全链路低时延AI体感推理开发指南

文件标识：RK-KF-YF-F16

发布版本：V1.0.1

日期：2025-08-05

文件密级：□绝密 □秘密 □内部资料 ■公开

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，由其各自拥有者所有。

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

RK3576、RK3588等芯片，具有CPU、GPU、VPU和NPU算力，适合体感游戏机顶盒形态的产品。 体感游戏机顶盒一般集成多种体感检测模型(骨骼关键点、人脸关键点、手势识别、手掌关键点)做体感检测，然后再关联游戏业务逻辑。

概述

本文档介绍全链路低时延AI体感推理的技术方案。该技术方案涉及：摄像头框架、图像预处理、RKNN体感模型和游戏操控逻辑等主要环节。

产品版本

芯片名称	Android版本
RK3576/RK3588	14.0

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

日期	版本	作者	修改说明
2025-08-04	V1.0.0	Martin.Cheng	初始版本
2025-08-05	V1.0.1	Owen.Qiu、Martin.Cheng	增加打开V4L2双流的流程

目录

1. 技术概述

本文档介绍全链路低时延AI体感推理的技术方案。该技术方案涉及：摄像头框架、图像预处理、RKNN体感模型和游戏操控逻辑等主要环节。

1.1 时延分析

体感推理全链路中，摄像头框架的时延一般在2~3帧；图像预处理/RKNN体感模型的时延一般在1帧；游戏操控逻辑/视频预览的时延一般在2帧。

- **摄像头框架：** 包含Android Camera2框架和V4L2框架，仅适用基础的摄像头曝光处理和ISP图像处理，能够将摄像头框架时延控制在<2帧。
- **图像预处理/RKNN体感模型：** 图像格式、图像大小和模型效率对模型推理效率关系。体感模型rockx-pose具有16ms/帧的处理能力。
- **游戏操控逻辑/视频预览：** 体感模型的后处理操作。视频预览和POSE关键点处理需要经过SurfaceFlinger+VOP后端。

影响因素	光照环境 曝光时间	3A复杂度 ISP性能	CPU性能 NPU性能	GPU性能 Surflinger/VOP负载
关键步骤	Sensor曝光	ISP-3A处理	体感推理 RockxBodyPose	BodyPose渲染 视频预览
耗时分析	< 16ms	< 16ms	< 16ms	< 32ms

以1080P@60FPS摄像头为例
BodyPose推理时延：小于3Frame/48ms
BodyPose视频预览：小于4Frame/64ms

1.2 性能分析

RK3576/RK3588机顶盒通过将显示器分辨率设置到1080P@60FPS。运行RockxBodyPose.apk查看推理帧率和预览帧率。

查看性能参数，常用命令如下：

```
# 查看npu负载
rk3576_gamebox:/ # cat /sys/kernel/debug/rknpu/load
NPU load:  Core0: 49%, Core1:  0%
# 查看cpu负载
rk3576_gamebox:/ # busybox top
  PID  PPID USER     STAT   VSZ %VSZ CPU %CPU COMMAND
 3089   535 u0_a45   S <  4025m102.9   3 13.0 {ckchips.rockiva} com.rockchips.rockiva
rk3576_gamebox:/ # logcat -c; logcat -v time --pid=$(pidof com.rockchips.rockiva)
rk3576_gamebox:/ # setprop persist.rockx.log.level 5
```

2. 软件安装和部署方法

2.1 安装和部署虚拟摄像头补丁包

- **虚拟摄像头功能：** Android-14-Media-Streaming-SDK已经合并了虚拟摄像头功能。如果SDK不存在该功能，请在hardware/rockchip/camera目录，打上add-virtual-camera.patch；并编译camera.rk30board.so。更新板级库/vendor/lib64/hw/camera.rk30board.so
- **更新摄像头配置：** 参考camera3_profiles.xml配置参数；更新板级配置/vendor/etc/camera/camera3_profiles.xml

  在原先camera节点的基础上再添加一个m00-sub节点,两个节点信息要一样：

  ```
  <Profiles cameraId="0" name="ov5695" moduleId="m00">
  <Profiles cameraId="1" name="ov5695" moduleId="m00-sub">
  ```

  同时将 `<sensorType value="SENSOR_TYPE_SOC"/>` 这个属性全部配置成SENSOR_TYPE_SOC。

2.2 安装和部署启动脚本

产品级部署，考虑部署在/vendor/etc/init/bodypose.rc。启动时完成配置。

```
./data/high_perf_freq.sh
./vendor/bin/rkaiq_3A_server_64 &
chmod 777 /dev/video12
```

2.3 安装和部署RockxBodyPose.apk应用

```
adb install -r RockxBodyPose.apk
```

3. 体感模型低时延推理方案

3.1 摄像头框架优化

- 使能虚拟双camera功能

Android-14-Media-Streaming-SDK已经合并了虚拟摄像头功能。如果不存在该功能，请按照以下步骤操作。

1. 在hardware/rockchip/camera目录下:

打上0001-add-virtual-camera.patch补丁包，并在Android.mk打开添加

```
LOCAL_CFLAGS += -DRK_VIR_CAMERA
```

2. 参考附件的camera_profiles.xml的imx386的配置(注意要配置2个imx386节点)

3. 如何确认功能是否正常

```
dumpsys media.camera
```

- 查看camera数目是否是 2

3.2 体感推理使用V4L2接口取流

采集之前，需要将ISP的对应的selfpath节点的对应的video节点的权限放开，查看具体的ISP的videoX,可以通过media-ctl -p -d /dev/media1,查看selfpath对应的video的具体节点。

```
chmod 777 videoX
./vendor/bin/rkaiq_3A_server_64 &
```

