天数智芯

IxStream 流分析工具包使用指南

版本：V4.0.0-MR

日期：2024.5.16

适用产品：智铠 50| 智铠 100

1 声明

1.1 版权声明

1.2 免责声明

天数智芯可以随时对本文档或本文档中描述的产品进行改进和/或更改。本文档包括与天数智芯产品有关的信息，作为说明典型应用的一种方式, 因此，不一定提供足以进行生产设计的完整信息。对于本文档中内容的准确性或完整性，天数智芯不做任何陈述或保证。

1.3 联系方式

地址：上海闵行区陈行公路 2168 号 3 幢

电话：021-68886607

网址：www.iluvatar.com

Contents

2 IxStream 流分析工具包使用指南

2.1 修订记录

- COREX01-MR400-UG11-01：2024/5/16
  - 更新Python 示例程序：
    - 增加以下示例：
      - ixstream_sample1_osd
      - ixstream_sample1_usbcam
      - ixstream_sample2
    - 删除示例 ixstream_tiler_sample
  - 增加C++ 示例程序
- COREX01-MR400-UG11-00：2024/4/3

  本次文档发布内容与 V3.2.1-MR 文档相比 (COREX01-MR321-UG11-00)，有以下更新：
  - 更新本次发布的 IxStream 版本为 v1.0.5
  - 更新兼容的天数智算软件栈的版本为 v4.0.0
  - 更新安装和配置 IxStream 小节中的 IxStream 所需依赖
  - 新增基于 gst-launch-1.0 构建的简单的 Pipeline 小节，提供一个 Pipeline 供用户快速上手 IxStream
  - 新增IxStream 示例程序 章节
  - 更新如何获取元数据 小节，更新头文件
  - 更新IxFrameMeta 元数据中关于”num_obj_meta” 上限的说明
  - 新增Gst-IxMultiStreamTiler Plugin
  - 更新各 Plugin 属性
  - 更新常见问题 章节

2.2 概述

IxStream (Iluvatar Stream) SDK 是天数智芯提供的面向 AI 智能视频分析 (IVA) 以及多传感器数据流处理的工具包，旨在支持用户能够在部署了天数智芯加速卡和软件栈的硬件平台上快速开发和部署实时数据流分析 Pipeline。

开发者只需要专注于构建核心的神经网络模型任务，无需从头开始设计端到端的数据流处理 Pipeline。一个常见的 IVA 工作流程如下图所示：

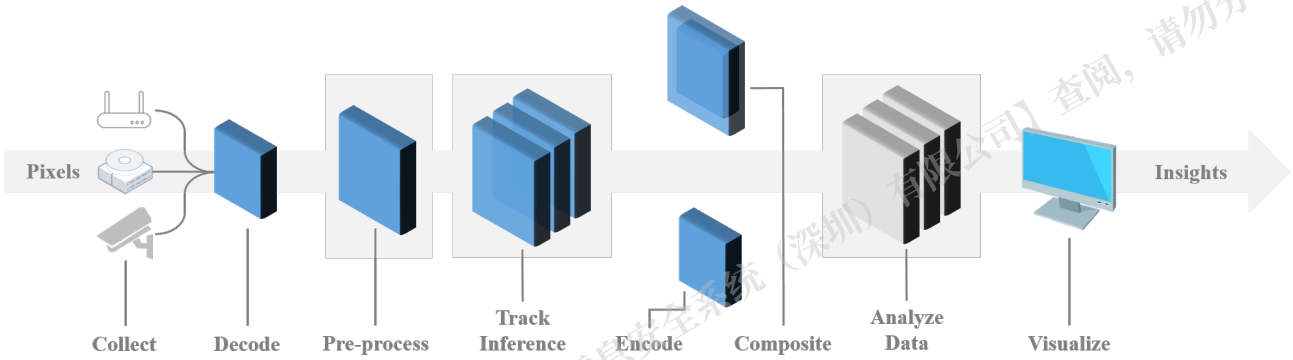

Figure 1: IVA 工作流程

2.3 IxStream 功能模块

IxStream 是基于 GStreamer 框架实现的，GStreamer 是一个用于开发流媒体应用的跨平台开源框架，采用了 Plugin 和 Pipeline 的体系结构。GStreamer 框架中的所有功能模块都被实现成可以插拔的 Plugin，并且能够很方便地插入到任意一个 Pipeline 中。

当前，IxStream 主要包含以下功能模块：

- CODEC 功能模块：基于 VPU/JPU 的视频/图片编解码方案
- Mux/Demux 功能模块：对多路输入输出做统一管理
- Infer 功能模块：基于 IGIE 推理框架对输入的图像或对象进行推理
- Tracker 功能模块：跟踪检测到的具有持久 ID 的对象
- Osd 功能模块：对输出的图片进行绘制

IxStream 核心功能由多个硬件加速器 Plugin 组成，这些 Plugin 使用 GPU、VPU、JPU 等加速器。通过在专用加速器中执行所有计算密集型操作，IxStream 可以实现视频分析应用程序的最高性能。了解更多 Plugin 信息，请参考IxStream Plugin 开发指南。

IxStream 构建在天数智算软件栈的各种库之上，例如 CUDA、CODEC、CV-CUDA、IGIE 等。IxStream 在 Plugin 中对这些库进行了封装并提供了简单的属性配置接口，使开发人员可以轻松构建视频分析管道，无需单独学习这些库。

2.4 IxStream 快速上手

本小节旨在介绍安装和配置 IxStream 的操作步骤，并提供一个基于 `gst-launch-1.0` 构建的简单的 Pipeline。

2.4.1 安装和配置 IxStream

2.4.1.1 支持的平台和操作系统

下表提供了有关安装和配置 IxStream 兼容的平台和软件版本信息：

平台/软件	版本
IxStream	1.0.5
CPU 架构	x86 ARM
天数智芯加速卡	MR-V50 MR-V100
操作系统	Ubuntu 18.04 CentOS 7.9
天数智算软件栈	4.0.0
GStreamer	1.10.5

