墨迹大侠 自动钓鱼脚本

程序使用图像定位点击实现自动钓鱼，仅能在pc版微信小程序上使用(手机投屏下无法正常点击，原因未知)，需要挂在前台;

程序基于`python3.11`，使用了`opencv`与`pyautogui`库，在1080p分辨率下无法正常识别(待修复)， python是现学的故代码质量不高;

受到程序截图和识别时间间隔的限制，不能保证满蓄力条，不能保证拿到全部的宝箱，无法处理鱼箱容量满的情况;

若你想在本程序的基础上二次开发或修复bug，见程序原理解释 **本脚本做不到开箱即用**，若你想使用本程序进行钓鱼，见程序依赖安装与截选识别目标及采样点校准

程序运行展示

图1 程序运行动图

程序原理解释

图2 绝对点获取和识别区域及采样点计算

1. 准备和采样阶段中，对 `fishing_reel.jpg` 和 `handle.jpg` 进行位置识别，分别获取二者中心点位置(图2中绿色点1和红色点2)

2. 利用点1和2计算处相对位置点3(图中橙色点)，随后以点3为钓鱼条圆弧中心点向外偏置一定距离(为图中偏置半径，蓄力条与钓鱼条具有不同的偏置半径)，从5度至175度每5度计算一个采样点位置，记录采样点的像素坐标，记录于autofish.py的`fish_mark_pos`变量中

3. 进行第2条类似操作，计算蓄力条采样点的相对位置像素坐标，记录于autofish.py的`prog_mark_pos`变量中

4. 在点1和点2的基础上，计算出弧形钓鱼条的外接矩形(为图2中识别区域)左上与右下点在屏幕上的坐标位置，对该矩形区域持续截图，至此准备阶段完成

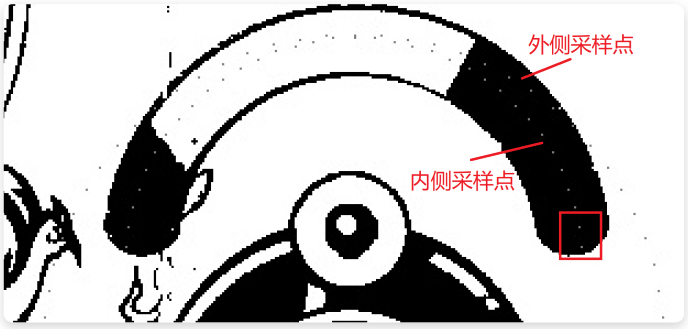

图3 蓄力阶段采样与判断

5. 蓄力阶段中，首先令鼠标移动至图2中点1位置，按下鼠标，蓄力条出现，蓄力条内白色段为实际蓄力条，黑色段为未满蓄力条

6. 图中钓鱼条内的两轮细密白点，较内侧白点为蓄力条采样点，较外侧白点为钓鱼条采样点，当图3红框内两个采样点的灰度值均为255时，表示钓鱼条蓄力值接近满，松开鼠标，此时蓄力条消失，钓鱼条即将出现

7. 在蓄力条消失至钓鱼条出现期间，通过死循环判断截图区域的平均灰度值，当发生灰度值跳变时表示钓鱼条出现，进行钓鱼

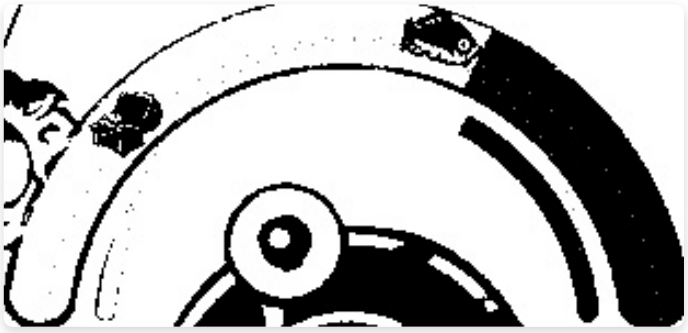

图4 识别区域二值化图

8. 钓鱼阶段中，钓鱼弧形框内存在3样显著元素：钓鱼条、宝箱和鱼，通过获取各个采样点处的灰度值，可以计算处钓鱼条的长度与当前钓鱼条的起始终止位置(`autofish.py中fishbar_begend`变量)，两个元素差值为钓鱼条长度

