**JSS MAHAVIDYAPEETHA**
**JSS SCIENCE AND TECHNOLOGY UNIVERSITY**
JSS Technical Institutions Campus, Mysuru – 570006

# "Implementation of Digital Image Processing Techniques"

Mini project report submitted in partial fulfillment of curriculum prescribed for the Digital Image Processing (CS662) course for the award of the degree of

**BACHELOR OF ENGINEERING**
**IN**
**COMPUTER SCIENCE AND ENGINEERING**
*by*

*Akshay S H*
**(01JST17CS014)**

*Bharath Choudhary*
**(01JST17CS033)**

*Under the Guidance of*

**Prof. Shruthi N M**
Assistant Professor,
Dept. Of CS & E
JSS STU Mysore

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**
**2020**

**JSS MAHAVIDYAPEETHA**
**JSS SCIENCE AND TECHNOLOGY UNIVERSITY**
JSS Technical Institutions Campus, Mysuru – 570006

## Abstract

Vision is a major source of information for human beings. Earlier it was impossible to achieve but due to the development of new technologies it has been made possible. Image processing has its impact on communication devices also. By digital image processing we can enhance the image, extract the text from the image, edges of images can be detected and we can apply other effects also . We can get any details about the images . There are many applications of digital image processing. Almost this technique is used in every field , medical field , robotics, neural networking , also useful in the Crime branch for investigation.

The project mainly deals with one of the Various Digital Image Processing techniques like Histogram Equalization, Image Segmentation, Image Smoothing, Image Negative, Image Sharpening, Morphology, Image Gradient, Skeletonization, and Template Matching. We applied these techniques on different types of images.

# Table of Contents

# Chapter 1: Introduction

Digital image processing is concerned with processing of an image. Image processing is a method to perform operations on images like enhancing images, extracting text from image, detecting edge of image and many other operations. In digital image processing we take an image and convert that image in different forms. Like if we take a color image we can convert it into grey image. In this both the input and output is an image.  Usually Image Processing systems include treating images as two dimensional signals while applying already set signal processing methods to them.

Today, it is rapidly growing technology. It forms a core research area within engineering and computer science disciplines too. Image processing has its wide applications in robotics, machine learning, neural networking, signal processing, medical field, graphics and animations and in many other fields.

In our project we have done a detailed analysis of different Image Processing Techniques which were taught in the class and  have come up with the results for each of the respective techniques.

The techniques used are Image Negative, Image Segmentation, Morphology, Image Gradient, Skeletonization etc..

# Chapter 2: Applications

Almost in every field, digital image processing puts a live effect on things and is growing with time to time and with new technologies. These are the few examples:

1) **Image sharpening and restoration-**

   It is the process in which we can modify the image. We can convert the color image to grey image, sharpening, enhancement of the image, detecting edges, and recognition of images.

2) **Medical field-**

   Nowadays if we have brain tumor through image processing the tumor is detected where the tumor is. Also it is used to detect any kind of cancer.
   X Ray imaging, medical CTScan , UV imaging depends on the functioning of digital image processing.

3) **Robot -Vision-**

   There are several robotic machines which work on this technique. Through this technique robots find their ways. Like they can detect the hurdle and line follower robot.

4) **Pattern- recognition-**

   It involves study of image- processing. It is also combined with artificial intelligence such that computer-aided diagnosis, handwriting- recognition and images- recognition can be easily implemented.

5) **Video processing-**

   The collection of frames and pictures are arranged in such a way that movement of pictures becomes faster. It involves frame rate , motion detection, reduction of noise and color space conversion etc.

# Chapter 3: Tools and Technology Used

**Jupyter Notebook :** The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning.

**Sublime Text :** Sublime text is a cross-platform source code editor with Python Programming Interface.It natively supports many Programing Languages and markup languages,functions can be added by users with plugins, typically community-built and maintained under free-software licenses.

**Google Docs :** Google Docs is a platform where we create,edit and manipulate text.It is useful in creating Documents for professional purposes.

**Google Meet :** It is a tool for Real time meetings where we can create and join high quality video meetings which are end to end encrypted and can be used for conference meetings and for presentation purposes.

**Github :** It is a United States-based global company that provides hosting for software development version control using Git. It is a subsidiary of Microsoft, mainly used for version control and building projects etc.

**Python :** Python is an interpreted, high-level, general-purpose programming language.It has various applications in terms of Images Processing, Data Science,Game Design,Web Development etc.,corresponding to its vivid application it provides a huge set of inbuilt libraries to do the job efficiently and with a few lines of code.
The Libraries used are

1. **OpenCV** for operations and manipulation of Images.
2. **Matplotlib** for displaying of Images.

# Chapter 4: Design

Systems design is the process of defining the architecture, modules, algorithms, and data for a system to satisfy specified requirements. System design is the first step once the planning and requirements are clear it just gives the blueprint of the entire implementation in an abstract manner.

Our Project has the following modules under System Design

1) **Histogram Equalisation :-**

   Histogram equalization is a method in image processing of contrast adjustment using the image's histogram. Histogram equalization is the best method for image enhancement. It provides better quality of images without loss of any information.

2) **Image Segmentation** :-

   Image segmentation is the process of partitioning a digital image into multiple segments (sets of pixels, also known as image objects). The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze.

