# Optical Character Recognition

**J Component Project**
**for**
**ITE1015 SOFT COMPUTING**


***Submitted by***

Aradhya Mathur(17BIT0146)

Shagnik Ghosh( 16BCE0224)

Saksham Sharma( 16BCE0350)


**To**

Prof. Balakrushna Tripathy

**In**

**G1+TG1 SLOT**



**S.I.T.E.**

**VIT University, Vellore**

**Tamil Nadu - 632 014**


April 2019


VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

# SITE

## J Component Project

## Soft Computing (ITE1015)

It is certified that the project entitled "*Optical Character Recognition*" is the bonafide work for J component of Soft Computing by the following students

**Aradhya Mathur(17BIT0146)**

**Shagnik Ghosh( 16BCE0224)**

**Saksham Sharma( 16BCE0350)**

of Computer Science and Information Technology under my supervision in G1+TG1 slot during the Winter Semester 2018-19 at V.I.T. University, Vellore-632 014.

Faculty Signature:

# TABLE OF CONTENTS

## Abstract

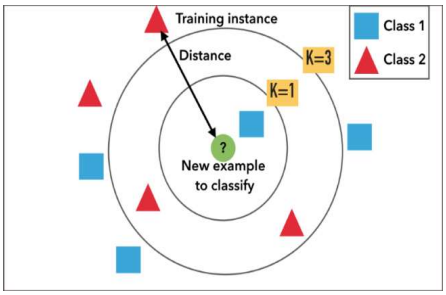
To design an OCR which is fast and reliable enough to be used in daily basis with maximum accuracy rate which can be able to distinguish different english language characters and able to read them and convert them digitally using supervised learning algorithms. OCR is one of the most pertinent problems of computer vision. OCR is the machine replication of human reading and has been the subject of intensive research for more than three decades. OCR can be described as Mechanical or electronic conversion of scanned images where images can be handwritten, typewritten or printed text. It is a method of digitizing printed texts so that they can be electronically searched and used in machine processes. It converts the images into machine-encoded text that can be used in machine translation, text-to-speech and text mining.

# Introduction

Optical Character Recognition (OCR) is a piece of software that converts printed text and images into digitized form such that it can be manipulated by machine. Unlike human brain which has the capability to very easily recognize the text/ characters from an image, machines are not intelligent enough to perceive the information available in image. Therefore, a large number of research efforts have been put forward that attempts to transform a document image to format understandable for machine. OCR is a complex problem because of the variety of languages, fonts and styles in which text can be written, and the complex rules of languages etc. Hence, techniques from different disciplines of computer science (i.e. image processing, pattern classification and natural language processing etc. are employed to address different challenges. This paper introduces the reader to the problem. It enlightens the reader with the historical perspectives, applications, challenges and techniques of OCR.

# KNN Algorithm

KNN algorithm is one of the simplest classification algorithms and it is one of the most used learning algorithms. KNN is a non-parametric, lazy learning algorithm. Its purpose is to use a database in which the data points are separated into several classes to predict the classification of a new sample point. KNN Algorithm is based on feature similarity: How closely out-of-sample features resemble our training set determines how we classify a given data point:




KNN can be used for classification—the output is a class membership (predicts a class—a discrete value). An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors. It can also be used for regression—output is the value for the object (predicts continuous values). This value is the average (or median) of the values of its k nearest neighbors.

# Stepwise KNN for OCR

**Step1:** Pick a value for K.

**Step2:** Search for the K observations in the training data that are "nearest" to the measurements of the unknown iris

**Step 3:** Use the most popular response value from the K nearest neighbors as the predicted response value for the unknown iris

**Step4**: Image Pre-Processing

**Step 5:** Digits Extraction and Training / Testing Data Preperation

**Step 6:** Feature Extraction

**Step 7**: Training

# Python Library Modules

## Python sys module

Python sys module provides easy functions that allow us to interact with the interpreter directly.
The functions python sys module provides allows us to operate on underlying interpreter, irrespective of it being a Windows Platform, Macintosh or Linux.
Sys functions:

- Python sys.modules
- Python sys.argv
- Python sys.path
- Python sys.stdin
- Python sys.copyright
- Python sys.exit
- Python sys.getrefcount

The sys module provides information about constants, functions and methods of the Python interpreter. dir(system) gives a summary of the available constants, functions and methods. Another possibility is the help() function. Using help(sys) provides valuable detail information.

