

华中科技大学

本科生毕业设计[论文]

基于 iOS 的头像萌化系统的设计与实现

院 系 电子信息与通信学院

专业班级 通信工程 1106 班

姓 名 罗显扬

学 号 U201113430

指导教师 许炜 副教授

2015 年 5 月 30 日

学位论文原创性声明

本人郑重声明：所呈交的论文是本人在导师的指导下独立进行研究所取得的
研究成果。除了文中特别加以标注引用的内容外，本论文不包括任何其他个人或
集体已经发表或撰写的成果作品。本人完全意识到本声明的法律后果由本人承担。

作者签名： 年 月 日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保障、使用学位论文的规定，同意学校保
留并向有关学位论文管理部门或机构送交论文的复印件和电子版，允许论文被查
阅和借阅。本人授权省级优秀学士论文评选机构将本学位论文的全部或部分内
容编入有关数据进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学
位论文。

本学位论文属于 1、保密口，在 年解密后适用本授权书

2、不保密口 。

（请在以上相应方框内打“√”）

作者签名： 年 月 日

导师签名： 年 月 日

摘 要

随着移动互联网的飞速发展,出现了很多优秀的社交软件,而每个用户在使用这些软件的时候都需要设置头像。如何选择一個满意的头像成为了用户需要解决的问题。目前,有很多卡通头像软件正在流行,但这些软件的操作都较为繁琐,需要用户手动地选择人脸的各个部位。本文针对如何快速便捷地生成卡通头像做出了探索。

针对上面的问题,本文开展了人脸特征提取以及特征分类的研究工作,设计了一个基于 iOS 的头像萌化系统。本系统能对眼睛、眉毛、嘴巴等面部特征进行分类。首先,需要从网络上选取大量自拍照来构成图片库用于 SVM 模型的训练。然后使用人脸检测技术标定图片上人脸的五官位置。随后,使用 LBP、边缘检测等方法提取面部器官的纹理特征,使用面部定位点提取其几何特征。然后使用 SVM 对特征进行分类,识别出五官的类别。最后,选取与五官类别对应的图片,拼接成一张卡通头像。测试结果表明,本系统能对五官进行大致分类,生成的卡通头像也与原图的相似性也能令人满意。

本文针对如今头像软件操作复杂的问题,开发了头像萌化系统。进行了人脸特征提取和 SVM 分类等工作,能够生成和用户自拍照相似的卡通头像,可以为用户选择头像提供帮助。

关键词: 人脸坐标定位; 人脸特征提取; 支持向量机

Abstract

As the Internet developing rapidly, a lot of social softwares have become popular. Every software user have to set a profile photo. But how to set a satisfied photo has become a problem. Nowadays, there are many software helping users to create a profile photo. But these softwares have a lot of steps to do when creating a profile photo. User has to choose every facial feature. This paper explore the method of creating a profile photo quickly.

To solve the problems above, this paper designs a picture sprouting system, based on iOS platform, and carrys out research on human facial feature extraction and feature classification. The system is capable of facial features eyes, eyebrows, mouth and other classification. First, I need to select photos from the Internet to form a dataset for SVM training. Then using the face detection technology to get the landmarks of face. Subsequently, using LBP or edge detection method to get the texture feature, or using the landmarks to get the geometric features. Then using SVM to classify features. At last, selecting the features category corresponding picture to form a cartoon profile photo. The test results show that the system can broadly classified features, and the profile photo is satisfying.

To solve the operating complex issues of profile photo software, this paper develops a picture sprouting system. I conducted face feature extraction and SVM classification work and then forming a similar profile photo and helping user to pick a profile photo.

Keyword: Face coordinate positioning; Facial feature extraction; Support vector machine

目录

摘 要	I
Abstract	II
1. 绪论	1
1.1. 课题研究背景	1
1.2. 国内外相关研究现状	2
1.3. 课题来源及内容	3
1.4. 论文组织结构	3
2. 技术背景综述	5
2.1. LibSVM 工具简介	5
2.2. OpenCV 简介	9
2.3. LBP 简介	9
2.4. Xcode 与 Objective-C	11
3. 需求分析与功能设计	15
3.1 系统需求分析	15
3.2 系统架构设计	16
3.3 功能模块设计	17
3.4 系统界面设计	21
3.5 本章小结	22
4. 系统实现与测试	24
4.1 软硬件环境	24
4.2 人脸特征点定位模块	24
4.3 人脸特征的提取与建模	26
4.4 SVM 分类模块的实现	37
4.5 系统的整体实现	38
4.6 系统测试与结果分析	39
4.7 本章小结	42
5. 总结与展望	43
5.1 现有工作总结	43
5.2 未来方向展望	43
致谢	44
参考文献	45

1. 绪论

1.1. 课题研究背景

自从上世纪 60 年代互联网产生以来, 互联网一直在高速发展。而互联网的发展也经历了几个阶段, 首先是在美国军事系统和大学中被使用, 然后逐步走进平民的生活中, 以 PC 互联网的形式存在。而发展到现在, PC 互联网已经趋于饱和, 正在井喷式发展的是移动互联网。

仅在中国, 截至 2014 年 4 月, 移动互联网用户在我国的数量已经达到了 8.4 亿, 占总移动电话用户的 70%^[1]。比起 PC 互联网几十年的发展, 移动互联网仅用十年左右的时间就达到如此惊人的渗透率。伴随着移动互联网的发展, 社交网络也迎来了颠覆性的转变^[2]。自从 2008 年以后, 社交网络就开始加速发展, 用户的规模在不断增大, 其价值也在不断攀升。仅仅过了五年, 中国就有一半以上的网民通过移动互联网进行联系以及信息分享。

伴随着移动互联网和社交网络的发展, 出现了像微信、微博、人人这类改变人们生活习惯和生活方式的颠覆性软件。例如 2011 年上线的微信, 仅仅过了 4 年时间, 就让用户发送短信的数量急剧减少, 短信甚至沦为验证码收件箱。说明移动互联网和社交网络的发展对传统电信行业的打击是巨大的。

在社交网络中, 每个人都有自己的头像。用户可以使用真实的照片, 也可以使用卡通图案。2014 年 5 月底的时候, 一款叫做脸萌的软件出现在朋友圈中。这是一个由 90 后年轻团队打造的产品, 据统计, 脸萌上线 6 个月后, 用户就达到了 2000 万, 火爆的时候仅在五天内脸萌在 iOS 和 Android 两大平台的下载量就突破了 500 万, 并顺利登上排行榜榜首。这体现了移动社交的强大。

脸萌的成功表明用户对头像类的软件有兴趣。经过对脸萌的研究, 发现其仍然有缺点: 需要手动选择脸型、发型、眼睛、鼻子、嘴巴等人脸特征, 而每个特征又需要从几十种图片中选择, 这样增加了生成头像的复杂性。针对这个问题, 本课题提出使用人脸识别技术和人脸特征点定位技术, 根据用户的自拍照, 对其面部的特征进行分类, 比如眉毛的浓淡、眼睛的大小、嘴唇的厚薄、头发的长短, 然后自动地生成一张卡通头像。本系统相对于脸萌的优点是操作简单, 仅仅需要用户上传自拍照即可, 且生成的卡通头像的准确率也能满足用户的需求。将人脸识别和头像软件结合在一起, 这将是一个非常具有创新的应用。

1.2.国内外相关研究现状

本课题中对人脸面部特征的提取包含两个方面的工作：人脸检测、面部特征提取。其中人脸检测技术在近十几年来已经发展地较为成熟，并已经在很多地方切实得到应用^[3]。该技术从 20 世纪 60 年代末开始发展，至今取得了非常丰硕的研究成果。早在 1888 年和 1910 年 Galton 就在《Nature》发表了两篇关于使用人脸识别技术来进行身份认证的文章^[4]。近年来，在人脸识别领域已经涌现出了很多新的技术方法，取得了长足的发展^[5]。

在得到面部检测数据之后，就需要对面部区域进行特征提取。特征提取就是获取面部各个部位的分类信息。本课题主要进行的工作就是提取面部特征并对其进行分类。

在过去的研究中，已经出现了很多特征提取的方法，常见的有两大类：基于几何的方法和基于统计的方法^[6]。基于几何的方法可以直接使用面部的坐标点来提取人脸特征，譬如眼睛的大小；而基于统计的方法需要大量的训练，而人脸图像的维数非常高，这样就导致该方法的速度和鲁棒性不好，因此，需要首先对数据进行降维。下面介绍几种常见的特征提取方法。

主成分分析（Principal component analysis, PCA）是一种数学计算方法，目的是对数据进行降维。Kirby 和 Sirovich 最先提出了面部数据的降维方法^[7]，并使用 PCA 方法来对面部特征进行降维。M.Turk 针对传统 PCA 的不足提出了改进的方法^[8]。如 NFL 方法^[9]、多向量线性方法等^[10]。

A.L.Yuille 等提出了基于模板的特征提取方法^[11]。由于不同人脸的特征差异较大，且结果很复杂，系统很难使用固定的模板来确定每个人的特征，因此，采用可变模板的方法就具有很大的优势。

主动形状模型（Active Shape Model, ASM）是由 T.F.Cootes 等人提出的基于统计模型的方法。该方法是分析描述目标轮廓特征的形状向量，然后建立一个能反应目标形状变换规律的模型。ASM 相比于其他方法，有着高效率、低成本、容易扩展到三维图像的优势^[12]。ASM 方法已经在特征提取、医学图像分割等领域得到广泛应用。

而对于头像类的软件，国内外有很多成熟且成功的产品。和脸萌相似的有 MakeMe，能快速制作专属的 2D 头像，拥有简约而直观的界面、细分的栏目以

及 1000 多个各式各样的配饰。iMadeFace 也曾长时间占据排行榜第一名，它融合波普艺术创作理念，将用户自己或者他人的样貌作为创作的母体，绘制出具波普风格的头像，它能轻松甩动就获得意想不到的随机面部组合。而 Dots 可以把用户的照片改成像素版的照片，不过操作较为繁琐，需要用户逐个像素点去调整颜色，但是制作出来的效果非常好。而 Bobbleshop 是一个摇头头像制作的软件，优势是可以制作全身的样子，拥有成千上万的漂亮的编辑区域，比同类的软件都多，甚至可以选择皱纹、眼部或嘴部的阴影。

从头像软件的国内外研究现状，可以得到的结论是，没有一个软件会与人脸识别技术结合起来，所以本课题将在这一点上有所创新，将人脸识别技术做到头像软件中。

1.3.课题来源及内容

本课题来源是华中科技大学电子信息与通信学院互联网中心的自拟项目。课题的目标是生成一个在 iOS 平台运行的 APP，能根据用户的自拍照，自动生成和用户相似的卡通头像。其实现流程是使用手机的摄像头拍摄自拍照或选择本地照片之后，使用训练好的 SVM 模型，来对人脸的各个部分的特征进行判断，然后根据特征自动生成类似脸萌的卡通头像，并保证一定的准确度。因此，本课题主要有如下工作内容：

- 1) 从网上选取自拍照构成自拍图片库，供系统训练；
- 2) 使用人脸识别服务器提供的 API，能成功识别人脸，并得到面部特征坐标；
- 3) 根据自拍图片库训练生成 SVM 模型；
- 4) 使用 SVM 模型预测的标签生成卡通头像。

1.4.论文组织结构

本文共分为五章，其余各章节的内容安排如下

第二章简要介绍了本系统开发过程中涉及的关键技术，包括 LibSVM 工具、OpenCV 计算机视觉库、LBP 局部二值特征、Xcode 及使用到的第三方库。

第三章主要介绍了本系统的需求分析与功能设计。首先对系统进行需求分析，

然后对整个系统的架构进行设计，随后对各个模块的功能进行设计，最后对系统的界面进行了设计。

第四章主要介绍了本系统的实现方法以及测试结果。首先介绍了人脸特征点定位模块，然后介绍了各个特征的提取和建模过程，随后介绍了 SVM 分类模块的实现以及系统的整体实现，最后对整个系统进行测试。

第五章对本系统的成果和贡献做出总结，并提出了还需要解决的问题，并对系统可以优化的方向做出了展望。

2. 技术背景综述

本课题研究的基于 iOS 的头像萌化系统是利用 LibSVM 进行特征预测, 使用 OpenCV 进行图片处理, 基于 Xcode 开发环境, 利用 Objective-C 编程语言予以实现。因此, 本章介绍开发本系统的基本工具, 即 SVM; 然后介绍了图像处理所用到的计算机视觉库 OpenCV; 接着介绍了处理纹理特征时需要用到的算法 LBP; 最后对开发环境 Xcode 和开发语言 Objective-C 进行简略说明。

