DIGITAL IMAGE PROCESSING LABORATORY

(V semester of B.Tech)

As per the curriculam and syllabus

Of

Bharath Institute of Higher Education & Research

PREPARED BY
DR.A.KUMARAVEL

NEW EDITION

Department of information technology

Bharath
INSTITUTE OF HIGHER EDUCATION AND RESEARCH
(Declared as Deemed - to - be - University under section 3 of UGC Act 1956)
ACCREDITED WITH 'A' GRADE BY NAAC

DIGITAL IMAGE PROCESSING

(V semester of B.Tech)

As per the curricullam and syllabus

Of

Bharath Institute of Higher Education & Research

PREPARED BY
DR.A.KUMARAVEL

Department of information technology

Bharath
INSTITUTE OF HIGHER EDUCATION AND RESEARCH
(Declared as Deemed - to - be - University under section 3 of UGC Act 1956)
ACCREDITED WITH 'A' GRADE BY NAAC

SCHOOL OF COMPUTING

DEPARTMENT OF INFORMATION TECHNOLOGY

LAB MANUAL

SUBJECT NAME: DIGITAL IMAGE PROCESSING LABORATORY

SUBJECT CODE: U20ITCJ06

Regulation - 2020

VISION AND MISSION OF THE INSTITUTE

VISION

"Bharath Institute of Higher Education & Research (BIHER) envisions and constantly strives to provide an excellent academic and research ambience for students and members of the faculties to inherit professional competence along with human dignity and transformation of community to keep pace with the global challenges so as to achieve holistic development."

MISSION

- To develop as a Premier University for Teaching, Learning, Research and Innovation on par with leading global universities.
- To impart education and training to students for creating a better society with ethics and morals.
- To foster an interdisciplinary approach in education, research and innovation by supporting lifelong professional development, enriching knowledge banks through scientific research, promoting best practices and innovation, industry driven and institute-oriented cooperation, globalization and international initiatives.
- To develop as a multi-dimensional institution contributing immensely to the cause of societal advancement through spread of literacy, an ambience that provides the best of international exposures, provide health care, enrich rural development and most importantly impart value-based education.
- To establish benchmark standards in professional practice in the fields of innovative and emerging areas in engineering, management, medicine, dentistry, nursing, physiotherapy and allied sciences.
- To imbibe human dignity and values through personality development and social service activities.

VISION AND MISSION OF THE DEPARTMENT

VISION

To be an excellence in education and research in Information Technology producing global scholars for improvement of the society

MISSION

- To provide sound fundamentals, and advances in Information Technology, Software Engineering, data Communications and Computer Applications by offering world class curriculum.
- To create ethically strong leaders and expert for next generation IT.
- To nurture the desire among faculty and students from across the globe to perform outstanding and impactful research for the benefit of humanity and, to achieve meritorious and significant growth.

PROGRAM EDUCATIONAL OBJECTIVES (PEO)

The Program Educational Objectives (PEOs) of Information technology are listed below: The graduate after 3-5 years of programme completion will

PEO1: PREPARATION
To provide students with sound fundamental in Mathematical, Scientific and Engineering fundamentals necessary to formulate, analyse, and comprehend the fundamental concepts essential to articulate, solve and assess engineering problems and to prepare them for research & development and higher learning.

PEO2: CORE COMPETENCE
To apply critical reasoning, quantitative, qualitative, designing and programming skills, to identify, solve problems and to analyze the experimental evaluations, and finally making appropriate decisions along with knowledge of computing principles and applications and be able to integrate this knowledge in a variety of industry and inter-disciplinary setting.

PEO3: PROFESSIONALISM
To broaden knowledge to establish themselves as creative practicing professionals, locally and globally, in fields such as design, development, problem solving to production support in software industries and R&D sectors.

PEO4: SKILL
To provide better opportunity to become a future researchers / scientist with good communication skills so that they may be both good team-members and leaders with innovative ideas for a sustainable development.

PEO5: ETHICS
To be ethically and socially responsible solution providers and entrepreneurs in Computer Science and other engineering discipline.

PROGRAMME OUTCOMES

PO 1	**ngineering Knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO2	**oblem Analysis:** Identify, formulate, review research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences.
PO 3	**sign/Development of Solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO 4	**nduct Investigations of Complex Problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions for complex problems.
PO 5	**odern Tool Usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
PO 6	**e Engineer and Society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO 7	**vironment and Sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO 8	**hics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO 9	**dividual and Team Work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO 10	**mmunication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO 11	**oject Management and Finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
PO 12	**fe-long Learning:** Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change.

PROGRAMME SPECIFIC OUTCOME

PSO 1	**Programming Design :** Design and develop algorithm for real life problems using latest technologies and solve it by using computer programming languages and database technologies .
PSO 2	**IT Business Scalable Design :** Analyze and recommend computing infrastructures and operations requirements and Simulate and implement information networks using configurations, algorithms, suitable protocol and security for valid and optimal connectivity.
PSO 3	**Intelligent Agents Design :** Design and execute projects for the development of data modeling, data analytics and knowledge representation in various domain.

U20ITCJ06- DIGITAL IMAGE PROCESSING

PART A- INTRODUCTION OF THE COURSE

Course Code	U20ITCJ06	**L**	**T**	**P**	**C**
		3	**0**	**2**	**4**
Course Title	**Digital Image Processing**				
Course Category	Professional Core (C)	**Contact Hrs**	**75**		
Pre-requisite		**Co-Requisite**	Nil		
Name of the Course Coordinator		**DR.A.KUMARAVEL**			
Course offering Dept./School		**IT / SoC**			

Course Objective and Summary

-

Course Outcomes (COs)

CO1	
CO2	
CO3	
CO4	
CO5	

Mapping / Alignment of COs with PO & PSO

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	2	1	-	-	2	2	3	2	2	2	2	2	2	3	2
CO2	2	1	-	-	2	2	3	2	2	2	2	2	2	3	2
CO3	3	2	1	1	3	3	3	3	3	3	3	3	3	3	3
CO4	3	2	1	1	3	3	3	3	3	3	3	3	3	3	3
CO5	3	2	1	1	3	3	3	3	3	3	3	3	3	3	3

(Tick mark or level of correlation: 3-High, 2-Medium, 1-Low)

PART B

CONTENT OF THE COURSE

S.No	Summary of Course Content	Hr(s)	Alignment to COs
1	Steps in Digital Image Processing	1	CO1
2	Components – Elements of Visual Perception	1	CO1
3	Image Sensing and Acquisition	1	CO1
4	Image Sampling and Quantization	1	CO1
5	Relationships between pixels	1	CO1
6	Color image fundamentals - RGB	1	CO1
7	HSI models	1	CO1
8	Two-dimensional mathematical preliminaries	1	CO1
9	2D transforms - DFT, DCT	1	CO1
10	Spatial Domain: Gray level transformations	1	CO2
11	Histogram processing	1	CO2
12	Basics of Spatial Filtering	1	CO2
13	Smoothing and Sharpening Spatial Filtering,	1	CO2
14	Frequency Domain: Introduction to Fourier Transform	1	CO2
15	Smoothing and Sharpening frequency domain filters	1	CO2
16	Ideal, Butterworth and Gaussian filters	1	CO3
17	Homomorphic filtering	1	CO3
18	Color image enhancement	1	CO3
19	Image Restoration	1	CO3
20	degradation model, Properties, Noise models	1	CO3
21	Mean Filters – Order Statistics	1	CO3
22	Adaptive filters	1	CO3
23	Band reject Filters	1	CO3
24	Band pass Filters	1	CO3
25	Notch Filters	1	CO3
26	Optimum Notch Filtering	1	CO3
27	Inverse Filtering – Wiener filtering	1	CO4
28	Edge detection, Edge linking via Hough transform – Thresholding	1	CO4
29	Region based segmentation	1	CO4
30	Region growing	1	CO4
31	Region splitting and merging	1	CO4
32	Morphological processing	1	CO4
33	erosion and dilation,	1	CO4
34	Segmentation by morphological watersheds	1	CO4
35	basic concepts of watersheds	1	CO4
36	Dam construction Watershed segmentation algorithm.	1	CO5
37	Need for data compression, Huffman, Run Length Encoding, Shift codes, Arithmetic coding,	1	CO5
38	JPEG standard, MPEG.	1	CO5
39	Boundary representation, Boundary description,	1	CO5
40	Fourier Descriptor	1	CO5
41	Regional Descriptors	1	CO5
42	Topological feature	1	CO5

43	Texture	1	CO5
44	Patterns and Pattern classes	1	CO5
45	Recognition based on matching	1	CO5

UNIT I DIGITAL IMAGE FUNDAMENTALS

Steps in Digital Image Processing – Components – Elements of Visual Perception – Image Sensing and Acquisition – Image Sampling and Quantization – Relationships between pixels - Color image fundamentals - RGB, HSI models, Two-dimensional mathematical preliminaries, 2D transforms - DFT, DCT.

