PaperPass检测报告简明打印版

比对结果（相似度）：
总　体：16 % *（总体相似度是指本地库、互联网的综合比对结果）*
本地库：16 % *（本地库相似度是指论文与学术期刊、学位论文、会议论文数据库的比对结果）*
互联网：0 % *（互联网相似度是指论文与互联网资源的比对结果）*

编　号：57491AC943434YNEZ
标　题：人脸跟踪与识别系统实现
作　者：杨晨
长　度：17870 字符(不计空格)
句子数：592句
时　间：2016-5-28 12:12:57
比对库：学术期刊、学位论文（硕博库）、会议论文、互联网资源
查真伪：http://www.paperpass.com/check

句子相似度分布图：

本地库相似资源列表（学术期刊、学位论文、会议论文）：

1. **相似度：1 % 篇名:《人脸检测与识别技术的研究》**
   来源：学位论文 重庆大学 2010 作者: 张建慧
2. **相似度：1 % 篇名:《基于ARM移动机器人中的人脸识别》**
   来源：学位论文 暨南大学 2014 作者: 张立峰
3. **相似度：1 % 篇名:《浅析人脸识别技术的现状和发展趋势》**
   来源：学术期刊 《通讯世界》 2015年13期 作者: 陈奇毅
4. **相似度：1 % 篇名:《基于层次聚类的集成学习方法及应用研究》**
   来源：学位论文 湖南工业大学 2014 作者: 周济
5. **相似度：1 % 篇名:《基于fast-AdaBoost算法的人脸检测与识别方法研究》**
   来源：学位论文 太原理工大学 2014 作者: 齐光景
6. **相似度：1 % 篇名:《表情与光照变化下几种人脸识别方法的研究与改进》**
   来源：学位论文 哈尔滨工业大学 2009 作者: 陈昌凤

互联网相似资源列表：
没有找到与互联网相似度高的资源！

全文简明报告：

摘要

{45%：随着互联网越来越多的渗透进人们的生活，计算机技术已经越来越多的应用的我们的生活中。} {53%：而计算机图形图像技术在生活中被也越来越多的运用起来。} {50%：人脸跟踪与识别系统，涉及到人脸的检测与人脸的识别。} {43%：将其运用到Android系统之中，从而实现了通过手机摄像头识别人脸的功能。}

{43%：本文首先阐述了人脸检测Haar分类器、人脸识别LBP算法的相关内容，然后根据实际的Android开发内容编写了需求报告。} {40%：首先通过Android中Camera相关类获取视频流的一帧，然后使用Haar分类器进行人脸检测，获取人脸矩阵信息。} 对录入的人脸进行训练后再使用LBP算法进行人脸识别，最后在屏幕上绘制出人脸的矩阵与识别出的人的姓名。 如此循环，从而实现了对视频流的人脸识别。

关键词： 人脸识别； haar分类器； LBP算法； Android

Abstract

With the increasing penetration of the Internet into people ’ s lives, computer technology has been more and more used in our lives. The computer graphics technology is more and more used in our life. Face tracking and recognition system, related to face recognition and face detection. As it applied to Android system, enabling moblile phone camera face recognition function.

The paper describes the face detection Haar classifier, the content LBP face recognition algorithm, and then write a report based on the actual needs of the Android development content. Get through the first JavaCV Camera Class a video stream, and then use the Haar classifier for face detection, face matrix to obtain information. On the entry face training before using LBP face recognition algorithm, and finally draw the human face of the matrix and identified the person ’ s name on the screen. In the next frame, then the above operations, in order to achieve the recognition of the video stream.

keywords： face recognition； haar classifier； LBP algorithm； Android

目录

第1章 绪论

{64%：1.1 视频中人脸识别系统的研究背景及国内外研究现状}

{51%：在最近10年中，计算机科学技术取得了巨大的发展。} 在很多行业中，对人员进行的信息进行确认与身份的辨别的需求越来越大。 例如设置摄像头监控地铁，火车站等人流量大的地方，来追踪可能存在的罪犯。 在人脸识别还没有发展起来时这些任务都是由人工完成的，消耗了大量的人力，也消耗很多的时间； {41%：在银行取钱时，银行需要对取钱的用户进行人脸识别，判断其是否是本人；} {52%：在海关，同样也需要人脸识别技术来确认出入境人员是否为本人，这些在生活中切实存在的需求极大的推动了近些年人脸识别技术的飞速发展与应用[1]。}

{45%：人脸识别技术涉及到计算机图形学与机器学习，提取人脸的特征，通过这些特征，与训练好的样本进行对比，从而进行身份验证的一种技术。} 人脸特征与人身上其他的特征一样，例如指纹和虹膜，都具有唯一性和不容易被伪造的特性，这是可以用来作为身份鉴别的前提； {43%：相比于其他的身份识别技术，人脸识别技术不需要任何物理上的操作，结果更直观同时还具有一定的隐蔽性。} {42%：因此，在刑事案件的侦破，信息安全领域等都有广泛的应用前景。}

在早期，人脸识别技术有很大的局限性，只能对拥有单一背景的正面灰度图像进行识别。 {47%：这时的人脸识别技术是纯粹是停留在在二维层面的。} {42%：随着对多种姿态下人脸识别的研究，包括正面和侧面，人脸识别正在向三维方向发展，在维度提升的同时，识别率也在逐步提高。} 在这个过程中，人脸识别技术越来越成熟，但是还是存在着很多的问题，比如在复杂的背景下，系统很难识别出其中的人脸，跟踪也存在比较大的问题。 {44%：[2]人脸识别技术并非纯粹的数字技术，它需要研究人员拥有计算机图形学、计算机视觉、生物特征技术等学科的知识，} 是一个融合了多种学科的技术，因此对研究人员有非常高的要求。 环境和人脸本身都有很大的不确定性，在环境方面，光照和图像采集工具会对采集到的图像有比较大的影响， 而人脸本身由于表情，身体姿态和脸上戴的物件的不同，也会影响人脸识别的正确率。