2.4.1.2 开始之前

1. 请确认已安装天数智算软件栈、IGIE 推理框架和 IxRT 推理引擎，详情请参考《软件栈安装指南》、《IGIE 推理框架使用指南》和《IxRT 推理引擎使用指南》。
2. 请确认已安装 GStreamer 相关依赖。
   - Ubuntu 环境请使用以下命令安装 GStreamer：

```
$ apt-get install glib2.0 libgstreamer1.0 gstreamer1.0-tools gstreamer1.0-plugins-base
↪ gstreamer1.0-plugins-good gstreamer1.0-plugins-bad gstreamer1.0-plugins-ugly
↪ libgstrtspserver-1.0 libopenblas-dev
```

   - CentOS 环境请使用以下命令安装 GStreamer：

```
$ yum -y install glib2 gstreamer1 gstreamer1-plugins-base gstreamer1-plugins-good
↪ gstreamer1-plugins-bad-free gstreamer1-plugins-ugly-free gstreamer1-rtsp-server-
↪ devel
```

3. 请确认已安装 PyGObject。
   - Ubuntu 环境请使用以下命令安装 PyGObject：

```
$ apt-get install libgirepository1.0-dev libcairo2-dev
$ pip3 install pycairo PyGObject
```

   - CentOS 环境请使用以下命令安装 PyGObject：

```
$ yum install pygobject3-devel cairo-devel cairo-gobject-devel pkg-config python3-devel
$ pip3 install pycairo PyGObject==3.44.0
```

2.4.1.3 安装 IxStream

天数智芯提供了 IxStream 的 C++ 和 Python 安装包 (Python 包需要依赖 C++ 包)，请您按需向您的应用工程师获取。

通过以下命令安装 IxStream：

```
$ ./ixstream-{version}_corex-{v.r.m}_{ARCH}.run # C++ 安装
$ pip3 install pyixstream-{version}-py3-none-linux_{ARCH}.whl # Python 安装
```

2.4.1.4 设置环境变量

IxStream 安装完成后，需要设置环境变量，参考命令如下：

```
$ export GST_PLUGIN_PATH=$GST_PLUGIN_PATH:/usr/local/corex/lib64/plugins
$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/corex/lib64:/usr/local/lib/python3.7/dist-
↪ packages/tvm
```

2.4.1.5 检查环境

1. 以上环境配置完成后，您可以执行 **gst-inspect-1.0** 命令查看到 IxStream 相关的 Plugin 信息 (为防止缓存影响，执行命令前需要删除 GStreamer 的缓存)：

   ```
   $ rm -rf /root/.cache/gstreamer-1.0/ # 删除缓存
   $ gst-inspect-1.0 | grep Gst # 查看 Plugin 信息
   ```

2. 输出以下信息表示环境配置成功：

   ```
   ixstream_jpuencode: Gst-IxJpuEncode: Gst-IxJpuEncode plugin
   ixstream_osd: Gst-IxOsd: Gst-IxOsd plugin
   ixstream_streammux: Gst-IxStreamMux: Gst-IxStreamMux plugin
   ixstream_h264encode: Gst-IxH264Encode: Gst-IxH264Encode plugin
   ixstream_infer: Gst-IxInfer: Gst-IxInfer plugin
   ixstream_streamdemux: Gst-IxStreamDemux: Gst-IxStreamDemux plugin
   ixstream_tracker: Gst-IxTracker: Gst-IxTracker plugin
   ixstream_vpudecode: Gst-IxVpuDecode: Gst-IxVpuDecode plugin
   ixstream_converter: Gst-IxConverter: Gst-IxConverter plugin
   ixstream_preprocess: Gst-IxPreprocess: Gst-IxPreprocess plugin
   ```

   如果输出信息与上述有不同，请参考开始之前 和设置环境变量 小节重新进行准备工作，并检查环境是否配置成功。

2.4.2 基于 gst-launch-1.0 构建的简单的 Pipeline

2.4.2.1 配置 Plugin 属性

在构建 Pipeline 之前，您需要熟悉各个 Plugin 的属性，以针对不同的需求进行差异化配置。您可以通过 **gst-inspect-1.0** 命令查看对应的 Plugin 的属性，如下图所示：

```
root@           :~# gst-inspect-1.0 Gst-IxVpuDecode
Factory Details:
  Rank                     none (0)
  Long-name                Gst-IxVpuDecode plugin
  Klass                    Gst-IxVpuDecode functionality
  Description              gstreamer vpu decode element
  Author                   Iluvatar CoreX

Plugin Details:
  Name                     ixstream_vpudecode
  Description              Template ixvpudecode
  Filename                 /usr/local/corex/plugins/libixstream_vpudecode.so
  Version                  1.0.0
  License                  LGPL
  Source module            ixstream
  Binary package           IxStream
  Origin URL               http://                                  /ixstream.git

GObject
 +----GInitiallyUnowned
       +----GstObject
             +----GstElement
                   +----GstBaseTransform
                         +----GstIxVpuDecode

Pad Templates:
  SRC template: 'src'
    Availability: Always
    Capabilities:
      ANY

  SINK template: 'sink'
    Availability: Always
    Capabilities:
      ANY

Element has no clocking capabilities.
Element has no URI handling capabilities.

Pads:
  SINK: 'sink'
    Pad Template: 'sink'
  SRC: 'src'
    Pad Template: 'src'

Element Properties:
  drop-frame-interval : Drop frame
                        flags: readable, writable, changeable only in NULL or READY state
                        Unsigned Integer. Range: 1 - 30 Default: 1
  frame-height        : Height of each frame. This must be same as the input size.
                        flags: readable, writable, changeable only in NULL or READY state
                        Unsigned Integer. Range: 0 - 4294967295 Default: 1080
  frame-width         : Width of each frame. This must be same as the input size.
                        flags: readable, writable, changeable only in NULL or READY state
                        Unsigned Integer. Range: 0 - 4294967295 Default: 1920
  gpu-id              : Set GPU Device ID
                        flags: readable, writable, changeable only in NULL or READY state
                        Unsigned Integer. Range: 0 - 4294967295 Default: 0
  name                : The name of the object
                        flags: readable, writable
                        String. Default: "ixvpudecode0"
  parent              : The parent of the object
                        flags: readable, writable
                        Object of type "GstObject"
  qos                 : Handle Quality-of-Service events
                        flags: readable, writable
                        Boolean. Default: false
  video-format        : video format of each frame.
                        flags: readable, writable, changeable only in NULL or READY state
                        Enum "Gst-IxVpuDecodeVideoFormat" Default: 0, "h264"
                           (0): h264             - H264 Format
                           (12): h265            - H265 Format
                           (14): avs2            - AVS2 Format
```

