Rockchip AiServer介绍

文件标识：RK-KF-YF-537

发布版本：V1.1.1

日期：2021-08-11

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

aiserver通过rockit框架构建媒体数据流pipeline，可对单个或者多个通路的media stream进行配置重组，实现下面这些功能的排列组合：

1. 摄像头设备采集。
2. 视频编码。
3. 支持rockx、rga、eptz、rkvo、fec等filter插件。
4. 用户可根据需求自定义扩展功能组件，如参考rockx插件实现第三方AI算法。

产品版本

芯片名称	内核版本
RV1109	Linux 4.19
RV1126	Linux 4.19
RK3568	Linux 4.19

读者对象

本文档主要适用于以下工程师：

- 技术支持工程师
- 软件开发工程师

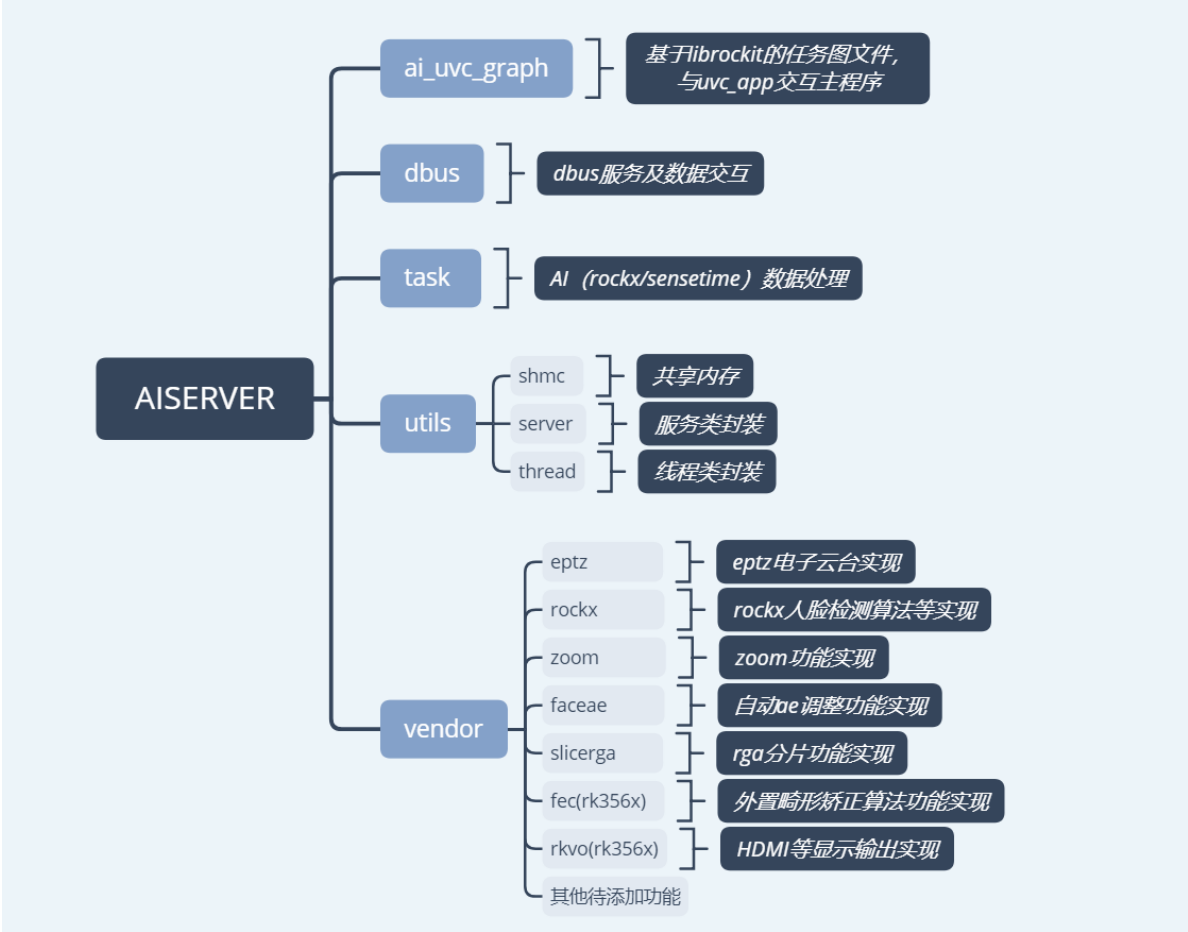
修订记录

日期	版本	作者	修改说明
2021/01/25	1.0.0	LQH	初始版本
2021/08/04	1.1.0	WT	更新功能说明
2021/08/11	1.1.1	WT	更新格式

