**Digital Image Processing**

**Title: Morphological Image Processing**

**Tools Used:** Python

**Procedure:** Open IDLE and perform the following tasks

import cv2

import numpy as np

img = cv2.imread('j.png',0)

kernel = np.ones((5,5),np.uint8)

erosion = cv2.erode(img,kernel,iterations = 1)

dilation = cv2.dilate(img, kernel, iterations = 1)

opening = cv2.morphologyEx(img, cv2.MORPH\_OPEN, kernel)

closing = cv2.morphologyEx(img, cv2.MORPH\_CLOSE, kernel)

# Rectangular Kernel

>>> cv2.getStructuringElement(cv2.MORPH\_RECT,(5,5))

array([[1, 1, 1, 1, 1],

[1, 1, 1, 1, 1],

[1, 1, 1, 1, 1],

[1, 1, 1, 1, 1],

[1, 1, 1, 1, 1]], dtype=uint8)

# Elliptical Kernel

>>> cv2.getStructuringElement(cv2.MORPH\_ELLIPSE,(5,5))

array([[0, 0, 1, 0, 0],

[1, 1, 1, 1, 1],

[1, 1, 1, 1, 1],

[1, 1, 1, 1, 1],

[0, 0, 1, 0, 0]], dtype=uint8)

# Cross-shaped Kernel

>>> cv2.getStructuringElement(cv2.MORPH\_CROSS,(5,5))

array([[0, 0, 1, 0, 0],

[0, 0, 1, 0, 0],

[1, 1, 1, 1, 1],

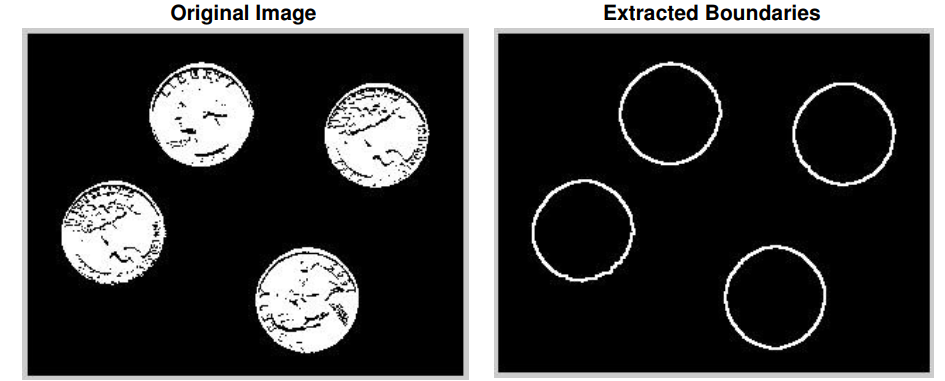
[0, 0, 1, 0, 0],

[0, 0, 1, 0, 0]], dtype=uint8)



**Task 1**

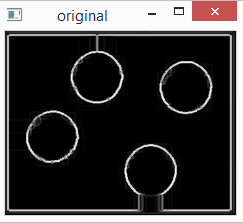
Read the image ‘eight.tif’. Write a function named ‘myMorphology’ to extract the boundaries of coins from the read image.   
[HINTS]:  
**1. First close the image then perform erosion.  
2. Take the difference of two images to find boundaries.**



**Code:**

import cv2 as cv  
import numpy as np  
image=cv.imread('coins.JPG'**,0**)  
kernel=np.ones((**15,15**)**,** np.uint8)  
closing = cv.morphologyEx(image**,**cv.MORPH\_CLOSE**,**kernel)  
kernel=cv.getStructuringElement(cv.MORPH\_ELLIPSE**,**(**5,5**))  
erosion=cv.erode(closing**,**kernel**,**iterations=**1**)  
boundaryImage=closing-erosion  
cv.imshow("original"**,**boundaryImage)  
cv.waitKey(**0**)  
cv.destroyAllWindows()

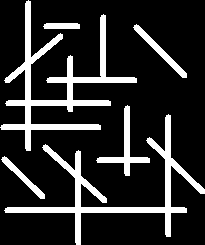
**Output:**



**Task 2**

Read the image ‘lines.png’. Use the opening operator to separate horizontal and vertical lines.

