

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：	V1.2
	作 者：	HPC&AI Team
	完成日期：	2019-02-13
	审 核：	熊伟 卓鸿添
	完成日期：	2019-02-13

福州瑞芯微电子股份有限公司

Fuzhou Rockchips Semiconductor Co., Ltd

(版本所有, 翻版必究)

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
V1.1	杨华聪	2019-11-15	1.添加 Python 接口使用说明 2.初始化和释放添加 config 配置说明 3.导入 SDK 库添加 rockx-data 的说明	熊伟 卓鸿添
V1.2	杨华聪	2019-02-13	1.更新系统依赖版本	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	RK180X 系统依赖.....	5
3	SDK 使用说明	6
3.1	C 接口使用说明.....	6
3.1.1	示例应用.....	6
3.1.2	导入 SDK 库.....	6
3.1.3	裁减说明.....	7
3.1.4	初始化和释放.....	8
3.1.5	接口调用.....	9
3.1.6	API 参考指南.....	10
3.2	PYTHON 接口使用说明.....	10
3.2.1	安装 WHEEL 包.....	10
3.2.2	示例应用.....	10
3.2.3	API 参考指南.....	12
4	性能指标	12
4.1	模块精度指标.....	12
4.1.1	目标检测.....	12
4.1.2	人脸检测.....	13
4.1.3	人脸识别.....	13
4.1.4	车牌识别.....	14
4.1.5	人脸属性分析.....	14

4.1.6	人脸特征点定位.....	14
4.1.7	人体骨骼点关键点.....	15
4.2	模块运行性能.....	15

1 主要功能说明

Rock-X SDK 是基于 RK3399Pro/RK180X 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。

Rock-X SDK 当前支持 Python/C 编程语言，支持运行于 RK3399Pro Android/Linux 平台、RK180X Linux 平台以及 PC Linux/MacOS/Windows（需要接 RK1808 计算棒）。

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 1.2.0 以上。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 1.2.0 以上。

```

=====
RKNN VERSION:
  API: 1.2.0 (1190a71 build: 2019-09-25 12:39:14)
  DRV: 1.2.0 (57b1656 build: 2019-09-04 09:27:47)
=====

```

2.2 RK180X 系统依赖

在 RK180X Linux 平台上，本 SDK 提供的库和应用程序需要 rknn_runtime 版本在 1.3.0 以上，在 RK180X 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 1.3.1 (7c5d3d8 build: 2020-01-10 14:11:03 base:  
1112)
```

3 SDK 使用说明

3.1 C 接口使用说明

3.1.1 示例应用

C 接口的示例应用代码包括命令行执行程序示例和 Android 程序示例。

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、RK180X Linux 平台以及 PC（需要接 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。

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。

3.1.2 导入 SDK 库

各平台的 Rock-X SDK 库位于 sdk 目录下，如下所示：

```
sdk/  
├── rockx-data  
├── rockx-rk1808-Linux  
├── rockx-rk1806-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

Linux 平台为例:

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

# Include RockX Header
include_directories(${RockX_INCLUDE_DIRS})

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

# install
install(PROGRAMS ${RockX_LIBS} DESTINATION lib)
install(PROGRAMS ${RockX_BINS} DESTINATION lib)
install(PROGRAMS ${RockX_DATA} DESTINATION lib)
```

注意，各个模块所需的模型/数据文件位于 `sdk/rockx-data` 目录下，`rockx` 通过以下顺序来查找对应的文件

- 1) 将 `rockx-data` 部署到任意路径，然后通过设置 `ROCKX_DATA_PATH` 环境变量（Linux 平台）/ `persist.sys.rockx.data.path` 属性值（Android 平台）来指定路径；
- 2) 将 `rockx-data` 部署到任意路径，然后在 `rockx_create` 时通过 `config` 传入路径；
- 3) 将 `rockx-data/`目录下所需的 `data` 文件放置到和 `librockx.so` 同一目录下，如果上述两种搜索不到，会在 `librockx.so` 所在的目录下搜索。如上 `CMakeLists.txt` 中的方式。

3.1.3 裁减说明

SDK 提供的 `sdk/rockx-data` 目录下的 `data` 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文件，这样可以减少应用的体积。表 5-1 列出了 SDK 各模块所对应的 `data` 文件，对于没有使用的模块，可以直接移除相应的文件。

表 5-1 rockx-data 文件说明

文件名	对应模块
carplate_align.data	ROCKX_MODULE_CARPLATE_ALIGN
carplate_detection.data	ROCKX_MODULE_CARPLATE_DETECTION
carplate_recognition.data	ROCKX_MODULE_CARPLATE_RECOG
face_attribute.data	ROCKX_MODULE_FACE_ANALYZE
face_detection.data	ROCKX_MODULE_FACE_DETECTION
face_landmark5.data	ROCKX_MODULE_FACE_LANDMARK_5
face_landmarks68.data	ROCKX_MODULE_FACE_LANDMARK_68
face_recognition.data	ROCKX_MODULE_FACE_RECOGNIZE
head_detection.data	ROCKX_MODULE_HEAD_DETECTION
object_detection.data	ROCKX_MODULE_OBJECT_DETECTION
pose_body.data	ROCKX_MODULE_POSE_BODY
pose_finger.data	ROCKX_MODULE_POSE_FINGER_3
pose_hand.data	ROCKX_MODULE_POSE_FINGER_21

3.1.4 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，其中可以通过 `rockx_config_t` 来配置 `data` 文件路径或指定运行在哪个目标设备，示例代码如下所示：

