

密级状态：绝密( ) 秘密( ) 内部( ) 公开(√ )

# Rockchip Developer Guide Android SVEP SR

(图形计算平台中心)

文件状态： [ ] 正在修改 [√] 正式发布	文档标识：	RK-KF-YF-410
	当前版本：	2.1.0
	作 者：	李斌
	完成日期：	2023-12-28
	审 核：	熊伟
	完成日期：	2023-12-28

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd

(版本所有,翻版必究)

Rockchip

## 免责声明

本文档按“现状”提供，瑞芯微电子股份有限公司（“本公司”，下同）不对本文档的任何陈述、信息和内容的准确性、可靠性、完整性、适销性、特定目的性和非侵权性提供任何明示或暗示的声明或保证。本文档仅作为使用指导的参考。

由于产品版本升级或其他原因，本文档将可能在未经任何通知的情况下，不定期进行更新或修改。

## 商标声明

“Rockchip”、“瑞芯微”、“瑞芯”均为本公司的注册商标，归本公司所有。

本文档可能提及的其他所有注册商标或商标，由其各自所有者所有。

## 版权所有 © 2023 瑞芯微电子股份有限公司

超越合理使用范畴，非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

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](http://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](mailto:fae@rock-chips.com)

## 更新记录

版本	修改人	修改日期	修改说明	核定人
V1.1.1	李斌	2022-03-15	<b>【新增】</b> 集成实现 SR 画质增强功能 <b>【新增】</b> 实现 SR 控制接口属性	熊伟
V1.2.1	李斌	2022-04-11	<b>【新增】</b> 支持 4K SR 模型 <b>【修复】</b> 解决若干内部软件问题 <b>【新增】</b> 新增画质增强强度调整接口；	熊伟
V1.3.1	李斌	2022-05-11	<b>【优化】</b> 完成 SR 库内存优化 <b>【新增】</b> 增加 SR 调试节点 <b>【优化】</b> 修改部分属性名称；	熊伟
V1.4.1	李斌	2022-06-09	<b>【新增】</b> 完成 VendorStorage 授权逻辑 <b>【新增】</b> 完成 OSD 字幕修改接口	熊伟
V1.5.0	李斌	2022-06-20	<b>【优化】</b> 优化 SR 系统负载 <b>【优化】</b> 优化 SR OSD 字幕效果 <b>【修复】</b> 解决若干内部软件问题	熊伟
V1.6.0	李斌	2022-08-13	<b>【优化】</b> 优化 SR 效果 <b>【优化】</b> 优化左右对比模式效果 <b>【新增】</b> FAQ 4.7 问题	熊伟
V1.7.7	李斌	2023-07-11	<b>【优化】</b> 优化 SR 带宽负载 <b>【新增】</b> 第三方应用白名单支持应用列表 <b>【新增】</b> OSD-Online 模式	熊伟
V2.0.5	李斌	2023-08-29	<b>【优化】</b> 优化 SR 用户接口 <b>【修复】</b> 修复若干内部软件问题 <b>【修改】</b> 补充 2.5/2.7 章节内容 <b>【修改】</b> 更新文档配图	熊伟
V2.1.0	李斌	2023-12-28	<b>【优化】</b> 优化用户调用接口 <b>【优化】</b> RK3588 支持 SRAM 优化 <b>【新增】</b> 支持 RK3568 平台 <b>【新增】</b> 集成说明章节 <b>【修改】</b> 调整 Android libsvep 目录结构 <b>【修复】</b> 修复内部稳定性问题	熊伟

# 目 录

<b>1 SVEP 简介 .....</b>	<b>3</b>
1.1 软件架构 .....	3
1.2 支持模式 .....	4
1.2.1 RK3588 .....	4
1.2.2 RK3568 .....	5
<b>2 集成说明 .....</b>	<b>7</b>
2.1 软件架构 .....	7
2.2 数据流处理说明 .....	8
2.3 版本匹配说明 .....	8
2.3.1 版本号匹配查询 .....	9
2.3.2 目录结构更新说明: .....	9
2.4 编译说明 .....	10
<b>3 接口说明 .....</b>	<b>12</b>
3.1 版本号查询 .....	12
3.2 模式使能 .....	13
3.2.1 效果展示 .....	13
3.3 对比模式 .....	14
3.3.1 效果展示 .....	14
3.4 增强强度 .....	15
3.4.1 效果展示 .....	15
3.5 OSD 字幕接口 .....	16
3.5.1 效果展示 .....	16
3.5.2 限制信息 .....	17

<b>4 算法匹配说明</b>	<b>18</b>
4.1 匹配规则说明	18
4.2 名单匹配规则说明	19
4.2.1 白名单	19
4.2.2 黑名单	20
4.2.3 配置方法	20
4.3 匹配日志说明	22
<b>5 算法授权说明</b>	<b>23</b>
5.1 授权码申请	23
5.2 授权码存储位置	23
5.2.1 简易授权码导入工具	23
5.3 授权验证	24
5.3.1 快速授权验证方法	24
5.3.2 Android 授权日志说明	25
5.4 授权实现流程	26
<b>6 常见问题</b>	<b>27</b>
6.1 简单介绍 SR 具体做了哪些处理？	27
6.2 SR 效果的评价手段和指标有哪些？	27
6.3 非标准分辨率输出存在绿边问题	27
6.4 SR 帧率低问题	29
6.5 最新版本获取	30

# 1 SVEP 简介

SVEP（Super Vision Enhancement Process，超级视觉增强处理），是一项利用深度学习实现的图像增强处理技术，目前实现算法主要有 SR 与 MEMC 两类：

- **SR（Super Resolution，超级分辨率）**，SR 算法利用深度神经网络补充图片纹理细节，将原始低分辨率输入重建为清晰高分辨输出，以提升视频清晰度获得更高主观感知。
- **MEMC（Motion Estimation and Motion Compensation，运动补偿）**，MEMC 算法利用深度神经网络从前后两帧原始帧计算出中间的预测帧，提高视频帧率以获得较原始视频更流畅的感官体验。

SVEP 算法主要面向电视盒子、教育平板等领域，提供对在线、离线图像的超分辨率（SR）和运动补偿（MEMC）图像增强功能，提高视觉效果。

## 1.1 软件架构

SR 软件架构如下图 1-1 SR 软件架构图所示：

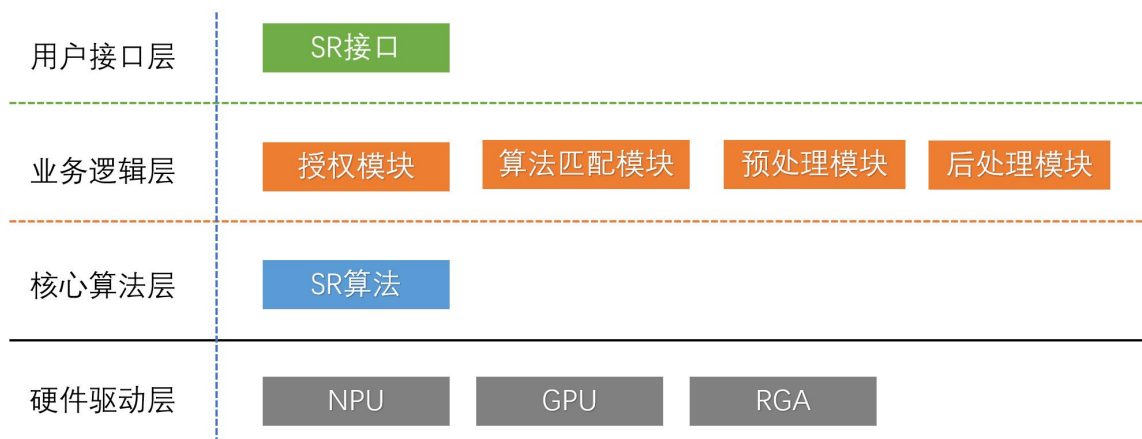


图 1-1 SR 软件架构图

- **用户接口层**：主要提供简洁的 SR 用户调用接口，也是本文档介绍的重点；
- **业务逻辑层**：主要由授权模块、算法匹配模块组成、预处理模块，后处理模块组成：
  - **授权模块**：主要用于算法软件授权，未经授权不允许调用相关核心算法；

