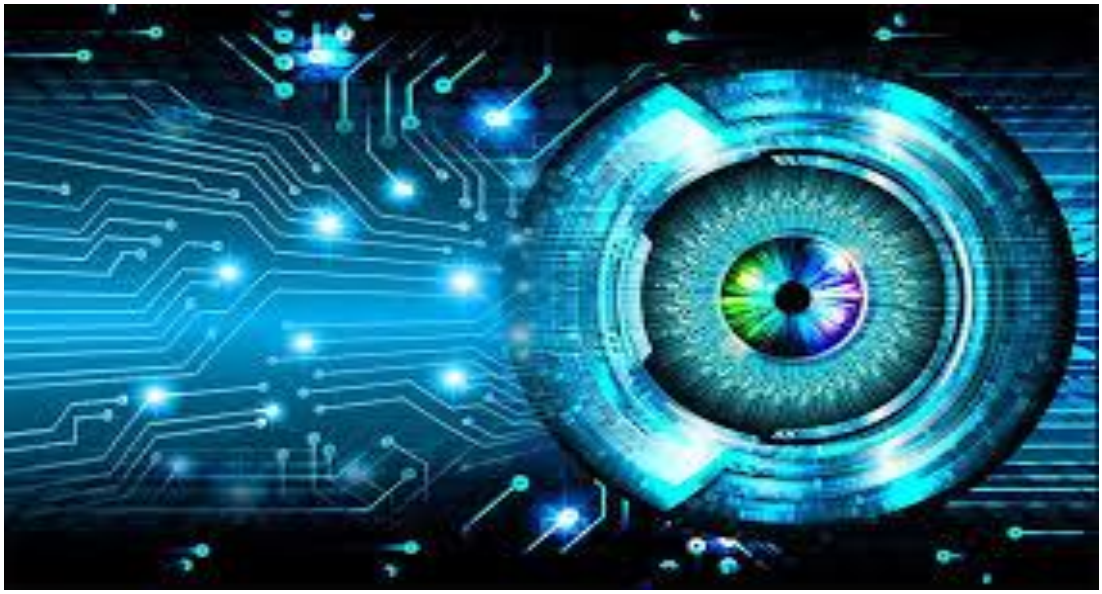
Computer Vision(CSE 365)

Project(OCR)



Name : Ahmed Essam Mohamed Ramzy

ID: 17p8229

Group : 2

Written Code with comments:

```
import pytesseract as ocr
import cv2 as cv
import imutils
import numpy as np
import re
height = 900
width  = 900

img1 = cv.imread('input3.png') # we read the image
#functions to be used in our algorithm
def printImages():
    #I resized every variable so that i can print it aside
    imgProcessedFit=cv.resize(imgProcessed,(380,380))
    contorsFit=cv.resize(contors,(380,380))
    ImagePerspectedFit=cv.resize(ImagePerspected,(380,380))
    imgWarpWarpFit=cv.resize(ImagePerspectedGrey,(380,380))
    cv.imshow('Canny Edges before Contouring',imgProcessedFit)
    cv.imshow('ImageContours',contorsFit)
    cv.imshow('The wrapped perspective',ImagePerspectedFit)
    cv.imshow('The gray image ',imgWarpWarpFit)
   # I rescale our input image to be of width 800
def imageRescaling(img):
    img = imutils.resize(img, width=width-100)
    return img
    # Here i have my input image being preprocessed to be ready for contoring phas
e by detecting all images using canny detector
def imagePreProcessing(img):
    grey = cv.cvtColor(img, cv.COLOR_BGR2GRAY) #My image is changed into
grey

    blur = cv.GaussianBlur(grey, (5,5), 0) # We smoothed the grey image by a 5*5 g
aussian Mask
    b=cv.resize(blur,(380,380))
    cv.imshow('Blureed image',b)
    edged = cv.Canny(blur, 75, 200) # Here we detected the edges using canny by gi
ving 2 threshold values

    return edged
```

```
    #Contor Manipulation function is used to get all the edged image contours and al
so the largest one
def ContourManipulation(imageProcessed):
    imgContours = img.copy() #We get a copy of the original image as find cotour f
unction rewrites on the image
    contours, hierarchy = cv.findContours(imageProcessed, cv.RETR_EXTERNAL,
 cv.CHAIN_APPROX_SIMPLE) #RETR_TREE Where we compute the relations
hip between contours and CHAIN_APPROX_SIMPLE.
    cv.drawContours(imgContours, contours, -
1, (0, 255, 0), 10) # I drawed all the contours on the imgContours variables
    contours=sorted(contours,key=cv.contourArea,reverse=True)[:5]# Then i sort th
em
    Largest=np.array([])
    for i in contours:
        perimeter = cv.arcLength(i, True)
        approximation = cv.approxPolyDP(i, 0.02 * perimeter, True)
        if len(approximation) == 4:
            Largest = approximation
            break
        else:
            return img1,imgContours
    return Largest,imgContours #return the largest contour and all contours image
    # This function is used to get the biggest contour points and reorderd them by a s
pecific manner Where:-
def reorderAndDraw(biggest):
    biggest = biggest.reshape((4, 2))
    PointsOrdered = np.zeros((4, 1, 2), dtype=np.int32)
    add = biggest.sum(1)
    PointsOrdered[0] = biggest[np.argmin(add)] #Here point 0 has the minimum su
m
    PointsOrdered[3] =biggest[np.argmax(add)] #Here point 3 has the minimum su
m
    diff = np.diff(biggest, axis=1)
    PointsOrdered[1] =biggest[np.argmin(diff)] #Here point 1 has the minimum diff
rence
    PointsOrdered[2] = biggest[np.argmax(diff)] #Here point 2 has the biggest diffre
nce
    return PointsOrdered
    # This function is used to get the largest contour and wrap into to be in readable
transform perspective
```

```
def ImagePerspective(largestPoints):
    if largestPoints.size != 0 and largestPoints.size==8 :
        largestPoints=reorderAndDraw(largestPoints)
        Matrix_1 = np.float32(largestPoints)
        Matrix_2 = np.float32([[0, 0],[width, 0], [0, height],[width, height]])
        matrix = cv.getPerspectiveTransform(Matrix_1, Matrix_2)
        ImagePerspected = cv.warpPerspective(img, matrix, (width, height))
        return ImagePerspected
    else:
        return largestPoints
# It is the last function that takes the processed image and change it to string wriite
n in the OCR.txt
def imgToString(img):
    data = ocr.image_to_string(ImagePerspectedGrey)
    with open('Ocr.txt', mode = 'w') as f:
        f.write(data)
        print("The number of charachters in this text file is ")
        print (len(re.sub(r"\W","",data))) # Here we used the regular expression modu
le to count the data string charachters after transforming from the image and print i
t to the terminal

# Here our algorithm starts

img=imageRescaling(img1) # I Rescaled the input image


imgProcessed=imagePreProcessing(img) #  pre-processed the input image

ProcessedCopy=imgProcessed.copy() # I took a copy from the pre-
processed to not affect the original image

biggest,contors=ContourManipulation(ProcessedCopy)#Here we recieve biggest c
ontor and all contours in the image

ImagePerspected=ImagePerspective(biggest) # Here we apply our easy prespective
 for human for the not well taken photos

ImagePerspectedGrey=  cv.cvtColor(ImagePerspected, cv.COLOR_BGR2GRAY)
#We transformed the colored image into gray-level
```

```
# These are postprocess functions for enhancement of the image depending on the quality of the image but it is not udes in the given samples
# ret,Thresh= cv.threshold(ImagePerspectedGrey,145,255,cv.THRESH_BINARY)
# ret,greyThresh = cv.threshold(grey, 100, 255, cv.THRESH_BINARY)
# blur=cv.medianBlur(greyThresh,3)

imgToString(ImagePerspectedGrey) #We send our transformed image to get written to the OCR.txt

printImages()# We printed the images of every phase

cv.waitKey(0)

cv.destroyAllWindows()
```

Instructions manual and Algorithm explanation:

I) Tunable parameters :

I) The 2 thresholded value of canny (75,200).
II) Gaussian smoothing filter of 5*5.
III) The images width and height that are predefined early in the code.

II) Algorithm explained

- After importing the needed libraries and modules and installation of pytesseract
- I read the image from projects folder.
- We give this image to the function imageRescaling to rescale the image with the given parameters.
- We then give the rescaled image to the imagePreProcessing function :-
    - To change coloured image to gray-level.
    - Blur the grayed image using gaussian 5*5 mask.
    - Then we called the canny edge detector function using the 2 chosen thresholded values.

- We created a copy from the preProcessed image and give it to the contourManipulation function which :-
    - Here we find all contours and draw them and then sort them.
    - And loop over the contours to get the largest.

    - And then return both the largest and all contours.
- After finding all contours and the largest we send the largest conoturs to imagePerspective function where:-
    - We called the reorderAndDraw function that orders our contours using the needed criteria and return the ordered points.
    - And draw the matrix needed to get the perspective of bird-view human readable document and return this transformed image.
- We then called the imgToString function of the transformed grey image and write the objects(characters drawn from the image into an external file called Ocr.text and prints in the terminal the length of the characters recognized.

III) Manual for OCR system

III.1) We installed the used packages on any IDE :-

- Pip install opencv-python
- PIP install numpy
- Pip install regex.
- Pip install pytesseract.
- After installing the tesseract.exe on windows and setup it we added the folder directory to environmental path.

III.2) To make the code works successfuly

- After importing and installed the above given libraries and modules
    - You need to create an OCR.txt file at the projects folder.
    - Clear it after any usage.
    - At line 9 replace the image you want inside imread function.

III.3) Instruction manual for the code step by step

As shown in the code we put all needed functions at the beginning of the file.

Here we read the image and change the parameter with any image the user wants to scan.

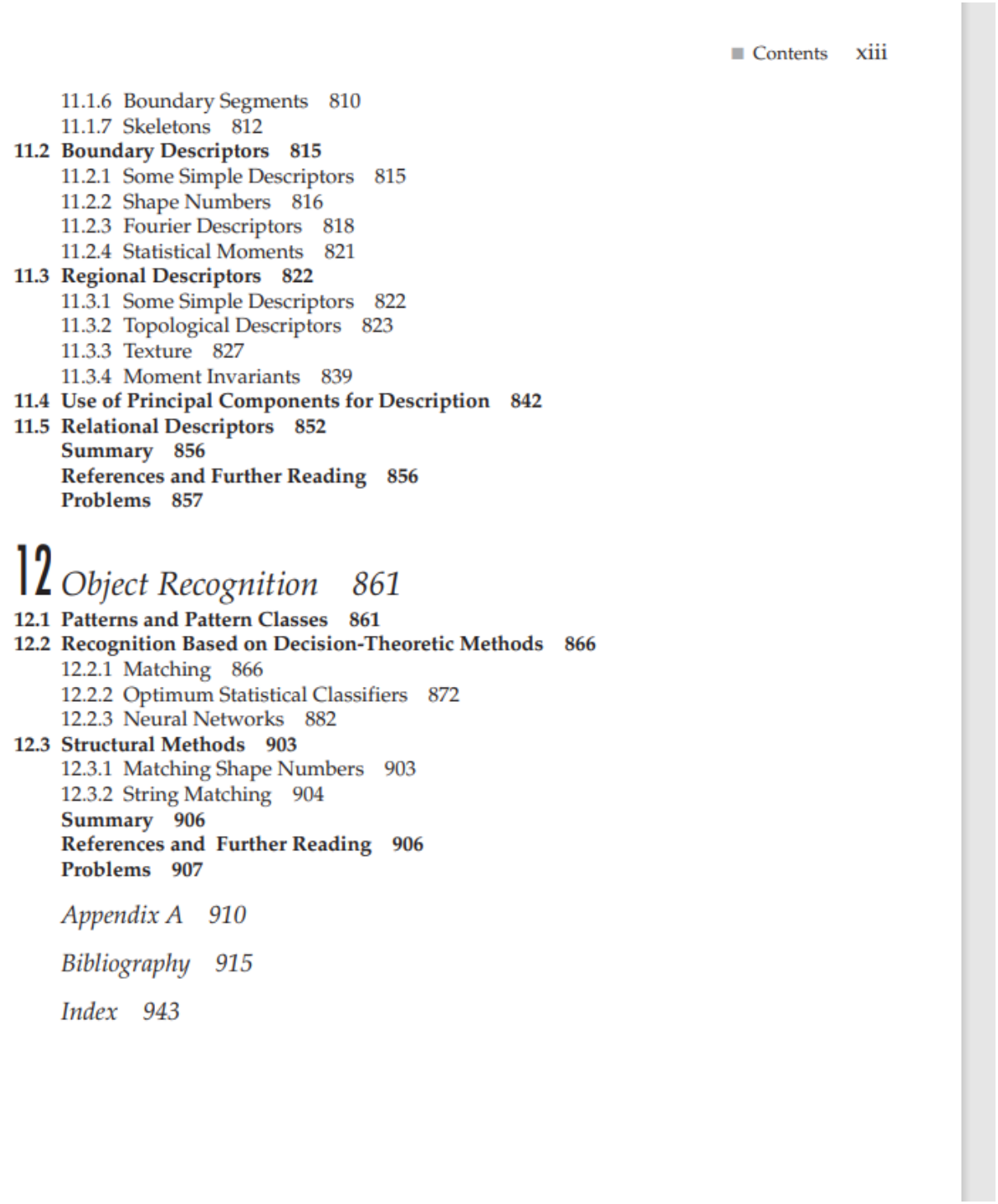
- Then we send the input image to the ImageRescaling where we scale the image to be of width 800 or any needed width depending on the image
    - Where we called the imutils imported library to have 2 parameters the image and the width.
- After that we get the image rescaled and put in the parameter of ImagePreProcessing function where we change the colored image to gray and smoothed the grey image by the 5*5 gaussian Mask.
- And then depending on the image we put the 2 suitable threshold values.
- Finally, I have my input image being preprocessed to be ready for contoring phase by detecting all images using canny detector.

