### 系统功能性需求分析

本系统是一个基于生活场景帮助人们快速识别文字（OCR）并将识别结果播报（语音合成）出来，有助于不方便使用手机等手持设备时识别内容，并且能根据人们的需求实现语音识别功能。

本系统是Android 客户端的应用程序，方便用户使用手机等安卓手持设备实现识别文字（OCR）等功能。

### 文字识别需求核心技术

1. 文字识别过程主要有：预处理、特征提取、特征匹配、识别后处理。
2. 由于普通拍摄图片会存在干扰，所以要对图片进行预处理，增强图片，降低干扰。
   1. 图片灰度化技术

要对采集到的图片进行灰度化处理，主要是因为灰度图片相对彩色的图片抛出了一部分的数据信息，数据量更小，更容易处理，并且更能显示文字的特点。

RGB图片灰度化技术，通俗点说就是对图像的RGB三个分量进行加权平均得到最终的灰度值。最常见的加权方法如下：

1）Gray=B；Gray=G；Gray=R

2）Gray=max(B+G+R)

3）Gray=(B+G+R)/3

4）Gray= 0.072169B+ 0.715160G+ 0.212671R

5）Gray= 0.11B+ 0.59G+ 0.3R

这三种方法中，第一种为分量法，即用RGB三个分量的某一个分量作为该点的灰度值；第二种方法为最大值法，将彩色图像中的三分量亮度的最大值作为灰度 图的灰度值。第三种方法将彩色图像中的三分量亮度求平均得到一个灰度图；后两种都是属于加权平均法，其中第四种是OpenCV开放库所采用的灰度权值，第 五种为从人体生理学角度所提出的一种权值（人眼对绿色的敏感最高，对蓝色敏感最低）。

* + 1. 二值技术

图像的二值化处理就是将图像上的点的灰度值为0或255，也就是将整个图像呈现出明显的黑白效果。即将256个亮度等级的灰度图像通过适当的阈值选取而获得仍然可以反映图像整体和局部特征的二值化图像。在数字图像处理中，二值图像占有非常重要的地位，特别是在实用的图像处理中，以二值图像处理实现而构成的系统是很多的，要进行二值图像的处理与分析，首先要把灰度图像二值化，得到二值化图像，这样子有利于在对图像做进一步处理时，图像的集合性质只与像素值为0或255的点的位置有关，不再涉及像素的多级值，使处理变得简单，而且数据的处理和压缩量小。

为了得到理想的二值图像，一般采用封闭、连通的边界定义不交叠的区域。所有灰度大于或等于阈值的像素被判定为属于特定物体，其灰度值为255表示，否则这些像素点被排除在物体区域以外，灰度值为0，表示背景或者例外的物体区域。 如果某特定物体在内部有均匀一致的灰度值，并且其处在一个具有其他等级灰度值的均匀背景下，使用阈值法就可以得到比较的分割效果。

如果物体同背景的差别表现不在灰度值上（比如纹理不同），可以将这个差别特征转换为灰度的差别，然后利用阈值选取技术来分割该图像。动态调节阈值实现图像的二值化可动态观察其分割图像的具体结果。

OpenCV中有两个函数可以实现图片的二值化：

cvThreshold( dst, dst,230 , 255, CV\_THRESH\_BINARY\_INV);

cvAdaptiveThreshold( dst, dst, 255, CV\_ADAPTIVE\_THRESH\_MEAN\_C,CV\_THRESH\_BINARY, 9, -10);

方法（1）是手动指定一个阈值，以此阈值来进行二值化处理。其中的第四个参数决定了该方法的结果：

threshold\_type=CV\_THRESH\_BINARY:

dst(x,y) = max\_value, if src(x,y)>threshold 0, otherwise.

threshold\_type=CV\_THRESH\_BINARY\_INV:

dst(x,y) = 0, if src(x,y)>threshold; dst(x,y) = max\_value, otherwise.

threshold\_type=CV\_THRESH\_TRUNC:

dst(x,y) = threshold, if src(x,y)>threshold; dst(x,y) = src(x,y), otherwise.

threshold\_type=CV\_THRESH\_TOZERO:

dst(x,y) = src(x,y), if (x,y)>threshold ; dst(x,y) = 0, otherwise.

threshold\_type=CV\_THRESH\_TOZERO\_INV:

dst(x,y) = 0, if src(x,y)>threshold ; dst(x,y) = src(x,y), otherwise.

