A PDF reader with gesture & voice interaction

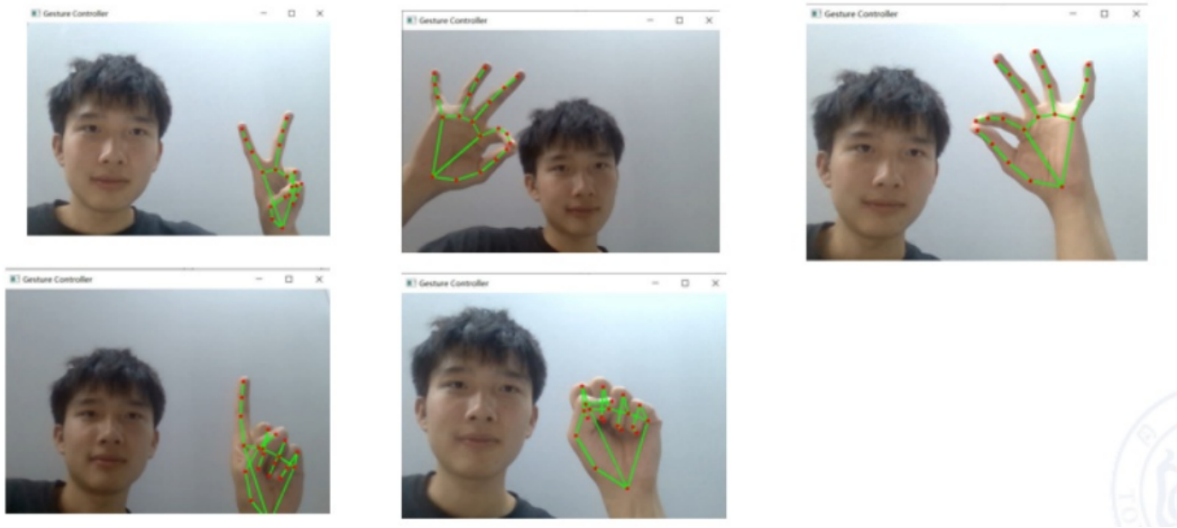
> Human Computer Interaction Final Project
>
> Group Member: 2052134 Liu Zhihua 2052717 Chen Chen 2053872 Zhu Xunyuan
>
> Instructor: Professor Shen Ying

1 A brief description of the project

This project is based on mediapipe, iFLYTEK CO.LTD. voice interface, and Baidu translation interface. Users can use gestures and voice to realize remote control of smart devices, thereby realizing mouseless operation, and providing new ideas and realizations for human-computer interaction.

Specifically:

- Users can perform operations such as turning pages, zooming in and out, turning on voice input, and exiting through six simple gestures.



- Users can also control the display pdf by voice, and with the help of voice, they can control the number of pages to jump, automatically generate notes, and automatically translate. Users can also control the display time by voice.

2 The implemented requirements

2.1 Gesture recognition

This part is based on Mediapipe's gesture recognition technology. Considering the user's gesture selection tendency, we have selected six gestures that are most convenient for the user. These gestures can perform various operations, including adjusting the brightness of the display, turning pages up and down, controlling the progress bar, moving Mouse, exit the current interface, call voice assistant and other functions.

Through gesture control, users can interact with the reader at a distance without touching the physical device, which improves the user experience and convenience. This technology has broad application prospects and can be used not only in readers, but also in other electronic devices, such as smart phones and tablet computers, to provide users with more intelligent and humanized interaction methods.

2.2 Voice aspects

By combining iFLYTEK voice dictation and Baidu translation API, our voice recognition technology can realize more intelligent voice control. Users can easily use voice control to perform various operations, such as displaying the time, jumping to the number of PDF pages, OCR selection to generate notes, translation and other functions. In this way, users can achieve a more natural and efficient interactive experience without manual operation, which greatly improves the efficiency of use.

At the same time, our technology also supports multi-thread processing. Using Qthread, users can perform other operations during voice input, such as browsing web pages, editing documents, etc., without blocking the use of other functions due to voice input. After stopping the voice for 1 second, the software will automatically cancel the reading and recognize the current recording, which improves the accuracy and stability of voice recognition.

Cooperating with gesture control technology, our voice recognition technology can further enhance the convenience and intelligence of the interactive experience. Users can choose a more convenient interaction method according to their habits and preferences to achieve more flexible operations. This multi-interaction design can not only improve user satisfaction and experience, but also bring more choices and fun to users.

To sum up, the speech recognition technology based on iFLYTEK voice dictation and Baidu translation API can not only achieve more intelligent and efficient voice control, but also cooperate with gesture control technology to improve the convenience and intelligence of the interactive experience . This technology has broad application prospects and will play an increasingly important role in the future intelligent life

3 Advantages and disadvantages

3.1 Advantages

1. Choose the most user-friendly six gestures to perform various operations, including adjusting display brightness, turning pages up and down, controlling the progress bar, moving the mouse, exiting the current interface, calling voice assistants, etc., which improves the user's operating experience and convenience.
2. Gesture control allows users to interact with the reader from a distance without touching the reader, which improves the freedom and comfort of use.
3. Based oniFLYTEK CO.LTD voice dictation and Baidu translation API, users can realize functions such as voice control display time, pdf page number jump, OCR selection to generate notes, translation, etc., providing more diversified interaction methods.
4. Using Qthread, users can perform other operations during voice input, which improves the response speed and efficiency of the software.
5. Cooperating with gesture control, it improves the convenience and experience of use.

3.2 Disadvantages

1. For gesture control, the user may need to carry out certain training and adaptation, otherwise there may be misoperation or the operation is not smooth enough.
2. The accuracy and stability of voice control may be affected by environmental noise interference, or the user's pronunciation is not clear enough.

4 How to improve our program

4.1 Improvements to be made

1. In terms of gesture opening speech, due to the calculation of response time and silence frame, if the user fails to grasp the time to start speaking, the recognition may be incomplete or empty. Consider adding a prominent popup when the microphone starts and closes automatically when it stops speaking, so the user knows when they can start talking. We didn't add this feature very well due to timing issues.
2. With gesture control, there is a delay in the response of the page to the action. For example, the user has stopped moving gestures, but the system is still zooming in on the page. On the one hand, this may be related to my personal environment and equipment. On the other hand, it may be that the sensitivity of gesture detection needs to be improved. We used mediapipe as a toolkit for gesture keypoint recognition because it was easy to use, but it might be better to try something else like JAVAFX.
3. For voice input, the user needs to first click the microphone through gesture or mouse each time. This doesn't fit with our philosophy of making interaction as simple as possible. Consider always turning on user microphone permissions. The instruction is recognized and executed as soon as the user speaks it. This way, you don't have to account for the delay caused by gestures turning on speech. We didn't do it due to lack of sufficient api calls and implementation difficulties.

