**DEEP LEARNING BASED BRAILLE-TEXT-SPEECH**

**CONVERSION**

***A PROJECT REPORT SUBMITTED BY***

Ebin Antony : URK18CS021

Ashby Issac : URK18CS055

Jensen Thomas : URK18CS059

Joswin V Jaison : URK18CS097

***in partial fulfillment for the award of the degree of***

**BACHELOR OF TECHNOLOGY**

***in***

**COMPUTER SCIENCE AND ENGINEERING**

***under the supervision of***

**Dr.Kumudha Raimond**





**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING SCHOOL OF ENGINEERING AND TECHNOLOGY**

**KARUNYA INSTITUTE OF TECHNOLOGY AND SCIENCES**

(Declared as Deemed-to-be-University under Sec-3 of the UGC Act, 1956)

**Karunya Nagar, Coimbatore - 641 114, India.**

# APRIL 2022

**ACKNOWLEDGEMENT**

First and foremost, we praise and thank **ALMIGHTY GOD** for giving us the will power and confidence to carry out our project.

We are grateful to our beloved founders Late **Dr. D.G.S. Dhinakaran**, **C.A.I.I.B, Ph.D.,** and **Dr. Paul Dhinakaran**, **M.B.A., Ph.D.,** for their love and always remembering us in their prayers.

We extend our thanks to **Dr. P. Mannar Jawahar, Ph.D.,** our honorable vice chancellor, **Dr. E. J. James, Ph.D.,** and **Dr. Ridling Margaret Waller, Ph.D.,**our honorable Pro-Vice Chancellor(s) and **Dr. R. Elijah Blessing, Ph.D.,** our respected registrar for giving us the opportunity to carry out this project.

We are thankful to **Dr. G. Prince Arulraj, M.E., Ph.D.,** Dean (Engineering & Technology) for his support and encouragement.

We would like to place our heart-felt thanks and gratitude to **Dr. J. Immanuel Johnraja, Ph.D.,** HOD, Department of Computer Science and Engineering for his encouragement and guidance.

We are grateful to our guide, **Dr.Kumudha Raimond** ,Professor, Department of Computer Science and Engineering for her valuable support, advice and encouragement.

We also thank all the staff members of the Department for extending their helping hands to make this project work a success. We would also like to thank all my friends and my parents who have prayed and helped me during the project work.

# BONAFIDE CERTIFICATE

Certified that this project report **“Deep Learning based braille-text-speech conversion”** is the bonafide work of “Ebin Antony(URK18CS021),Ashby Issac(URK18CS055),Jensen Thomas(URK18CS059),Joswin V Jaison(URK18CS097) **”** who carried out the project work under my supervision.

**SIGNATURE SIGNATURE**

# Dr. J. Immanuel Johnraja Dr.Kumudha Raimond

**Head of the Department Department of Computer Science and Engineering**

**Professor**

**Department of Computer Science and Engineering**

Submitted for the Project Viva Voce held on………29/1/2022……………….

**Examiner**

**Content**

|  |  |
| --- | --- |
| Chapter 1 | Introduction |
| Chapter 2 | Explanation of proposed system |
| Chapter 3 | Implementation details |
| Chapter 4 | Conclusion |
| Chapter 5 | References |
| Appendix A | Source Code |
| Appendix B | Screenshots |

**Chapter 1:Introduction**

The World Health Organization (WHO) reports that there are about 285 million people who are visually impaired worldwide. The number of people suffering from visual impairments and blindness is increasing each year. As the visually impaired and blind cannot write and read texts, they use the braille system. Braille is not a language but many people around the world use it to communicate with each other. Braille is a text recognition system that enables the visually impaired and blind people to access and respond to any written documents. Visually impaired and blind individuals can read the Braille system by using their fingers to trace raised dots, while sighted people can read it with their eyes.

Braille is a writing system that consists of cells in which there are six raised dots and each raised dot will be having a number from one to six which are organized in two columns. To make the lives of the visually impaired people more easier and efficient for communicating with others we need to provide Braille-assisted technology. Even though many blind people around the world have used the braille system most sighted people cannot understand the braille documentation of visually impaired people. For the sighted people to interact and help those individuals who are having visual impairments the sighted people also have to access the braille system. Even in the case of medicine, for blind people, the information will be printed in the braille system but some visually impaired people don’t know to use the braille system. To solve this an assistive technology is required. The introduction of assistive technology would help us to convert braille symbols into texts or even voices which would then help in supporting students in schools or universities. Another issue is that sighted people could also face difficulty in reading and understanding braille symbols that are designed for the visually impaired people as the braille system will script in different subjects such as chemistry or mathematics.

As the braille cells don’t consist of continuous strokes like the characters of natural languages braille images cannot be recognized with the help of traditional optical character recognition systems. The Optical Braille Recognition system is a method that is used for capturing and processing the braille documents and converting those documents into natural language. There are three main phases in optical braille recognition system: image capturing, image segmentation, and braille to text representation. We propose a deep learning based approach for optical braille recognition system. The rest of the sections describes about the related studies in the literature and their limitations and also details about the materials and methods used in this research, including a short description of the proposed approach. The final section presents the conclusions and future work of the study.

