



# **Image to Speech System**

Hanci Quan

MSc Advanced Computer Science

Supervisor: Dr Bailin Deng

School of Computer Science and Informatics

Cardiff University

December 2022

## Abstract

The advent of the 5G era has led to artificial intelligence profoundly influencing our living environment and how we live. As a significant branch of artificial intelligence, research in computer vision has gradually tended towards deep learning. Optical character recognition has been deployed in various smart devices, such as document scanning, bank card number recognition, etc. However, the recognition environment is complex and must be placed within a defined box for recognition. In addition, character recognition can also be used in more complex scenarios, such as license plate detection and signage recognition.

The popularity of electronic devices has led to more and more people using mobile phones, computers, and other devices, which has led to an explosive growth in the number of people with visual impairment. So, to help extraordinary people who cannot read and look at newspapers, this paper investigates the detection and recognition of text in natural scenes through deep learning techniques. In order to help particular groups of people unable to read and watch newspapers, this paper investigates text detection and recognition in natural scenes using Tesseract-OCR, CRNN, PGNet and PaddleOCR through deep learning techniques. Furthermore, I designed and implemented an image-to-speech system. Because the detection environment is complex and variable, resulting in blurred and distorted images, the images are pre-processed, and the engine is optimized to improve recognition accuracy based on an OCR engine and a deep learning model. Moreover, the performance of different models in different scenarios was analyzed to improve the accuracy of text recognition.

This image-to-speech system is designed and implemented based on deep learning and OCR engines to meet the practical needs of visually impaired people. This project is a convenient way to implement image-to-speech, which can significantly help the visually impaired.

## Acknowledgements

Firstly, I want to thank my supervisor Dr Bailin Deng for explaining and communicating the problems and ideas I encountered during the dissertation. Also, very grateful to Dr Bailin Deng for his time during this period.

Finally, I would like to thank the school for all their support and help.

## Table of Contents

1. Introduction.....	1
2. Background.....	3
3. Literature Review.....	4
3.1 Image preprocessing.....	4
3.2 Text Recognition .....	5
4. Deep learning-based on text recognition (OCR) .....	7
4.1 Tesseract-OCR.....	7
4.2 CRNN.....	8
4.3 PGNet.....	9
4.4 PaddleOCR.....	11
5. Image Pre-Processing.....	13
5.1 Edge detection .....	13
5.2 Outline .....	14
5.3 Perspective Transformation.....	15
6. Design and Implementation .....	16
6.1 UI design .....	16
6.2 Get image function implementation .....	16
6.2.1 Local Camera.....	17
6.2.2 IP Camera.....	17
6.2.3 Local File upload .....	17
6.3 Tesseract.....	18
6.3.1 Implementation details.....	18
6.3.2 jTessBoxEditor training and test results .....	19
6.4 CRNN.....	20
6.4.1 Implementation details.....	20
6.4.2 Dataset training and test results .....	21
6.5 PGNet.....	22

6.5.1	Implementation details.....	22
6.5.2	Dataset training and test results .....	23
6.6	PaddleOCR.....	24
6.6.1	Implementation details.....	24
6.6.2	Dataset training and test results .....	25
6.7	Text-to-speech based on pyttsx3 .....	27
6.8	System functional testing .....	27
7.	Analysis.....	29
8.	Future work.....	30
9.	Conclusion .....	31
10.	Reflection .....	32
11.	Reference.....	34

# 1. Introduction

With the development of information technology, including big data, cloud computing and the Internet, the trend towards artificial intelligence is becoming more and more apparent. Ubiquitous networks and computing platforms such as graphics processors have driven the rapid development of artificial intelligence technologies. Technologies such as text recognition, text-to-speech, image classification, knowledge quizzes and driverless news are experiencing explosive growth, intending to enable people to have a more convenient and comfortable life through technology. OCR (Optical Character Recognition) text recognition is a branch of computer vision research under pattern recognition and artificial intelligence. It refers to the analysis and recognition of image files of text material to obtain text and layout information. Typically, image information is acquired and stored in image files by devices such as cameras and scanners. OCR software then reads, analyses, and extracts the strings in the image file through character recognition and converts them into a format acceptable to computers and understandable to humans. OCR technology is changing our lives. For example, mobile phones can scan and upload information such as ID cards and A4 paper to the Internet through a small app or Face ID when people need to pay bills.

Paper documents are the primary source for people to disseminate and access information. However, using a large number of paper documents to keep information is undoubtedly time-consuming and does not facilitate the dissemination of information. Over time, paper documents are difficult to store and easily damaged and can be inconvenient for people with visual impairments to read. A mobile device with a camera or a computer with a photo function that converts paper documents into electronic documents that can be read aloud is undoubtedly the best option for document preservation and reading.

Some visually impaired people report many obstacles in using electronic products, such as difficulties in payment and verification because they cannot see the screen of their mobile

phones. Despite the obstacles they encounter, more and more application developers have taken notice of them, and they have developed software such as document scanning. Based on the current situation, I intend to develop an image-to-speech system based on TTS (text-to-speech). The aim is to take a picture of an article or book with a camera, then use optical symbols to recognize the text on the picture and intelligently translate it into speech to be read aloud fluently to people with visual impairments. This system will be lighter, with a clean page design, good conversion, and fast recognition of medium to long texts.

This research will promote a better understanding of image-to-speech. Furthermore, it will give people a sense of how technology can make people's lives more accessible. Finally, this image-to-speech system will continue to be improved in the future.

## 2. Background

This report focuses on solving the problem of text reading for people with visual impairment. The World Health Organization says that over 2.2 billion people are visually impaired or blind [1]. The eyes are the first human organs to undergo ageing, and long hours of work and reading can lead to dryness, stinging, and even short periods of blurred vision. With the advancement of technology, mobile phones have become an indispensable part of people's lives. While people relax and unwind, they also neglect the most critical issue of eye health. The image-to-speech function makes reading books easier for people with visual impairment so that people with special needs can really "see" the outside world and communicate smoothly, all through the OCR image-to-speech function.

In this report, I have studied the specific method of recognizing text and propose improving the recognition accuracy. After research, it was discovered that the photographic environment and the device significantly impact text recognition. Therefore, I performed pre-processing algorithms on the images (edge detection, contour detection, binarization and image correction algorithms). Secondly, I needed to study and learn the theory of OCR technology, including how to use the Tesseract engine and utilize the deep neural network CRNN, Paddle and PGNet algorithms. I also compared and analyzed the advantages and disadvantages of each algorithm. Finally, I use a text-to-speech library to produce sounds. The overall front-end content will be displayed using a simple GUI to make it easier for the user.



### 3. Literature Review

Data show that more than 161 million people worldwide were visually impaired in 2002 [2], and the number of people with visual impairment is rapidly increasing. By 2010 [3] it had grown to approximately 300 million people.

In the literature [4], the authors state that it is possible to convert recognized text to speech through OCR and TTS modules. The authors use OCR components provided by Microsoft and have a recognition accuracy of 98%. However, no corresponding studies are proposed for various forms, such as surface anomalies, slope distribution, wrinkle distortion and incompleteness. In the literature[5], the author demonstrates the desire of visually impaired people to read to enrich their lives and presents three needs of visually impaired people: visual reading, tactile reading, and auditory reading. Based on the author's three needs, this paper will continue investigating the image-to-speech function in depth based on 'auditory reading'.

#### 3.1 Image preprocessing

Image pre-processing aims to eliminate irrelevant information from the image and recover valuable and accurate information. With the development of the times, pre-processing techniques are also receiving more and more attention from scholars.