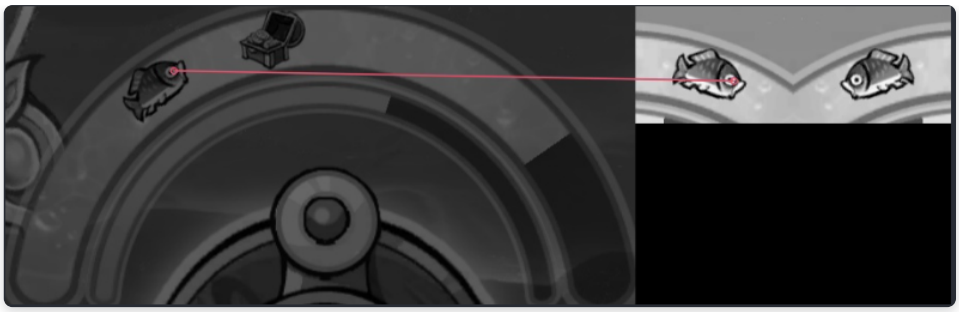

图5 ORB识别鱼的位置，调试模式下见pic/keypoints_match.jpg

9. 通过对截图和pic/fish.jpg分别使用ORB提取特征点匹配，选择出前3个最佳匹配位置的像素坐标，找出离这3个点的中心点最近的采样点位置，即为鱼最可能在的位置，通过按下或松开鼠标令该点始终被包含在钓鱼条内，直至钓鱼结束

以上为完整的钓鱼流程原理讲解，可以将autofish.py中bool_mark_debug变量改为True，进行程序运行调试(分别见控制台输出与pic/keypoints_match.jpg)，调试功能实际钓鱼的时候最好关闭，下图为调试流程动图(#表示钓鱼条，字母F表示鱼的位置), 忽略我的战力。

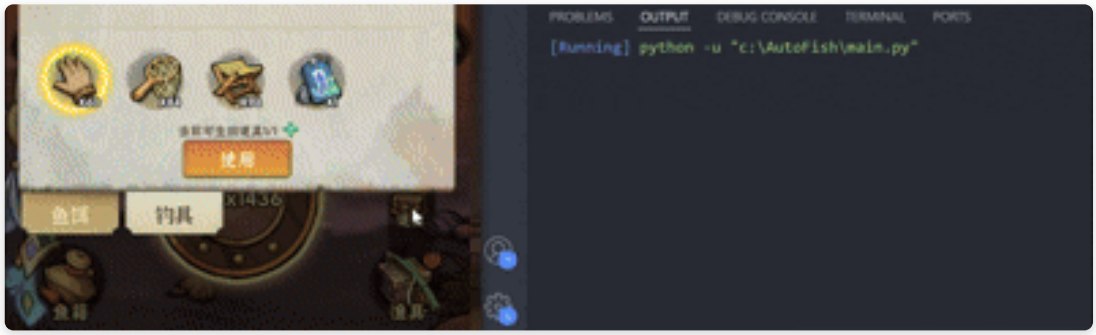

图6 调试模式下钓鱼及控制台输出

需要注意的是，图像识别需要一定的实时性，故程序运行稍微吃CPU单核性能，一般而言只要不是很老的CPU都不会无法运行程序，应该不会比我写程序的这台电脑还差。

程序依赖安装与采样点校准及截选识别目标

由于本程序不能自动适配多分辨率及图像识别存在差异性，计算出的采样点位置不一定在蓄力条和钓鱼条内部，故在钓鱼前需要对采样点进行校准和截选检测目标，以确保程序能够正常使用。