**Chapter 2 : Explanation of proposed system**

a)Architecture

Braille to text conversion

Removal of noise in the image using gaussian blur function

Perform erosion and dilation in the image

Contour detection

Convert image into a grid of 0s and 1s

Find the difference between consecutive x coordinates of bounding boxes.

Create an array of coordinates

Produce a matrix of window size 3x2

Convert the position of one's in the matrix into a list

Transform the list into letters using a dictionary

Text to speech conversion

Text

Preprocessing of text

Linguistic analysis

Grapheme to phoneme conversion

Prosodic pattern determination

Speech signal generation

Generation of sound

b)Methodology

There are two methods primarily which we have researched on the implementation of braille to text conversion. The first method is based on optical braille recognition.It uses image processing instead of artificial neural networks .This technique is contour based.

It is the basis of variety of tasks like object detection and image segmentation. Firstly we can use image morphing technique to reduce the noise and increase the sharpness of dots. We can do this by the conversion of RGB/BGR scale to Gray scale. Then we can apply segmentation using binary thresholding and OTSU thresholding. Binary thresholding is an image processing technique that create a binary image by giving a threshold value .Otsu thresholding convert the image into two groups foreground and background . This technique is used to perform automatic image thresholding. We can remove the noise by the process of blurring . Blurring is like applying low pass filter to an image. Then we are repeating the above three steps . Then we are converting the result of final thresholding to binary image.

Then we are extracting contours from the binary image . This method is used for shape analysis , object detection and recognition We find contours by a method proposed by Satoshi Suzuki et el. This method is already implemented in Open CV so we need not define the entire algorithm as it exceeds the scope of this study. After the particular technique we use braille character segmentation . Segmentation is very simply built and unique to our study. The main feature about this segmentation is that this research can be further improved with Artificial Neural Networks if desired as this segmentation allows us to separate individual braille character with all the pixel data required for Convolutional Neural Networks. The Segmentation process is as follows,

1. Calculate the Mean Radius and Diameter of each dots in each Braille Character using the Contours.

2. Using the Radius and Diameter, Find the Center Point of all dots in each Braille Characters.

3. Find the smallest Center Point of all dots in each Braille Characters in both X and Y Axis.

4. With the X and Y points find all possible Braille Characters as3x2matrix.

5. Get each Braille Character bounding Rectangle Coordinates.

6. With the bounding Rectangle Coordinates, Each Braille Character can be Segmented.

With each Braille Character segmented as separate images we can do simple image processing to convert the image to a3x2matrix, this matrix can be converted to vector with single dimension. Each combination of such vectors will represent a character in English, Thus a sentence can be constructed.

The second technique by which we can implement the conversion of braille to text is through Open CV. It is a library which can be used for computer vision and deep learning applications.It is also useful in image processing. When considering our project we have used it with the help of edge detection technique.It is used to find the particular boundaries of objects within an image.Deep learning techniques require greater amount of image segmentation and data extraction task. Inorder to perform these operation successfully edge detection methods can be used in computer vision and deep learning.

If I[a,b] be our image in 2 directions where ‘a’ can be considered as the width and ‘b’ can be considered as the height.Edge can be considered as a part of l[a,b].And there will be a change in colour intensity.

After the conversion of braille images into text the particular text need to be converted into speech.Text to speech technology helps in the reading of words and convert them into audio.The functioning of text to speech technology is in such a way that first it will convert the written text to phonemic representations .Then the phonemic representations will be converted to waveforms.It is a deep neural network which helps in the generation of audio.Firstly text will undergo preprocessing.Then linguistic analysis will be performed on the preprocessed text.It includes morphology,syntactic structure and markup.Then the graphemes are converted into phonemes.The next stage in text to speech is prosodic pattern determination.Prosodic patterns involve accent , stress , intonation , rhythm etc.After that phoneme is converted to diphone and speech signal is generated.In order to convert text to speech there are several API’s available. We have implemented the problem statement of converting text to speech with the help of gTTS API.gTTS can convert the text to speech and it will be saved as an MP3 file.Several Indian languages as well as foreign languages are supported by gTTS. For using the operating system related functionality while converting text to audio we can use OS module.

