基于Python的人脸识别技术

摘要：

随着计算机技术的高速发展和人们安全意识的不断提高，传统的身份识别手段已经不能满足人们的日常需求，因此生物特征识别应运而生，它主要运用计算机视觉、图像处理等技术来提取人类的生理特征，并通过深度学习提高识别的准确性和抗干扰性。在多种不同的生物特征识别方法中，人脸识别的方式更为自然，且具有非接触性、非强制性、高并发性和简便性等特点，同时也能运用至证件核实、身份识别、人事考勤等诸多实际领域。Dlib和OpenCV是开源的C++工具库，通过pythonAPI可以调用到图像处理的很多通用算法，可以很便捷地应用于人脸识别技术。python作为一门动态语言，拥有极高地开发效率和基于不同场景和需求完成对程序地开发地能力，因此本文使用python和Dlib、OpenCV来完成对人脸识别程序地开发。程序中结合SURF、HOG和FHOG三种特征算法有机结合实现人脸68特征点地提取，用已训练好地模型进行68特征点至128维向量地转化，运用非关系型数据库mongoDB存储人脸数据，运用pycharm搭建了基于python的人脸识别程序，具有识别图片中人脸的位置并比对数据库中的资料判断人脸身份；通过摄像头实时检测是否有人脸出现并识别其身份；通过图片向数据库中新增人脸身份信息等功能。本文对程序各个模块和整体流程也进行了测试，测试结果良好，基本完整实现了基于python的人脸识别技术。

关键词：

python，dlib，opencv，人脸识别

Abstract

With the rapid development of computer technology and the continuous improvement of people's security awareness, traditional identification methods can no longer meet people's daily needs. Therefore, biometric identification came into being. It mainly uses computer vision, image processing and other technologies to extract human Physiological characteristics, and improve the accuracy and anti-interference of recognition through deep learning. Among the many different biometric recognition methods, the face recognition method is more natural and has the characteristics of non-contact, non-mandatory, high concurrency and simplicity. It can also be applied to document verification, identity recognition, Many practical areas such as personnel attendance. Dlib and OpenCV are open source C ++ tool libraries. Many common algorithms for image processing can be called through pythonAPI, which can be easily applied to face recognition technology. As a dynamic language, Python has extremely high development efficiency and the ability to complete program development based on different scenarios and requirements. Therefore, this article uses Python, Dlib, and OpenCV to complete the development of face recognition programs. The program combines the three feature algorithms of SURF, HOG, and FHOG to realize the extraction of 68 feature points on the face. The trained model is used to transform the 68 feature points to 128-dimensional vectors. The nosql database mongoDB is used to store the face data. , Using pycharm to build a python-based face recognition program, which recognizes the position of the face in the picture and compares the information in the database to determine the identity of the face; detects whether a human face appears and recognizes its identity in real time through the camera; to the database through the picture Added features such as face identity information. This article also tested each module and the overall process of the program. The test results are good, and the face recognition technology based on python is basically completely implemented.

Keywords:

Python，opencv， dlib， Face recognition

目录

第一章 绪论

第一章 绪论

1.1 研究背景和意义

随着信息技术的飞速发展，它已经在人们不知不觉中渗入生活的各个方面，从日常的出行、饮食、购物、通信，到工作中的自动化办公、远程会议等，信息技术发展在为人们带来便利的同时也在信息安全中埋下隐患。利用漏洞非法的套取个人信息，刺探个人隐私等安全问题屡见不鲜，人们也开始重视信息安全的相关问题，而传统的身份识别方法诸如：身份证件、密码、钥匙等已经不足以在日新月异的非法破解下保护人们的信息安全，因此，一种更安全、更可靠、更高效、也更难以入侵与破解的身份识别技术——生物特征识别技术就应运而生，在如深度学习和机器学习的人工智能技术的加持下飞速发展和成熟，得以进入人们的视野并在实际应用中取得发展与突破。