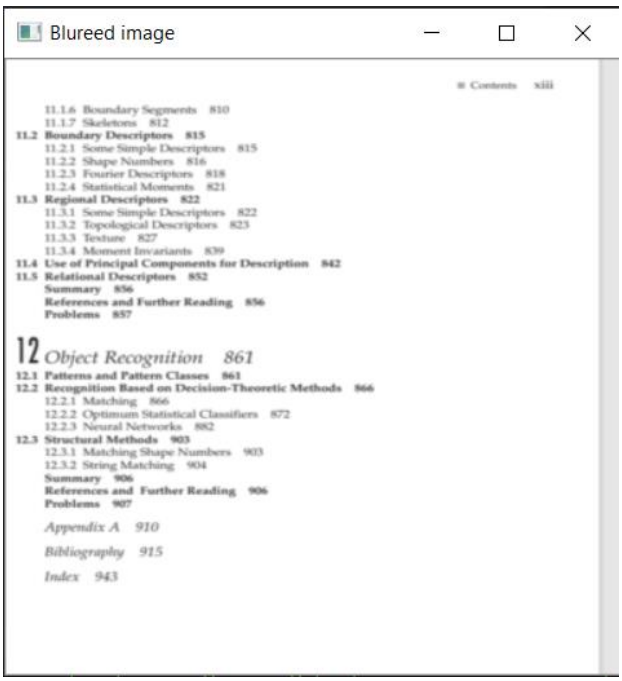
- We created a copy from processed image and put it in the contorManipulation function where we get all the edged image contours and also the largest one by :-
    - We get a copy of the original image as find cotour function rewrites on the image.
    - compute the relationship between contours and chain_approx and retr_external constants offered by opencv by the findContours function.
    - I drew all the contours on the imgContours variables and sort them with the cv.conoturArea and then iterate to find the largest of these contours.
    - After Executing this function we return the largest and all contours as explained in the images.
- After getting the largest contour we send it through the ImagePerspective function to call the reOrderAndDraw()
    - Where we ordered the four points and return them ordered.
    - And return to the ImagePerspective to form the matrix where we can extract the perspective of human-readable form and facilitates the scanning of the document and force the images to be of the same width and height for scalability of OCR system.
- We transformed the colored image into gray-level and depending on the input image we can do some thresholding or not to send the processed image and change it to string wriiten in the OCR.txt.

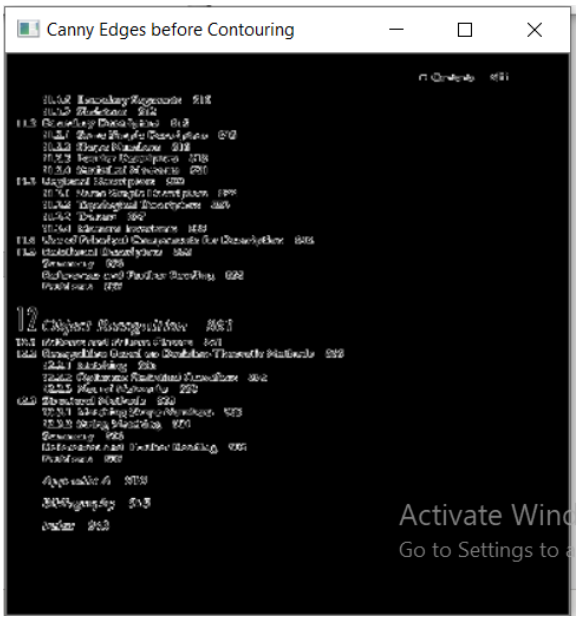
- By creating a txt file called OCR.txt.
- And then writing the transformed imageToString characters into the OCR.txt and printing on the terminal the number of characters

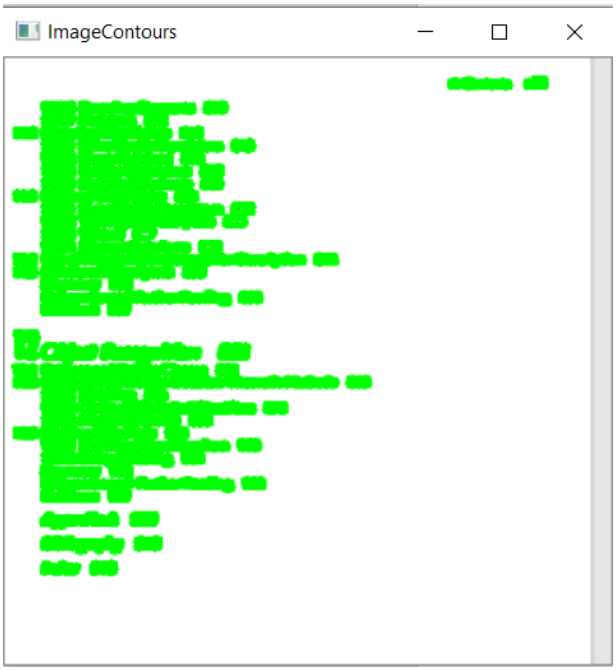
Chosen Input images:

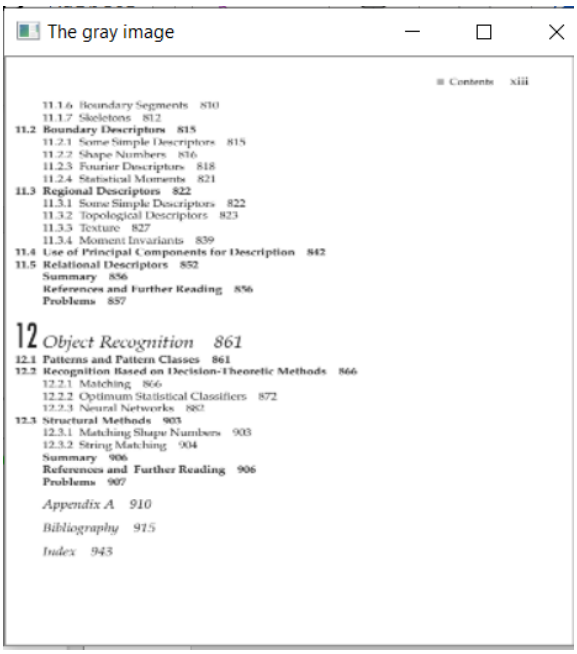
1) I choose an image from our reference contents which has a lot of words and numbers of different fonts and locations and my OCR system detects it.




Blureed image
Contents xiii
11.1.6 Boundary Segments 810
11.1.7 Skeletons 812
11.2 Boundary Descriptors 815
11.2.1 Some Simple Descriptors 815
11.2.2 Shape Numbers 816
11.2.3 Fourier Descriptors 818
11.2.4 Statistical Moments 821
11.3 Regional Descriptors 822
11.3.1 Some Simple Descriptors 822
11.3.2 Topological Descriptors 823
11.3.3 Texture 827
11.3.4 Moment Invariants 839
11.4 Use of Principal Components for Description 842
11.5 Relational Descriptors 852
Summary 856
References and Further Reading 856
Problems 857
12 Object Recognition 861
12.1 Patterns and Pattern Classes 861
12.2 Recognition Based on Decision-Theoretic Methods 866
12.2.1 Matching 866
12.2.2 Optimum Statistical Classifiers 872
12.2.3 Neural Networks 882
12.3 Structural Methods 903
12.3.1 Matching Shape Numbers 903
12.3.2 String Matching 904
Summary 906
References and Further Reading 906
Problems 907
Appendix A 910
Bibliography 915
Index 943


Canny Edges before Contouring
Activate Wind
Go to Settings to


ImageContours


The gray image
Contents xiii
11.1.6 Boundary Segments 810
11.1.7 Skeletons 812
11.2 Boundary Descriptors 815
11.2.1 Some Simple Descriptors 815
11.2.2 Shape Numbers 816
11.2.3 Fourier Descriptors 818
11.2.4 Statistical Moments 821
11.3 Regional Descriptors 822
11.3.1 Some Simple Descriptors 822
11.3.2 Topological Descriptors 823
11.3.3 Texture 827
11.3.4 Moment Invariants 839
11.4 Use of Principal Components for Description 842
11.5 Relational Descriptors 852
Summary 856
References and Further Reading 856
Problems 857
12 Object Recognition 861
12.1 Patterns and Pattern Classes 861
12.2 Recognition Based on Decision-Theoretic Methods 866
12.2.1 Matching 866
12.2.2 Optimum Statistical Classifiers 872
12.2.3 Neural Networks 882
12.3 Structural Methods 903
12.3.1 Matching Shape Numbers 903
12.3.2 String Matching 904
Summary 906
References and Further Reading 906
Problems 907
Appendix A 910
Bibliography 915
Index 943

Text file screenshot :

Ocr.txt - Notepad

File Edit Format View Help

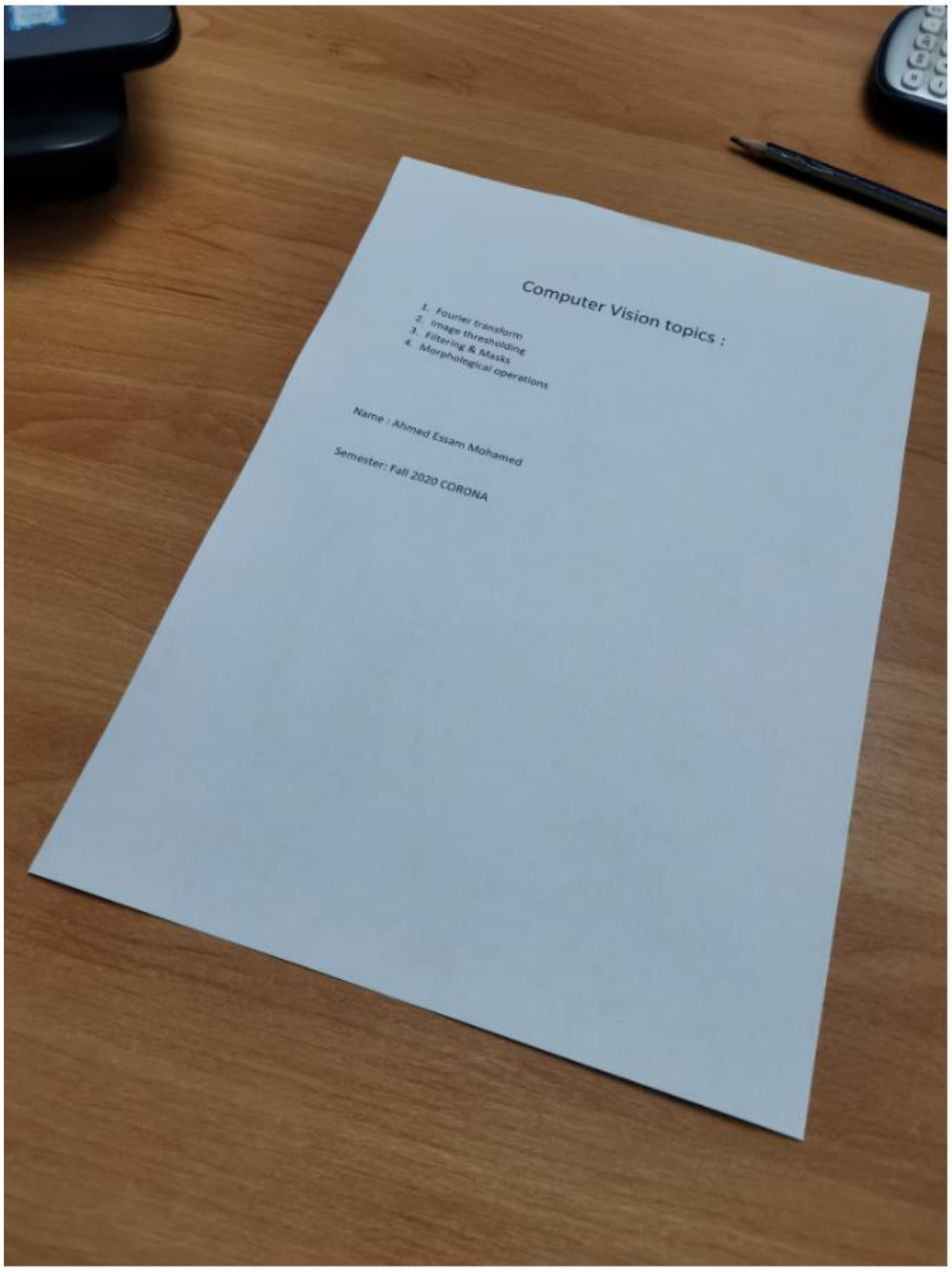
```
= Contents xiii

11.16 Boundary Segments 810
11.17 Skeletons 812
11.2 Boundary Descriptors 815
11.2.1 Some Simple Descriptors 815
11.2.2 Shape Numbers 816
11.23 Fourier Descriptors 818
11.2.4 Statistical Moments 821
11.3 Regional Descriptors 822
11.3.1 Some Simple Descriptors 822
11.3.2 Topological Descriptors 823
11.3.3 Texture 827
11.3.4 Moment Invariants 839
11.4 Use of Principal Components for Description 842
115 Relational Descriptors 852
Summary 856
References and Further Reading 856
Problems 857

l 2 Object Recognition 861

12.1 Patterns and Pattern Classes 861

12.2 Recognition Based on Decision-Theoretic Methods 866
12.2.1 Matching 866
12.2.2 Optimum Statistical Classifiers 872
12.2.3 Neural Networks 882

12.3 Structural Methods 903
12.3.1 Matching Shape Numbers 903
12.3.2 String Matching 904
Summary 906
References and Further Reading 906
Problems 907

Appendix A 910
Bibliography 915
Index 943
```

