**Final Report**

**项目名称: 文字识别安卓程序**

**学科: 软件架构**

**组长姓名: 郑宜静(201630666516)**

**联系方式: 13546817022**

**Email: 13546817022@163.com**

**小组成员: 李胜蓝(201630664840)**

**李榷基(201630664833)**

**岳靖喆(201630666295)**

**时间：2018年12月22日**

1. **项目名称**

文字识别安卓应用程序

1. **项目概述**

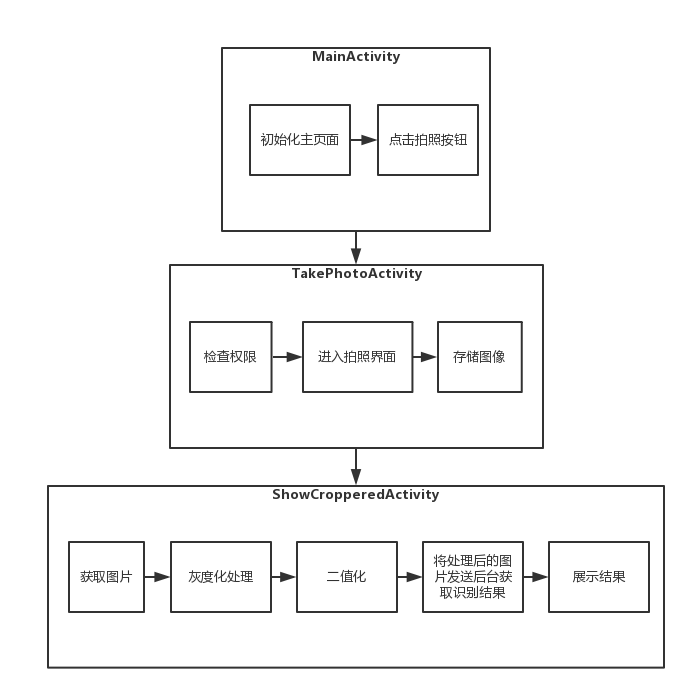
实现使用手机摄像头识别英文字符功能，基于Google Tesseract-OCR实现，由于这是基于C++开发，Android中不能直接使用，项目中使用tess-two是对于Android的分支，tess4j是对Java的分支，在识别之前先对图片进行预处理以增强识别效果。

1. **详细设计**
   1. **客户端流程图**
      1. **在手机客户端直接完成识别**

图片包含 屏幕截图

自动生成的说明

* + 1. **在客户端完成图像预处理再发送服务器获取识别结果**



* + 1. **在客户端获取原图发送服务器端获取识别结果**

图片包含 屏幕截图

自动生成的说明

* 1. **服务器端流程图**
     1. **对图像进行预处理流程**

图片包含 屏幕截图

自动生成的说明

* + 1. **对图像直接识别流程**

**图片包含 屏幕截图

自动生成的说明**

* 1. **模块结构**
     1. **图片获取模块**

这个模块在安卓客户端实现，功能是获取带有英文字符的图片，通过调用用户设备相机或用户从手机储存中调用一张图片。

* + 1. **图片处理模块**

图片的处理流程主要是增强图片亮度、图片灰度化、图片二值化，这些操作可以增强文字识别的精确度。

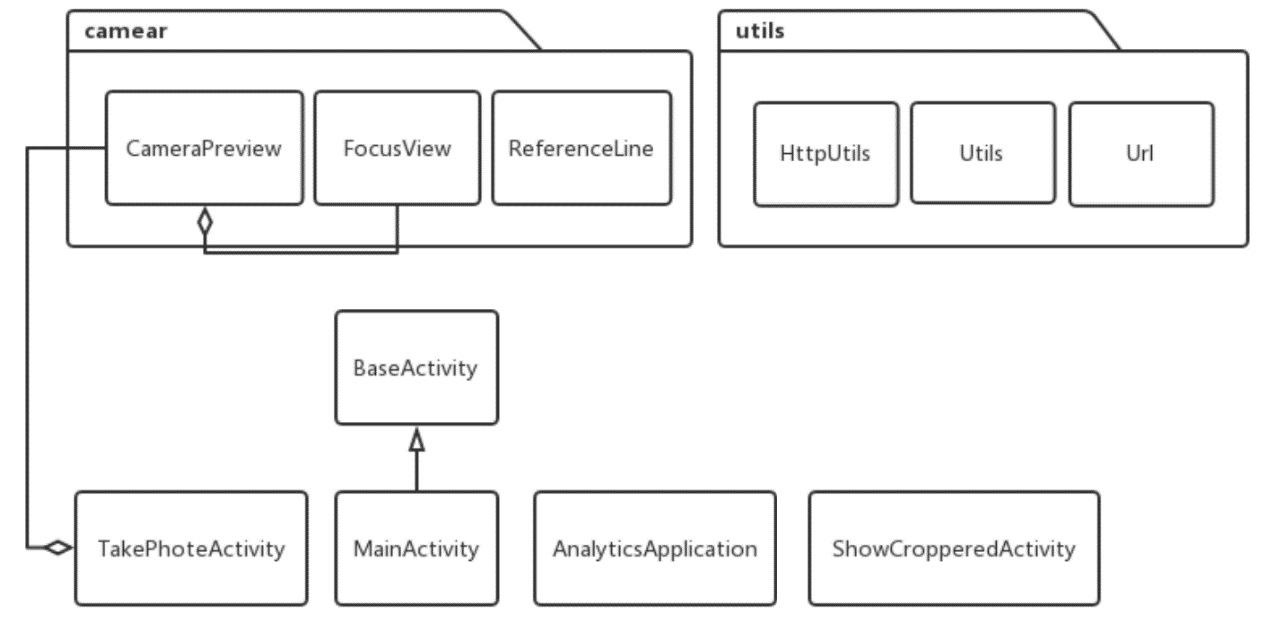
我们分别在手机客户端和服务器端实现了这个模块，用于比较客户端处理图片和服务器端处理图片速度的差异和对系统整体性能的影响。

* + 1. **Tesseract-OCR 模块**

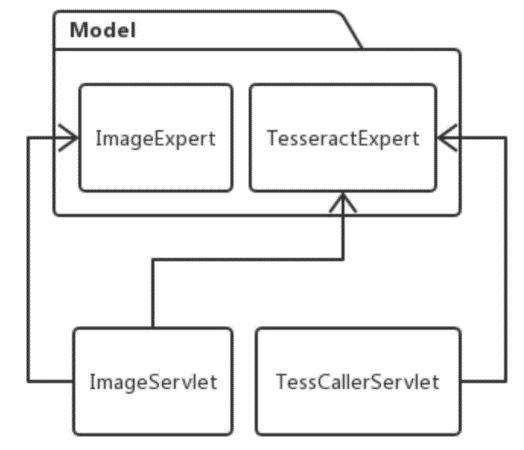
这个模块是基于Google Tesseract-OCR实现对图片中英文字符的识别，首先获取对应语言库文件(英语)，然后调用内置API进行识别。

我们同样在手机客户端和服务器端分别实现了这个模块，用于比较客户端和服务器端计算速度的差异。

* 1. **类**
     1. **前端类**



* + - 1. **MainActivity**
         1. 初始化页面，复制出assets中的语言库文件，点击跳转到拍照页面
      2. **TakePhoteActivity 主要执行拍照、裁剪操作，**
         1. 拍照成功后回调接口，存储图片并显示截图界面
         2. 剪裁操作，完成后跳转到识别界面
      3. **ShowCropperedActivity 主要是识别操作**
         1. 初始化识别器
         2. 进行图片灰度化和二值化处理，去除杂色可以提高准确度
         3. 识别
    1. **后端类**