4.2 Future outlook

Speech recognition and gesture control technology are important directions in the field of intelligent interaction in the future. With the continuous development and application of artificial intelligence technology, the application prospects of these two technologies are becoming more and more broad. Here are some thoughts and reflections on their future prospects:

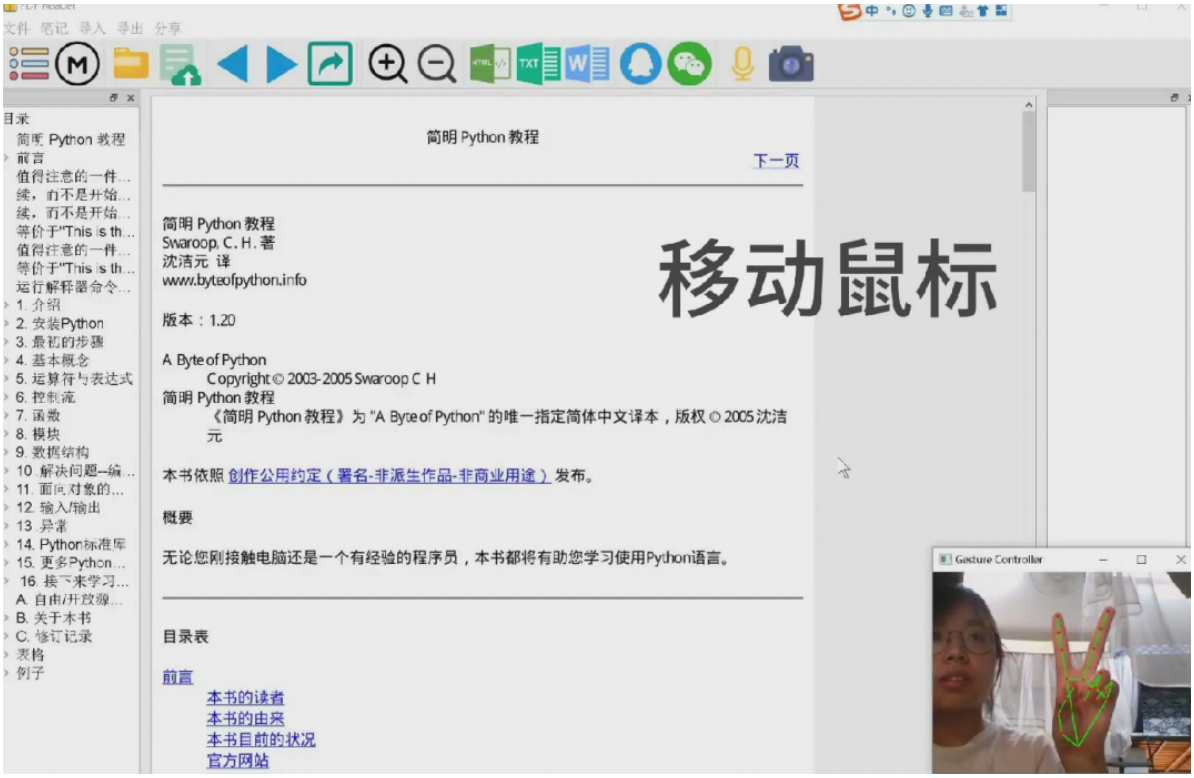
1. The degree of intelligence will continue to increase: With the continuous development of artificial intelligence technology, voice recognition and gesture control technology will continue to improve its intelligence, enabling more accurate, natural and intelligent interaction methods.
2. Multimodal interaction will become a trend: In the future, intelligent interaction will pay more and more attention to multimodal interaction, that is, the mixed use of voice, gesture, touch and other interactive methods to provide richer, natural and convenient interaction methods .
3. Privacy issues need to be paid attention to: speech recognition and gesture control technologies involve user privacy issues, which need to be paid attention to. Future technologies should provide more intelligent and convenient interaction methods on the premise of protecting user privacy.
4. Interaction methods suitable for different groups of people: intelligent interaction should be suitable for the needs of different groups of people, including the visually impaired, hand-disabled and other specific groups, and it is necessary to provide diversified and personalized interaction methods.

5. Design and humanization considerations: Future intelligent interaction design should be more humane, able to better fit with the natural communication methods of human beings, and provide a more natural and smooth interactive experience to meet people's needs for interactive methods.

5 Operation implementation

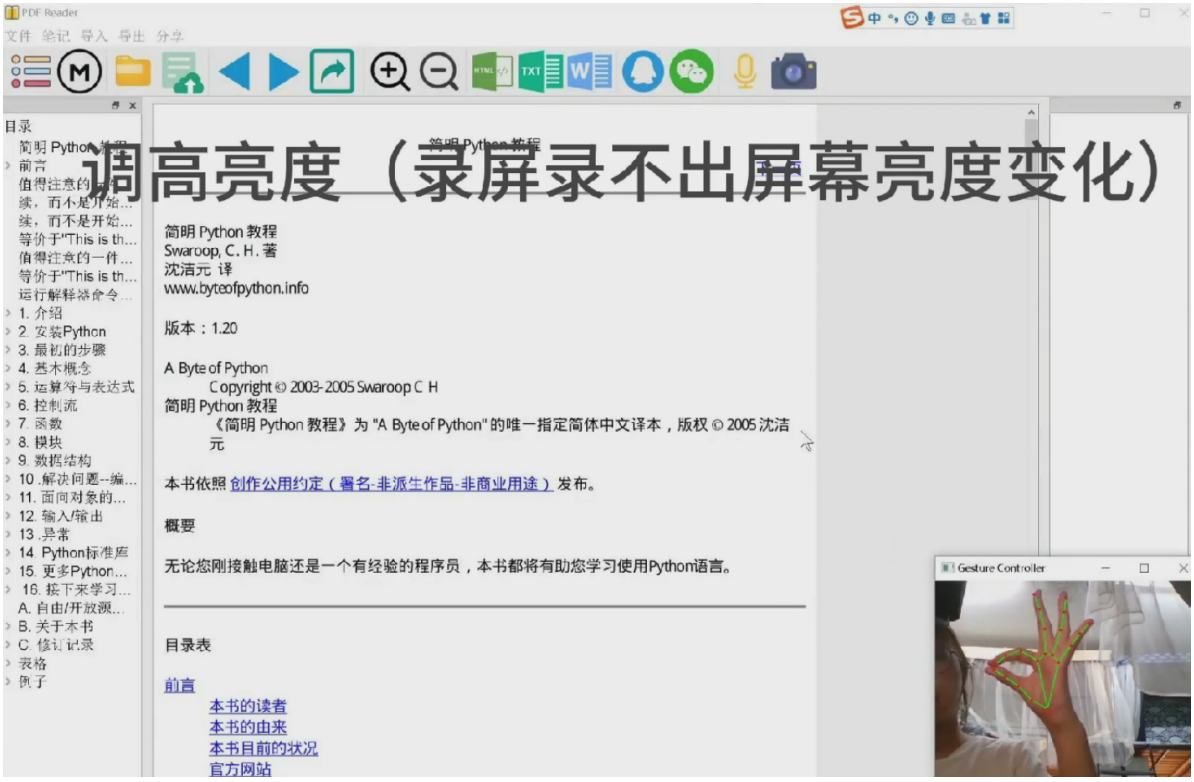
5.1 Move the mouse

After opening the pdf reader, the state of five fingers apart is changed to two fingers, and the mouse can be moved by moving the two fingers.