In 1992 MF McNitt-Gray [6] acquired images with different features from multiple modalities. After experiments, 93% of the CT images were automatically rotated to the correct orientation, improving medical image processing. In the study of edge detection algorithm techniques for image pre-processing, R Main et al. [7] compare various edge detection techniques and analyze their performance. The authors highlight the importance of edge detection for image pre-processing; irrelevant information in the image is eliminated, interference and noise are filtered out, helpful information is recovered, new detectability about the image is enhanced, and data is simplified to the maximum. According to experimental results, Canny's edge detection algorithm outperforms others in noisy operation conditions. However, Canny's edge detection

method outperforms these operators in most cases, but Canny is the costliest of these algorithms.

Image binarization has long been a popular research topic. Binarization can change the grey scale value of pixels in an image to 0 or 255, resulting in a distinct black-and-white effect. In the literature[8], the authors propose new methods for the adaptive binarization of documents, that is, global and local threshold segmentation methods. Because of the source document's lighting variations, resolution, and poor quality, more than a single global threshold is required. It was found that the proposed algorithm is very adaptive, adapting the algorithm in time, thus analyzing the image significantly better than other algorithms.

ZHAI Jun-hai et al. [9], the authors illustrate the extraction methods, characteristics, and current state of development of image features and state that image feature extraction is the most critical step in image preprocessing. Four commonly used feature extraction methods (colour, texture, image algebra and image transform coefficient features) are presented separately. Finally, an outlook is given for combining transform coefficient and algebraic methods. Li Rong [10] shows that the wavelet transform LBP algorithm has a significantly better recognition rate for texture feature extraction than the traditional LBP algorithm.

## **3.2 Text Recognition**

Following the previous image pre-processing, text recognition technology is described below. Because text extraction is divided into two parts: detection and recognition, the detection aspect will be covered first.

Cong Yao [11] focuses on the problem of natural image text detection, noting that detecting text in different orientations is challenging. Based on novel text algorithms and the inclusion of various datasets, the art algorithm outperforms similar algorithms in detecting text in different orientations and performs significantly in complex scenarios. Kwang In Kim et al. [12] proposed a text detection method based on texture features, which effectively solves the problem of extracting text from complex backgrounds. A fast-directed text discovery network

was proposed by Xuebo Liu et al. [13]. It has undergone testing and performed 5% better than prior state-of-the-art results.

In order to recognize text, as early as 1994, J.J. Hull et al. [14] described a handwritten text-based recognition image database and divided a training and test set. After more than a decade of development in the literature[15], for previous studies, the authors improved the problem of word detection and recognition in natural images and, in some cases, outperformed traditional OCR engines. This literature [16] presents a top-down text recognition-based framework with significant improvements on two more difficult datasets.

Hewlett-Packard originally developed the Tesseract between 1985 and 1994 [17]. This article comprehensively describes Tesseract OCR, introducing what is new and different about the OCR engine[18]. In 2011, the literature [19] mentioned Tesseract OCR as the most accurate open-source image recognition engine today and implemented the Tesseract Wrapper available for the .Net platform. Deep learning development is expected to accelerate in the coming years, resulting in the recognition efficiency of Tesseract OCR being far lower than other text detection algorithms. In the literature[20], the author uses OpenCV to perform a series of preprocessing on images, thereby improving the English recognition rate of Tesseract-OCR.

In 2016 Baoguang[21] Shi et al. proposed a new network structure, CRNN, based on combining feature extraction, transcoding, etc., into the same network to achieve end-to-end text recognition and demonstrated that this algorithm outperforms most techniques. In 2018 Jiuxiang Gu et al. [22] provided an in-depth discussion of recent research, such as text detection, visual recognition, etc., concluding with the need to continue to improve the design of CNNs. Monica Jangpangi[23] solves handwriting recognition with an accuracy of 93%.

## 4. Deep learning-based on text recognition (OCR)

### 4.1 Tesseract-OCR

Tesseract is an open-source OCR [24] engine that currently supports the recognition of most of the world's text. Tesseract comes with its character library, and we can also continuously train our library so that the ability to convert images to text continues to grow. An important part of Tesseract is the analysis of related areas and character outlines in images. Where page layout analysis can capture areas of approximate text, contiguous domain analysis captures more precise blocks of text. The workflow of tesseract is shown in the figure 1 below.

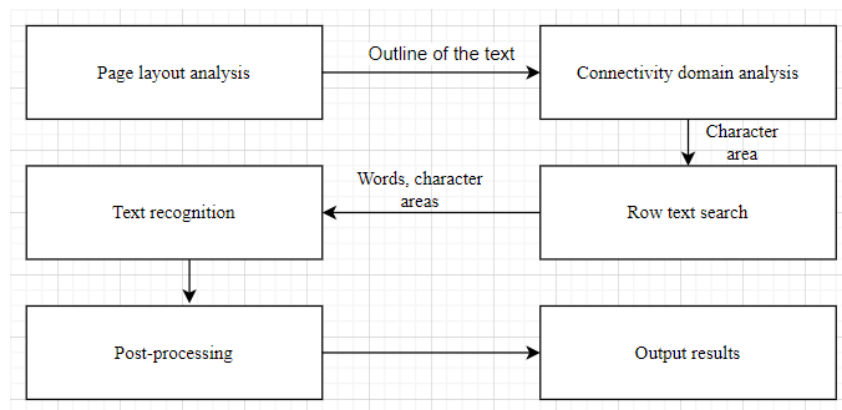


Figure 1 workflow of tesseract

Tesseract can be divided into four parts. The first part is to analyze the region of interest: detecting the region's outline and the character region's sub-outline and setting as block regions. Two methods of analyzing text lines: Fixed scenes segmented by character cells. The next is to split the scene according to proportional text by spaces or fuzzy space. If the first recognition is not good, tesseract will perform a second process to resolve the blurred text.

Tesseract uses text baselines to align lines of text, generally through the baseline, midline, ascending and descending lines to position the characters. The figure 2 clearly illustrates the Latin stylized baseline. Blue is the rising line, red is the middle line, purple is the baseline, and green is the falling line.

# TessBaseAPICreate 6090

Figure 2 Tesseract text baselines

The second part is to find block regions, detect character outlines and divide text and words into spaces. The third part is to find text lines and words. This step uses an adaptive classifier and performs word analysis twice. The final part is recognizing the text and identifying ambiguous spaces, stroke heights and lowercase letters.

## 4.2 CRNN

Deep learning methods have achieved great success in image recognition as artificial intelligence technology has advanced, owing to deep neural networks' powerful feature extraction ability. More network architectures with excellent performance have emerged. They are also gradually applied to optical character recognition, influenced by their advanced performance. After the classification of single characters and the detection and recognition of single characters, the classical CRNN[25] model was proposed. The CRNN model is the first algorithm that can be applied to the recognition of uncertain long character sequences. It can achieve end-to-end training and detection from the input image to the recognized text and adapt to most character recognition scenarios to achieve high recognition speed and detection accuracy. Figure 3 depicts the CRNN model's architecture, which consists of three layers: the convolutional layer, the recurrent layer, and the transcription layer.

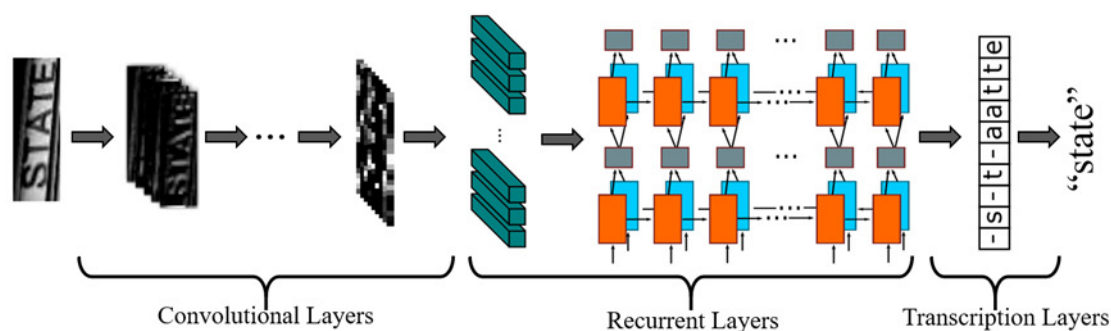


Figure 3 The architecture of the CRNN[25] model.