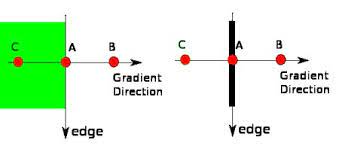
c) Background and related works

Optical braille recognition is the technique we are integrating with the respective project. Optical braille recognition is the act of capturing and processing braille characters into natural characters. It is used to convert braille documents for people who cannot read them. In 1984 a group of researchers at the Delft University of Technology designed a braille reading tablet in which a reading head with the photosensitive cells moved along the ruler to capture braille text line by line. We have come across fifteen research papers related to the implementation of our project. The first research paper is about Braille character detection using deep neural networks for an educational robot for visually impaired people. It helps to simplify the learning process for blind people .They have used a specific type of convolutional neural network called Mask R Cnn. Another algorithm with which they have implemented the project is Yolo . The second research paper is about a Deep Learning-Based Recognition Approach for the Conversion of Multilingual Braille Images. They Proposed a deep learning approach for conversion of braille images into multilingual texts. They have used a type of convolutional neural network called Deep convolutional neural network. It is used to identify patterns in videos and images .The next research paper we have gone through deals with a Braille Recognition approach for Reducing Asymmetric Communication between the Blind and Non-Blind . Here they have used convolutional neural network method . The next research paper is about a Deep Learning Method for Braille Recognition which solves the problems of automatic feature extraction and dimension reduction in Braille recognition. The sixth research paper is mainly concerned about a deep learning scheme for character prediction with position-free touch screen-based Braille input method. The proposed Braille input method for touch screen devices can predict the user's input with high accuracy. Classification model is proposed. It predict the new values or labels based on previous data. Object detection method is also used for the conversion of braille document into text. The particular method is used for the identification as well as to locate each object in images or frames. The braille documents can be written in different languages. We have gone through a research paper which deals with a braille document written in Arabic language. Dot part detection method is used for the implementation. It can detect dots composing braille characters in a document. We have also come across another research paper which convert the braille document written in Tamil language into text by dot detection method. Braille cell recognition and braille cell transcription are the other 2 techniques which have been used for the implementation of project. It can be achieved by using image processing technique. Three classes of Grey level are used. They are background, recto dots and verso dots. Image acquisition and Image deskewing are the two techniques which are a part of image processing. Image acquisition is the action of retrieving an image from a source. The source can be cameras, sensors etc. While image deskewing is the process of straightening an image. There are also various surveys regarding the benefits of conversion of braille document to text. The digitization of braille document is currently emerging as a need.

d) Algorithm

Object detection is a computer vision technique for locating instances of objects in images or videos. Object detection algorithms typically leverage machine learning or deep learning to produce meaningful results. When humans look at images or video, we can recognize and locate objects of interest within a matter of moments. The goal of object detection is to replicate this intelligence using a computer. Object detection is a key technology behind advanced driver assistance systems (ADAS) that enable cars to detect driving lanes or perform pedestrian detection to improve road safety. Object detection is also useful in applications such as video surveillance or image retrieval systems. We can use a variety of techniques to perform object detection. Popular deep learning–based approaches using convolutional neural networks (CNNs), such as R-CNN and YOLO v2, automatically learn to detect objects within images.We can choose from two key approaches to get started with object detection using deep learning. Create and train a custom object detector. To train a custom object detector from scratch, you need to design a network architecture to learn the features for the objects of interest. You also need to compile a very large set of labeled data to train the CNN. The results of a custom object detector can be remarkable. That said, you need to manually set up the layers and weights in the CNN, which requires a lot of time and training data. Use a pre trained object detector. Many object detection workflows using deep learning leverage transfer learning, an approach that enables you to start with a pre trained network and then fine-tune it for your application. This method can provide faster results because the object detectors have already been trained on thousands, or even millions, of images. The Standard deviation of the Gaussian function plays an important role in its behaviour.The values located between +/- σ account for 68% of the set, while two standard deviations from the mean (blue and brown) account for 95%, and three standard deviations (blue, brown and green) account for 99.7%. account for 99.7%.This is very important when designing a Gaussian kernel of fixed

length. The Gaussian function is used in numerous research areas .It defines a probability distribution for noise or data. It is a smoothing operator. It is used in mathematics.The Gaussian function has important properties which are verified with The Gaussian function has important properties which are verified with respect to its integral: In probabilistic terms, it describes 100% of the possible values of any given space when varying from negative to positive values. We need to produce a discrete approximation to the Gaussian function. The Gaussian filter works by using the 2D distribution as a point-spread function. This is achieved by convolving the 2D Gaussian distribution function with the image. We need to produce a discrete approximation to the Gaussian function. This theoretically requires an infinitely large convolution kernel, as the Gaussian distribution is non-zero everywhere. Fortunately the distribution has approached very close to zero at about three standard deviations from the mean. 99% of the distribution falls within 3 standard deviations. This means we can normally limit the kernel size to contain only values. This means we can normally limit the kernel size to contain only values within three standard deviations of the mean.



Canny edge detection

Edge detection is a technique of image processing used to identify points in a digital image with discontinuities, simply to say, sharp changes in the image brightness. These points where the image brightness varies sharply are called the edges (or boundaries) of the image. It is one of the basic steps in image processing, pattern recognition in images and computer vision. When we process very high-resolution digital images, convolution techniques come to our rescue. In our project we have used canny edge detection. This is the most commonly used highly effective and complex compared to many other methods. It is a multi-stage algorithm used to detect/identify a wide range of edges. The following are the various stages of the Canny edge detection algorithm

1.Convert the image to grayscale

2.Reduce noise – as the edge detection that using derivatives is sensitive to noise, we reduce it.