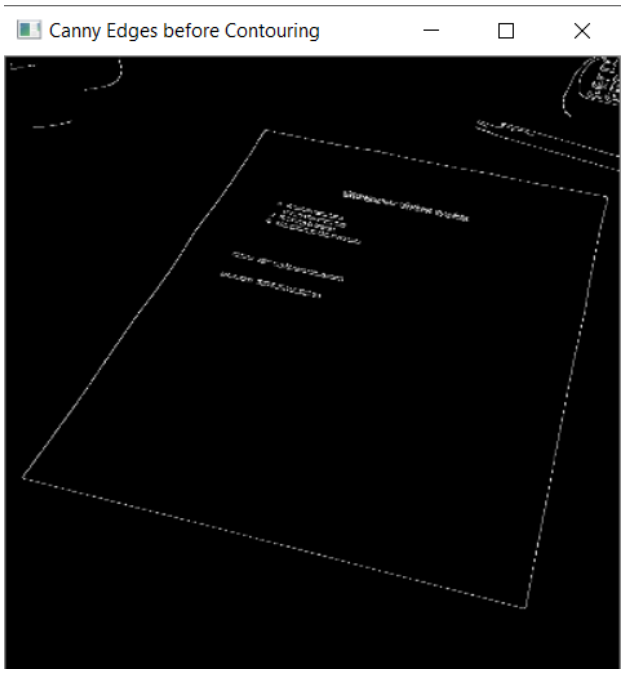
Ln 11, Col 35 | 100% | Windows (CRLF) | UTF-8

```
PS E:\3rd Computer\Computer Vision\OCR> & C:/Users/aeram/AppData/Local/Programs/Python/Python38/python.exe "e:/3rd Computer/Computer Vision/OCR/OCRproject.py"
The number of charachters in this text file is
744
PS E:\3rd Computer\Computer Vision\OCR>
```


Number of characters

2) I made an image and print it and picture it using a normal camera phone which is considered a challenging photo as there is different objects appear(calculator, pencil, edge of laptop) and the document I want to scan is perspected and not fit to process but my OCR system detects it.





Canny Edges before contouring

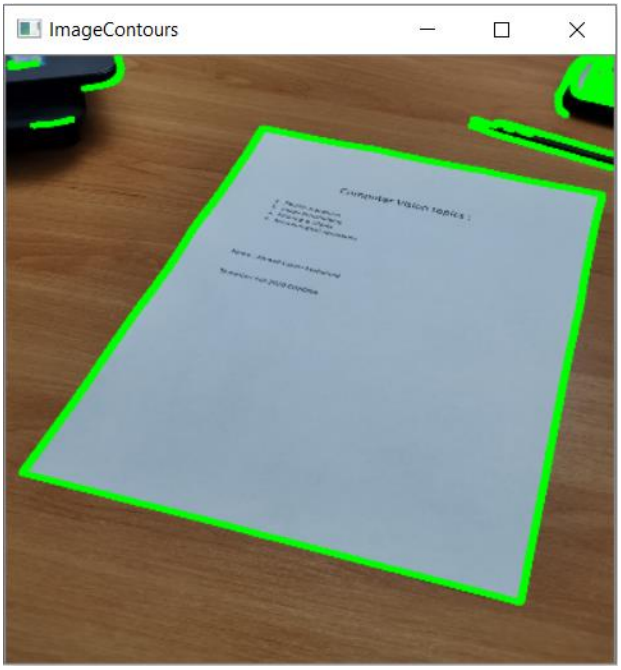


Image Contours detected

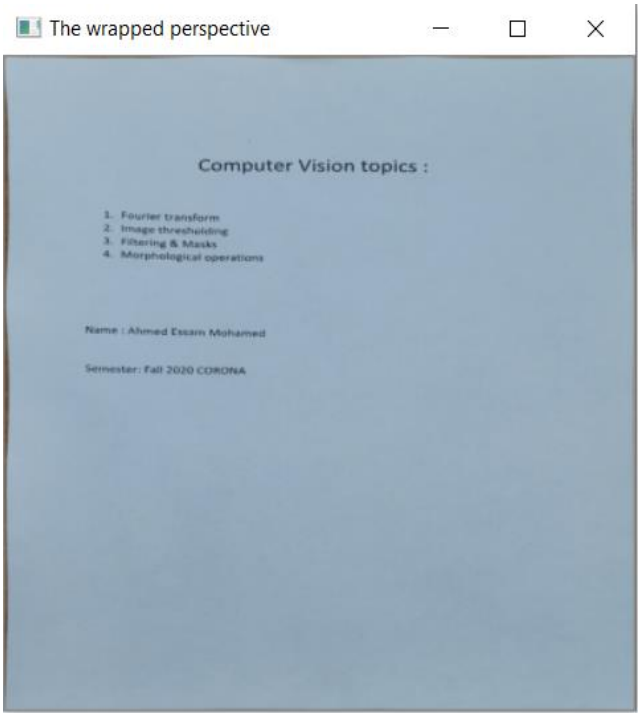
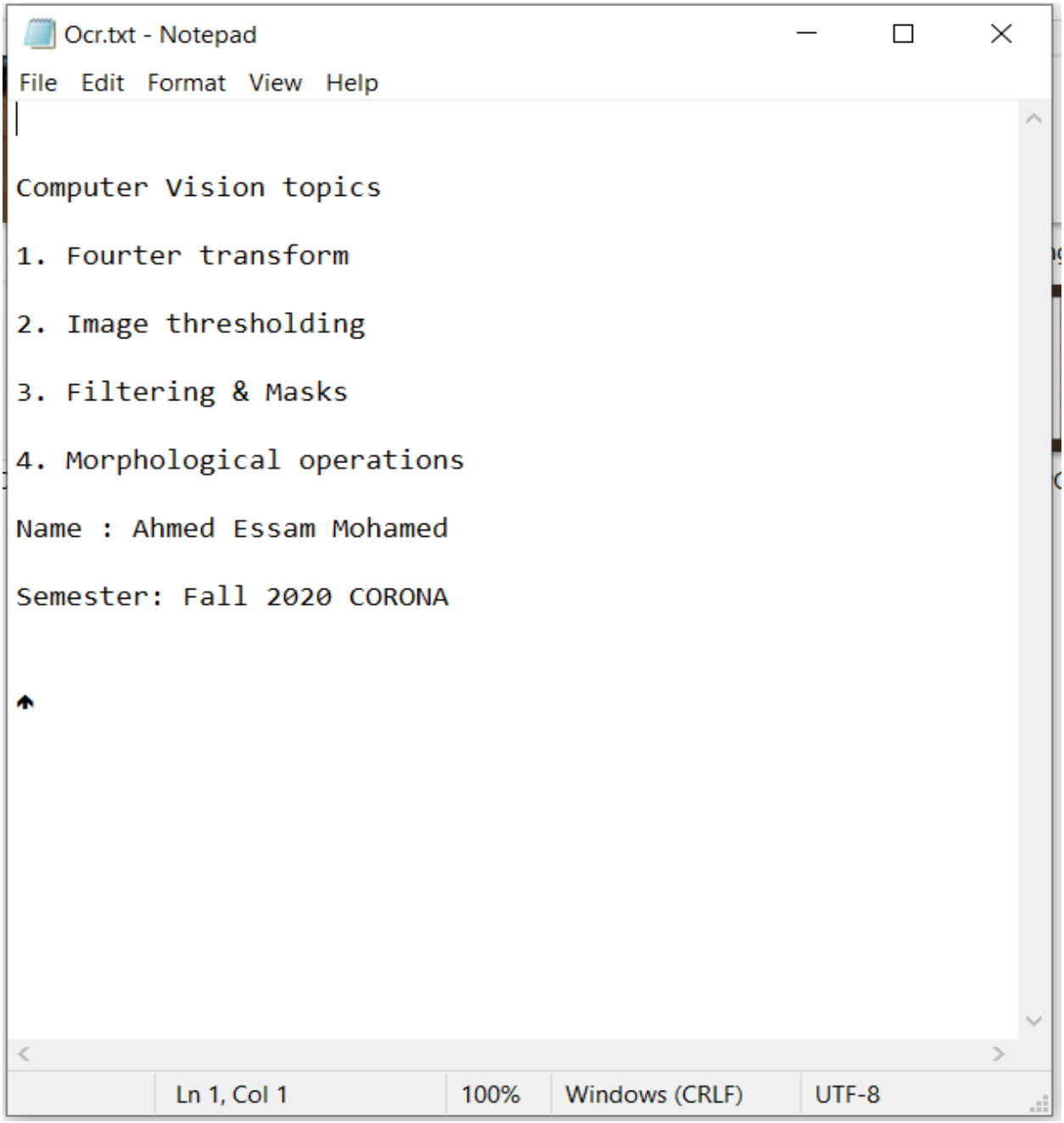


