**Roll No: 120A2036**

**Name: Varad Patil**

**Class: TE-EXTC**

**Batch: A2**

**EXPERIMENT NO. 5**

**TO PERFORM EROSION AND DILATION OF IMAGE AND ITS APPLICATION FOR OPENING, CLOSING, BOUNDARY EXTRACTION**

**EXPERIMENT NO. 5: Morphological Processing**

**AIM: -** To perform erosion and dilation on a black and white image and its application for Opening, Closing, boundary extraction

**OBJECTIVES:**

1. To apply and compare performance of erosion and dilation on binary images.
2. To perform Opening and closing using erosion and dilation.
3. To perform boundary extraction using erosion.
4. Observe and understand applications from the output.

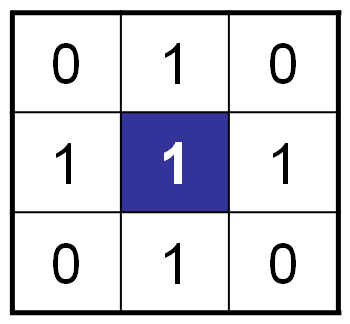
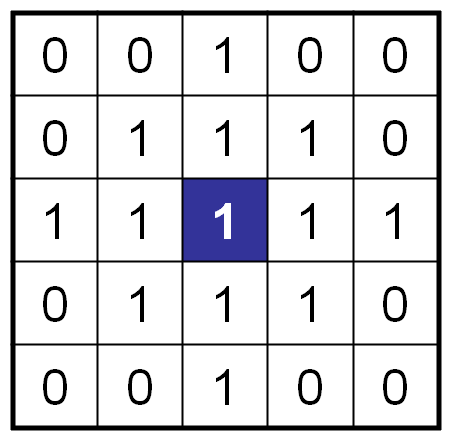
**EQUIPMENTS/SOFTWARE:** Python

**THEORY: -**

Once segmentation is complete, morphological operations can be used to remove imperfections in the segmented image and provide information on the form and structure of the image. Morphological image processing (or *morphology*) describes a range of image processing techniques that deal with the shape (or morphology) of features in an image

Morphological operations are typically applied to remove imperfections introduced during segmentation, and so typically operate on bi-level images.

Structuring elements can be any size and make any shape. However, for simplicity we will use rectangular structuring elements with their origin at the middle pixel.



**Morphological Operations**

Fundamentally morphological image processing is very like spatial filtering

The structuring element is moved across every pixel in the original image to give a pixel in a new processed image

The value of this new pixel depends on the operation performed.

There are two basic morphological operations: erosion and dilation.

**Dilation** of image *f*  by structuring element *s* is given by *f*   *s*

The structuring element s is positioned with its origin at *(x, y)* and the new pixel value is determined using the rule:

**Original Image Processed Image with Dilated Pixels Structuring Element**

****

**Original image Dilation by 3\*3 square Dilation by 5\*5 square**

**structuring element structuring element**

**Erosion** Erosion of image *f* by structuring element *s* is given by *f* ⊖ *s*

The structuring element s is positioned with its origin at *(x, y)* and the new pixel value is determined using the rule:

**Compound Operations**

More interesting morphological operations can be performed by performing combinations of erosions and dilations

