



全国大学生集成电路创新创业大赛  
CHINA COLLEGE IC COMPETITION

# 智慧之眼： 视障人士的户外安全出行伴侣

第八届 全国大学生集成电路创新创业大赛

飞腾杯

团队名称：飞腾风驰队

队伍编号：CICC1085





# 目录

## 1 背景介绍

传统辅助方式存在局限  
智慧之眼实现自由出行

## 3 应用场景

- (1) 视障人士端
- (2) 监护人端

## 5 关键技术

- (1) 功能应用创新点
- (2) 系统应用创新点

## 7 总结

2大系统, 4大出行场景  
5大功能模块, 8大关键技术

## 2 系统概况

系统层次与技术规格  
简介

## 4 系统多任务布局 与核间通信分配

## 6 运行效果

- (1) 功能实测
- (2) 耗时实测



# 1 背景介绍



视障者的日常生活面临诸多挑战，实现自由安全的户外出行困难重重

## 传统辅助出行 方式存在局限

**探路杖**: 仅可感知局部信息障碍

**无法协助主动避障**

**导盲犬**: 仅有引路、避障功能

犬类色盲，**无法安全识别交通灯**，  
过马路场景风险高

**无法提供自然语音提示**

**监护人陪伴**: 难以实现独自出行需求，  
**出行机会受制于人**

改进后

## 智慧之眼 实现自由出行

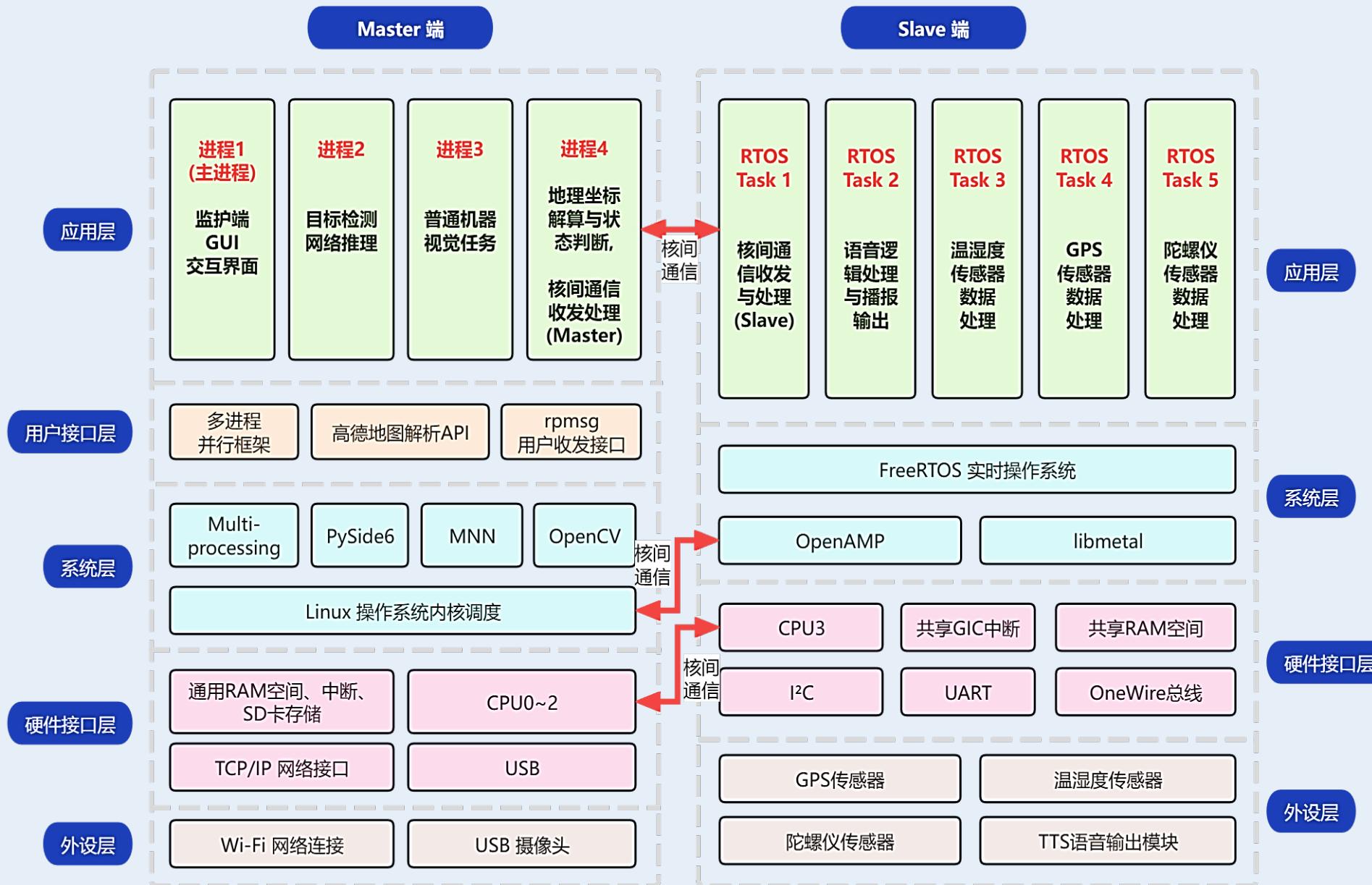
**全方位覆盖户外城市人行道路  
出行核心场景**

**AI智慧视觉支撑**: **障碍物方位、斑马线走向、  
交通灯状态、梯级数目**多种感知一步到位

**便捷语音提示**: **实时自然语音关怀**，一听  
就懂，协助视障者出行时准确决策判断

**监护人实时监控**: **实现独立出行自由**，同  
时提供监护人客户端，远程**保障双重安全**

# 2 系统概况



## 技术规格

飞腾派 E2000Q 开发平台  
Ubuntu Linux / FreeRTOS 系统  
**无外置神经网络加速器或显卡**  
**外接:** 温湿度传感器、GPS、  
三轴加速度指南针、TTS中文语音  
播报模块、USB摄像头