使用V4L2命令测试是否可以正常出流(videoX改成具体的video节点)

```
v4l2-ctl -d /dev/videoX --set-fmt-video=width=1920,height=1080,pixelformat=NV12 --stream-mmap=5 --stream-skip=3  --stream-count=1000000 --stream-poll
```

采集代码参考{RockxBodyPose}/app/src/main/image_reader.cpp的V4L2采集代码部分。

3.3 RKNN体感模型优化

- Rocx-Pose模型：玩家ID会变化，最好提供具有唯一性的ID。
- Rocx-Pose模型：快速运动和局部遮挡时，局部存在误检。

3.4 体感推理全链路优化

3.4.1 RockxBodyPose模型使用V4L2直接从ISP读取数字图像

V4L2封装见: {RockxBodyPose}/app/src/main/image_reader.cpp

```
    NVImageReader* nv_image_reader_reader(NVImageReaderType readerType);
    void* nv_image_reader_open(NVImageReader* reader, const char* uri, int width, int
height);
    int nv_image_reader_dequeue(NVImageReader* reader, NVImage* image);
    int nv_image_reader_enqueue(NVImageReader* reader, NVImage* image, int index);
    int nv_image_reader_close(NVImageReader* reader);
```

3.4.2 RockxBodyPose模型使用native线程驱动图像取流和体感推理

推理流程见: {RockxBodyPose}/app/src/main/image_detect_rockx.cpp

```
// 摄像头图像的读取线程
void* thread_worker_reader(void* arg) {
    gTaskCtx.task_run = 1;
    while(gTaskCtx.task_run > 0) {
        // read image from reader
        int64_t time_now = nv_time_ms();
        NVImage image = {0};
        int index = nv_image_reader_dequeue(gTaskCtx.reader, &image);
        if (index >= 0) {
            pthread_mutex_lock(&gTaskCtx.buffer_mutex);
            while (gTaskCtx.buffers.size() >= 2) {
                // release used buffers.
                NVImage img_old = gTaskCtx.buffers.back();
                nv_image_reader_enqueue(gTaskCtx.reader, &img_old, img_old.index);
                gTaskCtx.buffers.pop_back();
            }
            image.index     = index;
            image.timestamp = nv_time_ms();
            gTaskCtx.buffers.push_front(image);
            pthread_cond_signal(&gTaskCtx.buffer_ready);
            pthread_mutex_unlock(&gTaskCtx.buffer_mutex);
        }
    }
    return nullptr;
}

// BodyPose推理和推理结果回调线程
void* thread_worker_inference(void* arg) {
    gTaskCtx.task_run = 1;
    while(gTaskCtx.task_run > 0) {
        char *json_bodypose = nullptr;
        rockx_task_process(nullptr, &json_bodypose);
```

```
        if (nullptr != gTaskCtx.onListener) {
            // 推理结果保存在JSON，通过JNI回调到JAVA层
            gTaskCtx.onListener(json_bodypose, nullptr);
        }
        if (nullptr != json_bodypose) {
            free(json_bodypose);
        }
    }
    return nullptr;
}
```

3.4.3 RockxBodyPose应用显示视频预览和体感推理结果

- 应用activity见，{RockxBodyPose}/app/src/main/java/com/rockchips/rockiva/BodyPoseActivity.java
- 视频预览见，{RockxBodyPose}/app/src/main/java/com/rockchips/rockiva/util/Camera2Reader.java
- 体感推理见，{RockxBodyPose}/app/src/main/java/com/rockchips/rockiva/NVTaskRockx.java

```
public class BodyPoseActivity extends AppCompatActivity {
    // 省略部分代码(仅说明意图)

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // 省略部分代码(仅说明意图)
        // Camera2Reader封装了android camera2预览的代码

        // initCamera();
        mSurfaceView = findViewById(R.id.camera_preview);
        mSurfaceView.getHolder().addCallback(new SurfaceHolder.Callback() {
            @Override
            public void surfaceCreated(SurfaceHolder holder) {
                if (mCameraReader== null) {
                    mCameraReader = new Camera2Reader(BodyPoseActivity.this,
holder.getSurface());
                }
                mCameraReader.startCamera(mOnImageListener);
            }
        }

        // NVTaskRockx封装了体感推理的JNI代码，JNI通过回调触发OnTaskListener，然后JAVA解析和渲染体感
识别结果。
        mTaskRockx = new NVTaskRockx(null, null);
        mTaskRockx.onCreate();
        mTaskRockx.setTaskListener(new NVTaskRockx.OnTaskListener() {
            @Override
            public void onListener(String jsonString) {
                // parse and render detected objects
                parseJsonObjects(jsonString);
            }
        });
```

```
        // 省略部分代码(仅说明意图)
    }
```

3.4.4 调整系统性能参数

参考 附件:high_perf_freq.sh

```
echo "CPU available frequencies:"
cat /sys/devices/system/cpu/cpufreq/policy0/scaling_available_frequencies
cat /sys/devices/system/cpu/cpufreq/policy4/scaling_available_frequencies
echo "Fix CPU max frequency:"
echo userspace > /sys/devices/system/cpu/cpufreq/policy0/scaling_governor
echo 2208000 > /sys/devices/system/cpu/cpufreq/policy0/scaling_setspeed
cat /sys/devices/system/cpu/cpufreq/policy0/scaling_cur_freq
echo userspace > /sys/devices/system/cpu/cpufreq/policy4/scaling_governor
echo 2304000 > /sys/devices/system/cpu/cpufreq/policy4/scaling_setspeed
cat /sys/devices/system/cpu/cpufreq/policy4/scaling_cur_freq
```