目 录

1. 代码模块说明

1.1 目录结构



1.2 配置说明

SDK默认使用external\rockit\sdk\conf\aicamera_rockx.json配置文件（RK356X系列为external\rockit\sdk\conf\arch64\），编译后将默认安装在/oem/usr/share/aiserver/aicamera.json。

Json文件中的具体节点含义，请参考external\rockit\doc下《Rockchip_Developer_Guide_Linux_Rockit_CN.pdf》自动构建rockit应用章节。开发注意事项：

1. 对于已有的node_x，请保持原有编号。
2. 替换第三方算法模块，请使用node_11节点进行开发。
3. json文件下发的link_mode为内置连接通路，可根据需求进行修改。不建议新增自定义link_mode通路。

2. AIUVCGraph

AIUVCGraph是基于rockit开发（rockit相关资料查看《Rockchip_Developer_Guide_Linux_Rockit_CN.pdf》）的uvc图。主要实现从isp取流给uvc_app编码、eptz、zoom等流程处理。源码为ai_uvc_graph.cpp。

2.1 说明

这里基于rv1126的aicamera_uvc.json进行简单说明，查看link部分有如下表述，其中link_name为uvc的是正常预览所选择的通路，对应的为节点node_0、node_7，即isp的scale0到link_out的node7。

```
"node_1": {
    "node_opts": {
        "node_name"       : "rkisp"
    },
    "node_opts_extra": {
        "node_buff_type"  : 0,
        "node_buff_count" : 3,
        "node_buff_size"  : 5529600
    },
    "stream_opts": {
        "stream_output"   : "isp_bypass",
        "stream_fmt_out"  : "image:nv12"
    },
    "stream_opts_extra": {
        "opt_entity_name" : "rkispp_m_bypass",
        "opt_width"       : 2560,
        "opt_height"      : 1440,
        "opt_vir_width"   : 2560,
        "opt_vir_height"  : 1440,
        "opt_buf_type"    : 1,
        "opt_mem_type"    : 4,
        "opt_use_libv4l2" : 1,
        "opt_colorspace"  : 0
    }
},
"node_7": {
    "node_opts": {
        "node_name"       : "link_output"
    },
    "node_opts_extra": {
        "node_buff_type"  : 1,
        "node_buff_count" : 0
    },
    "stream_opts": {
        "stream_input"    : "uvc_link_out_in",
        "stream_output"   : "uvc_link_out_out",
        "stream_fmt_in"   : "image:nv12",
        "stream_fmt_out"  : "image:nv12"
    }
},
"link_0": {
    "link_name"          : "uvc",
    "link_ship"          : "0,7"
},
"link_7": {
    "link_name"          : "uvc_rga",
```

```
    "link_ship"          : "0,21,7"
}
```

2.2 FAQ

1. 如何选择link

   查看aiserver中的selectLinkMode代码部分。通过uvcMask来选择需要的link通路，用户也可自己增加link或者修改link_ship的组合来达到通路的选择或者切换。

2. 如何修改预览的分辨率

   如上所述，正常预览是通过scale0的输出，当uvc_app通过host选择分辨率后会传到aiserver的doUpdateCameraParams函数中进行通知，设置对应的分辨率。AIUVCGraph检测到设置的分辨率与上次不同会运行setupGraphAndWaitDone->setCameraParams进行参数的重新加载。

3. 如何修改其他isp节点的分辨率

   bypass分辨率只能输出最大分辨率，通过获取CAMERA_MAX_WIDTH及CAMERA_MAX_HEIGHT的环境变量作为bypass分辨率，不能修改为其他分辨率。

   scale1/2的分辨率默认通过json配置，如果需要动态修改则参考预览分辨率的修改方式，在切换分辨率的时候进行修改。

4. 如何获取节点的数据流

   以自定义节点60（自定义的节点序号请勿与AIUVCGraph中已经使用的重复，已经定义使用的节点请查看ai_uvc_graph.cpp开头部分。），获取双摄第二个sensor的scale0为例，`"opt_camera_index"=1`说明是双摄中的第二个sensor。AIUVCGraph中默认没有对60节点的分辨率进行修改，即60节点的分辨率为json配置的是1280*720。link使用默认的uvc通路，即预览情况下会走到这个通路。增加60,8这个链路，数据会走到8节点中，参考observeUVCOutputStream函数对节点8进行监听即可取到自定义60节点的输出数据。