生物特征识别技术主要运用计算机视觉和图形处理技术处理并提取人类的生理特征信息，通过大量数据进行深度学习的训练，从而能达到对个人身份完成验证的效果。每一个人都有各自不同的生理特征和行为特征，这些特征点都是可以被测量并记录的，同时也能通过各种手段和技术进行比较和验证。生理特征包括指纹、虹膜、掌纹、人脸特征、DNA等，行为特征则包括声音、笔记、步态等；对于这些生物特征，目前已经研发成熟并能投入实际使用的有：人脸、指纹、声音等，这些识别方式不需要人们随身携带各种证件、钥匙或记忆繁琐的口令，由此免去了因丢失或遗忘所带来的各种麻烦与隐患，只需要认定人本身即可，这无疑极大地提高了安全性。此外，虽然人脸识别相较于虹膜或指纹等识别方式地正确识别率略低，但是人脸识别不直接接触人体，避免了对人体的侵害性，而且这种识别模式更为自然、便捷，因此人脸识别也就顺理成章的成为了目前更容易被广泛接受和应用的生物识别技术。

就当下生物特征识别技术的研究进度和发展进程而言，人脸识别相较于其他生物识别技术的优势主要有以下几点：

1） 非接触性：

在获取人脸图像信息和验证人脸图像信息的过程中，图像采集设备不会直接以物理方式接触到用户，从而避免用户产生被本体侵犯负面情绪，更容易被用户所接纳。

2） 非强制性：

人脸图像的采集不需要强制用户的配合，可以在用户毫无察觉的情况下验证其身份，使得身份验证的过程更为自然和隐蔽。

3） 高并发性：

人脸识别技术可以针对同一幅图像中出现的多个人脸图像进行识别和验证，提高了验证效率。

4） 简便性：

人脸识别对硬件设备的要求底，只需要基础的摄像头或相机就能完成对人脸图像的采集。

基于以上优势，人脸识别技术得以运用于广泛领域之中，诸如：

1） 国家安全

国家安全是我国能长久稳定发展的必要条件，人脸识别技术的发展和应用能对维护国家安全稳定起到重要的帮助作用，人脸识别技术为公安机关提供了一种高效、快速、隐蔽、准确地手段来分辨犯罪分子的身份，并提供持续性的追踪、定位其位置的方法。在人脸识别技术的帮助下，公安机关可以迅速抓捕犯罪分子，排除其造成更大人身伤害或财产损失的隐患，从而能促进社会的和谐与稳定。

2） 电子支付

随着网络的发展与普及，淘宝、京东等电商平台为人们生活购物提供了极大的遍历，而电子支付也就成为了电子商务中炙手可热的产品，人脸识别技术可以有效地为用户提供身份认证，消除用户身份被盗用、受到财产损失的隐患，保障了用户的信息安全，使用户能更放心的使用电子支付平台进行交易。

3） 访问权限

对于一些涉及高安全、高机密单位，如政府机关、医院、公安机关、法院，或是一些保密性高的企业，人脸识别技术可以成为构建其安全防卫系统的重要基石。所属人员的进出都需要通过人脸识别技术的验证，个人电脑、单位或企业的机房、服务器等设备也都需要进行人脸识别的身份验证其权限后方可使用，这样可以高效且便捷的保证企业或单位的信息安全性和机密性，同时也不会让用户耗费大量的时间精力于繁琐的身份验证环节，在保证安全机密的条件下简化了流程，提高了工作效率。

4） 核对、查验证件

在乘坐飞机、火车，入住酒店，参加各类考试，在各政府机构办理特定业务的过程中，核对核实身份信息都是必不可少的环节，人脸识别技术可以高效迅速的查验持证人是否和证件所有者为同一人，可以加快业务的办理速度或是人口流通速度，在保证安全性、有效性的同时极大优化了了办理人和受理机构的办理流程，提高了用户体验感。