## 2 大系统

- Master 系统: 高性能核
- Slave 系统: 高实时核

## 5 大模块

- Master系统 应用模块:**
  - 机器视觉处理模块
  - 系统状态与感知模块
  - 监护客户端显示模块
- Slave系统 应用模块:**
  - 传感器处理与语音反馈模块  
(共使用4个rpmsg endpoint)
- 底层模块:**
  - 跨系统核间通信模块

### 3 应用场景: (1) 视障人士端

有效覆盖视障人士户外人行道出行  
所有核心应用场景



#### 机器视觉处理模块

1. 障碍物目标检测: YOLOv8n 自训练模型, 50 classes

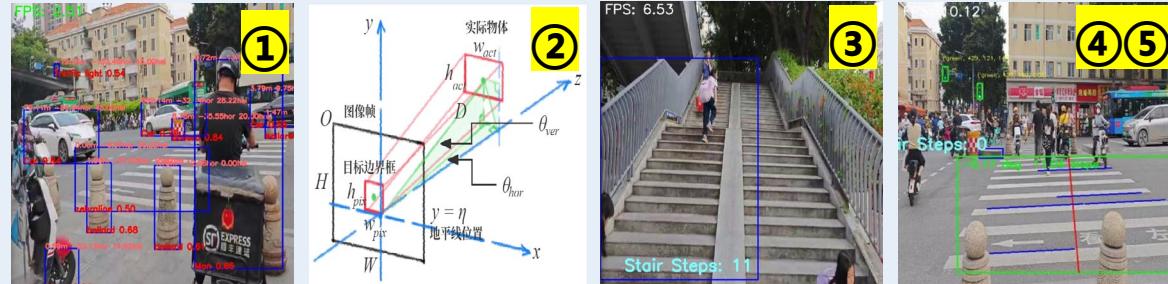
2. 目标距离角度解析:

单目视觉 + 实际尺寸推断 + 距离、高度角、水平角坐标变换

3. 人行天桥楼梯级数识别: 基于目标检测框图像部分  
边缘&直线线段检测 → 斜率  $k$  与截距  $b$  直方图提取 → 波峰数输出

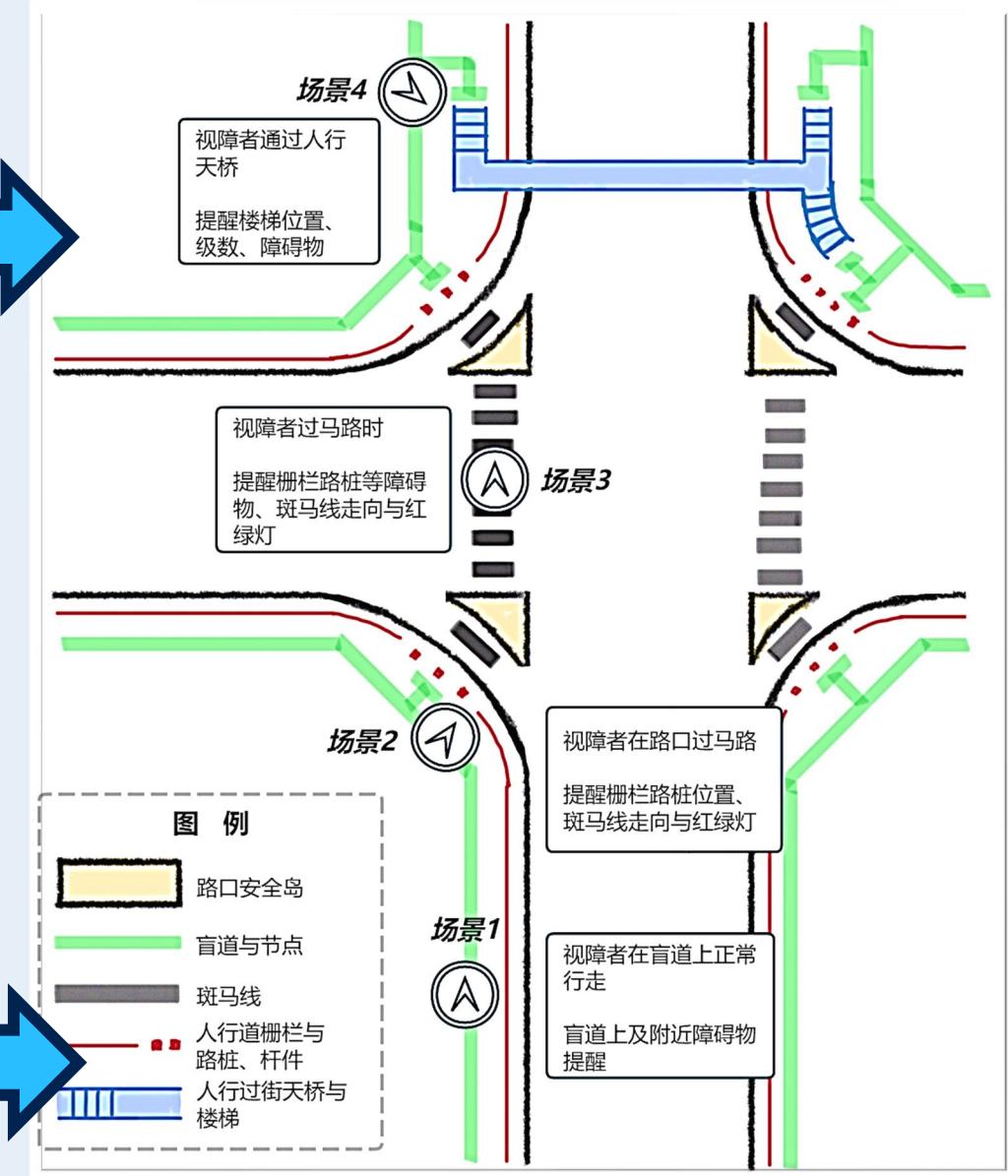
4. 斑马线走向方位检测: 基于目标检测框图像部分  
边缘&直线线段检测 → 长线段筛选&密集线段筛除 → 垂直线输出