人脸识别有悠久的研究历史。 {56%：高尔顿分别在1888年和1910年发表了关于利用人脸来识别身份的文章。} {64%：这两篇文章对人类自身如何进行人脸的识别进行了分析。} 在当时的科学文化环境下，并没有涉及到人脸的自动识别。

{63%：1965年陈和布莱德索在Panoramic Research Inc.发表了关于自动人脸识别的研究论文。} {75%：它历史性的创建了一个半自动化的人脸识别系统。} {60%：人脸特征点的间距和比率等特征是这个人脸识别系统的依据。} 这种思想也成为之后一段时间的主流。 这种人脸识别思想的特点有： {51%：①将人脸拆分成若干个部件，作为特征进行识别，主要利用的是各个部件的信息和部件之间的几何关系。} 这是一种很直观的方法，对人脸图像有很高的要求。 如果人脸图像中的人脸不是正面照或者表情有变化的话识别率就会很低。 {50%：为了弥补这种方法的缺点，后来出现了更好的人脸识别方法。} {41%：其中有一种是模板匹配法，其原理是用样本和待识别人脸的灰度图来进行比较识别人脸。} ②人脸识别对环境有非常高的要求。 通常在单一背景下，人脸的位置和大小等信息非常容易获得，相反，在复杂的背景下，人脸的位置非常难获得。 人脸识别由于这个原因，在当时并没有被应用到现实场景当中[3]。

{55%：从1964年至1990年，这26年是人脸识别的第一个阶。} {54%：这个阶段大多数研究者使用基于人脸几何特征的方法。} {60%：人工神经网络也曾被用于人脸识别的问题中。} 早期，除了布莱德索在进行自动人脸识别之外还有戈登斯泰因、金出武雄等。 {48%：金出武雄是人脸识别领域较活跃的人物，1973年，金出武雄完成了京都大学关于自动人脸识别的博士论文。}

1991年至1997年，这是人脸识别的第二个阶段。 {55%：在这个阶段，人脸识别的研究进入了高潮期，取得了很多的成就。} 许多经典的人脸识别算法在这个时期内诞生。 {58%：比如著名的FERET人脸识别算法，“特征脸”方法。} “特征脸”方法是这一时间段内的代表作，衍生出了之后许多的算法。 {44%：现在普遍使用的方法是特征脸与归一化协相关方法。} {58%：麻省理工学院人工智能实验室的布鲁内里和波基奥进行了基于结构特征的方法与模板匹配方法的对比，} {57%：得出了模板匹配要优于基于特征的方法这个结论，著名的Fisherface人脸识别方法便是在这个时期诞生的。}

第三个阶段是从1998年至现在。 为了解决在光照，姿态等非理想的情况下人脸识别鲁棒性较差的问题，研究者们重点研究了这一部分。 {47%：为了适应多姿态和多光照，吉奥盖迪斯等人提出基于光照锥模型的人脸识别方法。} {48%：还有一种适应多姿态多光照的识别方法是基于3D变形模型的。} 它是基于合成的分析技术被提出。 {50%：这种方法使得人脸可以用简单的举证特征作为特征，将大量弱分类器组合成了强分类器，} {44%：并且使用联级技术提高检测速度，为实时人脸识别打下了很好的基础。} {48%：同样是为了解决光照变化的问题，一种基于熵图像的人脸识别和绘制技术被提出。}

现在人脸识别的算法主要分成3种。

(1)基于肤色划分的人脸检测方法。 {42%：在确定肤色模型后从肤色区域中找出可能存在人脸的区域，再检测出人脸区域中有类肤色的非人脸物体。} 这种方法在复杂背景的环境下会有比较大的误差。 在复杂背景下，人脸区域可能和其他肤色区域混杂在一起，从而造成混淆，在获得肤色区域后任然无法判断出此区域是人脸区域。 另一种情况是光照和面部表情造成的影响。 光照和面部表情会对人脸检测造成影响，区域特征会受到影响，在这种情况下会使用聚类，归并，验证的方案来减少其带来的影响。 {45%：首先将人脸肤色像素按照严格的几何关系去拆分，再将其按照一定规则合并到一起，在合并过程中使用一些其他特征进行验证。}

(2)基于人工神经网的方法。 由于人脸轮廓的复杂性，人脸很难用数学模型表示，而神经网是一种可以表示复杂模型的方法。 人工神经网的优势是方便建模和较高的鲁棒性。 但是人工神经网在运算速度上有比较大的欠缺，这是制约它发展的一个因素。

(3)基于启发式模型的方法。 {54%：这种方法从图像中抽取若干特征进行人脸检测。} 由人脸局部特征衍

生到整体特征。 也就是利用人脸五官分布特征和知识模型从人的五官的检测到人的五官的相对位置的检测。 由于抽取的特征较少，所以这种方法有比较快的检测速度。 {41%：这种方法的主要障碍也是由于抽取特征较少，在复杂环境下的人脸检测成功率较低[4]。}

{67%：1.2 基于Android平台人脸识别系统研究目的及意义}

{63%：人脸识别在生活中有广泛的应用场景。} 比较常见的有安检，监控，身份认证等。 {44%：随着人脸识别技术渐渐成熟，人脸识别会更多的出现在我们生活中的各个角落。} 人脸识别的优势是新颖的交互方法和人脸的不变性，劣势是复杂的环境下识别率低。 研究的目的便是实现人脸识别，并且提高人脸识别的准确率，并且将其运用到实际生活中[5]。

随着支付宝刷脸支付， FaceU检测人脸实时添加表情，腾讯 QQ人脸登录等人脸识别技术在移动端的应用， {57%：人脸识别技术在移动端的应用也是越来越广泛。} {54%：学习人脸识别技术在移动端的应用具有重要的意义。}

