项目：人脸识别登录的图书系统

人脸识别：算法的研究而已，没有目的性

图书爬取：在于量，不在于技术

系统：图书管理

前——>后----->爬虫----->数据分析—>数据挖掘

sklearn包(导包侠)

(老程序员不爱学的东西[前端，创新，偏向于机器学习的算法) 巧

带（学徒）给你任务让你做

前端：框架/原生 vue

功能：捕捉视频中的图像内容，拍照后在网页上显示。

网页显示的是拍照后的照片。

只注重技术点的讲解，难点在哪里，怎么解决的。（本着自己给自己找麻烦）不注重功能实现的详细程度

简单界面：

拍照后的结果

摄像头

拍照

开启摄像头

摄像头：video视频标签

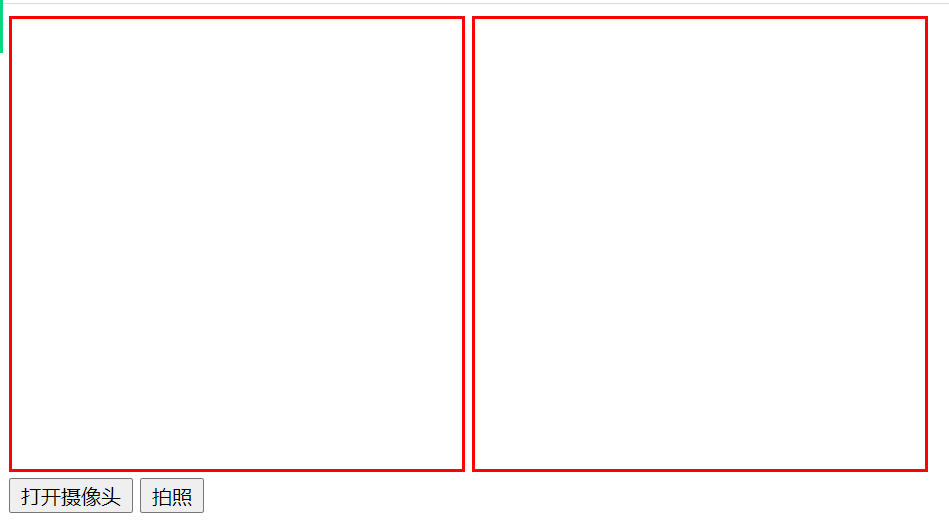
拍照后的结果：canvas(HTML5画布）

按钮:button

界面代码:

<!DOCTYPE **html**>  
<**html lang="en"**>  
<**head**>  
 <**meta charset="UTF-8"**>  
 <**title**>拍照界面</**title**>  
 <**style**>  
 **video**,**canvas**{  
 **border**:2**px solid red**;  
 }  
 </**style**>  
</**head**>  
<**body**>  
 <**video width="300" height="300"**></**video**>  
 <**canvas width="300" height="300"**></**canvas**><**br**/>  
 <**button** >打开摄像头</**button**>  
 <**button**>拍照</**button**>  
</**body**>  
</**html**>

浏览器中出现的结果如下。



js代码:

逻辑:

打开摄像头的逻辑:

1. 首先获取摄像头，先获取网页上面摄像头部分的元素，利用取id的方法

getElementById

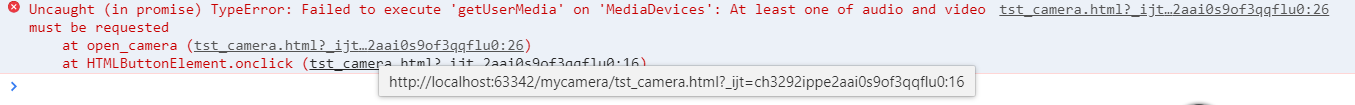
1. 首先取硬件的实际摄像头

前端navigator.mediaDevices.getUserMedia

1. 把页面上的摄像头框和硬件建立关联。把页面上的框当成参数传入实际摄像头里面去，
2. 利用then方法表示连接摄像头以后的动作，把摄像头的内容给显示框。

video标签对应的播放方法：play()

