Rockchip Linux Edge Python SDK 开发指导

文件标识：RK-KF-YF-861

发布版本：V0.4.1

日期：2022-04-13

文件密级：□绝密 □秘密 □内部资料 ■公开

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址： 福建省福州市铜盘路软件园A区18号

网址： www.rock-chips.com

客户服务电话： +86-4007-700-590

客户服务传真： +86-591-83951833

客户服务邮箱： fae@rock-chips.com

前言

概述

本文档为RK边缘计算SDK for Python Edition（以下简称PythonSDK）接口介绍。PythonSDK主要目的是将RK的芯片加速单元融入Python丰富的边缘计算生态环境中，无缝兼容主流的Python第三方计算库，实现快速开发。开发人员无需了解RK的众多加速单元，即可使用Python在RK芯片上快速开发。

产品版本

芯片名称	内核版本
RK3588	4.19、5.10

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

修订记录

版本号	作者	修改日期	修改说明
V0.0.1	Leok Wu	2021-12-13	创建初始接口文档
V0.0.2	Jeffery Zhang	2021-12-30	更新接口和文档模版
V0.0.5	Jeffery Zhang	2022-02-08	修正错别字以及一些描述措辞，发布Alpha版本
V0.3.4	Jeffery Zhang	2022-02-16	新增PipeCapture接口
V0.4.1	Jeffery Zhang	2022-04-13	新增HdmiCapture和CameraCapture接口

目录

1. 框架设计

1.1 设计约束

- 兼容Numpy、OpenCV、SciKit、Tensorflow、Pytorch等常见计算库，但不与任何库以及特定版本耦合。
- 隐藏RK加速单元的代码实现。
- 隐藏内存管理、线程管理、资源调度、缓存等具体实现。
- 支持交互式调用，命名规则尽可能和OpenCV保持一致。
- Python极简接口风格，隐藏任何非必要参数。
- 极简设计，不做冗余功能开发，保持库尽可能小。

1.2 适用范围

- 适合Python开发者、科研学者、学生使用。
- 适合Python交互式命令使用。
- 不适合深度定制以及对资源管理有高度要求的项目，复杂的开发需求可以直接使用C的API接口，而非该Python SDK。

1.3 加速单元

- GPU：Mali 图形处理单元
- RGA：RK 2D图形辅助计算单元
- VPU：RK 视频硬件编解码单元

1.4 模块介绍

1.4.1 Capture

音视频输入模块，其输入可以来自于网络，或者本地设备。

1.4.2 Writer

音视频输出模块，其输出可以通过网络，或者本地设备。

1.4.3 GraphicBuffer

所有read返回均为一个GraphicBuffer，而write均需要提供一个GraphicBuffer参数，他是由本SDK管理的DMABuffer，非物理连续。

1.4.4 SoundBuffer

（TBD）

1.5 命名规则

- XxxCapture：音视频输入接口
- XxxWriter：音视频输出接口
- read：读取一帧
- write：写入一帧

2. 使用方法

1. 执行以下命令进行安装

```
sudo apt install python3-toybrick
```

2. 在python中通过以下代码引用

```
import toybrick as toy
```

3. API总览

3.1 全局工具函数 Utils

函数名	描述
toy.version	获取版本信息
frame = toy.copy_from(nparray)	从numpy数组拷贝创建一个物理buffer

3.2 输入流 Capture

函数名	描述
stream = toy.RtspCapture(url, usr, pwd, isTCP)	建立Rtsp输入流
stream = toy.PipeCapture(format)	建立Pipe输入流
stream = toy.HdmiCapture(width, height, fps)	建立Hdmi输入流
stream = toy.CameraCapture(videoid, width, height, fps)	建立ISP Camera输入流
ret, frame = stream.read(width, height, format)	读取一帧图像

3.3 输出流 Writer

函数名	描述
stream = toy.RtspWriter(path, encoder, port)	建立Rtsp输出流
stream.write(frame, width ,height)	输出一帧图像

3.4 显示 Display

函数名	描述
disp = toy.Display(name, width, height, fullscreen, displayport)	新建显示设备
w = disp.width()	获取显示Buffer宽度
h = disp.height()	获取显示Buffer高度
view = disp.addview(x, y, w, h)	新增显示区域
disp.mvview(view, x, y, w, h)	移动显示区域
disp.rmview(view)	删除显示区域
disp.imshow(frame, view)	显示一帧

3.5 图像操作 Graphic

函数名	描述
dst = frame.rotate(degree)	图像旋转
dst = frame.resize(width, height)	图像缩放
dst = frame.crop(x, y, w, h)	图像剪裁
nparr = frame.asarray()	转为numpy数组，可给cv、numpy使用

