基于 ELF 2 的低光条件下的目标检测系统

摘要

在低光环境的安防监控、智能交通和工业检测等场景中，光照不足会导致目标漏检。为解决这一难题，本文基于瑞芯微 RK3588 芯片的飞凌 ELF 2 嵌入式平台开发了一套双模式低光目标检测系统。系统集成了图像采集、低光图像增强、自适应图像融合和 YOLOv5s[1]目标检测功能，提供实时视频流检测和高精度静态图像检测两种模式。实时模式采用 CPU 执行的传统增强算法[2]结合自研轻量级 ImageFuse 融合模块和 YOLOv5s 检测模型，形成五级异构流水线并行处理，大幅提升吞吐量和实时性能，实现吞吐率约为 30.84fps。其中 ImageFuse 模块采用双层卷积网络对原始与增强图像进行像素级权重融合，有效抑制过曝伪影并恢复细节。静态模式则将低光增强模型 SCI(Self-CalibratedIllumination)[3]、YOLOv5s 串联成端到端 SY 模型独立部署于一个 NPU 上，实现高质量的离线检测分析。

第一部分 作品概述

1.1 功能与特性

本系统部署在飞凌 ELF 2 嵌入式平台，针对低光条件下目标检测精度低、漏检率高的问题，设计了双模式检测架构，实现以下核心功能：

（1）实时视频流检测：采用高效的传统图像增强算法组合（双边滤波降噪、CLAHE 对比度受限直方图均衡和 Gamma 亮度校正），由 CPU 处理得到增强图像。然后，将增强后图像与原始低光图像一并输入部署在 NPU Core 1 上的 ImageFuse 自适应融合模块，进行像素级的图像融合以抑制过度曝光。融合结果再送入部署在 NPU Core 0 上的 YOLOv5s 目标检测模型，实时产出目标识别结果。能够在平均总延时 135.89ms 的情况下实现约 30.84fps 的处理吞吐率。用户可通过可视化界面实时监控原始画面、融合增强画面、检测结果及关键性能指标

（2）静态图像增强检测：针对单帧图像或离线视频，采用端到端的高精度 SY 模型进行处理。SY 模型将低光增强（使用 Self-CalibratedIllumination，SCI）、YOLOv5s 检测两个子模型串联整合，在 NPU Core 2 上独占运行。该模式以质量

优先，一次性完成图像增强和目标识别的串行推理，输出增强且标注了检测结果的图像。静态模式适合作为服务器端离线分析使用，提供更高的图像亮度质量和检测精度，适用于对实时性要求不高但追求结果准确性的场景。

系统的核心特性体现为四大优势：

（1）实时性：多线程流水线与 NPU 多核绑定设计保障低延迟处理；

（2）鲁棒性：融合机制抑制过增强伪影，提升弱光场景稳定性；

（3）嵌入式友好性：轻量化模型与功耗优化适配边缘设备长期运行；

（4）可扩展性：模块化架构支持算法升级扩展。

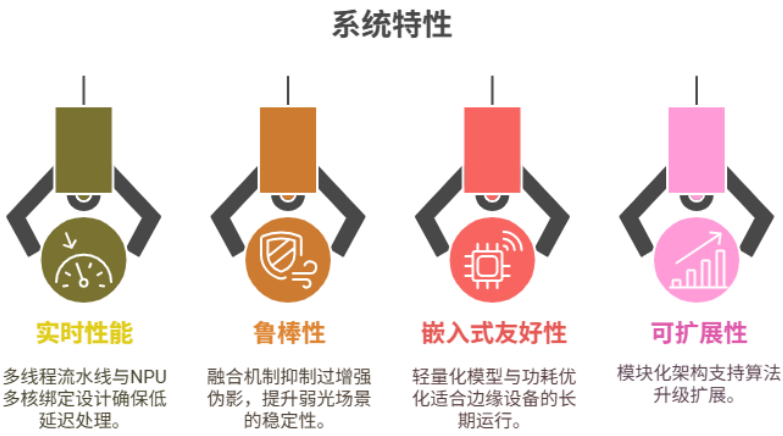

图 1：系统特性

1.2 应用领域

本项目具备广泛的应用价值，尤其适用于低光或夜间环境中的目标识别任务。通过图像增强、自适应融合和目标检测等多模块协同机制，系统在低光条件下依然能够输出稳定、准确的检测结果。主要应用领域包括：

（1）智能安防监控：适用于无照明园区周界监控、夜间仓库巡检等场景。系统增强人车目标的检出能力，降低低光条件下漏检与误报风险，提升夜间安防可靠性；

（2）智能交通管理：可部署于隧道进出口、低照度路段等交通场景，增强车辆与行人目标检测的稳定性，辅助事故预警；

（3）无人系统感知：支持仓储机器人导航、地下管廊巡检等低光作业，强化环境中障碍物的识别可靠性，保障无人设备在弱光环境下的运行安全；

（4）工业视觉质检领域，在工厂车间等光照不足环境下，克服视觉盲区，实现对物料缺陷的自动化检测与异常监测。通过低光图像增强与自适应融合，系统为边缘端工业质检提供可靠的视觉解决方案。

