目录

[1. 简介 3](#_Toc393023256)

[2. 平台设计与实现 4](#_Toc393023257)

[2.1 平台模型 4](#_Toc393023258)

[2.1.1 模型概述 4](#_Toc393023259)

[2.1.2 Server 4](#_Toc393023260)

[2.1.3 Master 5](#_Toc393023261)

[2.1.4 Client 5](#_Toc393023262)

[2.2 架构实现 6](#_Toc393023263)

[2.2.1 Server类 6](#_Toc393023264)

[2.2.2 Master类 7](#_Toc393023265)

[2.2.3 Client类 8](#_Toc393023266)

[2.3 模块划分 9](#_Toc393023267)

[2.4 开发环境 10](#_Toc393023268)

[2.5 依赖库 10](#_Toc393023269)

[3 查询图书馆模块 11](#_Toc393023270)

[3.1数据抓包 11](#_Toc393023271)

[3.1.1 查询图书 11](#_Toc393023272)

[3.1.2 查询个人信息 13](#_Toc393023273)

[3.2 编程实现 14](#_Toc393023274)

[3.2.1 python 14](#_Toc393023275)

[3.2.2 C++ 15](#_Toc393023276)

[3.3 查询结果 16](#_Toc393023277)

[3.3.1 个人信息 16](#_Toc393023278)

[3.3.1 借阅的书籍 16](#_Toc393023279)

[3.3.4 借阅过的所有书籍 17](#_Toc393023280)

[3.3.5 财务情况 18](#_Toc393023281)

[3.3.6 查询书籍 18](#_Toc393023282)

[4 语音模块 21](#_Toc393023283)

[4.1语音模块简介 21](#_Toc393023284)

[4.2 SpeechToText类 22](#_Toc393023285)

[4.2.1 record( )录制音频wav文件 23](#_Toc393023286)

[4.2.2 run\_iat( )音频文件转换为文本 24](#_Toc393023287)

[4.3 TextToSpeech类 25](#_Toc393023288)

[4.3.1 void play\_wav() 25](#_Toc393023289)

[4.3.2 text\_to\_speech( ) 26](#_Toc393023290)

[4.4 音频文件格式 27](#_Toc393023291)

[4.5 测试 27](#_Toc393023292)

[4.5.1 SpeechToText功能测试 27](#_Toc393023293)

[4.5.2 TextToSpeech功能测试 28](#_Toc393023294)

[5 ServerCamera模块 29](#_Toc393023295)

[5.1 Opencv简介 29](#_Toc393023296)

[5.2 ServerCamera模块功能划分 30](#_Toc393023297)

[5.2.1 GetCamera调用摄像头 31](#_Toc393023298)

[5.2.2 GetPicture拍照 31](#_Toc393023299)

[5.2.3 MoveDetect移动中的人脸识别 32](#_Toc393023300)

[5.2.4 CutImage对阅读证数字切割 33](#_Toc393023301)

[5.3 测试 35](#_Toc393023302)

# 1. 简介

小Q机器人（Qrobot）是智能互联网机器人产品，是中国科学院与腾讯公司重大战略合作项目。Qrobot由腾讯公司、中科院深圳先进技术研究院和深圳市中科睿成智能科技有限公司联合研发。小Q机器人能够通过语音指令等多维交互方式为用户提供新闻、天气、音乐、股票、教育、娱乐、办公等资讯和服务应用。同时，Qrobot有丰富的应用平台，通过互联网即可下载更多的功能。

本项目是以小Q机器人为应用实例，设计开发的一个机器人软件平台。本软件平台使用Master-Server-Client多层次结构，包含图书查询、语音、图像三个模块。图书查询模块包含查询个人信息、借阅图书、借阅过的所有图书、财务情况、查询书籍等功能。语音模板包含声音转文本，文本合成为声音等功能。图像模块包含调用摄像头、拍照、移动中的人脸识别、对阅读证数字切割等功能。用户可以通过继承Server类的方式增加其它功能模块。也可以在查询图书、语音、图像模块中增加其它功能函数。

# 2. 平台设计与实现

## 2.1 平台模型

### 2.1.1 模型概述

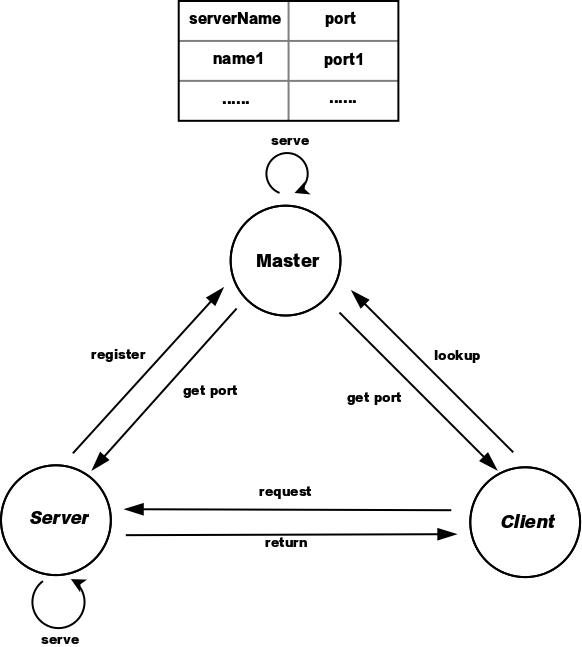
本软件平台使用Master-Server-Client多层次结构。

Master作为中心节点，拥有一个始终固定的通信标识符。Master负责记录和管理所有Server的通信标识符，需要在软件平台使用过程中始终保持在运行状态，

Server提供各类服务，可以在需要时开启，并在Master处进行注册，获得一个唯一的通信标识符。Server开启后被动地等待来自于Client的服务请求，进行处理并返回结果。

Client用于实现具体应用的业务逻辑，主动向Server请求服务。当需要某种服务时，先从Master处查询到对应Server的通信标识符，之后向Server请求服务。

架构模型示意图如下



### 2.1.2 Server

Server用于提供特定的服务，并根据服务的数目和所需的参数，定义通信消息的结构。

每种Server拥有一个唯一且固定的名称，并利用该名称，在Master处进行注册。Master将Server名称与为其分配的通信标识符进行绑定，并记录在表中，以备Client查询。

Server获得通信标识符后，即可开启服务，等待来自于Client的服务请求。

为了使Client能够正确地请求服务，Server需要定义一个自己特定的消息结构，指明为了向自己请求服务，需要填写的申请内容。

当Server收到请求服务时，读取请求服务消息，并根据消息的具体内容执行预定义的操作，进而返回特定的结果给Client。之后，Server继续等待下一个服务请求的到来。

### 2.1.3 Master

Master是软件平台的中心节点，记录和维护Server的注册信息。

事实上，Master可以看做一种特殊的Server，为除自己之外的其他所有Server提供注册服务，并为Client提供查询指定名称Server的服务。

Master拥有一个输入消息类型定义和一个输出消息类型定义。当请求服务者为Server时，该Server需要填写注册信息，包括表明此次请求是来自于Server的标志，以及自己的Server名称。当Master接收到来自于Server的注册服务请求时，会将该Server的名称与一个唯一的通信标识符绑定，进行保存，并返回该通信标识符。此时，输出消息包含的是为该Server分配的通信标识符。