3.Calculate the gradient – helps identify the edge intensity and direction.

4.Non-maximum suppression – to thin the edges of the image.

5.Double threshold – to identify the strong, weak and irrelevant pixels in the images.

6.Hysteresis edge tracking – helps convert the weak pixels into strong ones only if they have a strong pixel around them.

**Chapter 3: Implementation details**

a. Removing noise using Gaussian Blur function

Images can contain noise. These noise can be because of camera sensors and different sources by which we retrieve the image. Inorder to remove these noise we can use image smoothing approaches. A variety of approaches can be used with openCV for the image smoothing approach. Image smoothing technique is also called Blurring.We have used Gaussian filter in our project.Gaussian filter gives significance to pixels near the edge.There are various advantages of using Gaussian filtering.They are easy to implement and it helps in automatic sensoring.It helps to produce images that is symmetric. The main demerit of Gaussian filter is that it will affect fine details and contrast.If we are working with medical images that require some of the finest detail or point in the image,Gaussian image smoothing technique will not be useful.Gaussian filter is effective than many linear filters like mean filter. Non linear filters like median filters can also be used. Median filters can preserve edge while removing noise. Gaussian filters are primarily used in blurring the image and reducing the noise.On the other hand median filter is used to reduce the amount of intensity variation from one pixel to other pixel. The syntax of Gaussian blur function is as follows:

Dst = cv2.GaussianBlur(src, ksize, sigmaX[, dst[, sigmaY[, borderType=BORDER\_DEFAULT]]] )

src stands for input image , dst stands for output image, ksize represents the Gaussian kernel size. SigmaX is the kernel standard deviation along X axis. SigmaY is the kernel standard deviation along Y axis. Border type represents the image boundaries .We are also performing two morphological image processing operations . They are erosion and dilation. They modify the geometrical structure in an image. These operations helps to remove noise , isolate individual elements and identify intensity bumps or holes in the image. For expanding an image we prefer dilation. It add the pixels to the boundaries of image. It is controlled by structuring element which can be defined as a matrix of 0 and 1. In dilation the regions of white shade increases. While the regions of darker shade decreases. In the case of erosion the regions of darker shade increases and the regions of white shade decreases. Simply we can say that dilation add pixels to boundaries while erosion remove pixels from boundaries.

cv2.dilate(src, dst, kernel): dilation syntax

cv2.erode(src, dst, kernel): erosion syntax

b. Finding contours

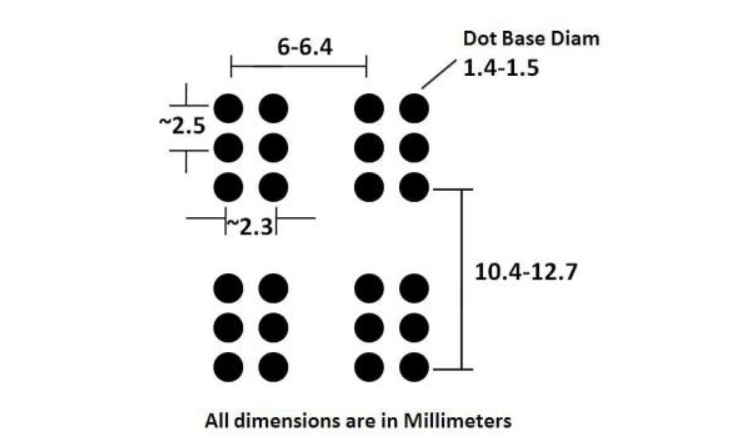
Contours are the line joining all the points that may have the same intensity. By the process of contour detection we can detect the border in an image. It is highly used in shape analysis ,object detection and recognition. There are two algorithms for contour detection. They are CHAIN\_APPROX\_SIMPLE and CHAIN\_APPROX\_NONE. In this project we have used CHAIN\_APPROX\_SIMPLE. In order to find contours first we need to read the image and convert it into grayscale format. After that we need to apply binary thresholding or canny edge detection to the grayscale image. Then use the findContours() function to detect contours followed by drawContours() function. The syntax for findContours() function is written below:

cv2.findContours(image, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

RETR\_EXTERNAL means that only external contours will be kept. After contour detection we know that the image is comprised of mostly black dots of same size. So the diameter of one could be calculated by finding the most frequently occurring height or width among the bounding boxes. In order to find the contours of all circles we are selecting only circles that have width or height which falls within a certain range of determined diameter.

c.Convert into a grid of 1’s and 0’s

To translate the image into text we are converting the image into a grid of 1 and 0.1 is the black dot.0 can be the missing ones. Even though finding the location of 1 is an easy task finding the location of missing dots is a difficult task.We need to calculate some distances for finding the location of missing dots.