## Python OS Module

Python OS module provides easy functions that allow us to interact and get Operating System related information and even control processes up to a limit.
The OS module in Python provides a way of using operating system dependent functionality.
The functions OS module provides allows us to operate on underlying Operating System tasks, irrespective of it being a Windows Platform, Macintosh or Linux. In this lesson, we will review these functions and what we can do with these.
The functions that the OS module provides allows you to interface with theunderlying operating system that Python is running on be that Windows, Mac or Linux.
OS functions

- import os
- os.system()
- os.environ()
- os.getcwd()
- os.getgid()
- os.getuid()
- os.getpid()
- os.umask(mask)
- os.uname()
- os.chroot(path)

- os.listdir(path)
- os.mkdir(path)
- os.makedirs(path)
- os.remove(path)
- os.removedirs(path)
- os.rename(src, dst)
- os.rmdir(path)

## Python NumPy module

NumPy is a general-purpose array-processing package. It provides a highperformance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

• A powerful N-dimensional array object
• Sophisticated (broadcasting) functions
• Tools for integrating C/C++ and Fortran code
• Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data.

Arbitrary data-types can be defined using Numpy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

## Python cv2 module

OpenCV-Python is a library of Python bindings designed to solve computer vision problems.

Python is a general purpose programming language started by Guido van Rossum that became very popular very quickly, mainly because of its simplicity and code readability. It enables the programmer to express ideas in fewer lines of code without reducing readability.

Compared to languages like C/C++, Python is slower. That said, Python can be easily extended with C/C++, which allows us to write computationally intensive code in C/C++ and create Python wrappers that can be used as Python modules. This gives us two advantages: first, the code is as fast as the original C/C++ code (since it is the actual C++ code working in background) and second, it easier to code in Python than C/C++. OpenCV-Python is a Python wrapper for the original OpenCV C++ implementation.

OpenCV-Python makes use of Numpy, which is a highly optimized library for numerical operations with a MATLAB-style syntax. All the OpenCV array structures are converted to and from Numpy arrays. This also makes it easier to integrate with other libraries that use Numpy such as SciPy and Matplotlib.

# DATA DESCIRPTION

We took a photo in which Capital Alphabets(A-Z), and numbers (0-9) are written. Using cv2 module we read this image into our python code. We convert image to grayscale using convert color command. Image is blurred using Gaussian Blur command. Image is filtered from grayscale to black and white. Using commands, we change pixels so that threshold is full white. Colors are inverted. Copy of thresh image is made as find contours modifies the image


# LITERATURE

| Sr. NO | NAME OF PAPER | AUTHOR AND YEAR | PROPOSED METHOD |
|---|---|---|---|
| 1 | OCR Performance Prediction using Cross-OCR Alignment | Ahmed Ben Shala, 2015 | OCR performance analysus using cross OCR allignment using suport vector regression. |
| 2 | Improving OCR Accuracy on Early Printed Books by Utilizing Cross Fold Training and Voting | Uwe Springmann ; Christoph Wick ; Frank Puppe, April 2018 | A method to significantly improve the CER on early printed books by utilizing cross fold training and con-fidence based voting was proposed. |

| | | | |
|---|---|---|---|
| 3 | Optical Character Recognition for Sanskrit Using Convolution Neural Networks | Meduri Avadesh, 2018 | approach of using convnets as classifiers for Indic OCRs. they show that convnet are more suitable than SVMs and ANNs, for multi-class image classification problems. |
| 4 | An improved scene text and document image binarization scheme | Ranjit Ghosal, 2018 | This work provides an improved scene text and document image binarization methodology. It uses both the edge and variance information of the input image. |
| 5 | Dependence models for searching text in document images | Ismet Zeki Yalniz, 2018 | a Markov Random Field (MRF) framework is proposed for searching document images and shown to be effective for searching arbitrary text in real time for books printed in English (Latin script), Telugu and Ottoman scripts. The English experiments demonstrate that the dependencies between the visual terms and letter bigrams can be automatically learned using noisy OCR output. |
| 6 | A review on using Augmented Reality in text translation | Lamma Tatwany, 2017 | Use of already existing text ocr OpenCV feature detector MSER, ABBYY cloud OCR |