UNIT II IMAGE ENHANCEMENT

Spatial Domain: Gray level transformations – Histogram processing – Basics of Spatial Filtering– Smoothing and Sharpening Spatial Filtering, Frequency Domain: Introduction to Fourier Transform– Smoothing and Sharpening frequency domain filters – Ideal, Butterworth and Gaussian filters, Homomorphic filtering, Color image enhancement.

UNIT III IMAGE RESTORATION

Image Restoration - degradation model, Properties, Noise models – Mean Filters – Order Statistics – Adaptive filters – Band reject Filters – Band pass Filters – Notch Filters – Optimum Notch Filtering – Inverse Filtering – Wiener filtering

UNIT IV IMAGE SEGMENTATION

Edge detection, Edge linking via Hough transform – Thresholding - Region based segmentation – Region growing – Region splitting and merging – Morphological processing- erosion and dilation, Segmentation by morphological watersheds – basic concepts – Dam construction – Watershed segmentation algorithm.

UNIT V IMAGE COMPRESSION AND RECOGNITION

Need for data compression, Huffman, Run Length Encoding, Shift codes, Arithmetic coding, JPEG standard, MPEG. Boundary representation, Boundary description, Fourier Descriptor, Regional Descriptors – Topological feature, Texture - Patterns and Pattern classes - Recognition based on matching.

INDEX

S. No.	Date	Name of the Experiment	Page No.	Signature
1		Basic Image Operations using OpenCV	1 - 2	
2		Image Enhancement Techniques such as Histogram Equalization and Contrast Stretching	3 - 5	
3		Image Filtering and Convolution	6 - 7	
4		Frequency Domain Analysis and Filtering using the Fourier Transform	8 - 9	
5		Image Compression Techniques using Run-Length Encoding (RLE) and Discrete Cosine Transform (DCT)	10 - 11	
6		Morphological Image Processing Techniques	12 - 13	
7		Image Segmentation Techniques using OpenCV	14 - 15	
8		Finding Contours and Calculating Region-Based Properties	16 - 17	
9		Colour Balance Correction using White Balance Algorithms	18 - 19	
10		Edge Detection using the Canny Edge Detector	20 - 21	

Exp - 1	Basic Image Operations using OpenCV
Date -	

AIM: To write a program for which covers basic image operations using OpenCV in Python.

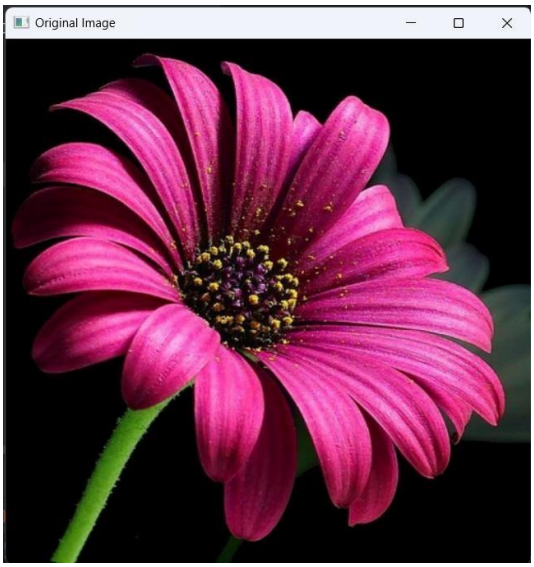
ALGORITHM:

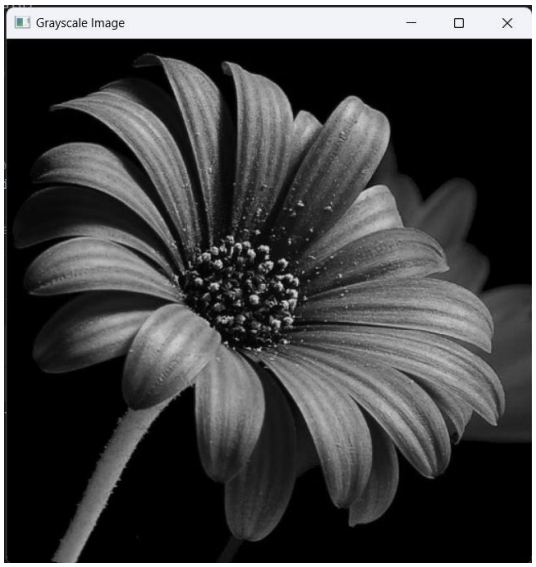
1. Import Libraries.
2. Load the Image.
3. Use OpenCV's cv2.imread() function to load the image from the specified file path
4. Display the Original Image Using OpenCV.
5. Use cv2.imshow() to display the original image.
6. Convert the Image to Grayscale.
7. Use cv2.cvtColor() to convert the loaded color image to grayscale (BGR to grayscale).
8. Display the Grayscale Image Using OpenCV.
9. Use cv2.imshow() to display the grayscale image.
10. Save the Grayscale Image.
11. Display Both Images Using Matplotlib.
12. Use plt.imshow() to display the original image (converted from BGR to RGB for proper visualization).
13. Use plt.imshow() to display the grayscale image using a grayscale colormap.
14. Use plt.tight_layout() to adjust the spacing between subplots.
15. Use plt.show() to display the Matplotlib plot containing both images.

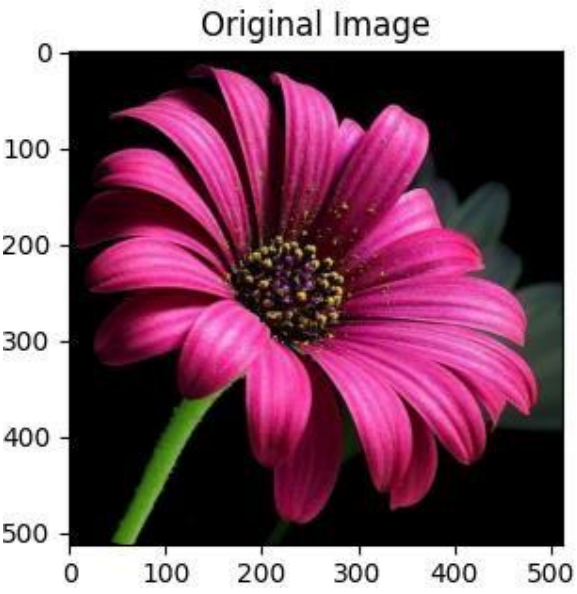
PROGRAM:

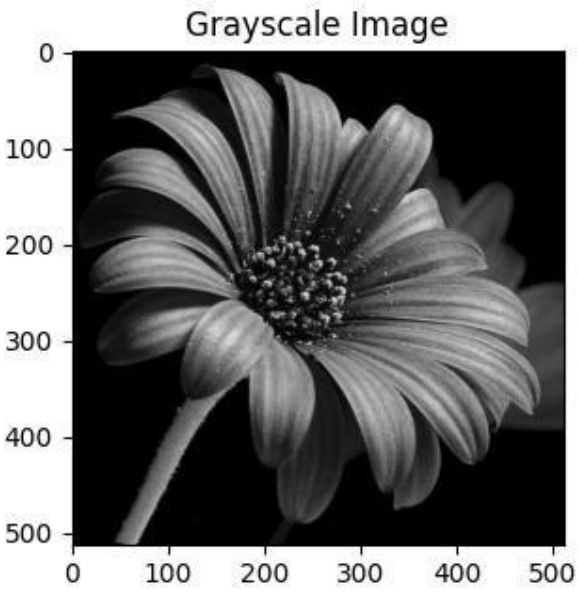
```
import cv2
import matplotlib.pyplot as plt

image_path = "1.png"
image = cv2.imread(image_path)

cv2.imshow("Original Image", image)
cv2.waitKey(0)
cv2.destroyAllWindows()

gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

cv2.imshow("Grayscale Image", gray_image)
cv2.waitKey(0)
cv2.destroyAllWindows()

output_path = "grayscale_image.jpg"
cv2.imwrite(output_path, gray_image)

plt.subplot(1, 2, 1)
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
plt.title("Original Image")

plt.subplot(1, 2, 2)
plt.imshow(gray_image, cmap="gray")
plt.title("Grayscale Image")
plt.tight_layout()
plt.show()
```

OUTPUT:









RESULT: Hence, the program that covers basic image operations using OpenCV in Python is executed successfully.

