Practical File

NIT HAMIRPUR

"COMPUTATIONAL TOOLS AND TECHNIQUES"

CSD-325

NAME : Manish

ROLL : 16583

BRANCH : CSE (4 YEAR)

NAME OF THE FACULTY:

Dr. Kamlesh dutta

(HOD , Department Of CSE)

Ms. Shobhana Kashyap

EXPERIMENT-01

AIM: To Demonstrate OCR ("Character Extractor") in **MATLAB**.

➢ DESCRIPTION:

Recognizing text in images is a common task performed in computer vision applications. For example, you can capture video from a moving vehicle to alert a driver about a road sign. Segmenting out the text from a cluttered scene helps with related tasks such as optical character recognition (OCR).

This mini project shows how to detect regions in an image that contain text. This is a common task performed on unstructured scenes. Unstructured scenes are images that contain undetermined or random scenarios. For example, you can detect and recognize text automatically from captured video to alert a driver about a road sign. This is different than structured scenes, which contain known scenarios where the position of text is known beforehand.

Segmenting text from an unstructured scene greatly helps with additional tasks such as optical character recognition (OCR). The automated text detection algorithm in this example detects a large number of text region candidates and progressively removes those less likely to contain text. This can be used for applications like license plate recognition, OCR, Text to speech converter and other applications.

Firstly, we will read the image. After that we will convert our RGB image into gray scale image for further proceedings. The gray scale images are quite easy to handle in image processing. Now convert this gray scale image to Binary image for removing the background by thresholding. Remove all Boundary connected Objects using **bwareaopen** function. Now we have to label the connected components in image. We use **bwlabel** for this purpose. Now we will measure the properties of the Image regions and Plot the bounding Box. We use **regionprops** and **bounding box** in this segment for this purpose. Now We will segment all the letters in the image to obtain the final extracted character from image.

➢ CODING:

```
%%Image segmentation and extraction

%%Read Image

imagen=imread('Hindi.png');

 imagen=255-imagen;

%%Show image

figure(1)

imshow(imagen);

title('INPUT IMAGE WITH NOISE')

%%Convert to gray scale

if size(imagen,3)==3 % RGB image
```

```
  imagen=rgb2gray(imagen);
end
%%Convert to binary image
threshold = graythresh(imagen);
imagen =~im2bw(imagen,threshold);
%%Remove all object containing fewer than 30 pixels
imagen = bwareaopen(imagen,30);
pause(1)
%%Show image binary image
figure(2)
imshow(~imagen);
title('INPUT IMAGE WITHOUT NOISE')
%%Label connected components
[L Ne]=bwlabel(imagen);
%%Measure properties of image regions
propied=regionprops(L,'BoundingBox');
hold on
%%Plot Bounding Box
for n=1:size(propied,1)
  rectangle('Position',propied(n).BoundingBox,'EdgeColor','g','LineWidth',2)
end
hold off
pause (1)
%%Objects extraction
figure
for n=1:Ne
  [r,c] = find(L==n);
  n1=imagen(min(r):max(r),min(c):max(c));
  imshow(~n1);
  pause(0.5)
end
```

➢ SCREENSHOTS :

