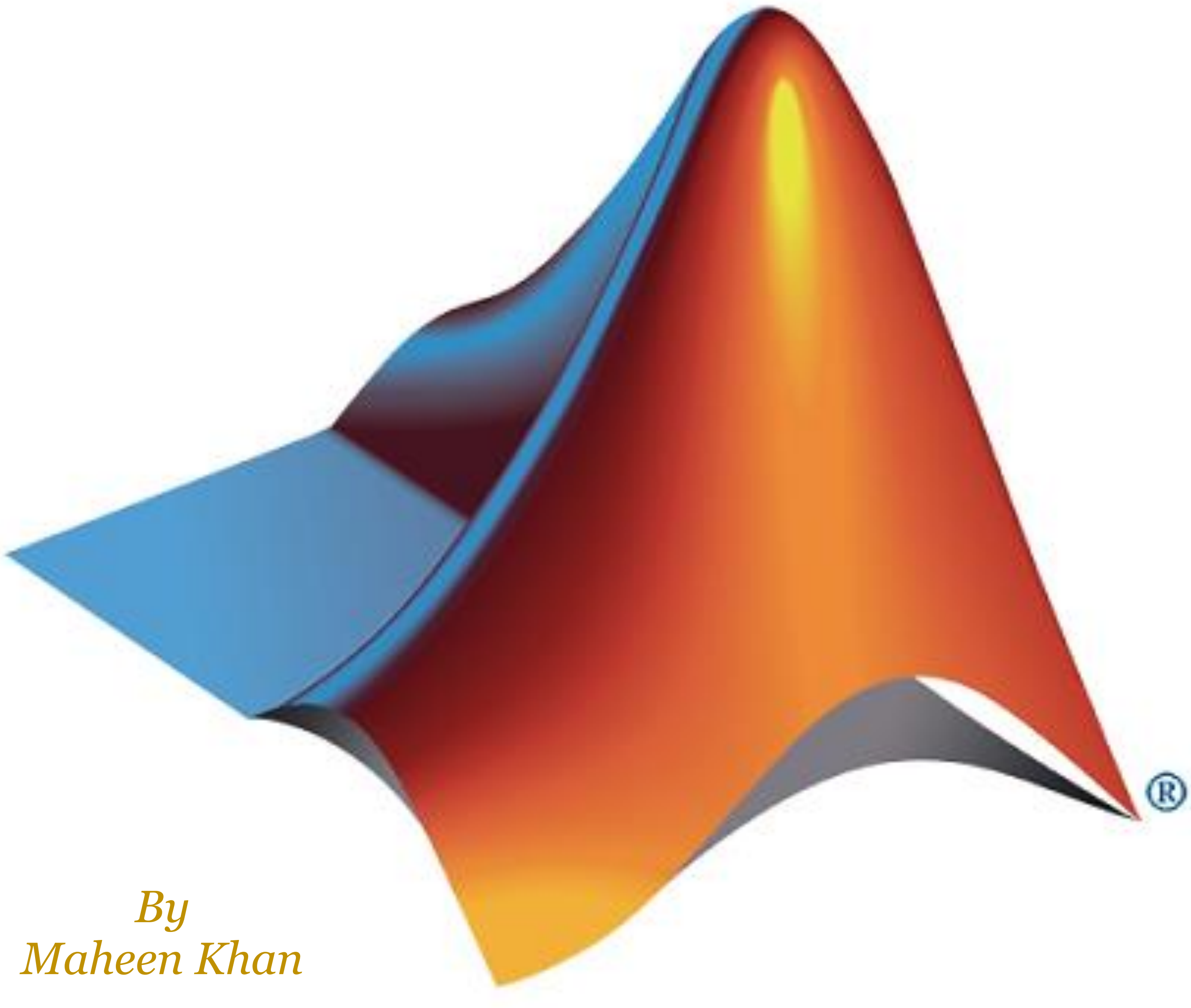
MATLAB

GUI Project

EDGE DETECTION IN IMAGE PROCESSING



By
Maheen Khan

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

% Last Modified by GUIDE v2.5 22-Oct-2023 10:58:58

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Edge_OpeningFcn, ...
                   'gui_OutputFcn',  @Edge_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 Edge is made visible.
function Edge_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 Edge (see VARARGIN)

