Application Note

利用 RGB-IR 摄像头构建驾驶员与乘员监控系统

Jianzhong Xu and Reese Grimsley

摘要

本应用手册详细阐述了利用 AM62A 平台上的 RGB-IR 传感器，实施视觉流水线的情况。本文档介绍了一种采用 OX05B1S 传感器与 AM62A SK EVM 并支持视频电话功能的驾驶员与乘员监控系统的参考设计。

内容

插图清单

表格清单

商标

1 简介

随着对可见光与红外光中图像传感需求的不断增长，利用单个摄像头同时捕捉 RGB 与 IR 图像的 RGB-IR 传感器变得越来越受欢迎。对于驾驶员监控与乘员监控（又称“座舱监控”）、机器人科学、安全监控以及智能家居系统等人工智能应用来说，有效处理与利用 RGB-IR 数据具有重要意义。AM62A SoC 是构建该等智能系统的理想选择，详情参阅 [1]。

本应用手册的重点是在 AM62A SoC 上实现从 RGB-IR 摄像头到人工智能引擎的视觉处理流水线，用于支持视频通话或视频录制的驾驶员与乘员监控系统（DMS 或 OMS）。本文档首先概述了基本的硬件与软件组件，然后介绍了参考设计与基准测试。

2 构建 RGB-IR 视觉流水线模块

如 [1] 所示，AM62A 具备了一系列独特功能，也正因此，该器件成为构建需要 RGB 与 IR 图像数据的应用的绝佳选择。以下各部分详细介绍了构建 RGB-IR 图像处理流水线的相关元件。

2.1 CSI 接收器

AM62A 搭载的摄像头串行接口 (CSI) 接收器 (Rx) 符合 MIPI CSI v1.2 标准，最多支持 16 个虚拟通道。该接收器包含一个 DMA 包装器，该包装器可通过 DMA 将采集的图像数据传输到存储器。该包装器可创建多个 DMA 上下文，每个 DMA 上下文专用于将数据从一个虚拟通道存储到存储器中。

用于 AM62A 的处理器 SDK Linux (或 EdgeAI SDK) 中的 CSI Rx 驱动程序 [2] 符合 V4L2 框架。该驱动程序能够为每个 DMA 上下文创建一个 V4L2 视频设备节点，用于从每个虚拟通道采集图像数据，然后存储到存储器中。然后，用户空间应用程序可通过与每个虚拟通道关联的视频节点访问采集的图像数据。

RGB-IR 传感器通常使用包含彩色与红外像素的 4x4 马赛克图案。外部 IR 照明器可用于提供充足的 IR 照明，可以定期开启与关闭，也正因此，传感器能够采集彩色 (RGB) 为主的图像或 IR 为主的图像。传感器可以与照明器同步。在照明器关闭时，通过一个虚拟通道发送 RGB 为主的图像。在照明器打开时，通过另一个通道发送 IR 为主的图像。为了单独处理该等图像，CSI Rx 驱动程序创建了两个视频设备节点：一个用于接收 RGB 数据（当 IR 照明器关闭时），另一个用于接收 IR 数据（当 IR 照明器打开时）。

2.2 图像信号处理器

AM62A 器件搭载的图像信号处理器 (ISP) 又称“视觉预处理加速器 (VPAC)”，能够提供像素级的基本视觉处理功能。ISP 由视觉成像子系统 (VISS)、多标量 (MSC) 与镜头畸变校正 (LDC) 三个子模块共同组成。

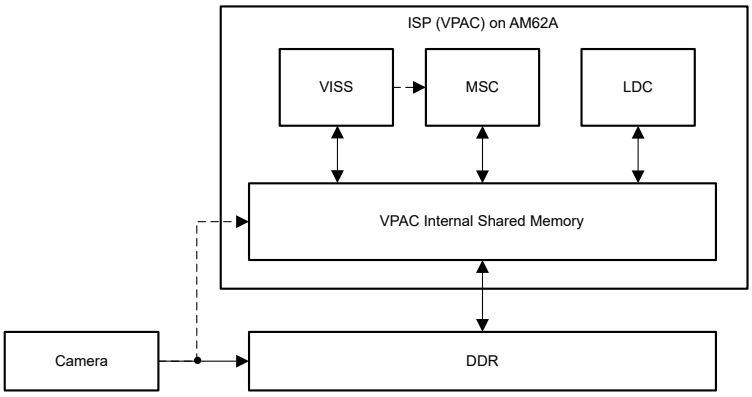

图 2-1. AM62A 图像信号处理器概述

VISS 子模块能够处理原始图像，生成去马赛克彩色图像与 IR 图像。VISS 有时也称 ISP。MSC 子模块最多可生成 10 幅缩放或裁剪图像。LDC 子模块能够进行透视与几何变换，主要用于校正镜头畸变。

AM62A ISP (VPAC) 的子模块能够在内存到内存模式或动态模式下运行：

- **内存到内存模式**：子模块能够从存储器读取数据并将处理后的数据再写回内存。
- **动态模式**：摄像头数据会直接发送到视觉成像子系统 (VISS)，VISS 输出会直接发送到多标量 (MSC)，而不会先存储至存储器。

VISS 能够以 4x4 RGB-IR 模式处理原始图像数据，并且产生两个输出流：一个 RGB 流与一个 IR 流，如 图 2-2 所示。

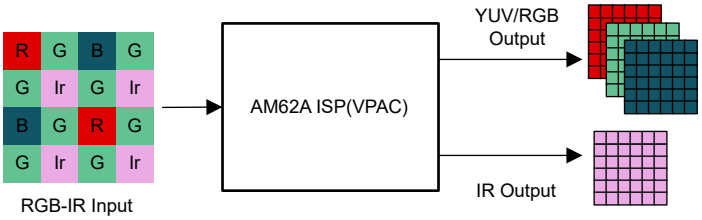

图 2-2. 通过 AM62A ISP 进行 RGB-IR 处理

对于 4x4 图像数据，要先进行原始像素处理（例如：解压缩、WDR 合并、缺陷像素校正以及镜头阴影校正）。进行原始像素处理以后，可将图像数据分为 RGB 处理路径与 IR 处理路径两条路径。

- **RGB 处理路径**：该路径包括典型的颜色处理功能（例如：去马赛克、自动白平衡、色彩校正以及噪声滤波）。此外，还会从 RGB 数据中去除 IR 污染。
- **IR 处理路径**：该路径较为简单，仅涉及上采样与色调映射。

对于驾驶员监控等只需要 IR 图像的应用，可使用 IR 输出。对于需要彩色图像的视频通话或录制等应用，可使用 ISP 的 YUV 或 RGB 输出。此外，乘员（或座舱）监控还可以利用 RGB 输出以提高检测精度。

2.3 视频处理单元

AM62A SoC 包含一个适用于编码与解码的硬件加速器，称为视频处理单元 (VPU)，可支持 H.264 与 H.265 编码与解码。VPU 可以对 ISP 生成的 RGB 流封包，用于视频通话或录制。如需了解 VPU 应用与性能的更多详细信息，可参阅 [3]。

2.4 德州仪器 (TI) 深度学习加速技术

深度学习与神经网络正日益成为从图像与其他数据中提取含义与信息的热门策略。德州仪器 (TI) 的 AM6xA 与 TDA4x SoC 采用内部开发的硬件 IP——C7xMMA，并借助德州仪器深度学习 (TIDL) 软件，加速神经网络推理。

C7xMMA 是一款紧密耦合的 C7x SIMD DSP 与矩阵乘法器加速器 (MMA)。卷积神经网络 (CNN) 是一种用于视觉处理的常见神经网络类型。对于该等神经网络，该架构非常有效。在大多数 CNN 中，矩阵乘法与类似运算至少占到总运算的 98%。因此，MMA 对适用于视觉任务（例如：物体检测、像素级分割以及关键点检测）的神经网络加速的计算效率有着很大影响。

图 2-3 展示了适用于 AM6xA 与 TDA4x 处理器搭载的 TIDL 的一般开发流程。可以从多个点进入该开发流程。德州仪器 (TI) 提供了基于 GUI 与基于命令行的工具，用户能够：

- 引入数据 (BYOD)，训练德州仪器 (TI) 模型
- 引入定制架构的预训练模型 (BYOM)
- 评估来自德州仪器 (TI) Model Zoo 的预训练与预优化模型。

每个开发步骤都会为下一步提供支持。开发人员能够为目标 SoC 编译模型，并在部署到目标以前，在 PC 上测试精度。编译工具与加速器可通过 Tensorflow Lite、ONNX Runtime 或 TVM 等开源运行时框架调用。该等运行时框架提供了熟悉的 API，允许未加速层在 Arm® A 核心上运行，从而为广泛的模型提供便捷的使用体验。该等开源运行时框架 (OSRT) 在底层利用了 TIDL 运行时框架 (TIDL_RT)。

