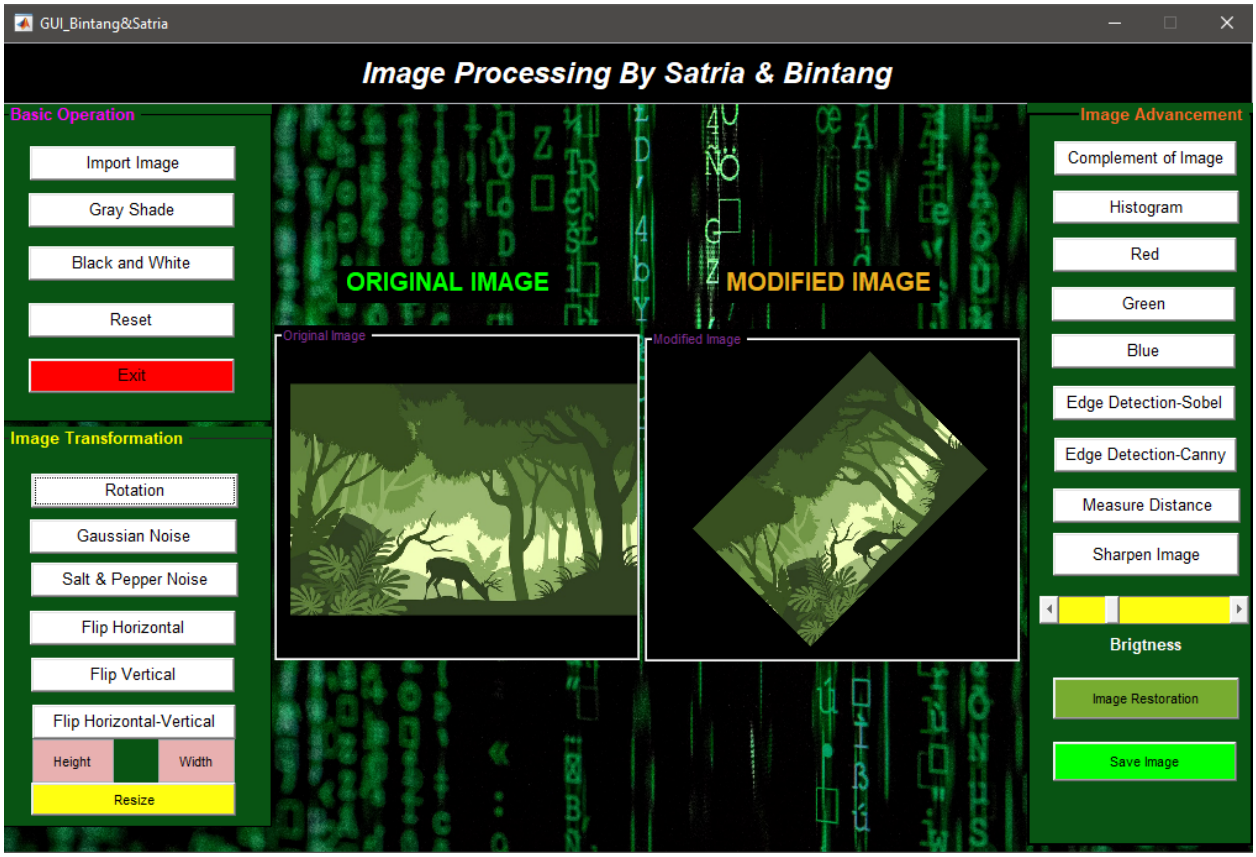
In exit function we have some message shown like alert, we used msgbox(‘’) to shown it, and do pause for 5 second before the program close, the function for close the program is only “close()”



6. Rotation Button

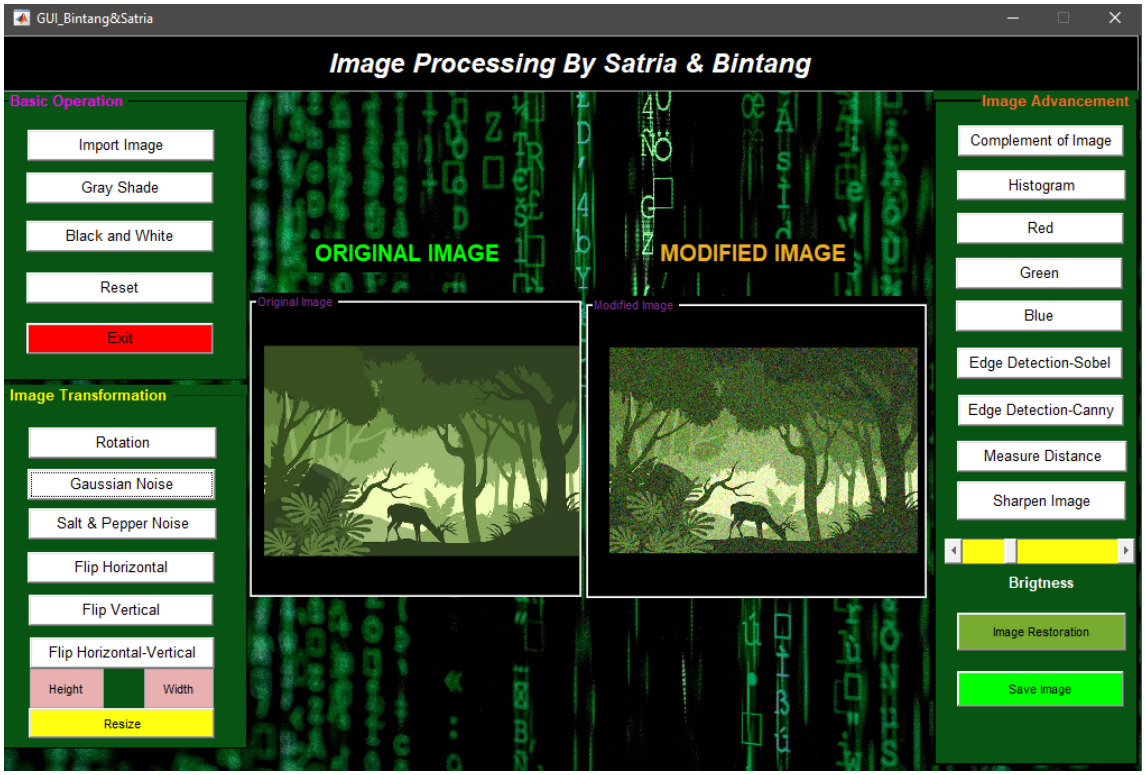
This button is used to rotate the image. In our program, it is 45 degrees counterclockwise.

For the function to rotate an image we use “imrotate(a,degree)” so the “a” is the variable from the original image and “degree” means how much degree you want to rotate the image. For me, We use 45 degrees to rotate the image counterclockwise.



