MULTIMEDIA 使用手册

发布 0.10.0

SOPHGO

2024 年 06 月 19 日

目录

CHAPTER 1

声明

注意

您购买的产品、服务或特性等应受算能商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，算能对本文档内容不做任何明示或默示的声明或保证。由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

技术支持

地址 北京市海淀区丰豪东路 9 号院中关村集成电路设计园（ICPARK）1 号楼

邮编 100094

网址 https://www.sophgo.com/

邮箱 sales@sophgo.com

电话 +86-10-57590723 +86-10-57590724

CHAPTER 2

Release note

版本	发布日期	说明
V0.1.0	2022.08.10	第一次发布，包含 sophon ffmpeg 和 sophon opencv
V0.2.4	2022.08.30	正式发布，增补文档
V0.6.0	2023.02.28	增加新功能：水印、马赛克、统一内存分配方式
V0.6.2	2023.03.31	增补 sample 安装和使用方法
V0.7.0	2023.05.16	1, bm_opencv 支持 imshow; 2, fix bug.
V0.7.1	2023.07.11	1, bm_ffmpeg 支持 jpeg loop dec; 2, 新增算子 bayer2rgb.
V0.7.3	2023.10.18	1, fix bug.
V0.8.0	2023.08.01	1, fix bug.
V0.10.0	2024.04.11	1, 开放 bmvid 接口; 2, vpu dec 支持外部分配物理内存; 3, 更新视频编解码固件.

CHAPTER 3

安装 sophon-mw

sophon-mw 在不同的 Linux 发行版上提供不同类型的安装方式。请根据您的系统选择对应的方式，不要在一台机器上混用多种安装方式。以下描述中“0.10.0”仅为示例，视当前实际安装版本会有变化。

下文中 $arch $system 根据实际架构进行配置:

- 主机为 x86 处理器的，$arch 为 amd64, $system 为 x86_64
- 主机为 arm64 或飞腾处理器的，$arch 为 arm64, $system 为 aarch64
- 主机为 sg2042 处理器的, $arch 为 riscv64, $system 为 riscv_64

如果使用 Debian/Ubuntu 系统:

sophon-mw 安装包由四个文件构成:

- sophon-mw-sophon-ffmpeg_0.10.0_$arch.deb
- sophon-mw-sophon-ffmpeg-dev_0.10.0_$arch.deb
- sophon-mw-sophon-opencv_0.10.0_$arch.deb
- sophon-mw-sophon-opencv-dev_0.10.0_$arch.deb

如果使用 CentOS 系统:

sophon-mw 安装包由四个文件构成:

- sophon-mw-sophon-ffmpeg_0.10.0_$arch.rpm
- sophon-mw-sophon-ffmpeg-dev_0.10.0_$arch.rpm
- sophon-mw-sophon-opencv-abi0_0.10.0_$arch.rpm
- sophon-mw-sophon-opencv-abi0-dev_0.10.0_$arch.rpm

如果使用 Fedora 系统：

sophon-mw 安装包由四个文件构成：

- sophon-mw-sophon-ffmpeg_0.10.0_$arch.rpm
- sophon-mw-sophon-ffmpeg-dev_0.10.0_$arch.rpm
- sophon-mw-sophon-opencv_0.10.0_$arch.rpm
- sophon-mw-sophon-opencv-dev_0.10.0_$arch.rpm

其中：

sophon-ffmpeg/sophon-opencv 包含了 ffmpeg/opencv 运行时环境（库文件、工具等）；sophon-ffmpeg-dev/sophon-opencv-dev 包含了开发环境（头文件、pkgconfig、cmake 等）。如果只是在部署环境上安装，则不需要安装 sophon-ffmpeg-dev/sophon-opencv-dev。

sophon-mw-sophon-ffmpeg 依赖 sophon-libsophon 包，而 sophon-mw-sophon-opencv 依赖 sophon-mw-sophon-ffmpeg，因此在安装次序上必须先安装 libsophon，然后 sophon-mw-sophon-ffmpeg，最后安装 sophon-mw-sophon-opencv。