Image get perspected

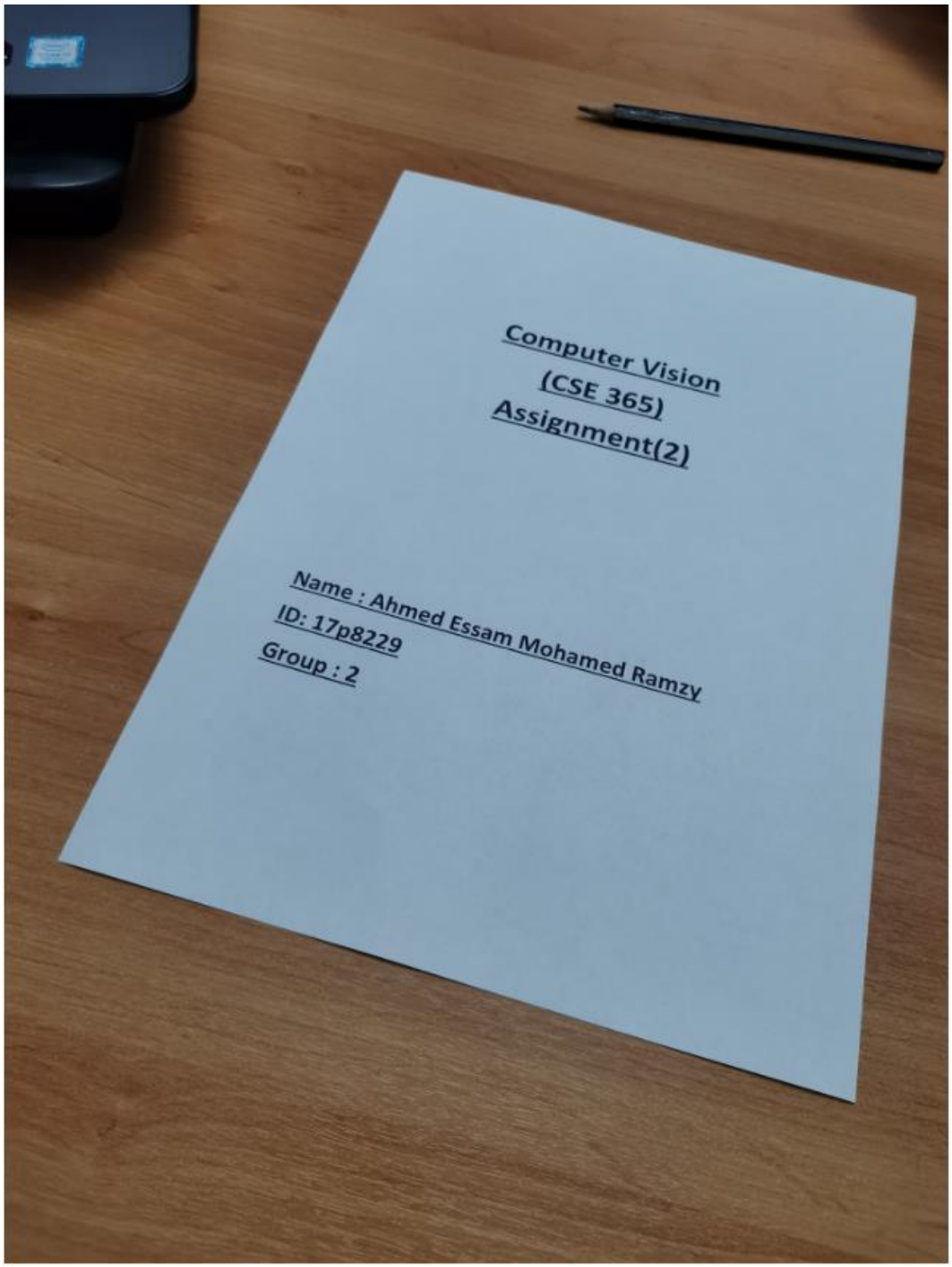
The perspected gray image

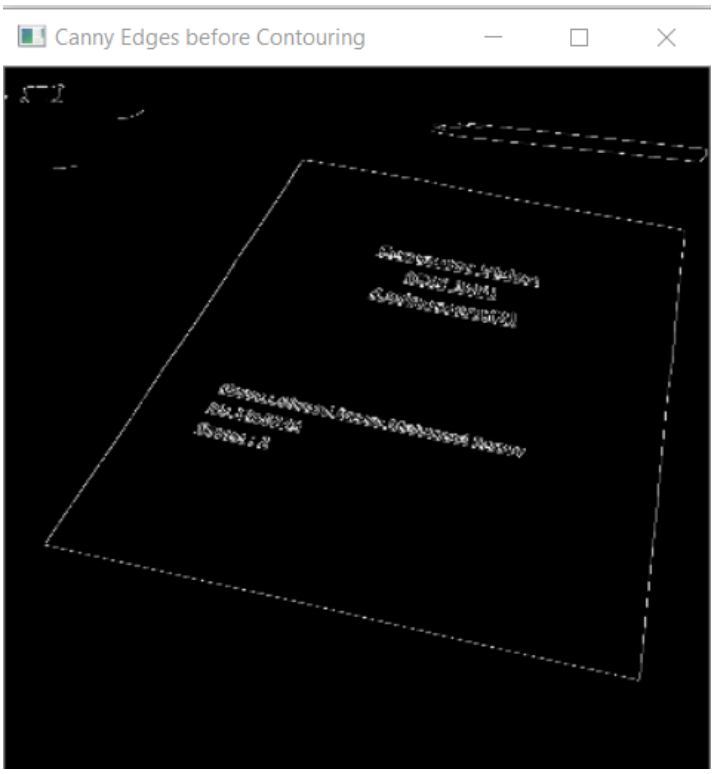
Text file screenshot :

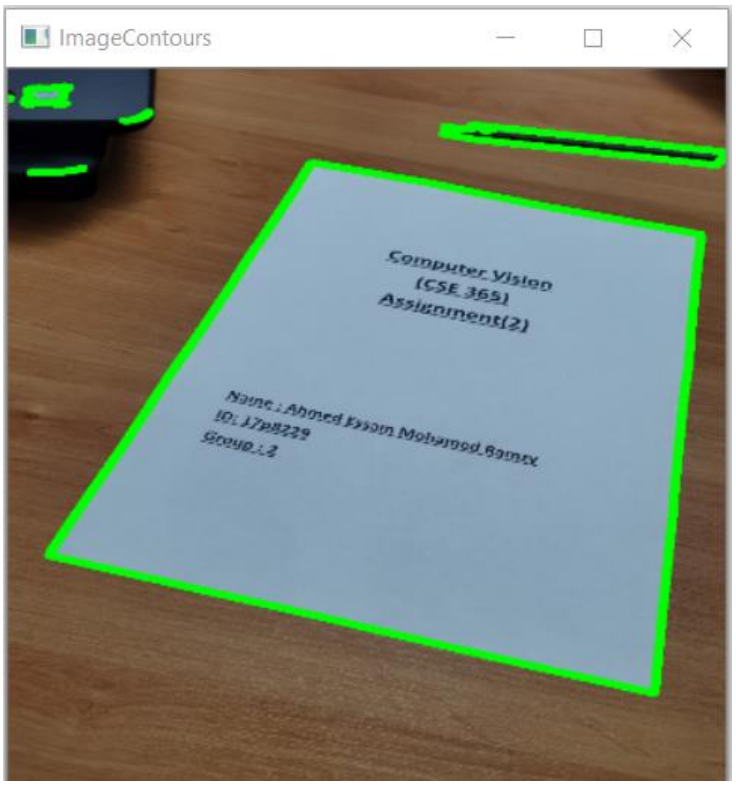


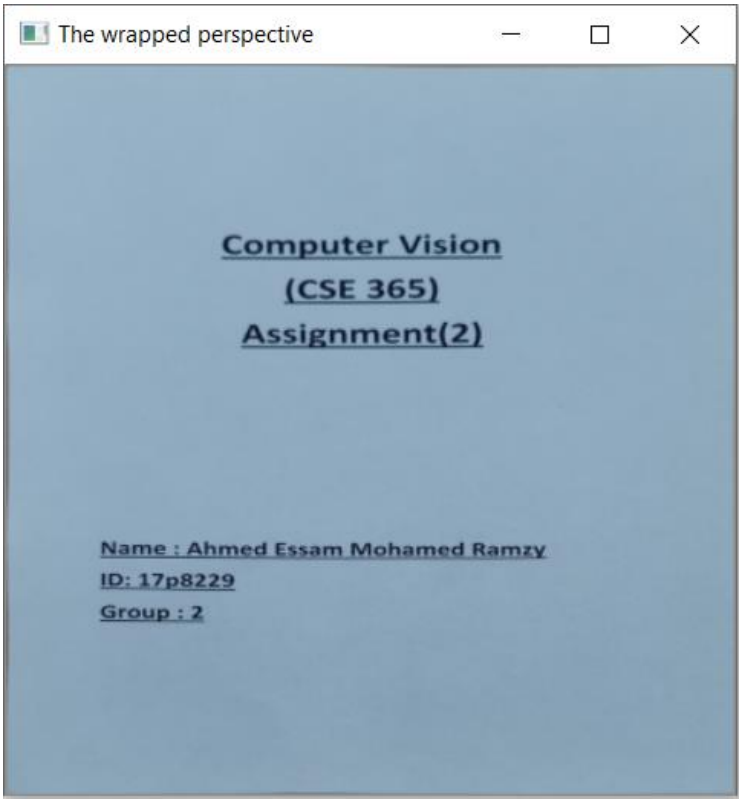

```
PS E:\3rd Computer\Computer Vision\OCR>
PS E:\3rd Computer\Computer Vision\OCR> & C:/Users/aeram/AppData/Local/Programs/Python/Python38/python.exe "e:/3rd Computer/Computer Vision/OCR/OCRproject.py"
The number of charachters in this text file is
137
```

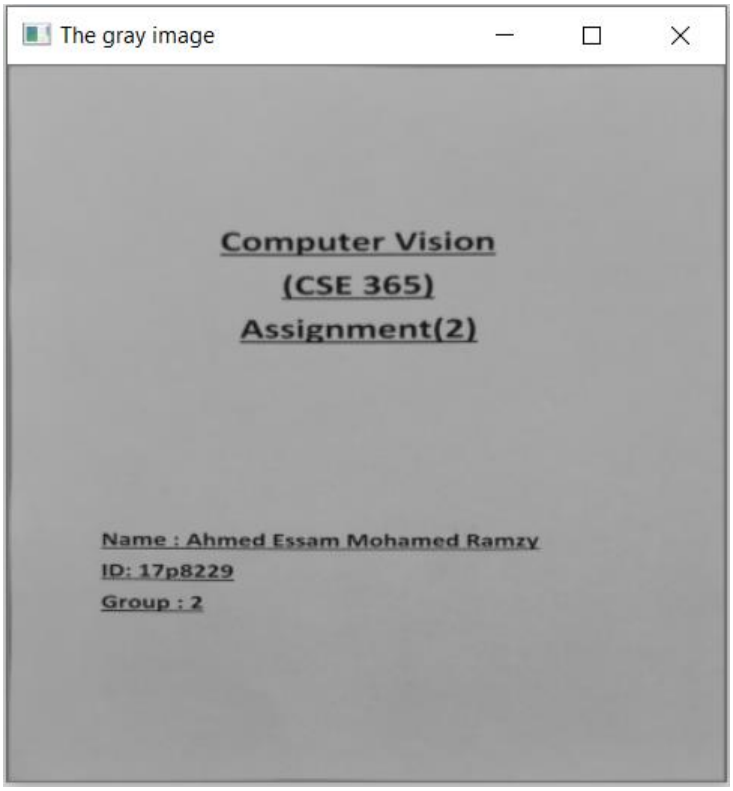
3) I made another challenging image and print it and picture it using a normal camera phone as there is pencil object and edge of laptop and the document I want to scan is perspected and not fit to process but my OCR system detects it.



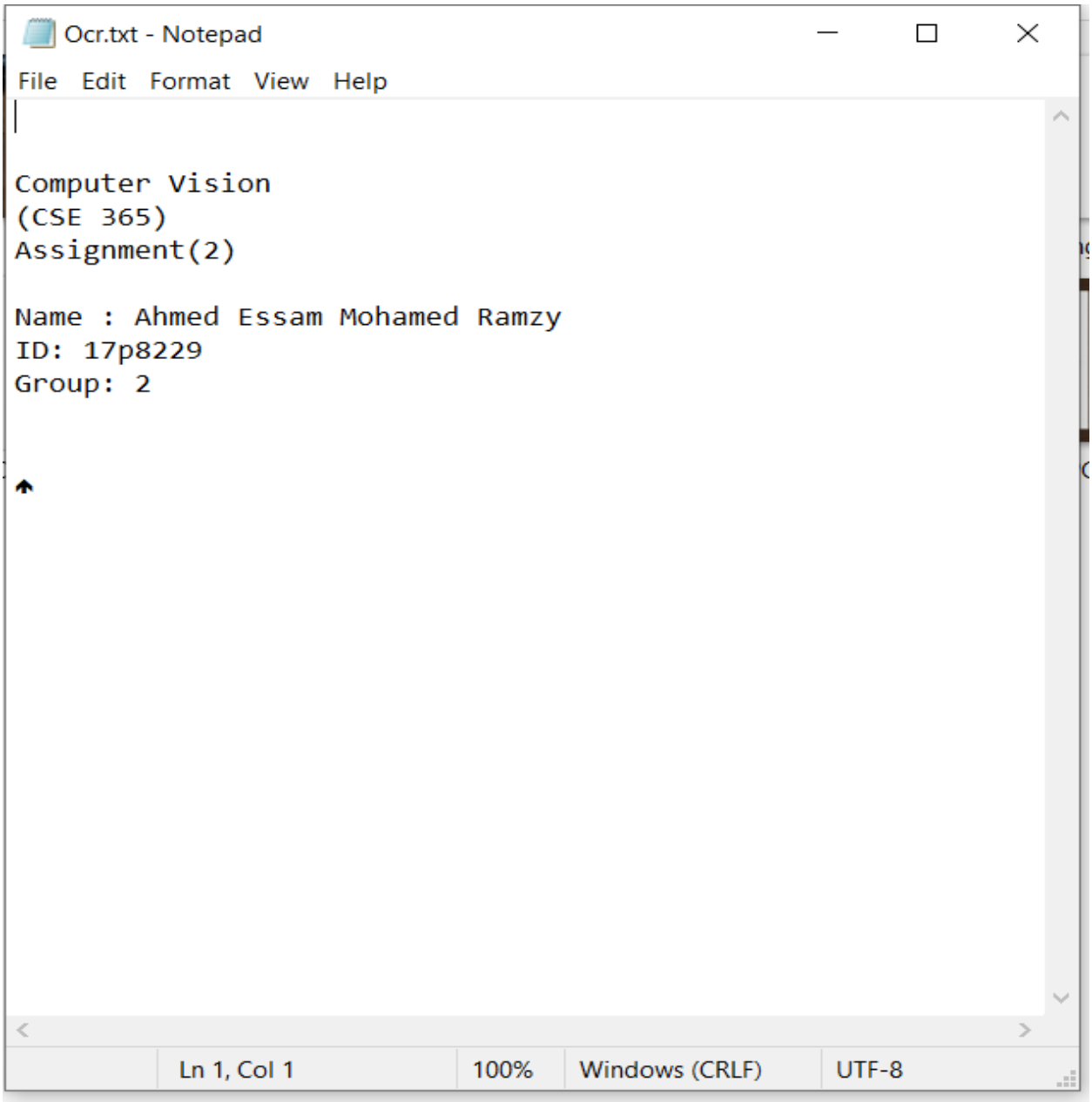

Canny Edges before Contouring


ImageContours
Computer Vision
(CSE 365)
Assignment(2)


The wrapped perspective
Computer Vision
(CSE 365)
Assignment(2)
Name : Ahmed Essam Mohamed Ramzy
ID: 17p8229
Group : 2


The gray image
Computer Vision
(CSE 365)
Assignment(2)
Name : Ahmed Essam Mohamed Ramzy
ID: 17p8229
Group : 2

Text file screenshot :




```
PS E:\3rd Computer\Computer Vision\OCR> & C:/Users/aeram/AppData/Local/Programs/Python/Python38/python.exe "e:/3rd Computer/Computer Vision/OCR/OCRproject.py"
The number of charachters in this text file is
72
```

4) I choose an image from our reference with huge number of characters and my OCR detects it.

