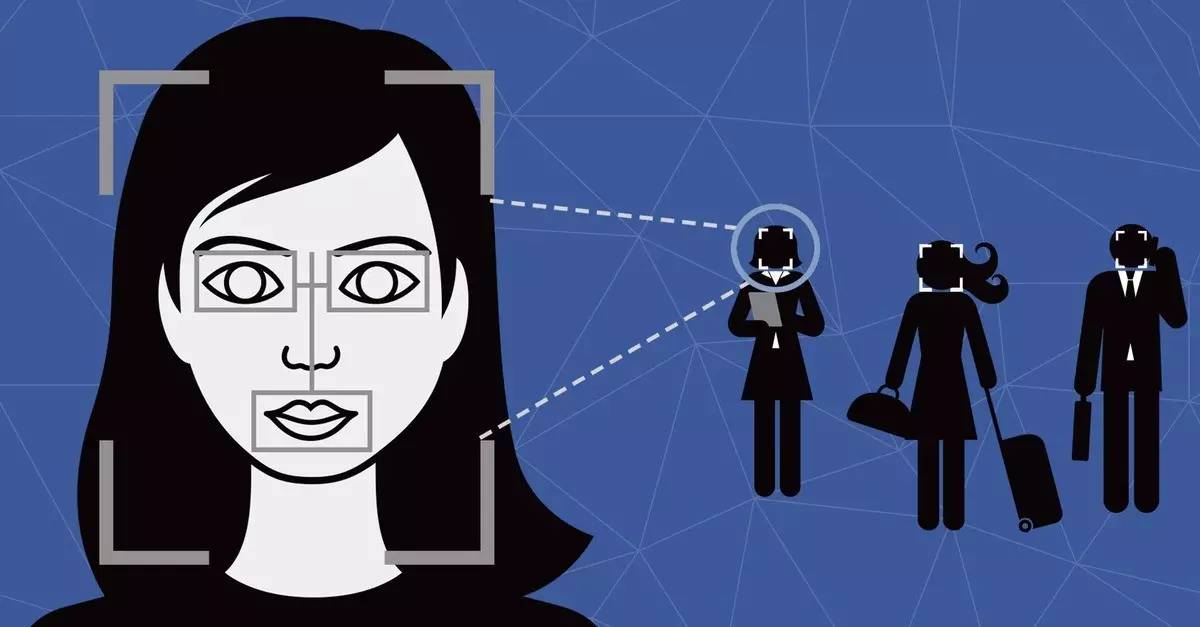
刷脸签到系统回顾

2017-01-19 19:09:14

Table of Contents

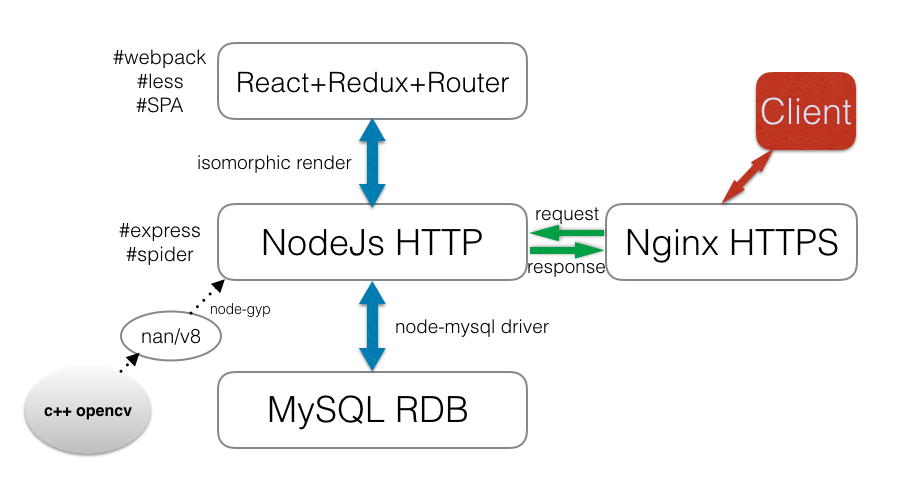
[毕设 👉 南师大刷脸签到系统 👉 face.moyuyc.xyz](https://face.moyuyc.xyz) 