如果向Master请求服务的是Client，则需要在输入消息中写明此次请求来自于Client，并指出之后需要的服务的Server名称。Master接收到该请求后，在注册信息表中查找该Client所需要的Server的通信标识符，并返回给Client。此时，输出消息包含的是Client查询的Server的通信标识符。

### 2.1.4 Client

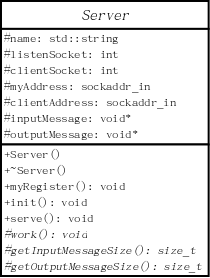
Client主要用于实现具体应用程序的业务逻辑，在需要时向Server请求服务。

当Client需要某个Server提供的服务时，首先利用该Server的名称，向Master请求查询服务，获得Server的通信标识符。之后，Client可以直接与Server进行通信，而不再需要Master的干预。Client遵守Server所定义的消息规范，向Server发送消息请求服务，并接收返回结果。

## 2.2 架构实现

本软件平台采用多进程运行模式，利用socket进程间通信机制进行消息传递。

### 2.2.1 Server类



Server类是所有服务提供类的抽象基类，一个服务器进程只需要创建一个Server派生类的对象，并执行Server::serve()操作开启服务即可。

Server派生类的对象被创建时，利用Server::myRegister()在Master类处进行注册。Master类为该Server派生类对象分配一个唯一的socket描述符，并将其作为通信标识符，与Server派生类对象的名称进行绑定，记录下来。

之后，该对象可以调用Server::init()进行初始化，然后执行Server::serve()操作，利用socket进行进程间通信，等待来自于Client的服务请求，进行相应的处理并返回一定的结果。

为了提供具体的某种服务，开发者需要继承Server类，在构造函数中为Server派生类命名并调用注册函数Server::myRegister()，并实现其中的纯虚函数getInputMessageSize()，getOutputMessageSize()和work()。

为了向Client说明如何向Server派生类对象请求服务，开发者需要为Server派生类定义输入消息结构体和输出消息结构体，用以标明如何向Server派生类对象请求某个服务，该服务所需要的参数，以及服务返回消息的数据结构。

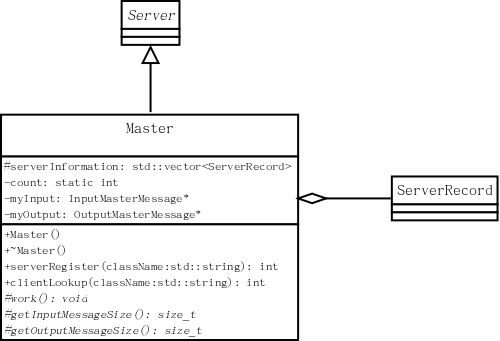
开发者需要实现Server类中的纯虚函数getInputMessageSize()和getOutputMessageSize()，返回输入消息结构体的大小和输出消息结构体的大小，为软件平台内置的通信功能提供必要的支持。

此外，开发者需要实现Server类中定义的纯虚函数work()，在其中编写具体的服务代码。在Server::work()函数中，需要从输入消息结构体对象所在的inputMessage缓冲区中读取服务请求消息，针对不同的消息内容，执行不同的任务，并将结果写入输出消息结构体对象所在的缓冲区outputMessage中。

Server::serve()函数执行后，等待Client请求服务。当收到服务请求时，执行开发者实现的Server::work()操作，并通过socket将结果返回给Client。之后，继续执行Server::serve()操作，等待下一个服务请求，重复上述操作。

### 2.2.2 Master类



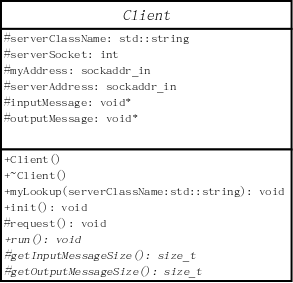


Master类作为一种特殊的Server类，继承自Server类，并实现了纯虚函数getInputMessageSize()，getOutputMessageSize()和work()。

Master类对象运行在一个单独的进程，在使用软件平台的整个过程中始终保持运行，并拥有一个众所周知且唯一的socket描述符作为通信标识符。

Master类实际上就是一个提供具体服务的Server派生类，工作机制与Server基本相同。serve()函数执行之后，等待来自于Server派生类对象的注册服务请求，或者是来自于Client派生类对象的查询服务请求。work()从inputMessage缓冲区中读取请求消息。根据InputMasterMessage::type判断此次请求的发起者，并提供相应的服务，再将结果写入outputMessage缓冲区。之后，由serve()中的操作通过socket将结果返回给请求者。

### 2.2.3 Client类



Client类是所有实现具体应用程序业务逻辑代码的类的抽象基类。

每个Client派生类对象运行在一个独立的进程中，利用需要服务的Server名称，通过周知的Master类对象的socket描述符，向Master查询Server的socket描述符。之后，Client进程可以直接通过socket与Server进程进行消息传递，请求服务并获得结果，而不再需要Master的干预。

开发者需要继承Client类，并实现getInputMessageSize()，getOutputMessageSize()以及run()三个纯虚函数。

向Server派生类对象请求服务时，必须遵守该Server派生类所定义的通信消息规范。getInputMessageSize()和getOutputMessageSize()分别返回提供该Server派生类所定义的输出消息结构体大小和输出消息结构体大小。

run()函数中，可以实现业务逻辑。需要某个Server服务时，调用Client::myLookup()向Master查询该Server派生类对象的socket描述符，之后执行run()操作。在run()函数中，根据Server派生类所定义的消息结构体，向outputMessage缓冲区写入消息内容。

之后，调用Client::request()函数发送消息，并接收返回结果。返回结果会保存在inputMessage缓冲区中。

可以看出，当请求服务的Server派生类确定之后，该Server派生类的输入消息，实际上就是Client派生类此时的输出消息。该Server派生类的输出消息则为Client派生类此时的输入消息。

当Client需要不同的服务时，随着Client::myLookup()请求的Server名称不同，应该对inputMessage和outputMessage的类型做相应的调整。

## 2.3 模块划分

在软件平台的基础上，我们开发了多种Server，提供了语音输入、语音输出、华南理工大学图书馆查询、摄像头、Qrobot机器人等服务，并利用一个包含了业务逻辑的Client，利用各个Server提供的服务，组成了一个完整的应用程序。

**语音输入**

语音输入模块的主要功能由SpeechToText类提供。SpeechToText类利用音频支持库ALSA，从音频输入设备获取语音，并对其进行识别，从而生成对应的文本。