| | | | |
|---|---|---|---|
| | | | and Google cloud translation API |
| 7 | Feature extraction using geometrical features for Malayalam handwritten character recognition system | K Thushara, 2017 | Feature extraction and classification for Malayalam OCR by using geometrical properties |
| 8 | Character Recognition via a Compact Convolutional Neural Network | Haifeng Zhao, 2017 | proposed to use the deep learning method directly on the scene character and word recognition with- out involving mancraft post-processing steps |
| 9 | Optical character recognition using Artificial Neural Networks | T. K. Das, 2017 | The proposed method uses neural network approach for the task of optical character recognition, but it is not effective way by using features rather than wholesome comparison, which is unsuitable for handwritten documents. |

# CODE
# FOR GENERATING DATA

```
# GenData.py

import sys
import numpy as np
import cv2
import os

# module level variables
##########################################################################
```

```
MIN_CONTOUR_AREA = 100

RESIZED_IMAGE_WIDTH = 20
RESIZED_IMAGE_HEIGHT = 30

###################################################################################################
def main():
    imgTrainingNumbers = cv2.imread("training_chars.png")           # read in training numbers image

    if imgTrainingNumbers is None:                          # if image was not read successfully
        print "error: image not read from file \n\n"        # print error message to std out
        os.system("pause")                                  # pause so user can see error message
        return                                              # and exit function (which exits program)
    # end if

    imgGray = cv2.cvtColor(imgTrainingNumbers, cv2.COLOR_BGR2GRAY)          # get grayscale image
    imgBlurred = cv2.GaussianBlur(imgGray, (5,5), 0)                        # blur

                                        # filter image from grayscale to black and white
    imgThresh = cv2.adaptiveThreshold(imgBlurred,                           # input image
                                      255,                                  # make pixels that pass the threshold full white
                                      cv2.ADAPTIVE_THRESH_GAUSSIAN_C,       # use gaussian
                                      cv2.THRESH_BINARY_INV,                # invert so foreground will be white, background will be black
                                      11,                                   # size of a pixel neighborhood used to calculate threshold value
                                      2)                                    # constant subtracted from the mean or weighted mean

    cv2.imshow("imgThresh", imgThresh)      # show threshold image for reference

    imgThreshCopy = imgThresh.copy()        # make a copy of the thresh image, this in necessary b/c findContours modifies the image

    imgContours, npaContours, npaHierarchy = cv2.findContours(imgThreshCopy,        # input image, make sure to use a copy since the function will modify this image in the course of finding contours
                                              cv2.RETR_EXTERNAL,                # retrieve the outermost contours only
                                              cv2.CHAIN_APPROX_SIMPLE)          # compress horizontal, vertical, and diagonal segments and leave only their end points

                        # declare empty numpy array, we will use this to write to file later
```

```
                    # zero rows, enough cols to hold all image data
  npaFlattenedImages =  np.empty((0, RESIZED_IMAGE_WIDTH * RESIZED_IMAGE_HEIGHT))

  intClassifications = []         # declare empty classifications list, this will be our list of how we are classifying our chars from user input, we will write to file at the end

                    # possible chars we are interested in are digits 0 through 9, put these in list intValidChars
  intValidChars = [ord('0'), ord('1'), ord('2'), ord('3'), ord('4'), ord('5'), ord('6'), ord('7'), ord('8'), ord('9'),
                   ord('A'), ord('B'), ord('C'), ord('D'), ord('E'), ord('F'), ord('G'), ord('H'), ord('I'), ord('J'),
                   ord('K'), ord('L'), ord('M'), ord('N'), ord('O'), ord('P'), ord('Q'), ord('R'), ord('S'), ord('T'),
                   ord('U'), ord('V'), ord('W'), ord('X'), ord('Y'), ord('Z')]

  for npaContour in npaContours:                          # for each contour
    if cv2.contourArea(npaContour) > MIN_CONTOUR_AREA:          # if contour is big enough to consider
      [intX, intY, intW, intH] = cv2.boundingRect(npaContour)         # get and break out bounding rect

                    # draw rectangle around each contour as we ask user for input
      cv2.rectangle(imgTrainingNumbers,           # draw rectangle on original training image
                    (intX, intY),                 # upper left corner
                    (intX+intW,intY+intH),        # lower right corner
                    (0, 0, 255),                  # red
                    2)                            # thickness

      imgROI = imgThresh[intY:intY+intH, intX:intX+intW]                                # crop char out of threshold image
      imgROIResized = cv2.resize(imgROI, (RESIZED_IMAGE_WIDTH, RESIZED_IMAGE_HEIGHT))     # resize image, this will be more consistent for recognition and storage

      cv2.imshow("imgROI", imgROI)                    # show cropped out char for reference
      cv2.imshow("imgROIResized", imgROIResized)      # show resized image for reference
      cv2.imshow("training_numbers.png", imgTrainingNumbers)      # show training numbers image, this will now have red rectangles drawn on it

      intChar = cv2.waitKey(0)                     # get key press

      if intChar == 27:                   # if esc key was pressed
        sys.exit()                    # exit program
```

```
            elif intChar in intValidChars:      # else if the char is in the list of chars we are looking for . . .

                intClassifications.append(intChar)                                          # append classification char to integer list of chars (we will convert to float later before writing to file)

                npaFlattenedImage = imgROIResized.reshape((1, RESIZED_IMAGE_WIDTH * RESIZED_IMAGE_HEIGHT))  # flatten image to 1d numpy array so we can write to file later
                npaFlattenedImages = np.append(npaFlattenedImages, npaFlattenedImage, 0)                    # add current flattened impage numpy array to list of flattened image numpy arrays
            # end if
        # end if
    # end for

    fltClassifications = np.array(intClassifications, np.float32)                 # convert classifications list of ints to numpy array of floats

    npaClassifications = fltClassifications.reshape((fltClassifications.size, 1))   # flatten numpy array of floats to 1d so we can write to file later

    print "\n\ntraining complete !!\n"

    np.savetxt("classifications.txt", npaClassifications)           # write flattened images to file
    np.savetxt("flattened_images.txt", npaFlattenedImages)          #

    cv2.destroyAllWindows()             # remove windows from memory

    return

###################################################################################################
if __name__ == "__main__":
    main()
# end if
```