写该文是为了准备写毕业论文的材料，所以文字介绍较多，比较面向大众程序员。

## 任务概要

本刷脸系统主要分为学生签到、人脸录入、管理员模块。管理员模块，可以对学生的人脸样本，学生相信操作；该系统核心模块：学生签到，将动态监控摄像头，一旦识别出人脸便向服务器发出请求进行预处理 => 人脸检测 => 人脸比对。该系统将采用 Web 架构实现，前端使用react+redux+router+webpack技术栈，结构清晰，复用率高；后端将采用 nodejs+express 搭建服务器，C++ 实现核心人脸识别比对算法，通过 js 调用 C++ 核心算法；后续可以使用 electron 将前端界面打包成跨平台app，方便师生使用。

## 技术栈（Technology Stack）



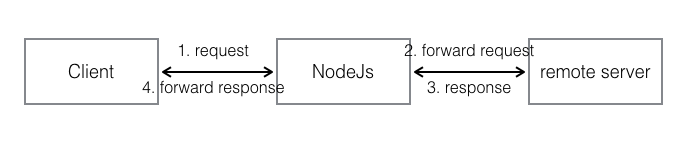
ClipboardImage

下面进行一些较为粗略的介绍，蜻蜓点水说说涉及的技术

### 前端（Front-End）

1. 单页Web应用（single page web application，SPA），就是只有一张Web页面的应用。单页应用程序 (SPA) 是加载单个HTML 页面并在用户与应用程序交互时动态更新该页面的Web应用程序。[1] 浏览器一开始会加载必需的HTML、CSS和JavaScript，所有的操作都在这张页面上完成，都由 JavaScript 来控制。因此，对单页应用来说模块化的开发和设计显得相当重要。
2. 使用主流 [Webpack](https://webpack.github.io/) 构建，进行前端模块自动化管理。
3. 使用Facebook提出的 [React](https://facebook.github.io/react/) , 将HTML DOM进行上层抽象，提出Component概念，一套理念，实现了Server render, Web UI, mobile UI的统一。 Learn Once, Write Anywhere
4. [Redux](https://github.com/reactjs/redux)，随着 JavaScript 单页应用开发日趋复杂，JavaScript 需要管理比任何时候都要多的 state (状态)，state 在什么时候，由于什么原因，如何变化已然不受控制。 当系统变得错综复杂的时候，想重现问题或者添加新功能就会变得举步维艰, Redux则是为了解决该痛点而产生。
5. [React Router](https://github.com/ReactTraining/react-router) 是一个基于 React 之上的强大路由库，它可以让你向应用中快速地添加视图和数据流，既保证了单页应用的畅快，同时保持页面与 URL 间的同步。
6. \*[Babel](https://babeljs.io/) => 使用 JavaScript 实现的编译器，正如官网所说的那样 Use next generation JavaScript, today. ，可以利用 Babel 书写最新的 JavasScript 语法标准，如 ECMAScript 6 ，搭配 Webpack 使用更佳。
7. \*[ECMAScript6](http://es6-features.org/) => 2015 年提出的JavaScript标准，目标是使得JavaScript语言可以用来编写复杂的大型应用程序，成为企业级开发语言。ECMAScript和JavaScript的关系是，前者是后者的规格，后者是前者的一种实现。ES 6 具有一系列简明的语法糖，更佳的书写体验。但为了保证浏览器, Node 环境兼容性，往往配合 Babel 书写。
8. \*less => 一种 CSS 预处理语言，增加了诸如变量、混合（mixin）、函数等功能，让 CSS 更易维护、方便制作主题、扩充。
9. 使用 HTML5 的 getUserMedia 方法，调用计算机音频视频等硬件设备。为了安全问题，Chrome 只能在本地地址上调用该方法，外网地址则只能在通过证书检测的 HTTPS 服务中调用。