5） 人事考勤

不论学校、企业或政府机关，考勤都是日常不可或缺的环节，利用人脸识别技术判断该学生或员工是否按时到岗，能有效的防止传统考勤制度中仿冒、造假的缺陷与漏洞。

由此可见，人脸识别技术的研究与发展不仅对于国家安全和信息安全有着重要的作用，而且同样也为人们的日常生活、工作和出行带来了舒适感和效率的提高。一言蔽之，人脸识别技术是当下科技高速发展的环境中解放和发展生产力的重要助力之一，对于人脸识别技术在政治、经济、文化方面的良好运用将会极大的推动生产力的发展，使人民富裕幸福，使国家繁荣富强。

1.2 国内外发展现状

近年来，人工智能技术在各个领域有着令人惊叹的发展与开拓，人脸检测和人脸识别技术的发展应用也是各科技领军企业的重中之重。早在上世纪七十年代，科学家们就已经开始着手这一领域的研究，当时的研究重点在于以几何特征方法对人脸进行解析，以这种方式处理人像则需要极高的图像质量，即使如此，分析获得的特征点也较少，在比对相似度方面也有较高的错误率，因此在这一时间段，人脸识别技术暂时仅仅停留在实验室阶段，并未与人们的现实生活产生任何可以挂勾的领域。

到上世纪九十年代，随着计算机硬件的急速发展，计算机的计算、处理数据的能力大幅提升，人脸识别技术也趁势而起，以代数特征和统计理论为基础Pentland提出了名为EigenFace的方法，这个方法成为后续所有人脸识别技术的衍生算法的基石。而后，另一个名为FisherFace的方法被提出，他创新性地提出先对人脸进行降维处理，然后采用线性判别方法对降维后得到的数据进行分析，成功地解决了特征提取的问题，由此，人脸识别技术也渐渐步入成熟阶段，开始从实验室渐渐转向实际应用方面发展。

进入二十一世纪后，人脸识别技术的研究重心逐渐从识别人脸的方法转向了人脸识别的精度，研究者们力图消除光照、肤色或复杂背景对识别结果的影响。同时，随着神经网络和深度学习的兴起，人脸识别技术可以利用深度学习从原始图像空间直接学习特征的表示方法。

1.3论文研究内容和结构

本文对基于python的人脸识别技术进行了设计与实现。利用OpenCV和Dlib两个库开发实现人脸识别、提取特征点并转化为128维向量的处理，基于pycharm进行程序的搭建，非关系型数据库mengoDB储存数据，numpy进行数据分析，最终实现人脸识别技术的程序。

第一章 绪论。介绍了人脸识别技术研究的背景和意义，同时还对论文研究的主要内容和总体结构进行了详细的介绍。

第二章 系统相关技术的介绍。对OpenCV和Dlib进行简单介绍，并分析python语言的优势，同时也对sqlite和numpy做了简要概述。

第三章 系统设计。介绍了系统的功能设计、模块设计，对系统将要实现的具体功能作相关介绍和阐述。

第四章 系统实现。对具体的模块和功能的实现进行详尽的介绍。

第五章 系统测试。对系统的算法、功能和整体方面进行了相关测试，证明了系统的可靠性。

第六章 结论和展望。对本文进行的工作进行了总结，并介绍后续可能的扩展。

第二章 系统相关技术的介绍

2.1 Python简介

Python是一种易于学习，功能强大的编程语言。它具有高效的高级数据结构和简单但有效的面向对象编程方法。Python优雅的语法和动态类型以及其解释的特性使它成为大多数平台上许多领域中脚本编写和快速应用程序开发的理想语言。Python解释器可轻松扩展为以C或C++（或从C调用的其他语言）实现的新功能和数据类型。Python也适合作为可定制应用程序的扩展语言。Python可在Windows，Mac OS 和Linux操作系统上使用。Python是一种解释性语言，因为不需要编译和链接，因此可以在程序开发过程中节省大量时间。解释器可以交互使用，这使得在自下而上的程序开发过程中可以轻松地尝试语言的功能，编写一次性程序或测试功能。

Python易于使用，但它是一种真正的编程语言，与Shell脚本或批处理文件相比，它为大型程序提供了更多的结构和支持。另一方面，Python还比C提供更多的错误检查，并且作为一种非常高级的语言，它内置了高级数据类型，例如灵活的数组和字典。由于Python具有更通用的数据类型，因此它比Awk甚至Perl都适用于更大的问题域。

