Design of text extraction function based on Raspberry Pi

Dong Fan Qian

September 2021

Student number:C2034950

School of Computer Science and Informatics

Cardiff University

Supervisor: Bailin Deng

Table of content

Declaration of statement----------1

Abstract----------3

Acknowledgement----------4

1. Introduction----------4

2. Background ----------8

2.1. Raspberry pi ----------8

2.2. Optical Character Recognition (OCR) ----------9

2.3. Text to Speech (TTS) ----------11

2.4. OpenCV----------12

2.5. GPIO----------13

3. Text Extraction Platform Design----------13

3.1. Programming language selection----------13

3.2. Raspberry pi modules selection----------13

3.2.1. Camera module----------13

3.2.2. Button module and T transfer board----------15

3.2.3. Platform Build----------15

3.3. Optical Character Recognition (OCR) ----------16

3.3.1. Use Face Detection to be familiar with OpenCV----------16

3.3.2. Tesseract----------20

3.3.3. Pi camera setting----------22

3.3.4. Realize the OCR function ----------23

3.3.5. OCR brief summary----------25

3.4. Text to Speech (TTS) ----------25

3.4.1. Speech engine: Espeak ----------25

3.4.2. Using Pyttsx3 to control Espeak engine----------28

3.4.3. TTS brief summary ----------29

4. Application by TTS and OCR----------29

4.1. Image capturing and image to text conversion----------29

4.2. Text to speech----------32

4.3. GPIO applications----------33

4.3.1. Led lights exercise----------34

4.3.2. Button module usage----------36

4.4. Application summary----------39

4.5. Improving accuracy of OCR----------41

5. Conclusion and expectation----------44

6. Reflections----------46

7. Reference----------49

1 Introduction

In the whole world, there are at least 2.2 billion people suffering the vision impairment. Between the 2.2 billion people, there are nearly 39 million people suffering from completely blind. We can't ignore the blind life, just image if you are blind, what's life become and what you need in your daily life. As an ordinary people, we use eyes to see the knowledge to input the words from your eyes. Without eyesight the blind use their ears to listen the world, sounds become their tool to think and study. Thanks for developing of technology, there are many advanced software or hardware appeared to help the visually impaired people read the paper without much difficulty. Our project is to help the people who suffer the visually impaired read the text from newspaper or other articles, so that they will have a better life experience. In this project, we want the tool to extract the text from printed articles or images and read the text out. Inspiring by Scanner, I decide to use raspberry pi to realize this function. I separate the whole process into three parts, from studying Text to Speech (TTS) to Optical Character Recognition (OCR) and the final assemble the OCR and TTS into the final device. In the TTS learning I will learn and decide the speech engine. In the OCR learning I will learn how to use camera catch the target that I wanted and transfer it to TTS. Finally use a physical switch to control the device to read the text out. The raspberry pi is initially connected to the internet through Wi-Fi, the related software and packages are installed by command line. The raspberry pi camera module is manually targeted straightly to the text. After the user press the button,

the image is captured and processed through raspberry pi 4b, then the words on the image will be said by the speak or headphone which plugged into raspberry pi by audio jack.

Abstract

With the development and usage of the electronic products, even the technologies of manufacturing screens are more advanced, we can still check there are many new products equip with 'Night Mode' to help the customers' eyes received less blue-ray, in addition to protect people's eyes. But according to research on Sciencealert[1], the half of the population on earth will be shorted-sighted. Especially in my motherland: China, with high pressure on work or study, people have no time to care their treasured eyes. We can also see that many smartphone/computers manufacture push out TTS (Text to speech) function to help the blind to use the phone or other electronic products, but in my perspective, blind people need another eye to help them to see the books or the restaurant menu. In the future, with the aging of the population, we face a more challenging situation to avoid visual problems.

My project is intending to use raspberry to build a device which can help the blind to capture the words and let the blind to listen these words even there is no internet. In my dissertation I will use a raspberry pi 4b, a camera module and one button module to realize the function that I mentioned.

Keyword: Raspberry pi, Camera, TTS, OCR, Python

Acknowledgement

I would like to thank my supervisor BaiLin Deng for his critical guidance and help in my tough time on the project.

Also thanks my friends in Cardiff University, who give me tons of supports on my life in Cardiff.

2 Background

2.1 Raspberry pi

The raspberry pi had been a famous product in Computer technology and education area. You can see it as a small pc, the different things between our normal pc are all its chips is already on board, just like a mini laptop mother board. Also, it doesn't run Windows system, so there are many operations we have to depend on command window on raspberry to realize what we want to do. It has been nearly 9 years until it first released to the customer.

At first, the meaning of inventing raspberry pi is to build a modern version of BBC Micro (a micro pc in 1981, it enlightened the British child to involve in Computer Science area, for this reason the first raspberry pi has two model called Model A and Model B). In 2012.2.29 the first version raspberry pi Model B released, but it didn't look like the BBC Micro, it didn't equip with any extensions such like mouse, keyboard, it just a board. At that time the pc was an expensive machine for majority of people, the raspberry pi depended on much lower price won the millions of potential customers. And the first generation received tons of good feedback from educational workers and tech-media. After on year, the Model A launched, in 2014 the Model A+ and Model B+ released with lower price. Toady the advanced version of raspberry pi is Model 4B (see figure 2.1),

Figure 2.1 Raspberry Pi 4B

if you want to use it you just need to follow the official website introduction then you will get a portable computer. On the raspberry pi web, you can but many kinds of extension module, such as camera, ultrasonic distance module to do everything you want (because of huge user basement, there are many tutorials that you can find to help you control different modules). In this project I will use one camera module one speaker module and one button to realize the text extract function.

2.2 Optical Character Recognition (OCR)

The technology of OCR is based on optical technologies and computer vision to let the computer to read the written words or printing character, and transfer these signals into a language that can be recognized by computer. For example, if an app wants you to add your credit card, you can let the camera focus on your card surface, the app will recognize the card number and other information, so that you don't have to input your card number by yourself. When you drive your car to a carpark, there is a camera by the side of the gate, when your car move into a specific distance between the camera, it will record your license plate number, in the whole process you have no chance to touch anything, it's quite important under the threat of coronavirus.

The concept of OCR is birthed in 1929, the German scientist Tausheck generate the concept and apply for the patent, but at that time there is no machine can realize his thought. In the 1960s, this technology is started to deep researched, at initial stage of research, the OCR can only recognize the number between 0-9. For example, in 1965s-1970s there are prototype can recognize the postal code and help post office to deliver the parcels into different regions, that's the reason why the postal code is always the stated form about your location.

OCR is a technology which equips indeterminacy, the rate of correct recognition about the character is just like an infinite approaching function, you know the approaching value but can't reach it. Because there are a lot of factors to stop the OCR reach the 100% accuracy. The custom of writing, the quality of printing, the level of scanner, the recognition approach and so on. Even there are much uncertainty, but we can see the overwhelming tendency about this technology, it brings great convenience in our daily life. Today there are four popular approaches to recognize the character:

1 Google's Tesseract engine

2 the third-party companies' OCR API

3 use the traditional way to extract the features of the characters, and distinguish them by different features

4 the CNN character recognition based on deep learning.

All of these approached have its own advantages and disadvantages. In my project I use the first way to realize the OCR function. The Google's Tesseract is

one of the open-source OCR engines and it has been supported by google for many years. The latest version of Tesseract support recognizes many kinds of languages. The accuracy on recognizing English and number is pretty and easy to use, but if I want to recognize the Chinese, I need to make further adjustments.

2.3 Text to speech (TTS)

TTS is one of the technologies that transfer texts to sound, the technology is one of parts of people communicate with pc. With the huge development potential, there are many areas uses TTS, such as Voice assistant (Siri, Bixby), the function helps the blind to use smartphone and computer. Also, it's not difficult to realize this function, I just need a speaker connects to my raspberry pi headphone jack. Nowadays, there are several TTS engine will help computer to transfer the text into voice.

1 Cepstral Text to speech

2 Festival Text to Speech.

3 Espeak Text to Speech

4 Google Text to Speech.

Cepstral Text to speech engine is one of the best offline speech engines on Raspberry pi. It equips with high quality of sound effects, it's a pity that the Cepstral Text to speech is not free and the sound resource is depended on the usage of the engine

Festival Text to Speech is a very old speech engine, in general it has decent performance but the sound is not smooth and like an old robot.

Espeak is a more modern speech engine than Festival Text to speech. They are all offline but the Espeak has a much better sound quality, and from the knowledge from the video that record the blind people life. I found that the TTS sound from their smartphone is much similar with the sound of Espeak. Using Espeak will drop down the learning price.

Google's Text to Speech engine is an online TTS engine. the theory of online TTS engine is the text that you scanned will send to Google's servers to generate the speech file which will be transferred by the internet. Because it depends on internet connection, I want my device can be used in every circumstance, even the sound quality is excellent (google translate), but I must give it up.

In my project I choose the Espeak engine, compare with Google Text to Speech, the Espeak doesn't need internet connection, I believe it's the key point that I give the Google TTS up, I need to make sure that the people who can use function anytime. If compare with the other two TTS engine, the Espeak has the best sound quality even it sounds like a robot.

2.4 OpenCV

The OpenCV is a critical open source, multi-platform computer vision library. It can be run on Linux, Windows, Android and Mac OS. It's formed from the C++ functions and categories, very light weight and high efficiency. Also, it supports Python to quote its function to realize the image process, the OpenCV just like the foundation of my project. With time going by, the OpenCV provides many

useful features such as text recognition, image recognition, creation of depth maps, and machine learning. Without OpenCV on y raspberry pi, I can do nothing.

2.5 GPIO

RPI.GPIO is one of the python modules, because the raspberry had already equipped with the GPIO pin on the board, so the GPIO had already built-in GPIO module, we can use it directly by python. In my project I will use button module to control the camera to capture images

3 Text Extraction Platform Design

3.1 programming language selection

The raspberry pi support three programming language: Python, Java, C++ and so on. In my semester in Cardiff University, it's the first time that I touch the Python programming, I find it's easy to understand and control if compare with the Java and C++. When I first time open the pi, I found the Raspbian (raspberry pi official system) had already equipped the python program software: Thonny. After choosing the Python as the main program language, I can find many resources about the how to set your pi and kinds of modules, the Python bring me much convenience when I learn pi knowledge. But using Python to control pi's module meets a problem, I can't adjust some specific parameters about camera module. Even so, use Python to control camera is sufficient for my project.