Exp - 2	Image Enhancement Techniques such as Histogram Equalization and Contrast Stretching
Date -	

AIM: To implement a program for which covers image enhancement techniques such as histogram equalization and contrast stretching using OpenCV in Python.

ALGORITHM:

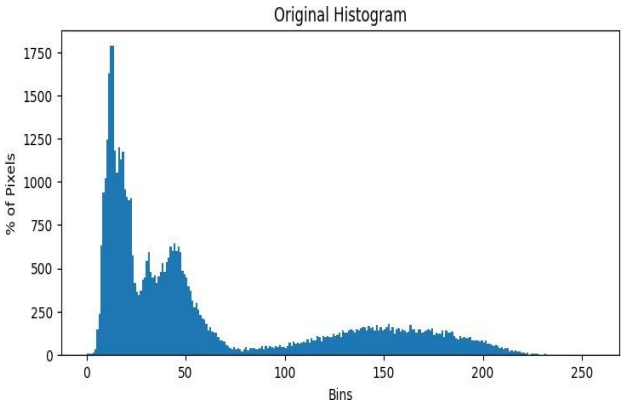
1. Import Libraries.
2. Load the Image.
3. Use OpenCV's cv2.imread() function to load the image in grayscale mode (cv2.IMREAD_GRAYSCALE).
4. Perform Histogram Equalization.
5. Use cv2.equalizeHist() to perform histogram equalization on the loaded grayscale image.
6. Perform Contrast Stretching.
7. Display Original Image and Histogram.
8. Use plt.imshow() to display the original image in the first subplot.
9. Display Equalized Image and Histogram.
10. Use plt.imshow() to display the equalized image in the first subplot.
11. Display Stretched Image and Histogram.
12. Use plt.imshow() to display the stretched image in the first subplot.
13. Show Plots: plt.tight_layout(), plt.show().

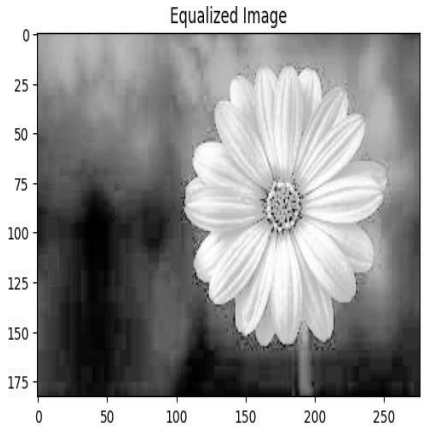
PROGRAM:

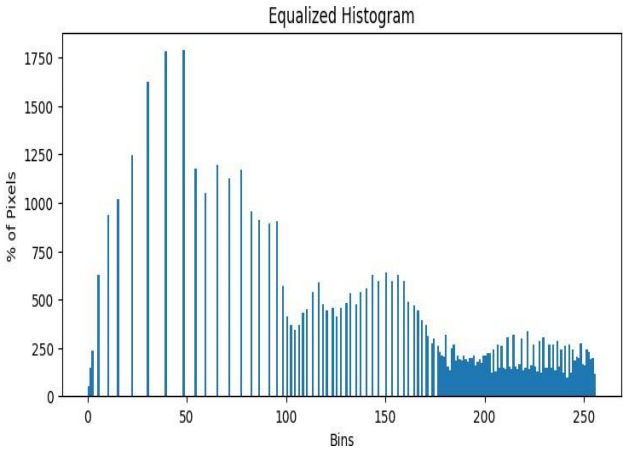
```
import cv2
import numpy as np
import matplotlib.pyplot as plt

image_path = "2.png"
image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
equalized_image = cv2.equalizeHist(image)
min_intensity = np.min(image)
max_intensity = np.max(image)
a = 0
b = 255
stretched_image = np.clip((image - min_intensity) * (b - a) /
(max_intensity - min_intensity) + a, 0, 255).astype(np.uint8)

plt.figure(figsize=(10, 8))
plt.subplot(2, 2, 1)
plt.imshow(image, cmap='gray')
plt.title("Original Image")

plt.subplot(2, 2, 2)
plt.hist(image.ravel(), 256, [0, 256])
plt.ylabel('% of Pixels')
plt.xlabel('Bins')
plt.title("Original Histogram")

plt.subplot(2, 2, 3)
plt.imshow(equalized_image, cmap='gray')
plt.title("Equalized Image")
```

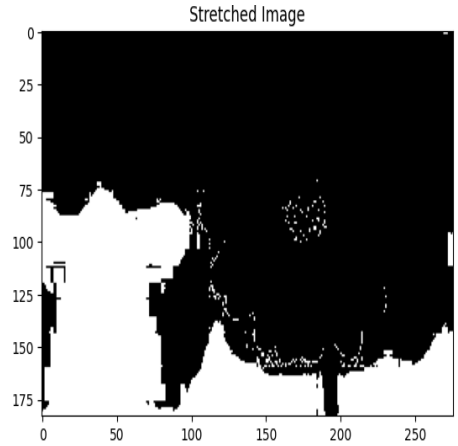
```
plt.subplot(2, 2, 4)
plt.hist(equalized_image.ravel(), 256, [0, 256])
plt.ylabel('% of Pixels')
plt.xlabel('Bins')
plt.title("Equalized Histogram")

plt.tight_layout()
plt.show()

plt.figure(figsize=(10, 8))
plt.subplot(2, 2, 1)
plt.imshow(stretched_image, cmap='gray')
plt.title("Stretched Image")

plt.subplot(2, 2, 2)
plt.hist(stretched_image.ravel(), 256, [0, 256])
plt.title("Stretched Histogram")
plt.ylabel('% of Pixels')
plt.xlabel('Bins')

plt.tight_layout()
plt.show()
```

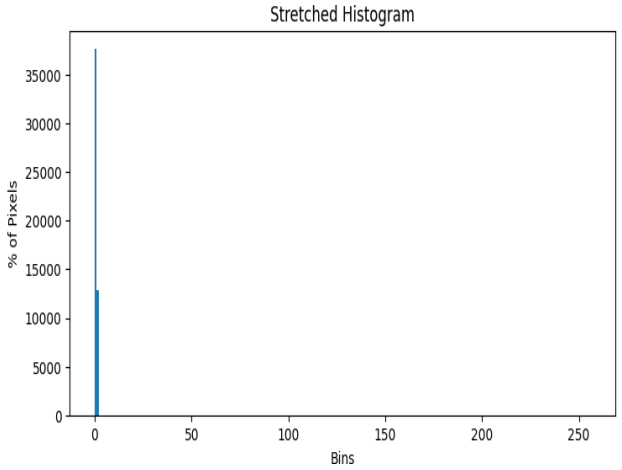
OUTPUT:











RESULT: The program for which covers image enhancement techniques such as histogram equalization and contrast stretching using OpenCV in Python is implemented successfully.

Exp - 3	Image Filtering and Convolution
Date -	

AIM: To write a program and execute for which covers image filtering and convolution using OpenCV in Python.