****

* + - 1. **ImageExpert**
         1. 用于对图片进行灰度化和二值化处理
      2. **TesseractExpert** 
         1. 对图片进行识别提取图片中的英文及数字字符
      3. **ImageServlet**
         1. 接收未经过预处理图片
         2. 进行图片灰度化和二值化处理
         3. 识别
         4. 将识别结果返回给客户端
      4. **TessCallerServlet** 
         1. 接收经过预处理完成的图片
         2. 识别
         3. 将识别结果返回给客户端

1. **实验与结果**
   1. **实验目的**

实现一个计算密集型和延迟敏感的移动应用程序来实现并测试其性能，找到有哪些影响该应用程序性能的因素以及它们是如何影响性能的。

* 1. **实验流程**
     1. 选择一个计算密集型和延迟敏感的移动应用程序（具体为上述的文字识别安卓程序）
     2. 分析程序的功能及其模块的划分
     3. 设计图像预处理流程及API模块
     4. 解决安卓程序源代码中的bug并改进图像处理和识别环节
     5. 完成后端API的实现及测试
     6. 编写安卓客户端的网络请求，并改进结果展示部分的代码
     7. 对安卓进行线程阻塞以顺利展示网络数据
     8. 进行时间测试及计算

数据测试手段：代码中插入获取时间函数并在控制台输出差值

* 1. **环境设置**

**开发环境：**

操作系统：Windows 10、

IDE：Android Studio、Intellij IDEA

**实验环境（在开发环境的基础上增加手机设备）：**

手机设备：

Xiaomi MI 5 (Android 8.0，API 26)

HUAWEI M2-801W (Android 6.0，API 23)

Nexus 5X (Virtual Device，API 28)

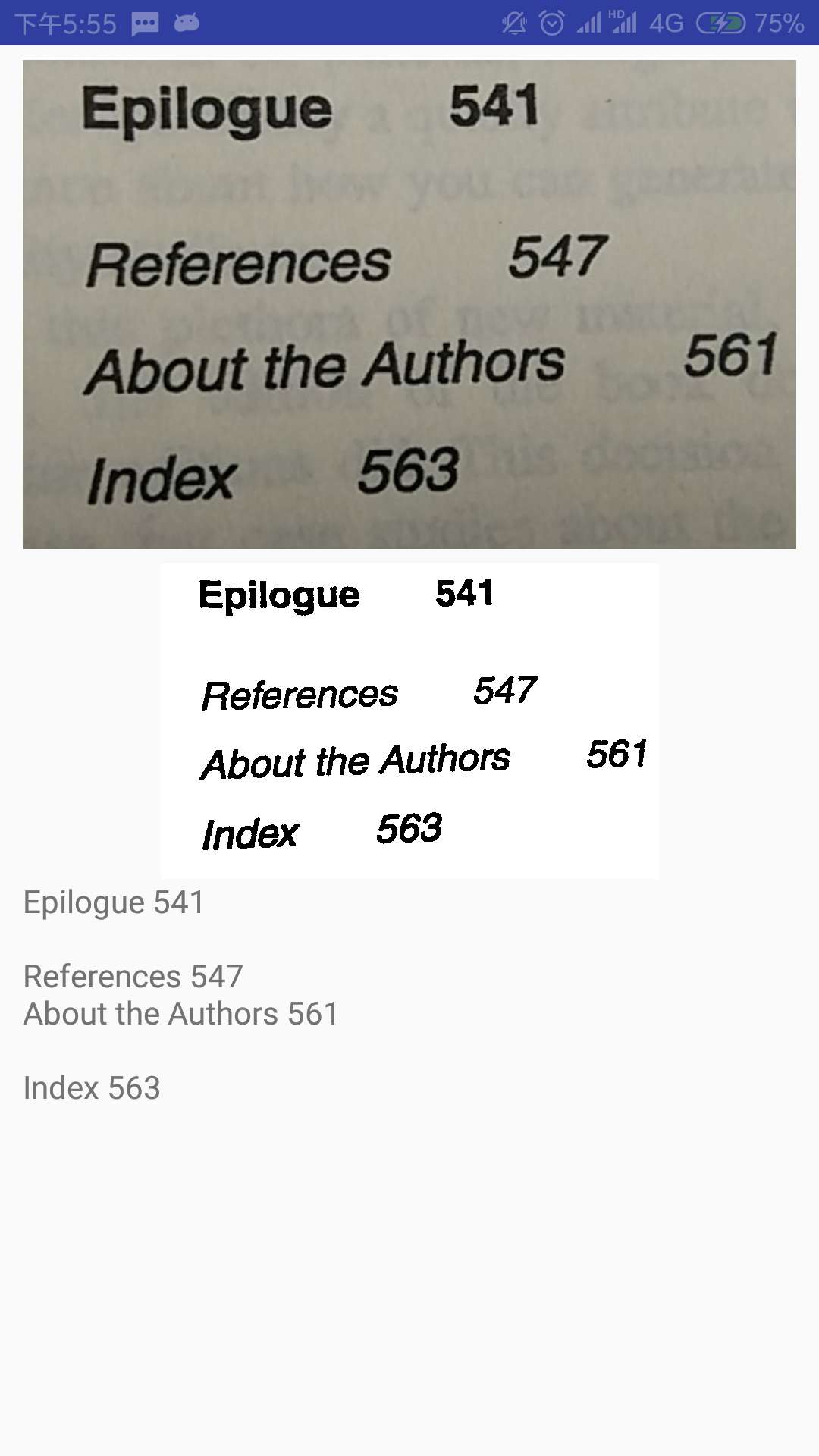
**生产环境：**

操作系统：Android API 21以上、Windows 10

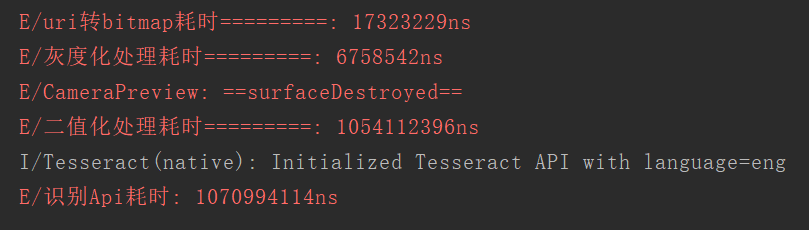
* 1. **实验结果(附图)**
     1. **将图片处理和识别模块置于安卓客户端（无需网络）**
        1. **测试1**
           1. **测试条件**