7. Gaussian Noise Button

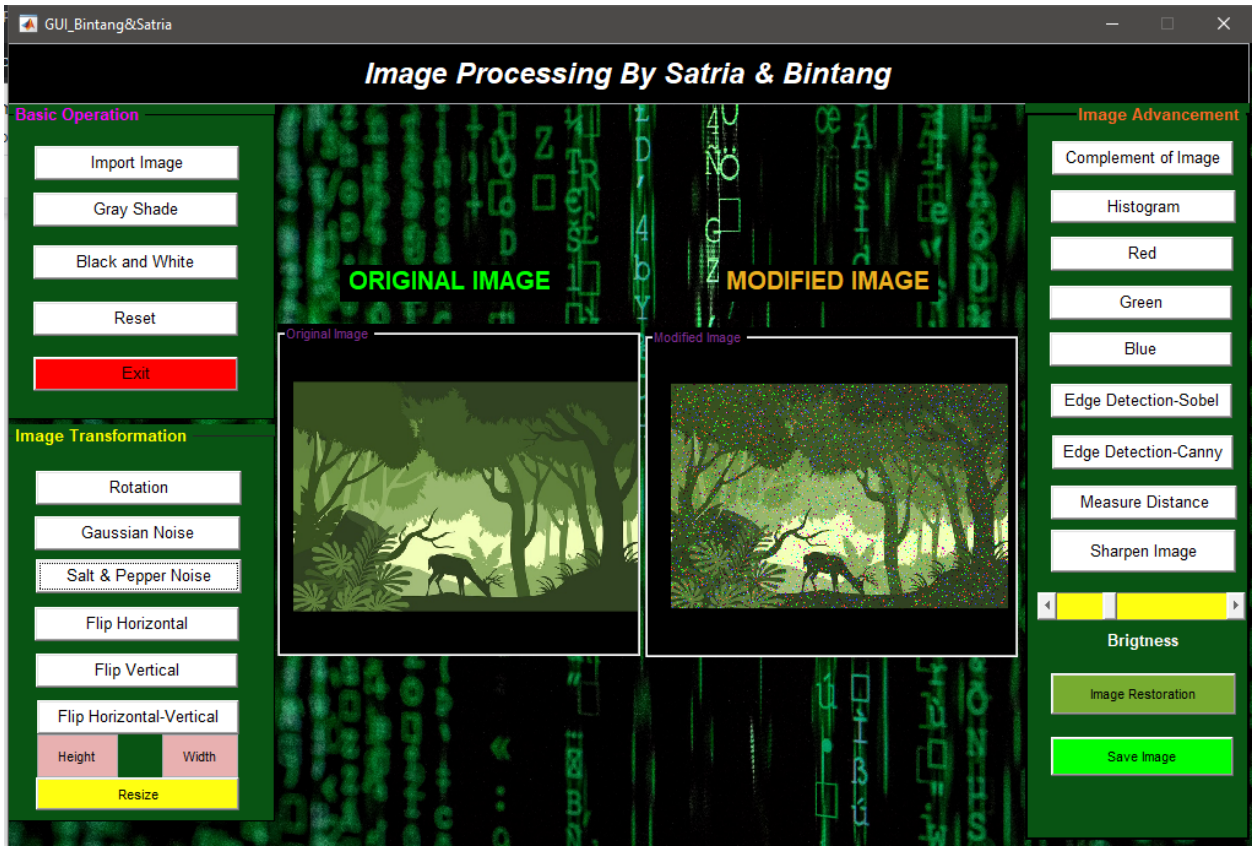
This button is used to apply Gaussian Noise into our image. Gaussian noise is a statistical noise having a probability density function equal to normal distribution. The function is “imnoise(a, 'gaussian');”. So, We insert parameter “a” with the image, and the Gaussian Noise is implemented.



8. Salt & Pepper Noise Button

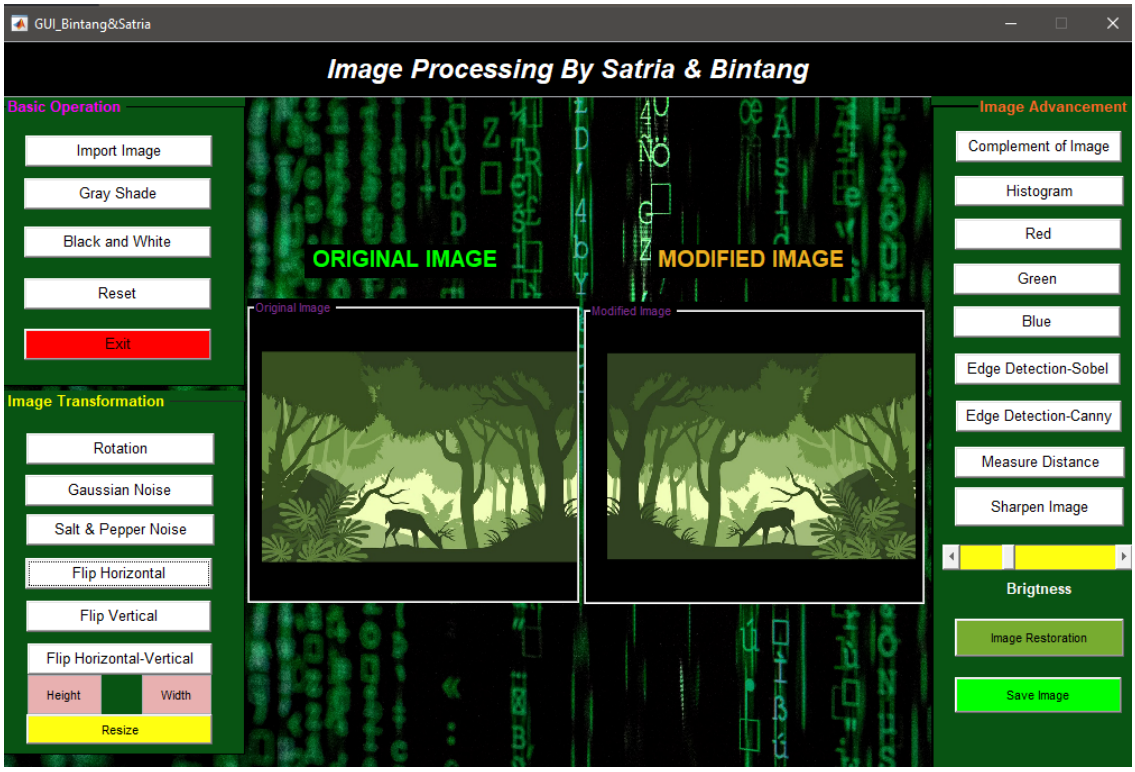
This button is used to apply Salt & Pepper Noise into the image. Salt & Pepper Noise is a form of noise typically seen on images which represents itself as randomly occurring white and black pixels.

Function use is “imnoise(a, 'salt & pepper')” So, We insert parameter “a” with the image, and the Salt & Pepper Noise is implemented.



9. Flip Horizontal Button

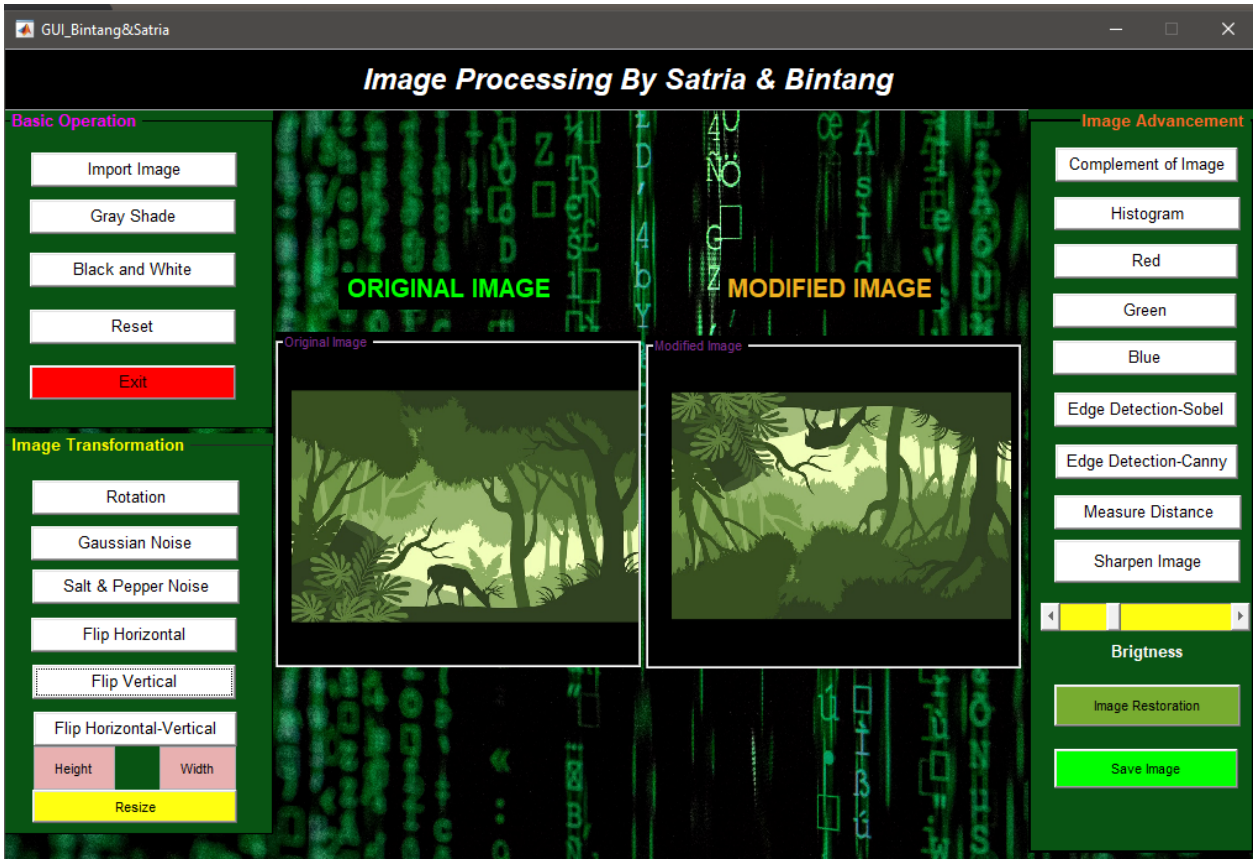
This button is used to flip the image horizontally. The function is “flipdim(I,2);”. We only need to insert the parameter I with the image, and it will be flipped horizontally.