The CRNN model begins by scaling the input images, scaling all input images to the same height of 32 by default, and the width can be arbitrarily long. Then the convolutional layer performs feature extraction on the preprocessed images. The layer structure is similar to that of a VGG[26] network, consisting of convolutional layers, pooling layers, and batch operation layers. The extracted vectors are arranged on the feature map from left to right as input to the recursive layer. Each feature vector represents a feature on a specific image width. The width is set to one by default, representing a single pixel. Because the CRNN model scaled the input image to the same height, only features at a specific width are required.

In the recurrent layer, a bidirectional LSTM[27] neural network is used to predict the label distribution of each feature vector in the feature sequence. The CRNN model regards the sequence width as the LSTM's time steps since the LSTM requires a time dimension. Error feedback of the recurrent layer and conversion with feature sequences are the primary uses of the "Map-to-Sequence" custom network layer. The mistake can effectively be feedback from the cyclic layer to the convolutional layer, acting as a bridge between both.

The transcription layer combines and creates the final output from the feature sequences that the LSTM network predicted. It is mostly implemented via the CTC module. The CTC[28] model (Connectionist Temporal Classification), which can perform end-to-end training and output the results of variable-length sequences, is used to solve the alignment problem of input data with a given label. Character spacing issues and image distortion lead to different representations of the exact text for the input text images of natural scenes. Therefore, the CTC model is introduced to remove the spacing characters and duplicate characters from the results.

### **4.3 PGNet**

The PGNet[29] model introduces a fully convolutional Point Gathering Network (PGNet) to achieve real-time recognition of curved text in most natural images. Three major problems are solved in the PGNet model: (1) the high time-consuming problem of using NMS and RoI to determine text position and orientation. (2) the time-consuming and laborious problem of

character-level annotation of the training dataset. (3) the failure of using predefined rules to recognize unconventional text orientation. The PGNet model uses a single-stage text reader and a point-gathering operation based on multi-task learning. In Figure 4, its architecture is displayed.

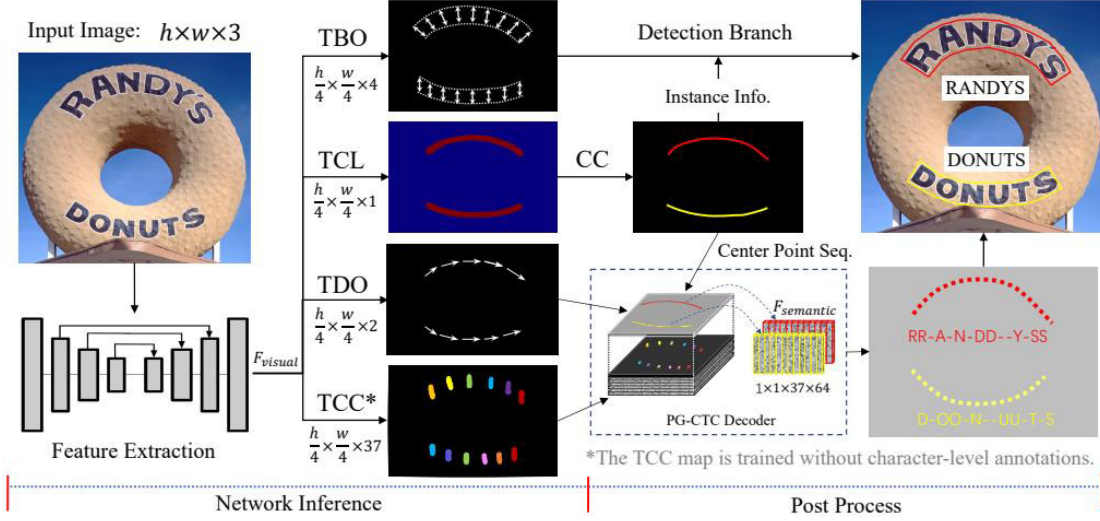


Figure 4 The architecture of the PGNet[29] model.

As shown in Figure 2, the image will first be input into a backbone network with FPN (Feature Pyramid Networks) for feature extraction and output of the feature map  $F_{\text{visual}}$ .  $F_{\text{visual}}$  is an original feature map that is used to predict the four feature maps: TCL, TBO, TDO, and TCC in a parallel multitasking manner. These four feature maps take up 1/4 of the input image's width and height. The scale-labelled feature maps that oversee TCL, TBO, and TDO during training are utilized to train the pixel-level TCC feature maps using the PG-CTC losses (PG-CTC loss solves the problem of requiring character-level labelling). Each text instance's centroid sequence is taken from the TCL feature map during inference. The TDO feature map is then utilized to rank the centroid sequence to recover the correct recognition order for the text. They are allowing PGNet to recognize text in unconventional text orientation. Based on the corresponding boundary offset information provided by the TBO feature map, PGNet achieves polygon detection for each text instance. Additionally, the high-level 2D TCC feature map may be serialized into a character classification probability sequence using the PG-CTC decoder, and the probability sequence can then be decoded to get the final text recognition result.

The training and inference stages of PGNet heavily rely on the PG (point gathering) process. It can help to get rid of the NMS and RoI operations of character-level annotation. In PGNet, the TCC is a feature map of 37 characters (37 channels) containing 26 letters, 10 Arabic digits, and one background class. The centroid of each forms the foundation of the PG operation. Based on the centroid of each text instance, the PG operation aggregates the character classification probability sequence from the TCC feature map. The entire PG operation process can be summed up as follows:

$$P_{\pi} = \text{gather}(TCC, \pi) \quad (1)$$

where a series of centroids with length  $N$  is called  $\pi = \{p_1, p_2, \dots, p_N\}$ .  $P_{\pi}$  is a sequence of character classification probabilities of size  $N \times 37$ . Using PG-CTC loss during the training phase makes it possible to train pixel-level TCC feature maps without the necessity for character set annotation. The NMS and RoI procedures are eliminated throughout the inference process thanks to the PG-CTC decoder's simplified end-to-end arbitrary shape text recognizer processes.

## 4.4 PaddleOCR

PaddleOCR[30] is a complete and powerful text recognition library built based on Baidu's paddle deep learning framework. It integrates the latest and most robust word recognition algorithm. It provides a set of rich, advanced, and practical OCR tool libraries which can meet general industrial production and actual needs. For general text recognition, PaddleOCR uses three stages of the text box detection model, the angle classification model, text, and recognition model to cascade to complete text detection of input images. The discrete text box's location in the picture is determined using the text detection model, which can select a regression-based EAST[31] model, segmentation-based DB[32] model, and regression and segmentation-based SAST[33] model. The angle classifier is used to adjust the angle of the text box to better identify the text in the detection box. It is an embedded model of PaddleOCR, which can be selected for or not used. The text recognition model is used to identify and output the text in the detected text box in a good order, which can select the CTC-based Rosetta model, CRNN model, STAR-Net models, the Attention-based RARE model, and the Transformer



based SRN[34] model.

PaddleOCR provides the training and deployment versions of the above text recognition package, which can be trained on individual models at each stage or used directly. It also allows users to train on personal data sets combined with official data sets, and after training, can be converted into an inference version for deployment to use. In addition, PaddleOCR provides several model versions with different CPU and GPU sizes, which can be flexibly selected according to the scenario.

