

# MediaPi

## 基于MediaPipe的手势识别网站——2013114张广慧

### 一、关于项目

- 基于MediaPipe和OpenCV开发
- 包括必要的前端界面与后端管理系统
- 搭建了数据库，实现用户登录
- 实现基本手势识别功能

### 二、技术栈

- 前端：HTML、CSS、JavaScript
- 后端：Django、Python
- 数据库：SQLite
- 后端包管理：pipenv、pip
- 文档撰写：markdown、飞书
- 开发工具：VSCode



### 三、解决问题

- 实现最基础的手势识别
- 能够在此基础上实现更完善的手势识别和需要使用到手势识别的应用

## 四、项目部署

项目主要通过Django开发，前端与后端均部署在本地，运行 `.bat` 文件即可安装所需依赖并启动项目。

另外，撰写了 `README.md` 文档记录部署流程和启动方式。

 <code>install.bat</code>	2023/1/9 19:21	Windows 批处理...	1 KB
 <code>README.md</code>	2023/1/9 19:18	Markdown 源文件	2 KB

尝试使用 `pyinstaller` 打包 Django 项目，但是部署时失败了。

## 五、实现过程

### 5.1 前端

前端使用 HTML 开发、存放于 `templates` 文件夹中。对于 Django 而言，加载 HTML，Django 会通过 `templates` 中放置的 HTML 来加载。

```
▼ templates
  <> demo.html
  <> main.html
  <> signin.html
  # style.css
  <> wrong.html
```

