**ANKARA UNIVERSITY**

**ENGINEERING FACULTY**

**DEPARTMENT OF COMPUTER ENGINEERING**



**REPORT I**

**Optical Character Recognition**

**Mustafa Ahmet DENİZ 14290087**

**Mustafa GÖKSEVER 14290099**

# 1. Introduction

Optical character recognition (OCR) is a process of converting a printed document or scanned page into ASCII characters that a computer can recognize.

Optical Character Recognition is broadly divided into two parts, offline recognition and online recognition. Offline recognition deal with the system where input is either an image or a scanned form of the document. In this paper we are dealing only with offline recognition technique.

**Software Requirements:**

1. **Python**

It is a high-level programming language which puts emphasis on code readability. Its user-friendly syntax assists in expressing concepts in fewer lines of code.

2. **OpenCV** (Open Source Computer Vision)

It is a library of programming functions mainly aimed at real-time computer vision, which is the field of processing and analyzing digital images.

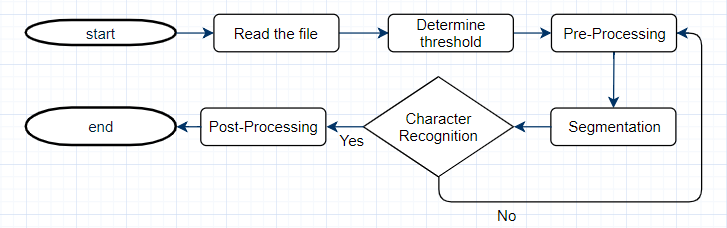


Figure number figure name ne olsun la :D

# 2. OCR Methodology

There are three basic process on which OCR works: Preprocessing, Segmentation and Character Recognition.

## 2.1 Browse

In this step, we store the path of the chosen image from the browse menu. Chosen image was read by cv2.imread() method. All file formats which is .png, .jpg etc. are supported.