## 5. Image Pre-Processing

### 5.1 Edge detection

The first pre-processing step is converting the image to a grey scale. As the edges of the image are very susceptible to noise, the second step is usually to filter the image to remove the noise to avoid detecting false edge information. Filtering aims to smooth out some non-edge areas with a soft texture to get a more accurate edge. The size of the filter is also variable, and the size of the Gaussian kernel plays an essential role in the effectiveness of edge detection. The larger the filter's kernel, the less sensitive the edge information is to noise. However, the larger the kernel, the greater the localization error of the edge detection. Generally, a 5 x 5 kernel is sufficient for most cases and implements edge detection using `cv2.Canny()`.

Incorrect edges may appear in the edge image due to noise or the image itself. Given the data, we need to define two thresholds. One for the high maximum value `maxVal` and one for the low minimum value `minVal`. Edges with a gradient value higher than or equal to `maxVal` are marked as strong edges. Edges with a gradient value between `maxVal` and `minVal` are marked as false edges. If the gradient value of edges is lower than `minVal`, it has considered suppressed edges. Figure 5

The figure 6 below displays the method's detection results after the edge extraction based on the Canny algorithm is finished.

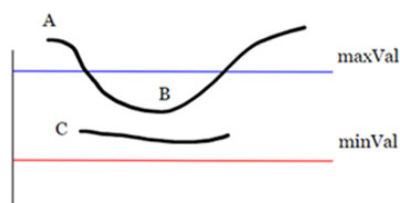


Figure 5 edge threshold

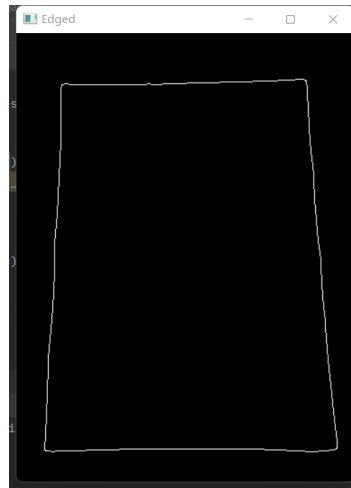


Figure 6 detection results

## 5.2 Outline

Once the edge detection is complete, the next step is contour detection, where we need to use the image after the edge detection. The outermost contour is the one with the most extended perimeter or the most significant area, where we can use an outer rectangle for sorting. The next step is to traverse the outline. First, we need to calculate the approximation of the outline, as the outline we observe may differ from the outline displayed by the computer. The outline may not be a rectangle and may consist of many points, so we need the `approxPolyDP` function to produce an approximate rectangle. In the end, we get a four-coordinate point, which means we get the outline of the text. The result of the contour detection is shown below. Figure 7

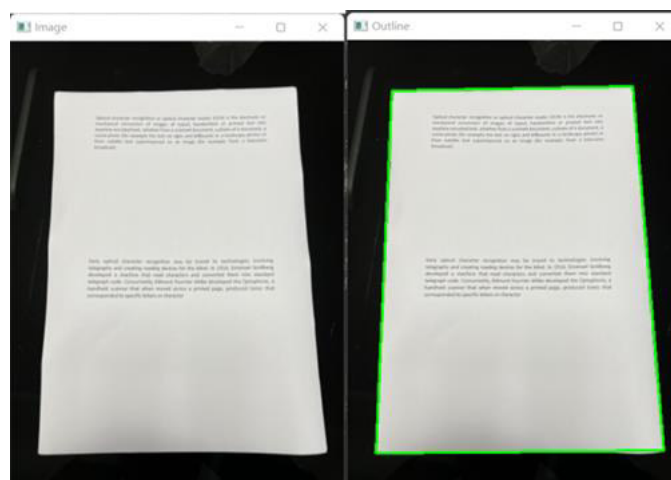


Figure 7 Result of the contour detection

### 5.3 Perspective Transformation

Once the outline has been detected, because the initial angle is random, we need to convert it to look like an electronic document, where we need to use a perspective transformation. Perspective transformation is projecting an image from one plane of view to another, called Projection Mapping. In simple terms, it changes a two-dimensional figure viewed obliquely into a two-dimensional image viewed from above. The perspective transformation is used quite often in computer vision, as the image captured by the computer is not regular.

As figure 8 shown below, after the above contour detection, we know the four ABCD coordinate points; we need to know this rectangle's h (height) and w (width). We can calculate each coordinate value's final conversion result with this information. The image is taken in two dimensions. Next, it will be mapped into three dimensions and converted back to two dimensions. This series completes the perspective transformation.

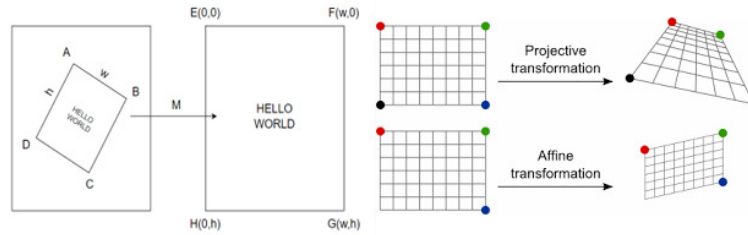


Figure 8 perspective transformation

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{31} & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad (2)$$

We also require binarization, where each point in the matrix has a separate RGB value and is shown in a different color, to provide a crisper image. By "binarizing," the most important portions of the image are preserved by having the RGB values of each point in the matrix either be (255, 255, 255) [white] or (0, 0, 0) [black].

## 6. Design and Implementation

### 6.1 UI design

The GUI links entire steps and provides a user-friendly interface so that the user does not have to type in filenames or use a command line interface to run the program. The GUI uses the PySimpleGUI library. PySimpleGUI is a Python package that collects all the major popular GUI modules. The most important aspect is the small amount of code and the simplicity of writing it, reducing the cognitive burden on the user. This project has a series of buttons designed to make it easier and faster for the user to operate the image-to-speech conversion. Figure 9 show image-to-speech project.