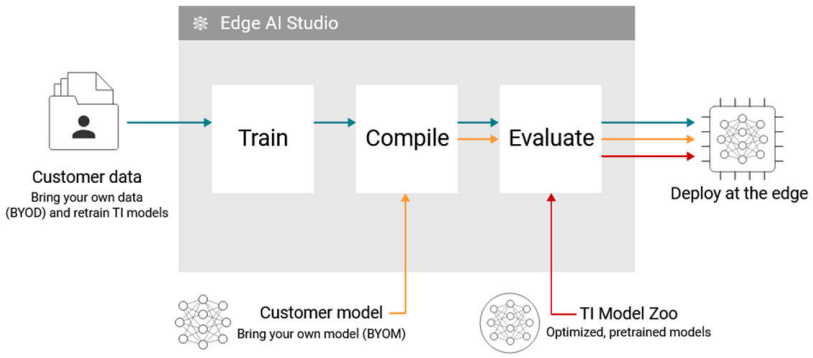

图 2-3. 德州仪器 (TI) 深度学习开发流程

2.5 GStreamer 与 TIOVX 框架

AM6xA 器件支持两种视觉框架：GStreamer 与 OpenVX。确切地说，德州仪器 (TI) 已实施并且符合 OpenVX 标准；该实施方案被命名为德州仪器 OpenVX (TIOVX)。该等框架支持片上硬件加速器（例如：C7xMMA 与 ISP）。在底层架构中，该等框架能够处理 IPC 与数据管理，以便降低应用级复杂性。

GStreamer (GST) 是一个基于 Linux 的开源多媒体流水线框架。流水线由能够实现特定功能的插件构成（例如：采集图像、更改数据格式、缩放大小、压缩或写入文件）。虽然支持使用多种社区插件，但使用德州仪器 (TI) 提供的插件（利用片上加速器）能够获得最佳性能。在 Edge AI SDK 中，可提供该等插件，源代码可用于修改或扩展德州仪器 (TI) 插件。GST 对于快速构建与测试流水线特别有效。除了 C++、Python 以及其他语言支持以外，GST 流水线还可以直接通过命令行运行。例如：

```
v4l2src device=/dev/video-usb-cam0 ! image/jpeg, width=1280, height=720 ! jpegdec ! video/x-raw,
format=NV12 ! kmssink driver-name=tidss sync=true
```

上述流水线由以“!”分隔符分隔的单个插件组成。该流水线能够从利用 v4l2 的 USB 摄像头读取图像，解码 JPEG 编码图像，将帧编码转换为 NV12，并且通过 Linux 的 KMS/DRM 接口将帧推送到显示子系统 (DSS)，以便在显示器上实现可视化。流水线很可能非常庞大和复杂，但仍然可以通过命令行运行，而无需进行其他编码。

TIOVX 是用于在异构 SoC 上构建视觉流水线的底层框架。与 GST 相类似，流水线由非循环图中的节点组成，每个节点能够在目标内核上运行一个函数。例如，在 ISP 上处理原始帧或在 C7xMMA 上运行神经网络。TIOVX 应用采用 C/C++ 编写，且需对 SOC 有更多的了解。在底层架构中，GST 利用 TIOVX 与硬件加速器进行通信。GST 需要与 Linux 进行更多交互，以便传递来自插件的控制信号，而 TIOVX 则允许内核更直接地通信。TIOVX 能够在包括 Linux 与 QNX 的诸多操作系统间移植，适用于功能安全 (FuSa) 认证应用。也正因此，TIOVX 成为汽车级与其他 ASIL 和 SIL 等级用例的绝佳选择。在中断使用方面，TIOVX 框架的效率略高于 GST；但是，在帧速率、加速器利用率以及 DDR 带宽方面，GST 与 TIOVX 几乎不相上下。

请注意，对于深度学习模型，TIOVX 应用将用户限制为使用 TIDL_RT；运行时，不能通过 TIOVX 使用 ONNX Runtime 等开源运行时框架 (OSRT)。OSRT 的作用之一是为 TIDL 不支持的层提供备用实施。因此，通过仅使用 TIOVX 的应用程序直接调用 TIDL_RT，这意味着，对于神经网络中的所有层均必须由 TIDL 提供支持。

大多数情况下，GST 足以在运行 Linux 的 TI AM6xA SoC 上构建视觉处理应用。如果需要功能安全与其他操作系统，TIOVX 是更合适的选择。

3 性能注意事项与基准测试工具

构建视觉应用时，需要考虑以下性能指标，并且需要进行基准测试：

- **端到端延迟**：必须最大限度缩短采集图像与生成析结果之间的延迟，以便及时做出决策，采取响应措施。
- **视频吞吐量（每秒帧数）**：必须以所需帧率采集与处理图像，不得出现掉帧现象。
- **CPU 负载**：视觉流水线对通用 CPU 内核（对于 AM62A 则为 A53）的负载必须降至最低，因为所有图像处理都是在硬件加速器上完成的。
- **DDR 利用率**：对于视觉流水线的 DDR 读取与写入操作，必须为其他系统任务留出充足带宽。

- **硬件加速器 (HWA) 负载（ISP、VPU、C7x/MMA）**：HWA 专用于特定功能，不能用于其他目的。通过视觉流水线，HWA 的利用率可高达 100%，并且留有一定余量。

适用于 AM62A 的 EdgeAI SDK 提供了几种能够对该等性能指标进行基准测试的工具：

- **Perf_stats 工具 [5]**：测量 CPU 内核与 HWA 上的负载以及 DDR 利用率。
- **GStreamer 调试跟踪**：通过设置环境变量 GST_DEBUG_FILE，可将 GStreamer 调试信息重定向至一个文件。EdgeAI SDK 脚本（/opt/edgeai-gst-apps/scripts/gst_tracers/parse_gst_tracers.py）能够处理该等消息并估算 GStreamer 流水线每个元素的处理时间。
- **GStreamer 插件 *fpsdisplaysink***：以帧/秒（fps）为单位，显示流水线的吞吐量。
- **自定义 GStreamer 插件 *tiperfoverlay***：在显示屏显示（或在终端控制台打印）CPU 负载、DDR 利用率、HWA 负载以及 fps。

4 参考设计

本部分介绍了一款适用于驾驶员与乘员监控的视觉流水线参考设计，可利用单个 RGB-IR 图像传感器实现视频通话或录制功能。本部分涵盖硬件设置、软件设计、工具以及支持基准测试结果的示例应用。

4.1 摄像头模块

Leopard Imaging 的 LI-OX05B1S-MIPI-137H 摄像头模块用于捕捉 RGB-IR 图像。该摄像头通过两根适配器电缆连接至 AM62A SK EVM：FAW-1233-03 与 LI-FPC22-IPEX-PI，如 图 4-1 所示。

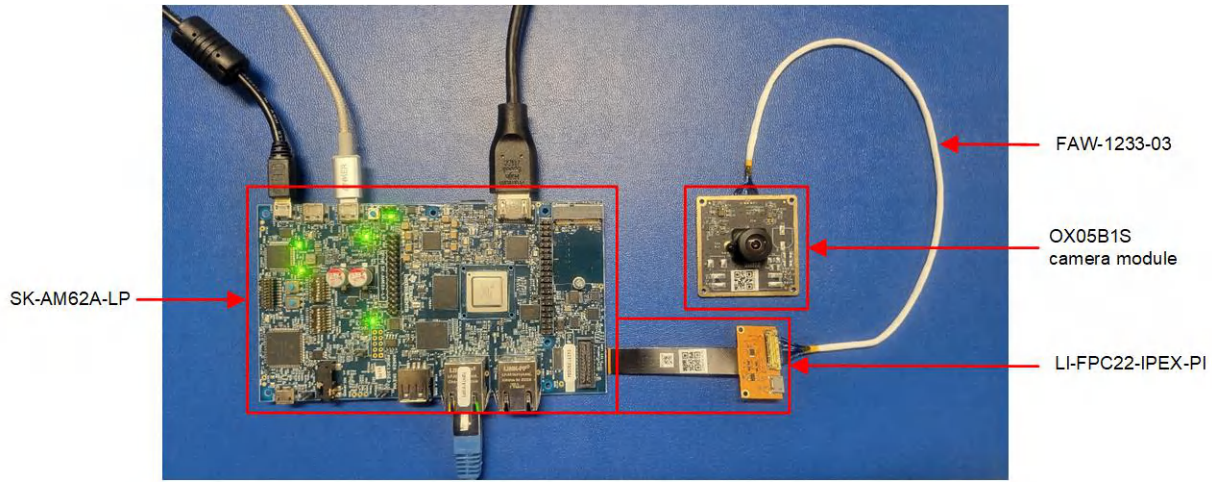

图 4-1. 基准设计硬件设置

4.2 传感器驱动程序

OX05B1S 传感器的驱动程序通过适用于 AM62A 的处理器 SDK Linux（或 EdgeAI SDK）提供，位于 linux-kernel 中的 drivers/media/i2c/ox05b1s.c 源文件中。该驱动程序可将传感器配置为两种工作模式：