- **算法匹配模块：**主要针对输入图像参数进行算法匹配；
- **预处理模块：**主要对输入图像进行图像预处理，使输入数据满足算法相关约束；
- **算法处理模块：**主要调用核心算法进行图像处理；
- **后处理模块：**主要对算法输出模块进行图像后处理，例如添加 OSD，实现对比模式等；
- **核心算法层：**提供 SR 算法实现；
- **硬件驱动层：**底层驱动提供 NPU/GPU/RGA 硬件支持；

## 1.2 支持模式

### 1.2.1 RK3588

#### 1.2.1.1 算法支持模式

SR 算法针对不同输入分辨率提供专门的优化模式，目前支持的 SR 模式如下表 1-1 RK3588 SR 算法模式支持列表：

表 1-1 RK3588 SR 算法模式支持列表

SR 模式	输入分辨率	输出分辨率	实时支持帧率	放大倍数
480p	768x480	2304x1440	30	3.0
540p	960x540	2880x1660	30	3.0
720p	1280x720	3840x2160	30	3.0
1080p	1920x1080	3840x2160	30	2.0
2160p	3840x2160	3840x2160	30	1.0
4320p	3840x2160	7680x4320	30	2.0

**注意：**若输入分辨率不满足标准 SR 模型的分辨率，系统端会适配到更大一级的 SR 模型 进行画质增强。例如：若输入 1024x600 分辨率视频，则采用 1280x720 SR 模型进行处理。利用此方法执行 SR 操作的输出 Buffer 将会出现绿边，需要用后续业务流程进行裁剪读取有效图像区域，详情见[“8.3 非标准分辨率输出存在绿边问题”](#)



### 1.2.1.2 色域空间支持

RK3588 SR 算法色域空间支持情况，如下表 1-2 RK3588 SR 算法色域空间支持情况所示：

表 1-2 RK3588 SR 算法色域空间支持情况

输入色域空间	RGB 格式支持情况	YUV 格式支持情况	输出色域空间
BT601 Limit Range	不存在	支持	BT601 Limit Range
BT601 Full Range	不存在	支持	BT601 Limit Range
BT709 Limit Range	不存在	支持	BT709 Limit Range
BT709 Full Range	支持	支持	BT709 Limit Range
BT2020	不支持	不支持	不支持

## 1.2.2 RK3568

### 1.2.2.1 算法支持模式

SR 算法针对不同输入分辨率提供专门的优化模式，目前支持的 SR 模式如下表 1-3 RK3568 SR 算法模式支持列表：

表 1-3 RK3568 SR 算法模式支持列表

SR 模式	输入分辨率	输出分辨率	实时支持帧率	放大倍数
480p	864x480	1728x960	30	2.0
720p	1280x720	1920x1080	30	1.5
1080p	1920x1080	1920x1080	30	1.0

**注意：**若输入分辨率不满足标准 SR 模型的分辨率，系统端会适配到更大一级的 SR 模型 进行画质增强。例如：若输入 1024x600 分辨率视频，则采用 1280x720 SR 模型进行处理。利用此方法执行 SR 操作的输出 Buffer 将会出现绿边，需要用后续业务流程进行裁剪读取有效图像区域，详情见[“8.3 非标准分辨率输出存在绿边问题”](#)

### 1.2.2.2 色域空间支持

RK3566/RK3568 SR 算法色域空间支持情况，如下表 1-4 SR 算法色域空间支持情况所示：

表 1-4 RK3568 算法色域空间支持情况

输入色域空间	RGB 格式支持情况	YUV 格式支持情况	输出色域空间
BT601 Limit Range	不存在	支持	BT601 Limit Range
BT601 Full Range	不存在	支持	BT601 Limit Range
BT709 Limit Range	不存在	支持	BT709 Limit Range
BT709 Full Range	支持	支持	BT709 Limit Range
BT2020	不支持	不支持	不支持

## 2 集成说明

### 2.1 软件架构

目前 SR 深度集成在 Android 显示框架 HardwareComposer 服务中,整体架构图如下图 2-1 Android 系统集成架构所示:

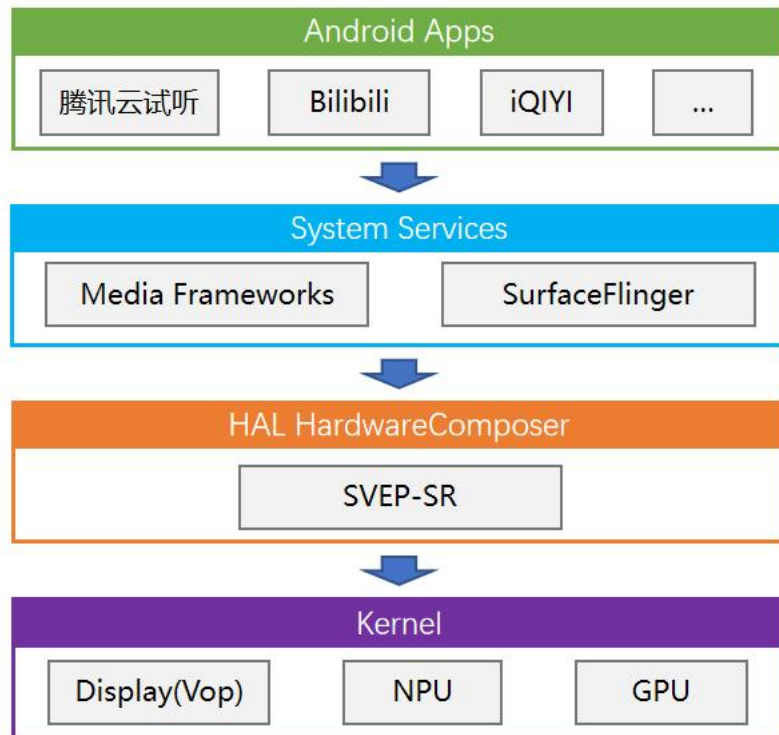


图 2-1 Android 系统集成架构图

HardwareComposer (简称 HWC) 服务负责处理所有应用图像上屏请求,服务内部可以调用 SR 算法实现对特定应用的图像进行 SR 处理后上屏显示。具体实现流程如下:

1. Android Apps 运行, 请求 Media Frameworks 播放视频;
2. Media Frameworks 解码片源, 并向 SurfaceFlinger 提交视频帧;
3. SurfaceFlinger 请求 HWC 服务处理视频帧;
4. HWC 服务调用 SR 算法对视频帧进行 SR 处理并提交屏幕送显;

有关 Android 显示框架的相关资料可参考: [Android Graphics docs](#)

## 2.2 数据流处理说明

以视频播放场景举例，视频数据流 SR 处理流程图如下图 2-2 视频流 SR 处理说明：

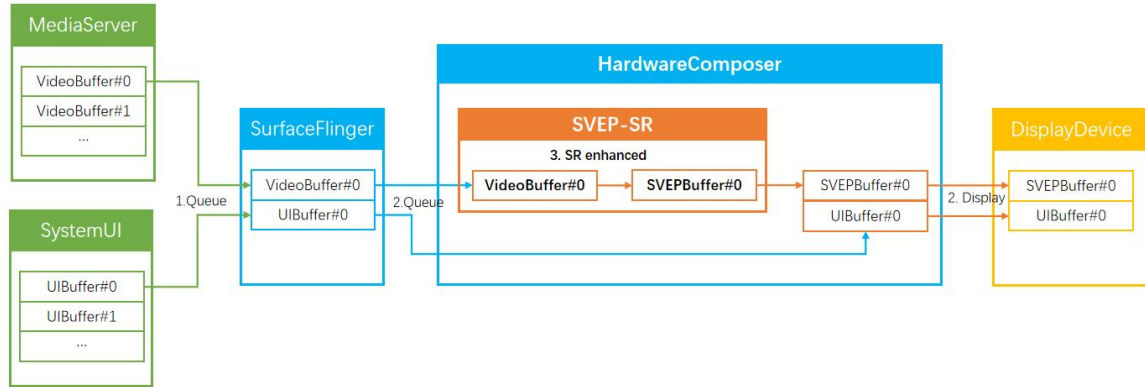


图 2-2 视频流 SR 处理说明

1. 视频应用请求 MediaServer 解码视频帧，提交 VideoBuffer#0 给 SurfaceFlinger 显示服务；
2. 系统 UI 应用渲染当前系统 UI，提交 UIBuffer#0 给 SurfaceFlinger 显示服务；
3. SurfaceFlinger 收集所有应用提交的待显示图像（VideoBuffer#0 与 UIBuffer#0），提交给 HWC 服务，请求上屏显示；
4. HWC 接收到 VideoBuffer#0 利用 SR 进行画质增强，并且输出 SVEPBuffer#0 替代 VideoBuffer#0，并且与 UIBuffer#0 打包成同一帧，提交 DisplayDevice 送显；
5. DisplayDevice 显示 SR 处理后的视频图像与当前的系统 UI；