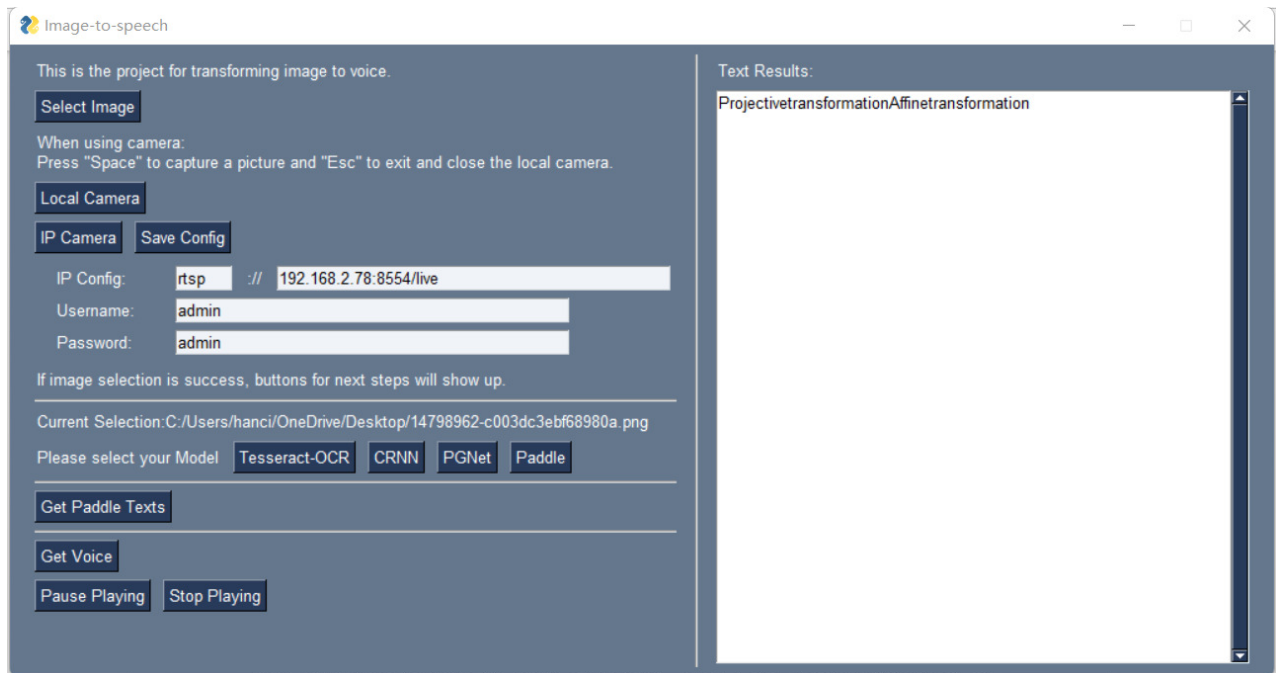


Figure 9 Image-to-speech project

### 6.2 Get image function implementation

There are three types to obtain pictures: local upload and camera photography, which users can use according to their needs. The image will be affected by the environment and light during shooting, likely affecting the recognition efficiency. Please upload as clear pictures as much as possible.

### 6.2.1 Local Camera

The first type is using the local camera either embedded in the computer or connected to the computer. If multiple cameras are connected, the first one in the list will be used. The python library for open-cv is used for this functionality. It will show another window with the camera's view, and users can capture the image by pressing "SPACE" and close/exit the window by pressing "ESC". The advantages of a local camera are that it is easy, fast and does not require a program to be installed. The disadvantages are bad pixels and low recognition accuracy.

### 6.2.2 IP Camera

The second way is using the IP camera figure 10, which is connected to the local area network. There are input forms for users to type in the camera's IP address and the username and password for authorization. These configurations can also be saved so that the user will not need to input the same values the next time running this program. Once the appropriate information has been loaded into the program, the same window will be opened as the local camera window, except the camera view is from the IP camera configured. This project uses software called "IP Camera Lite", which is a free IP camera system on apple store or google play that the user can set up within a few minutes via an IP address camera. The advantage is the high pixel count, which makes it easy for people to take pictures to identify text. The disadvantage is that it needs to configure and download mobile phone software.

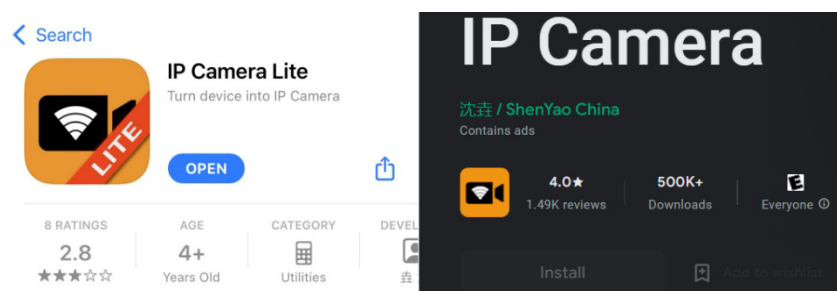


Figure 10 IP Camer in apple store and google play store

### 6.2.3 Local File upload

The last way is to upload image files locally. Image files must be in the format "JPG", "PNG"

and "GIF". Please note that the name of the image is in English and not in other languages, such as Chinese. The advantage is that it can recognize screenshots, but the disadvantage is that only three forms of files are currently supported.

## 6.3 Tesseract

### 6.3.1 Implementation details

The character recognition process is divided into analyzing related areas, finding blocks of areas, finding lines of text and words, and recognizing text. In this part of the implementation, we mainly input the image pre-processed photos to the Tesseract engine portal and process the images through the relevant APIs inside the Tesseract framework. The core code is shown figure11.

```
# recognize the text from the scanned image
def get_text_from_scan_image(scan_image_dir):
    preprocess = 'blur'
    image = cv2.imread(scan_image_dir + '/scan.jpg')
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    if preprocess == "thresh":
        gray = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY | cv2.THRESH_OTSU)[1]

    if preprocess == "blur":
        gray = cv2.medianBlur(gray, 3)

    filename = scan_image_dir + "/grey.png"
    cv2.imwrite(filename, gray)

    text = pytesseract.image_to_string(Image.open(filename))
    return text
```

Figure 11 Image\_to\_string

For more information about the pyTesseract image\_to\_string () function inside the code, the first and most straightforward way to implement it is to pass in 2 parameters. One is the image's file name, and the other is the type of language package used for recognition (the training language library is mentioned in the next chapter). In addition to passing in the image file name, the other method is to pass in the NumPy array of images, which can be used with modules such as pillow and OpenCV. First, do some pre-processing using pillow or OpenCV etc. and

then pass in the `image_to_string ()` for recognition. `image_to_string ()` can also be used to configure the command options for tesseract with the `config` parameter. All the optional parameters form a string to be passed to `config` using the following method:

```
text = ts.image_to_string(img, lang.config='--psm 6 --oem 1 --loglevel ALL')
```

### 6.3.2 jTessBoxEditor training and test results

In tesseract image text recognition, the jTessBoxEditor can be used for training to improve the recognition rate of image text. In tesseract's image text recognition, we use the official English font provided. However, these only sometimes meet our needs, so we can train with the jTessBoxEditor to improve the recognition rate of specific image text.

First, we need to prepare the font training images; we can use the tools that come with jTessBoxEditor to create TIF files. When all the photos generate a TIF file, we need to generate the .box file and execute the command as follows.

```
tesseract test_1.font.exp0.tif test_1.font.exp0 -l eng batch.nochop makebox
```

Next, we use the jTessBoxEditor to adjust the .box training file and correct the recognition characters and recognition boxes for each photo trained. This process is essential and determines the accuracy of the improved character recognition. Figure 12

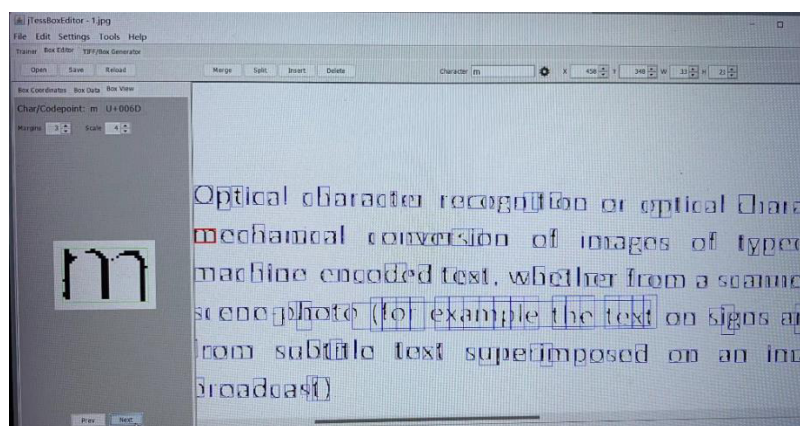


Figure 12 Recognition Box and Correction Recognition Characters

Then the font feature file is created, the training file and the character set file are generated, and the current directory generates the `test_1.traineddata` file is the same as the official font



library, only for the current type of text recognition. We can see figure 13 that the top part of the graph below shows the recognition accuracy after using the product, while the bottom part shows the accuracy using the original text library.