设备：小米手机5

* + - * 1. **识别结果**

****

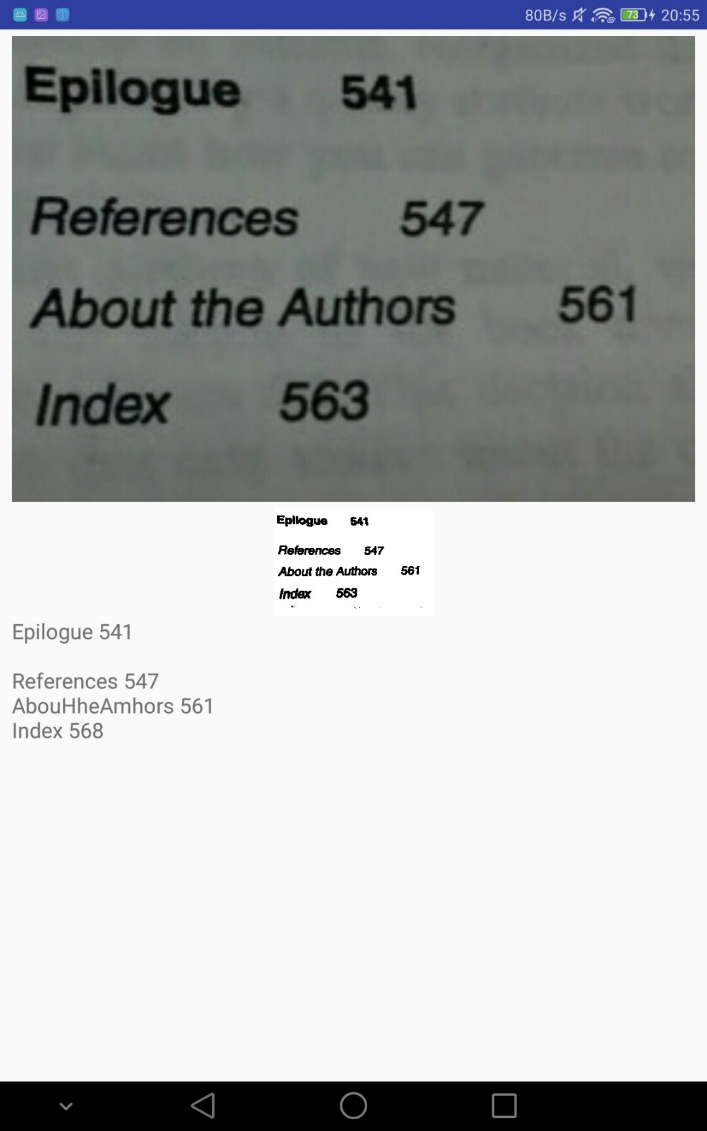
* + - * 1. **前端反馈**

****

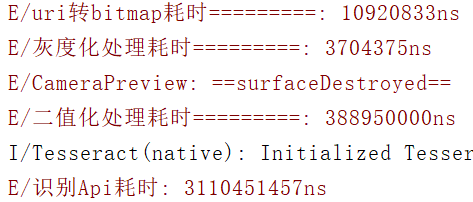
* + - 1. **测试2**
         1. **测试条件**

设备：华为平板M2-801W

* + - * 1. **识别结果**

****

* + - * 1. **前端反馈**

****

* + - 1. **其他结果**



* + 1. **将图片处理模块和识别英文字符模块均置于后台（由于图像处理模块置于后台，此模式下不展示处理后图片）**
       1. **测试1**
          1. **测试条件**

