密级状态：绝密(　　)　　秘密(　　)　　内部(　　)　　公开(√)

Rock-X SDK 开发指南

（技术部，图形显示平台中心）

文件状态： [　] 正在修改 [√] 正式发布	当前版本：	V1.0
	作　　者：	HPC&AI Team
	完成日期：	2019-06-11
	审　　核：	熊伟 卓鸿添
	完成日期：	2019-06-11

福州瑞芯微电子股份有限公司

Fuzhou　Rockchips　Semiconductor　Co . , Ltd

更新记录

版本	修改人	修改日期	修改说明	核定人
V1.0	杨华聪、吴丽娟	2019-06-11	初始版本	熊伟 卓鸿添

目 录

1 主要功能说明 5

2 系统依赖说明 5

2.1 RK3399PRO 系统依赖 5

2.2 RK1808 系统依赖 5

3 示例应用 6

3.1 命令行执行程序示例 6

3.2 ANDROID 程序示例 6

4 性能指标 7

4.1 模块精度指标 7

4.1.1 目标检测 7

4.1.2 人脸检测 7

4.1.3 人脸识别 8

4.1.4 车牌识别 8

4.1.5 人脸属性分析 9

4.1.6 人脸特征点定位 9

4.1.7 人体骨骼点关键点 9

4.2 模块运行性能 10

5 SDK 使用说明 10

5.1 SDK 库引入 10

5.2 SDK 裁减说明 11

5.3 RK1808 计算棒配置 12

5.4 初始化和释放 12

5.5 接口调用 13

5.6 API 参考指南 14

1 主要功能说明

Rock-X SDK 是基于 RK3399Pro/RK1808 平台的一套 AI 组件库。开发者通过 Rock-X SDK 提供的 API 接口能够快速构建 AI 应用。

当前 SDK 提供的功能如表 1-1 所示。

表 1-1 Rock-X SDK 主要功能

类别	功能
目标检测	人头检测、人车物检测
人脸	人脸关键点、人脸属性分析、人脸识别
车牌	车牌检测、车牌识别
人体关键点	人体骨骼关键点、手指关键点

2 系统依赖说明

2.1 RK3399Pro 系统依赖

在 RK3399Pro 平台上，SDK 所提供的库和应用程序需要 RKNN 驱动版本为 0.9.6。在 RK3399Pro Android/Linux 平台上运行 Demo 应用以后，通过日志能够看到如下的驱动信息，请确保 DRV 版本为 0.9.6。

```
=============================================
RKNN VERSION:
   API: 0.9.5 (a949908 build: 2019-05-07 22:20:52)
   DRV: 0.9.6 (c12de8a build: 2019-05-06 20:10:17)
=============================================
```

2.2 RK1808 系统依赖

在 RK1808 Linux 平台上，本 SDK 提供的库和应用程序需要 NPU 驱动版本需要 rknn_runtime 版本在 0.9.8 以上，在 RK1808 平台上查看 rknn_runtime 版本的方法如下所示：

```
$ strings /usr/lib/librknn_runtime.so |grep "librknn_runtime version"
librknn_runtime version 0.9.8 (8dcfdc7 build: 2019-05-31 11:23:34 base:
110)
```

3 示例应用

Rock-X SDK 中附带的示例应用能够帮助开发者评估 SDK 的基本功能和理解 SDK 接口，开发者可以直接基于示例应用进行修改或参考开发。示例应用代码包括命令行执行程序示例和 Android 程序示例。

3.1 命令行执行程序示例

Rock-X SDK 提供了所有接口的命令行执行程序代码示例，示例程序支持在 RK3399Pro Android/Linux 平台、RK1808 Linux 平台以及 x86-64 Linux（仅支持 RK1808 计算棒）平台上运行，编译和运行方法请参见“demo/command_line_demo”目录下的 README 文件。

3.2 Android 程序示例

Android 示例程序支持在 RK3399Pro Android 平台上运行。所有的 Android 示例程序代码位于“demo/rk3399pro_android_demo”目录下。将对应的 Android 示例程序解压，再通过 Android Studio 打开后，可以直接进行编译、运行和开发。

4 性能指标

4.1 模块精度指标

4.1.1 目标检测

表 4-1 检测类性能表

模块	数据集	性能指标
人头检测	Brainwash	mAP@IOU0.5 = 0.704
车牌检测	CCPD	mAP@IOU0.5 = 0.983
人车物检测	MSCOCO_VAL2017	mAP@IOU0.5 = 0.565