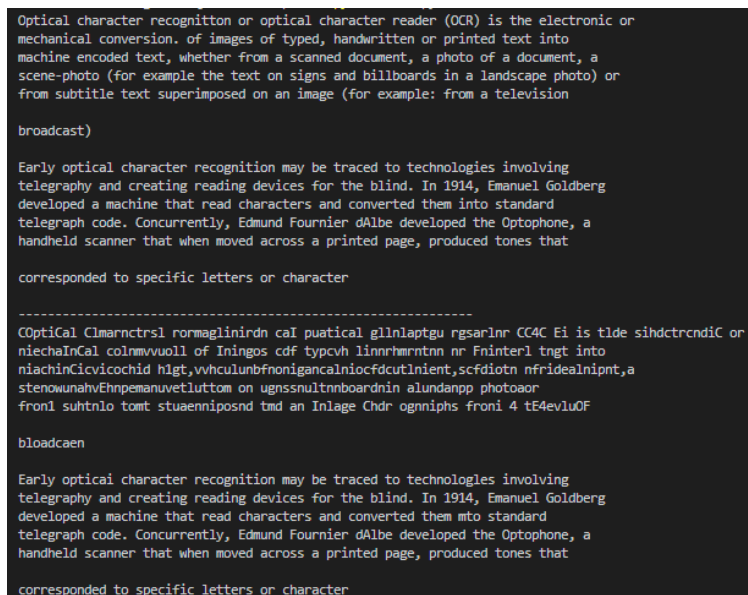


Figure 13 Optimization of training character library

The jTessBoxEditor tool is a basic training tool for samples; its function is to execute the above script commands automatically. In practice, imperfections exist, such as the inability to add psm parameters, and the program often crashes when generating shapes. We can see that after specific training, our conversion accuracy for image text has improved a lot.

## 6.4 CRNN

### 6.4.1 Implementation details

We refer to the source code of CRNN, use the PyTorch deep learning framework to build the whole network structure, and design the corresponding CTC loss function to train the network model based on the CRNN paper. The CTC loss function accurately describes the transformation path from the original output characters to the correct character sequence, which solves the problem of unaligned characters during training. The following formulation represents the CTC loss function:

$$p(l|x) = \sum_{\pi \in B^{-1}(l)} p(\pi|x) = \sum_{\pi \in B^{-1}(l)} \prod_{t=1}^T y_{\pi_t}^t \quad (3)$$

$$L_{CTC} = \sum_{\pi \in B^{-1}(l)} -\log(p(\pi|x)) \quad (4)$$

where the LSTM network's input and output characters are  $x$  and  $l$ , respectively.  $B$  is the transformation mapping from the sequence with spaces and repeated characters to the sequence of correctly ordered characters.  $\pi$  is one of the transformation paths.  $t$  are the moments of each output character under the current path  $\pi$ .  $y$  is the probability of the corresponding character output at the time  $t$  under the current path  $\pi$ . The CTC loss function uses the idea of dynamic programming and borrows the forward-backward algorithm of HMM to quickly traverse all transformation paths and accurately calculate the input-to-output loss to effectively train the network model. Table 1 displays the specifics of the CRNN model.

Model Training Parameter	Value
Epoch	100
Batch Size	10
Max Text Length	25
Architecture Backbone	ResNet
Layers	34
Loss	CTC Loss
Optimizer	Adam
Learning Rate	0.0005

Table 1 The implementation details of the CRNN model.

#### 6.4.2 Dataset training and test results

We train and test the model using the dataset used in the original CRNN paper. The experiment was equipped with Intel(R) Core (TM) i7-8750H CPU @ 2.20GHz, NVIDIA GeForce RTX 2060. The software platform is Pytorch1.7.1 framework and Python3.8. Finally, we deploy the trained model on equipment to detect images of multiple text recognition scenarios. The results of the detection are shown in Table 2.





Images	CRNN Results
	Future
	SLOW
	PAIN
	SPRI

Table 2 The detection results of the CRNN model

As shown in Table 2, the CRNN model can only detect characters of a single shape and direction in the rectangular box and cannot achieve arbitrarily shaped image text recognition. Furthermore, it has poor recognition of blurred, skewed character orientation images.

## 6.5 PGNet

### 6.5.1 Implementation details

We use the source code of PGNet for reference, build the whole network model with PyTorch deep learning framework, and design the corresponding PG-CTC loss function to train the network model according to the PGNet paper. Although the CTC loss function of CRNN solves the problem of inconsistent length between the source sequence and target sequence, it will convert the height of the feature map to 1, which is easily affected by background noise when recognizing curved text. Therefore, PGNet proposed the PG-CTC loss function to solve this problem. It can be formulated as follows:

$$L_{PG-CTC} = \sum_{i=1}^M CTC_{loss}(P_{\pi_i}, L_i) \quad (5)$$

where  $P_{\pi_i} \in \{\pi_1, \pi_2, \dots, \pi_M\}$  is the character classification probability sequence.  $L_i \in \{L_1, L_2, \dots, L_M\}$  is the corresponding transcription label. CTC\_loss is shown in equation (5). Table 3 displays the specifics of the PGNet model.

Model Training Parameter	Value
--------------------------	-------

Epoch	600
Batch Size	14
Max Text Length	50
Architecture Backbone	ResNet
Layers	50
Loss	PG-CTC Loss
Optimizer	Adam
Learning Rate	0.001

Table 3 The implementation details of the PGNet model.

## 6.5.2 Dataset training and test results

We trained the PGNet model using the hardware environment and software environment described in Section 6.4.2 and the dataset used in the PGNet paper. We tested its recognition performance with the following experimental results. Finally, we deploy the trained model on Raspberry Pi to detect images of multiple text recognition scenarios. Table 4 displays the detection's findings.

Images	PGNet Results
	
	



Table 4 The detection results of the PGNet model

The text area in the image and the text in the area of any shape can both be precisely identified by the PGNet model, as shown in Table 4. However, it is only for the task of text recognition. Its recognition effect is poor for many natural and document scenes, such as license plate recognition and card and bill information recognition.

## 6.6 PaddleOCR

### 6.6.1 Implementation details

We established the operating environment using the Paddle deep learning framework according to PaddleOCR's open-source code repository and the official tutorial and reproduced the entire text recognition process. Considering the speed and accuracy of camera detection during actual

deployment, we used the datasets officially provided by PaddleOCR to train the three-stage model with the lightest CPU version. Finally, the CPU version model was optimized and deployed on the camera to realize the text recognition function. The specific details of the PaddleOCR are shown in Table 5.

Model Training Parameter	Value
Epoch	600
Batch Size	14
Max Text Length	50
Detection Model	EAST, DB, SAST
Recognition Model	Rosetta, CRNN, STAR-Net, RARE, SRN
Optimizer	Adam
Learning Rate	0.001

Table 5 The implementation details of the PaddleOCR.

### 6.6.2 Dataset training and test results

We train the PGNet model using the hardware environment and software environment described in Section 6.4.2. Finally, we deploy the trained model on the camera to detect images of multiple text recognition scenarios. The results of the detection are shown in Table 6.