设备：小米手机5

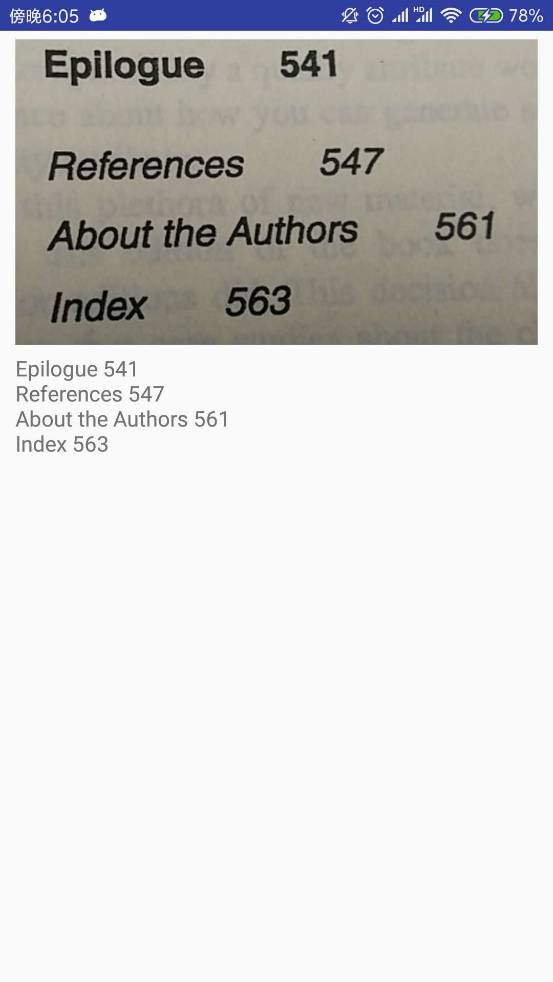
网络：华工图书馆校园网：

Download speed: 4.7 Mbps

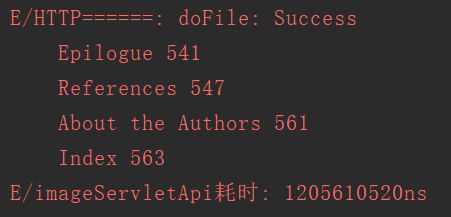
Upload speed: 5.4 Mbps

Ping: 6 ms

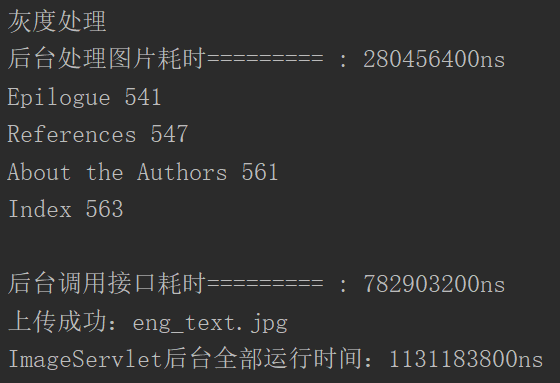
* + - * 1. **识别结果**



* + - * 1. **前端反馈**



* + - * 1. **服务器端反馈**



* + - 1. **测试2**
         1. **测试条件**

设备：小米手机5

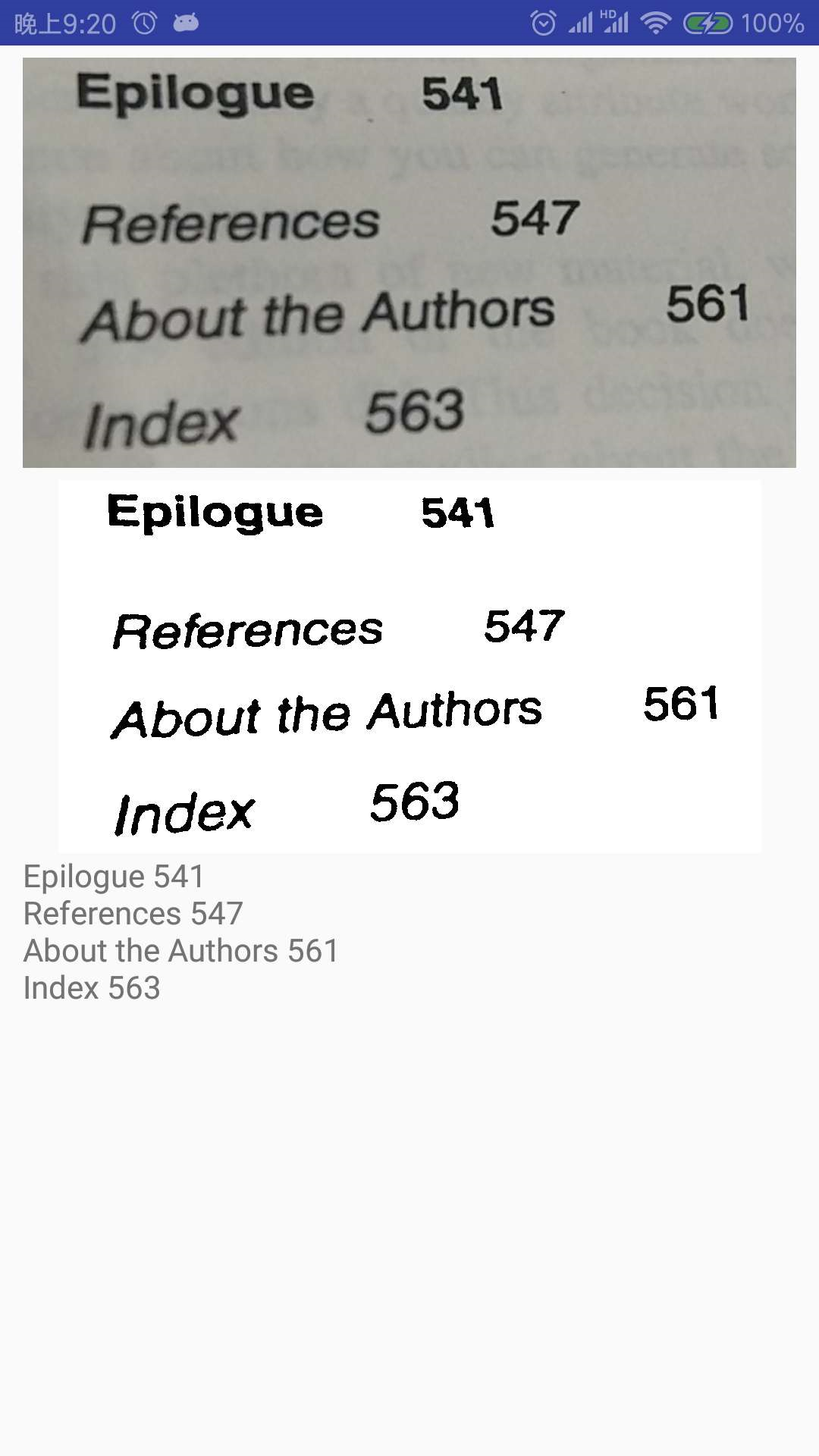
网络：华工宿舍校园网：

Download speed: 6.4 Mbps

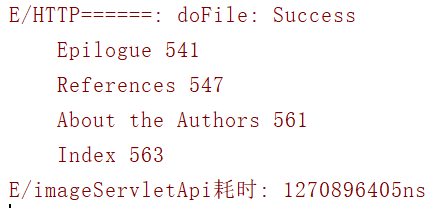
Upload speed: 2.6 Mbps

Ping: 7 ms

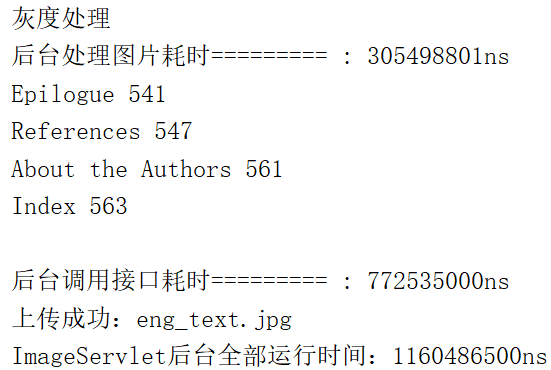
* + - * 1. **识别结果**



* + - * 1. **前端反馈**



* + - * 1. **服务器端反馈**



* + - 1. **测试3**
         1. **测试条件**

设备：华为平板M2-801W

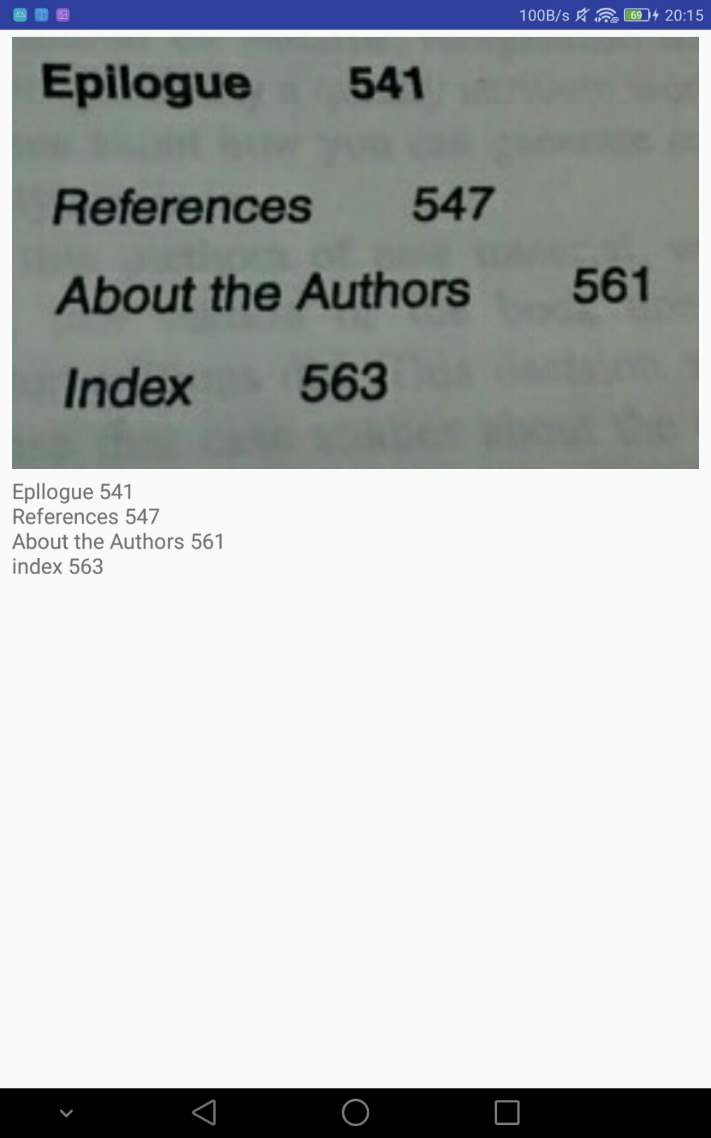
网络：华工宿舍校园网：

Download speed: 5.8 Mbps