3.2 Raspberry pi modules selection

3.2.1 Camera module

In the raspberry pi hardware market, there are three version of camera that I can choose, Raspberry pi camera Module 2 (see figure 3.3), Raspberry pi high quality camera (see figure 3.1 figure 3.2), Raspberry pi camera Module 2 NoIR (see figure 3.4).

Figure 3.1

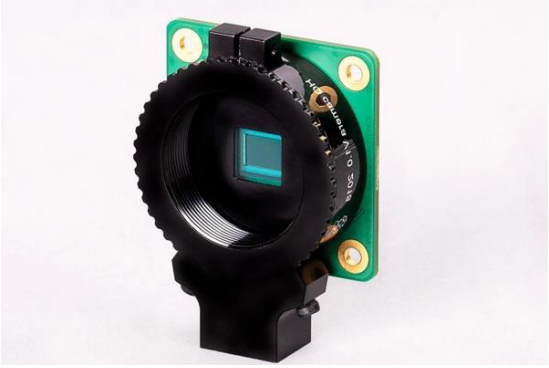


Figure 3.2

The high-quality camera likes a professional camera on DSLR, it has much better image quality than the other two especially in the dark light circumstances because of bigger sensor and changeable lens. But the lens and high quality module are more expensive (nearly 4 times price of Module 2 and Module 2 NoIR). In view of my project aims to help the blind have a more convenient life, so I give the high-quality camera module up.

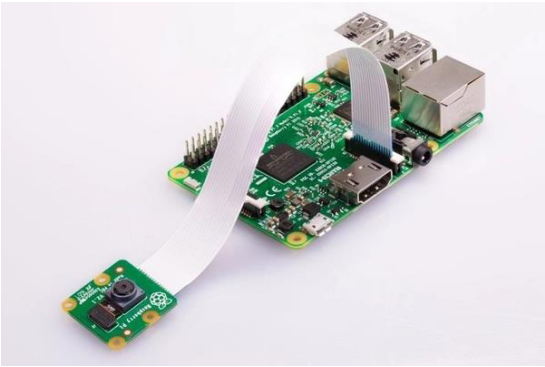


Figure 3.3

Figure 3.4

The difference between Module 2 (left) and Module 2 NoIR (right) is NoIR has better night performance but in daytime the image quality can't match the

Module 2, finally I choose the Module 2 as my pi camera to realize the image capturing.

3.2.2 Button module and T transfer board

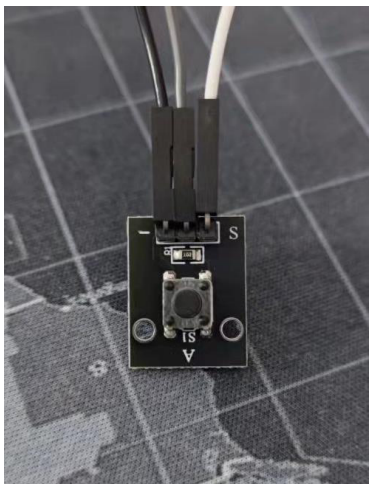


Figure 3.5

In my version of code, I use a keyboard switch to control the camera to shoot the image of words, but when I review the program, I found that button (see figure 3.5) is significant in my project, in my assumption, the user can use the button to control the pi to execute the OCR to TTS process.

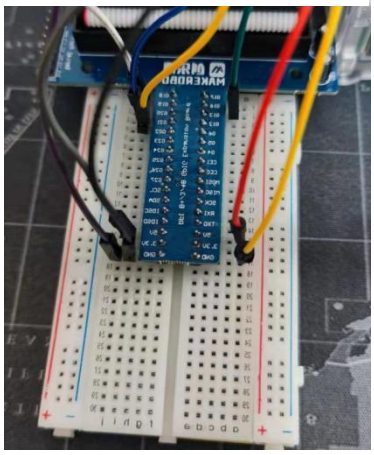


Figure 3.6

The T transfer board (see figure 3.6) is a critical module which can help pi to lengthen the GPIO slot. The GPIO slot like the USB port on pc, the raspberry can use GPIO to communicate with different modules like button and LED lights even ultrasonic distance sensor and temperature sensor. Based on the GPIO, the raspberry pi can build many kinds of projects.

3.2.3 Platform Build

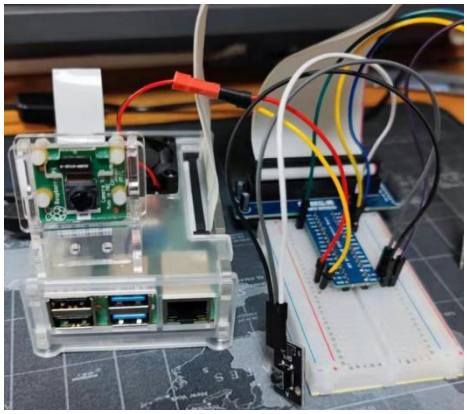


Figure 3.7

The final platform (see figure 3.7) equips with one pi camera Module 2, one button module and one T transfer board. For software, I had followed the official website introduction install the 32-bit pi system: Raspbian, it's the official recommended system for the new customer who want to build some projects by Raspberry Pi.

Figure 3.8

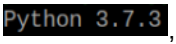
It's quite different with Windows (see figure 3.8), all of software and setting hide in the upper left corner little raspberry symbol (the function just like the Windows symbol).

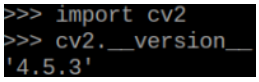
3.3 Optical Character Recognition (OCR)

When I want to program to control the camera to realize the OCR function, I found that I don't know how to make use of OpenCV to help me. After doing some research, I found that many experts recommend the new user who want to use OCR start from try to finish the Face Detection by the pi camera, in this process you will receive many new concepts about the OpenCV and many useful libraries

from the OpenCV to help you finish harder job in the future. So, I follow the expert's recommendation start to realize the face detection on raspberry pi.

3.3.1 Use face Detection to be familiar with OpenCV

First of all, according to OpenCV website introduction I finished installing the latest version of OpenCV. In pi command window , input 'python3', you will move into a python3 virtual environment to realize what you want python to do. You can also check your python3 version in the first line 

, after that in this environment input '**import cv2**' and 

'**cv2.__version__**'

After install the OpenCV, we will use a Haar+Cascade classifier to realize the face detection.

The classifier based on Haar+Cascade is an object detect method put forward from the thesis written by Paul Viola and Michael Jone in 2001. It's a machine learning approach which based on the cascade equation training by tons of specific pictures, the first thing we need to know is to understand how the face detection worked. Firstly, the algorithm needs a lot of positive examples that the pictures equip with people's face, and a lot of negative examples that has no people face to train the classifier (let it know the 'face'). Then we need to extract the features from these pictures. This is the Haar to extract features (see figure 3.9), just like conventional kernel, each feature coms from the sum of white box

minus the sum of the black box. In this process, in every kernel, we process all the possible size and location (for example, a 24x24 size slip window has nearly 160000 features). The hard thing is for every feature we have to calculate the sum of pixel from the white box and black box. For handling the giant calculation, the scientist quotes the concept: Integral images. The integral images technology simplifies the calculation of pixel, after that we just need to calculate the level of quantity of pixels and every calculation just involve 4 pixels.

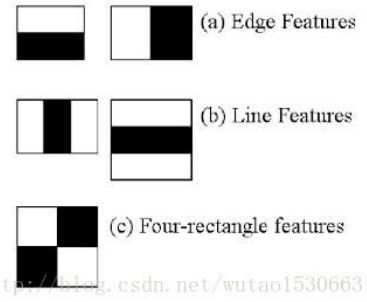


Figure 3.9

The majority of features calculation is inessential, for example (see figure 3.10), the first line shows two decent features, the first one seems based on the skin around the eyes is blacker than the nose and face, the second one shows the image around the eyes is darker than bridge of nose. Therefore, there is no meaning to use the same windows to detect the face and other part of our face. In order to get the best feature from the 160000 features, the Adaboost can help. So, we apply all of the features into the training pictures, for each feature, we need to find the best threshold value to detect face. Obviously, not every classifier is absolute correct, we just need to catch these features which have lowest error rate. At first, every picture will have the same value, after classifier, we increase the value of worry pictures and find these worry features in order to make a better classifier (in this process, we get

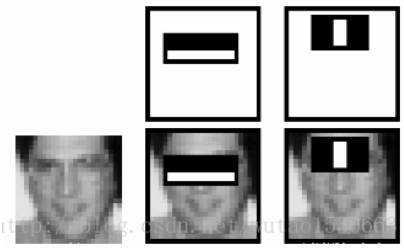


Figure 3.10

the more precise feature), finally when we reach the time requirements we stopped, this is the Adaboost. The final classifier is the sum of the weaker classifier, actually the reason why we call it weaker classifier because they reach their limit that can't classify pictures anymore, so we add them together became a stronger classifier. The classifier just need 200 features can reach nearly 95% accuracy in detection area, normally the final classifier will equip with nearly 6000 features. In fact, in one picture, the majority of places are not face, so we use Cascade classifier to separate these 6000 features into different level, just like a pyramid, the base level has few features, we can see the picture as many different windows, if the window cannot pass the base level, it will be abandoned, if it passes the base level into the second level, the window will face more different features. if the window passes all of the classifier, this window region is the face area.