## OUTPUT


# REVIEW 2 CONCLUSION:

We created a python program for generating data. We read the image in the program. This image contains alphabets and numbers. We assign the value of the characters by manually substituting different values. It stores their value and treat the alphabet or number's value as substituted manually.

# CODE
# FOR TRAINING AND TESTING:

```
# TrainAndTest.py

import cv2
import numpy as np
import operator
import os

# module level variables
##########################################################################
MIN_CONTOUR_AREA = 100

RESIZED_IMAGE_WIDTH = 20
RESIZED_IMAGE_HEIGHT = 30

##########################################################################################
###################
class ContourWithData():
```

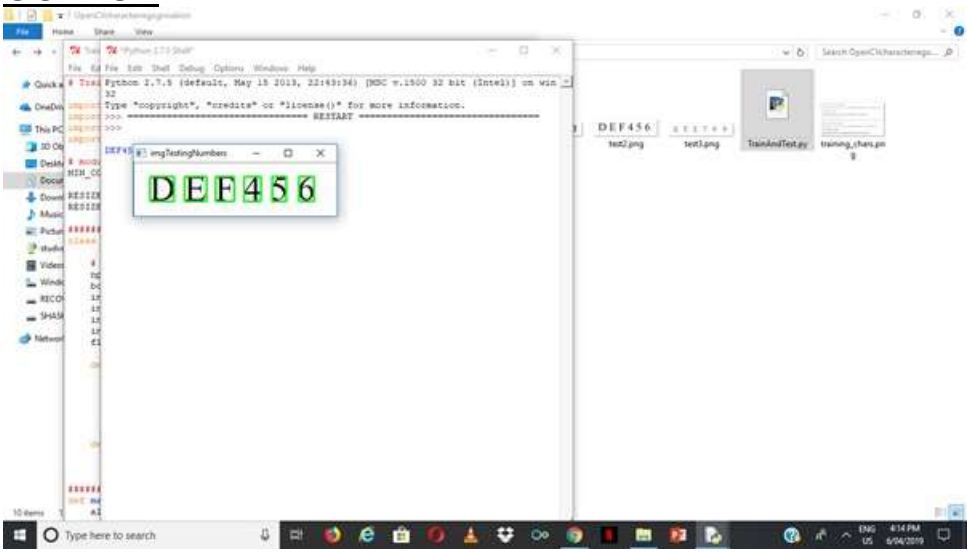
```
    # member variables
##########################################################################################
    npaContour = None           # contour
    boundingRect = None         # bounding rect for contour
    intRectX = 0                # bounding rect top left corner x location
    intRectY = 0                # bounding rect top left corner y location
    intRectWidth = 0            # bounding rect width
    intRectHeight = 0           # bounding rect height
    fltArea = 0.0               # area of contour

    def calculateRectTopLeftPointAndWidthAndHeight(self):               # calculate bounding rect info
        [intX, intY, intWidth, intHeight] = self.boundingRect
        self.intRectX = intX
        self.intRectY = intY
        self.intRectWidth = intWidth
        self.intRectHeight = intHeight

    def checkIfContourIsValid(self):                            # this is oversimplified, for a production grade
program
        if self.fltArea < MIN_CONTOUR_AREA: return False        # much better validity checking
would be necessary
        return True

##########################################################################################
###################
def main():
    allContoursWithData = []                # declare empty lists,
    validContoursWithData = []              # we will fill these shortly

    try:
        npaClassifications = np.loadtxt("classifications.txt", np.float32)                 # read in training
classifications
    except:
        print "error, unable to open classifications.txt, exiting program\n"
        os.system("pause")
        return
    # end try

    try:
        npaFlattenedImages = np.loadtxt("flattened_images.txt", np.float32)                # read in training
images
    except:
        print "error, unable to open flattened_images.txt, exiting program\n"
        os.system("pause")
        return
    # end try

    npaClassifications = npaClassifications.reshape((npaClassifications.size, 1))       # reshape numpy
array to 1d, necessary to pass to call to train
```

```
    kNearest = cv2.ml.KNearest_create()                 # instantiate KNN object

    kNearest.train(npaFlattenedImages, cv2.ml.ROW_SAMPLE, npaClassifications)

    imgTestingNumbers = cv2.imread("test2.png")         # read in testing numbers image

    if imgTestingNumbers is None:                       # if image was not read successfully
        print "error: image not read from file \n\n"        # print error message to std out
        os.system("pause")                                  # pause so user can see error message
        return                                              # and exit function (which exits program)
    # end if

    imgGray = cv2.cvtColor(imgTestingNumbers, cv2.COLOR_BGR2GRAY)       # get grayscale image
    imgBlurred = cv2.GaussianBlur(imgGray, (5,5), 0)                    # blur

                                            # filter image from grayscale to black and white
    imgThresh = cv2.adaptiveThreshold(imgBlurred,                       # input image
                                      255,                              # make pixels that pass the threshold full white
                                      cv2.ADAPTIVE_THRESH_GAUSSIAN_C,       # use gaussian rather than mean, seems to give better results
                                      cv2.THRESH_BINARY_INV,                # invert so foreground will be white, background will be black
                                      11,                               # size of a pixel neighborhood used to calculate threshold value