错误提示:



错误的原因是参数中没有指定video

代码如下:

<!DOCTYPE **html**>  
<**html lang="en"**>  
<**head**>  
 <**meta charset="UTF-8"**>  
 <**title**>拍照界面</**title**>  
 <**style**>  
 **video**,**canvas**{  
 **border**:2**px solid red**;  
 }  
 </**style**>  
</**head**>  
<**body**>  
  
 <**video width="300" height="300" id="myvideo"**></**video**>  
 <**canvas width="300" height="300"**></**canvas**><**br**/>  
 <**button onclick="***open\_camera*()**"**>打开摄像头</**button**>  
 <**button**>拍照</**button**>  
 <**script**>  
 **function** *open\_camera*(){  
 **myvideo**=***document***.getElementById(**"myvideo"**)  
 */\*指定外部摄像头设备的宽和高\*/  
 /\*外部设备分为视频video或者是音频audio\*/* **cons**={  
 **video**:{**width**:300,  
 **height**:300}  
 }  
 **cameras**=***navigator***.**mediaDevices**.getUserMedia(**cons**)  
 */\*一般情况下,then方法里面都是函数,参数就是摄像头中的内容\*/* **cameras**.then(**function**(stream){  
 **myvideo**.**srcObject**=stream  
 **myvideo**.play()  
 })  
 }  
 </**script**>  
</**body**>  
</**html**>

下面进行拍照逻辑:

1. 点击按钮时候发生，在按扭上面加方法，catch\_camera
2. 照像就是画图像，画图像在canvas画布上来画，获取canvas画布，利用id
3. 利用canvas画布的方法drawImage画图像，注意canvas画图之前需要说明画的2d图还是3d图，如果画的是2d，获取2d的图像画布.

ctx=canvas.getContext(“2d”)

再调用drawImage，参数

Ctx.drawImage(被画到画布上的图像，从画布的哪个一个位置x，从画布的哪个一个位置y，画到哪一个位置的x，画到哪一个位置的x)

在当前的例子中,左上角是坐标（0，0），画到右下角，右下角的坐标为（300，300）

ctx.drawImage(myvideo，0，0，300，300)

1. 画布就可以显示图像了。

代码:

<!DOCTYPE **html**>  
<**html lang="en"**>  
<**head**>  
 <**meta charset="UTF-8"**>  
 <**title**>拍照界面</**title**>  
 <**style**>  
 **video**,**canvas**{  
 **border**:2**px solid red**;  
 }  
 </**style**>  
</**head**>  
<**body**>  
  
 <**video width="300" height="300" id="myvideo"**></**video**>  
 <**canvas width="300" height="300" id="canvas"**></**canvas**><**br**/>  
 <**button onclick="***open\_camera*()**"**>打开摄像头</**button**>  
 <**button onclick="***catch\_camera*()**"**>拍照</**button**>  
 <**script**>  
 **myvideo**=***document***.getElementById(**"myvideo"**)  
 **function** *open\_camera*(){  
  
 */\*指定外部摄像头设备的宽和高\*/  
 /\*外部设备分为视频video或者是音频audio\*/* **cons**={  
 **video**:{**width**:300,  
 **height**:300}  
 }  
 **cameras**=***navigator***.**mediaDevices**.getUserMedia(**cons**)  
 */\*一般情况下,then方法里面都是函数,参数就是摄像头中的内容\*/* **cameras**.then(**function**(stream){  
 **myvideo**.**srcObject**=stream  
 **myvideo**.play()  
 })  
 }  
 **function** *catch\_camera*(){  
 **canvas**=***document***.getElementById(**"canvas"**)  
 **ctx**=**canvas**.getContext(**"2d"**)  
 **ctx**.drawImage(**myvideo**,0,0,300,300)  
 }  
 </**script**>  
</**body**>  
</**html**>

把代码做一下修改,项目中的问题就出来了.