CentOS 系统中使用的 libstdc++ 库使用旧版本 ABI 接口，请使用 sophon-mw-sophon-opencv-abi0_0.10.0_$arch.rpm 安装 sophon-mw-sophon-opencv。

安装步骤如下：

安装libsophon依赖库(参考《LIBSOPHON使用手册》)
安装sophon-mw

如果使用**Debian/Ubuntu**系统：

```
sudo dpkg -i sophon-mw-sophon-ffmpeg_0.10.0_$arch.deb sophon-mw-sophon-ffmpeg-dev_0.10.0_$arch.deb
sudo dpkg -i sophon-mw-sophon-opencv_0.10.0_$arch.deb sophon-mw-sophon-opencv-dev_0.10.0_$arch.deb
```

如果使用**CentOS**系统：

```
sudo rpm -ivh sophon-mw-sophon-ffmpeg_0.10.0_$arch.rpm sophon-mw-sophon-ffmpeg-dev_0.10.0_$arch.rpm
sudo rpm -ivh sophon-mw-sophon-opencv-abi0_0.10.0_$arch.rpm sophon-mw-sophon-opencv-abi0-dev_0.10.0_$arch.rpm
```

如果使用**Fedora**系统，请在安装之前卸载已安装的包：

已安装包目录可以通过此命令列举：

```
dnf list installed | grep sophon-ffmpeg
```

然后卸载：

```
sudo rpm -e sophon-mw-sophon-opencv-dev
sudo rpm -e sophon-mw-sophon-opencv
sudo rpm -e sophon-mw-sophon-ffmpeg-dev
sudo rpm -e sophon-mw-sophon-ffmpeg
```

再重新安装:

```
sudo rpm -ivh sophon-mw-sophon-ffmpeg_0.10.0_$arch.rpm sophon-mw-sophon-ffmpeg-dev_0.10.0_$arch.rpm
sudo rpm -ivh sophon-mw-sophon-opencv_0.10.0_$arch.rpm sophon-mw-sophon-opencv-dev_0.10.0_$arch.rpm
```

在终端执行如下命令，或者logout再login当前用户后即可使用安装的工具:

```
source /etc/profile
```

注意：位于SOC模式时，系统已经预装了:

sophon-mw-soc-sophon-ffmpeg

sophon-mw-soc-sophon-opencv

只需要按照上述步骤安装:

sophon-mw-soc-sophon-ffmpeg-dev_0.10.0_arm64.deb

sophon-mw-soc-sophon-opencv-dev_0.10.0_arm64.deb

安装位置为:

```
/opt/sophon/
├──── libsophon-0.10.0
├──── libsophon-current -> /opt/sophon/libsophon_|ver|
├──── sophon-ffmpeg_|ver|
│    ├──── bin
│    ├──── data
│    ├──── include
│    ├──── lib
│    │    ├──── cmake
│    │    └──── pkgconfig
│    └──── share
├──── sophon-ffmpeg-latest -> /opt/sophon/sophon-ffmpeg_|ver|
├──── sophon-opencv_|ver|
│    ├──── bin
│    ├──── data
│    ├──── include
│    ├──── lib
│    │    ├──── cmake
│    │    │    └──── opencv4
│    │    └──── pkgconfig
```

```
│   ├── opencv-python
│   ├── share
│   └── test
└── sophon-opencv-latest -> /opt/sophon/sophon-opencv_|ver|
```

deb 包安装方式并不允许您安装同一个包的多个不同版本，但您可能用其它方式在/opt/sophon 下放置了若干不同版本。在使用 deb 包安装时，/opt/sophon/sophon-ffmpeg-latest 和/opt/sophon/sophon-opencv-latest 会指向最后安装的那个版本。在卸载后，它会指向余下的最新版本（如果有的话）。