                                      2)                                # constant subtracted from the mean or weighted mean

    imgThreshCopy = imgThresh.copy()        # make a copy of the thresh image, this in necessary b/c findContours modifies the image

    imgContours, npaContours, npaHierarchy = cv2.findContours(imgThreshCopy,             # input image, make sure to use a copy since the function will modify this image in the course of finding contours
                                              cv2.RETR_EXTERNAL,        # retrieve the outermost contours only
                                              cv2.CHAIN_APPROX_SIMPLE)  # compress horizontal, vertical, and diagonal segments and leave only their end points

    for npaContour in npaContours:                          # for each contour
        contourWithData = ContourWithData()                                             # instantiate a contour with data object
        contourWithData.npaContour = npaContour                                         # assign contour to contour with data
        contourWithData.boundingRect = cv2.boundingRect(contourWithData.npaContour)     # get the bounding rect
        contourWithData.calculateRectTopLeftPointAndWidthAndHeight()                    # get bounding rect info
```

```
        contourWithData.fltArea = cv2.contourArea(contourWithData.npaContour)          # calculate
the contour area
        allContoursWithData.append(contourWithData)                                    # add contour with data
object to list of all contours with data
    # end for

    for contourWithData in allContoursWithData:                 # for all contours
        if contourWithData.checkIfContourIsValid():             # check if valid
            validContoursWithData.append(contourWithData)       # if so, append to valid contour list
        # end if
    # end for

    validContoursWithData.sort(key = operator.attrgetter("intRectX"))         # sort contours from left
to right

    strFinalString = ""         # declare final string, this will have the final number sequence by the end
of the program

    for contourWithData in validContoursWithData:            # for each contour
                                    # draw a green rect around the current char
        cv2.rectangle(imgTestingNumbers,                                    # draw rectangle on original testing
image
                      (contourWithData.intRectX, contourWithData.intRectY),     # upper left corner
                      (contourWithData.intRectX + contourWithData.intRectWidth,
contourWithData.intRectY + contourWithData.intRectHeight),      # lower right corner
                      (0, 255, 0),              # green
                      2)                    # thickness

        imgROI = imgThresh[contourWithData.intRectY : contourWithData.intRectY +
contourWithData.intRectHeight,     # crop char out of threshold image
                           contourWithData.intRectX : contourWithData.intRectX +
contourWithData.intRectWidth]

        imgROIResized = cv2.resize(imgROI, (RESIZED_IMAGE_WIDTH,
RESIZED_IMAGE_HEIGHT))            # resize image, this will be more consistent for recognition
and storage

        npaROIResized = imgROIResized.reshape((1, RESIZED_IMAGE_WIDTH *
RESIZED_IMAGE_HEIGHT))      # flatten image into 1d numpy array

        npaROIResized = np.float32(npaROIResized)       # convert from 1d numpy array of ints to 1d
numpy array of floats

        retval, npaResults, neigh_resp, dists = kNearest.findNearest(npaROIResized, k = 1)     # call
KNN function find_nearest

        strCurrentChar = str(chr(int(npaResults[0][0])))                                        # get character from
results
```

```
        strFinalString = strFinalString + strCurrentChar            # append current char to full string
    # end for

    print "\n" + strFinalString + "\n"                # show the full string

    cv2.imshow("imgTestingNumbers", imgTestingNumbers)      # show input image with green
boxes drawn around found digits
    cv2.waitKey(0)                                    # wait for user key press

    cv2.destroyAllWindows()             # remove windows from memory

    return

###################################################################################################
if __name__ == "__main__":
    main()
# end if
```

