# 视频标注及追踪

## 1. 环境

追踪部分:需安装OpenCV2

标注及预处理:需安装Python3以及PIL等库

### 2. 运行

追踪部分:安装配置好OpenCV2后,执行以下命令

g++ -L CV2库路径 -I CV2头文件路径 main.cpp ./KCF/kcftracker.cpp ./KCF/fhog.cpp -o main.o

标注及预处理: 直接 python3 preprocessing.py

## 3. 文件

/KCF: KCF Tracker核心代码 main.cpp:视频追踪主函数

Preprocessing.py:视频标注及预处理代码

### 4. 说明

注意:对于要追踪的视频,应确保视频的移动速度足够慢,否则结果会有较大误差!

本次追踪,由于视频比较多,故要对所有视频进行批处理。这就需要将不同的视频依次放在名为0,1,2...的文件夹下,便于生成下面提到的images.txt、region.txt以及多视频自动连续追踪。

#### 追踪部分

运行后输入要追踪的视频总数

输入文件说明:

images.txt:包含要追踪视频的所有图片帧的路径,如下:



region.txt:追踪视频第一帧所选区域的四个顶点的宽高信息,如下:



#### 输出文件说明:

output.txt : 依次存储所有视频的每一帧的追踪结果,格式如下:

```
Open with TextEdit

filename, d1, d2, d3, d4, d5, d6, d7, d8, d9, d10, d11, d12, d13, d14, d15, d16, d17, d18, d19, d20, d21, d22, d23, d24, d25, d26, d27, d28, d29, d30, d31, d32, d33, d34, d35, d36, d37, d38, d39, d40, d41, d42, d43, d44, d45, d46, d47, d48, d49, d50, d51, d52, d53, d54, d55, d56, d57, d58, d59, d60, d61, d62, d63, d64, d65

0, e. png, 17148426518413,
0, 1. png, 17584426518413,
0, 2. png, 1688426518413,
0, 4. png, 1688426518413,
0, 5. png, 1558415518413,
0, 6. png, 1558415518413,
0, 7. png, 152840849813,
0, 9. png, 1488408409813,
0, 10. png, 134884084013,
0, 11. png, 1398398498413,
0, 12. png, 1388398498413,
0, 13. png, 1358388488411,
0, 15. png, 1368388486411,
0, 15. png, 13183388448411,
0, 17. png, 13183388448411,
0, 19. png, 128388448411,
0, 19. png, 128388448411,
0, 19. png, 128388448411,
0, 20. png, 128388448411,
0, 21. png, 1383884844811,
0, 22. png, 138388448411,
0, 22. png, 1383884844811,
0, 22. png, 1158388448411,
0, 23. png, 128388448411,
0, 24. png, 128388448411,
0, 25. png, 1158388448411,
0, 25. png, 1158388484811,
0, 25. png, 1158388484811,
0, 25. png, 1158388484811,
0, 25. png, 1158388484811,
0, 25. png,
```

#### 标注及预处理部分

对于文件路径,直接看代码即可,有注释。

### 函数说明:

frameNameProcessing():自动生成images.txt

regionProcessing():标注,运行后,会依次对所有视频的第一帧进行标注,在弹出的图片中选中4个点即可

downsample():处理后的图片尺寸较大,不便于跑CNN,故需下采样

classify():用于提取训练集和测试集,并生成train\_data.txt和test\_data.txt