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

set(handles.axes1,'visible','off')
set(handles.axes3,'visible','off')

% Update handles structure
guidata(hObject, handles);


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


% --- Outputs from this function are returned to the command line.
function varargout = Edge_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)
global GRAY
[file,path] = uigetfile({'*.*'});
img_loc= fullfile(path, file);
RGB= imread(img_loc);
axes(handles.axes1);
imshow(RGB);
GRAY= rgb2gray(RGB);


% --- 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 GRAY
K= get(handles.popupmenu1,'value');

switch K
    case 2
        EDGE= edge(GRAY, 'Sobel');
    case 3
```

```
             EDGE= edge(GRAY, 'Prewitt');
        case 4
            EDGE= edge(GRAY, 'Roberts');
        case 5
             EDGE= edge(GRAY, 'Log');
        case 6
             EDGE= edge(GRAY, 'Canny');
        case 7
            EDGE= edge(GRAY, 'LAPLACIAN TECHNIQUE');
        otherwise
            disp('No filter');
end

axes(handles.axes3);
if isequal(K,1)
    imshow(GRAY);
else
    imshow(EDGE);
end
% --- Executes on selection change in popupmenu1.
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 popupmenu1 contents as
%        contents{get(hObject,'Value')} returns selected item from popupmenu1


% --- 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,'defaultUicontrolBackgrou
    set(hObject,'BackgroundColor','white');
end
```

Published with MATLAB® R2013a

We have used MATLAB GUI to create using GUIDE for image edge detection. Here are explanations for ten key operations or functions in this code:

❖ **Opening Function (`Edge_OpeningFcn`):**
- This function is automatically executed when the GUI is opened.
- It initializes some properties of the GUI, such as making two axes invisible.

❖ **Output Function (`Edge_OutputFcn`):**