2.1.LibSVM 工具简介

2.1.1. SVM 简介

支持向量机 (Support Vector Machine, SVM) 是一个在机器学习领域一个有监督的学习模型, 通常用来进行模式识别、分类以及回归分析^[13]。SVM 为了实现样本线性可分的目的, 使用非线性映射算法, 将在低维空间线性不可分的样本转换到高维特征空间使其线性可分, 从而使得高维特征空间可以采用线性算法对样本的非线性特征进行线性分析^[14]。从低维空间到高维空间的转换如图 2-1 所示

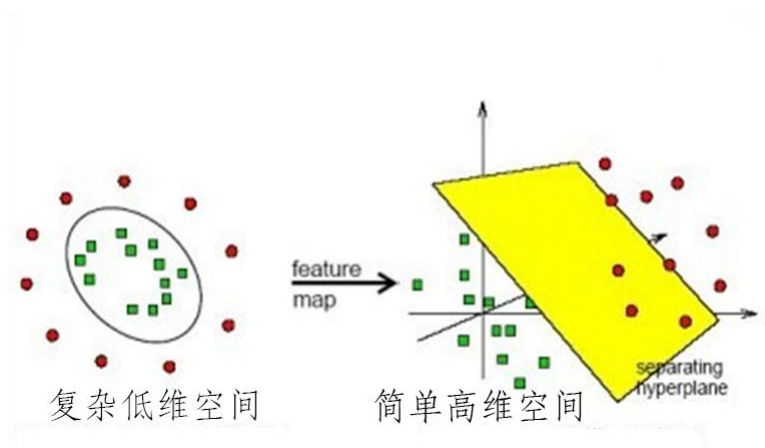


图 2-1 SVM 将低维空间转换到高维空间的转换示意图

将样本转换到高维之后, SVM 就会去求解最优的分类面来将不同的特征分开^[15]。最优分类面使得两个点集到此平面的最小距离最大, 即两个点集中的边缘点到此平面的距离最大^[16]。在图 2-2 中, H 面即是最优分类面, H 是线性平面, 这样就可以在高维特征空间进行线性分割计算。

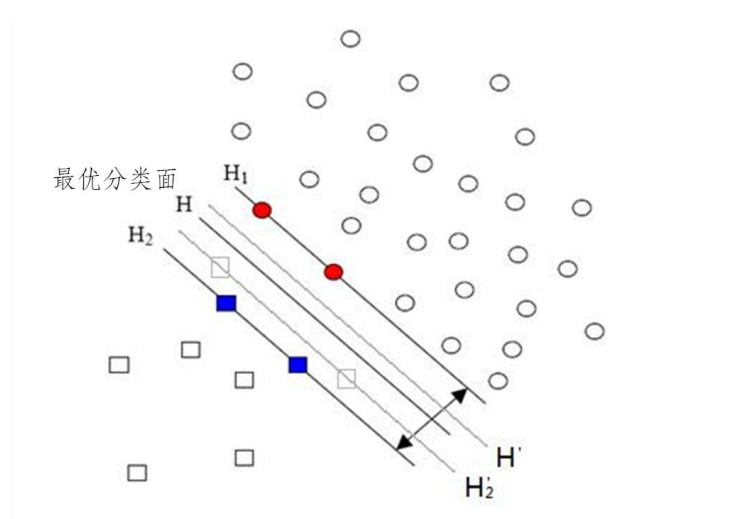


图 2-2 SVM 最优分类面

将样本向高维映射之后会导致运算的复杂性,甚至可能导致维数灾难,但是 SVM 通过引入核函数,应用核函数的展开定理,就不需要知道非线性映射的显性表达式,这样就能够很巧妙地解决高维空间中的内积运算^[17]。核函数的选择对于一个 SVM 的性能的好坏十分关键。选择核函数有两个部分的工作:

- 1) 选择核函数的类型
- 2) 确定核函数类型之后选择与核函数相关的参数

使用 SVM 工具的一个难点就是,如何根据数据的情况选择核函数。目前有以下三种核函数较为常用:

- 1) 多项式核函数

$$K(x,y) = [r(x * y) + coef]^q \quad \text{公式 2-1}$$

q 是该多项式核函数的阶数,当 q 等于 1 时,该核函数就是线性核函数

- 2) Sigmoid 核函数

$$K(x,y) = \tanh[r(x * y) + coef] \quad \text{公式 2-2}$$

使用该核函数, SVM 可实现一个隐层的多层感知神经网络

- 3) 径向基函数

$$K(x,y) = \exp(r||x - y||^2) \quad \text{公式 2-3}$$

使用该核函数的 SVM 是一个径向基函数分类器^[18]

2.1.2. LibSVM 工具

LibSVM (A Library for Support Vector Machine) 是易于使用以及高效的 SVM 模式识别和回归工具。拥有 C、Matlab、Java 等数十种语言的版本, 本课题中采用其中的 Matlab 版。

LibSVM 的使用步骤很简单, 如下

- 1) 准备训练数据集, 其格式要遵循 LibSVM 工具要求的格式
- 2) 为了更好的效果, 对数据集进行简单的缩放(归一化)
- 3) 根据数据集的特征来考虑选用核函数的类型
- 4) 通过交叉验证选择最佳的 c 和 g 参数
- 5) 使用 c 和 g 参数和训练样本来训练, 生成 SVM 模型
- 6) 使用 SVM 模型进行测试和预测

以上步骤中用到的两个最重要的函数就是 `svmtrain()` 和 `svmpredict()`, 有时也会用到 `svmscale()`。

1) `svmtrain()`

此函数使用训练数据集和参数来训练, 得到 SVM 模型^[19]。通过参数寻优得到最优的 c 和 g 之后, 就可以调用此函数。

此函数用法是 `svmtrain [options] training_set_file [model_file]`

其中 options 是 LibSVM 的参数设置, 主要参数如下

- -s 设置 SVM 的类型
- -t 设置核函数类型
- -d 设置核函数中的 degree 值
- -g 设置核函数中的 gamma 值
- -v 设置 n 折交叉验证模式

training_set_file 是输入到 SVM 的训练集, model_file 是输出的模型文件

2) `svmpredict()`

此函数使用训练好的 SVM 模型文件, 对输入的预测数据进行预测, 输出判断的标签^[20]。

此函数用法是 `[predict_label, accuracy, decision_values/prob_estimates] = svmpredict(test_label, test_matrix, model)`

其中参数含义如下

- -test_label 测试标签
- -test_matrix 测试数据
- Model 训练好的 SVM 模型

3) svmScale()

此函数的作用是对原样本的数据进行缩放, 缩放到[0,1]或者是[-1,1]的范围, 缩放的目有如下两点

- 以防某一个特征值过大或过小, 导致训练得到的模型不准确
- 过大或过小的特征值在 exp 运算或者内积运算时会造成计算缓慢, 缩放后可以加快计算速度

2.1.3. 参数寻优

参数在 SVM 的训练过程中非常重要, 参数的差异会导致训练生成的模型的预测精度发生变化^[21]。因此, 寻找最优的参数是使用 SVM 过程中一项重要的工作^[22]。

本课题中使用交叉验证来寻找最优的参数。交叉验证的流程是从给定的样本中, 选出大部分数据来进行建模, 小部分数据进行预测以及误差计算, 并记录误差的平方加和。完成一次运算后, 就换一部分样本进行预测, 直到所有的样本都被预测且仅被预测了一次。根据每个样本的平方加和, 来判断参数是否合适。

常见的交叉验证有三种, 如下

1) Holdout 验证

随机从原始样本中选取少于三分之一的数据构成验证数据, 剩余的数据构成训练数据, 进行训练和预测。

2) K 折交叉验证

将原始样本分为 K 份, 其中一份作为验证数据, 其余 K-1 份构成训练数据, 进行训练和预测, 重复 K 次, 最终使用 K 次的预测结果来评判参数是否合适^[10]。一般来说, 10 折交叉验证是最常用的。

3) 留一验证

此方法是 K 折交叉验证的特殊形式, 从原始样本中选取一份作为验证数据, 其余数据构成训练数据, 进行训练和预测, 直到所有的样本都被预测过一次, 综

合评判参数的好坏

2.2.OpenCV 简介

开源计算机视觉库（Open Source Computer Vision Library, OpenCV），实现了图像处理和计算机视觉方面的很多算法^[23]。其通过对代码的优化实现了执行速度的显著提升，在计算效率上与其他主流视觉函数相比具有很大的优势。并且，由于 OpenCV 是开源且免费的，可以节省商业软件的成本。

OpenCV 可用于解决如下领域的问题

- 1) 物体识别
- 2) 人机交互
- 3) 人脸识别
- 4) 图像分区
- 5) 动作识别

2.3.LBP 简介

局部二值模式（Local Binary Patterns, LBP）是一种用来描述图像局部特征的算子，它的作用是进行提取图像的纹理特征，并且是局部提取^[24]。它的显著特点是旋转不变性和灰度不变性^[25]。LBP 算子存在原始版本和改进后的算子

- 1) 原始 LBP 算子

原始的 LBP 算子是在一个 3*3 的窗口内使用的，将外围的 8 个像素的灰度与窗口中心的像素值对比，如果外围的像素值比中心像素值大，则这个点被标记为 1，否则被标记为 0，比较完成后就会得到一个 8bit 数，将其转换成 10 进制后就得到该窗口的 LBP 值，用来反映这个局部区域的纹理特征，如图 2-3 所示

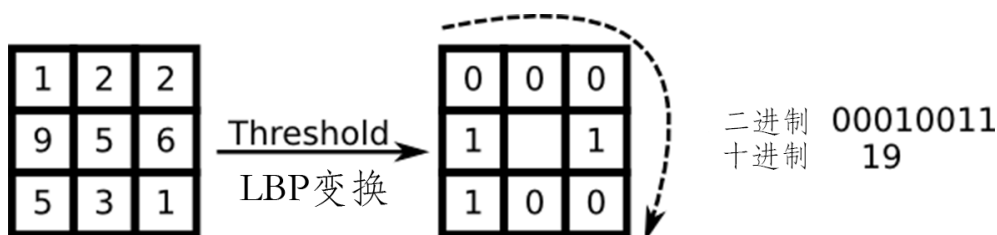


图 2-3 原始 LBP 变换

2) 改进后的 LBP 算子

● 圆形 LBP 算子

由于原始的 LBP 算子只计算了一小块区域, 且是固定范围之内, 为了满足不同尺寸图片的需要, 以及适应不同尺寸的纹理特征, Ojala 提出使用圆形的 LBP 算子, 将 3×3 扩展到半径为 R 的圆形区域内的所有像素点^[26]。这样也能达到旋转不变性和灰度不变性的要求。如图 2-4 所示。

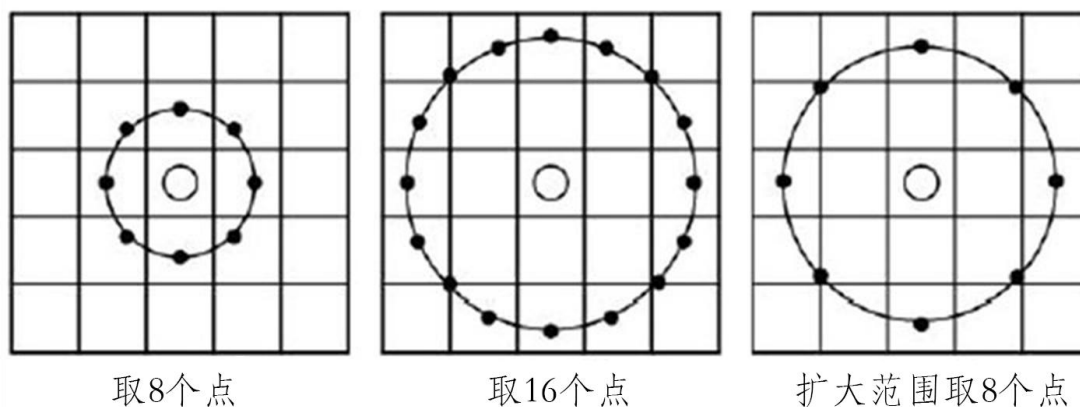


图 2-4 圆形 LBP 变换

● LBP 旋转不变模式

根据 LBP 的定义可以看出, 原始的 LBP 算子在旋转不变性方面仍有缺陷, 于是 Maenpaa 等人提出了具有旋转不变性的 LBP 算子^[27]。通过对圆形邻域的不断旋转得到一组 LBP 的初始值, 并从其中取得最小值作为该领域的 LBP 值^[28]。