3) **Image Smoothing** :-

   Smoothing is often used to reduce noise within an image or to produce a less pixelated image. Most smoothing methods are based on low pass filters.

4) **Image Negative** :-

   A negative is an image, usually on a strip or sheet of transparent plastic film, in which the lightest areas of the photographed subject appear darkest and the darkest areas appear lightest.

5) **Image Sharpening :-**

Sharpening then, is a technique for increasing the apparent sharpness of an image.

6) **Morphology** :-

Morphological image processing is a collection of non-linear operation related to the shape or morphology of features in an image. Morphological operations rely only on the relative ordering of pixel values, not on their numerical values, and therefore are especially suited to the processing of binary images.

7) **Image Gradient :-**

An image gradient is a directional change in the intensity or color in an image.The gradient of the image is one of the fundamental building blocks in image processing.

8) **Skeletonization** :-

Skeletonization is a process for reducing foreground regions in a binary image to a skeletal remnant that largely preserves the extent and connectivity of the original region while throwing away most of the original foreground pixels.

9) **Template Matching**:-

Template matching is a technique in digital image processing for finding small parts of an image which match a template image. It can be used in manufacturing as a part of quality control, a way to navigate a mobile robot, or as a way to detect edges in images

# Chapter 5: Implementation

## 1) Histogram Equalisation

```
from PIL import Image
import matplotlib.pyplot as plt
import cv2
import numpy as np

img = cv2.imread('hist2.tif',0)
intensity_count = [0] * 256

height,width = img.shape[:2]
N = height * width

high_contrast = np.zeros(img.shape)
for i in range(0,height):
    for j in range(0,width):
        intensity_count[img[i][j]] += 1

L = 256

intensity_count,total_values_used =
np.histogram(img.flatten(),L,[0,L])
pdf_list = np.ceil(intensity_count*(L-1)/img.size)
cdf_list = pdf_list.cumsum()

for y in range(0, height):
    for x in range(0, width):
        #Apply the new intensities in/our new image
        high_contrast[y,x] = cdf_list[img[y,x]]
cv2.imwrite('high_contrast.png', high_contrast)
plt.hist(img.ravel(),256,[0,256])
plt.xlabel('Intensity Values')
plt.ylabel('Pixel Count')
plt.show()
plt.hist(high_contrast.ravel(),256,[0,256])
plt.xlabel('Intensity Values')
plt.ylabel('Pixel Count')
plt.show()

#Displaying the Images
img = cv2.imread('hist2.tif',0)
cv2.imshow('Actual Image',img)
cv2.waitKey(0)
# cv2.destroyAllWindows()

#Display High Contrast Images
img = cv2.imread('high_contrast.png')
cv2.imshow('High Contrast Image',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

## 2) Image Segmentation

```
#Importing the libraries
import cv2
import numpy as np

#Reading the image
img = cv2.imread("image.png",0)

#Apply Gausian blurr with Kernel size 7 to remov noise
blurred_image = cv2.GaussianBlur(img,(7,7),0)

#Apply Otsus thresholding to binarize image
retval ,binarized_image =
cv2.threshold(blurred_image,40,255,cv2.THRESH_BINARY)

# Applying Closing to fill in the holes
filter = np.ones((3,3),np.uint8)
closed_image = cv2.morphologyEx(binarized_image, cv2.MORPH_CLOSE,
filter)
```

```
# Using connected components to label the image
retval, markers = cv2.connectedComponents(closed_image)

# Mapping the component labels to hue val
label_hue = np.uint8(120*markers/np.max(markers))
blank_ch = 255*np.ones_like(label_hue)
labeled_image = cv2.merge([label_hue, blank_ch, blank_ch])

# changing from HSV to RGB again to show
labeled_image = cv2.cvtColor(labeled_image, cv2.COLOR_HSV2BGR)

# background label set to black
labeled_image[label_hue==0] = 0

# getting the unique colors in the image
unique_colors = np.unique(labeled_image.reshape(-1,
labeled_image.shape[2]), axis=0)

print("Colors available in labeled image:")
for x in range(unique_colors.shape[0]):
    print(str(x+1)+"=> B:"+str(unique_colors[x,0])+"
G:"+str(unique_colors[x,1])+"   R:"+str(unique_colors[x,2])+" ")

print ("\nSelect one of the colors and give its RGB values ")

r = input("B : ")
g = input("G : ")
b = input("R : ")

# making an output image
output_image = np.zeros_like(labeled_image)

# getting the object of user input color
for x in range(labeled_image.shape[0]):
    for y in range(labeled_image.shape[1]):
        if (labeled_image[x,y,0] == int(r) and
labeled_image[x,y,1] == int(g) and labeled_image[x,y,2] ==
int(b)):
            output_image[x,y,0:3] = labeled_image[x,y,0:3]