注：

1) mAP@IOU0.5=0.704 表示 IOU=0.5 时对应的 mAP=0.704。

2) Brainwash 是用于人头检测的公开数据集，主要场景为咖啡店，共 5007 张。

3) CCPD(Chinese City Parking Dataset)是国内车牌数据集，从中随机抽取 10000 张进行测试。

4) MSCOCO_VAL2017 是目标检测公开数据集，使用该数据集中的 5000 张验证集测试，共 91 类别。

5) 人头检测模块的最小检测尺寸为图像分辨率的 1/19。

6) 车牌检测模块支持检测国内蓝色、黄色和绿色车牌。

4.1.2 人脸检测

表 4-2 人脸检测性能

参数	性能指标
适应角度	平面内人脸左右旋转±45° 侧脸左右偏转±60° 侧脸上偏转 60° 侧脸下偏转 45°
最大距离	11 米(测试摄像头 FOV=60°)
mAP	mAP@IOU0.5=0.857

注：

1) 图像质量较差时，支持的检测角度会减小。

2) 最大检测距离与摄像头 FOV 等参数有关。

3) 检测的最小人脸尺寸为图像分辨率的 1/19。

4.1.3 人脸识别

表 4-3 人脸识别性能

参数	性能指标
识别角度	平面内人脸左右旋转±45° 侧脸左右偏转±60° 侧脸上偏转 60° 侧脸下偏转 45°
识别距离	11 米(测试摄像头 FOV=60°)
识别精度(LFW 标准数据集)	99.65%±0.00088
参考精度	TPR=0.992@FAR=0 TPR=0.995@FAR=0.001

注：

1) 实际应用中，对距离和角度稍加限制，能获得更好的识别结果，用户可根据实际情况进行质量筛选。

4.1.4 车牌识别

表 4-4 车牌识别性能表

数据集	性能指标
CCPD	83.31%(8331/10000)

注：

1) CCPD(Chinese City Parking Dataset)是国内车牌数据集，从中随机抽取 10000 张进行测试。

2) 支持识别国内蓝色、绿色和黄色车牌。

3) 可识别的车牌字符如表 4-5 所示。

表 4-5 车牌识别字符表

字符类别	可识别字符
省份中文字符	京 沪 津 渝 冀 晋 蒙 辽 吉 黑 苏 浙 皖 闽 赣 鲁 豫 鄂 湘 粤 桂 琼 川 贵 云 藏 陕 甘 青 宁 新
数字和字母	0 1 2 3 4 5 6 7 8 9 A B C D E F G H J K L M N P Q R S T U V W X Y Z
车牌用途中文字符	港 学 使 警 澳 挂 军 北 南 广 沈 兰 成 济 海 民 航 空

4.1.5 人脸属性分析

表 4-6 性别年龄性能

数据集	年龄精度	性别精度
UTK_asian	4.823283	92.96%(2220/2388)

注：

1) UTK_asian 是 UTK 公开数据集的亚洲人部分，使用 7-70 岁数据进行测试，共 2388 张。

2) 年龄精度为平均年龄偏差。

4.1.6 人脸特征点定位

表 4-7 人脸特征点定位（68 点）性能

数据集	误差
300w_cropped	6.01%

注：

1) 误差计算公式如下

$$\text{error} = \frac{\sum_{j=1}^{68}[euclidean(d(j) - g(j))]}{(68 * d)}$$

$\text{euclidean}\big(d(j) - g(j)\big)$ 表是第 j 个检测点与标注点之间的欧式距离。

d 表示左外眼角和右外眼角的欧式距离。

4.1.7 人体骨骼点关键点

表 4-8 人体骨骼关键点定位性能

数据集	性能指标
MSCOCO_VAL2017	mAP@OKS0.5=0.623

注：

1) mAP@OKS0.5=0.623 表示 OKS=0.5 时对应的 mAP=0.623。

2) MSCOCO val2017 是 COCO 2017 Keypoint Detection Task 的验证集，共 5000 张，其中 2000 多张有关键点。

4.2 模块运行性能

各模块运行时间和所需内存如表 4-9 所示。

表 4-9 模块运行时间和消耗内存

模块	运行时间(ms)	消耗 NPU 内存(MB)
人车物检测	49	66
人头检测	38	43
人脸检测	41	24
人脸特征点定位（68 点）	11	34
人脸姿态	2	21
人脸矫正对齐	9	20
人脸识别特征提取	44	117
人脸属性分析	16	19
车牌检测	46	39
车牌矫正对齐	21	22
车牌识别	39	21
身体骨骼关键点	106	119
手指关键点（21 点）	153	89
手指关键点（3 点）	23	21
检测目标跟踪	1	18

注：

1) 图中所测的内存值为峰值内存。

5 SDK 使用说明

5.1 SDK 库引入

各平台的 Rock-X SDK 库位于 sdk 目录下，如下所示：

```
sdk/
├── rockx-rk1808-Linux
├── rockx-rk3399pro-Android
├── rockx-rk3399pro-Linux
└── rockx-x86-64-Linux
```

