Classification Level: Top secret ( )   Secret ( )   Internal ( )   Public (√)

Rock-X SDK Developer Guide

(Technology Department, Graphic Display Platform Center)

Mark: [ ] Editing [√] Released	Version:	V1.2
	Author	HPC&AI Team
	Completed Date	2020-02-26
	Reviewer	Xiong Wei/Zhuo HongTian
	Reviewed Date	2020-02-26

福州瑞芯微电子股份有限公司

Fuzhou Rockchips Semiconductor Co . , Ltd

Revision History

Version	Modifier	Date	Modify Description	Reviewer
V1.0	Yang Huacong	2019-06-11	Initial version	Zhuo HongTian
V1.1	Yang Huacong	2019-11-15	1. Add Python API 2. Add config configuration instructions 3. Add rockx-data description	Zhuo HongTian
V1.2	Yang Huacong	2020-02-13	1. Update Depenc	Zhuo HongTian

Table of Contents

1 Overview

Rock-X SDK is a set of AI components based on the RK3399Pro/RK1808 platform. Developers can quickly build AI applications through the API provided by SDK.

The Rock-X SDK currently supports the Python / C programming language and supports the RK3399Pro Android / Linux platform, RK180X Linux platform, and PC Linux / MacOS / Windows (requires RK1808 computing stick).

The functions provided by the SDK are as table 1-1 show:

Table 1-1 Rock-X SDK main functions

Classes	Functions
Object Detection	Head Detection / Object Detection
Face	Face Landmark / Face Analyze / Face Recognition
Carplate	CarPlate Detectin / Carplate Recognition
Human KeyPoint	Body Keypoint / Finger Keypoint

2 Dependencies

2.1 RK3399Pro System Dependencies

On the RK3399Pro platform, the libraries and applications provided by the SDK require the RKNN driver version to be 1.2.0 or higher. After running the Demo application on the RK3399Pro Android / Linux platform, the following driver information can be seen through the logs. Please ensure that the DRV version is 1.2.0 or higher.

```
===================================================
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 System Dependencies

On the RK180X Linux platform, the libraries and applications provided by this SDK require rknn_runtime version 1.3.0 or above, The method to view the rknn_runtime version on the RK180X platform is as follows:

```
    $ 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 Instructions

3.1 C API

3.1.1 Examples

The example of the C API includes command line execution program examples and android application examples.

1）Command Line Example

Rock-X SDK provides command line execution program code examples for all interfaces. The sample program supports running on RK3399Pro Android / Linux platform, RK180X Linux platform, and PC (requires RK1808 computing stick) platform. please refer to the README file under the "demo/command_line_demo" directory.

2）Android Application Example

The Android application example supports running on the RK3399Pro Android platform. All Android example is located in the "demo/rk3399pro_android_demo" directory. After decompressing the example and opening it through Android Studio, you can directly compile, run, and develop.

3.1.2 Import Libraries

The Rock-X SDK libraries for each platform are located in the "sdk" directory, as shown below:

```
sdk/
├── rockx-data
├── rockx-rk1808-Linux
├── rockx-rk1806-Linux
├── rockx-rk3399pro-Android
├── rockx-rk3399pro-Linux
└── rockx-x86-64-Linux
```

Developers only need to introduce the library of the corresponding platform in your CMakeLists.txt. Take the RK3399Pro Linux platform as an example:

```
# 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)
```

Note that the data files required for each module are located in the "sdk/rockx-data" directory, rockx finds the corresponding files for each module in the following order:

1) Specify the path by setting the ROCKX_DATA_PATH environment variable (on Linux platform) or persist.sys.rockx.data.path property value (on Android platform);

2) Pass the path through the config parameter when calling “rockx_create”;

3) Place the data file in the same directory as librockx.so. If the above two methods cannot be found, it will search in the directory where librockx.so is located. As in CMakeLists.txt above.