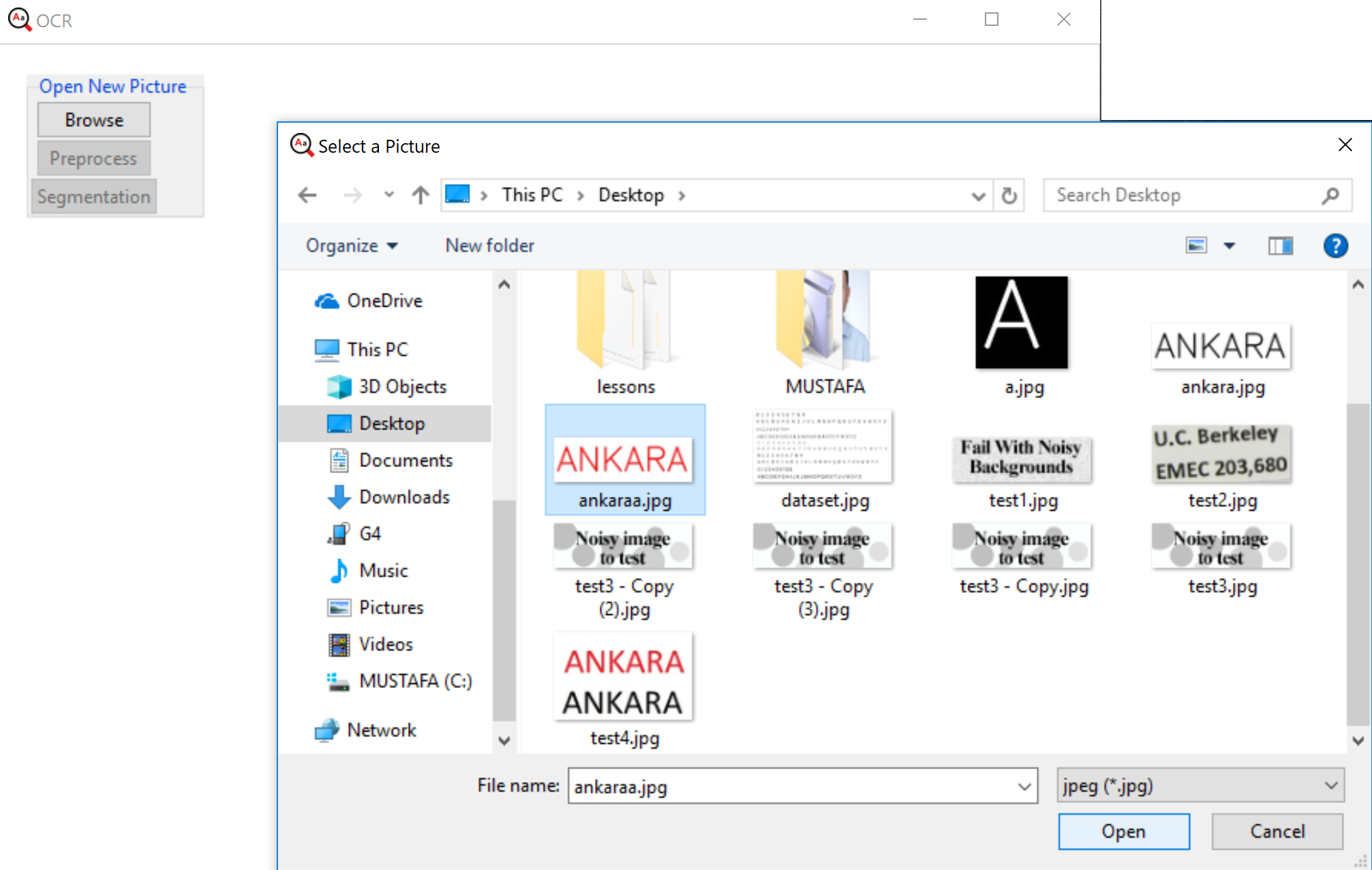


Figure Number Browse Menu

## 2.2. Preprocessing

There could be some noise, outlier object that affect the accuracy of text recognized through OCR. In order to achieve higher recognition rate, preprocessing algorithms makes the OCR system more robust by accurate image enhancement, noise removal, image thresholding, skew detection/correction

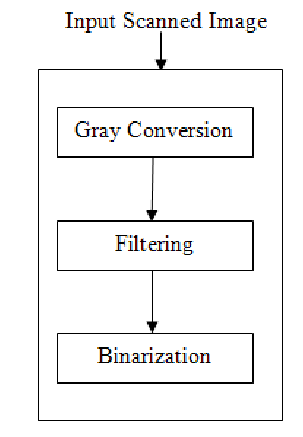


Figure number Preprocess Steps

### 2.2.1. Gray Conversion

Grayscale images have many shades of gray. For achieving accuracy input document should be grayscale. To convert a color from a color space based on an RGB color model to a grayscale representation following function is used

Y = 0.2126R+0.7152G+0.0722B

When the preprocess button was pressed, this chosen image was converted to gray image by cvtColor() method. In this method converts an input image from one color space to another. COLOR\_BGR2GRAY was used in our project. After the calculation gray image was shown.

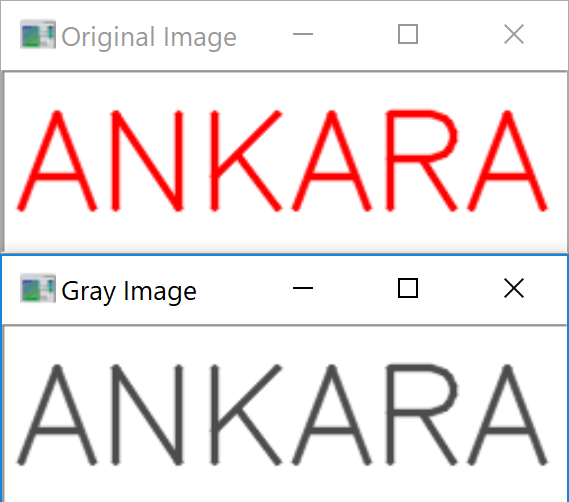


Figure number Original Image and Gray Image

### 2.2.2. Blurring

We can see that the image above needs further enhancement, therefore, we apply another blur to improve the looks

-cv2.GaussianBlur()



Figure number Gaussian Blur Image

### 2.2.3. Binarization

This step converts a multicolored image (RGB) to a black and white image. There are several algorithms to convert a color image to a binary image, ranging from simple thresholding to more sophisticated zonal analysis.