# show the output image
cv2.imshow("Selected", labeled_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

## 3) Image Smoothing

```
#Importing Libraries
import cv2
import numpy as np
from matplotlib import pyplot as plt
from Filter import applyFilter

plt.figure(figsize=(12,12))

#reading image from file
im = cv2.imread("inp1.tif", 0).astype(np.float)
size = int(input("> Enter the size of averaging filter: "))

#applying filter on image
output = cv2.blur(im,(5,5))

#writing image to image file
cv2.imwrite("averaging.jpg",output)

#plotting original image
plt.subplot(211)
plt.axis('off')
plt.title("Original Image")
plt.imshow(im, cmap="gray")

#plotting smoothed image
plt.subplot(212)
plt.axis('off')
plt.title("Smoothed Image (avg.
filter"+str(size)+"x"+str(size)+")")
plt.imshow(output, cmap="gray")
plt.show()

#Popping the window
cv2.imshow("Original",im)
# cv2.waitKey(0)
# cv2.destroyAllWindows()

cv2.imshow("Smooth Image",im)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

## 4) Image Gradient

```
from PIL import Image
import cv2
import numpy as np
get_ipython().run_line_magic('matplotlib', 'inline')
import matplotlib.pyplot as plt

# Reading a image in grayScale
imgGray = cv2.imread("lena.jpg", cv2.IMREAD_GRAYSCALE)
plt.imshow(imgGray, cmap="gray")
print("Original Image")

height, width = imgGray.shape[:2]

# Converting it into an numpy array
grayImg = np.asarray(imgGray)
print(grayImg)

for i in range(0, height):
    for j in range(0, width -1 ):

        #applying gradient
        a = min(grayImg[i][j+1], grayImg[i][j])
        if a == grayImg[i][j+1] :
            temp_arr = grayImg[i][j] - grayImg[i][j+1]
        else :
            temp_arr = grayImg[i][j+1] - grayImg[i][j]


        grayImg[i,j] = temp_arr

img = Image.fromarray(grayImg)
plt.imshow(img, cmap="gray")
print("Output Image")
```

## 5) Image Negative

```
# importing libraries
from PIL import Image
import cv2
import sys
import numpy as np
get_ipython().run_line_magic('matplotlib', 'inline')
import matplotlib.pyplot as plt

# reading image
img = cv2.imread("binary.jpg", cv2.IMREAD_GRAYSCALE)
plt.imshow(img, cmap="binary")
print("Original Image")

# converting to image negative
S = 255
invertedImg = [S - x for x in img]
plt.imshow(invertedImg, cmap="binary")
print("Inverted Image or Image Negative")

# reading image
img = cv2.imread("grayscale.png", cv2.IMREAD_GRAYSCALE)
plt.imshow(img, cmap="gray")
```

```
print("Original Image")

# converting to image negative
S = 255
invertedImg = [S - x for x in img]
plt.imshow(invertedImg, cmap="gray")
print("Inverted Image or Image Negative")

# reading image
img = cv2.imread("rgb.jpg", cv2.IMREAD_COLOR)
img = cv2.cvtColor(img, cv2.COLOR_RGB2BGR)
plt.imshow(img)
print("Original Image")

# converting to image negative
S = 255
invertedImg = [S - x for x in img]
plt.imshow(invertedImg)
print("Inverted Image or Image Negative")
```

## 6) Image Sharpening

```
# Importing Libraries
import cv2
import numpy as np
import matplotlib.pyplot as plt
import copy

# read image
img = cv2.imread("inp1.jpg", cv2.IMREAD_COLOR)
img = cv2.cvtColor(img, cv2.COLOR_RGB2BGR)
print("Original Image")
plt.imshow(img)

#converting color scale from BGR to GRAY
inputImg = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

```
#initialize black image of size equal to given image
outputImg = np.zeros(inputImg.shape)

#padding the image with zeros
inputImg = np.pad(inputImg, (1, 1), 'constant',
constant_values=(0))

#creating two filters for horizontal and vertical edge detection
fh = np.array([[-1.0,-2.0,-1.0],[0.0,0.0,0.0],[1.0,2.0,1.0]])
fy = np.array([[-1.0,0.0,1.0],[-2.0,0.0,2.0],[-1.0,0.0,1.0]])

#looping through image pixels
for row in range(1, inputImg.shape[0]-1):
    for col in range(1, inputImg.shape[1]-1):
        dx, dy = 0.0, 0.0

        #convolving both filters
        for x_filter in range(3):
            for y_filter in range(3):
                dx +=
inputImg[row+x_filter-1][col+y_filter-1]*fh[x_filter][y_filter]
                dy +=
inputImg[row+x_filter-1][col+y_filter-1]*fy[x_filter][y_filter]

        #magnitude of gradient (instead of just adding dx and dy.
we calculate magnitude)
        pixel = np.sqrt(dx * dx + dy * dy)
        outputImg[row-1][col-1] = pixel

#normalizing pixels
outputImg *= 255.0/np.max(outputImg)

print("Sharpened Image")
plt.imshow(outputImg, cmap="gray")
```