10.Flip Vertical Button

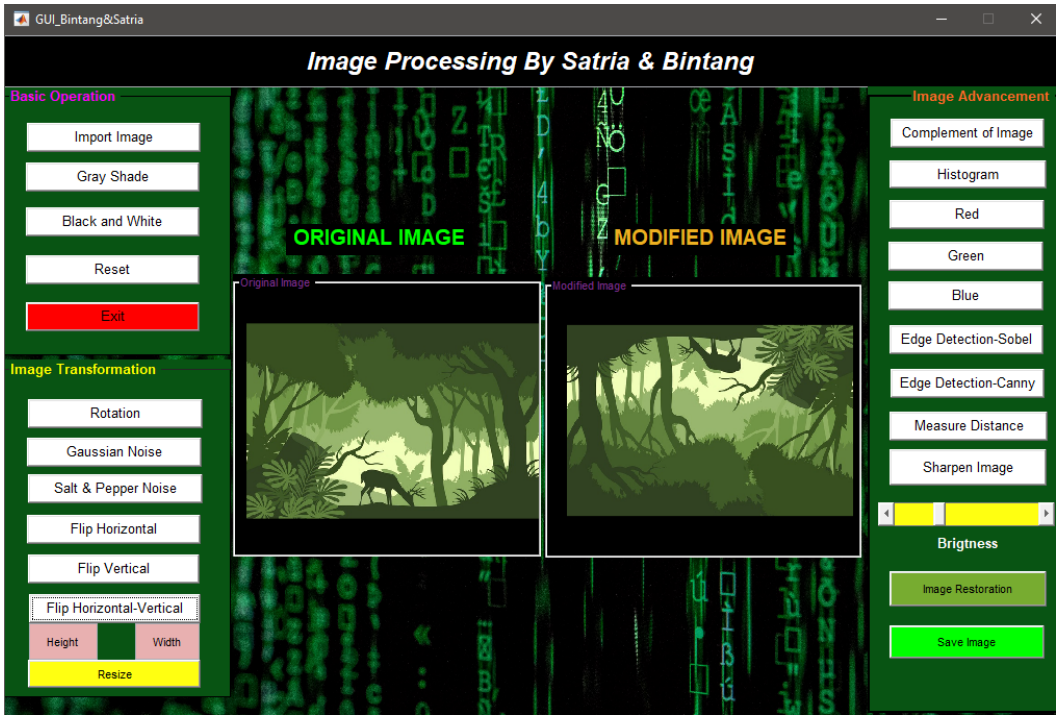
This button is used to flip the image vertically.

In vertical flip function there is a function name to use flip called “flipdim(I,1)” variable I is for the original image and 1 is for the vertical image.



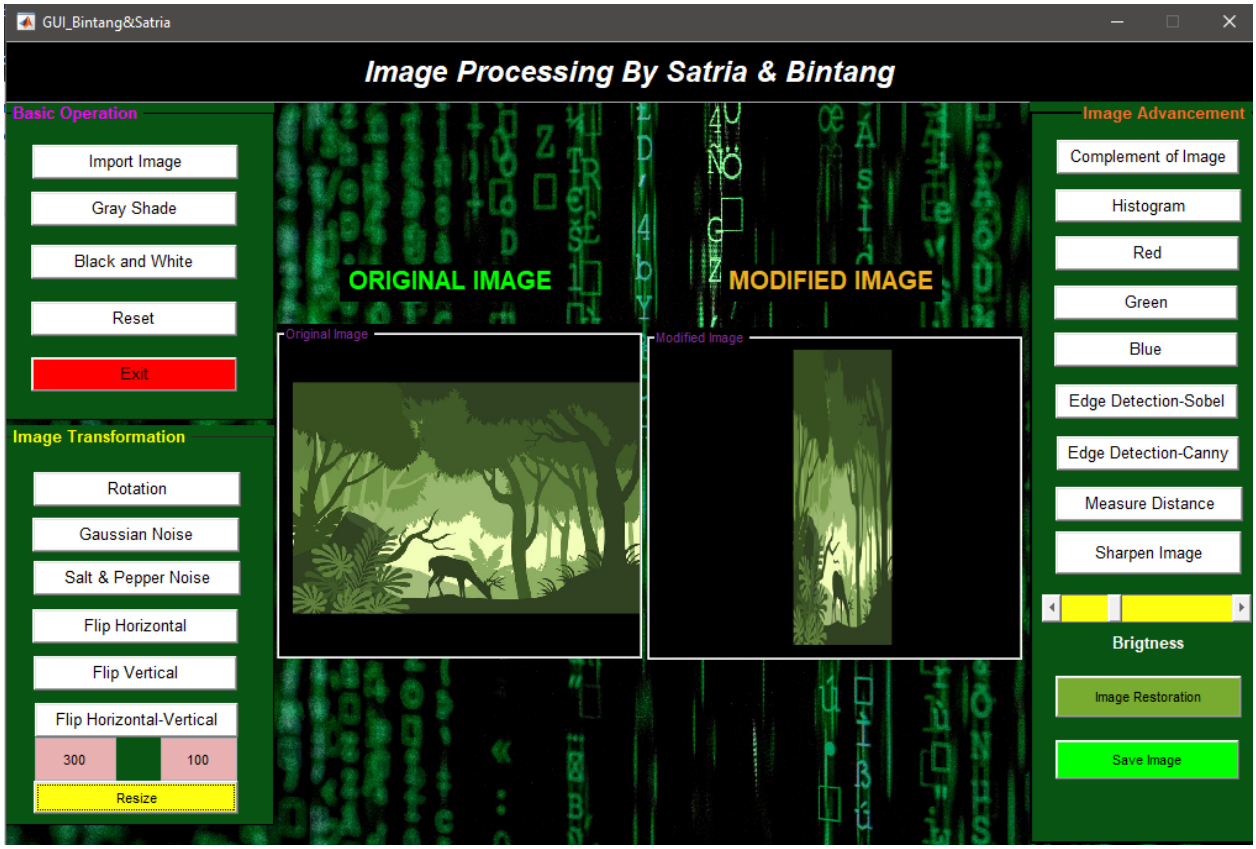
11. Flip Horizontal & Vertical Button

This button is used to flip the image both horizontally and vertically.
And the last flip is vertical and horizontal, in this function we use 2 function flipdim, The first for vertical and the second for horizontal