**语音输出**

语音输出模块的主要功能由TextToSpeech类提供。TextToSpeech类同样利用音频支持库ALSA，接收字符串作为输入，利用音频输出设备进行播放。

**图书馆查询**

图书馆查询模块的主要功能由Serverlibrary类提供。Serverlibrary类提供了查询用户个人信息、查询图书等功能。

**摄像头**

摄像头模块的主要功能由ServerCamera类提供。ServerCamera通过OpenCV库，

**Qrobot机器人**

Qrobot机器人模块的主要功能由ServerQrobot类提供。ServerQrobot类调用QrobotController类的操作，实现对机器人的间接控制。

**主客户端**

MainClient类提供了主客户端模块的功能。MainClient通过调用语音输入、语音输出、图书馆查询、摄像头、Qrobot机器人等模块提供的服务，为用户提供了一个完整的应用。

## 2.4 开发环境

编程语言： C++

Python

操作系统： Ubuntu 14.04 LTS

Ubuntu 13.10

编辑器： gVim

gedit

编译器： g++

版本管理工具： Git

版本库地址： https://github.com/Isilendil/socketPlatform

## 2.5 依赖库

图像处理支持库 OpenCV

音频支持库 ALSA

Qrobot支持库

Python HTML/XML处理模块 BeautifulSoup

python c++ 支持库

# 3 查询图书馆模块

使用软件平台开发应用的硬件平台式机器人小Q。我们做的应用的主要功能是可以通过小Q查询华南理工大学图书馆的图书和本校学生的图书证相关信息。使用小Q的麦克风输入要查询的图书，或者学生的图书证号和密码以查询私人信息。最后通过小Q的喇叭返回结果。

查询的信息源是华南理工大学图书馆（网址：http://www.lib.scut.edu.cn/index.jsp），界面如。



图 3‑1

## 3.1数据抓包

为了获得华南理工大学图书馆网站的数据，就要观察使用图书馆网站的时候，网站向哪个网址发送怎样的数据，就要对数据进行抓吧。使用IE11的开发人员工具中的网络功能模块监控华南理工大学图书馆网站的数据流向和发送数据的结构。

### 3.1.1 查询图书

在查询图书的部分，通过对查询图书的页面（网址：http://202.38.232.10/opac/servlet/opac.go）的数据监控（页面如），发现在输入框输入要查询的信息后，点击搜索，网站就会向<http://202.38.232.10/opac/servlet/opac.go>发送信息。发送的信息如下：

请求表头：

表格 3‑1

|  |  |
| --- | --- |
| 键 | 值 |
| 请求 | POST /opac/servlet/opac.go HTTP/1.1 |
| Accept | text/html, application/xhtml+xml, \*/\* |
| Referer | http://202.38.232.10/opac/servlet/opac.go |
| Accept-Language | zh-CN |
| User-Agent | Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko |
| Content-Type | application/x-www-form-urlencoded |
| Accept-Encoding | gzip, deflate |
| Host | 202.38.232.10 |
| Content-Length | 133 |
| DNT | 1 |
| Connection | Keep-Alive |
| Cache-Control | no-cache |
| Cookie | JSESSIONID=CC759B11D59F4B7677A3BB450EC46513 |

请求正文：

表格 3‑2

|  |  |
| --- | --- |
| cmdACT | simple.list |
| RDID | ANONYMOUS |
| ORGLIB | SCUT |
| FIELD1 | TITLE |
| VAL1 | &{搜索内容} |
| MODE | FRONT |



图 3‑2

那么，参考请求表头和请求正文的格式和内容，用代码模拟网站向网站发送信息，得到结果网页的源代码，通过分析网页源代码获取结果信息。

### 3.1.2 查询个人信息

学生的图书馆个人信息包括个人资料、书架、借阅、财务信息。查询个人信息的做法与查询图书的做法相似，也是通过数据抓包，模拟发送数据。不同的地方是，查询个人信息还要进行登录操作。通过对登录界面（网址：http://202.38.232.10/opac/servlet/opac.go?cmdACT=mylibrary.index）的数据流向监控发现（网站页面如）在用户填写完id和密码，点击登录后，网页会向” /opac/servlet/opac.go”发送数据，发送的数据如下：

请求表头：

表格 3‑3

|  |  |
| --- | --- |
| 键 | 值 |
| 请求 | POST /opac/servlet/opac.go HTTP/1.1 |
| Accept | text/html, application/xhtml+xml, \*/\* |
| Referer | http://202.38.232.10/opac/servlet/opac.go?cmdACT=mylibrary.index |
| Accept-Language | zh-CN |
| User-Agent | Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko |
| Content-Type | application/x-www-form-urlencoded |
| Accept-Encoding | gzip, deflate |
| Host | 202.38.232.10 |
| Content-Length | 107 |
| DNT | 1 |
| Connection | Keep-Alive |
| Cache-Control | no-cache |
| Cookie | JSESSIONID=CC759B11D59F4B7677A3BB450EC46513 |

请求正文：

表格 3‑4

|  |  |
| --- | --- |
| cmdACT | mylibrary.login |
| userid | &{用户id} |
| passwd | &{密码} |



图 3‑3

用代码模拟发送数据后，会返回cookie，把返回的cookie保存下来，就实现了模拟登陆，就可以用这个cookie获取用户的个人信息。

## 3.2 编程实现

考虑到python拥有urllib和urllib2这种成熟的网络库，还有BeautifulSoup这种网页源代码解析的第三方库，还考虑到python的易用性和在调试上的方便，遂用python语言实现获取数据的功能。

### 3.2.1 python

下表的代码是向网址发送数据，得到cookie，并利用cookie请求得到目标网页源代码的核心代码。主要使用了python的urllib和urllib2模块。使用urllib2模块模拟登陆，向登陆地址发消息，保存成功返回的cookie，那么在访问私人信息的时候，就可以发送这个cookie和请求信息（信息内容通过抓包获得）以获取结果的网页源代码。

表格 3‑5

|  |  |
| --- | --- |
| 1 | cookie= cookielib.CookieJar() |
| 2 | opener= urllib2.build\_opener(urllib2.HTTPCookieProcessor(cookie)) |
| 3 | postdata={“cmdACT”:”mylibrary.login”,”method”:”mylib”,”userid”:”B1321033460”,”passwd”:”5355365”,”user\_login”:”%E7%99%BB%E5%BD%95”} |
| 4 | postdata= urllib.urlencode(postdata) |
| 5 | headers={‘Accept’:’text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8’,’Content-Type’:’application/x-www-form-urlencoded’,’User-Agent’: ‘Mozilla/5.0(Windows NT6.1; WOW64; rv:30.0) Gecko/20100101 Firefox/30.0’, ‘Referer’: ‘http://202.38.232.10/opac/servlet/opac.go?cmdACT=mylibrary.index’} |
| 6 | req= urllib2.Request(homepage, postdata, headers) |
| 7 | result= opener.open(req) |
| 8 | html= result.read() |

下表的代码是使用python的第三方库BeautifulSoup来分析网页源代码的核心代码，把网页源代码传给BeautifulSoup做字符处理，通过对标签进行搜索和匹配，寻找结果。

