班　级　031111
学　号　03111047

西安电子科技大学

本科毕业设计论文

题　　目　手机拍照手势识别算法

学　　院　计算机学院

专　　业　计算机科学与技术

学生姓名　吕莹

导师姓名　郭杏莉

摘　要

随着 Android 设备的进一步普及，传统点触式交互方式已无法满足当今需求。人们日益期待一种解放双手的人机交互模式，基于手势识别的人机交互便应运而生。

而对于普通拍照应用，特别是自拍，人们更加渴望一种无需接触即可拍照的模式。本文便着力于这一需求，提出了一种手机拍照手势识别算法。

本文首先介绍了下 Android,OpenCV 的相关知识。随后重点叙述了算法中的几个关键技术，如图像预处理，手势特征提取，线程冲突解决等。接着我们详细介绍了算法的设计与实现，从手势选取，识别流程到算法封装发布，并验证了其理论可行性。最后，我们利用前文的手势识别算法编写了一个自拍相机 Demo，通过实验证明其具有实际可用性。

关键词： **安卓操作系统**　OpenCV　**手势识别**　**手势检测**　**人机交互**

ABSTRACT

With the further popularization of Android devices, the traditional touch-based interaction has been unable to meet the needs of today. People increasingly expect a hands free interactive mode, based on human-computer interaction gesture recognition have come into being.

For ordinary camera applications, particularly the self-timer, a no-contact with people more eager to take the picture mode. This article will focus on this need, a new camera phone gesture recognition algorithm.

This paper introduces the relevant knowledge under Android and OpenCV's. Then focus on the algorithm described in several key technologies, such as image preprocessing, feature extraction gesture, thread conflict resolution. Then we detail the design and implementation of the algorithm, selected from gesture recognition algorithm processes to release the package, and to verify the feasibility of the theory. Finally, we use gesture recognition algorithm previously written a self-timer camera Demo, proved by its practical usability.

Keywords: Android OS OpenCV Hand Gesture Recognition Gesture Detection Human Computer Interaction

目　录

第一章　绪论

1.1　课题研究背景及意义

在科技高速发展、研究层出不穷、需求不断更新的今天，外设硬件或屏幕触碰作为机械式输入设备，在某些方面来说，很难进行 3D 和高度自由的输入，这种交互方式对人们的日常生活来说并不方便，需要一个学习并且适应的过程。随着研究工作的更加深入，人们更多的开始关注人脸识别[1,2]、人眼跟踪、人体跟踪与识别及手势跟踪与识别等更加符合人类习惯的人机交互技术。

“手势”在人机交互中，不单单具有生动、形象而且直观的特点，更具有很强的视觉效果[3]，因此手势完全可以作为一种交互手段。研究人机交互形式，人体语言与有声语言的融合，对于提高机器视觉，进而提高人机交互接口的实用性是非常有意义的。

随着人机交互技术的发展，最初的交互方式大部分是以实体按键方式控制智能设备，而现在更多的发展为以屏幕触控方式为主，甚至有些设备已经完全淘汰了实体按键的交互方式。然而，随着用户需求的不断提高以及开发者头脑风暴的创意不断增加，不仅仅传统按键被市场渐渐淘汰，目前主流的屏幕触控方式也不能完全满足用户需求，这些操作方式都有一定的局限性，主要体现在用户需要接触到设备才能够完成操作，因此手势识别突破了距离的限制，给用户带来了全新的体验。随着这种以非接触式自然手势操作技术在 Android 系统中逐渐发展成熟，必将带来一场全新的技术革命，为开发者开拓眼界，更新了思维模式，提升在现在的发展中，很多软件都与手势识别相结合，如红外设备、Kinect 等。但是对于摄像头识别手势拍照系统并没有很成型的作品，在拍照过程中可能因为一手持手机，另一只手拿东西或其他原因对按键造成不便，能把 OpenCV 移植到 Android 平台上使用是对图像图像处理在了市场竞争力。对移动终端上方便的进行具有的巨大意义。因此在拍照过程中加入手势识别是很有发展空间与发展潜力的项目之一。

1.2 课题研究现状

1.2.1 Android 软件发展现状

Android 系统自推出以来，就以明显的优势逐渐扩大自大的市场份额，尤其在国外，其呼声日高，可谓是如日中天，正处于蓬勃发展的开拓阶段[4]。据美国某市场调研机构 2012 年发布的一份最新报告显示。2012 年一季度在美国，基于 Android 系统的智能手机的销售量已占据全美手机销售量的 28% 份额，而大名顶顶的 IPhone 手机其市场份额紧追其后，占到 21% 的市场份额，已经确定了 Android 系统的市场占有比。据业内人士分析，随着 Android 系统相应软件的不断开发应用，选择 Android 系统手机或者无线终端设备的人会越来越多，其市场霸主的地位在更新更好的系统出现之前是不可动摇的。

未来基于 Android 系统的应用软件将进入飞速发展的全新阶段。Android 系统的应用绝不仅局限于手机产业，几年来其迅速扩张到相关领域，例如平板电脑、车载系统、电视 STB、智能电器、智能会议系统等。如今，各 IT 厂商都在努力的研发前沿应用软件，以期在 Android 系统发展这一群雄逐鹿的关键阶段，占领更多的市场份额。

目前对 Android 的发展方向一类是偏向硬件驱动，一类是偏向软件应用[5]。从目前的招聘需求来看，后者的需求最大，包括手机游戏、手机终端应用软件和其他手机应用软件的开发。随着各种移动应用和手机游戏等内容需求日益增加，也将激励大中小型手机应用开发商加大对 Android 应用的开发力度，因此 Android 人才的就业前景也非常广泛。几乎每一个 android 手机用户都是游戏的需求者，都是潜在的顾客，现今的 android 用户不过是冰山一角，随着 android 手机市场进一步壮大，游戏的市场容量将具备较大的增长空间，游戏开发者不会愁吃不饱，只会愁胃口不够大。

如今国内的 Android 开发还是主要以应用开发为主，主要分成 3 类：为企业开发应用、开发通用应用（放到 Android Market 或者其他 App Market 销售）以及游戏开发（放到 Android Market 或者其他 App Market 销售)[6] 。第一类开发者一般身处规模较大的公司，这些公司主要为自有品牌或者其他品牌设计手机或者平板电脑的总体方案。除了根据需求对系统进行定制外，更多的工作在于为这些系统编写定制的应用。第二类开发者，一般处于创业型公司或者是独立开发者，他

们的盈利方式主要是 2 种：为国外公司进行外包开发，或者通过 Google 的移动广告（AdMob）通过广告点击分成。而理论上的通过付费下载的形式来盈利的，现在国内鲜见成功者。第三类开发者，目前和第二类开发者类似。

丰富的应用软件程序有游戏、生活、新闻、阅读器、记事本、天气预报、文件或行事历管理及硬件管理方面的服务软件，而工具型软件将随着 Android 系统日趋完善而减少，除了 UI 客制化及动作感测等结合硬件及软件的应用外，扩增实境应用也将逐渐进入 Android 应用领域中，更是值得关注的发展方向之一。

1.2.2　手势识别研究现状

手势是人类沟通交流不可分割的一部分：人们在相互交流时总会指手划脚。而手势识别技术又开辟了我们与机器、设备或电脑互动的新局面。手势识别的前景非常令人期待，特别是对于智能手机和平板电脑而言。这些设备已经深入人们生活的各个方面，而新的通信接口始终非常受人们的欢迎。