Figure 2: 查看对应 Plugin 属性

更多 IxStream Plugin 的属性说明，请参考IxStream Plugin 开发指南。

2.4.2.2 运行 Pipeline

以一个简单的编解码示例为例，运行以下命令即可运行 Pipeline 程序：

```
$ gst-launch-1.0 filesrc location="./sample_720p.h264" ! Gst-IxVpuDecode frame-width=1280 frame-
↪  height=720 video-format=0 ! Gst-IxStreamMux live-source=false ! Gst-IxJpuEncode save-file-
↪  path="./jpeg_out" ! fakesink
```

2.4.3 运行示例程序

请向您的应用工程师获取 IxStream Samples 运行示例，并执行相关示例程序。更多示例程序说明，请参考IxStream 示例程序。

2.5 IxStream 示例程序

请向您的应用工程师获取 IxStream Samples 运行示例。

2.5.1 Python 示例程序

ixstream_samples 中提供的 Python 示例程序简要说明如下。

示例程序	所在仓库中的目录	说明
ixstream_sample1	ixstream_samples/python/ixstream_sample1	基于 primary_detector 进行检测的简单示例，并对检测到的对象进行画框等操作。输入为 720P 的 H264 视频文件，输出为 JPEG 图片
ixstream_sample1_osd	ixstream_samples/python/ixstream_sample1_osd	基于 ixstream_sample1 增加了目标物体的统计功能，并将结果显示在图片上
ixstream_sample1_rtsp_out	ixstream_samples/python/ixstream_sample1_rtsp_out	基于 ixstream_sample1，将输出更改为 RTSP 流
ixstream_sample1_usbcam	ixstream_samples/python/ixstream_sample1_usbcam	基于 ixstream_sample1 将输入更改为摄像头输入
ixstream_sample2	ixstream_samples/python/ixstream_sample2	基于 ixstream_sample1 增加 tracker 和二级推理功能
ixstream_sample3	ixstream_samples/python/ixstream_sample3	基于 ixstream_sample1，增加多路输入/输出支持
ixstream_sample4	ixstream_samples/python/ixstream_sample4	基于 ixstream_sample3，将输出更改为输出到屏幕

示例程序	所在仓库中的目录	说明
ixstream_sample5	ixstream_samples/python/ixstream_sample5	基于 ixstream_sample3，支持更多的输入输出 (输入：H264 视频文件，MP4 视频文件，RTSP 流；输出：JPEG 图片，H264 视频文件)

2.5.2 C++ 示例程序

ixstream_samples 中提供的 C++ 示例程序简要说明如下。

示例程序	所在仓库中的目录	说明
ixstream_sample1	ixstream_samples/C++/ixstream_sample1	基于 primary_detector 进行检测的简单示例，并对检测到的对象进行画框等操作。输入为 720P 的 H264 视频文件，输出为 JPEG 图片
ixstream_sample1_osd	ixstream_samples/C++/ixstream_sample1_osd	基于 ixstream_sample1 增加了目标物体的统计功能，并将结果显示在图片上
ixstream_sample1_rtsp_out	ixstream_samples/C++/ixstream_sample1_rtsp_out	基于 ixstream_sample1，将输出更改为 RTSP 流
ixstream_sample1_usbcam	ixstream_samples/C++/ixstream_sample1_usbcam	基于 ixstream_sample1 将输入更改为摄像头输入
ixstream_sample2	ixstream_samples/C++/ixstream_sample2	基于 ixstream_sample1 增加 tracker 和二级推理功能
ixstream_sample3	ixstream_samples/C++/ixstream_sample3	基于 ixstream_sample1，增加多路输入/输出支持
ixstream_sample4	ixstream_samples/C++/ixstream_sample4	基于 ixstream_sample3，将输出更改为输出到屏幕

2.6 IxStream Plugin 开发指南

本小节旨在为开发人员介绍 IxStream 支持的 Plugin，并提供这些 Plugin 的输入和输出等信息，另外，在本小节中，您还可以了解 IxStream SDK 中使用的元数据。

下表列出了当前 IxStream SDK 中提供的 Plugin：

Plugin	说明
Gst-IxVpuDecode	基于 VPU 硬件加速的视频解码插件
Gst-IxJpuEncode	基于 JPU 硬件加速的 JPEG 图片编码插件
Gst-IxH264Encode	基于 CPU/GPU 实现的 H264 视频编码插件
Gst-IxConverter	格式转换插件
Gst-IxStreamMux	多路输入合并成一个 batch
Gst-IxStreamDemux	一个 batch 分解成对应的多路
Gst-IxInfer	基于 IGIE 实现的模型推理插件
Gst-IxTracker	用于跟踪检测到的具有持久 ID 的对象
Gst-IxOsd	对输入的帧进行绘制
Gst-IxMultiStreamTiler	将多路输入图片拼接成一张图片