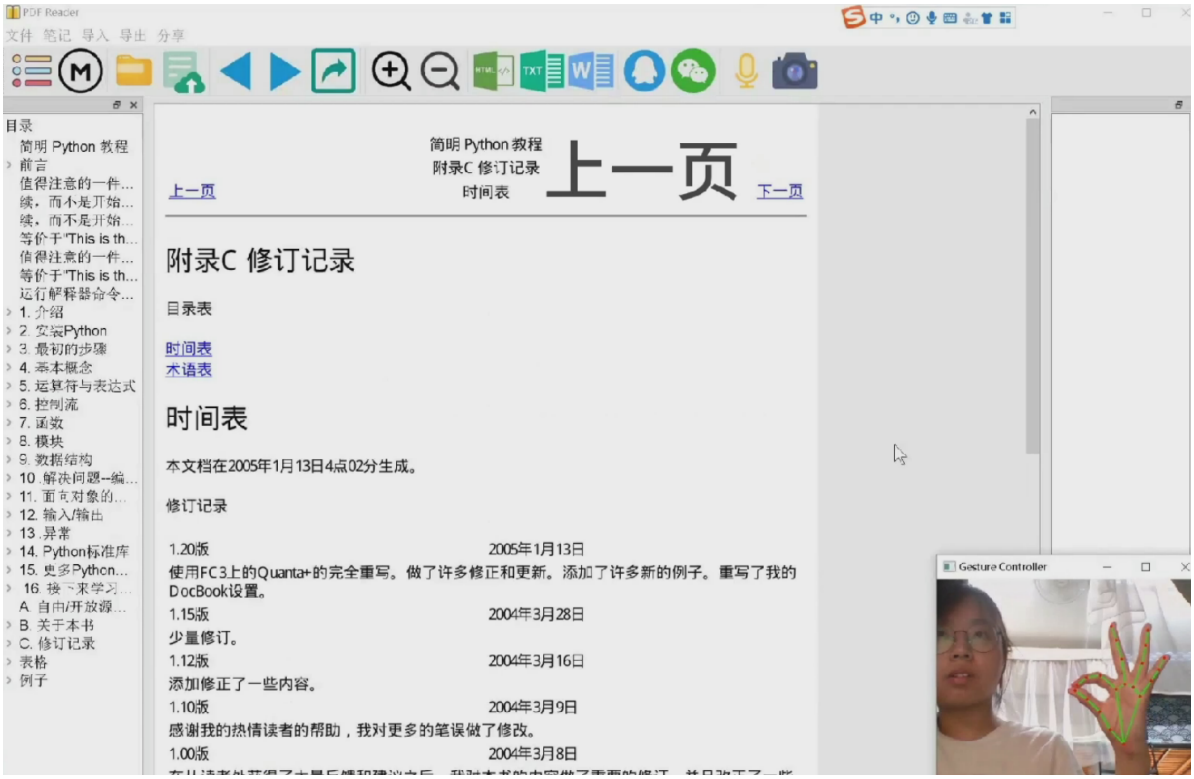
5.2 Dimming

The two fingers of the right hand make a pinch gesture and then move left and right to adjust the brightness



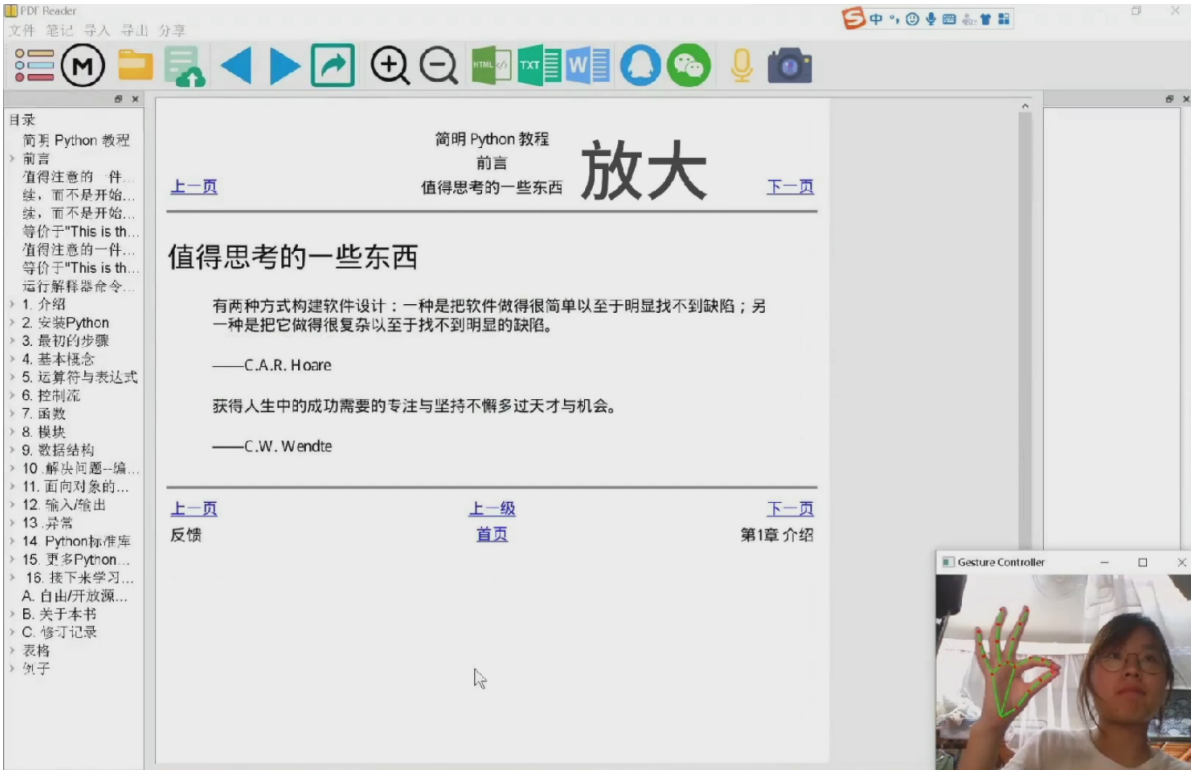
5.3 Page up and down

The two fingers of the right hand make a pinch gesture and then move up and down to turn the page.



