Ahsanullah University of Science & Technology

Department of Computer Science & Engineering

Course No: CSE4228
Course Title: Digital Image Processing Lab
Project Name: Real Time Edge Detection

Submitted by:
Sakib Ahmed (15.01.04.093)
Shamit Ahmed (15.01.04.101)
Sofia Neherin (15.01.04.107)

ABSTRACT

The main purpose of the project develop the real time detection. It will be useful in medical diagnosis centre, for x-ray or such types of scanning. Here we are executing the real time edge detection with multiple types of algorithms. At last we showing that canny is the best process for edge detection. The continuously captured system will be shown in discrete of edge detection type. Most computer systems normally do with computer vision with gray scale image. We cannot identify the features in gray scale intensities. so we are proposing edge detection with by threshold process with different types of image processing techniques, by using MATLAB software.

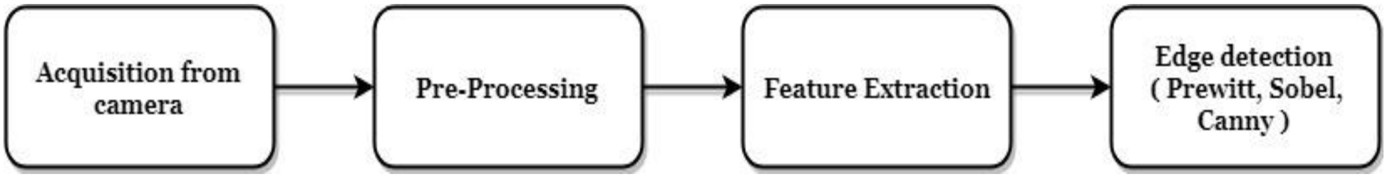
INTRODUCTION

Edge detection is extensively used in image segmentation to divide an image into areas corresponding to different objects. In a picture, an edge is normally defined as an abrupt change in colour intensity. Human's eyes use a much more complicated method to find edges. This is because we have two eyes (therefore stereoscopic vision and depth perception) as well as our incredible inference skills (we can "see" the grey square above, despite it being obscured by the circle). Despite this, most computer vision systems must do with one (normally grayscale) camera, so change in colour intensity is the next best thing. Edges occur in parts of the image with strong intensity contrast, which often represent object boundaries. Edges characterize object boundaries useful for identification of object in a scene such as an X-Ray image. Edges characterize object boundaries useful for identification of object in a scene such as an X-Ray image. Determining bone edges is important because it can provide surgeons with important information for diagnosis, which in turn enables them to give better treatment decision to their patients. Edge detection is extensively used in image segmentation to divide an image into areas corresponding to different objects. Image segmentation is widely used in many areas including.

PROPOSED METHOD

In this project our goal is to detect the boundaries with more edges. Here we proposing different types of algorithms to show that we approached maximum. The main advantage of the system we can go with real time tracking

BLOCK DIAGRAM



Application Description:

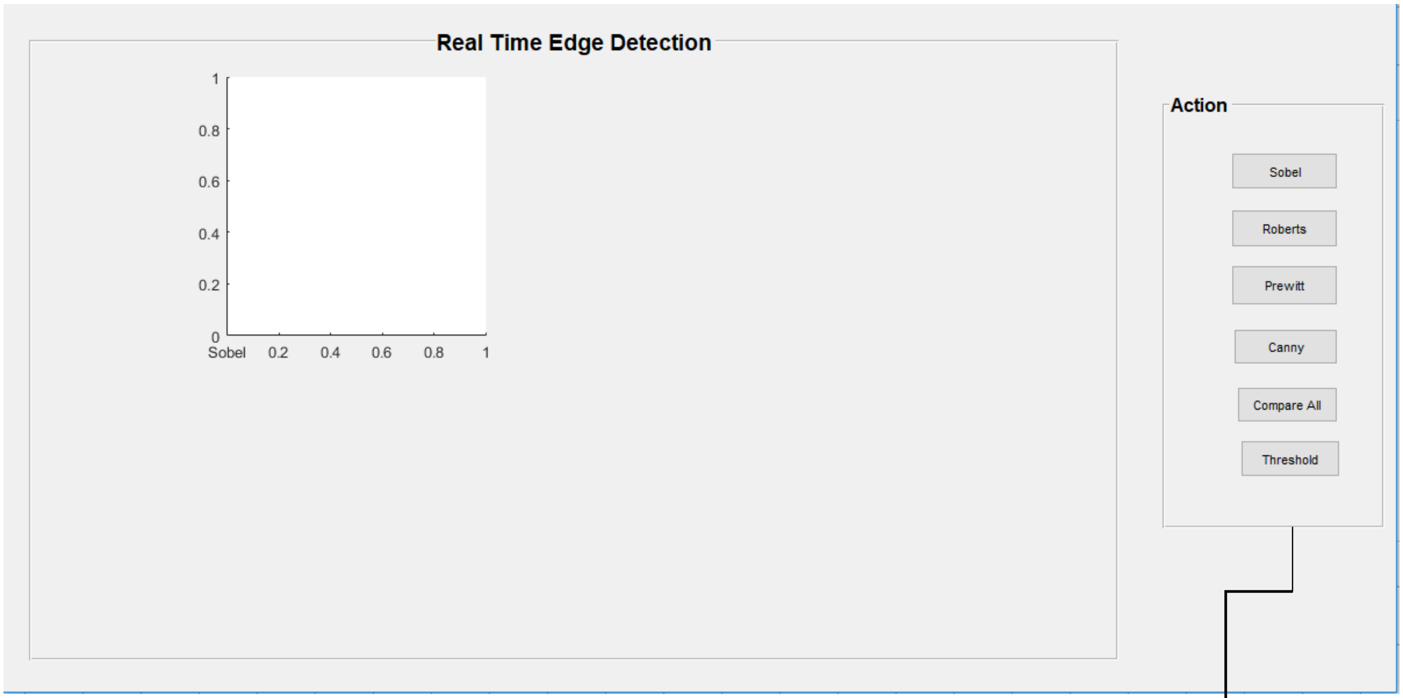
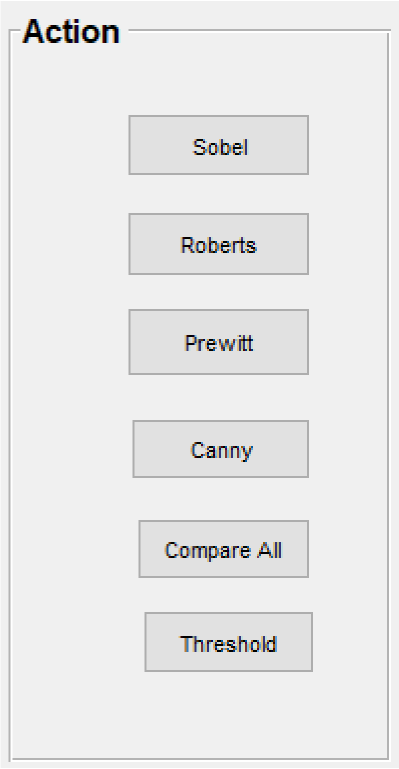


Figure: Real Time Edge Detection Application



Action Panel:

The action panel has four functions:

1. **Sobel:** Detects the edges with Sobel Algorithm
2. **Roberts**: Detects the edges with Roberts Algorithm
3. **Prewitt:** Detects the edges with Prewitt Algorithm
4. **Canny:** Detects the edges with Canny Algorithm
5. **Compare All:** Displays the outputs with all algorithms side by side
6. **Threshold:** Displays the outputs with all algorithms side by side

Figure: Action Panel

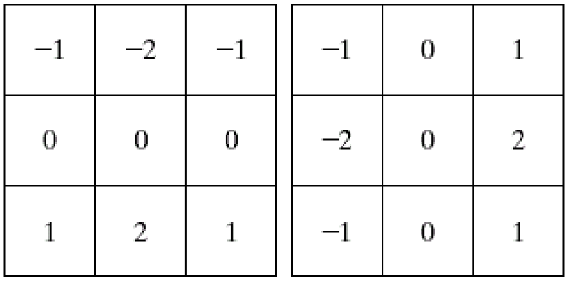
Image Acquisition and Pre-Processing:

Real time video is continuously being captured using the image acquisition tool of Matlab. From that video we are

- collecting snap shots after every 0.1s
- converting them to gray image
- pre-processing them using the imclose( ) function

The closing function is basically *dilation* followed by *erosion*.