## 2.3 版本匹配说明

HWC 模块与 SR 模块需要进行版本拉齐，建议严格按照版本匹配建议进行版本拉齐。

Android SDK 版本查询方法：

- SR 模块：V2.1.0 版本查询方法，更早期版本不支持此方法查询版本

```
$ cd hardware/rockchip/libsvcp/libsvcp/
$ strings lib/Android/rk3588/arm64-v8a/libsvcp.so | grep libsvcp
libsvcp version: 2.1.0 (4b25f45@2023-12-19T17:01:09)
```

- HWC 模块：

```
$ cd hardware/rockchip/hwcomposer/drmhwc2
$ cat include/rockchip/utls/drmdebug.h | grep GHWC_VERSION
#define GHWC_VERSION "HWC2-1.5.158"
```

### 2.3.1 版本号匹配查询

HWC 与 SR 模块版本匹配信息见下表 2-1 HWC 与 SR 模块版本对照表：

表 2-1 HWC 与 SR 模块版本对照表

SR 版本号	HWC 版本号	重要提示
1.1.1	1.2.55	
1.2.1	1.2.79	
1.3.1	1.2.92	
1.4.1	1.3.2	
1.5.0	1.3.5	
1.6.0	1.3.34	
1.7.7	1.4.11	
1.9.1	1.5.102	
2.0.5	1.5.143	SR 接口重构，无法向下兼容
2.1.0	1.5.158	SR 正式版本，无法向下兼容

**重要建议：**建议所有用户同步拉齐到至少 V2.1.0 版本。

### 2.3.2 目录结构更新说明：

为便于后续代码迭代，从 V2.1.0 版本开始，Android SDK 工程 libsvep 目录结构更新，路径如下：

hardware/rockchip/libsvep

早于 V2.1.0 版本目录结构如下：

- **Earlier than SR V2.1.0:**

```
$ hardware/rockchip/libsvep$ tree -L 2
.
├── Android.mk
├── common
├── include
│   ├── memc
│   └── sr
├── memc
└── sr
```

- Later than SR V2.1.0:

```
$ hardware/rockchip/libsvcp$ tree -L 2
├── libsvcpmemc
│   ├── Android.mk
│   ├── CHANGELOG.md
│   ├── docs
│   ├── examples
│   ├── lib
│   ├── LICENSE
│   ├── README.md
│   ├── resources
│   └── tools
└── libsvcpshr
    ├── Android.mk
    ├── CHANGELOG.md
    ├── docs
    ├── examples
    ├── lib
    ├── LICENSE
    ├── README.md
    ├── resources
    └── tools
```

**建议：**建议后续代码版本更新直接将旧 libsvcp 目录删除，下载 SR 新版本，直接解压到指定目录即可。

最新代码版本可参考“[6.5 最新版本获取](#)”章节获取。

## 2.4 编译说明

Android SDK 使能 SR 功能，在 [2.3 版本匹配](#)的前提下，需要在 BoardConfig.mk 添加如下字段：

```
BOARD_USES_LIBSVEP_SR := true
```

以 RK3588 举例，对应修改如下：

```
12/device/rockchip/rk3588$ git diff BoardConfig.mk
diff --git a/BoardConfig.mk b/BoardConfig.mk
index 71d04b0..1c0c399 100644
--- a/BoardConfig.mk
+++ b/BoardConfig.mk
@@ -124,3 +124,6 @@ BOARD_BASEPARAMETER_SUPPORT := true

#pcie ethernet
PRODUCT_HAVE_PCIE_ETHERNET := true
+
+# SVEP
+BOARD_USES_LIBSVEP_SR := true
```

**额外说明：** Earlier than SR V2.1.0 版本使用的条件编译选项如下：

```
BOARD_USES_LIBSVEP := true
```

目前代码内部也兼容此编译选项，等同于 BOARD\_USES\_LIBSVEP\_SR。

**编译报错日志说明：**

- 编译 drmhwc2，无法找到 libsvepsr 目录，请更新 libsvep 目录代码版本：

```
Directory hardware/rockchip/libsvep/libsvepsr does not exist, Please  
upgrade the libsvepsr to V2.1.0 version!
```

### 3 接口说明

Android HWC 服务集成 SR 接口，应用可以通过 Android Property 设置 SR 使能状态，即可实现 SR 相关功能设置，目前提供给应用设置的接口属性名如下表 3-1 SR 接口属性名说明：

表 3-1 SR 接口属性名说明

接口说明	接口属性名
模式使能	persist.sys.svep.mode
对比模式使能	persist.sys.svep.contrast_mode
对比偏移值	persist.sys.svep.contrast_offset_ratio
增强强度	vendor.svep.enhancement_rate
OSD 关闭	persist.sys.svep.disable_sr_osd
单行 OSD 模式	persist.sys.svep.enable_online_osd
单行 OSD 模式等待时间	persist.sys.svep.online_osd_wait_second

Android Property 属性调试可通过以下命令行进行查询与设置：

- 设置：

```
$ adb shell setprop persist.sys.svep.mode 1
```

- 查询：

```
$ adb shell getprop persist.sys.svep.mode
```

#### 3.1 版本号查询

版本号查询接口说明如下表 3-2 版本号查询接口说明所示：

表 3-2 版本号查询接口说明

接口说明	接口属性名
SR 版本号查询	vendor.svep.version
HWC 版本号查询	vendor.ghwc.version

可通过以下命令获取版本信息：



- SR 版本号查询：

```
$ adb shell getprop vendor.svep.version  
SR-2.1.0
```

- HWC 版本号查询：

```
$ adb shell getprop vendor.ghwc.version  
HWC2-1.5.157
```

## 3.2 模式使能

模式使能接口说明如下表 3-3 模式使能接口说明所示：

表 3-3 模式使能接口说明

接口说明	接口属性名	可选值	值说明
模式使能	persist.sys.svep.mode	0	关闭 SR 模式
		1	开启 SR 模式

可通过如下命令设置与获取对应属性值：

- 模式使能查询：

```
$ adb shell getprop persist.sys.svep.mode
```

- 模式使能设置：

```
$ adb shell setprop persist.sys.svep.mode 1
```

### 3.2.1 效果展示

SR 模式使能后播放 720p 效果如下图 3-1 SR 模式使能效果图：



图 3-1 SR 模式使能效果图

### 3.3 对比模式

对比模式接口说明如下表 3-4 对比模式接口说明所示：

表 3-4 对比模式接口说明

接口说明	接口属性名	可选值	值说明
对比模式使能	persist.sys.svep.contrast_mode	0	关闭对比模式
		1	开启对比模式
对比偏移值	persist.sys.svep.contrast_offset_ratio	0	开启对比扫描模式
		[10,90]	固定对比模式

- 对比模式使能查询：

```
$ adb shell getprop persist.sys.svep.contrast_mode
1
$ adb shell getprop persist.sys.svep.contrast_offset_ratio
50
```

- 模式使能设置：

```
$ adb shell setprop persist.sys.svep.mode 1
$ adb shell setprop persist.sys.svep.contrast_offset_ratio 50
```