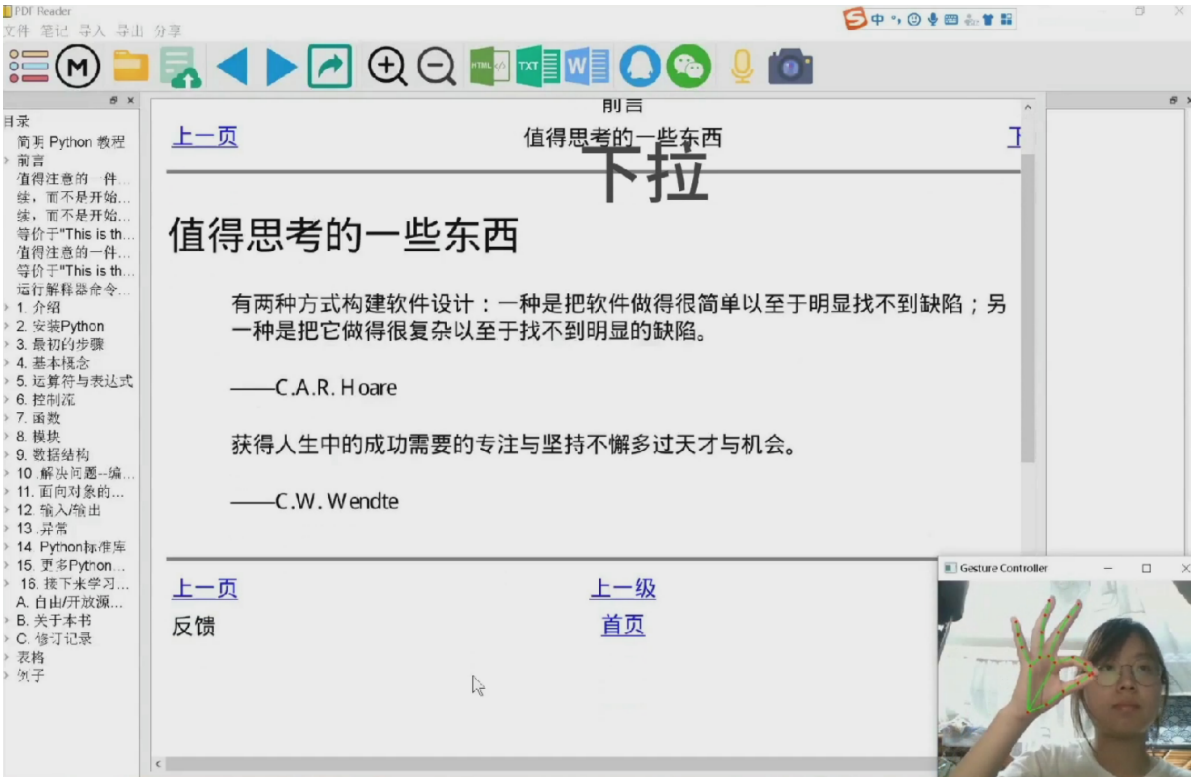
5.4 Zoom in/out explicitly

The two fingers of the left hand make a pinch gesture and then move left and right to zoom in/out the explicit pdf.



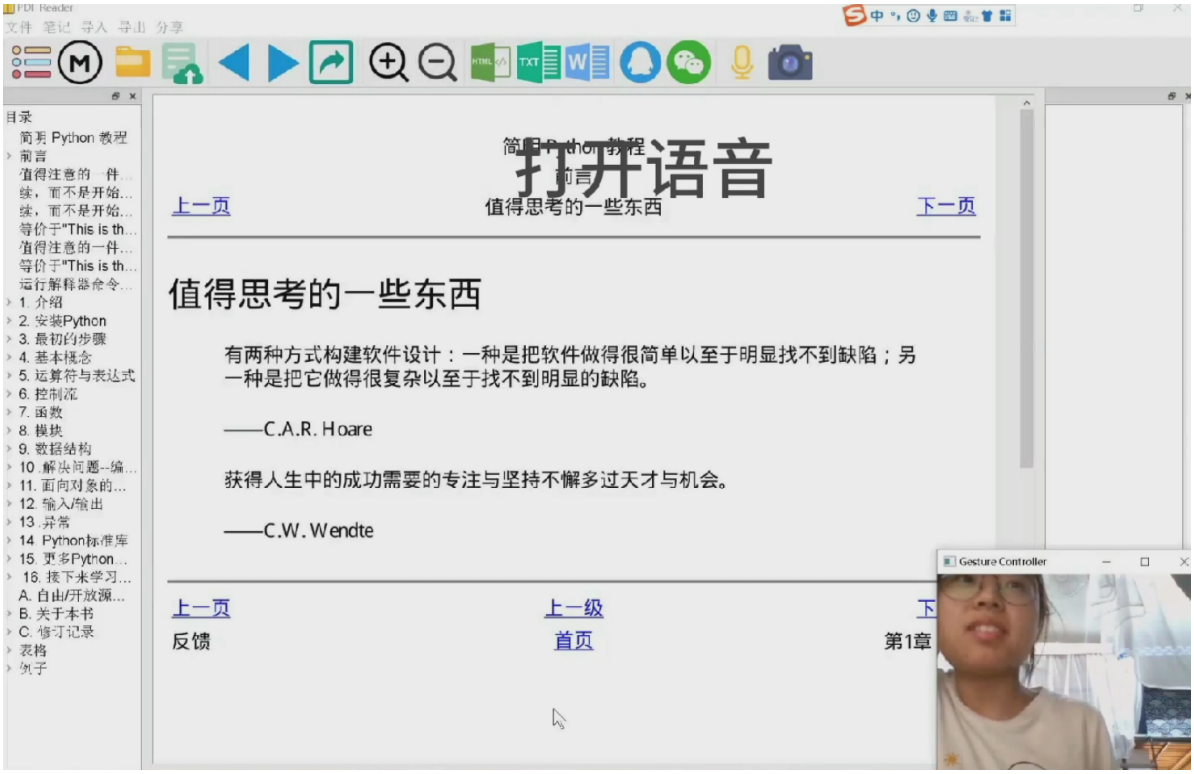
5.5 Pull down or pull up

The two fingers of the left hand make a pinch gesture and then move up and down to pull down/pull up the pdf.



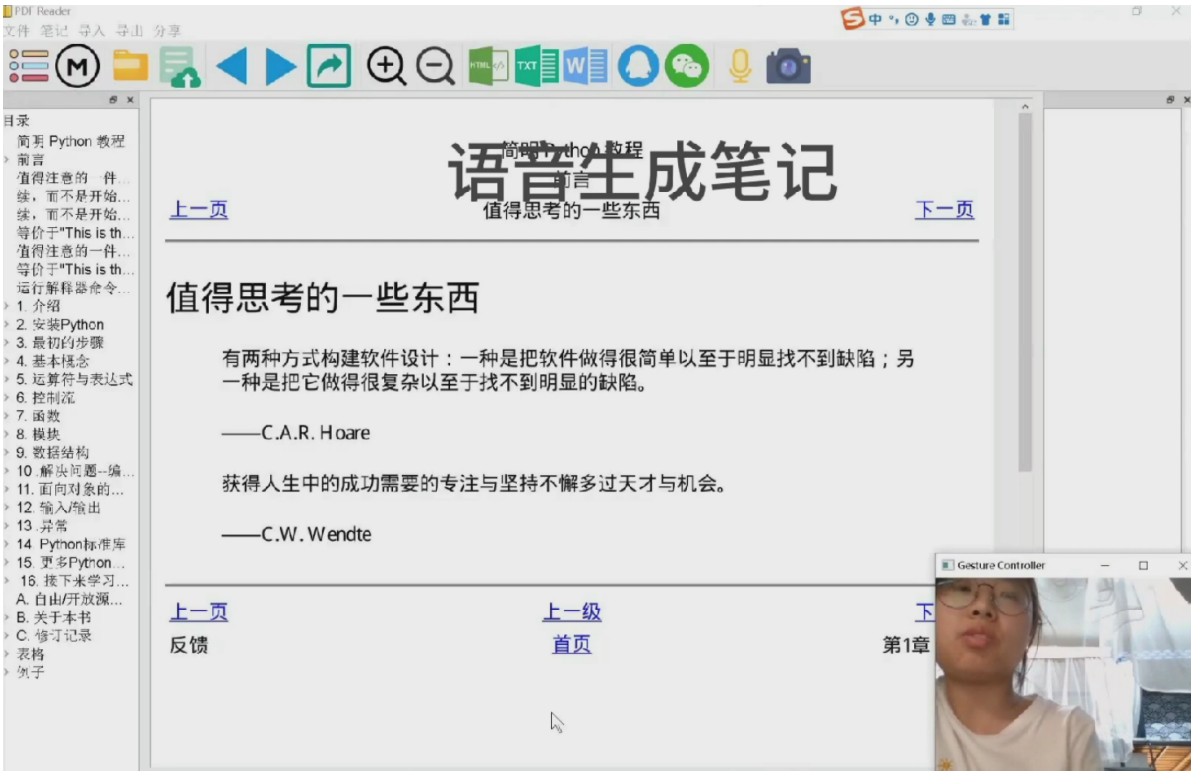
5.6 Open voice input

Switch the finger from two fingers to one finger to turn on the voice, and the user can speak.



