
分类号_____

密级_____

UDC ^{注1}_____



南京理工大学
NANJING UNIVERSITY OF SCIENCE & TECHNOLOGY

硕士专业学位论文

基于 Android 平台的服饰图像搜索 系统的设计与实现

李刚

指导教师姓名 _____

学 位 类 别 _____ 工程硕士

专 业 名 称 _____ 计算机技术

研 究 方 向 _____ 图像处理

论文提交时间 _____ 2015.12

声 明

本学位论文是我在导师的指导下取得的研究成果，尽我所知，在本学位论文中，除了加以标注和致谢的部分外，不包含其他人已经发表或公布过的研究成果，也不包含我为获得任何教育机构的学位或学历而使用过的材料。与我一同工作的同事对本学位论文做出的贡献均已在论文中作了明确的说明。

研究生签名：_____ 年 月 日

学位论文使用授权声明

南京理工大学有权保存本学位论文的电子和纸质文档，可以借阅或上网公布本学位论文的部分或全部内容，可以向有关部门或机构送交并授权其保存、借阅或上网公布本学位论文的部分或全部内容。对于保密论文，按保密的有关规定和程序处理。

研究生签名：_____ 年 月 日

摘 要

近年来,互联网的普及促进了电子商务的快速发展。人们在网络购物时,商品图像是购买与否的重要参考信息。然而目前商品图像并没有广泛的应用到商品搜索系统中,用户很难快速准确的找到自己需要的商品。与此同时,智能手机的普及和发展带动了互联网的移动化浪潮。手机购物成为新的趋势,移动终端的交易份额逐年增长。如何在移动终端上实现图像搜索成为电子商务平台商品搜索智能化的迫切需求。本文对此展开了研究,主要工作有:

分析和研究了基于内容图像检索的关键技术,尤其对项目中使用到的颜色特征、Uniform LBP 特征、SIFT 特征的相关理论和提取算法进行深入解析。

实现了一个基于 Android 平台的服饰图像搜索系统。其中,服务器端构建于开源 Python Web 框架 Django 之上,提供基于颜色特征搜索、基于 Uniform LBP 特征搜索、基于颜色特征和 Uniform LBP 特征的组合搜索以及基于 SIFT 特征搜索共四种搜索方式。颜色特征和 Uniform LBP 特征的组合方式是拼接,在本文中是把 72 维的颜色特征向量与 59 维的 Uniform LBP 特征向量拼接成 131 维的新的特征向量。前三种搜索方式的搜索流程为,计算查询图像与图像集中图像特征之间的欧式距离,使用 TOPK 算法对特征之间距离进行排序,距离最小的 K 幅图像即为相似度最高的 K 幅图像。基于 SIFT 特征的搜索中,首先利用视觉词袋模型离线生成视觉单词本,进而构造倒排索引结构,输入查询图像后,利用倒排索引结构对索引到的图像进行投票,得分最高的 K 幅图像即为相似度最高的 K 幅图像。本文在对图像进行特征提取之前,利用交互式图像分割方法 GrabCut 对图像进行前景提取,减少图像中无关元素对搜索的影响,提高搜索准确率。

本文同时实现了一个功能完善的 Android 客户端。用户通过拍照或从图库中选择得到一幅查询图像,经过裁剪上传至服务器,服务器搜索相似服饰并返回给客户端,客户端进行展示并支持用户收藏、保存、分享、购买这些服饰。

本文利用网络爬虫程序从电商网站上爬取服饰图像建立自己的图像集。最后,本文在自己的图像集上,对服务器的四种搜索方式进行了对比实验。实验结果表明使用颜色特征和 Uniform LBP 特征的组合特征进行搜索和使用 SIFT 特征进行搜索取得了比较好的效果。

关键词: Android, 基于内容的图像检索, LBP, SIFT, 特征提取

Abstract

In recent years, the popularity of the Internet has promoted the rapid development of Electronic Commerce. Product image information is the important consulting information for customers to shop online. However it is difficult for customers to find the products they need quickly and accurately, because product image is not widely used in commodity search system at present. At the same time, the popularity and development of smart phones driven by the wave of mobile internet has led to the tide of mobile Internet. The increase of mobile termination's trading volumes per year indicates that mobile shopping has become a new trend. How to realize image search on mobile terminal is urgent for Intelligent Product Search on E-commerce platform.

Detailed and in-depth studies are carried out on this issue. The primary work is as follows: Analysis and study of the content-based image retrieval of key technologies, especially used in the project to the color characteristics, Uniform LBP features, theories and SIFT feature extraction depth resolution.

It implemented a dress based on the Android platform image search system. Among them, the server is built on Django (an open-source Python Web framework) above, based color feature search feature searches based Uniform LBP combination of search based on color features and Uniform LBP feature and search based on SIFT features a total of four search methods. Uniform LBP features color features and combinations are spliced in this article is the color of 72-dimensional feature vector and Uniform LBP features 59-dimensional spliced feature vectors 141 into a new dimension. The first three search methods by the Euclidean distance between the feature vectors to measure the similarity between the images, feature extraction, and calculate the distance between the feature and use TOP K algorithm to the distance between the features you can get sort highest similarity former K images. Based on SIFT search mode, SIFT features quantified by visual word the bag model, the compressed image description. After the use of inverse document indexing technology to improve search efficiency. Before this paper, image feature extraction, use interactive image segmentation method GrabCut the image foreground extraction, reducing the impact of the image elements unrelated to the search.

Client-based Android platform, with improved functionality. Users take pictures or get a pair of the query image from the gallery selection, clipped uploaded to the server, search for similar clothing and returned to the client, the client display, users can view the information and complete the purchase of clothing in the client.

In this paper, the network's website crawlers crawl from the power take clothing image build your own image collection. Finally, in their own image collection server four search methods were compared experiments. Experimental results show that the use of color features and combinations of features Uniform LBP characteristics and use of SIFT features achieved relatively good results..

Key word:Android, CBIR, LBP, SIFT, Feature extraction

目录

摘 要.....	I
Abstract.....	II
1 绪论.....	1
1.1 课题的研究背景和意义	1
1.2 国内外研究现状.....	2
1.3 论文的主要工作	2
1.4 论文的组织结构.....	4
2 基于内容的图像检索技术	5
2.1 概述	5
2.2 颜色特征	6
2.3 纹理特征	7
2.4 SIFT 特征.....	9
2.5 GrabCut 算法	11
3 Android 平台相关技术.....	13
3.1 Android 平台的发展	13
3.2 Android 平台的组成.....	14
3.2.1 Linux 内核	14
3.2.2 系统运行库	14
3.2.3 应用程序框架.....	15
3.2.4 应用程序	15
3.3 Android 应用程序开发技术.....	15
3.3.1 Activity.....	15
3.3.2 Service.....	17
3.3.3 BroadcastReceiver.....	18
3.3.4 ContentProvider	19
4 基于 Android 平台的服饰图像搜索系统实现.....	20
4.1 系统架构	20
4.2 Android 客户端实现.....	21
4.2.1 逻辑架构.....	22
4.2.2 图像输入模块.....	24
4.2.3 网络模块.....	26
4.2.4 结果展示模块.....	28

4.2.5 其它辅助模块.....	31
4.3 图像集的建立.....	32
4.4 服务器端实现.....	34
4.4.1 Django 简介	36
4.4.2 GrabCut 模块	37
4.4.3 颜色特征提取模块.....	38
4.4.4 纹理特征提取模块.....	39
4.4.5 SIFT 特征提取模块.....	41
4.4.6 视觉词模块.....	41
4.4.7 倒排索引模块.....	42
4.4.8 排序模块.....	43
5 系统测试.....	44
5.1 测试环境.....	44
5.2 客户端测试.....	45
5.2.1 兼容性测试.....	45
5.2.2 性能测试.....	45
5.2.3 功能测试.....	46
5.3 服务器端测试.....	46
5.3.1 评价标准.....	47
5.3.2 实验结果.....	48
5.3.3 结果分析.....	53
6 总结及展望.....	54
6.1 论文总结.....	54
6.2 研究展望.....	54
致 谢.....	56
参考文献.....	57
附 录 A	60
附 录 B	61

1 绪论

传统的服饰搜索通常采用关键词搜索技术对服饰信息进行整合,忽略了对电子商务而言非常重要的图片信息^[1]。另一方面,用户并不总能对商品的关键词进行准确描述,而随着智能手机的普及,随时随地用手机拍摄的服饰照片则对实物具有更直观、准确的描述。因此,将基于内容的图像检索技术应用于商品检索领域具有广阔的应用前景和推广价值。

1.1 课题的研究背景和意义

随着互联网前所未有的发展,网上购物已经成为一种流行与时尚。网上购物者的数量在不断的增长。购物网站也相继涌现、发展和壮大。和实体店购物不同的是,网络购物者都是利用计算机和网络来浏览所需要的商品,由于缺乏现实购物过程中的视觉和触觉体验,网络购物者很难挑选到合适的商品。电子商务购物网站和一般网站相比具有其独特性,即为了直观呈现商品的信息,多使用图片作为信息载体。商品本身具有自己的特性,比如对时间的敏感性和对色彩的依赖性。大部分购物网站对于商品图片信息的检索都是基于标注的关键字来实现的^[2]。图像信息在数据库中的检索方式有两种,即全文索引和关键词索引^[3]。其基本步骤就是在网页信息自动采集和标引作为搜索引擎的重要组成部分的基础上,建立全文索引和关键词索引。全文索引就是用图像所在网页的全部文字信息作为图像的注释,网页上任何文字信息都认为与图像相关,显然这种图像标注查全率高但是查准率却是很低;基于关键词的索引使用若干关键词来表示图像信息,这些关键词一般都是从卖家对商品的图像的描述中获得,相比之下查准率高,但是查全率低。采用上述两种检索方式存在着以下问题:

(1) 商品销售方对商品信息分类方式的多样性。销售人员是网上信息的发布者,他们不可能按照有关分类法来进行信息的分类,而是根据自己的理解对图片文件加注标签,具有很强的主观性。因而必然会增加标引词的多样性,增加了检索的范围。

(2) 消费者对商品信息标签理解的歧义性。即由于个人认识的差异,不同的人对待相同的标签可能就会有不同的理解。

(3) 视觉依赖性强的商品类型划分不同。即对于这种视觉依赖性强的商品,不同的角度划分就会导致所属的类型不同。

(4) 商品信息标注的不完全性。即商品的文本标注不能全部反映图像的内容。

由于上述各种主观因素的存在导致基于文本的信息检索不能够很好的满足消费者所需商品的检索匹配。用户通过文本搜索检索不到自己心仪的商品,这在一定程度上将

会降低购物者的消费欲望。如果有一种检索方式可以避免图像文字标注的主观性和差异性所带来的匹配不精确问题,而直接客观地从商品的图像来检索匹配,不仅可以提高检索的查准率和查全率,还可以提高搜索的时间效率。于是基于内容的图像检索在电子商务中的应用成为一种迫切的需要与必然趋势。

1.2 国内外研究现状

基于内容的图像检索逐渐成为图像理解和计算机视觉领域的热门研究课题,国内外的研究结构已经投入了大量人力物力开展了对该课题的广泛研究,并且研制了一些商业系统和实验系统。比较知名的基于内容的图像检索系统有 IBM 公司的 QBIC 系统、由哥伦比亚大学研究开发的 VisualSEEK 和 WebSEEK 系统、由美国 Virage 公司开发的 Virage 系统、由美国 MIT 媒体实验室开发的 Photobook 系统等。

同时近年来,计算机视觉技术和电子商务得到了迅猛的发展。基于图像外部特征描述的商品检索方式已经不能很好的满足用户的需求,因此,国内外的研究机构开始将基于内容的图像检索技术引入到了商品的检索和浏览中,其中比较有代表性的图像搜索引擎如 QBIC、谷歌图像搜索以及移动端图像检索系统 Google Goggles。

(1) QBIC

QBIC 系统是由 IBM Almaden 研究中心开发的第一个商品化的基于内容的图像检索系统,它的系统框架、结构和技术对后来的图像检索系统有着深远的影响。QBIC 系统支持基于例子图像、手绘略图、选择的颜色、纹理等查询,不仅支持图像检索,还支持视频、文本和语音等多种形式的信息检索。QBIC 系统是少数几个考虑高维特征索引的系统。QBIC 系统使用的颜色特征是颜色直方图。纹理特征采用粗糙度、对比度和方向性描述。形状特征包括面积、离心率、主轴方向和不变矩。颜色、纹理和形状均采用加权的欧氏距离比较。

(2) 谷歌图像搜索系统

2011 年 6 月份谷歌发布了以图搜图的图片搜索服务,谷歌以其强大的搜索功能而著称,以图搜图的搜索技术也相对比较成熟。谷歌图像搜索融合了图像分析、模式识别、人机交互等多种技术,提供关键字、视觉特征相结合的检索模式,提高了查询的准确性,提升了用户体验。其操作流程比较简单,用户通过选择查询图像的相关特征如颜色、纹理等进行检索。Google 图像检索具有理想的检索效率,谷歌图像检索系统结合了图像特征和关键字对查询图像进行分析进而匹配出最相似的图片集合,提高了图像检索的准确率,优化搜索结果,改善了用户体验。

(3) Google Goggles

Google Goggles 是谷歌推出的一款移动端的图像搜索工具,利用手机摄像头拍摄当

地地标建筑、书籍封面、艺术作品、酒类标签以及产品商标等物体的照片后，软件就将会自动在 Google 上搜索相关信息，并予以识别显示。不过 Google Goggles 对查询图像的类别有一些限制，不能检索家具、服装之类的东西。

1.3 论文的主要工作

本论文围绕着基于内容的图像检索系统，进行了如下工作：

深入学习了基于内容的图像检索的流程以及关键技术，尤其对本文项目中将要使用的颜色特征、纹理特征、SIFT 特征、视觉词袋模型、倒排索引技术以及 GrabCut 交互式图像分割的理论和算法进行了深入的研究。同时，对 Android 应用程序开发的关键技术进行了认真学习，加深了对 Android 系统构成的理解，熟悉了 Android 四大组件的功能和使用场景。

设计和实现了一个基于 Android 平台的服饰图像搜索系统。服务器搭建在开源 Python Web 框架 Django 之上。系统图像搜索方式有四种，分别是基于颜色特征进行搜索、基于 Uniform LBP 特征进行搜索、基于颜色特征和 Uniform LBP 特征的组合（将颜色特征与 Uniform LBP 特征直方图向量进行拼接）进行搜索以及基于 SIFT 特征进行搜索。其中，前三种搜索方式通过特征之间欧式距离来度量相似度，然后使用 TOPK 算法对特征之间距离进行排序，得到特征距离最小的 K 幅图像，也就是相似度最高的 K 幅图像。由于一副图像包含较多的 SIFT 特征点并且每个 SIFT 特征均为 128 维，本文对 SIFT 特征使用视觉词袋模型进行量化，压缩图像描述，提高检索效率。在此基础上，基于 SIFT 特征的搜索利用倒排索引加快搜索速度。考虑到服饰图像中人脸以及背景会对搜索效率和搜索准确度产生影响，本文中使用 GrabCut 算法对图像进行无关元素过滤，提取感兴趣的服饰部分用于特征提取和匹配。本文还建立了自己的实验图像集。图像集中的服饰图像使用网络爬虫从电商平台爬取得到，一并爬取的还有服饰的品牌、价格等信息。

本文同时实现了一个功能完善的 Android 客户端。用户可以通过客户端的拍照或者图像选择功能选取一张查询图像，裁剪后上传至服务器，服务器进行搜索返回相似服饰列表，客户端对相似服饰进行展示。用户可以对服饰进行分享、收藏、保存和查看品牌价格信息等操作。同时，用户也可以在 Android 客户端内部实现购买服饰操作。除此之外，本文客户端还包含了用户系统以增加用户粘性，提高客户端的使用率。一个完整的客户端应该包含的版本升级、用户反馈等辅助功能本文 Android 客户端也都具备。

最后，本文对实现的基于 Android 平台的服饰图像搜索系统进行了测试。经测试，Android 客户端兼容性良好，性能良好，功能正常。服务器测试中，采用前 N 个结果 K 次平均正确率来评价搜索性能。结果表明，基于颜色特征搜索的平均准确率在 0.37 到

0.54 之间, 基于 Uniform LBP 特征搜索的平均正确率在 0.33 到 0.45 之间, 基于颜色特征和 Uniform LBP 特征的组合搜索的平均正确率在 0.37 到 0.56 之间, 基于 SIFT 特征搜索的平均正确率在 0.31 到 0.58 之间。实验结果表明, 就本文的图像集和评价标准而言, 颜色特征和 Uniform LBP 特征的组合以及 SIFT 特征取得了较好的搜索效果。

1.4 论文的组织结构

本文共划分为六个章节, 每章的主要内容如下所示。

第一章, 绪论。主要介绍了基于内容的图像检索研究的背景与意义, 说明了当前国内外对基于内容图像检索的研究实践现状, 并给出了本论文的主要工作以及论文组织结构。

第二章, 基于内容的图像检索技术。本章主要介绍了基于内容的图像检索原理, 着重解析了颜色特征、Uniform LBP 特征、SIFT 特征这三种特征的相关理论和提取算法。并简要介绍了 GrabCut 算法的原理。

第三章, Android 平台相关技术。本章首先介绍了 Android 的发展历程和发展现状, 然后阐述了 Android 平台的组成, 最后解析了 Android 应用开发的关键技术。

第四章, 基于 Android 平台的服饰图像搜索系统实现。详细介绍了 Android 客户端、服饰图像集、服务器端的实现过程。详细阐述了客户端图像输入模块、网络模块、结果展示模块以及其它辅助模块的实现效果与实现方法。介绍了图像集的建立过程以及规模大小。最后对服务器的工作流程进行了介绍, 着重介绍了 GrabCut 模块、颜色特征提取模块、Uniform LBP 特征提取模块、SIFT 特征提取模块、视觉词模块、倒排索引模块以及排序模块的实现。

第五章, 实验结果与讨论。对实现的图像检索系统进行测试, 分析搜索效果。

第六章, 研究总结及展望。在这一章将总结前面的研究及实践成果, 讨论并展望下一步研究方向。

2 基于内容的图像检索技术

为了实现基于 Android 平台的服饰图像搜索系统,本文首先深入研究和分析了基于内容的图像检索关键技术。本章首先对基于内容的图像检索基本原理进行概述,然后分别介绍了介绍本文系统中使用到的颜色特征、纹理特征、SIFT 特征的特性以及提取方法,最后介绍了可以提高搜索准确度的交互式图像分割 GrabCut 算法。

2.1 概述

随着互联网的发展和智能手机的普及,网络图片的数量飞速增长。原本基于文本的图像检索技术的弊端日益显现^[9]。基于文本的图像检索技术依赖于关键词标注。而一张图片的信息几个关键词无法完全涵盖。并且人工标注关键词费时费力,自动标注关键词准确性差。为了解决基于文本的图像检索中的诸多缺陷,人们想到了利用图像的内容进行检索。基于内容的图像检索(CBIR)应运而生。基于内容的图像检索是指提取图像特征,以图像特征的差异大小来衡量相似度,并给出相似度最高的图像结果^[7]。与基于文本的图像检索不同,基于内容的图像检索摒弃了关键词标注,而是直接从图片中提取特征,检索过程由关键词的比较变为特征的比较。与基于文本的图像检索技术相比,基于内容的图像检索技术具有三个主要优势,分别是:一,搜索基于图像的内容本身,而不是标注的关键词;二,基于内容的图像检索是一种近似匹配,检索范围大于基于文本的图像检索技术;三,搜索过程自动化,省去标注关键词的工作,同时也避免主观差异带来的标注误差。在基于内容的图像检索中,图像的内容由图像的特征描述符表示。图像的特征主要分为低层视觉特征和高层语义特征两大类别。目前,受到计算机视觉、心理学、生物学等学科发展水平的制约,基于高层语义特征的图像检索技术还很不成熟。因此,目前基于内容的图像检索技术的研究主要围绕低层视觉特征展开。

基于内容的图像检索系统主要流程如下:首先将图像集中的图像的特征提取出来,构成特征池。查询时,提取查询图像的特征并与特征池中的特征一一进行相似度的测量,然后对相似度进行排序,得到相似度最高的图像作为结果^[8]。

基于内容的图像检索的核心问题是选择特征描述符。选择的特征描述符要能很好的区分不同的图像。根据选择的特征描述符的不同,可以将基于内容的图像检索系统分为基于颜色特征的检索系统,基于纹理特征的检索系统以及基于形状特征的检索系统等等。使用单一特征很难以达到理想的检索效果,很多基于内容的图像检索系统利用若干特征的组合来描述图像。

2.2 颜色特征

颜色是一副图像非常重要的特征。我们在记忆和分辨图像时，颜色都起到了非常重要的作用。并且，颜色特征具有旋转不变性、尺度不变性，而且相比其它特征容易获取。因此，颜色特征在基于内容的图像检索技术中被广泛的采用^[11]。颜色通过颜色空间表示，常用的颜色空间有 RGB 颜色空间和 HSV 颜色空间。