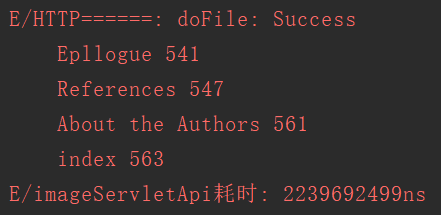
Upload speed: 8.1 Mbps

Ping: 7 ms

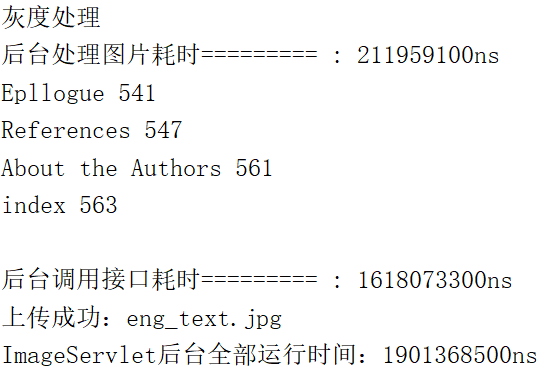
* + - * 1. **识别结果**



* + - * 1. **前端反馈**



* + - * 1. **服务器端反馈**



* + - 1. **测试4**
         1. **测试条件**

设备：华为平板M2-801W

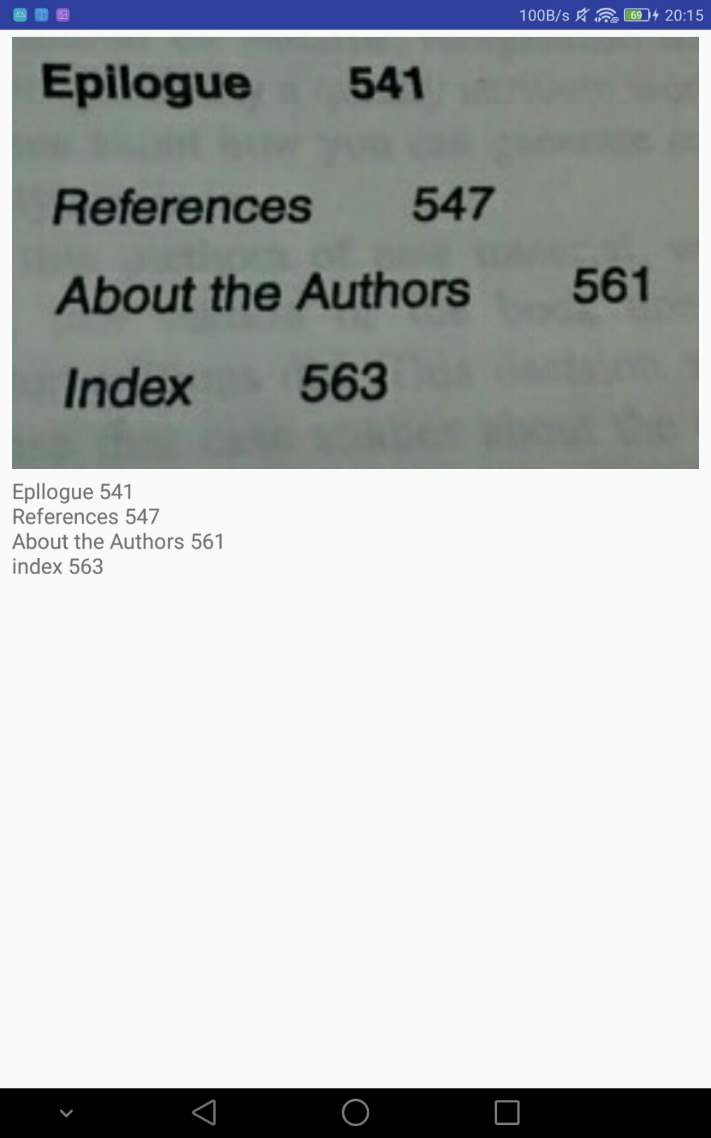
网络：手机4G网络

Download speed: 5.8 Mbps

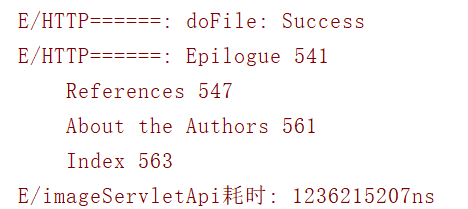
Upload speed: 8.1 Mbps

Ping: 7 ms

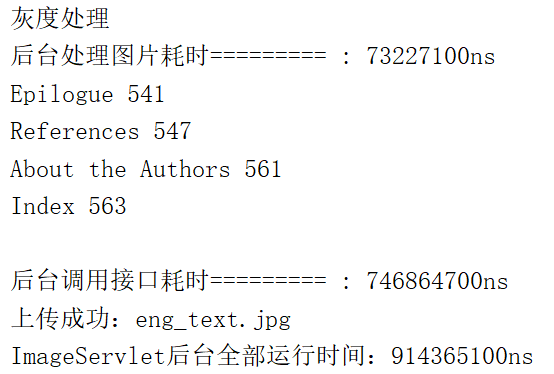
* + - * 1. **识别结果**



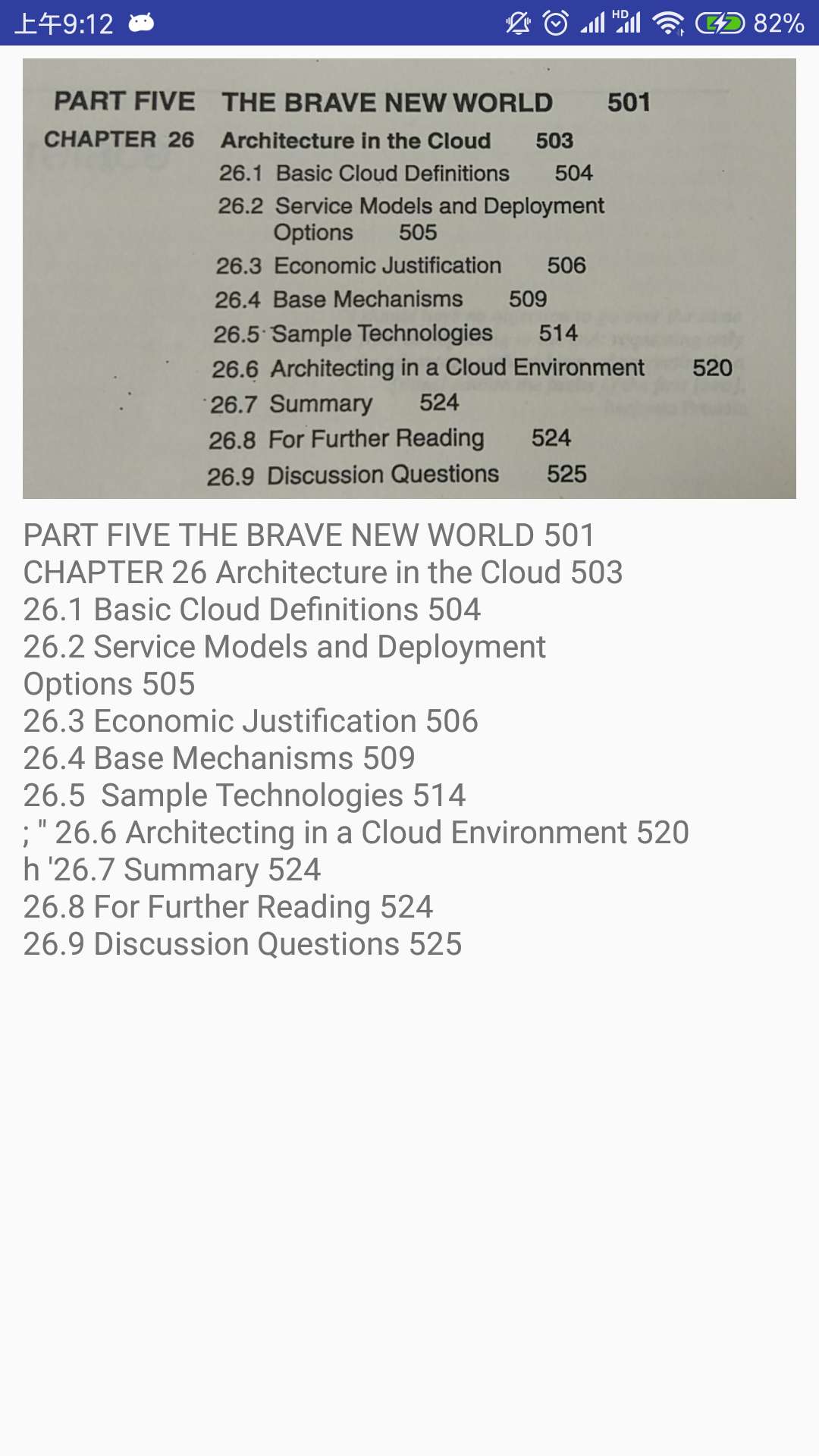
* + - * 1. **前端反馈**



* + - * 1. **服务器端反馈**



* + - 1. **其他结果**

****

* + 1. **将图片处理模块置于安卓端，将识别模块置于后台**
       1. **测试1**
          1. **测试条件**

设备：小米手机5

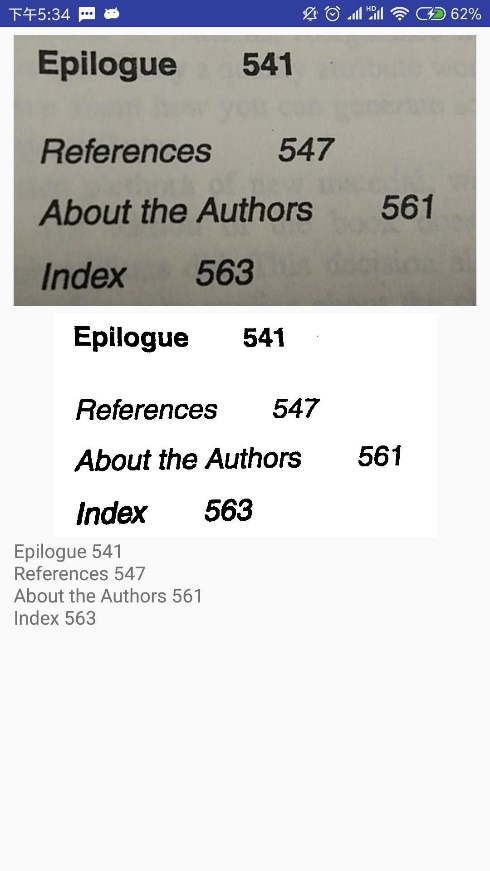
网络：华工图书馆校园网

Download speed: 3.7 Mbps