Sobel Edge Detection:

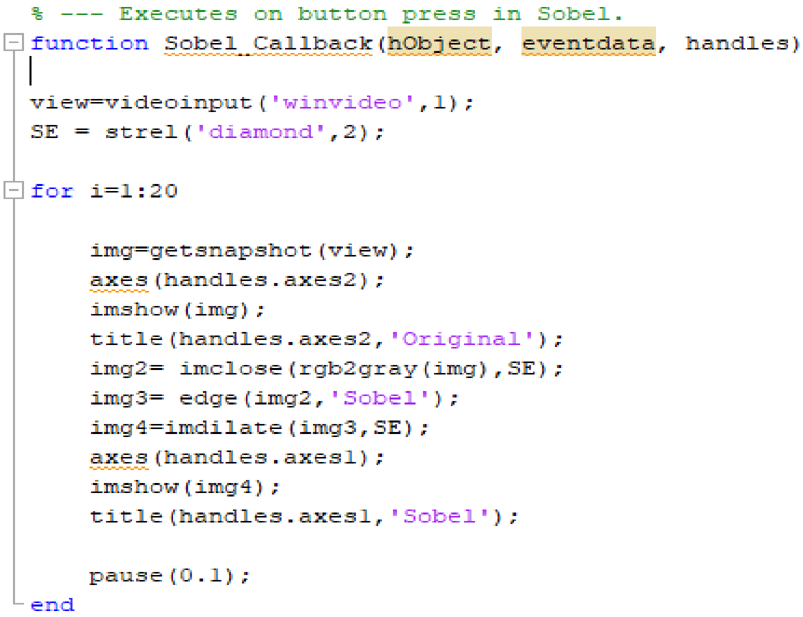


−1	−2	−1
0	0	0
1	2	1

−1	0	1
−2	0	2
−1	0	1

Sobel

Code Snippet:



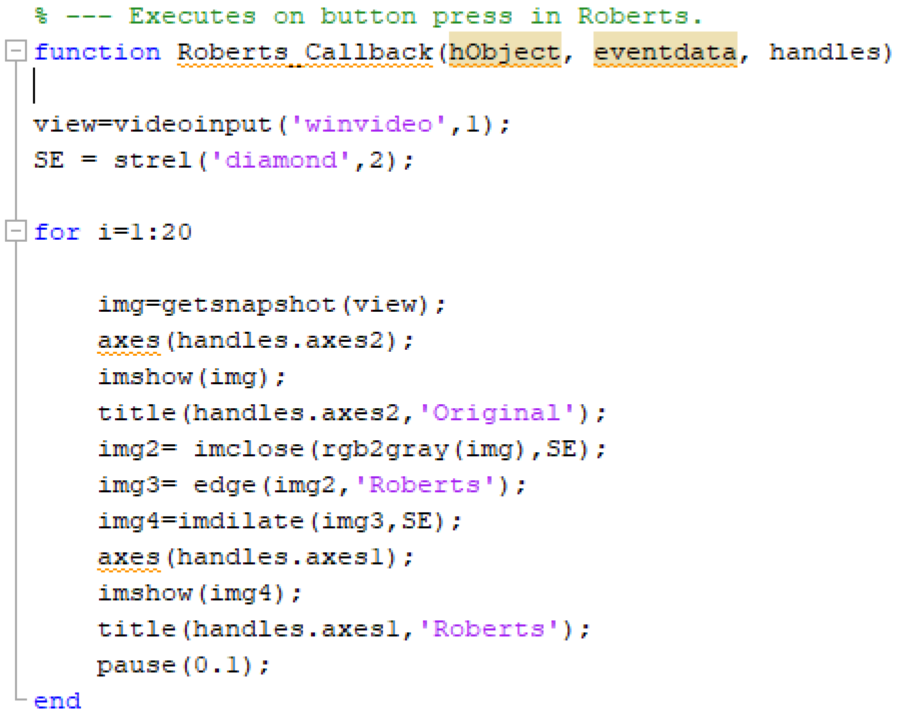
```
% --- Executes on button press in Sobel.
function Sobel_Callback(hObject, eventdata, handles)

view=videoinput('winvideo',1);
SE = strel('diamond',2);

for i=1:20

    img=getsnapshot(view);
    axes(handles.axes2);
    imshow(img);
    title(handles.axes2,'Original');
    img2= imclose(rgb2gray(img),SE);
    img3= edge(img2,'Sobel');
    img4=imdilate(img3,SE);
    axes(handles.axes1);
    imshow(img4);
    title(handles.axes1,'Sobel');

    pause(0.1);
end
```