❖ **Input image :**

❖ **Region Bounding:**

❖ **Output:**

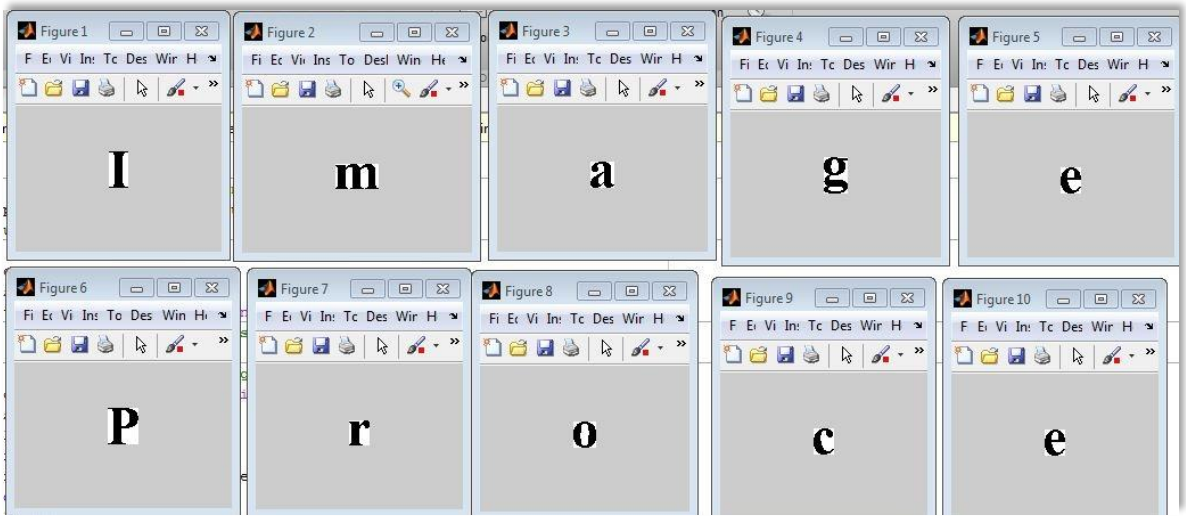


Figure 1
I
Figure 2
m
Figure 3
a
Figure 4
g
Figure 5
e
Figure 6
P
Figure 7
r
Figure 8
o
Figure 9
c
Figure 10
e

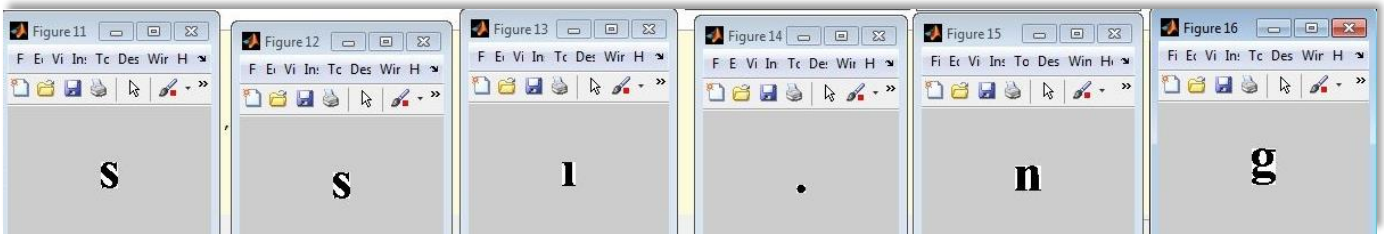


Figure 11
s
Figure 12
s
Figure 13
i
Figure 14
.
Figure 15
n
Figure 16
g

EXPERIMENT-02

AIM: To Demonstrate Image Editor in "**SciLab".**

- DESCRIPTION:

Scilab is a scientific software package for numerical computations providing a powerful open computing environment for engineering and scientific applications. Developed since 1990 by researchers from INRIA (French National Institute for Research in Computer Science and Control) and ENPC (National School of Bridges and Roads), it is now maintained and developed by Scilab Consortium since its creation in May 2003. Distributed freely and open source through the Internet since 1994, Scilab is currently being used in educational and industrial environments around the world. Scilab includes hundreds of mathematical functions with the possibility to add interactively functions from various languages (C, Fortran...). It has sophisticated data structures (including lists, polynomials, rational functions, linear systems...), an interpreter and a high level programming language. Scilab has been designed to be an open system where the user can define new data types and operations on these data types by using overloading.

Some of its features are listed below:

• Basic data type is a matrix, and all matrix operations are available as built-in operations.

• Has a built-in interpreted high-level programming language.

• Graphics such as 2D and 3D graphs can be generated and exported to various formats so that they can be included into documents.

• Scilab also provide feature of GUI through which we can create an interactive applications.