ALGORITHM:

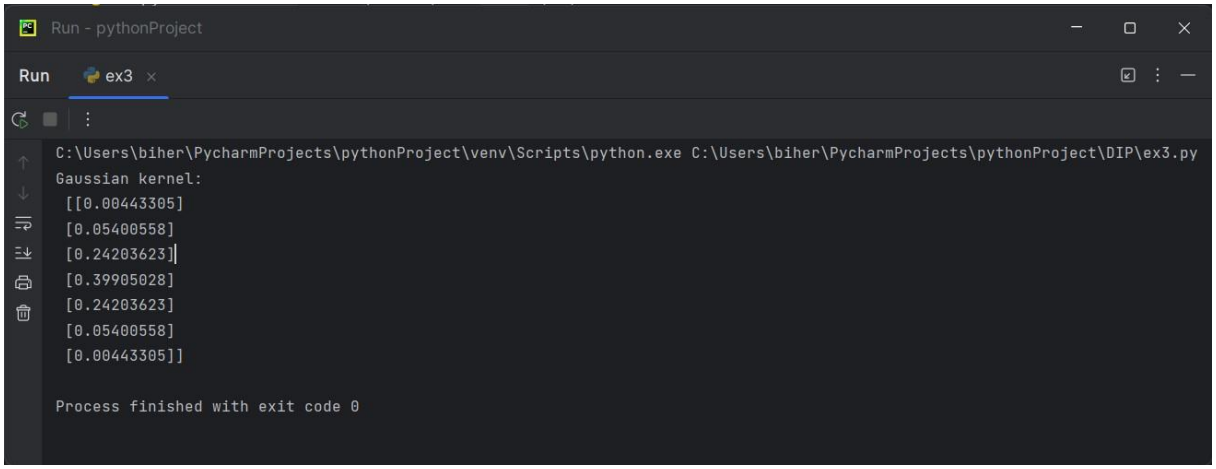
1. Import Libraries.
2. Load the Image.
3. Use OpenCV's cv2.imread() function to load the image in grayscale mode (cv2.IMREAD_GRAYSCALE).
4. Gaussian Blur for Noise Reduction.
5. Use OpenCV's cv2.imread() function to load the image in grayscale mode (cv2.IMREAD_GRAYSCALE).
6. Laplacian Filter for Edge Enhancement.
7. Apply cv2.Laplacian() to the original image for edge enhancement.
8. Sobel Filter for Edge Detection.
9. Apply cv2.Sobel() separately in the X and Y directions for edge detection
10. Display Original and Filtered Images.

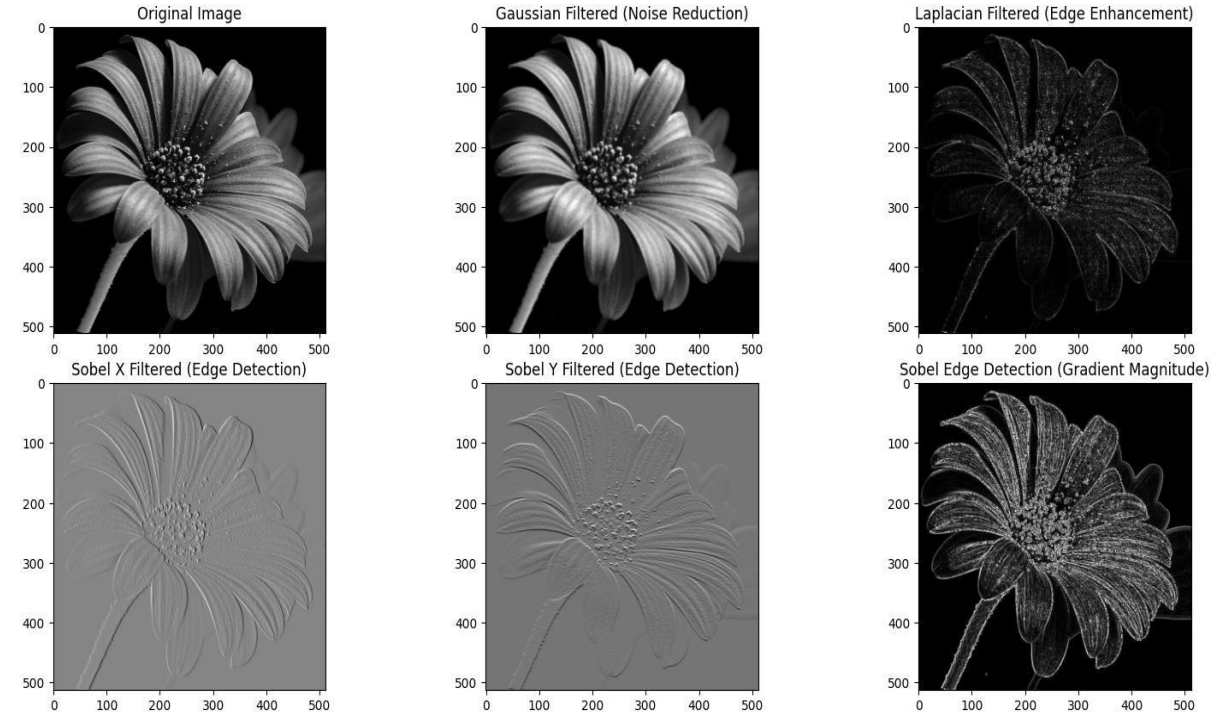
PROGRAM:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

image_path = "1.png"
image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
kernel_size = (7,7)
sigma = 1.0
gaussian_filtered = cv2.GaussianBlur(image, kernel_size, sigma)
laplacian_filtered = cv2.Laplacian(image, cv2.CV_64F)
laplacian_filtered = np.uint8(np.abs(laplacian_filtered))
sobel_filtered_x = cv2.Sobel(image, cv2.CV_64F, 1, 0, ksize=3)
sobel_filtered_y = cv2.Sobel(image, cv2.CV_64F, 0, 1, ksize=3)
sobel_filtered = cv2.magnitude(sobel_filtered_x, sobel_filtered_y)
sobel_filtered = np.uint8(sobel_filtered)

plt.figure(figsize=(12, 8))
plt.subplot(2, 3, 1)
plt.imshow(image, cmap='gray')
plt.title("Original Image")
plt.subplot(2, 3, 2)
plt.imshow(gaussian_filtered, cmap='gray')
plt.title("Gaussian Filtered (Noise Reduction)")
plt.subplot(2, 3, 3)
plt.imshow(laplacian_filtered, cmap='gray')
plt.title("Laplacian Filtered (Edge Enhancement)")
plt.subplot(2, 3, 4)
plt.imshow(sobel_filtered_x, cmap='gray')
plt.title("Sobel X Filtered (Edge Detection)")

plt.subplot(2, 3, 5)
```

```
plt.imshow(sobel_filtered_y, cmap='gray')
plt.title("Sobel Y Filtered (Edge Detection)")
plt.subplot(2, 3, 6)
plt.imshow(sobel_filtered, cmap='gray')
plt.title("Sobel Edge Detection (Gradient Magnitude)")
plt.tight_layout()
plt.show()