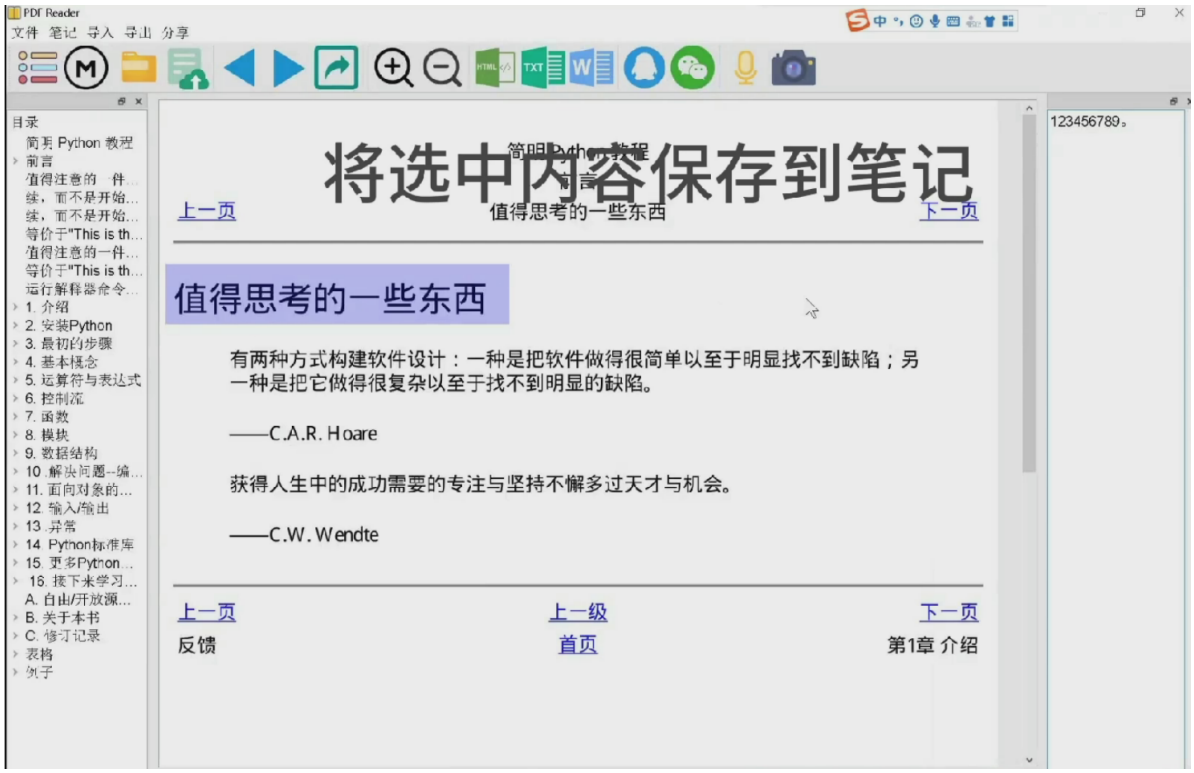
5.7 Generate notes

When the user continues to say the content of the note after the generated note, the pdf reader will record what the user said in real time.



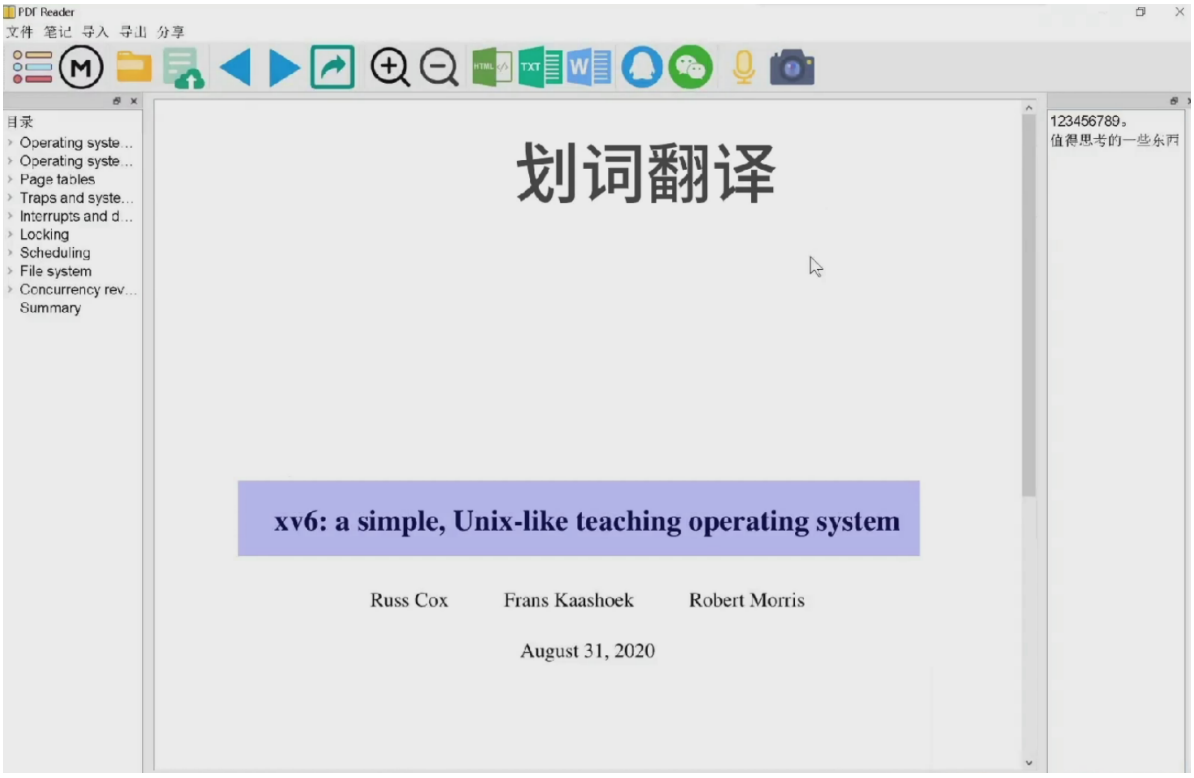
5.8 OCR recognition + save selected content

After the area selected by the user is highlighted, OCR recognition will be performed, and the user can input voice to let the system save the selected content.