1.3 基于Android平台人脸识别系统研究内容及目标

{43%：本文主要的研究内容是人脸检测与人脸识别在安卓终端的实现，其中主要内容有：}

(1)人脸检测算法，涉及到Haar分类器的研究。

{55%：(2)人脸识别算法，涉及到LBP人脸识别算法与样本训练。}

(3)学习使用基于opencv的JavaCV库。

(4)学习安卓客户端开发技术，。

项目的研究目标有：

(1)实现80%的人脸检测成功率。

(2)实现在非复杂背景下50%的人脸识别成功率。

{45%：(3)实现安卓客户端录入人脸样本并且完成训练。}

1.4 论文组织结构

{68%：本文一共分为7章，各章节具体介绍如下：}

第一章： 绪论。 {44%：主要介绍了人脸识别的发展前景与历史以及当前人脸识别主要使用的技术，讲述了研究人脸识别的目的和意义，最后列出了研究的具体内容与目标。}

第二章： 相关技术及开发工具简介。 简单介绍了人脸检测技术的定义和应用，人脸识别技术的定义和应用，介绍了Android IDE Android Studio，Java编程语言以及JavaCV图形图像库。

第三章： {51%：对基于安卓端的人脸识别系统进行了需求分析，分别进行了人脸检测模块分析，样本获取模块分析和人脸识别模块分析。}

第四章： 对算法进行了详细分析。 {51%：具体分析了Haar分类器算法和LBP人脸识别算法。}

第五章： 基于Android的视频人脸识别系统实现。 首先介绍了系统的整体结构，然后具体实现了每个子系统，并且最终将每个子系统联系到了一起。

第六章： 关键问题及解决方法。 {44%：主要讲述了开发过程中遇到的问题和相应的解决方法。}

第七章： {62%：对基于Android平台视频人脸识别技术的总结和展望。} {42%：对基于Android的人脸识别系统进行了总结，并且对系统可以提升和拓展的地方进行了展望。}

{67%：第2章 相关技术及开发工具简介}

2.1 人脸检测技术

{47%：人脸检测技术的输入是拥有或者不拥有人脸的图像，如果成功检测到人脸，输出人脸的所在的位置和大小。} 在最早的时候，人脸检测技术并没有获得与人脸识别技术相当的关注，直到研究者们发现在复杂环境下人脸获取极其困难， {44%：只有解决了人脸在复杂环境下的检测，才能顺利地进行人脸识别。} 比较常见的人脸检测方法有：

（1）参考模型法

（2）人脸规则法

（3）样品学习法

（4）肤色模型法

（5）特征子脸法

{60%：本文使用了基于Haar分类器的人脸检测方法进行人脸检测。} {44%：Haar分类器在人脸识别中使用，实际上就是对样本进行人脸和非人脸的分类。} {44%：Haar分类器进行分类的依据是haar-like特征，使用Adaboost算法来帮助机器学习，通过弱分类器的迭代可以快速准确地对非人脸进行过滤。} {50%：通过Android手机摄像头可以获取到视频流的一帧图像。} {48%：使用Haar分类器对这一帧图像进行分类，判断图像中是否有人脸存在，如果存在，则可以找到人脸的位置和大小信息，从而实现人脸的检测。}

2.2 人脸识别技术

{62%：人脸识别技术通常与人脸检测技术捆绑在一起。} 人脸识别技术一般分为3个过程：

(1)通过一定方法获取人面信息，生成面纹并且存储到面纹库。

(2)通过图像获取设备来获取当前人脸信息，生成面纹。

(3)将获取到的面纹与面纹库中的面纹通过一定方法进行对比，找到最相近的面纹，从而获取到识别出的对象。

现在主要的人脸识别方法有：

（1）几何特征的人脸识别方法

（2）基于特征脸的识别方法

（3）神经网络的人脸识别方法

（4）弹性图匹配的人脸识别方法

（5）线段Hausdorff距离的人脸识别方法

（6）支持SVM的人脸识别方法。

{72％：人脸识别技术具有非接触，非强制性，并发性等优点。} 人脸识别技术的缺点是人脸所

在背景对识别的准确度影响较大，人脸本身的表情和穿戴的装饰对人脸识别的结果也影响较大。

本文使用了LBP算法进行人脸识别。 {41％：LBP算法是一种纹理算法，在人脸识别的领域，LBP算法会将人脸分成很多小块，并且进行局部的描述，最终形成一个整体的描述。} {42％：通过与样本库的比较，判断当前图像是否匹配到了样本库的人脸，从而实现了人脸识别。}

2.3 开发工具

（1）Android Studio

Android Studio是Google推出的一个集成开发环境。 {43％：作为Eclipse的替代品，它为开发者提供了进行Android开发需要的开发工具与调试工具。} Android Studio使用Gradle进行项目管理。 Gradle构建系统同时支持本地文件系统和远程存储库支持的依赖，这样就不用把依赖库下载到本地了。 Android Studio 2.0提供了运行时修改代码，并且极大的提高了虚拟机的运行速度。 使用Android Studio可以方便的进行Android开发。 相比于之前流行的Eclipse，Android Studio在代码提示，界面，功能上显得更加优秀。 Android Studio唯一的缺点是对JNI编程不够友好。 使用JavaCV可以避免JNI编程，完美解决了问题。

2.4 开发技术

（1）Java语言介绍

想要进行Android开发，程序员必须学会使用Java语言。 Java编程语言作为一种面向对象语言，由于其拥有接

口和抽象类和异常处理等机制，使得使用Java写出来的程序往往有较强的健壮性。 使用Java进行编程不用担心内存问题，Java的内存回收机制可以自动在内存紧张的时候回收优先级低的对象，因此省去了内存操作的Java使用起来时相当方便的。 {40%：JDK是Java开发包，是一个开发工具的合集而JRE是一个纯粹的Java运行环境，包含了类库。} Java语言与JavaCV库的使用可以降低人脸识别系统制作的难度。