## 7) Morphology

```python
#Importing Libraries
import cv2
import numpy as np

img = cv2.imread('signature.png', 0)
r, img = cv2.threshold(img, 130, 255, cv2.THRESH_BINARY_INV)
cv2.imshow("Original", img)
cv2.waitKey(0)
cv2.destroyAllWindows()

#Setting up the kernel
kernel = np.ones((5,5),np.uint8)

dilated = cv2.dilate(img,kernel,iterations = 1)
cv2.imshow("Dilation", dilated)
cv2.imwrite("dilation.png", dilated)
cv2.waitKey(0)
cv2.destroyAllWindows()

eroded = cv2.erode(img,kernel,iterations = 1)
cv2.imshow("Erosion", eroded)
cv2.imwrite("erosion.png", eroded)
cv2.waitKey(0)
cv2.destroyAllWindows()

opened = cv2.morphologyEx(img, cv2.MORPH_OPEN, kernel)
cv2.imshow("Opening", opened)
cv2.imwrite("opening.png", opened)
cv2.waitKey(0)
cv2.destroyAllWindows()

closed = cv2.morphologyEx(img, cv2.MORPH_CLOSE, kernel)
cv2.imshow("Closing", closed)
cv2.imwrite("closing.png", closed)
cv2.waitKey(0)
cv2.destroyAllWindows()

img = cv2.imread('inp.jpg',0)
```

```
ret,bin = cv2.threshold(img,127,255,cv2.THRESH_BINARY_INV)

#Showing the Image
kernel = np.ones((3,3),np.uint8)
opened = cv2.morphologyEx(bin, cv2.MORPH_OPEN, kernel)
cv2.imshow("opened", opened)
cv2.waitKey(0)
cv2.destroyAllWindows()

#performing the operations
kernel = np.ones((5,5),np.uint8)
closed = cv2.morphologyEx(opened, cv2.MORPH_CLOSE, kernel)
ret, output = cv2.threshold(closed,127,255,cv2.THRESH_BINARY_INV)
cv2.imshow("segment", output)
cv2.imwrite("segment.png", output)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

## 8) Skeletonization

```
# importing libraries
import numpy as np
import cv2
get_ipython().run_line_magic('matplotlib', 'inline')
import matplotlib.pyplot as plt

# reading an image
img = cv2.imread('Thumb.png', cv2.IMREAD_GRAYSCALE)
ret, img = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY)

print("Original Image")
plt.imshow(img, cmap="binary")

# Function for skeletonizing the image
def findSkeleton(im):
    element = cv2.getStructuringElement(cv2.MORPH_CROSS,(3,3))
    out = np.zeros(im.shape,np.uint8)

    flag = 0
    while(not flag):
        eroded = cv2.erode(im, element)
        opened = cv2.dilate(eroded, element)
        opened = cv2.subtract(im,opened)
        out = cv2.bitwise_or(out,opened)
        im = eroded.copy()
        zeros = img.size - cv2.countNonZero(im)
        flag = 1 if (zeros == img.size) else 0

    return out


output = findSkeleton(img)

kernel = np.ones((3,3),np.uint8)
output = cv2.dilate(output,kernel)
output = cv2.medianBlur(output, 5)
ret,thresh = cv2.threshold(output,127,255,cv2.THRESH_BINARY_INV)

res = np.hstack((img, thresh))

cv2.imwrite("output.png", res)

# Final Output
print("Skeleton image")
plt.imshow(res, cmap="binary")
```

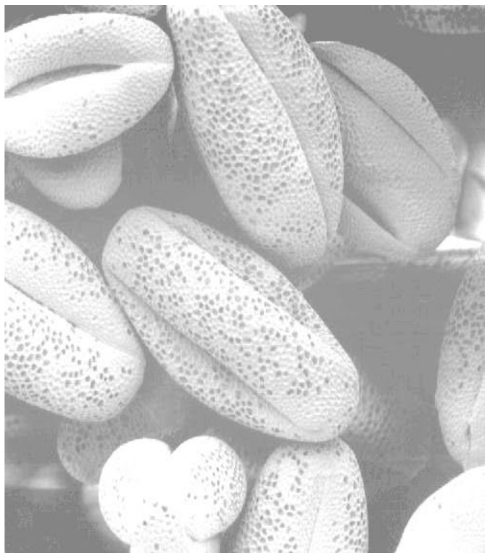
## 9) Template Matching

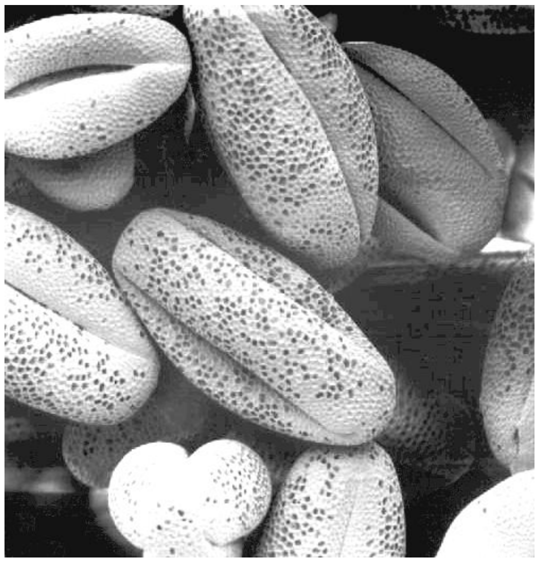
```
# importing libraties
import cv2
import numpy as np
get_ipython().run_line_magic('matplotlib', 'inline')
import matplotlib.pyplot as plt
```

```
# read image and template
image = cv2.imread('image.png')
template = cv2.imread('template.png')
(templateHeight, templateWidth) = template.shape[:2]

img = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
print("Original Image")
plt.imshow(image)