前端一共编写了两个页面，登录页面与主页面。

在本项目中，基于时间的项目重点考虑，弱化了前端 UI，参考了开源的 HTML 页面。

#### 5.1.1 登录页面



登录页面主要包括一个二级标题标签和一个重要的表单标签（包括三个输入标签：username、password、submit）。

在填写正确的用户名和密码后，点击 SUBMIT 按钮进入主页面。

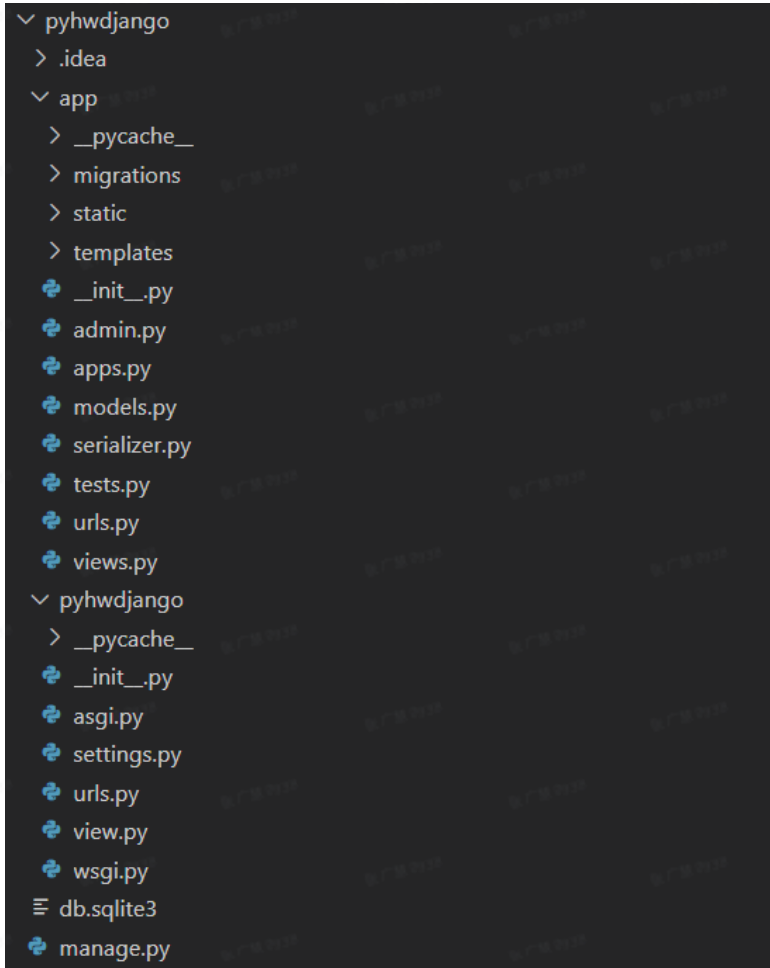
### 5.1.2 主页面



主页面与登录界面类似，在盒子内显示数据、文字与图片，点击 START 开始进行手势识别。

## 5.2 后端

后端使用 Django 开发，后端项目架构如下：



其中，pyhwdjango 是项目包文件夹：

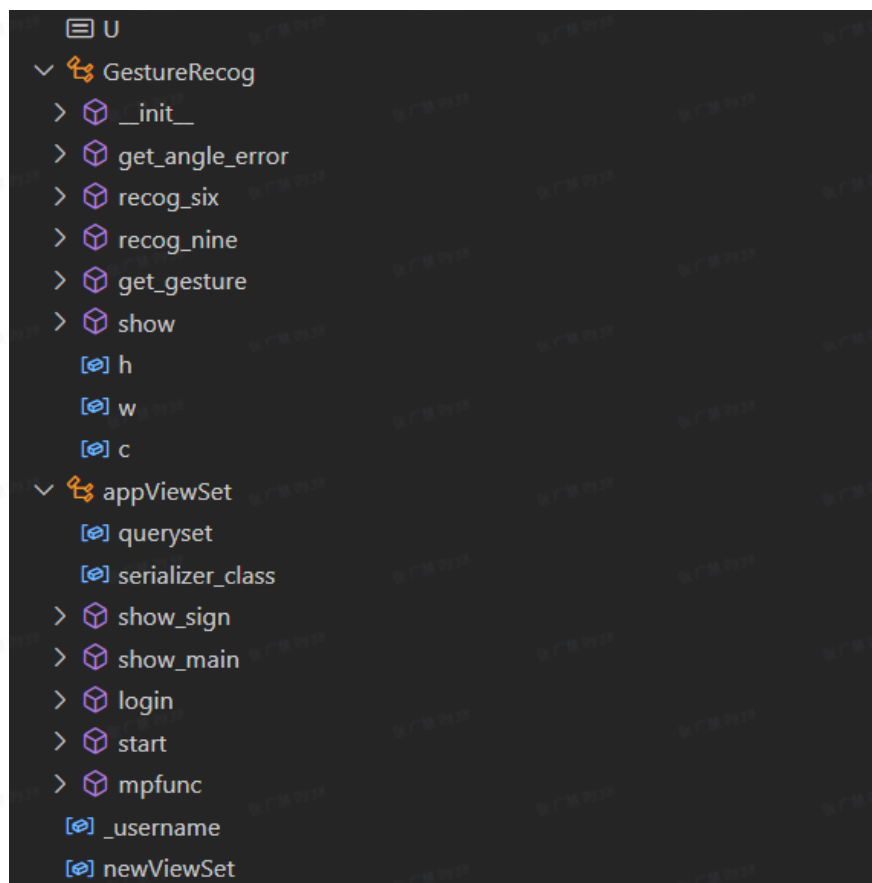
- setting.py 包含了django项目启动的所有配置项
- urls.py 配置项目的路由
- view.py 是项目的视图文件

app 是用户自建的应用文件夹：

- 新建应用后，需要注册到 setting.py 中
- apps.py 启动类
- models.py 模型文件
- serializer.py 序列化文件，在此注册模型
- test.py 单元测试
- urls.py 配置应用的路由
- view.py 应用的视图

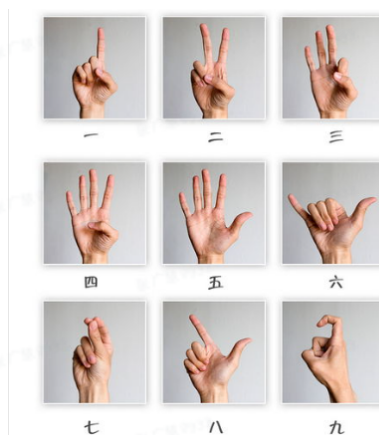
在构建应用的过程中，views.py 是编写应用功能的文件，在配置完其他文件后，在 views.py 编写手势识别和路由跳转触发代码：

手势识别类和视图集类大纲如下：



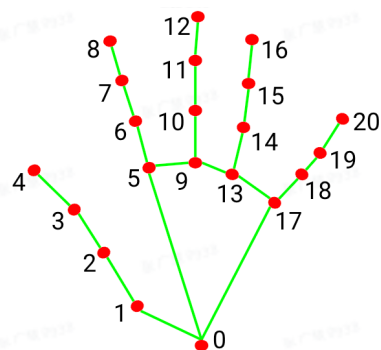
#### ■ 手势识别类：GestureRecog

识别的手势可参考下图：



手势识别通过手指的弯曲与伸直、关节处的夹角判断：

- 通过计算四个关节（如下图点1、2、3、4）处的反正切值求出四个关节处的角度，角度和小于一定值则视为手指伸直。
- 通过计算向量内积与叉积判断是否正交或平行，这一点用于判断手势六和手势九。
- 在这种基础上，可以进行扩展，识别更多手势。