（2）JavaCV库介绍

JavaCV库是Github上的一个开源库，它实际上是对OpenCV进行了Java层面的封装，其中应用了Jni机制来调用OpenCV的c或c++接口。 {78%：OpenCV是一个跨平台的计算机视觉库。} 在Windows，Linux和Mac OS上都可以使用。 OpenCV使用C++编写，它主要的接口是面向C++语言的。 但它也给其他的语言提供了接口，比如说：Python，Ruby，MATLAB。 {72%：实现了图片处理和计算机视觉方面的很多算法[8]。} 相较于其他计算机视觉库，例如： Libvideogfx、mVision，OpenCV的优点很突出。 OpenCV的开源性保证我们可以免费、放心的使用而不用担心法律问题，并且保证了代码的健壮性和高效性； OpenCV的跨平台性使得我们可以在Android平台上顺利的使用； {45%：OpenCV内置的分类器和人脸识别算法可以使我们方便的集成人脸识别功能；} OpenCV还有丰富的示例程序，可以降低学习成本。

第3章 人脸识别需求分析

3.1 功能需求

人脸识别系统划分为4个模块。 第一个是人脸样本获取模块，第二个是视频流中图像获取模块，第三个是人脸检测模块，第四个是人图像中人脸识别模块。 这4个模块相对独立，但是需要为彼此提供接口，最终实现从图像获取并且进行人脸检测，最后进行人脸识别的功能，示意图如下：

{69%：图3-1 人脸识别系统模块示意图}

3.1.1 人脸样本获取模块

{42%：为了在进行人脸识别时提供样本，人脸识别系统必须提供给用户输入人脸样本的接口。}

{44%：人脸样本需要有两个要素，第一点是人脸图像，第二点是人脸所对应的身份信息。} 并且需要将这两个要点对应起来作为一个样本保存到本地。

图3-1样本结构示意图

使用摄像头获取人脸图像时需要对人脸进行预览，以方便用户判断当前位置是否可以获取到可靠的样本。 需要给用户提供拍摄按钮，以便用户进行拍摄。 {40%：拍摄人脸图像时可以进行前置摄像头和后置摄像头的切换，以方便用户选择获取样本的对象。}

在页面中央需要有一个给用户输入姓名的编辑框，编辑框可以输入10个以内的字符，不能为数字或者标点符号，不允许为空。 在输入姓名的编辑框下面需要提供确认按钮和取消按钮。 {46%：确认按钮点击后进入人脸拍摄页面，取消按钮点击后返回上一个页面。}

因为拍摄时人脸所处的环境可能比较复杂，光线较强，会对人脸样本的可靠性造成影响，所以需要在拍摄完成

后对图片进行裁剪。 用户在按下拍照键后跳转到裁剪页面。 裁剪页面显示裁剪框，裁剪框外使用阴影表示裁剪后会被去除。 图片可以通过多点触控来进行放大和缩小，双击屏幕可以进行特定倍数的放大和缩小，手指按在图片上时可以对图片进行拖动， 以便将想要留下的图片部分拖入裁剪框内。 {42%：图片裁剪时需要统一图片的尺寸，质量，图片格式。}

由于单张人脸样本并不可靠，所以需要一次性拍摄三张人脸来对应一个身份信息。 因此在完成姓名输入后，需要进行三次人脸拍摄和三次图片裁剪。

再完成一次输入后需要将样本以文件的形式保存到本地。

{57%：3.1.2 视频流中单帧图像获取模块}

{41%：进行视频中的人脸识别需要首先需要获得视频流，再获得单帧图像，判断其中是否有人脸，再进行人脸识别。} {57%：因此基于Android的人脸识别系统需要具有获取图像的能力。}

{45%：视频中的人脸识别实际上是对每一帧图像进行人脸识别，系统首先需要能够获取到摄像头的视频流。} 用户可以对当前摄像头拍摄的视频画面进行预览，视频画面应占满屏幕，以更好的展示给用户。 {41%：视频流应当采取可以支持的最大分辨率，以提高图像质量。}

获取视频流应当通过前置摄像头，主要是完成对当前持有设备者的人脸识别。 在获取视频流时应当将方向调整为始终为横向，从而与人脸检测模块的输入相匹配。

从视频流中获取单帧图像对用户是不可见的，在进行数据处理时预览画面不应当出现卡顿。

在进行预览时，用户可以通过返回键退出当前预览画面。

3.1.3 人脸检测功能需求

人脸检测模块接收获视频中单帧图像获取模块获取到的单帧图像，然后对图像进行进行检测，如果检测到人脸则将人脸矩阵绘制在相应的位置。 {50%：检测结果要求显示清晰，明了，一眼可以看出检测到的人脸位置和大小。}

{48%：视频流中的人脸检测实际上是对视频流的每一帧都进行人脸检测，在极短时间内检测出人脸或非人脸。} {41%：然后将获得的人脸图像以矩形的形式表现出来，用户可以在预览画面中看到检测出的人脸的位置和大小。}

{44%：人脸检测模块需要在进入预览页面时完成初始化。} 初始化时要显示加载框，加载框内需要添加文字“正在进行初始化”，不能造成阻塞。

由于需要对视频的每一帧都进行人脸检测，这要求人脸检测算法可以在较短时间内完成人脸与非人脸的区分。如果检测出非人脸，则不需要在预览画面绘制任何数据，如果检测出人脸，则将人脸以矩形框的形式绘制出来。

{43%：对图像中的人脸检测应当可以同时检测出图像中所有的人脸，并且同时表现出检测到的所有人脸的位

置和大小。｝

｛41％：人脸检测的结果需要作为人脸识别模块的输入，因此要统一人脸数据的数据结构。｝

3.1.4 人脸识别功能需求

｛54％：在成功获取到一帧图像后，进行人脸检测，如果检测到人脸则进行人脸识别。｝

人脸识别模块的输入是人脸所在区域带有人脸图像信息和位置信息的数据结构，人脸识别模块的输出是人脸对应的身份，与样本输入的内容对应，例如输出姓名。

｛46％：人脸识别模块需要在进入预览页面时进行初始化，人脸识别模块的初始化紧随着人脸检测模块，｝在进行初始时需要显示加载框，加载框内显示“正在进行初始化”，不能造成阻塞。