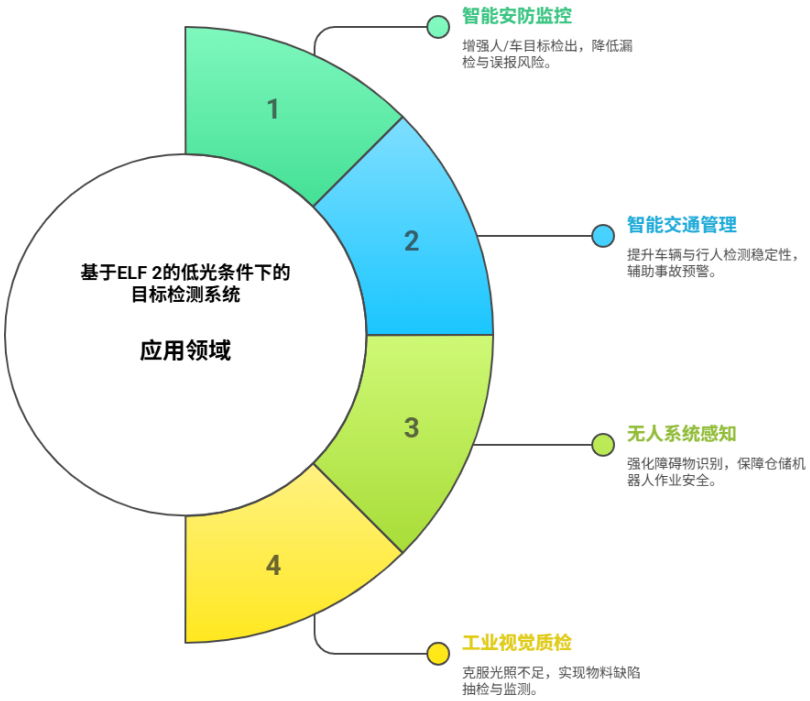

图 2：应用领域

1.3 主要技术特点

（1）多核 NPU 协同异构推理机制。系统充分发挥瑞芯微 RK3588 平台内置的多核 NPU 算力，设计了一种异构并行推理架构。分别将 YOLOv5s 检测模型、ImageFuse 融合模块以及端到端的 SY 模型独占绑定到不同 NPU 核心，传统增强算法则部署在 CPU 上并行执行。这种资源隔离方案避免了多任务间的资源竞争问题，确保各任务间的互不干扰，显著降低系统端到端处理延迟，提高整体推理吞吐率。

（2）实时与静态双模式检测架构。系统针对不同应用场景设计了两种独立但互补的检测模式：实时模式采用五级异构流水线进行多线程并行推理，以最大化吞吐量，保障对实时视频流的处理性能；静态模式采用端到端 SY 模型进行串

行推理，确保每帧图像在精度上的最优表现。这种架构实现了“一套硬件平台，支持两种应用场景”，在灵活适配不同任务需求方面具有明显优势。

（3）五级流水线并行调度架构。在实时模式中，通过细致划分图像增强、图像融合、目标检测、结果标注、结果显示五个任务阶段，并采用独立线程和缓冲队列进行任务解耦，形成流水线并行处理结构。流水线架构有效消除了各任务阶段的阻塞瓶颈，使系统整体吞吐率显著提高，达到近实时的视频检测性能（平均 30.84fps 左右）。

1.4 主要性能指标

基本参数	性能指标	优势
系统吞吐率	实时模式下平均值为 30.84fps	满足实时性需求
mAP（meanAveragePrecision）	实时模式：0.533 静态模式：0.505	相较原始低光输入精度提升
资源利用率	三核 NPU，CPU	充分发挥硬件计算并行能力
供电需求	12V	使用移动电源方便携带
是否有显示屏	是	实时显示结果

1.5 主要创新点

本系统的主要创新点侧重于模型设计和算法细节优化方面：

（1）自研 ImageFuse 自适应图像融合模块。针对传统图像增强算法存在的过曝现象，团队自主研发了一个轻量级卷积神经网络模块 ImageFuse。该模块采用双层卷积结构，自适应地从低光图像和增强图像中提取融合权重，逐像素加权融合。经过联合训练优化，融合结果在视觉质量上明显优于单纯的传统增强方法，有效抑制过曝伪影并恢复细节，极大提升了低光条件下的目标检测性能（融合后实时模式下 mAP@0.5 提升至 0.533）。

（2）轻量化算法与数据增强策略。为进一步提升算法在嵌入式平台上的部署性能，团队采取了多项模型优化措施：针对传统 SCI 深度增强模型延迟高的问题，实时模式中使用了传统增强算法取代 SCI；并通过对 YOLOv5s 检测模型进行 RKNN[4]量化，显著降低了单帧推理耗时。同时，为了提高模型的鲁棒性，构建

了特殊的混合训练数据集（Gamma 变换的低光数据与原始正常数据组合），增强了模型对不同光照条件的泛化能力。

（3）软硬件协同的边缘端闭环系统。本项目不仅实现了核心算法和模型优化，还针对嵌入式设备部署要求开发了完整的软硬件协同方案，包含从摄像头数据采集到界面实时展示的全流程闭环架构。系统集成了高效的界面显示（PyQt 本地 GUI 和 HTML 网页远程界面），实现实时性能监控与直观交互，展示了嵌入式低光视觉系统从算法设计到现场部署的全链条创新实践。