a = cv2.getGaussianKernel(7,sigma)
print("Gaussian kernel:\n", a)
```

OUTPUT:





RESULT: The program for which covers image filtering and convolution using OpenCV in Python is executed successfully.

Exp - 4	Frequency Domain Analysis and Filtering using the Fourier Transform
Date -	

AIM: To write a program for which covers frequency domain analysis and filtering using the Fourier Transform in Python with OpenCV.

ALGORITHM:

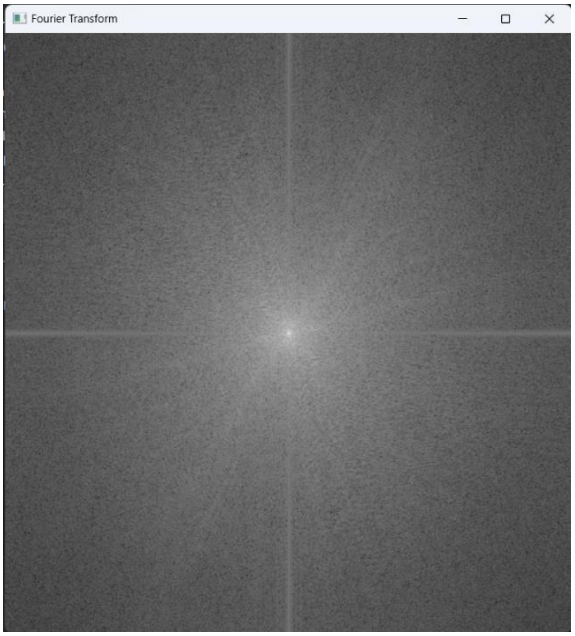
1. Import Libraries and load the Image.
2. Use OpenCV's cv2.imread() function to load the image in grayscale mode (cv2.IMREAD_GRAYSCALE).
3. Perform 2D Discrete Fourier Transform (DFT).
4. Use cv2.dft() to perform the 2D DFT on the loaded grayscale image.
5. Define Low-Pass Filter in Frequency Domain.
6. Create a 2D array for the low-pass filter (lp_filter) and set its values to 1 in the desired region.
7. Apply Low-Pass Filter to the Shifted DFT Image.
8. Store the filtered DFT image in a variable (dft_filtered).
9. Perform Inverse DFT to Get Filtered Image.
10. Use cv2.idft() to perform the inverse DFT on the filtered DFT image.
11. Display Original, DFT Magnitude, Filtered DFT Magnitude, Low-Pass Filter, and Filtered Image.
12. Use plt.imshow() to display the images in the respective subplots.
13. Add titles to subplots for clarity.
14. Use plt.tight_layout() to adjust the spacing between subplots.

PROGRAM:

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

image = "surprising.png"
gray = cv2.imread(image, cv2.IMREAD_GRAYSCALE)
fourier = cv2.dft(np.float32(gray), flags=cv2.DFT_COMPLEX_OUTPUT)
fourier_shift = np.fft.fftshift(fourier)
magnitude =
20*np.log(cv2.magnitude(fourier_shift[:,:,0],fourier_shift[:,:,1]))
magnitude = cv2.normalize(magnitude, None, 0, 255, cv2.NORM_MINMAX,
cv2.CV_8UC1)

cv2.imshow('Fourier Transform', magnitude)
cv2.waitKey(0)
cv2.destroyAllWindows()
ima = cv2.imread(image, 0)
DFT = cv2.dft(np.float32(ima), flags=cv2.DFT_COMPLEX_OUTPUT)
shift = np.fft.fftshift(DFT)
row, col = ima.shape
center_row, center_col = row // 2, col // 2
mask = np.zeros((row, col, 2), np.uint8)
mask[center_row - 30:center_row + 30, center_col - 30:center_col +
30] = 1
fft_shift = shift * mask
fft_ifft_shift = np.fft.ifftshift(fft_shift)
```

```
imageThen = cv2.idft(fft_ifft_shift)
imageThen = cv2.magnitude(imageThen[:,:,0], imageThen[:,:,1])

plt.figure(figsize=(10,10))
plt.subplot(121), plt.imshow(ima, cmap='gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122), plt.imshow(imageThen, cmap='gray')
plt.title('Magnitude Spectrum'), plt.xticks([]), plt.yticks([])
plt.show()
```

OUTPUT:



RESULT: The program for which covers frequency domain analysis and filtering using the Fourier Transform in Python with OpenCV is executed successfully.

Exp - 5	Image Compression Techniques using Run-Length Encoding (RLE) and Discrete Cosine Transform (DCT)
Date -	

AIM: To write a program and execute for which covers image compression techniques such as Run-Length Encoding (RLE) and Discrete Cosine Transform (DCT) using Python.

ALGORITHM:

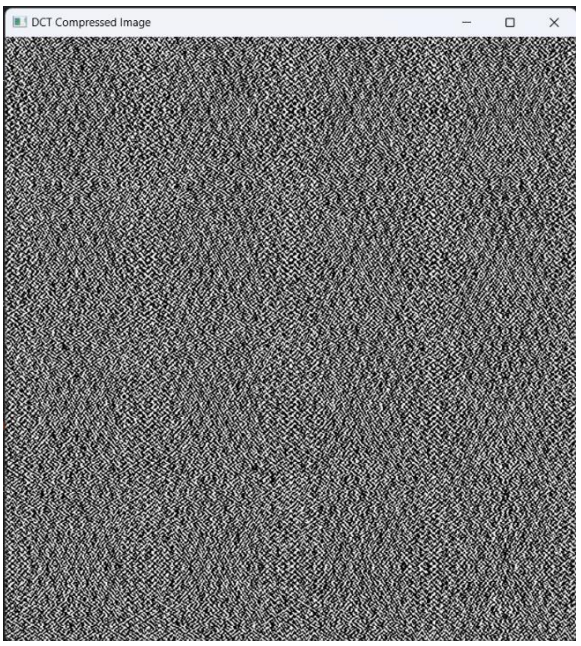
1. Load Image.
2. Run-Length Encoding (RLE) Compression.
3. Define a function run_length_encode(data) to perform RLE compression.
4. Discrete Cosine Transform (DCT) Compression.
5. Initialize an array (dct_compressed) with zeros, having the same shape as the original image.
6. Display Images.
7. Input: Original image (image), RLE compressed data (rle_compressed), DCT compressed data (dct_compressed).
8. Wait for User Input and Close Windows.
9. Input: User input from keyboard (cv2.waitKey(0)).

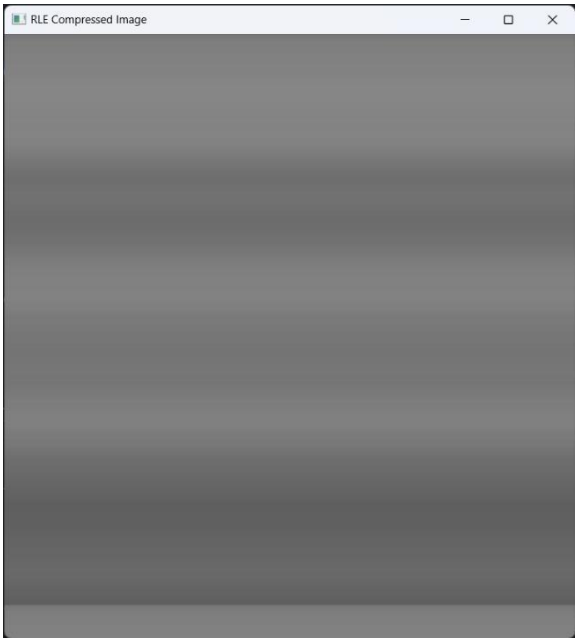
PROGRAM:

```
import cv2
import numpy as np
image_path = "2.png"
image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

def rle_encode_image(image):
    data = image.flatten()
    encoded = []
    count = 1
    for i in range(1, len(data)):
        if data[i] == data[i - 1]:
            count += 1
        else:
            encoded.append((data[i - 1], count))
            count = 1
    encoded.append((data[-1], count))
    return encoded
def rle_decode_image(encoded, image_shape):
    decoded_data = np.zeros(image_shape, dtype=np.uint8)
    current_pixel = 0
    for color, count in encoded:
        decoded_data[current_pixel:current_pixel + count] = color
        current_pixel += count
    return decoded_data

encoded_data = rle_encode_image(image)
rle_image = rle_decode_image(encoded_data, image.shape)
# Perform Discrete Cosine Transform (DCT) compression
dct_size = 8
dct_compressed = np.zeros_like(image)

for i in range(0, image.shape[0], dct_size):
    for j in range(0, image.shape[1], dct_size):
```

```
        dct_block = cv2.dct(np.float32(image[i:i+dct_size,
j:j+dct_size])/255.0)
        dct_compressed[i:i+dct_size, j:j+dct_size] = dct_block
cv2.imshow("Original Image", image)
cv2.imshow('RLE Compressed Image', rle_image)
cv2.imshow("DCT Compressed Image",
cv2.idct(np.float32(dct_compressed)))
cv2.waitKey(0)
cv2.destroyAllWindows()
```

OUTPUT:





RESULT: The program for which covers image compression techniques such as Run-Length Encoding (RLE) and Discrete Cosine Transform (DCT) using Python is executed successfully.

Exp - 6	Morphological Image Processing Techniques
Date -	

AIM: To write and execute a program for which covers morphological image processing techniques using Python and OpenCV.