- **模式 A（虚拟通道 0）**：该模式可与开启外部 IR 照明器同步。该模式下，传感器能够采集并发送以 IR 主导数据流。
- **模式 B（虚拟通道 1）**：该模式可与关闭外部 IR 照明器保持同步。该模式下，传感器能够采集并发送 RGB 主导数据流。

该等两种模式可交替使用。每种模式均可运行能够实现每个数据流所需帧率的不同帧数。例如，两个数据流都能够以 30fps 的速率运行，或一个数据流能够以 15fps 的速率运行，另一个数据流能够以 45fps 的速率运行。

4.3 CSI-2 Rx 驱动程序

符合 V4L2 标准的 CSI-2 Rx 驱动程序（包含在处理器 SDK 中）能够接收来自传感器的图像数据，并根据虚拟通道编号区分 RGB 主导流与 IR 主导流。然后，驱动程序能够利用专用 DMA 上下文，将每个数据流存储到 DDR 中。在用户空间创建两个视频设备节点，这样，应用程序能够分别检索 RGB 主导的图像数据与 IR 主导的图像数据。对于从传感器到视频设备节点的数据流，可通过 V4L2 框架，建模为媒体设备拓扑结构。该拓扑结构可通过 *media-ctl --print* 命令以文本形式显示，也可以利用 Linux dot 工具进行可视化。

图 4-2 显示了文本中基于 SDK 10.1 的媒体设备拓扑结构。该拓扑结构包含从传感器 (ox05b 4-0036) 到 CSI-2 Rx (cdns_csi2rx.30101000.csi-bridge)，再到 DMA 包装器 (30102000.ticsi2rx) 的两个数据流。DMA 包装器利用两个 DMA 上下文，将图像数据传输到 DDR，每个上下文均可链接一个设备节点（/dev/video3 与 /dev/video4）。然后，用户空间应用可通过该等两个设备节点，访问原始图像数据。

```
root@am62axx-evm:~# media-ctl -p

Device topology
- entity 1: 30102000.ticsi2rx (7 pads, 7 links, 2 routes)
            type V4L2 subdev subtype Unknown flags 0
            device node name /dev/v4l-subdev0
        routes:
                0/0 -> 1/0 [ACTIVE]
                0/1 -> 2/0 [ACTIVE]
        pad0: Sink
                [stream:0 fmt:SBGGI10_1X10/2592x1944 field:none colorspace:srgb]
                [stream:1 fmt:SBGGI10_1X10/2592x1944 field:none colorspace:srgb]
                <- "cdns_csi2rx.30101000.csi-bridge":1 [ENABLED,IMMUTABLE]
        pad1: Source
                [stream:0 fmt:SBGGI10_1X10/2592x1944 field:none colorspace:srgb]
                -> "30102000.ticsi2rx context 0":0 [ENABLED,IMMUTABLE]
        pad2: Source
                [stream:0 fmt:SBGGI10_1X10/2592x1944 field:none colorspace:srgb]
                -> "30102000.ticsi2rx context 1":0 [ENABLED,IMMUTABLE]

- entity 9: cdns_csi2rx.30101000.csi-bridge (5 pads, 2 links, 2 routes)
            type V4L2 subdev subtype Unknown flags 0
            device node name /dev/v4l-subdev1
        routes:
                0/0 -> 1/0 [ACTIVE]
                0/1 -> 1/1 [ACTIVE]
        pad0: Sink
                [stream:0 fmt:SBGGI10_1X10/2592x1944 field:none colorspace:srgb]
                [stream:1 fmt:SBGGI10_1X10/2592x1944 field:none colorspace:srgb]
                <- "ox05b 4-0036":0 [ENABLED,IMMUTABLE]
        pad1: Source
                [stream:0 fmt:SBGGI10_1X10/2592x1944 field:none colorspace:srgb]
                [stream:1 fmt:SBGGI10_1X10/2592x1944 field:none colorspace:srgb]
                -> "30102000.ticsi2rx":0 [ENABLED,IMMUTABLE]

- entity 15: ox05b 4-0036 (1 pad, 1 link, 2 routes)
             type V4L2 subdev subtype Sensor flags 0
             device node name /dev/v4l-subdev2
        routes:
                0/0 -> 0/0 [ACTIVE]
                0/0 -> 0/1 [ACTIVE]
        pad0: Source
                [stream:0 fmt:SBGGI10_1X10/2592x1944@1/60 field:none colorspace:srgb]
                [stream:1 fmt:SBGGI10_1X10/2592x1944@1/60 field:none colorspace:srgb]
                -> "cdns_csi2rx.30101000.csi-bridge":0 [ENABLED,IMMUTABLE]

- entity 21: 30102000.ticsi2rx context 0 (1 pad, 1 link)
             type Node subtype V4L flags 0
             device node name /dev/video3
        pad0: Sink
                <- "30102000.ticsi2rx":1 [ENABLED,IMMUTABLE]

- entity 27: 30102000.ticsi2rx context 1 (1 pad, 1 link)
             type Node subtype V4L flags 0
             device node name /dev/video4
        pad0: Sink
                <- "30102000.ticsi2rx":2 [ENABLED,IMMUTABLE]
```

2 DMA contexts at the CSI Rx driver

2 streams from the sensor

Video device node for IR stream

Video device node for RGB stream

图 4-2. 文本中的媒体设备拓扑

也可以利用 Linux dot 实用程序，对媒体设备拓扑结构进行可视化。在 EVM 上运行以下命令，以便生成 dot 文件：

```
root@am62axx-evm:~# media-ctl --print-dot > media.dot
```

然后，在 Linux PC 上运行以下命令，以便生成 png 图像文件，如下图所示。

```
$ dot -Tpng media-top.dot -o media-top.png
```

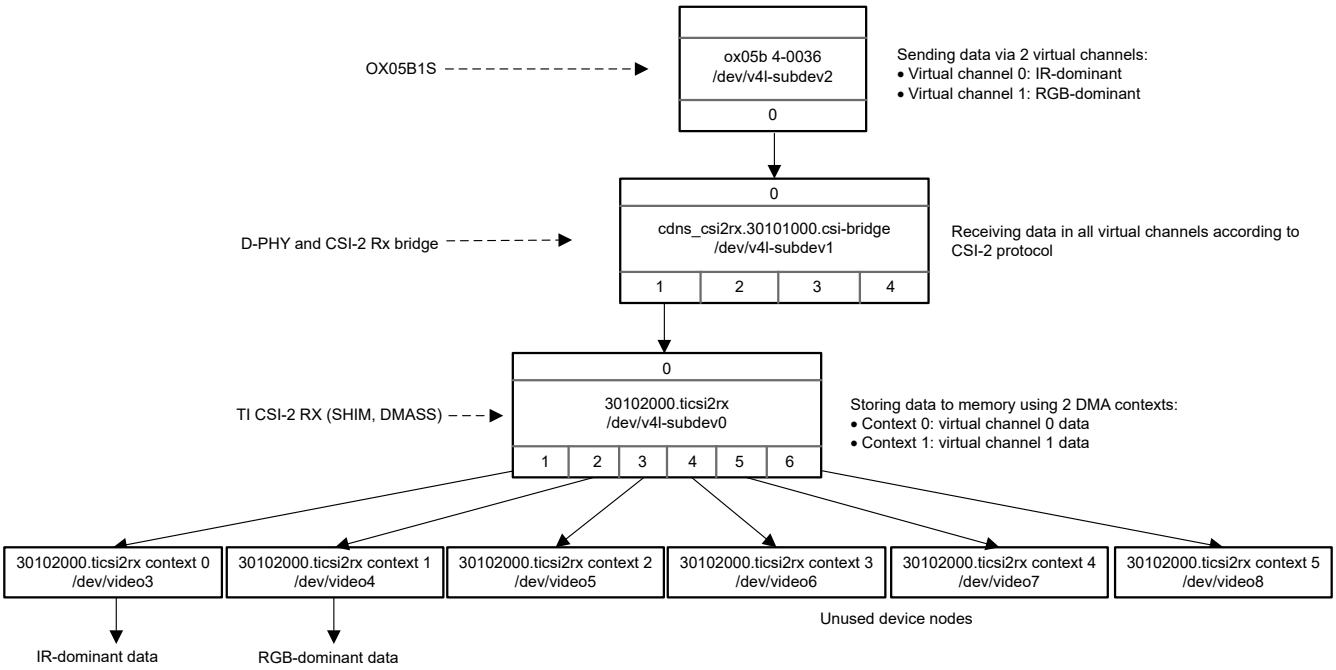

图 4-3. 媒体设备拓扑结构可视化

SDK 中的 OX05B1S 驱动程序能够将传感器配置为通过虚拟通道 0 传输 IR 主导数据，通过虚拟通道 1 传输 RGB 主导数据。DMA 上下文 0 用于存储虚拟通道 0 的数据。因此，CSI2 Rx 驱动程序创建的第一个视频设备节点（上例中的 /dev/video3）可用于接收 IR 主导数据。同样，第二个设备节点（上例中的 /dev/video4）可用于接收 RGB 主导数据。