# matching
matchResult = cv2.matchTemplate(image, template, cv2.TM_CCOEFF)
(_, _, minLoc, maxLoc) = cv2.minMaxLoc(matchResult)

topLeft = maxLoc
botRight = (topLeft[0] + templateWidth, topLeft[1] +
templateHeight)
roi = image[topLeft[1]:botRight[1], topLeft[0]:botRight[0]]

mask = np.zeros(image.shape, dtype = "uint8")
image = cv2.addWeighted(image, 0.25, mask, 0.75, 0)

image[topLeft[1]:botRight[1], topLeft[0]:botRight[0]] = roi

cv2.imwrite("matchedTemplate.png", image)

# matched image
img = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
print("Matched Template Image")
plt.imshow(image)
```

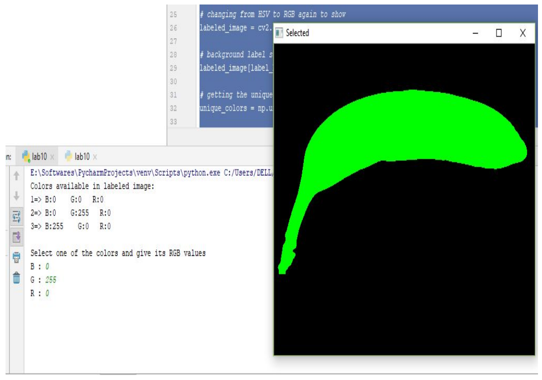
# Chapter 6: Result Analysis

## 1) Histogram Equalisation



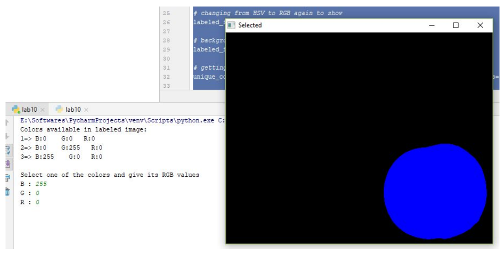


## 2) Image Segmentation



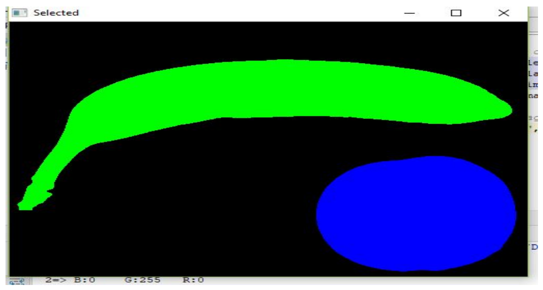
```
# changing from HSV to RGB again to show
labeled_image = cv2.

# background label s
labeled_image[label_

# getting the unique
unique_colors = np.u
```

```
E:\Softwares\PycharmProjects\venv\Scripts\python.exe C:/Users/DELL/
Colors available in labeled image:
1=> B:0    G:0   R:0
2=> B:0    G:255   R:0
3=> B:255    G:0   R:0

Select one of the colors and give its RGB values
B : 0
G : 255
R : 0
```

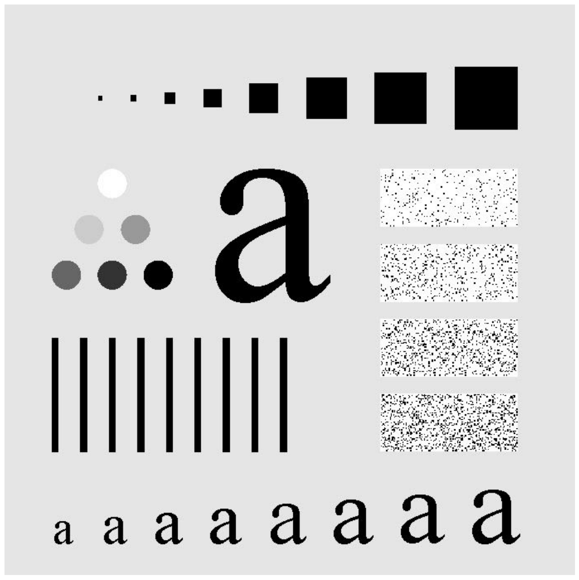
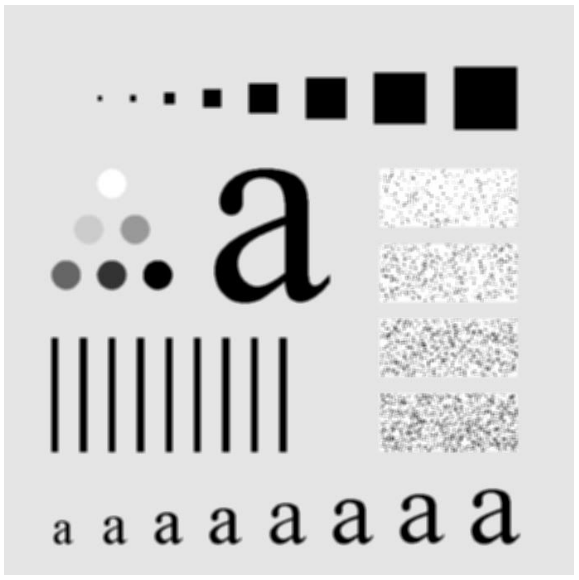