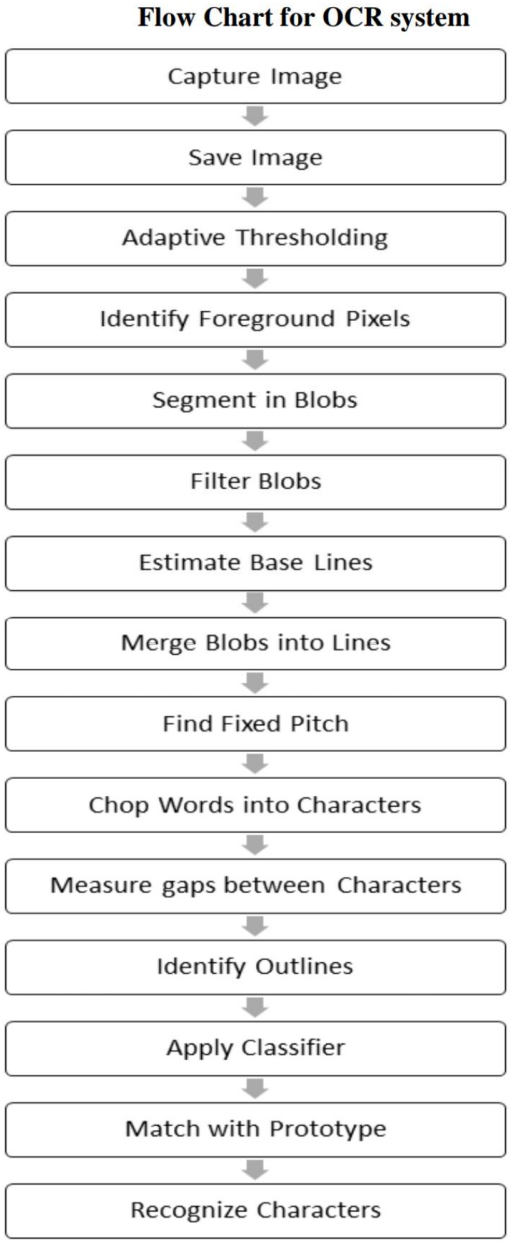
## OUTPUT




# CONCLUSION: -

We created another python connected to our first python program. The values of characters we entered manually in the first program is stored and passed on the second program. In this program we insert the image to be read. According to the values entered manually, it will read the image and provide the output.