1.6 设计流程

本项目的设计流程如图 3 所示。

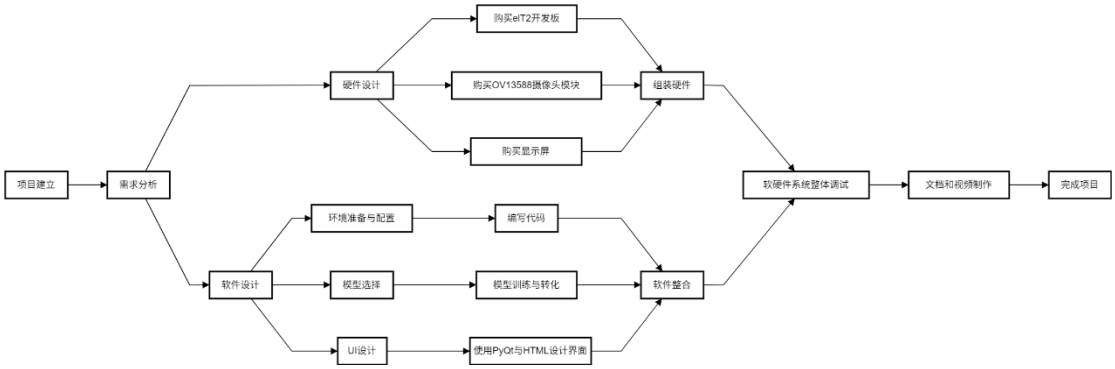

图 3：项目设计流程图

第二部分 系统组成及功能说明

2.1 整体介绍

本作品包含输入设备、开发板处理系统、显示屏界面显示，整体架构图如图 4 所示。其中，输入设备为摄像头或用户终端，开发板负责数据处理，显示终端用于结果呈现。系统部署在飞凌 ELF 2 嵌入式开发板，构建了一个适用于低光环境下的图像增强与目标检测平台。根据所采用的低光增强算法不同，系统实现了两种工作模式：一是面向视频流的实时模式，二是以开发板为服务器的静态模式。

实时模式突出高吞吐的并行处理能力，采用五级异构流水线架构让各处理阶段重叠执行，实现从图像获取到结果显示的流水线并行。相较而言，静态模式在接收到请求时以串行顺序执行完整的端到端模型推理流程，保证每次处理的最优精度。前者适用于对实时性要求高的场景，如视频监控即时报警，后者适用于对精度要求高的场景，如离线图像分析与取证。实时模式利用多线程和

多核 NPU 同时运行多个任务，资源占用上更强调并行调度；静态模式则采用单模型独占一个 NPU 核心运行，请求间隔期间其余计算资源空闲，以避免干扰实时任务的执行。两种模式通过合理的资源分配互不影响，支持系统同时提供实时监控和静态分析服务。系统的软件架构采用统一的多线程调度机制，将图像增强、融合、目标检测、标注、结果显示等任务分别分配到不同线程，充分利用 CPU 多核和 NPU 多核的并行能力，提高数据处理吞吐量和响应速度。借助上述框架，整个系统实现了从低光视频流实时监控到离线图像高精度分析的全场景覆盖，为嵌入式低光视觉应用提供了标准化解决方案。

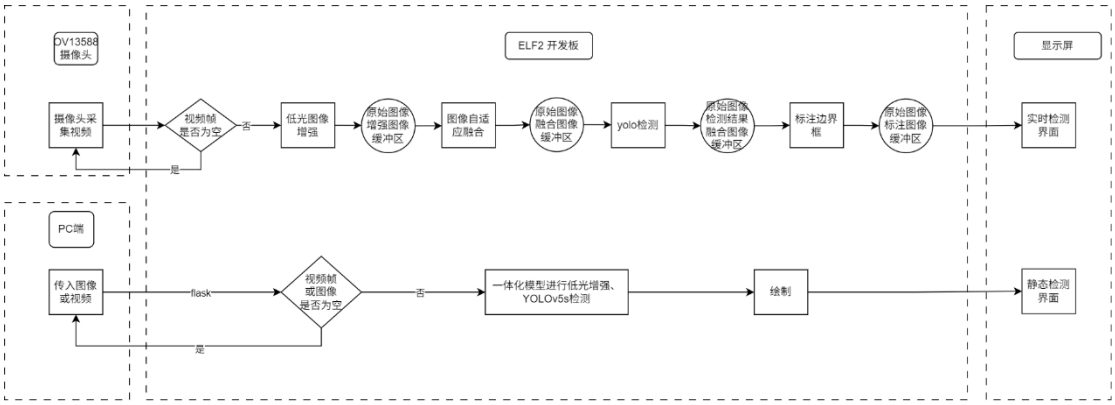

图 4：整体框架图

2.2 硬件系统介绍

2.2.1 硬件整体介绍；

从整体架构来看，本作品的硬件系统由以下三个核心模块构成：视频采集模块、视频帧处理模块和视频显示模块。三者协同工作，实现视频数据的获取、处理和实时呈现，构建了一个闭环的图像检测系统，硬件结构如图 5 所示。

（1）视频采集模块：本模块通过 GStreamer 视频管道，使用 v4l2src 元素从 OV13588 摄像头中采集实时图像数据，可根据实际摄像头配置多种分辨率与格式。

（2）视频帧处理模块：负责对采集的图像帧进行增强、融合和目标检测等多阶段处理。该模块运行于 ELF 2 开发板，依托其内置的多核 NPU 和 CPU 资源执行各算法，包括图像预处理增强、ImageFuse 模型融合和 YOLOv5s 检测等功能。视频帧处理模块是系统的核心算力单元，直接决定了检测的实时性和准确性。