其中 include 和 lib/cmake lib/pkgconfig 目录，分别由 sophon-mw-sophon-ffmpeg-dev 和 sophon-mw-sophon-opencv-dev 包安装产生

卸载方式：

如果使用 Debian/Ubuntu 系统：

```
sudo apt remove sophon-mw-sophon-opencv-dev sophon-mw-sophon-opencv
sudo apt remove sophon-mw-sophon-ffmpeg-dev sophon-mw-sophon-ffmpeg
或者：
sudo dpkg -r sophon-mw-sophon-opencv-dev
sudo dpkg -r sophon-mw-sophon-opencv
sudo dpkg -r sophon-mw-sophon-ffmpeg-dev
sudo dpkg -r sophon-mw-sophon-ffmpeg
```

如果使用 CentOS 系统：

```
sudo rpm -e sophon-mw-sophon-opencv-abi0-dev
sudo rpm -e sophon-mw-sophon-opencv-abi0
sudo rpm -e sophon-mw-sophon-ffmpeg-dev
sudo rpm -e sophon-mw-sophon-ffmpeg
```

如果使用 Fedora 系统：

```
sudo rpm -e sophon-mw-sophon-opencv-dev
sudo rpm -e sophon-mw-sophon-opencv
sudo rpm -e sophon-mw-sophon-ffmpeg-dev
sudo rpm -e sophon-mw-sophon-ffmpeg
```

如果使用其它 Linux 系统：

安装包由一个文件构成：

- sophon-mw_0.10.0_$system.tar.gz

可以通过如下步骤安装：

先按照《LIBSOPHON 使用手册》安装好 libsophon 包，然后，

```
tar -xzvf sophon-mw_0.10.0_$system.tar.gz
sudo cp -r sophon-mw_0.10.0_$system/* /
sudo ln -s /opt/sophon/sophon-ffmpeg_0.10.0 /opt/sophon/sophon-ffmpeg-latest
```

```
sudo ln -s /opt/sophon/sophon-opencv_0.10.0 /opt/sophon/sophon-opencv-latest
sudo ln -s /opt/sophon/sophon-sample_0.10.0 /opt/sophon/sophon-sample-latest
sudo sed -i "s/usr\/local/opt\/sophon\/sophon-ffmpeg-latest/g"  /opt/sophon/sophon-ffmpeg-latest/lib/pkgconfig/*.pc
sudo sed -i "s/^prefix=.*$/prefix=\/opt\/sophon\/sophon-opencv-latest/g"  /opt/sophon/sophon-opencv-latest/lib/pkgconfig/opencv4.pc
```

注意：以上安装步骤的命令，如果直接粘贴存在问题，可以手动敲打。

最后，安装 bz2 libc6 libgcc 依赖库（这部分需要根据操作系统不同，选择对应的安装包，这里不统一介绍）

然后是一些配置工作：

```
添加库和可执行文件路径:
sudo cp /opt/sophon/sophon-ffmpeg-latest/data/01_sophon-ffmpeg.conf   /etc/ld.so.conf.d/
sudo cp /opt/sophon/sophon-opencv-latest/data/02_sophon-opencv.conf   /etc/ld.so.conf.d/
sudo ldconfig
sudo cp /opt/sophon/sophon-ffmpeg-latest/data/sophon-ffmpeg-autoconf.sh   /etc/profile.d/
sudo cp /opt/sophon/sophon-opencv-latest/data/sophon-opencv-autoconf.sh   /etc/profile.d/
sudo cp /opt/sophon/sophon-sample-latest/data/sophon-sample-autoconf.sh   /etc/profile.d/
source /etc/profile
```