表格 3‑6

|  |
| --- |
| soup= bs(html) |
| a= soup.findAll('th') |

### 

### 3.2.2 C++

在用python实现了登陆华南理工大学图书馆查询图书信息和学生个人信息的功能后，要把python脚本集成到C++代码里面，封装成库供开发使用。Python官方提供了C++调用python脚本的库（头文件：Python.h；共享库：libpython2.7.so）。在linux平台下，通过在命令行输入sudo apt-get install python-dev就可以安装好所需的库文件，当然也可以通过下载python源码手动编译。

使用Python.h和libpython2.7.so提供的函数就可以找到相应的脚本文件，并把脚本里面的函数名导进来，通过python.h和libpython2.7.so，传递参数给所学要调用的函数，调用脚本的函数。实现用C++调用脚本。

虽然用C++调用python脚本的代码并不难，但是涉及到大量的数据的返回和转换。譬如python脚本中大量的list和字典字符串数据返回给C++，C++要用stl（C++标准库）容器保存下来。所以这里用了一个叫ffpython的库。ffpython不仅封装好了C++调用python的代码，只提供一个接口，而且ffpython支持stl容器。也就是说python中的字符串返回给C++可以用string保存，而python中的list和字典数据类型返回给C++可以用stl容器中的list和map保存。

使用C++封装好python脚本，封装成library类，使用使用用户的id和密码初始化library类，library自动查询用户信息，保存到类的实例中的私有成员变量。并且，可以直接使用library类内的全局静态函数searchbookbytitle（）、searchbookbyauthor（）、searchbookbyisbn（）来查询书籍，这里提供了通过书名、作者和ISBN的三种方式查询书籍。

最后使用已经写好的软件平台实现应用。主要是实现查询图书馆信息的服务功能。根据需求，定义服务器端输入/客户端输出的格式为：

|  |
| --- |
| struct InputServerlibraryMessage {       char userid[15];       char passwd[10];       enum searchtype commandName;       char book[25]; }; |

服务端接受/客户端发送用户id、密码、任务种类、要查询的书信息

服务器端发送/客户端接受的格式为：

|  |
| --- |
| struct OutputServerlibraryMessage {       int status;       char path[100]; }; |

服务器端发送/客户端接受status，如果通过用户id和密码正确地获取信息，那么status为1，否则为0。Path保存的是数据保存在的本地路径。因为查询出来的数据格式和数量都不同，因此直接用自己写的软件平台传输这些数据比较困难，因此把结果写为本地文本，并且把路径返回，也就是path。

这样，通过使用python模拟登陆、获取网页源代码和分析源代码获取结果，使用C++调用python脚本，并封装成类。查询图书馆信息的服务继承Server类，重写Server类的的work函数，work函数根据客户端发送的指令和信息调用library类相应功能，并返回结果。这样，查询图书馆信息的模块就完成了。

## 

## 3.3 查询结果

查询到的信息会通过api返回，并且保存在本地硬盘为文本文件，可以通过打开文件查询。

### 3.3.1 个人信息

表格 3‑7

|  |
| --- |
| 信用分数 100 |
| 证件号码 20132103\*\*\*\* |
| 开户时间 2013-09-27 |
| IC卡号 B132103\*\*\*\* |
| 借书证号 B132103\*\*\*\* |
| 到期时间 2016-06-30 |
| 证件类型 |
| 身份证 |
| 状态 有效 |
| 现借书数量 |
| 欠费情况 0 |
| 读者类型 研究生 |
| 姓名 \*\*\* |

### 

### 3.3.1 借阅的书籍

|  |  |  |
| --- | --- | --- |
| 借阅时间 | 到期时间 | 书籍名 |
| 2014-05-28 | 2014-07-28 | 网络安全基础/William Stallings著 |
| 2014-05-28 | 2014-07-28 | Java Web开发实战/软件开发技术联盟编著 |
| 2014-06-13 | 2014-09-12 | JavaScript+jQuery从入门到精通/高金勇, 陈晓建等编著 |
| 2014-06-13 | 2014-09-12 | 密码编码学与网络安全/(美) William Stallings著 |
| 2014-05-18 | 2014-07-17 | 你必须知道的261个Java语言问题/梁建全编著 |
| 2014-04-30 | 2014-09-12 | 法语·1/马晓宏, 柳利编著 |
| 2014-04-30 | 2014-09-12 | 法语现代语法/毛意忠编著 |
| 2014-04-30 | 2014-09-12 | 法语语音渐进·初级/编著周林飞, (法) 吕西尔·夏尔利雅克 ...[等] |

### 

### 3.3.4 借阅过的所有书籍

|  |
| --- |
| UML面向对象建模与设计 |
| 离散数学导学 |
| 数据库原理及应用 |
| Java快速入门与商用项目培训 |
| 法语·1 |
| Visual C++开发实战 |
| 你必须知道的261个Java语言问题 |
| UNIX从入门到精通 |
| A first course in database systems =·数据库系统基础教程·HZ books·数据库系统基础教程 / |
| Programming pearls =·编程珠玑·图灵程序设计丛书·编程珠玑 / |
| 网络安全基础 |
| JavaScript+jQuery从入门到精通 |
| 法语现代语法 |
| 英语佳句荟萃 |
| 密码编码学与网络安全 |
| 数据结构与算法 |
| 大规模并行处理器编程实战 |
| 离散数学及其应用 |
| 法语语音渐进·初级 |
| 并行计算导论 |
| 世界是数字的·= D is for digital |
| 设计原本 |
| MySQL 5.6从零开始学 |
| 设计模式 |
| 法语语音渐进附练习四百题答案·初级 |
| 托业考试学习大全本领书 |
| 并发程序设计基础教程 |
| Java Web开发实战 |

### 

### 3.3.5 财务情况

|  |  |  |
| --- | --- | --- |
| 服务费 | 0 | 华南理工大学图书馆 2014-05-28 16:29:07.0 |
| 工本费 | 0 | 华南理工大学图书馆 2014-05-28 16:29:07.0 |
| 超期违约金 | 0.1 | 华南理工大学图书馆 2014-05-28 16:29:07.0 |
| 超期违约金 | 0.1 | 华南理工大学图书馆 2014-05-28 16:29:07.0 |
| 押金 | 0 | 华南理工大学图书馆 2014-05-28 16:29:07.0 |