Distance between dots and letters may not be same. So we need to calculate the distance for each image individually. We need to align all the dots in one line to have same Ycordinate and all the dots in one column need to have same X coordinate.For this we need to sort the dots in reading order. That is first we are considering left to right then we are considering top to bottom. Sometimes the dot may be out of order if the pixels from top of one dot is not aligned to the pixels from top of other dot. We need to use a tolerance variable for grouping similar x and y coordinates.

d. Find the differences between distance of consecutive x coordinates of the bounding boxes.

The 2 neighboring dots of the same alphabet will have the smallest distance. The whitespace between two letters is the next smallest distance. The same dots in adjacent letters will have the next smallest distance. In this manner we are actually breaking the image into a grid where each cell can be either an empty dot or a filled dot. In the code we are specifying prev=0 which means that the last place was outside of a letter.

e. Create an array of coordinates

We have drawn a vertical line that seperates consecutive x coordinates of bounding box. A bounding box can be defined as an imaginary rectangle that act as a point of reference for object detection. Data annotators help to draw bounding boxes over images defining its x and y coordinates. Now we need to create an array of coordinates for vertical lines.

f. Produce a matrix of window size 3x2

Now we will get a matrix of window size 3x2. This matrix will have a stride of 2. A stride is a parameter that defines the amount of movement in an image. Then we are converting the location of one’s in the matrix into a list. And then we are transforming these lists into letters using a specific dictionary. Suppose a point is converted into a list which is of the form [2,3,4]. It represents the letter j.

**Chapter 4 : Conclusion**

We have implemented the conversion of braille image to text with the help of edge detection technique.The resultant text is converted to speech with the help of google text to speech module.The conversion of the braille images to text will be helpful for the people who are not blind to understand the braille text. And on converting the braille text to speech the blind people will be able to hear and learn what is written and they could grasp more . We have used Open CV to implement edge detection.

**References**

1. A Deep Learning-Based Recognition Approach for the Conversion of Multilingual Braille Images :King Saud University

https://www.researchgate.net/publication/350022456\_A\_Deep\_Learning-Based\_Recognition\_Approach\_for\_the\_Conversion\_of\_Multilingual\_Braille\_Images

1. Deep learning scheme for character prediction with position-free touch screen-based Braille input method

<https://hcis-journal.springeropen.com/articles/10.1186/s13673-020-00246-6>

1. A Review of Optical Braille Recognition

https://www.researchgate.net/publication/277331678\_A\_Review\_of\_Optical\_Braille\_Recognition

1. Braille character detection using deep neural networks for an educational robot for visually impaired people.

<https://www.researchgate.net/publication/344952721_Braille_character_detection_using_deep_neural_networks_for_an_educational_robot_for_visually_impaired_people>

1. Optical Braille Recognition Using Object Detection CNN

<https://www.researchgate.net/publication/347796795_Optical_Braille_Recognition_Using_Object_Detection_CNN>

1. Braille Character Recognition Based on Neural Networks

<https://www.researchgate.net/publication/328087745_Braille_Character_Recognition_Based_on_Neural_Networks>

1. A STUDY ON OBJECT DETECTION

<https://www.researchgate.net/publication/338253407_A_STUDY_ON_OBJECT_DETECTION>

1. Text Image to Braille Code Converter

<https://www.technoarete.org/common_abstract/pdf/IJERECE/v5/i6/Ext_61295.pdf>

1. <https://github.com/antony-jr/OpticalBrailleRecognition>
2. <https://www.pharmabraille.com/>
3. Investigation on the effect of a Gaussian Blur in image filtering and segmentation

<https://www.researchgate.net/publication/261278360_Investigation_on_the_effect_of_a_Gaussian_Blur_in_image_filtering_and_segmentation>

1. <https://www.javatpoint.com/opencv>

**Appendix A – Source Code**

from imutils.perspective import four\_point\_transform as FPT

from collections import Counter

import matplotlib.pyplot as plt

from imutils import contours

from skimage import io

import numpy as np

import imutils

import cv2

import re

from gtts import gTTS

import os

language = 'en'

import warnings

warnings.filterwarnings("ignore")

def get\_image(url, iter = 2, width = None):

image = io.imread(url)

if width:

image = imutils.resize(image, width)

ans = image.copy()

accumEdged = np.zeros(image.shape[:2], dtype="uint8")

# convert image to black and white

gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

# blur to remove some of the noise

blurred = cv2.GaussianBlur(gray, (5, 5), 0)

# get edges

edged = cv2.Canny(blurred, 75, 200)

accumEdged = cv2.bitwise\_or(accumEdged, edged)

# get contours

ctrs = cv2.findContours(edged.copy(), cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

ctrs = imutils.grab\_contours(ctrs)

docCnt = None

# ensure that at least one contour was found

if len(ctrs) > 0:

# sort the contours according to their size in

# descending order

ctrs = sorted(ctrs, key=cv2.contourArea, reverse=True)

# loop over the sorted contours

for c in ctrs:

# approximate the contour

peri = cv2.arcLength(c, True)

approx = cv2.approxPolyDP(c, 0.02 \* peri, True)

# if our approximated contour has four points,

# then we can assume we have found the paper

if len(approx) == 4:

docCnt = approx

break

paper = image.copy()

# apply Otsu's thresholding method to binarize the image

thresh = cv2.threshold(gray, 0, 255, cv2.THRESH\_BINARY\_INV | cv2.THRESH\_OTSU)[1]

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

# erode and dilate to remove some of the unnecessary detail

thresh = cv2.erode(thresh, kernel, iterations = iter)

thresh = cv2.dilate(thresh, kernel, iterations = iter)

# find contours in the thresholded image

ctrs = cv2.findContours(thresh.copy(), cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

ctrs = imutils.grab\_contours(ctrs)

return image, ctrs, paper, gray, edged, thresh

# plot image without axes

def display(img):

fig = plt.figure(figsize = (8,12))

plt.imshow(img)

plt.axis('off')

plt.show()

def sort\_contours(ctrs):

BB = [list(cv2.boundingRect(c)) for c in ctrs]

# choose tolerance for x, y coordinates of the bounding boxes to be binned together

tol = 0.7\*diam

# change x and y coordinates of bounding boxes to their corresponding bins

def sort(i):

S = sorted(BB, key = lambda x: x[i])

s = [b[i] for b in S]

m = s[0]

for b in S:

if m - tol < b[i] < m or m < b[i] < m + tol:

b[i] = m

elif b[i] > m + diam:

for e in s[s.index(m):]:

if e > m + diam:

m = e

break

return sorted(set(s))

# lists of of x and y coordinates

xs = sort(0)

ys = sort(1)

(ctrs, BB) = zip(\*sorted(zip(ctrs, BB), key = lambda b: b[1][1]\*len(image) + b[1][0]))

# return the list of sorted contours and bounding boxes

return ctrs, BB, xs, ys

def get\_circles():

questionCtrs = []

for c in ctrs:

(x, y, w, h) = cv2.boundingRect(c)

ar = w / float(h)

# in order to label the contour as a question, region

# should be sufficiently wide, sufficiently tall, and

# have an aspect ratio approximately equal to 1

# if w >= 20 and h >= 20 and 0.9 <= ar <= 1.1:

if diam\*0.8 <= w <= diam\*1.2 and 0.8 <= ar <= 1.2:

questionCtrs.append(c)

return questionCtrs

def get\_diameter():

boundingBoxes = [list(cv2.boundingRect(c)) for c in ctrs]

c = Counter([i[2] for i in boundingBoxes])

mode = c.most\_common(1)[0][0]

if mode > 1:

diam = mode

else:

diam = c.most\_common(2)[1][0]

return diam

def draw\_contours(questionCtrs):

color = (0, 255, 0)

i = 0

for q in range(len(questionCtrs)):

cv2.drawContours(paper, questionCtrs[q], -1, color, 3)

cv2.putText(paper, str(i), (boundingBoxes[q][0] + boundingBoxes[q][2]//2, boundingBoxes[q][1] + boundingBoxes[q][3]//2), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (255, 0, 0), 2)

i += 1

def get\_spacing():

def spacing(x):

space = []

coor = [b[x] for b in boundingBoxes]

for i in range(len(coor)-1):

c = coor[i+1] - coor[i]

if c > diam//2: space.append(c)

return sorted(list(set(space)))

spacingX = spacing(0)

spacingY = spacing(1)

# smallest x-serapation (between two adjacent dots in a letter)

m = min(spacingX)

c = 0

d1 = spacingX[0]

d2 = 0

d3 = 0

# for x in range(len(spacingX)):

# if spacingX[x+1] > spacingX[x]\*1.1:

# c += 1

# if d2 == 0: d2 = spacingX[x+1]

# if c == 2:

# d3 = spacingX[x+1]

# break

for x in spacingX:

if d2 == 0 and x > d1\*1.3:

d2 = x

if d2 > 0 and x > d2\*1.3:

d3 = x

break

linesV = []

prev = 0 # outside

linesV.append(min(xs) - (d2 - diam)/2)

for i in range(1, len(xs)):

diff = xs[i] - xs[i-1]

if i == 1 and d2\*0.9 < diff:

linesV.append(min(xs) - d2 - diam/2)

prev = 1

if d1\*0.8 < diff < d1\*1.2:

linesV.append(xs[i-1] + diam + (d1 - diam)/2)

prev = 1

elif d2\*0.8 < diff < d2\*1.1:

linesV.append(xs[i-1] + diam + (d2 - diam)/2)

prev = 0

elif d3\*0.9 < diff < d3\*1.1:

if prev == 1:

linesV.append(xs[i-1] + diam + (d2 - diam)/2)

linesV.append(xs[i-1] + d2 + diam + (d1 - diam)/2)

else:

linesV.append(xs[i-1] + diam + (d1 - diam)/2)

linesV.append(xs[i-1] + d1 + diam + (d2 - diam)/2)

elif d3\*1.1 < diff:

if prev == 1:

linesV.append(xs[i-1] + diam + (d2 - diam)/2)

linesV.append(xs[i-1] + d2 + diam + (d1 - diam)/2)

linesV.append(xs[i-1] + d3 + diam + (d2 - diam)/2)