For the convenience, the OpenCV had already equipped with many trained classifiers, such as face, eye and smile. When you install OpenCV, all of this thing is located in a specific fold. So, you can use these trained classifiers to detect the face.

```
import numpy as np
import cv2

faceCascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
```

Figure 3.11

The faceCascade is a OpenCV model for face, then I use camera setting in raspberry pi

```
cap = cv2.VideoCapture(0)
cap.set(3,640) # set Width
cap.set(4,480) # set Height
```

Figure 3.12

this step is to confirm the widow's size when we run the code. After setting camera, I start to use camera to capture live view and useHaar+Cascade classifier to detect which part of the view is face (see figure 3.14). This is the code and test result, after detection click ESC to quit the program.

```
import numpy as np
import cv2

faceCascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

cap = cv2.VideoCapture(0)
cap.set(3,640) # set Width
cap.set(4,480) # set Height

while True:
    ret, img = cap.read()
    #img = cv2.flip(img, -1)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = faceCascade.detectMultiScale(
        gray,

        scaleFactor=1.2,
        minNeighbors=5
        ,
        minSize=(20, 20)
    )

    for (x,y,w,h) in faces:
        cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = img[y:y+h, x:x+w]


    cv2.imshow('video',img)

    k = cv2.waitKey(30) & 0xff
    if k == 27: # press 'ESC' to quit
        break

cap.release()
cv2.destroyAllWindows()
```

Figure 3.13

After try face detection on raspberry pi, I know how to use OpenCV to detect face and other objects, I can use the OpenCV inbuilt face model to create a decent classifier to help me realize face detect. But there are some regrets, if I want raspberry pi to show each face's name by the side of the blue rectangle, I need more pictures about some specific people, and much time to train the camera. I believe if I have more time, I can do face recognition based on raspberry pi. The other important thing is I know how to set the camera parameters in this face detection process, the OpenCV is a useful tool for computer vision.

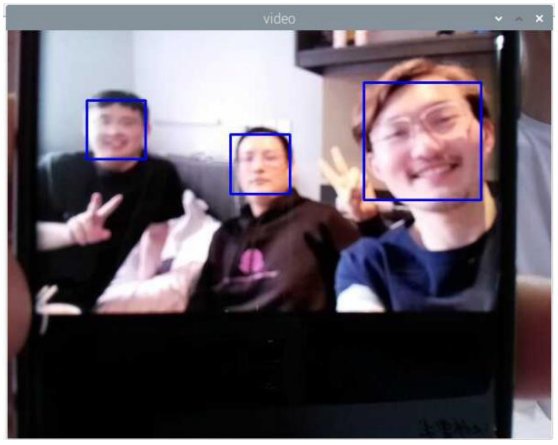


Figure 3.14

3.3.2 Tesseract

Tesseract is an open-source OCR engine, it had been developed in 1984-1994 by HP Bristol laboratory, at first, the Tesseract as OCR engine for improve the performance of HP scanner. In 2005, HP let the Tesseract became an open-source engine. Now the Tesseract is developing by Google Project. In our daily life, there are many regions are using Tesseract to improve the work efficiency. Tesseract recognizes and read the text present in your selected images. It can read all image types and make a large contribution in scanned documents area. Tesseract has Unicode (UTF-8) support and can recognize more than 100 languages. So, the Tesseract is the most critical tool for me to realize the OCR function.

```
sudo apt install tesseract-ocr --fix-missing
sudo apt install libtesseract-dev
sudo pip install pytesseract
```

Figure 3.15

Because the OpenCV has no Tesseract, I need to install the Tesseract engine on raspberry pi first (see figure 3.15. I can also download the Tesseract trained Chinese model to realize the Chinese recognition. In order to test the Tesseract engine, I use a short code and one article picture (see figure 3.16) to check the effect of Text-recognition. And the result is pretty good, except some characters in fancy style and special symbols can't be correct recognized. But all of test is just base on the downloaded pictures instead of the

```
from PIL import Image
from pytesseract import *
img0 = Image.open('/home/pi/Downloads/test.png')
mychars = image_to_string(img0,'eng').strip()
print(mychars)
```

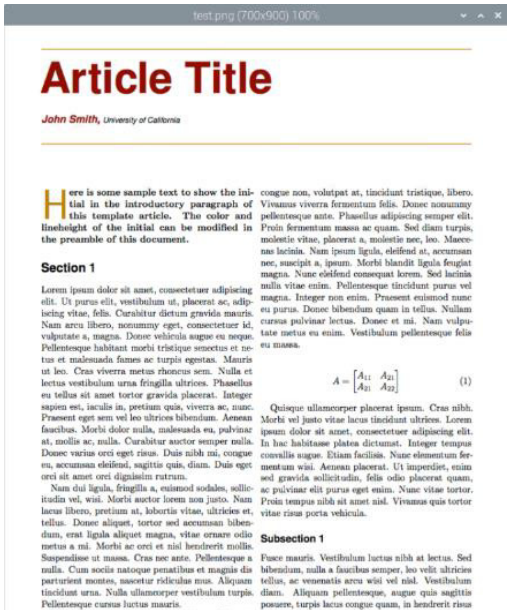


Figure 3.16

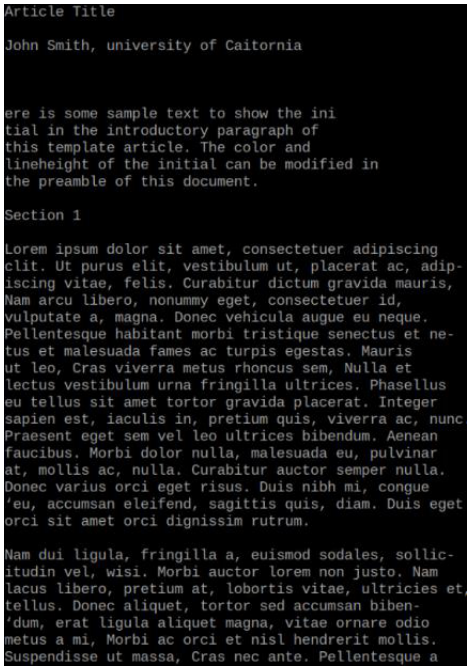


Figure 3.17

camera view. The next step is to find the way to connect the pi camera with the Tesseract engine.

This is the test result (see figure 3.17) from the no-preprocessing Tesseract engine.

After the face detection exercise, I understand some basic pi camera setting, but they are not sufficient when the camera faces a dark light circumstance.

3.3.3 Pi camera setting

At first, when I finished the face detection, I review the image quality is not good, because the Module 2 camera cannot realize the auto-fucus function, if I push the object or pictures too close by the camera, the pi can recognize nothing, after doing research, I find there is a circular ring built-in the camera module, if I turn it left, the focus point will be further, on the contrary the focus point will be closer. Finally, I find a decent focus point for camera to see the paper characters.

Of course, there are more setting by software. In the program I need to import PiRGBArray and PiCamera in my code to call the camera module. I can set the camera resolution and frame rate by code, the (640,480) is the resolution of

```
camera = PiCamera()
camera.resolution = (640, 480)
camera.framerate = 30
camera.iso=400
camera.shutter_speed=12000
camera.brightness=60
rawCapture = PiRGBArray(camera, size=(640, 480))
```

Figure 3.18

the camera and 30 is the framerate. In the last line **PiRGBArray()** (see figure 3.18). before this command the pi camera gives me an original RGB array image that has not been encoded. Because the frame hasn't been encoded so the **PiRGBArray()** will encode the frame in **Numpy array**. Comparing with other formats, the NumPy array can be used straightly by Raspberry pi instead of transferring to OpenCV format. So that we can have a better performance.

The iso is the same parameter with the film industry, is a parameter to measure the light sensitiveness by camera sensor, normally higher iso means better dark light performance, but in the viewer, you will see more noise in the image, so I try to use a low iso to make sure the image quality (higher image quality will help the Tesseract to extract the characters on the paper).

After test the different setting about the camera, I have to admit that if I want to use the camera in a dark light circumstance, the image quality is unsatisfactory because of the small sensor.

3.3.4 Realize the OCR function

After the Face Detection and Tesseract test, I understand how can I control camera and Tesseract, the first main target is to build a OCR function, I will combine camera and Tesseract engine to realize the OCR function. Except what I did before, if I want to use camera to shoot a picture and transfer it to Tesseract, I need do

more work. The same camera setting with the Face Recognition, I need a loop to call camera running until I click one button to capture the image, after that I will use the Tesseract to extract the text from the image and use one parameter to behalf of the text and print it out (see figure 3.19).

```
for frame in camera.capture_continuous(rawCapture, format="bgr", use_video_port=True):
    image = frame.array
    cv2.imshow("Frame", image)
    key = cv2.waitKey(1) & 0xFF

    rawCapture.truncate(0)

    if key == ord("s"):
        text = pytesseract.image_to_string(image)
        print(text)
        cv2.imshow("Frame", image)
        cv2.waitKey(0)
        break

cv2.destroyAllWindows()
```

Figure 3.19

The biggest difference are the codes in for loop. The function **capture_continuous()** has three parameters: **rawCapture**, **format** and **use_video_port**. The **rawCapture** is tell the camera we will read every frame in this whole process. It's an essential command (if you want to use camera). For **format** you can choose **bgr** or **rgb.** In my program, the reason why I select the **bgr** is for the convenience of OpenCV. According to the related articles, the OpenCV libraries will have a better performance in BGR than RGB mode. For the accuracy of OCR and Tesseract, I give the RGB up. For the parameter **use_video_port,** True means that we can use a window to have a live view about the camera sight. I can use the live view to check if the words aim the camera accurately. For the function of **cv2.waitkey()**, just as the meaning of the words:

wait key. When execute the function, the raspberry pi will detect the keyboard signal to compare it with the number in the brackets. If the key that you pressed is the same as the bracket's number, the program will move, **waitkey(0)** in this program means it will always wait for the key 's' pressed.

The **rawCapture.truncate(0)** will help the raspberry pi to clear the prepared frame in continued frame captures.

3.3.5 OCR brief summary

After trying Face Detection and OCR function, I am further familiar with the process that how to use OpenCV built-in library to control pi's camera and other components. In my results, I also found that I can do better in some areas, such as the dark lights performance, when it comes to dark light, the camera cannot supply decent images, even with the powerful Tesseract engine, the final result isn't perfect. I believe if I can do further adjustments on camera and Tesseract engine, I will have a better result. Also thanks the other pi user to help me find how to use OpenCV, Tesseract, I found many materials from the internet, they really give me tons of help.

3.4 Text to Speech (TTS)

Just like what I mentioned in Background, there are four speech engines for me to choose. When I consider this function will make sure the user can use it even offline, so I give the best Google speech engine. In the other three, I found many examples to listen their sound quality, even Espeak sounds like a robot, but has the clearest sound, I can do further adjustments to improve the sound quality.