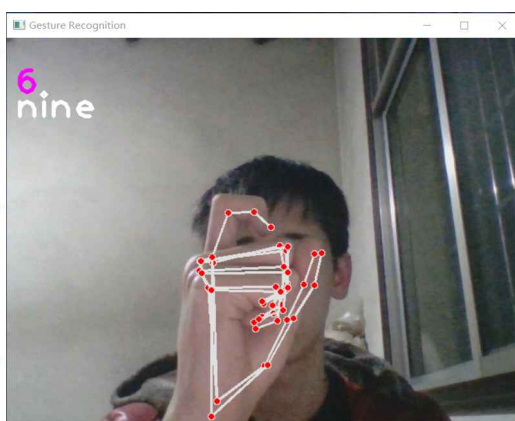
- 0. WRIST
- 1. THUMB\_CMC
- 2. THUMB\_MCP
- 3. THUMB\_IP
- 4. THUMB\_TIP
- 5. INDEX\_FINGER\_MCP
- 6. INDEX\_FINGER\_PIP
- 7. INDEX\_FINGER\_DIP
- 8. INDEX\_FINGER\_TIP
- 9. MIDDLE\_FINGER\_MCP
- 10. MIDDLE\_FINGER\_PIP

- 11. MIDDLE\_FINGER\_DIP
- 12. MIDDLE\_FINGER\_TIP
- 13. RING\_FINGER\_MCP
- 14. RING\_FINGER\_PIP
- 15. RING\_FINGER\_DIP
- 16. RING\_FINGER\_TIP
- 17. PINKY\_MCP
- 18. PINKY\_PIP
- 19. PINKY\_DIP
- 20. PINKY\_TIP

## ■ 视图集类：ViewSet

ViewSet 主要用于加载和跳转 HTML 页面、运行手势识别的函数。

最后的使用效果如下：可以看到能够识别出数字九。（左上角粉色数字为帧数）



## 5.3 数据库

数据库使用SQLite，在部署项目时创建模型并使用命令在数据库中生成表，即可在管理页面看到生成的 User 表：



本项目只实现了登录验证功能，在后端获取到 HTML 的 POST 请求后，获取表单输入的用户名和密码，使用 filter 和 get 获取数据库中的匹配的数据对象，验证是否正确。

## 六、跨域交互

### 6.1 后端接收到前端请求

在 HTML 文件中，使用表单发送数据，定义表单的 method 属性设置发送的请求为 POST 请求。

在定义视图集中的方法时，必须传入一个 request 参数，用于接收 HTML 发送的请求，获取数据。

```
try:
    corr_pwd = User.objects.filter(username=_username).values()[0].get('userpwd')
    if corr_pwd != password:
        print('密码不正确')
        messages.add_message(req, messages.INFO, 'Wrong Password')
        return redirect('http://127.0.0.1:8000/')
except IndexError:
    print('用户名不正确')
    # tkinter.messagebox.showinfo('提示', '登用户名或密码不正确')
    messages.add_message(req, messages.INFO, 'Wrong Name')
    return redirect('http://127.0.0.1:8000/')
```

## 6.2 后端向前端发送请求

引入 django.shortcuts 库中的 render 方法，可以渲染指定页面并向其发送数据。

引入 django.contrib.messages 库中的 messages 模型，向 HTML 中的发送消息，在 HTML 的中设置拦截这个消息后的响应，插入 script 代码，生成弹窗。

```
{% if messages %}
<ul class="messages">
    {% for message in messages %}
    <script>alert('用户名或密码不正确')</script>
    {% endfor %}
</ul>
{% endif %}
```

## 七、功能测试

开发过程中，前端直接点击页面，测试是否能产生正确的交互；后端使用 POSTMAN 发送请求，测试是否能收发请求；或者在前端操作，测试终端是否有正确的输出。

### 7.1 登录

测试是否能正常登录：

- ☐ 输入用户名或密码
- ☐ 点击 SUBMIT 按钮



测试结果要求：

1. 输入错误时，产生弹窗，并刷新页面；
2. 输入正确时，进入主页面，并显示“欢迎！你的用户名”字样

## 7.2 手势识别

测试是否能成功打开窗口并识别手势：

- ☐ 点击 START
- ☐ 做出手势
- ☐ 按 ESC 关闭



测试结果要求：

1. 成功弹出窗口
2. 识别出正确的手势
3. 成功关闭窗口返回主页面

## 八、参考文档