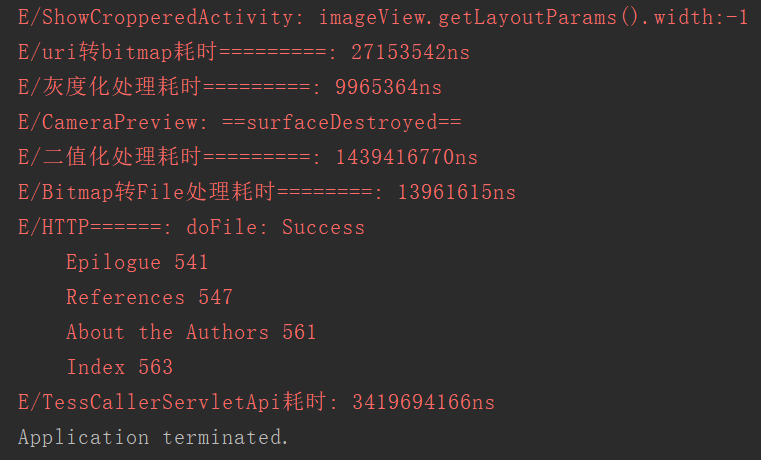
Upload speed: 2.4 Mbps

Ping: 7 ms

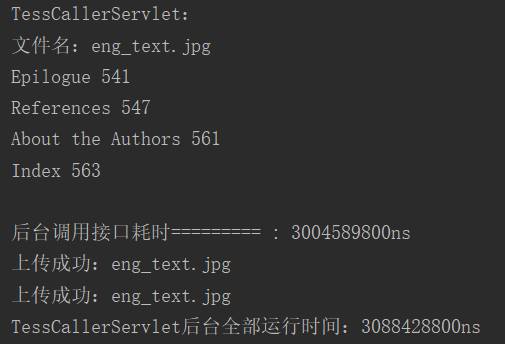
* + - * 1. **识别结果**



* + - * 1. **前端反馈**



* + - * 1. **服务器端反馈**



* + - 1. **测试2**
         1. **测试条件**

设备：小米手机5

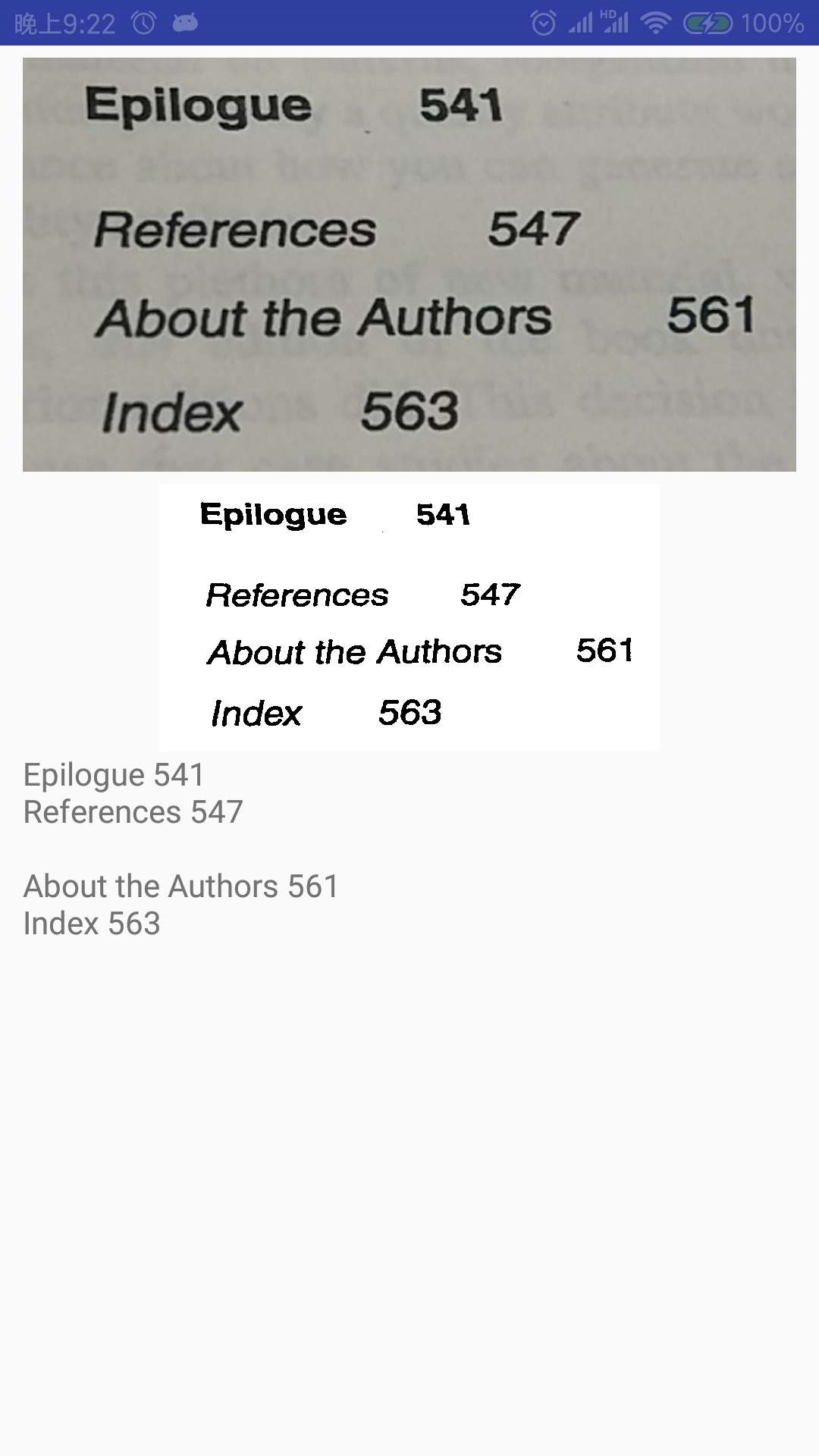
网络：华工宿舍校园网

Download speed: 7.1 Mbps

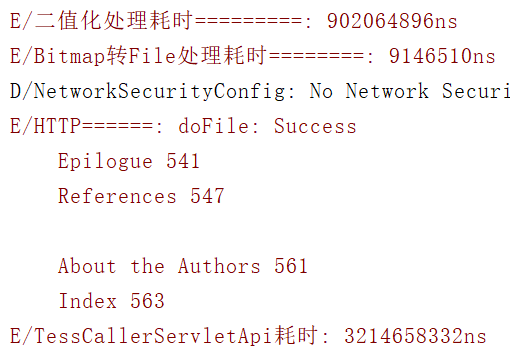
Upload speed: 4.1 Mbps

Ping: 7 ms

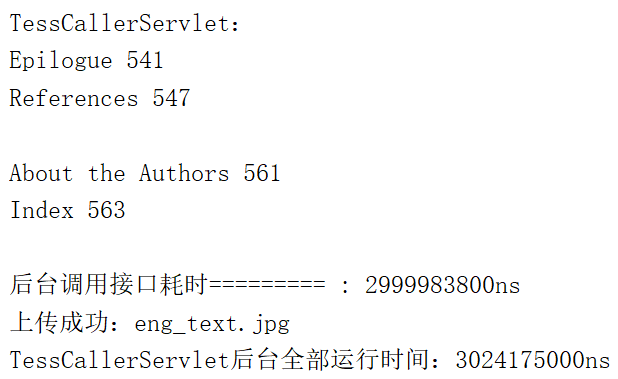
* + - * 1. **识别结果**



* + - * 1. **前端反馈**



* + - * 1. **服务器端反馈**



* + - 1. **测试3**
         1. **测试条件**

设备：华为平板M2-801W

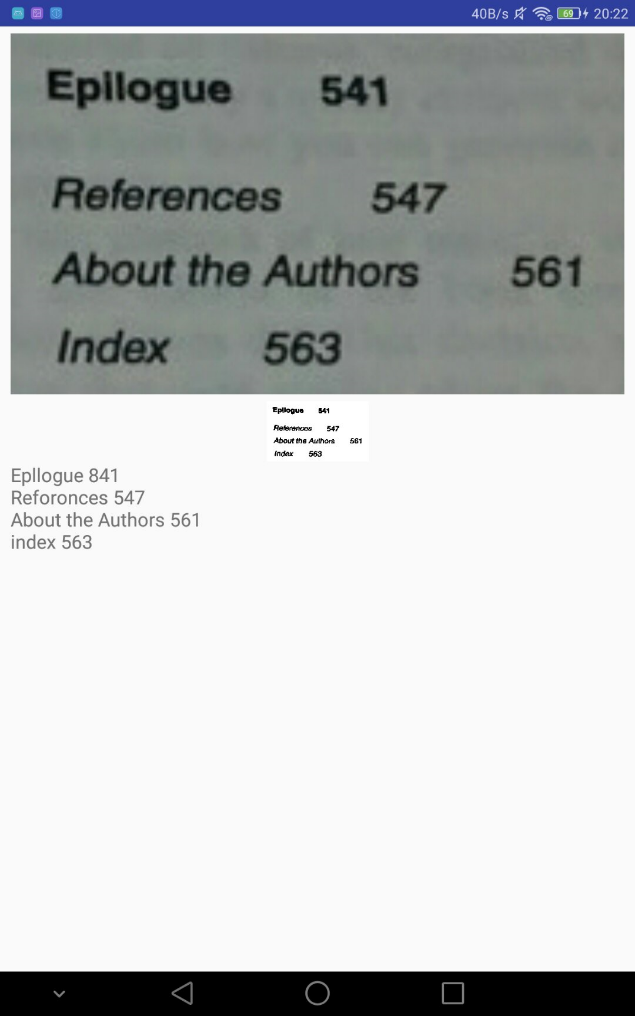
网络：华工宿舍校园网

Download speed: 7.1 Mbps

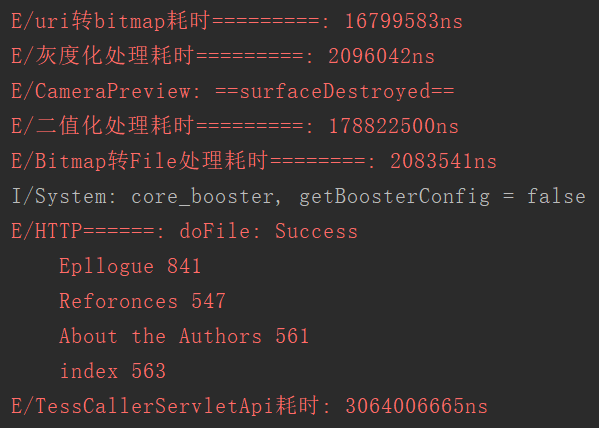
Upload speed: 4 Mbps

Ping: 7 ms

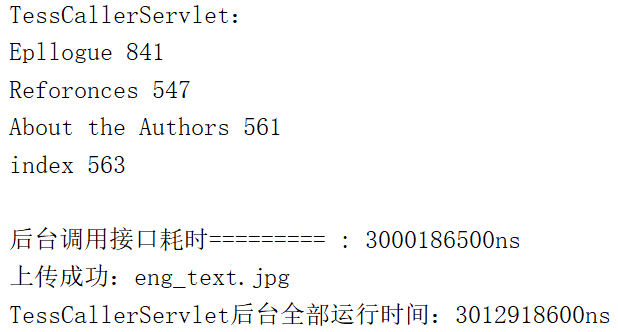