用一个样式表集中定义了video,canvas

<!DOCTYPE **html**>  
<**html lang="en"**>  
<**head**>  
 <**meta charset="UTF-8"**>  
 <**title**>拍照界面</**title**>  
 <**style**>  
 **video**,**canvas**{  
 **border**:2**px solid red**;  
 **width**:300**px**;  
 **height**:300**px**;  
 }  
 </**style**>  
</**head**>  
<**body**>  
  
 *<!--<video width="300" height="300" id="myvideo"></video>  
 <canvas width="300" height="300" id="canvas"></canvas><br/>-->* <**video id="myvideo"**></**video**>  
 <**canvas id="canvas"**></**canvas**>  
 <**button onclick="***open\_camera*()**"**>打开摄像头</**button**>  
 <**button onclick="***catch\_camera*()**"**>拍照</**button**>  
 <**script**>  
 **myvideo**=***document***.getElementById(**"myvideo"**)  
 **function** *open\_camera*(){  
  
 */\*指定外部摄像头设备的宽和高\*/  
 /\*外部设备分为视频video或者是音频audio\*/* **cons**={  
 **video**:{**width**:300,  
 **height**:300}  
 }  
 **cameras**=***navigator***.**mediaDevices**.getUserMedia(**cons**)  
 */\*一般情况下,then方法里面都是函数,参数就是摄像头中的内容\*/* **cameras**.then(**function**(stream){  
 **myvideo**.**srcObject**=stream  
 **myvideo**.play()  
 })  
 }  
 **function** *catch\_camera*(){  
 **canvas**=***document***.getElementById(**"canvas"**)  
 **ctx**=**canvas**.getContext(**"2d"**)  
 **ctx**.drawImage(**myvideo**,0,0,300,300)  
 }  
 </**script**>  
</**body**>  
</**html**>

开发过程中的问题1：

研究过程，在于canvas画布的特点。

1. canvas有原大小，原大小为300,150
2. canvas在css里面进行拉伸的话，相当于canvas画布在300\*150设置也被拉伸，意思是画布被拉伸。
3. 把canvas在标签里面进行设置宽和高。

结论：canvas只能把宽和高设置在标签上，不能使用css样式来设置，如果使用了，就是对原有画布大小进行拉伸。

后端功能

功能:接收前端的图像数据，需要把图像数据进行存储，设定数据库mongodb，当用户再次调用摄像头，拍照之后，比对这两张脸是不是同一张脸，这比对就是两个数像数据的比对，用到模块face-recognition进行脸的对比。当前拍照的脸与数据里的脸数据一致或者近似，可以登录。

分库分表：人脸比对用的mongodb

图书系统还是使用mysql

第一实现步骤：把前端的数据上传，

前端可以通过**canvas**.toDataURL把一个图片变成数据。

通过canvas.toDataURL获取的数据如下：

data:image/png;base64,

注意数据的前面一部分：

data:image/png;base64,

表征了图像数据的格式png格式，同时编码base64，前面编码说明“，”后的就是具体的数据了。上传到后台一定是”data:image/png;base64,”这一串后面的数据,因为数据的前面不是数据的具体内容，只是格式说明，要把前面的这几位去掉，去掉的方式可以切片的方法。

后台的框架：flask

flask实现一个hello world

请求的地址:/upload

get方法地址栏显示，点击的按钮提交，不需要地址栏做一些呼应，这里采用post方法。jquery的post方法就是$.post

$.post代码如下：

*/\*需要使用jquery ajax进行上传\*/*$.post(**"/upload"**,{**"myimg"**:**datas**},**function**(res){  
 ***console***.log(res)  
})