Python是一种解释性语言，因为不需要编译和链接，因此可以在程序开发过程中节省大量时间。解释器可以交互使用，这使得在自下而上的程序开发过程中可以轻松地尝试语言的功能，编写一次性程序或测试功能。

2.2 Dlib简介

Dlib是使用C++编写的跨平台软件库，对外提供C++接口和python接口，是一个轻量级但又高效的开源库。它在工业和学术界被广泛的使用，包括机器人技术、嵌入式设备、移动电话和大型高性能计算环境。

Dlib的主要特点和功能包括：

• 高品质可移植代码

o 良好的单元测试覆盖率。代码的单元测试行与代码的库行的比率约为1到4。

o 该库可以在Windows、Mac Os、Linux等系统上使用

o 所有特定于操作系统的代码都隔离在OS抽象层中，这些抽象层应保持尽可能小。库的其余部分位于操作系统抽象层之上，或者是纯ISO标准C++。

• 机器学习算法

o 深度学习

o 基于传统SMO的支持向量机进行分类 和 回归

o 大规模分类 和回归的降秩方法

o 相关向量机进行分类 和回归

o 通用多类分类工具

o 一个多类SVM

o 一种解决与结构支持向量机有关的优化问题的工具 。

o 用于序列标记的结构化SVM工具

o 用于解决分配问题的结构化SVM工具

o 用于图像中对象检测的结构化SVM工具以及用于对象检测的功能更强大（但速度较慢）的深度学习工具。

o 用于在图中标记节点的结构化SVM工具

o 大规模SVM-Rank实现

o 在线内核RLS回归算法

o 在线SVM分类算法

o 半定度量学习

o 在线核化质心估计器 /新颖性检测器和离线支持向量一类分类

o 聚类算法：线性 或核k均值， Chinese Whispers和 Newman聚类。

o 径向基函数网络

o 多层感知器

• 图形模型推断算法

o 连接树算法用于贝叶斯网络中的精确推断。

o Gibbs采样器 markov链蒙特卡洛算法，用于在贝叶斯网络中进行近似推断。

o 在链结构图， Potts或 通用因子图中执行MAP推理的例程 。

• 图像处理

o 用于读取和 写入常见图像格式的例程。

o 各种像素类型之间的自动色彩空间转换

o 常见的图像操作，例如边缘查找和形态学操作

o 实现SURF， HOG和FHOG 特征提取算法。

o 用于检测图像中对象的工具，包括 正面面部检测和 对象姿态估计。

o 高品质人脸识别

2.3 OpenCV简介

OpenCV项目于1999年正式启动，最初是一项英特尔研究计划，旨在推进CPU密集型应用程序的开发，这是一系列项目的一部分，其中包括实时 光线追踪和3D显示墙。OpenCV的主要特点为：

通过不仅为基本视觉基础设施提供开放代码而且优化代码来推进视觉研究。不再需要重新发明轮子了。

通过提供开发人员可以在其上构建的通用基础结构来传播视觉知识，从而使代码更易于阅读和传递。

通过免费提供可移植的，性能优化的代码，从而提高基于视觉的商业应用程序的使用-无需许可即可公开或免费释放代码的许可证。

OpenCV的应用领域包括：

• 2D和3D功能工具包

• 自我估计

• 面部识别系统

• 手势识别

• 人机交互（HCI）

• 移动机器人

• 运动理解

• 对象识别

• 细分与识别

• 立体视觉立体视觉：从2个摄像头的深度知觉

• 运动结构（SFM）

• 运动追踪

• 增强现实

2.4 mengoDB简介

MongoDB是一个跨平台的面向文档的数据库程序。分类为NoSQL数据库程序，MongoDB中的记录是一个文档，它是由字段和值对组成的数据结构。MongoDB文档类似于JSON对象。字段的值可以包括其他文档，数组和文档数组。使用文档的优点是：文档（即对象）对应于许多编程语言中的本机数据类型;嵌入式文档和数组减少了对昂贵连接的需求;动态模式支持流畅的多态性。