● LBP 等价模式

在圆形 LBP 变换中, 若在一个圆内有 25 个采样点, 那么就会产生 $2^{25}=1073741824$ 种二进制模式, 这样大量的数据对纹理特征的读取、分类以及存储都是不利的, 而且数据过于分散也不利于来表达图像的信息。因此, 需要对原始的 LBP 进行降维运算, 才能既完整地表达图像的信息, 又便于处理数据^[29]。

为了减少模式的种类, Ojala 等人提出了 LBP 的等价模式。Ojala 等人认为, 在实际的 LBP 变换中, 绝大多数模式最多只包含两次从 0 到 1 或从 1 到 0 的跳变, 符合这个规则的模式就称为一个等价模式类, 如 000000001 有两次跳变

(0 跳变到 1, 1 跳变到 0)。其余所有两次以上跳变的模式统称为混合模式类, 这样就极大地减少了模式的总数, 例如原始 LBP 降维之后, 就从 $2^8=256$ 维降到了 59 维, 使得数据能更快地被处理, 也不会丢失图像的信息^[30]。

LBP 算子可以用来匹配图像的局部特征, 进行 LBP 变化之后, 是利用其生成的直方图来判断特征, 具体步骤如下

- 1) 切割出图像中需要进行 LBP 变换的部分
- 2) 对于每个像素点, 将其周围的八个像素值与其比较, 若周围的像素值大则取 1, 否则取 0, 全部计算完成之后, 得到一个 8 位的二进制数, 即该像素点的 LBP 值
- 3) 对所有的像素点进行 LBP 值计算, 通过等价模式进行降维, 将其降到 59 维
- 4) 计算每个值出现的概率, 生成统计直方图, 并对直方图进行归一化处理

以上步骤完成只会, 就可以使用 SVM 对数据进行机器学习了, 图 2-5 是一张人脸在不同光照条件下的 LBP 变化结果。

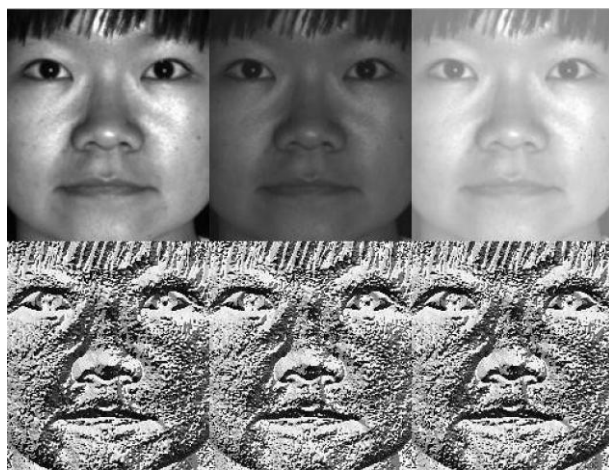


图 2-5 不同光照条件时同一张脸部 LBP 变换结果

2.4.Xcode 与 Objective-C

2.4.1 Xcode 简介

Xcode 是由苹果公司向开发者提供的集成开发环境 (Integrated Development Environment, IDE), 用于开发 MAC OS X 和 iOS 应用程序, 并可以通过 Xcode

便捷地进行用户界面的设计、软件的编写、调试、模拟测试、性能测试、内存泄露检测以及最终的上传到 App Store。开发者可以使用 Cocoa、Carbon、Java 等开发模式。

2.4.2 Objective-C 简介

面向对象的 C 语言 (Objective-C) 是一种通用的、高级的、面向对象的编程语言,它是标准 C 语言的超集,将消息传递机制和面向对象的机制加入 ANSI C 中, GCC 和 LLVM 是其主要编译器^[31]。

在 Objective-C 中使用 C 语言是完全合法的,也可以直接编译通过,但是 Objective-C 在语法上和 C 语言有很大的差异,例如类的定义, Objective-C 中强制要求将类的定义 (interface) 与实现 (implementation) 分为两个部分,一个类的定义以 @interface 开头,以 @end 结尾,如下:

```
@interface MyObject : NSObject {
    int memberVar1; // 实体变量
    id memberVar2;
}

+(return_type) class_method; // 类方法

-(return_type) instance_method1; // 实例方法
-(return_type) instance_method2: (int) p1;
-(return_type) instance_method3: (int) p1 andPar: (int) p2;
@end
```

以上定义中,定义了一个名为 MyObject 的类,继承自 NSObject,并具有两个变量,以及一个类方法和三个实例方法。

而在实现部分,则以 @implementation 开头,以 @end 结尾,如下:

```
@implementation MyObject
+(return_type) class_method {
}

-(return_type) instance_method1 {
}

-(return_type) instance_method2: (int) p1 {
}

-(return_type) instance_method3: (int) p1 andPar: (int) p2 {
}

@end
```

在 Objective-C 中, 得益于 Smalltalk 式的消息传递机制, 使得其方法相比于 C 语言和 C++ 语言更具有可读性。

2.4.3 AFNetworking 简介

AFNetworking 是在 iOS 和 MAC OS X 开发中使用的一个网络通信类库, 它是一个非常高效的网络模块, 它把复杂的网络底层操作封装成友好的类和方法, 并加入了异常处理, 拥有非常丰富的 API, 它为 iPhone、iPad 以及 Mac 的软件开发做出了很大的贡献。

它的使用非常简单, 能够方便地向服务器发送 GET 和 POST 请求, 用法如下

Get 请求:

```
AFHTTPRequestOperationManager *manager = [AFHTTPRequestOperationManager manager];
[manager GET:@"http://example.com/resources.json" parameters:nil
success:^(AFHTTPRequestOperation *operation, id responseObject) {
    NSLog(@"JSON: %@", responseObject);
} failure:^(AFHTTPRequestOperation *operation, NSError *error) {
    NSLog(@"Error: %@", error);
}];
```

以上代码向服务器发送了一个异步 GET 请求, 并在成功的时候返回 responseObject, 失败的时候返回错误信息

POST 请求:

```
AFHTTPRequestOperationManager *manager = [AFHTTPRequestOperationManager manager];
NSDictionary *parameters = @{@"foo": @"bar"};
[manager POST:@"http://example.com/resources.json" parameters:parameters
success:^(AFHTTPRequestOperation *operation, id responseObject) {
    NSLog(@"JSON: %@", responseObject);
} failure:^(AFHTTPRequestOperation *operation, NSError *error) {
    NSLog(@"Error: %@", error);
}];
```

以上代码向服务器发送了一个异步 POST 请求, 包含了一个请求参数, 并在成功的时候返回 responseObject, 失败的时候返回错误信息。

2.4.4 MBProgressHUD 简介

MBProgressHUD 是一个 iOS 的第三方库, 它能够展示一个半透明的提示框,

并在上面添加指示或者文字来提示后台进程的进行情况。由于 iOS 中提示框是私有的，所以 `MBProgressHUD` 为开发者提供了一个很好的解决方案。

`MBProgressHUD` 很容易使用，只需要设置好要展示的样式、文字、以及所表征的后台进程，就可以很好地将进程的进度表示出来。其用法如下：

```
[MBProgressHUD showHUDAddedTo:self.view animated:YES];
dispatch_time_t popTime = dispatch_time(DISPATCH_TIME_NOW, 0.01 * NSEC_PER_SEC);
dispatch_after(popTime, dispatch_get_main_queue(), ^(void){
    // Do something...
    [MBProgressHUD hideHUDForView:self.view animated:YES];
});
```

以上代码先展示了一个 HUD，然后开始一个后台进程，在进程结束后隐藏 HUD。

2.4.5 JSONModel 简介

在面向对象开发中经常使用到的一种模式是模型、视图、控制器（`Model`，`View`，`Controller`，`MVC`）模式^[32]，其中 `Model` 用来存储和管理数据。大部分数据从服务器上下载好后都是 JSON 格式的，这时需要将其转换成 `Model` 来使用，而 `JSONModel` 就提供了一个非常简便的方法来解析以及转换 JSON 数据，不仅非常高效，而且非常安全。

`JSONModel` 的用法很简单，只需要将 `Model` 继承自 `JSONModel`，并写好对应的标签，就可以自动地从 JSON 数据中生成对应的 `Model`，一个简单的 `Model` 声明如下：

```
#import "JSONModel.h"
@interface CountryModel : JSONModel
@property (assign, nonatomic) int id;
@property (strong, nonatomic) NSString* country;
@property (strong, nonatomic) NSString* dialCode;
@property (assign, nonatomic) BOOL isInEurope;
@end
```

以上代码中声明了一个 `CountryModel` 的类，继承自 `JSONModel`，拥有四个成员变量，只要对应 JSON 数据中也是这四个标签，就可以使用 `initWithDictionary` 方法来生成 `Model` 了，若是存在错误，也会输出相应的报错信息。

3. 需求分析与功能设计

本章主要阐述基于 iOS 的头像萌化系统的总体设计以及模块设计方案。首先从整体上对系统进行需求分析，然后介绍本系统的总体架构设计，随后分别对各个功能模块进行设计规划，最后对系统的界面进行设计。


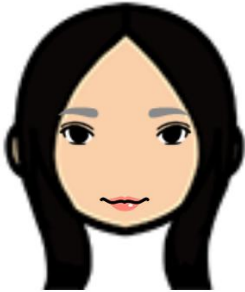
3.1 系统需求分析

现在移动互联网的浪潮下，社交软件在井喷式发展，而在社交网络中，每个人都需要有自己的头像，有一部分用户会选择使用仿真的卡通头像，于是一款叫做“脸萌”的软件曾排上下载量的榜首。

而脸萌仍然有缺点：用户只能手动选择脸型、发型、眼镜等特征，来组成一张卡通头像，这么做的不足是用户的操作比较复杂，需要选择多达十个左右的特征才能完成头像的生成。

针对现有脸萌 APP 的局限性，本课题试想设计一种基于 iOS 的头像萌化系统。该系统的需求如下：基于 iOS 平台完成，用户只需要下载该软件，选择图片后，系统即可进行人脸识别，对人脸的各个部位的特征进行判断以及分类，这些特征包括眼睛、眉毛、嘴巴、头发和眼镜，分类好后便选择合适的图片后生成一张类似于脸萌的卡通头像，并保证一定的准确度。其效果如表 3-1 所示

表 3-1 原图和卡通头像对比

原自拍照	生成的卡通头像
	

根据需求，用户需要将图片上传到人脸识别和特征点定位的服务器进行处理，而图片的来源有两种：选择手机本地照片或直接拍摄照片；而为了使本系统所生成的卡通头像和原图较为相似，就需要挑选合适的发型、五官和配饰等特征，这个过程便是分类。下面将详细介绍系统的流程设计。

3.2 系统架构设计

本系统采用分层的体系结构，将系统的功能划分成不同的层次，如图 3-1 所示，本系统主要有如下四个层次

1. 用户交互层

用户交互层展示了用户的交互界面，提供操作按钮，并展示了生成卡通头像的结果

2. 系统应用功能层

系统应用功能层包含了三个功能模块：人脸特征点定位模块、SVM 分类模块和视图模块。

- 1) 人脸特征点定位模块负责负责手机与服务器的通信，主要功能是上传照片并获取到面部坐标点数据；
- 2) SVM 分类模块负责使用处理好的数据，使用 SVM 模型来预测特征标签；
- 3) 视图模块负责根据特征标签选取合适的图片，拼接成一张卡通头像