5. 交通灯状态检测: 基于目标检测框图像部分  
目标框 $6 \times 8$ 网格分割 → 阈值判断亮灯状况(红, 绿, 非亮灯) → 输出



#### 语音反馈播报输出

根据障碍物密度、路口状况、人行天桥状况高频播报  
实时为视障者提供前方状况语音反馈



### 3 应用场景：(2) 监护人端



#### 监护人端 GUI 界面设计



#### 监护客户端GUI显示模块

- 实时画面显示：**视障者前方视野画面、障碍物物体检测信息
- 实时状态显示：**视障者当前位置信息、摔倒状态、视角状态、温湿度信息
- 特色小地图功能：**俯视+前方视角  
支持视障者前方视野内 10m 不同方位物体直观显示
- 系统性能监控：**Master 端 CPU 各核心使用率、系统响应时间、核间通信响应时间

#### 系统状态与感知模块

##### 加速度传感器：

- 摔倒检测：**任一方向线加速度  $> 3\text{m/s}^2$

##### 陀螺仪传感器：

- 视角状态：**保证视角位于视障者正前方，当视角发生垂直旋转倾斜，提醒视障者复位

##### 温湿度传感器：

- 舒适度检测：**保障视障者出行舒适安全、提供天气说明

#### 逆地理坐标解析

① GPS 传感器坐标  
高德地图在线 API  
逆地理解析

② 建构周边  
道路图拓扑

$$(x_1, y_1, x_2, y_2)$$

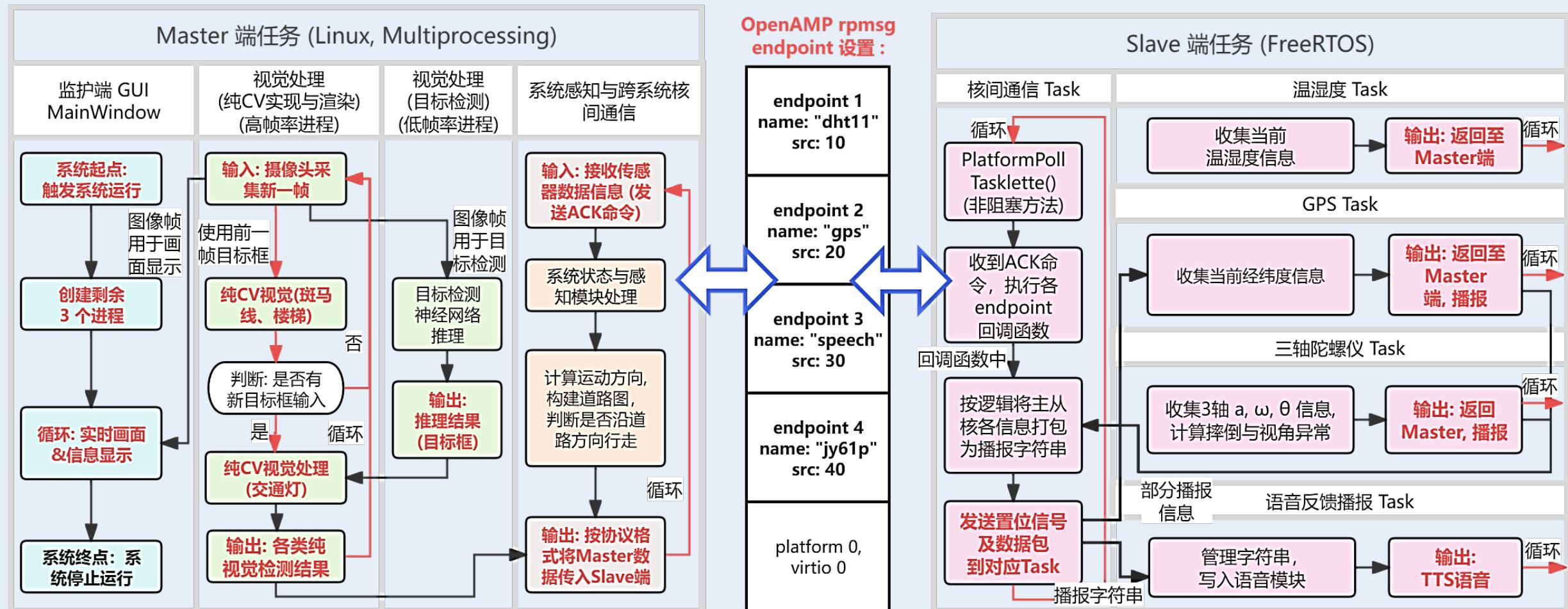
③ 滑动窗口  
计算当前视障  
者运动方向

$$\begin{aligned}\Delta y &= (y_1 - y_4) + (y_2 - y_5) + (y_3 - y_6) \\ \Delta x &= (x_1 - x_4) + (x_2 - x_5) + (x_3 - x_6) \\ \theta &= \arctan(\frac{\Delta y}{\Delta x}) - 90^\circ\end{aligned}$$

④ 找到最近  
的道路，计算  
角度偏移  
判断是否沿道  
路方向行走



# 4 系统多任务布局与核间通信分配



## Master 系统

**主进程:** 监护端 GUI MainWindow (QThread)  
**子进程:** 共3个, Multiprocessing 进程实例, 并行互不干扰, 负责功能具体实现; 生命周期由主进程控制

## OpenAMP rpmsg

4 个静态 endpoint, 位于同一 remoteproc platform 的同一 virtio; 并行信道通信, 互不干扰  
 每个 endpoint 负责各自的传感器或语音模块传输