卸载方式：

```
sudo rm -f /etc/ld.so.conf.d/01_sophon-ffmpeg.conf
sudo rm -f /etc/ld.so.conf.d/02_sophon-opencv.conf
sudo ldconfig
sudo rm -f /etc/profile.d/sophon-ffmpeg-autoconf.sh
sudo rm -f /etc/profile.d/sophon-opencv-autoconf.sh
sudo rm -f /etc/profile.d/sophon-sample-autoconf.sh
sudo rm -f /opt/sophon/sophon-ffmpeg-latest
sudo rm -f /opt/sophon/sophon-opencv-latest
sudo rm -f /opt/sophon/sophon-sample-latest
sudo rm -rf /opt/sophon/sophon-ffmpeg_0.10.0
sudo rm -rf /opt/sophon/sophon-opencv_0.10.0
sudo rm -rf /opt/sophon/sophon-sample_0.10.0
sudo rm -rf /opt/sophon/opencv-bmcpu_0.10.0
```

注意事项：

- **如果需要用 sophon-opencv 的 python 接口，手动设置环境变量：**

  ```
  export PYTHONPATH=$PYTHONPATH:/opt/sophon/sophon-opencv-latest/opencv-python
  ```

CHAPTER 4

使用 sophon-sample

sophon-sample 在不同的 Linux 发行版上提供不同类型的安装方式。请根据您的系统选择对应的方式, 不要在一台机器上混用多种安装方式。以下描述中 “0.10.0” 仅为示例, 视当前实际安装版本会有变化。

下文中 $arch $system 根据实际架构进行配置:

- 主机为 x86 处理器的,$arch 为 amd64, $system 为 x86_64
- 主机为 arm64 或飞腾处理器的,$arch 为 arm64, $system 为 aarch64
- 主机为 sg2042 处理器的, $arch 为 riscv64, $system 为 riscv_64

如果使用 Debian/Ubuntu 系统:

sophon-sample 安装包由以下文件构成:

- sophon-mw-sophon-sample_0.10.0_$arch.deb

如果使用 Fedora/CentOS 系统:

sophon-sample 安装包由以下文件构成:

- sophon-mw-sophon-sample_0.10.0_$arch.rpm

其中:

- sophon-mw-sophon-sample 包含了数个用于测试 sophon-ffmpeg/sophon-opencv 的应用程序;
- sophon-mw-sophon-sample 依赖上一章节的 sophon-ffmpeg/sophon-opencv 包。

安装步骤如下:

安装 libsophon 依赖库 (参考《LIBSOPHON 使用手册》)
安装 sophon-mw(参考上一章节)
安装 sophon-sample

如果使用 Debian/Ubuntu 系统：

- sudo dpkg -i sophon-mw-sophon-sample_0.10.0_$arch.deb

如果使用 Fedora/CentOS 系统，请在安装之前卸载已安装的包：

已安装包目录可以通过此命令列举：

- dnf list installed | grep sophon-ffmpeg

然后卸载：

- sudo rpm -e sophon-mw-sophon-sample_0.10.0_$arch.rpm

再安装：

- sudo rpm -ivh sophon-mw-sophon-sample_0.10.0_$arch.rpm

安装位置为：

```
/opt/sophon/
├── libsophon- 0.10.0 ├── libsophon-current -> /opt/sophon/libsophon 0.10.
↪0 ├── sophon-ffmpeg 0.10.0 ├── sophon-ffmpeg-latest -> /opt/sophon/sophon-
ffmpeg 0.10.0 ├── sophon-opencv 0.10.0 ├── sophon-opencv-latest -> /opt/
↪sophon/sophon-opencv 0.10.0 ├── sophon-sample_ 0.10.0 │   ├── bin
│   │   ├── test_bm_restart
│   │   ├── test_ff_bmcv_transcode
│   │   ├── test_ff_scale_transcode
│   │   ├── test_ff_video_encode
│   │   ├── test_ff_video_xcode
│   │   ├── test_ocv_jpubasic
│   │   ├── test_ocv_jpumulti
│   │   ├── test_ocv_vidbasic
│   │   ├── test_ocv_video_xcode
│   │   ├── test_ocv_vidmulti
│   │   └── water.bin
│   ├── data
│   └── sample
└── sophon-sample-latest -> /opt/sophon/sophon-sample_ 0.10.0
```

deb 包安装方式并不允许您安装同一个包的多个不同版本，但您可能用其它方式在/opt/sophon 下放置了若干不同版本。在使用 deb 包安装时/opt/sophon/sophon-sample-latest 会指向最后安装的那个版本。在卸载后，它会指向余下的最新版本（如果有的话）。