2.6.1 IxStream SDK 中的元数据

Gst Buffer 是 GStreamer 中数据传输的基本单位，每个 Gst Buffer 都有关联的元数据。

IxStream SDK 附加的主要元数据对象如下图所示：

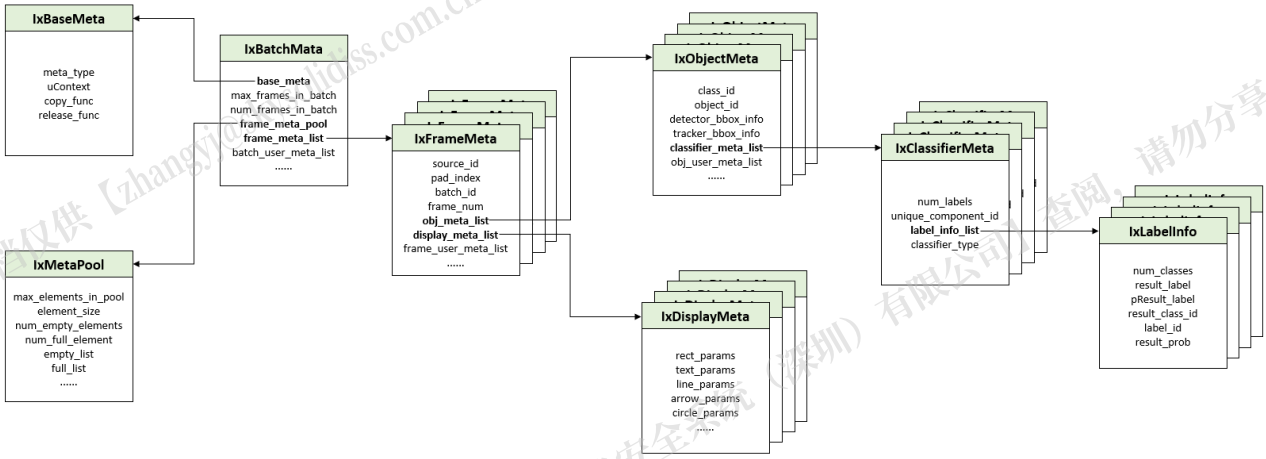

Figure 3: IxStream SDK 附加的主要元数据对象

您可以通过相关接口获取这些元数据 (参考如何获取元数据)，同时也可以通过您自定义的接口 (如 xxx_user_meta_list) 向其中添加自定义元数据 (参考如何添加用户自定义元数据)。

了解更多关于 Gst Buffer 的信息，请参考 GStreamer 文档。

2.6.1.1 如何获取元数据

C++ 获取元数据方式

1. 引入必要的头文件：

```
#include<gst/gst.h>
#include<gst_ixstream_meta.h>
```

2. 通过 IxStream 对象的指针，获取该对象的 sinkpad (或 srcpad) 的指针：

```
GstPad* sink_pad = gst_element_get_static_pad(ixstream_obj, "sink");
或
GstPad* src_pad = gst_element_get_static_pad(ixstream_obj, "src");
```

3. 向 sinkpad (或 srcpad) 添加 probe 函数 (指定为 GST_PAD_PROBE_TYPE_BUFFER 类型)：

```
gst_pad_add_probe(sink_pad, GST_PAD_PROBE_TYPE_BUFFER,  sink_pad_buffer_probe, NULL, NULL)
```

4. 在 probe 函数中获取IxBatchMeta 对象：

```
static GstPadProbeReturn sink_pad_buffer_probe(GstPad *pad, GstPadProbeInfo *info, gpointer
↪  u_data) {
   GstBuffer *buf = (GstBuffer *) info->data;
   IxBatchMeta *pBatchMeta = gst_buffer_get_ix_batch_meta(buf);
}
```

Python 获取元数据方式

1. 导入必要库：

```
import gi
gi.require_version('Gst', '1.0')
from gi.repository import GObject, Gst
import pyixstream
```

2. 通过 IxStream 对象，获取该对象的 sinkpad (或 srcpad)：

```
sink_pad = ixstream_obj.get_static_pad('sink');
或
src_pad = ixstream_obj.get_static_pad('src');
```

3. 向 sinkpad (或 srcpad) 添加 probe 函数 (指定为 GST_PAD_PROBE_TYPE_BUFFER 类型)：

```
sink_pad.add_probe(Gst.PadProbeType.BUFFER,  sink_pad_buffer_probe, NULL)
```

4. 在 probe 函数中获取IxBatchMeta 对象：

```
def sink_pad_buffer_probe(pad, info, gpointer u_data):
    gst_buf = info.get_buffer()
    batch_meta = pyixstream.gst_buffer_get_ix_batch_meta(hash(gst_buf))
```

2.6.1.2 如何添加用户自定义元数据

C++ 添加用户自定义元数据方式

1. 引入必要的头文件：

```
#include<gst/gst.h>
#include<gst_ixstream_meta.h>
```

2. 获取IxBatchMeta 对象指针：

```
static GstPadProbeReturn sink_pad_buffer_probe(GstPad *pad, GstPadProbeInfo *info, gpointer
↪  u_data) {
   GstBuffer *buf = (GstBuffer *) info->data;
   IxBatchMeta *pBatchMeta = gst_buffer_get_ix_batch_meta(buf);
}
```

3. 通过IxBatchMeta 对象，创建一个 IxUserMeta 对象：

```
IxUserMeta *pUserMeta = ix_acquire_user_meta_from_pool(pBatchMeta);
```

4. 指定 IxUserMeta 对象的拷贝函数、析构函数及数据：

```
pUserMeta->base_meta.copy_func = user_metadata_copy_func;
pUserMeta->base_meta.release_func = user_metadata_release_func;
pUserMeta->user_meta_data = user_meta_data_ptr;
```

5. 将 IxUserMeta 对象添加到相应的元数据上 (如IxFrameMeta)：

```
ix_add_user_meta_to_frame(pFrameMeta, pUserMeta);
```

Python 添加用户自定义元数据方式

1. 导入必要库：

```
import gi
gi.require_version('Gst', '1.0')
from gi.repository import GObject, Gst
import pyixstream
```

2. 获取IxBatchMeta 对象指针：

```
def sink_pad_buffer_probe(pad, info, gpointer u_data):
    gst_buf = info.get_buffer()
    batch_meta = pyixstream.gst_buffer_get_ix_batch_meta(hash(gst_buf))
```

3. 通过IxBatchMeta 对象，创建一个 IxUserMeta 对象：

```
user_meta = pyixstream.ix_acquire_user_meta_from_pool(batch_meta)
```

4. 指定 IxUserMeta 对象的拷贝函数、析构函数及数据：

```
user_meta.base_meta.copy_func = user_metadata_copy_func
user_meta.base_meta.release_func = user_metadata_release_func
user_meta.user_meta_data = user_meta_data_ptr
```

5. 将 IxUserMeta 对象添加到相应的元数据上 (如IxFrameMeta)：

```
pyixstream.ix_add_user_meta_to_frame(frame_meta, user_meta)
```