## Slave 系统

基于 FreeRTOS, 共 5 个 Task, 时间片轮转运行  
 分别完成 rpmsg 传输、各传感器及播报处理输出  
 充分利用核间传输等待时空间隙性能, 减轻Master负担

# 5 关键技术：(1) 功能应用创新点



## 1. 楼梯级数识别算法迭代

### 改进前：[版本1]

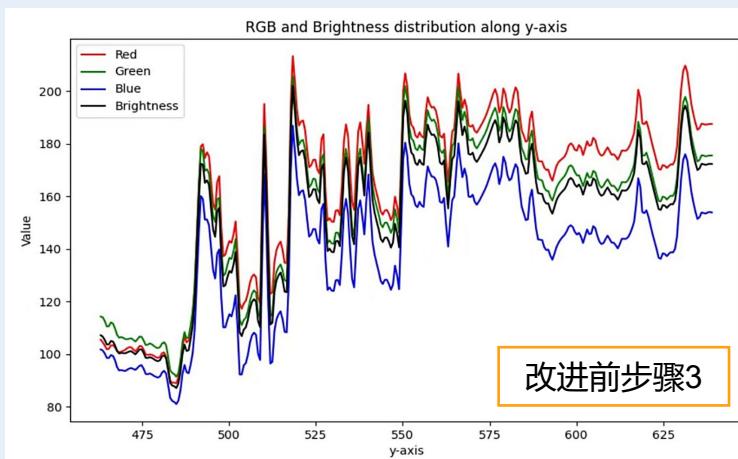
步骤1：目标框内线段提取

步骤2：根据斜率判断楼梯走向

步骤3：过框中点沿走向作线段  $l$ , 每  $y$  值左右11像素点平均采样, 利用RGB通道亮度值直方图波峰分析

总时间复杂度：

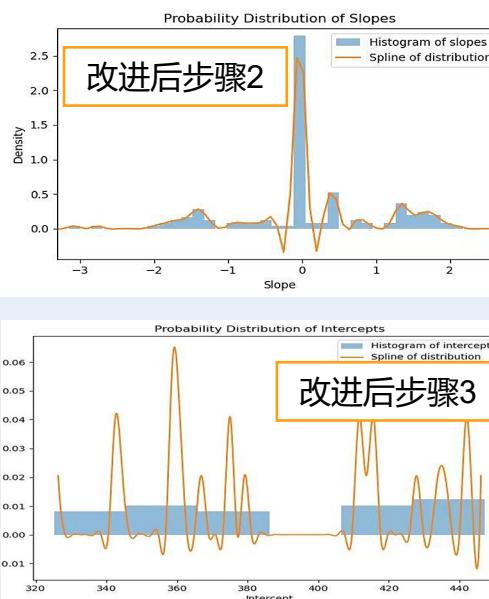
$$O(6xy + 3xy^2 + n)$$



(注:  $x$  目标框横轴像素数,  $y$  纵轴,  $n$  线段数)

### 存在问题：

- 不同亮度、纹理楼梯, 波峰形态各异, 滤波阈值设定困难
- 不同纹理下楼梯走向判断复杂, 侧视楼梯效果不佳
- 数据量过大, 性能低



### 改进后：[版本2]

步骤1：目标框内线段提取:

Canny 算子检测边缘 + Hough 变换检测线段

步骤2：

斜率  $k$  频率分布直方图

$$h_j = \#\{s_i | b_j \leq s_i < b_{j+1}\}$$

$$x_j = \frac{b_j + b_{j+1}}{2}$$

提取  $k \approx 0$  波峰附近线段

$$k = \arg \max_{j \in P} h_j$$

$$\{l'_i\} = \{l_i | x^* - \delta \leq l_i^0 \leq x^* + \delta\}$$

$$x^* = x_k$$

步骤3：

截距  $b$  直方图

$$\{b_i\}_{i=1}^n$$

$$H(b, 60)$$

曲线拟合平滑化

$$S(x) = \sum_{j=1}^m B_j(x) \cdot \beta_j$$

$$\min \left\{ \sum_{i=1}^n [c_i - S(x_i)]^2 + s \int [S''(x)]^2 dx \right\}$$

分析截距  $b$  直方图峰数

总时间复杂度：

$$O(2xy + 2n)$$



## 5 关键技术：(1) 功能应用创新点



## 2. 自主训练目标检测网络与轻量化处理

## ① 数据集原始采集与标注

针对实际场景，拍摄自己的原始数据集。

**36.4GB, 4.8k原始图片, 80k labels**

**标注方法：**利用OIV7预训练模型进行预标注，同时添加符合项目需求的 class

## 数据增强：5种方式，提升泛化能力

**采集地点:** 广州市海珠区赤岗街道

# 训练模型：YOLOv8n 目标检测网络

**训练轮次:** 175 epochs



## 验证集测试效果

## ② 模型训练结果

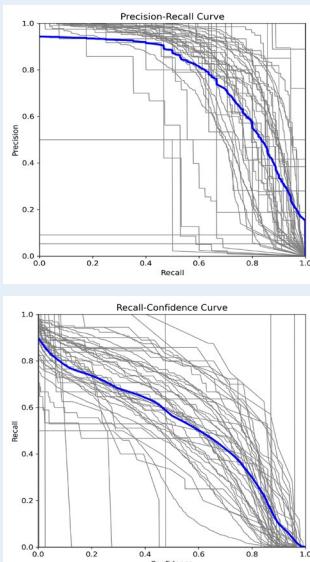
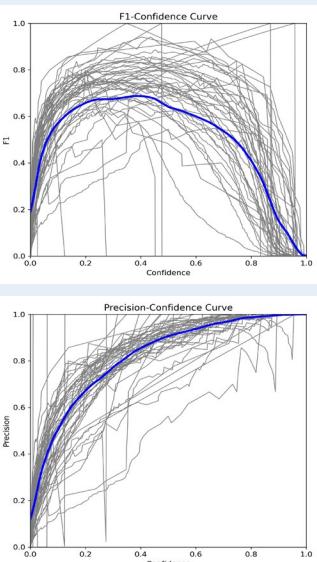
由于主要class和次要class数据量悬殊，