采用摄像头跟踪进行手势识别的技术实际上已经使用了一段时间。领先的游戏机，如微软的 Xbox 和索尼公司的 Play Station，都配置了手势识别设备（分别是 KINECT 和 Play Station Eye）。目前，这两种设备已经升级至第七代和第八代。不过，移动设备的手势识别技术仍面临几个重要问题，包括在不利的光线条件下，该技术能够实现的效果，背景的变化与高功耗等。然而，业内普遍认为，这些问题可以通过不同的跟踪解决方案和新技术克服。

这两年来智能移动终端设备特别是 Android 设备数量迅猛增长[7]，很多开发者也将目光投降了除了触控以外其它的交互方式，例如语音识别技术，现在已日趋成熟，手势等肢体语言的交互形式将是另一个发展趋势，并有可能成为发展主体。近年来陆续出现了一些使用肢体语言交互的应用，如眨眼拍照的相机应用、笑容触发的相机应用、眼球转动来控制的阅读器应用等。相比之下更直观的手势交互应用必将成为热门。

1.3　论文主要内容安排

本论文共分为六章，内容安排如下：

- 第一章: 绪论。本章主要介绍了课题研究背景及意义、Android 软件发展现状及手势识别的发展现状。

- 第二章: 相关知识。本章介绍了 Android 平台，NDK，OpenCV 和常见的图像处理方法，为后文详细叙述算法打下基础。

- 第三章: 关键技术。本章主要介绍了算法过程中的几个关键技术，包括图像预处理技术（灰度处理，平滑处理），手势特征提取技术（Canny 检测轮廓，直线检测等等）还有相机线程冲突的解决方案。由于本算法中我们为了加速运算，用到了部分 C 语言，所以最后还介绍了 Android 平台下 C 和 Java 的交叉编译方法。

- 第四章: 算法设计与实现。本章详细介绍了算法的设计与实现，从手势选取方案到算法具体流程，再到最后的封装结果。

- 第五章:Demo 实现与实验结果。本章给出了 Demo 的测试环境，框架结构，实现上的关键难点和实验结果。

- 第六章: 总结与展望。本章总结了本文的主要工作，说明了具体的研究成果与不足，给出了下一步的工作方向。

第二章　相关知识

本章介绍了 Android 平台，NDK，OpenCV 和常见的图像处理方法，为后文详细叙述算法打下基础。

2.1　Android 平台

2.1.1　Android 起源及发展状况

Android 一词最早出现于法国作家利尔亚当在 1886 年发表的科幻小说《未来夏娃》中。他将外表像人的机器起名为 Android。2010 年 2 月 3 日，Linux 内核开发者 Greg Kroah-Hartman 将 Android 的驱动程序从 Linux 内核“状态树”（“staging tree”）上除去，从此，Android 与 Linux 核心开发分道扬镳。

Android 是 Google 开发的基于 Linux 平台的开源手机操作系统[8] 。它该平台由操作系统、中间件、用户界面和应用软件组成，是一个全面整合的软件栈，号称是首个为移动终端打造的真正开放和完整的移动软件平台。you 谷歌与开放手机联盟合作开发了 Android，这个联盟由包括中国移动、摩托罗拉、高通、宏达和 T-Mobile 在内的 30 多家技术和无线应用的领军企业组成。通过与运营商、设备制造商、开发商和其他有关各方结成深层次的合作伙伴关系，我们希望借助建立标准化、开放式的移动电话软件平台，在移动产业内形成一个开放式的生态系统。我们认为此举必将推进更好、更快的创新，为移动用户提供不可预知的应用和服务。

Android 作为谷歌企业战略的重要组成部分，将进一步推进" 随时随地为每个人提供信息" 这一企业目标的实现。由数据表明，全球为数众多的移动电话用户从未使用过任何基于 Android 的电话。谷歌的目标是让移动通讯不依赖于设备甚至平台。出于这个目的，Android 将会将其补充，而不会替代谷歌长期以来奉行的移动发展战略：通过与全球各地的手机制造商和移动运营商结成合作伙伴[9] ，开发既有用又有吸引力的移动服务，并推广这些产品。开放手机联盟的成礼和 Android 的推出是对现状的重大改变，在带来初步效益之前，还需要不小的耐心和高昂的投入。但是，我们认为全球移动用户从中能获得的潜在利益是值得付出这些努力

的。如果你也是一个开发者，并对我们的想法感兴趣，就请再给我们一星期的时间，届时谷歌便能提供 SDK 了。

Google 与开放手机联盟（Open Handset Alliance）合作开发了 Android 移动开发平台，这个联盟由摩托罗拉、高通、宏达电和 T-Moblie、中国移动等在内的 30 多家移动通讯领域的领军企业组成。Google 与运营商、设备制造商、开发商和其他第三方结成了深层次的合作伙伴关系，希望通过建立标准化、开放式的移动电话软件平台，在移动产业内形成一个开放式的生态系统。

Android 作为 Google 企业战略的重要组成部分，将进一步推进“随时随地为每个人提供信息”这一企业目标的实现。全球为数众多的移动电话用户从未使用过任何基于 Android 的移动通讯设备，Google 的目标是让移动通讯不依赖于设备甚至平台。出于这个目的，Android 将补充而不会代替 Google 长期以来奉行的移动发展战略：通过与全球各地的手机制造商和移动运营商结成合作伙伴，开发即有用又有吸引力的移动服务，并推广这些产品。

2.1.2 Android 平台及其优点

Android 是开源的，开源是一种的完全开放的开发模式，有着很多移动终端平台的优点，主要有以下几点：

1. 简单性

   开源软件解决方案很容易找到和很容易实施，许多架构师和开发人员都熟悉这个技术的架构。开源软件团体推动开源软件开发人员提供使用方便的框架和平台。开源软件解决方案还能够让企业迅速创建一些解决方案以提供有形的和可衡量的好处。

2. 开放性

   开源软件本身的灵活性允许比专有软件产品更大的自由和个性化。这就意味着一个机构能够从开源软件的安装中看到与自己的业务关系更密切的更大的价值。

3. 价格负担低

   Android 是一款基于 Linux 平台的开源操作系统，从而避开了阻碍市场发展的专利壁垒，是一款完全免费的智能手机平台。而 WindowsMobile 高达 20

多美元的单台授权费相比，采用 Android 系统的终端可以有效的降低产品成本;Android 系统对第三方软件开发商也是完全开放和免费的。Android 系统承载着 Google 帝国的梦想。

2.1.3　Android 系统架构原理

对操作系统而言, 必须做到设计合理、层次分明, 同时还需考虑整个系统的结构要聚耦适当,Android 系统是基于 Linux 内核的, 因此还必须具备开源的特性, 以符合开源人员共同工作。

从系统的组成要件来讲,Android 平台架构包括硬件设备、板级支持包、驱动程序、操作系统内核、程序运行库, 运行框架, 应用程序等, 它们的有机结合和协同工作共同完成了整个系统的正常运行和对事务的处理。

依据 Google 开源资料可知[10] , 整个系统由 Linux 内核、程序库、Android Runtime、应用程序框架和应用程序等 5 部分组成，系统架构如图2.1所示。

参照图2.1 , 由上而下对组成系统各部分的主要组件作以下描述。

1. Linux 内核

   Android 基于 Linux 2.6 内核, 但并非完全照搬内核, 而是对内核作了部分增删和修改, 在 Linux 2.6 内核的基础上,Android 核心系统实现了安全性、内存管理、进程管理、网络协议栈和驱动模型等功能,Linux 内核也同时作为硬件和软件栈之间的抽象层。