#### 3.3.1 效果展示

对比模式使能后效果图如下图 3-2 SR 对比模式效果图：



图 3-2 SR 对比模式效果图

### 3.4 增强强度

SR 增强强度是通过调整 SR 模型内部权重系数，来实现对纹理细节效果的调整，强度设置越高，细节越清晰。增强强度接口说明如下表 3-5 增强强度接口说明所示：

表 3-5 增强强度接口说明

接口说明	接口属性名	可选值	值说明
增强强度	vendor.svep.enhancement_rate	0	关闭增强强度
		5	设置强度值为 5
		10	设置强度值为 10

- 增强强度查询：

```
$ adb shell getprop vendor.svep.enhancement_rate
0
```

- 增强强度设置：

```
$ adb shell setprop vendor.svep.enhancement_rate 5
```

#### 3.4.1 效果展示

如下图 3-3 SR 增强强度效果图所示：



图 3-3 SR 增强强度效果图

**注意：**SR 强度设置大于 0 的值会显著增加 GPU 端的负载，建议客户根据项目情况合理配置强度值。

## 3.5 OSD 字幕接口

OSD 字幕接口可在 SR 输出图像上叠加一层 OSD 字幕，用于提示 SR 相关信息。默认 OSD 显示模式为三行 OSD 模式，OSD 相关接口如下表 3-6 OSD 字幕接口说明表：

表 3-6 OSD 字幕接口说明表

接口说明	接口属性名	可选值	值说明
OSD 关闭	persist.sys.svep.disable_sr_osd	0	开启 OSD 模式
		1	关闭 OSD 模式
单行 OSD 模式	persist.sys.svep.enable_online_osd	0	关闭单行 OSD 模式
		1	开启单行 OSD 模式
单行 OSD 模式 等待时间	persist.sys.svep.online_osd_wait_second	0	无等待时间直接开启单行 OSD 模式
		1	进入 SR 后三行 OSD 模式 显示 1s 后再进入单行模式

- OSD 查询：

```
$ adb shell getprop persist.sys.svep.disable_sr_osd
$ adb shell getprop persist.sys.svep.enable_online_osd
$ adb shell getprop persist.sys.svep.online_osd_wait_second
```

- 增强强度设置：

```
$ adb shell setprop persist.sys.svep.disable_sr_osd 0
$ adb shell setprop persist.sys.svep.enable_online_osd 0
$ adb shell setprop persist.sys.svep.online_osd_wait_second 0
```

### 3.5.1 效果展示

- 三行 OSD 模式，如下图 3-4 三行 OSD 效果图：

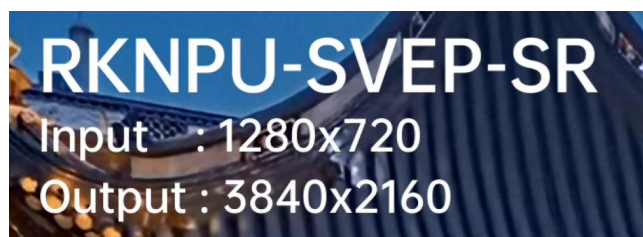


图 3-4 三行 OSD 效果图

- 单行 OSD 模式，去除 Input/Output OSD，如下图 3-5 单行 OSD 模式效果图：



图 3-5 单行 OSD 模式效果图

### 3.5.2 限制信息

- **长度限制：**自定义 OSD 对长度存在限制，限制信息如下表 3-7 OSD 字幕长度限制：

表 3-7 OSD 字幕长度限制

	大写英文字母	小写英文字母	简体中文	阿拉伯数字
支持最大长度	19	22	14	29

- **修改限制：**目前仅支持对第一行 OSD 字幕进行修改，剩余 Input/Output 信息无法修改。

## 4 算法匹配说明

### 4.1 匹配规则说明

SR 算法集成在 HWC 模块，HWC 模块收集所有应用的上屏数据，依据匹配规则指定其中图层进行 SR 处理，本章节介绍 SR 算法的匹配规则，主要匹配规则如下图 4-1 算法匹配规则：

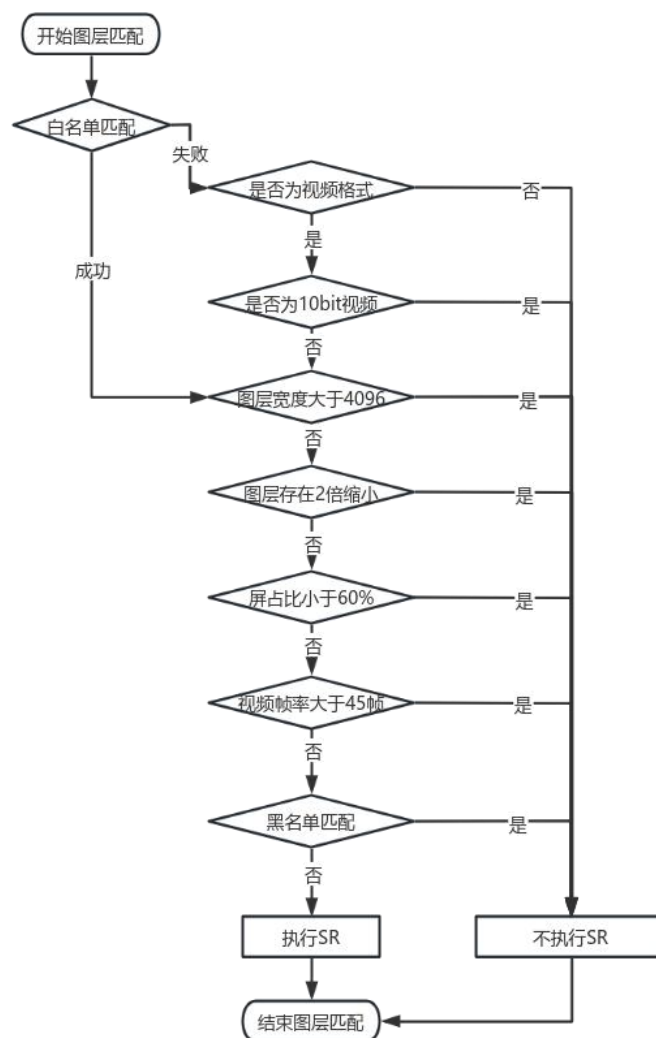


图 4-1 算法匹配规则

具体匹配规则描述如下：

1. 格式匹配：目前 SR 主要针对视频数据进行增强处理，若非视频数据请添加至白名单中。



- (1) 格式是否为视频格式，多数为 NV12 格式或者其他 YUV 格式，不支持 10bit 格式；
- (2) 非视频格式是否在白名单中存在，若存在则忽略视频格式限制；
2. 输入数据尺寸判断：判断输入图像尺寸是否大于 4096，大于 4096 的不进行 SR 处理；
3. 缩小行为判断，若图层存在 2 倍以上缩小，则不进行 SR 处理；
4. 屏占比判断：默认设置若屏占比小于 60%，则不进行 SR 处理；此项可通过以下命令修改，修改屏占比边界为 40%，即屏占比小于 40% 不使用 SR：

```
$ adb shell setprop vendor.hwc.disable_svep_dis_area_rate 40
```

5. 视频帧率判断：视频帧率大于 45 帧，则不进行 SR 处理，SR 处理能力不支持 45 帧的视频处理，建议处理帧率为 30 帧。图层刷新率可通过以下命令查看：

```
$ adb shell dumpsys SurfaceFlinger
```

对应的 FPS 参数可参考如下图 4-2 图层 FPS 查看：

DisplayId=0, Connector 380, Type = HDMI-A-1, Connector state = DMV_MODE_CONNECTED NumLayers=2, activeNodeid=3, 1920x1080p60.00, colorMode = 0, bStandardSwitchResolution=0														
id	z	sf-type	hwc-type	handle	transform	bind	source crop (l,t,r,b)			frame	dataspace	mFps	name	
0014   000	Device	Device	00b4000077abf53c30	None	None		0.0,	0.0,	1920.0,	1080.0	0,	0, 1920, 1080	10c10000   30.0	SurfaceView[com.rockchips.mediacenter/com.rockchips.mediacenter.MainActivity]
00005da1														
0006   001	Cursor	Device	00b4000077abf53990	None	Premultipl		0.0,	0.0,	29.0,	37.0	254, 149,	283, 186	0   0.0	bbq-adaptee#0(BLAST Consumer)0   0x11100006d7b