在获取到人脸检测模块检测出的结果后，人脸通过某一算法与本地人脸面纹进行对比，如果人脸匹配到本地的样本，则将对应的人脸信息输出。 如果人脸未匹配到本地样本，则不输出任何内容。

｛40％：人脸识别模块同样要求屏幕是在横向的，这样才能进行图像方向的统一。｝

｛41％：如果匹配到人脸，需要将对应的人脸信息显示到屏幕中，显示的位置位于屏幕中央偏上位置。｝ 例如： ｛44％：在人脸样本库中存在杨晨的人脸，杨晨使用此系统进行人脸识别，｝｛40％：首先人脸检测模块检测到人脸，将人脸处用红色框框出，然后人脸识别模块识别出杨晨的身份，｝ 则在屏幕中央偏上位置显示红色的文字“姓名： 杨晨”。

人脸识别模块需要有较快的响应速度，这样当获取到一帧图像后可以在较短时间内将人脸识别的结果显示到预览画面。

｛55％：人脸识别模块也需要有较高的人脸识别成功率。｝ 保证系统的可靠性。

3.2 非功能需求

｛41％：为了提高用户体验，人脸识别系统也需要有以下的非功能需求：｝

（1）可用性： ｛41％：样本收集，人脸识别的过程应当减少用户的操作，通过尽量少的步骤完成整个人脸识别过程。｝ 检测和识别结果应当简单明了的显示出来。

（2）可靠性： ｛40％：人脸检测和人脸识别的成功率应当比较高，可以适应复杂环境下的人脸识别需求。｝

（3）健壮性： 可以适配不同安卓版本，不同机型的终端，对异常有优秀的处理。 ｛46％：在没有检测到人脸或者未识别到人脸时有友好的提示。｝

（4）可维护性： 人脸识别系统的每一个模块应当相对独立，减少相对之间的依赖，提取公共函数。 ｛50％：保证可以方便的更换人脸检测和人脸识别算法，可以方便的修改系统的各个部分。｝

第4章 算法分析

4.1 Haar分类器

人脸检测算法在最初是不被重视的，研究者们在研究人脸识别时发现在复杂环境下人脸检测有很大的难度，因此很多研究者投入到人脸检测这个曾经不被他们重视的课题的研究中。

{54％：人脸检测的方法有很多，总体上可以分为利用先验知识的基于知识的人脸检测方法和利用统计的基于统计的方法。} {49％：后者是把人脸当成一个整体的模式，通过大量的人脸样本，使用统计的方法来构造人脸模式空间，利用相似度来判断是否存在人脸。} 由于两者方法都有自己的优势和缺点，所以目前很多方法是把基于知识和基于统计结合在一起使用的。

Haar分类器包含了Adaboot算法。 分类器属于数据挖掘的范畴，顾名思义分类器就是将样本分成若干个类，任何一个样本都会被分到一个特定的类别中。 {44％：在人脸检测的研究中，所有的人脸会被Haar分类器分成人脸和非人脸两种类别，分类的依据便是haar-like特征。}

Haar分类器实际上是由三部分完成的：

（1）Boosting方法的Adaboot分类器增强算法。

（2）Haar-like人脸分类特征

（3）提高计算速度的低分图方法

本项目使用的是OpenCV图形图像库。 在OpenCV中有对Haar分类器的封装，可以直接读取本地的分类器文件进行分类器的初始化，也就是说只需要有训练好的分类器文件，那么就可以方便的使用Haar分类器了。

4.1.1 Haar-Like特征

{44％：Haar-like特征是在人脸分类器中经常使用，用来表现人脸特征的。} Haar特征分为四种，如下：

（1）边缘特征

（2）线性特征

（3）中心特征

（4）对角线特征

特征模板将多种特征组合到一起。 {68％：如果将特征模板用黑，白两种颜色来区分，定义模板的特征值即为白色矩形像素减去黑色矩形像素和。} Haar特征值反映的是图像在灰度上的变化。 其在人脸上的表现为：眼睛比皮肤的颜色深； 嘴巴颜色比皮肤颜色深等等。

图4-1组成特征模板的四种特征

{52%：对于图中共有4中特征，其中A，B，D的特征数值计算公式都是是：} v=白色的和-黑色的和。 但是对于C来说，计算公式为： v=白色的和-2*黑色的和。 {43%：通过将黑色区域的面积乘二来使是矩形区域中的像素数量相同。}

可以在图像的任何位置放置一个矩形特征，矩形特征可以任意设置大小，可以自定义矩形模板。 如果将人脸划分成非常多的矩形小窗口，与自定义的模板特征进行匹配，那么匹配的结果可以很好的刻画出人脸的轮廓。

4.1.2 AdaBoost

{45%：AdaBoost并不是局限于某一种算法的，而是用来提升分类器的性能的。} AdaBoost可以帮助我们更好地选择矩阵特征的组合。

{58%：AdaBoost使用同一个训练集训练处若干个弱分类器，最后将这些弱分类器通过一定的方式组合成一个强分类器。}

Adaboost算法是通过迭代实现的，其中重要的一步是更改数据的分布。 {41%：其核心思想是根据样本在分类器中检测的结果来赋予样本权值，然后把划分权值的分类器传递给下层分类器进行训练，} {69%：将多个弱分类器组合在一起形成一个强分类器。} 以haar人脸分类器的训练为例：