   硬件驱动程序: 完成与各种硬件的通信,Linux 内核提供了大部分设备的驱动程序, 如显示屏, 摄像头, 内存, 键盘, 无线网络, 音频设备, 电源等组件。

   系统内存管理: 对所有可用的内存进行统一编码管理, 定义一整套内存定位, 使用与回收的策略。

   系统进程管理: 内核管理进程的创建与销毁, 管理进程间的通信, 以及采取必要的措施避免死锁等内容。

   网络管理系统: 无线网络设备工作原理, 内核掌控如何读取网络设备中的缓存数据。

2. 程序库

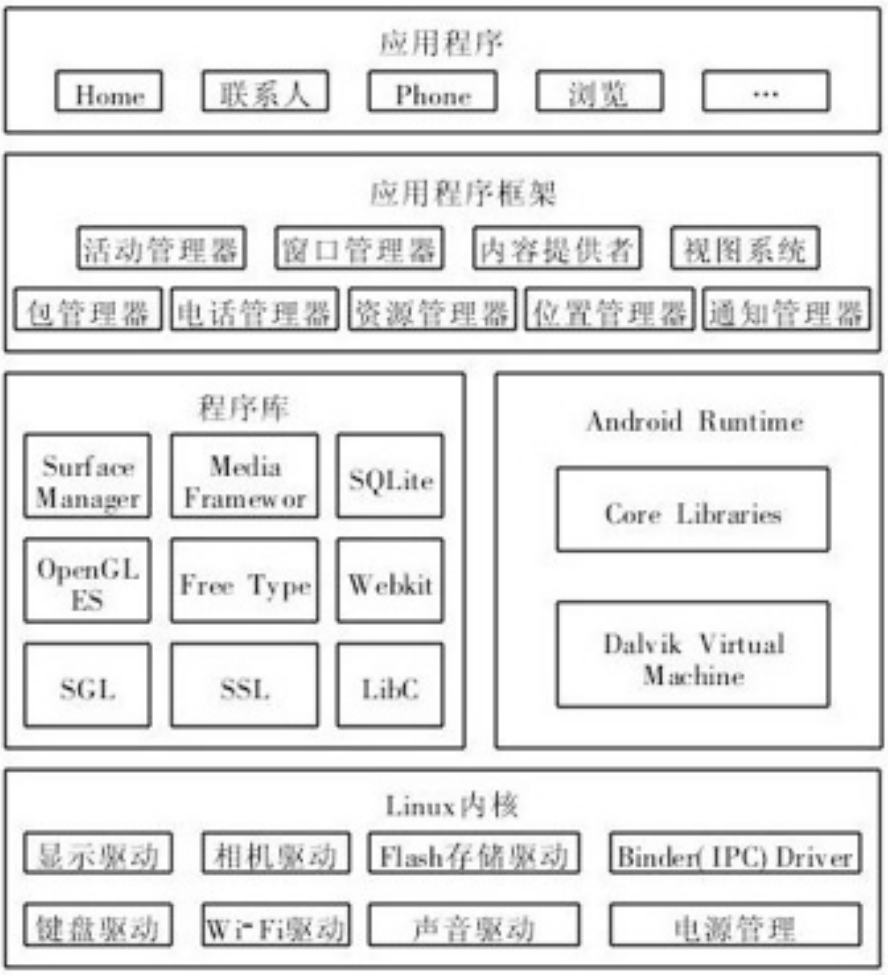

图 2.1 Android 系统架构图

程序库是指可供使用的各种标准程序、子程序、文件以及它们的目录等信息的有序集合,Android 包含一些 C/C++ 库,Android 系统中不同的组件通过应用程序框架可以使用这些库, 以下是一些核心库:

Surface Manager: 管理显示子系统, 并且为多个应用程序提供 2D 和 3D 图层的无缝融合。

Media Framework: 基于 OpenCORE 的多媒体框架, 支持多种常用的音频、视频格式文件的回放和录制, 同时支持静态图像文件。

SQLite: 一个对于所有应用程序可用, 功能强劲的轻型关系型数据库引擎。

OpenGL ES:3D 图形库, 用于 3D 图形渲染, 该库可以使用 3D 硬件加速。

FreeType: 位图 (Bitmap) 和矢量 (Vector) 字体显示。

WebKit: 支持 Android 浏览器和一个可嵌入的 Web 视图。

SGL:2D 图形库, 用于 2D 图形渲染。

LibC: 一个从 BSD 继承的标准 C 系统函数库, 它是专门为基于嵌入式 Linux 设备定制的。

3. Android 运行库 (Android Runtime)

Android 运行库包括两部分: 一是核心库, 二是自身的虚拟机。

核心库提供 Java 编程语言核心库的大多数功能。Dalvik 虚拟机是 Google 专为 Android 开发的, 比 SunJava 虚拟机的效率更高, 功能也更为复杂, 以更好的支撑 Android 平台, 并拥有独立的版权。每一个 Android 应用程序都在自己的进程中运行, 都拥有一个独立的 Dalvik 虚拟机实例, Dalvik 虚拟机执行.dex 的可执行文件, 该格式文件针对小内存的使用进行了优化, 同时虚拟机是基于寄存器实现的, 所有的类由 Java 编译器编译, 然后通过 SDK 中的相应工具转化成.dex 格式, 最后由虚拟机执行。

4. 应用程序框架

应用程序框架是指定义了一个应用程序运行所必须的全部功能组件, 开发者也可以访问核心应用程序所使用的 API 框架。该应用程序的架构设计简化了组件的重用; 任何一个应用程序都可以发布它的功能块, 并且任何其他的应用程序都可以使用其所发布的功能块 (应该遵循框架的安全性限制)。同样, 该应用程序的重用机制也使用户可以方便地替换程序组件。

5. 应用程序

Android 系统发布时, 会同一系列核心应用程序和常用程序一起发布, 如常用的手机功能程序, 包括语音电话、通讯录、短信收发、照相、话机设置等; 数据应用程序, 包括邮件工具, 日程表, 浏览器, 地图导航等, 以及 Android Market 上的各种应用程序; 所有的应用程序都是使用 Java 语言编写。

2.2　NDK 简介

NDK 即 Native Development Kit，NDK 是 Google 发布的一系列开发工具集合，这个工具可以帮助 Android 应用程序开发者编译来自 C/C++ 语言编写的源代码，然后嵌入到他们自己的应用包中，供应用程序调用。Android 的虚拟机在应用程

序的源代码中通过 JNI 调用方法实现本地代码。在内核层面，这意味着你的应用程序源代码可以用关键字 native 方式声明一个或多个方法，用于指出这些方法是通过本地代码方式实现的；另外你必须提供一个本地共享库包含上述用关键字 native 声明的方法的具体实现代码。这个库是要被打包到应用程序的 apk 文件中，而且库的命名也要跟 Unix 的规则，即 lib<something>.so，将包含一个标准的 JNI 入口点。