2.6.1.3 IxBatchMeta

IxStream 使用可扩展的元数据标准结构，IxBatchMeta 是其中最重要的元数据结构，可以在Gst-IxStreamMux Plugin 中创建。

IxBatchMeta 包含的元素及说明如下：

- base_meta：IxStream 基础元数据结构，包含以下属性：
  - meta_type
  - uContext
  - copy_func
  - release_func
- max_frames_in_batch：一个 batch 中可填充的最大 frame 个数，即 batch_size。该值在Gst-IxStreamMux Plugin 中设置，由输入路数决定
- num_frames_in_batch：当前 batch 中实际已填充的 frame 个数。该值在Gst-IxStreamMux Plugin 中设置
  - 当出现 Timeout 时，num_frames_in_batch <= max_frame_in_batch
  - 当非 Timeout 时，num_frames_in_batch = max_frame_in_batch
- frame_meta_pool：IxFrameMetaList 对应的 IxMetaPool，该 pool 在Gst-IxStreamMux Plugin 中初始化
- obj_meta_pool：IxObjectMetaList 对应的 IxMetaPool，该 pool 在Gst-IxStreamMux Plugin 中初始化
- classifier_meta_pool：IxClassifierMetaList 对应的 IxMetaPool，该 pool 在Gst-IxStreamMux Plugin 中初始化
- display_meta_pool：IxDisplayMetaList 对应的 IxMetaPool，该 pool 在Gst-IxStreamMux Plugin 中初始化
- user_meta_pool：IxUserMetaList 对应的 IxMetaPool，该 pool 在Gst-IxStreamMux Plugin 中初始化
- label_info_meta_pool：label_info_meta 对应的 IxMetaPool，该 pool 在Gst-IxStreamMux Plugin 中初始化
- frame_meta_list：指向 IxFrameMetaList 的指针
- batch_user_meta_list：指向 IxUserMetaList 的指针
- meta_mutex：MetaData 操作锁

2.6.1.4 IxFrameMeta

IxFrameMeta 包含的元素及说明如下：

- base_meta：IxStream 基础元数据结构

- pad_index：当前 frame 对应的 sink_pad index，该值与 source_id 一一对应
- batch_id：当前 frame 在对应 batch 中的 index
- source_frame_width：当前 frame 对应输入源的 frame_width
- source_frame_height：当前 frame 对应输入源的 frame_height
- num_obj_meta：当前 frame 包含的目标对应个数，该值最大为 100，超过该值赋值 100 并抛出 Warning 信息
- obj_meta_list：指向 IxObjectMetaList 的指针
- display_meta_list：指向 IxDisplayMetaList 的指针
- frame_user_meta_list：指向 IxUserMetaList 的指针
- pipeline_width：streammux 后的 frame_width
- pipeline_height：streammux 后的 frame_height

2.6.1.5 IxObjectMeta

IxObjectMeta 包含的元素及说明如下：

- base_meta：IxStream 基础元数据结构
- class_id：当前对象的 class index
- detector_bbox_info：检测到的目标对象的 bbox 信息，包含宽、高及位置信息
- confidence：保存由推理组件设置的对象的置信度值
- classifier_meta_list：指向 IxClassifierMetaList 的指针
- obj_user_meta_list：指向 IxUserMetaList 的指针

2.6.1.6 IxClassifierMeta

IxClassifierMeta 包含的元素及说明如下：

- base_meta：IxStream 基础元数据结构
- num_labels：分类器输出的分类总数
- label_info_list：指向 IxLabelInfoList 的指针

2.6.1.7 IxLabelInfoList

IxLabelInfoList 包含的元素及说明如下：

- base_meta：IxStream 基础元数据结构
- pResult_label：指向保存当前分类 label 信息的指针
- label_id：当前分类的 index

2.6.1.8 IxDisplayMeta

IxDisplayMeta 包含的元素及说明如下：

- base_meta：IxStream 基础元数据结构
- num_rects：当前帧包含的目标框的个数
- num_labels：当前帧包含的目标 label 的个数

- num_lines：当前帧包含的线的个数
- num_arrows：当前帧包含的箭头个数
- num_circles：当前帧包含的圆的个数
- rect_params：当前帧中目标框的位置信息数组
- text_params：当前帧中用于绘制文本框的参数信息数组
- line_params：当前帧中用于绘制多边形的线的参数信息数组
- arrow_params：当前帧中用于绘制箭头的参数信息数组
- circle_params：当前帧中用于绘制圆的参数信息数组

2.6.2 Gst-IxVpuDecode

Gst-IxVpuDecode Plugin 提供视频流硬解码功能。

支持输入的视频流格式有：

- H264
- H265
- AVS2

支持输出的数据格式有：

- YUV420

输入与输出

- 输入：H264、H265、AVS2 视频流
- 输出：基于帧的 YUV420 数据

功能

下表总结了 Gst-IxVpuDecode Plugin 的功能：

功能	发布版本
H264 格式的硬件解码	IxStream 1.0
H265 格式的硬件解码	IxStream 1.0
AVS2 格式的硬件解码	IxStream 1.0
对解码后的帧按一定比例丢帧	IxStream 1.0

属性

下表总结了 Gst-IxVpuDecode Plugin 的 Gst 属性：

属性	说明	类型与范围
gpu-id	用于操作的 GPU ID，该值需与其他 Plugin 对应值保持一致	Unsigned int，0~4294967295，默认值：0
frame-width	输入视频流的宽 (需设置为输入视频对应真实值，否则会丢弃当前帧)	Unsigned int，0~4294967295，默认值：1920
frame-height	输入视频流的高 (需设置为输入视频对应真实值，否则会丢弃当前帧)	Unsigned int，0~4294967295，默认值：1080
video-format	输入视频流的格式 (需设置为输入视频对应真实值，否则解码结果异常)	枚举，{0:H264，12:H265，14:AVS2}，默认值：0
drop-frame-interval	丢帧频率 (每 N 帧输出一帧)	Unsigned int，1~30 (大于 30 设置为 30)，默认值：1

2.6.3 Gst-IxJpuEncode

Gst-IxJpuEncode Plugin 提供 JPEG 图片的硬解码功能。

支持输入的视频流格式有：

- YUV400
- YUV420
- YUV422
- YUV440
- YUV444

输入与输出

- 输入：基于帧的 YUV400、YUV420、YUV422、YUV440、YUV444 格式数据
- 输出：基于帧的 JPEG 格式数据

功能

下表总结了 Gst-IxJpuEncode Plugin 的功能：

功能	发布版本
JPEG 格式的硬件解码	IxStream 1.0

属性

下表总结了 Gst-IxJpuEncode Plugin 的 Gst 属性：

属性	说明	类型与范围
gpu-id	用于操作的 GPU ID，该值需与其他 Plugin 对应值保持一致，否则抛出异常	Unsigned int，0~4294967295，默认值：0
save-file-path	保存的 JPEG 文件目录(该值必须设置，否则抛出异常)	String，无默认值