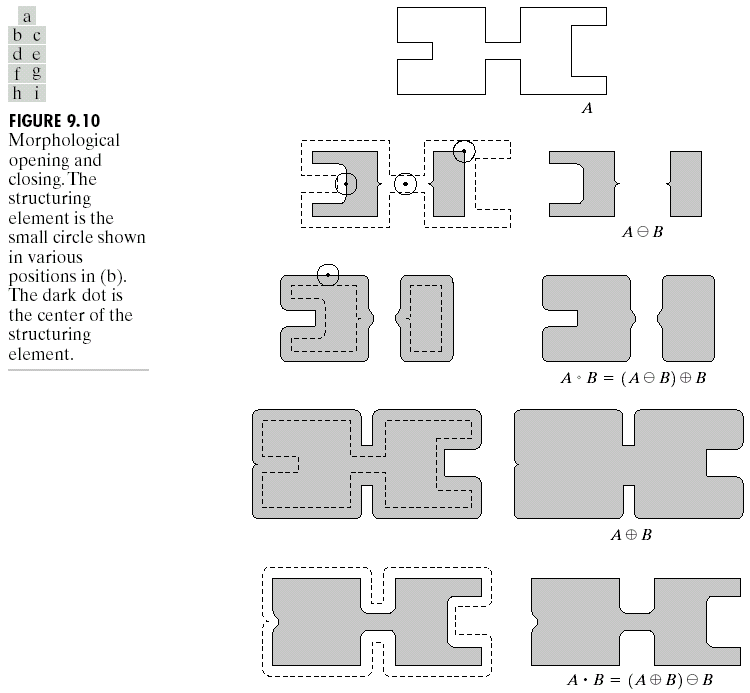
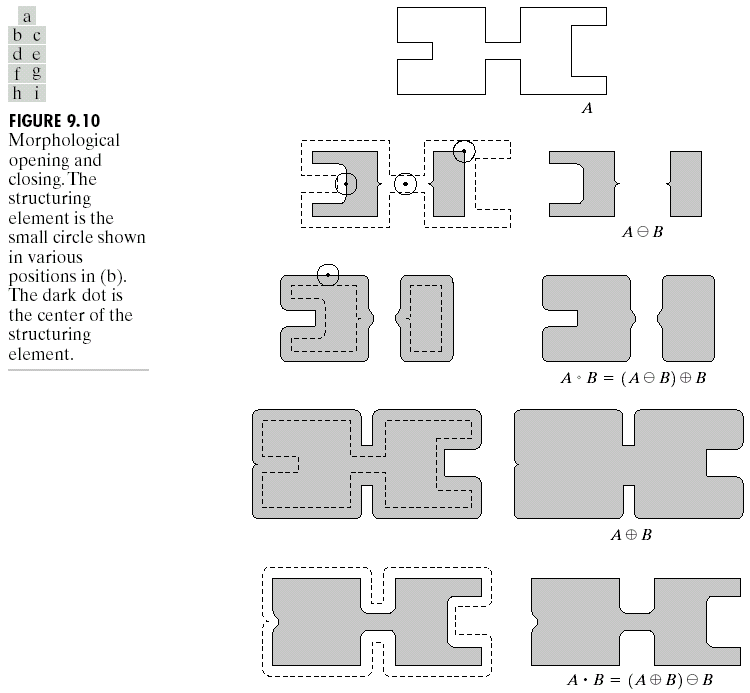
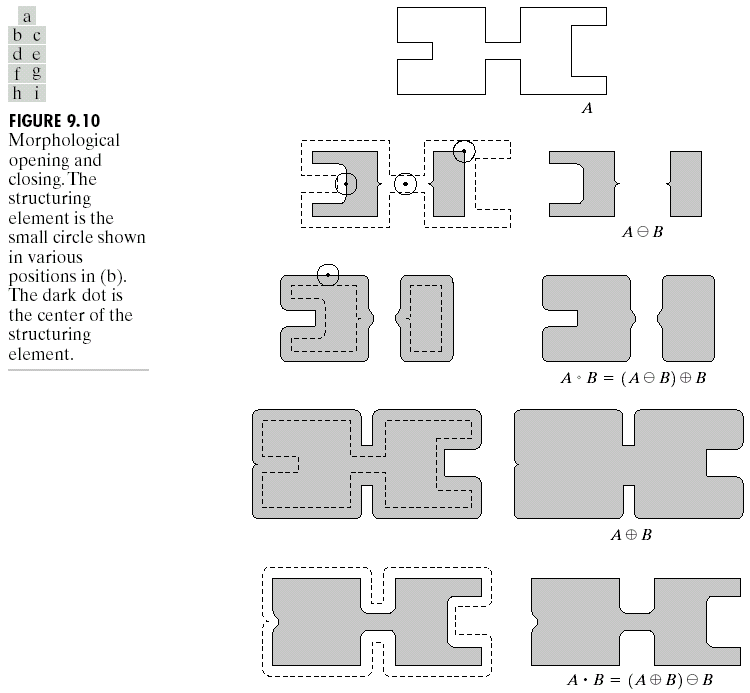
The most widely used of these *compound operations* are:

* + Opening
  + Closing

**Opening**

The opening of image *f* by structuring element *s,* denoted *f* ○ *s* is simply an erosion followed by a dilation