（3）视频显示模块：负责将处理结果进行输出展示。在本项目中，通过 Type-

C 接口连接的显示屏可实时显示摄像头采集的原始画面、增强融合后的图像以及标注了检测框的结果画面。同时，显示模块还在 GUI 界面上动态更新系统处理帧率、平均延迟等性能参数；对于静态模式的结果，系统通过网页界面返回处理后的图像或视频，方便用户查看。

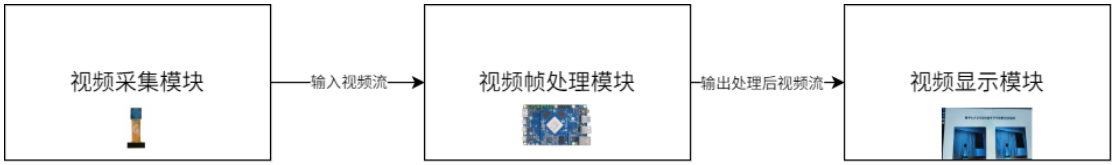

图 5：硬件结构图

2.3 软件系统介绍

2.3.1 软件整体介绍

本软件系统运行于飞凌 ELF 2 开发板之上，由数据采集、模型推理、结果显示等模块组成，其结构如图 6、图 7、图 8 所示。各软件模块协同执行，实现对硬件功能的充分调度：数据采集模块负责从摄像头或网络接口获取图像数据，模型推理模块执行图像增强、融合及目标检测算法，结果显示模块负责将处理后的结果通过本地 GUI 或网页界面反馈给用户。软件架构中同时包含实时和静态模式两种服务管线（前者采用分阶段多线程流水线，后者采用单模型串行推理）。这种架构保证了系统可以同时支持实时视频流处理和离线批量图像处理，覆盖从前端监控到后端分析的全场景需求。