2.6.4 Gst-IxH264Encode

Gst-IxH264Encode Plugin 提供 H264 视频的编码功能 (包含 CPU 实现和 CUDA 加速实现)。

支持输入的视频流格式有：

- YUV420

输入与输出

- 输入：基于帧的 YUV420 格式数据
- 输出：H264 视频文件

功能

下表总结了 Gst-IxH264Encode Plugin 的功能：

功能	发布版本
基于 CPU 实现的 H264 视频编码功能	IxStream 1.0
CUDA 加速的 H264 视频编码功能	IxStream 1.0
CPU 亲和性设定	IxStream 1.0

属性

下表总结了 Gst-IxH264Encode Plugin 的 Gst 属性：

属性	说明	类型与范围
gpu-id	用于操作的 GPU ID，该值需与其他 Plugin 对应值保持一致，否则抛出异常	Unsigned int，0~4294967295，默认值：0
save-video -path	保存的视频文件目录绝对路径	String，无默认值

属性	说明	类型与范围
save-video -name	保存的视频文件的文件名	String，默认值：h264out.h264
threads-num	CPU 编码部分并行线程数	Unsigned int，1~128，默认值：1
use-cuda	是否使用 CUDA 加速	Bool，true/false，默认值：false
cpu-affinity	CPU 亲和性 (以”,” 号进行分割)	String，默认值：all

2.6.5 Gst-IxConverter

Gst-IxConverter Plugin 提供基于帧数据的格式转换功能。

支持以下格式间的转换：

- YUV420
- RGB

输入与输出

- 输入：基于帧的 YUV420、RGB 格式数据
- 输出：基于帧的 YUV420、RGB 格式数据

功能

下表总结了 Gst-IxConverter Plugin 的功能：

功能	发布版本
YUV420 -> RGB 格式转换	IxStream 1.0
RGB -> YUV420 格式转换	IxStream 1.0

属性

下表总结了 Gst-IxConverter Plugin 的 Gst 属性：

属性	说明	类型与范围
gpu-id	用于操作的 GPU ID，该值需与其他 Plugin 对应值保持一致，否则抛出异常	Unsigned int，0~4294967295，默认值：0

2.6.6 Gst-IxStreamMux

Gst-IxStreamMux Plugin 将多个输入源的数据合并成一个 batch 数据。如果输入源的分辨率不同，会对输入做 resize 并统一成同一分辨率 (可以使用 width 和 height 属性指定此分辨率)。如果 batch 中数据已满或超时达到 batched-push-timeout (收集到 batch 中的第一个缓冲区时，超时开始运行) 时，Gst-IxStreamMux 会将 batch 数据向下游推送。当 live-source 属性设置为 true 时，Gst-IxStreamMux 将执行实时非阻塞策略 (输出 buffer 满则自动丢弃，该策略适用于 rtsp 等实时输入流)。当 live-source 属性设置为 false 时，Gst-IxStreamMux 将执行阻塞策略 (输出 buffer 满则进行等待，该策略适用于视频文件输入流)。

输入与输出

- 输入：Gst buffer
- 输出：Gst buffer

功能

下表总结了 Gst-IxStreamMux Plugin 的功能：

功能	发布版本
可配置的批处理超时	IxStream 1.0
允许具有不同分辨率的多个输入流	IxStream 1.0
允许具有不同帧率的多个输入流	IxStream 1.0
可以 resize 到用户指定的分辨率	IxStream 1.0

属性

下表总结了 Gst-IxStreamMux Plugin 的 Gst 属性：

属性	说明	类型与范围
gpu-id	用于操作的 GPU ID，该值需与其他 Plugin 对应值保持一致，否则抛出异常	Unsigned int，0~4294967295，默认值：0
width	mux 后的帧的 width	Unsigned int，0~4294967295，默认值：1920
height	mux 后的帧的 height	Unsigned int，0~4294967295，默认值：1080
live-source	输入源是否为实时源	Bool，true/false，默认值：true

属性	说明	类型与范围
batched-push-timeout	超时时间，单位为 ms	Unsigned int，0~4294967295，默认值：4294967295
enable-padding	resize 操作是否执行 padding	Bool，true/false，默认值：false

2.6.7 Gst-IxStreamDemux

Gst-IxStreamDemux Plugin 将 batch 帧 demux 到单独的缓冲区中。它为 batch 中的每个帧创建一个单独的 Gst 缓冲区，它不复制视频帧。每个 Gst 缓冲区都包含一个指向 batch 中相应帧的指针。

输入与输出

- 输入：Gst buffer
- 输出：Gst buffer

功能

下表总结了 Gst-IxStreamDemux Plugin 的功能：

功能	发布版本
根据输入路数将 batched 输入分成多路输出	IxStream 1.0

属性

下表总结了 Gst-IxStreamDemux Plugin 的 Gst 属性：

属性	说明	类型与范围
gpu-id	用于操作的 GPU ID，该值需与其他 Plugin 对应值保持一致，否则抛出异常	Unsigned int，0~4294967295，默认值：0

2.6.8 Gst-IxInfer

Gst-IxInfer Plugin 使用 IGIE 推理框架对输入数据进行推理。

Gst-IxInfer Plugin 目前支持以下网络的推理：

- 多类物体检测，已支持以下模型：

  - YOLOv5
  - YOLOv7
  - YOLOv8
  - YOLOxl
  - primary_detector 示例模型
- 多标签分类

Gst-IxInfer Plugin 可以在两种模式下工作：

- primary：在全帧上运行 (比如在一级推理中对所有输入的帧进行目标对象检测)
- secondary：对上游组件添加到元数据中的对象进行操作 (比如在二级推理中对上游一级推理中检测到的目标对象进行进一步分类)