```
# changing from HSV to RGB again to show
labeled_
# backgr
labeled_
# getting
unique_c
```

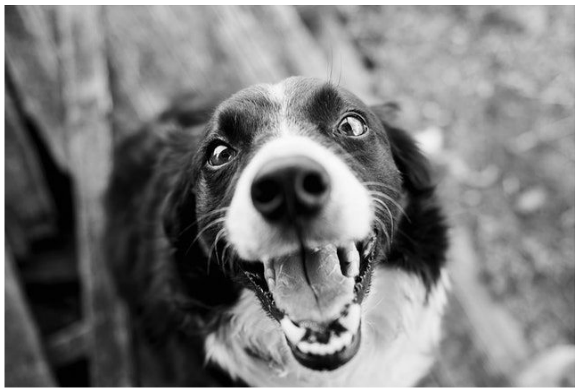
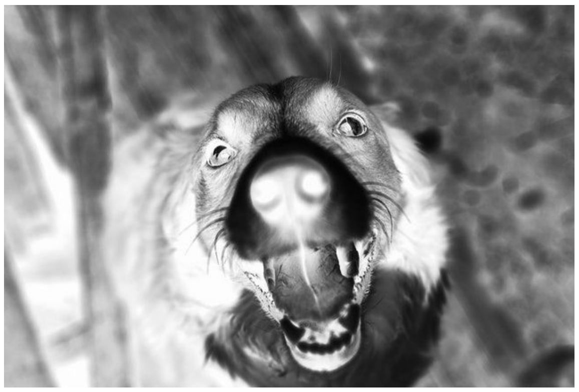
```
E:\Softwares\PycharmProjects\venv\Scripts\python.exe C:
Colors available in labeled image:
1=> B:0    G:0   R:0
2=> B:0    G:255   R:0
3=> B:255    G:0   R:0