### 后端（Back-End）

1. 采用 [nodeJs](https://nodejs.org/) 作为后端，采用 JavaScript 脚本语言开发。 nodeJs 具有异步事件驱动、非阻塞（non-blocking）IO 特性，采用 Google 的 V8 引擎来执行代码。
2. Node.js以单线程运行，使用非阻塞I/O调用，这样既可以支持数以万计的并发连接，又不会因多线程本身的特点而带来麻烦。众多请求只使用单线程的设计意味着可以用于创建高并发应用程序。Node.js应用程序的设计目标是任何需要操作I/O的函数都使用回调函数。 这种设计的缺点是，如果不使用cluster、StrongLoop Process Manager或pm2等模块，Node.js就难以处理多核或多线程等情况。
3. pm2 => https://segmentfault.com/a/1190000004621734
4. isomorphic render（同构渲染）=> 指的是前后端使用同一份代码。前端通过 Webpack 实现 CommonJs 的模块规范（Node亦是 CommonJs ）+ React 提出的 JSX ，使得 NodeJs 通过解析请求的 URL，适配 react-router 中的前端路由规则，得到 routing Props，还可以 dispatch(action) 同步或异步（一般是 isomorphic-fetch ），又或是直接读取数据，从而更新 store ，最后 nodeJs 通过 store 中的 state 渲染 JSX ，产生静态的 HTML，从而实现了前后端的同构渲染。
5. [nodeJs C++ Addons](https://nodejs.org/api/addons.html)，nodeJs 就是使用C++语言实现的，图像处理最强大的库 opencv 便是用 C++ 实现的，这就不得不需要 nodeJs 与 C++ 之前通信，通过 nodeJs 调用 opencv 的方法，[node-opencv](https://github.com/peterbraden/node-opencv) 便是利用 [nan](https://github.com/nodejs/nan) （解决平台间兼容性问题，将异步事件驱动封装）与 [v8](https://github.com/v8/v8) （javascript 对应的数据类型与 C++映射） ，通过 [node-gyp](https://github.com/nodejs/node-gyp) 工具，将 C++ 打包成 一个动态链接库 \*.node，通过 require 即可调用。
6. [node-mysql](https://github.com/mysqljs/mysql) ，由于 NodeJs 具有 non-blocking IO 与异步事件驱动的特性，所以很适合于 IO 密集型高并发业务，而访问数据库正是常用的 IO 操作。
7. NPM（全称Node Package Manager，即node包管理器），是Node预设的，通过国内 taobao 镜像可以加快下载速度。
8. [Express](http://expressjs.com/zh-cn/)（Node.js Web 应用程序框架），很方便的定义 restful api.
9. Spider，网络爬虫，通过转发客户端的 HTTP 或 HTTPs 请求，得到远程服务器的响应数据，然后再一次转发至客户端中，也就是代理的意思  关于[南师大的一些 API](http://njnu.chaiziyi.com.cn/) ，已经有前人用 Python 写过了，爬取教务系统数据，然后我只需要爬取对应的网站即可。
10. nginx，使用 C++ 实现的 Web 服务器，通过简单的配置就可以反向代理至正确的端口和应用层协议。
11. 由于浏览器安全性的考虑，对于外网地址使用摄像头需要在安全的HTTPs协议下，因此需要付费或免费地得到认可的证书，通过 nginx 配置，反向代理至 Node 进程即可。

## 涉及知识

1. javascript / react / redux / react-router / webpack / less / node / babel / es6 / tracking.js / isomorphic / promise
2. http / https / express / mysql
3. opencv / Eigenfaces FaceRecognizer / node addons 降维、特征提取、特征比对
4. homebrew / curl / bash script / electron / cross-env / npm script / screen command / pm2 / nginx / git / seo