主要功能:

高性能:

MongoDB提供高性能的数据持久性。尤其是对嵌入式数据模型的支持减少了数据库系统上的i/o活动;索引支持更快的查询，并且可以包括来自嵌入式文档和数组的键。

丰富的查询语言:

MongoDB支持丰富的查询语言以支持读取和写入操作（CRUD）以及：资料汇总;文本搜索和地理空间查询。

高可用性

MongoDB的复制工具（称为副本集）提供：自动故障转移;数据冗余。

副本集是一组维护相同数据集的MongoDB服务器，可提供冗余并提高数据可用性。

水平可伸缩性:

MongoDB提供水平可伸缩性作为其核心功能的一部分：分片在一组计算机集群分布数据。

支持多种存储引擎:

MongoDB支持多个存储引擎：WiredTiger存储引擎（包括对静态加密的支持）;内存中存储引擎。

2.5 本章小结

本章主要介绍了本文所使用的相关技术，介绍了基础编程语言Python，以及实现人脸识别技术的主要依赖库Dlib，及其主要功能与特点。

第三章 系统设计

3.1 程序的功能性需求分析

3.1.1 读入待识别的图像

实现人脸识别技术的第一步就是要将待识别的图像读入程序，以待用于人脸图像的识别和后续特征点的提取等操作。

读入单幅图片的功能是基于OpenCV的imread函数，它可以读取.JPG/.BMP/.JEPG/.PNG等多种格式的图片文件，并可以手动设置参数将图片转化为灰度、彩色等不同的格式。

读入视屏文件的功能是基于OpenCV的VideoCapture函数，可以读取.AVI/.MPG等格式的视频文件，将该函数的参数直接设置为0则会直接从当前设备的默认摄像头获取实时录像数据，将参数设置为1则可以从外部连接的其他设备获取实时录像数据。

3.1.2 识别人脸位置

得到图像后，就需要识别出图片内人脸的位置，以此为依据来为后续提取特征点提供更精确的数据.

确定人脸位置的功能基于Dlib的get\_frontal\_face\_detector函数，该函数会将人脸在图片中的具体位置范围作为返回值返回，利用OpenCV中提供的框图函数rectangle则能将人脸框出并显示在图片上

3.1.3 将人脸图像转化为可计算的128维向量

要进一步对人脸进行比对、辨别，则需要将人脸图片进一步处理为可计算的128维向量值

该功能基于Dlib中提供的已经训练完成的图像预测和人脸识别模型，加载这两个模型后，即可将目标图片转化为128维的向量值

3.1.4 人脸数据库

想要识别人脸并判别其身份，建立一个可以储存人脸身份信息的数据库必不可少，非关系型数据库mongoDB是一个不错的选择

选取清晰度较高的正脸照，文件名命名为人名，将人脸数据转化为128维向量后以字符串的形式保存进数据库中，主要存入三组数据人脸名字，人脸的128维向量和128维向量的欧氏长度（方便后续数据筛选）

3.1.5 人脸数据比对

人脸数据比对采用欧氏距离的方式，若两组128维特征向量的距离小于0.6（官方推荐的阈值），则可以认定为同一人

3.2 程序的整体设计

3.2.1 静态人脸测试模块

在该模块中主要是为了检测人脸识别的准确性和能力，测试当脸部有不同程度的遮挡物（如眼镜、手等）时能否正确识别到人脸位置，当人脸的方向发生变化（如侧脸）能否正确检测到人脸。

3.2.2 添加人脸数据到数据库模块

该模块主要是为了添加可作为判别身份的128维向量数据和人名信息，通过对指定位置文件夹中文件的提取，并识别其中的人脸，将其转化为128维向量，储存到mongoDB中。