RGB 颜色空间是通过红绿蓝三原色来描述颜色的颜色空间，是最基本、最常用的颜色空间^[12]。RGB 颜色空间是面向硬件的颜色空间。图像在计算机中存储，一般采用 RGB 颜色空间来表示。然而，RGB 颜色空间的三个分量受亮度影响太大，亮度的改变将使三个分量随之改变。所以，RGB 颜色空间不适合于基于内容的图像检索系统。

HSV 颜色空间是通过色调（H）、饱和度（S）、量度（V）3 个分量来描述颜色的颜色空间，是一种面向视觉感知的颜色空间^[14]。其中，色调是指图像的主色，如黄、绿、青代表不同的色调，取值范围为 0° 到 360° ；饱和度是指颜色的深浅程度，与颜色的纯度有关，取值范围为 0 到 1；亮度是指颜色的明暗程度，取值范围为 0 到 1。HSV 颜色模型与人眼的视觉特征比较接近，所以 HSV 颜色空间在图像颜色特征提取与分析中应用广泛。

基于以上理论，本文选取的颜色特征使用 HSV 颜色空间表示。而计算机中位图（Bitmap）颜色使用 RGB 颜色空间表示。所以，在提取颜色特征之前，我们需要把图像颜色从 RGB 颜色空间转换到 HSV 空间。转换公式如下式 2.1、式 2.2、式 2.3 所示。

$$V = \frac{1}{\sqrt{3}}[R+G+B] \quad (2.1)$$

$$S = 1 - \frac{1}{V} \min(R, G, B) \quad (2.2)$$

$$H = \begin{cases} \theta, & G \geq B \\ 2\pi - \theta, & G < B \end{cases} \quad (2.3)$$

$$\text{其中, } \theta = \cos^{-1} \left[\frac{\frac{1}{2}[(R-G)+(R-B)]}{\sqrt{(R-G)^2 + (R-B)(G-B)}} \right], H \in [0, 360^\circ], S \in [0, 1], V \in [0, 1]。$$

考虑到检索速度，我们需要对颜色进行量化，以压缩图像的颜色描述，提高搜索效率。本文对颜色的量化过程如下所示：

（1）按照式 2.4、式 2.5、式 2.6 所示，分别把色调（H）从 $[0^\circ, 360^\circ]$ 映射到 $\{0, 1, 2, 3, 4, 5, 6, 7\}$ ，饱和度（S）从 $[0, 1]$ 映射到 $\{0, 1, 2\}$ ，亮度（V）从 $[0, 1]$ 映射到 $\{0, 1, 2\}$ 。

$$H = \begin{cases} 0, & H \in [316, 20] \\ 1, & H \in [21, 40] \\ 2, & H \in [41, 75] \\ 3, & H \in [76, 155] \\ 4, & H \in [156, 190] \\ 5, & H \in [191, 270] \\ 6, & H \in [271, 295] \\ 7, & H \in [296, 315] \end{cases} \quad (2.4)$$

$$S = \begin{cases} 0, & S \in [0, 0.2] \\ 1, & S \in [0.2, 0.7] \\ 2, & S \in [0.7, 1] \end{cases} \quad (2.5)$$

$$V = \begin{cases} 0, & V \in [0, 0.2] \\ 1, & V \in [0.2, 0.7] \\ 2, & V \in [0.7, 1] \end{cases} \quad (2.6)$$

(2) 构造一维特征矢量。

$$W = HQ_SQ_V + SQ_V + V \quad (2.7)$$

其中, Q_S 和 Q_V 分别是 S、V 的量化级数, $Q_S = 3$, $Q_V = 3$ 。

式 2.7 实际上为:

$$W = 9H + 3S + V \quad (2.8)$$

W 的取值范围是 [0,1,...71]。至此, 每个像素的颜色被量化为一维向量 W。统计图像中像素的颜色分布, 获得颜色分布直方图。我们使用此颜色直方图描述图像的颜色, 颜色特征为一个 72 维向量。

2.3 纹理特征

纹理是图像的一项重要特征, 对服饰图像尤其重要。纹理特征主要刻画了一块区域像素灰度的空间分布, 宏观上表现为凹凸不平的沟纹或者光滑表面上的花纹^[16]。

局部二元模式 (local binary pattern, 简称 LBP) 是一种应用非常广泛的局部纹理描述算子, 具有计算简单、对光照变化不敏感等优势^[17]。基本 LBP 算子计算过程为: 将中心像素点 3*3 邻域内的 8 个像素点的灰度值分别与中心像素点的灰度值比较大小, 根据大小关系对 8 个相邻像素点进行二值化。即如果相邻像素点大于中心像素点的灰度值, 则将其置为 1, 否则置为 0。然后对邻域的值进行加权求和, 即可得到该像素点的 LBP 特征值。

基本 LBP 算子计算简单,但是由于窗口固定,无法提取大尺度纹理特征。为了提高 LBP 算子的有效性和完整性, Ojala 等人对基本 LBP 算子进行了改进, LBP 算子的计算不在局限于 3×3 窗口内的相邻的 8 个像素点,而是给定一个采样半径 R 和采样点数 P ,在中心像素点为圆心、 R 为半径的圆周上等间隔的采样 P 个点。常用的 P 、 R 取值组合有 $P=8, R=1.0$ 、 $P=12, R=2.5$ 、 $P=16, R=4.0$, 如图 2.1 所示。

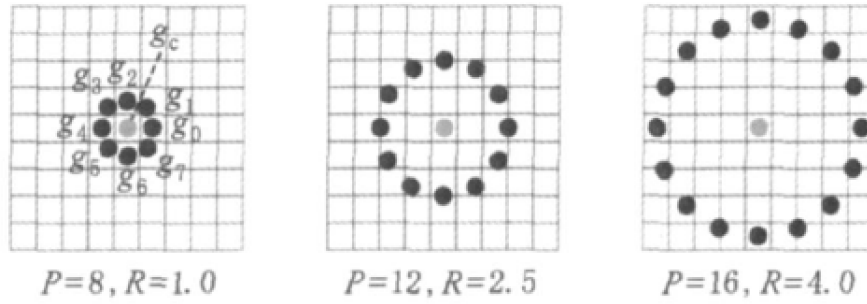


图 2.1 常用的 P 、 R 取值组合示例

现在介绍 $P=8, R=1.0$ 时改进 LBP 算子的计算过程,其它取值组合依次类推。设中心像素点的灰度值为 g_c , 相邻的 8 个像素点的灰度值依次为 g_0, g_1, \dots, g_7 , 那么该中心像素点的 LBP 算子可表示如下:

$$LBP_{8,1.0} = LBP_{P=8,R=1.0} = \sum_{i=0}^7 S(g_i - g_c) 2^{i-1} \quad (2.9)$$

$$s(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \quad (2.10)$$

$LBP_{8,1.0}$ 有 2^8 种不同的取值,如果我们采用 $LBP_{8,1.0}$ 特征直方图来表示一副图像的话,需要统计 2^8 种类别,这个向量将是 2^8 bin 的,不便于计算。Ojala 等人经过对大量纹理图像进行研究之后发现,如果将 LBP 算子二进制值首尾相连组成环,绝大多数的环至多存在 2 次 0 和 1 之间的跳变^[21]。以 $LBP_{8,1.0}$ 为例,00000000、01110000、01010000 分别包含 0、2、4 次 0 和 1 之间的跳变。基于这样的统计规律, Ojala 等人提出了均匀模式的 LBP 算子,即将至多存在 2 次 0 和 1 之间的跳变的 LBP 算子定义为均匀模式,分别归类。将存在 2 次以上 0 和 1 之间的跳变的 LBP 算子定义为混合模式,归为一类。压缩之后, LBP 算子的种类可由原来的 2^P 减少为 $P(P-1)+3$ 。

本文纹理特征的提取使用均匀模式的 $LBP_{8,1.0}$,将原来的 2^8 种 LBP 特征压缩为 $8 \times (8-1) + 3 = 59$ 种 LBP 特征。然后将均匀模式 LBP 特征映射为 0 到 57 之间的序号,将非均匀模式 LBP 特征映射为 58,即所有的 LBP 特征最终都用范围在 0 到 58 的一维向量表示。统计图像用一维向量表示了的 LBP 分布,得到分布直方图,图像的 LBP 特征最终为一个 59 维向量。

2.4 SIFT 特征

SIFT(Scale-Invariant Feature Transform)描述子由 Lowe 于 1999 年提出, 2004 年总结完善^[24]。SIFT 对旋转、尺度缩放、亮度变化具有不变性, 对视角变化、仿射变换、噪声的容忍度较高^[25]。SIFT 特征提取主要分为 4 个步骤:

(1) 构建高斯差分尺度空间 (DOG scale-space)

尺度空间理论的出现是为了模拟图像数据的多尺度特征。Koenderink 与 Lindeberg 证明了高斯卷积核是唯一的线性尺度核^[22]。一副图像的尺度空间被定义为:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \quad (2.11)$$

其中 $G(x, y, \sigma)$ 为尺度可变高斯函数:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \quad (2.12)$$

式中 (x, y) 为尺度坐标; σ 为尺度因子, 决定图像的平滑程度, 大尺度对应图像概貌特征 (低分辨率), 小尺度对应图像细节特征 (高分辨率)。高斯差分尺度空间利用不同尺度高斯差分核与图像卷积生成。 $D(x, y, \sigma)$ 是两个相邻尺度图像的差。

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) = L(x, y, k\sigma) - L(x, y, \sigma) \quad (2.13)$$

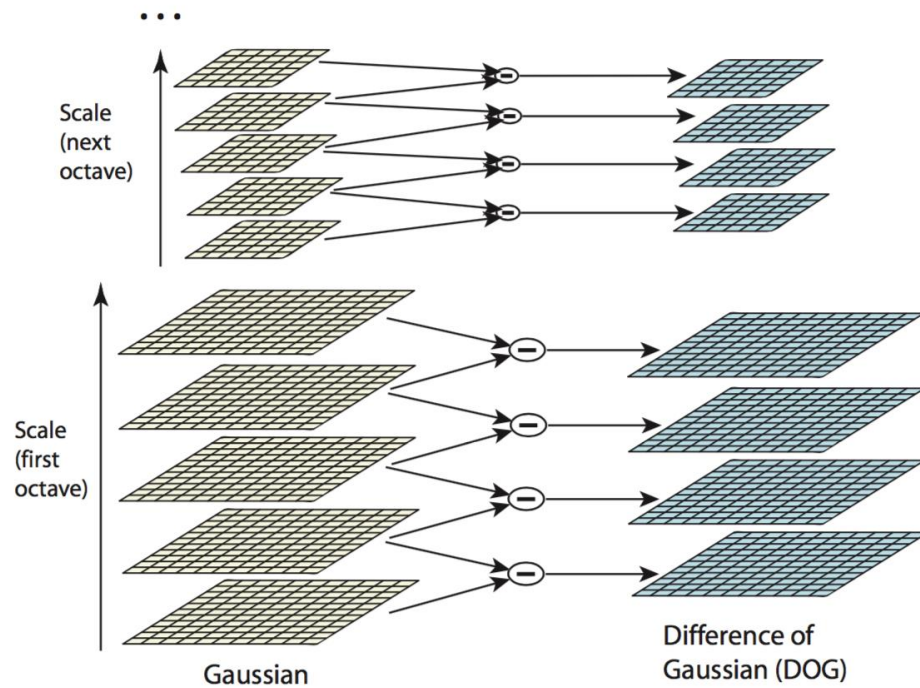


图 2.2 构建高斯差分尺度空间 (DOG scale-space)

(2) 关键点定位

如果一个像素在 DOG 尺度空间本层以及上下两层的领域中具有最大值或最小值时, 则认为该像素是图像在该尺度下的一个关键点。如图 2.3 所示。

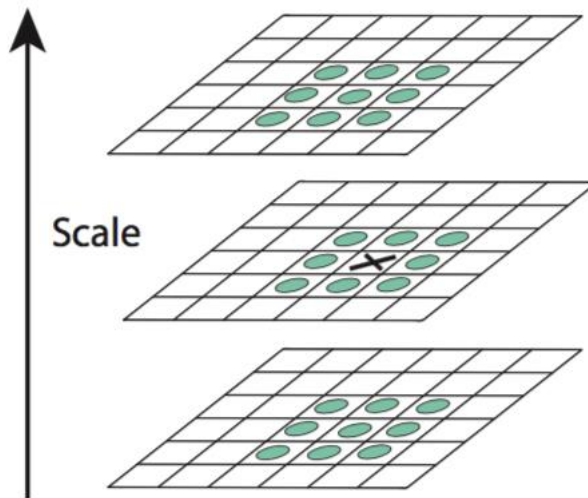


图 2.3 DoG 尺度空间关键点定位

中间的检测点和它同尺度的 8 个相邻点和上下相邻尺度对应的 9×2 个相邻点共 26 个相邻点进行比较，以确保在尺度空间和二维图像空间都检测到极值点。

(3) 关键点方向确定

基于关键点邻域像素的梯度分布特性，分配给每个关键点一个或多个方向参数。后续的对图像的操作均相对于关键点的方向、尺度和位置进行变换，从而提供对于这些变换的不变性。

$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2} \quad (2.14)$$

$$\theta(x, y) = \arctan\{[L(x, y+1) - L(x, y-1)] / [L(x+1, y) - L(x-1, y)]\} \quad (2.15)$$

式 2.14、式 2.15 分别为 (x, y) 处梯度的模值和方向公式。其中 L 所用的尺度为每个关键点各自所在的尺度。

在实际计算时，我们在以关键点为中心的邻域窗口内采样，并用直方图统计邻域像素的梯度方向。梯度直方图的范围是 $0 \sim 360$ 度，其中每 45 度一个柱，总共 8 个柱，或者每 10 度一个柱，总共 36 个柱。Lowe 在论文中建议使用高斯函数对直方图进行平滑，减少突变的影响。直方图的峰值代表该关键点处邻域梯度的主方向，用作该关键点的方向。图 2.4 是采用 7 个柱来统计邻域像素梯度方向并确定主方向的示例。

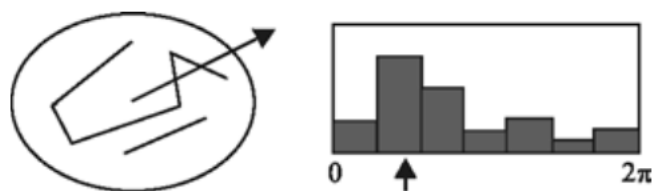


图 2.4 确定关键点方向

(4) 关键点描述子的生成

首先将坐标轴旋转为关键点方向，以确保旋转不变性。以关键点为中心取 8×8 的窗口。

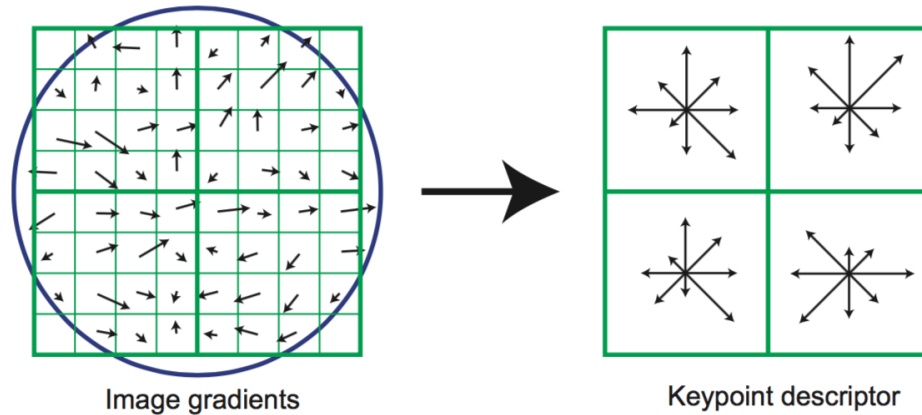


图 2.5 关键点描述子的生成

图 2.5 左部分的中央为当前关键点的位置，每个小格代表关键点邻域所在尺度空间的一个像素，利用式 2.14 求得每个像素的梯度幅值 $m_{i,j}$ ，利用式 2.15 求得每个像素的梯度方向 $\theta_{i,j}$ ，箭头方向代表该像素的梯度方向，箭头长度代表该像素的梯度模值，然后用高斯窗口对其进行加权运算。图中圈代表高斯加权的范围（越靠近关键点的像素梯度方向贡献越大）。然后在每 4×4 的小块上计算 8 个方向的梯度方向直方图，绘制每个梯度方向的累加值，即可形成一个种子点，如图 2.5 右面部分所示。图 2.5 中一个关键点由 2×2 共 4 个种子点组成，每个种子点有 8 个方向信息。这种邻域方向性信息联合的思想增强了算法抗噪声的能力，同时对于含有定位误差的特征匹配也提供了较好的容错性。

计算关键点周围的 16×16 的窗口中每一个像素的梯度，并使用高斯下降函数降低远离中心的权重。这样就可以对每个关键点形成一个 $4 \times 4 \times 8 = 128$ 维的描述子。将这一向量归一化之后，可进一步去除了光照的影响。

2.5 GrabCut 算法

图像分割是图像工程中目标检测、特征提取和参数测量的基础，是图像分析、模式识别、计算机视觉领域的关键问题之一，它使得高层次的图像理解成为可能^[30]。利用基于能量最小化框架的图割理论进行图像分割已成为近年来一个研究热点。它的优势包括它的全局最优求解能力以及结合了多种知识的统一图像分割框架。在此基础上，针对不同应用场景，人们提出了多种变种分割方法^[23]。

在 Interactive Graph Cuts 算法基础上，Rother 等人提出了 GrabCut 算法^[24]。改进的内容包括：（1）采用高斯混合模型（Gaussian Mixture Model, GMM）替代灰度直方图，支持彩色图像分割；（2）在 GMM 参数估计过程中，采用多次迭代算法替代一次最小估

计；(3) 算法采用非完全标号 (incomplete labelling) 的方式，降低了用户的交互工作量。

图像分割即把像素标为前景/背景，是典型的二元标号问题。首先构造一个能量函数，用于计算像素的标号值。之后借助网络流理论，把标号问题转化为最大流/最小割问题解决。

设 $G=(V,E)$ 为一无向图， c 是定义在边集 E 上的容量函数： $c:E \rightarrow R^+$ ，则无向图 G 及其边集 E 上的容量函数 c 构成一个 $s-t$ 网络，记作 $N=(G,s,t,c)$ ，其中， s 是网络的源点， t 是网络的汇点。借助最小化能量函数 $E(f)$ 把顶点集 V 划分为两个顶点集 S, T ，分别与源点 s 和汇点 t 相连，($s \in T, t \in T$)。

$$E(f) = E_{data}(f) + E_{smooth}(f) = \sum_{p \in V} D_p(f_p) + \sum_{(p,q) \in V} D_{p,q}(f_p, f_q) \quad (2.18)$$

其中， f 为 V 的一个标号， $f:P \rightarrow L(L=\{0,1\})$ ； $E_{data}(f)$ 为数据项，用来衡量和所观察到的数据的不一致性； $E_{smooth}(f)$ 为光滑项，用来衡量非分片光滑的程度； N 为相互作用的相邻顶点对。

网络中的顶点对应图像中的像素，网络边上的容量对应像素特征之间的差异或者相似度，即用 $s-t$ 网络表示一个图像。图像分割能量函数的最小值对应网络的最小割。根据最大流/最小割定理，网络的最大流与最小割是等价的。最终，能量函数的最小化问题转化为网络的最大流问题。

GrabCut 算法将图像分割问题定义为：对图像的每个像素点 $Z=(z_1, \dots, z_n, \dots, z_N)$ ，求其标号值 $\alpha, \alpha=(\alpha_1, \dots, \alpha_n, \dots, \alpha_N), \alpha_n \in \{0,1\}$ ，其中，1 代表前景；0 代表背景。 θ 代表前景/背景的概率密度模型。借助图割理论，图像分割问题可表示为

$$\alpha = \arg \min_{\alpha} E(\alpha, \theta) \quad (2.19)$$

GrabCut 算法基本步骤包括：首先，由用户在前景周围画一个矩形，通过“非完全标号”的方式来标定图像的背景区域 T_b (Trimap Background) 和未知区域 T_u (Trimap Unknown)，矩形框外为背景区域，矩形框内为未知区域。将未知区域内的像素标注为 1，背景区域内的像素标注为 0，目标区域 T_f (Trimap Foreground) 设为空。利用用户标定的背景区域和未知区域分别初始化前景/背景 GMM。然后将未知区域 T_u 划分为前景/背景两类，对新划分的前景/背景像素进行切割，更新 GMM。在新的 GMM 参数下继续对未知区域 T_u 进行划分，迭代直至满足收敛条件，最终确定 GMM 参数。利用最终的 GMM 对未知区域进行一次切割，最终得到前景图像。

3 Android 平台相关技术

2008 年 7 月 23 日, Android 1.0 正式发布, Android 1.0 是 Android 的一个商业版本。2015 年 10 月 5 日, Android 6.0 正式发布。目前, Android 已发展出 Android Wear、Android TV 和 Android Auto 等分支平台。其中, Android Wear 专为智能手表等可穿戴设备设计; Android TV 专为智能电视设计; Android Auto 专为汽车设计。Android 系统一直处于快速发展之中。2013 年 7 月 3 日, Google 宣布, 全世界激活的 Android 设备已经超过 10 亿台。2015 年 7 月, Google 宣布, 全世界访问网络的智能手机和平板中 59.1% 搭载了 Android 系统, 中国市场的比例为 68.3%, 美国市场的比例是 40.78%。