输入与输出

- 输入：Gst buffer、元数据
- 输出：Gst buffer、元数据

功能

下表总结了 Gst-IxInfer Plugin 的功能：

功能	发布版本
支持 primary 模式	IxStream 1.0
支持 secondary 模式	IxStream 1.0
支持 IGIE Infer	IxStream 1.0
支持 IxRT Infer	IxStream 1.0

配置文件说明

下表总结了 Gst-IxInfer Plugin 的配置文件说明：

属性	说明	类型与范围	适应网络模式
process-mode	推理模式，1：primary；2：secondary	Unsigned int，1/2，默认值：1	all
model-engine-file	IGIE/IxRT 生成的模型引擎文件，该值必须设置，否则抛出异常	String，无默认值	all
output-blob-names	模型输出名，IGIE 无需设置该属性 (不支持解析输出名)，IxRT 必须设置正确的输出名	String，无默认值	all

属性	说明	类型与范围	适应网络模式
infer-dims	模型输入维度 (C,H,W) 信息，该值必须设置，否则抛出异常	Vector3_i，默认值：{0, 0, 0}	all
network-input-order	模型输入类型，0：NCHW；1：NHWC	Unsigned int，0/1，默认值：0	all
batch-size	模型 batch 大小，该值必须设置，否则抛出异常	Unsigned int，0~4294967295，默认值：0	all
net-scale-factor	模型输入图片缩放因子，用于预处理	Float，>0.0，默认值：1.0	all
offsets	模型输入图片偏移因子，用于预处理	Vector3_f，默认值：{0.0,0.0,0.0}	all
maintain-aspect-ratio	resize 操作是否保持纵横比，用于预处理	Bool，true/false，默认值：true	all
symmetric-padding	resize 操作是否执行对称填充，用于预处理	Bool，true/false，默认值：false	all
num-detected-classes	检测网络的类别数，该值必须设置，否则抛出异常	Unsigned int，0~4294967295，默认值：0	Detector
parse-bbox-func-name	自定义边界框解析函数名称，用于后处理，该值必须设置，否则抛出异常	String，无默认值	Detector
custom-lib-path	自定义的模型后处理库的路径，该值必须设置，否则抛出异常	String，无默认值	all
filter-out-class-ids	过滤掉检测到的属于指定 class-id 的对象，如果为空则不进行过滤	Vector<gint>，无默认值	Detector
operate-on-class-ids	对上游模型检测到的指定 class-ids 的对象进行操作	Unsigned int，0~4294967295，默认值：0	secondary
labelfile-path	class 与 lable 的映射文件，支持的分隔符有：分号，回车	String，无默认值	Detector

2.6.9 Gst-IxTracker

Gst-IxTracker Plugin 用于跟踪检测到的具有持久 ID 的对象。

当前已实现的 tracker 算法如下：

- deepsort

输入与输出

- 输入：Gst buffer、元数据
- 输出：Gst buffer、元数据

功能

下表总结了 Gst-IxTracker Plugin 的功能：

功能	发布版本
支持 deepsort 算法	IxStream 1.0

属性

下表总结了 Gst-IxTracker Plugin 的 Gst 属性：

属性	说明	类型与范围
gpu-id	用于操作的 GPU ID，该值需与其他 Plugin 对应值保持一致，否则抛出异常	Unsigned int，0~4294967295，默认值：0
config-file-path	配置文件绝对路径，该值必须设置，否则抛出异常	String，无默认值

配置参数说明

下表总结了 Gst-IxTracker 中常见模块的配置参数：

模块	参数	说明	类型与范围	默认值	备注
Target Management	probationAge	试用期长度 (以帧数为单位)	Int，>=0	5	
Target Management	maxShadowTrackingAge	阴影跟踪最大长度	Int，>=0	38	

模块	参数	说明	类型与范围	默认值	备注
ReID	reidType	Re-ID 网络的类型 {DUMMY=0, NvDEEP-SORT=1, Reid based reassoc=2, DEEPSORT and reid based reassoc=3}	Int，[0, 3]	0	
ReID	batchSize	Re-ID 网络的 batch 大小	Int，>0	1	
ReID	reidFeatureSize	Re-ID 的特征大小	Int，>0	128	
ReID	reidHistorySize	特征库的大小，即为一个跟踪器保留的 Re-ID 特征的最大数量	Int，>0	100	
ReID	resize	Re-ID 网络预处理时 resize 的尺寸信息 (包含 W/H 两个维度)	Vector2_i	[0, 0]	当设置该属性时，先 resize 成对应大小再 crop 成 inferDims 的输入，否则直接 resize 成 inferDims 的输入
ReID	inferDims	Re-ID 网络基于 inputOrder 的输入维度信息 (仅包含 C/H/W 三个维度)	Vector3_i	[128, 64, 3]	
ReID	networkMode	Re-ID 网络推理精度 {FP32=0, FP16=1, INT8=2}	Int，[0, 2]	0	
ReID	inputOrder	Re-ID 网络模型输入类型 {NCHW=0, NHWC=1}	Int，[0, 1]	0	

模块	参数	说明	类型与范围	默认值	备注
ReID	colorFormat	Re-ID 网络输入颜色格式 {RGB=0, BGR=1}	Int，[0, 1]	0	
ReID	offsets	Re-ID 网络输入数据的偏移值 (基于每个颜色通道)	Vector3_f	[0.0, 0.0, 0.0]	
ReID	netScaleFactor	Re-ID 网络输入的缩放因子	Float，>0.0	1.0	
ReID	normalize	Re-ID 网络输入的标准化参数	Vector3_f	[1.0, 1.0, 1.0]	
ReID	keepAspc	调整输入对象大小到 Re-ID 网络时是否保持长宽比	Bool，true/false	true	
ReID	inputBlobName	Re-ID 网络输入层名称	String	”images”	
ReID	modelEngineFile	IGIE 生成的 Re-ID 模型引擎文件绝对路径	String	””	

2.6.10 Gst-IxOsd

Gst-IxOsd Plugin 可以对输入的帧绘制边界框、文本、箭头、线条、圆形和多边形。

该 Plugin 支持以下两种处理模式：

- CPU 处理模式 (通过 OpenCV 库实现，不支持中文字体)
- GPU 处理模式 (通过 NvOSD 库实现，支持中文字体)