   数据统一通过RTMediaBuffer进行传递，其中：

   RTMediaBuffer->getData()获取数据，

   RTMediaBuffer->getFd()获取数据硬件句柄fd，

   RTMediaBuffer->getLength()获取数据长度。

   如果上一级是isp输出，则可以通过以下两个函数来获取图像的宽w，高h。

   RTMediaBuffer->getMetaData()->findInt32(kKeyVCodecWidth, &w);
   RTMediaBuffer->getMetaData()->findInt32(kKeyVCodecHeight, &h);

```
"node_60": {
    "node_opts": {
        "node_name"       : "rkisp"
    },
    "node_opts_extra": {
        "node_buff_type"  : 0,
        "node_buff_count" : 3,
        "node_buff_size"  : 1382400
    },
    "stream_opts": {
        "stream_output"   : "isp1_scale_0",
        "stream_fmt_out"  : "image:nv12"
    },
    "stream_opts_extra": {
        "opt_entity_name" : "rkispp_scale0",
```

```
        "opt_width"       : 1280,
        "opt_height"      : 720,
        "opt_vir_width"   : 1280,
        "opt_vir_height"  : 720,
        "opt_buf_type"    : 1,
        "opt_mem_type"    : 4,
        "opt_use_libv4l2" : 1,
        "opt_colorspace"  : 0,
        "opt_camera_index": 1
    }
},
"node_8": {
    "node_opts": {
        "node_name"       : "link_output"
    },
    "node_opts_extra": {
        "node_buff_type"  : 1,
        "node_buff_count" : 0
    },
    "stream_opts": {
        "stream_input"    : "isp1_link_out_in",
        "stream_output"   : "isp1_link_out_out",
        "stream_fmt_in"   : "image:nv12",
        "stream_fmt_out"  : "image:nv12"
    }
},
"default_mode_link": "none",
"link_0": {
    "link_name"          : "uvc",
    "link_ship"          : "0,7-60,8"
},
```

5. 如何封装/解封装nn数据

   参考RTVfilterRockx.cpp中的fillAIResultToMeta，通过RtMetaData对nn数据进行封装。数据流通过nn注册的回调函数nn_data_output_callback，调用processAIData->getAIDetectResults进行nn数据包的获取，通过postNNData解析后发送到smart_display_service，进而通过rndis发送到host端。

6. 如何对节点进行特殊参数设置

   自定义插件的节点需要实现invokeInternal函数的继承，可以参考eptz节点的实现；调用者则参考setEPTZ函数实现通过对特定节点进行控制。

   调用者:

```
int32_t AISceneDirector::setEPTZ(const AI_UVC_EPTZ_MODE &mode, const int32_t
&val) {
    ...
    RtMetaData meta;
    meta.setInt32(kKeyTaskNodeId, ZOOM_NODE_ID); //ZOOM_NODE_ID为需要设置参数的
节点序号
    meta.setCString(kKeyPipeInvokeCmd, "set_pan"); //"set_pan"为设置命令
    meta.setInt32("value", val); //"value"为设置的数值
    mUVCGraph->invoke(GRAPH_CMD_TASK_NODE_PRIVATE_CMD, &meta);//invoke会调用到
对应节点的invokeInternal实现
    ...
}
```

被调用者：

```
RT_RET RTNodeVFilterZoom::invokeInternal(RtMetaData *meta) {
    ...
    RtMutex::RtAutolock autoLock(mLock);
    meta->findCString(kKeyPipeInvokeCmd, &command);
    RTSTRING_SWITCH(command) {
        RTSTRING_CASE("set_pan"):
            RT_ASSERT(meta->findInt32("value", &mPanValue));
            mPanValue = mPanValue * 2;
            break;
    ...
    }
    ...
    return RT_OK;
}
```