注意：soc 模式下，deb 安装包为

sophon-mw-soc-sophon-sample_0.10.0_arm64.deb

安装位置同上

卸载方式:

如果使用 Debian/Ubuntu 系统:

- sudo apt remove sophon-mw-sophon-sample 或者:
- sudo dpkg -r sophon-mw-sophon-sample_0.10.0_$arch.deb

如果使用 Fedora/CentOS 系统:

- sudo rpm -e sophon-mw-sophon-sample_0.10.0_$arch.rpm

用例介绍:

test_bm_restart

此用例主要用于测试 ffmpeg 模块下的视频解码功能和性能, 支持多路解码和断线重连功能。用户可以通过用例监测视频、码流的解码情况。

test_bm_restart [api_version] [yuv_format] [pre_allocation_frame] [codec_name] [sophon_idx] [zero_copy] [input_file/url] [input_file/url]

参数:

-api_version 指定解码过程使用的 ffmpegAPI 版本

0: 使用老版本的解码 avcodec_decode_video2 接口

1: 使用新版解码 avcodec_send_packet 接口

2: 使用 av_parser_parse2 的 API 用于抓包

-yuv_format 是否压缩数据,

0 表示不压缩

1 表示压缩

-pre_allocation_frame 允许的缓存帧数, 最多为 64

-codec_name 指定解码器, 可选择 h264_bm/hevc_bm,no 为不指定

-sophon_idx 若处于 SOC 模式, 该选项可以随意设置（不可为空）, 其值将会被忽略

-zero_copy 若处于 SOC 模式,0 表示启用 Host memory, 1 表示不启用若处于 PCIE 模式, 该选项可以随意设置（不可为空）, 其值将会被忽略

-input_file_or_url 输入的文件路径或码流地址

e.g

test_bm_restart 1 0 1 no 0 1 ./example0.mp4 ./example1.mp4 ./example2.mp4

test_ff_bmcv_transcode

此用例主要用于测试 ffmpeg 模块下的视频转码功能和性能, 通过调用 ff_video_decode,ff_video_encode 用例中的数据类型和函数, 来实现先解码后编码的转

码过程，以此保证解码和编码功能的正确性。同时此用例也可测试 ffmpeg 下的转码性能，运行时程序会输出即时转码帧率供参考。

test_ff_bmcv_transcode [platform] [src_filename] [output_filename] [encode_pixel_format] [codecer_name] [width] [height] [frame_rate] [bitrate] [thread_num] [zero_copy] [sophon_idx] <optional: enable_mosaic> <optional: enable_watermark>

参数：

-platform 平台: soc 或者 pcie

-src_filename 输入文件名如 x.mp4 x.ts 等

-output_filename 转码输出文件名如 x.mp4,x.ts 等

-encode_pixel_format 编码格式如 I420.

-encoder_name 编码 h264_bm,h265_bm.

-width 编码宽度 (32,4096]

-height 编码高度 (32,4096]

-frame_rate 编码帧率

-bitrate 编码比特率 encode bitrate 500 < bitrate < 10000

-thread_num 线程数量

-zero_copy PCie 模式: 0: 开启主从内存复制, 1: 关闭. SoC 模式: 任何数字均可以，但是无效

-sophon_idx PCie 模式: sophon 设备 id SoC 模式: 任何数字均可以，但是无效

-enable_mosaic 可选，在左上角添加马赛克，目前仅支持 bm1686 且-encode_pixel_format] 为 I420

-enable_watermark 可选，给视频添加水印，目前仅支持 bm1686 且-encode_pixel_format] 为 I420

e.g

pcie mode example:

```
test_ff_bmcv_transcode pcie example.mp4 test.ts I420 h264_bm 800 400 25 3000 3 0 0
```