3. 核心算法层

核心算法层包含了本系统最重要的模块：特征提取模块。该模块负责处理面部坐标信息，将其转换成 SVM 分类模块能处理的数据。

4. 系统支持层

系统支持层提供了各类第三方库来对系统提供支持，例如 OpenCV、JsonModel 等等。

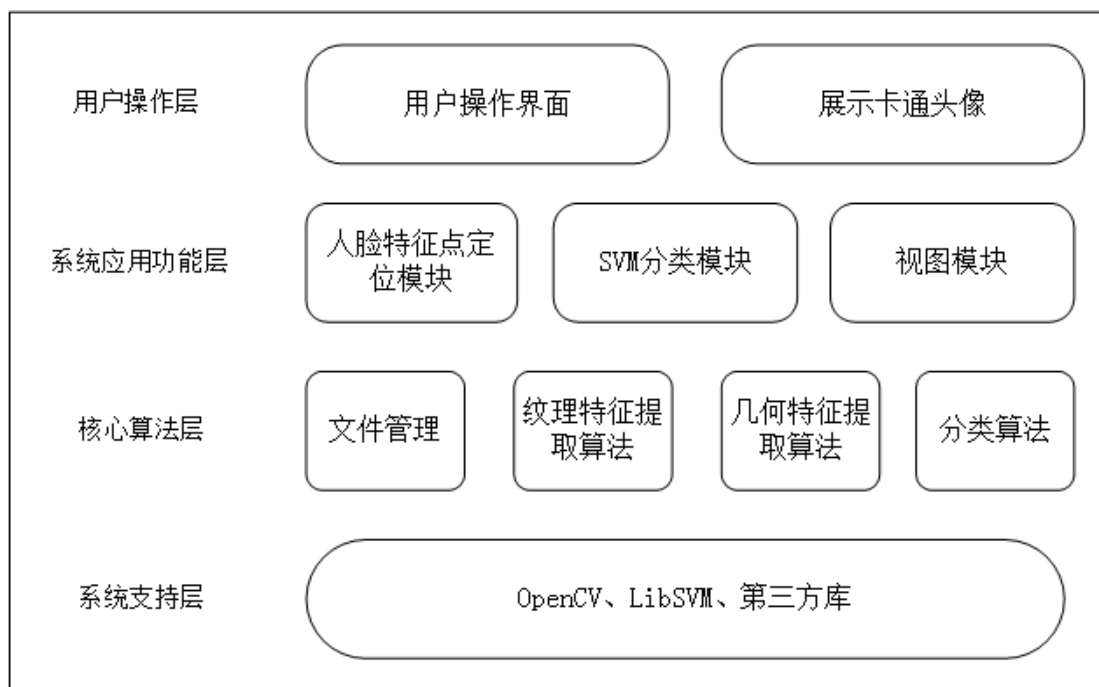


图 3-1 系统架构设计示意图

3.3 功能模块设计

根据上述系统总体设计可将系统分为 4 个不同的功能模块:人脸特征点定位模块、特征提取模块、SVM 预测模块、视图模块,这几个模块分工明确。人脸特征点定位模块主要实现图片的上传和人脸特征点信息的获取;特征提取模块根据不同的特征,使用对应的算法处理人脸特征点坐标信息,提取到能够被 SVM 分类的特征数据;SVM 分类模块对处理好的数据进行预测,输出相应的特征信息;视图模块则根据人脸每个部位的特征信息,选取对应的图片,拼接成萌化头像。本课题的模块图如图 3-2 所示

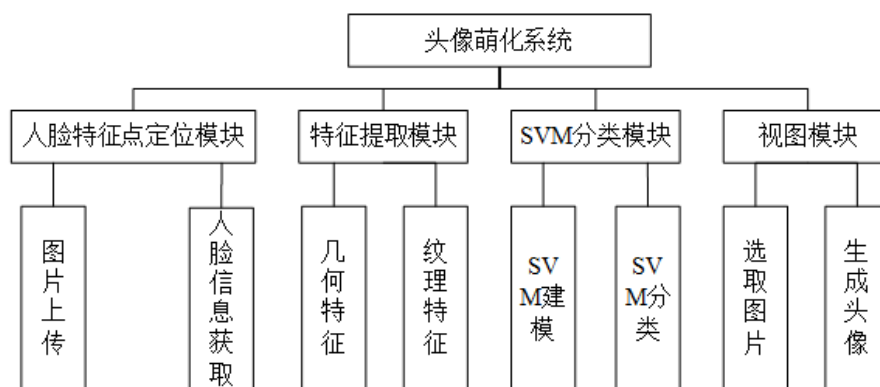


图 3-2 头像萌化系统的模块图

下面将对每个模块进行详细地阐述。

3.3.1 人脸特征点定位模块

由于人脸识别和人脸特征点定位的算法放到服务器端,所以需要人脸特征点定位模块来传递数据并获取数据。

人脸特征点定位模块主要实现的功能是将用户选择或者拍摄的图片上传到服务器,等待服务器对图片处理之后,返回人脸特征点的数据,这个模块是整个系统的基础,在没有网络的情况下,整个系统不能够正常的运行。其具体过程如图 3-3 所示

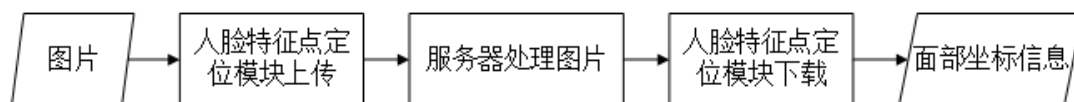


图 3-3 人脸特征点定位模块流程示意图

3.3.2 特征提取模块设计

由于从服务器获取到的特征点信息是点的坐标值,所以从服务器端获取到特征点的信息之后,需要对特征点信息进行数据处理,将其转换成 SVM 能够预测的信息,这个过程就是特征的提取。

对于每一种特征,数据处理的方式是不同的。在整体上,本课题将所有的特征分为两类,分别是几何特征和纹理特征,几何特征可以仅仅通过简单处理点的坐标信息得到,而纹理特征在得到点的坐标信息之后,还需要对图片进行算法变换,得到图片最终的特征数据。

本课题所选取的人脸特征包含人脸特征和配饰特征。人脸特征有如下特征

- 1) 眼睛
- 2) 眉毛
- 3) 嘴巴
- 4) 头发

配饰特征有如下特征

- 1) 眼镜

将以上的分类相结合,就得到一张特征提取方式表,如表 3-2 所示

表 3-2 特征提取方式

	眼睛	眼镜	眉毛	嘴巴	嘴唇	头发
几何特征	大/小	无	粗/细	张开/闭合	厚/薄	无
纹理特征	无	有/无	浓/淡	无	无	长/短 有刘海/无刘海

表 3-2 详细介绍了每个特征的分类方式,特征提取模块根据这些分类方式处理对应的数据,处理完成之后就交由 SVM 分类模块来预测特征对应的标签,特征提取模块的流程如图 3-4 所示

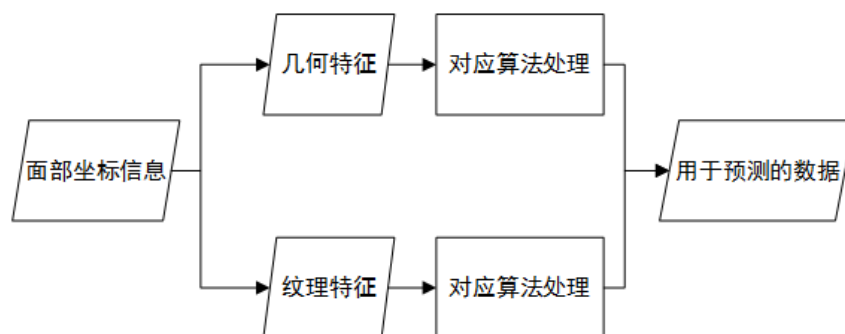


图 3-4 特征提取模块流程示意图

3.3.3 SVM 分类模块设计

当特征提取模块处理完面部坐标信息并提取到各个部位的特征之后,就需要 SVM 模块来预测数据,对于每个特征,都有一个 SVM 模型与之对应,这个模型需要在软件使用前被训练好,在使用时将直接调用预测函数来输出特征标签。由于模型的准确率将直接影响生成的头像的相似程度,所以生成准确的 SVM 模型非常重要。

本课题从网络上选取自拍照,共 569 张,形成自拍图片库,使用这些图片来训练 SVM 模型。SVM 分类模块的流程如图 3-5 所示。

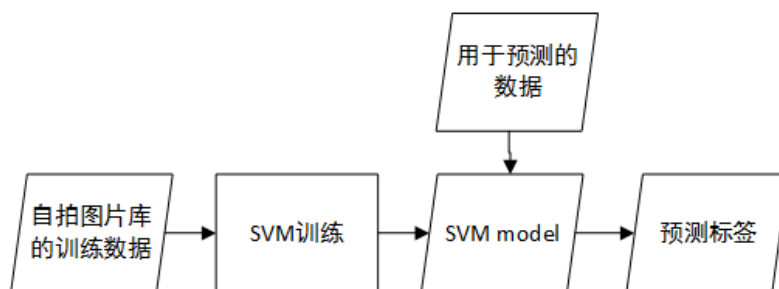


图 3-5 SVM 分类模块流程示意图

3.3.4 视图模块设计

在得到 SVM 输出的标签之后，就可以根据对应的特征来选择图片，并将图片拼接成一张完整的人脸，根据特征提取模块的特征分类方式，每个特征分类方式对应一张图片，本课题所用到的卡通图片如表 3-3 所示。

表 3-3 详细列出了本课题所使用的图片，以及与图片对应的特征，使用上述图片，针对每个特征从中选择一张图片，就可以完整地拼出一张人脸，在表 3-1 中已经给出范例。

视图模块的流程如图 3-6 所示。

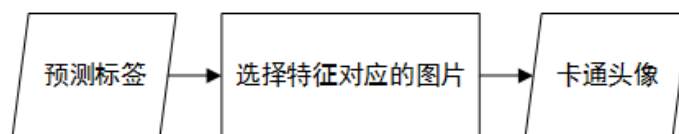




















图 3-6 视图模块流程示意图

表 3-3 特征对应图片表

眼睛	大眼睛			小眼睛		
						
眉毛	厚、浓眉毛	厚、淡眉毛		薄、浓眉毛	薄、淡眉毛	
						
嘴巴	厚嘴唇、张开	厚嘴唇、闭合		薄嘴唇、张开	薄嘴唇、闭合	
						
眼镜	有眼镜			无眼镜		
						
发型	女、长发、有刘海	女、长发、无刘海	女、短发、有刘海	女、短发、无刘海	男、有刘海	男、无刘海
						

3.4 系统界面设计

本系统的界面设计如图 3-7 所示，其画出了本系统的各个控件，包含三类，分别是 UIButton 类、UIImageView 类、UILabel 类，描述了各个控件的功能。

本系统的界面一共有三个 UIButton，显示在屏幕的最下方，分别是：

- 1) 选择照片按钮：点击后会请求照片图库权限，如果允许则打开本地照片图库

- 2) 拍摄照片按钮：点击后会请求相机权限，如果允许则打开相机界面
- 3) 上传图片按钮：点击后会使用人脸特征点定位模块上传用户选择的图片到服务器

两个 UIImageView，显示在屏幕的中间，分别是：

- 1) 原图片：显示用户选择的照片
- 2) 卡通头像：显示最终生成的卡通头像

一个 UILabel，显示在屏幕的最上方，作用是给用户操作提示。

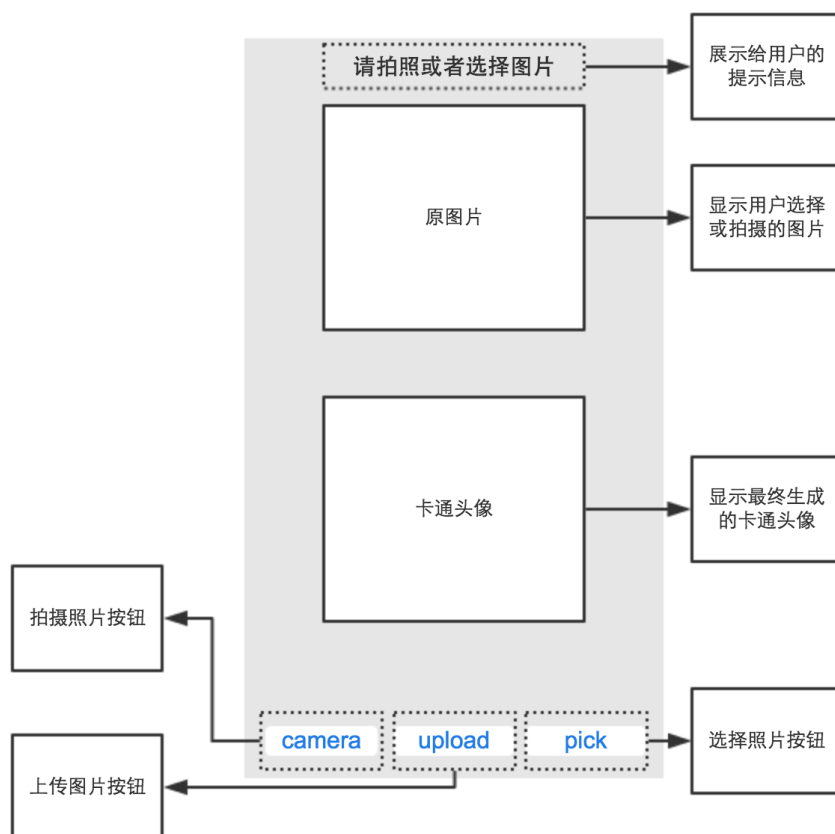


图 3-7 系统界面设计示意图

3.5 本章小结

本章主要介绍了基于 iOS 的头像萌化系统的设计流程，首先对本系统进行需求分析，然后介绍了本系统的架构设计，随后介绍了本系统的四个基本模块：人脸特征点定位模块、特征提取模块、SVM 分类模块以及视图模块。本系统的核心是 SVM 分类模块，主要用于 SVM 模型的训练，已经对数据的预测，其预测结果将直接影响最终生成的头像的效果；人脸特征点定位模块负责图片的上传和