4.4 图像处理

OX05B1S 是一款原始传感器，其采集的原始图像可通过 ISP 处理。为了在 RGB 应用中获得最佳图像质量，必须对 ISP 调优。如需了解适用于特定传感器的 ISP 调优方法更多详细信息，可参阅《AM6xA ISP 调优指南》[4]。对于适用于 OX05B1S 的预调优 ISP 配置二进制文件，请参阅目标设备 /opt/imaging/ox05b1s/linear 路径下的 SDK。

AM62A ISP 会为每个 RGB-IR 输入帧生成两个输出帧：一个 YUV 帧与一个 IR 帧。在本参考设计中，OX05B1S 配置为以 30fps 的速率交替输出 RGB 主导流与 IR 主导流。在通过 GStreamer 或 TIOVX 实现的视觉流水线中，输出 RGB 主导流时仅使用 RGB 输出帧，丢弃 IR 帧，反之亦然，输出 IR 主导流时，仅使用 IR 帧，丢弃 RGB 输出帧。

处理 RGB 主导流时，ISP 还会生成可用于自动曝光与增益控制 (AE) 以及自动白平衡 (AWB) 的统计数据。处理器 SDK 支持 AE 与 AWB 算法（又称“2A”）。在本参考设计中，该算法可用于调整传感器曝光与增益以及白平衡增益。曝光与增益调整会发送到传感器，白平衡增益会提供给 ISP。

4.5 适用于驾驶员与乘员监控的深度学习

从图像分析与深度学习角度来看，驾驶员监控系统 (DMS) 与乘员监控系统 (OMS) 通常是单独的处理路径。这两种情况下，通常都会使用来自 RBG-IR 摄像头的 IR 帧。这样，就能够利用非可见光，对车辆内部进行充分照明，从而在保持驾驶员夜视能力的同时实现精确监控。

因此，作为单通道灰度图像，分析相应图像。当前仅处理单通道数据而不提供三通道 RGB 数据，从而降低了处理要求与 DDR 带宽。不过，分析单通道（例如：灰度）图像意味着也要在该等数据上，训练神经网络模型，但是，典型模型是针对三通道 RGB 训练的。TIDL 完全能够处理任意数量的输入通道与分辨率。

适用于驾驶员监控的深度学习

驾驶员监控必须确定驾驶员什么时候在关注路况，什么时候没有关注路况。名义上可以理解为疲劳或分心。这两种情况下，首先要关注驾驶员的头部位置、目光以及眼睛。眼球与眼睑的运动速度非常快，因此，必须以适当帧率（通常约为 30 FPS）进行分析。地方法规标准（例如：Euro NCAP）能够改变这一要求。较简单的 DMS 系统能够仅进行头部姿态检测，但是无法处理复杂的“*蜥蜴型*”场景，即：驾驶员的头部朝向道路，但看向别处（例如：手机）。

DMS 的典型流程如 图 4-4 所示。请注意，有几种可行的方法与技术。例如，一些系统能够通过头部姿势检测（而非目光检测）判断驾驶员是否分心。

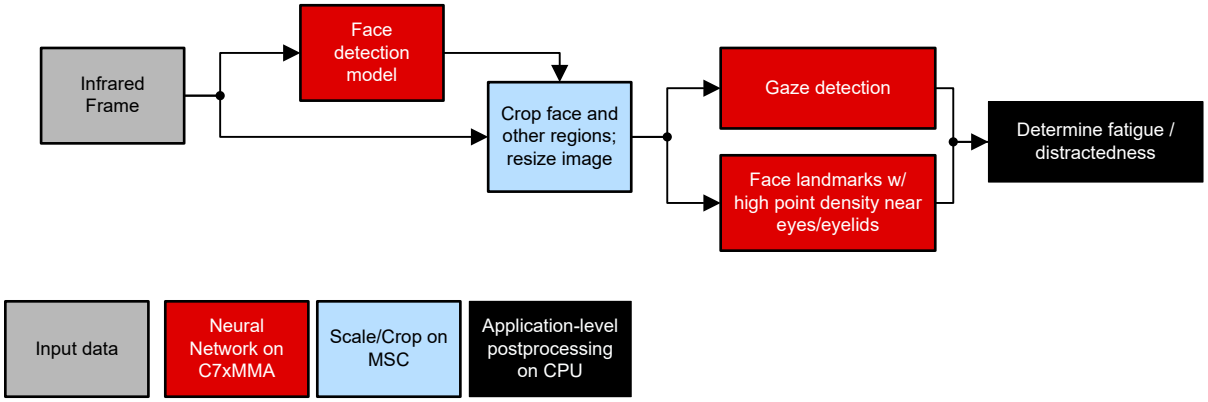

图 4-4. 驾驶员监控图像分析流程

深度学习模型能够提供有关驾驶员注意力的信息。不过，需要对各帧图像进行一定程度的后处理。例如，单帧显示闭眼可能是在眨眼，但连续数帧显示闭眼就可能是在瞌睡或微睡。同样，目光偏离车辆前方道路，可能是分心，也可能是必要的驾驶活动（例如：看向即将转弯的方向）。因此，适用于 DMS 的深度学习算法必须提供足够高的帧率，以便能够跨多帧进行该等跟踪。

适用于乘员监控的深度学习

乘员监控会收集关于车内座位乘坐情况以及安全带使用情况的信息。相较驾驶员头部位置与眼球运动，该等信息变化速率相对较慢，因此，对帧速率的要求相对较低；多数情况下，可以接受 1FPS 到 5FPS 的帧率。不过，需要关注更大的区域，通常是整个车辆内部，而并非是仅需关注驾驶员座椅。因此，必须以更高的分辨率运行相应模型，并且具有更高的处理要求。OMS 负责检查哪些座位有人、安全带是否正确使用以及发生碰撞时安全气囊如何展开。

适用于乘员监控的数据流示例如 图 4-5 所示。神经网络能够对单个图像进行多级处理，以便确定车内有多少乘员、乘员具体位置以及对安全气囊展开有何影响。

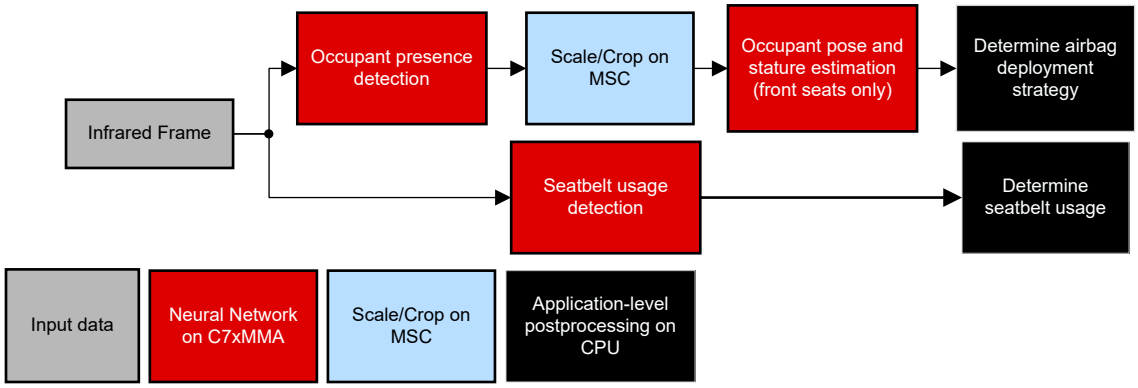

图 4-5. 乘员监控图像分析流程

利用 TIDL 运行多个模型

TIDL 允许同时加载多个深度学习模型。只要模型的权重与配置适合可用的持久 DDR 空间，应用程序就能够以任何顺序进行初始化并运行多个模型。无需对 TIDL 的并发调用进行特殊处理。