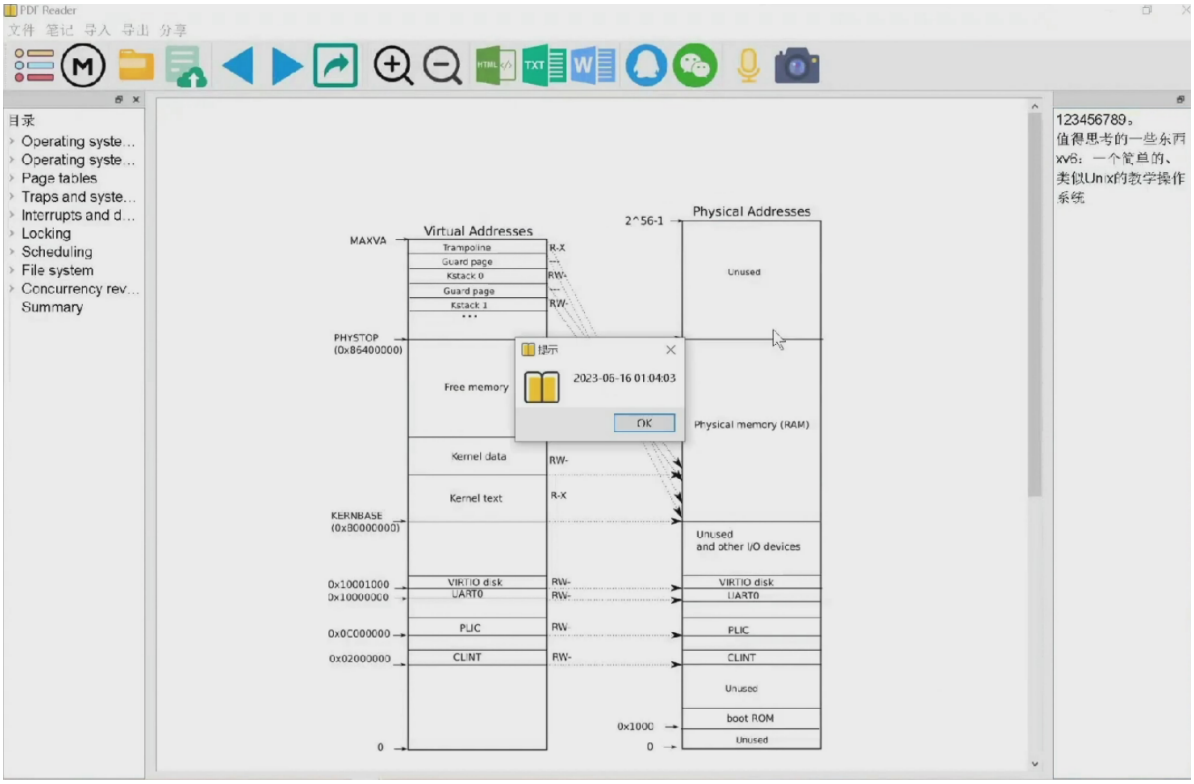
5.9 Word translation

After the selected area is highlighted, the user can select the translation function by voice input.



5.10 Ask time

Voice inquiry time can be displayed.



6 Core code

6.1 Realization of the total control interface

```
def generateToolBar(self):
        self.toolbar.setMinimumSize(QSize(200, 200))
        self.toolbar.setIconSize(QSize(80, 80))  # 设置工具栏图标大小
        # self.toolbar.setToolButtonStyle(Qt.ToolButtonTextBesideIcon)  # 文字在图
标旁边
        # self.toolbar.setToolButtonStyle(Qt.ToolButtonTextUnderIcon)  # 文字在图标
下方
        # 不设置以上两句话默认只显示图标
        ToC = QAction(QIcon('icon/目录 (5).png'), '目录', self.toolbar)
        openFile = QAction(QIcon('icon/file.png'), '打开文件', self.toolbar)
        saveFile = QAction(QIcon('icon/Save (3).png'), '保存文件', self.toolbar)
        prePage = QAction(QIcon('icon/分页 下一页 (1).png'), '上一页',
self.toolbar)
        nextPage = QAction(QIcon('icon/分页 下一页.png'), '下一页', self.toolbar)
        turnPage = QAction(QIcon('icon/跳转.png'), '跳转', self.toolbar)
        # insertPage = QAction(QIcon('icon/insert.png'), '添加页面', self.toolbar)
        # deletePage = QAction(QIcon('icon/delete.png'), '删除当前页面',
self.toolbar)
        # extractPage = QAction(QIcon('icon/pdf.png'), '提取pdf页面',
self.toolbar)
        enlargePage = QAction(QIcon('icon/放大 (1).png'), '放大', self.toolbar)
        shrinkPage = QAction(QIcon('icon/缩小.png'), '缩小', self.toolbar)
        toHTML = QAction(QIcon('icon/html (3).png'), '导出为HTML', self.toolbar)
        toTXT = QAction(QIcon('icon/txt.png'), '导出为TXT', self.toolbar)
        toDocx = QAction(QIcon('icon/word.png'), '导出为Docx', self.toolbar)
```

```
        # toKindle = QAction(QIcon('icon/kindle.png'), '发送到kindle',
self.toolbar)
        toQQ = QAction(QIcon('icon/QQ.png'), '分享到QQ', self.toolbar)
        toWechat = QAction(QIcon('icon/wechat.png'), '分享到微信', self.toolbar)
        # toEmail = QAction(QIcon('icon/email.png'), '通过邮件发送', self.toolbar)
        editor = QAction(QIcon('icon/markdown_2.png'), "编辑器", self.toolbar)
        microphone = QAction(QIcon('icon/microphone.png'), "语音命令",
self.toolbar)
        camera = QAction(QIcon('icon/camera.png'), "手势控制", self.toolbar)
        editor.triggered.connect(self.onDoc2)

        nextPage.setShortcut(Qt.Key_Right)
        prePage.setShortcut(Qt.Key_Left)

        # self.shareConnect(toEmail, toKindle, toQQ, toWechat)
        openFile.triggered.connect(self.onOpen)
        ToC.triggered.connect(self.onDock)
        saveFile.triggered.connect(self.onSave)
        microphone.triggered.connect(self.onMicrophone)
        camera.triggered.connect(self.onCamera)
        prePage.triggered.connect(self.onPrepage)
        nextPage.triggered.connect(self.nextpage)
        turnPage.triggered.connect(self.turnpage)
        enlargePage.triggered.connect(self.enlargepage)
        shrinkPage.triggered.connect(self.shrinkpage)
        self.tofileConnect(toDocx, toHTML, toTXT)
        # self.pageConnect(deletePage, extractPage, insertPage)
```

6.2 Call API

wsParam is the parameter for connecting to the websocket, followed by a call to the iFLYTEK voice-to-text interface.

```
# 设置websocket参数
wsParam = Ws_Param(APPID='7d8fd752',
APISecret='OWYxMTkyNWE4YWRmNzJlMTAxZDE5Y2Fk',
                       APIKey='14798c746e7527cec1fef4b9b9dd1920',
                       AudioFile='')

# if __name__ == "__main__":
def main(filename) :
    # 测试时候在此处正确填写相关信息即可运行
    wsParam.set_filename(filename)
    time1 = datetime.now()
    global sentence
    sentence =''
    websocket.enableTrace(False)
    wsUrl = wsParam.create_url()
    ws = websocket.WebSocketApp(wsUrl, on_message=on_message, on_error=on_error,
on_close=on_close)
    ws.on_open = on_open
    ws.run_forever(sslopt={"cert_reqs": ssl.CERT_NONE})
```

```
    print(sentence)

    time2 = datetime.now()
    print(time2-time1)

    return sentence
```