ALGORITHM:

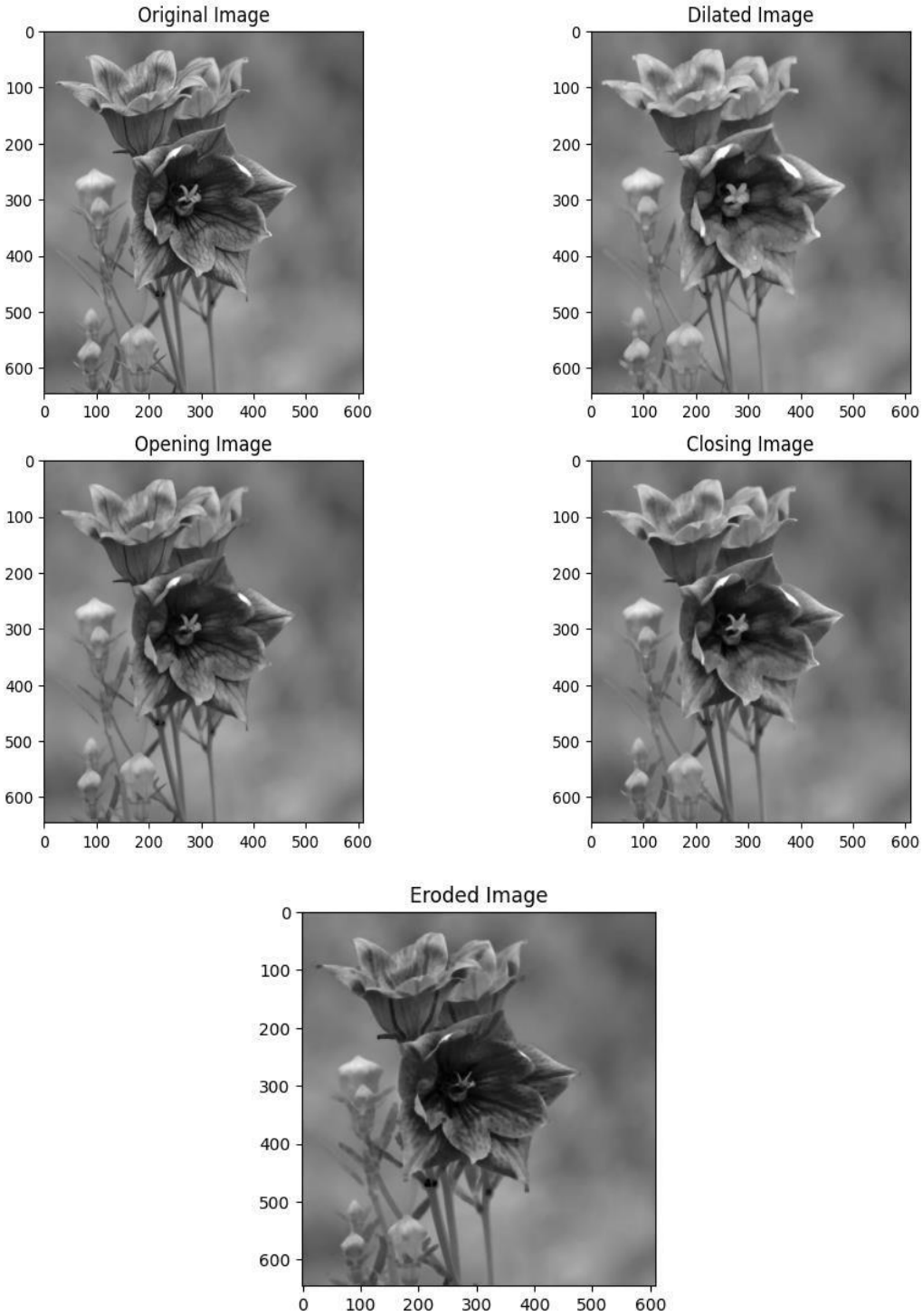
1. Load Image.
2. Define a Kernel for Dilation and Erosion.
3. a square kernel of size 5x5 using NumPy (np.ones((5, 5), np.uint8)).
4. Perform Dilation Operation.
5. Input: Original image (image), Kernel (kernel), Number of iterations (iterations = 1).
6. Perform Erosion Operation.
7. Input: Original image (image), Kernel (kernel), Number of iterations (iterations = 1).
8. Perform Opening Operation.
9. Input: Original image (image), Kernel (kernel).
10. Perform Closing Operation.
11. Input: Original image (image), Kernel (kernel).
12. Display Original and Processed Images.
13. Input: Original image (image), Dilated image (dilated_image), Eroded image (eroded_image), Opened image (opening_image), Closed image (closing_image).

PROGRAM:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
image_path = "surprising.png"
image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

kernel = np.ones((5, 5), np.uint8)
dilated_image = cv2.dilate(image, kernel, iterations=1)
eroded_image = cv2.erode(image, kernel, iterations=1)
opening_image = cv2.morphologyEx(image, cv2.MORPH_OPEN, kernel)
closing_image = cv2.morphologyEx(image, cv2.MORPH_CLOSE, kernel)

plt.figure(figsize=(12, 8))
plt.subplot(2, 3, 1)
plt.imshow(image, cmap='gray')
plt.title("Original Image")
plt.subplot(2, 3, 2)
plt.imshow(dilated_image, cmap='gray')
plt.title("Dilated Image")
plt.subplot(2, 3, 3)
plt.imshow(eroded_image, cmap='gray')
plt.title("Eroded Image")
plt.subplot(2, 3, 4)
plt.imshow(opening_image, cmap='gray')
plt.title("Opening Image")
plt.subplot(2, 3, 5)
plt.imshow(closing_image, cmap='gray')
plt.title("Closing Image")
```

```
plt.tight_layout()
plt.show()
```

OUTPUT:



RESULT: The program for which covers morphological image processing techniques using Python and OpenCV is implemented and executed successfully.

Exp - 7	Image Segmentation Techniques using OpenCV
Date -	

AIM: To write a program which covers image segmentation techniques using Python and OpenCV.

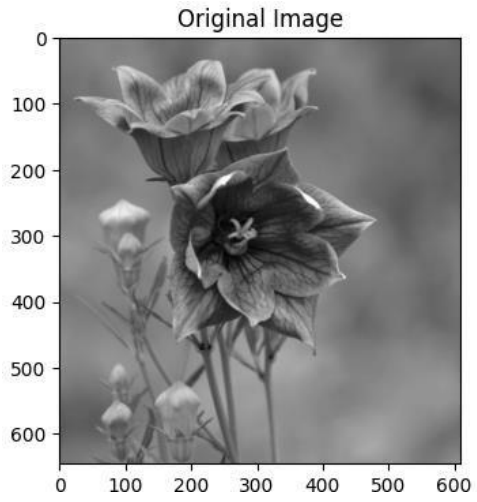
ALGORITHM:

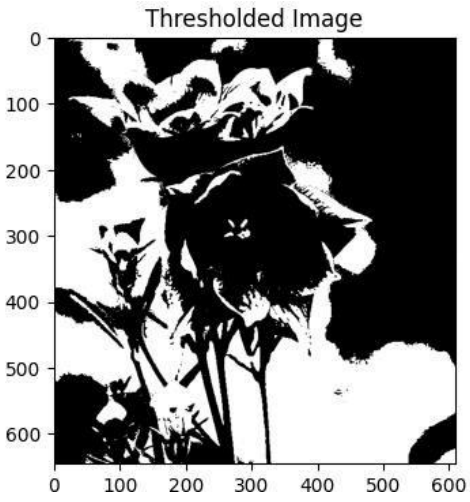
1. Import necessary libraries: `cv2`, `numpy`, and `matplotlib.pyplot`.
2. Specify the path to the input image.
3. Read the image in grayscale using `cv2.imread`.
4. Apply a thresholding technique using `cv2.threshold`.
5. Perform connected component labeling using `cv2.connectedComponents`.
6. Apply the Canny edge detector using `cv2.Canny`.
7. Create a 2x3 subplot for visualization using `matplotlib.pyplot`.
8. Display the original image, thresholded image, connected components, and Canny edge detection results using `plt.imshow`.
9. Set titles for each subplot.
10. Adjust layout for better visualization using `plt.tight_layout`.
11. Show the plot using `plt.show`.

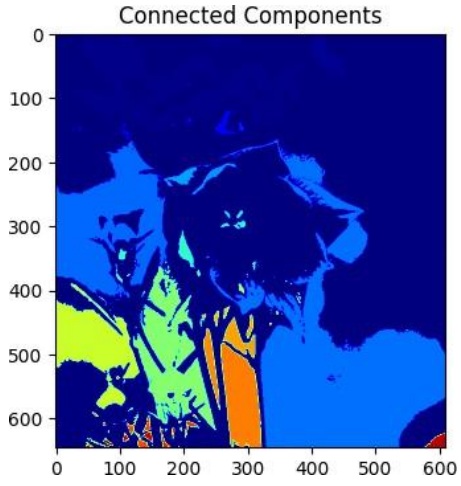
PROGRAM:

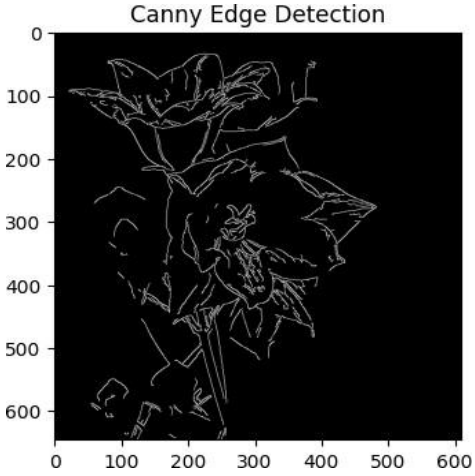
```
import cv2
import matplotlib.pyplot as plt

image_path = "surprising.png"
image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