程序依赖安装

程序基于python3.11开发，尽量保持版本号的一致，不会安装python的见b站或其他网站python安装教程，本文档不会详解。

首先安装程序依赖的库，具体使用库见requirements.txt，安装命令为

```
pip install -r requirements.txt
```

安装过程可能会很慢或者无法下载，可能是python环境变量未正确配置或网络问题，首先在控制台输入pip --version再按下回车观察版本号是否正确输出，如果是下载超时则可以使用魔法或者镜像源，本文档以清华镜像源为例，使用如下安装命令

```
pip install -r requirements.txt -i https://pypi.tuna.tsinghua.edu.cn/simple
```

若安装成功，则依赖安装步骤完成。

截选检测目标

程序受限于本人水平，做不到自适应分辨率，故在程序运行前需要手动对识别目标进行截图，需要进行截图的图像依次见下图。

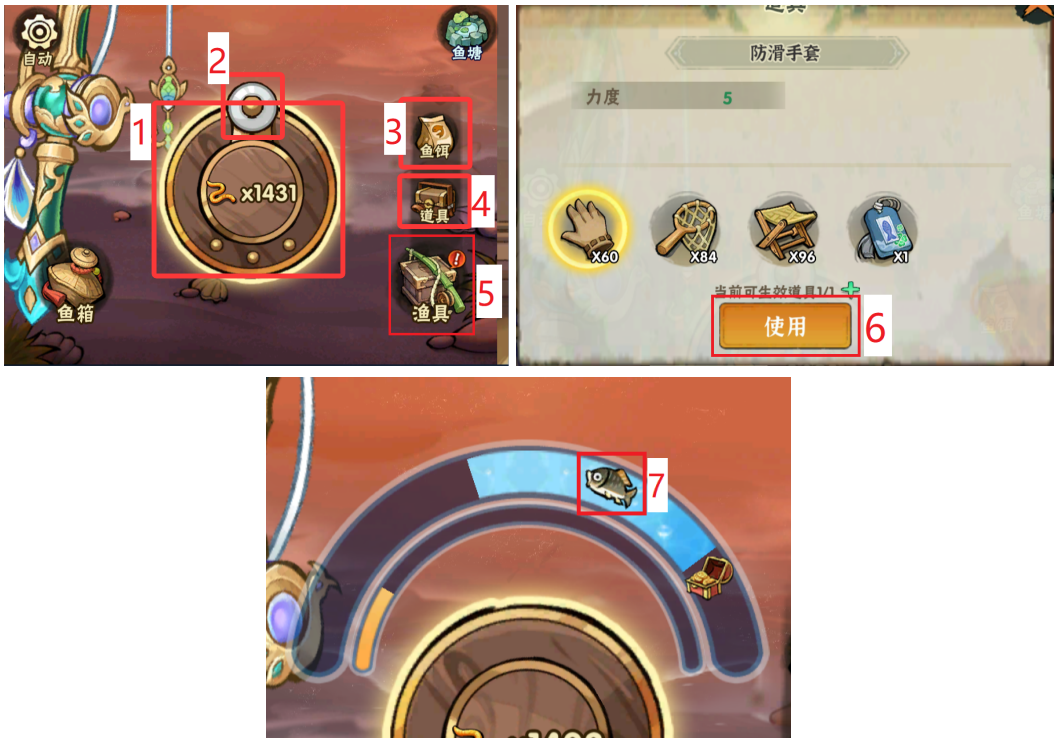

图7 需要进行手动截图的目标及其序号

图片序号	文件命名
1	fishing_reel.jpg
2	handle.jpg
3	bait.jpg
4	prop.jpg
5	fishing_gear.jpg
6	use.jpg
7	fish.jpg

文件后缀不强制为jpg，可以根据自己截图的文件后缀在程序中修改，修改`locate.py`中11-17行文件路径最后的jpg即可; 截图尤其是1和2，尽可能保证1和2的相切和居中，不要大了也不要小了，会影响后续的采样点校准； **将截取的图像放到pic文件夹中，注意不要打错文件名;**