6.3 Baidu translation interface core code

```
def translate(from_lang, to_lang, query):
    # Set your own appid/appkey.
    appid = '20230615001713542'
    appkey = 'i_boQF8PhuyUhu3sfjaI'

    endpoint = 'http://api.fanyi.baidu.com'
    path = '/api/trans/vip/translate'
    url = endpoint + path

    # Generate salt and sign
    def make_md5(s, encoding='utf-8'):
        return md5(s.encode(encoding)).hexdigest()

    salt = random.randint(32768, 65536)
    sign = make_md5(appid + query + str(salt) + appkey)

    # Build request
    headers = {'Content-Type': 'application/x-www-form-urlencoded'}
    payload = {'appid': appid, 'q': query, 'from': from_lang, 'to': to_lang,
'salt': salt, 'sign': sign}

    # Send request
    r = requests.post(url, params=payload, headers=headers)
    result = r.json()

    # Show response
    print(json.dumps(result, indent=4, ensure_ascii=False))

    ret = ""
    for res in result["trans_result"]:
        ret += res["dst"]

    return ret
```

6.4 Speech input and recognition

After clicking the microphone icon, it will jump to this function and start a new audio thread. After the thread is created, it will be linked to onMicrophoneResult after the end, so that it can jump to another function after the recording is over, and call the iFLYTEK CO.LTD. voice interface for voice conversion. Word.

```
def onMicrophone(self):
        # 执行语音输入
        t = AudioThread(getCmd, ('test',59))
        t.finishSignal.connect(self.onMicrophoneResult)
        t.start()
        time.sleep(0.1)
```

getcmd function, the top record_audio is recording, we will record the number of silent frames, once the sound is too low to reach 1s, it will automatically stop sound recording, but the audio format is wav, it needs to be converted to mp3 format before it can be used, getcmd finally The function.

The onMicrophoneResult function has obtained the text here, and after obtaining the speech-to-text result, it will execute the corresponding command according to the keywords in the result.

```
def onMicrophoneResult(self, cmd):
        print('onMicrophoneResult', cmd)
        # QMessageBox.about(self, "语音输入结果", cmd)
        # cmd = '跳转到第328964页'
        if cmd.__contains__('转到'):
            if not self.book_open:
                QMessageBox.about(self, "提示", "请先打开一个文件")
                return
            allpages = self.doc.pageCount
            print(allpages)
            # 取出数字
            page = re.findall(r"\d+\.?\d*", cmd)
            print(page)
            if len(page) == 0:
                QMessageBox.about(self, "提示", "未识别到页数")
                return
            if int(page[0]) > allpages:
                QMessageBox.about(self, "提示", "页数过大")
                return
            elif int(page[0]) <= 0:
                QMessageBox.about(self, "提示", "页数过小")
                return
            # 跳转到指定页面
            self.page_num = int(page[0]) - 1
            self.updatePdfView()
        elif cmd.__contains__('上一页'):
            self.onPrepage()
        elif cmd.__contains__('下一页'):
            self.nextpage()
        elif cmd.__contains__('几点'):
            # 返回当前时刻
            QMessageBox.about(self, "提示", time.strftime("%Y-%m-%d %H:%M:%S",
time.localtime()))
```

```
        elif cmd.__contains__('翻译'):
            # 取出翻译内容
            if self.book_open == False:
                QMessageBox.about(self, "提示", "请先打开一个文件")
                return
            elif self.selection_text == "":
                QMessageBox.about(self, "提示", "选中内容为空")
                return
            result = translate('en','zh', self.selection_text) # 英译中结果
            # content = '原文：'+self.selection_text+'\n'+'译文：'+result
            self.editor.appendText(result)
            self.doc2.setVisible(True)

        elif cmd.__contains__('保存'):
            if self.book_open == False:
                QMessageBox.about(self, "提示", "请先打开一个文件")
                return
            elif self.selection_text == "":
                QMessageBox.about(self, "提示", "选中内容为空")
                return
            self.editor.appendText(self.selection_text)
            self.doc2.setVisible(True)
        elif cmd.__contains__('笔记'):
            content = cmd.split('笔记')[1]
            self.editor.appendText(content)
            self.doc2.setVisible(True)
```

6.5 Select text OCR recognition function

After opening the file, you need to render the pdf view, all through the generatePDFView function, which can convert the pdf into a pixel image, and then implement OCR.

If the selection operation has been performed, execute the draw_selection function, where the original image is first drawn and highlighted as a rectangle, then the selected area is saved as a picture, and the OCR operation is continued to obtain the final text.

If no selection operation is performed, the statement after else is executed.

```
def generatePDFView(self):
        if not self.file_path or not self.doc:
            return
        pix = self.doc[self.page_num].getPixmap(matrix=self.trans)
        fmt = QImage.Format_RGBA8888 if pix.alpha else QImage.Format_RGB888
        pageImage = QImage(pix.samples, pix.width, pix.height, pix.stride, fmt)
        pixmap = QPixmap()
        pixmap.convertFromImage(pageImage)

        if self.area != None:
            self.draw_selection(pixmap)
            self.area = None
        else:
            self.pdfview.setPixmap(pixmap)
            # print('pixmap.size():', pixmap.size())
```

```
            self.pdfview.resize(pixmap.size())
```

We get the selected area through eventFilter, extract_selected_text, we read the area drawn by the left mouse button, get the selected area, and then execute the rendering operation again to get the picture.

```
def eventFilter(self, obj, event):
        if obj is self.pdfview:
            if event.type() ==  QEvent.MouseButtonPress:
                self.selection_start = event.pos()
            elif event.type() ==  QEvent.MouseButtonRelease:
                self.selection_end = event.pos()
                self.extract_selected_text()
        return super().eventFilter(obj, event)
```

6.6 Translate

If there is translation in the keyword spoken by the user, the Baidu translation interface is called to translate; the save operation is to add the OCR result to the last line of the note.

```
def translate(from_lang, to_lang, query):
    # Set your own appid/appkey.
    appid = '20230615001713542'
    appkey = 'i_boQF8PhuyUhu3sfjaI'

    endpoint = 'http://api.fanyi.baidu.com'
    path = '/api/trans/vip/translate'
    url = endpoint + path

    # Generate salt and sign
    def make_md5(s, encoding='utf-8'):
        return md5(s.encode(encoding)).hexdigest()

    salt = random.randint(32768, 65536)
    sign = make_md5(appid + query + str(salt) + appkey)

    # Build request
    headers = {'Content-Type': 'application/x-www-form-urlencoded'}
    payload = {'appid': appid, 'q': query, 'from': from_lang, 'to': to_lang,
'salt': salt, 'sign': sign}

    # Send request
    r = requests.post(url, params=payload, headers=headers)
    result = r.json()

    # Show response
    print(json.dumps(result, indent=4, ensure_ascii=False))

    ret = ""
    for res in result["trans_result"]:
        ret += res["dst"]
```

```
    return ret
```