开发者只需要在自己工程的 CMakeLists.txt 中引入对应平台的库即可，下面以 RK3399Pro Android 平台为例：

```
# Find RockX Package
set(RockX_DIR <path-to-rockx-sdk>/sdk/rockx-rk3399pro-Android)
find_package(RockX REQUIRED)

# Include RockX Header
include_directories(${RockX_INCLUDE_DIRS})

# Link RockX Libraries
target_link_libraries(target_name ${RockX_LIBS})
```

5.2 SDK 裁减说明

SDK 提供的库文件并不需要完全导入，开发者可以根据自己所用到的模块来引入所需的库，这样可以减少应用的体积。表 5-1 列出了 SDK 库文件说明，除了 librockx.so 和 librknn_api.so 两个库文件，对于没有使用的模块，可以直接移除相应的库文件。

表 5-1 SDK 库文件说明

库文件名	对应模块	可裁剪
libcarplate_align.so	ROCKX_MODULE_CARPLATE_ALIGN	是
libcarplate_detection.so	ROCKX_MODULE_CARPLATE_DETECTION	是
libcarplate_recognition.so	ROCKX_MODULE_CARPLATE_RECOG	是
libface_attribute.so	ROCKX_MODULE_FACE_ANALYZE	是
libface_detection.so	ROCKX_MODULE_FACE_DETECTION	是
libface_landmark5.so	ROCKX_MODULE_FACE_LANDMARK_5	是
libface_landmarks68.so	ROCKX_MODULE_FACE_LANDMARK_68	是
libface_recognition.so	ROCKX_MODULE_FACE_RECOGNIZE	是
libhead_detection.so	ROCKX_MODULE_HEAD_DETECTION	是
libobject_detection.so	ROCKX_MODULE_OBJECT_DETECTION	是
libpose_body.so	ROCKX_MODULE_POSE_BODY	是
libpose_finger.so	ROCKX_MODULE_POSE_FINGER_3	是
libpose_hand.so	ROCKX_MODULE_POSE_FINGER_21	是
librockx.so	SDK 基础库	否
librknn_api.so	SDK 依赖库	否

5.3 RK1808 计算棒配置

当需要在 x86-64 Linux 平台上使用 RK1808 计算棒运行 RockX SDK 的程序时，首次需要安装通信代理服务。

1） 安装通信代理服务

首次运行需要安装通信代理服务，命令如下所示：

```
./sdk/rockx-x86-64-Linux/bin/npu_transfer_proxy/install.sh
```

2） 卸载通信代理服务

如果需要将通信代理服务卸载，可以执行以下命令：

```
./sdk/rockx-x86-64-Linux/bin/npu_transfer_proxy/uninstall.sh
```

5.4 初始化和释放

Rock-X 各个模块都通过 rockx_create 函数进行初始化，通过传入不同的 rockx_module_t

枚举值来初始化不同的模块，示例代码如下所示：

```
rockx_ret_t ret;
rockx_handle_t face_det_handle;
ret = rockx_create(&face_det_handle,
                   ROCKX_MODULE_FACE_DETECTION,
                   nullptr, 0);
if (ret != ROCKX_RET_SUCCESS) {
    printf("init rockx module error %d\n", ret);
}
```

创建完成之后将得到一个 rockx_handle_t 类型的句柄，后面可以使用该句柄来调用相应的接口函数。

如果不再需要使用该模块，可以通过调用 rockx_destroy 函数来释放掉该句柄，示例代码如下所示：

```
rockx_destroy(face_det_handle);
```

5.5 接口调用

Rock-X SDK 所包含模块的接口函数如表 5-2 所示。

表 5-2 RockX 各模块接口函数

类别	函数名	函数功能
目标检测	rockx_object_detect	人车物检测
	rockx_head_detect	人头检测
人脸	rockx_face_detect	人脸检测
	rockx_face_landmark	人脸特征点定位
	rockx_face_pose	人脸姿态
	rockx_face_align	人脸矫正对齐
	rockx_face_recognize	人脸识别特征提取
	rockx_face_feature_similarity	人脸特征对比
	rockx_face_attribute	人脸属性分析
车牌	rockx_carplate_detect	车牌检测
	rockx_carplate_recognize	车牌识别
	rockx_carplate_align	车牌矫正对齐
人体关键点	rockx_pose_body	身体骨骼关键点定位
	rockx_pose_finger	手指关键点定位
其他	rockx_object_track	检测目标跟踪

模块接口函数示例代码如下：

```
rockx_object_array_t face_array;
memset(&face_array, 0, sizeof(rockx_object_array_t));

rockx_ret_t ret = rockx_face_detect(face_det_handle, &input_image,
                                    &face_array, nullptr);
if (ret != ROCKX_RET_SUCCESS) {
    printf("rockx_face_detect error %d\n", ret);
    return -1;
}
```

5.6 API 参考指南

详细的接口描述请参考 API 文档(doc\rockx_api_doc\html\index.html)。