NDK 集成了交叉编译器，并提供了相应的 MK 文件隔离 CPU、平台、ABI 等差异。MK 文件相当于配置文件，开发人员在 MK 文件中设置一些参数、欲编译的文件及编译特性的要求等, 这样编译器就可以根据 MK 文件创建出动态库 so 文件。为了更好的与 C/C++ 开发相结合，也在不断完善开发中，目前 NDK 提供了一些稳定 API 头文件声明，但是在功能上还是相当有限，包括 C 标准库 (libc)、标准数学库 (libm)、压缩库 (libz)、Log 库 (liblog)。随着版本的不断升级，功能将越来越强大。

2.3 OpenCV 简介

OpenCV 是近年来推出的开源、免费的计算机视觉库，利用其所包含的函数可以很方便地实现数字图像和视频处理[11] 。同时利用面向对象的 VC++ 6.0 编程工具，用 C++ 语言进行程序编写，大大提高了计算机的运行速度。本文首先阐述了 OpenCV 的特点以及结构，然后以平滑处理、图像形态学为例介绍了 OpenCV 在数字图像处理中的典型应用。OpenCV 算法库为 VC++ 编程处理数字图像提供了很大的方便，其必将成为图像视频处理领域的强有力的工具。

利用 OpenCV 视觉库，科研开发人员只需添加自己的编写程序，直接调用 OpenCV 中的函数即可实现，这样不仅降低了开发程序的难度，而且缩短了相关程序的开发周期。

2.4 常见图像处理方法

在手势识别中有很多常用的图像操作，例如图像的剪切、图像的缩放、图像的旋转以及图像的亮度调整[3] 。我们以 *Pixe1Array[x,y]* 来表示原有图像的二维像素矩阵, 原有图像的高度用 H 来表示, 原有图像的宽度用 *W* 来表示, 用

PjxelArrayNew[x,y] 来表示结果图像的二维像素矩阵, 结果图像的高度为 *HNew*, 宽度为 *WNew*。

2.4.1　图像的剪切

图像的剪切操作的函数原型为 *void CImage::eliplmage(eReet&reet)*, 需要输入的参数为要剪切的图像的具体的位置, 即要剪切图像左上角的位置和右下角的位置。

假定要剪切图像左上角位置为 *(left,top)*, 右下角的位置 *(right,bottom)*，那我们只需要调整图像的宽度为 *(righi-left)*, 调整图像的高度 *(bottom-top)*，从图像的二维数组中提取左上角到右下角之间的数据。剪切后图像的数据可以表示为:

$$PixelArray[x,y],\quad left \le x \le right \ \ \wedge \ \ top \le y \le bottom$$

2.4.2　图像的缩放

图像的缩放操作的函数原型为 *void CImage::CollapseOrExpandlmage(double dbRatioX,double dbRatioY)*[12]。输入的参数为图像水平方向需要缩放的尺度以及图像垂直方向需要缩放的尺度。

假定水平方向缩放的比例为 *a*, 垂直方向缩放的比例为 *b*, 那么, 缩放后图像的高度为 *bH*, 缩放后图像的宽度为 *aW*, 对于原有图像的任一点 *PixelArray[x,y]*, 在缩放后对应的像素为:

$$PixelArray[i,j],\quad ax \le i \le a(x+1) \ \ \wedge \ \ by \le j \le b(y+1)$$

令 *CountArray[i,j]* 为结果图像中点 *(i,j)* 对应原有图像像素的计数, 那么每次有点映射到点 *(i,j)* 的像素时, 将结果图像中点 *(i,j)* 的像素值加上该原有图像点 *(x,y)* 的像素值, 并将 *CountArray[i,j]* 加 1。

$$PixelArrayNew[i,j] = PixelArrayNew[i,j] + PixelArray[x,y]$$

$$CountArray[i,j] = CountArray[i,j] + 1$$

由于经缩放后, 结果图像中的某一点的像素可能对应原有图像中的多个像素, 所以对于这种情况需要对该点的像素求取平均值:

$$PixelArrayNew[i,j] = \frac{PixelArrayNew[i,j]}{CountArray[i,j]}$$

图像缩放操作可以用一个简单的示意图 (图2.2) 来表示:

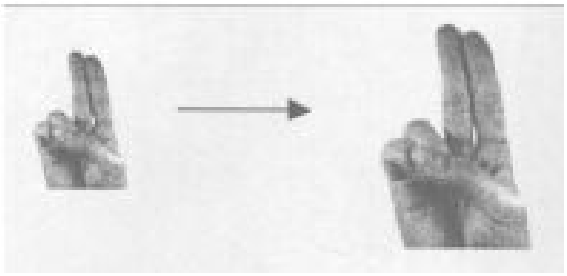

图 2.2 缩放示意图

2.4.3 图像的旋转

图像的旋转在所有的通用操作中是最复杂的, 它的函数原型为 *void CImage::Rotatelmage(double dbAngle)*。从中可以看出, 传入的参数是需要旋转的角度, 该角度以弧度值来表示, 正数表示的是逆时针方向旋转, 负数表示的是顺时针方向旋转。

旋转操作的示意图如图2.3所示 (逆时针旋转 Q 度):

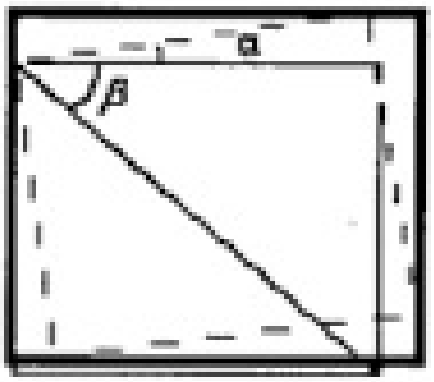

图 2.3 旋转示意图

图中细黑色的表示原来图像的范围, 粗黑色的为旋转后图像的范围, 虚线的为原来图像经旋转后在结果图像中的位置。

旋转操作首先需要判断该角度属于哪一个象限, 不同的象限所使用的变换公式是不同的。假定输入的旋转角度为 θ, 令 θ_1 为 θ 经变换后在 $[0, 2\pi]$ 之间的角度,

令 α 为 θ 经变换后在 $[0,\pi/2]$ 之间的角度, 其中:

$$\alpha = \begin{cases} \theta_1 & 0 \le \theta_1 \le \pi/2 \\ \theta_1 - \pi & \pi/2 < \theta_1 \le \pi \\ \theta_1 - \pi & \pi < \theta_1 \le 3\pi/2 \\ \theta_1 - 3\pi/2 & 3\pi/2 < \theta_1 \le 2\pi \end{cases}$$

令

$$NW = Wcos\alpha + Hsin\alpha$$

$$NH = Wsin\alpha + Hcos\alpha$$

对于原图像中的点 (x,y)，在新图像中对应的坐标为 (x',y')，那么

1. θ 在第一象限时

$$HNew = NH, WNew = NW$$

$$x' = xcos\alpha + ysin\alpha$$

$$y' = (W - x - 1)sin\alpha + ycos\alpha$$

2. θ 在第二象限时

$$HNew = NW, WNew = NH$$

$$x' = (W - x - 1)sin\alpha + ycos\alpha$$

$$y' = NW - 1 - xcos\alpha - ysin\alpha$$

3. θ 在第三象限时

$$HNew = NH, WNew = NW$$

$$x' = NW - 1 - xcos\alpha - ysin\alpha$$

$$y' = NH - 1 - (W - x - 1)sin\alpha - ycos\alpha$$

4. θ 在第四象限时

$$HNew = NH, WNew = NW$$

$$x' = NH - 1 - (W - x - 1)sin\alpha - ycos\alpha$$

$$y' = xcos\alpha + ysin\alpha$$

这样在任意的旋转角度下, 原图像中的每一点都可以对应到新图像中的某一点中, 考虑到经过旋转后在新图像中对应原图像部分的内部可能存在某些点在原图像中不存在对应点, 所以需要对这些点作平滑处理, 使图像达到连续的效果。假定对于图像中的任一点 $PixelArrayNew[x, y]$，如满足条件:

$$\begin{aligned} & PixelArrayNew[x, y] \ \wedge \\ PixelArrayNew[x-1, y] \neq 0 \ \ & \wedge \ \ PixelArrayNew[x+1, y] \neq 0 \\ PixelArrayNew[x, y-1] \neq 0 \ \ & \wedge \ \ PixelArrayNew[x, y+1] \neq 0 \end{aligned}$$