### 

### 3.3.6 查询书籍

这个模块提供三种方式查书，包括数名，作者名和ISBN。以搜索书名为例，搜索“人工智能”，结果如下表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 王永庆著 | 人工智能 | 7-5605-0646-1 | 西安交通大学出版社 | TP18 8 | 1994 | 中文图书 |
| 李卫华等编译 | IBM-PC机编译型PROLOG语言 |  | 武汉大学出版社 | TP312PR 6 | 1987.3 | 中文图书 |
| 杨祥金,蔡庆生编著 | 人工智能 |  | 科学技术文献出版社重庆分社 | TP18 21 | 1988.1 | 中文图书 |
| 涂序彦编著 | 人工智能及其应用 | 7-5053-0264-7 | 电子工业出版社 | TP18 23 | 1988.10 | 中文图书 |
| (美)尼尔森(Nilsson,N.J.)著 | 人工智能原理 |  | 科学出版社 | TP18 12 | 1983.2 | 中文图书 |
| (苏)恰基柯著 | 人工智能的故事 |  | 科学普及出版社 | TP18 15 | 1986.12 | 中文图书 |
| 石纯一等编著 | 人工智能原理 |  | 清华大学出版社 | TP18 58 | 1993.10 | 中文图书 |
| 俞瑞钊,史济建编著 | 人工智能原理与技术 | 7-308-01258-1 | 浙江大学出版社 | TP18 59 | 1993.11 | 中文图书 |
| 黄国兴,徐国定编 | 人工智能原理及其实现 | 7-5439-0154-4 | 上海科学技术文献出版社 | TP18 57 | 1993.8 | 中文图书 |
| 田盛丰等编著 | 人工智能原理与应用 | 7-81013-735-2 | 北京理工大学出版社 | TP18 54 | 1993.8 | 中文图书 |
| (英)T.雷蒙德著·赵沁平译 | 人工智能中的逻辑 |  | 北京大学出版社 | TP18 34 | 1990.2 | 中文图书 |
| 施鹏飞,姚远编 | 人工智能教程 | 7-313-01241 | 上海交通大学出版社 | TP18 5 | 1993.12 | 中文图书 |
| 张玉志,魏宏森著 | 人工智能与社会进步 |  | 吉林教育出版社 | G301 3 | 1990.3 | 中文图书 |
| 林尧瑞等编著 | 专家系统原理与实践 | 7-302-00018-2 | 清华大学出版社 | TP11 27 | 1988.4 | 中文图书 |
| 王树林主编 | 人工智能辞典 | 7-115-04625-5 | 人民邮电出版社 | TP18-61 1 | 1992.2 | 中文图书 |
| 钟义信等著 | 智能理论与技术 | 7-115-04768-5 | 人民邮电出版社 | TP18 51 | 1992.12 | 中文图书 |
| (美)E.里奇著 | 人工智能 |  | 世界图书出版公司 | TP18 53 | 1993.1 | 中文图书 |
| 叶景楼主编 | 人工智能.专家系统.程序设计 | 7-5610-1825 | 辽宁大学出版社 | TP18 48 | 1992.8 | 中文图书 |
| 周旋芳,王林祥编译 | TURBO PROLOG人工智能程序设计 | 7-80038-436-5 | 兵器工业出版社 | TP18 47 | 1992.6 | 中文图书 |
| (美)班纳吉(Banerji,R.B.)著 | 人工智能理论方法 | 7-313-00856-2 | 上海交通大学出版社 | TP18 46 | 1991.5 | 中文图书 |

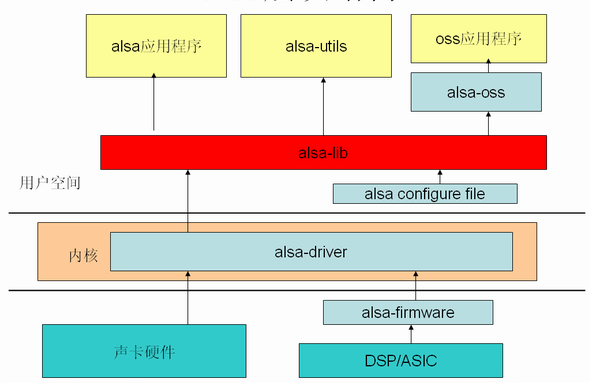
# 4 语音模块

## 4.1语音模块简介

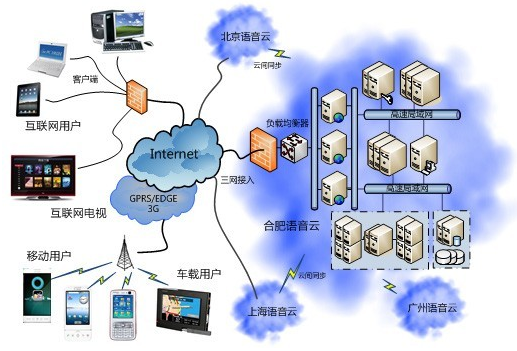
语音模块主要实现四个功能，（1）录制声音成wav音频文件，（2）把wav音频文件转换成文本，（3）把文本合成为语音wav文件，（4）播放音频wav文件。其中，音频转文本、文本合成为语音文件的功能通过调用iFly科大讯飞接口实现，音频的播放录制依赖ALSA提供的库实现，音频的编码、解码方式需要符合iFly科学讯飞语音平台接口的要求。

语音模块主要包含两个类：Speech\_To\_Text类、Text\_To\_Speech类，下面分别介绍。

ALSA结构如下图所示：



iFLY Mobile Speech Platform(MSP)产品的典型网络拓扑结构：



## 4.2 SpeechToText类

Speech类继承Server类，主要实现将声音转为文本输出的功能，分为两个方法实现：record( )，run\_iat( )。

|  |
| --- |
| Class SpeechToText : Server |
| 方法： |
| void work();  size\_t getInputMessageSize();  size\_t getOutputMessageSize();  void record();  void run\_iat(const char\* src\_wav\_filename , const char\* param, char \*result) |

在SpeechToTextMessage.h中定义了消息结构体，与Client通讯。SpeechToText实例化后向Master注册，Client即可直接调用SpeechToText提供的服务。如下图：



### 4.2.1 record( )录制音频wav文件

录制音频wav文件功能主要依赖ALSA音频处理库实现，ALSA的声音录制是采用数据IO接口(read/write)实现，设备的参数采用ioctl来控制。录制音频的编码方式需要符合iFly科大讯飞语音平台的接口要求。主要功能代码如下所示：

|  |
| --- |
| //SpeechToText.cpp |
| **if** ((fd **=** open(filename, O\_WRONLY **|** O\_CREAT, 0644)) **==** **-**1) { |
| fprintf(stderr, "Error open: [%s]/n", filename); |
| **return** ; |
| } |
|  |
| **if** (snd\_output\_stdio\_attach(**&**record.log, stderr, 0) **<** 0) { |
| fprintf(stderr, "Error snd\_output\_stdio\_attach/n"); |
| **goto** Err; |
| } |
|  |
| **if** (snd\_pcm\_open(**&**record.handle, devicename, SND\_PCM\_STREAM\_CAPTURE, 0) **<** 0) { |
| fprintf(stderr, "Error snd\_pcm\_open [ %s]/n", devicename); |
| **goto** Err; |
| } |
|  |
| **if** (SNDWAV\_PrepareWAVParams(**&**wav) **<** 0) { |
| fprintf(stderr, "Error SNDWAV\_PrepareWAVParams/n"); |
| **goto** Err; |
| } |
|  |
| **if** (SNDWAV\_SetParams(**&**record, **&**wav) **<** 0) { |
| fprintf(stderr, "Error set\_snd\_pcm\_params/n"); |
| **goto** Err; |
| } |
| snd\_pcm\_dump(record.handle, record.log); |
|  |
| SNDWAV\_Record(**&**record, **&**wav, fd); |
|  |
| snd\_pcm\_drain(record.handle); |
|  |
| close(fd); |