If pixel value is greater than a threshold value, it is assigned 0, else it is assigned 256. We applied thresholding with value 127 using cv2.threshold() method.



Figure number Binary Image

Preprocess is done. Now segmentation button is available to press.

# 3. Segmentation

Segmentation partitions an image into distinct regions containing each pixel with similar attributes. To be meaningful and useful for image analysis and interpretation, the regions should strongly relate to depicted objects or features of interest.

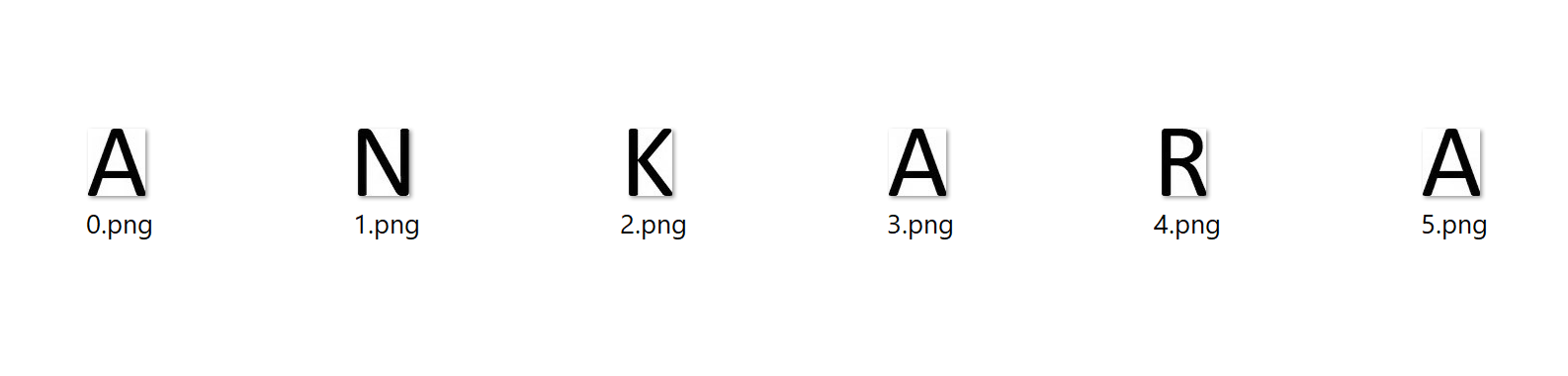
-cv2.findContours()

When the segmentation button was pressed, Contours in binary image was found and drawn yellow rectangle around the characters one by one. After that contours were sorted from left to right.



Figure number Segmentation

After that contours were sorted from left to right. Characters were found and saved in IMAGEROI Folder. It’s shown below.



# 4. Characters Recognition

cv2.matchTemplate()

Template matching is a method that check how much two images are similar to each other. Template matching works by "sliding" the template across the original image. As it slides, it compares or matches the template to the portion of the image directly under it. This method calculates all the correlation coefficients for every displacement between the input images. After the template matching, you can filter the results by decent threshold, let’s say 0.85, to check if the images are similar (this threshold depends on your input images, e.g. lightning, different sensor types etc.), then look for the max value displacement.

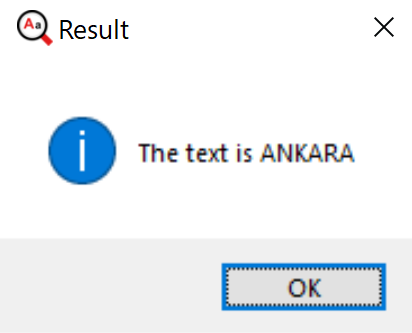


Figure number the result text from image

# 5. Future Works

Applications of our approach are varied. We hope to implement our ideas further on the various image for multiple fonts, lower case, and different size. The development of OCR to directly using machine learning algorithm which is known KNN is also something we would like to pursue in the near future.

# 6. Conclusion

We have shown that matching template can be implemented successfully in optical character recognition. The system has image preprocessing, segmentation and recognition modules for image.