值得一说的是threshold\_type可以使用CV\_THRESH\_OTSU类型，这样该函数就会使用大律法OTSU得到的全局自适应阈值来进行二值化图片，而参数中的threshold不再起作用。比如：cvThreshold( dst, dst,300 , 255, CV\_THRESH\_OTSU | CV\_THRESH\_BINARY\_INV);这种方法对于灰度直方图呈现二峰特征的图片处理起来效果很好。

方法（2）是一个自适应阈值二值化方法，通过设定最后两个参数来调整效果。

* + 1. 基于统计模型（隐马尔可夫模型）语义识别与纠错

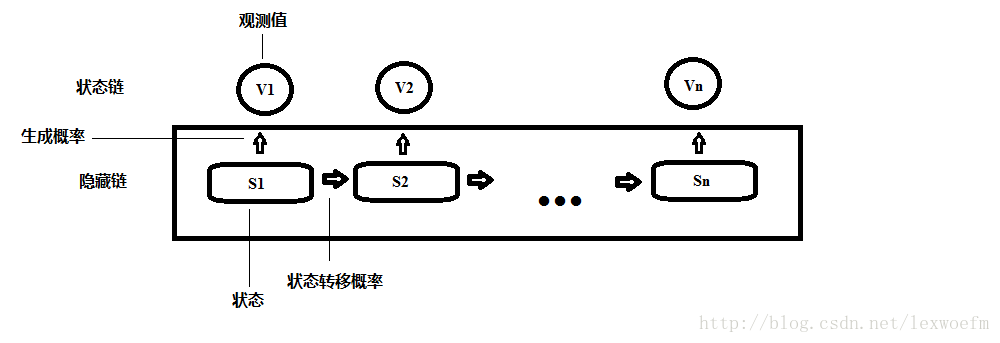
隐马尔可夫模型（Hidden Markov Model，HMM）创立于20世纪70年代。主要用于行为识别，语音识别，文字识别等。它用来描述一个含有隐含未知参数的马尔可夫过程。其难点是从可观察的参数中确定该过程的隐含参数。然后利用这些参数来作进一步的分析，例如模式识别。

是在被建模的系统被认为是一个马尔可夫过程与未观测到的（隐藏的）的状态的统计马尔可夫模型。

* + 1. 隐马尔可夫原理简述

隐马尔可夫模型由五个部分组成：状态空间S，观测空间O，初始状态概率空间PI，状态概率转移矩阵P以及观测值生成概率矩阵Q。另外，隐马尔可夫模型还包括一条观测链，一条隐藏链。（后面将详述）

下面是隐马尔可夫模型示意图：



因此整个过程就是观测值随状态的转移而生成，而我们所关心的是通过已有的观测值来判断其隐藏的状态，即通过一长串的观测序列推算导致这一结果的可能的状态序列。

* + 1. **实现方法**

要实现上面所述原理就必须解决三个问题：评估问题（evaluation），解码问题（decoding）和学习问题（learning）

1. **评估问题，即评估当前状态为真实状态的可能性。**

最简单的方法有前向算法和后向算法（当然也可以联合使用这两种算法）。

**前向算法：**从前递归，一层一层计算概率，最后再求总和。

1）t=0（事实上 t 的首项应该为1，但是考虑到编程的方便这里就设首项为0）

https://img-blog.csdn.net/20131117152419484?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvbGV4d29lZm0=/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/SouthEast，即 alpha(i,t)=PI(i)\*Q(i,t)（伪代码，这只是为了表示方便易懂，与之后的代码可能会有出入）

PS：alpha(i,t)指t时刻状态为Si的概率（下面同义），PI(i)为状态Si的初始概率，Q(i,t)指的是 t 时刻观测值Vt由状态Si生成的概率。

2）t>0 && t<=n

https://img-blog.csdn.net/20131117152517812?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvbGV4d29lZm0=/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/SouthEast，即alpha(t,i)=Sum[ alpha(j,t-1)\*P(j,i)\*Q(i,t) ]

PS：P(j,i)指由状态 Sj 转移到 Si 的概率

3）

https://img-blog.csdn.net/20131117152605859?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvbGV4d29lZm0=/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/SouthEast，即将3）所算的所有状态Si的结果再求和。