### 4.2.2 run\_iat( )音频文件转换为文本

音频wav文件转换为文本功能主要依赖iFly科大讯飞提供的语音识别接口实现。

iFly提供的主要功能函数有：

|  |  |
| --- | --- |
| QISRSessionBegin | 开始一个ISR会话 |
| QISRGrammarActivate | 传入语法 |
| QISRAudioWrite | 写入用来识别的语音 |
| QISRGetResult | 获取识别结果 |
| QISRSessionEnd | 结束一路会话 |
| QISRGetParam | 获取识别/语音听写业务相关参数 |

主要代码如下所示：

|  |
| --- |
| //SpeechToText.cpp |
| sessionID **=** QISRSessionBegin(NULL, param, **&**ret); |
| f\_pcm **=** fopen(src\_wav\_filename, "rb"); |
| **if** (NULL **!=** f\_pcm) { |
| fseek(f\_pcm, 0, SEEK\_END); |
| pcmSize **=** ftell(f\_pcm); |
| fseek(f\_pcm, 0, SEEK\_SET); |
| pPCM **=** (**char** **\***)malloc(pcmSize); |
| fread((**void** **\***)pPCM, pcmSize, 1, f\_pcm); |
| fclose(f\_pcm); |
| f\_pcm **=** NULL; |
| } |
| ret **=** QISRAudioWrite(sessionID, (**const** **void** **\***)**&**pPCM[pcmCount], len, audStat, **&**epStatus, **&**recStatus); |
| **const** **char** **\***rslt **=** QISRGetResult(sessionID, **&**recStatus, 0, **&**ret); |
| ret**=**QISRAudioWrite(sessionID, (**const** **void** **\***)NULL, 0, 4, **&**epStatus, **&**recStatus); |
| ret**=**QISRSessionEnd(sessionID, NULL); |

## 4.3 TextToSpeech类

TextToSpeech类主要实现文本转为声音功能，继承Server类，主要分为两个方法实现：void play\_wav( )，int text\_to\_speech(const char\* src\_text ,const char\* des\_path ,const char\* params)。

|  |
| --- |
| Class SpeechToText : Server |
| 方法： |
| void work();  size\_t getInputMessageSize();  size\_t getOutputMessageSize();  void play\_wav();  int text\_to\_speech(const char\* src\_text ,const char\* des\_path ,const char\* params) |

在TextToSpeechMessage.h中定义了消息结构体，与Client通讯。TextToSpeech实例化后向Master注册，Client即可直接调用TextToSpeech提供的服务。如图：



### 4.3.1 void play\_wav()

void play\_wav()实现播放音频wav文件功能，依赖ALSA库实现，ALSA的声音播放是采用数据IO接口(read/write)实现，设备的参数采用ioctl来控制，主要代码如下所示：

|  |
| --- |
| //TextToSpeech.cpp |
| fd = open(filename, O\_RDONLY); |
| if (WAV\_ReadHeader(fd, &wav) < 0) { |
| fprintf(stderr, "Error WAV\_Parse [%s]/n", filename); |
| goto Err; |
| } |
| if (snd\_output\_stdio\_attach(&playback.log, stderr, 0) < 0) { |
| fprintf(stderr, "Error snd\_output\_stdio\_attach/n"); |
| goto Err; |
| } |
| if (snd\_pcm\_open(&playback.handle, devicename, SND\_PCM\_STREAM\_PLAYBACK, 0) < 0) { |
| fprintf(stderr, "Error snd\_pcm\_open [ %s]/n", devicename); |
| goto Err; |
| } |
| if (SNDWAV\_SetParams(&playback, &wav) < 0) { |
| fprintf(stderr, "Error set\_snd\_pcm\_params/n"); |
| goto Err; |
| } |
| snd\_pcm\_dump(playback.handle, playback.log); |
| SNDWAV\_Play(&playback, &wav, fd); |
| snd\_pcm\_drain(playback.handle); |
| close(fd); |
| free(playback.data\_buf); |
| snd\_output\_close(playback.log); |
| snd\_pcm\_close(playback.handle); |

### 4.3.2 text\_to\_speech( )

text\_to\_speech( )主要实现将文本转换为音频的功能，依赖iFly科大讯飞语音合成功能接口实现。iFly提供的语音合成主要函数如下所示：

|  |  |
| --- | --- |
| QTTSSessionBegin | 开始一个TTS会话 |
| QTTSTextPut | 输入要合成的文本 |
| QTTSAudioGet | 获取合成语音 |
| QTTSAudioInfo | 获取音频相关信息 |
| QTTSSessionEnd | 结束一路会话 |
| QTTSGetParam | 获取合成业务相关参数 |

主要代码如下所示：

|  |
| --- |
| //TextToSpeech.txt |
| fp = fopen(des\_path,"wb"); |
| sess\_id = QTTSSessionBegin(params, &ret); |
| ret = QTTSTextPut(sess\_id, src\_text, text\_len, NULL ); |
| if ( ret != MSP\_SUCCESS ) |
| { |
| printf("QTTSTextPut: qtts put text failed Error code %d.\n",ret); |
| QTTSSessionEnd(sess\_id, "TextPutError"); |
| return ret; |
| } |
| fwrite(&pcmwavhdr, sizeof(pcmwavhdr) ,1, fp); |
| while (1) |
| { |
| const void \*data = QTTSAudioGet(sess\_id, &audio\_len, &synth\_status, &ret); |
| if (NULL != data) |
| { |
| fwrite(data, audio\_len, 1, fp); |
| pcmwavhdr.data\_size += audio\_len;//修正pcm数据的大小 |
| } |
| if (synth\_status == 2 || ret != 0) |
| break; |
| } |
| //修正pcm文件头数据的大小 |
| pcmwavhdr.size\_8 += pcmwavhdr.data\_size + 36; |
| //将修正过的数据写回文件头部 |
| fseek(fp, 4, 0); |
| fwrite(&pcmwavhdr.size\_8,sizeof(pcmwavhdr.size\_8), 1, fp); |
| fseek(fp, 40, 0); |
| fwrite(&pcmwavhdr.data\_size,sizeof(pcmwavhdr.data\_size), 1, fp); |
| fclose(fp); |
| ret = QTTSSessionEnd(sess\_id, NULL); |

## 4.4 音频文件格式

iFly科大讯飞语音平台对音频文件的编码方式有特定要求，且有固定的解码方式，不满足编码要求的音频无法实现转换，因此在录制、播放音频时都需要制定特定的音频格式，如下所示：

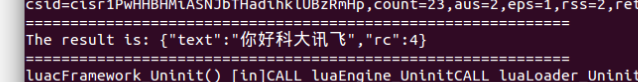
|  |
| --- |
| //SpeechToText.cpp |
| #define DEFAULT\_CHANNELS (1) |
| #define DEFAULT\_SAMPLE\_RATE (16000) |
| #define DEFAULT\_SAMPLE\_LENGTH (16) |
| #define DEFAULT\_DURATION\_TIME (5) |

其中，采样频率为16000，单声道，采样长度为16，采样时间为5.