不仅仅局限与以上。以上工具、理论、技术可能只是项目简单地使用，或是学习过程中触碰过而已。

## 学习记录

* 2016年12月9日
  1. 初步确定课题：基于南师大本科生学生照片，进行人脸识别+特征提取+人脸相似度对比，判断输入人物图片是否存在于库中。若时间允许，精力有余，考虑加上声音比对，提高识别正确率。
  2. 成功下载南师大本科生照片，代码见gp-image-download文件夹，使用bash脚本+node实现，可以方便的跨平台。
* 2016年12月10日 初步了解人脸识别检测。 产生怀疑：
  1. 纯粹自己实现系列识别，特征提取，模式匹配等等。仅是识别算法理论，需要的数学功底较复杂...数学已经丢的差不多了..进度缓慢，无法入手，恐怕较难完成。
  2. 调用opencv接口或者调用网上api，恐怕工作量不够(应用交互设计完备点？)
  3. 换个课题？备选moka(1)或者iNjnu App(2)
* 2016年12月11日
  1. 搭建学生签到系统开发环境，采用react + redux + react-router + webpack技术栈，利用web前端技术实现界面，后续可以使用electron打包为跨平台app
  2. 注册face++账号，打算采用第三方人脸识别比对api
* 2016年12月12日
  1. 初步开发前端页面+后端服务(node express), 前后端分离
  2. 使用Sublime编辑器，默认缩进为Tab，书写脚本updateIndent.js，批量修改Tab为四空格
  3. 课题确定，《南师大学生刷脸签到系统》 对于人脸识别+比对方面实现，初步考虑3个解决方案：
  4. 人脸识别+比对算法完全自己实现。
  5. 调用opencv人脸识别api + 自己实现人脸比对算法。
  6. 调用网上较成熟的人脸识别+人脸比对接口，如Face++。 以上三种解决方案工作量递减（或者工作量可以在系统功能完备性方面体现），但是识别比对准确率递增。
* 不知道老师对以上三种方案有什么看法。 如果采用1或2，本人不知道应该看什么相关书籍入门(数学已经丢的差不多了，非考研党)，以及所用时间和最终效果都可能不尽人意。
* 还有一个问题：面向学生的教务系统好像没有输入课程号，教师ID，输出全部选课学生ID的接口。或者在面向教师的教务系统才有提供，但我没有教师账号密码，不能自己爬取。不知道我应该联系谁，才可以提供该接口。
  1. [获取学号接口](http://urp.njnu.edu.cn/authorizeUsers.portal?start=0&limit=100&term=191301)
  2. 根据爬取的学生照片，创建数据库stu(id, audio), sign(id, time, stu\_id); 班级的判断，通过已经得到ids.txt前6位得到（去掉非纯数字的，位数不同的）
  3. 构思管理员入口 => 登录(判断是否已经登录) => 查看学生信息(根据学号，姓名，班级号) + 签到信息查看
* 2016年12月13日
  1. 重拾数学 [积分](https://zh.wikipedia.org/wiki/%E7%A7%AF%E5%88%86) [方差](https://zh.wikipedia.org/wiki/%E6%96%B9%E5%B7%AE) [协方差](https://zh.wikipedia.org/wiki/%E5%8D%8F%E6%96%B9%E5%B7%AE) [统计独立] 矩阵 [协方差矩阵\*(写得好)](http://blog.csdn.net/itplus/article/details/11452743)
  2. 方差描述的是它的离散程度，也就是该变量离其期望值的距离。
  3. 协方差表示的是两个变量的总体的误差，这与只表示一个变量误差的方差不同。 如果两个变量的变化趋势一致，也就是说如果其中一个大于自身的期望值，另外一个也大于自身的期望值，那么两个变量之间的协方差就是正值。 如果两个变量的变化趋势相反，即其中一个大于自身的期望值，另外一个却小于自身的期望值，那么两个变量之间的协方差就是负值。
  4. [face\_camera](https://trackingjs.com/bower/tracking.js/examples/face_camera.html) （Js人脸检测插件，去掉手动拍照，监控摄像头识别人脸，可以根据searching和是否有人脸进行对比操作，只发送多个人脸部分图像）**看下源码，学习识别算法**
* 2016年12月14日
  1. opencv install:
  + brew tap homebrew/science  
    brew install opencv
  + [install-opencv-3-on-yosemite-osx-10-10-x](https://www.learnopencv.com/install-opencv-3-on-yosemite-osx-10-10-x/) [linux\_install](http://docs.opencv.org/2.4/doc/tutorials/introduction/linux_install/linux_install.html) [install-opencv-3-0-and-python-2-7-on-osx](http://www.pyimagesearch.com/2015/06/15/install-opencv-3-0-and-python-2-7-on-osx/)
  1. [添加opencv依赖](http://docs.opencv.org/2.4/doc/tutorials/introduction/linux_gcc_cmake/linux_gcc_cmake.html)
  2. brew更改源 替换formula 索引的镜像（即 brew update 时所更新内容）
  + cd "$(brew --repo)"  
    git remote set-url origin https://mirrors.tuna.tsinghua.edu.cn/git/homebrew/brew.git  
      
    cd "$(brew --repo)/Library/Taps/homebrew/homebrew-core"  
    git remote set-url origin https://mirrors.tuna.tsinghua.edu.cn/git/homebrew/homebrew-core.git  
      