PS：对应后面的Java类为 AlgorithmFront.java

**后向算法：**与前向算法相反

1）t>=0 && t<n

https://img-blog.csdn.net/20131117152734484?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvbGV4d29lZm0=/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/SouthEast，即beta(i,t)=Sum[ beta(j,t+1)\*P(i,j)\*Q(j,t+1) ]

PS：beta(i,t)表示t时刻状态为Si的条件下，从t+1时刻到n生成相应观测序列的概率。

2）t=n

https://img-blog.csdn.net/20131117152705906?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvbGV4d29lZm0=/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/SouthEast

PS：因为下一个时刻就已结束，所以无论是什么状态都是确定的，所以概率都为1。

3）

https://img-blog.csdn.net/20131117152823265?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvbGV4d29lZm0=/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/SouthEast，与前向算法相似，最后也是将所有结果进行求和。

PS：对应后面的Java类为 AlgorithmBack.java

1. **解码问题，即如何根据观测值，状态转移概率矩阵，生成概率矩阵得到真正的状态序列。（有时候你完全可以根据先验知识给参数设值，这样就无需 学习步骤（Learning） 便可以解码了）**

Viterbi算法：基本原理就是计算概率每一步最高时对应的状态序列

1）初始化

https://img-blog.csdn.net/20131117152900015?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvbGV4d29lZm0=/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/SouthEast

2）递归

https://img-blog.csdn.net/20131117152929968?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvbGV4d29lZm0=/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/SouthEast

3）终止

https://img-blog.csdn.net/20131117153130046?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvbGV4d29lZm0=/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/SouthEast，https://img-blog.csdn.net/20131117153200265?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvbGV4d29lZm0=/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/SouthEast

PS：https://img-blog.csdn.net/20131117153448859?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvbGV4d29lZm0=/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/SouthEast表示 n 时刻沿着X1,X2,...Xn 且在 n 时刻状态Xn=Si 产生相应观测序列的最大概率

https://img-blog.csdn.net/20131117153343156?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvbGV4d29lZm0=/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/SouthEast保存着状态序列信息。

4）回溯

根据 https://img-blog.csdn.net/20131117153316984?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvbGV4d29lZm0=/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/SouthEast 的结果便可知道相应的状态序列了。

PS：对应的后面的Java类为 HMMDecisionVbImp.java

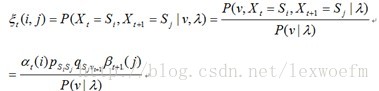
1. **学习问题，即如何通过观测值来获取初始状态概率，状态转移概率矩阵以及生成概率矩阵。**

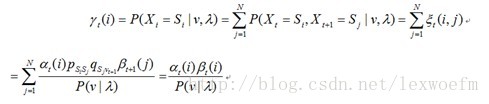
Baum-Welch算法：

Step1: 随机产生一组参数，并代入评估函数（evaluation，例如前向算法），计算结果。

Step2: 利用参数估算初始状态概率，状态转移概率矩阵以及生成概率矩阵

由于：

，即kis(i,j,t)=alpha(i,t)\*P(i,j)\*Q(j,t+1)\*beta(j,t+1)

，即gamma(i,t)=alpha(i,t)\*beta(i,t)

PS：a）伪代码中并没有除以https://img-blog.csdn.net/20131117154033984?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvbGV4d29lZm0=/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/SouthEast，这主要是为了减少运算量，因为之后计算状态概率矩阵、生成矩阵这项都会被约掉。

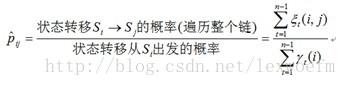
b）kis(i,j,t)即，表示t时刻为状态Si，t+1时刻为状态Sj的概率

c）gamma(i,t)即，表示t时刻状态为Si的概率

d）相应的Java类为 Gammas.java，Ksis.java

所以：

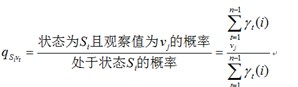
1）估计概率转移概率矩阵



2）估计初始状态概率

https://img-blog.csdn.net/20131117154207078?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvbGV4d29lZm0=/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/SouthEast，（注意，实际编程实现时这里还需除以之前漏除的https://img-blog.csdn.net/20131117154225453?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvbGV4d29lZm0=/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/SouthEast）