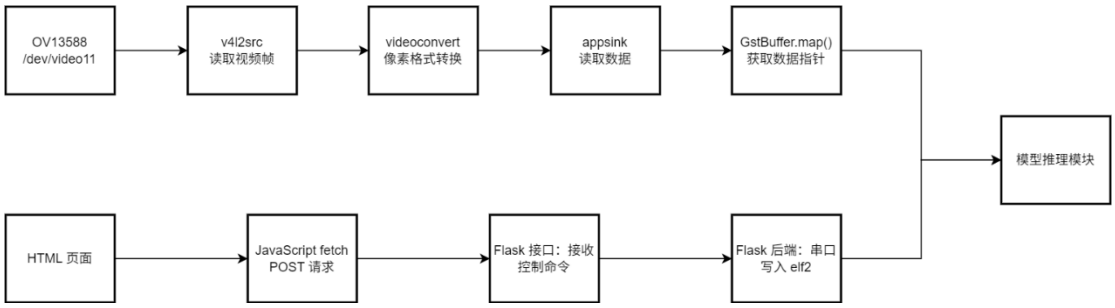

图 6：数据采集模块

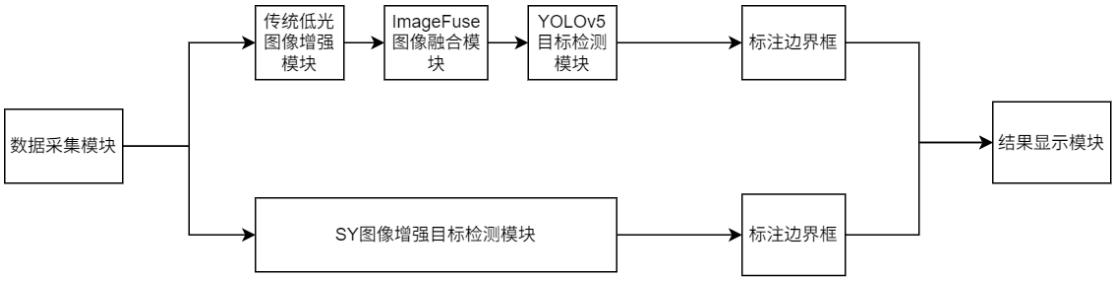

图 7：模型推理模块

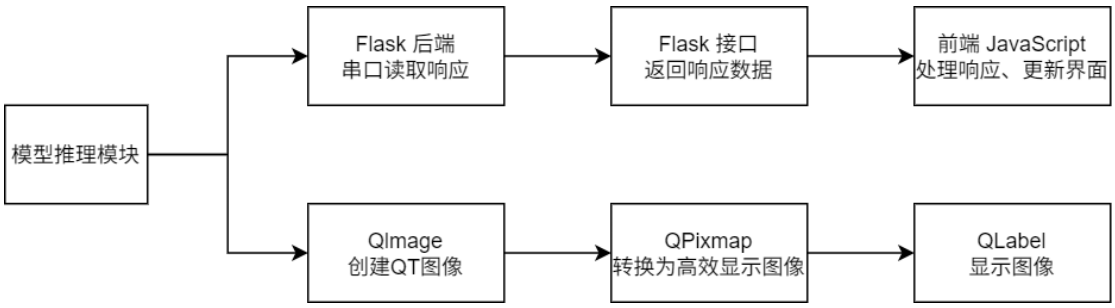

图 8：结果显示模块

2.3.2 软件各模块介绍

（1）模型推理模块

该模块承担图像增强和目标检测的核心推理任务。设计目标是实现端到端的实时推理能力，因此对模型的选择和优化非常关键。最初，本项目尝试使用深度学习的低光增强模型 SCI（Self-CalibratedIllumination）来提高暗光图像亮度，但实测单帧处理延时超过 70ms，无法满足实时性要求。经比较，对于实时模式，最终采用了传统图像增强算法组合（双边滤波+CLAHE+Gamma 校正）替代 SCI 执行实时增强，其平时处理延时降至 30ms 以内。目标检测方面，选用了轻量化的 YOLOv5s 网络，并通过 RKNN 工具将其量化为 RK3588 平台可加速的模型格式。量化优化后，YOLOv5s 在 NPU 上推理单帧约 30ms。针对直接增强可能导致正常光照区域过曝的问题，本项目自主研发了一个轻量级卷积神经网络（ImageFuse）用于提取图像融合权重，并在此基础上构建了图像自适应融合模块。ImageFuse 由双卷积层组成。其首先计算从低光原始图和增强后图像提取像素级的权重系数；随后按照各自权重对原始图和增强图进行逐像素加权相加，生成融合图像。该融合机制能够有效压制增强过程中过亮的区域，并在一定程度上恢复传统增强丢失的细节信息。为提高融合后的图像对检测模型的适配性，将 ImageFuse 模型与 YOLOv5s 模型进行了联合训练：冻结 YOLOv5s 的参数，仅训练 ImageFuse 的权重，使二者形成一个连续优化的组合模型。

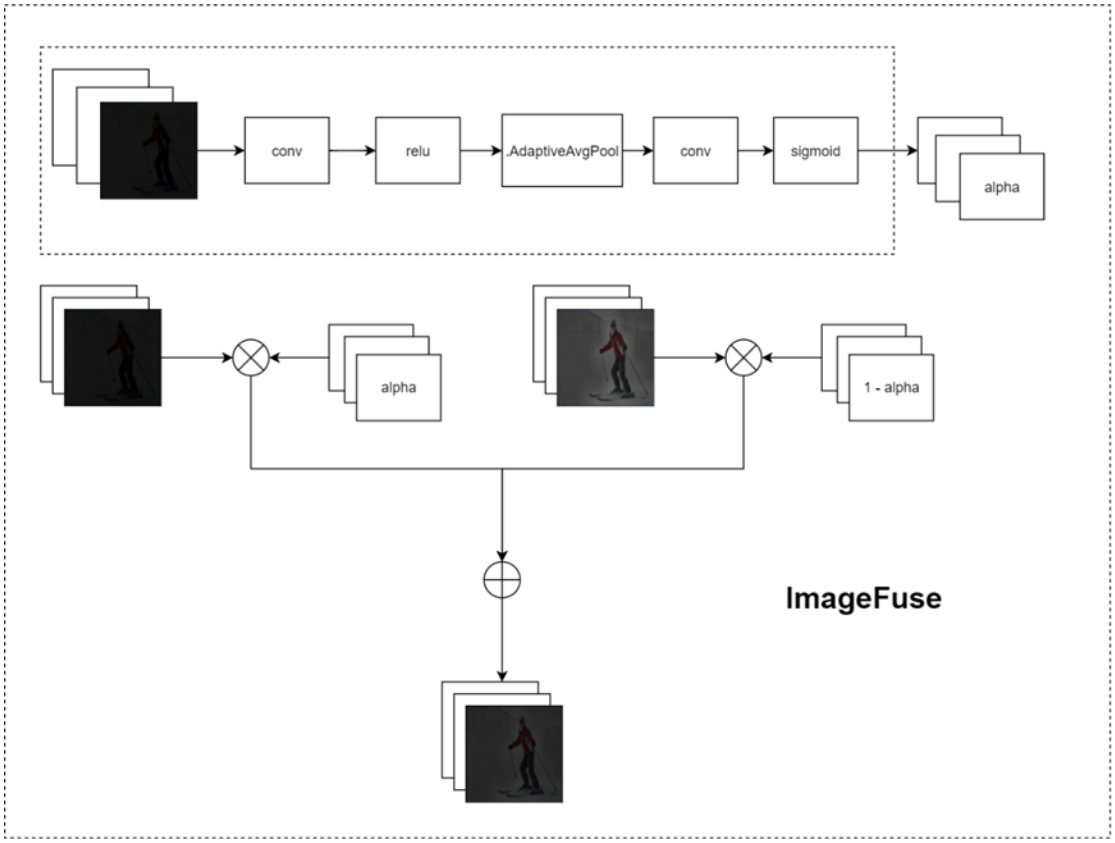

图 9：图像自适应融合模块结构图

考虑到该模块含有多个模型单纯的串行时延较长不符合实时显示要求，于是对于实时模式，我们利用 ELF 2 开发板多 NPU 的特性，将 YOLOv5s 部署于 NPU Core 0，ImageFuse 模型部署于 NPU Core 1，传统图像增强算法部署于 CPU 上执行。同时利用多线程机制和缓冲池将整个流程分为五段流水线分别是图像增强、自适应融合、目标检测、图像标注、实时显示。