需要注意的是，以后运行程序时窗口必须与截图时窗口大小一致，最简单的方法是始终将窗口拉到最大(上下顶格)。

采样点校准

受到截图分辨率和实际截图的误差，校准点不一定能准确被包含在弧形条内，从而影响蓄力和钓鱼的判断，故需要进行采样点校准，采样点校准在窗口大小不变的情况下只要校准一次，以后都可以直接使用。

首先，打开`locate.py`，将第25行的`mark_point_calibration` 变量改为`True`, 随后运行`main.py`，如果上一步截选检测目标正确执行了，程序运行后应该是至少可以进入到蓄力部分的。

部分人可能截图截得比较好，直接可以像图1一样走完整个钓鱼流程，那么恭喜你，不需要看后面的部分；如果你不能走完整个钓鱼流程，请务必看完下面的采样点校准部分。

程序运行之后，如果出现了钓鱼失败的界面，直接结束程序的运行。这时在项目中你会看见新建了一个 `test` 文件夹，打开文件夹之后，你会看见以数字依次命名的很多图像。

从中挑选出两张典型的图像：蓄力部分和钓鱼部分(见图8图9)

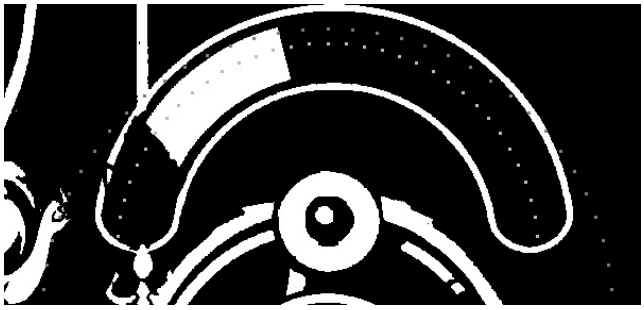

图8 蓄力时截图

图9 钓鱼时截图

可见内部存在两轮白色的采样点(见图3)，通过调整`autofish.py`中第26-33行的偏置参数，分别令两轮采样点都被包含在各自的弧形条中(图8、9为正确的示例)；

```
# 钓鱼条采样点校准参数
fish_radius = 0.88  # 钓鱼条采样点偏置半径
fish_mark_x_offset = 0
fish_mark_y_offset = 0
# 蓄力条采样点校准参数
prog_radius = 0.65  # 蓄力条采样点偏置半径
prog_mark_x_offset = 0
prog_mark_y_offset = 0
```

根据图中坐标系的指示修改对应的偏置参数，偏置单位为像素，如想将蓄力条的采样点整体向下侧移动30个像素，则将`prog_mark_x_offset`变量后的数字改为`30`；如想将蓄力条的采样点整体向左侧移动50个像素，则将`prog_mark_y_offset`变量后的数字改为`-50`，修改后代码和示例图图下：

```
# 参数修改示例
prog_mark_x_offset = 30
prog_mark_y_offset = -50
```

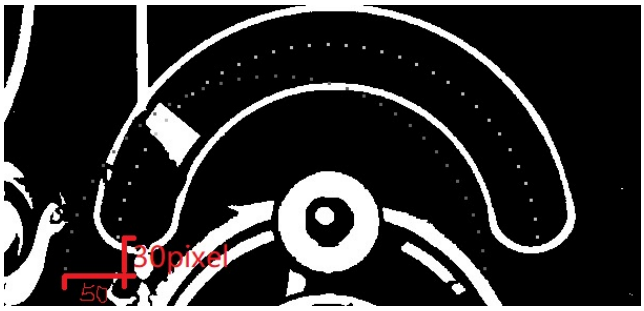

图10 蓄力条采样点向左下偏置示例图

经调整后，运行```main.py```进行钓鱼测试，直至两轮采样点均符合图8、9的示例，应当可以走完整个钓鱼流程，自此采样点校准和掉本准备工作完成，可以进行自由钓鱼。

总结

当前脚本使用前确实需要进行不少操作，后续可能会进行简化，开盒即用，不过那也是以后的事了...