Acknowledgments

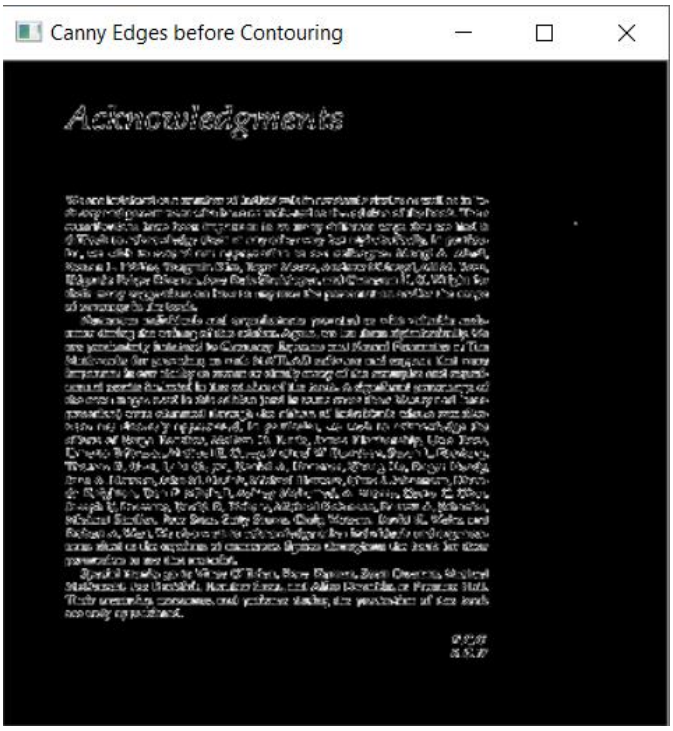
We are indebted to a number of individuals in academic circles as well as in industry and government who have contributed to this edition of the book. Their contributions have been important in so many different ways that we find it difficult to acknowledge them in any other way but alphabetically. In particular, we wish to extend our appreciation to our colleagues Mongi A. Abidi, Steven L. Eddins, Yongmin Kim, Bryan Morse, Andrew Oldroyd, Ali M. Reza, Edgardo Felipe Riveron, Jose Ruiz Shulcloper, and Cameron H. G. Wright for their many suggestions on how to improve the presentation and/or the scope of coverage in the book.

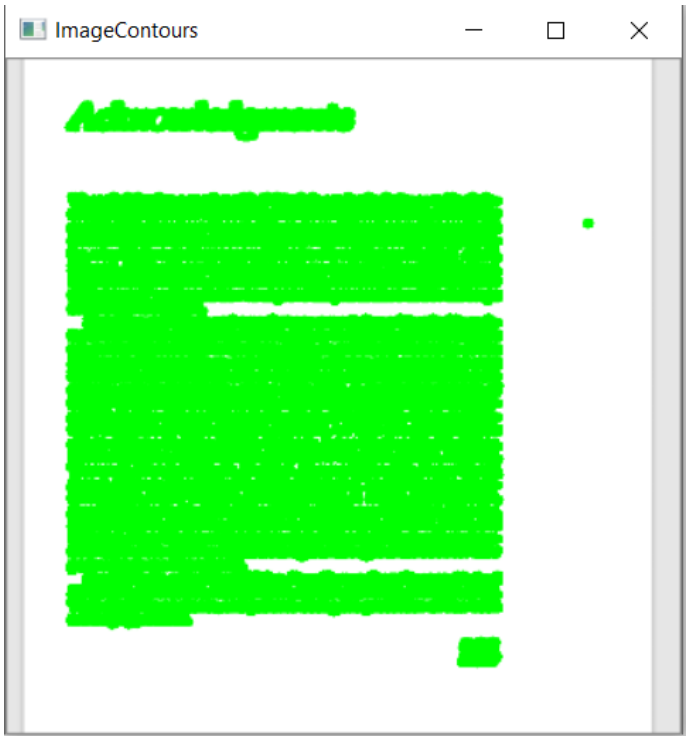
Numerous individuals and organizations provided us with valuable assistance during the writing of this edition. Again, we list them alphabetically. We are particularly indebted to Courtney Esposito and Naomi Fernandes at The Mathworks for providing us with MATLAB software and support that were important in our ability to create or clarify many of the examples and experimental results included in this edition of the book. A significant percentage of the new images used in this edition (and in some cases their history and interpretation) were obtained through the efforts of individuals whose contributions are sincerely appreciated. In particular, we wish to acknowledge the efforts of Serge Beucher, Melissa D. Binde, James Blankenship, Uwe Boos, Ernesto Bribiesca, Michael E. Casey, Michael W. Davidson, Susan L. Forsburg, Thomas R. Gest, Lalit Gupta, Daniel A. Hammer, Zhong He, Roger Heady, Juan A. Herrera, John M. Hudak, Michael Hurwitz, Chris J. Johannsen, Rhonda Knighton, Don P. Mitchell, Ashley Mohamed, A. Morris, Curtis C. Ober, Joseph E. Pascente, David. R. Pickens, Michael Robinson, Barrett A. Schaefer, Michael Shaffer, Pete Sites, Sally Stowe, Craig Watson, David K. Wehe, and Robert A. West. We also wish to acknowledge other individuals and organizations cited in the captions of numerous figures throughout the book for their permission to use that material.

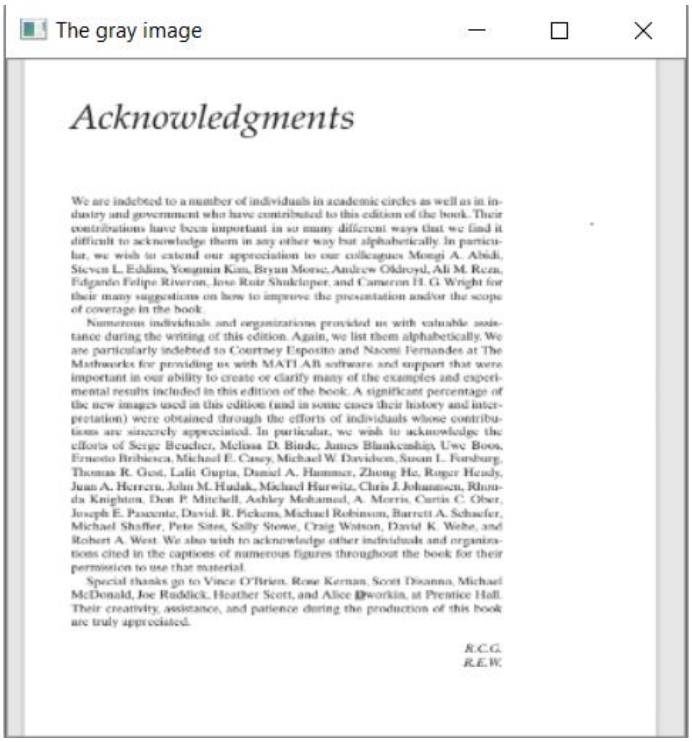
Special thanks go to Vince O'Brien, Rose Kernan, Scott Disanno, Michael McDonald, Joe Ruddick, Heather Scott, and Alice Dworkin, at Prentice Hall. Their creativity, assistance, and patience during the production of this book are truly appreciated.

R.C.G.
R.E.W.

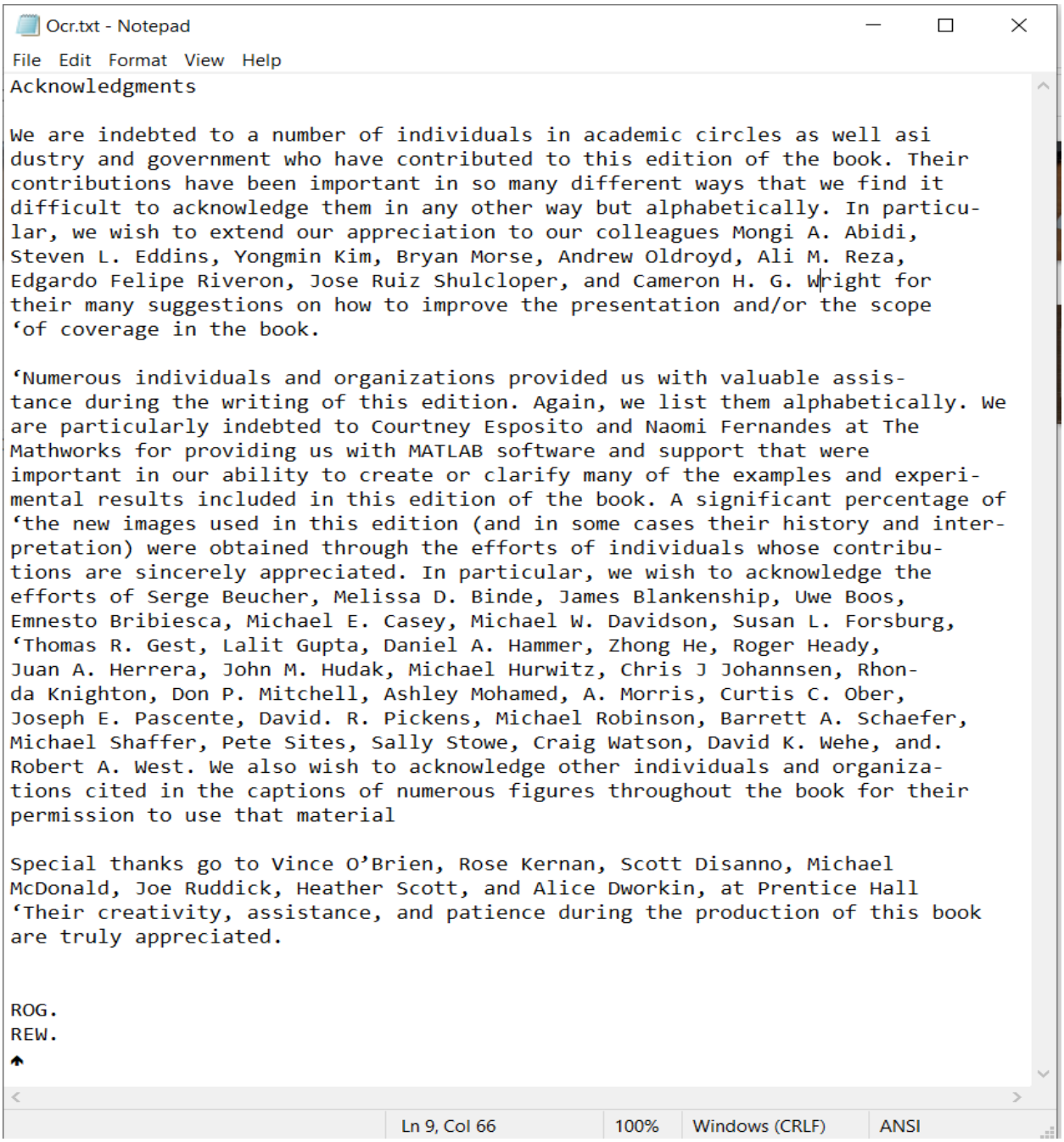
Here as the sample is already perspected we don’t need to draw the wrap perspective and don’t need the ImagePerspective function.