图 4-2 图层 FPS 查看

6. 黑名单匹配：存在于黑名单内的图层不进行 SR 处理；

## 4.2 名单匹配规则说明

### 4.2.1 白名单

部分应用可能需要对非 YUV 格式数据执行 SR 处理，故提供白名单配置方法，绕过视频格式的限制。具体白名单处理流程如下图 4-3 白名单处理流程图所示：

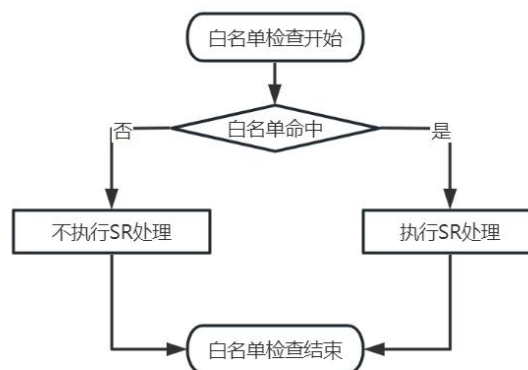


图 4-3 白名单处理流程图

### 4.2.2 黑名单

部分应用可能存在无需对视频进行 SR 处理的场景，如要求低延迟的云游戏场景，故提供黑名单配置，允许客户将应用图层名加入黑名单，系统 SR 处理流程检测黑名单上的图层后，不进行 SR 处理增强。流程图如下图 4-4 黑名单处理流程图所示：

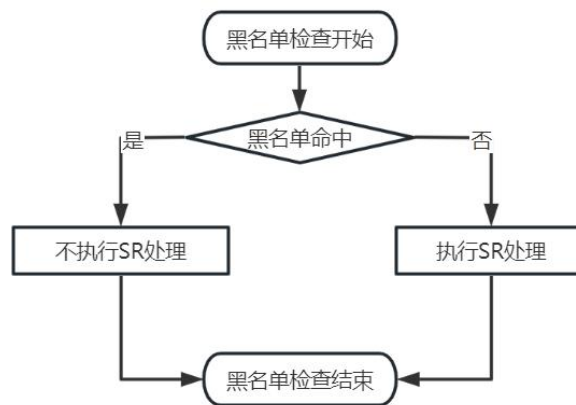


图 4-4 黑名单处理流程图

### 4.2.3 配置方法

黑白名单目前通过 XML 文件配置，具体文件路径为：

- 工程路径：

```
sdk_path/hardware/rockchip/hwcomposer/drmhwc2/configs/HwcSvepEnv.xml
```

- 设备端路径：

```
/vendor/etc/HwcSvepEnv.xml
```

**XML 文件格式：**参考下面 XML 源文件代码

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Svep xml -->
<HwcSvepEnv Version="1.1.1" >

  <Blacklist> <!-- 黑名单头部结构 -->
    <!-- 黑名单的关键词，通常为应用图层名字 -->
    <BlackKeywords>
SurfaceView[android.rk.RockVideoPlayer/android.rk.RockVideoPlayer.VideoPlayActi
vity</BlackKeywords>
    <!-- 可添加多个并列 BlackKeywords -->
    <BlackKeywords> RockVideoPlayer</BlackKeywords>
```



```

<!-- 可添加多个 BlackKeywords -->
<BlackKeywords>
android.rk.RockVideoPlayer.VideoPlayActivity</BlackKeywords>
</Blacklist>

<Whitelist>
<!-- 爱奇艺: 标准、Lite、巴布版本适用如下白名单 -->
<WhiteKeywords>SurfaceView[com.qiyi.video</WhiteKeywords>
<!-- 抖音 -->
<WhiteKeywords>ViewRootImpl[SplashActivity]</WhiteKeywords>
<!-- 优酷视频 -->
<WhiteKeywords>SurfaceView[com.youku.phone/</WhiteKeywords>
<!-- 哔哩哔哩 -->
<WhiteKeywords>SurfaceView[tv.danmaku.bili</WhiteKeywords>
<!-- 虎牙直播 -->
<WhiteKeywords>SurfaceView[com.duowan.kiwi</WhiteKeywords>
<!-- 腾讯课堂 -->
<WhiteKeywords>ViewRootImpl[RecruitAvActivity]</WhiteKeywords>
<WhiteKeywords>ViewRootImpl[FCourseDetailActivity]</WhiteKeywords>
<WhiteKeywords>ViewRootImpl[WebOpenUrlActivity]</WhiteKeywords>
<!-- 掌门一对一辅导 -->

<WhiteKeywords>ViewRootImpl[RecordedLessonDetailActivity]</WhiteKeywords>
</Whitelist>
</HwcSvepEnv>

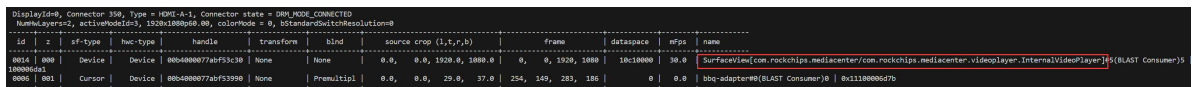
```

### XML 文件添加应用名方法举例:

1. 在目标场景按如下命令抓打印日志:

```
$ adb shell dumpsys SurfaceFlinger
```

2. 输出文件日志如下图红框部分即为应用图层名字, 如下图 4-5 应用图层名字获取说明:



id	z	ef-type	hwc-type	handle	transform	blend	source crop (l,t,r,b)	frame	dataspace	mfps	name
0004	000	Device	Device	0004000077d5f3c30	None	None	0.0, 0.0, 1920.0, 1080.0	0, 0, 1920, 1080	10c10000	30.0	SurfaceView[com.rockchips.mediocenter/com.rockchips.mediocenter.video.player.internal.VideoPlayer]45(BLAST Consumer)5
0005	001	Cursor	Device	0004000077d5f3c30	None	Premultipl	0.0, 0.0, 29.0, 37.0	254, 149, 283, 188	0	0.0	bbq-adaptor@0(BLAST Consumer)0   0x1100000d7b

图 4-5 应用图层名字获取说明

3. XML 文件中白名单位置添加如下字符串:

```

<WhiteKeywords>SurfaceView[android.rk.RockVideoPlayer/android.rk.Rock
VideoPlayer.VideoPlayActivity]</WhiteKeywords>

```

**注意:** 黑白名单匹配支持模糊匹配规则, 只要应用名中包含添加的字符串即认为名单命中, 建议添加的字符串具备应用的特征名, 例如爱奇艺应用的话, 包含“com.qiyi.video”字符串即可。

## 4.3 匹配日志说明

目前可通过如下命令打印 SR 算法匹配日志：

```
$ adb shell setprop vendor.hwc.log debug
$ adb shell logcat | grep SvpSrAllowed
```

目前以 60 帧片源播放作为例子，说明匹配日志。打开上述日志开关后，会出现如下图 4-6 高帧率片源无法使能 SR 日志打印：

```
SvpSrAllowedByLocalPolicy,line=284 disable-svp: ffps=60.000000 name=SurfaceView[com.rockchips.mediacenter/com.rockchips.mediacenter.videoplayer.InternalVideoPlayer]#21(BLAST Consumer)21
SvpSrAllowedByLocalPolicy,line=284 disable-svp: ffps=60.000000 name=SurfaceView[com.rockchips.mediacenter/com.rockchips.mediacenter.videoplayer.InternalVideoPlayer]#21(BLAST Consumer)21
SvpSrAllowedByLocalPolicy,line=284 disable-svp: ffps=60.000000 name=SurfaceView[com.rockchips.mediacenter/com.rockchips.mediacenter.videoplayer.InternalVideoPlayer]#21(BLAST Consumer)21
SvpSrAllowedByLocalPolicy,line=284 disable-svp: ffps=60.000000 name=SurfaceView[com.rockchips.mediacenter/com.rockchips.mediacenter.videoplayer.InternalVideoPlayer]#21(BLAST Consumer)21
```

图 4-6 高帧率片源无法使能 SR 日志打印

下列列举其他无法匹配日志说明：

- 源数据宽度大于 4096：