3.1.3 Data Files

The data files in the "sdk/rockx-data" directory provided by the SDK do not need to be fully imported. Developers can import required files according to the modules being use, which can reduce the size of the application. Table 3-1 lists the data files corresponding to the SDK modules. For unused modules, you can directly remove the corresponding data files.

Table 3-1 rockx-data file description

Data Files	Module
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 Create and Destroy Module

Rock-X modules are initialized by the rockx_create function, and different modules are initialized by passing in different rockx_module_t enumeration values. You can configure the data file path or specify which target device to run on through rockx_config_t. The sample code is as follows:

```
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);
}
```

After the creation is complete, you will get a handle of type rockx_handle_t, which can be used later to call the corresponding interface function.

If you don't need to use this module, you can release the handle by calling the rockx_destroy function. The sample code is as follows:

```
rockx_destroy(face_det_handle);
```

3.1.5 Modules Interface

Table 3-2 shows the interface functions of the modules included in the Rock-X SDK.

Table 3-2 RockX module interface functions

Classes	Functions	Description
Object Detection	rockx_object_detect	91 Classes Object Detection
	rockx_head_detect	Head Detection
Face	rockx_face_detect	Face Detection
	rockx_face_landmark	Face KeyPoint Landmark
	rockx_face_pose	Face Angle
	rockx_face_align	Face Align
	rockx_face_recognize	Face Recognition
	rockx_face_feature_similarity	Compare Two Face Feature
	rockx_face_attribute	Face Attribute Analyze
Carplate	rockx_carplate_detect	CarPlate Detection
	rockx_carplate_recognize	CarPlate Recognition
	rockx_carplate_align	Carplate Align
Human Keypoints	rockx_pose_body	Human Body Keypoint
	rockx_pose_finger	Human Finger Keypoint
Others	rockx_object_track	Track Detection Object