3）估计概率生成矩阵



Step3: 将刚估计的参数代入 评估函数 进行计算，并与上一次评估的结果做比较，若差异小于某个阈值（thresh，例如 0.05）则接受。否则继续迭代计算。

https://blog.csdn.net/xiaotianlan/article/details/38925197

* + 1. HMM 的基本概念

HMM 是马尔可夫链的一种，用于描述随机过程统计特性。 包括双重随机过程，一个是隐含的状态，它的状态不能直接观察 到。另一个是观测向量序列，每一个观测向量是由一个具有相 应概率密度分布的状态序列产生。

HMM 包括 5 个元素: ( 1) 隐含状态的数目 N，状态集为 S = { s1，s2，…，sN } ; ( 2) 观察序列的数目 M，观察序列集为 V = { v1，v2，…，vm } ; ( 3) 状态转移矩阵 A，A = { aij} ，aij = P［qt + 1 = sj | qt = si］， 1≤i，j≤N。其中 qt为在时刻 t 的状态。A 为 N × N 的方阵，行和 列都对应所有的状态，表示状态之间转移的概率; ( 4) 观察序列概率矩阵 B，B = { bj ( k) } ，bj ( k) = P［vt at t qt = sj］，1≤j≤N，1≤k≤M，连续型 HMM 的 B 通过一个连续的 函数得到观察序列与状态的关系，常用的是混合高斯概率密度 函数。 ( 5) 初始状态分布概率 ∏ = { πi} ，πi = P{ s1 = q1 } ，其中 1 ≤ i ≤ N。显然有∑ N i = 1 πi = 1。

HMM 主要可以解决 3 个基本问题:

1) 评估问题即给定一个模型 λ = { A，B，π} 和一组观察序 列 O = O1，O2，…，OT，计算在此模型下产生该观察序列的概率 P( O λ) 。评估问题用前向或后向算法解决。

2) 解码问题，即给定一个模型 λ = { A，B，π} 和一组观察序 列 O = O1，O2，…，OT，求解对应的最大概率的状态序列 Q = q1， q2，…，qT。解码问题用 Viterbi 算法解决。

3) 学习问题，即给定观察序列 O = O1，O2 …，OT，如何调整 模型参数 A，B，π，使得 P( O| λ) 最大。学习问题用 Baum-Welch 算法解决。

* + 1. OCR技术难点是什么？

复杂背景、艺术字体、低分辨率、非均匀光照、图像退化、字符形变、多语言混合、文本行复杂版式、检测框字符残缺，等等。

如克服这些难点的？从几个方面入手。一是使用场景，另一方面是从技术上的改进文本检测技术方进行了深度优化，通过设计合理的网络结构来提升各尺度的文字检测／提取能力。引入RNN多层自适应网络和Refinement结构来提升检测完整性和准确性。

##### Android 客户端需求分析

Android客户端应用程序具备多个文字识别的内容，帮助用户在不同场景下快速实现完成文字识别的功能（OCR），并且不方便查看识别结果时将结果语音合成播报，也需要满足人们不方便打字时通过语音识别功能识别人们话语。

图1 详细的分析安卓应用所需要的需求：



图 1

* 1. 系统功能
     1. 通用文字识别

具备调用手机摄像头拍摄任意带有文字的内容，比如一张图片中的文字，纸巾包装上的文字等。

* + 1. 身份证识别

可以识别身份证正反面的内容，识别出姓名，出生年月日，性别等内容。

* + 1. 银行卡识别

可以通过手机摄像头拍摄银行卡识别出银行卡的卡号与发卡行。

* + 1. 车牌识别

可以通过手机摄像头拍摄车牌号（可以是图片），识别出车牌号码。

* + 1. 车型识别

可以通过手机摄像头拍摄常见的车型（可以是图片），系统根据拍摄的图片识别出车型。

* + 1. 语音识别

可以识别出用户的语音，并且显示到手机屏幕上

* + 1. 语音合成

可以将文字识别的结果合成语音，通过手机扬声器播放出来，能设置语速，播讲人等参数。

* 1. 系统非功能性需求分析

整体应用对性能有较高的要求，调取摄像头时要注意防止内存泄漏，也不能出现程序无响应异常，需要站在用户的角度上体验整个应用，关键时刻出现错误，非常影响用户体验。

除此之外，进行应用开发时，心里对整个应用要保持着简洁、美观的要求，一个美观的界面与功能完整但是界面简陋，用户更加倾向于美观简洁的界面，同时，作为开发人人员要具备产品思维，一个好的应用，就就必须注重细节。另外还需要注意的是，Android 手机发展多年，最新版本也发展到 Android 9.0（2019年），我们做出来的应用不仅要注意适配市面上常见的手机设备，还要注意适配Android的版本，这里是非常难的。

