

基于Python的智能监控平台文档

雷怡然 2016013274

徐天依 2016

1.环境配置

网站使用了 `Flask` 框架，安装 `Flask`：

```
1 | pip install Flask
```

`Flask` 连接 `mysql` 数据库需要库 `flask_sqlalchemy` 和库 `pymysql`，安装：

```
1 | pip install flask_sqlalchemy  
2 | pip install pymysql
```

试图建立 `WebSocket` 连接，需要库 `flask_socketio`，安装：

```
1 | pip install flask_socketio
```

图像处理需要库 `opencv` 和 `pillow`，安装：

```
1 | pip install opencv-python  
2 | pip install pillow
```

图像分析使用了 `tensorflow` 的 `object_detection`：

官网链接：

接：https://github.com/tensorflow/models/tree/master/research/object_detection

安装过程：

```
1 | pip install tensorflow  
2 |  
3 | pip install --user Cython  
4 | pip install --user contextlib2  
5 | pip install --user pillow  
6 | pip install --user lxml  
7 | pip install --user jupyter  
8 | pip install --user matplotlib
```

然后下载 `tensorflow` 模型:

模型官网: <https://github.com/tensorflow/models>

执行命令:

```
1 | git clone https://github.com/tensorflow/models
```

或者

```
1 | wget https://github.com/tensorflow/models
```

即可完成下载,得到一个 `models` 文件夹, 将刚才下载的 `models` 文件夹移动到刚刚安装的 `tensorflow` 的文件夹中。

可以通过如下方法获知 `tensorflow` 的安装文件夹, 在通过 `pip` 安装完成后, 再次运行 `pip install tensorflow`, 命令行会显示已经安装, 并显示 `tensorflow` 所在的路径。之后安装 `tensorflow` 需要的 `COCO API`:

```
1 | git clone https://github.com/cocodataset/cocoapi.git
2 | cd cocoapi/PythonAPI
3 | make
4 | cp -r pycocotools <tensorflow文件夹的路径>/models/research/
```

在 `tensorflow/models/research` 文件夹下进行 `Protobuf` 编译:

```
1 | # From tensorflow/models/research/
2 | protoc object_detection/protos/*.proto --python_out=.
```

设置环境变量,在 `tensorflow/models/research` 文件夹下运行:

```
1 | # From tensorflow/models/research/
2 | export PYTHONPATH=$PYTHONPATH:`pwd`:`pwd`/slim
```

`tensoflow object detection` 安装完成! 可以通过以下命令测试是否安装成功:

```
1 | python object_detection/builders/model_builder_test.py
```

然后, 打开项目文件夹中的 `camera.py` 和 `od.py`, 将文件中的 `TENSORFLOW_DIR` 修改为当前的 `tensorflow`所在路径。至此, 所有环境配置完成, 还需要初始化数据库。

打开 `mysql`, 建立一个名为 `2018Python` 的数据库:

```
1 | create database 2018Python;
2 | use 2018Python;
```

在 `2018Python` 数据库中添加名为 `user` 和 `rootuser` 两个表，分别代表普通用户数据库和管理员数据库：

```
1 | create table user (
2 |   username char(20) primary_key,
3 |   password char(20) not null
4 | );
```

```
1 | create table rootuser (
2 |   username char(20) primary_key,
3 |   password char(20) not null
4 | );
```

之后向管理员数据库 `rootuser` 添加一个管理员：

```
1 | insert into rootuser values('root','1234');
```

以后也通过类似的方式添加管理员，然后在网页上通过管理员添加普通用户。
至此，所有初期准备工作完成。

(PS:以上所有配置在Mac OS下可以使程序成功运行)

2.文件结构介绍

`templates` 文件夹中是网页模版。

`static` 文件夹中是一些静态文件js和css文件。

`res` 文件夹中是 `object detection` 需要的训练好的模型和标签和其他的资源文件。

`output` 文件夹中放置的是 `object detection` 分析的结果。`camera.py` 是摄像机行为和图像处理的文件。

`od.py` 是物体检测的具体实现。

`tmo.py` 是追踪动态物体的具体实现。

`app.py` 是程序入口,运行

```
1 | python app.py
```

然后进入 `127.0.0.1:5000` 即可运行程序。

3.功能介绍

1.登陆使用了 `mysql + cookies` 实现，分为普通用户和管理员。

管理员登陆

用户名
密码

登陆以后进入功能管理页面，在该页面普通用户和管理员可以选择相应功能，并可以选择注销。

功能列表

- [原视频流](#)
- [实时捕捉动态物体](#)
- [修改密码](#)
- [添加用户](#)
- [查看,删除用户](#)