The module interface function call example code is as follows:

```
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 Reference

Please refer to API documentation (doc\rockx_api_doc\html\index.html)。

3.2 PYTHON API

3.2.1 Installation

The Python installation package is located in the “sdk/python” directory. Developers first need to install the Python3 and python3-pip environments. Then install RockX wheel with the following command:

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

In addition, in order to run the python demo, you need to install the opencv-python package.

3.2.2 Examples

Demo is included in the Python installation package, and can be run directly after installation is complete. The code of each demo is located in the "sdk/python/test" directory, and developers can use it for reference development.

1） Camera Example

Note that you need to plug in the USB camera before running the camera example application (you can also use the laptop ’s own camera).

a) Object Detection Example

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

b) Head Detection Example

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

c) Face Attribute Analysis Example

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

d) Face Recognition Example

First you need to import face pictures. Put the pictures you want to import (each picture should have only one face), and then execute the following command. After the import is complete, a face.db file will be generated.

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

Face recognition test can be performed after the face import is completed

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

e) Finger Example

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

f) Human Body Example

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

All the above commands can specify the camera to be used with the -c parameter, such as using camera 1:

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

If there are multiple computing sticks inserted, you can specify the target device ID to run through the -d parameter. For the device ID, please refer to the computing stick documentation. The reference command is as follows:

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

2）Image Example

Run the following command to run a sample application for object detection with an input image:

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

3.2.3 API Reference

You can view the API functions of the RockX Python SDK through the help function that comes with python, as shown below:

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

4 Performance

4.1 Accuracy

4.1.1 Object Detection

Table 4-1 Performance of detection algorithms

Module	Dataset	Performance
Head Detection	Brainwash	mAP@IOU0.5 = 0.704
Car plate	CCPD	mAP@IOU0.5 = 0.983
Object Detection	MSCOCO_VAL2017	mAP@IOU0.5 = 0.565

Note:

1) mAP@IOU0.5=0.704 means that the corresponding mAP = 0.704 when IOU = 0.5.

2) Brainwash is a public data set for head detection. The main scene is a coffee shop. Has a total of 5,007 photos.

3) CCPD (Chinese City Parking Dataset) is a Chinese carplate dataset, from which 10,000 samples are randomly selected for testing.

4) MSCOCO_VAL2017 is a public dataset for target detection. It uses 5,000 validation set tests in this dataset, with a total of 91 categories.

5) The minimum detection size of the human head detection module is 1/19 of the image resolution.

6) The car plate detection module supports the detection of blue, yellow and green car plates in China.

4.1.2 Face Detection

Table 4-2 Face detection performance

Parameter	Performance
Adaptation angle	Roll 45 ° Yaw 60 ° Head Up 60° Head Down 45°
Maximum distance	11 meters (test camera FOV = 60 °)
mAP	mAP@IOU0.5=0.857

Note:

1) When the image quality is poor, the supported detection angle will decrease.

2) The maximum detection distance is related to parameters such as camera FOV.

3) The minimum face size detected is 1/19 of the image resolution.

4.1.3 Face Recognition

Table 4-3 Face recognition performance

Parameter	Performance
Adaptation angle	Roll 45 ° Yaw 60 ° Head Up 60° Head Down 45°
Maximum distance	11 meters (test camera FOV = 60 °)
Accuracy (on the LFW dataset)	99.65%±0.00088

	TPR=0.992@FAR=0
	TPR=0.995@FAR=0.001

Note:

1) In practical applications, limiting the distance and angle can get better recognition results, and developers can also perform face image quality filtering according to actual conditions.

2) Face feature comparison using Euclidean distance.

4.1.4 Car plate Recognition

Table 4-4 Car plate recognition performance table

Dataset	Performance
CCPD	83.31%(8331/10000)

Note:

1) CCPD (Chinese City Parking Dataset) is a Chinese car plate data set, from which 10,000 samples are randomly selected for testing.

2) Support to recognize Chinese blue, green and yellow car plates.

3) Recognizable car plate characters are shown in Table 4-5.

Table 4-5 Car plate recognition characters

Type	Characters
Province	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
Numbers and letters	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
Others	港 学 使 警 澳 挂 军 北 南 广 沈 兰 成 济 海 民 航 空

4.1.5 Face Attribute Analyze

Table 4-6 Sex and age performance

Dataset	Age Accuracy	Gender Accuracy
UTK_asian	4.823283	92.96%(2220/2388)

Note:

1) UTK_asian is the Asian part of UTK's public dataset. It was tested using 7-70 years old data, a total of 2388 images.

2) Age accuracy is mean age deviation.

4.1.6 Face Landmark

Table 4-7 Face lanmark (68 points) performance

Dataset	Error
300w_cropped	6.01%

Note:

1) The error calculation formula is as follows

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

$\text{euclidean}(d(j) - g(j))$ Represents the Euclidean distance between the j-th detection point and the labeled point.

"d" represents the Euclidean distance between the left outer corner and the right outer corner.

4.1.7 Pose

Table 4-8 Human body key point performance of human bones

Dataset	Accuracy
MSCOCO_VAL2017	mAP@OKS0.5=0.623

Note:

1) mAP@OKS0.5=0.623 means the corresponding mAP = 0.623 when OKS = 0.5.

2) MSCOCO val2017 is the verification set of the COCO 2017 Keypoint Detection Task, a total of 5,000, of which more than 2,000 have key points.

4.2 System Consume

Table 4-9 shows the running time and required memory of each module.

Table 4-9 Module run time and memory consumption

Module	Run Time(ms)	Used NPU Memory(MB)
Object Detection	49	66
Head Detection	38	43
Face Detection	41	24
Face Landmark(68 point)	11	34
Face Angle	2	21
Face Align	9	20
Face Recognition	44	117
Face Analyze	16	19
Carplate Detection	46	39
Carplate Align	21	22
Carplate Recognition	39	21
Human Body Keypoints	106	119
Finger Keypoints(21 points)	153	89
Finger Keypoints(3 points)	23	21
Deection Object Track	1	18

Note

1) Peak memory measured in the table.