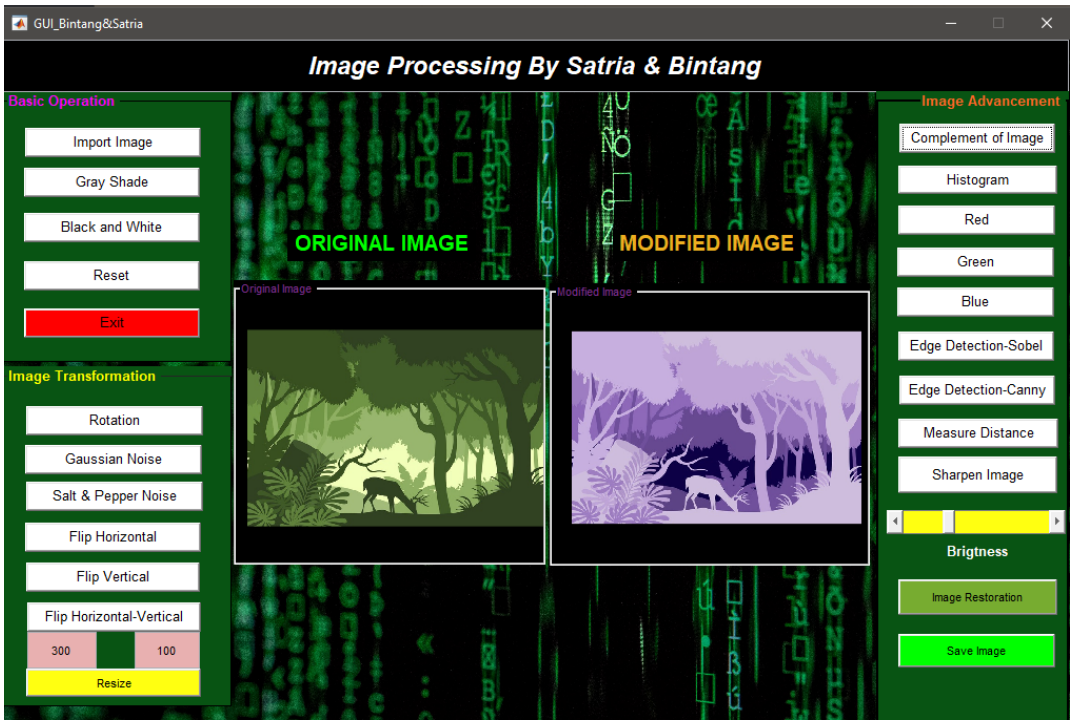
12. Resize Button

This button is used to transform the size of the image based on the width and height that we specified. The function for this is called “imresize(a,[w,h]);”. The variable w & h should be of type double. The parameter “a” is the image that we want to process. So, this way the image will be transformed into the width and height that we want.



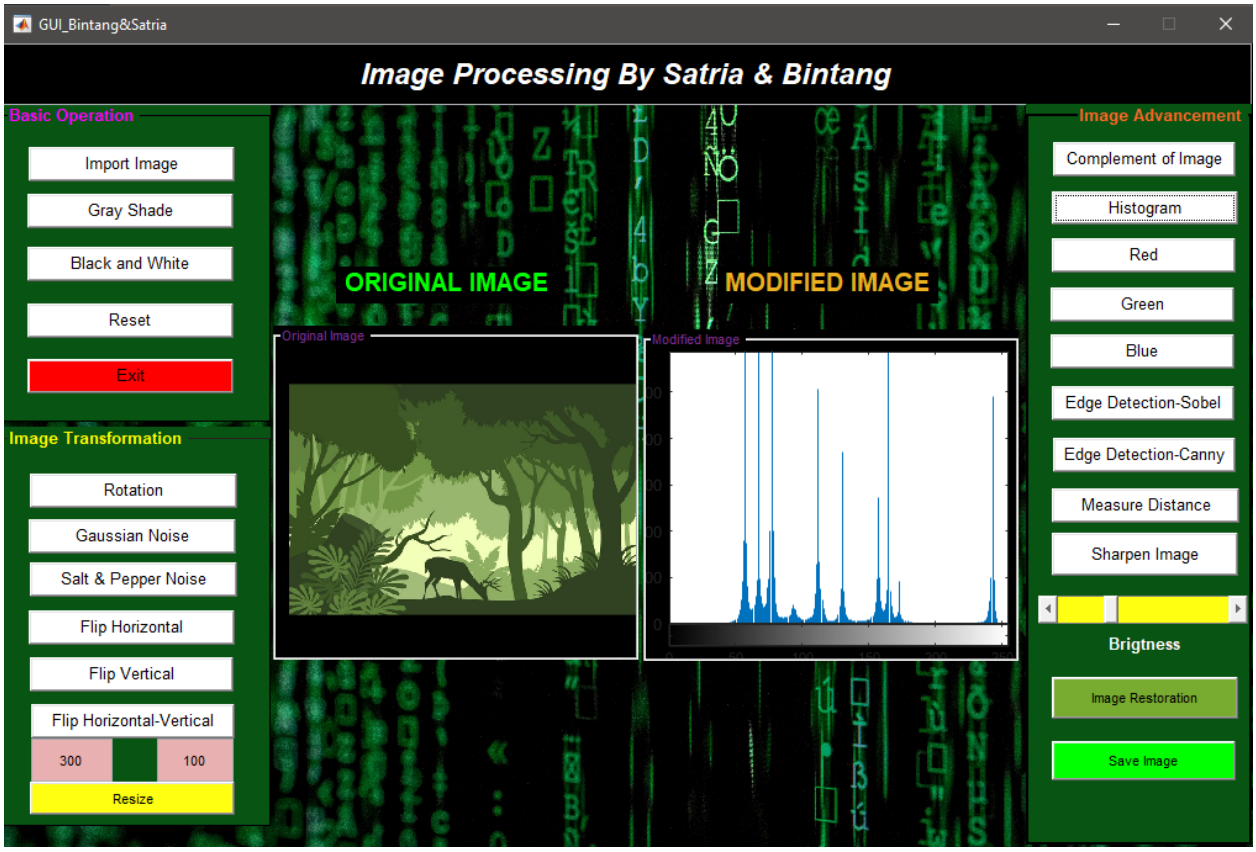
13. Complement of Image Button

This button is used to apply complement into the image colors. In the complement of a binary image, zeros become ones and ones become zeros. Black and white are reversed. In the complement of a grayscale or color image, each pixel value is subtracted from the maximum pixel value supported by the class (or 1.0 for double-precision images).the function use to make implement complement image is “imcomplement(a)” where “a” is the variable from original image



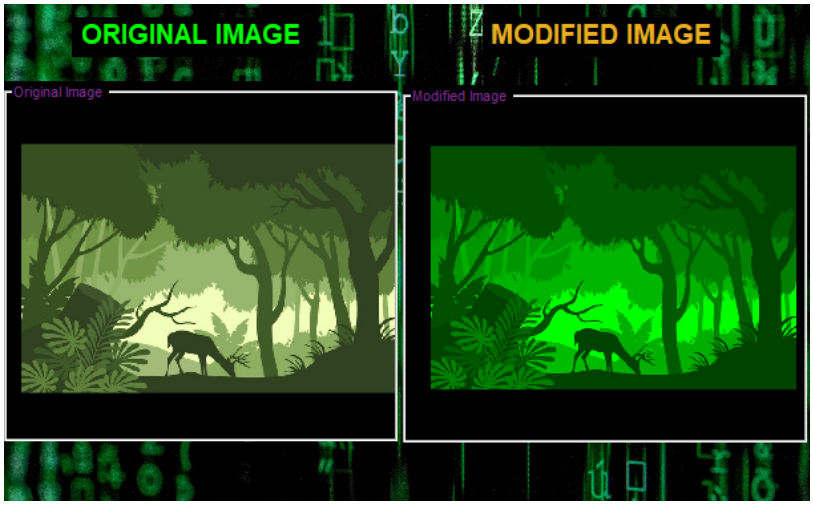
14. Histogram Button

This button is used to show a histogram of the pixel intensity values found in the image. For showing histogram we must change the original image into grayscale first using “rgb2gray()” then only show the image but not use imshow() anymore however use imhist() to show histogram.