3.2.3 静态人脸识别模块

该模块主要是实现了提取图片中人脸的数据并于数据库中已有的信息比对，找出与其欧氏距离小于0.6并且最小的一组数据，输出其人名。

3.2.4 动态人脸识别模块

该模块主要实现在一段录像视频文件或摄像头得到的影像中识别出其中人脸的位置，并在数据库中找出与该人脸128维向量欧氏距离小于0.6且最近的一组数据，输出其人名。

3.3 本章小结

本章介绍了基于python的人脸识别技术的程序设计，人脸识别系统主要包含识别静态人脸，识别动态人脸，录入数据库和测试模块四个部分，其中人脸的识别和比对是本文的重要内容，基于python和Dlib来实现，数据库使用的是mongoDB，也就是应用很广泛的非关系型数据库，本文后续的程序实现就是基于本章的程序设计来完成的。

第四章 程序实现

4.1模块实现

4.1.1 静态人脸测试模块

本程序中将待测试的图片以不同风格分为三类：normal为普通情况无特殊遮挡的人脸；shelter为面部存在眼镜或手等遮挡物不同程度遮挡的人脸；side为方向不同的人脸，如侧脸或半侧脸等；将不同类型的照片依次放入对应特点的文件夹就可以开始测试了。

静态人脸测试模块的第一步是从上述这些文件夹中读取图片，然后利用Dlib的人脸识别函数检测图片中所包含的人脸，如果识别出人脸，则将人脸框选并保存到对应文件夹（normal，shelter或side）下的result子文件夹。

静态人脸测试模块的代码实现如下：

models.py 该文件包含一些预编译的变量或函数，提高代码复用性

import dlib

import pymongo

# 模型及初始化

predictor\_path = "models/shape\_predictor\_68\_face\_landmarks.dat"

face\_rec\_model\_path = "models/dlib\_face\_recognition\_resnet\_model\_v1.dat"

shape\_predictor = dlib.shape\_predictor(predictor\_path)

face\_rec\_model = dlib.face\_recognition\_model\_v1(face\_rec\_model\_path)

detector = dlib.get\_frontal\_face\_detector()

connection = pymongo.MongoClient('localhost', 27017)

db = connection['graduation']

main\_sheet = db['main']

detect\_face\_interference.py 人脸检测测试模块

import os

import cv2

from models import \*

path\_glasses = 'photos/glasses'

path\_normal = 'photos/normal'

path\_shelter = 'photos/shelter'

path\_side = 'photos/side'

def detect\_face(path):

photos = [file for file in os.listdir(path) if file.endswith('.jpg')]

fault = []

# window = dlib.image\_window()

for photo in photos:

target = cv2.imread(path + '/' + photo, cv2.IMREAD\_COLOR)

dets = detector(target, 1) # 获取目标的特征点,第二个参数设置为1，否则当出现多张人脸时会仅显示三处

if not dets:

fault.append(photo)

continue

'''window.clear\_overlay() # 清除显示窗

window.set\_image(target) # 加载图片

window.add\_overlay(dets) # 框选人脸'''

for index, face in enumerate(dets):

cv2.rectangle(target, (face.left(), face.top()), (face.right(), face.bottom()), (255, 0, 0), 2)

# 参数设置： 目标文件，（左，上），（右，下），颜色，边框粗细

dlib.save\_image(target, path + '/result/' + photo)

print('over')

print(fault)

detect\_face(path\_normal)

detect\_face(path\_glasses)

detect\_face(path\_shelter)

detect\_face(path\_side)

4.1.2 添加人脸数据到数据库模块

本程序中数据库使用mongoDB，将指定文件夹add下的图片读取，检测其中的人脸，提取128维向量数据，以数组形式保存，同时计算出该128维向量的欧氏长度，以便后续查找时能避免无意义的计算，加快运行速度，最后以