特别的，管理员可以查看普通用户并删除。

2.修改密码，修改成功或者失败都会有网页提示。

修改密码

用户名: root
原密码:
新密码:
再输入一次新密码:

[返回首页](#)

3.管理员添加用户。

新用户名
密码

[返回首页](#)

管理用户。

用户列表

用户名 密码 是否删除

Jack 123 [删除](#)

leiyiran 123 [删除](#)

[返回首页](#)

4. 读取摄像头，应该支持RTSP协议的网络摄像头，在 `camera.py` 中的函数 `init_camera` 中可以修改摄像头参数（我在网上找了一个 rtsp 流的视频是可以播放的，实验室的摄像头连接好像有点问题，在 `camera.py` 文件中可以修改，目前是默认系统摄像头）。

前端显示视频通过 `M-JPEG` 格式的流媒体实现。

5. 视频分析有两部分，第一部分是 `tensorflow` 的物体检测，第二部分是 `动态物体追踪`。

在功能选择网页上可以打开 `原视频流` 和 `实时捕捉动态物体`，无论打开哪一个网页，都会创建一个新的进程在后台进行物体检测，物体检测每一帧处理需要一定的时间，当处理完一帧后再传入下一帧处理，处理完后的结果存储在 `output` 文件夹中，图片以时间命名。

如果打开 `实时捕捉动态物体` 网页，则会进行动态物体追踪，该功能逐帧分析并且及时显示在网页上，刚打开的时候需要先拍摄背景帧，因此设置延迟3秒的延迟时间（背景帧意思是拍摄一帧图片，该图片上尽可能不包含移动的物体，方便后计算），在运行的过程中，可以随时调整该功能的相关参数值并重新设置背景帧。

（在此说明，`高斯模糊` 值越大，细节就会被去掉；`亮度阈值` 越大，和背景颜色比较相似的物体就越不容易被检测出来）

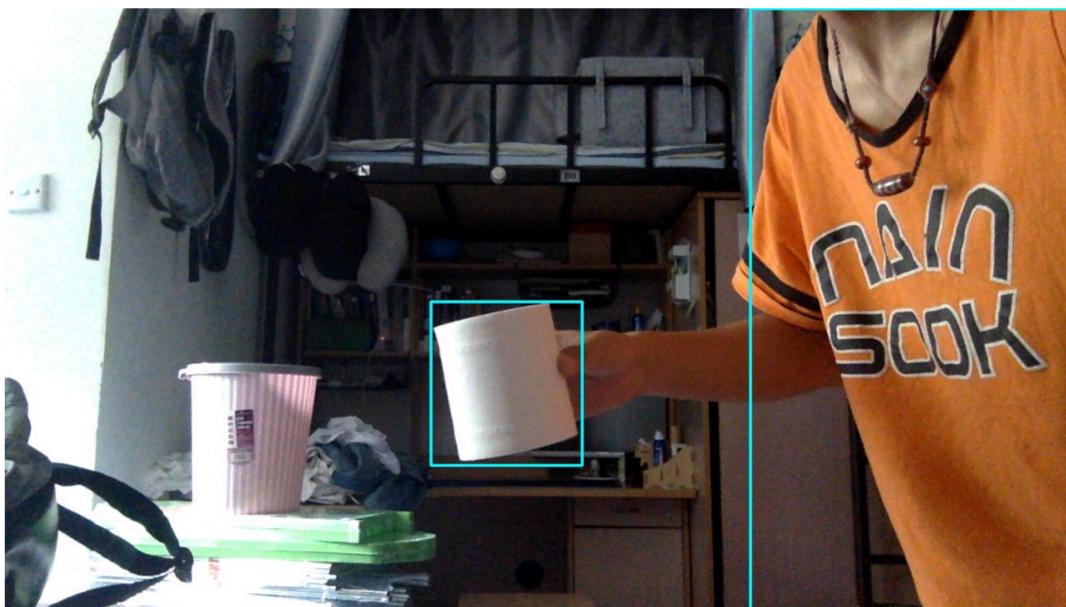
`object detection` 检测效果示例：(忽略我的脸)



动态追踪物体 效果示例：



动态物体捕捉



比较占CPU的 `object detection` 在另一个进程运行，可以充分利用多核CPU的性能，并且保证了在分析时网页播放视频时不会卡顿。

6. 报警功能，本来想通过 `object detection` 实现当画面中出现特定物品就报警的功能，然后将报警信息存入数据库。数据库和物品检测的接口已经写好了，在 `app.py` 和 `od.py` 中的可以查看到。但是 `Flask` 的 `websocket` 的用法和 `Flask` 网站响应很玄学，导致python主动推送的功能实现不了，就暂时鸽了这个功能。