后台分出get和post方法，不同的方法对应不同的处理方式，get方法对应对网页模板进行渲染。post方法实现图像数据的获取。

代码如下：

**if** request.method == **"POST"**:  
 *# 需要前端上传一个数值对应的变量* print(**"1111111"**)  
 imgdata = request.form.get(**"myimg"**)  
 print(imgdata)  
 *# 获取到的是base64的数据，需要base64模块进行decode解码，解码后存在图片* img\_base64\_data = base64.b64decode(imgdata)  
 **with** open(**"a.png"**, **"wb"**) **as** f:  
 f.write(img\_base64\_data)  
**return** render\_template(**"tst\_camera.html"**)

图像数据已上传到服务器，做的实质是人脸，看上传中的图像中是否包含人脸，需要调用face-recognition模块。

调用face\_encodings查找人脸的部位。注意前端拍照的时候一定全脸的数据，不然后端脸部是没有数据的。

代码：

**if** request.method == **"POST"**:  
 *# 需要前端上传一个数值对应的变量* print(**"1111111"**)  
 imgdata = request.form.get(**"myimg"**)  
 print(imgdata)  
 *# 获取到的是base64的数据，需要base64模块进行decode解码，解码后存在图片* img\_base64\_data = base64.b64decode(imgdata)  
 **with** open(**"a.png"**, **"wb"**) **as** f:  
 f.write(img\_base64\_data)  
 *#1、使用face\_recognition读取图片 load\_image\_file* imgfile=face\_recognition.load\_image\_file(**"a.png"**)  
 *#2、当imgfile被读取的时变成array  
 #face\_encodings就是在图片中找出人脸的相关信息* facedata=face\_recognition.face\_encodings(imgfile)  
  
 *#注意后面的操作一定有一个前提，前提条件就是facedata不能为空* **if** facedata!=[]:  
 facedata=facedata[0]  
 print(facedata)  
 *#存储到mongodb数据库中* **else**:  
 **return "脸部数据报错"  
  
  
return** render\_template(**"tst\_camera.html"**)

接下来，把数据存储到数据库中，安装flask-pymongo

Pip3 install flask-pymongo

开发中的思路变通:

上传的数据是base64编码，face-recognition里面原理还是机器学习，机器学习基础是ndarray。把 base64变成ndarrayLoad\_image\_file读出文件同时数据ndarray。还要识别图片当中是否含有脸部区域，最终存在mongodb数据库中一定是脸部区域，face\_encodings把脸部区域变成ndarray。

开发中的第二个问题:

mongodb是否能存储ndarray数据，推导到mysql，如果有一个ndarray数据，能不能存储以mysql。

Pickle dumps把特殊类型转成python类型,loads把数据取出来

Json dumps把json数据转成python类型 loads把python类型转成json类型，

代码句子：

face\_data=pickle.dumps(facedata,protocol=-1)

人脸识别登录功能：

上传第二张脸，不符合脸部数据库中存储的记录，那就是非法用户，如果在脸部数据库中可以找到这个数据,compare\_faces进行脸部比较，比较可以允许误差。

实现人脸登录，只能改变一下页面，有一张脸是上传的，上传的数据是数据库中没有的。

改一下前端逻辑：

文件框，读取文件，把文件的数据上传。

前端代码:

**function** *upload\_pic*(){  
 */\*取文件选择框按钮\*/* **filename**=***document***.getElementById(**"file"**)  
 **filename**=**filename**.**files**[0]  
 */\*前端有一个文件读取类\*/* **reader**=**new *FileReader***()  
 **reader**.readAsDataURL(**filename**)  
 */\*onloadend表示文件读取结束时\*/* **reader**.**onloadend**=**function**(){  
 **datas**=**reader**.**result**.substring(22)  
 ***console***.log(**datas**)  
 $.post(**"/check"**,{**"myimg"**:**datas**},**function**(res){  
 ***console***.log(res)  
 })  
 }