3.4.1 Speech engine: Espeak

Espeak is a offline Text to speech engine, because it had been existed for a quite long time, it's convenient for raspberry to install. Before install the Espeak, I need to make sure the pi has no issues on sound function, in the command line, I input this command: `aplay /usr/share/sounds/alsa/*` , I heard a female voice for almost 10 seconds. That's prove the pi's sound function is ok. Next in the command I just need to input `sudo apt-get install espeak`, after execute this command, my raspberry can use Espeak command to speak words. To test the Espeak engine by using Espeak command. If I want the Espeak to say 'Text to speech engine is me', in the command line input `espeak 'text to speech engine is me'`, then I will hear a robotic voice says the words. But at first use the Espeak, I forget to add quotation by the side of the sentence, the Espeak can only recognize the first word.

Then I start to create my python program that speak words because I nee d Espeak to help me say the words from the OCR result. From the basic f unction count numbers. I will download num2words python package to hel p me transfer the numbers into strings, when it finishes the transfer proces

s (see figure 3.20 figure 3.21), the pi will read the number.

```python
from num2words import num2words
from subprocess import call

cmd_beg= 'espeak '
cmd_end= ' 2>/dev/null' # To dump the std errors to /dev/null


x = int(input("Enter a number: "))
count = num2words(x)+' Count Down Starts'
print(count)

#Replacing ' ' with '_' to identify words in the text entered
count = count.replace(' ', '_')
#Calls the Espeak TTS Engine to read aloud a Text
call([cmd_beg+count+cmd_end], shell=True)

#To do a Count Down
for i in range(x,-1,-1): # To count numbers down from the entered number till zero
    cmd=num2words(i) #To convert the Numbers to Text
    print(i)
    #Calls the Espeak TTS Engine to read aloud the Numbers
    call([cmd_beg+cmd+cmd_end], shell=True)
```

Figure 3.20

```
pi@raspberrypi:~/Downloads $ python3 count_number.py
Enter a number: 5
five Count Down Starts
5
4
3
2
1
0
```

Figure 3.21

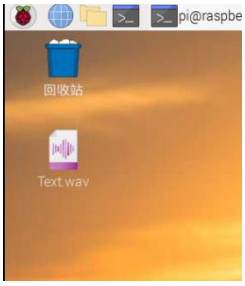
There is a critical command in 15th line, it's different way to use Espeak in command window (see figure 3.22). Except the number I will try to use py thon call Espeak to say words.

```python
from num2words import num2words
from subprocess import call


cmd_beg= 'espeak '
cmd_end= ' | aplay /home/pi/Desktop/Text.wav  2>/dev/null'
# To play back the stored .wav file and to dump the std errors to /dev/null
cmd_out= '--stdout > /home/pi/Desktop/Text.wav '
# To store the voice file

text = input("Enter the Text: ")
#
print(text)

#Replacing ' ' with '_' to identify words in the text entered
#text = text.replace('\n', ' ')
#print(text)
text = text.replace(' ', '_')

#Calls the Espeak TTS Engine to read aloud a Text
call([cmd_beg+cmd_out+text+cmd_end], shell=True)
```

Figure 3.22

The difference between the count number is the program supports to store the sound file, for example if I want Espeak to say 'hello world'. There is a .wav file appear on the pi desktop window you can re-listen the words

```
pi@raspberrypi:~/Downloads $ python3 text_speech.py
Enter the Text: hello world
hello world
```



Not only support English speaking, the Espeak packages also support more than 30 languages such as French, Spanish and American accent English by change the command,

```
espeak -ves 'hello world'
```

```
espeak -vfr 'hello world'
```

```
espeak -ven-us 'hello world'
```

the **-ves** means Spanish version, **-vfr** is French and **-en-us** is American accent. Of course, the Espeak also support to change the voice (7 male voices and 4 female voices).

This is for male voices `espeak -ven+m5 'hello world'` and this is for female voices `espeak -ven+f4 'hello world'`. There are many other adjustments such as speech speed and pause between specific word, but in my project, I don't have to use all of these functions. But I find it's still a little complicated to use Espeak. I did some research about if there is a upper platform can control these speech engine in raspberry pi, so I find Pyttsx3.

3.4.2 Using Pyttsx3 to control Espeak engine

If we see the speech engine as the works, the Pyttsx3 just like their boss, in different system the Pyttsx3 can call different speech engines to speak. On the raspberry pi we use Linux Espeak engine to speak words, but like the command in the 3.4.1, I must finish some complicated work. If we use Pyttsx3 to control the

Espeak will be much easier. When I first time touched the Pyttsx3 I did some python programs to check if it gives me more convenience.

Just like the first time I use Espeak, I try to use Pyttsx3 say some words, and I found

```
import pyttsx3
engine = pyttsx3.init()

engine.say('hello world')
engine.say('my dear friends')

engine.runAndWait()
```

the sound quality has no difference because of the Espeak engine, but I just need to use **engine.say(‘’)** one command to realize the function, it’s more convenient than the Espeak directly. The first thing if you want to use it is **import pyttsx3** from python3 packages, and do not forget to initialize the pyttsx3. Then the pyttsx3 allow you to use **engine.say(‘’)** to say the words. Because the pyttsx3 use Espeak, so we can use pyttsx3 to adjust the speak setting, we can set new speech rate by

```
rate = engine.getProperty('rate')
engine.setProperty('rate', rate - 50)
```

, the default rate is 200words in a minute. You can also set the volume and voice style. After using the pyttsx3, I find my old program can be easier, the next step is to use pyttsx3 to realize a simple project to help me familiar with the pyttsx3.

3.4.4 TTS brief summary

From the learning of TTS function, I found that if I want to use python to realize the speak function I not difficult, because there are many speech engines, and if I go browser their official website, I can find the detailed introduction to tell you how to install this type of speech engine on your device. And there are many experts article to help you how to use your speech engine. But I am not sure the decision of choosing the offline speech engine, because the online speech engines have much better sound quality and sounds like a human. From the

studying of pyttsx3, I found the code based on python is concise and easy to understand, I have the confidence to finish the combination of OCR and TTS.

4 Application by TTS and OCR

In this part, I will combine the OCR function and the TTS function together, the raspberry pi will use the pi camera to capture the words that you want to read and the Tesseract extract the words bring it to the Pyttsx3 to speak out.

4.1 Image capturing and image to text conversion

Just like my OCR studying in Chapter 3, I will use the camera to capture the article and use a variable to behalf of result of the Tesseract engine. the first thing imports the essential modules from the python3: pyttsx3, cv2, pytesseract to realize the function of TTS, and PiRGBArray, PiCamera to control the camera to realize the OCR function. Because the camera image quality is the critical part in my project, so I make some adjustments by pi camera setting in python3 (see figure 4.1), I change the iso to 400 for better dark light performance, 12000 as my camera shutter speed for letting lighter move into the camera sensor. 30 framerates have better image quality than 60fps because of the performance of the chips. So, when I run the program, there is a 640x480 window jump out, we can see the view of the camera and decide which parts of article that you want to transfer.

```
camera.resolution = (640, 480)
camera.framerate = 30
camera.iso=400
camera.shutter_speed=12000
camera.brightness=60
rawCapture = PiRGBArray(camera, size=(640, 480))
```

Figure 4.1

because in the OCR exercise I build a practical OCR program to

recognize the characters, so I can move the code into the final combination. If I want to realize the TTS, I just need to add pyttsx3 code in (see figure 4.2).

```
for frame in camera.capture_continuous(rawCapture, format="bgr", use_video_port=True):
    image = frame.array
    cv2.imshow("Frame", image)
    key = cv2.waitKey(1) & 0xFF

    rawCapture.truncate(0)

    if key == ord("s"):
        text = pytesseract.image_to_string(image)
        print(text)
        cv2.imshow("Frame", image)
        cv2.waitKey(0)
        break
cv2.destroyAllWindows()
```

Figure 4.2

1. Capture window (live view)

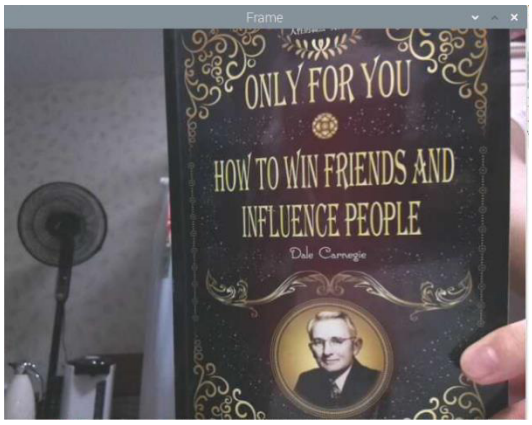


Figure 4.3

I will use one of pages of 《How to win friends and influence people》to show the process of OCR function.(See figure 4.3)

2. When you move your book and find the appropriate paragraphs that you want to use OCR, you need to aim the precisely and press 's' on your keyboard, then in the window you will find the

camera takes a picture of the page. Finally, the Tesseract will show the characters in this page on the command window.(see figure 4.4)

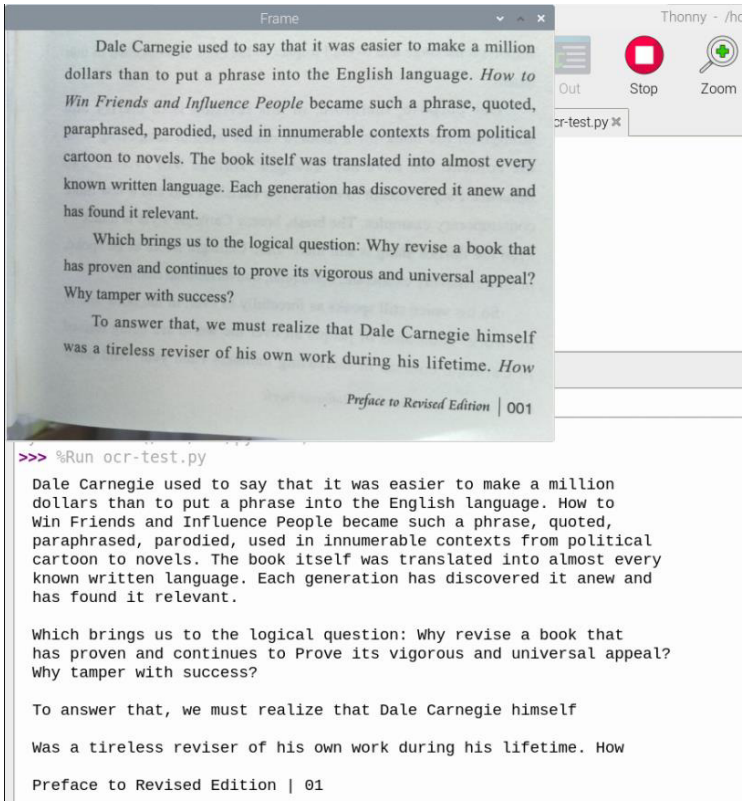


Figure 4.4

From the results, we can see that the accuracy is appropriate, and some of the fancy style characters can also be recognized (I believe the light circumstance plays a major role in the OCR process)