# if d2 + d3 < diff:

# linesV.append(xs[i-1] + 2\*d3 - (d2 - diam)/2)

prev = 0

else:

linesV.append(xs[i-1] + diam + (d1 - diam)/2)

linesV.append(xs[i-1] + d1 + diam + (d2 - diam)/2)

linesV.append(xs[i-1] + d1 + d2 + diam + (d1 - diam)/2)

linesV.append(xs[i-1] + d1 + d3 + diam + (d2 - diam)/2)

# if d2 + d3 < diff:

# linesV.append(xs[i-1] + d1 + 2\*d3 - (d2 - diam)/2)

prev = 1

linesV.append(max(xs) + diam\*1.5)

if len(linesV)%2 == 0:

linesV.append(max(xs) + d2 + diam)

return linesV, d1, d2, d3, spacingX, spacingY

def display\_contours(figsize = (15,30), lines = False):

fig = plt.figure(figsize = figsize)

plt.rcParams['axes.grid'] = False

plt.rcParams['axes.spines.left'] = False

plt.axis('off')

plt.imshow(paper)

if lines:

for x in linesV:

plt.axvline(x)

plt.show()

def get\_letters(showID = False):

Bxs = list(boundingBoxes)

Bxs.append((100000, 0))

dots = [[]]

for y in sorted(list(set(spacingY))):

if y > 1.3\*diam:

minYD = y\*1.5

break

# get lines of dots

for b in range(len(Bxs)-1):

if Bxs[b][0] < Bxs[b+1][0]:

if showID: dots[-1].append((b, Bxs[b][0:2]))

else: dots[-1].append(Bxs[b][0])

else:

if abs(Bxs[b+1][1] - Bxs[b][1]) < minYD:

if showID: dots[-1].append((b, Bxs[b][0:2]))

else: dots[-1].append(Bxs[b][0])

dots.append([])

else:

if showID: dots[-1].append((b, Bxs[b][0:2]))

else: dots[-1].append(Bxs[b][0])

dots.append([])

if len(dots)%3 == 0 and not dots[-1]:

dots.append([])

# for d in dots: print(d)

letters = []

count = 0

for r in range(len(dots)):

if not dots[r]:

letters.append([0 for \_ in range(len(linesV)-1)])

continue

else:

letters.append([])

c = 0

i = 0

while i < len(linesV)-1:

if c < len(dots[r]):

if linesV[i] < dots[r][c] < linesV[i+1]:

letters[-1].append(1)

c += 1

else:

letters[-1].append(0)

else:

letters[-1].append(0)

i += 1

# print(letters[-1])

for l in range(len(letters)):

if l%3 == 0: print()

print(letters[l])

print()

return letters

def translate(letters):

alpha = {'a': '1', 'b': '13', 'c': '12', 'd': '124', 'e': '14', 'f': '123',

'g': '1234', 'h': '134', 'i': '23', 'j': '234', 'k': '15',

'l': '135', 'm': '125', 'n': '1245', 'o': '145', 'p': '1235',

'q': '12345', 'r': '1345', 's': '235', 't': '2345', 'u': '156',

'v': '1356', 'w': '2346', 'x': '1256', 'y': '12456', 'z': '1456',

'#': '2456', '^': '6', ',': '3', '.': '346', '\"': '356', '^': '26',

':': '34', '\'': '5'}

nums = {'a': '1', 'b': '2', 'c': '3', 'd': '4', 'e': '5', 'f': '6', 'g': '7', 'h': '8', 'i': '9', 'j': '0'}

braille = {v: k for k, v in alpha.items()}

letters = np.array([np.array(l) for l in letters])

ans = ''

for r in range(0, len(letters), 3):

for c in range(0, len(letters[0]), 2):

f = letters[r:r+3,c:c+2].flatten()

f = ''.join([str(i + 1) for i,d in enumerate(f) if d == 1])

if f == '6': f = '26'

if not f:

if ans[-1] != ' ': ans += ' '

elif f in braille.keys():

ans += braille[f]

else:

ans += '?'

if ans[-1] != ' ': ans += ' '

# replace numbers

def replace\_nums(m):

return nums.get(m.group('key'), m.group(0))

ans = re.sub('#(?P<key>[a-zA-Z])', replace\_nums, ans)