For **GUI** based **Image Editor**, we will use guibuilder. Here we will use different buttons for different image processing tools and axes for displaying the images. We will having following buttons on our **GUI** screen:

- **Capture Image:** This button will allow you to capture the image through webcam.
- **Browse Image:** This button will allow you to select image from your Computer.
- **Negative Image:** This button will convert your image into negative image.
- **Gray Image:** This button will let you to change your image into grayscale.
- **B W Image:** This button will convert your image into binary image.
- **Cartoon Filter 1:** This is a special filter which you can apply on your images.
- **Cartoon Filter 2:** This is a special filter which you can apply on your images.
- **Cartoon Filter 3:** This is a special filter which you can apply on your images.
- **Histogram Image:** This button creates Histogram equalized image.
- **Edge Detection:** This detects edges in your image.
- **Threshold Image:** This gives threshold image of your given Image.
- **Blur Image:** This reduces sharpness of the image.
- **Rotate Image:** This rotates the image by 45^0.
- **Crop image:** This button will provide you a rectangular box through which you can crop your image to your desire.
- **Save Image:** This button will let you to save your image to desired location.

➢ CODING:

```
// This GUI file is generated by guibuilder version 4.2.1

//////////

// Callbacks are defined as below. Please do not delete the comments as it will be used in coming version

//////////


function Capture

global I;

global J;

//Write your callback for  Image  here

n = camopen(0);

im = camread(n); //get a frame

imshow(im);

tic();

for cnt = 1:80

    im = camread(n);

    imshow(im);

end

camclose(1);

I=im;

endfunction

function browse

//Write your callback for  B_image  here

[filename, filepath]=uigetfile("*.*","img");

file_path = filepath+"\"+filename;

global I;

I=imread(file_path);

imshow(I);

endfunction

function negativeimg

//Write your callback for  neg  here

global I;

global J;

im1 = 255 - I;

imshow(im1);
```

```
J=im1;

endfunction

function gray

//Write your callback for  gr  here

global I;

global J;

im2=rgb2gray(I);

imshow(im2);

J=im2;

endfunction

function BW

//Write your callback for  Bw  here

global I;

global J;

im3 = im2bw(I,0.5);

imshow(im3);

J=im3;

endfunction




function red

//Write your callback for  rf  here

global I;

global J;

filter = fspecial('sobel');

imf = filter2(I,filter);

imshow(imf)

J=imf;

endfunction

function green

//Write your callback for  gf  here

global I;

global J;

filter = fspecial('sobel');

imf = imfilter(I, filter);

imshow(imf);

J=imf;
```

```
endfunction

function blue

//Write your callback for  bf  here

global I;

global J;

filter = fspecial('sobel');

imf = filter2(I, filter);

imshow(imf);

J=imf;

endfunction

function histogram

//Write your callback for  Hist  here

global I;

global J;

t=rgb2gray(I);

J1 = imhistequal(t);

imshow(J1);

J=J1;

endfunction




function thresolding

//Write your callback for  thres  here

global I;

global J;

th = imgraythresh(I);

S2 = im2bw(I,th);

figure(); imshow(S2);

J=S2;

endfunction

function enhance

//Write your callback for  enhance  here

global I;

global J;

J1 = imrotate(I,45);

figure(); imshow(J1);

J=J1;
```

```
endfunction

function crop

//Write your callback for  crop  here

global I;

global J;

I2 = imcropm(I);

imshow(I2);

J=I2;

endfunction

function blur

//Write your callback for  blur  here

global I;

global J;

A = imnoise(I,'Gaussian',0.04,0.003);

figure,imshow(A);

I7 = double(A);

sigma = 1.76;   %Standard Deviation

sz = 3; %Box size

[x,y]=meshgrid(-sz:sz,-sz:sz);

M = size(x,1)-1;

N = size(y,1)-1;

%Gaussian

Exp_comp = -(x.^2+y.^2)/(2*sigma*sigma);

Kernel= exp(Exp_comp)/(2*pi*sigma*sigma);

Output=zeros(size(I7));

I7 = padarray(I7,[sz sz]);

 %Convolution

for i = 1:size(I7,1)-M

    for j =1:size(I7,2)-N

        Temp = I7(i:i+M,j:j+M).*Kernel;

        Output(i,j)=sum(Temp(:));

    end

end

Output = uint8(Output);

figure,imshow(Output);

J=Output;

endfunction
```