因为传统图像增强算法在低光环境下对于图像细节处理效果没有神经网络模型好，于是我们决定将传统图像增强方法替换为 SCI 模型提供静态检测服务。考虑到 NPU 的空闲数量仅剩一个同时，我们将 SCI、YOLOv5s 模型拼接为 SY 模型并转换为 RKNN 模型单独部署在 NPU Core 2 上。静态模式下 SY 模型的总推理延迟相对较高，但运行在独立 NPU 核心上且只在有用户请求时启动，不影响实时检测服务的持续运行。

（2）数据采集模块

该模块负责获取系统所需的输入图像数据流。

在实时模式下，数据来源为本地摄像头。系统通过 GStreamer 框架调用

v4l2src 元素从 OV13588 摄像头实时读取原始视频帧，并使用 videoconvert 元件将图像格式转码为统一的处理格式后输出至应用接收端（appsink）。利用底层提供 GstBuffer.map()接口，可高效地将摄像头帧数据映射到内存缓冲供后续处理，系统支持 YUY2、NV12 等多种像素格式的输入。

在静态模式下，数据采集模块通过网络接口接收用户提交的离线图像或视频。具体实现上，在 ELF 2 开发板上运行了基于 Flask 框架的轻量级服务端，PC 端用户可通过网页上传图像或视频文件，Flask 后端接收到上传数据后写入开发板存储或内存，触发静态检测流程。该模块对实时和静态两种来源的数据进行统一封装，使后续处理流程可以透明地获取输入帧，无论其来自摄像头还是网络上传。

（3）界面控制模块

该模块负责系统的人机交互界面和结果展示。

在实时模式下，采用 PyQt5 构建本地 GUI 界面，提供直观的监控视图。GUI 包含两个并行的视频窗口：一个显示摄像头捕获的原始低光画面，另一个显示经过增强融合和目标识别后的结果画面。界面上方实时刷新当前处理帧率（FPS）和系统平均延迟（Latency）等指标，方便用户观察系统性能。当用户关闭界面时，程序会自动释放摄像头和模型占用的资源，确保系统能够稳定运行和资源及时回收。

对于静态检测结果展示，界面控制模块通过 PC 的浏览页面与 Flask 服务进行交互。我们使用 HTML 语言搭建的显示界面实现人机交互，用户可以选择传输图片或视频，在开发板处理完成后，显示界面会接收返回数据并刷新界面，供用户查看处理结果。