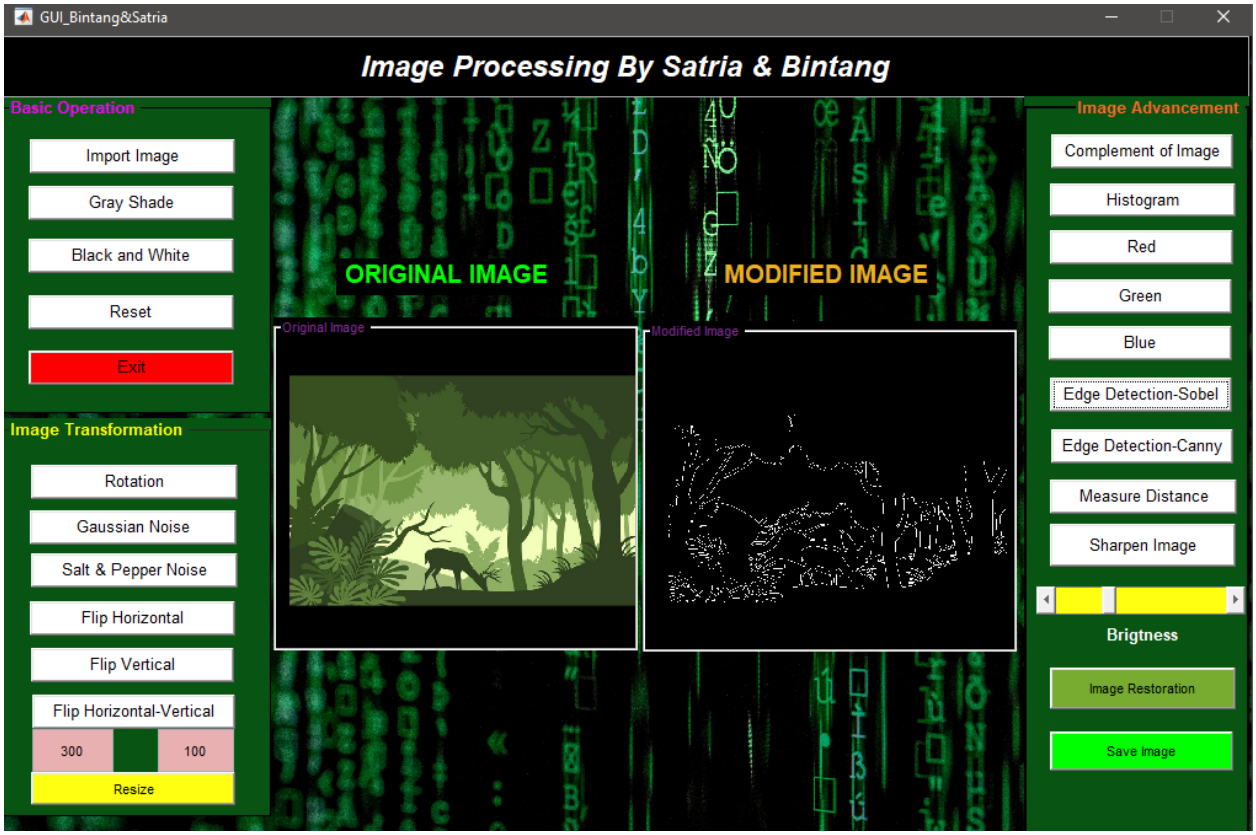
15. Red, Green, Blue Button

These 3 buttons are used to change the image color into Red, Green, or Blue based on the original image color intensity.



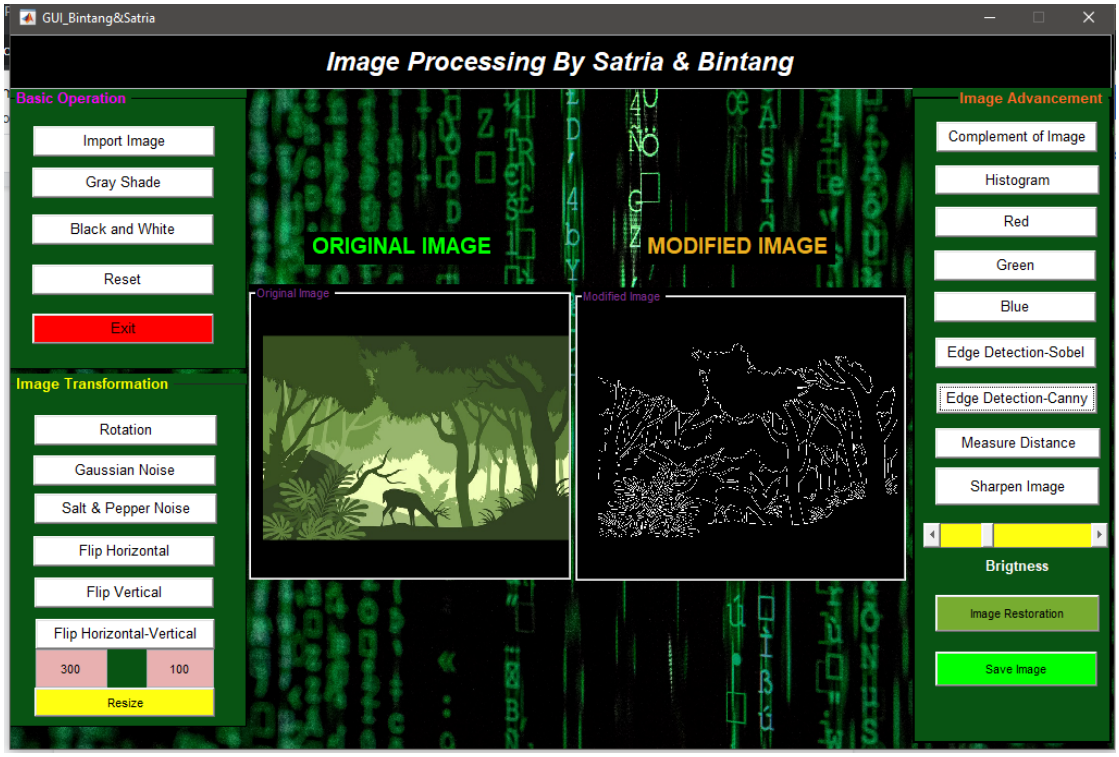
16. Edge Detection Sobel Button

This button is used to apply Edge Detection Sobel into the image. Sobel detection refers to computing the gradient magnitude of an image using 3x3 filters. Where "gradient magnitude" is, for each pixel, a number giving the absolute value of the rate of change in light intensity in the direction that maximizes this number.