3. 插件

3.1 EPTZ

Rockchip Linux平台支持EPTZ电子云台功能，指通过软件手段，结合智能识别技术实现预览界面的“数字平移- 倾斜- 缩放/变焦”功能。配合RK ROCKX人脸检测算法，快速实现预览画面人物聚焦功能，可应用于视屏会议等多种场景。

3.1.1 集成说明

EPTZ模块支持库为libeptz.so，通过对EptzInitInfo结构体进行配置，实现相应的操作。相关代码位于SDK以下路径：

```
app/aiserver/src/vendor/samples/filter/eptz/
```

具体接口可参考eptz_algorithm.h文件，eptz版本说明详见app\aiserver\src\vendor\samples\filter\eptz\release_note.txt。

从v1.0.5版本开始支持两种模式：1.灵动模式；2.会议模式。

3.1.2 功能验证

RV1126/RV1109使用EPTZ功能，需将dts中的otp节点使能，evb默认配置中已将其使能。

```
&otp {
    status = "okay";
};
```

在RV1126/RV1109中，提供三种方案进行AUTO EPTZ功能验证及使用。

- 环境变量：在启动脚本（例如：RkLunch.sh）中添加环境变量export ENABLE_EPTZ=1，默认开启EPTZ功能，在所有预览条件下都将启用人脸跟随效果。
- XU控制：通过UVC扩展协议，参考5.1中描述进行实现。当uvc_app接收到XU的CMD_SET_EPTZ(0x0a)指令时，将根据指令中所带的int参数1或0，进行EPTZ功能的开关，以确认下次预览时是否开启人脸跟随效果。
- dbus指令：最新版本已支持通过dbus指令通知aiserver进程跨进程动态启动AUTO EPTZ能力。

```
#开启命令
dbus-send --system --print-reply --type=method_call --
dest=rockchip.aiserver.control
/rockchip/aiserver/control/graph rockchip.aiserver.control.graph.EnableEPTZ
int32:1
#关闭命令
dbus-send --system --print-reply --type=method_call --
dest=rockchip.aiserver.control
/rockchip/aiserver/control/graph rockchip.aiserver.control.graph.EnableEPTZ
int32:0
```

RV1126/RV1109显示预期效果:

- 单人：在camera可视范围内，尽可能将人脸保持在画面中间。
- 多人：在camera可视范围内，尽可能的显示人多画面，且将其保持在画面中间。

3.1.3 测试用例

1. 单人场景
    - 横向移动

测试项目	**camera预览横向跟踪**
测试目的	检测单人横向跟踪效果
测试步骤	1.camera端打开eptz功能 2.camera预览中仅存单人进行横向移动
预期结果	camera画面跟随人脸，在预览范围使人脸尽可能保持画面中心。

    - 纵向移动

测试项目	**camera预览纵向跟踪**
测试目的	检测单人纵向跟踪效果
测试步骤	1.camera端打开eptz功能 2.camera预览中仅存单人进行起立，下蹲活动
预期结果	camera画面跟随人脸，在预览范围使人脸尽可能保持画面中心。

    - 远近距离移动

测试项目	**camera预览横向跟踪**
测试目的	检测单人自动zoom聚焦效果
测试步骤	1.camera端打开eptz功能 2.camera预览中仅存单人，从距离摄像头0.5米开始，逐渐移到到5米左右距离。 3.camera预览中仅存单人，从距离摄像头5米开始，逐渐移到到0.5米左右距离。
预期结果	camera画面跟随人脸，在远距离时聚焦人脸，在近距离时将视角放大显示。

2. 多人场景
    - camera zoom

测试项目	**camera zoom功能**
测试目的	检测camera zoom 放大效果
测试步骤	1.camera端打开eptz功能 2.camera预览中仅存单人，在预览界面左边或右边边缘加入1名人员。 3.当两名人间之间的距离从小变大，或从大到小时，将会产生对应的视觉变化效果。
预期结果	1.当在预览界面左边或右边边缘加入人员，camera将实现zoom放大。 2.当两名人间之间的距离从小变大，在一定范围内视角变大。 3.当两名人间之间的距离从大变小，在一定范围内视角变小。 4.使多人计算的总范围尽可能保持画面中心。