```
disable-sr: input too big, input-info (5120,2160) name=SurfaceView..
```

- 格式非视频格式且不在白名单名单内：

```
disable-sr: Not-YUV, can't find in Whitelist name=SurfaceView..
```

- SurfaceFlinger 服务强制请求非 SR 合成请求：例如请求高斯模糊效果，或者 HDR 效果：

```
disable-sr: SF request Client, name=SurfaceView..
```

- 视频 10bit 数据，SR 不支持：

```
disable-sr: is 10bit YUV, SR unsupport, name=SurfaceView..
```

- 图层本身存在 2 倍以上缩小：

```
disable-sr: scale-rate is too big fHScaleMul_=2.5 fVScaleMul_=2.5 SR
unsupport, name=SurfaceView..
```

- 开机动画阶段关闭 SR 请求：

```
disable-sr: during bootanim disable SR, name=SurfaceView..
```

- 图像屏幕占比小于 60%：

```
disable-sr: video_area_rate=50%, name=SurfaceView..
```

- 视频刷新率大于 35，则不对该图层执行 SR 处理：

```
disable-sr: video_max_fps=60, name=SurfaceView..
```

- 白名单命中：

```
Sr uid=50 is SurfaceView.. in Whitelist! force to SR.
```

- 黑名单名字：

```
Sr SurfaceView.. in BlackList! not to SR.
```

## 5 算法授权说明

SR 算法须要软件授权才能使用，目前是采用激活码授权方式进行终端网络授。

### 5.1 授权码申请

SR 授权码可联系 RK 业务申请；

### 5.2 授权码存储位置

在获得 SR 算法授权码后，需要将授权码导入设备指定的储存位置，SR 算法接口库初始化时才可进行算法鉴权。目前授权码设计存储在芯片 VendorStorage 分区，分区 ID=60，关于 VendorStorage 分区可参考以下路径文档，进行阅读了解：

[libsvepsr/docs/Rockchip\\_Application\\_Notes\\_Storage\\_CN.pdf](#)

生产的授权码写码工具请参考《Rockchip\_Application\_Notes\_Storage\_CN.pdf》文档，用户自行开发。

#### 5.2.1 简易授权码导入工具

本章节提供本地测试的简易写授权码工具 vendor-storage-test。

工具名称：vendor-storage-test，源码地址位于：

[libsvepsr/vendor-storage-test](#)

工具帮助信息：直接执行 vendor-storage-test 即可输出以下日志

```
$ adb shell vendor-storage-test
Usage : vendor_storage id len [code]
Config Code : vendor_storage 17 50 71581714-9736-4EEE-C029-B...
Read Code   : vendor_storage 17 50
```

授权码导入说明：超分授权码存放 ID 为 60，长度为 50 char，命令执行如下：

```
$ adb shell vendor-storage-test 60 50 \
71581714-9736-4EEE-C029-B65E54280A91
```

导入日志：导入后，输出日志如下：

```
vendor-storage-test 60 50 71581714-9736-4EEE-C029-B65E54280A91
vendor write:
```

```
tag=0x56524551 id=60 len=50
71581714-9736-4EEE-C029-B65E54280A91
vendor read:
tag=0x56524551 id=60 len=50
71581714-9736-4EEE-C029-B65E54280A91
```

## 5.3 授权验证

确保设备端满足以下条件后，即可进行授权验证：

1. 设备端 VendorStorage ID=60 存在可靠授权码；
2. 设备已连接互联网；

### 5.3.1 快速授权验证方法

本地快速搭建授权验证环境可利用 libsvpsr 工具提供的 sr\_demo 工具，源码位于以下路径：

```
libsvpsr/examples/sr-demo
```

利用以下命令进行授权验证：

```
$ sr_demo -i 1280x720+0+0:1280x720@NV12
```

授权情况日志可通过以下命令打印：

```
$ logcat -s SvpAuth
```

首次授权需要联网，授权成功输出日志如下图 2-1 初次授权成功日志 所示：

```
rk3588_box:/ # logcat -s SvpAuth
----- beginning of main
12-01 07:45:12.714 15299 15299 I SvpAuth: ReadAuthCode,line=475, CheckAuth: AuthCode: tag=0x56524551 id=60 len=50 Code: 3FE6CC5D-7BA6-54F8-99DC-1EA79BEED465
12-01 07:45:12.852 15299 15299 I SvpAuth: DoAuth,line=380, CheckAuth: rkauth_activate2 auth code success, Write license to /data/system/svp_key.lic
```

图 2-1 初次授权成功日志

授权成功后，SR 算法库会将离线的授权文件保存在以下路径：

```
/data/system/svp_key.lic
```

再次授权会直接校验离线的授权文件，授权成功后日志如下图 2-2 再次授权成功日志：

```
rk3588_box:/ # logcat -s SvpAuth
----- beginning of main
12-01 07:48:56.950 15376 15376 I SvpAuth: DoAuth,line=359, CheckAuth: rkauth_verify_license /data/system/svp_key.lic Success. ret=RKAUTH_OK
```

图 2-2 再次授权成功日志

**注意：**由于离线授权文件需要存在在 /data/system/ 目录，若设备端没有这个目录建议提前创建，否则离线授权文件将无法输出；

**注意：**授权码授权成功后，将于 CPU efuse 绑定，已授权的激活码无法再次授权第二颗 CPU，请妥善使用授权码资源。

### 5.3.2 Android 授权日志说明

Android 默认不会初始化 SR 模块，需要使能 SR 模式后系统才会初始化 SR 模块，所以要验证 SR 算法授权情况，首先需要使能 SR 模式，可通过如下命令使能：

```
$ adb shell setprop persist.sys.svep.mode 1
```

通过以下命令抓打印授权情况日志：

```
$ adb shell logcat | grep SvpAuth
```

#### 5.3.2.1 初次授权

初次日志打印如下图 4-1 Android 初次授权日志打印所示：

```
I SvpAuth: ReadAuthCode,line=494, CheckAuth: AuthCode: tag=0x56524551 id=60 len=50 Code: 841EA94C-7559-57E8-0B15-441B41B58B4C
E SvpAuth: DoAuth,line=407, CheckAuth: rkauth_activate auth code fail because networks err=RKAUTH_NETWORK_ERROR, try again.
I SvpAuth: DoAuth,line=399, CheckAuth: rkauth_activate2 auth code success, Write license to /data/system/svep_key.lic
```

图 4-1 Android 初次授权日志打印

日志说明如下：

- ReadAuthCode: 从 VendorStorage 分区获取算法激活码信息；
- DoAuth,line=407: 校验算法授权码失败，错误码为 RKAUTH\_NETWORK\_ERROR，网络错误，休眠 10s 后继续尝试在线授权；
- DoAuth,line=399: 校验算法授权码成功，将离线授权文件 svep\_key.lic 保存至 /data/system 路径，下次授权即可实现离线授权。

#### 5.3.2.2 再次授权

再次授权日志打印如下图 4-1 Android 再次授权日志打印所示：