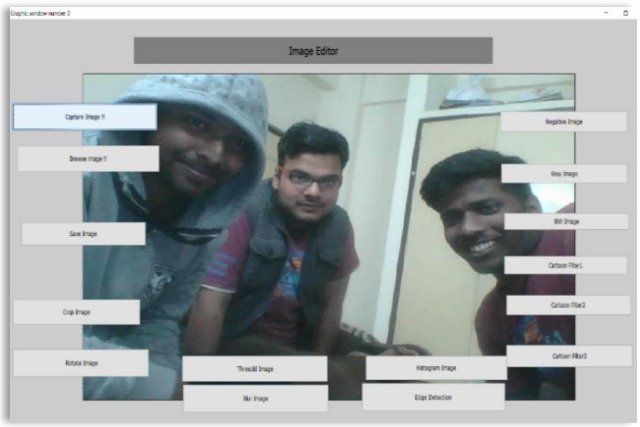
```
function bitslice

//Write your callback for  bit slice  here

global I;

global J;

im = rgb2gray(I);

E = edge(im, 'sobel');

imshow(E);



E = edge(im, 'canny', [0.06, 0.2]);

imshow(E);



E = edge(im, 'prewitt');

imshow(mat2gray(E));

endfunction

function crop

//Write your callback for  crop  here

global I;

global J;

I2 = imcropm(I);

imshow(I2);J=I2;

endfunction

function creators

//Write your callback for  obj15  here

disp("Manish  16583");

disp("Harshit  16570");

disp("Saurav  16555");

disp("Submitted to: Ms Shobhna Kashyap");

endfunction

function save_image

    global J;

    global I;

//Write your callback for  Save  here

imwrite(J,'new.jpg');

endfunction
```

➢ SCREENSHOTS :

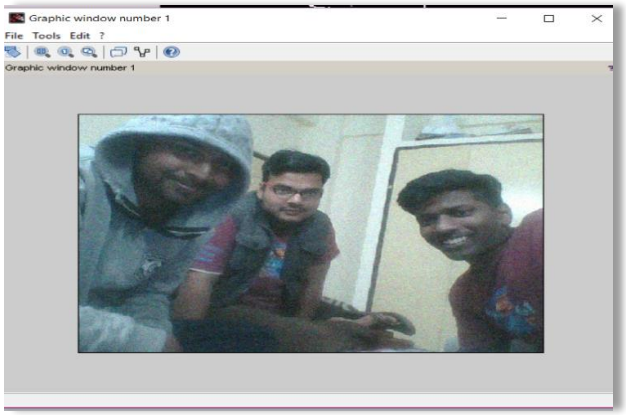
❖ **GUI Screen:**

❖ **Functions of Buttons :**

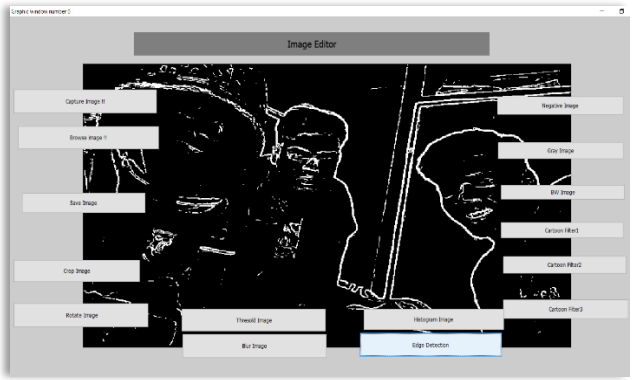
Capture Image



Blur Image



Gray Scale Image



Edge Detection

EXPERIMENT-03

AIM: To Demonstrate a Game in "**PYTHON**".