{'name': name, 'vector': vector,'length': length}的形式保存。

代码实现如下：

models.py

import dlib

import pymongo

# 模型及初始化

predictor\_path = "models/shape\_predictor\_68\_face\_landmarks.dat"

face\_rec\_model\_path = "models/dlib\_face\_recognition\_resnet\_model\_v1.dat"

shape\_predictor = dlib.shape\_predictor(predictor\_path)

face\_rec\_model = dlib.face\_recognition\_model\_v1(face\_rec\_model\_path)

detector = dlib.get\_frontal\_face\_detector()

connection = pymongo.MongoClient('localhost', 27017)

db = connection['graduation']

main\_sheet = db['main']

add\_data.py

import os

import cv2

from models import \*

import numpy

def get\_data(photo):

img = cv2.imread('add/' + photo)

name = photo.split('.jpg')[0]

dets = detector(img, 1)

for index, face in enumerate(dets):

shape = shape\_predictor(img, face)

face\_descriptor = face\_rec\_model.compute\_face\_descriptor(img, shape)

vector = list(face\_descriptor)

length = numpy.linalg.norm(numpy.array(face\_descriptor))

info = {'name': name, 'vector': vector, 'length': length}

main\_sheet.insert\_one(info)

photos = [file for file in os.listdir('add') if file.endswith('.jpg')]

for photo in photos:

get\_data(photo)

print(photo, ' Done!')

4.1.3 静态人脸识别模块

本程序将提取制定文件夹find\_who下待识别的图片，提取出其中人脸数据并与数据库中的文件相比对，查找出该人脸的身份

实现代码如下：

models.py

import dlib

import pymongo

# 模型及初始化

predictor\_path = "models/shape\_predictor\_68\_face\_landmarks.dat"

face\_rec\_model\_path = "models/dlib\_face\_recognition\_resnet\_model\_v1.dat"

shape\_predictor = dlib.shape\_predictor(predictor\_path)

face\_rec\_model = dlib.face\_recognition\_model\_v1(face\_rec\_model\_path)

detector = dlib.get\_frontal\_face\_detector()

connection = pymongo.MongoClient('localhost', 27017)

db = connection['graduation']

main\_sheet = db['main']

find\_who.py

import cv2

import os

import numpy

from models import \*

def pair(img):

dets = detector(img, 1)

target = []

for index, face in enumerate(dets):

shape = shape\_predictor(img, face)

face\_descriptor = face\_rec\_model.compute\_face\_descriptor(img, shape) # 计算人脸的128维的向量

target.append(numpy.array(face\_descriptor))

result = []

if target: # 识别出的人脸

for face in target: # 对每个人脸进行比对

length = numpy.linalg.norm(face)

temp = [[x['name'], x['vector']] for x in main\_sheet.find({'length': {'$lt': length + 0.6,

'$gt': length - 0.6}})]

temp1 = []

for name, vector in temp:

if numpy.linalg.norm(face - numpy.array(vector)) < 0.6:

temp1.append([name, numpy.linalg.norm(face - numpy.array(vector))])

if temp1: # 在阈值0.6的条件下，找出相似度最高的选项

similar = ['', 0.6]

for i in temp1:

if i[1] < similar[1]:

similar = i

result.append(similar[0])

else:

result.append('unpaired')

print(result)

return '-'.join(result)

else:

return 'unrecognized'

files = [file for file in os.listdir('photos/find\_who') if file.endswith('.jpg')]

for file in files:

cv2.imwrite('photos/find\_who/result/' + pair(cv2.imread('photos/find\_who/' + file)) + file,

cv2.imread('photos/find\_who/' + file))

print(file + ' Done!')

4.1.4 动态人脸识别模块

本程序主要实现了对视频文件或摄像头获取的实时影像文件中的人脸的检测并比对数据库中的数据找出该人脸的身份

models.py

import dlib

import pymongo

# 模型及初始化

predictor\_path = "models/shape\_predictor\_68\_face\_landmarks.dat"

face\_rec\_model\_path = "models/dlib\_face\_recognition\_resnet\_model\_v1.dat"

shape\_predictor = dlib.shape\_predictor(predictor\_path)

face\_rec\_model = dlib.face\_recognition\_model\_v1(face\_rec\_model\_path)

detector = dlib.get\_frontal\_face\_detector()

connection = pymongo.MongoClient('localhost', 27017)

db = connection['graduation']

main\_sheet = db['main']

detect\_in\_video.py

import numpy

import cv2

from models import \*

def find(frame):