* + - * 1. **识别结果**



* + - * 1. **前端反馈**



* + - * 1. **服务器端反馈**



* + - 1. **测试4**
         1. **测试条件**

设备：华为平板M2-801W

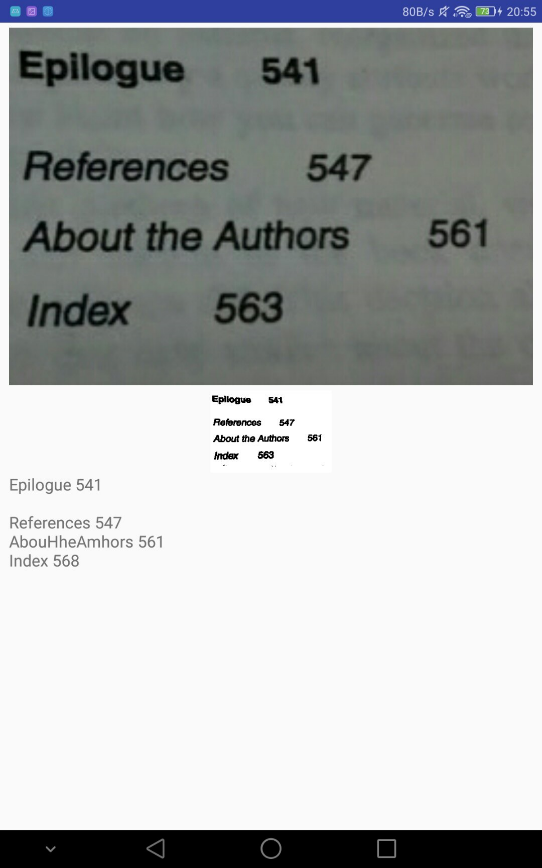
网络：手机4G网络

Download speed: 1.15 Mbps

Upload speed: 0.61 Mbps

Ping: 95.43 ms

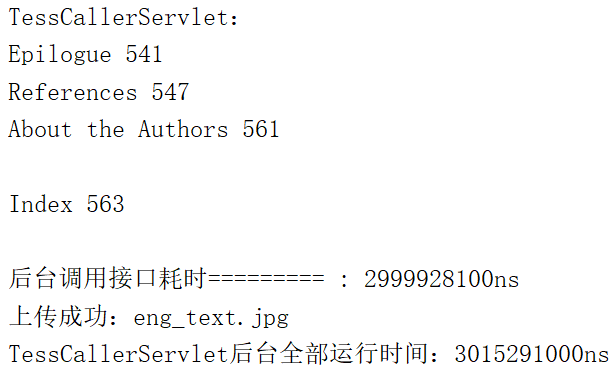
* + - * 1. **识别结果**



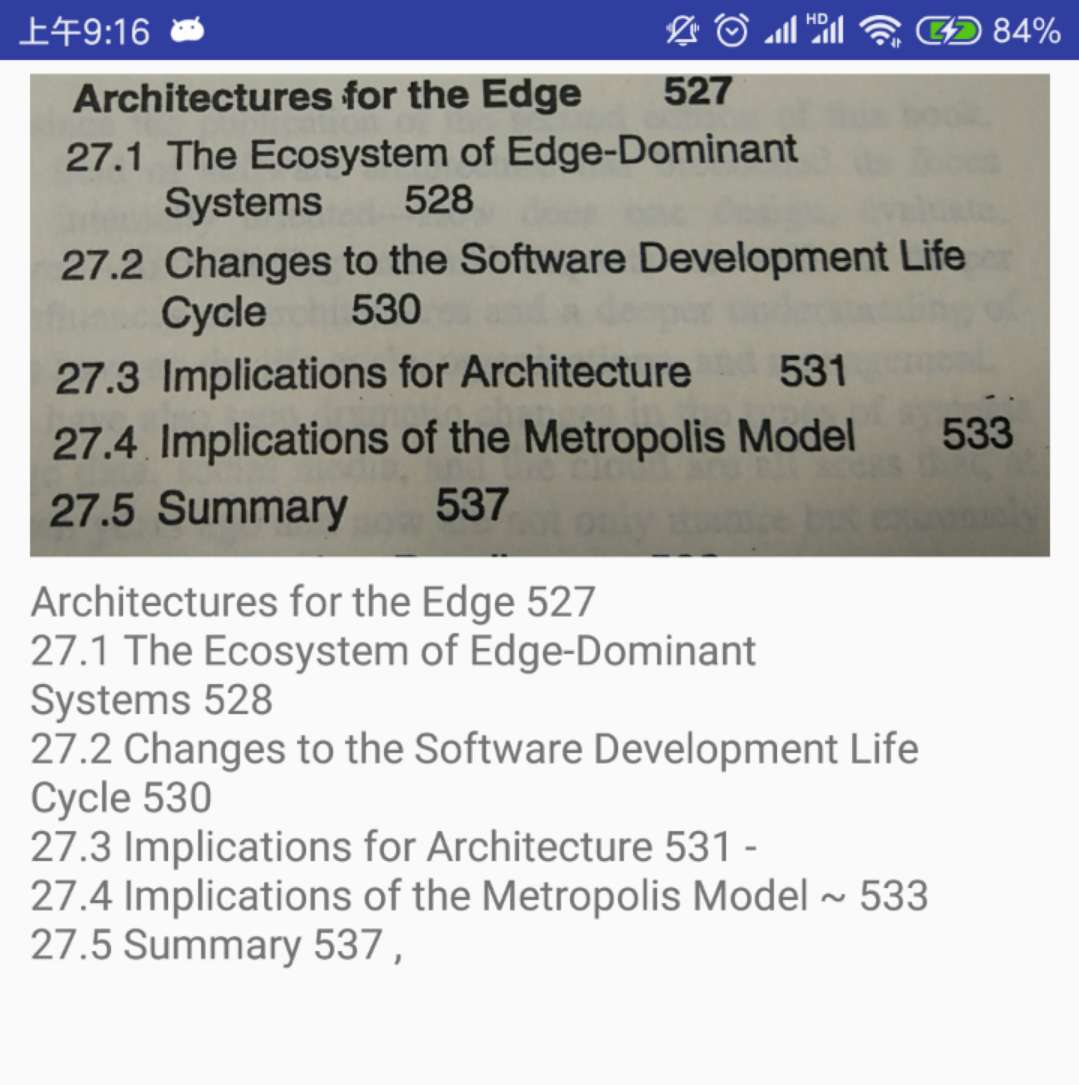
* + - * 1. **前端反馈**



* + - * 1. **服务器端反馈**



* + - 1. **其他结果**

****

1. **性能分析**
   1. **硬件因素**
2. **手机硬件配置：**

**在高性能的手机硬件配置情况下，可以提高系统性能从而减少计算耗时**。在实验中，发现相同条件下，小米手机5的数据明显比华为平板M2的数据优秀。

**手机摄像头的配置影响照片的质量进而影响识别的精确度。**华为平板由于摄像头较差，会频繁出现识别不准确的情况。

1. **服务器端硬件配置：**

在高性能的服务器硬件配置情况下，可以提高系统性能从而减少计算耗时。

* 1. **网络带宽因素**