    brew update
  + 替换Homebrew 二进制预编译包的镜像
  + echo 'export HOMEBREW\_BOTTLE\_DOMAIN=https://mirrors.tuna.tsinghua.edu.cn/homebrew-bottles' >> ~/.bash\_profile  
    source ~/.bash\_profile
  1. [node-opencv](https://github.com/peterbraden/node-opencv)  
     [Issue](https://github.com/peterbraden/node-opencv/issues/380#issuecomment-191492421)
* 2016年12月16日
  1. 使用node-opencv检测人脸，挑选出效果相对好的分类模板lbpcascade\_frontalface.xml, 对比效果数据见backend/data/summary.json
  2. 死嚼PCA理论[1](http://blog.csdn.net/itplus/article/details/11451327), [2](http://blog.csdn.net/liulina603/article/details/7912950)
* 2016年12月18日
  1. [人脸检测模式匹配数据xml2json](http://www.freeformatter.com/xml-to-json-converter.html)(需VPN)
  2. 发现人脸识别效果不佳，分析原因， 第一，样本每个人只有一张 ，第二，几年下来，人的变化比较大。
  3. 新增样本输入模块，学生自主输入删除样本。（需要重新训练，存储）
  4. 采用opencv Eigenfaces 人脸识别算法
* 2017年1月11日 image-download去除对wget依赖，改用curl指令下载
* 2017年1月16日
  1. /usr/local/bin/mysql.server start 启动mysql
  2. 开始mysql数据库设计: face\_import table, 书写DAO代码
  3. 简化部分业务逻辑，删除非必须输入情况(本地图片，网络图片)
  4. 考虑到数据的迁移简便和服务器负载，使用sm.ms免费图床，存储用户导入的人脸图片
  5. 完成100%人脸录入逻辑。TODO: 每次启动服务器需要读取数据库，得到smms图片数据，进行训练。（为了保证Dev环境启动速度，暂时不做）
  6. 引入cross-env：跨平台设置环境变量NPM包, 区分Dev(父进程监听js文件改动，改动后则重启服务器)与Production环境
* 2017年1月17日
  1. 改善训练样本方法，加上了smms外链的图片训练(一大串Promise)
  2. 完成前后端分离的管理员登录状态控制(本地存储+md5编码)，完成管理员样本查看功能。
* 2017年1月18日
  1. 完成每次启动服务器需要读取数据库，得到smms图片数据，进行训练。
  2. 完成删除/添加人脸样本，重新训练逻辑(异步，不保证实时性)。
  3. 对于学生证照，lbpcascade\_frontalface, scale=1.95左右人脸检测效果较好
  4. [distance convert to precentage](http://stackoverflow.com/questions/13652778/what-is-confidence-in-opencvs-facerecognizer)
  5. confidence<?, 认为是正确
  6. 整体大致已经完成，阈值的确定尚未完成
  7. [iterm2, 快捷键设置(useful!)](https://coderwall.com/p/h6yfda/use-and-to-jump-forwards-backwards-words-in-iterm-2-on-os-x)
* 2017年1月19日
  1. Server读取app代码时，node\_module回环加载，出现错误，修改成只使用最外层node\_modules
  2. react Server端渲染，require() 样式文件的解决方法？(ignore, only无效)
  3. HTTPs才能使用摄像头，因此搞了个免费证书，配置下nginx
     + [https://aotu.io/notes/2016/08/16/nginx-https/](值得细嚼)
     + [https://www.zhihu.com/question/19578422](免费证书)
     + http://serverfault.com/questions/67316/in-nginx-how-can-i-rewrite-all-http-requests-to-https-while-maintaining-sub-dom
  4. mac迁移到ubuntu一系列的问题
     1. http://stackoverflow.com/questions/12335848/opencv-program-compile-error-libopencv-core-so-2-4-cannot-open-shared-object-f
     2. unix平台的快速安装脚本=> start.sh
  5. [nativefier](https://github.com/jiahaog/nativefier) => 站点打包成App解决方案, ELECTRON\_MIRROR=https://npm.taobao.org/mirrors/electron/
  6. app:packager scripts
  + "app:mac64": "set ELECTRON\_MIRROR=https://npm.taobao.org/mirrors/electron/ && nativefier -a x64 -p mac --name \"南师大刷脸签到\" \"https://face.moyuyc.xyz/\" -i logos/logo.icns --disable-dev-tools --disable-context-menu",  
    "app:mac32": "set ELECTRON\_MIRROR=https://npm.taobao.org/mirrors/electron/ && nativefier -a ia32 -p mac --name \"南师大刷脸签到\" \"https://face.moyuyc.xyz/\" -i logos/logo.icns --disable-dev-tools --disable-context-menu",  
    "app:win32": "set ELECTRON\_MIRROR=https://npm.taobao.org/mirrors/electron/ && nativefier -p win32 -a x64 --name \"南师大刷脸签到\" \"https://face.moyuyc.xyz/\" -i logos/logo.png --disable-dev-tools --disable-context-menu",  
    "app:win64": "set ELECTRON\_MIRROR=https://npm.taobao.org/mirrors/electron/ && nativefier -p win32 -a ia32 --name \"南师大刷脸签到\" \"https://face.moyuyc.xyz/\" -i logos/logo.png --disable-dev-tools --disable-context-menu"
* 2017年1月20日
  1. 同构渲染(优化seo+首屏渲染) [css module打包的解决方案](http://www.aliued.com/?p=3077), [webpack-isomorphic-tools](https://github.com/halt-hammerzeit/webpack-isomorphic-tools)
  2. [远端发布 screen指令](https://github.com/chenzhiwei/linux/tree/master/screen)
* 2017年1月21日
  1. google+baidu 收录，添加 robots.txt 与 sitemap.txt，如今在google下搜索 南京师范大学 刷脸 即可