面部坐标信息的获取；特征提取模块负责处理面部坐标信息，提取面部特征，将其处理成能够用于 SVM 预测的数据；而视图模块则负责利用 SVM 输出的标签生成最终的头像。最后介绍了本系统的界面设计。

4. 系统实现与测试

本章在对基于 iOS 的头像萌化系统进行设计与分析后,在一定的软硬件环境之下,分别实现了各个模块,并在 iOS 系统上实现了模块的功能。本节主要介绍了人脸眼部、眉毛和眼镜的特征提取方法和建模方法,以及人脸特征点定位模块和 SVM 预测模块。最后对系统进行测试,并分析结果。本系统主要使用的语言是 Matlab 语言和 Objective-C,主要工具是 Xcode。

4.1 软硬件环境

4.1.1 软件环境

本系统主要用到的软件有 LibSVM、Matlab、OpenCV、Xcode。其所实现的功能介绍如下

- 1) LibSVM: 在建模阶段训练 SVM 模型,在使用阶段对数据进行 SVM 预测。
- 2) Matlab: 在建模阶段配合 LibSVM 进行 SVM 模型的训练
- 3) OpenCV: 在使用阶段实现切割图片、边缘检测等功能
- 4) Xcode: 完成 iOS 软件的编写

4.1.2 硬件环境

本系统主要用到的硬件为 MacBook 和 iPhone,MacBook 负责软件的编写,iPhone 负责系统的运行和测试。

4.2 人脸特征点定位模块

人脸特征点定位模块负责图片的上传以及人脸面部坐标数据的获取,该模块是系统的基础。本系统中使用了名为 AFNetworking 的第三方库,能够方便地进行异步网络请求。

该模块在系统中是以 HttpClient 类的形式出现的,在控制层初始化这个类,用户进入系统并成功确定图片之后,便可以通过点击 uploadBtn 上传图片,调用人脸特征点定位模块的 getFaceModel 方法,人脸特征点定位模块便开始工作,如果成功从服务器获取数据,则将数据交给特征提取模块,如果获取失败,则提示用户未成功检测人脸,请重新上传。

人脸面部坐标是通过面部的 29 个坐标点表现出来的, 将一张人脸照片上传到人脸检测服务器便可获取到这 29 个点的信息。成功获取到数据之后, 就使用 FaceModel 将数据封装成模型, 坐标点的信息保存在 LandmarkModel 中, 封装好数据之后, 人脸特征点定位模块便将 FaceModel 传给特征提取模块。该模块的实现流程如图 4-1 所示。

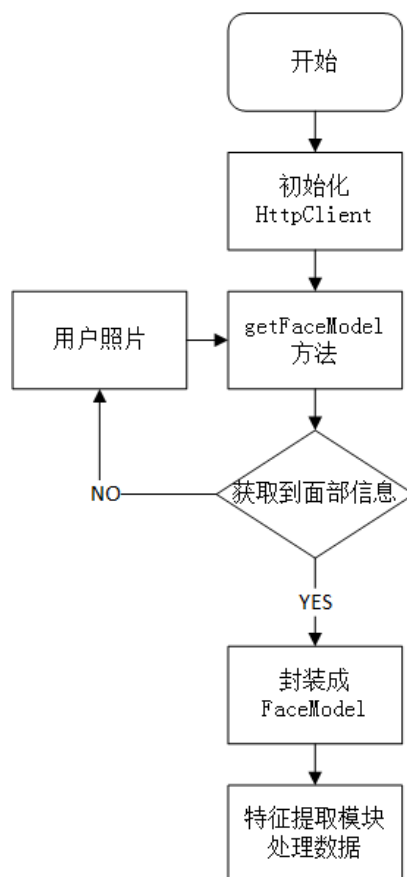


图 4-1 人脸特征点定位模块实现流程

主要代码如下:

```

AFHTTPRequestOperationManager *manager; // 初始化 manager
manager POST:@ "http://120.24.247.190/RCPR/detect_landmark"; // 向服务器发送 post 请求
if success 则 successBlock(face, aDic); // 给特征提取模块来提取特征
else 则 failBlock(error); // 给出错误提示信息
    
```

以上代码通过一个 AFHTTPRequestOperationManager 来进行网络通信, 使用 block 的方式来发送异步的 POST 请求, 请求成功后, 将获取到 JSON 格式的数据, 如下

```

{
    "imgWidth": 356, //原图宽度
    "imgHeight": 452, //原图高度
    "imgPath": "http://120.24.247.190/upload/test.jpg", //图片路径
    
```

```

"state": 0,          //状态, 0=成功, 1=出错, 2=没有检测到脸
"landmarks": [      //landmark 数组, 多个脸的 landmark
    {
        //第一个脸的 landmark
        "x": 64,      //脸的坐标 x
        "y": 112,     //脸的坐标 y
        "width": 225, //脸的宽度
        "height": 225, //脸的高度
        "landmark": [ //landmark 坐标集合
            {
                "x": 98, //landmark 第一个点的坐标 x
                "y": 183, //landmark 第一个点的坐标 y
                "index": 1 //landmark 序号
            }, ...      //接下来的 28 个点
        ]
    }, ...             //接下来的脸
]
}

```

一旦获取面部坐标信息成功, 就使用 FaceModel 来封装 JSON 数据, FaceModel 将保存人脸部的坐标信息, 此时将信息交给特征提取模块来处理。

4.3 人脸特征的提取与建模

本节介绍人脸各个特征的提取方法以及建模过程。为了实现本系统的识别功能, 需要首先提取出人脸各个部位的特征并针对每个特征建立相应的模型。对于每一张从服务器成功获取到面部信息的人脸, 会得到 29 个面部的坐标点, 其分布如图 4-2 所示。从图中可以看出, 这 29 个点分布在眉毛、眼睛、嘴巴以及下巴, 将这 29 个点记为 marks[i](i=0,1,...,28), 本课题将利用这些信息点来提取特征并建模。

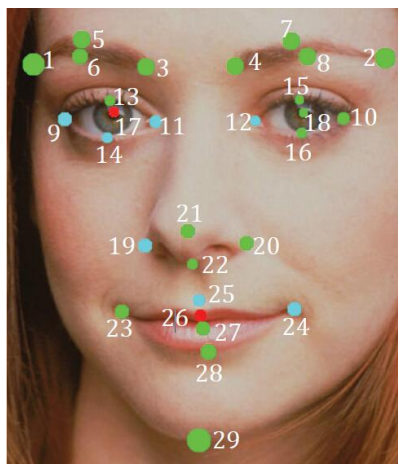


图 4-2 29 个点在人脸上的分布图

显然直接使用这 29 个点的纵横坐标来表征几何特征是不准确的，因为每张图片的分辨率不同，且点在图片上的位置也会有很大差异，如果不经归一化，这样是无法得到人脸的特征的。由于大部分人脸的双瞳孔之间的距离相近，所以瞳距则能作为归一化的桥梁^[33]，所以本课题中，使用瞳距作为归一化参数，在下面的实现过程中记瞳距为 L 。

而在建模之前，需要建立训练图片库，本课题所有训练图片均从网络获取。获取到图片之后需要针对每个特征对图片进行筛选，得到适合训练的图片集。

图 4-3 展示了每个特征的 SVM 模型训练的流程。

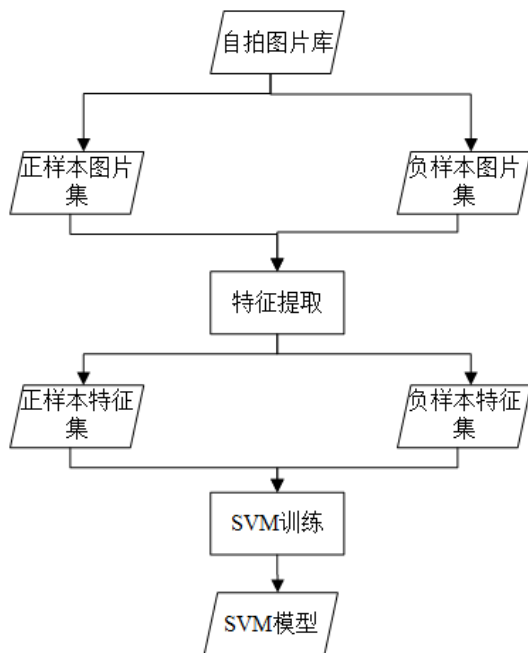


图 4-3 SVM 模型训练流程

下面将首先介绍训练数据，然后针对脸部的每个部位的特征的提取与建模详

细介绍其实现流程。

4.3.1 筛选训练数据

为了训练出准确的模型,本课题从网络上下载并筛选出了共 569 张无眼镜的人脸照以及 100 张有眼镜的照片,样张如图 4-4 所示



图 4-4 自拍图片库样本示意图

在这个训练集中,针对某一特征,有的图片是无法使用的,例如眉毛被遮挡、干扰信息过多等,为了保证模型的准确性,在对每个特征进行训练之前需要对图片进行筛选,经过筛选后,对于每个特征用来训练的图片数量如表 4-1 所示。

表 4-1 面部特征训练集图片数量表

训练集名称	图片总数(张)	正样本数(张)	负样本数(张)
眉毛浓(正)淡(负)	70	21	49
眉毛厚(正)薄(负)	70	23	47
眼睛大(正)小(负)	279	157	122
眼镜有(正)无(负)	60	30	30
嘴巴开(正)闭(负)	199	100	99
嘴唇厚(正)薄(负)	89	39	50
女头发长(正)短(负)	20	10	10
刘海有(正)无(负)	20	10	10

4.3.2 眼部特征的提取与建模

本课题中对眼睛的识别主要是识别眼睛的大小。这个特征归为几何特征。可

以使用点的坐标计算得来而不需要对图片进行进一步的处理。眼睛的大小可以使用眼睛的张开角度来表征^[34]，如图 4-5 所示。



图 4-5 眼部几何特征示意图

在图 4-5 中，本系统计算了双眼的角度特征， f_1 和 f_2 分别代表张开的角度值，如果 f_1 或者 f_2 越大，那么眼睛越大，表 4-2 中给出了眼部角度特征的定义以及计算方法

表 4-2 眼部特征定义

特征标号	含义	计算方法
f_1	左眼睛眼角的开合角度	$f_1 = \text{distance}(\text{mark}[13], \text{mark}[14]) / \text{distance}(\text{mark}[9], \text{mark}[11])$
f_2	右眼睛眼角的开合角度	$f_2 = \text{distance}(\text{mark}[15], \text{mark}[16]) / \text{distance}(\text{mark}[10], \text{mark}[10])$
f_3	左右眼眼角开合角度的平均值	$f_3 = (f_1 + f_2) / 2$

根据以上定义，得到提取眼部特征的代码如下

```
float    leftEyeAngle    =    [self    distanceOfTwoLandMark:faceModel.landmark[12]
another:faceModel.landmark[13]] / [self    distanceOfTwoLandMark:faceModel.landmark[8]
another:faceModel.landmark[10]];    // 左眼角度
float    rightEyeAngle    =    [self    distanceOfTwoLandMark:faceModel.landmark[15]
another:faceModel.landmark[14]] / [self    distanceOfTwoLandMark:faceModel.landmark[9]
another:faceModel.landmark[11]];    //右眼角度
float avergEyeAngle = (leftEyeAngle + rightEyeAngle) / 2;    // 平均角度
```

确定了眼部特征的计算方法之后，就需要对所有的训练图片做特征提取。在训练阶段，本课题使用 excel 来完成数据的计算，所有训练图片上传到服务器进行处理之后，所有的面部信息将保存在 excel 表格里面，利用简单的公式计算出以上三个特征。将眼部特征数据保存成 LibSVM 需要的格式，眼部数据示意表如表 4-3 所示。