输入与输出

- 输入：Gst buffer、元数据
- 输出：Gst buffer、元数据

功能

下表总结了 Gst-IxOsd Plugin 的功能：

功能	发布版本
支持 CPU 处理模式	IxStream 1.0

功能	发布版本
支持 GPU 处理模式	IxStream 1.0
支持绘制箭头	IxStream 1.0
支持绘制圆	IxStream 1.0
支持绘制边界框	IxStream 1.0
支持绘制多边形	IxStream 1.0
支持绘制文本	IxStream 1.0
GPU 处理模式，支持中文字体	IxStream 1.0

属性

下表总结了 Gst-IxOsd Plugin 的 Gst 属性：

属性	说明	类型与范围
gpu-id	用于操作的 GPU ID，该值需与其他 Plugin 对应值保持一致，否则抛出异常	Unsigned int，0~4294967295，默认值：0
process-mode	控制相关操作使用的硬件平台	枚举，{0:CPU，1:GPU}，默认值：0
display-bbox	是否显示边界框	Bool，true/false，默认值：false
display-text	是否显示文本	Bool，true/false，默认值：false
font-file-path	中文字体路径，仅在 GPU 模式下支持	String，默认值：/usr/share/fonts/ixstream/WenQuanYiMicroHei.ttf

2.6.11 Gst-IxMultiStreamTiler

Gst-IxMultiStreamTiler Plugin 可以将多个输入源的图片数据拼接成一个图片数据。拼接后的图片大小由属性 width 和 height 指定。拼接后的图片布局遵循以下原则：

- 行优先原则
- 当输入源个数 <= rows*columns 时，不足的位置做 padding
- 当输入源个数 > rows*columns 时，忽略传入的 rows 和 columns 参数，重新对 rows 和 columns 数进行设置，设置规则如下：
  - rows = (int)(sqrt(input_channels))

- columns = (int)((input_channels + rows - 1) / rows)

输入与输出

- 输入：Gst buffer
- 输出：Gst buffer

功能

下表总结了 Gst-IxMultiStreamTiler Plugin 的功能：

功能	发布版本
将多个输入源的图片数据拼接成一个图片数据	IxStream 1.0

属性

下表总结了 Gst-IxMultiStreamTiler Plugin 的 Gst 属性：

属性	说明	类型与范围
gpu-id	用于操作的 GPU ID，该值需与跟其他 Plugin 对应值保持一致，否则抛出异常	Unsigned Int，0~4294967295，默认值：0
width	拼接后的帧的 width	Unsigned int，0~4294967295，默认值：1920
height	拼接后的帧的 height	Unsigned int，0~4294967295，默认值：1080
rows	拼接的行数	Unsigned int，0~4294967295，默认值：1
columns	拼接的列数	Unsigned int，0~4294967295，默认值：1

2.7 常见问题

2.7.1 环境相关问题

问题 1：gi 版本不匹配

问题描述

在执行 **import gi** 时报以下错误：

```
ImportError: cannot import name '_gi' from 'gi' (/usr/lib/python3/dist-packages/gi/
↪  __init__.py)
```

解决方法 1

执行以下命令解决问题：

```
# x86 环境执行以下命令
$ sudo ln -s /usr/lib/python3/dist-packages/gi/_gi.cpython-{36m,37m}-x86_64-linux-gnu.so

# ARM 环境执行以下命令
$ sudo ln -s /usr/lib/python3/dist-packages/gi/_gi.cpython-{36m,37m}-aarch64-linux-gnu.so
```

解决方法 2

执行以下命令解决问题：

```
$ python3 -m pip install --ignore-installed PyGObject
```

问题 2：GStreamer 无法加载 Gst-IxInfer Plugin

问题描述

在执行 **gst-inspect-1.0** 时报以下错误：

```
(gst-plugin-scanner:2280): GStreamer-WARNING **: 06:27:46.886: Failed to load plugin '/opt/
↪  ixstream/plugins/libixstream_infer.so': libtvm.so: cannot open shared object file: No
↪  such file or directory
```

解决方法

参考以下命令将 `libtvm.so` 的路径添加到 LD_LIBRARY_PATH 变量中：

```
$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib/python3.7/dist-packages/tvm/cpp
```

问题 3：PyGObject 安装失败

问题描述

在执行 **pip3 install PyGObject** 时报以下错误：

```
Requested 'glib-2.0 >=2.64.0' but version of Glib is 2.56.1
```

解决方法

安装 PyGObject 时指定版本安装，相关命令如下：

```
$ pip3 install PyGObject==2.44.0
```

2.8 IxStream 与 DeepStream 配置接口对比一览表

IxStream	DeepStream	说明
Gst-IxVpuDecode	Gst-nvvideo4linux2(decoder)	
Gst-IxH264Encode	Gst-nvvideo4linux2(encoder)	Gst-nvvideo4linux2 使用专用的硬件加速实现，Gst-IxH264Encode 使用 CUDA Core 实现
Gst-IxJpuEncode	Gst-nvjpegenc	
Gst-IxConverter	Gst-nvvideoconvert	
Gst-IxStreamMux	Gst-nvstreammux	
Gst-IxStreamDemux	Gst-nvstreamdemux	
Gst-IxInfer	Gst-nvinfer	
Gst-IxTracker	Gst-nvtracker	Gst-IxTracker 目前仅支持 deepsort 一种算法
Gst-IxOsd	Gst-nvdsosd	

3 商标声明

- 天数智芯、天数智芯 logo、Iluvatar CoreX 等商标、标识、组合商标为上海天数智芯半导体有限公司之注册商标或商标，受法律保护。
- 除了天数智芯的注册商标外，本内容中使用的其他产品名称及标志仅用于识别目的，该等名称及标志可能是归属于其各自公司的商标。我们否认对该等名称及标志的所有权利。
- CentOS 标识为 Red Hat 公司的商标。
- Docker 为 Docker 公司在美国和其他国家的商标或注册商标。
- Linux 为 Linus Torvalds 在美国和其它国家的注册商标。
- NVIDIA 和 CUDA 为 NVIDIA 公司在美国和/或其它国家的商标和/或注册商标。
- PyTorch 为 Facebook 公司的商标。

- TensorFlow 为 Google 公司的商标。
- Ubuntu 为 Canonical 公司的注册商标。