4.2 Text to speech

According to the TTS project based on pyttsx3, comparing with the original OCR code, I need to add few critical to call the pyttsx3. Firstly, **import pyttsx3** from the python3, and use variable **engine** to initialize the pyttsx3, because variable **text** is the result from the pytesseract (the pytesseract recognize the image's characters into string), follow the TTS project I will use **engine.say(text)** and **engine.runAndWait()** to realize one speech progress.(see figure 4.5) Because the raspberry pi has headphone jack, I can use a small 3.5mm headphone jack sound box connect raspberry pi to output the Espeak sound.

```
engine=pyttsx3.init()

for frame in camera.capture_continuous(rawCapture, format="bgr", use_video_port=True):
    image = frame.array
    cv2.imshow("Frame", image)
    key = cv2.waitKey(1) & 0xFF

    rawCapture.truncate(0)

    if key == ord("s"):
        text = pytesseract.image_to_string(image)
        print(text)
        engine.say(text)
        engine.runAndWait()
        cv2.imshow("Frame", image)
        cv2.waitKey(0)
        break
```

Figure 4.5

4.3 GPIO applications

The button module will simulate the reality situation. If the user wants to use this device to read something, he/she just need to click the small button, and the device will help him read the articles. But when I link the button with the raspberry by T transfer board, I find it's not easy to control the button. The first thing I need to understand is the GPIO ports on my raspberry pi 4b.

In the RPI.GPIO, there are two versions of pi's IO pins, the first is BOARD serial number, this is the original pi's pin number. When you use BOARD version pin, you can ignore the pi's version, because every raspberry pi has the same BOARD serial number, if you want to update or change raspberry pi, you don't have to change your connecter location. The other one is BCM serial number, it's the channel code of Boardcom Soc, if you want to use this pin version, you need to check the pi's BCM diagram. (See figure 4.6)

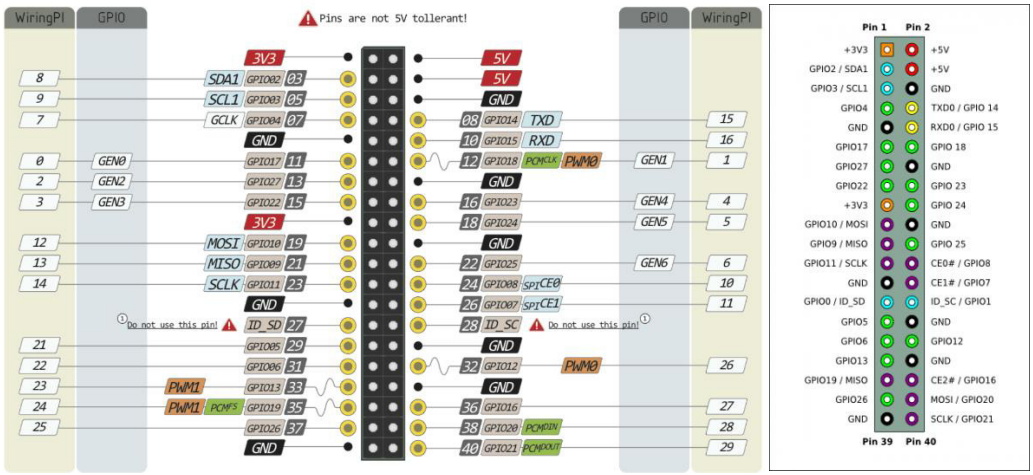


Figure 4.6

From the two-pin picture I check the pins position. In the software, you can select different version of pins by **GPIO.setmode(GPIO.BOARD)** or **GPIO.setmode(GPIO.BCM)**. Normally if you want use python to control the IO, the BCM is a better choice for user. Because if you use the BCM version, you will have more GPIO ports left, you can control more modules by different GPIO pins. After understanding the rule of GPIO, I will use a led test to familiar how to use the GPIO ports.

4.3.1 Led lights exercise

The LED module (see figure 4.7) has four separate pins control the red, green and blue lights and a GND pin to connect the ground. I will use BCM version to control the LED light. Firstly, import GPIO module by import RPi.GPIO as gpio, for controlling the light time, there is a critical module called **time (import time)** if don't use time module to control the LED, we can't see the flashing effect, then set the BCM version: **gpio.setmode(gpio.BCM).** I will use the 16th pin to connect led module:

Figure 4.7

gpio.setup(16,gpio.OUT), this means I will set the 16th pin as output mode to control the high or low electric level. Then use a while loop to control the LED realize a red light flashing. By using **time.sleep(0.1),** I can let the red interval time became 0.1s. So, we will check a quick flash by red light. The **gpio.output(R,gpio.HIGH/LOW)**, means the different electric level on 16th pin, when 'HIGH', the red light on, when 'LOW', the red light off. After trying to control the red light, I will try to flash the light that follow the regulation of red-green-blue-red. (see figure 4.8)

```
import RPi.GPIO as gpio
import time
R = 16
gpio.setwarnings(False)
gpio.setmode(gpio.BCM)
gpio.setup(R,gpio.OUT)
while 1:
        gpio.output(R,gpio.HIGH)
        time.sleep(0.1)
        gpio.output(R,gpio.LOW)
        time.sleep(0.1)
```

Figure 4.8

Just follow the single-color operation, we need to set green and blue color into different GPIO pin, in my mind, I use 20th pin as green color and 21 pin as blue color. And use an endless loop to let the red, green and blue alternant flashing. (See figure 4.9 figure 4.10)

```
while(1):
        gpio.output(R,gpio.HIGH)
        time.sleep(1)
        gpio.output(R,gpio.LOW)
        time.sleep(1)
        gpio.output(G,gpio.HIGH)
        time.sleep(1)
        gpio.output(G,gpio.LOW)
        time.sleep(1)
        gpio.output(B,gpio.HIGH)
        time.sleep(1)
        gpio.output(B,gpio.LOW)
        time.sleep(1)
```

Figure 4.9

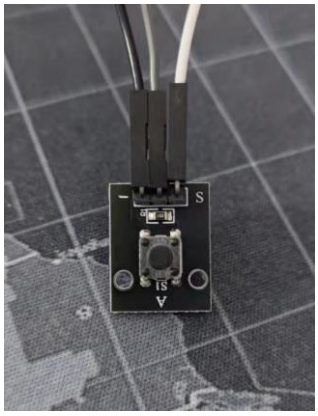
```
import time
import RPi.GPIO  as gpio

gpio.setmode(gpio.BCM)

R    = 16
G    = 20
B    = 21
```

Figure 4.10

4.3.2 button module usage

After trying this led light experiments, I have a basic impression on how to use GPIO to help me control the different modules, so next step I will study how to

use button module to control the camera in my project. In my opinion, I want to use the button module as a switch control the all processes.



the first thing that I need to do is to connect the right pin, there are three pins on this module, GND, VCC, and SIG. the significant pin is SIG, I need it to connect the GPIO pin that I want to control the button. This time I choose the 18th pin to control the button. After connecting, I need to know the theory.

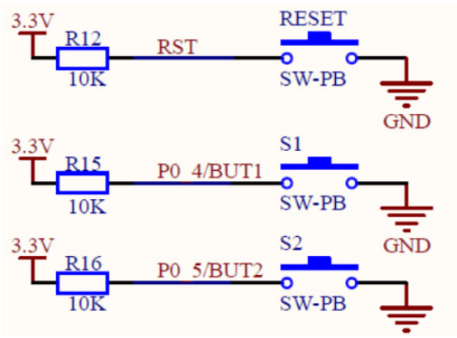


Figure 4.11

This is the button circuit diagram (see figure 4.11), for example, the button is S1. When we press the S1, the circuit will connect to the ground, the P04 is low level, if we loosen the S1, the circuit is stopped, the P04 will show the high-level. So, I can use this regulation to control the button by Python. The first we need to import GPIO: **import RPi.GPIO as GPIO** (use GPIO module to control the IO ports), and we set BCM mode: **gpio.setmode(gpio.BCM)**, then set 18th pin to control the button SIG pin. For button module, before it pressed, we don't know the input value in the circuit, so we prefer to set pull-up resistor and pull-down resistor to handle this problem. In this way we can set the default input value. This time we use pull-up resistor, then use an endless loop to test if the button is pressed. (See figure 4.12 figure 4.13)

```
GPIO.setup(channel, GPIO.IN, pull_up_down=GPIO.PUD_UP)
  # or
GPIO.setup(channel, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
```

```python
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
KEY = 18
GPIO.setup(KEY,GPIO.IN,GPIO.PUD_UP)
while 1:
        time.sleep(0.05)
        if(GPIO.input(KEY)==0):
             print ("key press")
```

Figure 4.12

But when I press the button one time, there are three or four 'key press' appear. Even just touch the button there is many 'key press' out. I did some researches to figure it out.

```
>>> %Run button.py
 key press
 key press
 key press
```

Figure 4.13

Because the button has mechanical dither (see figure 4.14), so when I touch it or press it, the switch will think it has been pressed for many times.