dets = detector(frame, 1)

if not dets:

return 'no face'

else:

target = []

for index, face in enumerate(dets):

shape = shape\_predictor(frame, face)

face\_descriptor = face\_rec\_model.compute\_face\_descriptor(frame, shape) # 计算人脸的128维的向量

target.append(list(face\_descriptor))

result = []

if target: # 识别出的人脸

for face in target: # 对每个人脸进行比对

for face in target: # 对每个人脸进行比对

length = numpy.linalg.norm(face)

temp = [[x['name'], x['vector']] for x in main\_sheet.find({'length': {'$lt': length + 0.6,

'$gt': length - 0.6}})]

temp1 = []

for name, vector in temp:

if numpy.linalg.norm(face - numpy.array(vector)) < 0.6:

temp1.append([name, numpy.linalg.norm(face - numpy.array(vector))])

if temp1: # 在阈值0.6的条件下，找出相似度最高的选项

similar = ['', 0.6]

for i in temp1:

if i[1] < similar[1]:

similar = i

result.append(similar[0])

else:

result.append('unpaired')

return '-'.join(result)

else:

return 'unrecognized'

cap = cv2.VideoCapture(0) # 获取摄像头

# 如果要实现视频的读取，就将cap = cv2.VideoCapture('test.avi');循环条件改为cap.isOpened()

while 1:

ret, frame = cap.read() # 得到摄像头的参数

dets = detector(frame, 1)

for index, face in enumerate(dets): # 人脸框图

cv2.rectangle(frame, (face.left(), face.top()), (face.right(), face.bottom()), (255, 0, 0), 2)

cv2.putText(frame, find(frame), (40, 50), cv2.FONT\_HERSHEY\_PLAIN, 2.0, (0, 0, 255), 2) # 文本

cv2.imshow('frame', cv2.cvtColor(frame, cv2.IMREAD\_COLOR))

if cv2.waitKey(1) & 0xFF == ord('q'): # 输入q结束程序

break

cap.release()

cv2.destroyAllWindows()

4.2 算法实现

使用数据库在搜索最匹配的人脸时，为了避免对数据库中的每一项都进行128维数据的欧式距离求解从而导致计算量过大，影响程序的运行时间，因此在存入128维数据时同时也计算出该向量的欧式长度，这样在计算欧氏距离前先比较两者欧式长度的差值是否小于阈值，若欧式长度的差值小于阈值，则其欧式距离必不可能小于阈值，以此方法进行筛选以减少计算量。

4.3 本章小结

本章详细介绍了基于python的人脸识别技术的程序实现，通过对第三章中程序设计的详细分析，以及对Dlib中类函数的详细学习、研究，基于Dlib19.16并结合OpenCV、numpy等库和mongoDB数据库实现了人脸识别技术的功能实现。

第五章 系统测试。

5.1 功能测试

5.1.1 静态人脸测试模块

基于python实现的人脸识别技术的静态人脸测试可以实现对图像的读取，识别出图像内的人脸并将其框选。经过测试，大部分遮挡不严重的，以及人脸部分大部分清晰可见的图片可以被正确识别出人脸，对于少部分人像被遮挡过于严重或不清晰的图片则无法识别出人脸。

5.1.2 添加人脸数据到数据库模块

通过读取指定文件夹中的人像图片，能正确将图片人脸的特征点提取出并且转化为128维的向量，以文件名给出的字段为人名，正确将数据录入数据库中。

5.1.3 静态人脸识别模块

通过在指定位置读取待识别的图片，并对其进行人脸识别操作，提取特征点，转化为128维向量，同时与数据库内的数据进行欧氏距离计算，找出欧氏距离小于0.6并最小的一项，将其输出。

5.1.4 动态人脸识别模块

通过对摄像头的调用或视频文件的读取，识别动态图像中的人脸，对其提取特征点，转化为128维向量后与数据库内的数据进行欧式距离计算，找出欧氏距离小于0.6并最小的一项，将其输出在屏幕上。

第六章 总结与展望