## 系统设计

本系统的文字识别引擎与语音合成引擎皆采用百度公司的识别引擎，可以应用于多门语言，比如Java语言，PHP语言，Python语言，C#语言，C++语言，更包括可以应用于Android 与 iOS 平台上。

### 百度文字识别支持系统与硬件

支持的系统和硬件版本

系统：支持 Android 4.0（API Level 15）到Android7.0（API Level 25）系统。需要开发者通过minSdkVersion来保证支持系统的检测。

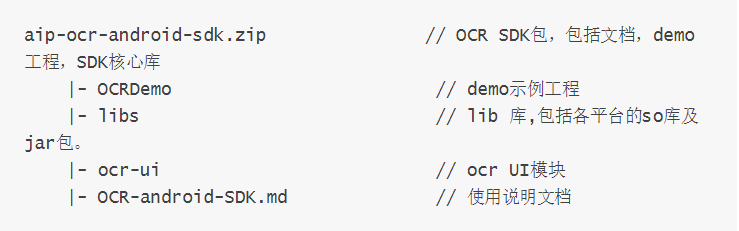
CPU架构：armeabi，arm64-v8a，armeabi-v7a，x86

机型：手机和平板皆可

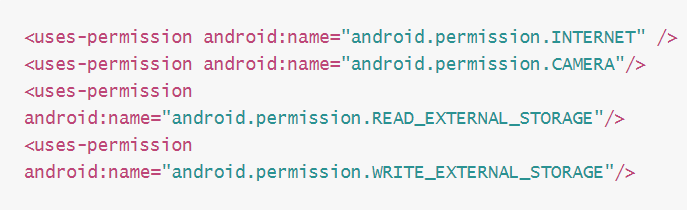
硬件要求：要求设备上有相机模块。

网络：支持WIFI及移动网络，移动网络支持使用NET网关及WAP网关（CMWAP、CTWAP、UNIWAP、3GWAP）。

* 1. 开发包说明



### 程序开始前需要添加的权限



各个权限的用途说明见下表：



### 系统界面设计

Android 界面布局设计，通常两种，一是静态布局文件设计，二是Java代码动态设置。

本应用两者皆有采用，主体不需要变动内容，采用静态布局文件设计，而需要根据不同结果采用不同的内容则采用Java代码动态设置。

### 首页设计

首页作为主要设计页面，可以让用户通过点击Android控件完成数据采集功能。其效果图如图：



图 2

用户打开本应用，在主页面上选择想要实现的功能，系统会调用相应的引擎进行处理，最后返回结果，可以通过语音合成播报识别结果。

文字识别时序图：



系统功能时序图 1

详细流程如下：

1. 启动本应用，系统将进行初始化操作
2. 用户通过点击屏幕相应的功能，开启识别功能。
3. 系统启动手机的摄像头，用户拍摄图片，系统系统已拍摄图片。用户点击确定，调用文字识别引擎，处理图片。
4. 识别完成后将数据返回给首页，调用对话框弹出返回的识别结果，
5. 系统调用语音合成，进行语音播报
   1. 系统相机获取图片