***f* ○ *s = (f* ⊖*s)***  ***s***

****

**Original Image Processed Image**

**Closing**

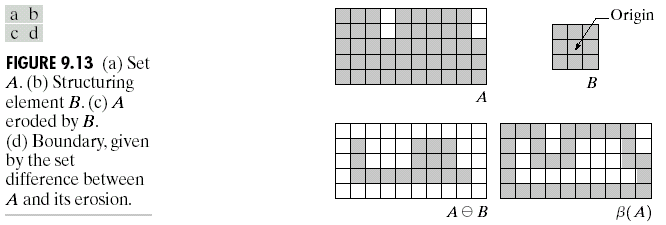
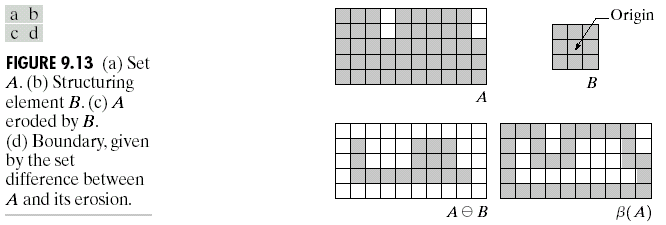
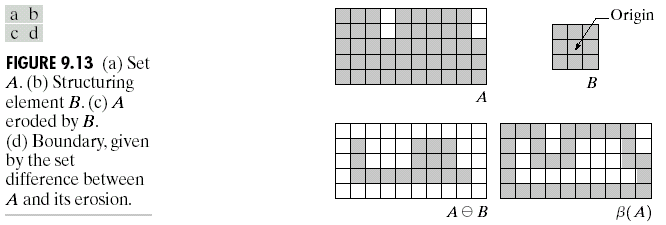
The closing of image f by structuring element s, denoted f • s is simply a dilation followed by an erosion

 f • s = (f s)⊖s

**Boundary Extraction**

Extracting the boundary (or outline) of an object is often extremely useful

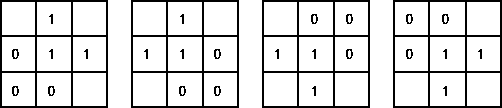
The boundary can be given simply as  *β(A) = A – (A⊖B)*

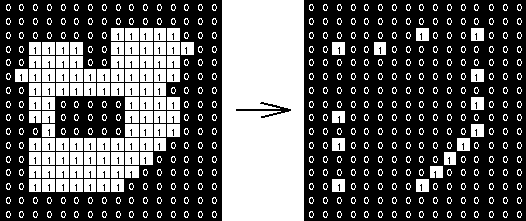
****

**Hit or Miss Transform**

The hit-and-miss transform is a general binary morphological operation that can be used to look for particular patterns of foreground and background pixels in an image. It is actually the basic operation of binary morphology since almost all the other binary morphological operators can be derived from it. As with other binary morphological operators it takes as input a [binary image](http://homepages.inf.ed.ac.uk/rbf/HIPR2/binimage.htm) and a [structuring element](http://homepages.inf.ed.ac.uk/rbf/HIPR2/strctel.htm), and produces another binary image as output.