    - 横向移动

测试项目	**camera多人预览横向跟踪**
测试目的	检测多人下横向跟踪
测试步骤	1.camera端打开eptz功能。 2.camera预览中存在多人。 3.多人同时向左边或右边进行移动。 4.部分人员保持不动，部分人员向左边移动。 5.部分人员保持不动，部分人员向右边移动。 6.部分人员向左移动，部分人员向右移动。
预期结果	camera预览使多人人脸确定的总范围尽可能保持画面中心。 针对第3点，camera预览界面跟随多人向左或向右移动。 针对第4点，camera预览界面保持不动或向左移动。 针对第5点，camera预览界面保持不动或向右移动。 针对第6点，camera预览界面基本保持不动。

    - 纵向移动

测试项目	**camera多人预览纵向跟踪**
测试目的	检测多人下纵向跟踪
测试步骤	1.camera端打开eptz功能。 2.camera预览中存在多人。 3.多人同时站于高位或下蹲。 4.部分人员保持不动，部分人员下蹲。 5.部分人员保持不动，部分人员站在高位。 6.部分人员下蹲，部分人员站于高位。
预期结果	camera预览使多人人脸总范围尽可能保持画面中心。 针对第3点，camera预览界面跟随多人向上或向下移动。 针对第4点，camera预览界面保持不动或向下移动。 针对第5点，camera预览界面保持不动或向上移动。 针对第6点，camera预览界面基本保持不动。

3.1.4 调试手段

- touch /tmp/eptz_face_debug：动态输出人脸坐标信息
- touch /tmp/eptz_zoom_debug：动态打开zoom日志，显示当前人脸范围和裁剪比例数据
- 设置环境变量export eptz_log_level=3：打开eptz控制算法，坐标计算debug信息
- touch /tmp/eptz_mode1: 动态切换到灵动模式，跟随灵敏。
- touch /tmp/eptz_mode2: 动态切换到会议模式，人物稳定后开始画面切换。

3.1.5 ROCKX算法模型替换

SDK默认使用rockx_face_detect_v3模型，但rockx同时有提供rockx_face_detect_v2及rockx_face_detect_v2_h模型，三者数据对比如下：

场景	识别有效距离	AI数据帧率	DDR带宽	aiserver Uss 内存
1080mjpeg 预览 scale1 720 nn rockx_face_detect_v2	5米左右	10fps	3172MB/s	106740K
1080mjpeg 预览 scale1 720 nn rockx_face_detect_v2_h	5米左右	15fps	3112MB/s	105232K
1080mjpeg 预览 scale1 720 nn rockx_face_detect_v3	2米左右	30fps	2881MB/s	105596K

EPTZ对AI数据帧率要求不高，因此近距离场景建议使用**rockx_face_detect_v3**模型，较远距离场景使用**rockx_face_detect_v2_h**模型。

SDK默认使用**rockx_face_detect_v3**，模型替换为**rockx_face_detect_v2_h**需执行以下步骤：

1. 将external/rockx/目录下的**rockx_face_detect_v2_horizontal.data**打包到usr/lib或oem/usr/lib目录下

```
diff --git a/sdk/rockx-rv1109-Linux/RockXConfig.cmake b/sdk/rockx-rv1109-
Linux/RockXConfig.cmake
index dd77dc7..151ed97 100644
--- a/sdk/rockx-rv1109-Linux/RockXConfig.cmake
+++ b/sdk/rockx-rv1109-Linux/RockXConfig.cmake
@@ -39,7 +39,7 @@ if(EXISTS "${CMAKE_CURRENT_LIST_DIR}/../rockx-data-rv1109")
        set(ROCKX_DATA_FILES  ${ROCKX_DATA_FILES}
"${CMAKE_CURRENT_LIST_DIR}/../rockx-data-rv1109/carplate_recognition.data")
    endif()
    if(${WITH_ROCKX_FACE_DETECTION})
-        set(ROCKX_DATA_FILES  ${ROCKX_DATA_FILES}
"${CMAKE_CURRENT_LIST_DIR}/../rockx-data-rv1109/face_detection_v3.data")
+        set(ROCKX_DATA_FILES  ${ROCKX_DATA_FILES}
"${CMAKE_CURRENT_LIST_DIR}/../rockx-data-
rv1109/rockx_face_detect_v2_horizontal.data")
    endif()
    if(${WITH_ROCKX_FACE_RECOGNITION})
        set(ROCKX_DATA_FILES  ${ROCKX_DATA_FILES}
"${CMAKE_CURRENT_LIST_DIR}/../rockx-data-rv1109/face_recognition.data")
```