- DESCRIPTION:

The **pygame** library is an open-source module for the **Python** programming language specifically intended to help you make games and other multimedia applications. Built on top of the highly portable SDL (Simple Direct Media Layer) development library, **pygame** can run across many platforms and operating systems. **Pygame** is usually used in creating games.

Here we are creating a snake game. Here we are generating food by using random function. The size of snake is proportional to the score i.e as the score increases, the size of snake increases. The speed of snake(FPS) also increases as the size increases. The snake is being controlled by the arrow keys. When Snake touces screen of game window, the game stops and your final score appears on the game window.

- CODING:

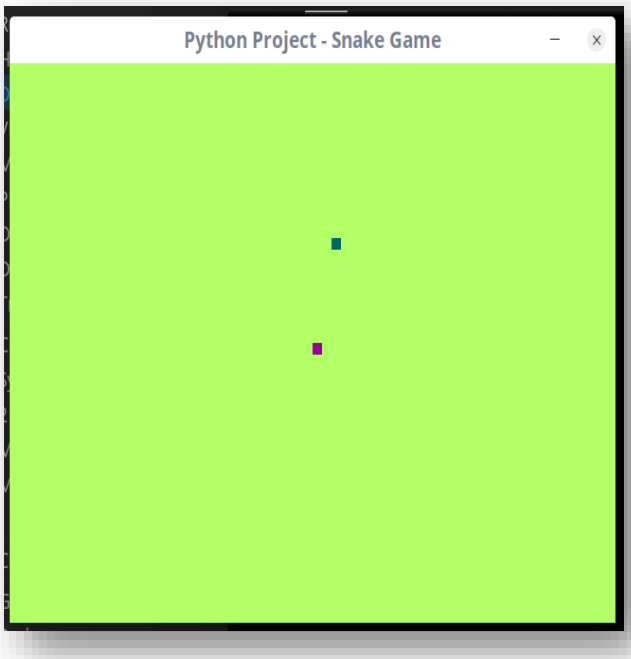
```
import pygame

import time

import random

pygame.init()

green = (178, 255, 102)

black = (153, 0, 153)

blue = (0, 102, 102)

gray = (64, 64, 64)

window_width = 640

window_height = 480

gameDisplay = pygame.display.set_mode((window_width,window_height))

pygame.display.set_caption('Python Project - Snake Game')

clock = pygame.time.Clock()

blockSize = 10

noPixel = 0

font = pygame.font.SysFont(None, 25, bold = True)

def snake(blockSize, snakelist):

    for size in snakelist:

        pygame.draw.rect(gameDisplay, black,[size[0],size[1],blockSize,blockSize])


def message_to_screen(msg, color):

    screen_text = font.render(msg, True, color)

    gameDisplay.blit(screen_text, [50, window_height/2])
```

```
def gameLoop():

    FPS = 15

    score = 0

    gameExit = False

    gameOver = False

    lead_x = window_width/2

    lead_y = window_height/2

    change_pixels_of_x = 0

    change_pixels_of_y = 0

    snakelist = []

    snakeLength = 1

    randomAppleX = round(random.randrange(0, window_width - blockSize) / 10.0) * 10.0

    randomAppleY = round(random.randrange(0, window_height - blockSize) / 10.0) * 10.0

    while not gameExit:

        while gameOver == True:

            gameDisplay.fill(green)

            string = "Game Over | Your Score is " + str(score) + " | Press C to restart | Press Q
to QUIT";

            message_to_screen(string, gray)

            pygame.display.update()

            for event in pygame.event.get():

                if event.type == pygame.QUIT:

                    gameOver = False

                    gameExit = True

                if event.type == pygame.KEYDOWN:

                    if event.key == pygame.K_q:

                        gameExit = True

                        gameOver = False

                    if event.key == pygame.K_c:

                        gameLoop()

        for event in pygame.event.get():

            if event.type == pygame.QUIT:

                gameExit = True

            if event.type == pygame.KEYDOWN:

                leftArrow = event.key == pygame.K_LEFT
```

```
                rightArrow = event.key == pygame.K_RIGHT

                upArrow = event.key == pygame.K_UP

                downArrow = event.key == pygame.K_DOWN



                if leftArrow:

                    change_pixels_of_x = -blockSize

                    change_pixels_of_y = noPixel

                elif rightArrow:

                    change_pixels_of_x = blockSize

                    change_pixels_of_y = noPixel

                elif upArrow:

                    change_pixels_of_y = -blockSize

                    change_pixels_of_x = noPixel

                elif downArrow:

                    change_pixels_of_y = blockSize

                    change_pixels_of_x = noPixel

            if lead_x >= window_width or lead_x < 0 or lead_y >= window_height or lead_y < 0:

                gameOver = True

        lead_x += change_pixels_of_x

        lead_y += change_pixels_of_y

        gameDisplay.fill(green)

        AppleThickness = 10

        pygame.draw.rect(gameDisplay, blue,
[randomAppleX,randomAppleY,AppleThickness,AppleThickness])

        snakehead = []

        snakehead.append(lead_x)

        snakehead.append(lead_y)

        snakelist.append(snakehead)

        if len(snakelist) > snakeLength:

            del snakelist[0]

        for eachSegment in snakelist [:-1]:

            if eachSegment == snakehead:

                gameOver = True

        snake(blockSize, snakelist)
```

```
            pygame.display.update()

            if lead_x >= randomAppleX and lead_x <= randomAppleX + AppleThickness:

                if lead_y >= randomAppleY and lead_y <= randomAppleY + AppleThickness:

                    randomAppleX = round(random.randrange(0, window_width-blockSize)/10.0)*10.0

                    randomAppleY = round(random.randrange(0, window_height-blockSize)/10.0)*10.0

                    snakeLength += 1

                    score = score + 1

                    FPS = FPS + 3

            clock.tick(FPS)

        pygame.quit()

        quit()

gameLoop()
```