第三部分 完成情况及性能参数

3.1 整体介绍

本作品的正面视图如图 10 所示，斜 45°视图如图 11 所示，全局图如图 12 所示。

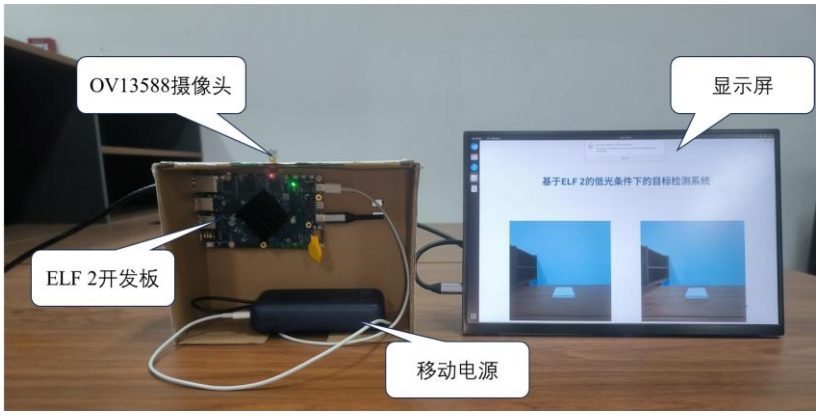

图 10：作品正面视图

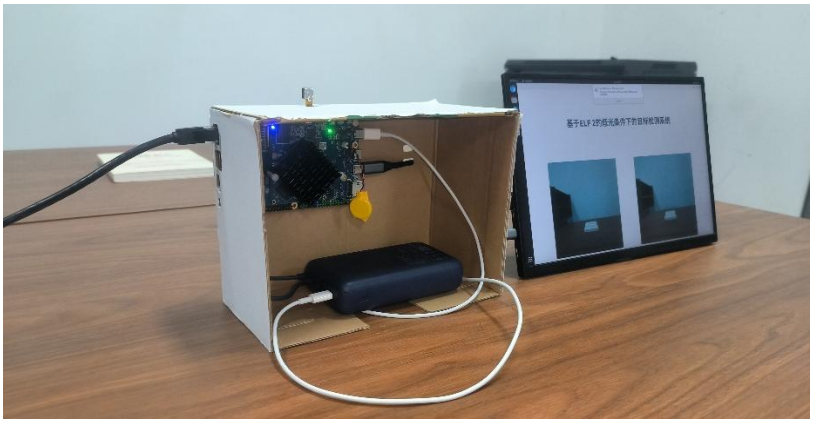

图 11：作品斜 45°视图

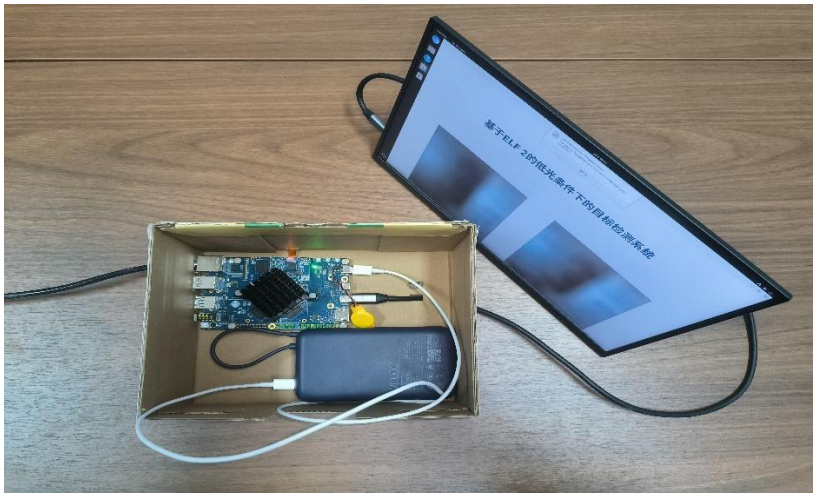

图 12：作品全局图

3.2 工程成果

3.2.1 机械成果。本作品的机械成果可从图 10 中看到。

3.2.2 电路成果。本作品无电路成果。

3.2.3 软件成果。本作品实时模式的下界面如图 13 所示，本作品静态模式下

的界面如图 14 所示。图 13 与图 14 的两个显示框中左边都是原始的视频或图像，右边都是处理后的视频或图像。

图 13：实时模式下的界面

图 14：静态模式下的界面

3.3 特性成果

在本节中，将对实时模式和静态模式采用的模型的分类性能进行评估，并且还将对实时模式下的时间性能进行分析。

3.3.1 分类性能评估

为了评估不同处理流程对目标检测性能的影响，我们定义如表 1 所示的以下模型组合进行对比实验。

表 1：模型定义表

模型	模型组成	处理流程
M1	YOLOv5s	直接将原始图像输入 YOLOv5s 检测模型
M2	传统图像增强算法（CPU）+YOLOv5s	使用传统图像增强算法处理后输入 YOLOv5s 检测模型
M3(实时模式模型）	传统图像增强算法（CPU）+ImageFuse+YOLOv5s	传统增强算法处理+ImageFuse 模块融合增强图与原始图后输入 YOLOv5s 检测模型
M4(静态模式模型）	SY 神经网络模型	使用 SCI 神经网络模型出力后输入 YOLOv5s 检测模型

在模型训练阶段，我们使用 COCO128 数据集构建训练集：首先使用原始 COCO128 图像作为正常光照样本，然后对其应用 Gamma 校正生成对应的低光图像样本（伪低光），最后合并这两部分数据形成最终的训练集。该训练集包含多种光照条件，用于训练模型适应不同光照环境的能力。

在模型验证阶段，我们基于 val2017 作为验证集：从 val2017 中选取 1000 张正常光照图像作为原始数据集，并对其应用相同的 Gamma 校正生成 1000 张低光数据集（伪低光）。随后将原始数据集（1000 张）与低光数据集（1000 张）合并，形成 2000 张图像的混合数据集。**错误!未找到引用源。**中的所有检测结果均在此独立验证集上测试得到，确保评估结果与训练数据完全分离。

表 2：验证集检测结果

数据集	模型	Images	Labels	P	R	mAP@0.5	mAP@0.5:0.95
混合数据集	M1	2000	14688	0.64	0.448	0.498	0.322
	M2	2000	14688	0.644	0.473	0.522	0.377
	M3（实时）	2000	14688	0.647	0.487	0.533	0.345
低光数据集（伪）	M1	1000	7344	0.588	0.379	0.427	0.271
	M2	1000	7344	0.615	0.431	0.481	0.310
	M3（实时）	1000	7344	0.652	0.43	0.497	0.381
	M4（静态）	1000	7344	0.634	0.455	0.505	0.324

表 3：参数指标

指标	表示	偏重方面
P	精确率	少误报（FP）
R	召回率	少漏检（FN）
mAP@0.5	平均精度@0.5IoU	预测是否大致正确
mAP@0.5:0.95	平均精度（10 个 IoU）	预测是否精确重合

3.3.2 实时模式时间性能分析

针对实时模式，我们提取了 1000 帧的各个模块的时延和总时延，并统计了它们的最大值、最小值以及平均值，以及平均吞吐率，如表 4 所示。

表 4：实时模式模块时间开销

时延 \ 数值	平均值(ms)	最小值(ms)	最大值(ms)
图像增强模块时延	16.77	12.17	50.36
图像融合模块时延	38.97	24.52	66.18
目标检测模块时延	33.68	20.77	74.00
图像标注模块时延	45.02	27.75	73.41
实时模式 1000 帧的平均总时延:135.89ms			
实时模式 1000 帧的平均系统吞吐率:30.84fps			

第四部分 总结

4.1 可扩展之处

本系统基于模块化架构设计，具备良好的可扩展性与优化潜力，后续可在以下方向如图 15 所示进行深化与拓展：

（1）通信协议扩展：静态模式下当前使用 FlaskHTTP 接口，未来可拓展 MQTT、WebSocket 等协议，增强远程监控与云端对接能力。

（2）多模态感知融合：为提升在极端低光或恶劣天气条件下的感知鲁棒性，可以考虑融合其他传感器数据。例如，将红外热成像、微光夜视仪甚至毫米波雷达的感知结果与本系统的视觉检测结果结合，构建多模态的信息融合检测系统，从而获得在全黑环境或大雾雨雪天气下依然可靠的目标检测能力。