[HINTS]: Experiment with structuring elements of sizes 7x3, 9x3, 11x3 etc. to remove horizontal lines. Use the transpose of these structuring elements to eliminate vertical lines.

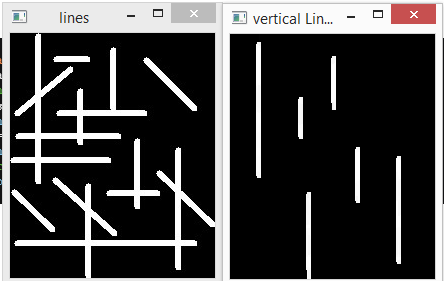


**Code:**

import cv2 as cv  
import numpy as np  
image=cv.imread('lines.jpg'**,0**)  
cv.imshow("lines"**,**image)  
kernel=np.ones((**15,2**)**,**np.uint8)  
#kernal1=np.ones(3,13)  
verticalLines=cv.erode(image**,**kernel**,**iterations=**1**)  
#horizontalLines=cv.erode(image,kernal1,iterations=1)  
cv.imshow("vertical Lines"**,**verticalLines)  
#cv.imshow("horizontal Lines",horizontalLines)  
cv.waitKey(**0**)

**Output:**

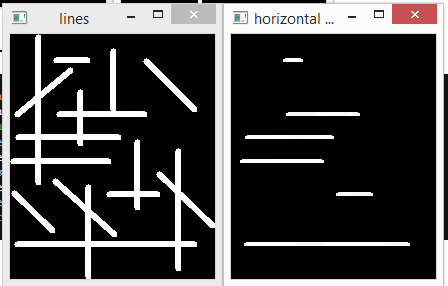
**For vertical lines:**



**For horizontal lines:**

import cv2 as cv  
import numpy as np  
image=cv.imread('lines.jpg'**,0**)  
cv.imshow("lines"**,**image)  
#kernel=np.ones((15,2),np.uint8)  
kernal1=np.ones((**3,17**)**,**np.uint8)  
#verticalLines=cv.erode(image,kernel,iterations=1)  
horizontalLines=cv.erode(image**,**kernal1**,**iterations=**1**)  
#cv.imshow("vertical Lines",verticalLines)  
cv.imshow("horizontal Lines"**,**horizontalLines)  
cv.waitKey(**0**)

**Output:**



**Task 3**

You are provided with a printed document image (Image.png). You need to find the approximate number of lines and words in the given image.  
Hints:

* Binarize the image
* dilation with a horizontal structuring element to merge all characters in a line.
* Apply connected component labeling algorithm (connected component) to find the number of lines.
* Use a smaller horizontal structuring element to merge characters in a word together and again use the CC labeling algorithm to find the number of words in the image.

**Code:**

import numpy as np  
import cv2  
  
img = cv2.imread('text.png'**,0**)  
kernel=np.ones((**3,15**)**,**np.uint8)  
lines=cv2.erode(img**,**kernel**,**iterations=**1**)  
  
totalLabels**,**labels**,**stats**,**centroids=cv2.connectedComponentsWithStats(~lines**,8,**cv2.CV\_32S)  
print("total lines are:"**,**totalLabels-**1**)  
  
colors=np.random.randint(**0,255,**size=(totalLabels**,3**)**,**dtype=np.uint8)  
colors[**0**]=[**0,0,0**]  
colored\_components=colors[labels]  
  
cv2.imshow('output'**,**~lines)  
cv2.imshow('colored image'**,**colored\_components)  
cv2.waitKey(**0**)