则

$$\begin{aligned} & PixelArrayNew[x, y] = \\ & \quad (PixelArrayNew[x-1, y] + PixelArrayNew[x+1, y] \\ & \quad + PixelArrayNew[x, y-1] + PixelArrayNew[x, y+1])/4 \end{aligned}$$

第三章　关键技术

本章主要介绍了算法过程中的几个关键技术，包括图像预处理技术（灰度处理，平滑处理），手势特征提取技术（Canny 检测轮廓，直线检测等等）还有相机线程冲突的解决方案。由于本算法中我们为了加速运算，用到了部分 C 语言，所以最后还介绍了 Android 平台下 C 和 Java 的交叉编译方法。

3.1　手势图像预处理

3.1.1　灰度处理

在进行视频流目标识别与跟踪时，通常第一个步骤就是对采集到的彩色图像进行灰度化[13]，这是因为黑白照片数据量小，相比彩照更易实现实时算法，另一方面黑白照片是由未处理的光线所形成的照片，因此从图像处理学角度来看，这种未经特殊滤光处理的图片所涵盖的信息更有价值。

由于在 OpenCV 中自带函数可以实现图像灰度化，因此在该问题处理中直接调用函数:*cvCreateImage*。实现代码如下。

```
1 cvNamedWindow("image",CV_WINDOW_AUTOSIZE );//创建窗口，窗口名字为image
2 cvShowImage("image",img);//在刚创建的image窗口中载入图像
3 //创建一个与img相同大小的图像img1
4 IplImage *img1 = cvCreateImage(cvGetSize(img),IPL_DEPTH_8U,1);
5 //色彩空间转换，将源彩色图像img转化成目标灰色图像imag1
6 cvCvtColor(img,img1,CV_BGR2GRAY); //关键
7 cvNamedWindow("GrayImage",CV_WINDOW_AUTOSIZE);//创建窗口，窗口名字GrayImage
8 cvShowImage("GrayImage",img1);//载入转化后的图像
9 cvSaveImage("/lena_gray.jpg",img1,0);
```

3.1.2　平滑处理

消除图像中随机噪声的技术。对平滑技术的基本要求是在消去噪声的同时不使图像轮廓或线条变得模糊不清。常用的平滑方法有中值法、局部求平均法和 k 近邻平均法。局部区域大小可以是固定的，也可以是逐点随灰度值大小变化的。此外，有时应用空间频率域带通滤波方法。

对于平滑处理，在 OpenCV 函数库中也有相应的函数[14]:

CV_BLUR_NO_SCALE (简单不带尺度变换的模糊) - 对每个象素的 $param1 \times param2$ 领域求和。如果邻域大小是变化的，可以事先利用函数 *cvIntegral* 计算积分图像。

CV_BLUR (simple blur) - 对每个象素 $param1 \times param2$ 邻域求和并做尺度变换 $1/(param1 \times param2)$。

CV_GAUSSIAN (gaussian blur) - 对图像进行核大小为 $param1 \times param2$ 的高斯卷积。

CV_MEDIAN (median blur) - 对图像进行核大小为 $param1 \times param1$ 的中值滤波 (i.e. 邻域是方的)。

CV_BILATERAL (双向滤波) - 应用双向 3×3 滤波，彩色 $sigma = param1$，空间 $sigma = param2$。param1 平滑操作的第一个参数。param2 平滑操作的第二个参数。对于简单/非尺度变换的高斯模糊的情况，如果 param2 的值为零，则表示其被设定为 param1。

3.2 手势特征提取

特征提取是计算机视觉和图像处理中的一个概念。它指的是使用计算机提取图像信息，决定每个图像的点是否属于一个图像特征。特征提取的结果是把图像上的点分为不同的子集，这些子集往往属于孤立的点、连续的曲线或者连续的区域。

特征提取与选择一般分为三个步骤: 特征形成、特征提取、特征选择。

特征形成有信号获取或测量得到原始测量，之后处理后得到图像特征，这样的特征称作原始特征，如手势图像属于数字图像，它的原始测量是各点灰度值, 但是这些测量不适于作为特征, 还要经过计算产生一组原始特征。

特征提取原始特征的数量可能很大, 或者说样本是处于一个多维空间中, 通过映射 (变换) 将原始特征变换为较少的新特征, 这个过程称作特征提取。映射后的特征又称作二次特征, 它们是原始特征的某种组合, 所以特征提取在广义上就是一种特征映射。如公式(3-1)所示。

$$A : Y \to X \tag{3-1}$$

其中 Y 是测量空间,X 是特征空间,A 称为特征映射, 又称作特征提取器。

特征选择从原始特征中挑出一些最具有代表性和分类性最好的特征, 这个过程称为特征选择, 它可以起到降低特征空间维数的作用。

3.2.1　Canny 检测轮廓

Canny 边缘检测器利用 Canny 算子进行检测，是目前最精确的检测器, 并且已经被大量运用于程序中。从目前看来，canny 边缘检测在做图像轮廓提取方面是最优秀的边缘检测算法。

canny 边缘检测采用双阈值值法，高阈值用来检测图像中重要的、显著的线条、轮廓等，而低阈值用来保证不丢失细节部分，低阈值检测出来的边缘更丰富，但是很多边缘并不是我们关心的。最后采用一种查找算法，将低阈值中与高阈值的边缘有重叠的线条保留，其他的线条都删除。

函数如下:

```
1  int main()
2  {
3      Mat I=imread("../cat.png");
4      cvtColor(I,I,CV_BGR2GRAY);
5
6      Mat contours;
7      Canny(I,contours,125,350);
8      threshold(contours,contours,128,255,THRESH_BINARY);
9      namedWindow("Canny");
10     imshow("Canny",contours);
11     waitKey();
12     return 0;
13 }
```

效果如图3.1所示。

3.2.2　直线检测

直线在图像中出现的频率非常之高，而直线作为图像的特征对于基本内容的图像分析有着很重要的作用，我们通常通过 OpenCV 中的 hough 变换来检测图像中的线条。

首先展示最基本的 Hough 变换函数 HoughLines，它的原型如下所示。

图 3.1 Canny 轮廓检测效果图

```
1 void HoughLinesP(InputArray image, OutputArray lines, double rho, double
    theta,int threshold, double minLineLength=0, double maxLineGap=0 );
```

它的输入是一个二值的轮廓图像，往往是边缘检测得到的结果图像；它的输出是一个包含多个 *Vec2f* 点的数组，数组中的每个元素是一个二元浮点数据对 *<rou,theta>* 而言，*rou* 代表直线离坐标原点的距离，*theta* 代表角度。第 3 和第 4 个参数代表步长，因为 *Hough* 变换实际上是一个穷举的算法，*rho* 表示距离的步长，*theta* 代表角度的步长。第 5 个参数是一个阈值设置直接的最低投票个数，知道 *Hough* 原理的，这个参数应该很容易理解。