3.1 Android 平台的发展

安迪·鲁宾 (Andy Rubin) 被称为“Android 之父”。他于 2003 年创立了 Android 科技公司 (Android Inc.)。2005 年 7 月 11 日, Android 科技公司被 Google 收购, 成为 Google 的全资子公司。安迪·鲁宾加入 Google, 继续基于 Linux 内核的 Android 系统的研发。2007 年 11 月 5 日, Google 牵头成立了开放手持设备联盟 (OHA, Open Handset Alliance), 最初的成员有 Broadcom、HTC、Intel、LG、Marvell、Samsung (三星) 等。开放手持设备联盟成立的同一天也推出了第一部搭载 Android 系统的智能手机。小米、华为、索尼和 ARM 等也于 2008 年 12 月 9 日加入 OHA。2007 年, AOSP (Android Open Source Project) 项目启动, 它是 Android 源码开发更新项目。众所周知, Android 系统是开源的, 代码公开免费。其中, Android 的大部分源码以 Apache 开源条款 2.0 发布, 而 Linux 内核部分继承 GPLv2 开源。Android 发展至今, 比较重要的版本包括: Cupcake (1.5)、Donut (1.6)、Eclair (2.0–2.1)、Froyo (2.2–2.2.3)、Gingerbread (2.3–2.3.7)、Honeycomb (3.0–3.2.6)、Ice Cream Sandwich (4.0–4.0.4)、Jelly Bean (4.1–4.3.1)、KitKat (4.4–4.4.4, 4.4W–4.4W.2)、Lollipop (5.0–5.1.1)、Marshmallow (6.0–6.0.1)。

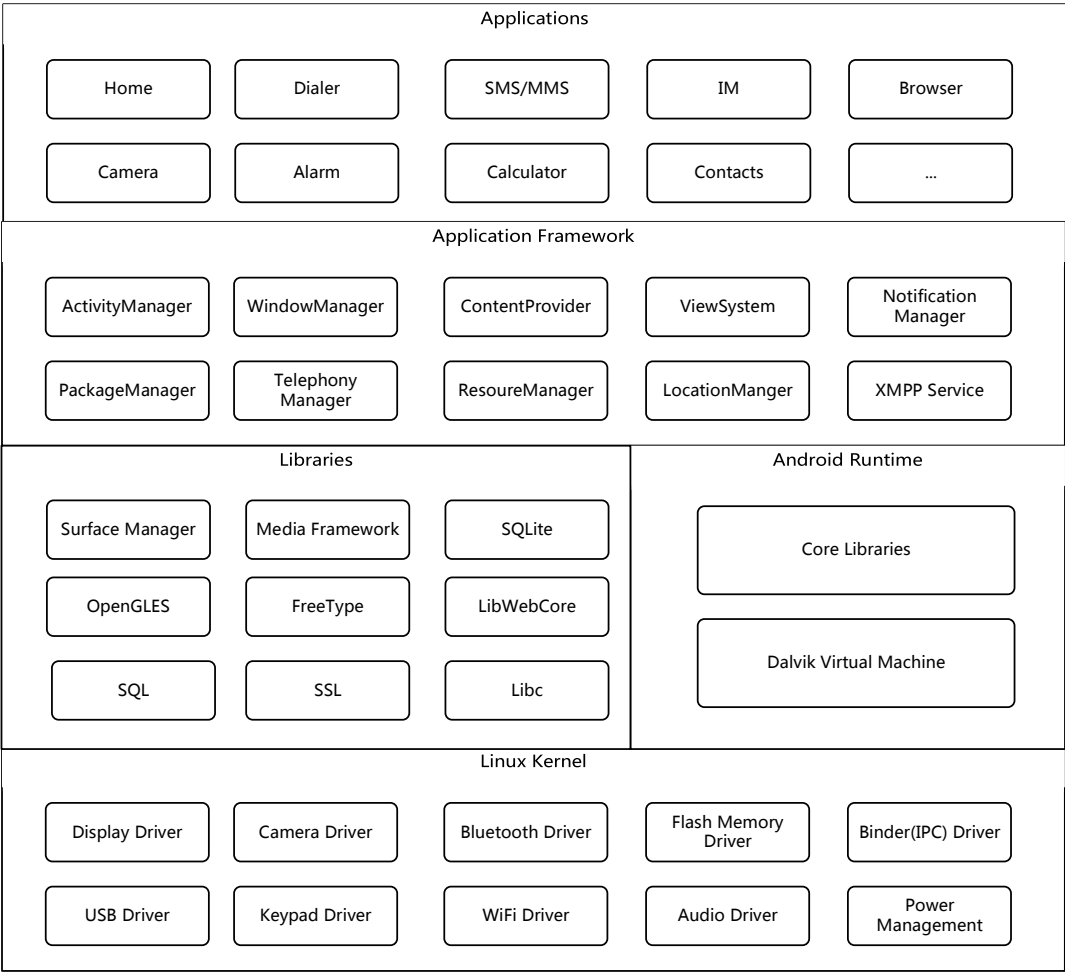


图 3.1 Android 系统架构

3.2 Android 平台的组成

如图 3.1 所示，Android 系统大致可以分为五层，由上到下依次为应用程序层、应用程序框架层、函数库层、Android 运行层和 Linux 内核层。各层各司其职，上层使用下层提供的服务，屏蔽本层及下层的差异，为上层提供统一的服务。分层结构具有高内聚、低耦合的优点。

3.2.1 Linux 内核

Linux 核心层是 Android 的最底层，也是 Android 的基础。Andriod 依赖 Linux2.6 内核提供包括进程管理、内存管理、网络、硬件驱动等核心系统服务。同时，Linux 核心层也是硬件和软件之间的抽象层，它屏蔽硬件差异，为上层提供统一服务。

3.2.2 系统运行库

Android 运行层的核心是 Dalvik 虚拟机。在 Android 系统中，每一个 Android 应用

程序对应一个 Dalvik 虚拟机。Dalvik 虚拟机在设计上也使得一台移动设备可以高效的运行多个它的实例。Dalvik 虚拟机可执行文件格式是 dex。dex 格式是专为 Dalvik 虚拟机设计的压缩格式, 适合内存和处理器速度有限的移动设备。开发 Android 应用时, Java 程序通过编译生成.class 文件, 还需要通过 SDK 中提供的 dx 工具转化为 dex 文件才能在虚拟机上执行。一个 dex 文件通常会包含多个 class。Dalvik 虚拟机依赖下层的 Linux 内核来实现线程和内存管理等功能。

3.2.3 应用程序框架

应用程序框架层是与 Android 开发者联系最紧密的一层, 它为 Android 开发提供了一套服务, 主要有:

(1) 视图 (View)。Android 应用框架层为开发者提供了文本视图 (TextView)、按钮 (Button)、列表视图 (ListView) 等基本视图, 也提供了线性布局 (LinearLayout)、相对布局 (RelativeLayout)、帧布局 (FrameLayout) 等布局来组织基本视图。视图以树状结构组织起来构成我们所见到的各式各样的 UI 界面。

(2) 内容提供者 (Content Providers)。内容提供者实现了一种 Android 应用程序之间的数据共享机制。应用程序可以访问、修改其他应用程序共享的数据, 也可以共享自己的数据。

(3) 资源管理器 (Resource Manager)。在 Android 开发中, 界面可以通过布局文件 (xml 文件) 来定义。布局文件即是这里所说的资源的一种。资源是相对代码而言的, 可以说 Android 应用程序是有代码和资源组成的。除了上述的布局文件, 资源还包括图标、字符串等等。资源管理器统一管理这些资源, 为开发者提供一致的资源获取接口。

(4) 通知管理器 (Notification Manager)。在 Android 开发中, 经常有在状态栏通知中心显示通知的需求。通知管理器是应用程序框架层为 Android 开发提供的服务, Android 开发者只需调用通知管理器的 API 即可实现通知推送的功能。

3.2.4 应用程序

应用程序层在 Android 系统的最上层, 面向用户提供服务。应用程序层主要包括系统应用和第三方应用。我们平常使用的拨号、消息、计算器等应用均位于此层。

3.3 Android 应用程序开发技术

Android SDK 为 Android 开发者提供了四大应用程序组件, 是 Android 开发不可或缺的强大工具, 分别是 Activity、Service、Broadcast、ContentProvider。

3.3.1 Activity

Activity 称为活动, 是应用程序的表示层, 每个程序包括一个或多个 Activity。应用

程序的每个屏幕显示都通过继承和扩展 Activity 基类实现。在 Android 中，不同应用程序的 Activity 可以相互调用，共同协作完成一个任务。这种协作是通过任务栈机制实现的。任务栈机制将来自不同应用程序的 Activity 放在同一个任务栈中，对于用户来说，好像一个应用程序完成了这个任务。任务栈本身仅是 Activity 的组织结构，无法设置参数。任务栈的特性通过 Activity 的启动参数来确定。根据不同的启动参数，可以把 Activity 的启动模式分为四种，分别是 Standard、SingleTop、SingleTask 和 SingleInstance。

(1) Standard。Standard 模式为默认模式，以 Standard 模式启动一个 Activity，总是创建一个新的 Activity 实例，并把这个实例置于当前任务栈栈顶。

(2) SingleTop。以 SingleTop 模式启动一个 Activity 时，将首先判断当前任务栈栈顶是不是该 Activity 的实例，如果不是，接下来的操作与 Standard 模式相同；如果是，调用栈顶 Activity 实例的 onNewIntent 方法，而不再创建一个新的实例。

(3) SingleTask。以 SingleTask 模式启动一个 Activity 时，将首先判断当前任务栈中是否存在该 Activity 的实例，如果不存在，接下来的操作与 Standard 模式相同；如果存在，调用 Activity 实例的 onNewIntent 方法，并销毁其之上的 Activity，让该 Activity 实例居于任务栈栈顶，接受用户响应，而不再创建一个新的实例。

(4) SingleInstance。以 SingleInstance 模式启动一个 Activity 时，将首先查找系统中是否存在该 Activity 的实例，如果不存在，创建一个新的任务栈并创建一个新的 Activity 实例置于新的任务栈的栈底，并且保证此任务栈自始至终只有这一个 Activity；如果存在，调用 Activity 实例的 onNewIntent 方法。

Activity 从创建到销毁可能经历多种状态，在不同状态之间切换时，将调用不同的回调方法 (CallBack)，如图 3.2 所示。Activity 生命周期中包含四种状态，分别是活动状态、暂停状态、停止状态和非活动状态。

(1) 活动 (Active/Running) 状态。处于屏幕前台，可以响应用户操作的 Activity 处于活动状态。此时，它处于任务栈的栈顶。同一时刻至多只有一个 Activity 处于活动状态。

(2) 暂停 (Paused) 状态。对用户仍可见但失去焦点，无法响应用户操作的 Activity 处于暂停状态。比如，一个 Activity 被另一个 Activity 部分遮挡时，它处于暂停状态。一个 Activity 透明或者尺寸小于屏幕尺寸，会部分遮挡启动它的 Activity。暂停的 Activity 仍处于存活状态，它保留着所有的状态和成员信息。Activity 的 UI 视图也保持着与窗口管理器 (WindowsManager) 的连接。仅当系统资源极度紧张时，暂停状态的 Activity 才可能会被系统销毁以回收资源。

(3) 停止 (Stopped) 状态。被另一个 Activity 完全遮挡的 Activity 处于停止状态。停止的 Activity 仍然保留着所有的状态和成员信息。但 Activity 的 UI 视图已与窗口管理器 (WindowsManager) 断开了连接。当系统资源紧张时，停止状态的 Activity 往往会被

系统销毁以回收资源。

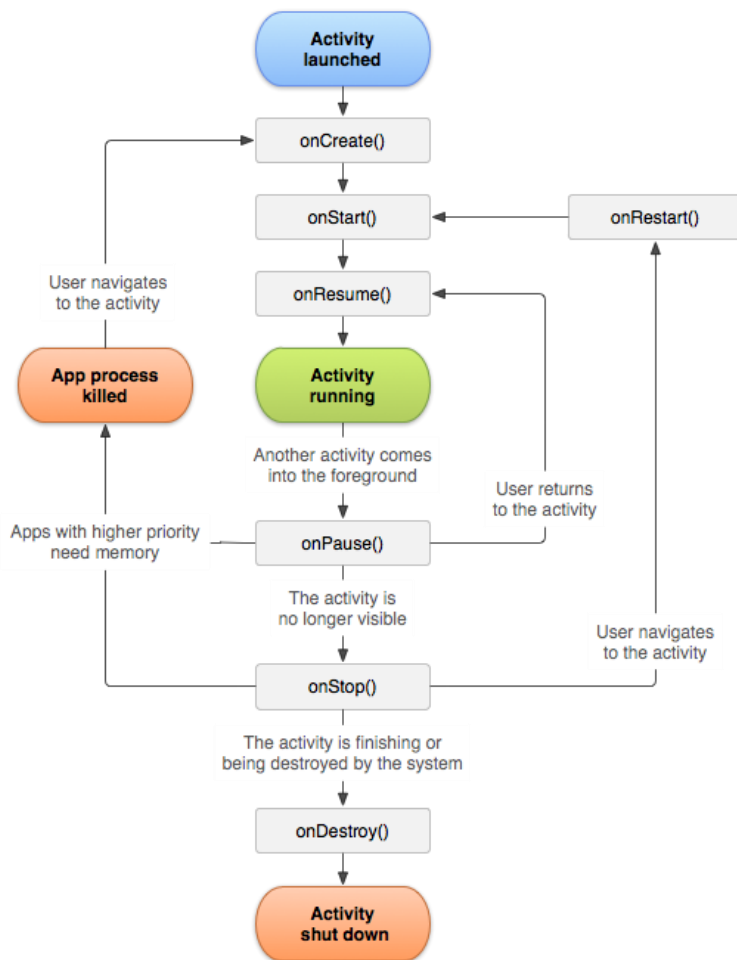


图 3.2 Activity 生命周期

（4）非活动（Dead）状态。尚未被启动、已经被终止、已经被系统销毁的 Activity 处于非活动的状态。非活动的 Activity 不保存任何状态或者信息。

3.3.2 Service

Service 跟 **Activity** 的级别相同，区别是 **Service** 运行在后台，不提供用户界面。一个 **Service** 是一个可以长期运行在后台的应用程序组件。**Service** 可以与其他应用程序组件交互，另一个应用程序组件可以启动一个服务，它将继续在后台运行，即使用户切换到另一个应用程序。此外，一个组件可以绑定到一个服务与之交互，甚至执行进程间通信（**IPC**）。例如，一个服务可以在后台处理网络交易、播放音乐、执行文件 **I/O**，或与内容提供者交互等。一个服务基本上有两种形式：

(1) Started (启动)

Service 由其它应用程序组件（例如 **Activity**）调用 **startService** 启动。一个服务一旦启动，可以无限期地在后台运行，即使启动它的组件被摧毁。通常情况下，开始服务执行一个操作，不向调用者返回一个结果。例如，它可能通过网络下载或上传文件。当操

作完成，服务应该自动销毁。

（2）Bound（绑定）

其它应用程序组件通过调用 `bindService` 绑定到一个服务。一个绑定服务提供了一个客户端—服务器接口，允许组件与服务交互，发送请求并得到的结果，甚至跨进程通信（IPC）。一个 `Service` 可以同时和多个客户绑定，当所有客户都解除绑定之后，系统会销毁 `Service`。

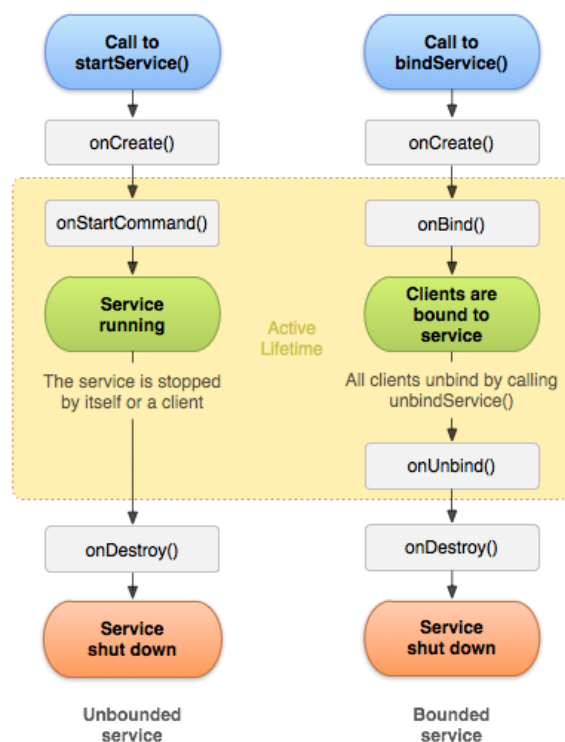


图 3.3 Service 生命周期

需要注意的是，一个服务运行在其宿主进程中，服务不会创建自己的线程，除非另行指定。这意味着，如果服务是用做进行任何耗时操作（例如 MP3 播放或网络下载等），应该在该服务中创建一个新的线程来执行该工作。通过使用一个单独的线程，将会减少应用程序的不响应（ANR）错误，应用程序的 UI 主线程可以继续致力于响应用户的操作。

3.3.3 BroadcastReceiver

在 Android 中，广播分为两种，系统广播和自定义广播。常用的系统广播包括启动广播、低电量广播、网络可用广播等等。自定义广播可以用于 APP 内部通信。BroadcastReceiver 使得应用可以对感兴趣的外部事件（如电话呼入、数据网络变为可用）进行接收并做出响应。广播接收器没有用户界面，但是可以启动一个 Activity 或 Service 来响应它们收到的信息，也可以用 NotificationManager 来通知用户。通知一般来说是在

状态栏上放一个持久的图标，用户可以点击打开它并获取消息。使用 Broadcast 需要注意两点。

(1) BroadcastReceiver 生命周期只有十秒左右，如果在 onReceive() 内做超过十秒内的事情，就会报 ANR(Application No Response) 程序无响应的错误信息。如果需要完成一项比较耗时的工作，应该通过发送 Intent 给 Service，由 Service 来完成。这里不能使用子线程来解决，因为 BroadcastReceiver 的生命周期很短，子线程可能还没有结束 BroadcastReceiver 就先结束了。BroadcastReceiver 一旦结束，此时 BroadcastReceiver 的所在进程很容易在系统需要内存时被优先杀死，因为它属于空进程（没有任何活动组件的进程）。如果它的宿主进程被杀死，那么正在工作的子线程也会被杀死。所以采用子线程来解决是不可靠的。

(2) 动态注册广播接收器还有一个特点，就是当用来注册的 Activity 关闭后，广播也就失效了。静态注册无需考虑广播接收器是否被关闭，只要设备是开启状态，广播接收器就是有效的。也就是说即便 APP 本身未启动，APP 订阅的广播在触发时也会对它起作用。

3.3.4 ContentProvider

ContentProvider 为存储和获取数据提供了统一的接口。无论应用程序是否启动，其它应用程序都可以通过接口来操作该应用程序的内部数据，包括增加数据、删除数据、修改数据、查询数据等。一般来说 ContentProvider 是单例模式的，当多个应用程序通过 ContentResolver 操作 ContentProvider 的数据时，ContentResolver 调用的数据将会委托给同一个 ContentProvider 处理。

4 基于 Android 平台的服饰图像搜索系统实现

基于上文所述的相关理论和研究成果，本文设计并实现了一个基于 Android 平台的服饰图像搜索系统。本章将详细介绍系统的架构设计以及客户端、服务器的实现过程。

4.1 系统架构

本文系统由三个部分组成，一是 Android 客户端；二是图像集；三是服务器端。系统架构如图 4.1 所示。系统工作流程如下：移动端以拍照或从图库选图的方式获得待搜索图像上传至服务器；服务器进行图像搜索，得到相似度最高的服饰图像列表返回给客户端；移动端得到相似服饰图像列表并向用户展示，用户可以保存、收藏、分享、购买这些服饰；特征池的建立是离线的，首先使用网络爬虫从电商平台爬取服饰图像建立服饰图像集，然后提取特征建立特征池。Android 客户端开发使用 Android Studio 集成开发环境。服务器端开发使用 Pycharm 集成开发环境。