# capitalize

def capitalize(m):

return m.group(0).upper()[1]

ans = re.sub('\^(?P<key>[a-zA-Z])', capitalize, ans)

return ans

#url = 'https://i.imgur.com/NwLqmz2.jpg' # works

#url = 'https://i.imgur.com/4nC067a.jpg' # works

#url = 'https://i.imgur.com/osNCAx3.jpg' # works

# url = 'https://i.imgur.com/maU4r0t.jpg' # works

# url = 'https://i.imgur.com/OdyYxp1.jpg' # not works :< (because letters aren't aligned vertically)

#url = 'https://i.imgur.com/ttq5PzE.jpg' # works

# url = 'https://i.imgur.com/EjBz4nI.jpg' # works (iter = 0, width = 1500)

# url = 'https://i.imgur.com/4ggIni9.jpg' # not works :<

#url = 'https://i.imgur.com/UBqs60s.jpg' # works

# url = 'https://i.imgur.com/ihU7tFt.jpg' # works (iter = 0, width = 1500)

url = 'C:\\Users\\Acer\\Desktop\\de.png' # works (iter = 0, width = 1500)

image, ctrs, paper, gray, edged, thresh = get\_image(url, iter = 0, width = 1500)

diam = get\_diameter()

dotCtrs = get\_circles()

questionCtrs, boundingBoxes, xs, ys = sort\_contours(dotCtrs)

draw\_contours(questionCtrs)

linesV, d1, d2, d3, spacingX, spacingY = get\_spacing()

letters = get\_letters()

ans = translate(letters)

from textwrap import wrap

plt.axis('off')

io.imshow(image)

plt.show()

for l in wrap(ans, width = 80):

print(l)

myobj = gTTS(text=l, lang=language, slow=False)

myobj.save("welcome.mp3")

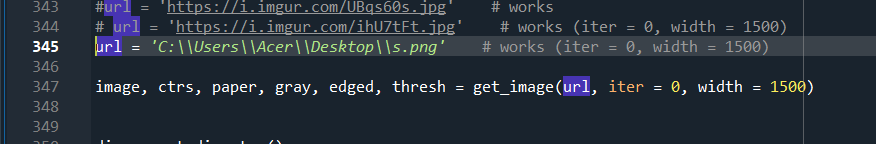
# Playing the converted file

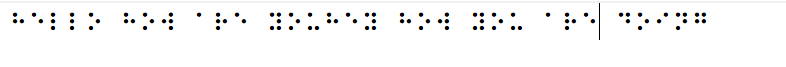
os.system("welcome.mp3")

**Appendix B: Screenshots**

a.Giving an image location/url as input

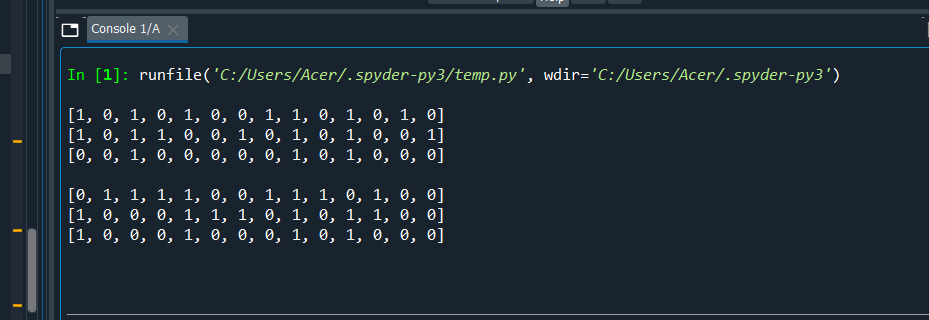
As the initial phase of testing we are giving the url/location of an image as input. Here the input is as follows.





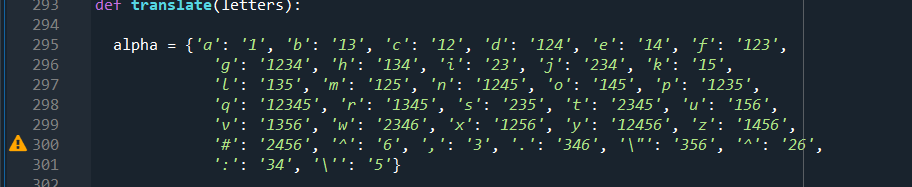
b. Producing a matrix of the window size 3x2.

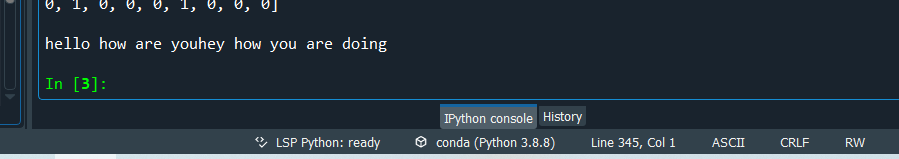
This is the next phase of the testing stage of our project. We are arriving at a matrix of window size 3x2.Then we are converting the position of one’s in the matrix into lists.



c. Arriving at the required text output by transforming the lists.

The next step is to transform the lists into letters using a dictionary. Then we are getting the required text output.





d. Getting the required speech output.

We are converting the text to speech using the gTTS library. The speech is saved as an audio file . The audio file is played by interacting with the operating system through os module.