从这个函数的输出结果我们可以看出，得到的直线并没有指定在图像中的开始点与结束点，需要我们自己去计算，如果我们想把直接显示在图像中就会比较麻烦，而且会有很多角度接近的直线，其实它们是重复的，为了解决上面这些问题，*OpenCV* 又提供了一个函数 *HoughLinesP()*。它的输出是一个 *Vector of Vec4i*。*Vector* 每一个元素代表一条直线，是由一个 4 元浮点数组构成，前两个点一组，后两个点一组，代表了在图像中直线的起始和结束点。

3.2.3 轮廓的提取与描述

在目标识别中, 首先要把感兴趣的目标提取出来，而一般常见的步骤都是通过颜色或纹理提取出目标的前景图（一幅黑白图像，目标以白色显示在图像中），接

下来要对前景图进行分析进一步地把目标提取出来，而这里常常用到的就是提取目标的轮廓。

OpenCV 里提取目标轮廓的函数是 *findContours*，它的输入图像是一幅二值图像，输出的是每一个连通区域的轮廓点的集合：*vector<vector<Point»*。外层 *vector* 的 *size* 代表了图像中轮廓的个数，里面 *vector* 的 *size* 代表了轮廓上点的个数。

提取到轮廓后，最重要的就是如何把这些轮廓转换为可以利用的特征，也就是涉及到轮廓的描述问题，这时就有多种方法可以选择，比如矢量化为多边形、矩形、椭圆等。OpenCV 里提供了一些相应函数，在这里就不详细介绍了。

3.3　照相机线程冲突解决方案

由于 Android 系统原因，Camera 为一个非共享资源，即同一时间只能由一个线程使用。而我们的拍照手势识别算法的目的就是用手势控制相机，故必然要有相机的预览。这时就存在着一个难点: 如何让预览和 OpenCV 的 VideoCapture 同时获得图像。

传统上，面对此问题有两种解决方案:

1. 利用两个线程分时占用相机，交替获得控制权。

2. 利用 VideoCapture 从相机获得图片，再传入前端 View 渲染。

这两种方法固然可以解决，但是方法 1，由于交替切换控制权，存在大量延时。而方法 2 除去编码复杂，更是 OpenCV 的 VideoCapture 采集的祯尺寸一般是线程初始化是就预设好的，而为了处理速度，一般都设为较小尺寸，但是如果直接在前端渲染，则就会过于模糊。因此，传统上并没有解决此问题的好方法。

本算法中，我们采用类似于方法 2 的方法，但是我们采用了间隔采集和预设尺寸两种方法兼顾了速度与相片质量。

1. 间隔采集：顾名思义，就是每隔一定的间隔才采集一帧画面，我们采取每 40ms 采集一次，即 25FPS，保证了预览的流畅，同时降低了运算复杂度，保证了速度。

2. 预设尺寸：前文说到，VideoCapture 是在线程初始化时预设好尺寸的，因此我们就在一开始预设了较大的尺寸，而由于间隔采集，大尺寸并不会造成太

大的处理压力，我们先将大尺寸彩色图片传至前端渲染，再将其缩小灰度化后传入后端计算。从而保证了预览质量和相片质量。

3.4 Android 平台 C 和 Java 交叉编译

Android 平台下，C 和 Java 的交叉编译主要通过 NDK 实现。首先在基于 Android 应用程序框架下进行 Java 端的开发[15]；然后通过 OpenCV 与 JNI 接口编写本地的 C/C++ 代码，并利用 Android NDK 对其进行编译得到 JNI 接口 (so 文件)，然后运用 Java 端编译后生成的 Java 代码可调用动态链接库 so 文件，最后通过 SDK 编译打包并生成应用程序，整体逻辑图如图3.2所示。实际上可以完全交由智能 IDE(如 Android Studio) 完成。

图 3.2 交叉编译逻辑图

第四章　算法设计与实现

本章详细介绍了算法的设计与实现，从手势选取方案到算法具体流程，再到最后的封装结果。

4.1　手势选取

考虑到本算法为手势拍照设计的专用算法，故手势不需太多。为了保证识别性能，我们选取了五种基本手势，即上划，下划，左划，右划和点击。

4.2　算法流程

本算法流程图如图4.1所示。

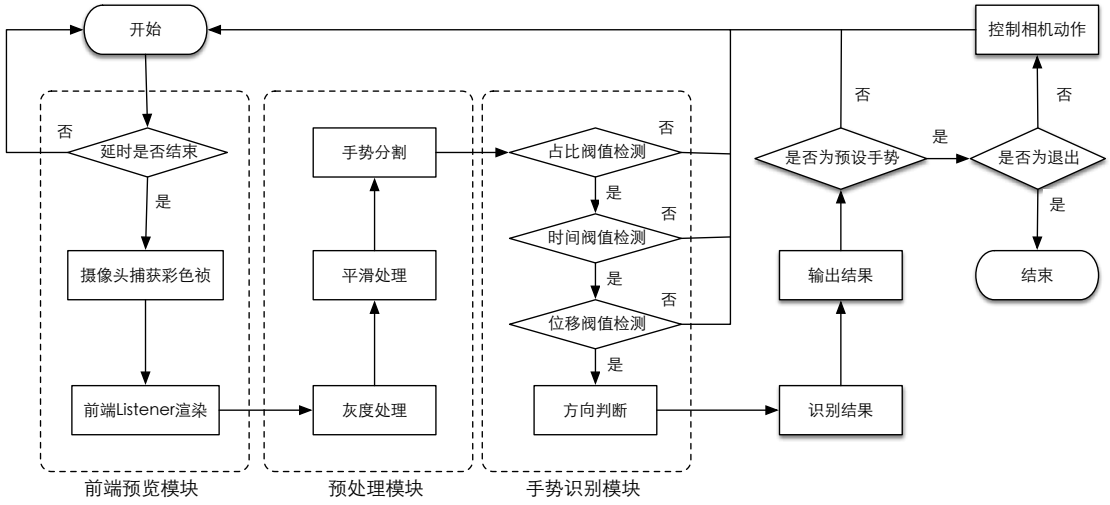

图 4.1 算法流程图

本算法可分为三个模块，分别是前端预览模块，预处理模块和手势识别模块。

4.2.1　前端预览模块

前端预览模块包含从祯捕获到前端展示所有过程。首先用一个单独的线程控制倒计时延时，当时间到达后捕获一个彩色祯，本文中为一个 640*480 大小的祯。

捕获完成后传至前端进行渲染。本文采用观察者设计模式，故渲染过程只需将祯传递给前端监听者即可，不需本算法实现。

4.2.2 预处理模块

本模块为手势识别的预处理阶段，包含三部分，灰度处理，平滑处理，手势分割。

1. 首先我们将采集到的彩色祯利用库函数进行灰度处理。
2. 随后我们再进行平滑处理，我们利用 Opencv 库函数对图像进行平滑处理，并且为了有效识别点击效果，我们采用去 100 祯画面的灰度平均值作为一个衡量因子。
3. 最后，通过比较前后两祯的差异确定运动物体，为了加速比对过程，我们采用 C 语言进行两祯比对。