2. 修改app/aiserver/src/vendor/CMakeLists.txt

```
diff --git a/src/vendor/CMakeLists.txt b/src/vendor/CMakeLists.txt
index d3c8774..ffd177a 100755
--- a/src/vendor/CMakeLists.txt
+++ b/src/vendor/CMakeLists.txt
@@ -25,7 +25,7 @@ if (${ENABLE_SAMPLE_NODE_EPTZ})
    )
endif()

-option(ENABLE_SAMPLE_NODE_ROCKX  "enable sample node rockx" OFF)
+option(ENABLE_SAMPLE_NODE_ROCKX  "enable sample node rockx" ON)
```

修改后重新编译rockx模块、aiserver模块即可。

后续可以通过修改aicamera.json文件node_4、node_11中的opt_rockx_model，替换其他算法模型进行效果验证，推荐rockx_face_detect_v2_h和rockx_face_detect_v3模型。

若运行时提示 xxx model data not found, 需将对应模型文件从external/rockx/sdk/rockx-data-rv1109/下拷贝到/usr/lib或oem/usr/lib目录下。

备注：若sensor为2K以上分辨率，建议将RTNodeVFilterEptzDemo.cpp中的eptz_npu_width和eptz_npu_height修改为1280和720，同时将aicamera.json中的node_2节点opt_width、opt_height、opt_vir_width、opt_vir_height修改为1280、720、1280、720，可以提高人脸检测的识别率和准备率。若sensor做多只支持bypass 1920x1080输出，则相应节点需修改为640x360。

3.1.6 第三方检测算法集成

第三方算法集成，可参考external/rockit/doc/《Rockchip_Developer_Guide_Linux_Rockit_CN.pdf》文档进行开发。

具体代码demo可参考以下目录：

```
app/aiserver/src/vendor/samples/filter/rockx/
```

结合EPTZ功能使用时，需注意RTNodeVFilterEptzDemo.cpp中传入的AI数据结构要同步修改。

3.1.7 Q&A

1. 人脸识别率不够，或识别距离较近该如何改善？

   目前EPTZ算法，人脸检测基于RK ROCKX模块，使用ROCKX中的NPU人脸检测算法实现。可参考3.1节使用rockx_face_detect_v2_h进行验证，2K分辨率以上sensor使用1280、720输入给算法。若还不能满足场景要求，建议使用第三方算法。

2. eptz初始化参数详细代表什么意思？

   - eptz_src_width、eptz_src_height：sensor bypass支持的最大输出分辨率，RkLunch.sh中通过环境变量获取。
   - eptz_dst_width、eptz_dst_height：eptz算法检测出的目标区域，即裁剪放大区域。默认设置为同eptz_src_width、eptz_src_height一致，即第一次打开预览全视角显示。
   - camera_dst_width、camera_dts_height：当前实际预览分辨率，由uvc_app传入，可以做相关变量参数配置使用。
   - eptz_npu_width、eptz_npu_height：输入算法的图像分辨率大小。
   - eptz_facedetect_score_shold：人脸检测算法数据中的阈值，通过该值可以将低质量的人脸过滤排除。
   - eptz_zoom_speed：zoom速度调整，可设置1,2,3，默认为1，其中3的速度最快。
   - eptz_fast_move_frame_judge：忽略人物x帧内快速移动的数据，人物移动防抖阈值。
   - eptz_zoom_frame_judge：忽略人物x帧内zoom比例变化数据，人物移动ZOOM防抖阈值。
   - eptz_threshold_x、eptz_threshold_y: x，y方向移动阈值，当x，y的范围变化超过设定值时进行移动调整。
   - mLastXY：初始化时，第一次显示的范围区域大小。

3. 人物移动时，发现画面人物跟随不上如何解决？

   - 可自行添加打印查看人脸检测算法帧率，AI帧率过低则可能存在现象，可以考虑算法优化。
   - 小幅度修改eptz_iterate_x和eptz_iterate_y的值可加快跟随速度，但修改值太大会造成画面移动不顺滑。