17. Edge Detection Canny Button

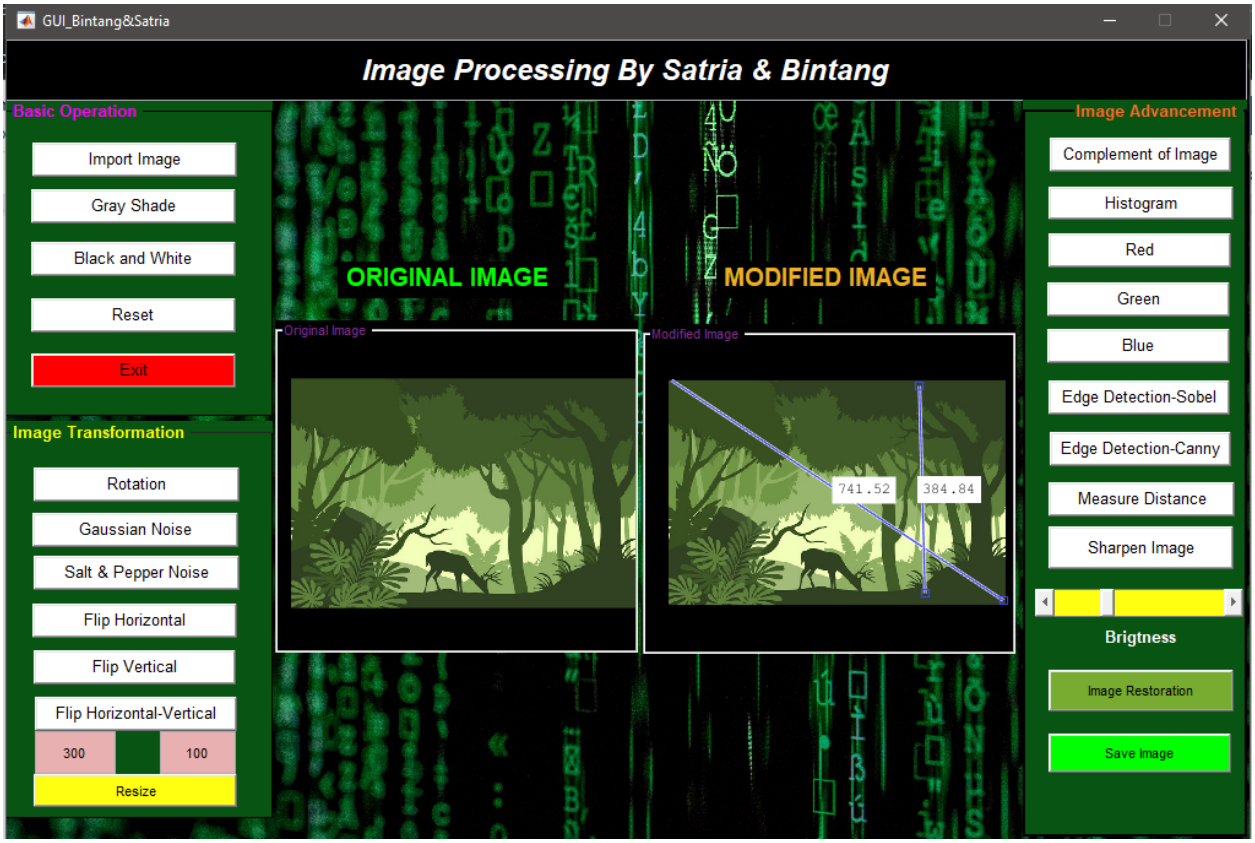
This button is used to apply Edge Detection Canny into the image. This edge detection is better than the Sobel, because this edge detection goes a bit further by removing speckle noise with a low pass filter first, then applying a Sobel filter, and then doing non-maximum suppression to pick out the best pixel for edges when there are multiple possibilities in a local neighborhood.



18. Measure Distance Button

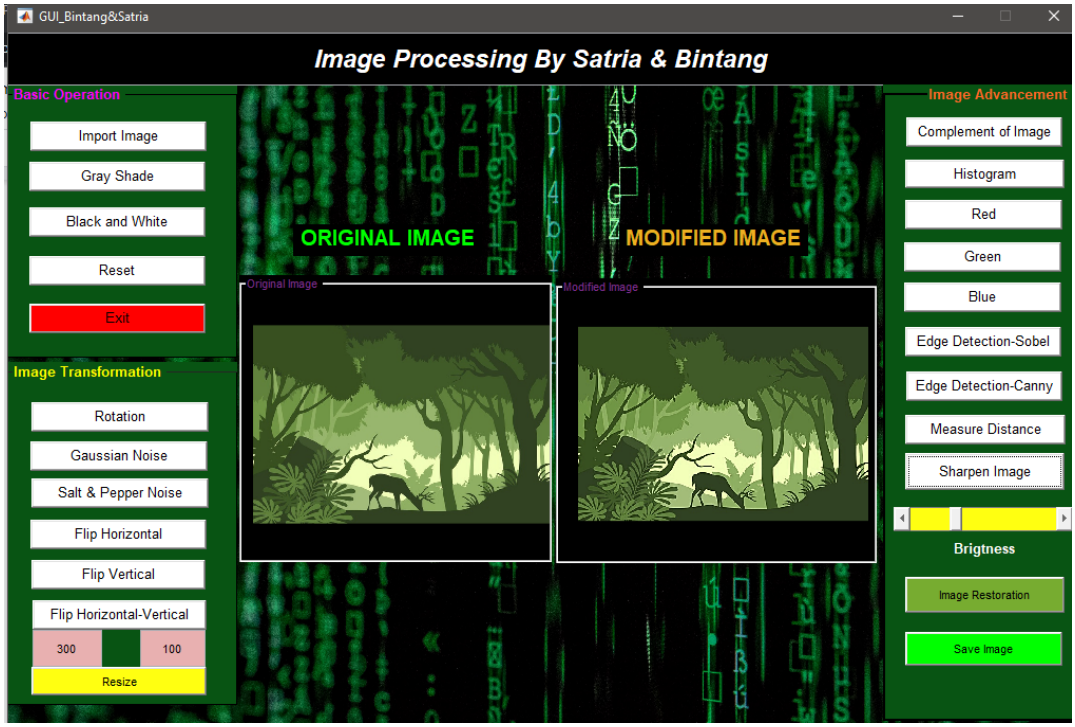
This button is used to measure the distance between one pixel and another pixel in the image based on what we specified.

We use 2 function to show measure distance that are “imdistline()” for getting line in image and “getDistance()” to measure distance



19. Sharpen Image Button

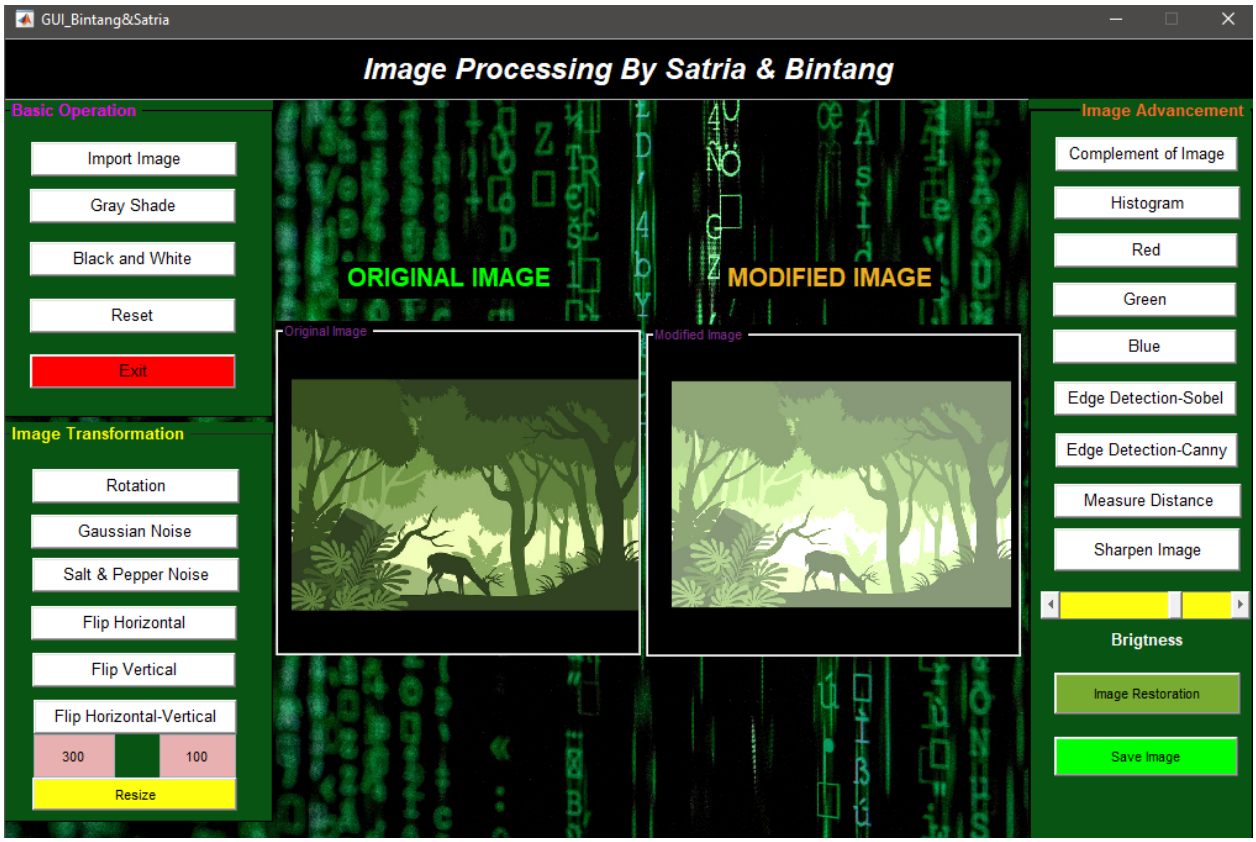
This button is used to give a sharper appearance to the image. The function for this is called “(a,'Radius',2,'Amount',1);”. Parameter “a” is the image to be sharpened, ‘Radius’ is the specified size of the region around edge pixels that will be affected by the image sharpening, parameter “Amount” is the strength of the sharpening effect.