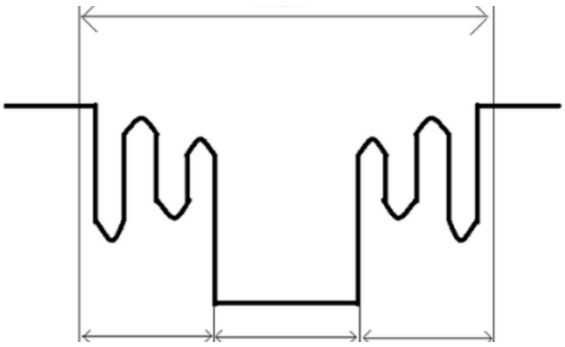


Figure 4.14

Normally the time of mechanic dither last for 10~20ms (depend on the quality of the button). In the 10~20ms, the raspberry pi can execute the program for thousands of times, so I need to find a way to avoid the mechanic dither. I will use time module to avoid it. **Time.sleep(0.02)** means that the program will wait me (see figure 4.15)

```python
while 1:
    if GPIO.input(KEY)==0:
       time.sleep(0.02)
       if(GPIO.input(KEY)==0):
          while(GPIO.input(KEY)==0):
              pass
          print('key press')
```

Figure 4.15

for 20ms, then I will re-test the 18th voltage, if it is low-level, the program will move into a while loop until I loosen the button. In this way, I can avoid the mechanic dither. This time when I press the button, only has one 'key press' appear. (See figure 4.16)

```
>>> %Run button1.py
 key press
```

Figure 4.16

After that I will add codes to control the camera in my program. When I press the button there is nothing happened, when I loosen the button, the program will follow the procedures in TTS and OCR to extract words and read them out. (See figure 4.17)

```
while 1:
    if GPIO.input(KEY)==0:
       time.sleep(0.02)
       if(GPIO.input(KEY)==0):
          while(GPIO.input(KEY)==0):
               pass
          for frame in camera.capture_continuous(rawCapture, format="bgr", use_video_port=True):
            image = frame.array
            cv2.imshow("Frame", image)
            key = cv2.waitKey(1) & 0xFF
            rawCapture.truncate(0)

            if GPIO.input(KEY)==0:
               time.sleep(0.02)
               if(GPIO.input(KEY)==0):
                  while(GPIO.input(KEY)==0):
                   pass
                  text = pytesseract.image_to_string(image)
                  print(text)
                  engine.say(text)
                  engine.runAndWait()
                  cv2.imshow("Frame", image)
                  break

          cv2.destroyAllWindows()
```

Figure 4.17

The first time press the button will call the camera, if you press the button again, the camera will capture the image and transfer it to OCR and TTS engine to realize the text extraction function. That's my goal

4.4 Application summary

In this project designing, it's my first time touch the raspberry pi 4b. Thanks me supervisor gives me tons of guidance when I confused to choosing the project. From selecting one of the raspberry pi versions to determine which extensions that I want to use in my project, I meet many obstacles. In choosing the raspberry pi, there are many price levels on official website, after doing researches, there are many raspberry pi users recommend new user to but the latest raspberry pi if you

can accept the price, because the newer version means support more things and have more powerful performance, so I choose the raspberry pi 4b. After discussing with my supervisor, I choose to realize a text recognition based on raspberry pi, at first, I am worried about if I can finish it in time, because even I learned much knowledge about python in my postgraduate semester, but the raspberry pi is a brand-new area for me. At the beginning of project, I naively think I just need a camera as my extension, but when I finished the combination of OCR and TTS, I believe I need have a button to control nearly everything, because the user won't bring a laptop and a raspberry pi out and press a key to help him/her to read the articles. So, I add a button module to help the user to control this function. I found that it's a self-improvement processing, when I finish the OCR function, I was many exercises, from the face detection to use the Tesseract to extract the text in images, I found that there are many useful build-in functions have already been installed by raspberry pi, if I want to use them, I just need to call these packages/libraries by python3. The tesseract is a useful text extract engine, thanks OpenCV has trained tesseract decently, I don't have to do further adjustments about the tesseract and can get a better recognition effect. But when I did the test in the daytime and night (even open all lights in my room), there is a non-negligible difference about the recognition accuracy, even I do some researches about how to enhance the camera dark light performance, such as pull up the ISO, slow down the shutter speed to let lighter move into the sensor, increase the brightness of the camera. But all these operations can't make up for the

deficiencies of the small sensor, so I need to make sure the circumstance light to help the tesseract to recognize the text. I had the decision of use the photo resistance to detect the luminance of the circumstance and add a fill-in light to simulate the daytime situation. When the photo resistance detects the luminance is too low will turn the fill-in light on. But it's difficult for me to realize a decent effect on this function. It's also the first time I use many exercises to learn how to realize the OCR and familiar with the OpenCV. After successfully finish the OCR test, I get more confidence on TTS function, in the background chapter, I said there are four speech engines that I can select. From the best offline and free speech engine: Espeak to decent online speech engine Google TTS, if I don't have to concern about the internet condition, I will choose the Google TTS. The Espeak sound is more robotic, I will try to do further adjustments on the voice. Thanks to the developments of OpenCV, I can use pyttsx3 to call Espeak engine in one command: **engine.say()**.

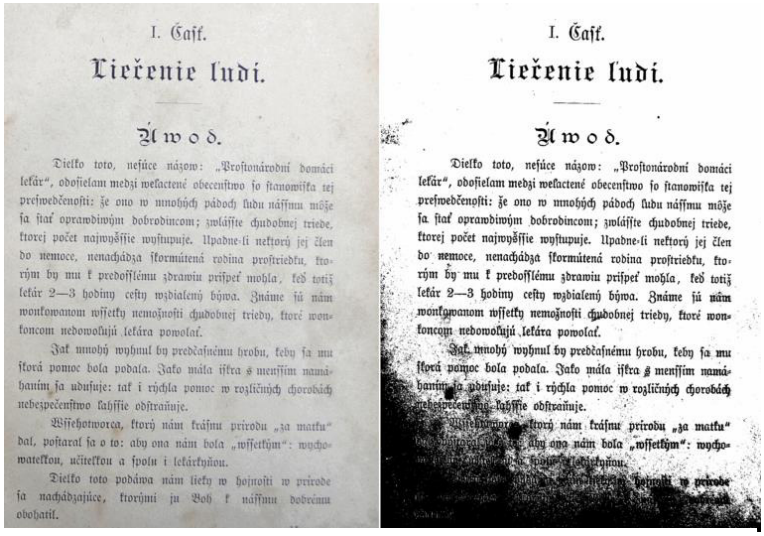
Finally, I combine the OCR and TTS together, because I had finished many small exercises, some of them have decent codes so that I can assemble them and did little changes. In the process of combination, the critical issue is adding the button module. The GPIO is a good function for raspberry pi to use more extensions such as ultrasonic sensor to detect the distance and so on. According to different T transfer board protocols, the usage of pin is different, for me the T trans board help me push all the GPIO pin more forward than the GND and VCC, so I can select the io port easily. In this processing of familiar with the new hardware, I also

use small works to train me how to use this tool. Of course, there are many articles show tutorial to help me understand how to use GPIO if I have no experiences. Because of I make mistake of the voltage level about my button, in the exercise of testing key press, I cannot stop the **print ('key press'),** until I found that that when the button hasn't been pressed the voltage level is HIGH if you press it, the voltage level is LOW. For handling this problem, I use one night to test the T transfer board and the way of using GPIO. Finally, this project is finished and the result is not bad. However, it still has obvious deficiencies about the accuracy, so I made further research to find the way to improve the image quality.

4.5 Improving accuracy of OCR

The original Tesseract has multiple methods to realize image processing by built-in packages (using the Leptonica library) before doing the actual OCR function. The Leptonica library plays a crucial part in pre-processing the image, nonetheless, the built-in library is old. If we depend on the program, there are always have something wrong happened. So why not do some preparation for better recognition accuracy in OCR.

The first is **Rescaling**, according to the official introduction, the Tesseract engine will work better on image which equip with a DPI doesn't lower than 300, you can consider the relation is higher resolution of image means higher recognition accuracy. If the images are under 300 dpi, the tesseract will have a unsatisfying result, because higher resolution will help Tesseract to resize images.



I. Část.

Liečenie ľudí.

Úwod.

Dielko toto, nesúce názow: „Prostonárodní domácí lekár“, odosielam medzi welactené obecenstwo so stanowiska tej preswedčenosti: že ono w mnohých pádoch ľudu náššmu môže sa stať oprawdiwým dobrodincom; zwláště chudobnej triede, ktorej počet najwyššie wystupuje. Upadne-li nektorý jej člen do nemoce, nenachádza sformútená rodina prostriedku, ktorým by mu k predošlému zdrawiu prispeť mohla, keď totiž lekár 2—3 hodiny cesty wzdialený býwa. Známe ju nám wonkowanom wšetky nemožnosti chudobnej triedy, ktoré wonkoncom nedowoľujú lekára powolať.

Jak mnohý wyhnul by predčasnému hrobu, keby sa mu skorá pomoc bola podala. Jako mála iskra s menšim namáhaním sa udušuje: tak i rýchla pomoc w rozličných chorobách nebespečenstwo lahšie odstraňuje.

Wšehotworca, ktorý nám krásnu prirodu „za matku“ dal, postaral sa o to: aby ona nám bola „wšetkým“: wychowatelkou, učitelkou a spolu i lekárkyňou.

Dielko toto podáwa nám lieky w hojnosti w prirode sa nachádzajúce, ktorými ju Boh k náššmu dobrému obohatil.

Figure 4.18

Second: Binarization (see figure 4.18), in this process will convert an image to black and white version, for convenience, Tesseract will use Otsu algorithm(built-in) to do this work, but the result may be suboptimal, especially when meet the page which background is of uneven darkness.

Third: Noise Removal, noise is something that we cannot totally removed, it's random variation of brightness or color in an image, just like the noise on the picture that captured by our phone in the dark light, there are tons of colorful spots in the dark side of the images. The noise will also make the text of the image more difficult to recognize. Certain types of noise cannot be removed by Tesseract in the binarization step, which can cause accuracy rates to drop. So, before recognition we need to help the tesseract has a decent luminous environment, the brighter circumstance will significantly decrease the noise.

The fourth: Dilation and Erosion (see figure 4.19), in the normal articles we can see various of characters, some of them are bold the others are thin. The different between characters will impact the recognition of details and reduce the

recognition accuracy. After researching, there are many image processing programs prefer to depending on the rules of Dilation and Erosion. When meet the characters which color is too shallow to be recognized precisely, the program will deepen the color which belong to the characters' edge, on the contrary, if the ink of the character is much expanded that disturb the OCR function, the program will shallow the ink and shrink the character. This technology can also be used to help the museum extract the words from the ancient articles, because the ink will expand with moisture into the paper. By using this technology, the archaeologists will get more information from the cultural relics and historic sites.