4. 是否支持动态调整eptz初始化参数，如何进行修改配置？

   支持动态修改阈值等参数，可在open或process方法中直接修改mEptzInfo对象参数，可参考RTNodeVFilterEptzDemo.cpp。

5. eptz_zoom.conf需要如何调整？其中参数对应关系指什么？

   area_ratio_data 是检测到的人脸范围，clip_ratio_data 是对应的裁剪范围。

   eptz_zoom-3840x2160.conf指初略适配3840x2160sensor的配置文件，其中的内容：

   ```
   area_ratio_data:0.035,0.07,0.3,0.55,0.75,1.0
   clip_ratio_data-480p:0.10,0.30,0.45,0.70,0.85,1.0
   clip_ratio_data-720p:0.10,0.30,0.45,0.70,0.85,1.0
   clip_ratio_data-1080p:0.10,0.30,0.45,0.70,0.85,1.0
   clip_ratio_data-1440p:0.10,0.30,0.45,0.70,0.85,1.0
   clip_ratio_data-2160p:0.10,0.30,0.45,0.70,0.85,1.0
   ```

   - 当人脸的范围在画面中所占比例小于等于0.035时，显示范围区域为sensor全视角的0.10，产生聚焦效果。
   - 当人物靠近sensor或人数增多时，画面占比逐渐变大，如超过0.3小于0.55，显示范围区域为sensor全视角的0.7，视角放大。
   - clip_ratio_data-480p代表预览分辨率为480P时，eptz模块读取的对应比例参数，clip_ratio_data-720p代表预览分辨率为720P时，eptz模块读取的对应比例参数，以此类推。

SDK默认提供最大分辨率为1440P和2160P的配置文件eptz_zoom-2560x1440.conf和eptz_zoom-3840x2160.conf。对应的NPU输入为1280x720。

用户定制对应eptz_zoom.conf文件时，通过touch /tmp/eptz_zoom_debug动态打开日志，查看当前人脸范围和裁剪比例数据 face_ratio[%.2f] eptz_clip_ratio[%.2f]。固定使用同一预览分辨率，分别在0.5m、1m、2m、3m|、5m（**距离仅作举例说明，可以减少或增加其中的部分数据**）等距离测量单人及多人时的face_ratio数值，配上期望的eptz_clip_ratio数值。如在1080P分辨率下，测试出一组满意的eptz_clip_ratio数据后，如:

```
area_ratio_data:0.035,0.07,0.3,0.55,0.75,1.0
clip_ratio_data-1080p:0.10,0.30,0.45,0.70,0.85,1.0
```

可以将该比例copy到其他预览分辨率数据下，如下所示：

```
clip_ratio_data-480p:0.10,0.30,0.45,0.70,0.85,1.0
clip_ratio_data-720p:0.10,0.30,0.45,0.70,0.85,1.0
clip_ratio_data-1080p:0.10,0.30,0.45,0.70,0.85,1.0
clip_ratio_data-1440p:0.10,0.30,0.45,0.70,0.85,1.0
clip_ratio_data-2160p:0.10,0.30,0.45,0.70,0.85,1.0
```

若存在360P或其他分辨率，则会自适应最接近的分辨率使用其参数。clip_ratio_data-480p、clip_ratio_data-720p、clip_ratio_data-1080p、clip_ratio_data-1440p、clip_ratio_data-2160p数据都存在eptz_zoomm.conf即可。

6. 是否支持eptz_zoom.conf文件动态修改？

   支持，可调用changeEptzConfig接口，使用其他eptz_zoon.conf文件来进行参数更新。

7. 其他

   其他EPTZ问题，建议redmine上提交，并复上对应的日志以及视频文件。我司将安排人员进行分析解决。

3.2 RKVO

RK356X系列拥有HDMI、VGA等接口，部分客户有需求将Camera数据输出至显示器上，为此开发了RKVO插件，可将VI数据发送至HDMI、VGA等输出设备上。

相关代码位于SDK以下路径：

```
app/aiserver/src/vendor/samples/filter/rkvo/
```

VO接口为Rockchip rockit媒体框架的一部分，具体接口可查阅rockit VO文档，文档位于`external\rockit\doc\Rockchip_Developer_Guide_MPI_VO.pdf`。