表 4-3 用于训练的眼部数据示意图

特征标签	左眼张开角度	右眼张开角度	平均张开角度
1	0.477690192	0.386510341	0.432100267
1	0.465633074	0.5	0.482816537
1	0.488846181	0.436648003	0.462747092
-1	0.316227766	0.21227496	0.264251363
-1	0.376949801	0.330572761	0.353761281
1	0.405346702	0.468164589	0.436755646
-1	0.379432302	0.427502773	0.403467537
-1	0.378474891	0.392837101	0.385655996
-1	0.39223227	0.357192115	0.374712193
1	0.413096192	0.413096192	0.413096192

将表 4-3 中的数据保存成文本文件格式的数据, 就可以使用 matlab 里的 LibSVM 工具进行训练了, 模型的训练过程分为以下步骤

- 1) 从数据文件加载数据, 并分别得到 eye_label 和 eye_data, eye_label 是存储所有标签的矩阵, eye_data 是存储所有数据的矩阵

```
[eye_label, eye_data] = libsvmread('part1_eye');
```

- 2) 取一部分数据为训练数据, 另一部分为预测数据

```
train_label = eye_label(1:240,:); // 训练标签
train_data = eye_data(1:240,:); // 训练数据
test_label = eye_label(241:279,:); // 测试标签
test_data = eye_data(241:279,:); // 测试数据
```

- 3) 将数据归一化到[0,1]的范围, 避免过大或过小的数据对训练的结果造成影响

```
[train_data, pstrain] = mapminmax(train_data);
pstrain.ymin = 0;
pstrain.ymax = 1;
[train_data, pstrain] = mapminmax(train_data, pstrain);
train_data = train_data; // 数据归一化之后的结果
```

- 4) 第一次交叉验证, 得到最优 c 和 g 的范围

```
[bestacc,bestc,bestg] = SVMcg(train_label, train_data,-10, 10, -10, 10);
disp('c & g'); // 寻找最优的 c 和 g 参数
str = sprintf( 'Best Cross Validation Accuracy = %g%% Best c = %g Best g = %g',bestacc,bestc,bestg);
disp(str);
```

以上代码使用 SVMcg 工具, 能自动进行交叉验证, 找到最优的 c 和 g 参数, 这里运行完之后将给出最后的 c 参数和 g 参数的范围, 本次输出的范围是最优的

$\log_2 c$ 分布在 $[-2,8]$, 最优的 $\log_2 g$ 分布在 $[-4,4]$, 将根据这一范围取得更加精确的 c 和 g 参数

5) 第二次交叉验证, 使用第四步得到的 c 和 g 的范围, 得到 $bestc$ 和 $bestg$,

```
[bestacc,bestc,bestg] = SVMcg(train_label, train_data,-2,8,-4,4,10,0.5,0.5,0.9);
disp('c & g');
str = sprintf( 'Best Cross Validation Accuracy = %g%% Best c = %g Best g
= %g',bestacc,bestc,bestg);
disp(str);
```

以上代码运算完之后便会在屏幕上打印最佳的 c 和 g , 以及训练集的准确率, 如下所示, 最佳 c 参数为 181.019, 最佳 g 参数为 16, 训练集的准确率为 91.6667%

```
Cross Validation Accuracy = 90.8333%
c & g
Best Cross Validation Accuracy = 91.6667% Best c = 181.019 Best g = 16 //输出的最佳 cg
参数
```

6) 训练 SVM 模型: 得到 c 和 g 参数后, 就可以使用该参数、训练标签和数据来生成 SVM 模型了, 如下

```
cmd = ['-c ',num2str(bestc),' -g ',num2str(bestg)];
model = svmtrain(train_label,train_data,cmd);
```

7) 测试 SVM 模型的准确率: 得到 SVM 模型之后, 就可以利用模型来对预测数据进行预测, 得到其准确率, 如下

```
[predict_label,accuracy, prob_estimize] = svmpredict(test_label, test_data, model);
total = length(test_label);
right = sum(predict_label == test_label);
disp('accuracy');
str = sprintf( 'Accuracy = %g%% (%d/%d)',accuracy(1),right,total);
disp(str);
```

这段代码将输出该模型的准确率, 如下所示, 输出中显示 39 个预测样本成功预测了 29 个, 准确率为 74.359%

```
optimization finished, #iter = 2651
nu = 0.171009
obj = -7150.835249, rho = -0.111650
nSV = 57, nBSV = 35
Total nSV = 57
Accuracy = 74.359% (29/39) (classification) // 输出的准确率
```

至此,判断眼镜大小的 SVM 模型已经训练完成,保存后就可以用其对眼镜大小进行判断。

4.3.3 眉毛特征的提取与建模

本课题中对眉毛的特征提取包括两个特征:几何特征和纹理特征。几何特征判断眉毛的厚薄,而纹理特征则判断眉毛的浓密。下面将针对这两个特征分别讲述实现过程。

1. 几何特征

对于几何特征,其提取过程较为简便。如图 4-2 所示,眉毛的厚薄可以使用 mark[4]和 mark[5]的垂直距离以及 mark[6]和 mark[7]的垂直距离来表征,如图 4-6 所示



图 4-6 眉毛几何特征示意图

在图 4-10 中, f_1 和 f_2 分别代表左眉毛和右眉毛的归一化高度,其值越大,那么眉毛越厚,反之越薄。由于每张图片的分辨率不同,故不可以直接使用像素距离,需要对该特征进行归一化,本系统将绝对像素距离除以瞳距 (L) 得到归一化距离。表 4-4 中给出了眉毛几何特征的定义以及计算方法

表 4-4 眉毛几何特征定义

特征标号	含义	计算方法
f_1	左眉毛的归一化高度	$f_1 = \text{distance}(\text{mark}[4], \text{mark}[5]) / L$
f_2	右眉毛的归一化高度	$f_2 = \text{distance}(\text{mark}[6], \text{mark}[7]) / L$

根据以上定义,得到提取眉毛几何特征的代码如下

```
float leftBrowThickness = [self distanceOfTwoLandMark:faceModel.landmark[4]
another:faceModel.landmark[5]] / eyeDistance; // 左眉毛归一化高度

float rightBrowThickness = [self distanceOfTwoLandMark:faceModel.landmark[6]
another:faceModel.landmark[7]] / eyeDistance; // 右眉毛归一化高度
```

在成功提取到眉毛的几何特征之后,就可以利用自拍图片库中的图片数据训练 SVM 模型,其过程与 4.3.2 中眼部特征的建模过程完全相同,这里不再赘述,最终得到的最优 c 和 g 参数分别是 1 和 4.35。

2. 纹理特征

相比于几何特征的处理过程,纹理特征的处理较为复杂,不可以仅仅使用面部坐标点的数据,而需要根据坐标对相应的图片进行处理,并提取器特征,再训练成模型。本课题中对眉毛的纹理特征的处理结果是区分眉毛的浓密。使用 LBP 变换来提取纹理特征。

在本文的 2.3 中已经对 LBP 进行了简要介绍, LBP 变换能提取图像的局部纹理特征,并将其以灰度直方图的形式表现出来。本课题中使用的 LBP 变换是原始 3×3 的 LBP 变换,然后对 256 维特征进行了降维,使之只有 59 维,便于数据的处理。

下面将使用图 4-4 中的第一张图片作为示例来讲述眉毛纹理特征的提取和建模的过程:

- 1) 根据面部坐标点确定眉毛的位置,将眉毛的图片从原图中切割出来。以左眉毛为例,由图 4-2 可知眉毛的四个边缘点,分别为 $mark[0]$ 、 $mark[2]$ 、 $mark[4]$ 、 $mark[5]$,利用这四个点可以获取眉毛的矩形,其算法如表 4-5 所示

表 4-5 眉毛矩形区域的计算方式

标号	含义	计算方法
LOriginX	左眉毛原点的横坐标	$LOriginX = mark[0].x$
LOriginY	左眉毛原点的纵坐标	$LOriginY = mark[4].y$
LWidth	左眉毛宽度	$LWidth = mark[2].x - mark[0].x$
LHeight	右眉毛宽度	$Max((mark[0].y - mark[4].y), (mark[5].y - mark[4].y), (mark[2].y - mark[4].y))$

根据以上算法,得到眉毛的区域提取方法的代码如下

```
int leftOriginX = landmark1.x;
int leftOriginY = landmark5.y;
int leftWidth = landmark3.x - landmark1.x;
int leftHeight = [self Max:(landmark1.y - landmark5.y) b:(landmark6.y - landmark5.y)
c:(landmark3.y - landmark5.y)];
```

根据以上算法可以得到眉毛的区域，便可在原图上切割出眉毛，切割图如图 4-7 所示



图 4-7 切割出来的左右眉毛示意图

- 2) 对切割出来的眉毛的图片进行 LBP 变换，LBP 变换使用一个双重循环实现，其关键代码如下

```
for(int j=1;j<width-1;j++)
{
    for(int i=1;i<height-1;i++)
    {
        uchar neighborhood[8]={0};
        neighborhood[7] = CV_IMAGE_ELEM( src, uchar, i-1, j-1);
        neighborhood[6] = CV_IMAGE_ELEM( src, uchar, i-1, j);
        neighborhood[5] = CV_IMAGE_ELEM( src, uchar, i-1, j+1);
        neighborhood[4] = CV_IMAGE_ELEM( src, uchar, i, j+1);
        neighborhood[3] = CV_IMAGE_ELEM( src, uchar, i+1, j+1);
        neighborhood[2] = CV_IMAGE_ELEM( src, uchar, i+1, j);
        neighborhood[1] = CV_IMAGE_ELEM( src, uchar, i+1, j-1);
        neighborhood[0] = CV_IMAGE_ELEM( src, uchar, i, j-1);
    }
}
```

经过双重循环得到的 neighborhood 即是处于 $\langle i, j \rangle$ 点的 LBP 值表征，将其和降维表对应就得到其降维之后的值。眉毛的 LBP 变换图片如图 4-8 所示



图 4-8 左右眉毛的 LBP 变换图

- 3) 提取 LBP 变换后图片的直方图, 对于左眉毛和右眉毛, 分别生成一个 59 个标签的数据, 其中每个标签对应的数据是降维表中对应数据在所有像素中出现的概率。该数据即是眉毛的特征。
- 4) 在得到眉毛的特征数据之后, 使用 4.3.2 中相同的方法使用数据建模, 就可以得到 SVM 模型。最终该模型的 c 和 g 参数分别是 1.6 和 0.176777。

4.3.4 眼镜特征的提取与建模

眼镜是人脸上的重要特征, 是否佩戴眼镜会对人的样貌有很大的影响。本课题对眼镜的特征提取是处理其纹理特征。本课题对于眼镜的检测方法是边缘检测, 它能检测图片中色彩变化明显的点^[35]。由于现在的眼镜几乎都在鼻梁上有框架, 所以检测鼻梁区域是否存在镜框能方便地得到是否佩戴眼镜。并且鼻梁区域的皮肤颜色较为相近, 同时也基本不会被遮挡。本课题对眼镜的特征提取步骤如下:

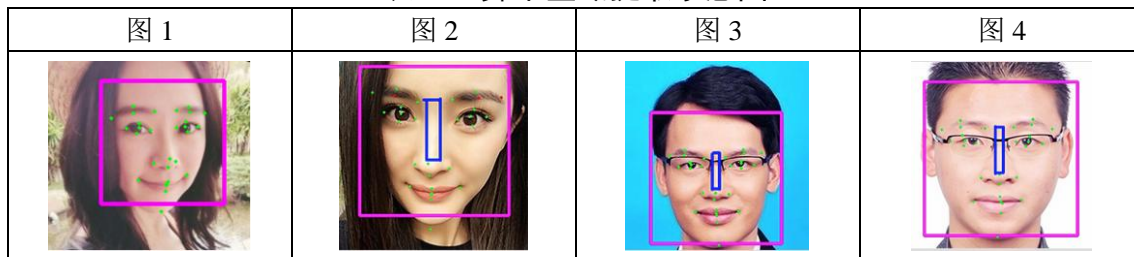
- 1) 根据面部坐标点确定鼻梁的位置, 将鼻梁的图片从原图中切割出来。从图 4-2 中可知, 可以使用左眉毛的右边缘点, 右眉毛的左边缘点以及鼻尖点来确定鼻梁的位置。分别是 mark[2]、mark[3]、mark[20]。利用这几个点可以得到鼻梁的矩形区域, 其计算方式如表 4-6 所示