实际F1-置信度平均曲线更高(如图)

F1曲线平台区右倾，表明模型平均精确率

(Precision) 效果优于平均召回率(Recall) 效果

F1-置信度	<b>0.69</b> (图示偏低)	P-R <b>mAP@0.5</b>	<b>0.758</b>
P-置信度	1.0 at 0.984	R-置信度	0.90

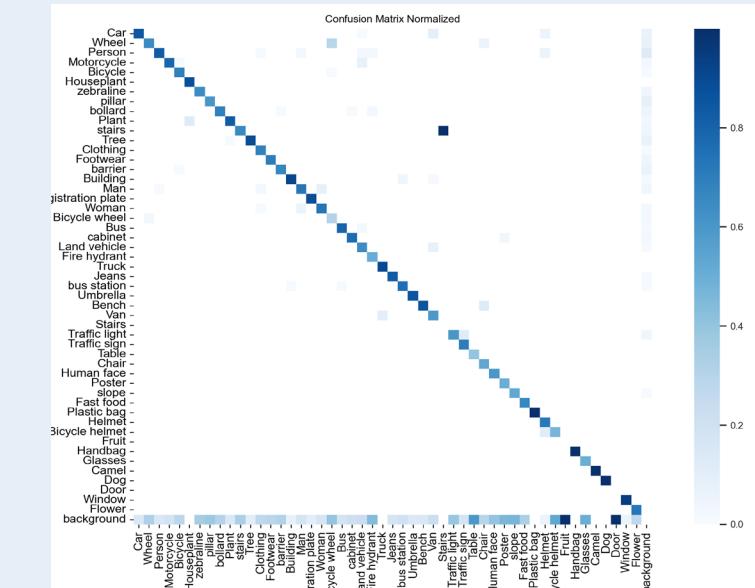


### ③ 数据集评价

训练测试集比：14% 测试集，86% 训练集

绝大多数关键class, TP值大小 > 0.8

部分数据集较少class, TP值大小 > 0.6



## ④ 轻量化处理

针对实际开发板紧张性能，由 **FLOAT16**  
量化转换为 **INT8** 量化，速度大幅提升

# 5 关键技术：(1) 功能应用创新点



## 3. 单目距离角度解算 + 小地图

**步骤1：**基于障碍物目标推理结果，根据**边界框大小&位置**，以及不同class**实际物体宽/高经验值**，推断**实际尺寸**

**步骤2：**经由公式计算**物体与视角之间距离、水平角、垂直角**，绘制**小地图 (GUI界面)**

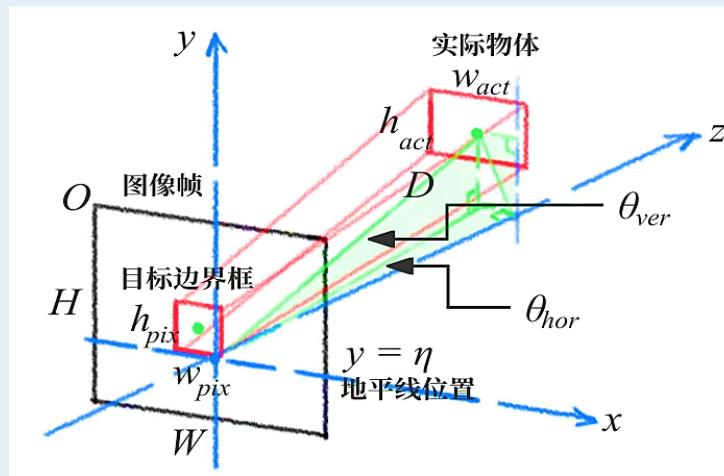
**距离计算：**

$$D = \frac{h_{act}}{h_{pix}} \times \frac{f}{\cos(\text{abs}(\theta_{hor}) \times \cos(\text{abs}(\theta_{ver}))}$$

$$D = \frac{w_{act}}{w_{pix}} \times \frac{f}{\cos(\text{abs}(\theta_{hor}) \times \cos(\text{abs}(\theta_{ver}))}$$

**水平角度计算：**  $\theta_{hor} = \left( \frac{\Delta_{pix}}{W/2} \times \theta_{hormax} + \theta_{horbias} \right) \times k_{\theta_{hormax}}$

**仰角计算：**  $\theta_{ver} = \theta_{vermax} \times \left( \frac{H \times \eta - y_{center}}{H} \right) \times k_{\theta_{ver}}$



## 4. 原创判断视障人士沿路行走方法

### ① 高德API逆地理数据解析

GPS模块经纬度原始坐标 → 逆地理解析API → 返回参数 → 获取**道路线段信息**

$(x, y)$

<location>坐标点</location>

$(x_1, y_1, x_2, y_2)$

**② 周边道路图建构：** 合并距离较近 (40m) 的道路节点，并构建**周边道路图信息**

**③ 当前运动方向计算：** 滑动窗口法，根据最近的位置信息序列得到**当前运动方向**

$$\Delta y = \frac{(y_1 - y_4) + (y_2 - y_5) + (y_3 - y_6)}{3} \quad \Delta x = \frac{(x_1 - x_4) + (x_2 - x_5) + (x_3 - x_6)}{3} \quad \theta = \arctan\left(\frac{\Delta y}{\Delta x}\right) - 90 \text{ deg}$$