4. 枚举定义

```
# Image Format
FMT_YUV420SP = 'NV12'
FMT_NV12 = 'NV12'
FMT_RGB888 = 'RGB888'
FMT_BGR888 = 'BGR888'
FMT_ARGB8888 = 'XRGB8888'
FMT_XRGB8888 = 'XRGB8888'

# Sound Format
FMT_PCM = 'PCM'

# Video Encoder
FMT_H264 = "H.264"
FMT_H265 = "H.265"
FMT_MJPEG = "MJPEG"

# Display Port
DISP_HDMI_A_1 = 'HDMI_A_1'
DISP_HDMI_A_2 = 'HDMI_A_2'
DISP_EDP_1 = 'EDP_1'
DISP_DSI_1 = 'DSI_1'
DISP_DP_1 = 'DP_1'
DISP_WINDOW = 'WINDOW'
```

5. 错误信息

定义	值	描述
	0x0	未定义
RET_SUCCESS	0x7000	成功
RET_TRY_AGAIN	0x7001	没有错误发生，请再试一次

（TBD）

6. 接口描述

6.1 全局函数

6.1.1 copy_from

函数描述：从numpy数组拷贝并构建一个GraphicBuffer。

函数参数：

参数	描述
ndarray	拷贝源，目前仅支持numpy数组
gformat	拷贝的图片格式

6.2 RtspCapture

6.2.1 描述

Rtsp输入流。需要注意的是，交互模式下并不能阻塞住Rtsp服务端发送数据，所以就算不调用read等函数来获取数据，其依然会在后台工作不断的接收数据，而read函数永远只会获取cache里最新的那帧。

6.2.2 构造参数

参数	描述
url	Rtsp服务地址
usr	用户名，默认为空
pwd	密码，默认为空
isTcp	是否使用TCP连接，默认为False

6.2.3 成员函数

6.2.3.1 read

函数描述：读取一帧图片，该函数按照开发者要求获取一帧指定宽高和格式的图片，如果没有指定宽高格式，那么将返回帧原始图片，不做任何转换。

函数参数：

参数	描述
width	想要的宽，默认为0
height	想要的高，默认为0
gformat	想要的图片格式，默认为FMT_RGB888

返回值：

返回值	描述
errno	错误代码
frame	一帧数据GraphicBuffer

6.2.4 示例代码

```
rtsp = toy.RtspCapture("rtsp://172.16.9.230/cam/realmonitor?
channel=1&subtype=0",usr="admin", pwd="admin", istcp=False)

while True:
    ret, frame = rtsp.readv(1920, 1080, toy.FMT_RGB888)
```

6.3 RtspWriter

6.3.1 描述

Rtsp输出流。新建一个RtspServer，等待客户端连接，并往所有连接上发送编码后的数据包。需要注意的是，就算没有任何连接，依然会发送(丢弃)数据包，并不会因此而阻塞。

6.3.2 构造参数

参数	描述
method	本地路径，即 rtsp://:/格式中名字
encoder	发送编码格式，默认为 FMT_H264
port	监听端口号，默认为 554

6.3.3 成员函数

6.3.3.1 write

函数描述：写入一帧图片，往所有连接发送编码后的数据。

函数参数：

参数	描述
frame	一帧GraphicBuffer
width	需要编码的宽，默认为0，即原始图像宽度
height	需要编码的高，默认为0，即原始图像高度

返回值：

返回值	描述
errno	错误代码

6.3.4 示例代码

```
rtsps = toy.RtspWriter("/live", toy.Encoder.H_264, 554)
while True:
    ret = rtsps.write(frame, 1920, 1080)
```

6.4 PipeCapture

6.4.1 描述

通过进程管道获取数据。不建议通过管道传输大数据，目前仅支持裸H.264/H.265二进制数据。可以配合其他第三方应用传递脱包后的裸数据。

6.4.2 构造函数

参数	描述
popen	SubProcess.popen 返回的进程管道
format	管道数据格式，目前仅支持H.264/H.265数据

6.4.3 成员函数

6.4.3.1 read

函数描述：读取一帧图片，该函数按照开发者要求获取一帧指定宽高和格式的图片，如果没有指定宽高格式，那么将返回帧原始图片，不做任何转换。

函数参数：

参数	描述
width	想要的宽，默认为0
height	想要的高，默认为0
gformat	想要的图片格式，默认为FMT_RGB888

返回值：

返回值	描述
errno	错误代码
frame	一帧数据GraphicBuffer

6.4.4 示例代码

```
import toybrick as toy
import <第三方程序>

process = (
    <第三方程序>
```

```
        .input('H264_8k_43Mbps_30fps_46Mbps.mp4')
        .output('-', format='h264', codec='copy')
        .run_async(pipe_stdout=True)
)

disp = toy.Display("File", 1920, 1080, True)
pipe = toy.PipeCapture(process, toy.FMT_H264)
pipe.start()

while True:
    ret, frame = pipe.read(1920, 1080)
    if frame is None:
        time.sleep(0.01)
        continue
    disp.imshow(frame)
```