{41%：（1）想要获取最初的弱分类器需要学习样本数为N的训练人脸样本。}

（2）使用测试人脸样本进行测试，将分错的人脸样本和其他新的训练人脸样本构成一个新的样本数为 N的训练人脸样本， 学习得到第二个弱分类器。

（3）将若干新的人脸训练样本和前两步测试出来分错的人脸样本结合到一起组成N个训练样本，学习得到第三个弱分类器。

{47%：（4）反复进行上面的步骤，最终可以学习得到若干个弱分类器，并且最终组合成一个强分类器。}

{43%：AdaBoost算法对分错的人脸样本进行了加权，也就是说如果某次训练时把人脸识别为非人脸或把非人脸识别为人脸，} 则会把训练的焦点放在了比较难区分的训练人脸样本上。 {60%：该算法还对弱分类器进行了加权处理，使得对人脸和非人脸分类效果较好的分类器具有比较高的权重，分类效果差的分类器具有较小的权重。} [9]

图4.2权重处理

{41%：因此，在人脸检测时遇到非人脸时，由于每一级弱分类器就会排除非人脸图像，因此排除非人脸图像的速度是很快的。}

4.1.3 积分图

积分图的定义是： 灰度图上的某一点到灰度图左上角那个点所围成的矩形中所有像素点的灰度的和。

图4-2 灰度图示意图

如图4-2中所示，根据定义可以得到，1点的灰度图是Ra区域所有像素点的灰度和， 2点的灰度图是Ra区域与Rb区域所有像素点的灰度和，3点的灰度图是Ra区域与Rc区域所有像素点的灰度和， R4区域是Ra， Rb， Rc，Rd四个区域的所有像素点的灰度和

所以就可以方便的得到Rd区域的积分图了。 用公式表示为： Sum（Rd）=I1+I4-（I2+I3）。

{52％：在Haar分类器中，分类是依据Haar-like特征，将积分图的方法来计算Haar-like特征可以极大的减少计算量。}

图4-3 haar-like边缘特征

{42％：要计算Haar-like分类特征，实际上就是计算上图中黑白两个区域中的像素之差。}

图4-4 haar-like边缘特征在某一小窗口中

假设某一haar-like边缘特征在某一小窗口中的位置如上图所示，那么要计算A区域和B区域的像素之差，只需要使用如下公式：

4.2 LBP算法

LBP算法，英文全称为Local Binary Patterns，中文名称为局部二值模式。 LBP算法通过局部特征来对图像进行判别。 使用LBP时，图像必须为灰度图。 {47％：每张灰度图都是由一个一个像素点组成，每个像素点都有自己的灰度。}

{49％：相比较于其他人脸识别算法，LBP算法的优势是对光照并不敏感。} 这在人脸识别时是非常重要的一点，在实际的识别过程中，识别的人脸可能处于不同的光照强度之下， 如果与样本的光照强度差别太大，并且识别算法对光照较敏感，则很有可能无法正确的识别出人脸。 这是LBP算法的一大优势。 LBP算法的另一个优势是计算速度快。 LBP算法的计算方法特别简单，可以在非常短的时间时间内完成计算，并且拥有较高的识别成功率。

LBP算法最初是定义在一个3×3的一个邻域内的，以中间的点作为阈值， {46％：找到与它相邻的8个像素，与中间像素的灰度值进行对比，如果大于中心点的灰度值，} 则记为1，如果小于中心点的灰度值，则记为0。这样在每个点对比过后，可以得到一个8位的2进制数，即这个邻域的LBP值。 这个值表现的是这个邻域的纹理信息[12]。

图4.5 LBP算法示意图

正式公式为：

其中 代表的是中间的点。 {54％：ic表示中间点的灰度值，ip表示邻域的点的灰度值。} S（x）是符号函数，有如下的定义：

最初的LBP算法存在着缺陷，它无法应对不同尺寸和不同频率纹理以及旋转的情况。 奥加拉对LBP算法进行了相应的改进。 他将3乘3的邻域拓展成任意大小的邻域，由于正方形的形状的局限性，它将领域改成了圆形。类似于下图：

图4-6 LBP算法圆形邻域

假设圆的半径为R，其中有P个采样点，其中每个采样点的值可以用下面的公式表示：

其中 为中心点， 为某个采样点，通过上式可以计算出每个采样点的坐标值，但是坐标并不一定是整数值，可以通过双线性插值计算得到采样点的像素：

在获取到人脸的LBP特征后还需要进行特征匹配，例如下面这张人脸图像，将其划分

图4-7 人脸的LBP划分

为7×7的子区域并计算得到每个邻域的LBP值并且统计其直方图。 在得到直方图后，下面两种方法都可以判断其相似性。 算法公式如下：

（1）直方图交叉核算法

（2）卡方统计方法

第5章 系统实现

5.1 系统总体结构

系统总体结构如图5.1

图5-1 总体结构图

本系统使用的Android开发IDE是Android Studio。 与Eclipse不同的是，Android Studio使用Gradle进行构建。

Project名字是FaceRecognizer，它代表了整个工作区。 faceRecognizer是Project中的一个Module。

Manifests表示配置，里面有个AndroidManifest.xml文件，用于对Android应用进行配置。

Java文件夹就是用来存放Java代码的。

jniLibs文件夹下放置的是不同平台下的动态链接库。

Res文件夹可以放置各种资源文件，例如图片，布局文件，动画等。

resources文件夹可以用来存放本地文件。 {45％：在项目中这个文件夹存放的是已经训练好的分类器文件。}

Gradle Scripts存放的是Gradle脚本，如下图所示：

图5-2 gradle脚本

其中build.gradle表示Project或者Module的构建脚本。 Proguard-rules.pro表示混淆规则。 gradle.properties表示gradle特性，可以对构建过程进行设置。 setting.gradle可以对Gradle的Module进行管理。 可以通过local.properties对SDK或NDK进行管理。