_, thresholded_image = cv2.threshold(image, 127, 255,
cv2.THRESH_BINARY)
_, labels = cv2.connectedComponents(thresholded_image)
canny_image = cv2.Canny(image, threshold1=100, threshold2=200)


plt.figure(figsize=(12, 8))
plt.subplot(2, 3, 1)
plt.imshow(image, cmap='gray')
plt.title("Original Image")
plt.subplot(2, 3, 2)
plt.imshow(thresholded_image, cmap='gray')
plt.title("Thresholded Image")
plt.subplot(2, 3, 3)
plt.imshow(labels, cmap='jet')
plt.title("Connected Components")
plt.subplot(2, 3, 4)
plt.imshow(canny_image, cmap='gray')
plt.title("Canny Edge Detection")
plt.tight_layout()
plt.show()
```

OUTPUT:









RESULT: Hence the program that covers image segmentation techniques using Python and OpenCV has been executed successfully.

Exp - 8	Finding Contours and Calculating Region-Based Properties
Date -	

AIM: The program aims to load an image, find contours in the image, draw the contours on a black background, calculate region-based properties (centroid coordinates), and display the original image along with the contour.

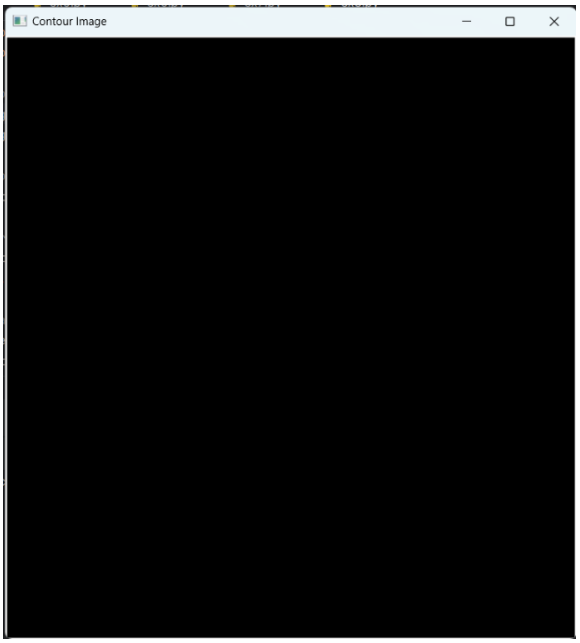
ALGORITHM:

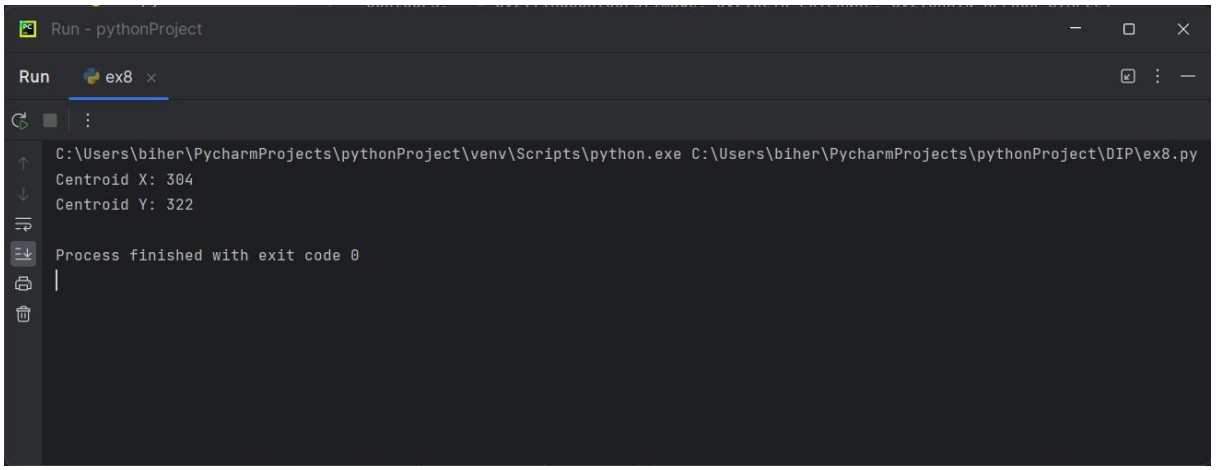
1. *Load Image: * Specify the file path of your image.
2. *Read Image: * Use OpenCV to read the image in grayscale mode.
3. *Find Contours: * Utilize the `findContours` function to identify contours in the image.
4. *Draw Contours: * Create a black background and draw the identified contours on it.
5. *Calculate Moments: * Compute image moments for region-based properties using `cv2.moments`.
6. *Extract Centroid Coordinates: * Calculate centroid coordinates from the moments.
7. *Display Images: * Use `cv2.imshow` to display the original and contour images.
8. *Wait for User Input: * Use `cv2.waitKey(0)` to wait until a key is pressed.
9. *Close Windows: * Close image display windows with `cv2.destroyAllWindows()`.
10. *Print Centroid Coordinates: * Output the calculated centroid coordinates (X and Y) to the console.

PROGRAM:

```
import cv2
import numpy as np

image_path = "surprising.png"
image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
contours, _ = cv2.findContours(image, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
contour_image = np.zeros_like(image)
cv2.drawContours(contour_image, contours, -1, 255, 2)
moments = cv2.moments(contours[0])
centroid_x = int(moments["m10"] / moments["m00"])
centroid_y = int(moments["m01"] / moments["m00"])

cv2.imshow("Original Image", image)
cv2.imshow("Contour Image", contour_image)
cv2.waitKey(0)
cv2.destroyAllWindows()

print("Centroid X:", centroid_x)
print("Centroid Y:", centroid_y)
```

OUTPUT:





RESULT: The program will display the original image and the contour image, while also printing the calculated centroid coordinates (X and Y) on the console.

Exp - 9	Colour Balance Correction using White Balance Algorithms
Date -	

AIM: The aim of this Python code is to demonstrate colour image processing techniques using the Balance White Algorithms.

ALGORITHM:

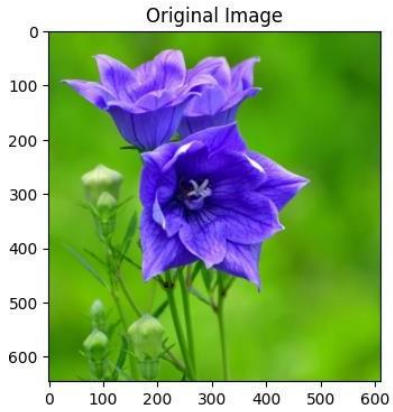
1. Load a color image from the specified file path using OpenCV's `cv2.imread()` function.
2. Convert the loaded image to three different color models:
   - Grayscale using `cv2.cvtColor()` with the `cv2.COLOR_BGR2GRAY` conversion.
   - HSV using `cv2.cvtColor()` with the `cv2.COLOR_BGR2HSV` conversion.
   - CIELab using `cv2.cvtColor()` with the `cv2.COLOR_BGR2LAB` conversion.
3. Display the original and converted images using Matplotlib. The original image is displayed along with the grayscale, HSV, and CIELab versions in a 2x2 grid
4. Apply color balance correction to the original image using the white balance algorithm from OpenCV's xphoto module. The `cv2.xphoto.createSimpleWB().balanceWhite()` function is used for this purpose.
5. Display the color balance corrected image using Matplotlib.

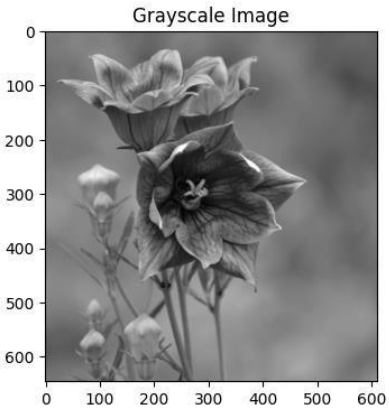
PROGRAM:

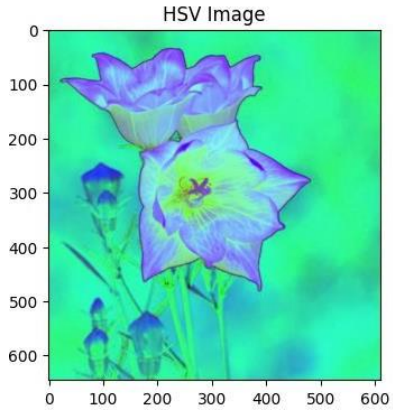
```
import cv2
import numpy as np
import matplotlib.pyplot as plt