本报告介绍的几种模型具有不同的帧率要求与不同的复杂程度。TIDL 能够对模型进行优先级排序与抢占，以便适合该等应用。例如，具有较高帧率要求的 DMS 模型可以设置较高的运行优先级，以便确保模型可以在帧间延迟方面保持较高的运行速率。对于具有较低 FPS 要求但规模较大的 OMS 模型，可以设置较低的运行优先级，以便利用 DMS 帧间未使用的周期。开发人员需要分析模型的运行时框架延迟，确保有充足余量以所需帧率在延迟限制范围以内运行每个模型。

4.6 参考代码与应用

在德州仪器 (TI) GitHub [6] 上查找适用于下一部分应用的示例代码。本应用手册的测试与基准测试是在 SK-AM62A-LP 入门套件板搭载的 AM62A74 处理器上进行的。器件运行适用于 Edge AI 应用的 Linux SDK（版本号：10.00.00.08）。

5 应用示例与基准测试

本部分介绍了使用 LI-OX05B1S-MIPI-137H 摄像头的几个示例以及适用于每个示例的基准测试结果。

5.1 应用 1：利用 GST 进行单数据流采集与可视化

这是一个从摄像机向显示器进行流式传输并且进行 RGB 数据可视化的简单示例。本示例演示了对 节 3 所列性能指标进行基准测试的具体方法。本示例的 GStreamer 流水线元件如下所示。

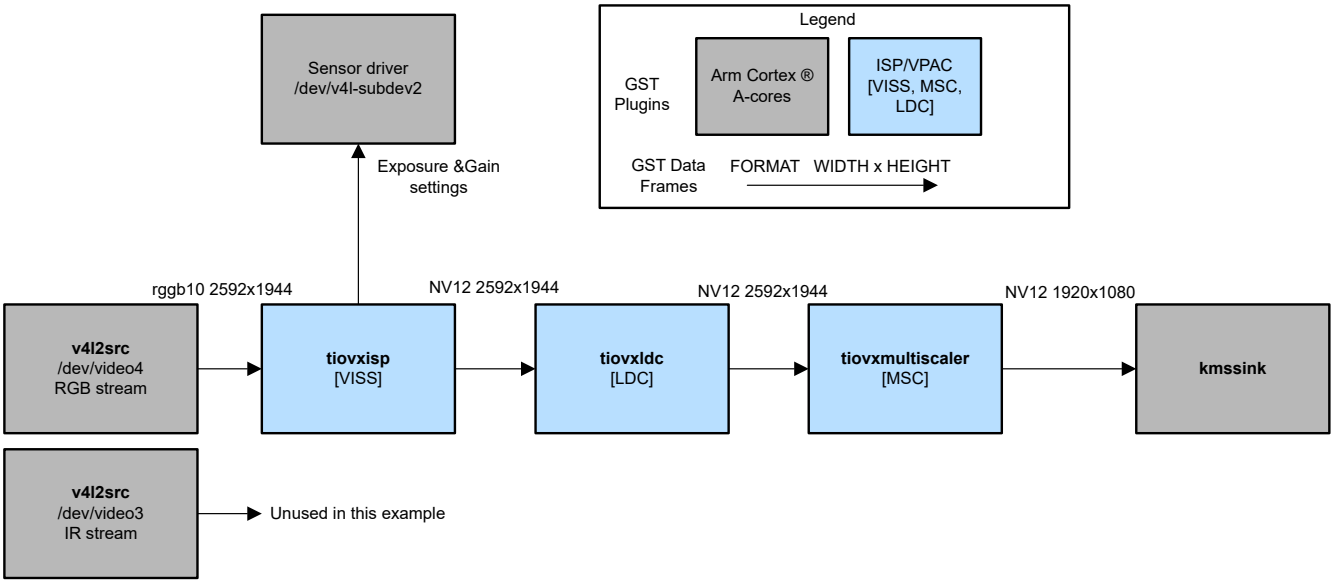

图 5-1. 利用 GStreamerr 进行 RGB 流采集与显示

相应的 GST 命令可通过 GitHub 存储库 [6]获取。为了对性能指标进行基准测试，可以在一个终端控制台运行该命令，同时在另一个终端控制台运行 perf_stats 工具 [5]。

- 在第一个控制台打印出流水线吞吐量 (fps)，如 图 5-2 所示。与此同时，在显示屏显示吞吐量。
- 对于 CPU 负载、HWA 负载以及 DDR 利用率，可以在第二个控制台打印，同时在显示屏显示。GST 命令运行期间，该等性能指标会不断更新。图 5-3 显示了单个更新的屏幕截图。
- 停止 GST 流水线后，运行“/opt/edgeai-gst-apps/scripts/gst_tracers/parse_gst_tracers.py /run/trace.log”，以便生成流水线中每个元素的延迟测量值，如 图 5-4 所示。图中所示 tiovxisp0 (VISS)、tiovxldc0 (LDC) 和 tiovxmultiscaler0 (MSC) 的延迟符合预期：
  - 对于 VISS 与 LDC，延迟时间约为 5MPixel/375MHz，开销约为 14-15msec，其中，375MHz 是 ISP(VPAC) 的工作时钟频率。
  - 对于 MSC，可以同时或单独处理 YUV 数据。单独处理时（默认配置），亮度 (Y) 平面延迟约为 14-15msec，与 VISS 和 LDC 相同，而色度 (UV) 平面延迟约为亮度平面延迟的一半，即：7msec。因此，MSC 的总延迟约为 21msec。通过配置 MSC 同时处理两个平面，可以将延迟缩短至 14msec。