## 4.5 测试

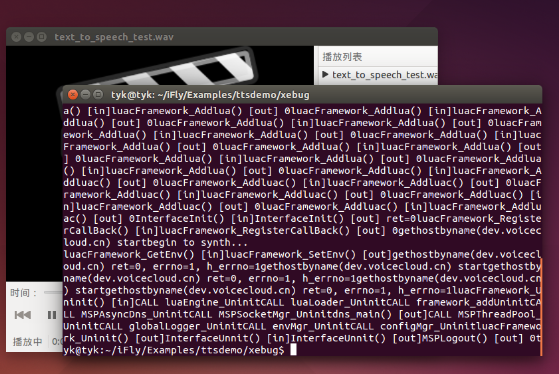
## 4.5.1 SpeechToText功能测试

对着小q说：“你好，科大讯飞”，获得文本如下。



## 4.5.2 TextToSpeech功能测试

如下图所示，文本成功转换为语音，并且播放。



# 5 ServerCamera模块

## 5.1 Opencv简介

OpenCV（Open Source Computer Vision Library）于 1999 年由 Intel 建立，现在由Willow Garage提供支持。 OpenCV 是一个基于 BSD 许可证授权（开源）发行的跨平台计算机视觉库，可以运行在 Linux、Windows、Mac OS和Android操作系统上。它轻量级而且高效——由一系列C函数和少量C++类构成，同时提供了 Python、Ruby、MATLAB 等语言的接口，实现了图像处理和计算机视觉方面的很多通用算法，不依赖于其它的外部库。

OpenCV 致力于真实世界的实时应用，通过优化的C代码的编写对其执行速度带来了可观的提升，并且可以通过购买 Intel 的 IPP 高性能多媒体函数库(Integrated Performance Primitives)得到更快的处理速度。本平台使用了OpenCV 2.4.3。

Opencv结构：

①cxcore

基础结构：CvPoint,CvSize,CvScalar等

数组操作：cvCreateImage,cvCreateMat等

动态结构：CvMemStorage,CvMemBlock等

绘图函数：cvLine,cvRectangle等

数据保存和运行时类型信息：CvFileStorage,cvOpenFileStorage等

错误处理和系统函数:cvGetErrStatus,cvAlloc,cvFree等

②cv

图像处理：cvSobel，cvCanny等

结构分析：ContourArea等 :（轮廓分析，轮廓提出等）

运动分析与目标跟踪：cvMeanShift等

模式识别：CvHaarFeature

摄像头定标与三维重建：cvCalibrateCamer2

③Machine Learning(ML)

包含许多聚类、分类和数据分析函数。如Bayes分类器，K近邻算法，支持向量机，决策树，神经网络等等。

④HighGUI

图像界面函数：cvNamedWindow

读图像和保存图像：cvLoadImage，cvSaveImage

读视频和写视频：CvCreateFileCapture

⑤cvcam

摄像机接口，在Opencv1.0以后的版本中已经被移除。

⑥cvaux

该模块中一般存放一些即将被淘汰的算法和函数（如基于嵌入式隐马尔科夫的人脸识别算法），还包含一些实验性的算法和函数（前景检测，背景剔除等）。

## 5.2 ServerCamera模块功能划分

ServerCamera模块共有4个功能，分别是GetCamera调用摄像头、GetPicture拍照、MoveDetect移动中的人脸识别、CutImage对阅读证数字切割。构造一个ServerCamera类，继承平台的Server类，将功能作为类的方法，类示意图如下：

|  |
| --- |
| Class ServerCamera:Server |
| 方法： |
| void work();  size\_t getInputMessageSize();  size\_t getOutputMessageSize();  bool get\_camera();  bool get\_picture();  bool cut\_image(char\* image);  bool move\_detect(); |

5‑1 ServerCamera类示意图

在ServerCameraMessage.h中定义了消息结构体，与客户端进行通讯。ServerCamera实例化后向Master注册，Client即可直接调用ServerCamera提供的服务。如图：

①Register

②Look up Server

③Communicate

5‑2 通讯结构示意图

### 5.2.1 GetCamera调用摄像头

通过OpenCV的获取视频的抽象接口结构CvCapture调用摄像头，将每一帧展示在窗口上，将其封装成get\_camera方法，核心代码如下：

|  |  |
| --- | --- |
|  | //ServerCamera.cpp |
| 1 | CvCapture\* pCap = cvCreateCameraCapture( -1 ); |
| 2 | while ((frame = cvQueryFrame(pCap)) != 0 && cvWaitKey(20) != ' ') |
| 3 | {  frame = cvQueryFrame(pCap);  cvShowImage("Camera", frame);  } |

### 5.2.2 GetPicture拍照

调用摄像头后，画出矩形框，在矩形框内扫描借阅证卡号区域，通过设置函数cvWaitKey（20）==‘ ’，即按下空格键，将当前帧存成图片保存至当前目录，完成拍照功能。核心代码如下：

|  |  |
| --- | --- |
|  | //ServerCamera.cpp |
| 1 | while (1) |
| 2 | { |
| 3 | if (key==' ') |
| 4 | {  CvRect rc=cvRect(pt1.x,pt1.y,200,50);  cvSetImageROI(frame, rc);  sprintf(fileName,"Image %d.jpg",++count);  cvSaveImage(fileName,frame);  cvXorS(frame,cvScalarAll(255),frame);  cvShowImage("Camera",img\_binary);    cvWaitKey(200);  } |
| 5 | } |

### 5.2.3 MoveDetect移动中的人脸识别

OpenCV中有检测人脸的函数，该函数还可以检测一些其他物体， 甚至还包含一些预先训练好的物体识别文件。人脸检测的主要步骤为：

1.加载分类器。

用cvLoad函数读入xml格式的文件。文件在OpenCV安装目录下的“data/haarcascades/”路径下haarcascade\_frontalface\_atl.xml和haarcascade\_frontalface\_atl2.xml

2.读入待检测图像。读入图片或者视频。

3.检测人脸。

核心代码如下：

|  |  |
| --- | --- |
|  | //ServerCamera.cpp |
| 1 | CvHaarClassifierCascade \* pCascade = 0; // the face detector |
| 2 | CvMemStorage \* pStorage = 0; // expandable memory buffer |
| 3 | CvSeq \* pFaceRectSeq; // list of detected faces |
| 4 | int i; |
| 5 | // initializations |
| 6 | pStorage = cvCreateMemStorage(0);      pCascade = (CvHaarClassifierCascade \*)cvLoad(("/home/becky/OpenCV-2.4.3/data/haarcascades/haarcascade\_frontalface\_alt.xml"), 0, 0, 0 ); |
| 7 | // detect faces in image |
| 8 | pFaceRectSeq = cvHaarDetectObjects          (frame, pCascade, pStorage,          1.1, // increase search scale by 10% each pass          3, // drop groups of fewer than three detections          CV\_HAAR\_DO\_CANNY\_PRUNING, // skip regions unlikely to contain a face          cvSize(0,0)); // use XML default for smallest search scale |
| 9 | } |