**④ 找到最近道路，计算角度偏移，判断是否沿道路行走**

$$d = \min \left( \frac{|(y_2 - y_1)x - (x_2 - x_1)y + x_2y_1 - y_2x_1|}{\sqrt{(y_2 - y_1)^2 + (x_2 - x_1)^2}} \right) \quad \theta_{road} = \tan^{-1} \left( \frac{y_2 - y_1}{x_2 - x_1} \right) \quad \Delta\theta = \theta_{road} - \theta_{current}$$

## 5. 原创斑马线方向与交通灯识别方法

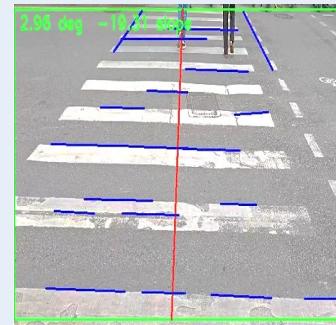
### 斑马线方向识别：

- ① 目标框内直线线段提取
- ② 长度 l 直方图，取较长线段集
- ③ 取垂线，做修正，得斑马线真实走向角

$$\bar{m} = \frac{1}{n} \sum_{i=1}^n m_i \quad m_{\perp} = -\frac{1}{\bar{m}}$$

$$m_{corrected} = m_{\perp} \cdot \theta$$

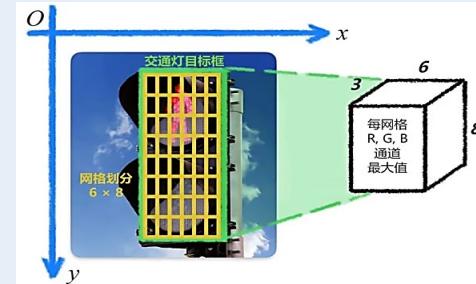
$$\alpha = \arctan(m_{corrected})$$



识别效果准确

### 交通灯颜色识别：

目标框划分 $8 \times 6$ 网格 + 亮灯阈值判断



# 5 关键技术：(2) 系统应用创新点



## 6. Master 端多进程并行框架

### 版本1：

- 实现方式：**使用 `PySide6::QThread` 类实现
- 运行过程：**程序将视频读取、目标网络推理、检测结果再处理分别置于三个独立线程下进行，通过 PySide 的信号与槽机制在不同线程间通信。
- 问题：**GIL限制；出现间歇性图像帧输出卡顿问题

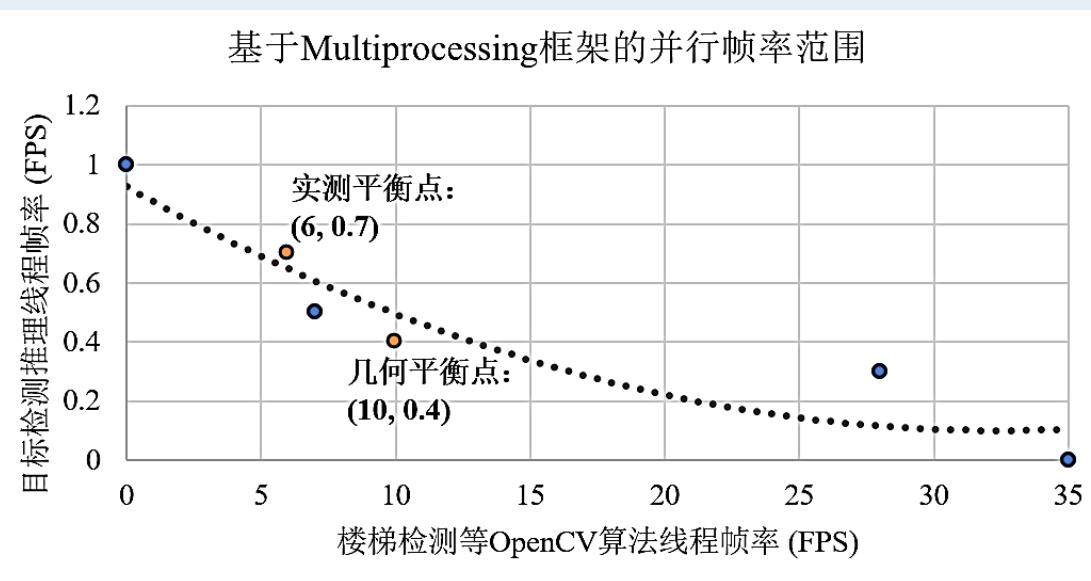
### 版本2：

- 实现方式：**使用 `Multiprocessing` 类实现
- 运行过程：**两个独立进程循环，纯CV循环进程频率较高，完成视频帧读取、CV算法执行；目标检测循环进程频率较低，从缓存中获取最新帧，完成耗时的目标检测过程。
- 优点：**实现真实并行运转，解决间歇性图像帧卡顿问题
- 缺点：**算力资源分配需要通过每个循环的适当延时平衡

**效果：**目标检测推理 0.7FPS、纯CV算法+监护端实时显示 6FPS，实现高性能与实时性任务同时运转

### 飞腾派硬件 (E2000Q) Master 端并行运转下， 纯CV算法与目标检测帧率间帧率规律的矛盾关系示意

Master 端运行 CPU	CPU0, 1, 2, 共3个 (CPU3关闭)
CPU 占用率	运行中CPU最低值均 > 90%
推理网络格式	MNN (OpenCV4) 格式 Float16 量化
纯CV算法环境	OpenCV4 (Python)