调用系统的相机模块共两种使用方式，用户可以拍摄图片或者选取手机中存储的图片，下图为调用系统相机的流程图：



图 3 调用系统相机获取图片流程图

* 1. Android XML布局部分代码展示：

<**Button**

android:id="@+id/general\_basic\_button"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="通用文字识别"

tools:ignore="HardcodedText" />

<**Button**

android:id="@+id/idcard\_button"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="身份证识别"

tools:ignore="HardcodedText" />

<**Button**

android:id="@+id/bankcard\_button"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="银行卡识别"

tools:ignore="HardcodedText" />

<**Button**

android:id="@+id/license\_plate\_button"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="车牌识别"

tools:ignore="HardcodedText" />

<**LinearLayout**

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical">

<**Button**

android:id="@+id/car\_model\_button"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="车型识别"

tools:ignore="HardcodedText" />

<**Button**

android:id="@+id/speechrecognition\_button\_start"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="语音识别开始"

tools:ignore="HardcodedText" />

<**Button**

android:id="@+id/speechrecognition\_button\_stop"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="语音识别停止"

tools:ignore="HardcodedText" />

<**RelativeLayout**

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

<**ImageView**

android:id="@+id/car\_model\_iv"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:scaleType="centerCrop" />

<**TextView**

android:id="@+id/scan\_value\_tv"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_below="@id/car\_model\_iv"

android:background="@color/colorAccent"

android:gravity="center"

android:text="结果扫描中"

android:textColor="@color/colorWhite"

android:textSize="20sp"

android:textStyle="bold"

android:visibility="gone" /

</RelativeLayout>

<**TextView**

android:id="@+id/re\_result\_tv"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content" />

</LinearLayout>

* 1. 关键代码部分展示：

*// 通用文字识别，跳转调用手机摄像头*

findViewById(R.id.general\_basic\_button).setOnClickListener(new View.OnClickListener() {

@**Override**

**public** **void** onClick(**View** v) {

if (!checkTokenStatus()) {

return;

}

**Intent** intent = new Intent(MainActivity.this, CameraActivity.class);

intent.putExtra(CameraActivity.KEY\_OUTPUT\_FILE\_PATH,

FileUtil.getSaveFile(getApplication()).getAbsolutePath());

intent.putExtra(CameraActivity.KEY\_CONTENT\_TYPE,

CameraActivity.CONTENT\_TYPE\_GENERAL);

startActivityForResult(intent, REQUEST\_CODE\_GENERAL\_BASIC);

}

});

*// 识别成功回调，通用文字识别*

if (requestCode == REQUEST\_CODE\_GENERAL\_BASIC && resultCode == Activity.RESULT\_OK) {

RecognizeService.recGeneralBasic(this, FileUtil.getSaveFile(getApplicationContext()).getAbsolutePath(),

new RecognizeService.ServiceListener() {

@**Override**

**public** **void** onResult(**String** result) {

Log.i(TAG, "onResult: == " + getUniverstalTextJsonBean(result));

infoPopText("识别结果,查看以下内容", getUniverstalTextJsonBean(result));

speak(getUniverstalTextJsonBean(result));

}

});

}

*//弹出对话框*

**private** **void** alertText(**final** **String** title, **final** **String** message) {

this.runOnUiThread(new Runnable() {

@**Override**

**public** **void** run() {

alertDialog.setTitle(title)

.setMessage(message)

.setPositiveButton("确定", new DialogInterface.OnClickListener() {

@**Override**

**public** **void** onClick(**DialogInterface** dialog, **int** which) {

mSpeechSynthesizer.stop();

}

}) .show();

}

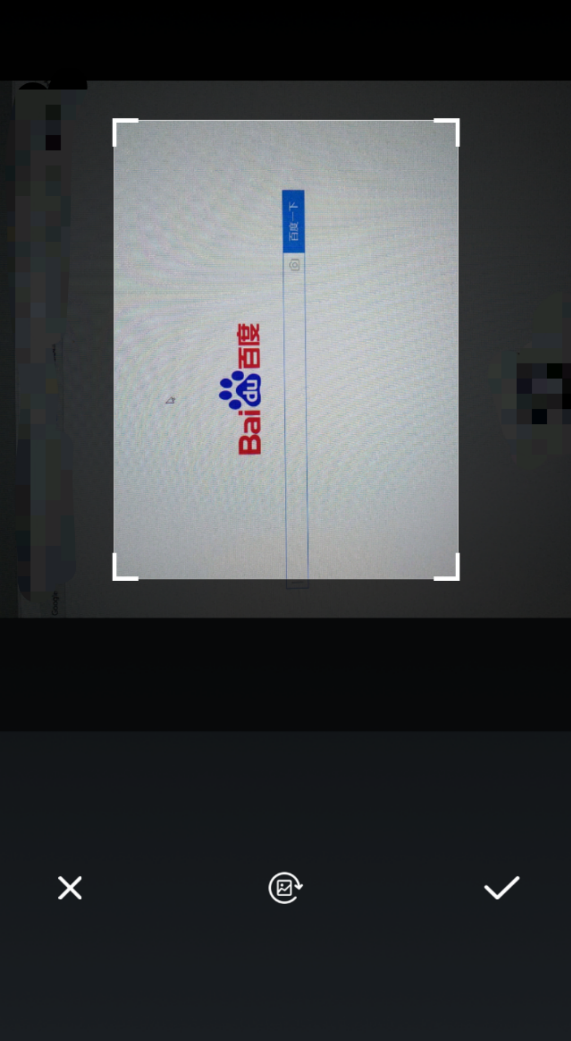
});