6.5 HdmiCapture

6.5.1 描述

获取Hdmi In图像流。

6.5.2 构造函数

参数	描述
width	希望的宽，建议1920
height	希望的高，建议1080
fps	设定最高帧率，例如30

6.5.3 成员函数

6.5.3.1 read

函数描述：读取一帧图片，该函数按照开发者要求获取一帧指定宽高和格式的图片，如果没有指定宽高格式，那么将返回帧原始图片，不做任何转换。

函数参数：

参数	描述
width	想要的宽，默认为0
height	想要的高，默认为0
gformat	想要的图片格式，默认为FMT_RGB888

返回值：

返回值	描述
errno	错误代码
frame	一帧数据GraphicBuffer

6.5.4 示例代码

```
hdmi = toy.HdmiCapture(1920, 1080)
while True:
  ret, frame = hdmi.read(1920, 1080)
```

6.6 CameraCapture

6.6.1 描述

获取V4L2 Camera图像流。仅支持ISP设备的Camera，不支持UsbCamera。

6.6.2 构造函数

参数	描述
deviceid	/dev/videoX的X
width	希望的宽，建议1920
height	希望的高，建议1080
fps	设定最高帧率，例如30

6.6.3 成员函数

6.6.3.1 read

函数描述：读取一帧图片，该函数按照开发者要求获取一帧指定宽高和格式的图片，如果没有指定宽高格式，那么将返回帧原始图片，不做任何转换。

函数参数：

参数	描述
width	想要的宽，默认为0
height	想要的高，默认为0
gformat	想要的图片格式，默认为FMT_RGB888

返回值：

返回值	描述
errno	错误代码
frame	一帧数据GraphicBuffer

6.6.4 示例代码

```
cam = toy.CameraCapture(8, 1920, 1080)
while True:
  ret, frame = cam.read(1920, 1080)
```

6.7 Display

6.7.1 描述

显示输出。当前SDK版本下为wayland输出，使用GPU进行加速和绘制。

6.7.2 构造参数

参数	描述
windowname	窗口名称（将会显示在任务栏上）
want_width	希望的显示宽，默认为640
want_height	希望的显示高，默认为360
fullscreen	是否全屏显示，默认为False
display_port	显示类型，例如设置为DISP_WINDOW为Wayland窗口模式，设置为DISP_HDMI_A_1为HDMI直接DRM输出（需要字符界面）

6.7.3 成员函数

6.7.3.1 addview

函数描述：新增一个显示区域。坐标系左上角为圆点(0, 0)，右下角(w, h)。

函数参数：

参数	描述
x	显示区域的x坐标
y	显示区域的y坐标
w	显示区域的宽
h	显示区域的高

返回值：

返回值	描述
view	返回当前view的id号

6.7.3.2 rmview

函数描述：删除一个显示区域。

函数参数：

参数	描述
view	view的id，即addview后的返回值

返回值：

返回值	描述
errno	错误代码

6.7.3.3 mvview

函数描述：移动（重设）一个显示区域。

函数参数：

参数	描述
view	view的id，即addview后的返回值
x	显示区域的x坐标
y	显示区域的y坐标
w	显示区域的宽
h	显示区域的高

6.7.3.4 imshow

函数描述：显示一帧数据，注意，如果没有给定一个view，默认为全窗口显示。

函数参数：

参数	描述
frame	一帧GraphicBuffer图像
view	显示区域view的id，即addview后的返回值

返回值：

返回值	描述
errno	错误代码

6.7.3.5 width

函数描述：获取显示Buffer宽度。

6.7.3.6 height

函数描述：获取显示Buffer高度。

6.7.4 示例代码

```
disp = toy.Display('Test', 1920, 1080, False)
view1 = disp.addview(0, 0, 640, 320)
view2 = disp.addview(640, 320, 640, 320)
while True:
  disp.imshow(frame1, view1)
  disp.imshow(frame2, view2)
```

6.8 GraphicBuffer

6.8.1 描述

所有图像Buffer的载体类。

6.8.2 构造参数

不支持从构造函数构造。

6.8.3 成员函数

6.8.3.1 width

函数描述：返回图像宽度。

6.8.3.2 height

函数描述：返回图像高度。

6.8.3.3 asarray

函数描述：转换numpy数组格式，可直接给opencv、numpy等常用运算库作为参数使用。

6.8.3.4 resize

函数描述：缩放图像。

函数参数：

参数	描述
widht	缩放目标宽
height	缩放目标高

返回值：一张新的图像。

6.8.3.5 crop

函数描述：裁剪图像。

函数参数：

参数	描述
x	相对于原图裁剪开始的位置坐标x
y	相对于原图裁剪开始的位置坐标y
w	裁剪宽度
h	裁剪高度

返回值：一张新的图像。

6.8.3.6 rotate

函数描述：旋转图像。

函数参数：

参数	描述
degree	旋转角度，仅支持90、180、270

返回值：一张新的图像。

6.8.4 示例代码

```
ret, frame = rtsp.read(1920, 1080)
dst1 = frame.rotate(90)
dst2 = frame.resize(1280, 720)
dst3 = frame.crop(0, 0, 640, 360)
```