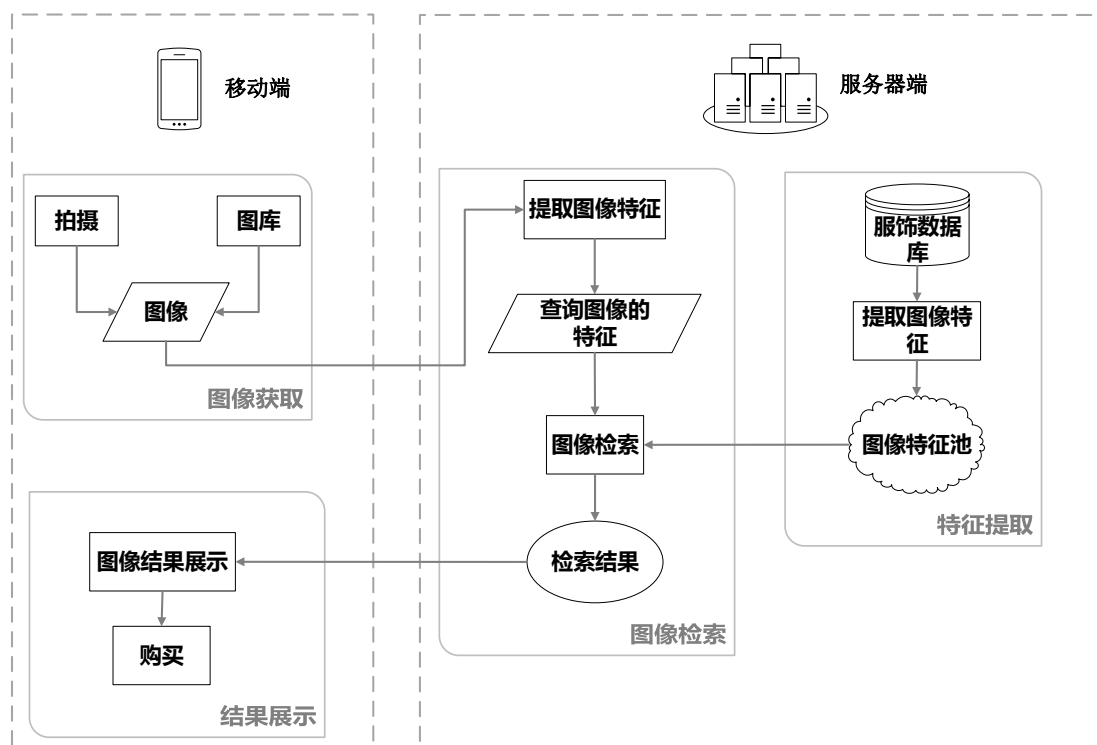


图 4.1 系统架构

4.2 Android 客户端实现

Android 客户端主要由获取查询图像、网络访问、展示搜索结果三个模块组成。查询图像的获取有两种方式，一是通过摄像头拍照得到；二是通过手机图库选择得到。网络访问模块，移动端与服务器端以 JSON 格式交换数据。搜索结果有两种展示模式，一是概要展示模式，将相似服饰图像缩略图以瀑布流的形式展示给用户；二是详情展示模式，将相似服饰图像原图以翻页的形式展示给用户。在详情展示模式下，支持用户保存、收藏、分享、购买等操作，也可以查看服饰品牌、价格、材质等相关信息。除查询图像的获取、网络访问和搜索结果的展示的三大主要模块之外，本文的 Android 客户端还包括新手引导模块、收藏模块、用户模块、通用模块等辅助模块。功能结构如图 4.2 所示。

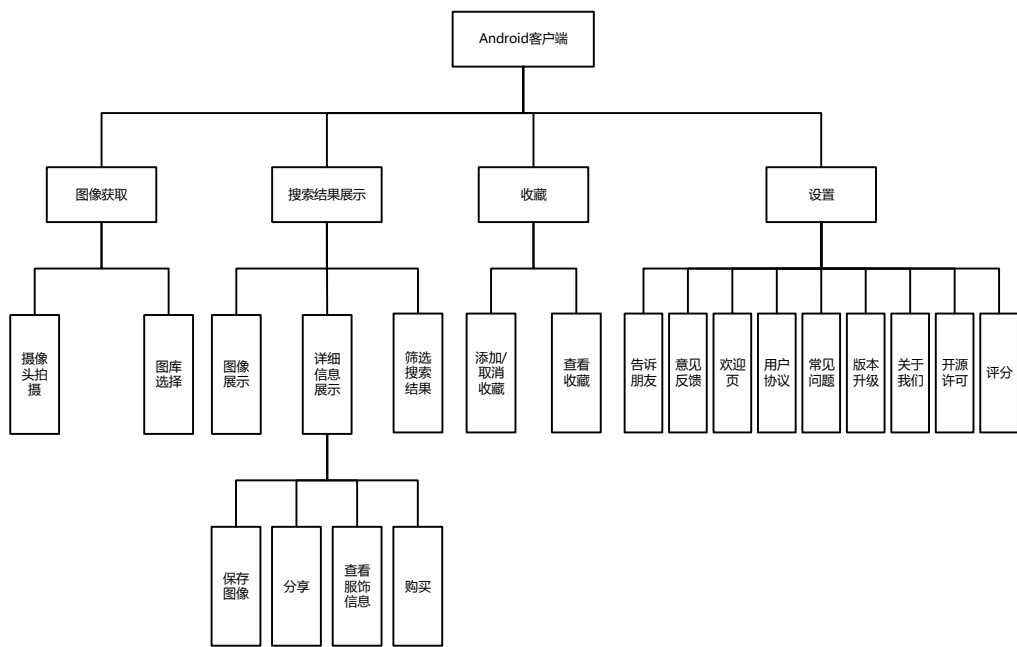


图 4.2 Android 客户端功能结构

Android 客户端硬件要求：摄像头；系统要求：Android 4.0 以上；权限要求：访问摄像头权限（CAMERA）、写外部存储权限（WRITE_EXTERNAL_STORAGE）、访问网络状态权限（ACCESS_NETWORK_STATE）、访问网络权限（INTERNET）、挂载外部文件系统权限（MOUNT_UNMOUNT_FILESYSTEM）。

图 4.3 展示的是本文 Android 项目的工程目录,其中 java 文件夹下是项目的源代码，res 文件下是项目的资源（主要是布局文件）。源代码共分为 activity、adapter、dialog、fragment、model、popupwindow、utility、widget 共计 8 个包。activity 包中是活动类，客户端的主要功能逻辑均在活动中实现。adapter 包中是存放适配器类，负责适配数据用于

视图进行展示。`dialog` 包中是对话框类，比如展示服饰品牌、价格等信息的对话框类。`fragment` 包中是碎片类，比如拍照 `fragment` 和图像选择器 `fragment`。`model` 包中是数据模型类，比如相似服饰模型类。`popupwindow` 包中是弹出窗口类，比如图像选择器中的弹出窗口。`utility` 包中是一些工具类，比如获取屏幕尺寸类、查看网络状态类等。`widget` 包中是一些自定义控件类，比如利用手势进行快速分享和收藏的自定义控件类。

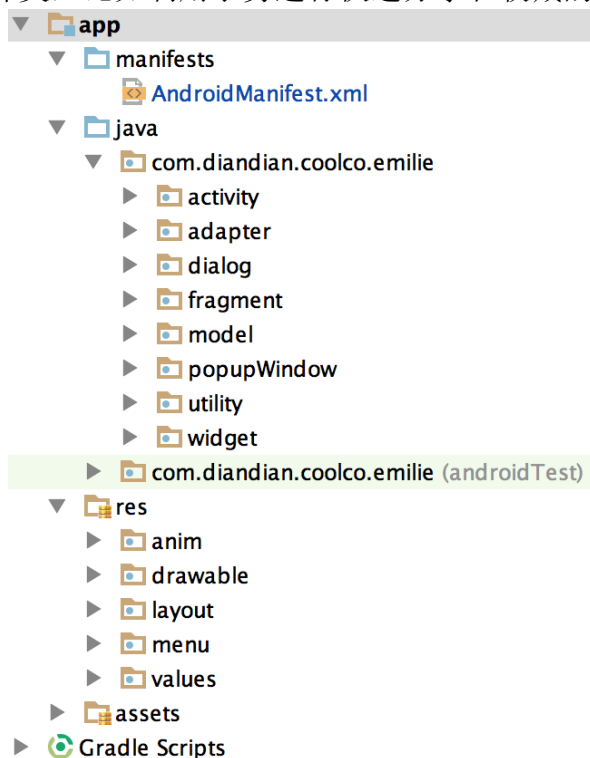


图 4.3 Android 项目工程目录

4.2.1 逻辑架构

在逻辑上，本文 Android 客户端可划分为四层，分别为界面层、核心层、接口层、模型层。界面层负责 UI 展示；核心层负责处理业务逻辑；接口层负责与服务器进行交互；模型层定义所有的数据模型。如图 4.4 所示，界面层依赖核心层、模型层，核心层依赖接口层、模型层，接口层依赖模型层，模型层不依赖任何层级。在构建项目，依照模型层、接口层、核心层、界面层的顺序依次构建。

模型层横跨所有层级，定义模型类，封装项目用到的数据。在本文项目中，主要定义了 `Image` 类，封装服饰图像的图像地址、购买地址、品牌、价格、材质等信息。

接口层的主要的功能是封装网络请求。接口层首先将请求封装好发送给服务器，然后将网络访问得到的数据转化为模型层定义的数据类型返回给上层的核心层。在本文项目中，客户端和服务端以 JSON 格式交互数据。发送请求使用 `HttpURLConnection` 系统类。请求结果处理上，本文利用 `Gson` 库将服务器返回的 JSON 格式服饰图像信息直接映射成为 `Image` 类型。

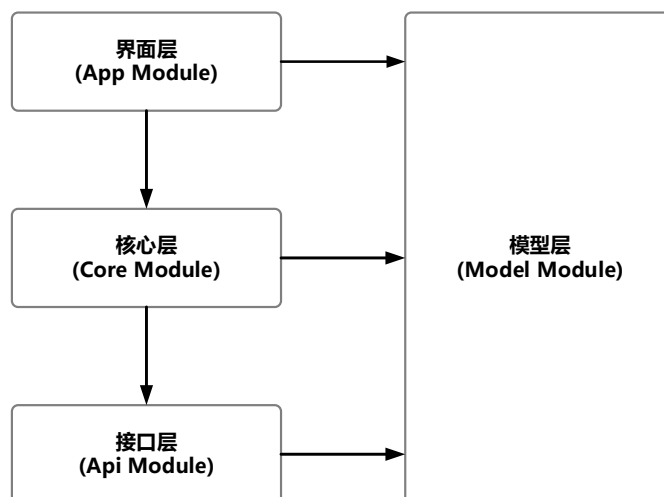


图 4.4 Android 客户端逻辑结构

核心层介于接口层和界面层之间，负责处理业务逻辑。向上，向层界面层提供待展示的数据。向下，调用接口层获取服务器的数据。由于网络访问是异步的，此层暴露给界面层的 API 都有一个 Callback 回调函数参数。界面层向核心层请求数据的时候，核心层会立即返回，在后台调用接口层向服务器请求数据，在得到数据并处理后通过 Callback 回调函数返回给界面层。Callback 模式的引入可以避免网络请求阻塞 UI 线程造成的应用程序无响应错误（ANR，Application Not Responding）。

界面层处于最上层，负责 UI 的展示，接收用户输入。根据不同类型划分，主要分为以下几个包：activity、adapter、fragment、util、wiget。其中 activity、adapter、fragment 各自都有一个基类，抽取相同的逻辑，比如定义了一些共用的常量、对象和方法等。



图 4.5 图像搜索主界面

4.2.2 图像输入模块

客户端支持拍照搜索，从本机图库中选取图像搜索两种搜索方式，如图 4.5 所示。点击拍照搜索将跳转到自定义的拍照页面进行拍照，点击选图搜索将跳转到自定义图片选择页面进行选图。尽管图像来源不同，拍照和选图都将返回一个图像的绝对路径，供后续网络模块使用。



图 4.6 拍照

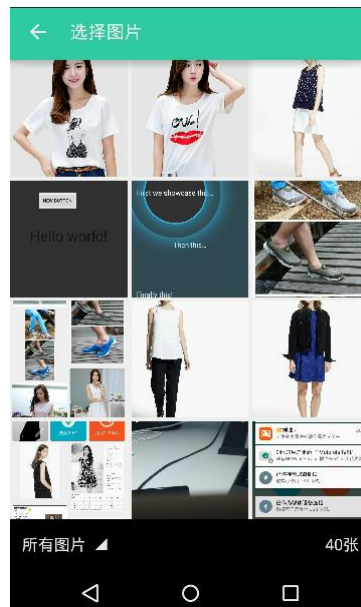


图 4.7 图片选择器

(1) 拍照

拍照功能可以调用系统服务完成。本文 Android 客户端采用自定义页面进行拍照的原因主要有两个。一是项目需要在拍照预览界面加矩形框，提醒用户将搜索目标置入矩形框内拍照，后续依据矩形框进行裁剪进而提高搜索准确度，而系统拍照服务并不支持界面的定制。二是不同版本的 Android 系统拍照界面也不同，低版本 Android 系统的拍照界面十分简陋。考虑到用户体验的一致性以及界面风格的统一，本文项目的拍照功能由自己实现，而非调用系统服务。自定义的拍照页面如图 4.6 所示。

页面主体为摄像头的实时预览，在实时预览上层有一个矩形框。框内透明，框外蒙着一层灰色，和矩形框上部的文字一起提醒用户将服饰目标置入矩形框内然后再拍照。矩形框和蒙层由一个自定义视图控件实现，显示效果完全由自己绘制。

页面底部为拍照按钮。点击拍照按钮进行拍照。然后播放转场动画（Transition Animation），拍照按钮向下坠落然后消失，取消、确定、重拍三个按钮向上升起然后出现。转场动画的存在让页面跳转更加友好。没有动画的过渡，用户会对界面的变化感到困惑。拍照后将出现取消、确定、重拍三个按钮。如果对拍照结果满意，点击确定按钮，跳转到搜索结果展示页面。如果对拍照结果不满意，点击重拍按钮。点击取消，跳转到搜索主界面。

如果点击了确定按钮，客户端将立即跳转到搜索结果展示页面，在此页面展示进度条（Progress Bar）提醒用户等待，避免出现应用程序无响应错误（ANR, Application Not Responding）。点击了确定按钮，拍照页面将把拍照所得图像做旋转处理并存储为临时图像文件，得到临时图像文件的路径。上述工作交由整个应用程序共用的后台线程池去执行，执行结束后通过 EventBus 异步的通知搜索结果展示页面，实现数据处理和页面跳转相互分离。之所以先跳转到搜索结果展示页面再通过 EventBus 把临时图像文件路径传递过来是为了减少一次进度条的展示。直接的做法是在拍照页面展示一次进度条，让用户等待客户端后台旋转图像并存储为临时图像文件，得到临时图像文件的路径后，让 Intent 夹带临时图像文件路径信息，跳转到搜索结果展示页面，再展示一次进度条，让用户等待服务器返回搜索结果。但是这样处理，将连续展示两次进度条，并且两次进度条之间用户没有任何操作，严重影响用户体验。本文客户端的处理方法，可以很好的合并进度条，提升用户体验。

（2）图像选择器

图像选择可以通过调用系统服务实现。之所以项目中选择自己实现图像选择器主要原因有两个。一是 Android 系统图像选择服务以文件夹的形式组织图像，选中一副图像至少需要两步（选中相册、选中图像），并且选择过程中需要进行界面的跳转，不够简单便捷。本文客户端中的自定义图像选择器把图像文件夹选择和图像选择置于同一页面中，图像文件夹的选择以底部弹窗（Popup Window）的形式出现。二是自定义图像选择器增加了所有图像这一逻辑文件夹，并按时间倒序排列图像，所有图像这一逻辑文件夹也是图像选择器的默认文件夹，所以用户打开自定义的图像选择器就可以看到最近拍摄、保存的图像，点击即可选择目标图像，操作更加高效。图片选择器页面如图 4.7 所示。

图像选择器中图像的展示采用 GridView 布局。Android 设备内存有限，在展示大量图像的时候容易导致 OOM（Out of Memory）错误。造成 OOM 的主要原因是图像文件在展示之前需要解压（decode）为位图（Bitmap），而位图非常占用内存。针对这一问题，本文图像选择器做了如下优化：

使用缩略图代替原图。在图像选择器页面中，每幅图像只占一格，其显示尺寸远远小于其实际尺寸。所以在解压图像的时候，可以一定的采样率去解压，然后进行伸缩变换，得到的位图的尺寸将与显示的尺寸相匹配，此时得到的位图占用的内存将远远小于将图像直接解压得到的位图占用的内存。

缓存缩略图。用户在选择图像的时候，经常会上下滑动。如果采取图像被滑出屏幕时被回收（recycle）、滑入屏幕时重新解压（decode）的策略有两个弊端。一、解压是高计算量操作，频繁解压将大量消耗计算资源。二、用户在上下滑动的时候，大量的图像被滑出屏幕和滑入屏幕，大量的位图将被创建和回收，导致频繁的 GC（垃圾回收），将造成内存抖动，进而导致界面卡顿。本文项目中，利用 LRU 策略缓存图像缩率位图，

在需要显示图像时,将首先到缓存查找对应的位图。如果命中,直接显示。如果未命中,从原图像文件中采样解压、缩放得到与显示尺寸匹配的位图,显示并加入缓存。图像在刚刚被滑出屏幕又被滑入屏幕时,不会被回收,也不需要重新从原图片文件中解压得到,很好的解决了上文描述的两个问题。经过优化,本文客户端图像选择器滑动流畅,并有效的避免了 OOM 错误。

(3) 图像裁剪

用户拍照后,客户端参照拍照预览上的矩形框自动对拍照所得图像进行裁剪。使用图片选择器选择图片后,将跳转到裁剪页面进行裁剪。裁剪页面如图 4.9 所示。用户可以拖动矩形框的边缘改变矩形框的大小,拖动其它位置改变矩形框的位置,直到矩形框将服饰框住,点击确定即可预览裁剪过后的图像,如果对裁剪结果不满意可以重新进行裁剪。在搜索结果列表展示页面,也有裁剪页面的入口,方便用户再次裁剪进行二次搜索。



图 4.8 用户引导



图 4.9 裁剪图像

4.2.3 网络模块

本文客户端的网络模块主要包括上传查询图像文件和解析服务器返回数据两个部分。客户端通过网络模块上传查询图像文件,服务器将返回相似服饰信息数据。相似服饰信息中最重要的信息是相似服饰图像的 URL。本文系统进行图像搜索时,首先返回的是相似服饰图像的 URL。在搜索结果展示模块,客户端再通过此 URL 去下载相似服饰图像。这么做的目的有两个,一是加快响应速度,提升用户体验;二是实现惰性加载,节省流量。一次搜索可能产生 50 个相似服饰图像结果,而用户并不一定会把它们全部浏览完。搜索时返回相似服饰图像的 URL,用户在浏览搜索结果时,浏览到哪一件相似

服饰时再去下载该相似服饰的图像可以减少下载不必要的图像而浪费流量。

(1) 上传查询图像文件

本文客户端使用 HTTPPOST 方法上传查询图像文件。在实现上借助 Android SDK 中的 HttpURLConnection 类，HttpURLConnection 实现了 HTTP 协议，可以用于发送请求，获得响应。发送请求时，首先设置报文头中的“Content-Type”属性为“multipart/form-data”，然后将位图（Bitmap）转化为字节数组填充到请求报文体中，发送报文，即可实现文件上传。关键代码见附录 B.1。

(2) 解析服务器返回的相似服饰信息数据

服务器返回的数据使用 JSON 格式。JSON 是目前应用非常广泛的数据交换格式，它具有易于阅读、信息密度高等优点。各个语言对它的支持都很好，封装和解析都十分方便。在本文中，服务器端使用 Python 标准模块 json 对数据进行封装和解析，Android 客户端使用 Google 的 Gson 开源库对数据进行封装和解析。

```
{
  "downloadUrl": "http://ec4.images-amazon.com/images/I/81v-020YfYL._UL1500_.jpg",
  "id": 13350,
  "brand": "Sandra Darren",
  "productName": "Sandra Darren Women's Petite Self Tie Sleeveless Printed Maxi Dress-服饰箱包-亚马逊中国-海外购 美亚直邮",
  "productId": "errorWishlist",
  "material": "95% 聚酯纤维/5% 氨纶",
  "price": "114",
  "specialPrice": null,
  "buyUrl": "http://www.amazon.cn/%E6%9C%8D%E9%A5%B0%E7%AE%B1%E5%8C%85/dp/B00YCEF014"
}
```

图 4.10 服务器返回 JSON 数据示例

```
@DatabaseTable(tableName = "image")
public class Image implements Parcelable {

    @DatabaseField(generatedId = true)
    private int id;
    @DatabaseField
    private String downloadUrl;
    @DatabaseField
    private String brand;
    @DatabaseField
    private String productName;
    @DatabaseField
    private String productId;
    @DatabaseField
    private String material;
    @DatabaseField
    private String price;
    @DatabaseField
    private String specialPrice;
    @DatabaseField
    private String buyUrl;
}
```

图 4.11 客户端相似服饰图像模型类

使用 Gson 可以方便的实现 Java 对象和 JSON 数据之间的相互转换，并且速度较快。客户端接受到服务端的响应之后，从响应中提取响应字符串。响应字符串是一个 JSON 数组。以 JSON 数组和预先定义好的相似服饰图像模型类为输入，Gson 即可将 JSON 数组转换为相似服饰图像模型类的对象数组。服务器返回的 JSON 数据样例如图 4.10 所示，包含的键（Key）有相似服饰图像下载地址（downloadUrl）、服饰品牌（brand）、服饰名称（productName）、服饰货号（productId）、服饰材质（material）、服饰价格（price）、