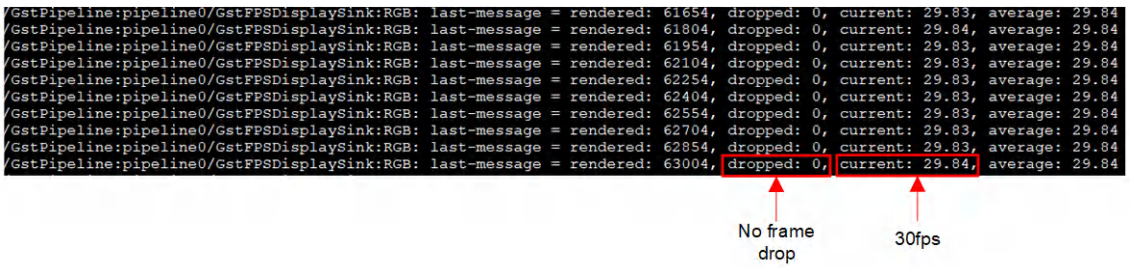

图 5-2. 单数据流采集与可视化的流水线吞吐量

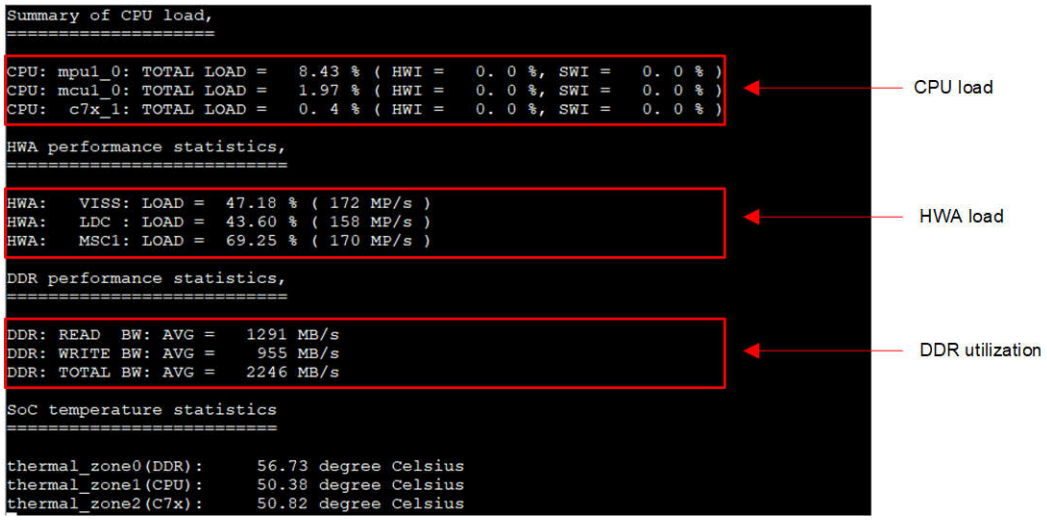

图 5-3. 单数据流采集与可视化的 CPU、HWA 和 DDR 负载

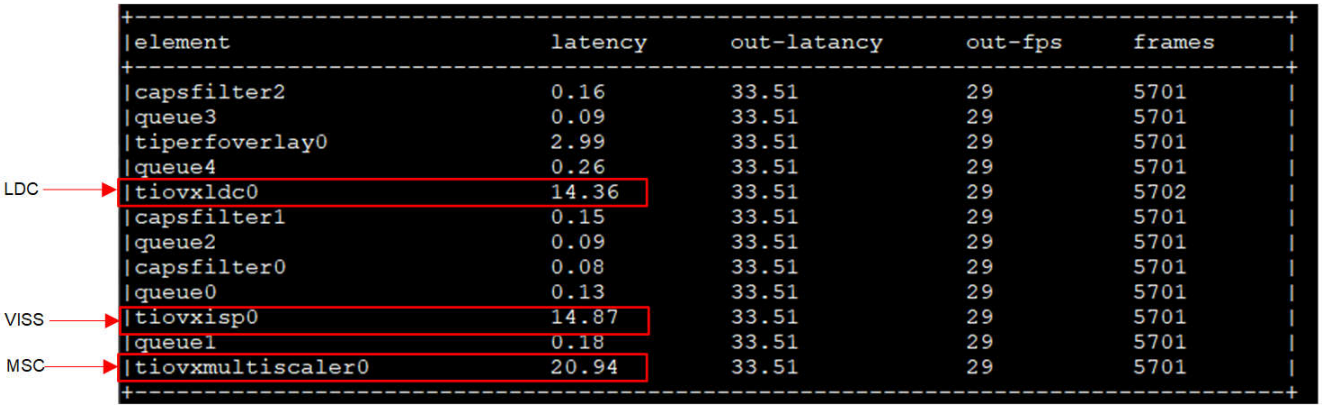

图 5-4. 单数据流采集与可视化的流水线元素延迟

5.2 应用 2：利用 GST 与 TIOVX 框架进行双流采集与可视化

本部分将分析一个同时对 RGB 流与红外流进行可视化的应用程序。

该应用由 GStreamer 与 TIOVX 共同组成，以便对两者进行对比。图 5-5 显示的是 GStreamer 中的应用程序，图 5-6 显示了利用 TIOVX 构建的等效应用程序。对于 GST 命令与 TIOVX 代码，可通过 GitHub [6] 获取。

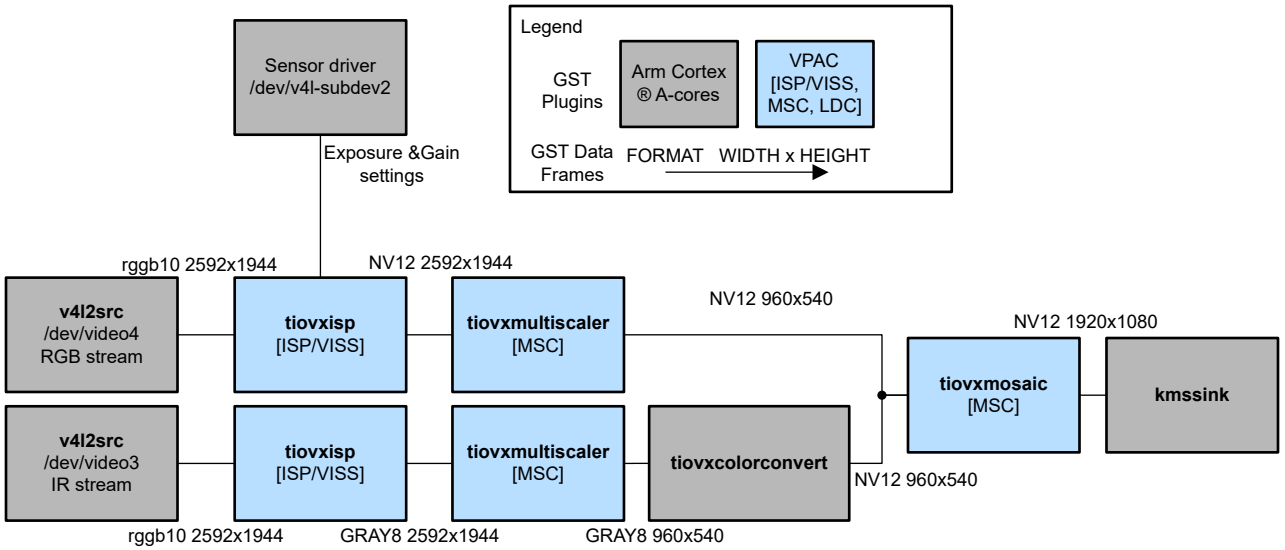

图 5-5. 利用 GStreamerr 进行双流采集与显示

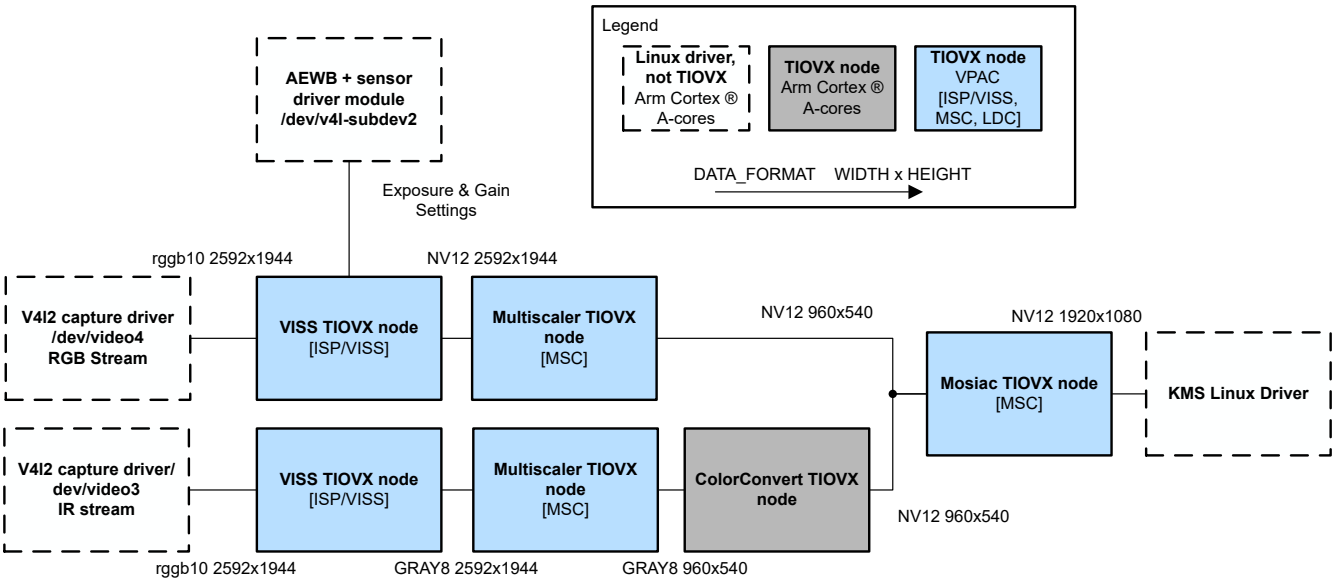

图 5-6. 利用 TIOVX 进行双流采集与显示

帧数据通过 v4l2，从与 RGB 和 IR 流对应的 /dev/videoX 条目到达，并且通过片上 ISP 进行处理。两个数据流都会缩放到适合显示器的分辨率，IR 数据流（灰度）会转换为与 RGB 数据流相同的色彩格式。然后，在通过 Linux KMS 或 DRM 接口显示到监视器以前，将该等数据流合并至支持*马赛克*功能的单个帧。

TIOVX 与 GStreamer 应用在处理功能方面是等效的，但也存在一些主要差异。TIOVX 应用程序会构建一个处理应用程序内部主体的 TIOVX 图形，在本例中，为 ISP、比例缩小、色彩转换以及图像合并（马赛克）功能。利用 TIOVX 图形外的 Linux 级 API，处理来自 V4L2 的输入与通过 KMS 或 DRM 的输出。不过，GStreamer 拥有很多可用插件，可利用该等插件，实施该等 API。TIOVX 应用程序会被编译为二进制应用程序并运行，而 GStreamer 流水线可利用能够通过命令行运行的单个字符串表示。

性能统计数据与 SoC 资源利用率

本部分将分析在运行该等应用程序时 AM62A74 的资源利用率。测量应用程序能够通过 TIOVX 接收远程内核利用率，并且通过存储器映射寄存器来读取 DDR 利用率和温度等其他信息。该 ***perf_stats*** 应用程序是 SDK 的一部

分，位于 /opt/edgeai-gst-apps/scripts/perf_stats 目录下。SoC 利用率的采样间隔为 500ms；在 20s 的时间窗口完成该等采样（每个 RGB 流与 IR 流为 600 帧），然后平均到 图 5-7 所示双条形图中。