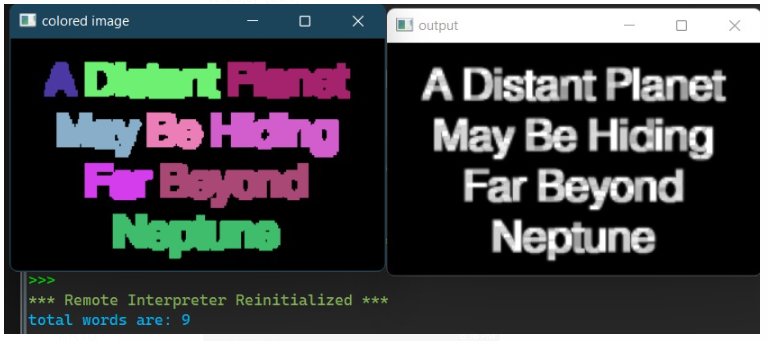
**Output:**



**Code:**

import numpy as np  
import cv2  
  
img = cv2.imread('text.png'**,0**)  
kernel=np.ones((**3,5**)**,**np.uint8)  
words=cv2.erode(img**,**kernel**,**iterations=**1**)  
  
totalLabels**,**labels**,**stats**,**centroids=cv2.connectedComponentsWithStats(~words**,8,**cv2.CV\_32S)  
print("total words are:"**,**totalLabels-**1**)  
  
colors=np.random.randint(**0,255,**size=(totalLabels**,3**)**,**dtype=np.uint8)  
colors[**0**]=[**0,0,0**]  
colored\_components=colors[labels]  
  
cv2.imshow('output'**,**~words )  
cv2.imshow('colored image'**,**colored\_components)  
cv2.waitKey(**0**)

**Output:**



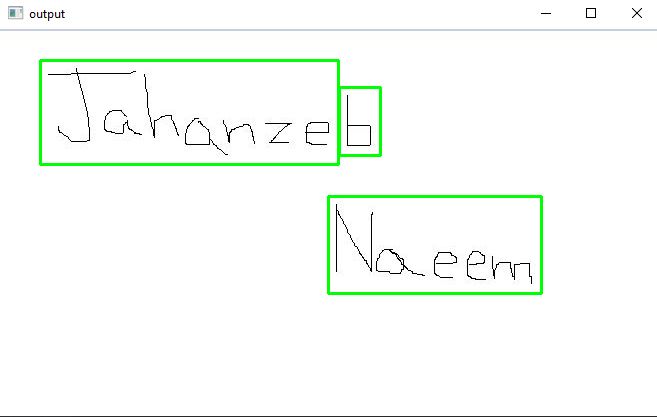
**Task 4**

Create an image through paint, snipping or even your mobile. The image should have a white background and your name written in it as foreground. Detect the text in the image and make a bounding box around it is using morphological operations.

**Code:**

import cv2  
# Read image from which text needs to be extracted  
img = cv2.imread("my\_image.JPG")  
  
# Convert the image to gray scale  
gray = cv2.cvtColor(img**,** cv2.COLOR\_BGR2GRAY)  
  
# Performing binarization through threshold  
ret**,** thresh1 = cv2.threshold(gray**, 0, 255,** cv2.THRESH\_OTSU | cv2.THRESH\_BINARY\_INV)  
  
# Specify structure shape and kernel size.  
# Kernel size increases or decreases the area  
# of the rectangle to be detected.  
# A smaller value like (10, 10) will detect  
# each word instead of a sentence.  
rect\_kernel = cv2.getStructuringElement(cv2.MORPH\_RECT**,** (**18, 18**))  
  
# Applying dilation on the threshold image  
dilation = cv2.dilate(thresh1**,** rect\_kernel**,** iterations=**1**)  
  
# Finding contours  
contours**,** hierarchy = cv2.findContours(dilation**,** cv2.RETR\_EXTERNAL**,** cv2.CHAIN\_APPROX\_NONE)  
  
# Creating a copy of image  
im2 = img.copy() # Why did we use the copy method instead of just assigning like im2 = img. Any idea?  
for cnt in contours:  
 x**,** y**,** w**,** h = cv2.boundingRect(cnt)  
  
 # Drawing a rectangle on copied image  
 rect = cv2.rectangle(im2**,** (x**,** y)**,** (x + w**,** y + h)**,** (**0, 255, 0**)**, 2**)  
  
 # Cropping the text block for giving input to OCR  
 cropped = im2[y:y + h**,** x:x + w]  
  
cv2.imshow('output'**,** im2)  
cv2.waitKey(**0**)

**output:**

****