Roberts Edge Detection:

$$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

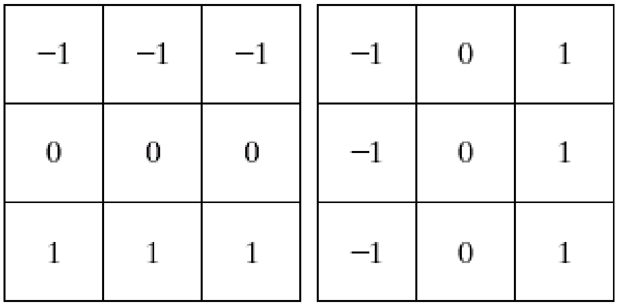
Roberts

Code Snippet:



```
% --- Executes on button press in Roberts.
function Roberts_Callback(hObject, eventdata, handles)

view=videoinput('winvideo',1);
SE = strel('diamond',2);

for i=1:20

    img=getsnapshot(view);
    axes(handles.axes2);
    imshow(img);
    title(handles.axes2,'Original');
    img2= imclose(rgb2gray(img),SE);
    img3= edge(img2,'Roberts');
    img4=imdilate(img3,SE);
    axes(handles.axes1);
    imshow(img4);
    title(handles.axes1,'Roberts');
    pause(0.1);
end
```

Prewitt Edge Detection:

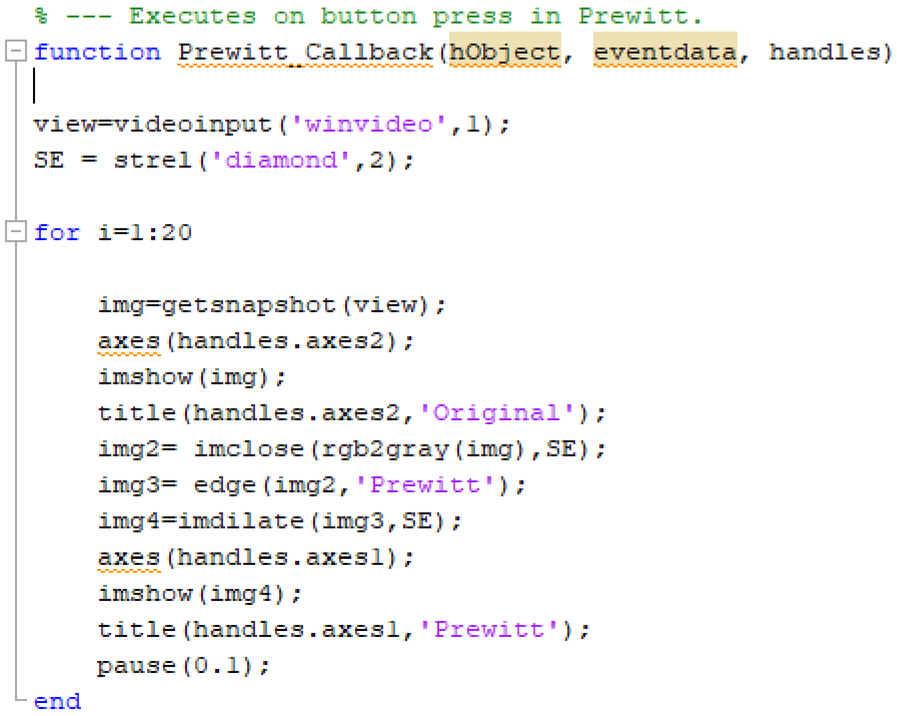


−1	−1	−1
0	0	0
1	1	1

−1	0	1
−1	0	1
−1	0	1

Prewitt

Code Snippet:

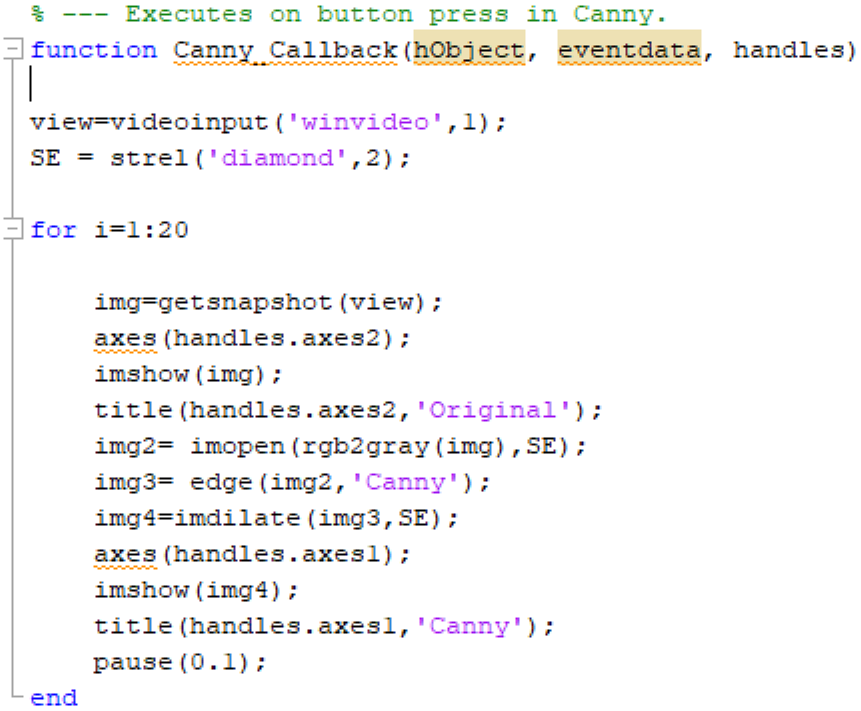


```
% --- Executes on button press in Prewitt.
function Prewitt_Callback(hObject, eventdata, handles)


view=videoinput('winvideo',1);
SE = strel('diamond',2);

for i=1:20

    img=getsnapshot(view);
    axes(handles.axes2);
    imshow(img);
    title(handles.axes2,'Original');
    img2= imclose(rgb2gray(img),SE);
    img3= edge(img2,'Prewitt');
    img4=imdilate(img3,SE);
    axes(handles.axes1);
    imshow(img4);
    title(handles.axes1,'Prewitt');
    pause(0.1);
end
```

Canny Edge Detection:

- Smooth the input image with a Gaussian filter
- Compute the gradient magnitude and angle images
- Apply nonmaxima suppression to the gradient magnitude image
- Use double thresholding and connectivity analysis to detect and link edges

Code Snippet:



```
% --- Executes on button press in Canny.
function Canny_Callback(hObject, eventdata, handles)

view=videoinput('winvideo',1);
SE = strel('diamond',2);

for i=1:20

    img=getsnapshot(view);
    axes(handles.axes2);
    imshow(img);
    title(handles.axes2,'Original');
    img2= imopen(rgb2gray(img),SE);
    img3= edge(img2,'Canny');
    img4=imdilate(img3,SE);
    axes(handles.axes1);
    imshow(img4);
    title(handles.axes1,'Canny');
    pause(0.1);
end
```