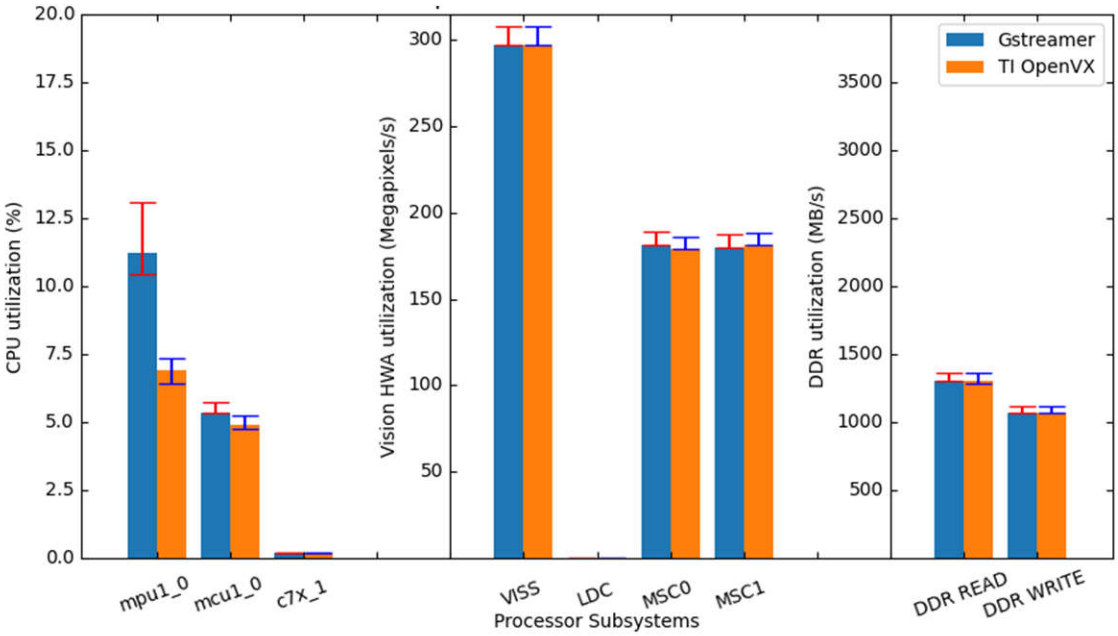

图 5-7. 适用于双流可视化流水线的 GStreamer 与 TIOVX 的利用率比较

该图表中的误差条表示第 25 百分位数与第 75 百分位数。两种框架下，当应用以 30FPS（每个输入流）运行时，加速器与 DDR 的利用率相当，不会出现掉帧现象。值得注意的是，MPU（SMP 模式下的四路 Arm® Cortex®-A53，图 5-7 中标记为 mpu1_0）在 GStreamer 中的利用率高于 TIOVX。这是因为，CPU 复合体与任何远程内核或加速器之间的信号增加。在该应用中，未使用 C7xMMA。除此之外，两个应用框架之间的核心或 HWA 的利用率非常相似，TIOVX 效率稍高。

表 5-1 显示了通过流水线各个元件的延迟。不包括帧采集与显示延迟。对于应用程序中的每个处理任务，对比 GStreamer 与 TIOVX 的延迟以及总延迟。因此可以认为，TIOVX 一般更快，特别是在色彩转换方面；红外路径受应用框架差异的影响最大。

表 5-1. GStreamer 延迟与 TIOVX 延迟

功能	GStreamer (ms)	TIOVX (ms)
VISS ISP (红外)	18.5	13.9
VISS ISP (RGB)	17.6	14.1
MSC 缩小(红外)	14.3	13.7
MSC 缩小(RGB)	21.2	20.5
色彩转换（红外->NV12）	19.2	0.64
马赛克合并图像（RGB + IR）	5.5	4.7
总延迟（IR 路径）	**57.5**	**32.9**
总延迟（RGB 路径）	**44.3**	**39.3**

GStreamer 在内部为每个插件实现了 TIOVX 节点，因此，TIOVX 节点的测量速度总是快于 GStreamer 中的同等插件。GStreamer 的测量是在插件从 Linux 运行前后捕获的，TIOVX 的测量可以通过运行操作的远程内核采集与报告。存在明显提升。[1] 所示），特别是从灰度到 NV12 格式的色彩转换延迟。[2]

对比该等应用程序的另一种方法是中断与处理器间通信的频繁程度（即：A53 向 C7x 发送信息，R5F 向 A53 发送信息等）。中断次数越少越好，因为这样能够让处理器更快地处理来自不同外设与加速器的待处理信号。测量 Linux 的中断次数，以便查看 A53（运行 Linux）在每个应用程序运行 600 帧前后收到的中断次数。

在 20 秒的持续时间内（每个流 600 帧），Linux 的中断次数。GStreamer 显示的中断次数多于 TIOVX，因为运行 Linux 的 Cortex A53 内核必须在每个插件/流水线元素之间发出通知。于单个内核邮箱的中断计数，通过 /proc/interrupts 获取。

表 5-2. GStreamer 中断次数与 TIOVX 中断计数

	GStreamer 应用程序	TIOVX 应用
DM R5F（管理 VPAC）	13,469	10,589
C7x	0（未使用）	0（未使用）
MCU R5F	0（未使用）	0（未使用）

表 5-2 中的数据反映了一个总体趋势，即：在 CPU 中断方面，GStreamer 的效率低于 TIOVX。这是因为，TIOVX 允许所有内核直接通信，但 GStreamer 要求内核通过 Linux 主机 (A53) 流动。利用 TIDL 增加 AI 处理也显示出了类似的 C7x 中断模式。

5.3 应用 3：GStreamerr 的代表性 OMS-DMS + 视频电话流水线

在本示例应用中，运行了一个具有代表性的 GStreamer 流水线，用于对来自 OX05B1S 摄像头的实时流进行乘员监控、驾驶员监控以及视频电话。

对于 OMS 与 DMS，红外帧分别通过分割模型与物体检测模型。该等模型分别模拟安全带检测与面部检测任务。RGB 帧经过 H.265 编码以后保存到文件中。这代表了 RGB-IR 座舱内监控应用的起点。请注意，此处所用深度学习模型并未进行优化。[3] 最终产品需要扩展，以便包含更多 DMS 与 OMS 模型以及后处理，从而打造一个功能强大且丰富的解决方案。

[1] 在一些 GStreamer 流水线中，单个插件的延迟可能会受到相邻插件与队列的影响，这是因为，该等队列会影响 GStreamer 选择多线程插件与部分流水线的具体方式。这也解释了为什么 表 5-1 中 GStreamer 应用程序的延迟超过了 TIOVX 中延迟时间的原因（如 图 5-4.
[2] 请注意，实际应用中，不需要进行色彩转换；加入色彩转换是为了通过拼接帧同时对 RGB 流与 Ir 流进行可视化。两个帧必须采用相同格式。生产应用程序不需要这一色彩转换步骤。
[3] 虽然这里介绍的模型使用的是 RGB 数据，但最佳模型使用红外帧为灰度图像。对于本应用手册介绍的模型，DDR 利用率高于为该用例设计的 DMS/OMS 模型。