5.2 视频预览实现

视频预览的实现涉及到Android的SurfaceView类和SurfaceView类。

SurfaceView继承于View，可以直接从内存或者DMA等硬件接口取得图像数据。 它的特点是： 通常在Android中， UI的绘制只能在主线程中进行，但是在绘图频繁进行的情况下，主线程会被阻塞， 而 SurfaceView可以实现在主线程以外的线程进行图像的绘制，使用 SurfaceView可以防止 UI线程阻塞，可以用于摄像头预览。 使用SurfaceView需要继承SurfaceView，并且实现SurfaceView的Callback接口，如下图：

在构造函数中首先为SurfaceView的Holder添加一个实现了Callback的回调。 这个Callback有3个回调函数：

在这3个回调函数里对Camera进行操作。

Camera类是在Android中是用来对摄像头进行一些操作的。 首先需要通过Camera.open()来获取一个Camera实例，然后使用setPreviewDisplay方法来为Camera设置一个SurfaceView对象，将其作为预览容器。 使用Camera.Parameters可以对Camera的各个参数进行设置。 常见的参数有预览的大小和摄像头预览的方向。 想要获取到每帧图像的大小，只要为camera设置一个PreviewCallback回调。 关键代码如下：

至此，已经完成了预览的流程。 实现效果如下：

图5-3视频预览画面

5.3 人脸检测实现

在上一节中提到需要为Camera对象设置一个PreviewCallback回调。 创建了一个继承于View的FaceView来负责人脸信息的绘制。 FaceView实现了Camera的PreviewCallback，重写了onPreviewFrame方法。 OnPreviewFrame接收摄像头获取到的图像数据，每当获取到一帧，就回调一次。

{60％： FaceRecognizer类负责人脸检测和人脸识别。} 在FaceView的构造函数中，对FaceRecognizer进行了初始化。 {45％：人脸检测的初始化是加载分类器，分类器保存在resources文件夹中。} {42％：由于加载分类器是一个耗时操作，因此要放到一个新的线程中使用。} {46％：主要使用JavaCV的CVHaarClassifierCascade类，在初始化时只需要传入分类器文件的路径。} 关键代码如下：

由于加载分类器是一个耗时的操作，因此会新开一个线程进行加载，在完成加载后调用主线程创建的

Handler来发送一个消息给主线程， 表明加载完成，然后进行相应的 UI更新。

而当FaceView接收到预览图像一帧的回调时，将获取到的图像数据传递给FaceRecognizer对象。 关键代码如下：

在processImage方法中会先将获取到的图像转换成OpenCV中图片的数据结构，即IplImage。 直接调用IplImage的create方法即可。 在将其转换成IplImage之后再将其转换成灰度图。 具体实现代码如下：

进行人脸识别的方法为cvHaarDetectObject方法，传入待检测的图片，分类器等参数，最终可以获得识别到的人脸信息。 OpenCV中使用CVSeq数据结构进行存储。 使用CVSeq数组faces存储检测到的人脸。

在完成这个过程后，由于在onPreviewFrame中调用了postInvalidate方法进行view的重绘，因此会重新调用faceview的onDraw方法。 在onDraw方法中会从FaceRecognizer中取出检测到的人脸，根据人脸的信息来在屏幕上绘制矩阵。 只需要调用canvas的drawRect方法就可以方便的进行矩阵绘制。 {58％：根据检测到的人脸数量绘制相应数量的人脸矩阵。} 实现的关键代码如下：

至此，人脸检测的过程就已经完成了。

5.4 人脸检测实现

{42％：人脸识别之前先要对所有的样本图片进行训练，在此之前要先进行样本的采集。} 首先在人脸采集页面添加EditText，使用户可以输入姓名。 {60％：在姓名输入完成后，点击保存按钮会跳转拍照页面。} 这里利用了Android的Intent机制，通过给Intent设置Action： ACTION_IMAGE_CAPTURE，可以调用系统的拍摄页面，只需要在onActivityResult中处理拍摄完成的结果即可。

在获取到人脸图像后，还需要对图像进行裁剪，来使得人脸占图像比例尽量大。 裁剪也是利用Intent机制，给Intent设置Action： com.android.camera.action.CROP。 并且通过Intent的putExtra方法来添加裁剪参数。

最后需要将拍摄的图片保存到本地。 使用Android的File类可以完成。 首先通过 File的 createTempFile方法创建一个图片文件，文件名使用姓名加上当前的时间戳，再将这个图片文件传入跳转裁剪页面的 Intent对象中，Android系统会在完成裁剪后将裁剪好的图像根据传入的图片文件保存到本地。

5.4人脸识别流程

人脸识别部分使用的OpenCV的FaceRecognizer类。 {41％：在完成人脸检测部分的初始化后会进行FaceRecognizer对象的初始化。} {44％：由于要对所有的人脸样本进行训练，所以首先要获取所有人脸样本。} 通过Android的文件系统，首先找到人脸样本的目录，通过File对象的listFiles可以获得所有人脸样本。 在获取到人脸样本后先要将其转换成灰度图，方法与人脸检测部分图片转换成灰度图相同。 通过createLBPFaceRecognizer方法可以新建一个FaceRecognizer对象。 训练图片只需要调用FaceRecognizer的train方法。 train方法要求传入MatVector对象和int数组。 也就是OpenCV图片向量与其对应的标签。 通过一个循环即可完成人脸样本到MatVector对象与对应标签的转换。 关键代码如下：

{43％：在人脸检测完成之后，如果检测到人脸就会调用predictFace方法来进行人脸识别，并且传入待检测的人脸。} 人脸识别时会调用FaceRecognizer的predict方法，这个方法会在识别完成后返回一个label，表示识别出的人

脸在人脸数组中的位置。 通过遍历即可获得识别出的人的姓名。

在Faceview中，FaceRecognizer的ProcessImage方法回返回识别到的人的姓名。 在FaceView的onDraw方法中会判断是否有识别到的人脸，如果存在，则调用canvas的drawText方法把姓名显示到屏幕上。 效果图如下：