```
rockx_ret_t ret;
rockx_handle_t face_det_handle;

rockx_config_t rockx_configs;
memset(&rockx_configs, 0, sizeof(rockx_config_t));
rockx_add_config(&rockx_configs,
                (char *)ROCKX_CONFIG_DATA_PATH,
                "/sdcard/rockx-data/");

ret = rockx_create(&face_det_handle,
                 ROCKX_MODULE_FACE_DETECTION,
                 &rockx_configs,
                 sizeof(rockx_config_t));
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);
```

3.1.5 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	<code>rockx_object_detect</code>	人车物检测
	<code>rockx_head_detect</code>	人头检测
人脸	<code>rockx_face_detect</code>	人脸检测
	<code>rockx_face_landmark</code>	人脸特征点定位
	<code>rockx_face_pose</code>	人脸姿态
	<code>rockx_face_align</code>	人脸矫正对齐
	<code>rockx_face_recognize</code>	人脸识别特征提取
	<code>rockx_face_feature_similarity</code>	人脸特征对比
车牌	<code>rockx_carplate_detect</code>	车牌检测
	<code>rockx_carplate_recognize</code>	车牌识别
	<code>rockx_carplate_align</code>	车牌矫正对齐
人体关键点	<code>rockx_pose_body</code>	身体骨骼关键点定位
	<code>rockx_pose_finger</code>	手指关键点定位
其他	<code>rockx_object_track</code>	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;
}
```

3.1.6 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)。

3.2 PYTHON 接口使用说明

3.2.1 安装 WHEEL 包

Python 安装包位于 sdk/python 目录下, 开发者首先需要安装 Python3 和 python3-pip 环境, 然后直接通过如下命令进行安装

```
pip3 install RockX-*-py3-none-any.whl
```

另外 Demo 的运行还需要依赖 opencv-python 包。

3.2.2 示例应用

Python 安装包中自带了 SDK 接口的 Demo, 安装完成后即可直接运行。各 Demo 的代码位于 sdk/python/test 目录下, 开发者可以用于参考开发。

1) 摄像头示例应用

注意运行摄像头示例应用前需要插上 USB 摄像头 (也可以使用笔记本自带摄像头)。

a) 目标检测

```
python3 -m rockx.test.camera.rockx_object_detection
```

b) 人头检测

```
python3 -m rockx.test.camera.rockx_head_detection
```

c) 人脸分析

```
python3 -m rockx.test.camera.rockx_face_analyze
```

d) 人脸识别

首先需要导入人脸图片，将需要导入的图片（每张图片最好只有一个人脸，图片名字为英文）放入一个文件夹内，然后执行以下命令，导入完成后会生成 face.db 文件

```
python3 -m rockx.test.camera.rockx_face_recog -b face.db -i  
<import_image_dir>
```

导入完成后可进行识别测试

```
python3 -m rockx.test.camera.rockx_face_recog -b face.db
```

e) 手指

```
python3 -m rockx.test.camera.rockx_finger
```

f) 人体骨骼关键点

```
python3 -m rockx.test.camera.rockx_pose
```

以上所有命令都可以通过-c 参数指定要使用的摄像头，如使用摄像头 1

```
python3 -m rockx.test.camera.rockx_face_analyze -c 1
```

如果有插多个计算棒，可以通过-d 参数指定要运行的目标设备 ID，设备 ID 的获取请参考计算棒文档，参考命令如下：

```
python3 -m rockx.test.camera.rockx_face_analyze -d 7e9f3eb02ede60e8
```

2) 图片示例应用

运行如下命令可以运行一个对输入图片进行人车物分类的示例应用：

```
Python3 -m rockx.test.image.rockx_object_detection -i xxx.jpg
```

3.2.3 API 参考指南

可以通过 python 自带的 help 函数来查看 RockX Python SDK 的 API 函数，如下所示：

```
$ python3
>>> from rockx import RockX
>>> help(RockX)
```

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 $\pm 45^\circ$ 侧脸左右偏转 $\pm 60^\circ$ 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 $\pm 45^\circ$ 侧脸左右偏转 $\pm 60^\circ$ 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% \pm 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。
- 2) 人脸比对使用欧式距离。

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

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

$\text{euclidean}(d(j) - g(j))$ 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。