- SCREENSHOTS:



EXPERIMENT-04

AIM: Implement Linear Regression in **WEKA** and **R** and Compare the Outputs.

- ### DESCRIPTION:

  "Linear regression" is a linear approach to modelling the relationship between a scalar response (or dependent variable) and one or more explanatory variables (or independent variables).

  In this experiment Linear Regression is calculated between two variables one is dependent variable (y) and another one is independent variable (x) in the form of $y = a * x + b$ where a and b are coefficient to model the relationship between x and y.

  - To implement Linear Regression in WEKA data is provided in the form ARFF file.
  - In R Language Linear Regression is implemented simply by inbuilt function "**lm**" and the data is provided in the form of .csv file.

- ### DATA:

X	1	2	3	4	5	11
Y	2	4	6	8	10	22

- ### CODING :

IN '**R**' LANGUAGE:

```
data<-read.csv('data.csv')

lm(data)

OUTPUT:

R:

Call:

lm(formula = data)


Coefficients:

(Intercept)            y

 -2.176e-15    5.000e-01
```

IN "**WEKA**"

```
=== Run information ===

Scheme:       weka.classifiers.functions.LinearRegression -S 0 -R 1.0E-8 -num-decimal-places 4

Relation:     linearRegression

Instances:    6

Attributes:   2
```

```
                x

                y

Test mode:    6-fold cross-validation


=== Classifier model (full training set) ===

Linear Regression Model

y =

      2      * x +

      0

Time taken to build model: 0.05 seconds


=== Cross-validation ===

=== Summary ===

Correlation coefficient                  1

Mean absolute error                      0

Root mean squared error                  0

Relative absolute error                  0       %

Root relative squared error              0       %

Total Number of Instances                6
```

- SCREENSHOTS :