## 系统剖析

源码地址：[Graduation-Project](https://github.com/moyuyc/graduation-project)

### 文件结构

顶层文件结构如下

Graduation-Project/  
├── desktop/  
├── gp-image-download/  
├── gp-njnu-photos-app/  
└── gp-njnu-photos-backend/

desktop/ 中放的是将站点打包成 PC Desktop 的 Logo，logo.icns 用在 osx 系统中，logo.png 则用于 linux 与 windows 系统中，打包成的 PC Desktop 默认也是放在该文件夹下。

desktop/  
└── logos/  
 ├── logo.icns  
 ├── logo.ico  
 └── logo.png

gp-image-download/ 文件夹里面放的是将教务系统的学生图片下载至该文件夹中，其中的 data/ 文件夹放的是各年入学的学生的学号（部分学号不正确），数据是 get-all-id.js 脚本得到，具体工作细节在后面会说到。保证获取到各年的学生学号集后，通过 download.sh Bash 脚本即可进行下载；下载的图片放在 images/ 中。

gp-image-download/  
├── data/  
│ └── ...  
├── download.sh  
├── get-all-id.js  
├── images/  
│ └── ...  
└── lib/  
 └── get-all-id.js

gp-njnu-photos-app/ 放的是前端代码，app/ 是开发用的源码，build/ 是用 Webpack 打包的压缩后的发布资源，包括 css/js/html/image...

gp-njnu-photos-app/  
├── app/  
│ └── ...  
├── build/  
│ └── ...  
├── index.js  
├── package.json  
├── webpack-assets.json  
└── webpack.config.js

gp-njnu-photos-backend/ 放的是后端的全部代码。  
- test/ nodejs 调用 opencv 接口的例子  
- data/ 放些人脸 xml 模板数据，服务器运行时生成的缓存数据，预处理后的学生照片，上一次保存的训练数据。  
- database/ 访问 mysql 的 JS 接口 - lib/ 一些通用的 JS 方法，比如爬虫接口，图片上传接口...  
- opencv/ 搭建 nodejs 与 C++ 桥梁的源码与产生的链接库 - pretreat 一些预处理接口，如人脸检测，图片灰化处理，样本数据的训练... - routes express 的路由控制文件 - ssl HTTPs 证书与密钥 - server.js HTTP 服务入口 - index.js 在调用 server.js 之前，获取前端数据，使得后端能够处理静态资源（image/css/...），为前后端公用一套代码提供解决方案。（一般在线上环境使用） - provider.js 创建子进程 server.js，同时监听后端开发目录代码的改动，一旦改动便杀死上一个子进程，并且再次创建子进程 server.js，可以实现后端代码的***热更新***。（一般在开发环境使用）

gp-njnu-photos-backend/  
├── test/  
│ └── ...  
├── data/  
│ └── ...  
├── database/  
│ └── ...  
├── lib/  
│ └── ...  
├── opencv/  
│ └── ...  
├── pretreat/  
│ └── ...  
├── routes/  
│ └── ...  
├── ssl/  
│ └── ...  
├── index.js  
├── package.json  
├── provider.js  
└── server.js

### 过程

#### 学生照片下载

#### 人脸识别理论学习

#### opencv 环境安装

#### node addons

#### 图片预处理（人脸检测...）

#### 识别算法测试与确定

#### 学生信息接口（爬虫）

#### 前端+后端

#### mysql

#### 同构渲染

#### 部署

#### nginx + https

#### SEO

#### PC Desktop

### 代码解析

#### 获取ID集合

#### 下载图片脚本

#### 图片预处理脚本

#### 样本训练脚本

#### 开发环境（热部署）脚本

#### MySQL 服务器、非服务器环境区别处理

#### 同构渲染核心

#### 一键部署脚本

#### Desktop打包脚本

### 系统模块

图片

### 系统截图

...

### 总结