# 5 关键技术：(2) 系统应用创新点



## 7. Master 端 Linux 内核核间通信驱动改进

**修改前：**virtio 的 endpoint 在 Linux 内核中的设备名称与随机分配的 **MINOR 设备号**相关，导致多 endpoint 静态生成时无法区分与一一对应，调用 ioctl 时无法正常开启

**修改后：**将 **endpoint /dev 设备名称**命名规则与 **src 字段绑定**，实现不同 endpoint 在 Linux 内核中的**设备名称的唯一性**

**修改内核函数** /drivers/rpmsg/rpmsg.c::rpmsg\_eptdev\_create()

```
1 static int rpmsg_eptdev_create(struct rpmsg_ctrldev *ctrldev,
2                                struct rpmsg_channel_info chinfo)
3 {
4     ...
5     u32 destnum; //用于存储终端的src编号
6     ...
7     destnum = ctrldev->rpdev->dst;
8     dev_set_name(dev, "rpmsg_%d", destnum); //设置名称
9     ...
10 }
```

**修改内核函数** /drivers/rpmsg/rpmsg.c::rpmsg\_chrdev\_probe()

```
1 static int rpmsg_chrdev_probe(struct rpmsg_device *rpdev)
2 {
3     ...
4     u32 destnum; //用于存储终端的src编号
5     ...
6     destnum = rpdev->dst;
7     dev_set_name(&ctrldev->dev, "rpmsg_ctrl_%d", destnum); //设置名称
8     ...
9 }
```

## 8. Slave 端 rpmsg 非阻塞平台轮询实现

由于飞腾 **phytium-freertos-sdk** 中 /third-party/openamp/ports/platform\_info.c 的轮询机制 **只有阻塞实现**，实际运行时也只能在 baremetal 环境中运行，而无法在具有 FreeRTOS 的环境下运行，这是因为它会**阻塞 FreeRTOS 的任务调度机制**。

**修改后的 platform\_poll() 函数：**实现与RTOS任务调度器共存

```
1 int PlatformPollTasklette(void *priv)
2 {
3     struct remoteproc *rproc = priv; //传参传入的是platform, remoteproc平台的指针
4     struct remoteproc_priv *prproc;
5     unsigned int flags;
6     int ret;
7
8     prproc = rproc->priv;
9     while (1) {
10        if (metal_io_read32(prproc->kick_io, 0) & 0x2) { //RPROC_M2S_SHIFT
11            ret = remoteproc_get_notification(rproc, RSC_NOTIFY_ID_ANY);
12            if (ret) {
13                return ret;
14            }
15            break;
16        }
17        (void)flags;
18        vTaskDelay(100 / portTICK_PERIOD_MS); //延时100ms, 这个需要放在while(1)内部
19    }
20    return 0;
21 }
```

经测试，新加入的 vTaskDelay 延时时长设定经验值：

静态 endpoint 数量为1 **1ms**

静态 endpoint 数量为4, 且加入各类回调函数处理 **100ms**



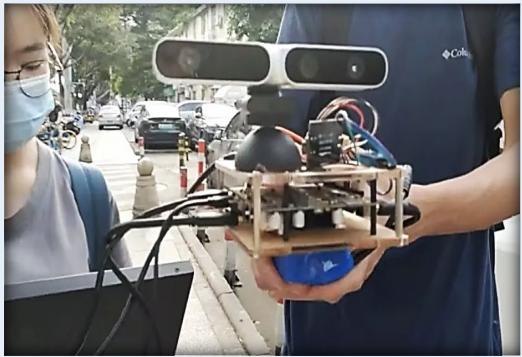
# 6 运行效果：(1) 功能实测

## 实测时间地点

**时间：**2024年5月，工作日（周一）下午16: 30~17: 30

模拟**交通繁忙的典型城市道路场景**

**地点：**广州市海珠区赤岗街道新港中路



## 人行道模式 场景1：一般人行道场景

**播报内容：仅有障碍物检测**

- “**前方宽松，中距多障碍**”
- “**11, 0, 2, 0, 1**” (从近到远：  
树木, 小轿车, 行人, 小轿车, 车轮)

没有提示视角异常、摔倒异常。

偏离道路方向位于**沿道路行走模式**，  
默认不播报。



## 人行道模式

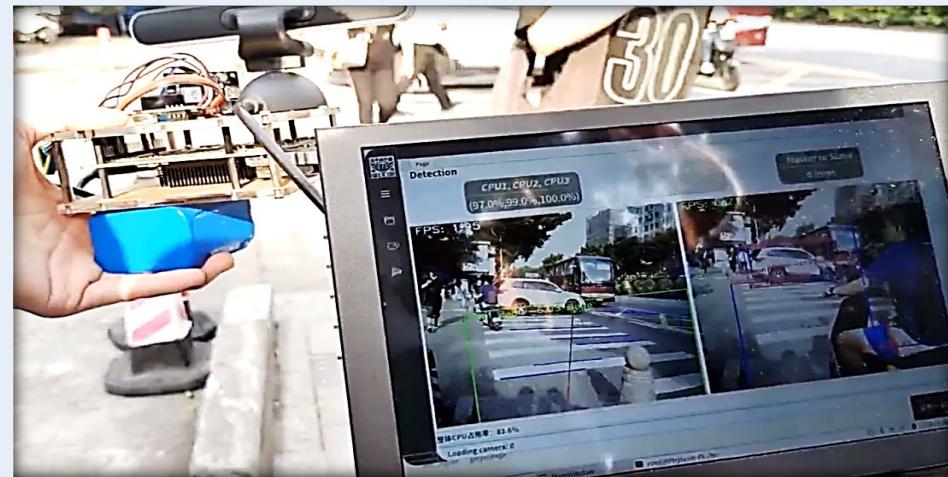
### 场景2：无交通灯支路路口场景

**播报内容：障碍物检测 + 斑马线检测** (由于出现支路路口斑马线)

- “**前方宽松，近距多障碍**”
- “**6, 8, 8, 0, 9**” (从近到远：斑马线, 短柱, 短柱, 小轿车, 绿化带)
- “识别到**1条斑马线，脚下斑马线在您右侧，指向右前方**”