**客户端网络宽带：**客户端网络宽带会直接影响到图像处理耗时以及发送接收时间。

**服务端网络宽带：**服务端网络宽带会直接影响到图像处理耗时以及发送接收时间。

**对比相同情况下的三种识别方法发现：**在网络情况比较好的情况下，性能最优的是直接将图片发送至后台，让后台处理所有的计算；其次是在手机本地完成所有的计算；最后是在手机客户端处理图片后到服务器完成识别。而在网络较差的情况下，在手机客户端直接完成所有工作消耗的时间不一定会比交付服务器计算所消耗的时间长。

* 1. **传输数据量因素**

**数据大小：**图片的大小会直接影响到图像处理耗时以及发送接收时间

**数据格式：**图片数据在预处理时有一个矩阵计算的过程，可以考虑将图片计算成小矩阵进行传输，再在后台恢复成图片模式

**我们经过截取不同大小的图片进行测试，发现发送大的图片耗时较长，小的图片耗时较短。**

1. **反思与总结**
   1. **程序未做的改进：**

在客户端应当先进行图片的压缩减小大小以缩短网络传输图片的时间消耗，由于小组对安卓程序不熟悉，此项工作并未开展。

* 1. **项目展开难点：**

源项目是纯粹的安卓应用，我们在阅读理解以及改动其代码时是其适配于我们的目标程序时遇到了许多阻碍。也因此有许多想法未能实现。

* 1. **心得体会：**

服务器端的计算速度远超过手机本地，因此复杂计算尽量部署到服务器上可以有效提高性能，但是网络带宽也是一个很大的影响因素，数据应当尽量在本地进行优化缩小以发送的数据量的体积，从而减小传输过程带来的时间损耗。

因此，若软件系统要做动态划分，可以根据实时网络情况进行权衡。例如，当软件检测到用户的网络时无线Wi-Fi时，说明此时用户的网络情况比较良好，应该优先考虑将大部分计算功能交给后台实现；相反，则应考虑在客户端本地实现项目功能。