4.2.3 手势识别模块

手势识别模块负责具体的手势识别过程。

- 首先，我们先检测了运动的手势是否占到了预设的最小画面比重阀值，即运动物体与整个画面的占幅比，如果比重过小则忽略。实际中，我们采用 $1/10$ 为最小占比阀值。
- 随后，我们又判断手势运动时间是否大于最小阀值，如果小于阀值，我们则认为手势并没有正确表达，因此忽略。实际中，我们采用 $1.5 \times 500ms$ 为最小时间阀值。
- 接着，我们又进行了位移阀值判断，判断手势运动距离有没有达到预定最小位移，如果没达到，则认为移动过小，可以忽略。实际中，我们采用画幅大小的三分之一作为最小位移阀值。
- 经过前 3 步检测，我们可以认定已检测到了手势。进而只需进行方向判断即可判断是上述前四种手势的哪一种。而对于点击手势，则是利用预处理阶段的 100 祯画面灰度平均值作为基准，如果灰度突然增加大于最小点击阀值（实际中，我们采用 0.3 倍平均灰度值）则判定为点击。至此，手势识别完成。

随后我们在 log 中输出识别结果，判断是否为预设手势，如果是，则先判断是否是退出手势，如果是退出，则直接退出程序，如果不是，则执行相应操作 (同样采用观察者模式)。

4.3　算法封装结构

为了方便算法的移植与使用，我们将本算法进行了封装，将其封装为了一个 library，其中 CameraGestureSensor 对外暴露方法和接口，如图4.2所示。Camer-

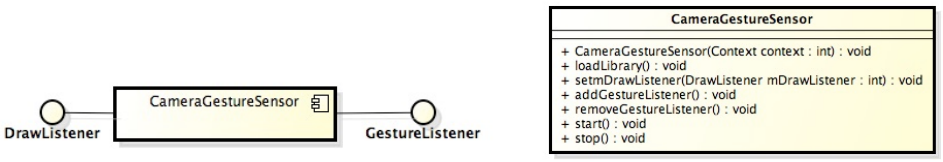

图 4.2 组件图和类图

aGestureSensor 类对外暴露两个接口均为观察者模式中的监听者接口:DrawListener 和 GestureListener。

1. DrawListener: 其中包含一个 draw(Mat) 抽象方法，监听者利用传入的 Mat 在前端渲染。

2. GestureListener: 包含五个抽象方法，分别为 onGestureUp,onGestureDown，onGestureLeft，onGestureRight，onClick。监听者利用重载这几个实现真正功能调用。

封装实际结果如图4.3所示。

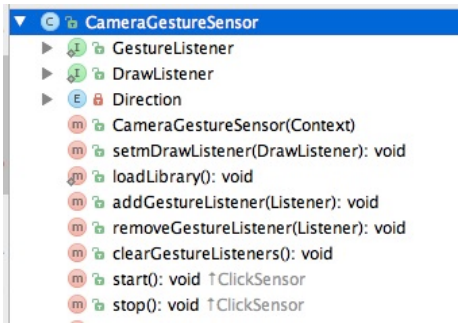

图 4.3 实际结构截图

第五章　Demo 实现与实验结果

前文我们详细介绍了我们提出的手机拍照手势识别算法，并从理论上分析了其可行性。本章我们将在实际情况下完成一个 Android 平台的 Demo，并进行测试，给出测试结果。

5.1　测试环境

测试手机为: 魅族 MX2 M045,Android 4.4.4。编译环境为:Mac OS X 10.10.3，1.7GHz Intel Core i7，8GB 1600MHz DDR3，128G SSD。所有代码均利用 Android Studio 1.2,Java 在 JRE 1.6.0，C++ 在 Clang 600.0.56 环境下编译完成。

5.2　Demo 框架

本 Demo 主要功能为测试算法可用性，所以功能上为简单的四点，左划延时 3s 拍照，右划退出，上划放大，下划缩小。Demo 的组件图如图5.1所示。Demo 的主类依赖于 OpenCV,Android OS，与 CameraGestureSensor 为相互依赖关

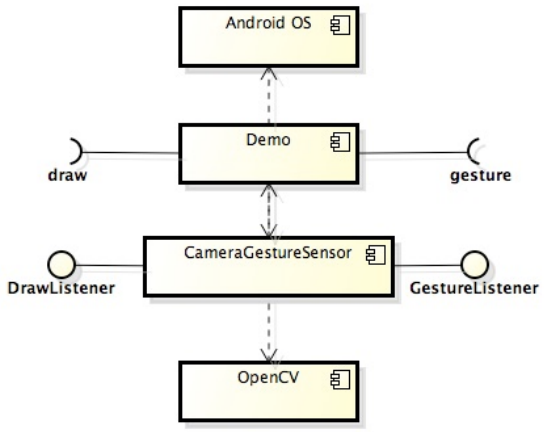

图 5.1 Demo 组件图

系。CameraGestureSensor 需要 Demo 对 OpenCV 进行初始化，而 Demo 则需要 CameraGestureSensor 提供接口。Demo 调用 Android OS，实现前端渲染和相机控制。

5.3 关键技术实现

本节将具体叙述 Demo 中两个难点，即前端渲染函数和相机控制的具体实现。

5.3.1 前端渲染

- 首先，我们在布局文件中放置一个 SurfaceView 用于手动绘制相机预览画面。
- 随后，将 CameraGestureSensor 传入的 Mat 利用 BitMap 转为一个位图
- 接着，利用当前缩放值 scale，进行矩阵变换，绘制一副缩放后的位图
- 最后，锁定画布，将当前画面替换为缩放后的位图，解锁画布。一次绘制完成。

实际实现关键代码如下:

```
1  Core.flip(currentFrame,currentFrame,1);
2  Bitmap bmp=processFrame(currentFrame);
3  Bitmap real;
4  if (bmp != null) {
5      Canvas canvas = mView.getHolder().lockCanvas();
6      if (canvas != null) {
7          Matrix matrix = new Matrix();
8          matrix.postRotate(90);
9          matrix.postScale((float) 1.0 * canvas.getWidth() / bmp.getHeight(),
               (float) 1.0 * canvas.getHeight() / bmp.getWidth());
10         real=Bitmap.createBitmap(bmp,0,0,bmp.getWidth(),bmp.getHeight(),
               matrix,true);
11         if(scale!=1) {
12             int x = (int) (canvas.getWidth() * (scale - 1) / 2.0);
                                int y = (int) (canvas.getHeight() * (scale - 1)
                   / 2.0);
13             matrix.reset();
14             matrix.postScale(scale, scale);
15             real = Bitmap.createBitmap(real, x, y, real.getWidth() - x,
                   real.getHeight() - y, matrix, true);
16         }
17         if(is_taken==true){
18             is_taken=false;
19             takePicture(real);
```

```
            }
            canvas.drawBitmap(real, 0, 0, null);
            mView.getHolder().unlockCanvasAndPost(canvas);
            real.recycle();
        }
        bmp.recycle();
    }
```

需要注意的一点就是 bmp 最后一定要释放，否则则容易出现内存问题。

5.3.2　相机控制

相机控制无非有 2 种，一种是缩放，一种是拍照。

在前段渲染中，我们看见了在第三步时利用了一个缩放值进行绘制，实验中，我们就是利用此方法进行缩放的，也就是传统意义上的数字变焦。即利用一个 scale 记录变焦倍数，默认为 1，恒大于 1，小于 5(因为大于 5 时已过于模糊，没有意义)。