表 4-6 鼻梁矩形区域的计算方式

标号	含义	计算方法
OriginX	鼻梁原点的横坐标	$\text{OriginX} = \text{mark}[20].x - L / 10$
OriginY	鼻梁原点的纵坐标	$\text{OriginY} = (\text{mark}[2].y + \text{mark}[3].y) / 2$
Width	鼻梁宽度	$\text{Width} = L / 10$
Height	鼻梁高度	$\text{Height} = L / 3$

根据以上方式截取出来的鼻梁区域如表 4-7 所示, 表中共有四幅图, 对其进行了图片编号, 图中显示的是四张人脸的鼻梁区域提取的结果

表 4-7 鼻梁区域提取示意图

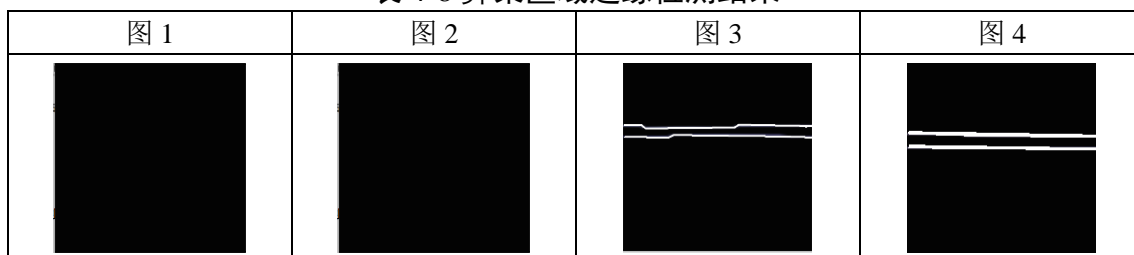


- 2) 对鼻梁区域进行边缘检测：提取到鼻梁区域之后，就可以对其进行边缘检测，本课题使用 canny 算子来实现，主要实现代码如下

```
// 使用 3x3 内核降噪
blur(cannySrc_gray, detected_edges, cv::Size(3,3));
// 运行 Canny 算子
Canny( detected_edges, detected_edges, 90, 90 * cannyRatio, kernel_size );
// 使用 Canny 算子输出边缘作为掩码显示原图像
cannyDst = Scalar::all(0);
cannySrc.copyTo(cannyDst, detected_edges);
```

运算的结果保存在 cannyDst 中，cannyDst 和鼻梁区域大小相同的图片，在有边缘的地方会有白色线条，如表 4-8 所示。

表 4-8 鼻梁区域边缘检测结果



从表 4-8 中可以看出，对于无眼镜的鼻梁，将显示全黑的图片，即图 1 和图 2，而对于有眼镜的鼻梁，即图 3 和图 4，会在图中存在白色线条，说明存在边缘特征，也就是存在镜框。

- 3) 计算检测到的图片中非黑点的数目：得到 cannyDst 之后，对于该矩阵做遍历，得到矩阵中非黑点的数目。代码如下

```
for (int i = 0; i < cannyDst.rows; i++) {
    for (int j = 0; j < cannyDst.cols; j++) {
        if (cannyDst.at<double>(i, j) != 0) {
            result += cannyDst.at<double>(i, j);
            count++;
        }
    }
}
```

以上代码中 count 即是输出也就是是否佩戴眼镜的特征。使用该标签和训练图片即可建立 SVM 模型。

4.4 SVM 分类模块的实现

在建立每个特征对应的 SVM 模型之后,就可以使用模型来预测对应的特征了。本系统使用 OpenCV 提供的 SVM 方法来预测。预测的流程如图 4-9 所示

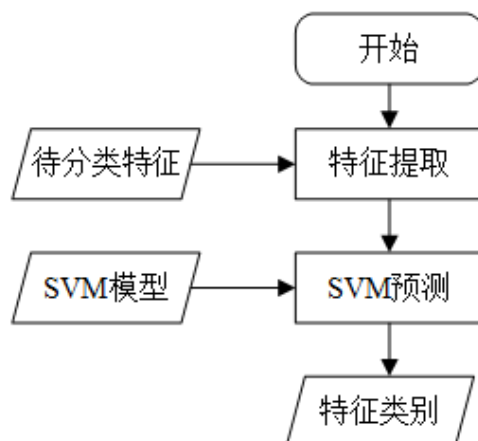


图 4-9 SVM 分类模块预测流程示意图

在本模块内,由 SVMModel.mm 文件实现所有的功能,下面以眼睛大小的实现方式介绍其实现的具体过程,简要代码如下

```

// 1. 判断眼睛大小
NSString *eyeModelPath = [[NSBundle mainBundle] pathForResource:@"part1_eye_model"
ofType:@"xml"]; // 定义判断眼镜的大小的 SVM 模型
CvSVM svmEye; // SVM 初始化
svmEye.load([eyeModelPath UTF8String]); // 加载模型
int predictEyeResult = svmEye.predict(predictEyeMat); // SVM 预测
NSString *eyeresult = [NSString stringWithFormat:@"%d", predictEyeResult];
predictResult.eye = eyeresult;
    
```

以上代码中 eyeModelPath 是判断眼镜大小的 SVM 模型的保存路径,定义一个 CvSVM 之后,使用 load 方法将 model 加载到 svmEye。predictEyeMat 是事先准备好的眼部数据,使用 predict 方法就可以直接得到眼镜大小特征的标签 predictEyeResult,并将结果保存到 predictResult 对象中。对于其他的特征,也是使用对应的 SVM 模型和数据来预测标签。经过 SVM 分类模块的运算之后,人脸部的所有特征均将获取到一个分类结果,其都保存在 predictResult 对象中,此时使用视图模块就可以针对每个特征选取对应的图片组合成一张卡通头像了。

4.5 系统的整体实现

本系统在 iOS 客户端的实现中, 主要是由 ViewController.m 来控制系统的, 其主要方法如下:

- 1) - (void)viewDidLoad: 在界面加载的时候执行, 负责初始化界面和模型, 并为各个按钮加上触发的方法
- 2) - (void)takePhoto:(id)sender: 拍照方法, 点击拍照按钮的时候执行, 首次执行时会要求获取拍照权限, 获得允许后会弹出相机界面, 拍照后提供简单的编辑功能
- 3) - (void)pickPhoto:(id)sender: 选择照片方法, 点击选择照片按钮的时候执行, 首次执行时会要求获取相册权限, 获得允许后会弹出相册界面, 选择照片后提供简单的编辑功能
- 4) - (void)upload:(id)sender: 上传照片方法, 用户确认照片后, 点击上传按钮的时候执行, 将用户所选择的照片上传到服务器处理, 成功后会返回人脸面部坐标信息
- 5) - (void)svmPredict:(FaceModel *)faceModel: SVM 预测方法, 成功获取到人脸面部坐标信息后会被自动执行, 使用 faceModel 和 SVMModel 来预测人脸面部的特征
- 6) - (void)setFaceQImage:(PredictModel *)predictResult: 生成头像方法, 在进行完 SVM 预测之后会自动执行, 利用预测的结果 predictResult 来选取对应的图片, 拼接成一张完整的头像

系统整体的流程图如 4-10 所示

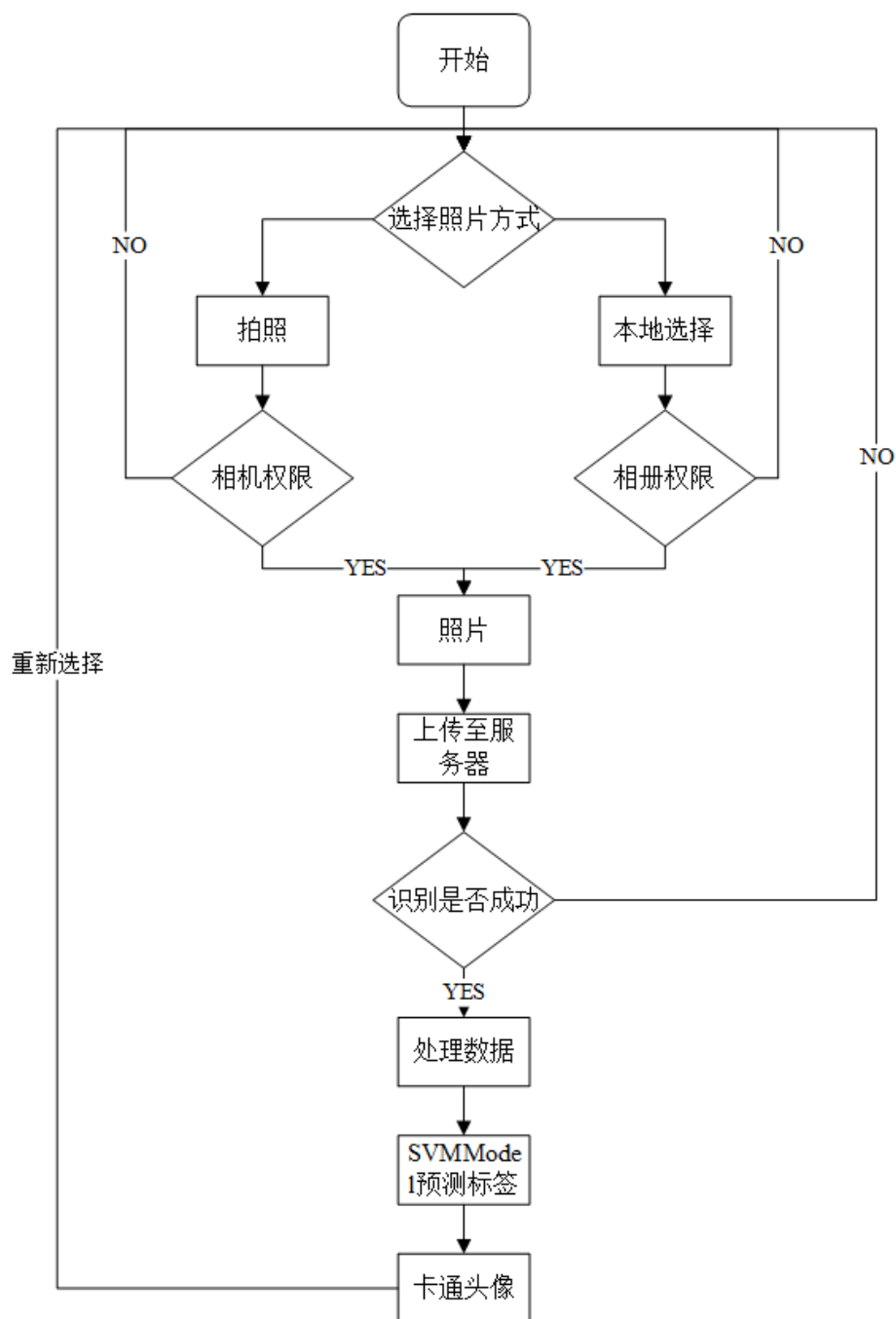


图 4-10 系统整体实现流程图

4.6 系统测试与结果分析

软件测试是系统实现的最后阶段。本节主要对开发的基于 iOS 的头像萌化系统进行测试。首先说明了测试方案，然后给出了 SVM 预测模块测试结果，最终对系统整体进行了测试与分析。

4.6.1 系统的测试方案

由于本系统是处理的是自拍照，且交互过程较为简单，所以从网上下载了 50 张照片作为测试照片（有别于自拍图片库中的照片），其样本如图 4-11 所示。



图 4-11 测试图片样例

首先对这些测试照片做出主观判断，即判断其眼睛的大小、眉毛的浓淡、是否佩戴眼镜等特征，记录在一张表中，表 4-9 列出了本测试集样本的特征分布。

表 4-9 面部特征测试集图片数量表

训练集名称	图片总数（张）	正样本数（张）	负样本数（张）
眉毛浓（正）淡（负）	50	21	29
眉毛厚（正）薄（负）	50	23	27
眼睛大（正）小（负）	50	30	20
眼镜有（正）无（负）	50	15	35
嘴巴开（正）闭（负）	50	13	37
嘴唇厚（正）薄（负）	50	19	31
女头发长（正）短（负）	25	18	7
刘海有（正）无（负）	50	37	13