## Flow Chart




## Software Language and Tool used: Python

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

# RESULT

After successfully training the program manually. Program is able to identify characters of the image successfully and we are able to encode and decode secret messages through images by random encoding training the program successfully.

# FUTURE SCOPE: -

### 1) ENCODING: -

This technique can be used for transferring important documents in the form of images. As the characters in the images will be trained as other characters. The image will look like codes that cannot be decoded by normally, can only be decoded through the characters manually substituted by us.

# CURRENT APPLICATION

### 1) BANKING: -

The Banking industry, along with other finance sector industries like insurance and securities, is a major consumer of OCR. The most frequent use of OCR is to handle cheques: a handwritten cheque is scanned, its contents converted into digital text, the signature verified and the cheque cleared in real time, all with human involvement. While near 100% accuracy has been achieved for printed cheques (with only the signature verification requiring a match against a pre-existing database), it is still a long way off for complete autonomy for handwritten cheques. Nevertheless, with deep learning AI techniques applied to handwriting OCR, this problem is not as unsurmountable as it might seem. A reduced turnaround time for cheque clearance is an economic gain for all, from payer to bank to payee.

### 2) LEGAL: -

Few industries generate as much paperwork as the legal industry, and so OCR has multiple applications herein. Reams and reams of affidavits, judgements, filings, statements, wills and other legal documents, especially the printed ones, can be digitised, stored, databased and made searchable using the simplest of OCR readers. With OCR technology extending to languages not using the Roman script, this brings into the digital world documents in Chinese, Arabic and other scripts, too. For an industry heavily reliant on judicial precedent, fast access to legal documents from millions of past cases is surely a place.

# References


[1] Patel C, Patel A, Patel D. Optical character recognition by open source OCR tool tesseract: A case study. International Journal of Computer Applications. 2012 Jan 1;55(10).

[2] Ye Q, Doermann D. Text detection and recognition in imagery: A survey. IEEE transactions on pattern analysis and machine intelligence. 2015 Jul 1;37(7):1480-500.

[3] Jain A, Dubey A, Gupta R, Jain N, Tripathi P. Fundamental challenges to mobile based ocr. vol. 2013 May;2:86-101.

[4] Moravec K. A Grayscale Reader for Camera Images of Xerox Data Glyphs. In BMVC 2002 Sep (pp. 1-10).

[5] Smith R, Antonova D, Lee DS. Adapting the Tesseract open source OCR engine for multilingual OCR. In Proceedings of the International Workshop on Multilingual OCR 2009 Jul 25 (p. 1). ACM.

[6]Gustav Tauschek. Reading machine. U.S. Patent 2026329,

http://www.google.com/patents?vid=USPAT2026329,

December 1935FLEXChip Signal Processor (MC68175/D), Motorola, 1996. [Accessed 23/11/2016]

[7] Paul W. Handel. Stat IST ical Machine. U.S. Patent 1915993,http://www.google.com/patents?vid=USPAT1915

993, June 1993. [Accessed 23/11/2016]

[8] Mori S, Suen CY, Yamamoto K. Historical review of OCR research and development. Proceedings of the IEEE. 1992 Jul;80(7):1029-58.

[9] Amin A. Off-line Arabic character recognition: the state of the art. Pattern recognition. 1998 Mar 1;31(5):517-30.

[10] Stallings W. Approaches to Chinese character recognition. Pattern recognition. 1976 Apr 30;8(2):87-98. Zhai X, Bensaali F, Sotudeh R. OCR-based neural network

for ANPR. In2012 IEEE International Conference on Imaging Systems and Techniques Proceedings 2012 Jul 16

(pp. 393-397). IEEE.

[11] Shamsher I, Ahmad Z, Orakzai JK, Adnan A. OCR for printed urdu script using feed forward neural network. In Proceedings of World Academy of Science, Engineering and Technology 2007 Aug (Vol. 23, pp. 172-175).

[12] Yetirajam M, Nayak MR, Chattopadhyay S. Recognition and classification of broken characters using feed forward neural network to enhance an OCR solution. International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume. 2012 Oct 28;1