而对于拍照，我们利用一个单独线程实现倒计时，当倒计时结束时我们直接将预览画面存至存储器中。需要注意的是在存储前要先判断存储文件夹是否存在，若不存在则需要自行新建。

5.4　实验结果

5.4.1　Log 记录

图5.2为算法后台运行的 log。可见，我们的间隔采集的理论速度应该是 25FPS，但是由于软硬件性能因素实际只有 16FPS，不过并不影响使用，仅为看起来略有卡顿。手势识别记录一切正常。

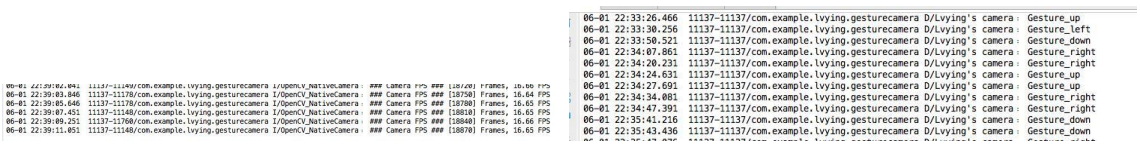

图 5.2 log 记录

5.4.2 CPU 与系统内存占用

CPU 与系统内存占用如图5.3所示。可见，本算法对内存及计算量的要求并不算高，并未占用太多系统资源，可以实际运用。

图 5.3 CPU 与内存消耗占比图

5.4.3 拍照实例

当手势被识别后，进行对应功能处理。如果为左划则进入拍照功能，在屏幕右上角显示倒计时 3 秒，并在倒计时结束时拍照。如果是放大或缩小功能，由于前置摄像头无法实现物理变焦，故采用数字放大/缩小实现该功能。因为识别为动态过程，不方便截图展示，因此我们只截取了一个识别到左划手势并开始拍照倒计时的界面，和一个拍照完成的提示。如图5.4所示。

图 5.4 拍照展示

经过实验，本算法已具有一定的实际使用可行性，可以进行试验性使用。

第六章　总结与展望

基于视觉的手势识别长期以来都是人机交互领域研究的热点。本课题在近两年来智能移动终端设备爆炸式增长的大背景下，针对智能移动终端设备的特点，对手势的检测和识别方法做了深入研究，并选取 Android 智能平台为应用背景，设计了一个 Android 平台上的实时手势交互系统，这是在智能移动设备上研究人机交互手段所做出的新尝试，本文所做的主要工作和取得的成果主要以下有几点：

- 第一，在 OpenCV 调用摄像头与处理图像占用 CPU 方面进行的分频处理，使得在处理时间和效率上得到了大大的提高，并且直线检测的速度较快，计算量较小，和同类处理方式相比适合在移动设备上使用。
- 第二，在手势识别方面，本文首先研究了手势分割技术，良好的手势分割是识别的前提，本文通过边缘检测和直线提取两个步骤得到了较好的手势分割效果。
- 第三，在应用方面，本文设计了一个基于 Android 平台的实时手势识别系统，还演示了通过调用该库实现的一个使用特定手势控制摄像头完成拍摄的应用案例，这是研究智能终端设备上人机交互技术的一次新的尝试。

目前在识别过程仍有一些不足，归纳起来，造成错误识别的主要原因有以下两条：

- 第一，手势是 3D 的图像，实验中采用了 2D 的形式来表现 3D 图像，造成了一些手势信息的丢失，这是错误识别的主要原因。
- 第二，获取的图像质量不高，背景中的景物比较复杂，这些由景物产生的像素对手势轮廓的提取造成了干扰，使最终的特征提取造成错误。

本课题在未来还可以在以下四个方面加以研究：

- 第一，在现有的手势检测基础上进行手势跟踪，本文没有研究手势跟踪的算法，但如果要达到更好的交互效果就应该加入手势跟踪。

- 第二，降低软件的 CPU 和内存的占用率, 目前完成的图像处理软件在这方面做得并不够好, 占用手机的资源比较多。

- 第三，实现图像亮度自动调整的功能, 由于日常生活中由于曝光不足的照片很常见, 亮度自动调整具有比较现实的意义。可以增加一些特定功能, 比如针对现有的年轻人群对照片的要求, 增加美白, 瘦脸, 瘦身等功能

- 第四，完善 Android 手势拍照应用，进一步提高软件的稳定性和识别的准确性，使其在实际生活中具有实用性，争取将其应用发布到 Android 应用市场，供广大 Android 设备用户使用。

致　谢

转瞬间四年的大学生活即将结束，在即将走出校门迈入工作生活之际，回首这三年来的学习历程，我满怀一颗感恩之心，感谢一路走来给予我帮助的各位老师、同学和亲人。

首先，我想把最诚挚的谢意和深深的祝福送给郭杏莉老师。本论文是在郭杏莉老师的指导和关怀下完成的，在此对郭杏莉老师对我的指导表示诚挚的感谢。使我在学习能力上得到了充分的锻炼，她，严谨的治学精神，实事求是的研究作风，对事业充满热情的投入和勤奋工作的态度不时地感染着我，教育着我，激励着我在学习和生活中不断进取。

感谢我的家人和男朋友，他们一直给予我精神上的鼓励、物质上的支持、学习上的督促。他们是我克服困难、不断追求进步的精神支柱和动力源泉！

最后感谢参加论文评审和答辩的各位老师对本文的认真审阅。

参考文献

[1] EROL A, BEBIS G, NICOLESCU M, et al. Vision-based hand pose estimation: A review[J]. Computer Vision and Image Understanding, 2007, 108(1): 52 – 73.

[2] LEE J, KUNII T L. Model-based analysis of hand posture[J]. Computer Graphics and Applications, IEEE, 1995, 15(5): 77 – 86.

[3] 何阳清. 基于几何特征的手势识别算法研究 [D]. 上海: 上海海事大学, 2004.

[4] 张华, 刘铁英. Android 应用软件发展趋势与关键技术探索 [J]. 商场现代化, 2012(28): 175 – 175.

[5] 张建平. 基于 Android 校园即时信息系统研究 [J]. 电子设计工程, 2013, 21(20): 117 – 120.

[6] 许瑾. 基于 Android 平台音乐播放器的设计与实现 [D]. 北京: 北京邮电大学, 2011.

[7] 王赞超. 基于 Android 平台的视觉手势识别研究 [D]. 西安: 西安电子科技大学, 2013.

[8] 单李旺. Android 操作平台的研究与应用 [D]. 天津: 南开大学, 2009.

[9] 罗翔. Symbian 平台软件下载管理系统的设计与实现 [D]. 北京: 北京邮电大学, 2010.

[10] 王茜. Android 嵌入式系统架构及内核浅析 [J]. 电脑开发与应用, 2011, 24(4): 59 – 61.

[11] 秦小文, 温志芳, 乔维维. 基于 OpenCV 的图像处理 [J]. 电子测试, 2011(7): 39 – 41.

[12] 殷涛. 基于几何矩的手势识别算法 [D]. 上海: 上海海事大学, 2004.

[13] 黄柏林. 基于边界特征的人脸识别 [D]. 上海: 上海海运学院, 2002.

[14] 王昭威, 李钊. 基于 OpenCV 的图像特征提取与匹配 [J]. 软件, 2013, 34(11): 147 – 148.

[15] 祝志远, 张庆辉. 基于 OpenCV 实现 Android 系统对摄像头的调用及操作 [J]. 信息与电脑(理论版), 2015, 2: 002.