Select one of the colors and give its RGB values
B : 255
G : 0
R : 0
```



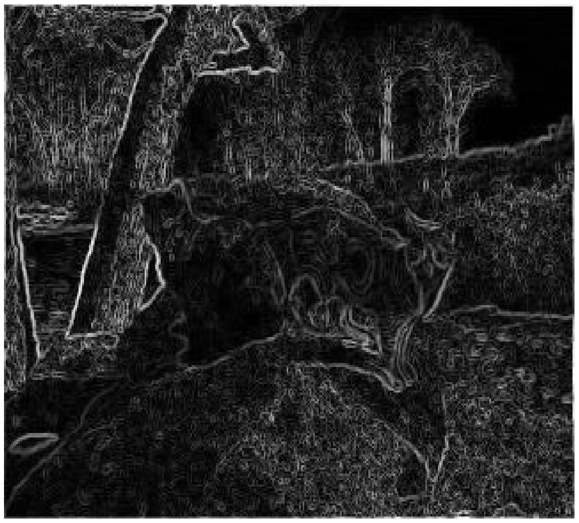
## 3) Image Smoothing

## 4) Image Negative

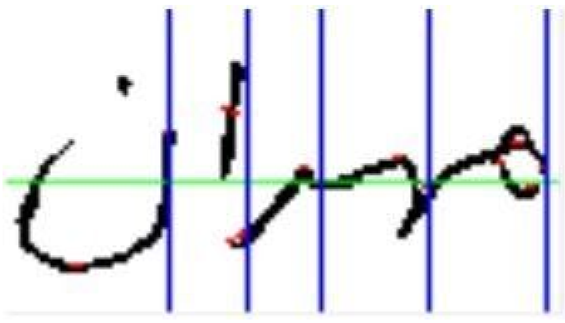
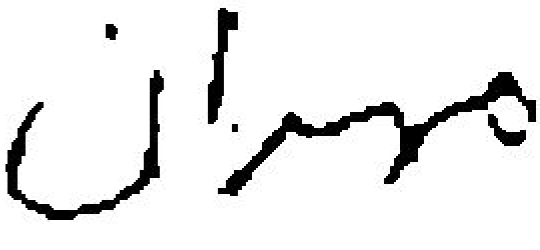



## 5) Image Sharpening

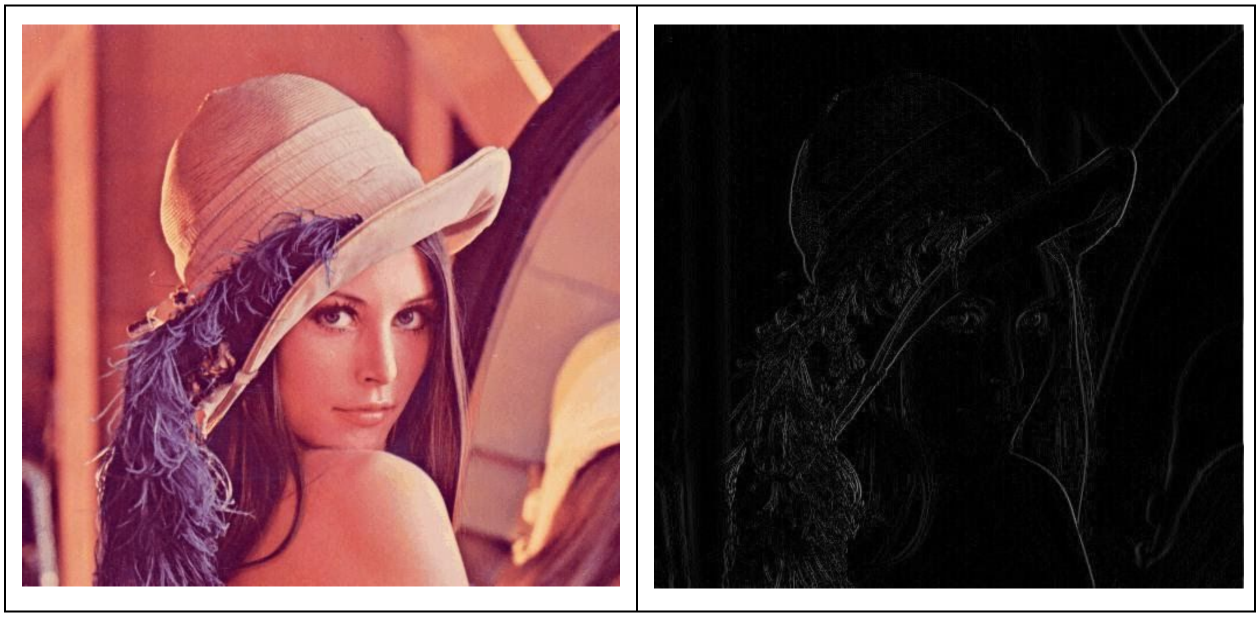


## 6) Morphology

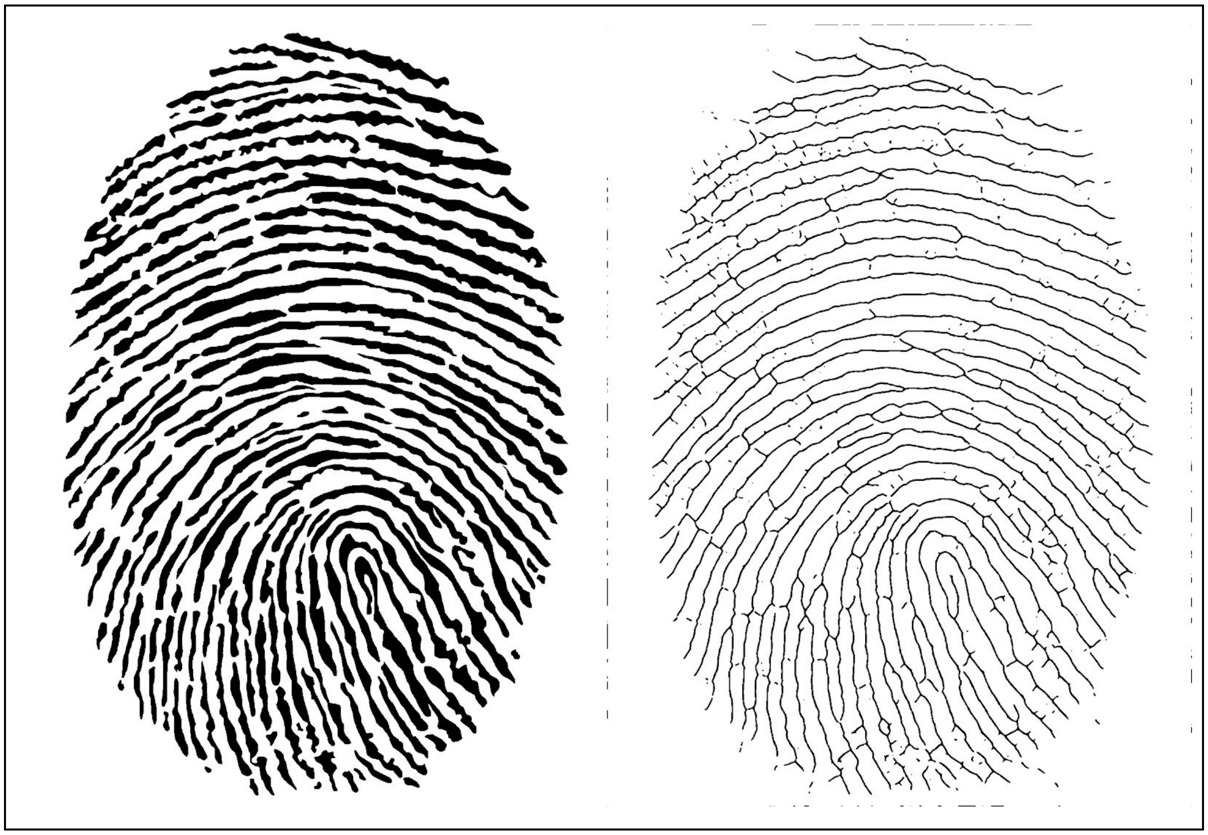
| | |
|---|---|
| | |
| | |

| | |
|---|---|
|  |  |

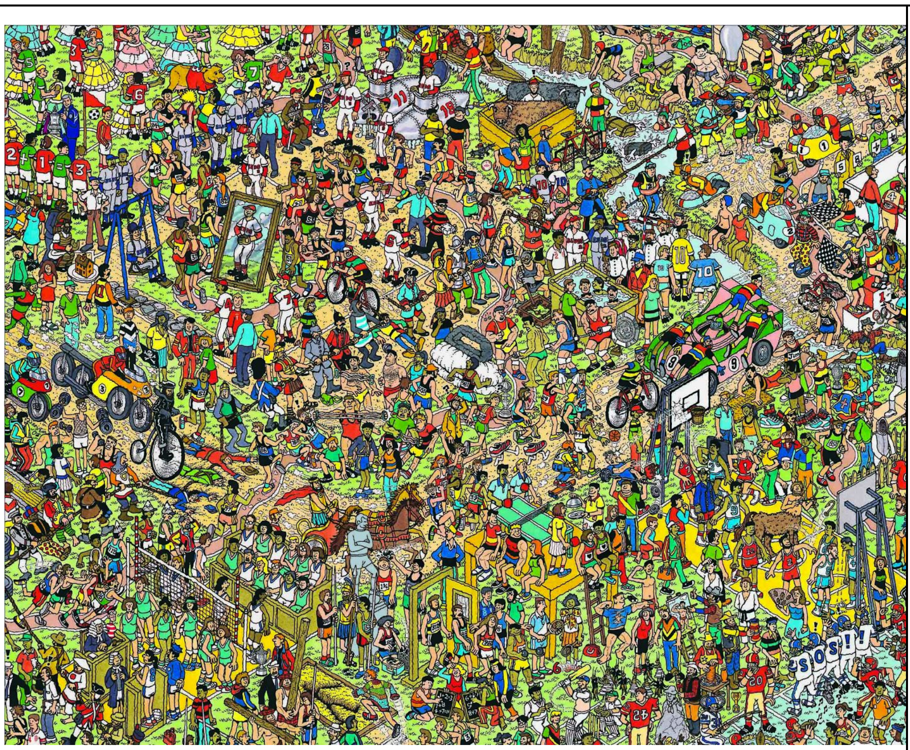
**7) Image Gradient**

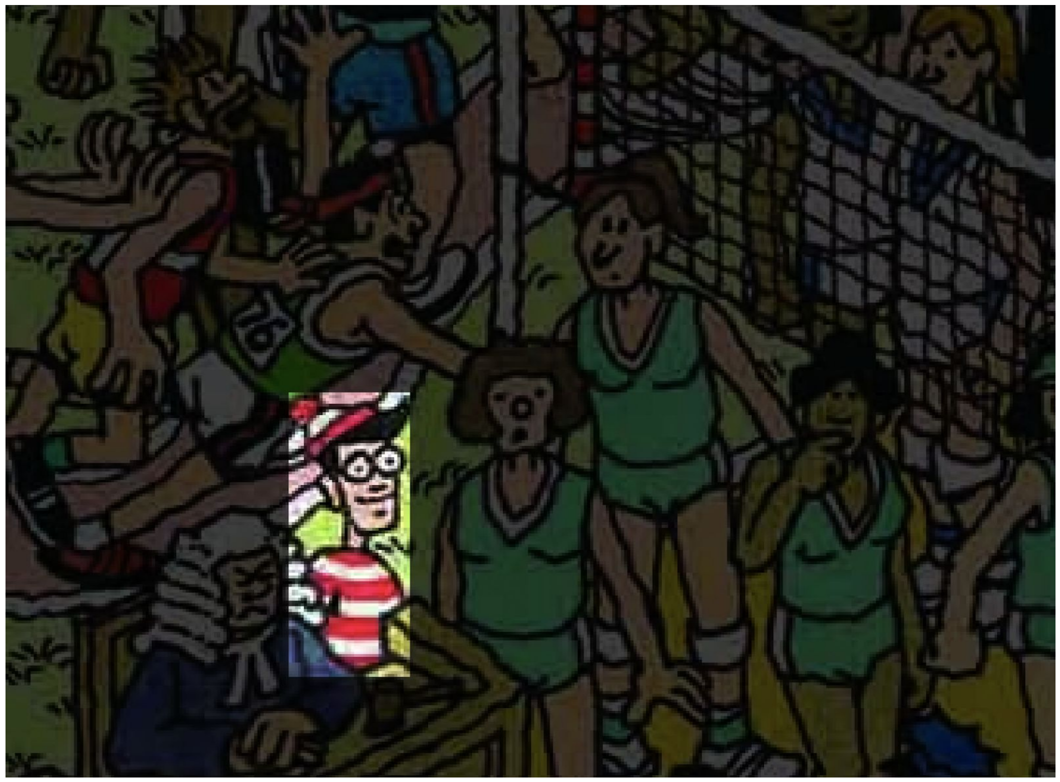


**8) Skeletonization**



## 9) Template Matching





# Chapter 7 : Conclusion

Due to the fast development in computer technology , the future of image-processing is going to be more flexible. Digital image processing has a wide range of applications. In future it has been expected to be less expensive. Due to advancement of this technology there will be millions of robots in the world. Advances in the image- processing and artificial intelligence will involve spoken commands, translation of languages, recognizing the finger prints, tracking of people and things, diagnosing medical conditions, performing surgery and automatically driving all forms of transport. With an increasing power of modern computing, the concept of computation can go beyond the present limits and in future, image processing technology will advance . The future trend in remote sensing will be towards improved sensors that record the same scene in many spectral channels. Graphics data is becoming important nowadays in image processing applications. In future image processing techniques will play an important role in space also.

Learned about the different techniques of Image processing and its real world applications.

## References

### Websites

1. https://github.com/coder-KB/imageProcessing
2. https://docs.opencv.org/2.4/doc/tutorials/tutorials.html

### Books

1. Rafael C Gonzalez and Richard E Woods, Digital Image Processing, Pearson Education, 4th edition , 2017