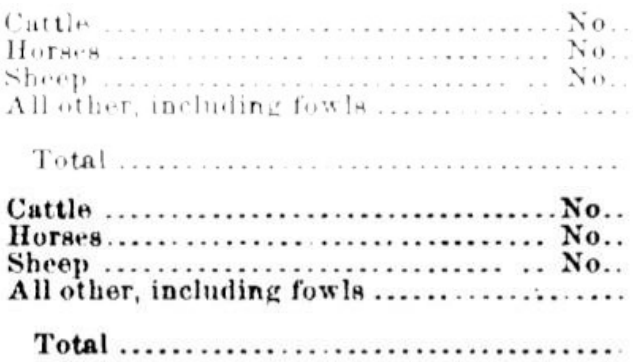


Figure 4.19

The fifth: Rotation (see figure 4.20), in my test, because I must hold the article by myself so I cannot sure the characters is in correct position, there is a critical deficiency, if the paper that you want to scan isn't locate at the appropriate position, the Tesseract recognition function will be affected critically. The worse Tesseract results means the OCR cannot have the

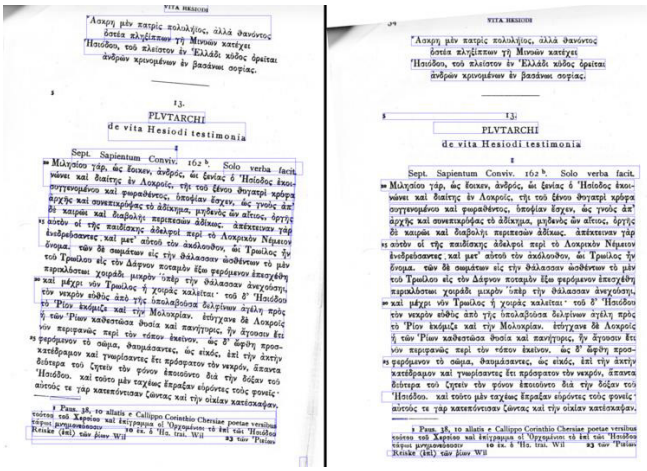


Figure 4.20

best working circumstance. So, in case of this issue happen, we need to make sure the paper will be scanned should on correct position.

Except these methods to help me improve the accuracy of OCR, the border of articles always plays a critical role in disturb the result of Tesseract. After follow these pre-processing, I found that my tesseract accuracy has a decent progress, but through the comparison, I found that the ambient brightness is the most significant factor, if the brightness changed, the tesseract results will have a big change.

5 Conclusion and Expectation

In a word, what I did is a device can help blind people to listen articles, for example, if I am the blind, someday I want to have a dinner outside because of celebrating birthday or others. Normally, the restaurant won't supply the braille menu, and the time that I come in is the busiest time for the restaurant, I don't want to disturb others, so I can use this device to read the menu, not have to very precisely read the dishes names but some critical materials should be fine, I can book my deal for tonight smoothly. I believe this device will give the blind more courage to go out, your ears are your eyes. Comparing with the reference materials, my project advantages is combining the OCR and TTS together. Of course, the whole device is based on raspberry pi 4b, because the raspberry pi is a small board with all essential components and it's very thin, even with the camera module and button, I can use a small case to install all the modules. So, if this device can be manufactured, it must be a portable equipment. At the beginning of considering

usage scenario, I think everything should be offline, but when I finished the project and have a review, I found that there is a SIM card module for raspberry pi. You can insert the SIM card for your pi and have internet all the time. If I have more time to realize this function if the sim card module is difficult, I can use a mobile hot spot to replace it. Nonetheless, there still have a big room for improvement. The first is the dark light performance, the Tesseract always can't recognize words accurately when meets dark light circumstance, if I have more time to improve this project, I will add a led fill-in light module by the side of the pi camera and use an ambient light sensor to detect the brightness, when the brightness under the specific value, the fill-in light will open to help the camera to capture the image, in this way the accuracy will have a big improvement. The second is the performance of resisting the rotation, because of the Tesseract prefer to using rectangles to mark the characters in each line. If you rotate the paper, you will have an irregular rectangle, so the recognition accuracy will be affected. I have more time I will find some methods to improve the ability of resisting rotation.

The main purpose is to want the blind to live a better life, I understand the smart phone has excellent offline built-in TTS system called Pico Text to speech. But it based on Android system, I try to transfer it to raspberry pi but there is no use. I did many researches about the functions that help the blind, but even the Samsung flagship has awful used experience for the blind, however the IOS phone have much better using experience, in my knowledge the if you open the android blind mode, it will separate the whole desktop by apps icon, when your finger

touches the different area on phone's screen, the phone will use speaker to read the app name to help user find the app that he/she wanted. If the user wants to know the contents of the screen, he/she just need to use fingers slide on this page, then he will listen the contents. There is a critical deficiency, because of android is open-source system, and the amount of blind user is little. So, there are almost no company want to spend much more time and money (comparing with the normal version of app) on developing the special version of app for the blind user. I want to make endeavor to help the blind. According to the scientific researches, the blind people will have a better hearing than the ordinary people. From some videos record the blind people daily life, I found if they use the phone built-in TTS function, they will use very fast speech speed to listen the contents, for ordinary people that speech speed is unacceptable, we need exercise for a long time to adapt this speed. In my project the Espeak engine sounds like the TTS engine on the phone. By the way, the reviews of the IOS and Android TTS engine show that Apple have better experience for the blind people. Because the IOS can control the app strictly so the apps on iPhone can be call easily by user. But if I want to use android or iOS, the price will much higher than the raspberry pi 4b. The 4B version equips with more function than the old raspberry pi. For convenience I choose the 4B instead of other pi. If I have more time, I will cut down the cost within nearly 60 pounds by replace the raspberry pi 4B by raspberry pi 3 or 2.

6 Reflections

In this project, I designed four steps to realize my target. Form OCR to TTS and GPIO, finally combine all of components together into the final work. I have to say I haven't touch any one of knowledge areas, but thanks to the studying in Cardiff university in my postgraduate semester, I learned the Python language. Through the assignments I understand the convenient of python, I used to build a small library book regulation system by python. I really love this coding language. So that I select Python as my coding language.

In the process of learning OCR technologies, I prefer to using two small exercises to help me learn the rules. The first is learning how to use OpenCV to realize face detection. After reviewing many examples of face detection, I understand OpenCV is treasure for OCR. There are tons of useful packages that you can utilize them to realize your work. Such as Tesseract engine. after realizing face detection, I can even realize the face recognition, this means that when the camera catches the face, the same with face detection, there is a rectangle by the side of the face, but this time the rectangle will mark the name of this face. Because through the first exercise, I learn how to train the camera to recognize different people's face. Of course, there is a premise that I need to have many photos about this specific person. After trying this exercise, I have the confidence to build an alarm system to detect the stranger. The second exercise is using program to recognize the characters in the selected image, in this process, I learned the theory of Tesseract engine, if I want to use it in my project, I just need to use camera module shoot a

picture of articles, and use the same method in second exercise to extract the characters in the image.

Like the OCR learning, I use exercises to learn the new knowledge in TTS area. Comparing with learning OCR, because there are many free speech engines and they have detailed introduction on their websites. Also, because it's free, there are many articles to guide me how to set speech engine properly. After learning, I tried different engine to test the sound quality. Finally, I select the Espeak, it's easy, offline and the sound like the blind user phone speak. After learning how to use OCR and OpenCV, I understand how to build a personal project by free resources. For my project, what I need to do is connecting the two parts together.

Thanks to using many exercises to familiar with the new knowledge, I was surprised at the combination work. I didn't do many researches, just need to follow the exercises procedures. After combination, according to the learning experience on courses, if I didn't add a physical switch, it's not appropriate in real life, so I decide to add a button to control the device to realize the function in my designing. For learning GPIO to control the button, I search the GPIO rules, because I had experience on control SCM by C++. But this time is python program, through the LED and Button exercises, I know how to use python to control the raspberry pi GPIO. After learning GPIO, I found that I can use unofficial hardware to extend the function of raspberry pi. There are many combinations on raspberry pi, I can build CCTV alarm system by buzzer module and camera. There is infinite

possibility for raspberry pi because of GPIO. Thanks to my supervisor recommend me to select the raspberry pi project.

In the past, when I meet one unfamiliar knowledge area, I will start to learn the knowledge from every basic knowledge. Next time when I touch unknow knowledge area, I will learn the knowledge that I needed and train it from small exercises, through the exercises I can understand the theory much quicker than before. Through this project I know the new methods of learning, and much of new knowledge.

7 Reference

[1] Ji, Y. 2019 A Static Analysis of Subjective and Object Image Quality Assessment on Different Image Categories. MSc Dissertation, Cardiff University.

[2] Oulkar Onkar S, D, K, P. 2019 Optical Character Recognition for Blind Using Raspberry PI. MSc Dissertation.

[3] Xia Shan, J. 2020. Raspberry pi button experiment by Python Available at: https://blog.csdn.net/qq_27320195/article/details/108980585?spm=1001.2014.3001.5501

[4] Muhammad, A. 2019. Optical Character Recognition Using Raspberry Pi with OpenCV and Tesseract, Available at: https://maker.pro/raspberry-pi/tutorial/optical-character-recognizer-using-raspberry-pi-with-opencv-and-tesseract

[5] Dexter, I. 2019. How to Make Your Raspberry Pi Speak, Available at: https://www.dexterindustries.com/howto/make-your-raspberry-pi-speak/

[6] Connor, M. 2018. Utilizes Raspberry Pi, Azure, Twilio, and AWS APIs to monitor for motion and use face recognition to send customized MMS, Available at: https://maker.pro/raspberry-pi/projects/raspberry-pi-security-camera-with-face-recognition

[7] Muhammad, A. 2019. Learn how you can use the open-source library OpenCV with a Raspberry Pi to create face and object detection, Available at: https://maker.pro/raspberry-pi/projects/raspberry-pi-security-camera-with-face-recognition

[8] Adrian, R. 2018. pip install OpenCV, Available at: https://www.pyimagesearch.com/2018/09/19/pip-install-opencv/

[9] Adrian, R. 2018. OpenCV Tutorials, Resources, and Guides, Available at: https://www.pyimagesearch.com/opencv-tutorials-resources-guides/

[10] Lan Teng, 2020.The technologies of Python: handle 'HTTP lid' issues, Available at: https://blog.csdn.net/qq_38161040/article/details/107849745