```
I SvpAuth: DoAuth,line=359, CheckAuth: sr rkauth_verify_license /data/system/svep_key.lic Success. ret=RKAUTH_OK
```

图 4-1 Android 再次授权日志打印

再次授权无需联网即可实现离线算法授权。

## 5.4 授权实现流程

SR 算法授权内部实现流程如下图 2-3 授权实现流程图所示：

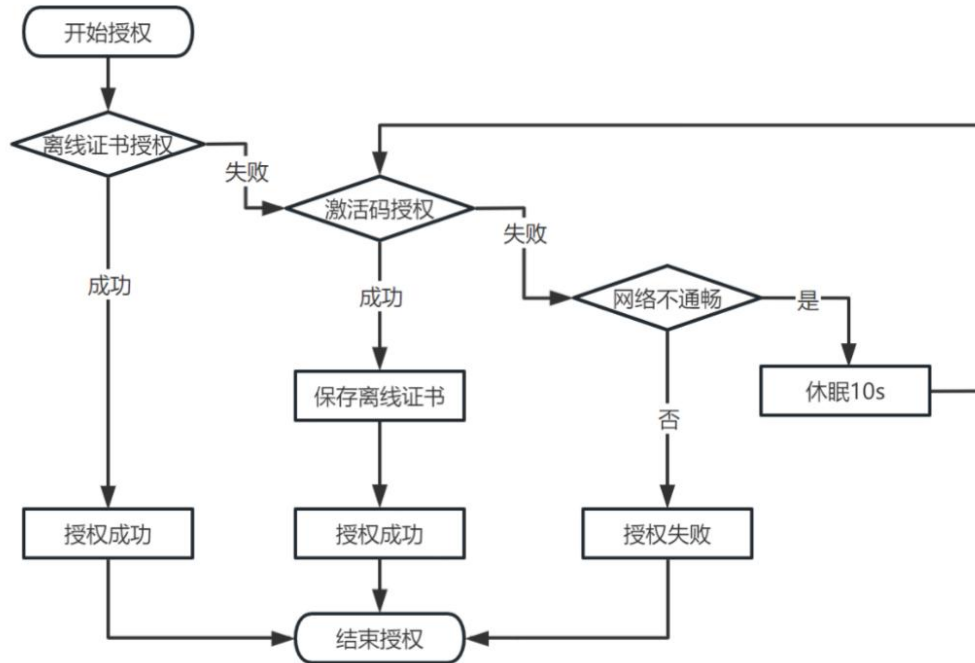


图 2-3 授权实现流程图

流程说明如下：

1. **离线证书授权：**初次授权，由于离线证书不存在，则离线证书授权失败，进入激活码授权流程；
2. **激活码授权：**获取 VendorStorage ID=60 激活码，进行网络授权：
  - a) 若由于网络原因导致授权失败，则休眠 10s 再尝试激活码授权；
  - b) 若激活码无效则直接判断授权失败；
2. **保存离线证书：**初次授权成功后，系统会将离线授权证书保存至 /data/system/svep\_key.lic 位置，用于下次离线授权；
3. **授权成功；**

## 6 常见问题

### 6.1 简单介绍 SR 具体做了哪些处理？

利用深度学习将低分辨率的图像放大为高分辨率图像，提高在低分辨率图像在高分辨率屏幕上显示效果。

### 6.2 SR 效果的评价手段和指标有哪些？

SR 效果评测可以利用图像和视频质量的评价指标比如 PSNR, SSIM, VMAF 等可以参考以下博客：  
视频流量在整个互联网流量的占比每年都在高速增长，为降低视频存储成本和数据传输通道的负载，视频压缩标准及算法在不断积极开发和改进。视频质量的评估在其中也起着至关重要的作用，尽管已经发展出了大量视频质量评估方法，但普遍接受度最高、最知名的评价方法还是经典的 PSNR、SSIM 以及 VMAF。

原文链接：[视频质量评价 VMAF，为何让人又喜又忧](#)

在评价视频质量时，单一指标数值的高低，并不能完全反映视频的画质好坏，与人眼主观感受相比会出现偏差。

所以通常评价视频画质时，要结合多项指标和人眼的主观感受进行综合评价。

### 6.3 非标准分辨率输出存在绿边问题

如果输入分辨率并不完全等于 SR 模型输入分辨率，目前内部做法是将源图像数据点对点拷贝到模型分辨率中，再执行 SR 处理，则输出图像会存在黑边问题，下面以 sr\_demo 举例说明：

sr\_demo 输入参数如下：

```
$ adb shell 'sr_demo -i 640x360+0+0:640x360@RG24 \  
-m +async+osd+spilt=50+en=0+rotate=0 \  
-f /data/bus_640x360_s640x360_RG24.bin \  
-c 1'
```

执行日志如下图 6-1 sr\_demo 非标准输入执行日志：



```
rk3588_box:/data # sr_demo -i 640x360+0+0:640x360@RG24 -m +async+osd+spilt-50+en=0+rotate=0 -f /data/bus_640x360_s640x360_RG24.bin -c 1
cmd_parse: crop[0,0,640,360] image[640,360,RG24] afbc=0 path=/data/bus_640x360_s640x360_RG24.bin async=1 osd=1 spilt=1 spilt-rate=50 en-rate=0 rotate=0x0
rga_api version 1.9.3 [1]
Svep Async-Thread(1) Wait success: Time = 120.59ms, FPS = 8.29, cnt=1
Please enter a word to dump dst data

Image output to /data/dump/I1_SrSrc_640x360_s640x360_RG24.bin
Image output to /data/dump/I2_SrDst_1920x1080_s1920x1080_NV12.bin
Please enter a release sr resource and exit!
```

图 6-1 sr\_demo 非标准输入执行日志

对应 info 日志如下图 6-2 非标准输入参数打印：

```
I SvepSr : DumpCtx, line=692, Magic=0x83991986 Version=2.0.10 Mode=SR-480p EnhancementRate=0 QsdMode=1 Contrast-Offset=1-50 Rotate=0x0
I SvepSr : DumpCtx, line=712, SrSrc: Fd=47 Buffer=[w,h,s,hs,size,f]=[640,360,640,360,691200, RG24] Buffer-Id=0x63ea00000001 Crop=[l,t,r,b]=[0,0,640,360] color=0x0 mask=0x0 AcqFence=-1
I SvepSr : DumpCtx, line=732, SrInt: Fd=1 Buffer=[w,h,s,hs,size,f]=[768,480,768,480,0, NV12] Buffer-Id=0x0 Crop=[l,t,r,b]=[0,0,640,1080] color=0x0 mask=0x0 AcqFence=-1
I SvepSr : DumpCtx, line=752, SrDst: Fd=50 Buffer=[w,h,s,hs,size,f]=[2304,1440,2304,1440,4976640, NV12] Buffer-Id=0x63ea00000002 Crop=[l,t,r,b]=[0,0,1920,1080] color=0x0 mask=0x0 AcqFence=-1
```

图 6-2 非标准输入参数打印

关注信息如下：

- 模型匹配到 480p 模型，符合向上模型匹配规则；
- 输出 crop 信息为 (0,0,1920,1080)，符合 480p 3 倍放大情况；

YUView 打开文件如下图 6-3 360p 输出图像预览图：

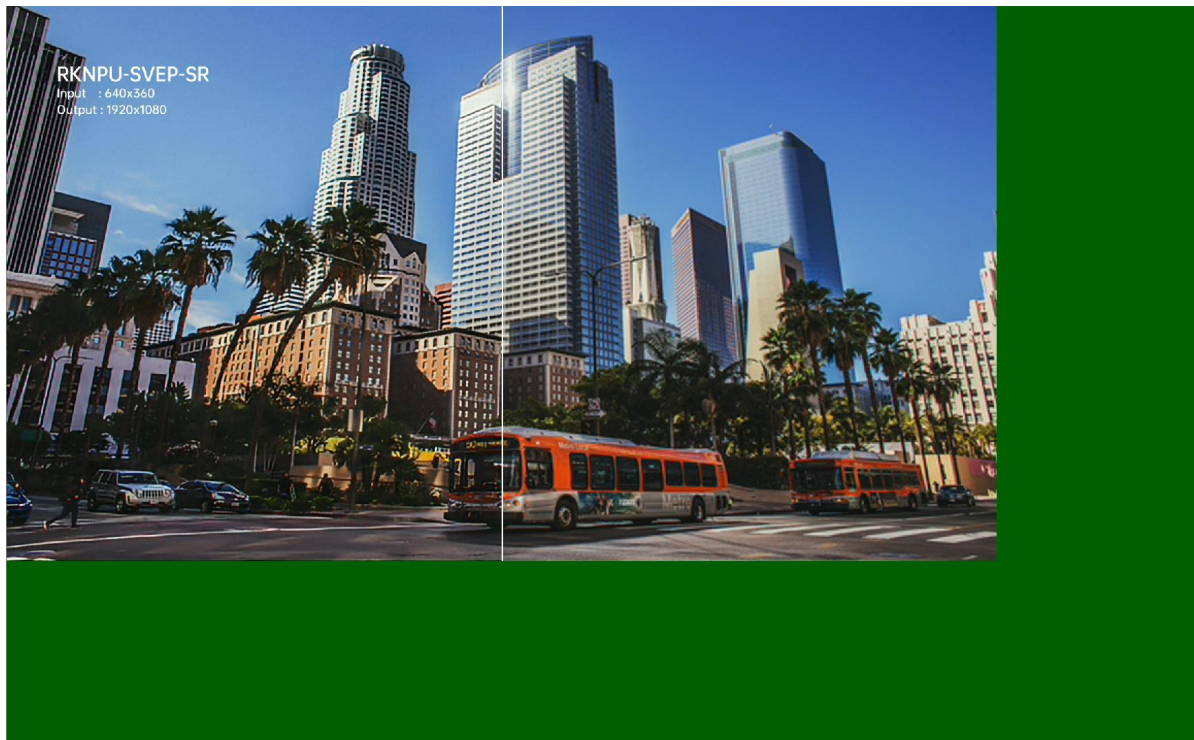


图 6-3 360p 输出图像预览图

关注点：实际内存大小为 2304x1440，有效图像存在于 (0,0,1920,1080) 区域内，符合预期。

外部模块要使用这份输出图像的话，需要设置 Crop 信息，否则将会存在绿边。

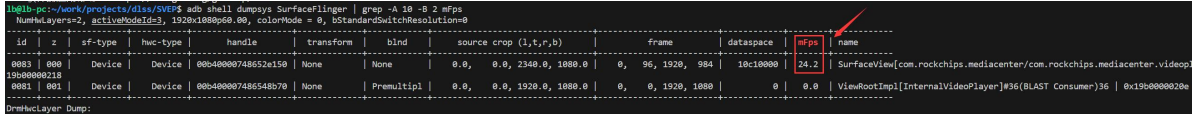


## 6.4 SR 帧率低问题

SR 帧率可通过以下命令获取：