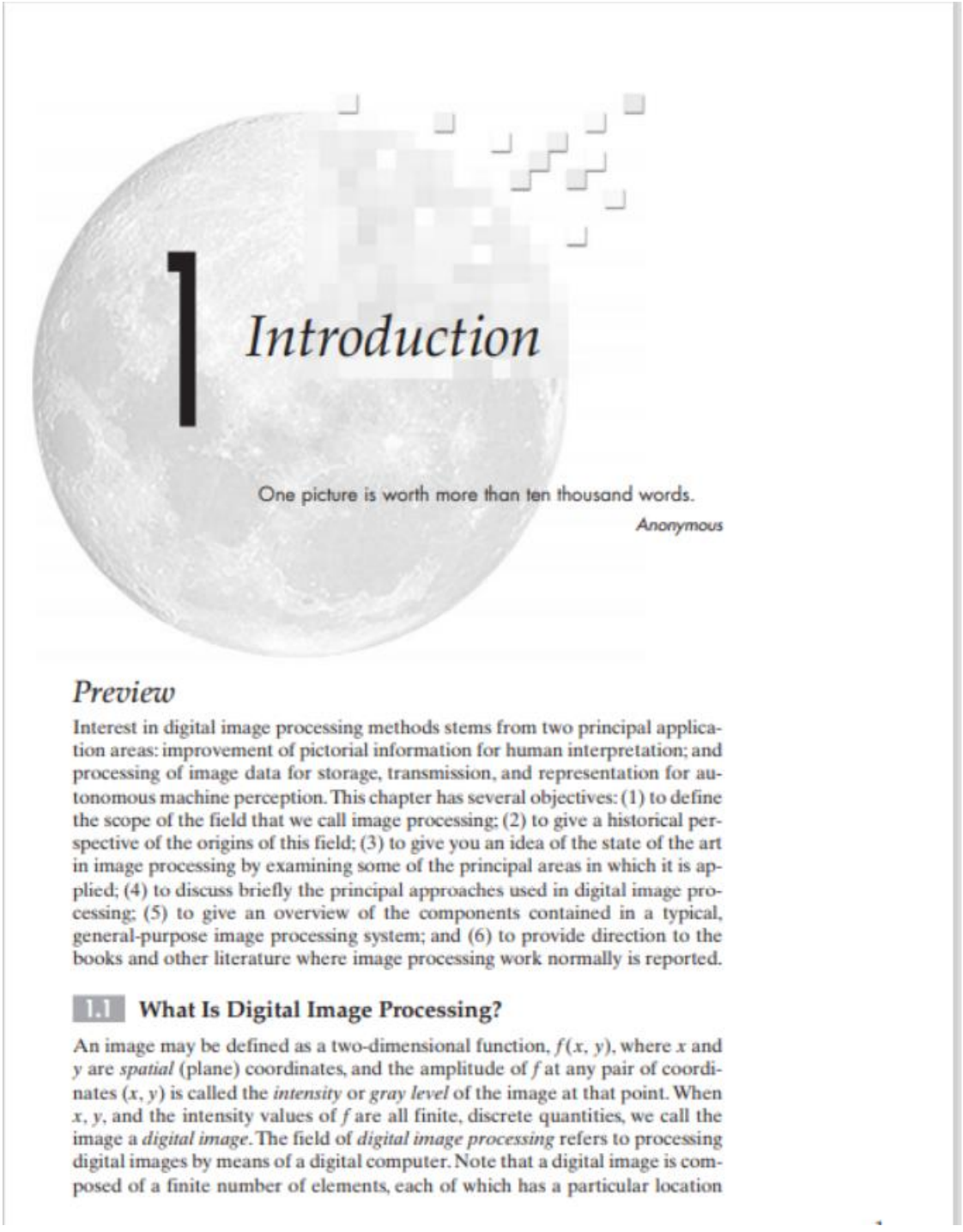


Text file screenshot :




```
PS E:\3rd Computer\Computer Vision\OCR> & C:/Users/aeram/AppData/Local/Programs/Python/Python38/python.exe "e:/3rd Computer/Computer Vision/OCR/OCRproject.py"
The number of charachters in this text file is
1839
PS E:\3rd Computer\Computer Vision\OCR>
```

5) I choose a challenging image from our reference with huge number of characters and a lot of objects and shapes where there is a text inside a big circle and my OCR detects it.



1 Introduction

One picture is worth more than ten thousand words.

Anonymous

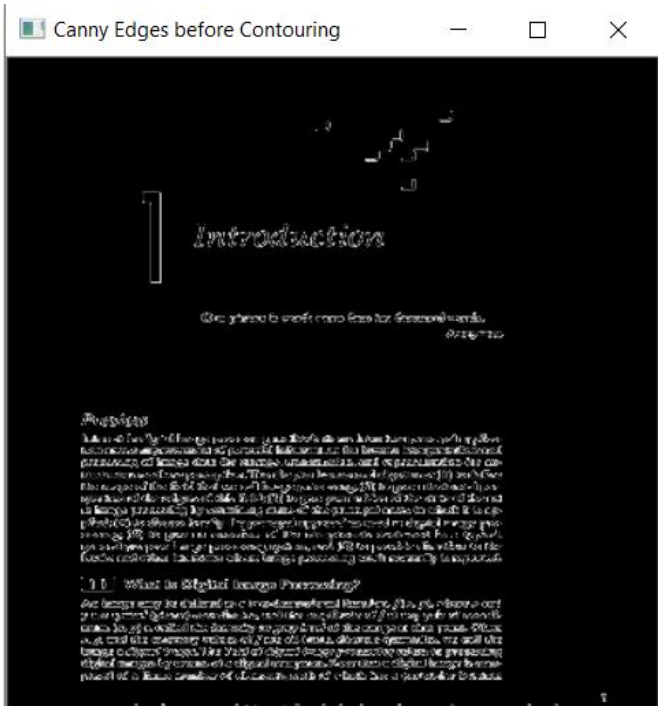
Preview

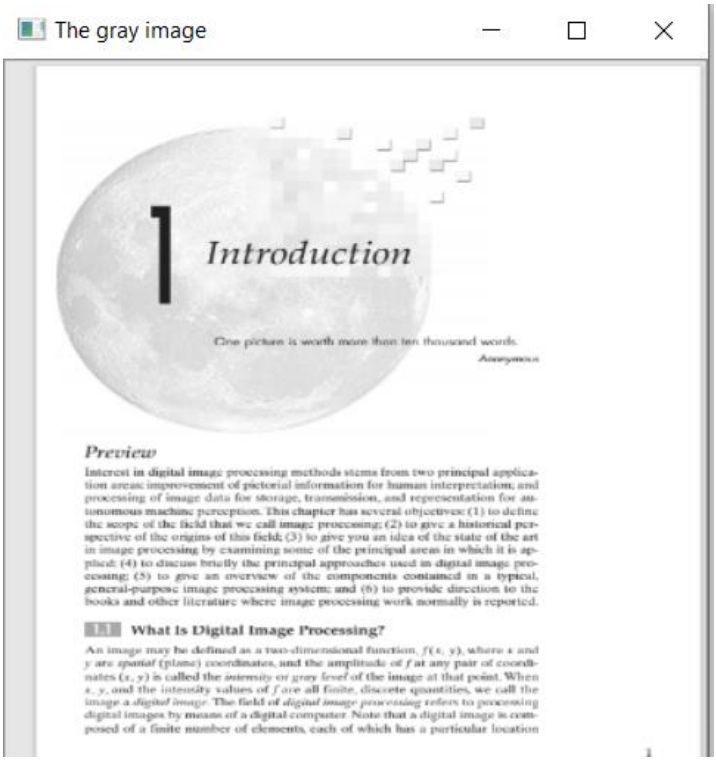
Interest in digital image processing methods stems from two principal application areas: improvement of pictorial information for human interpretation; and processing of image data for storage, transmission, and representation for autonomous machine perception. This chapter has several objectives: (1) to define the scope of the field that we call image processing; (2) to give a historical perspective of the origins of this field; (3) to give you an idea of the state of the art in image processing by examining some of the principal areas in which it is applied; (4) to discuss briefly the principal approaches used in digital image processing; (5) to give an overview of the components contained in a typical, general-purpose image processing system; and (6) to provide direction to the books and other literature where image processing work normally is reported.

1.1 What Is Digital Image Processing?

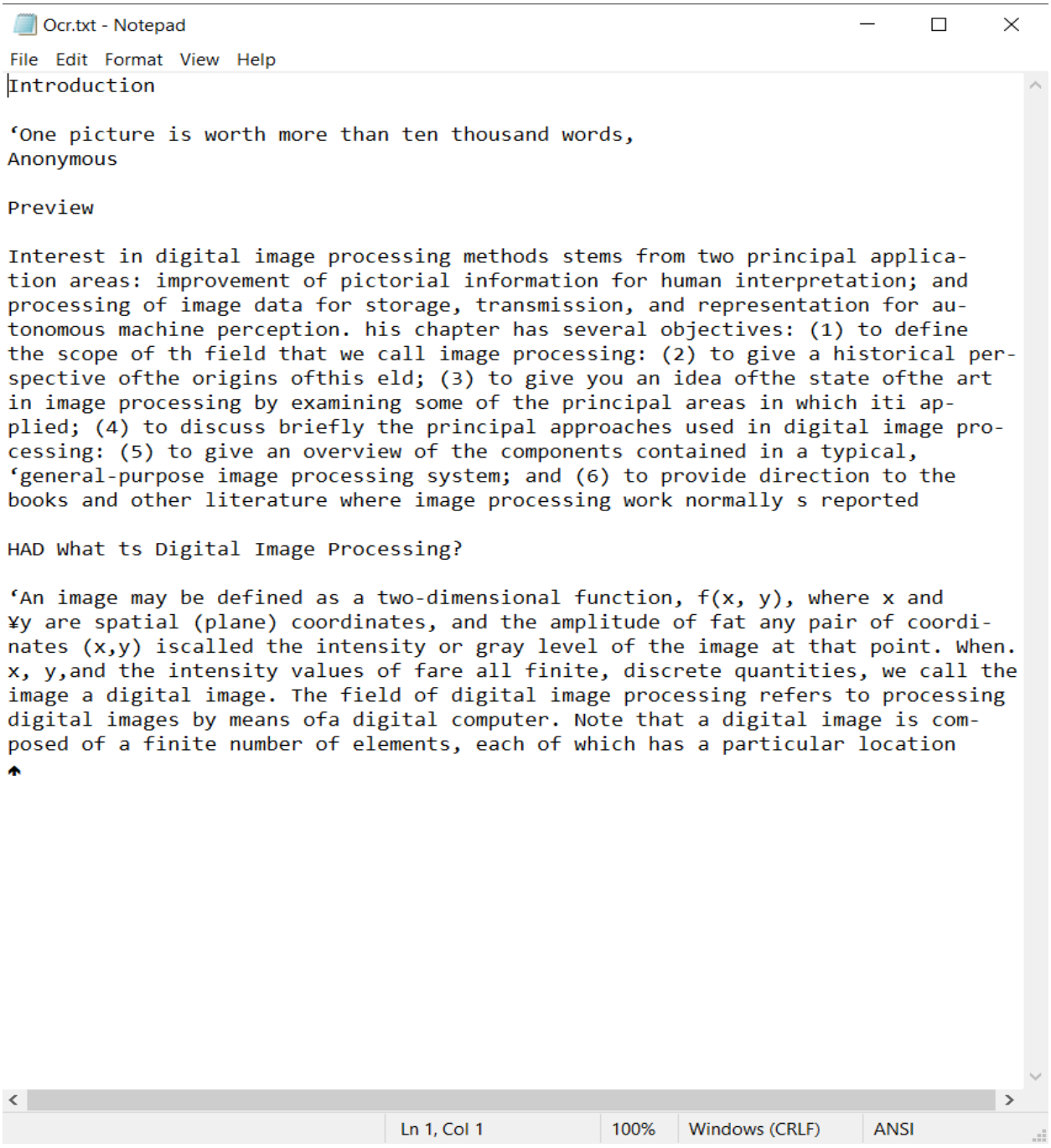
An image may be defined as a two-dimensional function, $f(x, y)$, where x and y are *spatial* (plane) coordinates, and the amplitude of f at any pair of coordinates (x, y) is called the *intensity* or *gray level* of the image at that point. When x, y, and the intensity values of f are all finite, discrete quantities, we call the image a *digital image*. The field of *digital image processing* refers to processing digital images by means of a digital computer. Note that a digital image is composed of a finite number of elements, each of which has a particular location

Here as the sample is already perspected we don’t need to draw the wrap perspective and don’t need the ImagePerspective function.