[11] Liu Wu, S, 2020.Install the OpenCV 4.2.0 based on Raspberry pi (Raspbian), Available at: https://blog.csdn.net/qq_27149279/article/details/105331034

[12] Py pi, 2020. Install the face-recognition 1.3.0 Available at: https://pypi.org/project/face-recognition/

[13] Liu Wu, S, 2020. Realizing the face recognition on Raspberry pi 4B. Available at: https://blog.csdn.net/qq_27149279/article/details/105307368?utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7Edefa

ult-3.control&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7Edefault-3.control

[14] Ling3ye, 2019. Install the OpenCV 4.1.2 on Raspberry Pi 4b. Available at: https://blog.csdn.net/ling3ye/article/details/103229033?utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromBaidu%7Edefault-5.no_search_link&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromBaidu%7Edefault-5.no_search_link

[15] v Tou Bi, 2017. Change the raspberry system version to Chinese , Available at: https://blog.csdn.net/la9998372/article/details/77886806?ops_request_misc=%257B%2522request%255Fid%2522%253A%252216311518441678026199550%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=163115184416780261995550&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_positive~default-2-77886806.first_rank_v2_pc_rank_v29&utm_term=%E6%A0%91%E8%8E%93%E6%B4%BE%E6%8D%A2%E6%BA%90&spm=1018.2226.3001.4187

[16] Zhgfn, 2021. Utilizing Tesseract engine to realize the Chinese recognition, Available at: https://blog.csdn.net/zhgfn4056100/article/details/118518491?ops_request_misc=&request_id=&biz_id=102&utm_term=%E6%A0%91%E8%8E%93%E6%B4%BE4b%E6%96%87%E5%AD%97%E8%AF%86%E5%88%AB&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduweb~default-0-118518491.first_rank_v2_pc_rank_v29&spm=1018.2226.3001.4187

[17] 39592315, 2020. The raspberry pi realizes the TTS_ Raspberry pi voice recognition, Available at: https://blog.csdn.net/weixin_39592315/article/details/110172275?utm_medium=distribute.pc_relevant.none-task-blog-2~default~baidujs_utm_term~default-0.no_search_link&spm=1001.2101.3001.4242

[18] Khito, 2020. The raspberry pi project based on OpenCV and Tesseract, Available at: https://blog.csdn.net/jiyotin/article/details/105935015?ops_request_misc=&request_id=&biz_id=102&utm_term=%E6%A0%91%E8%8E%93%E6%B4%BETesseract&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduweb~default-3-105935015.first_rank_v2_pc_rank_v29&spm=1018.2226.3001.4187

[19] Elinux.org, 2019. RPi Text to Speech (Speech Synthesis), Available at: https://elinux.org/RPi_Text_to_Speech_(Speech_Synthesis)

[20] Yin Li, 2018.The adjustments of raspberry pi camera , Available at: https://blog.csdn.net/m0_37509650/article/details/80449042?ops_request_misc=%257B%2522request%255Fid%2522%253A%252216318477241678027412034 1%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=163184772416780274120341&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduend~default-1-80449042.first_rank_v2_pc_rank_v29&utm_term=%E6%A0%91%E8%8E%93%E6%B4%BE%E6%91%84%E5%83%8F%E5%A4%B4%E5%AF%B9%E7%84%A6&spm=1018.2226.3001.4187

[21] Hou Zeng T, 2019. Set the raspberry camera through program, Available at: https://blog.csdn.net/weixin_44345862/article/details/91048083?ops_req

uest_misc=%257B%2522request%255Fid%2522%253A%252216318490241678027417101 7%2522%252C%2522scm%2522%253A%252220140713.130102334.%2522%257D&request_id=163184902416780274171017&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~baidu_landing_v2~default-1-91048083.first_rank_v2_pc_rank_v29&utm_term=camera.shutter_speed&spm=1018.2226.3001.4187

[22] Embed Debugger, 2021. The raspberry pi + OpenCV to realize the live view from pi camera, Available at: https://blog.csdn.net/weixin_47965042/article/details/113274237?ops_request_misc=%257B%2522request%255Fid%2522%253A%252216318458991678027157062 9%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=163184589916780271570629&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_click~default-2-113274237.first_rank_v2_pc_rank_v29&utm_term=%E6%A0%91%E8%8E%93%E6%B4%BE%E6%91%84%E5%83%8F%E5%A4%B4&spm=1018.2226.3001.4187

[23] Wei X, 2018. Use pyttsx3 to realize the TTS function (Chinese version). Available at: https://blog.csdn.net/weixin_39012047/article/details/82012306?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163247795016780269842234%2522%252C%2522scm%2522%253A%252220140713.130102334.%2522%257D&request_id=163247795016780269842234&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduend~default-1-82012

306.pc_search_ecpm_flag&utm_term=pyttsx3%E4%B8%AD%E6%96%87&spm=1018.2226.3001.4187

[24] Mo Y, 2018. Using introduction pytts3 based on python3. Available at: https://blog.csdn.net/dss_dssssd/article/details/82693742?ops_request_misc=%257B%2522request%255Fid%2522%253A%252216324774431678026981378 9%2522%252C%2522scm%2522%253A%252220140713.130102334.%2522%257D&request_id=163247744316780269813789&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~baidu_landing_v2~default-4-82693742.pc_search_ecpm_flag&utm_term=pyttsx3&spm=1018.2226.3001.4187

[25] Er G, 2020. Using Python pyttsx3 to read the text. (Multiple languages) Available at: https://blog.csdn.net/qq_35164554/article/details/105846824?utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7Edefault-1.no_search_link&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7Edefault-1.no_search_link

[26] Wait F, 2016. The series of learning Raspberry pi 8-usage of GPIO. Available at: https://blog.csdn.net/wait_for_taht_day5/article/details/52320035?utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7Edefault-2.no_search_link&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7Edefault-2.no_search_link

[27] You K, 2019.The raspberry pi realizes the LED flash by GPIO. Available at: https://blog.csdn.net/youknor/article/details/90037566?utm_medium=distri

bute.pc_relevant.none-task-blog-2~default~baidujs_utm_term~default-1.no_search_link&spm=1001.2101.3001.4242

[28] Wang C, 2019.The raspberry pi experiment: press the button module. Available at: https://blog.csdn.net/qq_37037348/article/details/93738103?ops_request_misc=%257B%2522request%255Fid%2522%253A%252216325395121678027 4194784%2522%252C%2522scm%2522%253A%252220140713.130102334.%2522%257D&request_id=163253951216780274194784&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~baidu_landing_v2~default-1-93738103.pc_search_ecpm_flag&utm_term=%E6%A0%91%E8%8E%93%E6%B4%BE%E4%BD%BF%E7%94%A8%E6%8C%89%E9%92%AE&spm=1018.2226.3001.4187

[29] Small A, 2020. The basic of raspberry pi GPIO: press the button to control the LED lights. Available at: https://blog.csdn.net/qq_41149269/article/details/104367656?ops_request_misc=%257B%2522request%255Fid%2522%253A%252216325600781678036652944 8%2522%252C%2522scm%2522%253A%252220140713.130102334.%2522%257D&request_id=163256007816780366529448&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduend~default-2-104367656.pc_search_ecpm_flag&utm_term=%E6%A0%91%E8%8E%93%E6%B4%BE%E6%8C%89%E9%92%AE&spm=1018.2226.3001.4187

[30] Xia S, 2020. The Raspberry pi experiment: press button. Available at: https://blog.csdn.net/qq_27320195/article/details/108980585?spm=1001.2014.3001.5501

[31] Cheng X, 2019. The history of Raspberry pi development. Available at: https://blog.csdn.net/chengxian6892/article/details/100838256?ops_request_misc=%257B%2522request%255Fid%2522%253A%252216329027381678027416615 1%2522%252C%2522scm%2522%253A%252220140713.130102334.%2522%257D&request_id=163290273816780274166151&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduend~default-1-100838256.pc_search_all_es&utm_term=%E6%A0%91%E8%8E%93%E6%B4%BE%E7%9A%84%E5%8F%91%E5%B1%95&spm=1018.2226.3001.4187

[32] Shu M, 2021 The introduction of different raspberry pi. Available at: https://blog.csdn.net/qq_41676577/article/details/112849618?ops_request_misc=%257B%2522request%255Fid%2522%253A%252216329027381678027416615 1%2522%252C%2522scm%2522%253A%252220140713.130102334.%2522%257D&request_id=163290273816780274166151&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduend~default-2-112849618.pc_search_all_es&utm_term=%E6%A0%91%E8%8E%93%E6%B4%BE%E7%9A%84%E5%8F%91%E5%B1%95&spm=1018.2226.3001.4187

[33] Cumifi, 2020, The pre-process of Tesseract: how to improve the accuracy of Tesseract. Available at: https://blog.csdn.net/cumifi2519/article/details/108157723?ops_request_misc=&request_id=&biz_id=&utm_medium=distribute.pc_search_result.none-task-blog-2~all~es_rank~default-8-108157723.pc_search_all_es&utm_term=%E5%A6%82%E4%BD%95%E6%8F%90%E9%AB%98tesseract&spm=1018.2226.3001.4187

[34] Claroja, 2018, Improving the Accuracy and Quality of Tesseract. Available at: https://blog.csdn.net/claroja/article/details/82992643?ops_request_misc=%257B%2522request%255Fid%2522%253A%252216330893121678026987273 7%2522%252C%2522scm%2522%253A%252220140713.130102334.%2522%257D&request_id=163308931216780269872737&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~baidu_landing_v2~default-1-82992643.pc_search_all_es&utm_term=%E5%A6%82%E4%BD%95%E6%8F%90%E9%AB%98tesseract&spm=1018.2226.3001.4187

[35] Fhqlong T, 2018, The service manual of Raspberry pi GPIO. Available at: https://blog.csdn.net/fhqlongteng/article/details/80395059?ops_request_misc=%257B%2522request%255Fid%2522%253A%252216330998511678027158669 4%2522%252C%2522scm%2522%253A%252220140713.130102334.%2522%257D&request_id=163309985116780271586694&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduend~default-1-80395059.pc_search_all_es&utm_term=GPIO%E6%A0%91%E8%8E%93%E6%B4%BE&spm=1018.2226.3001.4187

[36] Xia S J, 2020, The development of Raspberry pi: LED experiment. Available at: https://blog.csdn.net/qq_27320195/article/details/108969034