20. Brightness Slider

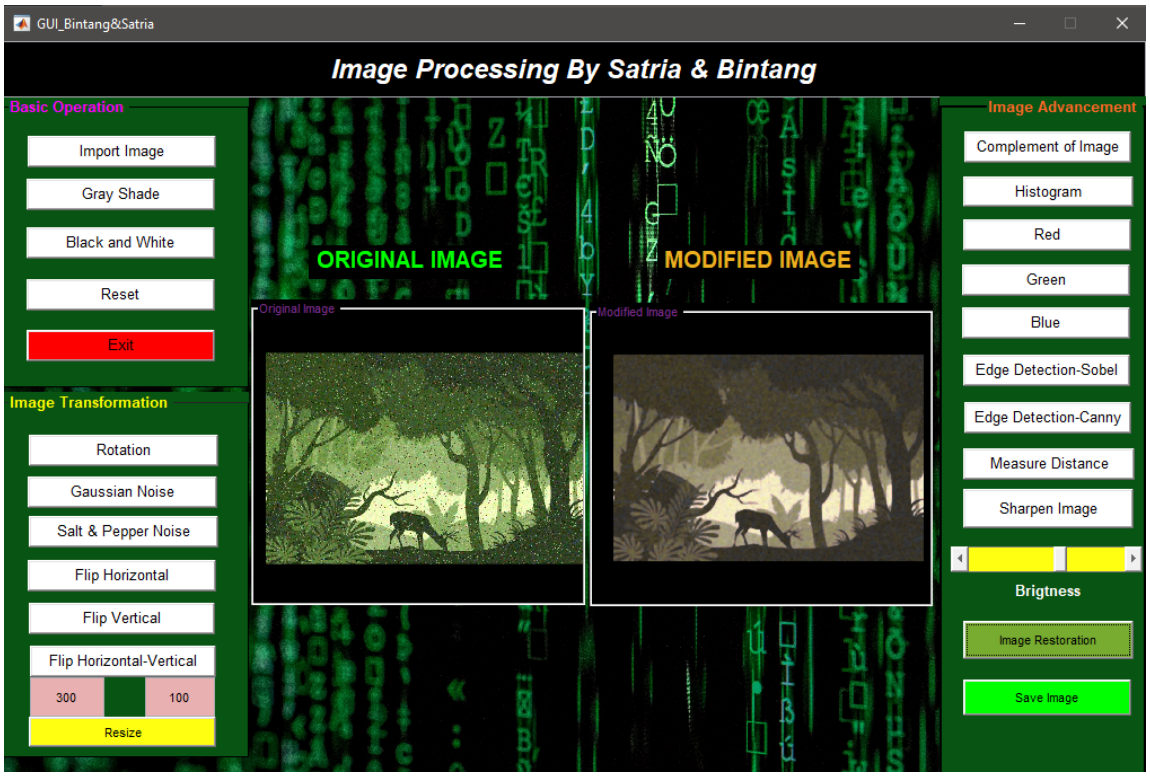
This slider performs brightness transformation to the image. The more to the right it is slid, the brighter the image. The more to the left it is slid, the closer the brightness to its original brightness.

For brightness we only add white color to image so it shows like brightening but actually we use the code “0.5*get(hObject, 'value')-0.5” to apply white color based on slider move.



21. Image Restoration Button

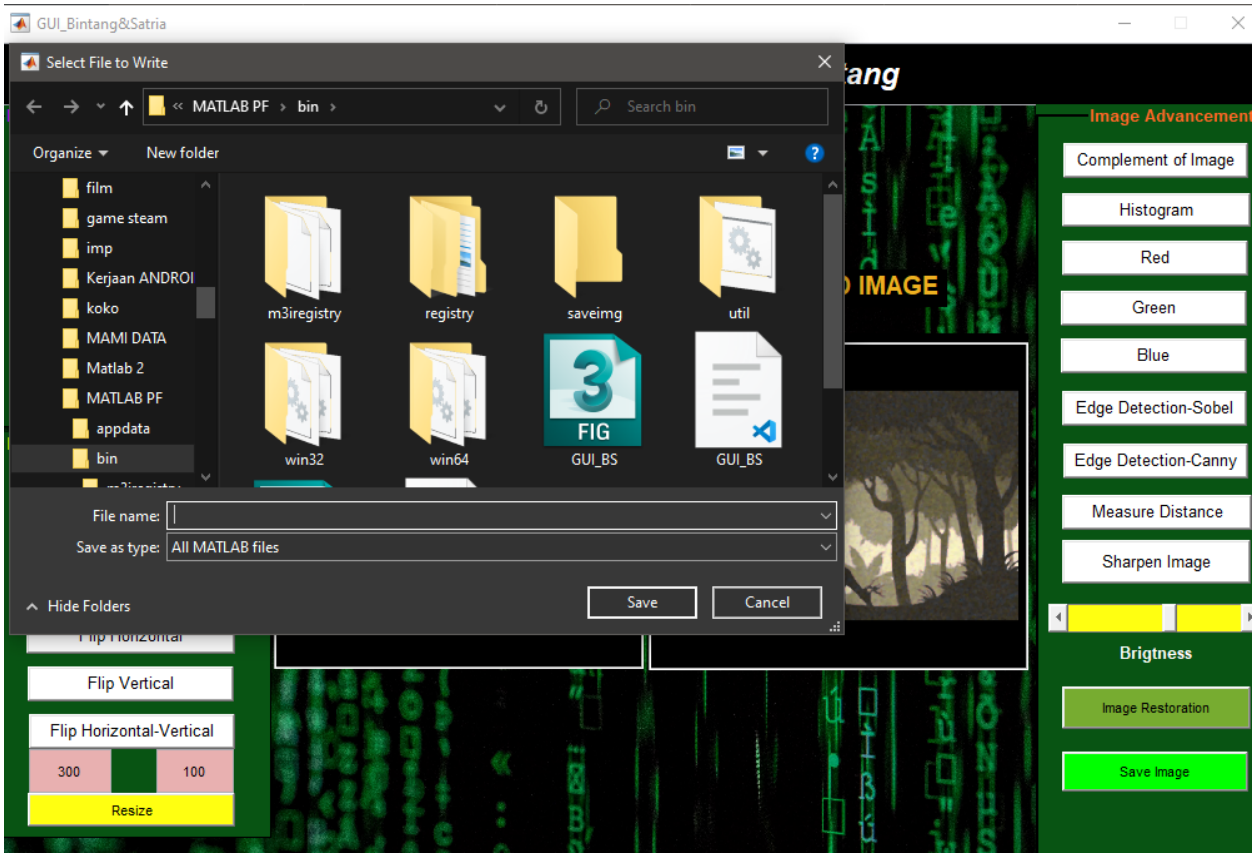
This button is used to restore all noises found in the image. The function for this is called “medfilt3(a, [5 5 3]);”. The parameter “a” is our processed image that is in the MODIFIED IMAGE section. The array [5 5 3] is the specified size of filters that we want when doing image restoration. We use medfilt3 which means we use median filtering technique. The median filter works by moving through the image pixel by pixel, replacing each value with the median value of neighbouring pixels.