3.2.1 使用说明

RKVO的节点配置位于`external\rockit\sdk\conf\arch64\aicamera_uvc_zoom_rkvo.json`，其中节点node_9即为RKVO的配置

```
"node_9": {
    "node_opts": {
        "node_name"        : "rkvo"
    },
    "node_opts_extra": {
        "node_buff_type"   : 1,
        "node_buff_count" : 0
    },
    "stream_opts": {
        "stream_input"     : "vo_in",
        "stream_output"    : "vo_out",
        "stream_fmt_in"    : "image:nv12",
        "stream_fmt_out"   : "image:nv12"
    }
}
```

在link_0中将从节点node_0或节点node_1中取出的Camera数据送至node_9后，即会在`app\aiserver\src\vendor\filter\rkvo\RTNodeVFilterVideoOutput.cpp`文件中将数据送至显示设备中。

```
"link_0": {
    "link_name"           : "uvc",
    "link_ship"           : "0,7-0,9"
}
```

客户可依据产品形态，对`/oem/usr/share/aiserver/aicamera.json`做出对应的修改，将Camera数据送至显示设备。

3.3 FEC（鱼眼矫正）

RV1109/RV1126/RK356X系列均支持大广角Camera，并对广角Camera的畸变数据进行反畸变处理，该模块即为Rockchip开发的一款进行Camera数据反畸变的插件。如有需要请参考使用说明。

相关代码位于SDK以下路径：

```
app/aiserver/src/vendor/samples/filter/fec/
```

3.3.1 使用说明

在`app/aiserver/src/vendor/CMakeLists.txt`中合并以下补丁：

```
diff --git a/src/vendor/CMakeLists.txt b/src/vendor/CMakeLists.txt
index e1cb534..c42786b 100644
--- a/src/vendor/CMakeLists.txt
+++ b/src/vendor/CMakeLists.txt
@@ -28,15 +28,15 @@ if (${ENABLE_SAMPLE_NODE_EPTZ})
     )
 endif()

-option(ENABLE_SAMPLE_NODE_FEC  "enable node fec" OFF)
-if ({ENABLE_SAMPLE_NODE_FEC})
+option(ENABLE_SAMPLE_NODE_FEC  "enable node fec" ON)
+#if ({ENABLE_SAMPLE_NODE_FEC})
     aux_source_directory(filter/fec/src SRC_FILES_VENDOR)
     include_directories(filter/fec)
     include_directories(filter/fec/headers)
     include_directories(../utils/drm)
     install(FILES ${CMAKE_CURRENT_SOURCE_DIR}/filter/fec/libdistortion.so
DESTINATION ../../oem/usr/lib/)
     set(SRC_DEPEND_LIBS ${SRC_DEPEND_LIBS} drm)
-endif()
+    #endif()

 option(ENABLE_SAMPLE_NODE_ROCKX  "enable node rockx" OFF)
 if (${ENABLE_SAMPLE_NODE_ROCKX})
```

FEC的节点配置位于`external\rockit\sdk\conf\arch64\aicamera_uvc_zoom_fec.json`，其中节点node_22为FEC的配置

```
"node_22": {
    "node_opts": {
        "node_name"       : "rkfec"
    },
    "node_opts_extra": {
        "opt_width"       : 3840,
        "opt_height"      : 2160,
        "opt_vir_width"   : 3840,
        "opt_vir_height"  : 2160,
        "node_buff_type"  : 0,
```

```
            "node_buff_count" : 2,
            "node_buff_size"  : 12441600
        },
        "stream_opts": {
            "stream_input"    : "fec_in",
            "stream_output"   : "fec_out",
            "stream_fmt_in"   : "image:nv12",
            "stream_fmt_out"  : "image:nv12"
        }
    }
```

在link_0通路中，将从node_0或node_1取出的数据送至node_22节点，即会对数据进行反畸变处理。

反畸变处理需要有Camera的反畸变矫正文件，将该文件放入设备指定目录才可正常进行反畸变处理，该文件的生成请联系对应的Camera调试工程师。

`/oem/usr/share/mesh`：由ISP生成的反畸变矫正文件，放入该目录即使用ISP进行反畸变处理。

`/oem/usr/share/distortion`：由算法生成的反畸变矫正文件，放入该目录即使用GPU进行反畸变处理。