```
adb shell dumpsys SurfaceFlinger | grep -A 10 -B 2 mFps
```

对应打印日志如下图 6-4 SR 图层帧率打印



id	z	sf-type	hwc-type	handle	transform	bind	source crop (l,t,r,b)	frame	dataspace	mFps	name
0083	000	Device	Device	00b40000748652e150	None	None	0.0, 0.0, 2340.0, 1080.0	0, 96, 1920, 984	10c10000	24.2	SurfaceView[com.rockchips.mediaceenter/com.rockchips.mediaceenter.videoop
10b00000218	003	Device	Device	00b400007486548b70	None	Premultipl	0.0, 0.0, 1920.0, 1080.0	0, 0, 1920, 1080	0	0.0	VideoRootImpl[InternalVideoPlayer]#36(BLAST Consumer)36   0x19b00000218e

图 6-4 SR 图层帧率打印

可能原因如下：

- SR 单帧处理耗时长：建议通过 `sr_demo async` 模式，执行 100 次对应操作，查看耗时打印，可参考《Rockchip\_API\_Reference\_SVEP\_SR\_CN.pdf》6 测试用例说明章节进行单元测试。
  - 确认 DDR/NPU/GPU 频率是否正常，可以尝试设置 `performance` 模式再测试；
  - 确认 DDR 跑分与公版差异，可采用 Antutu 应用跑分对比；
- SurfaceFlinger 触发丢帧逻辑：SR 单帧耗时通常大于一个 Vsync 时间，可能会触发 SurfaceFlinger 内部 FrameMiss 逻辑，建议集成 SVEP 的项目关闭该判断逻辑，补丁已上传至以下目录，请根据 Android 平台获取对应补丁：

```
resources/patch/android/..android-platform-dir../0001-Add-SVEP-compilati
on-option.patch
resources/patch/android/..android-platform-dir../0001-SurfaceFlinger-Runn
ing-SVEP-need-disable-Frame-Fence.patch
```

注意：..android-platform-dir.. 为具体 Android 版本及对应打补丁目录，例如：

```
./13/device/rockchip/common/
```

## 6.5 最新版本获取

SVEP 目前永久版本发布地址如下：

<https://console.box.lenovo.com/1/Q08d2o> 提取码：rksvep

目录结构如下图 6-5 SVEP 发布目录结构说明：

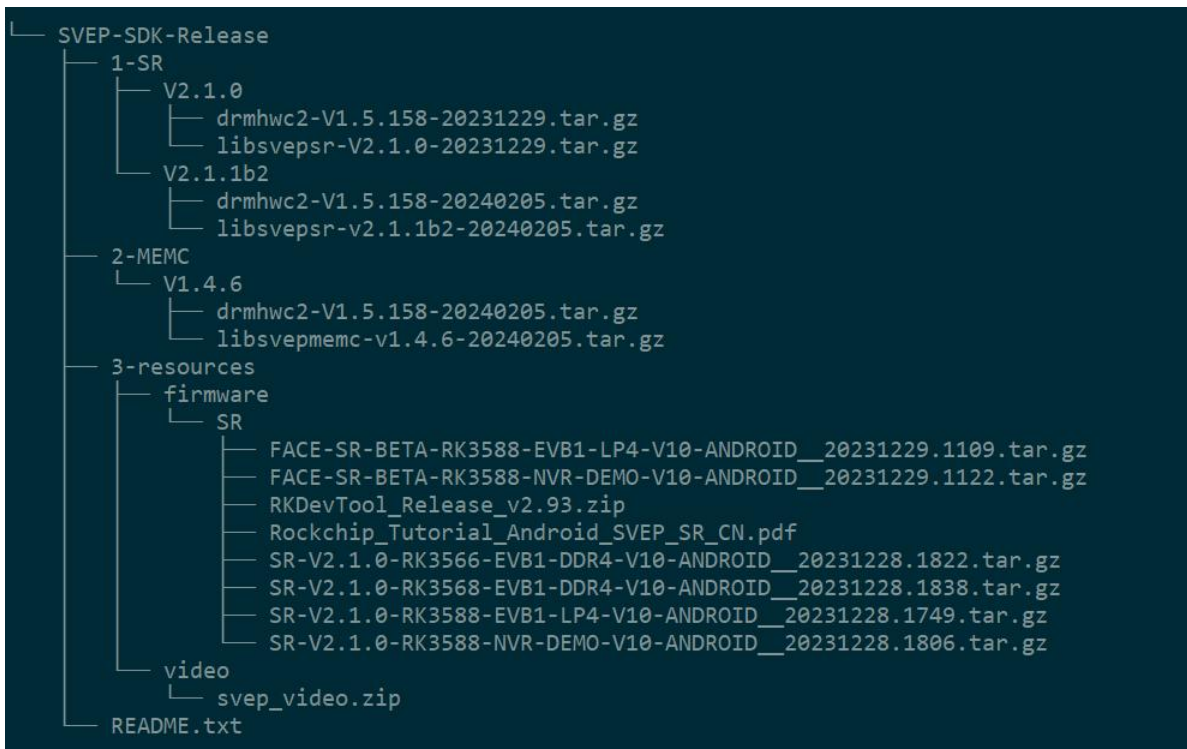


图 6-5 SVEP 发布目录结构说明

- 1-SR: SVEP-SR 版本发布目录说明
  - V2.1.0: 发布版本号
    - ◆ drmhwc2.tar.gz: 版本匹配的 HWC 源码版本, 请替换 SDK 如下目录:  
Path/to/sdk/hardware/rockchip/hwcomposer/drmhwc2
    - ◆ libsvepsr.tar.gz: 请替换 SDK 如下目录:  
Path/to/sdk/hardware/rockchip/libsvep/libsvepsr
- 2-MEMC: SVEP-MEMC 版本发布目录
  - V1.4.6: 发布版本号
- 3-Resource: 其他的资源文件, 例如视频, 演示固件等