22. Save Button

This button is used to save the processed image into our file explorer. The function for this is called “imwrite(a, complete_name);”. The parameter “complete_name” is the path and the filename that we want to give to our processed image. The parameter “a” is the processed image that is still in the MODIFIED IMAGE section.

Also for this function we use 2 function that are “uiputfile()” to input file and “fullfile()” to make the file into savable file and only imwrite() to save in our directory path



CODE MATLAB

```
function varargout = GUI_BS(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @GUI_BS_OpeningFcn, ...
                   'gui_OutputFcn',  @GUI_BS_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
% End initialization code - DO NOT EDIT


% --- Executes just before GUI_BS is made visible.
function GUI_BS_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   unrecognized PropertyName/PropertyValue pairs from the
%            command line (see VARARGIN)

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

% Update handles structure
guidata(hObject, handles);

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

```
buat_axes = axes('unit', 'normalized', 'position', [0 0  1 1]);
backgroundnya = imread('bg15.jpg');
imagesc(backgroundnya);
set(buat_axes, 'handlevisibility', 'off', 'visible', 'off')

% --- Outputs from this function are returned to the command line.
function varargout = GUI_BS_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 pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)%Histogram
% 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)
a=getappdata(0,'a');
input=a;
input=rgb2gray(input);
axes(handles.axes2);
imhist(input);

% --- Executes on button press in pushbutton9.
function pushbutton9_Callback(hObject, eventdata, handles)%measure distance by
pixel
% hObject    handle to pushbutton9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
l=imdistline();
msgbox('Measured in Pixels');
dist = getDistance(l);


% --- Executes on button press in pushbutton10.
function pushbutton10_Callback(hObject, eventdata, handles)%RGB to Red
```

```
% hObject    handle to pushbutton10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
a=getappdata(0,'a');
red=a;
red(:,:,2:3)=0;
setappdata(0,'filename', red);
setappdata(0,'ImRotation', red);
axes(handles.axes2);
imshow(red);

% --- Executes on button press in pushbutton1111.
function pushbutton1111_Callback(hObject, eventdata, handles) %rotate 45degree
% hObject    handle to pushbutton1111 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
a=getappdata(0, 'a');
rotate=imrotate(a,45);
axes(handles.axes2);
imshow(rotate);

% --- Executes on button press in pushbutton12.
function pushbutton12_Callback(hObject, eventdata, handles)%Flip horizontal
% hObject    handle to pushbutton12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
I=getappdata(0,'a');
I2=flipdim(I,2);
axes(handles.axes2);
imshow(I2);


% --- Executes on button press in pushbutton13.
function pushbutton13_Callback(hObject, eventdata, handles)%Complement of Image
% hObject    handle to pushbutton13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
a=getappdata(0,'a');
IM2=imcomplement(a);
axes(handles.axes2);
```

```
imshow(IM2);

% --- Executes on button press in pushbutton14.
function pushbutton14_Callback(hObject, eventdata, handles)%RGB to Green
% hObject    handle to pushbutton14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
a=getappdata(0,'a');
green=a;
green(:,:,1)=0;
green(:,:,3)=0;
setappdata(0,'filename', green);
setappdata(0,'ImRotation', green);
axes(handles.axes2);
imshow(green);


% --- Executes on button press in pushbutton15.
function pushbutton15_Callback(hObject, eventdata, handles)%RGB to Blue
% hObject    handle to pushbutton15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
a=getappdata(0,'a');
blue=a;
blue(:,:,1)=0;
blue(:,:,2)=0;
setappdata(0,'filename', blue);
setappdata(0,'ImRotation', blue);
axes(handles.axes2);
imshow(blue);


% --- Executes on button press in pushbutton16.
function pushbutton16_Callback(hObject, eventdata, handles)%Salt pepper noise
% hObject    handle to pushbutton16 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
a=getappdata(0,'a');
noise=imnoise(a, 'salt & pepper');
axes(handles.axes2);
```

```
imshow(noise);

% --- Executes on button press in pushbutton17.
function pushbutton17_Callback(hObject, eventdata, handles)%Gaussian noise
% hObject    handle to pushbutton17 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
a=getappdata(0,'a');
noise=imnoise(a, 'gaussian');
axes(handles.axes2);
imshow(noise);


% --- Executes on button press in pushbutton18.
function pushbutton18_Callback(hObject, eventdata, handles)%Flip Vertical
% hObject    handle to pushbutton18 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
I=getappdata(0,'a');
I3=flipdim(I,1);
axes(handles.axes2);
imshow(I3);


% --- Executes on button press in pushbutton19.
function pushbutton19_Callback(hObject, eventdata, handles)%Flip horizontal & Vertical
% hObject    handle to pushbutton19 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
I=getappdata(0,'a');
I2=flipdim(I,2);
I3=flipdim(I,1);
I4=flipdim(I3,2);
axes(handles.axes2);
imshow(I4);

% --- Executes on button press in pushbutton20.
function pushbutton20_Callback(hObject, eventdata, handles)%edge Detection Canny
% hObject    handle to pushbutton20 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
I=getappdata(0,'a');
I=rgb2gray(I);
BW2=edge(I,'canny');
axes(handles.axes2);
imshow(BW2);


% --- Executes on button press in pushbutton21.
function pushbutton21_Callback(hObject, eventdata, handles)%Edge detection Sobel
% hObject    handle to pushbutton21 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
I=getappdata(0,'a');
I=rgb2gray(I);
BW1=edge(I,'sobel');
axes(handles.axes2);
imshow(BW1);

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)% import Image
% 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)
a=uigetfile('*');
filename=a;
setappdata(0,'filename',filename);
a=imread(a);
axes(handles.axes1);
imshow(a);
setappdata(0,'a',a);
setappdata(0,'filename',a);
plot(handles.axes1,'a');


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)%grayscale
% 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)
a=getappdata(0,'a');
a_gray=rgb2gray(a);
setappdata(0,'filename', a_gray);
axes(handles.axes2);
imshow(a_gray);


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
a=getappdata(0,'a');
adjust=imadjust(a);
axes(handles.axes2);
imshow(adjust);


% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles) %Black and White
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
a=getappdata(0,'a');
a_bw=im2bw(a,.57);
axes(handles.axes2);
imshow(a_bw);
setappdata(0,'filename',a_bw);


% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)%Reset button
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
a=getappdata(0,'a');
imshow(a);
```

```
% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)%exit button
% 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)
msgbox('Thankyou Mam Rosalina, Have a Good Day ^ ^');
pause(5);
close();
close();

% --- Executes on button press in pushbutton82.
function pushbutton82_Callback(hObject, eventdata, handles)%resize based on Height
and Width
% hObject    handle to pushbutton82 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
a=getappdata(0,'a');
w=str2double(get(handles.edit3,'string'));
h=str2double(get(handles.edit4,'string'));
axes(handles.axes2);
y=imresize(a,[w,h]);
imshow(y);


% --- Executes on slider movement.
function slider4_Callback(hObject, eventdata, handles)%Brightness
% hObject    handle to slider4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
a=getappdata(0,'a');
val=0.5*get(hObject, 'value')-0.5;
imbright= a+val;
axes(handles.axes2);
imshow(imbright);
guidata(hObject, handles);

% --- Executes on button press in pushbutton83.
```

```
function pushbutton83_Callback(hObject, eventdata, handles)%sharpen image
% hObject    handle to pushbutton83 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
a=getappdata(0,'a');
b = imsharpen(a,'Radius',2,'Amount',1);
axes(handles.axes2);
imshow(b)


% --- Executes on button press in pushbutton84.
function pushbutton84_Callback(hObject, eventdata, handles)%save Image
% hObject    handle to pushbutton84 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
a= getimage(handles.axes2);
[filename, foldername] = uiputfile();
complete_name = fullfile(foldername, filename);
imwrite(a, complete_name);


% --- Executes on button press in pushbutton85.
function pushbutton85_Callback(hObject, eventdata, handles)%image restoration
% hObject    handle to pushbutton85 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
a=getimage(handles.axes1);
Output_med = medfilt3(a, [5 5 3]);
axes(handles.axes2);
imshow(Output_med);
```