image_path = "surprising.png"
image = cv2.imread(image_path)
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
hsv_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
lab_image = cv2.cvtColor(image, cv2.COLOR_BGR2LAB)
plt.figure(figsize=(12, 8))

plt.subplot(2, 2, 1)
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
plt.title("Original Image")
plt.subplot(2, 2, 2)
plt.imshow(gray_image, cmap="gray")
plt.title("Grayscale Image")
plt.subplot(2, 2, 3)
plt.imshow(hsv_image)
plt.title("HSV Image")
plt.subplot(2, 2, 4)
plt.imshow(lab_image)
plt.title("CIELab Image")
plt.tight_layout()
plt.show()

wb = cv2.xphoto.createGrayworldWB()
wb.setSaturationThreshold(0.99)
image = wb.balanceWhite(image)
gray_world_image = cv2.xphoto.createSimpleWB().balanceWhite(image)
gray_world_image = cv2.cvtColor(gray_world_image, cv2.COLOR_BGR2RGB)
```

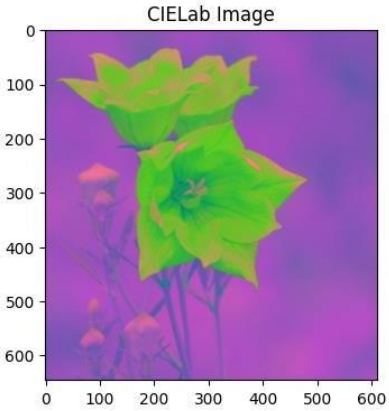
```
plt.figure(figsize=(8, 6))
plt.imshow(gray_world_image)
plt.title("Color Balance Corrected Image")
plt.axis("off")
plt.show()
```

OUTPUT:









RESULT: The result of running this code is a series of visualizations showing the original colour image and its conversions to grayscale, HSV, and CIE Lab colour models.

Exp - 10	Edge Detection using the Canny Edge Detector
Date -	

AIM: The aim of this Python code is to perform basic image analysis and Apply edge detection using the Canny edge detector.

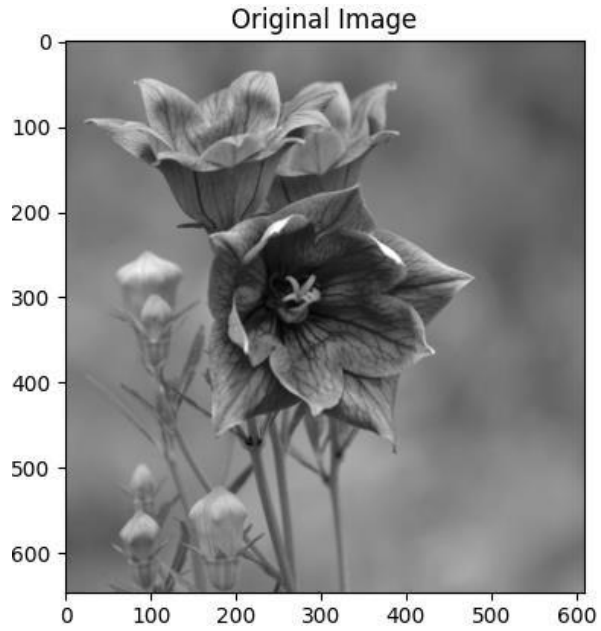
ALGORITHM:

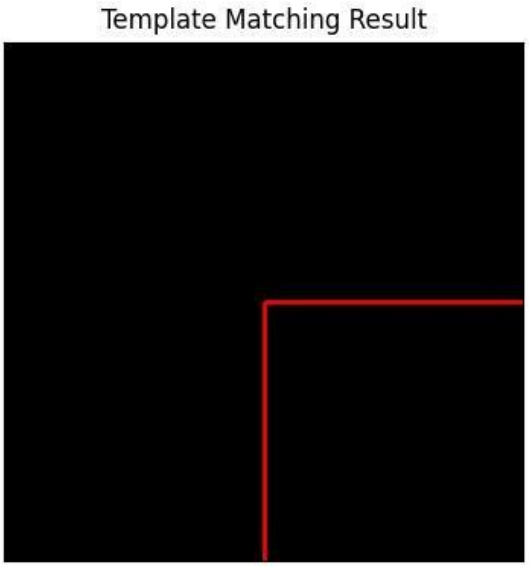
1. Load the original image and the template image using OpenCV's `cv2.imread` function.
2. Use the `cv2.matchTemplate` function to perform template matching on the original image with the template image. This results in a correlation map (`result`).
3. Find the location of the maximum correlation in the correlation map using `cv2.minMaxLoc`.
4. Highlight the region with the maximum correlation in the original image using a rectangle.
5. Apply the Canny edge detector to the original image using `cv2.Canny`.
6. Display the original image, the template matching result, and the edge detection result using matplotlib.

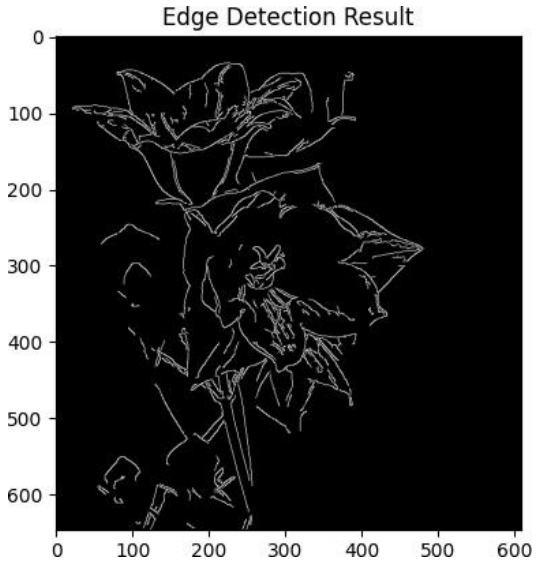
PROGRAM:

```
import cv2
import matplotlib.pyplot as plt
import matplotlib.patches as patches

image_path = "surprising.png"
image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
template_path = "surprising.png"
template = cv2.imread(template_path, cv2.IMREAD_GRAYSCALE)
result = cv2.matchTemplate(image, template, cv2.TM_CCOEFF_NORMED)
min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(result)
top_left = max_loc
bottom_right = (top_left[0] + template.shape[1], top_left[1] +
template.shape[0])

edges = cv2.Canny(image, threshold1=100, threshold2=200)
plt.figure(figsize=(12, 8))
plt.subplot(1, 3, 1)
plt.imshow(image, cmap='gray')
plt.title("Original Image")
plt.subplot(1, 3, 2)
plt.imshow(result, cmap='gray')
plt.title("Template Matching Result")
plt.xticks([]), plt.yticks([])  # Hide axes
rectangle = patches.Rectangle(top_left, template.shape[1],
template.shape[0], linewidth=2, edgecolor='r', facecolor='none')
plt.gca().add_patch(rectangle)  # Highlight the matching area
plt.subplot(1, 3, 3)
plt.imshow(edges, cmap='gray')
plt.title("Edge Detection Result")

plt.tight_layout()
plt.show()
```

OUTPUT:







RESULT: The result is a visual representation of the original image, the template matching result, and the edge detection result. The edge detection result is displayed using the Canny edge detector.