soc mode example:

```
test_ff_bmcv_transcode soc example.mp4 test.ts I420 h264_bm 800 400 25 3000 3 0 0
```

test_ff_scale_transcode

此用例主要用于测试 ffmpeg 下视频转码的功能和性能。此功能通过先解码再编码的过程实现，主要调用了 ff_video_decode，ff_video_encode 中的数据类型和函数。

test_ff_scale_transcode [src_filename] [output_filename] [encode_pixel_format] [codecer_name] [height] [width] [frame_rate] [bitrate] [thread_num] [zero_copy] [sophon_idx]

参数:

-src_filename 输入文件名如 x.mp4 x.ts…

-output_filename 输出文件名如 x.mp4,x.ts…

-encode_pixel_format 编码格式如 I420

-codecer_name 编码名如 h264_bm,hevc_bm,h265_bm

-height 编码高度

-width 编码宽度

-frame_rate 编码帧率

-bitrate 编码比特率

-thread_num 使用线程数

-zero_copy 0: copy host mem,1: nocopy.

-sophon_idx 设备索引

e.g

test_ff_scale_transcode example.mp4 test.ts I420 h264_bm 800 400 25 3000 3 0 0

test_ff_video_encode

此用例主要用于测试 ffmpeg 模块下视频的编码功能。输入的视频限制为 I420 和 NV12 格式。通过调用此用例用户可以得到封装好的视频文件,ffmpeg 支持的视频格式均可。

test_ff_video_encode <input file> <output file> <encoder> <width> <height> <roi_enable> <input pixel format> <bitrate(kbps)> <frame rate> <sophon device index>

-input_file 输入视频路径

-output_file 输出视频文件名

-encoder H264 或者 H265, 默认为 H264

-width 视频宽度, 输出与输入需一致,256 <= width <= 8192

-height 视频高度, 输出与输入需一致,128 <= height <= 8192

-roi_enable 是否开启 roi,0 表示不开启,1 表示开启

-input_pixel_format I420(YUV, 默认), NV12.

-bitrate 输出比特率,10 < bitrate <= 100000, 默认为帧率 x 宽 x 高/8

-framerate 输出帧率,10 < framerate <= 60, 默认为 30

-sophon_device_index pcie 模式可用, 指定运行的设备号, 最小为 0

e.g

- test_ff_video_encode <input file> <output file> H264 width height 0 I420 3000 30 2
- test_ff_video_encode <input file> <output file> H264 width height 0 I420 3000 30
- test_ff_video_encode <input file> <output file> H265 width height 0 I420
- test_ff_video_encode <input file> <output file> H265 width height 0 NV12
- test_ff_video_encode <input file> <output file> H265 width height 0

test_ff_video_xcode

此用例主要用于测试 ffmpeg 下视频转码的功能和性能。此功能通过先解码再编码的过程实现, 主要调用了 ff_video_decode,ff_video_encode 中的数据类型和函数。转码后的视频分辨率与原视频一致, 比特率不能超过 10000kbps 或小于 500kbps, 否则会被置为默认值 3000kbps。转码后的视频如比特率与原视频一致, 那么时长也应一致。有一些丢帧属于正常现象。

test_ff_video_xcode <input file> <output file> encoder framerate bitrate(kbps) isdmabuffer pcie_no_copyback sophon_idx

-input_file 输入文件

-output_file 输出文件

-encoder 编码器 H264 或者 H265.