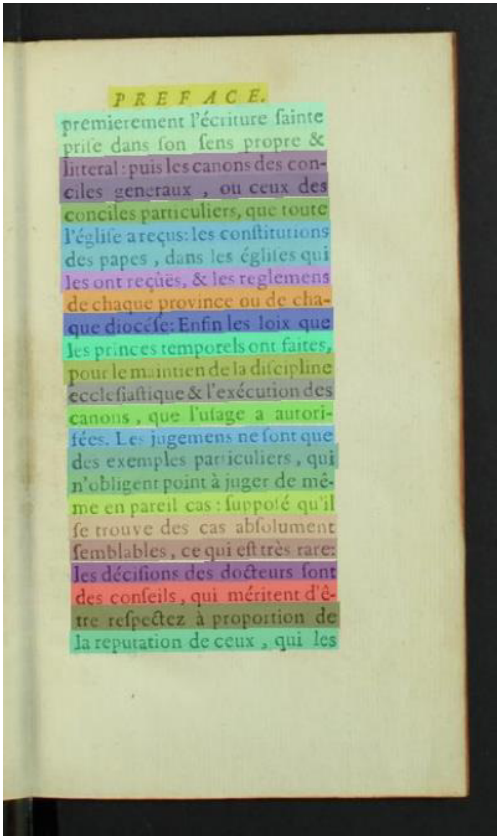
Images	PaddleOCR Results
	<p>1: Mairie du 1<sup>er</sup> 0.992  2: Palais du LOUVRE 0.992  3: les arts décoratifs 0.983  4: Musée du LOUVRE 0.990  5: Théâtre. 0.919  6: du PALAIS- ROVAL 0.893</p>
	<p>P K E F A 1 C . E .</p> <p>premierement l'écriture sainte  prise dans son sens propre &amp;  littéral: puis les canons des con-  ciles généraux , ou ceux des  conciles particuliers, que toute  l'église a reçus: les constitutions  des papes , dans les églises qui  les ont reçues, s les réglemens  de chaque province ou de cha-  que diocèse. Enfin les loix que  les princes temporels ont faites,  pour le maintien de la discipline  ecclésiastique &amp; l'exécution des  canons ; que l'usage a autori-  fés. Les jugemens ne font que  des exemples particuliers, qui  n'obligent point à juger de mé-  me en pareil cas : supposé qu'il  se trouve des cas absolument  semblables, ce qui est très rare:  les décisions des docteurs font  des conseils, qui méritent d'être  respectés à proportion de  la réputation de ceux , qui les</p>

Table 6 The detection results of the PaddleOCR

As shown in Table 6, the PaddleOCR can adopt the most of natural scenes and document scenes, such as license plate recognition, card and bill information recognition, and PDF and online document recognition. And it can achieve high recognition accuracy and meet the needs of practical applications.

## 6.7 Text-to-speech based on pyttsx3

pyttsx3 is a text-to-speech conversion library in Python. The texts-to-voice process applied the pyttsx3 library to transfer the texts we obtained from the second process and save it to the temporary directory. Then we used the Pygame module's sound mixer library to load the sound file we just saved and play it in another thread so that the main thread, which is the GUI, will not stop responding until the voice stopped. Two global variables and buttons are used to control the status of the currently playing voice. Processing another image will also stop the currently playing voice to void concurrently playing different voices.

## 6.8 System functional testing

After the method described in detail above, we have completed the development of the image-to-speech system. We have tested the system functions to ensure that the whole system can function properly and have tested the image upload, image taking, image-to-text, etc. The specific test table 7 is as follows.

Test content	Test Steps	Expected results	Test results	Note
Use local files uploads picture	Click on upload image, select the correct format image, upload successfully, and save it to a temporary picture	Successful image upload and display of image path	Pass	The correct format (PNG, JPG) needs to be uploaded
Use IP or local camera uploads picture	Click on Local Camera to connect to your local camera, take a picture and save it to a temporary picture	Open the local camera or IP camera normally and show the save path	Pass	No local configuration required; IP cameras require network address configuration



Text recognition	Select the model and click to recognize the text	Text recognition successful, results shown on the right	Pass	Differences in model and photo size can affect recognition wait times
Converting speech	If the text recognition is successful, the voice playback will be displayed, click on Play	Voice playback is successful and can be stop and paused	Pass	None

Table 7 System functional testing

Through extensive testing, this chapter provides an effective integration of image processing and several text recognition models proposed for the image-to-speech system, allowing users to take and upload images freely and to recognize and read aloud the text in the images.

## 7. Analysis

Tesseract-OCR has the highest recognition rate for neatly aligned text. However, when the light is dark, the text of the image is not aligned, and the paper is distorted. Tesseract cannot even recognize all the text. Tesseract-OCR is also poor at recognizing images with complex environments and has almost zero accuracies regarding shop names, billboards, and banners, so sometimes Tesseract is seriously polarized.

In this case, we have launched a study on the CRNN model. The CRNN model can achieve end-to-end detection of indeterminate length characters from images to character recognition. Moreover, its network structure is simple, fast, and accurate. However, it can only detect characters of a single shape and direction in the rectangular box and cannot achieve arbitrarily shaped image text recognition. Besides, the CRNN model requires the construction of special datasets during training, which has low universality. Therefore, we introduced the PGNet model. The PGNet model can adapt to natural scenes and achieve high text recognition accuracy and detection speed. It is the first text recognition algorithm with excellent performance that can be used in practice. However, it is only for text recognition and cannot meet the needs of real-time and accurate industrial production and practical applications. In many natural and document scenes, such as license plate recognition, card and bill information recognition, and PDF and online document recognition, the PGNet model is less effective in handling complex and diverse tasks and cannot meet the text detection requirements of this paper. Finally, we chose a mature and open-source text recognition library, PaddleOCR, to complete the text recognition function of this paper.

## 8. Future work

This report has done some research in image-to-speech and realized this function, using various models to find the best method. Although some expected results have been obtained, there are some shortcomings that need to be studied in the future:

In terms of text recognition accuracy, although PaddleOCR has a high recognition accuracy, it cannot accurately recognize all punctuation marks, so the break in sentences after the characters are recognized requires a lot of data support. Furthermore, I need to continue optimizing these frameworks, adding the Attention Model mechanism, and using a large amount of multi-scene data for training to improve recognition accuracy. For future research in the recognition field, the existing system should be improved to realize automatic recognition and broadcast of various special scripts, such as oracle bone script, Tangut script and other ancient Chinese scripts. We can improve the information collection and recording work based on deep learning of complex scene text recognition technology and use natural language processing technology to interpret the context and semantic understanding of the recognized ancient Chinese. In the future, this system will contribute to and help study ancient texts' literary and historical value.

The application scenario of image-to-speech needs further design. I have implemented the photo method by calling the phone's IP. However, this method is not very convenient for taking photos after my use because it cannot do these configurations easily for those special people. So, I need to improve my image uploading part and build personalized APPs, for example, by adding software to the phone where the user can hold the phone for content to speech, and the content inside can be shown immediately in the form of speech.

For the test of image-to-speech software, this paper only tests its functionality. The system's stability and code robustness have yet to be studied, so the code needs to be continuously improved in the future, and the pages and functions can be significantly improved.

## 9. Conclusion

This report designs an image upload module, a pre-processing image module, a text recognition module, and a text-to-speech module. Furthermore, based on this, each function is described in detail. We analyze the principles, advantages, and disadvantages of Tesseract-OCR, CRNN, PGNet and PaddleOCR methods and design a prototype of a PaddleOCR-based image-to-speech system.

The system first uploads the image to be converted using either a photo or a local readout, then pre-processes the uploaded image to highlight the text features and perform text recognition, and finally converts the recognized text into speech. In this report, we have carefully analyzed the principles of the four OCR algorithms mentioned above, studied the model structure of each part in depth, and summarized their technical advantages and applicable scenarios. In addition, based on the algorithms' advantages and disadvantages and considering the camera environment's variability and complexity, we have improved the OCR algorithms with some pre-processing using traditional image processing methods. The pre-processing includes contour detection, edge detection, perspective transformation and binarization, which can significantly improve the recognition accuracy of the image. We propose a complete and accurate conversion scheme from image to speech.

Although this paper has obtained good results for image-to-speech conversion, the task of scene text detection and recognition remains an arduous task. Objective factors such as scene complexity it has an impact on both the accuracy rate and the response time of recognition, so the text recognition model still needs to be improved.

## 10. Reflection

The first point is the literature review section. At first, I summarized all the literature in chronological order. However, this method of collating the literature could be more organized and provide a visual representation of the strengths and weaknesses. So, I have improved the literature review by presenting and analyzing the different areas chronologically to make it more intuitive for the reader.