（3）检测目标与功能扩展：基于 YOLOv5s 框架，可训练集成特定场景（如工业

缺陷分类、交通行为识别、安防入侵检测）定制模型，拓宽应用场景。

（4）模型轻量化与全嵌入式部署：探索先进压缩、量化或蒸馏技术，对高精度但计算开销大的模型进行优化，将其精简至嵌入式平台可承载的范围。

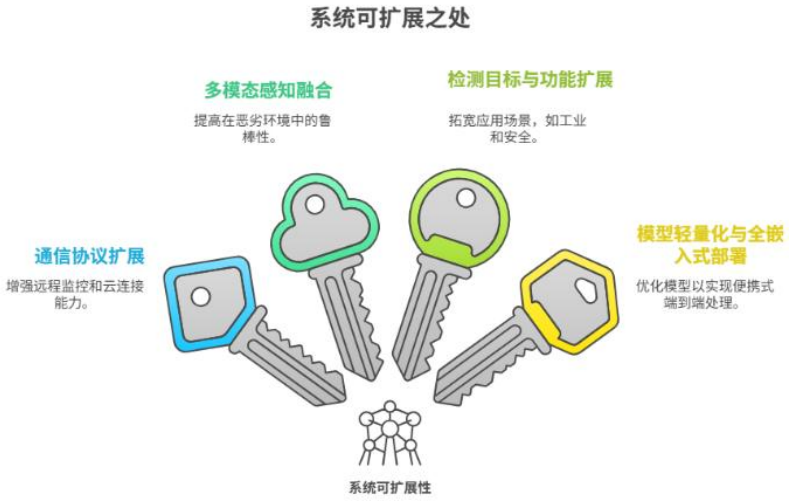

图 15:系统可拓展图

4.2 心得体会

本项目从解决低光环境目标检测的核心痛点出发，成功在飞凌 ELF 2 嵌入式平台上构建了一套低光图像增强、自适应融合与目标检测的双模式系统。整个研发过程深刻印证了“软硬协同优化”在嵌入式 AI 系统中的关键作用。

在技术方案上，我们放弃了单一模型处理低光图像的思路，创新性地设计了“实时+静态”双模式。实时模式通过 CPU 执行高效的传统图像增强算法（双边滤波+CLAHE+Gamma 校正）与 NPU 上部署的自研 ImageFuse 融合模块（NPU Core 1）及 YOLOv5s 检测模型（NPU Core 0）协同工作，确保了低延迟（平均总延迟 135.89ms，吞吐率 30.84fps）。静态模式则通过整合端到端的 SY 模型（SCI+ImageFuse+YOLOv5s）独占部署于 NPU Core 2，提供高精度离线分析能力。异构多核 NPU 协同推理架构是解决资源竞争、实现真正并行处理的核心突破点，经千帧实测验证显著降低了系统延迟。

在模型设计与优化上，针对传统增强易过曝、神经网络增强延迟高的问题，我们创新性地设计了轻量级的双层卷积 ImageFuse 模块，通过自适应权重融合原始图与增强图，有效抑制了过曝伪影并部分恢复了丢失的细节（实验数据表明融

合后，实时模式 mAP@0.5 由 0.498 提升至 0.53；静态模式 mAP@0.5 0.505 相对于相同情况下使用传统的增强算法 mAP@0.5 0.481 有所提高）。模型选型经历了实际验证：SY 模型中图像增强部分的 SCI 虽效果良好但延迟过高被用于静态模式；YOLOv5s 经 RKNN 量化优化后延迟降至 30ms 内，满足实时要求。

在系统工程实现上，对于实时模式采用五级异构流水线和基于队列的异步多线程调度机制，有效解耦任务，结合缓冲区管理，大幅提升了系统的吞吐率、响应速度和稳定性。

利用 PyQt5 和 Flask 分别构建了直观的实时监控 GUI 和静态模式下的 Web 界面，便于用户交互与结果观察。

调试过程中，诸如多模型 NPU 资源竞争、图像格式转换、流水线同步等问题带来了挑战，通过日志分析、逐模块验证和细致的资源管理（如模型绑定、线程安全队列、退出资源释放）得以逐一攻克。这使我们深刻认识到嵌入式 AI 系统开发中，算法效果、计算效率与资源管理必须紧密结合，细节决定成败。

综上所述，本项目不仅验证了在资源受限的嵌入式平台上实现高效低光目标检测的可行性，更通过双模式架构、多核 NPU 协同、自适应融合和流水线调度等创新设计，为边缘端智能视觉应用提供了一套实用性强、扩展性好的解决方案，在安防、交通、工业等领域具有广阔的应用前景。未来将持续优化模型性能与系统效率，探索更广泛的应用场景。

第五部分 参考文献

[1] https://github.com/ultralytics/yolov5

[2] https://blog.csdn.net/HanWenKing/article/details/141172106

[3] Ma,Long,etal."Towardfast,flexible,androbustlow-lightimageenhancement."*ProceedingsoftheIEEE/CVFconferenceoncomputervisionandpatternrecognition*.2022.

[4] 《Rockchip_User_Guide_RKNN_API_V1.2.0_CN》

[5] 《ELF 2 开发板快速启动手册》

[6] 《基于 RK3588 的 AI 模型训练到部署》

[7] https://github.com/airockchip/rknn_model_zoo