National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Department of Computing

EE 433: Digital Image Processing

Amal Saqib (282496)

Muhammad Hadi (305774)

Class: BSCS 9C

Lab 10: Image Morphology

Date: 22nd November 2021

Time: 2.00Pm to 5.00Pm

Instructor: Dr. Imran Malik

National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Task 1

Note: Same kernel sizes were used for all morphological operations. All morphological operations were performed on the original image itself.

Erosion

Dilation

Binary Opening

Binary Closing

Analysis

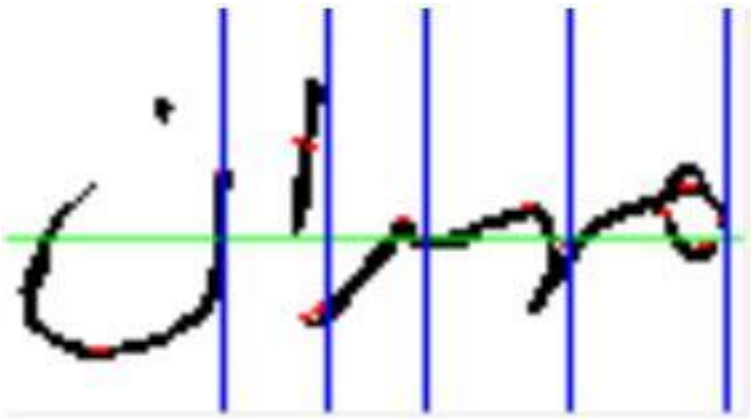
Binary Opening and Closing can be thought of as more refined uses of the more primitive erosion and dilation functions. The result of binary opening is much more suitable than the result of erosion, whilst the result of binary closing is much more suitable than the result of dilation.

Code

```python
# -*- coding: utf-8 -*-


import cv2 as cv
import numpy as np


def applyErosionAndDilation(binary_image,kernel_size):
    kernel=np.ones(shape=(kernel_size,kernel_size),dtype=np.uint8)
    eroded_image=cv.erode(binary_image,kernel,1)
    dilated_image=cv.dilate(binary_image,kernel,1)
    showImage("Eroded Image",eroded_image)
    showImage("Dilated Image",dilated_image)

def applyBinaryOpeningAndClosing(binary_image,kernel_size):
    kernel=np.ones(shape=(kernel_size,kernel_size),dtype=np.uint8)
    opened_image=cv.morphologyEx(binary_image,cv.MORPH_OPEN,kernel)
    closed_image=cv.morphologyEx(binary_image,cv.MORPH_CLOSE,kernel)
    showImage("Opened Image",opened_image)
    showImage("Closed Image",closed_image)

def convertImageToBinary(image):
    grayScale=cv.cvtColor(image,cv.COLOR_BGR2GRAY)
    _,binaryImage=cv.threshold(grayScale,127,255,cv.THRESH_BINARY)
    return binaryImage

def showImage(text,image):
    cv.namedWindow(text,cv.WINDOW_NORMAL)
    cv.imshow(text,image)
    cv.waitKey(0)

image=cv.imread('Picture5.png')
binaryImage=convertImageToBinary(image)
showImage("Original",binaryImage)
applyErosionAndDilation(binaryImage, 20)
applyBinaryOpeningAndClosing(binaryImage, 20)
```

Task 2

Original Image



Approach

Binarized the image to be able to perform morphological operations.

Dilated the image to remove the lines(Optimal kernel size obtained such that lines were completely removed whilst keeping the overall text shape recognizable).

Erode the dilated image with a kernel such that the text is easily readable.

Code

```python
# -*- coding: utf-8 -*-
"""
Created on Thu Nov 25 20:11:43 2021

@author: Hadi
"""

import cv2 as cv
import numpy as np

def convertImageToBinary(image):
    grayScale=cv.cvtColor(image,cv.COLOR_BGR2GRAY)
    _,binaryImage=cv.threshold(grayScale,127,255,cv.THRESH_BINARY)
    return binaryImage

def removeLines(image):
    dkernel=np.ones(shape=(4,4),dtype=np.uint8)
    dilated_image=cv.dilate(image,dkernel,1)     # Removing lines from image
    ekernel=np.ones(shape=(8,8),dtype=np.uint8)
    # Eroded kernel size is greater than dilated kernel size to rid of gaps in the text structure
    eroded_image=cv.erode(dilated_image,ekernel,1)  # Making the text a bit thicker, improving readability
    return eroded_image

image=cv.imread("Picture6.png")
binaryImage=convertImageToBinary(image)
result=removeLines(binaryImage)
cv.imshow("End result",result)
cv.waitKey(0)
```

Output

Difficulties

Some tweaking of the kernel size was required to obtain optimal results. Initially some holes were discovered, which were then dealt with by dilating with a bigger kernel. Overall,

the result obtained is satisfactory in my opinion, since no part of the text was lost(even the dot managed to survive the process).