- This function specifies the output arguments that are returned when the GUI is closed.
- In this case, it returns the `handles.output` property.

❖ **Push Button 1 (`pushbutton1_Callback`):**
- This callback function is executed when "Open Image" button is clicked.
- It opens a file dialog for selecting an image file.
- It reads and displays the selected image in `axes1`.

❖ **Push Button 2 (`pushbutton2_Callback`):**
- This callback function is executed when the "Process" button is clicked.
- It retrieves the selected filter from a `popupmenu`.Depending on the filter chosen, it applies edge detection using functions like `edge` with different methods (e.g., Sobel, Prewitt) on the grayscale image (`GRAY`).
- It displays the resulting edge-detected image in `axes3`.

❖ **Axes 1:**
- This callback function is executed when the user selects an option from the dropdown menu.
- It retrieves the selected filter method from the dropdown menu.

❖ **Axes 2:**
- This function is executed during the creation of the `popupmenu1`.
- It sets the background color of the dropdown menu to white.

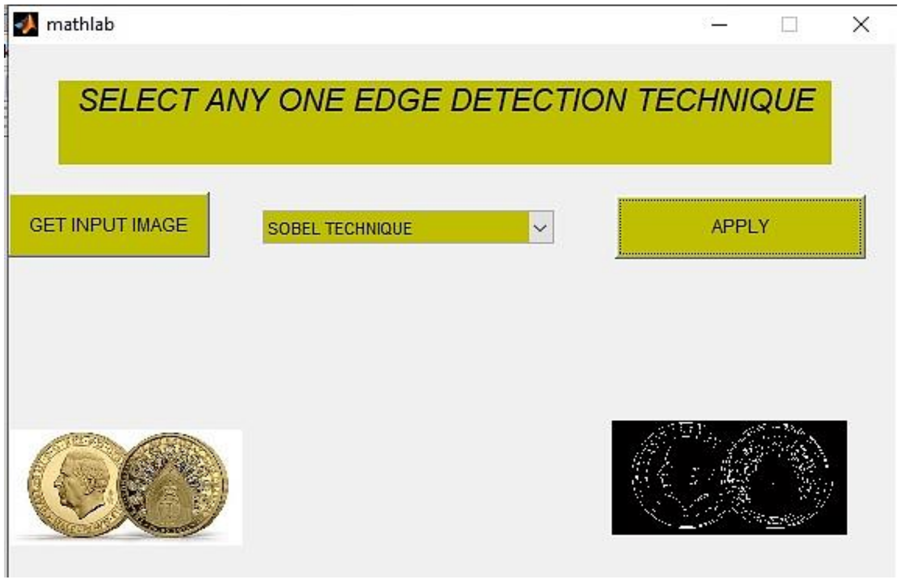
These functions collectively allow the user to open an image, choose an edge detection method from a dropdown menu, and apply edge detection to the selected image. The resulting edge-detected image is displayed in the GUI.

DIFFERENTIATING DIFFERENT TECHNIQUES USED IN OUR PROJECT:

Sobel, Canny, Prewitt, Roberts, and Laplacian are different techniques used for edge detection in image processing. Here are the key differences between these techniques in MATLAB:

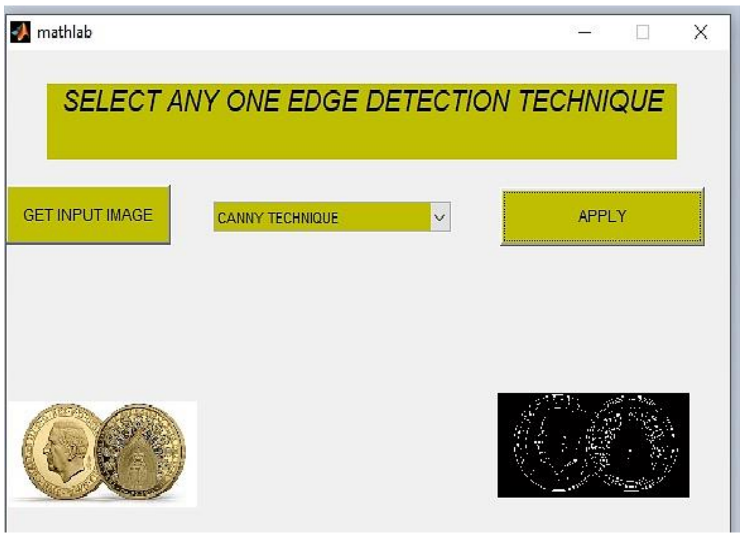
Sobel Operator:

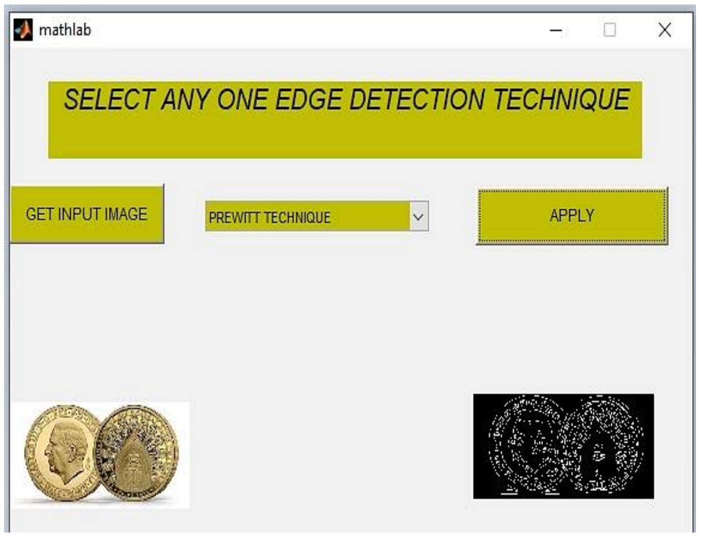
Sobel is a gradient-based edge detection method. It calculates the gradient of the image using convolution with two 3x3 kernels (one for horizontal changes and one for vertical changes).