后端始终返回前端的是json数据，专门名词叫接口

@app.route(**"/check"**,methods=[**"POST"**,**"GET"**])  
**def** check():  
 **if** request.method==**"POST"**:  
 *#收集前端base64的数据* imgdata=request.form.get(**"myimg"**)  
 img\_base64\_data=base64.b64decode(imgdata)  
 **with** open(**'b.png'**,**'wb'**) **as** f:  
 f.write(img\_base64\_data)  
 *#调用face\_recognition的load\_image\_file读取时ndarray* imgfile=face\_recognition.load\_image\_file(**"b.png"**)  
 *#找脸部区域* facedata=face\_recognition.face\_encodings(imgfile)  
 *#读取的脸部数据不能往数据存，需要跟数据库中存储的数据做比较  
 #先读取数据中所有的脸* faces=mongo.db.oldfaces.find()  
 **for** face **in** faces:  
 newface=pickle.loads(face[**"face"**])  
 compare\_result=face\_recognition.compare\_faces(newface,facedata)  
 *#compare\_result是一个布尔型的值* **if** compare\_result:  
 **return** {**"result"**:**"脸部识别成功,允许登录"**}  
 **else**:  
 **return** {**"result"**:**"不是允许登录的那张脸"**}  
  
 **return** render\_template(**"tst\_compare.html"**)

\*\*\*\*\*\*\*爬虫爬取的数据，写接口

需求：前端拍照动作发生后，后端会把脸部数据识别出来出来，形成脸部数据的图，传给前端，前端可以通过后台传来的脸部数据把切出来的脸显示出来，不显示拍照后整个视频的全貌。

（奥卡姆剃刀：如无必要，勿增实体）

开发人脸登录识别项目:

本来上传图片base64,转码b64decode正常,切分后,再进行编码,b64encode报错.如何处理.

类型是一个Array，使用pillow，image，通过矩阵产生图片，采用base64编码，最后转化utf8编码即可。

验证face-recognition切人脸准确性。

Face-recognition中load\_image\_file 加载图片

Face\_ecodings 把脸部区域变成ndarray

Compare\_faces 比较两张脸相同度

Face\_locations 输出脸的位置。

Face\_landmarks 五官的位置

利用pillow模块有ImageDraw画妆方法。

人脸识别登录：

问题一:

前端canvas如果css设置样式,基本把画布拉伸.

问题二:

如果存mongodb里的ndarray数据如何保存,通过pickle

问题三:

对图片中脸部进行切分, 返回前端base64,ndarray转成base64. Pillow比较保险的一种方法

问题四:

安装face-recognition, 安装visual studio dlib cmake, 一直要安装C++支持环境

图书系统

京东网站:

A href //list.jd.com/1713-3258-3297.html

302跳转

<https://list.jd.com/list.html?cat=1713,3258,3297>

把网站全部爬取

京东地址特点:

地址类型

<https://list.jd.com/list.html?cat=737,794,798>

手机地址:https://mall.jd.com/index-1000004259.html#!

电脑的地址:

<https://search.jd.com/Search?keyword=%E7%94%B5%E7%AB%9E%E6%98%BE%E7%A4%BA%E5%99%A8&enc=utf-8&wq=%E7%94%B5%E7%AB%9E%E6%98%BE%E7%A4%BA%E5%99%A8&pvid=0973b801dd644f948615e668640f0f65>

家居:

<https://search.jd.com/Search?keyword=%E5%9B%9B%E4%BB%B6%E5%A5%97&enc=utf-8&pvid=17242cf6723b42cdb77d6002c6214b72>

结论:

爬取京东列表的数据，只能相信下面地址结构

<https://list.jd.com/list.html?cat=737,794,798>

问题：如何获取京东中所有的分类列表全部都是上面类型地址。

问题二：是否触犯法律法规

<https://www.jd.com/robots.txt>

User-agent: \* /\*

Disallow: /?\* /?\*

Disallow: /pop/\*.html

Disallow: /pinpai/\*.html?\*

User-agent: EtaoSpider

Disallow: /

User-agent: HuihuiSpider

Disallow: /

User-agent: GwdangSpider

Disallow: /

User-agent: WochachaSpider

Disallow: /