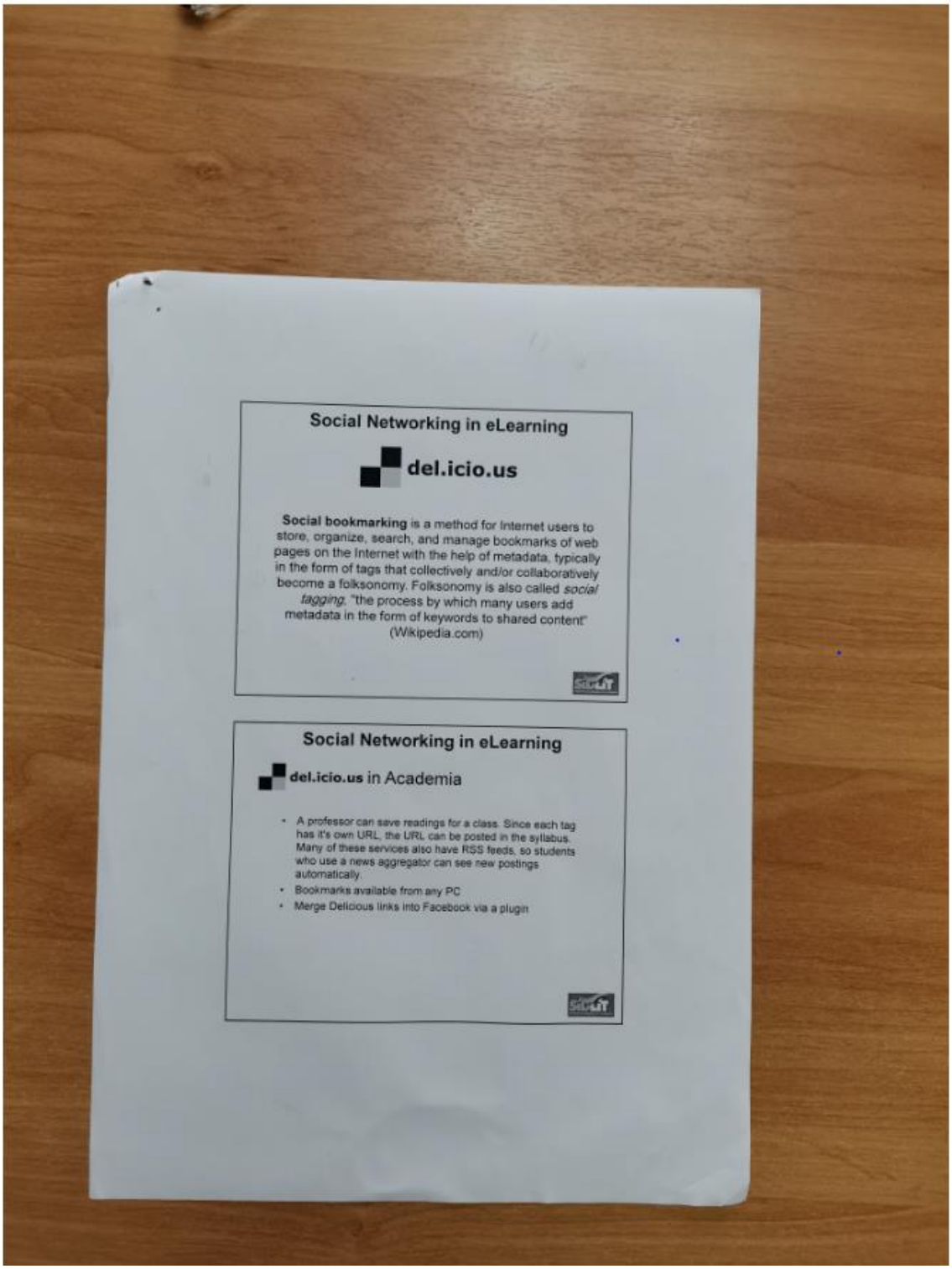


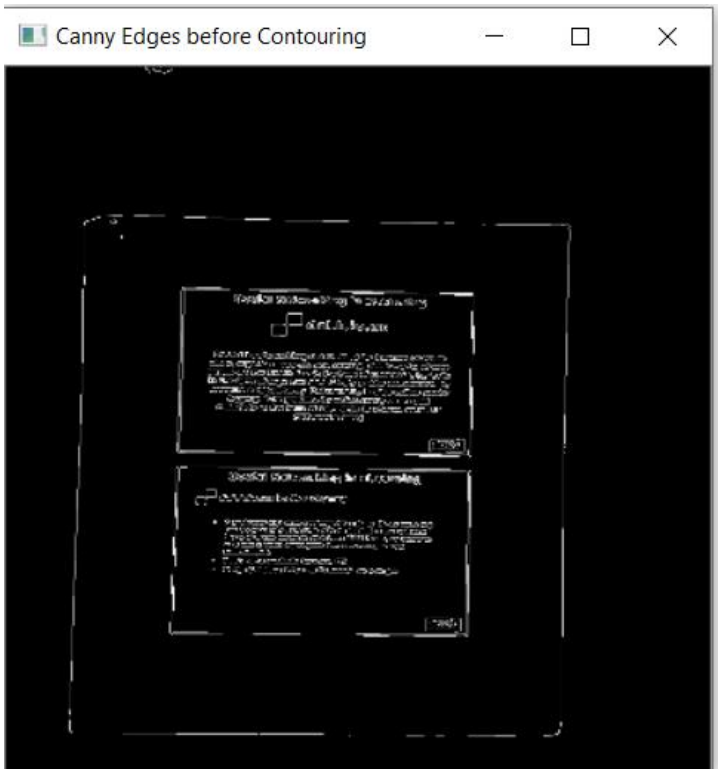
Text file screenshot :

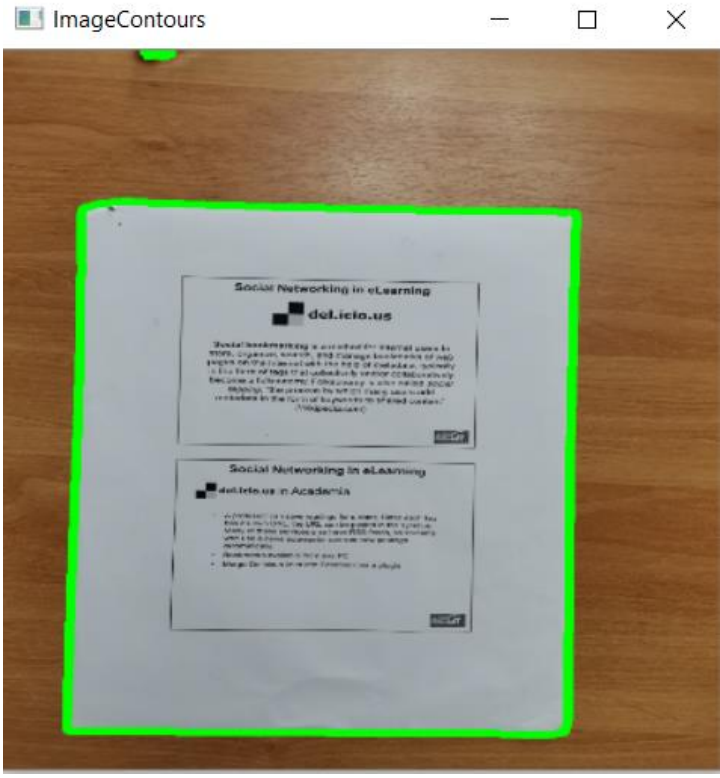


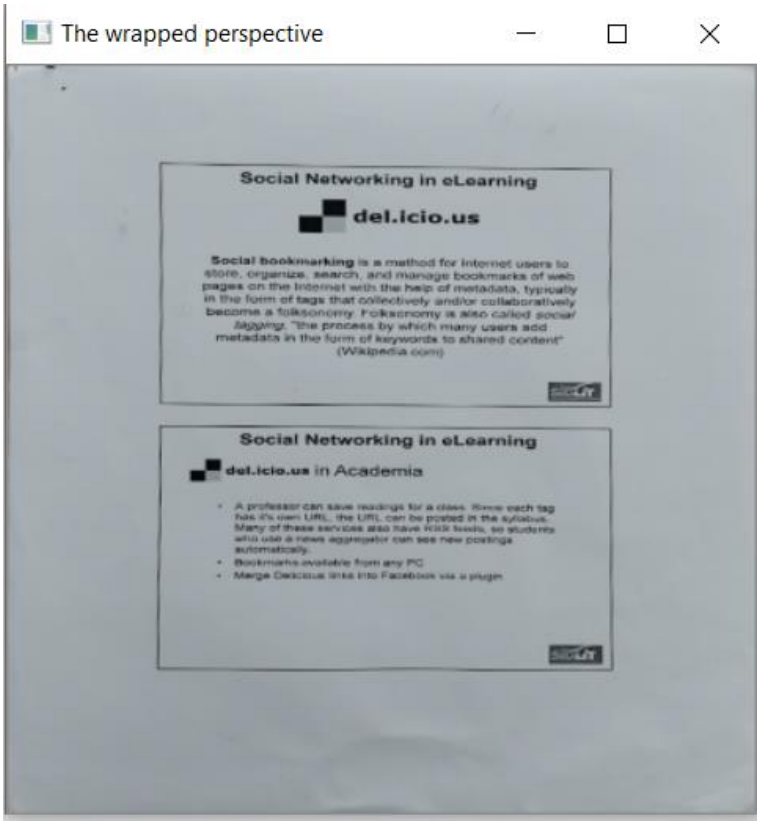

```
PS E:\3rd Computer\Computer Vision\OCR> & C:/Users/aeram/AppData/Local/Programs/Python/Python38/python.exe "e:/3r
d Computer/Computer Vision/OCR/OCRproject.py"
The number of charachters in this text file is
1242
```

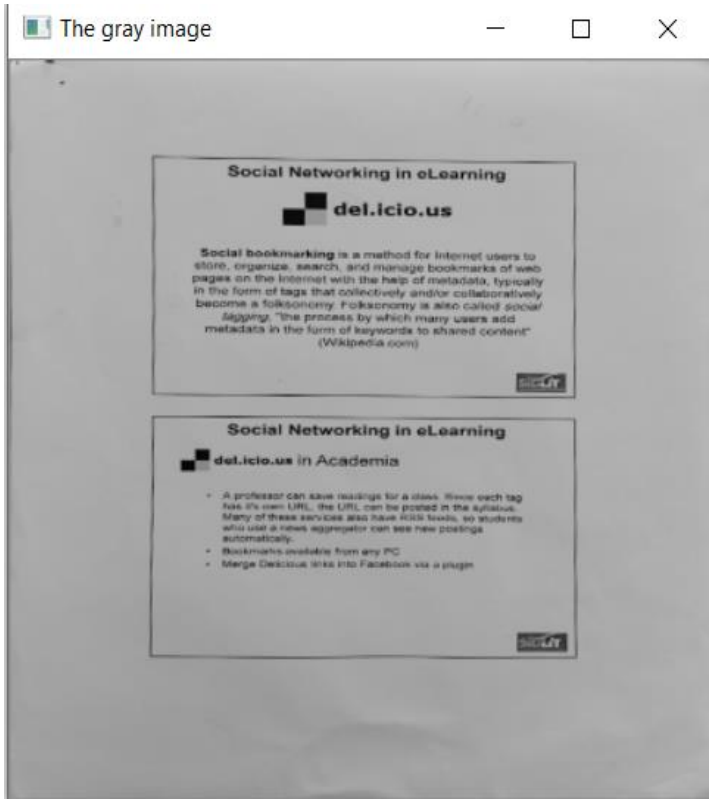
6) I choose an image which consists of 2 paragraphs and each has its set of characters and my OCR detects both the 2 paragraphs characters.