服饰优惠价格 (specialPrice)、服饰购买地址 (buyUrl)。相似服饰图像模型类如图 4.11 所示, 相似服饰图像类的成员变量与服务器返回的 JSON 对象的键一一对应, 以便使用 Gson 进行转换。

4.2.4 结果展示模块

(1) 搜索结果列表展示

搜索结果列表展示页面的主要功能是让用户上下滑动快速浏览系统搜索结果, 选择感兴趣的服饰点击跳转到搜索结果详细展示页面。此外, 页面还提供继续裁剪查询图像、搜索结果数量统计、搜索耗时统计等辅助功能, 如图 4.12 所示。搜索结果以瀑布流的形式分两列展示给用户, 从上到下相似度依次降低。

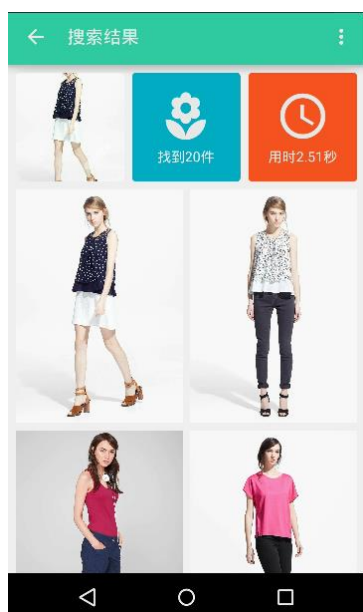


图 4.12 搜索结果列表



图 4.13 搜索结果详情

搜索结果图像的加载使用开源的图像加载组件 Fresco。Fresco 是一个强大的开源图片加载组件。解压后的图像将会占用大量的内存。大量内存的占用与释放势必引发更加频繁的 GC。在 5.0 以下, 频繁 GC 将会引发界面明显卡顿。在 5.0 以下系统, Fresco 将图像放到一个特别的内存区域, 不受系统垃圾管理器管理。当然, 在图像不显示的时候, 占用的内存会自动被释放。这会使得 APP 更加流畅, 减少因图像内存占用而引发的 OOM。Fresco 的设计遵循 MVC 模式。DraweeView 继承自 View, 负责图片的显示, 相当于 MVC 中的 V (View)。DraweeHierarchy 用于组织和维护最终绘制和呈现的图片, 占位图片和按下蒙层图片的设置均通过 DraweeHierarchy 实现, 相当于 MVC 中的 M (Model)。DraweeController 负责和 image loader 交互 (默认是 Fresco 中的 image pipeline), 实现对所要加载的图片的更多控制, 比如图片加载失败情况下单击重新加载, 相当于 MVC 中的 C (Controller)。本文客户端中, 服务器返回的相似图片 URL 交由 Fresco 去加载, 用户点击感兴趣的图片将跳转到服饰详情页面。

(2) 搜索结果详细展示模块

服饰详情页面全屏展示服饰图像。用户可左右滑动来查看上一幅图像和下一幅图像。左右滑动的实现使用的是 Android SDK 中的标准组件 ViewPager。为了给用户沉浸式体验，默认情况下，顶部 ActionBar（工具条）被隐藏，单击屏幕可显示。ActionBar 的更多（overflow）菜单中包括收藏、分享、保存到相册、去购买、查看服饰信息等菜单项，如图 4.13 所示。收藏操作是把当前服饰的相关信息保存到本地数据库中，之后可在收藏页面查看。保存到相册功能是把服饰图像保存到本地相册，方便用户在客户端外部查看服饰图像。去购买功能给用户提供了购买的渠道。购买地址是在使用网络爬虫抓取服饰图像时一并抓取得到的。用户点击去购买菜单，APP 将打开相应的购买网页，如图 4.14 所示。打开网页操作在 APP 内嵌的浏览器完成，并没有调用系统浏览器或第三方浏览器。内嵌浏览器保证了 APP 在各个 Android 版本的手机上使用体验一致。本文项目中实现的内嵌浏览器还提供了复制链接、用浏览器打开链接和分享三个辅助功能。其中，分享的内容并不仅限于服饰的购买地址，还包括 APP 相关信息。内嵌浏览器基于 Android SDK 中的标准组件 WebView 实现。点击查看服饰信息菜单将弹出对话框（Dialog）用于展示服饰信息，如图 4.15 所示。展示的服饰信息包括服饰品牌、服饰价格等信息。



图 4.14 购买服饰

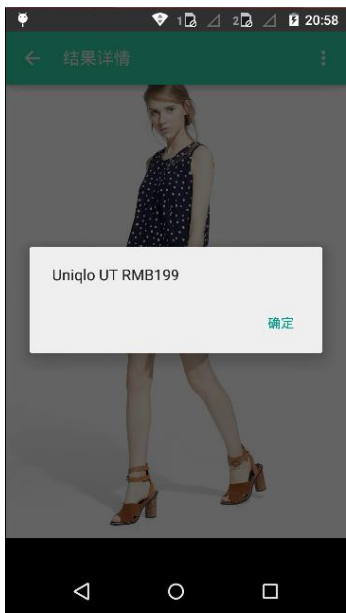


图 4.15 查看服饰信息

分享操作的基础是系统的分享功能。Android 有着完善的应用间通信机制，其与 Unix 的开发式工具链哲学一脉相承，是一个完整的体系。在 Android 中，所有的应用都向系统注册其所持有的组件和每个组件能够处理的任务。这样系统对于每个应用的每个组件能完成何种任务都了如指掌。当一个应用需要启动其它组件完成任务时，只需向系统发送一个消息，即可调用系统中可以完成该项任务的组件，而应用本身无需关心具体哪个应用的哪个组件最终处理了这个消息。这样跨应用的内容分享变得很方便。在本文项目

中，要分享的是纯文本，其中包含了图片的 URL 以及服饰相关信息，通过系统的分享功能实现。

收藏、分享、查看服饰信息等命令除了通过点击相应菜单项触发以外，还可以由手势（gesture）触发。收藏通过下拉触发，与常见的下拉刷新操作类似，如图 4.17 所示。如果已收藏，下拉则为取消收藏，添加收藏与取消收藏均有文字提示。分享通过上拉触发，如图 4.16 所示。查看服饰信息通过双击触发。双击显示服饰信息的过程为 3D 翻页动画效果，给用户服饰信息就在图片背后的感觉。翻页后，显示服饰信息，图片在背面。再次双击，再次翻页，图片回到正面，服饰信息回到背面。这种交互方式与生活经验相符，也易于理解。手势操作的优势是全局触发，相比菜单项必须点击某一矩形区域而言，手势操作更加方便快捷。除此之外，移动设备屏幕尺寸有限，使用手势代替菜单可以使界面更加简洁，同时释放大量视觉空间。手势操作被用户接受并熟悉后，会变成用户的直觉化操作，高效而便捷。

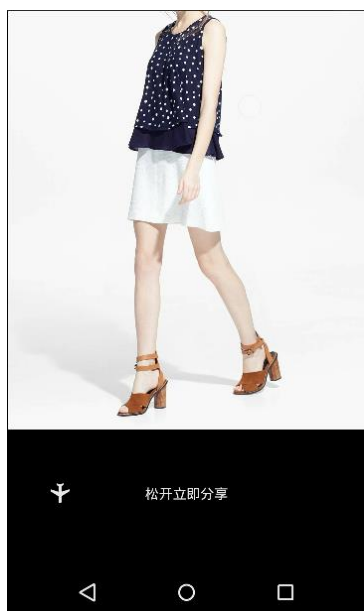


图 4.16 搜索结果详情页上拉进行快速分享服饰

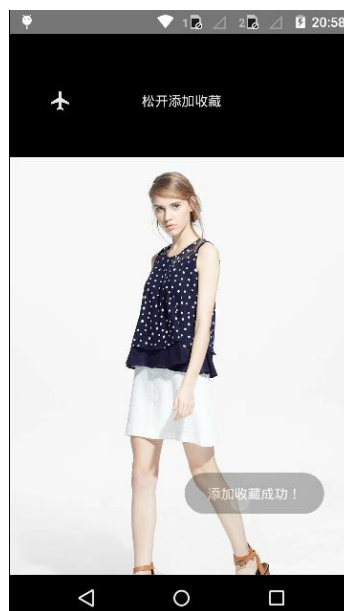


图 4.17 搜索结果详情页下拉快速收藏服饰

手势操作的实现有两个关键部分。一是头部视图（Header View，提醒用户松开添加收藏的视图）和底部视图（Footer View，提醒用户松开快速分享的视图）的隐藏与显示。Android 视图的显示过程主要分为三个步骤，依次是测量（measure）、布局（layout）和绘制（draw）。隐藏视图的实现方法是将头部视图显示在屏幕以上，底部视图显示在屏幕以下。具体实现方法是自定义线性布局（LinearLayout），从上至下添加头部视图、内容视图（Content View）和底部视图。重写 View 中的 onLayout 方法，改写头部视图的布局参数，使得头部视图的左下角与父视图（自定义线性布局）的左上角重合，把头部视图显示在屏幕以上，对用户不可见。同理，改写底部视图的布局参数，使得底部视图的左上角与父视图（自定义线性布局）的左下角重合，把底部视图显示在屏幕以下，对用

户不可见。头部视图和底部视图的显示通过调用 `scrollTo` 函数改变 `mScrollY` 实现。关键代码见附录 B.4。

二是对触摸事件（Touch Event）的处理和对手势的响应。在 Android 中，一个手势以一个 Down Event 开始，中间有若干个 Move Event，以一个 Up Event 或 Cancel Event 结束，三者需要分别处理。在本文项目中，通过 `setOnTouchEvent` 截获 TouchEvent。如果是 Move Event，滚动整个视图，露出头部视图或者底部视图。如果是 Up Event 或 Cancel Event，复原到初始状态。关键代码见附录 B.5。

4.2.5 其它辅助模块

（1）新手引导模块

本文 Android 客户端较多的使用了手势命令，而手势命令相对来说学习成本较高，为此本文客户端也设计并实现了新手引导模块，如图 4.8 所示。在用户第一次打开相关页面时，利用动画引导用户相关手势操作。动画引导相比文字引导、静态图片引导更加形象，可以最大程度的降低用户的学习成本。在第一次打开相关页面的时候引导该页面可以使用的手势命令，可以把手势学习分散到各个实际场景，更有利于用户接受和学习。引导动画的实现使用 Android 中的属性动画（ValueAnimator）。用 `SharedPreferences` 记录用户是不是第一次打开此页面。

（2）收藏模块



图 4.18 系统主菜单



图 4.19 收藏页面

本文 Android 客户端支持用户收藏感兴趣的服饰。用户收藏服饰时，客户端将服饰信息存入本地数据库。用户浏览自己的收藏时，客户端从本地数据库读取数据并显示，收藏页面如图 4.19 所示。本地数据库使用 SQLite。SQLite 是一个进程内的库，实现了自给自足的、无服务器的、零配置的、事务性的 SQL 数据库引擎。SQLite 直接访问其

存储文件。客户端使用 ORMLite 简化数据库操作。ORMLite 是一个轻量级的 Java 对象关系映射持久层框架,支持包括 MySQL、Postgres、Microsoft SQL Server、H2、Derby、HSQLDB 和 SQLite 等在内的数据库,提供灵活的 QueryBuilder 来构建复杂的数据查询以及强大的抽象 DAO (Database Access Object) 类。

(3) 用户模块

用户模块支持用户设置自己的头像和昵称,如图 4.20 所示。系统将统计用户的搜索次数、收藏次数,并由此计算成长值(积分)。系统将按照成长值为用户颁发勋章,鼓励用户使用 APP。

(4) 通用模块

通用模块包括“欢迎页”、“用户协议”、“常见问题”、“告诉朋友”、“意见反馈”、“版本升级”、“给我们评分”、“开源许可”、“关于我们”,如图 4.21 所示。其中,“欢迎页”介绍 APP 的核心功能,在 APP 首次打开时显示;“用户协议”声明用户需要遵守的行为规范,列举禁止事项(比如反向编译本客户端);“常见问题”列举用户反馈最多的问题并作出解答。常见问题页面是一个网页,方便时常更新;“告诉朋友”具有分享 APP 的作用;“意见反馈”收集用户的意见和建议,指导 APP 修复 Bug,不断完善;“版本升级”为 APP 内部升级渠道;“给我们评分”提供用户到应用商店为 APP 打分的途径;“开源许可”声明 APP 开发中使用第三方开源库,本文 APP 使用了 EventBus、Fresco、StaggerGridView 三个开源库;“关于我们”介绍系统开发成员。通用模块的存在使得 APP 功能更加完整。



图 4.20 用户页面



图 4.21 设置页面

4.3 图像集的建立

服饰图像集的建立是离线的。本文系统采用的服饰图像来自国内主流电商网站(唯

品会)，通过网络爬虫程序得到。网络爬虫不仅爬取服饰图像本身，也爬取服饰相关信息，比如品牌、价格、材质、货号。图 4.22 给出图像集的示例图像。

表 4.1 本文项目使用的图像集信息

项目	参数
图像数量	36960
图像尺寸	234×295
图像集大小	5941MB

本文网络爬虫程序使用 Python 实现，借助 pyquery 库。pyquery 利用 lxml 实现快速的 xml 和 html 操作。pyquery 支持在 xml 文件上做类似 jquery 查询，API 设计和 jquery 非常相似，可以非常方便的定位 HTML 标签并提取该标签的属性值，比如输入参数“div.cat-item-pic”可以定位到 class 为 cat-item-pic 的 div。这意味着，利用 pyquery 可以精准抓取服饰图像，而不是将服饰购买页上的所有图像都抓取下来之后过滤到非服饰图像。相比使用正则表达式在网页源码文本搜索感兴趣的字符串，借助 pyquery 实现网络爬虫更加简洁高效。



图 4.22 图像集示例图片

本文利用网络爬虫建立的图像集共包含 36960 张图片，尺寸为 234×295，大小为 5941MB，如表 4.1 所示。

网络爬虫程序爬行到一件服饰的购买页时，不仅抓取服饰的图片，服饰的品牌、商品名、货号、材质、价格、优惠价格以及当前购买页的地址都将一并被抓取下来。服饰

的相关文本信息以 JSON 的格式存放在文本中。本文爬取服饰文本信息示例如图 4.23 所示。

```
[
  {
    "downloadUrls": "http://ec4.images-amazon.com/images/I/61ykchX7Z6L._UL1500_.jpg;http://ec4.images-amazon.com/images/I/61K3pj2mm7L._UL1500_.jpg;http://ec4.images-amazon.com/images/I/71EFB1i04KL._UL1500_.jpg;http://ec4.images-amazon.com/images/I/61T55he2KIL._UL1500_.jpg;http://ec4.images-amazon.com/images/I/61s1xhzxdlL._UL1500_.jpg;http://ec4.images-amazon.com/images/I/61PPz7LahYL._UL1500_.jpg;http://ec4.images-amazon.com/images/I/61M1u257qyL._UL1500_.jpg;http://ec4.images-amazon.com/images/I/ZiWL._UL1500_.jpg;",
    "id": 1,
    "brand": "V.C欧美范",
    "productName": "色凹 欧洲站 女装2015秋装新款连衣裙秋 欧美修身显瘦荷叶边麂皮绒长袖连衣裙9192 【V.C欧美范】 服饰箱包 - 亚马逊中国",
    "productId": "errorWishlist",
    "material": " 麂皮绒空气层",
    "price": "880",
    "specialPrice": null,
    "buyUrl": "http://www.amazon.cn/%E6%9C%8D%E9%A5%B0%E7%AE%B1%E5%8C%85/dp/B017N4QU04"
  },
  {
    "downloadUrls": "http://ec4.images-amazon.com/images/I/71Rn240yNPL._UL1500_.jpg;http://ec4.images-amazon.com/images/I/71-B8EFRP7L._UL1500_.jpg;",
    "id": 2,
    "brand": "Calvin Klein Jeans 卡尔文克莱恩牛仔",
    "productName": "Calvin Klein Jeans Women's Button Back Summer Dress, Night Sky, Small 【Calvin Klein Jeans 卡尔文克莱恩牛仔】 服饰箱包 - 亚马逊中国-海外购 美亚直邮",
    "productId": "errorWishlist",
    "material": " 72% Modal/28% Polyester",
    "price": "223",
    "specialPrice": null,
    "buyUrl": "http://www.amazon.cn/%E6%9C%8D%E9%A5%B0%E7%AE%B1%E5%8C%85/dp/B00UBT1WFU"
  }
]
```

图 4.23 网络爬虫采集服饰信息示例

本文共爬取 19161 件服饰的文本信息，文本信息大小为 10.3MB，如表 4.2 所示。

表 4.2 网络爬虫采集服饰信息规模

项目	参数
服饰数量	19161
服饰信息大小	10.3MB

4.4 服务器端实现

本文服务器搭建在 Django 之上，Django 是一个开源的 Python Web 框架。本文主要依赖 Django 完成 URL 分发、网络请求接受和解析、响应的封装和发送等低层工作。系统核心的特征提取和特征匹配功能使用 Python 实现，是独立的模块。核心搜索模块向外暴露接口，供 Django 调用，与 Django 是松散耦合关系。服务器的基本工作流程如下：Django 接受到网络请求，解析请求，将查询图像传递给核心搜索模块，核心搜索模块进行图像搜索，将搜索结果返回给 Django，Django 封装后返回给客户端。

本文服饰图像搜索系统共有四种搜索方式，分别是基于颜色特征搜索、基于纹理特征搜索、基于颜色特征和纹理特征的组合搜索以及基于 SIFT 特征搜索。前三种搜索方式的工作流程如图 4.24 所示，基于 SIFT 特征搜索的工作流程如图 4.25 所示。

图 4.26 展示的是服务器的工程目录，其中 color.py 用于提取颜色特征；distance.py 用于计算特征之间的欧氏距离；grabcut.py 使用 GrabCut 算法对图像进行背景过滤；image.py 定义了图像模型；invert_index.py 实现了倒排索引；kmeans.py 用于对 SIFT 特征进行聚类构建视觉单词本；search_by_color.py 是基于颜色搜索的控制器，它调用 color.py、distance.py、top_k.py 等模块实现颜色特征的提取、颜色特征距离的计算以及颜色特征距离的排序并最终实现图像搜索功能；search_by_color_ulbp.py 是基于颜色特

征和 Uniform LBP 特征的组合搜索的控制器；search_by_sift.py 是基于 SIFT 特征搜索的控制器；search_by_ulbp.py 是基于 Uniform LBP 特征搜索的控制器；sift.py 实现了 SIFT 特征的提取；top_k.py 实现 TOP K 算法；ulbp.py 实现 Uniform LBP 特征的提取。其余文件为 Django 框架文件。

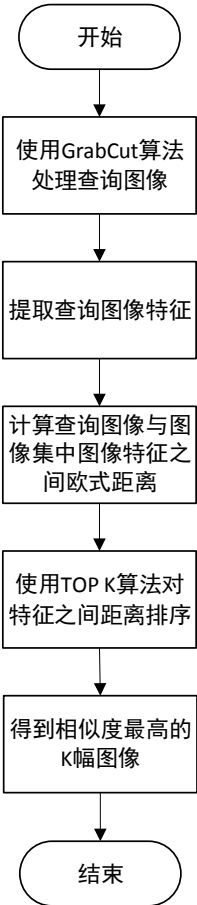


图 4.24 基于颜色特征、Uniform LBP 特征、颜色 和 Uniform LBP 组合特征的搜索流程

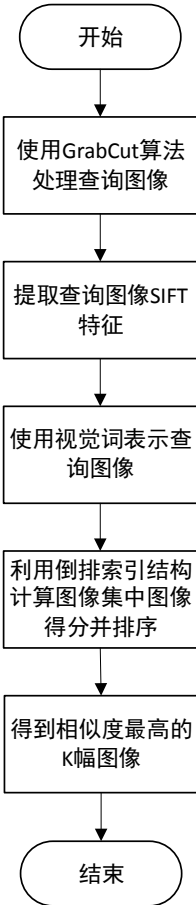


图 4.25 基于 SIFT 特征的搜索流程

服务端搜索核心模块主要包括 Grabcut 模块、颜色特征提取模块、纹理特征(Uniform LBP)提取模块、SIFT 特征提取模块、视觉词模块、倒排索引模块、排序模块。各个模块会在下面各个小节详细介绍。欧式距离的计算过程较为简单，在此不过多介绍。

