算法库调用说明 (Linux .so 版)
库文件：libCylinderTravelDetector.so
运行环境：Ubuntu 20.04/22.04 (x64)，依赖 OpenCV 4.x (core, imgproc) 及 ONNX Runtime。

头文件声明 (CylinderTravelDetector.h)
为了在 C++ 中使用，且支持 C 风格的 dlopen 动态加载，必须使用 extern "C" 包裹接口。

cpp
```
#ifndef CYLINDER_TRAVEL_DETECTOR_H
#define CYLINDER_TRAVEL_DETECTOR_H

#include <string>

// 检测结果结构体
struct DetectionResult {
    float confidence;      // 缸体识别置信度
    float stroke_px;       // 活塞杆行程像素距离
    float stroke_mm;       // 换算后的实际物理距离 (mm)
    float cx1, cy1;        // 行程起点坐标
    float cx2, cy2;        // 行程终点坐标
};

// 使用 extern "C" 防止 C++名称修饰 (Name Mangling)
#ifdef __cplusplus
extern "C" {
#endif

    // 初始化算法配置
    // 返回值：0 成功，其他失败
    int Init(const char* config_path);

    // 检测输入图像并计算行程
    // image_path：输入图片路径
    // px_per_mm：像素标定系数 (例如 1mm = 2.5 像素)
    // result：输出检测结果结构体指针
    int Detect(const char* image_path, float px_per_mm, DetectionResult* result);

    // 释放资源
    void Release();

#ifdef __cplusplus
}
#endif
```

#endif // CYLINDER_TRAVEL_DETECTOR_H
调用方式一：静态链接 (推荐)
在编译你的主程序 main.cpp 时，使用 -l 参数链接生成的 .so 文件。

bash
编译命令 (假设头文件在 include 目录，库文件在 lib 目录)
g++ main.cpp -I./include -L./lib -lCylinderTravelDetector -lopencv_core -lopencv_imgproc -o detector
调用方式二：动态加载 (dlopen / dlsym)
适用于不想在编译期硬依赖库，或者开发插件系统的场景。

cpp
```
#include <dlfcn.h>
#include <iostream>

int main() {
    // 1. 加载动态库
    void* handle = dlopen("./libCylinderTravelDetector.so", RTLD_LAZY);
    if (!handle) {
        std::cerr << "加载库失败: " << dlerror() << std::endl;
        return -1;
    }

    // 2. 获取函数指针
    typedef int (*InitFunc)(const char*);
    typedef int (*DetectFunc)(const char*, float, DetectionResult*);
    typedef void (*ReleaseFunc)();

    auto Init = (InitFunc)dlsym(handle, "Init");
    auto Detect = (DetectFunc)dlsym(handle, "Detect");
    auto Release = (ReleaseFunc)dlsym(handle, "Release");

    // 3. 调用
    Init("model_config.yaml");

    DetectionResult res;
    if (Detect("frame_001.jpg", 2.5f, &res) == 0) {
        std::cout << "置信度: " << res.confidence
                  << ", 行程: " << res.stroke_mm << " mm" << std::endl;
    }

    Release();

    // 4. 卸载库
```

```
    dlclose(handle);
    return 0;
}
```
(注意：编译上述代码时需加 -ldl 参数)