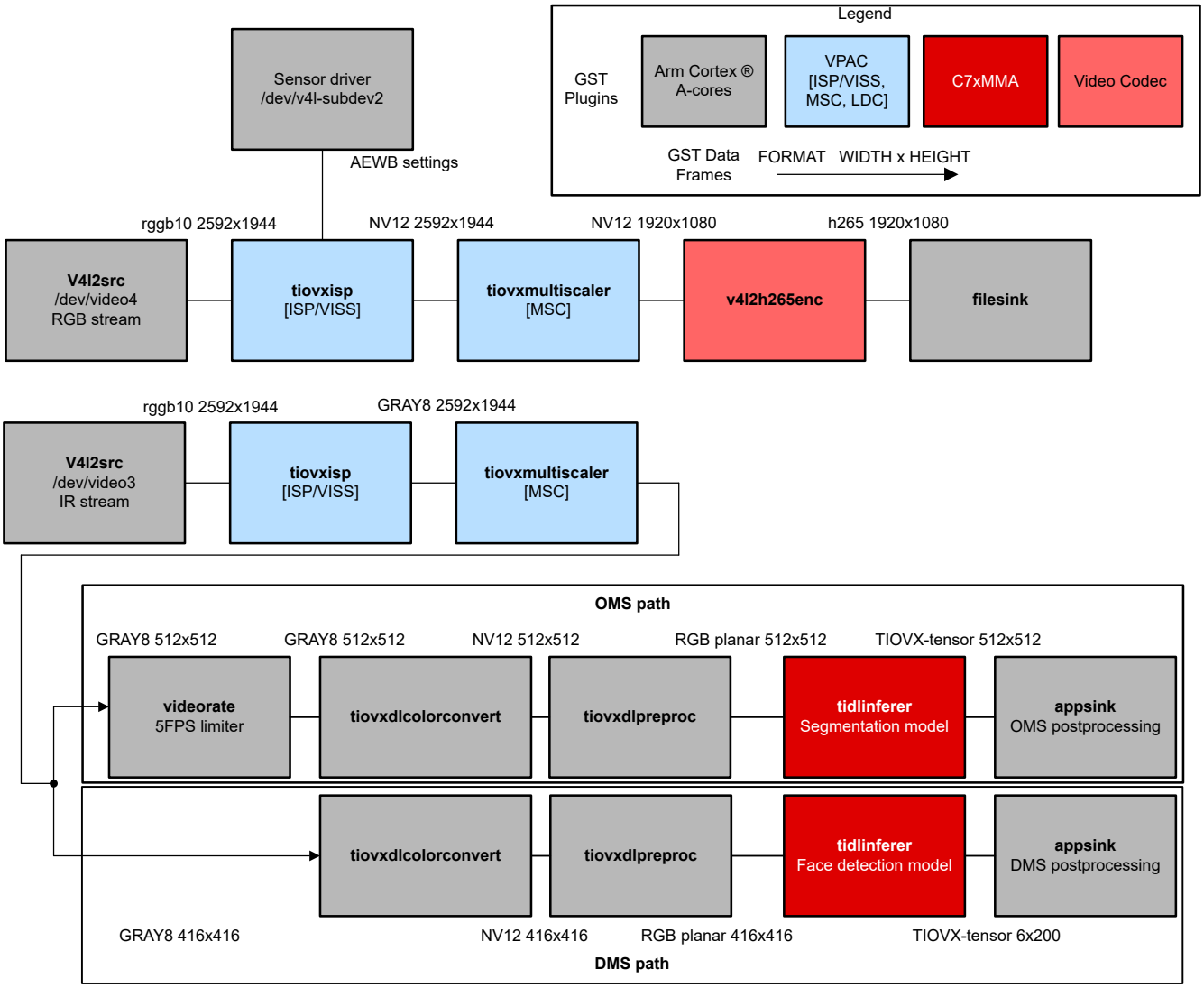

图 5-8. 支持视频记录功能的 DMS-OMS 双流 RGB-IR GStreamer 应用流程

在本应用中（如 图 5-8 所示），RGB 帧仅用于文件存储，通过 H.265 格式进行缩放与编码。在拆分为 OMS 与 DMS 路径以前，会对红外输入进行处理，并且按比例进行缩放。DMS 以 30FPS 速率运行，而 OMS 运行速率更慢，为 5FPS（丢弃 OMS 路径多余帧）。在此基础上，可通过缩放、将颜色空间转换为 RGB（尽管可针对灰度输入对最终模型进行优化）、预处理以及通过 C7x 深度学习加速器上的 CNN 运行，对适用于深度学习的红外帧做好准备。对于该等模型的任何后处理，均在 GStreamer *appsink* 插件公开的应用代码中运行。

图 5-9 显示了在运行该应用时 AM62A 的内核负载。VISS-ISP 硬件加速器几乎能够达到 315 MP/s 的最大容量。不过，处理内核仍有充足的余量。4x Arm® Cortex®-A53s 仅使用了约 26%，为应用程序代码与其他服务留出了充足空间。C7xMMA AI 加速器的负载不到 50%，为其他 AI 模型留出了空间，并可通过进一步优化最大程度提升 IR 图像分析能力。32 位 3200 MT/s DDR 总线显示利用率为 35%，在占用这一共享资源以前，利用率可以达到约 50-60%。请注意，在优化整个系统时，必须充分考虑 DDR 的利用率。

单独来看，流水线的 OMS、DMS 以及电话/录音部分的延迟时间分别约为 85ms、51ms 以及 67ms（不包括帧采集延迟）。

图 5-9 显示了适用于 DMS-OMS 参考 GStreamer 应用程序的计算、加速器以及 DDR 利用率。请注意，此处的模型是德州仪器 (TI) Model Zoo 的现成模型，用于模拟简单 DMS/OMS 应用的实际负载。还可以进行进一步的深度学习与图像分析算法（包括：优化与降低 DDR 负载的机会）。

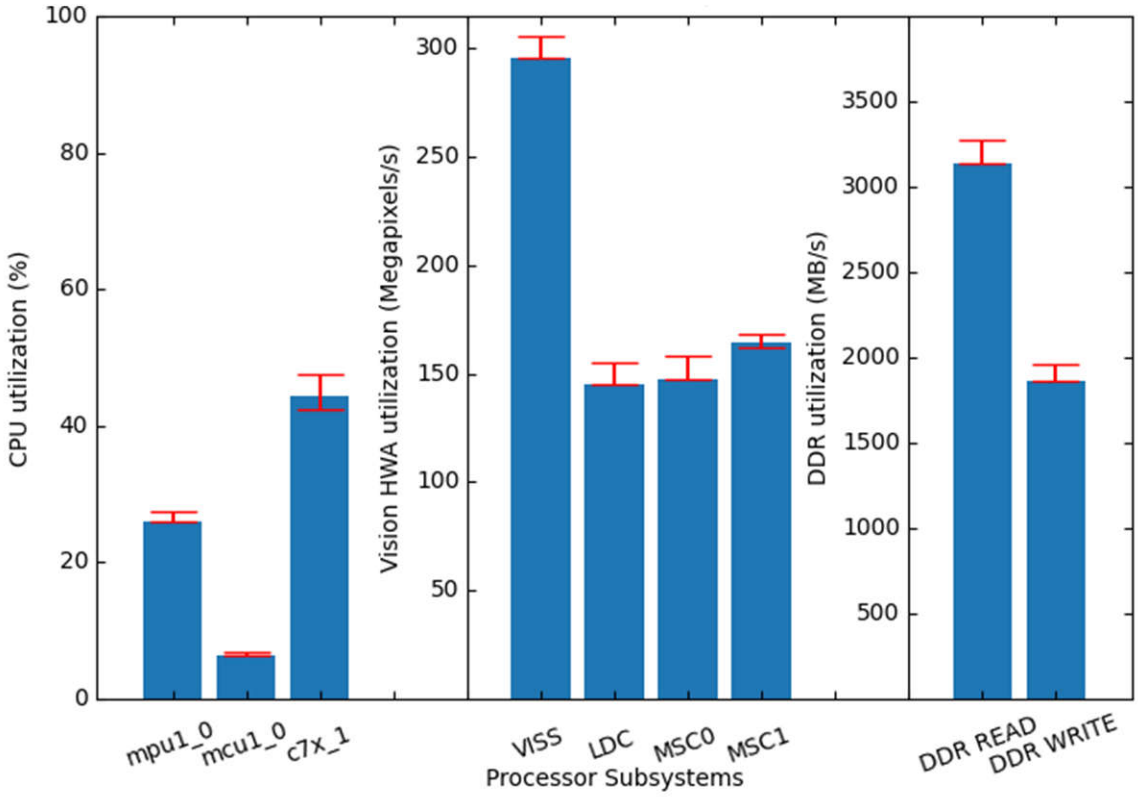

图 5-9. RGB-Ir DMS/OMS 应用的核心利用率

6 总结

本应用手册介绍与实施了利用 RGB-IR 图像传感器与 AM62A 处理器的驾驶员与乘员监控系统的参考设计，并且对其进行了基准测试。OX05B1S 摄像头模块用于创建 RGB 与红外视觉处理流水线，该流水线可利用 AM62A 上硬件加速功能实现 500 万像素用例。我们探讨了一个代表性 OMS/DMS 流水线的基本元件，并在 GStreamer 与 TI OpenVX 中测量了该等流水线的利用率与延时情况。

7 参考资料

1. 德州仪器 (TI)，AM62A 上的驾驶员和乘客监控系统技术白皮书
2. 德州仪器 (TI)，适用于 AM62A 上 Edge AI 应用的处理器 SDK Linux（网站）
3. 德州仪器 (TI)，《AM62A 上的多媒体应用》（应用手册）
4. 德州仪器 (TI)，《AM6xA ISP 调优指南》（应用手册）
5. 德州仪器 (TI)，Perf_stats 工具网站
6. 德州仪器 (TI)，EdgeAI-RGB-IR（代码库）

8 修订历史记录

注：以前版本的页码可能与当前版本的页码不同

Changes from Revision * (February 2025) to Revision A (March 2025)	Page
• 更新了 RGB 和 IR 主流与 Linux 视频器件节点之间的链接：/dev/video3 (IR) ， /dev/video4 (RGB).....	12

重要通知和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、某特定用途方面的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他功能安全、信息安全、监管或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的相关应用。 严禁以其他方式对这些资源进行复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。您应全额赔偿因在这些资源的使用中对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，TI 对此概不负责。

TI 提供的产品受 TI 的销售条款或 ti.com 上其他适用条款/TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

邮寄地址：Texas Instruments, Post Office Box 655303, Dallas, Texas 75265