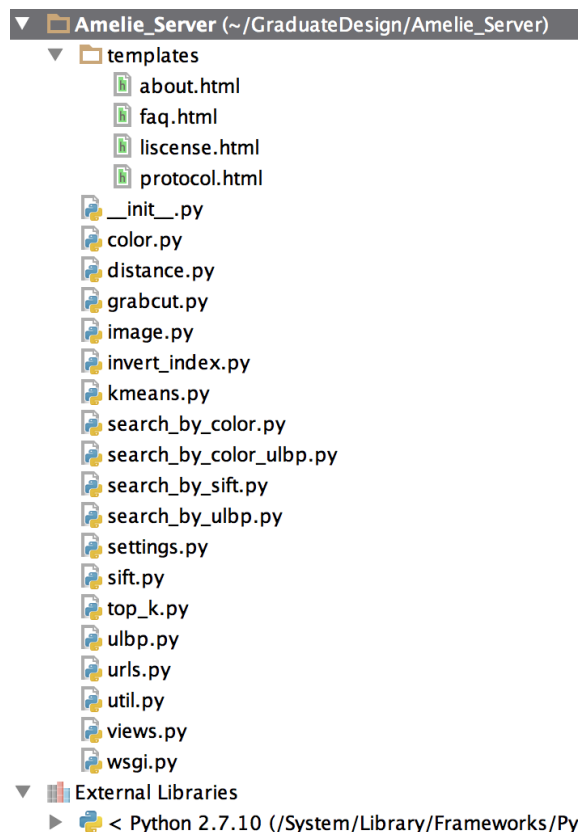


图 4.26 服务器工程目录

4.4.1 Django 简介

本文系统服务器采用 Python Django 架构。Django 是基于 Python 的开源 Web 框架，它有部署快速、安全性高、伸缩性强三个主要优势。图 4.27 为 Django 处理网络请求的流程图。

Django 以 MTV（Model-Template-View）模型架构，其中 Model 为数据模型，与数据库操作相关；Template 为模板系统，用于数据的格式化显示；View 为控制器，负责调用 Model、Template 进行业务逻辑的处理，是 MTV 模型的调度中枢。Django 作为 Web 架构提供的功能有：（1）Application 的可插入（Plug-in）管理；（2）ORM（对象关系映射）：使用 Django 开发 Web 应用，只需用 Python 定义数据模型即可，数据库将由类映射得到，数据库操作也都有相应的 API，不必自己写 SQL 语句；（3）URL 分发：在 Django 中，URL 的分发通过正则表达式实现，简洁高效；（4）模板系统：模板系统很好的分离了数据与显示。Django 的模板系统是可扩展的，能很好的满足个性化需求；（5）Cache 系统：Django 为开发者提供了各种粒度的缓存策略，方便开发者快速开发。

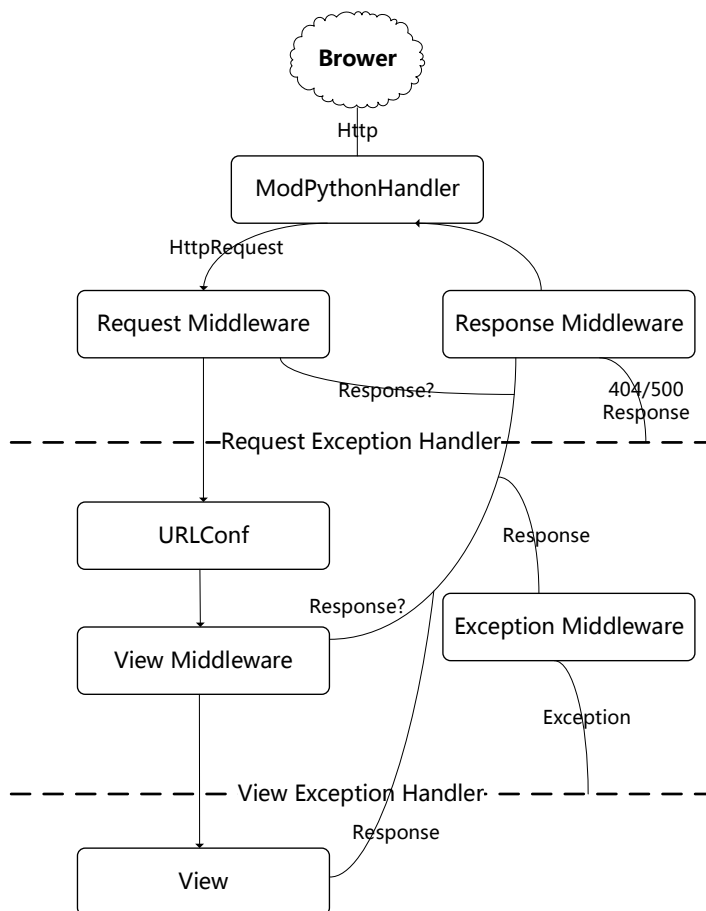


图 4.27 Django 处理网络请求流程

4.4.2 GrabCut 模块

服饰图像往往包含很多无关元素（比如人脸、背景等），而我们只对图片中的服饰部分感兴趣，无关元素的存在将降低图像特征提取的效率并影响图像搜索的准确度。本文选择使用交互式图像分割方法 GrabCut 来提取服饰图像中我们感兴趣的服饰部分，减少无关元素对图像特征提取的干扰以及对图像搜索准确度的影响。

在本文中，GrabCut 是预处理操作，利用 openCV 中的方法对图像集中的所有图像进行图像分割，过滤背景，将前景图像保存。后续的特征提取均在前景图像上进行。前景图像和源图像一一对应，前景图像用于图像搜索，源图像用于返回给客户端。OpenCV 中 GrabCut 初始化有两种方式：一是使用前景背景标记分别标记一部分前景像素和背景像素；二是使用一个矩形框标记未知区域和背景区域。本文项目使用第二种初始化方式。电商平台的服饰图像一般由专业摄影师拍摄，它们的构图基本一致。通过观察与实验发现，服饰图像左右两次各有 20% 的留白（背景），上下分别有 25% 的留白和 15% 的留白。矩形框的尺寸位置根据服饰图像的特点以及实验经验设置。长为图像长度的 60%，宽为图像宽度的 60%，矩形框水平居于图像中间，垂直上，距离图像上边缘距离为图像长度的 25%。本文项目中应用的 GrabCut 仅粗略的过滤了背景，在一定程度上提高搜索准确

度。使用 GrabCut 算法对查询图像进行处理时，初始化矩形尺寸与查询图像尺寸基本一致，仅在上下左右各有一个像素的留白。客户端上传至服务器的查询图像经过裁剪，此时查询图像的主体已经是想要搜索的服饰。



4.28 原始图像



图 4.29 前景图像

如图 4.28 和图 4.29 所示，使用 GrabCut 算法处理图像可以在一定程度上过滤模特脸部以及背景。本文项目中的 GrabCut 算法初始化矩形参数是根据经验设置，仅适用于本文项目图像集中的图像。

在项目中，使用 GrabCut 对图像集进行预处理，后续特征提取均在处理后的图像上进行。预处理操作是离线进行的，对图像集中 36960 图像进行 GrabCut 操作共计耗时 10.27 分钟，平均每张耗时 0.16 秒。


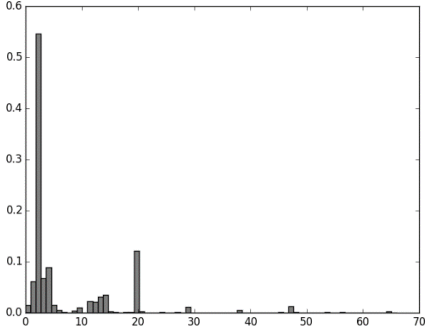

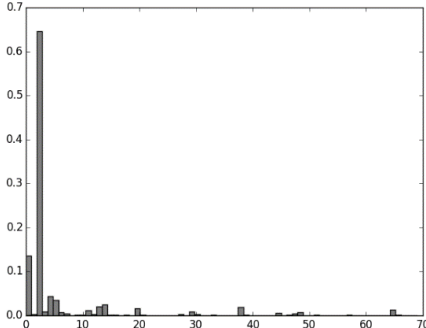

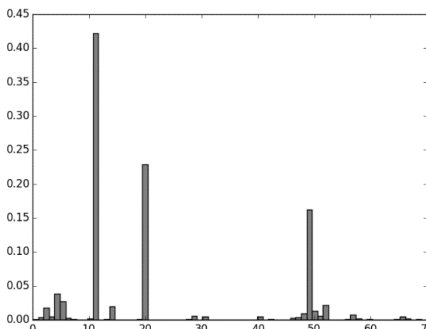
4.4.3 颜色特征提取模块

依据上文理论，颜色特征提取首先把数字图像的 RGB 颜色空间转换为 HSV 颜色空间，然后对 h、s、v 分别进行量化，量化后 $H \in [0,7]$ 、 $S \in [0,2]$ 、 $V \in [0,2]$ 。按照量化级数，把各个颜色分量合成为一维特征矢量，公式为 $W = 9H + 3S + V$ 。一维特征矢量 W 的范围是 $[0,71]$ ，统计其分布直方图，然后进行归一化，即可得到图像的 72 维颜色特征向量。关键代码见附录 B.2。

表 4.3 展示了三幅服饰图像以及它们的颜色直方图，直观上来看，第一幅示例图像与第二幅示例图像颜色更加接近，而与第三幅示例图像差距较大。从表中可以看到第一幅示例图像的颜色直方图与第二幅示例图像的颜色直方图形状更加相似，而与第三幅示例图像相差较大。通过计算，第一幅示例图像的颜色特征与第二幅图像的颜色特征之间

的欧式距离的平方为 0.0455；第一幅与第三幅之间的值为 0.4332。从示例中得知，本文选取的颜色特征模型可以较好的区分图像之间的颜色差异。

表 4.3 服饰图像颜色直方图示例

服饰图像	颜色直方图
	
	
	


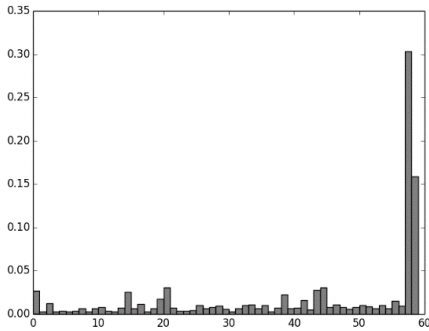

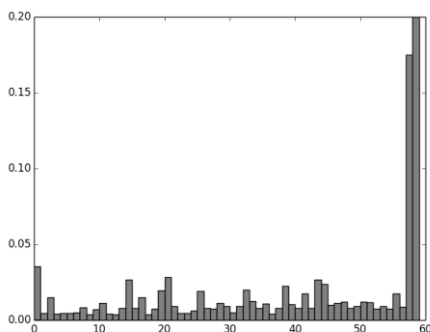

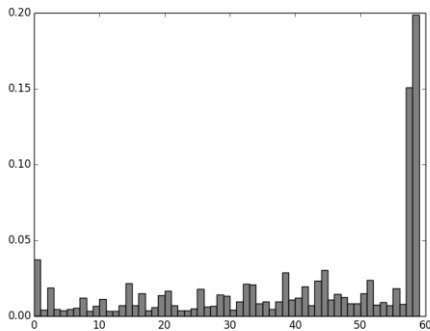
利用附录 B.2 中的代码对图像集中的 36960 幅图像进行颜色特征提取，总计耗时 33.21 分钟，平均每幅图像耗时 52.24 毫秒。

4.4.4 纹理特征提取模块

本文纹理特征采用 R=1、P=8 的 Uniform LBP 特征。特征提取算法如下：（1）定义均匀模式 LBP 特征向量映射表，将均匀模式 LBP 特征向量映射为一个序号值。（2）读取图像并转化为灰度图。（3）遍历每个像素点，针对每个像素点执行以下操作。将该像

素点与邻域内 8 个像素点进行比较, 若周围像素点灰度值大于中心像素, 则该像素点标记为 1, 否则标记为 0。从而从关键点邻域内得到八维二值向量 $pattern[]$ 。(4) 计算 $pattern$ 中包含的跳变 (由 0 变 1 或者由 1 变 0) 次数 $variations$ 。如果 $variations$ 大于 2, 表示该特征是非均匀模式特征。如果 $variations$ 小于等于 2, 表示该特征是均匀模式特征, 通过查表得到其序号值。(5) 统计均匀模式特征向量和非均匀模式特征向量的分布, 归一化后得到 Uniform LBP 特征向量。关键代码见附录 B.3。

表 4.4 Uniform LBP 直方图示例

原始图像	Uniform LBP 直方图
	
	
	

由表 4.4 可以看出, 所有非均匀模式的 LBP 特征加起来占总的 LBP 特征的比例低于 20%, 把它们算作一类进行统计并不会对准确度影响太大。而采用均匀模式的 LBP 来

描述图像，可以把直方图从 256 维压缩到 59 维，这将大大有利于提高后续的特征比较速度，进而提高图像搜索速度。

利用附录 B.3 中代码提取图像集中 36960 图像的 Uniform LBP 特征共花费 195.12 分钟，平均每幅图像耗时 316.8 毫秒。

4.4.5 SIFT 特征提取模块



图 4.30 原始图像



图 4.31 标注了 SIFT 特征点的图像

提取图像集中每幅图像的 SIFT 特征，每幅图像的 SIFT 特征存为一个文件。后续构建视觉词典操作直接读取文件获得图像的 SIFT 特征。对图像集中 36960 副图像进行 SIFT 特征提取，共提取到个 8676432 个 SIFT 特征，共用时 21.93 分钟，平均每副图像提取 234 个 SIFT 特征，平均每幅图像耗时 35.6 毫秒。提取 SIFT 特征平均数量以及平均耗时如表 4.5 所示。

表 4.5 SIFT 特征提取性能

图像尺寸	每张图片平均特征数量	每张图片特征提取平均耗时
234*295	234	0.0356s

4.4.6 视觉词模块

本文在基于 SIFT 特征的图像搜索中应用视觉词袋模型。SIFT 特征共有 2^{128} 种，利用视觉词袋模型的目的是减少 SIFT 特征的类别，为使用倒排索引做准备。

视觉词袋技术借鉴自文本检索中的词袋（Bag of words）技术。在词袋技术中，文档用单词直方图表示。借鉴过来，视觉词袋技术中，图像用视觉词直方图表示。它们的缺点是，词袋技术忽略了单词的顺序，视觉词袋技术忽略了视觉词的空间位置。视觉词袋技术把图像特征用距离最近的视觉词表示，也损失了一定的特征信息。视觉词袋的最大优势是它压缩图像描述，可以提高检索速度。视觉词袋实现过程如下：

步骤一：特征提取。提取图像集中每幅图像的局部特征，每幅图像提取结果是多个局部特征描述符。

步骤二：聚类生成视觉词典。步骤一中得到的所有特征构成一个特征空间，利用 K-means 进行聚类。设 (x_1, x_2, \dots, x_n) 是一组观测值序列，其中，每个观察值都是一个 d 维向量。利用 K-means 将这 n 个观察值划分到 k 个序列 $S = \{S_1, S_2, \dots, S_k\} (k < n)$ 中，使式 4.1 成立，其中 u_i 是 S_i 的均值。

$$\arg \min \sum_{i=1}^k \sum_{x_j \in S_i} \|x_j - u_i\| \quad (4.1)$$

步骤三：将图像用词典表示。将图像中的局部特征映射到距离最近的视觉词上，统计图像中各个视觉词的频率，得到分布直方图，最终用直方图 k 维向量表示图像。其中， k 为词典中包含的视觉词数量。局部特征之间的相似度用欧式距离度量。

对图像集中的每一幅图像进行 SIFT 特征提取，总共提取得到 8676432 个 SIFT 特征。然后由这些特征向量构成特征空间，采用均值聚类（K-Means）方法进行聚类，建立视觉单词本。聚类中心即为视觉词，聚类中心的集合即为视觉单词本。视觉词袋技术牺牲了一定的搜索准确度来换取搜索速度。视觉词典的大小对搜索准确度与搜索速度之间的平衡至关重要。视觉词典过大，图像特征类别基本未被压缩，无法起到加速搜索的效果。视觉词典过小，图像将变得难以区分，最终影响搜索准确度。通过实验，在本文中词典的大小取值为 10000。

4.4.7 倒排索引模块

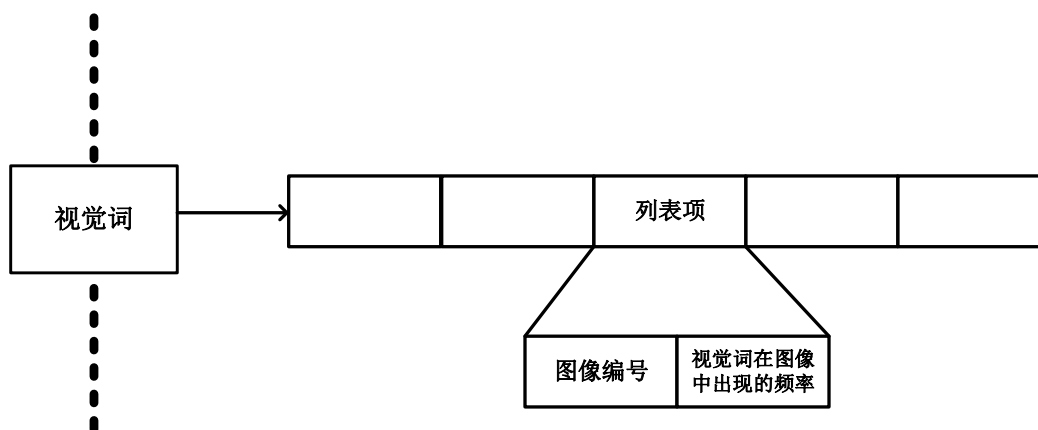


图 4.32 倒排索引结构

倒排索引是文本检索中广泛应用的技术。本文的倒排索引是依据视觉词反向查找图像的索引方法。其索引结构类似于邻接矩阵。索引结构中一行是图像集中包含该视觉词的所有图像的列表。列表项包括图像的编号，以及该视觉词在该图像中出现的频率。索引结构如图 4.32 所示。索引结构是离线生成的。利用此索引结构进行图像检索是一个计分的过程。输入一副查询图像，首先将查询图像用视觉词表示。然后遍历查询图像中包

含的视觉词，给其所处行图像列表中图像加分。分值为该视觉词在查询图像中出现的频率与此视觉词在被加分图像中出现的频率的乘积。再乘以一个权值 $\log(N/N_i)$ ，其中 N 表示的视觉词汇表的大小， N_i 表示的视觉词在整个图像库中出现的次数。用哈希表存储在列表中的图像得分。待查询图像的视觉词遍历完，获得图像的最终得分。按得分排序可以得到相似度最高的图像，完成搜索工作。

4.4.8 排序模块

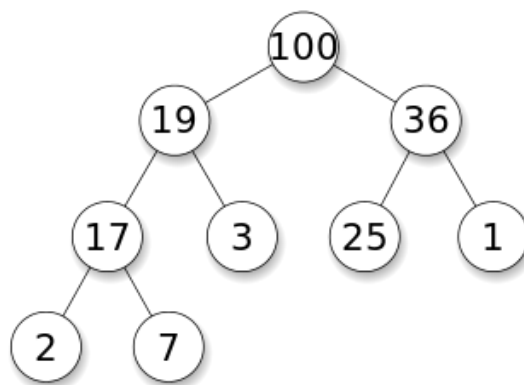


图 4.33 大顶堆示例

本文中的排序问题是典型的 Top K 问题，给定 N ($N \gg 10000$) 个特征间距离，求出其中前 K 个最小的距离。由于输入为大量数据，而只需要求得前 K 个最小值，因此对整个输入数据进行保存并排序是不必要的。

本文利用大顶堆数据结构来处理该问题。大顶堆示例如图 4.33 所示，每个非叶子节点的数值，一定不小于其孩子节点的数值。这样可以使用含有 K 个节点的大顶堆来保存 K 个目前的最小值，当前根节点是 K 个最小值中的最大值。每次有数据输入时，先与根节点比较，如果不小于根节点，则舍弃该数据；如果小于根节点，用该数据替代根节点，并进行大顶堆调整。数据输入完时，大顶堆保存的即是 N 个数据中前 K 个最小值。然后使用快速排序对这 K 个数据进行排序，从而得到从小到大有序的前 K 个最小距离。

本文对上述 TOP K 排序算法和常规快速排序算法进行了对比实验。实验过程如下：随机生成一个长度为 36960 的数组，分别使用上述 TOP K 排序算法和快速排序算法对数组进行排序，TOP K 排序算法中的参数 K 取值 50，然后重复进行 100 次实验，计算两种排序算法的平均耗时。实验结果如下：上述 TOP K 排序算法平均耗时 4.10ms；快速排序平均耗时 10.12ms。实验表明上述 TOP K 排序算法相比快速排序性能有较大提升。

5 系统测试

本章共包含三个小节，第一小节介绍测试环境，给出用于测试的手机配置以及服务器配置；第二小节给出 Android 客户端的测试结果，包括 Android 客户端兼容性、性能和功能的测试结果；第三节详细介绍服务器端测试结果，即系统搜索效果测试结果。本文服务器端有四种搜索模式可供选择，分别是基于颜色特征进行搜索、基于 Uniform LBP 纹理特征进行搜索、结合颜色特征和 Uniform LBP 纹理特征进行搜索以及基于 SIFT 特征进行搜索。本章分别对上述四种搜索方式的搜索性能进行了测试并给出测试结果。