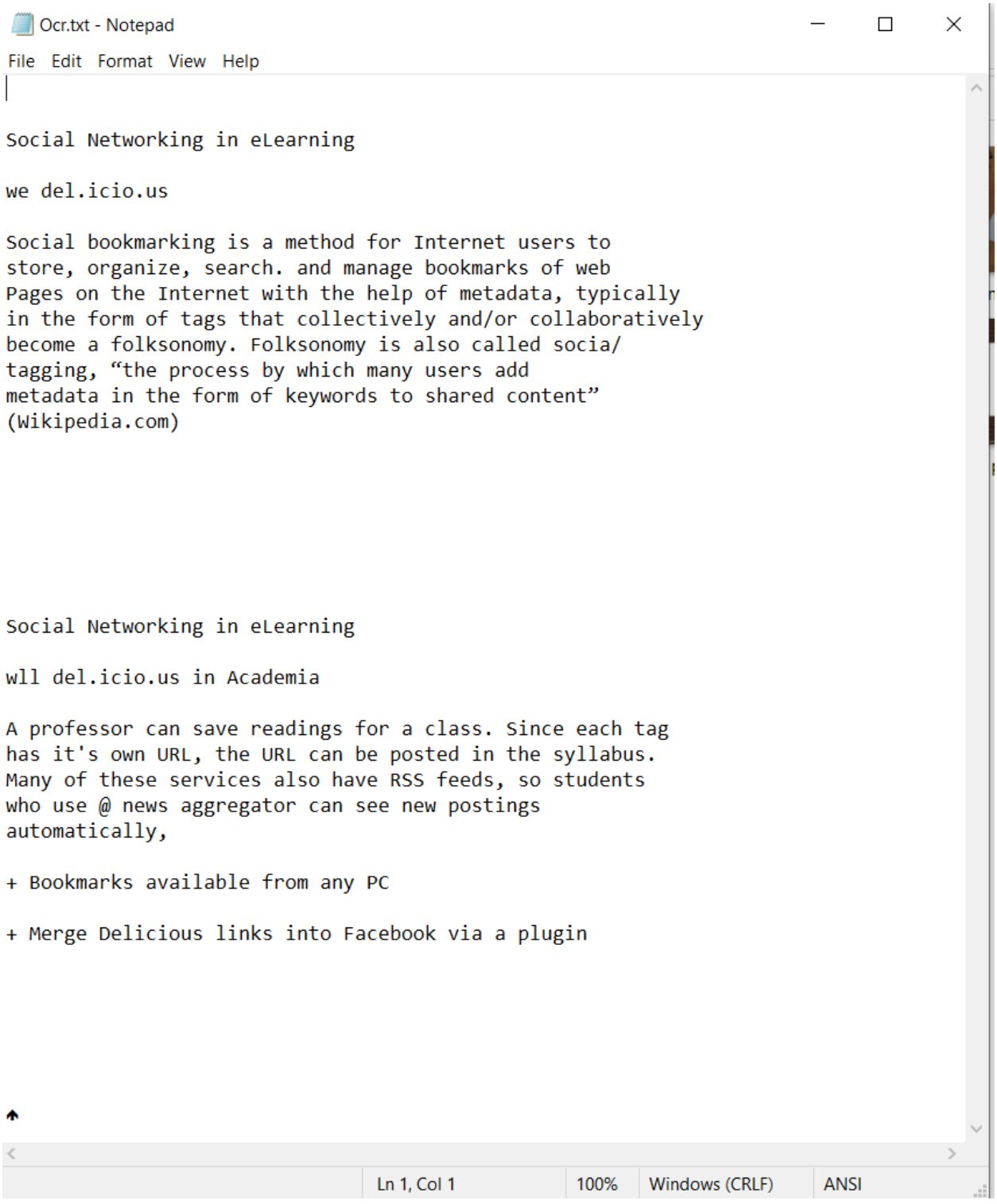



Canny Edges before Contouring


ImageContours
Social Networking in eLearning
del.icio.us
Social Networking in eLearning
del.icio.us in Academia

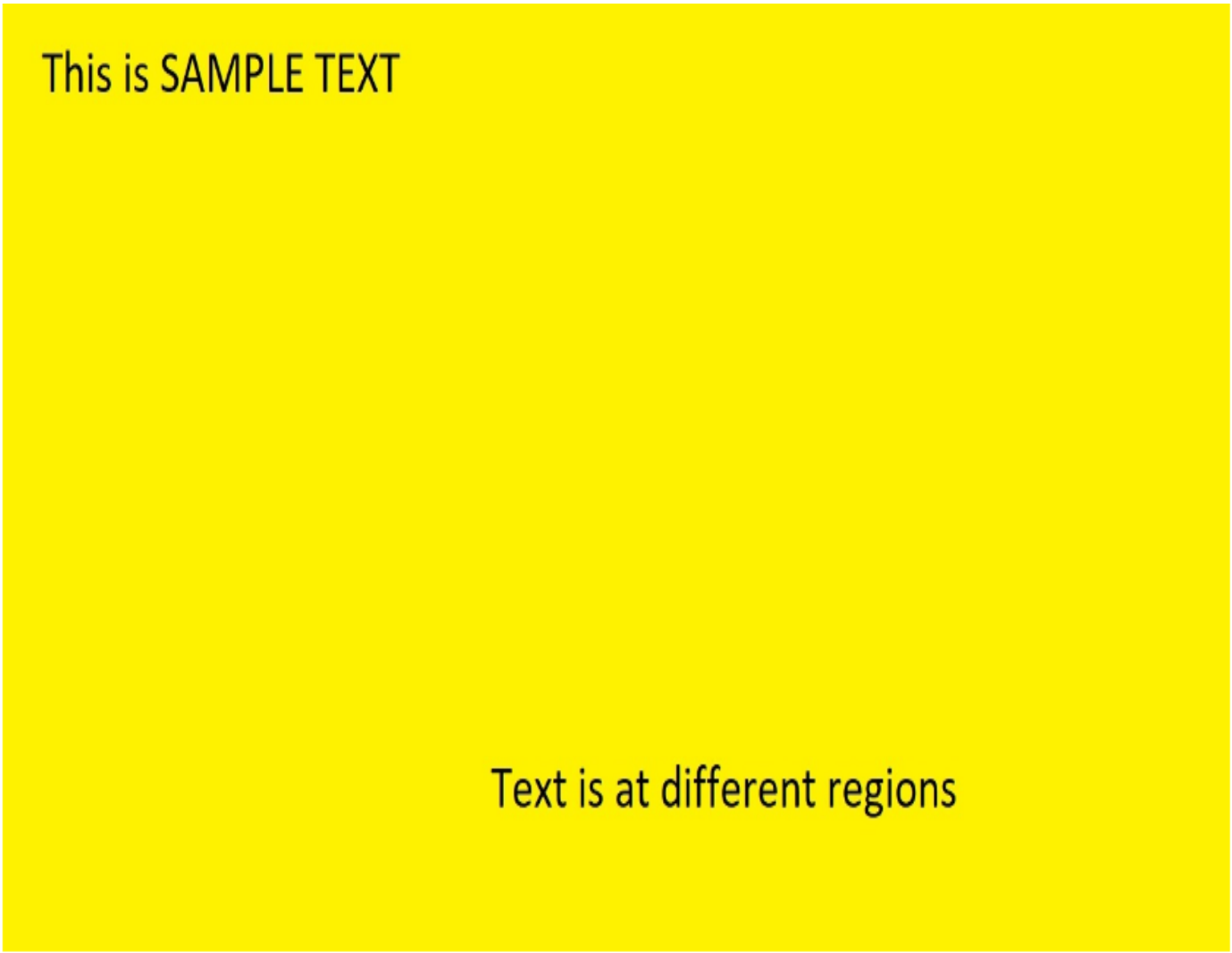

The wrapped perspective
Social Networking in eLearning
del.icio.us
Social bookmarking is a method for Internet users to store, organize, search, and manage bookmarks of web pages on the Internet with the help of metadata, typically in the form of tags that collectively and/or collaboratively become a folksonomy. Folksonomy is also called social tagging. "the process by which many users add metadata in the form of keywords to shared content" (Wikipedia.com)
Social Networking in eLearning
del.icio.us in Academia


The gray image
Social Networking in eLearning
del.icio.us
Social bookmarking is a method for Internet users to store, organize, search, and manage bookmarks of web pages on the Internet with the help of metadata, typically in the form of tags that collectively and/or collaboratively become a folksonomy. Folksonomy is also called social tagging. "the process by which many users add metadata in the form of keywords to shared content" (Wikipedia.com)
Social Networking in eLearning
del.icio.us in Academia

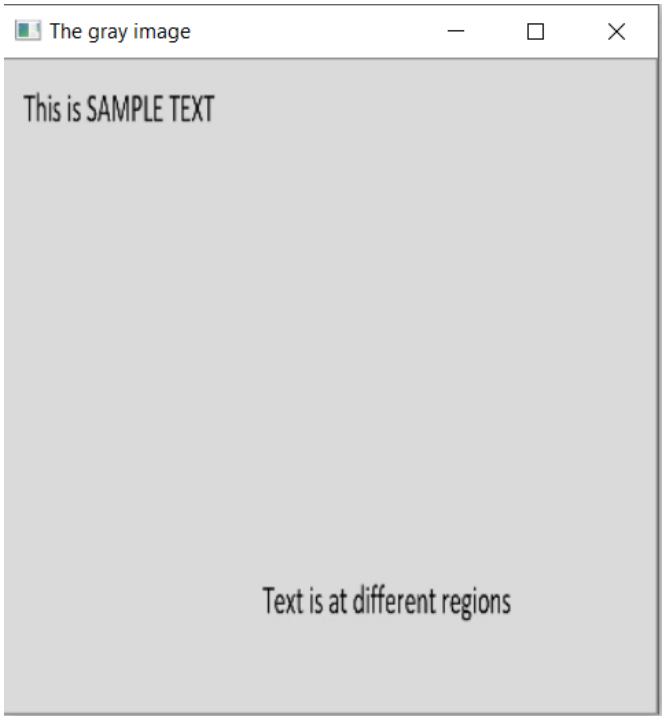


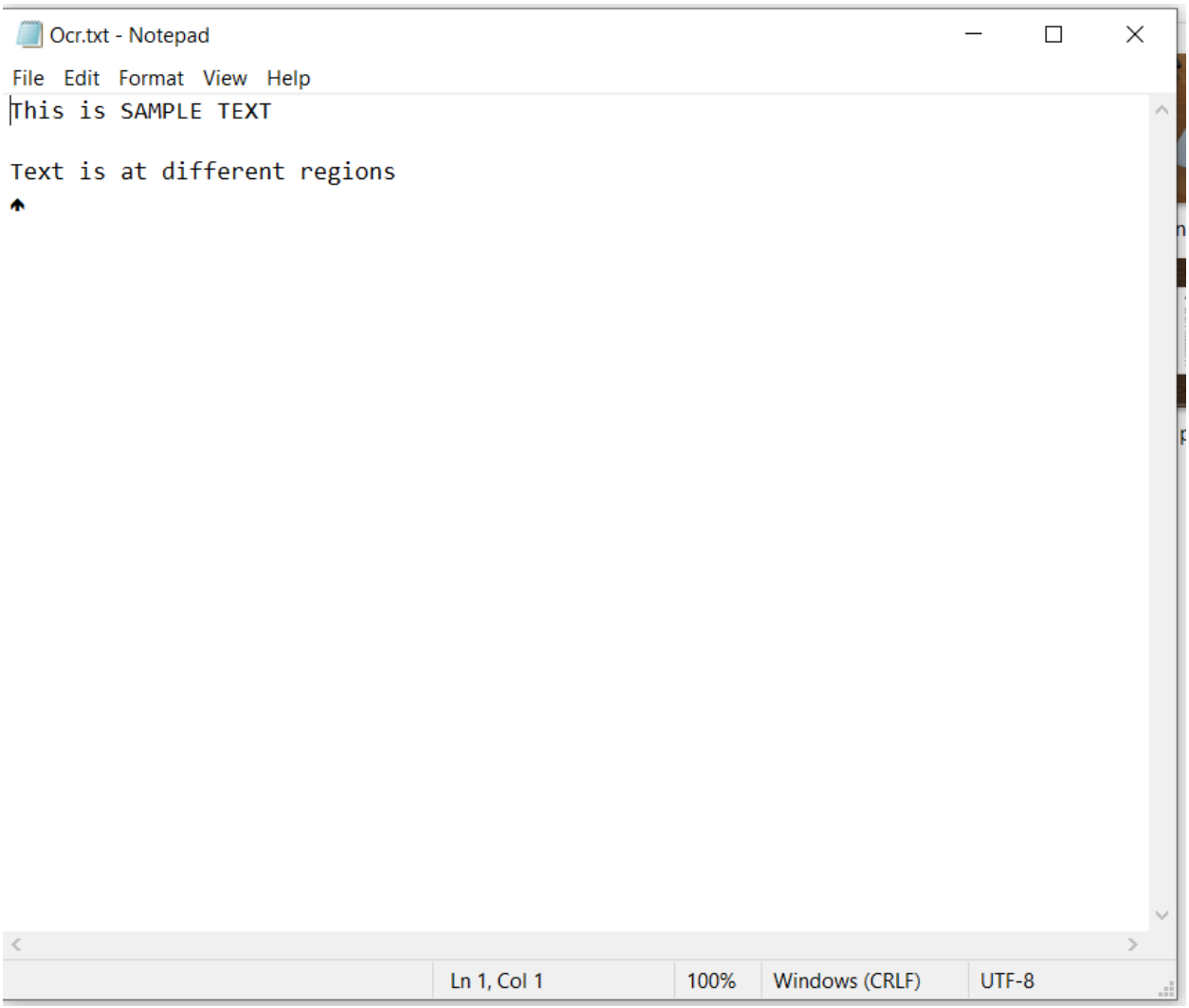

```
PS E:\3rd Computer\Computer Vision\OCR> & C:/Users/aeram/AppData/Local/Programs/Python/Python38/python.exe "e:/3rd Computer/Computer Vision/OCR/OCRproject.py"
The number of charachters in this text file is
657
```

7) I picked the given input image.



This image doesn’t need much preprocessing functions and don’t need ImagePerspective function.






```
PS E:\3rd Computer\Computer Vision\OCR> & C:/Users/aeram/AppData/Local/Programs/Python/Python38/python.exe "e:/3rd Computer/Computer Vision/OCR/OCRproject.py"
The number of charachters in this text file is
40
```