-isdmabuffer 是否开启内存一致,1 表示不开启,0 表示开启

PCIE 模式下可选:

-pcie_no_copyback 表示不将解码后 YUV 数据复制到 host 主机内存,0 表示复制 YUV 数据到 host 主机内存

-sophon_idx 指定 card 编号, 默认为 0

e.g

- test_ff_video_xcode ./file_example_MP4_1920_18MG.mp4 tran5.ts H264 30 3000 1 1 0
- test_ff_video_xcode ./file_example_MP4_1920_18MG.mp4 tran5.ts H264 30 3000 0 0 0
- test_ff_video_xcode ./file_example_MP4_1920_18MG.mp4 tran5.ts H264 30 3000 1 0 0

test_ocv_jpubasic

此用例主要用于测试图片编解码的基本功能。此用例进行了 3 种测试, 主要调用了 opencv 中的 imread,imwrite,imdecode,imencode 等接口用不同方式实现图片的编解码。

test_ocv_jpubasic <file> <loop> <yuv_enable> <dump_enable> [card]

-file 图片文件路径

-loop 循环次数

-yuv_enable 0 表示解码后输出 BGR 格式,1 表示解码后输出 YUV 格式

-dump_enable 1 表示输出 dump 文件,0 表示不输出

-card pcie 模型下可选。指定运行的 card 编号

e.g

test_ocv_jpubasic 1920x1080_gray.jpg 1000 1 1 0

test_ocv_jpumulti

此用例主要用于测试多路图片编解码的功能和性能。

test_ocv_jpumulti <test type> <inputfile> <loop> <num_threads> <outjpg> [card]

-test_type 测试类型,1 表示只解码,2 表示只编码,3 表示解码编码混合

-inputfile 输入文件

-loop 循环次数

-num_thread 线程数, 最小为 1, 最大为 12

-outjpg 是否输出图片,1 表示输出,0 表示不输出

-card PICE 模式下可选参数。指定运行的 card 编号

e.g

test_ocv_jpumulti 3 1088test1_420.jpg 10 2 1 0

test_ocv_vidbasic

此用例主要用于测试视频记录为 png 或 jpg 格式的功能。如果启用 dump, 则也可输出 BGR 或 YUV 格式的原始数据。

test_ocv_vidbasic <input_video> <output_name> <frame_num> <yuv_enable> [card] [WxH] [dump.BGR or dump.YUV]

-input_video 输入的视频文件路径

-output_name 输出文件名前缀

-frame_num 存储帧数

-yuv_enable 是否开启 yuv,0 表示输出 BGR 格式,1 表示输出 YUV 格式

-card PCIE 模式下可选参数, 指定运行的 card 编号

-WxH 可选参数, 指定输出文件的分辨率, 未指定则与原视频分辨率一致

-dumpBGR_or_dumpYUV 可选参数, 是否输出 dump.BGR 或 dump.YUV 文件

e.g

test_ocv_vidbasic station.avi test_basic 10 1 0 1920x1080

test_ocv_video_xcode

此用例主要用于 opencv 模块下视频转码的全方位测试, 对于 opencv 库支持的不同格式的视频输入, 提供 h264、h265 编码,yuv 格式输出,roi 区域等多种功能以及多种格式输出, 覆盖的格式以及功能范围非常广泛。

test_ocv_video_xcode <input> <code_type> <frame_num> <outputname> <yuv_enable> <roi_enable> [device_id] [encodeparams]

-input	输入文件名
-code_type	编码类型。H264enc 表示 H.264,H265enc 表示 H.265
-frame_num	需要处理的帧数
-outputname	指定输出文件名, 可以设置为 null
-yuv_enable	0 指定输出文件为 bgr 色彩格式; 1 指定输出文件为 yuv420 色彩模式
-roi_enable	0 表示禁用 roi 编码; 1 表示启用 roi 编码, 启用 roi 编码时, 输出文件名应指定为 null 并且在编码参数中加上 roi_eanble=1
-encodeparams	可选参数, 可以设置 roi_enable ,bitrate,min_qp,gop 等参数
-device_id	pcie 模式下可选参数, 指定 card 编号

e.g

test_ocv_video_xcode rtsp_url H265enc 10000 encoder_test265.ts 1 0 0 bitrate=1000

test_ocv_vidmulti

此用例主要用于 opencv 模块下视频以及码流的编解码性能测试, 测试设备在视频多路编解码转码时的性能, 终端可以输出每一个线程的编号、输入视频分辨率、重连次数、当前解码帧数、实时帧率、总平均帧率、丢帧数等, 以及编码功能相关信息。

test_ocv_vidmulti <thread_num> <input_video> [card] [enc_enable] <input_video> [card] [enc_enable]