[structuring element](http://homepages.inf.ed.ac.uk/rbf/HIPR2/strctel.htm)s

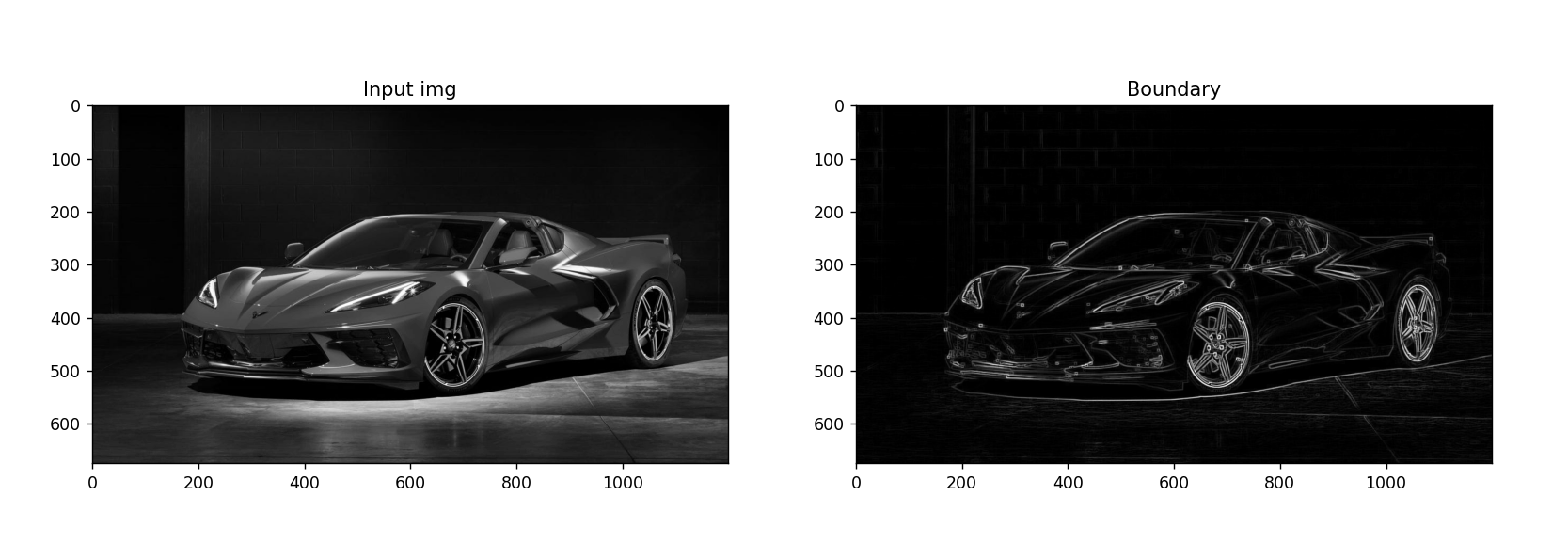


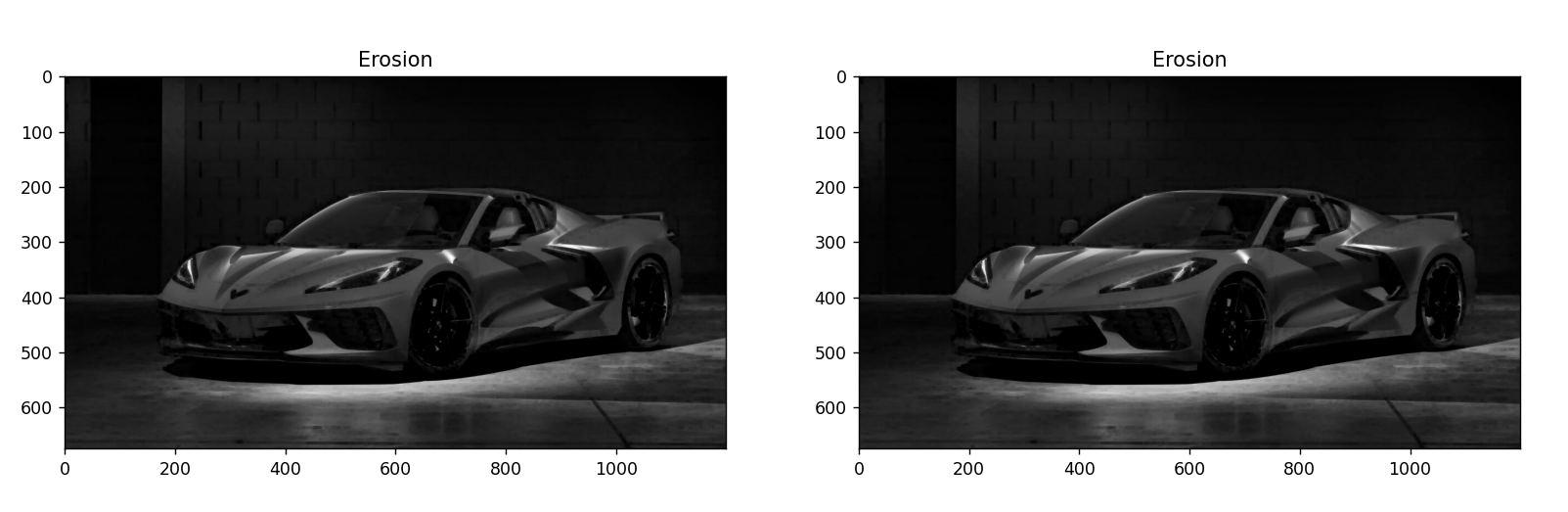


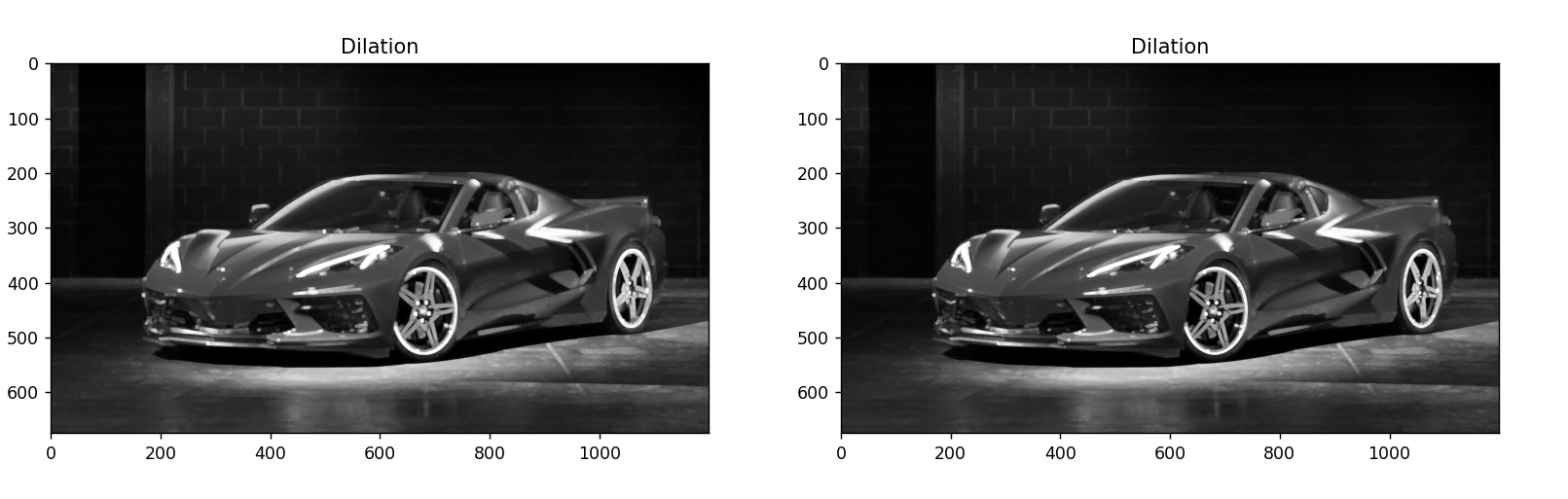
**Code:-**

import numpy as np # library used for working with arrays  
import matplotlib.pylab as plt # library used for ploting,graph  
import cv2  
img = cv2.imread('D:\\college related\\Third year\\sem6\\IPMV\\CODE\\car.jpg',0)  
#img = cv2.cvtColor(img1, cv2.COLOR\_BGR2GRAY)  
# cv2.imshow("Input image",img)  
# cv2.waitKey(0)  
kernel = np.ones((5,5),np.uint8)  
img\_erosion1 = cv2.erode(img, kernel, iterations=1)  
img\_dilate1 = cv2.dilate(img, kernel, iterations=1)  
img\_boundary = img\_dilate1-img  
# cv2.imshow("Erosion", img\_erosion1)  
# cv2.waitKey(0)  
# cv2.imshow("Dilation", img\_dilate1)  
# cv2.waitKey(0)  
# cv2.imshow("Boundary", img\_boundary)  
  
kernel1=cv2.getStructuringElement(cv2.MORPH\_RECT,(5,5))  
kernel2 =cv2.getStructuringElement(cv2.MORPH\_ELLIPSE,(5,5))  