The second point is the model selection of text recognition. The model selected at the beginning has a poor effect and low recognition efficiency, making it challenging to write the paper. Through communication with the supervisor, I extended the model and analyzed the recognition effect in different situations. Moreover, I compared and analyzed all the models, improving recognition accuracy.

The third point is image preprocessing. I have a relatively simple implementation for the image preprocessing part and have not studied the implementation of image processing in depth. In the future, different frameworks and image-processing algorithms need to be used to improve the results of image preprocessing.

The last point is whether this system is helpful for visually impaired people. Consider the condition of visually impaired people because everyone's impairment is different. The system must make higher requirements if the user's situation is serious. For example, deploying on mobile, setting shortcut keys, switching speech in real-time by taking pictures, speeding up the recognition of models and creating more concise interaction pages. Because of the current research needs, the UI interface design will have many buttons for model selection. If this system is deployed in the future, we need to specify the model with the best generality for text recognition and a more user-friendly speech.

Through this project, first as a brief accessibility developer, I know how difficult the life of

accessible people is, and I will continue to optimize this system and find the most suitable model. Secondly, I am weak in deep learning, need to understand the text recognition part of deep learning in more detail, and need to continue learning the underlying logic. In the literature review section for thesis writing, I learned how to sort out the research topic and summarize and analyze the previous research results. In my future study, I will continue to optimize and research this project and learn different areas based on deep learning methods.

## 11. Reference

- [1] <https://news.un.org/zh/story/2019/10/1043162>
- [2] Resnikoff, S., Pascolini, D., Etya'Ale, D., Kocur, I., Pararajasegaram, R., Pokharel, G.P. and Mariotti, S.P., 2004. Global data on visual impairment in the year 2002. *Bulletin of the world health organization*, 82(11), pp.844-851.
- [3] Pascolini, D. and Mariotti, S.P., 2012. Global estimates of visual impairment: 2010. *British Journal of Ophthalmology*, 96(5), pp.614-618.
- [4] Yu, T. & He, F. Him. (2012). Application of OCR components in smart readers. *Computer Knowledge and Technology* (14), 3385-3387.
- [5] Lin Ying. (2014). Research on Reading Problems of People with Visual Impairment. *Library Theory and Practice* (04), 22-25.
- [6] McNitt-Gray, M.F., Pietka, E. and Huang, H.K., 1992. Image preprocessing for a picture archiving and communication system. *Investigative radiology*, 27(7), pp.529-535.
- [7] Maini, R. and Aggarwal, H., 2009. Study and comparison of various image edge detection techniques. *International journal of image processing (IJIP)*, 3(1), pp.1-11.
- [8] Sauvola, J. and Pietikäinen, M., 2000. Adaptive document image binarization. *Pattern recognition*, 33(2), pp.225-236.
- [9] ZHAI Jun-hai, ZHAO Wen-xiu, WANG Xi-zhao. Research on the Image Feature Extraction[J]. *Journal of Hebei University (Natural Science Edition)*, 2009, 29(1): 106-112.
- [10] LI Rong XU Yan-hua. (2016). Research on the extraction algorithm of image feature based on visual information. *Electronic Design Engineering* (09), 188-190. doi:10.14022/j.cnki.dzsjgc.2016.09.056.
- [11] Yao, C., Zhang, X., Bai, X., Liu, W., Ma, Y. and Tu, Z., 2013. Rotation-invariant features for multi-oriented text detection in natural images. *PloS one*, 8(8), p.e70173.
- [12] Kim, K.I., Jung, K. and Kim, J.H., 2003. Texture-based approach for text detection in images using support vector machines and continuously adaptive mean shift algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(12), pp.1631-1639.
- [13] Liu, X., Liang, D., Yan, S., Chen, D., Qiao, Y. and Yan, J., 2018. Fots: Fast oriented text spotting with a unified network. In *Proceedings of the IEEE conference on computer vision*

and pattern recognition (pp. 5676-5685).

- [14]Hull, J.J., 1994. A database for handwritten text recognition research. *IEEE Transactions on pattern analysis and machine intelligence*, 16(5), pp.550-554.
- [15]Wang, K., Babenko, B. and Belongie, S., 2011, November. End-to-end scene text recognition. In 2011 International conference on computer vision (pp. 1457-1464). IEEE.
- [16]Mishra, A., Alahari, K. and Jawahar, C.V., 2012, June. Top-down and bottom-up cues for scene text recognition. In 2012 IEEE conference on computer vision and pattern recognition (pp. 2687-2694). IEEE.
- [17]<https://iopscience.iop.org/article/10.1088/17426596/1516/1/012017/pdf#:~:text=The%20Tesseract%20OCR%20engine%20was,%2B%2B%20to%201998%20%5B5%5D>.
- [18]Smith, R., 2007, September. An overview of the Tesseract OCR engine. In Ninth international conference on document analysis and recognition (ICDAR 2007) (Vol. 2, pp. 629-633). IEEE.
- [19]Chi Hao. (2011). Design and Implementation of a Wrapper Based on Tesseract OCR Engine. *Science and Technology Communication* (23), 199.
- [20]Guo Shiyi. (2019). Research on English Character Algorithm Based on OpenCV and Tesseract-OCR. *Computer Programming Skills and Maintenance* (06), 45-49. doi:10.16184/j.cnki.comprg.2019.06.017
- [21]Shi, B., Bai, X. and Yao, C., 2016. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE transactions on pattern analysis and machine intelligence*, 39(11), pp.2298-2304.
- [22]Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J. and Chen, T., 2018. Recent advances in convolutional neural networks. *Pattern recognition*, 77, pp.354-377.
- [23]Jangpangi, M., Kumar, S., Bhardwaj, D. *et al.* Handwriting Recognition Using Wasserstein Metric in Adversarial Learning. *SN COMPUT. SCI.* 4, 43 (2023). <https://doi.org/10.1007/s42979-022-01445-x>
- [24]<https://github.com/tesseract-ocr/tesseract>
- [25]Shi, B., Bai, X. and Yao, C., 2016. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE transactions on pattern analysis and machine intelligence*, 39(11), pp.2298-2304.



- [26]Simonyan, K. and Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
- [27]Hochreiter, S. and Schmidhuber, J., 1997. Long short-term memory. *Neural computation*, 9(8), pp.1735-1780.
- [28]Graves, A., Fernández, S., Gomez, F. and Schmidhuber, J., 2006, June. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning* (pp. 369-376).
- [29]Wang, P., Zhang, C., Qi, F., Liu, S., Zhang, X., Lyu, P., Han, J., Liu, J., Ding, E. and Shi, G., 2021, May. PGNET: Real-time arbitrarily-shaped text spotting with point gathering network. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 35, No. 4, pp. 2782-2790).
- [30]<https://github.com/PaddlePaddle/PaddleOCR>
- [31]Zhou, X., Yao, C., Wen, H., Wang, Y., Zhou, S., He, W. and Liang, J., 2017. East: an efficient and accurate scene text detector. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition* (pp. 5551-5560).
- [32]Liao, M., Wan, Z., Yao, C., Chen, K. and Bai, X., 2020, April. Real-time scene text detection with differentiable binarization. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 34, No. 07, pp. 11474-11481).
- [33]Wang, P., Zhang, C., Qi, F., Huang, Z., En, M., Han, J., Liu, J., Ding, E. and Shi, G., 2019, October. A single-shot arbitrarily-shaped text detector based on context attended multi-task learning. In *Proceedings of the 27th ACM international conference on multimedia* (pp. 1277-1285).
- [34]Yu, D., Li, X., Zhang, C., Liu, T., Han, J., Liu, J. and Ding, E., 2020. Towards accurate scene text recognition with semantic reasoning networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 12113-12122).