5.1 测试环境

测试手机终端选择使用摩托罗拉 Moto G。摩托罗拉 Moto G 的硬件配置如表 5.2 所示。

表 5.1 摩托罗拉 Moto G 硬件配置

系统配置	硬件参数
Operating System	Android 5.0
CPU	Qualcomm Snapdragon 400 四核 1.2GHz
GPU	Adreno 305
RAM	1GB
屏幕大小	5.0 英寸
屏幕分辨率	1280*720

本文服务器代码均部署在远程服务器上，拥有固定 IP 地址，客户端可以随时随地地进行访问，请求服务器的图像搜索服务。本文项目使用的服务器的硬件配置如表 5.3 所示。

表 5.2 服务器硬件配置

系统配置	硬件参数
Operating System	Windows 7
CPU	Intel(R) Xeon(R) CPU E5-2650 v2 2.6GHz 2.59GHz(双处理器)
RAM	8GB
硬盘大小	2TB

5.2 客户端测试

本文主要从兼容性、性能、功能可用性三个方面对 Android 客户端进行了测试。

5.2.1 兼容性测试

众所周知，Android 设备厂商众多，Android 设备系统版本、屏幕分辨率各式各样，碎片化问题严重。Android 应用在兼容性方面存在很大的挑战。本文中 Android 客户端在设计和实现上很好的考虑了兼容性问题，实现了对主流机型的适配。真机适配测试结果如表 5.1 所示。真机适配测试结果证明本文实现的 Android 客户端兼容性良好。

表 5.3: 真机适配测试结果

Android 手机型号	Android 系统版本	分辨率	测试结果
三星 GalaxyS3	Android 4.3	1280*720	UI 显示及功能均正常
华为荣耀 3C	Android 4.4	1280*720	UI 显示及功能均正常
SnoyXperia C6603	Android 4.1	1920*1080	UI 显示及功能均正常
摩托罗拉 Moto G	Android 5.0	1280*720	UI 显示及功能均正常

5.2.2 性能测试

对 Android 应用来说，CPU 和内存都是稀缺的资源，所以性能问题一直是 Android 开发人员考虑最多的问题之一。一般来说，一个 Android 应用程序（Application）最多可以使用 16MB 内存，有些机型是 24MB。同时，Android 应用程序也要避免长时间占用 CPU，长时间占用 CPU 不仅会影响系统正常运行，也会造成大量电量消耗。本文使用 Android Studio 中的 CPU Monitor 和 Memory Monitor 分别对本文的 Android 客户端运行时 CPU 占用和内存占用进行了监控，监控结果结果如下：

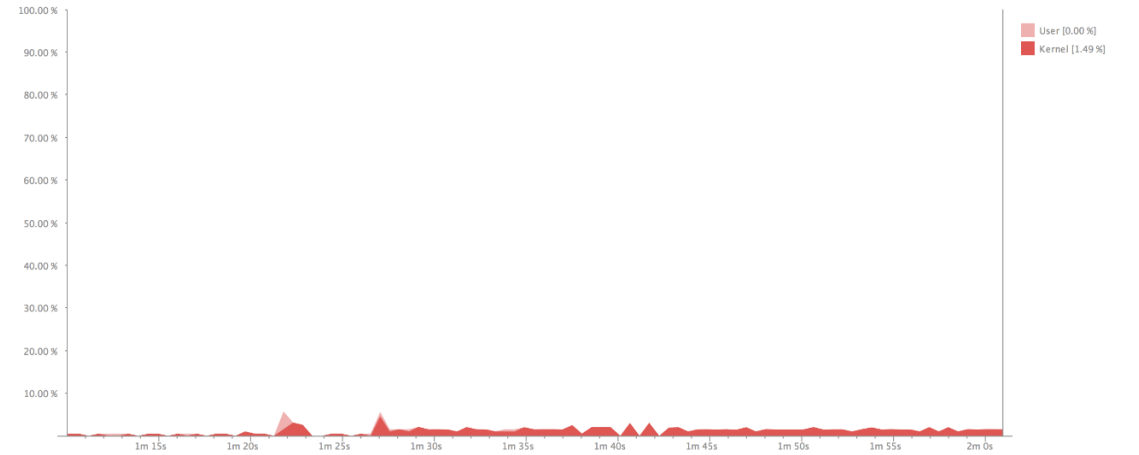


图 5.1 CPU 实时占用

(1) CPU 占用

从图 5.1 可以看出，本文实现的 Android 客户端并不十分消耗系统 CPU 资源，CPU

占用率一直处于 10% 以下。

(2) 内存占用

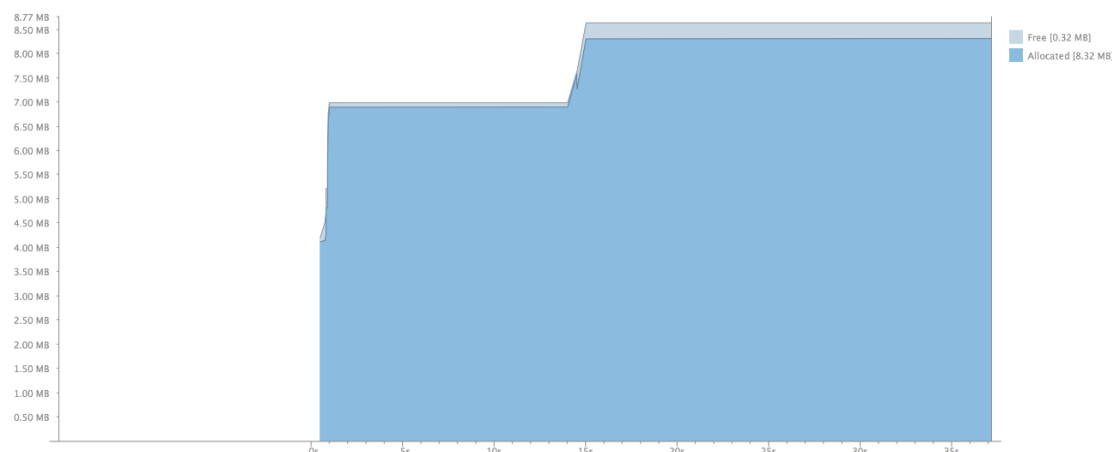


图 5.2 内存实时占用

从图 5.2 可以看出，本文实现的 Android 客户端一般情况下占用内存 6MB 左右，打开图像选择器占用内存 8MB 左右，滑动图像选择器页面没有出现内存抖动，也意味着不会引发频繁的垃圾回收（GC），不会造成系统卡顿。

综合 CPU 占用和内存占用情况，我们可以发现本文实现的 Android 客户端性能良好。

5.2.3 功能测试

本文 Android 客户端功能很多，测试时划分为核心搜索相关功能和其它辅助功能分别进行测试。核心搜索功能又细分为拍照、选择图像、裁剪图像、相似图像列表展示、相似图像详细展示、分享服饰、收藏服饰、保存服饰图像到本地相册、购买服饰、查看服饰信息等功能点。其它辅助功能则细分为分享 APP、给 APP 评分、意见反馈、查看收藏、个人主页等功能点。分别进行黑盒测试，即对按钮进行点击，观察 APP 处理是否正常，数据是否正常。另外本文也测试了 Android 客户端在 2G、3G、4G 和 WIFI 情况下的使用情况，以及无法访问网络时错误提醒是否正常。最终结果是本文实现的 Android 客户端所有功能均可用，效果良好。

5.3 服务器端测试

本文实验为系统集成实验，每次实验均有客户端发起搜索请求，服务器进行搜索，将搜索结果返回给客户端，客户端给予展示，人工评判返回图像结果是否与查询图像是否相似，并统计计算，得出平均正确率。

5.3.1 评价标准

在搜索性能评价标准中比较常用的有正确率、检索率以及前 N 个结果的正确率与检索率。下面简要介绍一下它们的概念和计算方法。

(1) 正确率 (precision) 与检索率 (recall):

在图像检索中, 正确率是指, 搜索结果中相似图像所占比例, 如式 5.1 所示, 分子为相似图像数量, 分母是搜索结果数量。检索率是指搜得的相似图像占图像集中所有相似图像的比例, 分子是搜得的相似图像数量, 分母是图像集中所有相似图像的数量。

$$P = \frac{A}{A+B} \quad (5.1)$$

$$P = \frac{A}{A+C} \quad (5.2)$$

其中, A 表示搜索结果中相似图像数量, B 表示搜索结果中不相似图像数量, A+B 表示搜索结果数量; C 表示图像集中与查询图像相似而未被搜索出来的图像数量, A+C 代表图像集中所有与查询图像相似的图像数量。所以, 正确率是反映了一次搜索的准确性, 而检索率反映的是一次搜索的全面性。

(2) 前 N 个结果的正确率与检索率:

前 N 个结果的正确率就是在返回的前 N 个结果中正确的比例, 定义 R 为某一相关的图像集合, q_i ($q_i \in R$) 为查询图像, 其返回的 N 个结果记为 (I_1, I_2, \dots, I_N) , 则前 N 个结果的正确率可由式 5.3 和式 5.4 计算得到。选取 K 个测试样例进行 K 次实验, 可以得到前 N 个结果的平均正确率, 计算公式如式 5.5。

$$P_N(q_i) = \sum_{j=1}^N \frac{f(I_j, R)}{N} \quad (5.3)$$

$$f(I_i, R) = \begin{cases} 0, & I_i \notin R \\ 1, & I_i \in R \end{cases} \quad (5.4)$$

$$P_N = \sum_{i=1}^k \frac{P_N(q_i)}{K} \quad (5.5)$$

同理, 前 N 个结果的检索率可由式 5.6 计算得到。选取 K 个测试样例进行 K 次实验, 可以得到前 N 个结果的平均检索率, 计算公式如式 5.7。

$$R_N(q_i) = \sum_{j=1}^N \frac{f(I_j, R)}{|R|} \quad (5.6)$$

$$R_N = \sum_{i=1}^k \frac{R_N(q_i)}{K} \quad (5.7)$$

5.3.2 实验结果

本文实验中，搜索性能使用 K 次实验前 N 个结果的平均正确率进行评价。其中， N 有 5 种取值，分别为 5、10、20、30、50， K 取值为 10，图像是否相似由人工主观判定。本文还对使用 GrabCut 与否做了对比实验，检验 GrabCut 对平均正确率的影响。本文图像集中图像数量较多，人工标注所有图像彼此是否相关难以实现，因此本文实验结果并不包含检索率。下面分别介绍各个搜索方式的搜索效果。

(1) 基于颜色特征进行搜索的实验结果



图 5.3 查询图像示例



图 5.4 基于颜色特征的搜索结果示例

图 5.3 和图 5.4 是一次基于颜色特征进行搜索的示例，前者是查询图像，后者是系统返回的前 6 个最相似图像结果。

图 5.5 是系统利用颜色特征进行搜索时的平均正确率。其中，Origin 表示查询图像以及图像集中的图像未经过 GrabCut 过滤背景时的平均正确率，GrabCut 表示查询图像以及图像集中的图像经过 GrabCut 过滤背景时的平均正确率。从图中可以看出，经过 GrabCut 过滤背景之后基于颜色特征的搜索平均正确率范围在 0.37 到 0.54 之间，随着返回搜索结果的数量增加，搜索平均正确率不断下降。经过实验可以得出结论：利用颜色特征进行搜索时，平均正确率良好。使用 Grabcut 过滤背景，平均正确率有所提升。

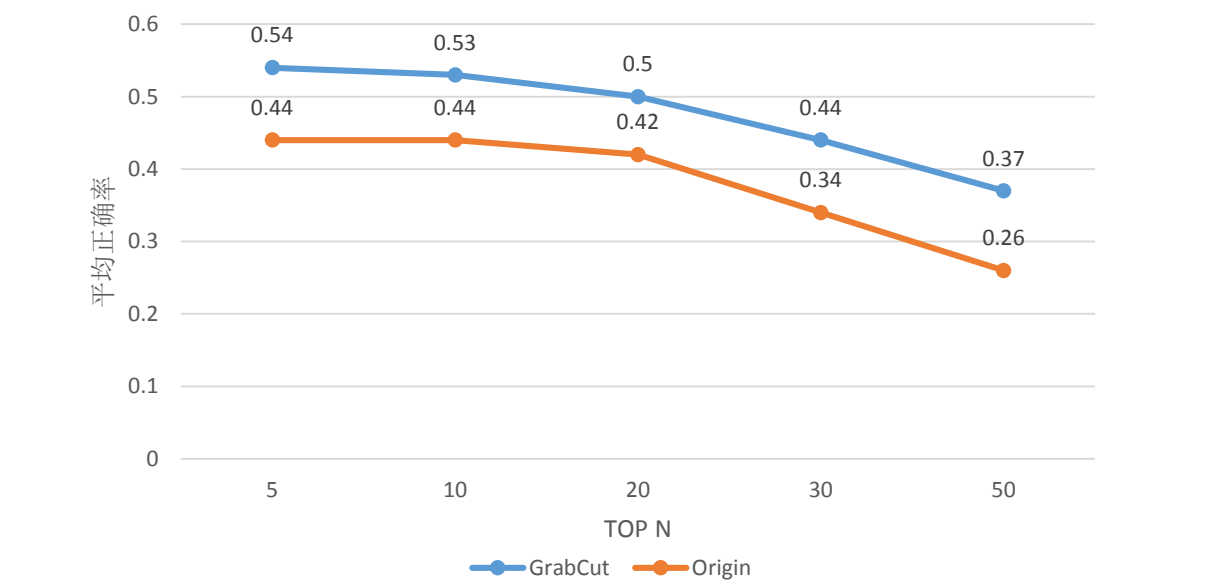


图 5.5 使用颜色特征搜索平均正确率

(2) 基于 Uniform LBP 特征进行搜索的实验结果



图 5.6 查询图像



图 5.7 基于 Uniform LBP 特征的搜索结果

图 5.6 和图 5.7 是一次基于 Uniform LBP 特征进行搜索的示例，前者是查询图像，后者是系统返回的前 6 个最相似图像结果。

图 5.8 是系统利用纹理特征进行搜索时的平均正确率。其中，Origin 表示查询图像以及图像集中的图像未经过 GrabCut 过滤背景时的平均正确率，GrabCut 表示查询图像以及图像集中的图像经过 GrabCut 过滤背景时的平均正确率。从图中可以看出，经过 GrabCut 过滤背景之后基于 Uniform LBP 特征的搜索平均正确率范围在 0.33 到 0.45 之

间，随着返回搜索结果的数量增加，搜索平均正确率不断下降。经过实验可以得出结论：利用 Uniform LBP 特征进行搜索时，平均正确率良好。使用 Grabcut 过滤背景，平均正确率有所提升。

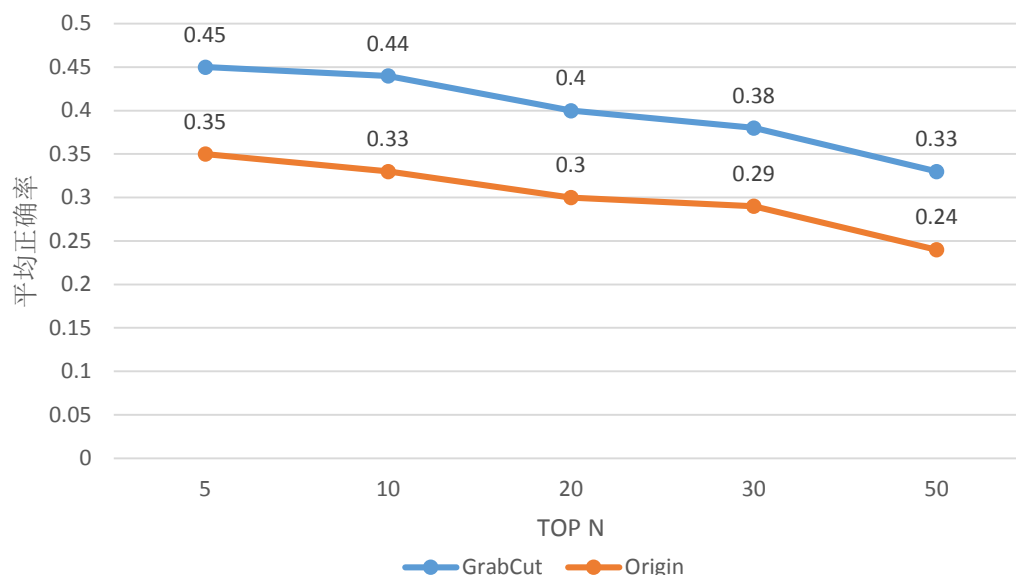


图 5.8 使用纹理特征搜索平均正确率

(3) 基于颜色特征和 Uniform LBP 特征进行搜索的实验结果



图 5.9 查询图像示例



图 5.10 基于颜色特征和 Uniform LBP 特征的搜索结果示例

图 5.9 和图 5.10 是一次基于颜色特征和 Uniform LBP 特征进行搜索的示例，前者是查询图像，后者是系统返回的前 6 个最相似图像结果。

图 5.11 是系统利用颜色特征和 Uniform LBP 特征的组合进行搜索时的平均正确率。其中, Origin 表示查询图像以及图像集中的图像未经过 GrabCut 过滤背景时的平均正确率, GrabCut 表示查询图像以及图像集中的图像经过 GrabCut 过滤背景时的平均正确率。从图 5.11 中可以看出, 经过 GrabCut 过滤背景之后基于颜色特征和 Uniform LBP 特征的组合的搜索平均正确率范围在 0.37 到 0.56 之间, 随着返回搜索结果的数量增加, 搜索平均正确率不断下降。经过实验可以得出结论: 利用颜色特征和 Uniform LBP 特征的组合进行搜索时, 平均正确率良好。使用 Grabcut 过滤背景, 平均正确率有所提升。

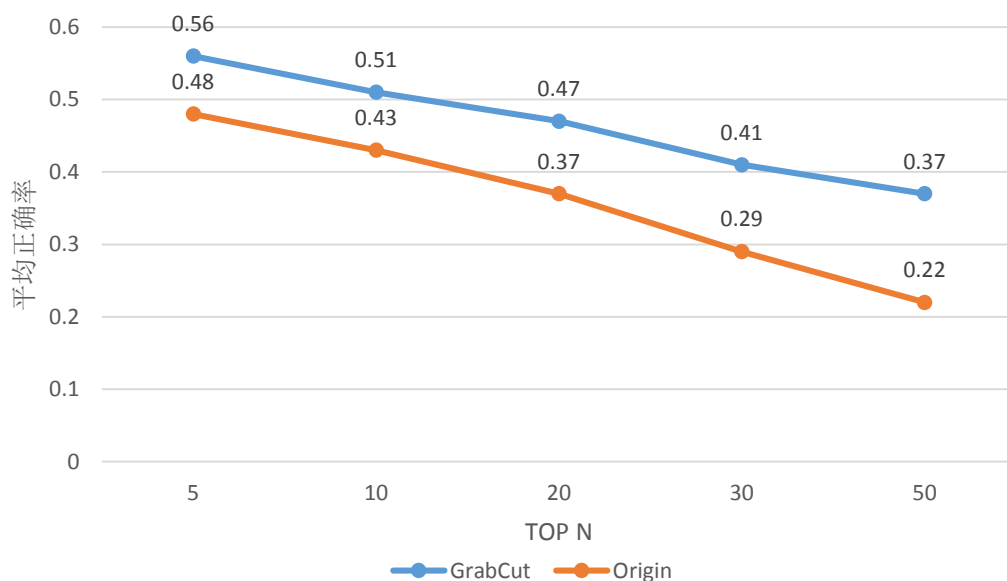


图 5.11 使用颜色特征和 UniformLBP 特征搜索平均正确率

(4) 基于 SIFT 特征进行搜索的实验结果



图 5.12 查询图像示例

图 5.13 基于 SIFT 特征的搜索结果示例

图 5.12 和图 5.13 是一次基于 SIFT 特征进行搜索的示例，前者是查询图像，后者是系统搜索前 6 个最相似的图像结果。

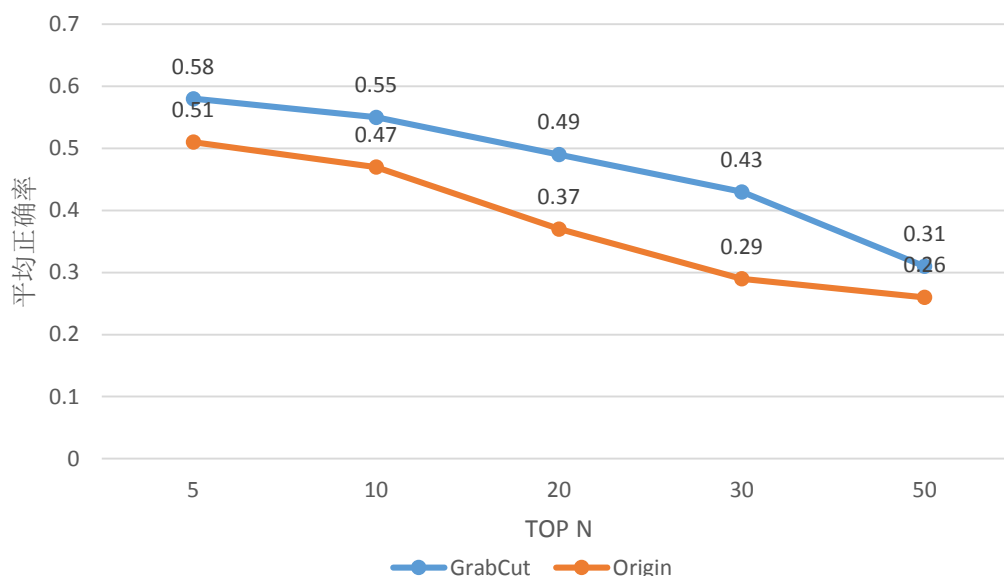


图 5.14 使用 SIFT 特征搜索平均正确率

图 5.14 是系统利用 SIFT 特征进行搜索时的平均正确率。其中，Origin 表示查询图像以及图像集中的图像未经过 GrabCut 过滤背景时的平均正确率，GrabCut 表示查询图像以及图像集中的图像经过 GrabCut 过滤背景时的平均正确率。从图 5.11 中可以看出，经过 GrabCut 过滤背景之后基于 SIFT 特征的搜索平均正确率范围在 0.31 到 0.58 之间，随着返回搜索结果的数量增加，搜索平均正确率不断下降。经过实验可以得出结论：利用 SIFT 特征进行搜索时，平均正确率良好。使用 Grabcut 过滤背景，平均正确率有所