kernel3= cv2.getStructuringElement(cv2.MORPH\_CROSS,(5,5))  
img\_erosion2 = cv2.erode(img, kernel1, iterations=1)  
img\_dilate2 = cv2.dilate(img, kernel2, iterations=1)  
  
plt.subplot(1,2,1)  
plt.imshow(img, cmap="gray")  
plt.title("Input img")  
  
plt.subplot(1,2,2)  
plt.imshow(img\_boundary, cmap="gray")  
plt.title("Boundary")  
plt.show()  
  
plt.subplot(1,2,1)  
plt.imshow(img\_erosion1, cmap="gray")  
plt.title("Erosion")  
  
plt.subplot(1,2,2)  
plt.imshow(img\_erosion2, cmap="gray")  
plt.title("Erosion")  
plt.show()  
  
plt.subplot(1,2,1)  
plt.imshow(img\_dilate1, cmap="gray")  
plt.title("Dilation")  
  
plt.subplot(1,2,2)  
plt.imshow(img\_dilate2, cmap="gray")  
plt.title("Dilation")  
plt.show()

**Output:-**

****

****

****

**CONCLUSION** :-

Erosion and dilation on a black and white image and its application for Opening, Closing, boundary extraction was performed successfully. Erosion and dilation using different kernel was understood.