6.7 Gesture control

```
def start(self, pdf):
                """
                Entry point of whole programm, caputres video frame and passes,
obtains
                landmark from mediapipe and passes it to 'handmajor' and
'handminor' for
                controlling.
                """

                handmajor = HandRecog(HLabel.MAJOR)
                handminor = HandRecog(HLabel.MINOR)

                with mp_hands.Hands(max_num_hands=2,
min_detection_confidence=0.5,
                                    min_tracking_confidence=0.5) as hands:
                    while GestureController.cap.isOpened() and
GestureController.gc_mode:
                        success, image = GestureController.cap.read()


                        if not success:
                            print("Ignoring empty camera frame.")
                            continue
                        img_height, img_width, img_channels = image.shape
                        image = cv2.cvtColor(cv2.flip(image, 1),
cv2.COLOR_BGR2RGB)
                        image.flags.writeable = False
                        results = hands.process(image)

                        image.flags.writeable = True
                        image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

                        if results.multi_hand_landmarks:
                            GestureController.classify_hands(results)

 handmajor.update_hand_result(GestureController.hr_major)

 handminor.update_hand_result(GestureController.hr_minor)

                            handmajor.set_finger_state()
                            handminor.set_finger_state()
                            gest_name = handminor.get_gesture()

                            if gest_name == Gest.PINCH_MINOR:
                                GestureController.handle_controls(gest_name,
handminor.hand_result, pdf)
                            else:
```

```
                                gest_name = handmajor.get_gesture()
                                GestureController.handle_controls(gest_name,
handmajor.hand_result, pdf)

                            for hand_landmarks in results.multi_hand_landmarks:
                                mp_drawing.draw_landmarks(image, hand_landmarks,
mp_hands.HAND_CONNECTIONS)
                        else:
                            GestureController.prev_hand = None

                        window_name = 'Gesture Controller'
                        cv2.namedWindow(window_name, cv2.WINDOW_NORMAL)

                        cv2.resizeWindow(window_name, int(img_width * 0.75),
int(img_height * 0.75))  # 设置窗口大小为800x600
                        gesture_window_x = pdf.screenWidth - int(img_width *
0.75) - 1
                        gesture_window_y = pdf.screenHeight - int(img_height *
0.75) - 40
                        cv2.moveWindow(window_name, gesture_window_x,
gesture_window_y)
                        cv2.imshow(window_name, image)
                        if cv2.waitKey(1) == 27:
                            break
                GestureController.cap.release()
                cv2.destroyAllWindows()

        gc1 = GestureController()
        gc1.start(self)
```

6.8 Gesture control core function

Perform various operations by recognizing gestures. Can recognize various gestures at the beginning.

```
def handle_controls(gesture, hand_result, pdf):
                """Impliments all gesture functionality."""
                x, y = None, None
                if gesture != Gest.PALM:
                    x, y = GestureController.get_position(hand_result)

                # flag reset
                if gesture != Gest.FIST and GestureController.grabflag:
                    GestureController.grabflag = False
                    pyautogui.mouseUp(button="left")

                if gesture != Gest.PINCH_MAJOR and
GestureController.pinchmajorflag:
                    GestureController.pinchmajorflag = False

                if gesture != Gest.PINCH_MINOR and
GestureController.pinchminorflag:
                    GestureController.pinchminorflag = False

                # implementation
```

```
                if gesture == Gest.V_GEST:
                    GestureController.flag = True
                    pyautogui.moveTo(x, y, duration=0.1)

                elif gesture == Gest.FIST:
                    if not GestureController.grabflag:
                        GestureController.grabflag = True
                    #     pyautogui.mouseDown(button="left")
                    # pyautogui.moveTo(x, y, duration=0.1)
                    # print('close--close--close--close--close--close--close--
close')
                    GestureController.gc_mode = 0

                elif gesture == Gest.MID and GestureController.flag:
                    pyautogui.click()
                    GestureController.flag = False

                elif gesture == Gest.INDEX and GestureController.flag:
                    # pyautogui.click(button='right')
                    # GestureController.flag = False
                    print('打开语音')
                    pdf.onMicrophone()
                    GestureController.flag = False

                elif gesture == Gest.TWO_FINGER_CLOSED and
GestureController.flag:
                    pyautogui.doubleClick()
                    GestureController.flag = False

                elif gesture == Gest.PINCH_MINOR:
                    if GestureController.pinchminorflag == False:
                        GestureController.pinch_control_init(hand_result)
                        GestureController.pinchminorflag = True
                    GestureController.pinch_control(hand_result,
GestureController.scrollHorizontal,

 GestureController.scrollVertical, pdf)

                elif gesture == Gest.PINCH_MAJOR:
                    if GestureController.pinchmajorflag == False:
                        GestureController.pinch_control_init(hand_result)
                        GestureController.pinchmajorflag = True
                    GestureController.pinch_control(hand_result,
GestureController.changesystembrightness,

 GestureController.changesystemvolume, pdf)
```

6.9 Gestures

```
# Gesture Encodings
class Gest(IntEnum):
    # Binary Encoded
    """
    Enum for mapping all hand gesture to binary number.
```

```
    """

    FIST = 0    #拳
    PINKY = 1   #小指
    RING = 2    #无名指，戴戒指的那个手指头
    MID = 4     #中指
    INDEX = 8   #食指
    THUMB = 16  #拇指
    PALM = 31   #手掌

    # Extra Mappings
    V_GEST = 33     #V型手势，相当于单纯移动鼠标位置
    TWO_FINGER_CLOSED = 34  #食指和中指并拢，鼠标左键
    PINCH_MAJOR = 35    #拿捏表情包的那个“捏”的动作，右手
    PINCH_MINOR = 36    #拿捏表情包的那个“捏”的动作，左手
```

```
def scrollVertical(pdf):
                """scrolls on screen vertically."""
                # pyautogui.scroll(120 if GestureController.pinchlv > 0.0 else
-120)
                currentVolumeLv = GestureController.pinchlv / 50.0
                if currentVolumeLv > 0.1:
                    currentVolumeLv = 0.1
                elif currentVolumeLv < -0.1:
                    currentVolumeLv = -0.1
                pdf.currentScrollBarlv += (-1.0) * currentVolumeLv
                if pdf.currentScrollBarlv > 1.0:
                    pdf.currentScrollBarlv = 1.0
                elif pdf.currentScrollBarlv < 0.0:
                    pdf.currentScrollBarlv = 0.0
                print('----------------------------------------',
pdf.currentScrollBarlv)
                pdf.scrollarea.verticalScrollBar().setValue(
                    pdf.currentScrollBarlv *
pdf.scrollarea.verticalScrollBar().maximum())
                # print(pdf.scrollarea.verticalScrollBar().maximum())
                # print(pdf.scrollarea.verticalScrollBar().minimum())
                time.sleep(0.5)
```

6.9.1 Function to zoom in/out the page

`currentVolumeLv = GestureController.pinchlv / 5.0` is the threshold adjustment. If you want to respond faster, choose a number greater than 5.

```
 def scrollHorizontal(pdf):
                currentVolumeLv = GestureController.pinchlv / 5.0
                print('----------------------------------------',
currentVolumeLv)
                if currentVolumeLv > 1.0:
                    currentVolumeLv = 1.0
                elif currentVolumeLv < -1.0:
                    currentVolumeLv = -1.0
                else:
```

```
                    if currentVolumeLv >= 0.0:
                        pdf.enlargepage()
                        print('放大页面')
                    else:
                        pdf.shrinkpage()
                        print('缩小页面页')
                # volume.SetMasterVolumeLevelScalar(currentVolumeLv, None)
```

7 How to run

```
conda create --name hci python=3.7.12
conda activate hci
pip install -r requirements.txt
python main.py
```