   4.检测结果表示。通过画个方框表示。

### 5.2.4 CutImage对阅读证数字切割

本函数提供了将图片中数字进行切割成独立的图片，可以作为分类器学习的数据集，将图片中的数字转成文本，完成了数字识别的功能。

该图像处理分为三个部分：

1.图像二值化

图像的二值化就是将图像上的像素点的灰度值设置为0或255，这样将使整个图像呈现出明显的黑白效果。图像的二值化使图像中数据量大为减少，从而能凸显出目标的轮廓。

通过OpenCV的cvCvtColor(src,dst\_gray,CV\_BGR2GRAY)函数得到灰度图，然后调用函数 cvThreshold 对单通道数组应用固定阈值操作。设置合适的阈值后既得到二值图。实验过程中，得出值为120时分辨效果最好。该函数支持的对图像取阈值的方法由 threshold\_type 确定。

2.轮廓检测

得到二值图后，调用cvFindContours函数从二值图像中检索轮廓，并返回检测到的轮廓的个数。

3.图片分割

检测出轮廓后对轮廓序列进行分割，将图片大小改为32\*32，使用cvRectangle 绘制矩形框，cvSetImageROI函数锁定轮廓，cvCopy复制到结果图片上，cvResize设置图片大小。核心代码如下：

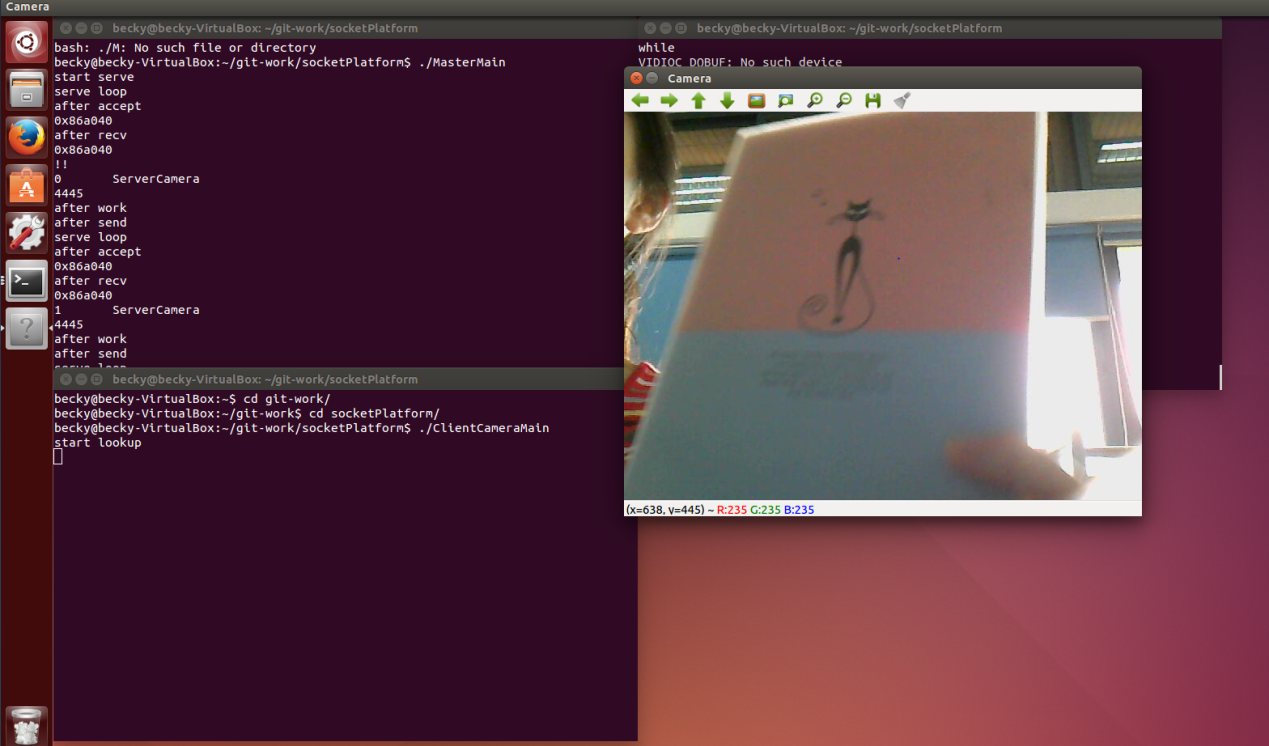
|  |  |
| --- | --- |
|  | //ServerCamera.cpp |
| 1 | for(;first\_contour!=0;first\_contour=first\_contour->h\_next |
| 2 | { |
| 3 | cnt++; |
| 4 | cvDrawContours(img\_temp, first\_contour, cvScalar(100), cvScalar(100), 1 ); |
| 5 | CvRect rect = cvBoundingRect(first\_contour,0); |
| 6 | if(rect.height<130&&rect.width>3) |
| 7 | { |
| 8 | CvSize size,dst\_size; |
| 9 | cvRectangle(img\_temp, cvPoint(rect.x, rect.y), cvPoint(rect.x + rect.width, rect.y + rect.height),CV\_RGB(255,255, 255), 1, 8, 0); |
| 10 | IplImage \*result; |
| 11 | size.width=rect.width; |
| 12 | size.height=rect.height; |
| 13 | result=cvCreateImage( size, img\_temp->depth, img\_temp->nChannels ); |
| 14 | //从图像中提取子图像  cvSetImageROI(img\_temp,rect);  cvCopy(img\_temp,result);  cvResetImageROI(img\_temp);  cvShowImage("resize", img\_temp);  sprintf(filename,"Image%d.jpg",++cnt);  dst\_size.width=32;      dst\_size.height=32;  IplImage\* file=0;  file=cvCreateImage( dst\_size, img\_temp->depth, img\_temp->nChannels );  cvResize(result, file, CV\_INTER\_LINEAR); //缩放源图像到目标图像  cvSaveImage(filename,file); |
| 15 | } |

## 5.3 测试

写客户端进程，客户端ClientCamera继承Client类，即可通过发送消息调用ServerCamera的功能。例如测试功能getcamera客户端代码如下：

|  |  |
| --- | --- |
| 1 | void ClientCamera::run() |
| 2 | { |
| 3 | myLookup("ServerCamera"); |
| 4 | myOutput->commandName = getcamera; |
| 5 | request(); |
| 6 | cout << myInput->returnValues[0] << '\t' <<myInput->returnValues[1] << endl; |
| 7 | } |

实验效果图如下：



5‑3 测试结果图

其他各功能模块测试类似，只需修改commandName的值，调用功能。