It's relatively simple and computationally efficient. It highlights edges by emphasizing changes in pixel intensity in both horizontal and vertical directions.



Canny Edge Detector:

Canny is a multi-stage edge detection method. It includes Gaussian smoothing, gradient calculation (usually with Sobel), non-maximum suppression, and edge tracking by hysteresis. Canny is more robust to noise and can detect thin edges accurately.
It can perform edge tracking to connect edges into continuous lines.



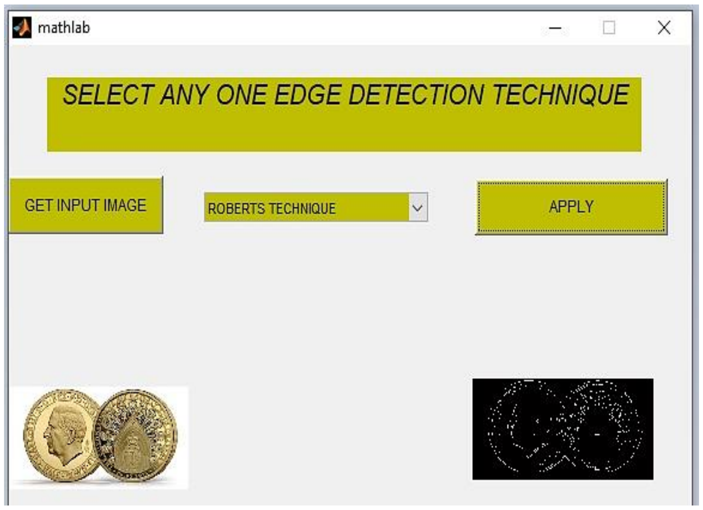


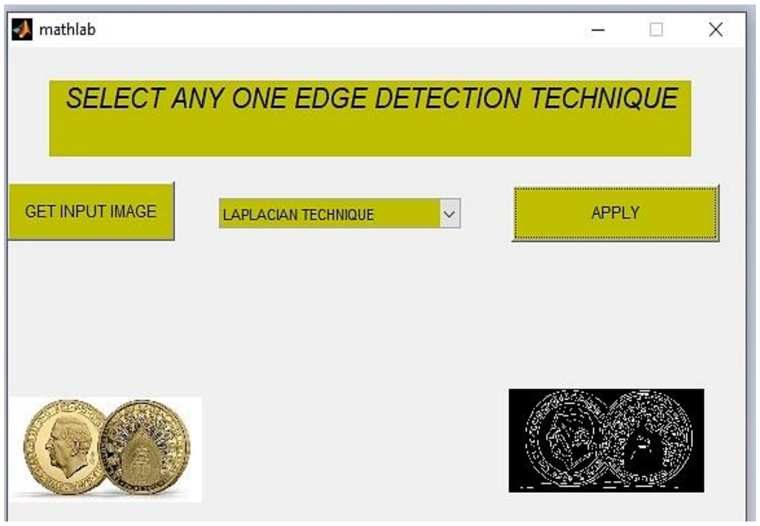
Prewitt Operator:

Prewitt, like Sobel, is a gradient-based edge detection method. It uses 3x3 convolution kernels to calculate horizontal and vertical gradient components. Prewitt focuses on detecting changes in pixel intensity along horizontal and vertical directions.

Roberts Operator:

Roberts is another simple gradient-based edge detection technique. It uses a pair of 2x2 convolution kernels to approximate the gradient in the diagonal directions. Roberts is computationally less expensive but may not perform as well as other methods for certain images.





Laplacian Operator:

Laplacian is a second-order edge detection technique.
It computes the Laplacian of the image to detect regions where the intensity changes (edges) by using a 3x3 kernel.
Laplacian can detect zero-crossings in the image to identify edges but may also detect noise as edges.

In summary, the main differences among these techniques are in their approach to gradient calculation, noise handling, and additional processing stages. Canny is often considered one of the best methods for general-purpose edge detection due to its multi-stage approach, which includes noise reduction and fine edge detection. The choice of technique depends on the specific requirements of your image processing task and the level of noise present in the images.