然后使用本系统对照片进行处理，用 SVM 模型预测其特征并输出卡通图片，将 SVM 输出的特征与主观判断的特征进行对比，得出测试结果。

4.6.2 SVM 预测模块的测试

对测试照片的处理和 4.3 中的特征提取方法相同，提取到特征之后就可以通

过 SVM 模型来进行分类了, 最终得到的结果如表 4-10 所示

表 4-10 测试集的预测结果

训练集名称	图片总数(张)	正样本数(张)	正样本预测正确数(张)	正样本准确率	正样本召回率	负样本数(张)	负样本预测正确数(张)	负样本准确率	负样本召回率
眉毛浓(正)淡(负)	50	21	15	62.5%	71.4%	29	20	76.9%	69.0%
眉毛厚(正)薄(负)	50	23	15	67.7%	65.2%	27	16	66.7%	69.3%
眼睛大(正)小(负)	50	30	25	86.2%	83.3%	20	16	76.2%	80.0%
眼镜有(正)无(负)	50	15	14	87.5%	93.3%	35	33	97.1%	94.3%
嘴巴开(正)闭(负)	50	13	8	60.0%	61.5%	37	25	83.3%	67.6%
嘴唇厚(正)薄(负)	50	19	13	56.5%	68.4%	31	21	77.8%	67.7%
女头发长(正)短(负)	25	18	16	94.1%	88.9%	7	6	75.0%	85.7%
刘海有(正)无(负)	50	37	34	91.9%	81.1%	13	10	76.9%	76.9%

由表 4-10 的测试结果可知, 使用边缘检测方式提取的特征, 其预测准确率较高, 而使用几何特征和 LBP 变换提取的特征, 其准确率略低。

4.6.3 系统的测试与结果

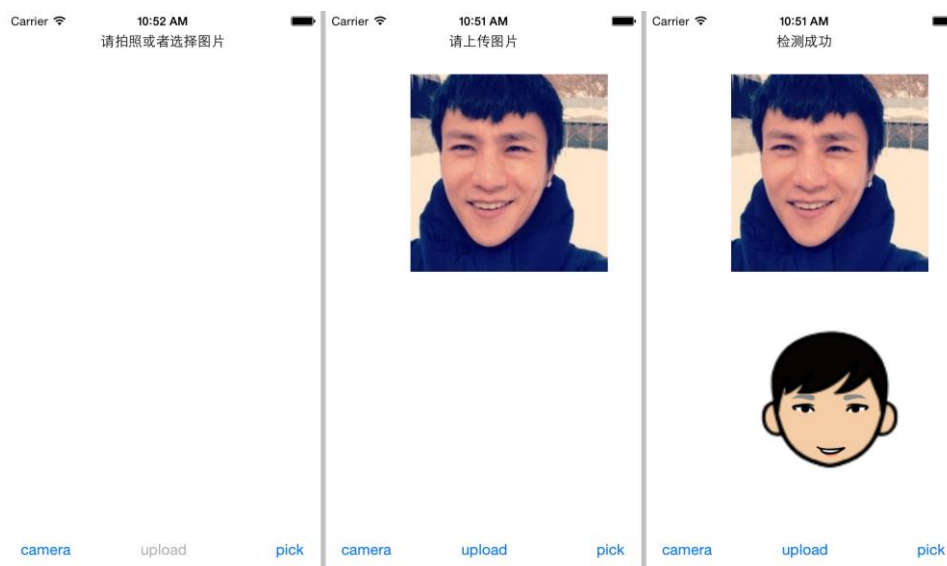







图 4-12 系统界面示意图

图 4-12 展示了系统的三个主要界面。第一个是初始化界面，允许用户选择照片或者拍着照片；第二个界面是选择成功的界面，允许用户上传照片进行检测或者重新选择图片；第三个界面是结果展示界面，展示了本系统的头像结果。在表 4-11 中展示了测试图片样例的头像结果。

表 4-11 测试结果样例表

原图	经过本系统变换之后的结果	原图	经过本系统变换之后的结果
			
			
			

4.7 本章小结

本章从系统开发的软硬件环境开始介绍，随后介绍了人脸特征点定位模块的实现，然后重点并详细介绍了人脸部各个特征的提取与建模过程，包括眼部特征、眉毛特征以及眼镜特征的提取与建模，之后介绍了 SVM 分类模块的实现和系统的整体实现，最后给出了系统的测试和测试结果。

5. 总结与展望

5.1 现有工作总结

考虑到移动互联网的发展迅猛发展,以及用户对便捷生成头像的需求,本课题综合多种图像处理方法,设计并实现了一个可以根据人脸照片自动生成卡通头像的系统。本课题的创新点可以总结为如下方面:

1. 将图像处理和支撑向量机分类的研究成果转换为了实际应用,并结合运用了多种图像处理算法,如 LBP 特征提取算法、边缘检测算法;
2. 可以有效地识别人脸的特征,并对其进行分类,平均识别率在 75%左右;
3. 通过用户的照片直接生成类似于脸萌的卡通头像,非常便捷,解决了用户选择头像的烦恼。

5.2 未来方向展望

本课题基于用户对头像的需求,实现了完整的机遇 iOS 的头像萌化系统,但是仍有不足和需要改善的地方,如下:

1. 没有对更加细致的人脸特征实现区分,例如脸型、胡子、肤色等特征,这些特征也会对人脸的区分有影响,这是需要进一步研究的特征分类;
2. 由于需要与服务器进行通信,所以生成头像所需要的时间较长,且在网络环境不好的时候会生成失败。为了改善这一缺点,可以将人脸识别和坐标点定位的算法放到手机端来实现;

致谢

大学四年的时光眨眼即逝，这大学四年是我学习知识最快也最多的时光。不仅收获到了知识，更交到了非常好的朋友。感谢这大学四年中帮助过我的老师和同学们。

首先我要感谢我的导师许炜副教授。许老师平时工作很繁忙，既要带研究生做项目还要给本科生上课，但是从我做毕业设计开始，许老师在每个阶段都给与我悉心的指导，让我能够很快地进入角色，也能顺利地完成这个课题。并且，许老师平时做事情的态度也影响着我，要求严格，细致入微，成为了我们学习的榜样。

然后，也要感谢在实验室给我帮助的学长学姐们以及同学们。王辉学长指导我如何使用服务器获取人脸数据，肖颖学姐指引我如何开展这个项目，余键夫和我一起讨论项目的方向以及实现的方法。没有他们，这个项目不会像现在这么顺利。

还需要感谢大学四年来的同学们，朋友们。大一的时候参加远征协会，完成了川藏线，体会到了身体在地狱，眼睛在天堂的感觉，在之后的日子遇到困难便会想想那个拼搏的暑假，再大的困难都能克服。大二下学期加入点团队，感谢刘玉老师、钟国辉老师，感谢各位队友，引导我进入互联网这个行业。大三暑假加入腾讯，开始第一份正式的实习工作，感谢各位同事，让我在公司不仅学会了如何做好一个项目，更学会了如何更好地和同事相处。大四去到一家创业公司实习，感谢各位朝气蓬勃的伙伴，能让我体会到创业公司那种努力的氛围。还要感谢大学四年的室友们，我们互相学习，互相提高。

最后，我要感谢我的父母。是他们尊重我的每一个选择，并给我极大的支持。我的每一滴成果都离不开他们的辛劳付出，感谢他们能给我带来这样的生活。

参考文献

- [1] 张世颖. 移动互联网用户生成内容冬季分析与质量评价研究: [博士学位论文]. 吉林: 吉林大学, 2014
- [2] 韩保华. 浅析当今移动互联网发展. 科技视界, 2012(22): 181-182
- [3] Xiaogang Wang, Xiaoou Tang. Random Sampling for Subspace Face Recognition. International Journal of Computer Vision. 2006(1): 86-89
- [4] 赵明华. 人脸检测和识别技术的研究: [博士学位论文]. 四川: 四川大学, 2006
- [5] Venu Govindaraju. Locating human faces in photographs. International Journal of Computer Vision. 1996 (2): 135-139
- [6] 汤德俊. 人脸识别中图像特征提取与匹配技术研究: [博士学位论文]. 辽宁: 大连海事大学, 2013
- [7] Sirovich L, Kirby M. Low-dimensional procedure for the characterization of human face. J. Opt. Soc. Am. A, 1986,4(3): 519-524
- [8] Kim K A, Oh S Y, Choi H C. Facial feature extraction using PCA and wavelet multi-resolution images[C]. Automatic Face and Gesture Recognition, 2004. Sixth IEEE international Conference on. IEEE, 2004: 439-444
- [9] San Z L, Juwei L. Face recognition using the nearest feature line method. IEEE Trans. On NN, 1998, 10(2): 300-311
- [10] Peng H, Zhang D. Dual eigenspace method for human face recognition. Electronics letters. 1997, 33(4): 283-284
- [11] Yuille A L, Hallinan P W, Cohen D S. Feature extraction from faces using deformable templates. International journal of computer vision, 1992, 8(2): 99-111
- [12] T.F.Cootes, C.J.Taylor, D.H.Cooper. Active shape models: their training and application. Computer Vision and Image understanding, 1995, 61(1):38-59
- [13] 张文娇, 闫德勤, 桑雨. 基于支持向量机和粗糙集的图像检索算算法. 计算机工程与应用. 2009(16): 20-31
- [14] 张春艳. 基于支持向量数据描述的累积和控制图: [博士学位论文]. 天津: 天津大学, 2011
- [15] 宋华军. 基于支持向量机的目标跟踪技术研究: [博士学位论文]. 中国科学院长春光学精密机械与物理研究所, 2006
- [16] 鲁珊, 雷英杰, 孔韦韦, 雷阳. 基于空间点特征和改 Hausdorff 距离的图像配准方法. 系统工程与电子技术, 2011(7): 1664-1667
- [17] 陈永义. 支持向量机方法与模糊系统. 模糊系统与数学, 2005(1): 1-11
- [18] 谢菲. 图像纹理特征的提取和图像分类系统研究及实现: [硕士学位论文]. 四川: 电子科技大学, 2009
- [19] 丁二锐, 曾平, 丁阳, 王义峰. 一种新的回归型约简多分辨率相关向量机. 控制与决策, 2008(1): 65-69
- [20] 周昌俊. 学习环境下人脸表情识别系统的研究与开发: [硕士学位论文]. 上海: 上海交通大学, 2011
- [21] Corinna Cortes, Vladimir Vapnik. Support-Vector Networks. Machine Learning . 1995 (3): 102-109
- [22] 林茂六, 陈春雨. 基于傅立叶核与径向基核的支持向量机性能之比较. 重庆邮电学院学

- 报, 2005(6): 647-650
- [23] 阴法明. 基于 OpenCV 的图像处理. 科技信息, 2009(32): 60-63
- [24] Ojala T, Pietikainen M, Maenpaa T. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Transactions on Pattern Analysis and Machine Intelligence . 2002: 71-80
- [25] 刘卓夫, 桑恩方. 一种新的灰度图像识别方法. 声学及电子工程, 2003(4): 6-8
- [26] 王玮, 黄非非, 李见为, 冯海亮. 使用多尺度 LBP 特征描述与识别人脸. 光学精密工程, 2008(4): 696-705
- [27] Zhao G Y, Pietikainen M. Improving Rotation Invariance of the Volume Local Binary Pattern Operator. IAPR Conference on Machine Vision Applications, 2007: 122-129
- [28] 宋本钦, 李培军. 加入改进 LBP 纹理的高分辨率遥感图像分类. 国土资源遥感, 2010(4): 40-45
- [29] Ahonen T, Hadid A, Pietikainen M. Face recognition with local binary patterns. Proceedings of the European Conference on Computer Vision, 2004: 130-135
- [30] Zhang W C, Shan S G, Gao W, et al. Local Gabor Binary Pattern Histogram Sequence (LGBPHS): a Novel Non-Statistical Model for Face Representation and Recognition. Proceedings of the 10th IEEE International Conference on Computer Vision (ICCV'05) , 2005: 75-84
- [31] Rosamund Rawlings. Objective-C: An Object-Oriented language for pragmatists: IEE Colloquium. London: IET. Nov, 1989: 129-139
- [32] 李霞. MVC 设计模式的原理与实现: [硕士学位论文]. 吉林: 吉林大学, 2004
- [33] 陈丽渊. 基于关键部位定位的人脸表情识别的研究与应用: [硕士学位论文]. 上海: 上海交通大学, 2013
- [34] 夏海英. 基于纹理和几何特征的表情分类研究: [硕士学位论文]. 湖北: 华中科技大学, 2011
- [35] 任明罡, 陈岳林, 蔡晓东, 杨超. 基于人脸图片边缘信息的眼镜检测. 软件导刊, 2014(7): 141-143