}

应用运行截图



图 4



* 1. 语音合成与语音识别核心技術

语音合成和[语音识别](https://baike.baidu.com/item/%E8%AF%AD%E9%9F%B3%E8%AF%86%E5%88%AB)技术是实现人机语音通信，建立一个有听和讲能力的口语系统所必需的两项关键技术。使电脑具有类似于人一样的说话能力，是当今时代信息产业的重要竞争市场。和语音识别相比，语音合成的技术相对说来要成熟一些，并已开始向产业化方向成功迈进，大规模应用指日可待。

语音合成，又称文语转换（Text to Speech）技术，能将任意文字信息实时转化为标准流畅的语音朗读出来，相当于给机器装上了人工嘴巴。它涉及声学、语言学、[数字信号处理](https://baike.baidu.com/item/%E6%95%B0%E5%AD%97%E4%BF%A1%E5%8F%B7%E5%A4%84%E7%90%86)、计算机科学等多个学科技术，是[中文信息处理](https://baike.baidu.com/item/%E4%B8%AD%E6%96%87%E4%BF%A1%E6%81%AF%E5%A4%84%E7%90%86)领域的一项前沿技术，解决的主要问题就是如何将文字信息转化为可听的声音信息，也即让机器像人一样开口说话。我们所说的“让机器像人一样开口说话”与传统的声音回放设备（系统）有着本质的区别。传统的声音回放设备（系统），如磁带[录音机](https://baike.baidu.com/item/%E5%BD%95%E9%9F%B3%E6%9C%BA)，是通过预先录制声音然后回放来实现“让机器说话”的。这种方式无论是在内容、存储、传输或者方便性、及时性等方面都存在很大的限制。而通过计算机语音合成则可以在任何时候将任意文本转换成具有高自然度的语音，从而真正实现让机器“像人一样开口说话”。

文语转换系统实际上可以看作是一个人工智能系统。为了合成出高质量的语言，除了依赖于各种规则，包括语义学规则、词汇规则、语音学规则外，还必须对文字的内容有很好的理解，这也涉及到自然语言理解的问题。下图显示了一个完整的文语转换系统示意图。

文语转换过程是先将文字序列转换成音韵序列，再由系统根据音韵序列生成语音波形。其中第一步涉及语言学处理，例如分词、字音转换等，以及一整套有效的韵律控制规则；第二步需要先进的语音合成技术，能按要求实时合成出高质量的语音流。因此一般说来，文语转换系统都需要一套复杂的文字序列到音素序列的转换程序，也就是说，文语转换系统不仅要应用数字信号处理技术，而且必须有大量的语言学知识的支持。[4]

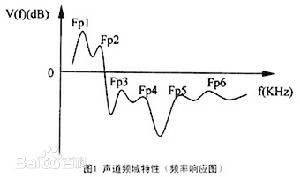


图 5

参考文献

1. 吕俊哲，图像二值化算法研究及其实现[J]，科技情报开发与经济，2004，（3）：125-129
2. 苏金明、王永利，MATALAB图形图像[M]，河北：华北电力大学，2008
3. 陈友仁、赵正校，基于隐马尔可夫模型的车牌自动识别技术，上海，2001
4. 张斌,全昌勤,任福继. 语音合成方法和发展综述[J]. 小型微型计算机系统,2016,(01):186-192.
5. 王志超、刘惠义，一种基于隐马尔可夫模型的人脸识别方法[A]，计算机应用与软件，2013 年，第 30 卷第 2 期