提升。

5.3.3 结果分析

从上一节给出的实验结果中可以看出，就本文的图像集和评价标准而言，在搜索性能上，颜色特征好于 Uniform LBP 特征，颜色特征和 Uniform LBP 特征的组合稍好于颜色特征，SIFT 特征稍好于颜色特征和 Uniform LBP 特征的组合。

6 总结及展望

6.1 论文总结

随着智能手机的普及和电子商务的兴起,手机购物越来越流行。面对海量的商品,传统关键字匹配的商品搜索方式逐渐显现出其局限性。图像作为人们购买商品时重要的参考内容包含的信息量巨大。近年来,在移动端商品搜索中加入图像作为输入受到了越来越多的关注。

本文首先介绍了基于内容的图像检索技术常用的特征,包括颜色特征、纹理特征、局部特征。着重介绍了 HSV 颜色空间下量化过的颜色特征、Uniform LBP 特征、SIFT 特征的优势及其提取算法。详细介绍了 GrabCut 图像分割算法、视觉词袋模型以及倒排索引技术。视觉词袋模型应用于 SIFT 特征上,可以有效的压缩图像的描述。倒排索引技术可以加速基于 SIFT 特征的搜索速度。GrabCut 图像分割算法可以在一定程度上过滤背景,排除干扰信息,提高检索准确度。Android 平台以其开放的优势,在各个领域都得到了广泛的应用。本文实现的服饰图像搜索系统客户端基于 Android 平台开发。Android 客户端共分为界面层、核心层、接口层、模型层四个逻辑层次,以及图像输入模块、网络模块、结果展示模块、其它辅助模块四个功能模块。用户可以通过客户端拍照或者从图库中选择一副服饰图像进行搜索,并以列表展示、详情展示两种方式展示搜索结果,并可在客户端内对服饰进行保存、收藏、分享、购买等操作。本文服务器支持四种搜索方式,分别是基于颜色特征的搜索、基于 Uniform LBP 特征的搜索、基于颜色特征和 Uniform LBP 特征组合的搜索以及基于 SIFT 特征的搜索。前三种搜索方式中,使用欧式距离度量特征之间相似度。搜索主要步骤为计算查询图像与图像集中图像之间的特征距离,使用 TOP K 算法对特征之间距离进行排序,距离最小的 K 幅图像即为相似度最高的 K 幅图像。基于 SIFT 特征的搜索中,首先利用视觉词袋模型离线生成视觉单词本,进而构造倒排索引结构,输入查询图像后,利用倒排索引结构对索引到的图像进行投票,得分最高的 K 幅图像即为相似度最高的 K 幅图像。通过实验证明基于颜色特征和 Uniform LBP 特征组合的搜索以及基于 SIFT 特征的搜索效果良好。

6.2 研究展望

本文实现的系统还有一些不足。本文项目使用的图像集仅有 36960 张图像,远远少于电商平台的服饰图像数量。因此,图像以普通方式存储,特征之间距离的计算和排序

操作计算量也并不大。而在实际应用场景中,图像文件是海量的。海量图像的存储与管理也是图像搜索系统面临的挑战。由于图像集数量有限,本文图像搜索算法在海量图像集上的搜索正确率以及响应时间不能被很好的估计。此外,本文实验采用平均正确率作为搜索性能评价标准。而在平均正确率的计算中,搜索结果图像与查询图像是否相似依靠人工主观判断。首先,人工判断使得实验过程非常耗时;其次,主观差异使得计算得出的平均准确率并不可靠。如果存在已经标注图像之间相似度的图像集,相似度不再需要主观判断,那么实验过程即可实现自动化,计算得出的平均准确率也将更加可靠。本文项目使用 GrabCut 算法进行背景过滤,GrabCut 是基于人机交互的,需要人工标注一定区域为目标或背景。在本文中,针对自己的图像集,依赖其中的图像构图具有规律性,用一个矩形框标注了图像的前景与背景。这种做法适应性差,效果不好。在这种应用背景下,使用无监督图像分割技术进行背景过滤更加恰当。

Android 应用程序开发中,由于移动设备电量、内存等硬件资源有限,性能优化受到广泛关注。在处理、展示图像时,性能优化的要求尤为迫切。同样,网络数据传输的安全性也值得注意,如今网络环境越来越复杂,客户端与服务器的交互安全需要每个系统开发人员关注。响应速度也是衡量用户体验的一个重要指标。网络传输大量图片的技术还需要进一步优化。

致 谢

不知不觉中，论文工作已经接近尾声。这篇论文的完成与一群人的帮助密不可分。

首先要感谢我的导师***教授。感谢她研究生阶段一直以来的关心与指导。

还要特别感谢***老师。这篇论文从选题目、做实验到写文章都在他的悉心指导下完成。不仅在学术上帮助我成长，***老师也教会我很多做人做事的道理。每次与***老师谈话都有很大的收获。我为能被这样一位优秀的老师指导而感到非常幸运。同时，我要感谢南京理工大学。

感谢任良成、张洪伟、张懿鑫三位好友。

感谢朱凌峰、张金康、李宁三位室友，是他们见证了我的成长！

最后要感谢我的爸爸、妈妈、哥哥、姐姐还有外甥女。虽然我的妈妈不知道论文为何物，但她听说我要写五六十页，她就为我骄傲，感谢她一直无条件的爱我。感谢我的爸爸，感谢他一直努力工作，让我可以不为钱发愁，有一个安稳小康的家。感谢我的哥哥，一直默默的关心我的，遇到事情喜欢挡在我的前面的哥哥（虽然有些事情，我觉得我可以处理好的）。感谢我的姐姐，永远把我当小孩儿看的姐姐，感谢这么多年来给我的这么多的红包。感谢我的小外甥女，她马上两周岁了，感谢她学会的第三个词就是舅舅，感谢她给我们带来那么多的快乐。感谢他们，因为有他们，我感到幸福。

参考文献

- [1] 梁玲玲. 面向购物搜索的目标提取算法研究及系统实现[D]. 西南交通大学, 2012.
- [2] Boykov Y Y, Jolly M P. Interactive graph cuts for optimal boundary & region segmentation of objects in ND images[C]//Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on. IEEE, 2001, 1: 105-112.
- [3] Wang Z, Stavrou A. Exploiting smart-phone usb connectivity for fun and profit[C]//Proceedings of the 26th Annual Computer Security Applications Conference. ACM, 2010: 357-366.
- [4] Schlegel R, Zhang K, Zhou X, et al. Soundcomber: A Stealthy and Context-Aware Sound Trojan for Smartphones[C]//NDSS. 2011, 11: 17-33.
- [5] 刘昌平, 范明钰, 王光卫, 等. Android 手机的轻量级访问控制倡[J]. 计算机应用研究, 2010, 27(7).
- [5] 陈莉君. Linux 内核的分析及应用[J]. 西安邮电学院学报, 2001 (1): 17-20.
- [6] 王飞, 罗东礼, 汤井田, 等. 一种基于图的交互式目标分割算法[J]. 计算机工程与应用, 2006, 42(24): 65-67.
- [7] 邹大海. 人体图像中周边物品检测分类技术研究[D]. 南京邮电大学, 2013.
- [8] 常朝稳, 徐江科. 终端行为可信评估及其访问控制方法研究[J]. 小型微型计算机系统, 2014, 3: 014.
- [9] 邓先平. 交互式服饰图像检索研究及系统实现 [D][D]. 西南交通大学, 2013.
- [10] 陈丹伟, 黄秀丽, 任勋益. 云计算及安全分析[J]. 计算机技术与发展, 2010 (2): 99-102.
- [11] Kal é L V. PARALLEL PROBLEM SOLVING![J]. Parallel Algorithms for Machine Intelligence and Vision, 2012, 51: 146.
- [12] Chandrasekhar V, Takacs G, Chen D, et al. Chog: Compressed histogram of gradients a low bit-rate feature descriptor[C]//Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on. IEEE, 2009: 2504-2511.
- [13] 杭燕, 杨育彬. 基于内容的图像检索综述[J]. 计算机应用研究, 2002, 19(9): 9-13.
- Lowe D G. Distinctive image features from scale-invariant keypoints[J]. International journal of computer vision, 2004, 60(2): 91-110.
- [14] 张骞. 基于文本的与基于内容的图像检索技术比较研究[J]. 情报探索, 2012, 1: 040.

- [15] 王璇. 基于内容图像检索中的对象提取与检索算法研究[D]. 北京工业大学, 2013.
- [16] Jiang Y G, Ngo C W, Yang J. Towards optimal bag-of-features for object categorization and semantic video retrieval[C]//Proceedings of the 6th ACM international conference on Image and video retrieval. ACM, 2007: 494-501.
- [17] Flickner M, Sawhney H, Niblack W, et al. Query by image and video content: The QBIC system[J]. Computer, 1995, 28(9): 23-32.
- [18] Jiang Y G, Ngo C W, Yang J. Towards optimal bag-of-features for object categorization and semantic video retrieval[C]//Proceedings of the 6th ACM international conference on Image and video retrieval. ACM, 2007: 494-501.
- [19] Diaz A. Through the Google goggles: Sociopolitical bias in search engine design[M]. Springer Berlin Heidelberg, 2008.
- [20] Van der Bruggen L J J, Lenstra J K, Schuur P C. Variable-depth search for the single-vehicle pickup and delivery problem with time windows[J]. Transportation Science, 1993, 27(3): 298-311.
- [21] 吴培君, 漆涛. android 系统上权限提升的安全增强框架[J]. 2013.
- [22] 任桂超, 丁丽萍, 贺也平. 带有空间上下文信息的细粒度 Android 安全强化机制的设计[J]. 计算机应用与软件, 2014, 31(5): 285-290.
- [23] 差沙. 用 Android 开发手机应用[J]. 程序员, 2008 (1): 56-61.
- [24] Siddiquie B, Feris R S, Davis L S. Image ranking and retrieval based on multi-attribute queries[C]//Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on. IEEE, 2011: 801-808.
- [25] Liu G, Lin Z, Yu Y. Robust subspace segmentation by low-rank representation[C]//Proceedings of the 27th international conference on machine learning (ICML-10). 2010: 663-670.
- [26] Hasan B, Hogg D. Segmentation using Deformable Spatial Priors with Application to Clothing[C]//BMVC. 2010: 1-11.
- [27] Lampert C H, Nickisch H, Harmeling S. Learning to detect unseen object classes by between-class attribute transfer[C]//Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on. IEEE, 2009: 951-958.
- [28] Lin Z, Chen M, Ma Y. The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices[J]. arXiv preprint arXiv:1009.5055, 2010.
- [29] Pan S J, Yang Q. A survey on transfer learning[J]. Knowledge and Data Engineering, IEEE Transactions on, 2010, 22(10): 1345-1359.

- [30] Kaelbling L P, Littman M L, Moore A W. Reinforcement learning: A survey[J]. Journal of artificial intelligence research, 1996: 237-285.
- [31] Song Z, Wang M, Hua X, et al. Predicting occupation via human clothing and contexts[C]//Computer Vision (ICCV), 2011 IEEE International Conference on. IEEE, 2011: 1084-1091.
- [32] Wang N, Ai H. Who blocks who: Simultaneous clothing segmentation for grouping images[C]//Computer Vision (ICCV), 2011 IEEE International Conference on. IEEE, 2011: 1535-1542.
- [33] Liu S, Song Z, Liu G, et al. Street-to-shop: Cross-scenario clothing retrieval via parts alignment and auxiliary set[C]//Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on. IEEE, 2012: 3330-3337.
- [34] Kalantidis Y, Kennedy L, Li L J. Getting the look: clothing recognition and segmentation for automatic product suggestions in everyday photos[C]//Proceedings of the 3rd ACM conference on International conference on multimedia retrieval. ACM, 2013: 105-112.
- [35] Bossard L, Dantone M, Leistner C, et al. Apparel classification with style[M]//Computer Vision—ACCV 2012. Springer Berlin Heidelberg, 2013: 321-335.
- Jammalamadaka N, Minocha A, Singh D, et al. Parsing clothes in unrestricted images[C]//Proc. of British Machine Vision Conference. 2013.
- [36] Wright J, Yang A Y, Ganesh A, et al. Robust face recognition via sparse representation[J]. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2009, 31(2): 210-227.
- [37] Yang M, Yu K. Real-time clothing recognition in surveillance videos[C]//Image Processing (ICIP), 2011 18th IEEE International Conference on. IEEE, 2011: 2937-2940.
- [38] Yang Y, Ramanan D. Articulated pose estimation with flexible mixtures-of-parts[C]//Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on. IEEE, 2011: 1385-1392.
- [39] Felt A P, Chin E, Hanna S, et al. Android permissions demystified[C]//Proceedings of the 18th ACM conference on Computer and communications security. ACM, 2011: 627-638.
- [40] Nauman M, Khan S, Zhang X. Apex: extending android permission model and enforcement with user-defined runtime constraints[C]//Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security. ACM, 2010: 328-332.

附 录 A

攻读硕士学位期间参加的科学研究情况：

申请专利《一种基于移动终端的服饰搜索方法》，申请号：201510624226.3

附录 B

B.1 上传图像文件关键源代码 (Android)

```
...
URL url = new URL(actionUrl);
con = (HttpURLConnection) url.openConnection();

/* 允许 Input、Output, 不使用 Cache */
con.setDoInput(true);
con.setDoOutput(true);
con.setUseCaches(false);

/* 设置传送的 method=POST */
con.setRequestMethod("POST");

/* setRequestProperty */
con.setRequestProperty("Connection", "Keep-Alive");
con.setRequestProperty("Charset", "UTF-8");
con.setRequestProperty("Content-Type",
    "multipart/form-data;boundary=" + boundary);

ds = new DataOutputStream(con.getOutputStream());
ds.write(jsonSB.toString().getBytes());

if (imageNames != null && bitmaps != null) {
    for (int i = 0; i < imageNames.size(); i++) {
        String picName = imageNames.get(i);
        Bitmap bitmap = bitmaps.get(i);

        ds.writeBytes(twoHyphens + boundary + end);
        ds.writeBytes("Content-Disposition: form-data; "
            + "name=\"upload_image\";filename=\"" + picName + "\""
            + end);
    }
}
```

```

        ds.writeBytes(end);
        ds.write(bitmap2Bytes(bitmap));
        ds.writeBytes(end);
        ds.writeBytes(twoHyphens + boundary + twoHyphens + end);
    }
}

```

```
ds.flush();
```

```
/* 取得 Response 内容 */
```

```
...
```

B.2 提取颜色特征关键源代码 (Python)

```

def color_feature(image_path):
    try:
        img = Image.open(image_path)
    except Exception, exception:
        print exception
        return []
    except IOError, error:
        print error
        return []

    w1 = [0]*72
    for_show = []
    for count, (r, g, b) in img.getcolors(img.size[0]*img.size[1]):
        h, s, v = colorsys.rgb_to_hsv(r/255.0, g/255.0, b/255.0)
        h = int(h*360)

        H, S, V = compress_hsv(h, s, v)

        W = 9*H + 3*S + V
        assert W < 72
        w1[W] = w1[W] + count

```

```

        for i in xrange(count):
            for_show.append(W)

```

```

    wl = normalize(wl)
    return wl

```

B.3 Uniform LBP 特征提取关键源代码 (Python)

```

def ulbp(img_path):
    try:
        img = cv2.imread(img_path, 0)
    except Exception, exception:
        print exception
        return []
    except IOError, error:
        print error
        return []

    uniform_table =
    [0,1,2,3,4,58,5,6,7,58,58,58,8,58,9,10,11,58,58,58,58,58,58,12,58,58,58,13,58,
    14,15,16,58,58,58,58,58,58,58,58,58,58,58,58,17,58,58,58,58,58,58,18,
    58,58,58,19,58,20,21,22,58,58,58,58,58,58,58,58,58,58,58,58,58,58,58,58,
    58,58,58,58,58,58,58,58,58,58,23,58,58,58,58,58,58,58,58,58,58,58,58,
    58,58,24,58,58,58,58,58,58,25,58,58,58,26,58,27,28,29,30,58,31,58,58,58,32,58,
    58,58,58,58,58,33,58,58,58,58,58,58,58,58,58,58,58,58,34,58,58,58,58,
    58,58,58,58,58,58,58,58,58,58,58,58,58,58,58,58,58,58,58,58,58,58,58,
    58,35,36,37,58,38,58,58,39,58,58,58,58,58,58,40,58,58,58,58,58,58,58,
    58,58,58,58,58,41,42,43,58,44,58,58,45,58,58,58,58,58,46,47,48,58,49,
    58,58,58,50,51,52,58,53,54,55,56,57]

    ulbp_feature = [0]*59
    if img is None:
        return []

    # loop each point
    for x in xrange(1, len(img)-1):
        for y in xrange(1, len(img[0])-1):
            center = img[x][y]
            lbp = 0

```

```

    if center <= img[x-1][y-1]:
        lbp = lbp + 1
    if center <= img[x-1][y]:
        lbp = lbp + 2
    if center <= img[x-1][y+1]:
        lbp = lbp + 4
    if center <= img[x][y+1]:
        lbp = lbp + 8
    if center <= img[x+1][y+1]:
        lbp = lbp + 16
    if center <= img[x+1][y]:
        lbp = lbp + 32
    if center <= img[x+1][y-1]:
        lbp = lbp + 64
    if center <= img[x][y-1]:
        lbp = lbp + 128
    unlbp_id = uniform_table[lbp]
    ulbp_feature[unlbp_id] = ulbp_feature[unlbp_id]+1

    return normalize(ulbp_feature)

```

B.4 快捷分享、快捷收藏手势操作中隐藏头部视图和底部视图的关键源代码（Android）

```
@Override
```

```
protected void onLayout(boolean changed, int l, int t, int r, int b) {
    int width = r - l;
    headerView.layout(0, t - headerView.getMeasuredHeight(), width, t);
    contentView.layout(0, t, width, b);
    footerView.layout(0, b, width, b + footerView.getMeasuredHeight());
}

```

```
@Override
```

```
protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
    super.onMeasure(widthMeasureSpec, heightMeasureSpec);
    contentView.measure(MeasureSpec.makeMeasureSpec(getMeasuredWidth(),
        MeasureSpec.EXACTLY), MeasureSpec.makeMeasureSpec(getMeasuredHeight(),
        MeasureSpec.EXACTLY));
}

```

B.5 快捷分享、快捷收藏手势操作中对触摸事件的处理的关键源代码（Android）

```
private void handleTouchEvent(MotionEvent event) {  
    if (downY == -1) {  
        downY = event.getY();  
    }  
    if (lastY == -1) {  
        lastY = event.getY();  
    }  
    float dy = event.getY() - lastY;  
    lastY = event.getY();  
    switch (event.getAction()) {  
        case MotionEvent.ACTION_DOWN:  
            break;  
        case MotionEvent.ACTION_MOVE:  
            if (scrolling) {  
                float calibratedTotalDy = (event.getY() - downY) * SCROLL_RATIO;  
                scrollTo(0, (int) -calibratedTotalDy);  
            } else {  
                float totalDy = event.getY() - downY;  
                if (Math.abs(totalDy) > touchSlop) {  
                    scrolling = true;  
                    float calibratedTotalDy = totalDy * SCROLL_RATIO;  
                    scrollTo(0, (int) -calibratedTotalDy);  
                }  
            }  
            updateHeaderFooterView();  
            break;  
        case MotionEvent.ACTION_UP:  
        case MotionEvent.ACTION_CANCEL:  
            if (getScrollY() >= footerView.getMeasuredHeight()) {  
                if (pullListener != null) {  
                    pullListener.onPullUp();  
                }  
            } else if (getScrollY() <= -headerView.getMeasuredHeight()) {
```

```
        if (pullListener != null) {  
            pullListener.onPullDown();  
        }  
    }  
    smoothScrollTo(0);  
    lastY = -1;  
    downY = -1;  
    break;  
}  
}
```