-thread_num	线程数, 最大不超过 512
-enc_enable	是否开启编码。0 表示不开启,1 表示开启
-card	PCIE 模式下可选参数, 指定运行每个文件对应的 card 编号

环境变量:

export VIDMULTI_DISPLAY_FRAMERATE=0 只显示帧数

export VIDMULTI_DISPLAY_FRAMERATE=1 会显示每一个 channel 的详细信息, 可以看丢帧情况等

e.g

test_ocv_vidmulti 3 a.264 0 1 b.264 1 1 c.264 0 0

CHAPTER 5

使用 sophon-mw 开发

在安装完 sophon-mw 后，用户可以用两种方式将 sophon-mw 的库链接到自己的编译程序中。

** 如果使用 Make 编译系统 **

推荐用户使用 pkgconfig 来寻找 sophon-mw 库。

在之前的安装中，我们已经将 sophon-mw 的 pkgconfig 路径加入到环境变量 PKG_CONFIG_PATH。因此，用户可以在 Makefile 中添加如下语句：

```
CFLAGS  = -std=c++11

# add libsophon dependency because sophon-ffmpeg rely on it
CFLAGS  += -I/opt/sophon/libsophon-current/include/
LDFLAGS += -L/opt/sophon/libsophon-current/lib -lbmcv -lbmlib -lbmvideo -lbmvpuapi -
↪lbmvpulite -lbmjpuapi -lbmjpulite -lbmion

# add sophon-ffmpeg
CFLAGS  += $(shell pkg-config --cflags libavcodec libavformat libavfilter libavutil libswscale)
LDFLAGS += $(shell pkg-config --libs libavcodec libavformat libavfilter libavutil libswscale)

# add sophon-opencv
CFLAGS  += $(shell pkg-config --cflags opencv4)
LDFLAGS += $(shell pkg-config --libs opencv4)
```

然后就可以在 Makefile 中使用 sophon-ffmpeg 和 sophon-opencv 的库。

注意：当系统在/usr/lib 或者/usr/local/lib 下还安装了另一份 ffmpeg 或者 opencv 的时候，要注意检查下，是否搜索到了正确的 sophon-ffmpeg/sophon-opencv 路径。如果搜索不正确，则需要显式地指定头文件位置和库文件位置

** 如果使用 CMake 编译系统 **

用户可以在 CMakeLists.txt 中添加如下语句：

```
# add libsophon
find_package(libsophon REQUIRED)
include_directories(${LIBSOPHON_INCLUDE_DIRS})
link_directories(${LIBSOPHON_LIB_DIRS})

# add sophon-ffmpeg
set(FFMPEG_DIR      /opt/sophon/sophon-ffmpeg-latest/lib/cmake)
find_package(FFMPEG REQUIRED NO_DEFAULT_PATH)
include_directories(${FFMPEG_INCLUDE_DIRS})
link_directories(${FFMPEG_LIB_DIRS})

# add sophon-opencv
set(OpenCV_DIR /opt/sophon/sophon-opencv-latest/lib/cmake/opencv4)
find_package(OpenCV REQUIRED NO_DEFAULT_PATH)
include_directories(${OpenCV_INCLUDE_DIRS})

add_executable(${YOUR_TARGET_NAME} ${YOUR_SOURCE_FILES})

target_link_libraries(${YOUR_TARGET_NAME} ${FFMPEG_LIBS} ${OpenCV_LIBS})
```

在用户的代码中即可以调用 sophon-ffmpeg 和 sophon-opencv 中的函数：

```
#include <opencv2/opencv.hpp>

int main(int argc, char const *argv[])
{
  cv::Mat img = cv::imread(argv[1]);

  return 0;
}
```