图5-4 人脸识别效果图

第6章 关键问题及解决方法

6.1 摄像头预览部分

在进行摄像头预览时在部分机型上未响应。 测试过后发现在部分Android6.0以上手机上未获取到摄像头权限。 Android M采用了不同于其他版本的动态权限管理，部分权限不会在安装时就获得，需要动态申请。 如果未获取到目标权限，相关操作不会有任何返回值。 需要将Target SDK设置为23或23以上，然后在代码判断是否获取到了权限，然后动态请求获取权限。

在进行摄像头预览时程序奔溃，提示setParameters failed。 在进行摄像头预览时需要设置预览时的宽度和高度。 这时需要先获取摄像头支持的宽度和高度，然后与用于预览的 SurfaceView的宽度和高度进行比较，当宽高比小于阈值并且高度与某一支持的高度最相近的时候， 就采用这个宽高作为预览时 Camera的输出宽高。 如果设置了Camera不支持的宽高，则程序运行时就会报错。

6.2 人脸识别部分

从主页面跳转到视频预览页面存在长时间的卡顿。 在初始化 FaceRecognizer的时候从本地读取分类器文件初始化 Haar分类器，从本地读取人脸样本并进行训练来初始化 OpenCV的 FaceRecognizer类， {41％：这个过程很耗时，并且在主线程中完成，导致主线程阻塞，无法更新 UI。} Android中耗时任务一般不在主线程中执行，以防止主线程被阻塞无法更新UI，因此，需要把耗时任务放到子线程中执行。 首先新建一个线程，把初始化任务放到新的线程中执行，在完成初始化工作后再通过主线程创建的 Handler来告诉主线程初始化工作完成， 进行相应的UI更新。

图像数据位深度不一致导致程序报错。 在OpenCV中存在多种图像位深度，使用分类器时需要相同位深度的图像深度。 通常从摄像头获取到的图像位深度都是8U的，因此，在进行cvCreateImage操作时统一使用8U这个深度。

第7章 总结与展望

随着移动互联网的发展和人们对新颖交互方式日趋强烈的需求，人脸识别在移动客户端上的应用愈来愈广泛。{45％：本文利用Android开发技术与OpenCV图形图像库完成了在Android设备上的人脸识别功能。} 现在对本文的工作进行一个总结：

本文首先对人脸识别的现状，发展历程以及生活场景中的应用进行了概括，然后总结对需要实现的系统进行了分析。 随后决定在Android客户端实现系统。 {43％：使用Haar分类器算法实现人脸检测部分，使用LBP算法实现人脸识别部分。} 并且使用JavaCV库来实现这两个算法。 {53％：最后对人脸识别系统具体的开发流程进行了介绍。}

由于时间和精力的以及研究深度上的限制，无法对算法进行深层次的探索，只是理解了算法的基本原理和库实现的方法。 在实现Android人脸识别系统后还有很大的改进空间：

1.没有使用运动跟踪算法。 系统对视频流中的每一帧进行人脸识别，从而实现了在视频中实时的人脸识别，但是这样效率较低，使用运动跟踪算法可以提高系统的效率。

2.人脸样本的获取方法局限于使用摄像头获取，样本数量较少，应当可以通过其他方式导入人脸样本，从而提高系统的易用性和识别成功率。

{41%：3.人脸识别成功率较低，直接使用 OpenCV中的人脸识别算法人脸识别识别错误率较高，} {52%：对获取到的人脸图片进行预处理，以提高识别成功率。}

4.可以对人脸样本库的人脸图片进行预览，从而方便用户进行比对，提高用户体验。

总而言之，人脸识别在Android端的应用会越来越多，为了提升用户体验，需要研究如何提高人脸识别的速度与成功率，以及如何简便的获取人脸样本。 只有沉下心来对算法进行深入的研究和实践，才能提高算法的效率。这些都是我今后要去完成的。

致谢

{42%：光阴似箭，四年大学时光即将结束，首先我要向母校武汉理工大学致以真诚的感谢。} 我在这里度过了美好又充实的4年。 {46%：论文的完成，离不开我的指导老师岑丽副教授的指点，从论文的选题，撰写，} 到完成，从系统的架构，实现，调试与测试岑老师都对我进行了精心的指导， {46%：提出了宝贵的意见，能够顺利完成毕业设计，我非常感谢岑丽老师。} 除此之外也要感谢身边的同学和老师对我的帮助以及鼓励，感谢他们的陪伴。

经过这一学期的努力，我的论文终于成功完成。 论文完成过程中，遇到问题时，岑老师都会抽空为我解答。{42%：岑老师学识渊博，治学态度严谨，工作作风优良，为人谦逊，非常值得我学习。} 此外，岑丽老师对我的生活也很关心，在生活上也为我提供了帮助，很平易近人。 {46%：能成为岑老师的一名学生是很幸运的一件事。}

{58%：我也要感谢计算机科学与技术学院的领导和老师们。} 作为软件工程的学生，我感受到了学院的老师们上课时认真的态度和解答问题的耐心，他们传授的知识是我能顺利完成论文的基础。 此外，学院丰富的教学资源能够帮助我更好的学习，接触最宝贵优秀的资源。 我还要感谢四年里一起学习生活的同学们，我们一起学习，一起生活，一起玩耍，他们在各方面都给我提供了帮助，一起讨论的过程中慢慢建立起了友谊，非常感谢他们。

{41%：我还要感谢我的父母，他们无私的奉献不求回报，为我提供他们能提供的最好的资源。} 在我心情低落的时候会鼓励我，在我遇到挫折的时候会给我温暖的拥抱，没有他们，就无法如此顺利地完成四年的学业。

即将毕业步入工作，只有努力工作，踏实学习才可以不辜负亲人，在社会中创造自己的价值，才能不辜负师长和朋友们对我的期望和谆谆教诲。

检测报告由PaperPass文献相似度检测系统生成