没有提示视角异常、摔倒异常。

偏离道路方向位于**沿道路行走模式**，默认不播报。



**实测全程**

**性能状态**

**保持良好**

**CPU0, 1, 2占用率 (平均)** 97.0%, 99.0%, 100.0%

**单帧平均循环时间 (平均)** 0.103s

**核间通信响应时间 (平均)** 90us



# 6 运行效果：(1) 功能实测

## 路口模式 场景3: 路口边缘→安全岛

**播报内容:** 障碍物+斑马线+交通灯

- “前方宽松，近距多障碍”
- “6, 3, 1, 3, 18, 8”  
(从近到远: 斑马线, 摩托车(左前方私人车辆), 车轮, 摩托车(右前方外卖车辆), 行人, 短桩)
- “识别到1条斑马线, 脚下斑马线在您右侧, 指向左前方”
- “识别到3个交通灯, 最近距离交通灯为红灯”



没有提示视角异常、摔倒异常。偏离道路方向位于**路口模式**, 默认不播报。

## 路口模式 场景4: 安全岛→下一个路口

**播报内容:** 障碍物+斑马线+交通灯

**[高难度: 同时识别脚下,附近斑马线]**

- “前方拥挤, 远距多障碍”
- “6, 6, 1, 3, 13, 11” (从近到远: 斑马线, 斑马线, 车轮, 摩托车, 鞋子, 树木)
- “识别到2条斑马线, 脚下斑马线在您左侧, 指向左前方”
- “识别到4个交通灯, 最近距离交通灯为绿灯”



没有提示视角异常、摔倒异常。偏离道路方向位于**路口模式**, 默认不播报。

## 人行天桥模式 场景5: 上行楼梯

**播报内容:** 障碍物+阶梯

- “前方宽松, 近距多障碍”
- “10, 3, 13, 13, 18” (从近到远: 楼梯, 摩托车, 鞋子, 鞋子, 行人)
- “识别到1个阶梯, 在您右侧, 阶梯数21”



没有提示视角异常、摔倒异常。偏离道路方向位于**路口模式**, 默认不播报。

## 人行道模式 场景6: 下行楼梯

成功识别下行楼梯级数为14。



## 偏离道路方向功能

模拟视障者擅自穿越马路,  
播报警报提示。





# 6 运行效果：(2) 耗时实测

## 1. 目标检测量化耗时

**对比：**YOLOv8n ONNX Float16 目标检测模型，使用 **Int8** 量化后，执行时间**大幅缩短**；修改为 **MNN** 格式，仍保持 Float16 量化，达到**精度与速度的平衡**

模块 (单帧图像)	ONNX Float16	ONNX <b>Int8</b>	MNN Float16
目标检测 推理时间 (ms) (3×CPU)	1775	<b>375</b>	<b>768</b>
楼梯级数 识别时间 (ms)*	5.05e-2	5.25e-2	5.05e-2
红绿灯识别 时间 (ms)	4.17	4.24	4.22
斑马线识别 时间 (ms)	29.57	30.70	32.13

\*注：两者均使用改进后的楼梯级数识别算法

## 2. 楼梯级数识别算法耗时

项目	平均执行时间 (ms)
改进前: 楼梯走向+RGB直方图法	77.8
改进后: 线段k,b直方图波峰分析法	<b>5.05e-2</b>

**原因分析：** ( $x$  为目标框横轴像素数,  $y$  为纵轴,  $n$  为线段数)

改进前后共同: 提取线段:  $O(xy)$  (特殊数据结构耗时极短)

改进前独有: RGB直方图分析:  $O(4xy+3xy^2)$

改进后独有: k,b 直方图波峰分析:  $O(2n)$

对比斑马线识别算法(20ms): 滤除过于密集的斜线:  $O(n^2+2n)$

## 3. rpmsg 核间通信耗时

项目	平均执行时间 (us)
Master端向Slave端写入数据 (1×ept)	135
Master端从Slave端读取数据 (1×ept)	390
Master端向Slave端写入数据 (4×ept)	<b>136</b>
Master端从Slave端读取数据 (4×ept)	<b>140</b>

系统使用**4个ept并行**，与单ept下串行写入耗时一致，读取耗时大幅减小。



# 智慧之眼：视障人士的户外安全出行伴侣

## 2 大系统

1. Master系统: 高性能核  
视觉 + 算法 + GUI

2. Slave系统: 高实时核  
传感器 + 语音播报输出

## 4 大出行场景

1. 人行道: 提示避障
2. 路口前: 报告路况
3. 过马路: 安全通行
4. 过天桥: 级数预报

## 5 大功能模块

Master系统 应用模块:

1. 机器视觉处理模块
2. 系统状态与感知模块
3. 监护客户端显示模块

Slave系统 应用模块:

4. 传感器处理与语音反馈模块

底层模块:

5. 跨系统核间通信模块

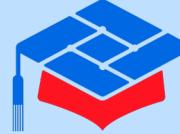
## 8 大关键技术

功能应用创新技术:

1. 楼梯级数识别算法迭代
2. 自主训练目标检测网络与轻量化处理
3. 单目距离角度解算 + 小地图
4. 原创判断视障人士沿路行走方法
5. 原创斑马线方向与交通灯识别方法

系统应用创新技术:

6. Master 端多进程并行框架
7. Master 端 Linux 内核核间通信驱动改进
8. Slave 端 rpmsg 非阻塞平台轮询实现



全国大学生集成电路创新创业大赛  
CHINA COLLEGE IC COMPETITION

# 感谢观看

第八届 全国大学生集成电路创新创业大赛  
飞腾杯

团队名称：飞腾风驰队

队伍编号：CICC1085

