密级状态：绝密(　　)　　秘密(　　)　　内部(　)　　公开(√)

Rockchip RK1808/RK3399Pro 人脸 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：	2019-11-26
	审　　核：	熊伟 卓鸿添
	完成日期：	2019-11-26

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	2019-11-26	初始版本	熊伟 卓鸿添

目 录

1 主要功能说明

RK1808/RK3399Pro 人脸 SDK 提供一系列人脸识别分析相关功能，充分利用了 NPU 对算法模型进行加速。开发者通过 SDK 提供的 API 接口能够快速构建人脸 AI 应用。

SDK 当前支持 C/C++进行开发，支持运行于 RK1808/RK3399Pro Linux/Android 平台。

当前 SDK 提供的功能如表 1-1 所示。

表 1-1 人脸 SDK 主要功能

类别	功能
人脸检测	人脸检测、人脸跟踪、人脸矫正对齐
人脸分析	人脸关键点、人脸属性分析、人脸角度
人脸识别	人脸特征提取、人脸比对、人脸搜索
活体检测	活体检测（需要特定红外摄像头）

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。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 RK1808 系统依赖

在 RK1808 Linux 平台上，本 SDK 提供的库和应用程序需要 rknn_runtime 版本在 1.2.0 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 1.2.0 (d6402c5 build: 2019-09-04 09:14:03 base:
112)
```

3 授权说明

SDK 需要获得授权后才能使用，客户首先需要向对应业务提出申请，获得授权使用的用户名和密码，然后通过 sdk/auth 目录下的授权工具进行授权就可以正常使用。人脸识别授权流程如下所示：

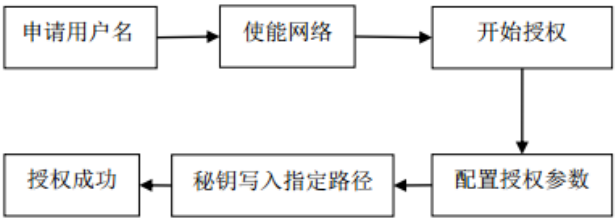

图 1 人脸识别授权流程

具体配置方法请参考 sdk/auth 目录中的说明文档。在获得密钥文件后，修改应用中设置授权 key 的路径，就可以正常进行人脸识别。

4 SDK 使用说明

4.1 C 接口使用说明

4.1.1 示例应用

人脸 SDK 提供了命令行执行程序代码示例，示例程序支持在 RK3399Pro/RK1808 Linux 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。

4.1.2 导入 SDK 库

SDK 库位于 sdk 目录下，如下所示：

```
sdk/
├── rockface-data
├── rockface-rk3399pro-Android
├── rockface-rk1808-Linux
└── rockface-rk3399pro-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 RK1808 Linux 平台为例：

```
# Find RockFace Package
set(RockFace_DIR <path-to-rockface-sdk>/sdk/rockface-rk1808-Linux)
find_package(RockFace REQUIRED)

# Include RockFace Header
include_directories(${RockFace_INCLUDE_DIRS})

# Link RockFace Libraries
target_link_libraries(target_name ${RockFace_LIBS})

# Install RockFace
install(PROGRAMS ${RockFace_LIBS} DESTINATION lib)
install(PROGRAMS ${RockFace_DATA} DESTINATION lib)
```

注意，各个模块所需的数据文件位于 sdk/rockface-data 目录下，可以通过以下两种方式来设置其路径：

1）将 rockface-data 部署到设备任意路径，然后通过 rockface_set_data_path 函数设置其路径；

2）将 rockface-data/目录下所需的 data 文件放置到和 librockface.so 同一目录下，如上 CMakeLists.txt 中的方式。

4.1.3 初始化和释放

SDK 通过 rockface_create_handle 函数来创建一个句柄对象，示意代码如下所示，示例代码如下所示：

```
rockface_ret_t ret;
rockface_handle_t face_handle = rockface_create_handle();
```

创建完之后可以通过调用 rockface_set_licence 函数设置 licence 文件（licence 文件的获取

请参见授权说明一节），示意代码如下所示：

```
ret = rockface_set_licence(face_handle, licence_path);
if (ret != ROCKFACE_RET_SUCCESS) {
    printf("Error: authorization error %d!", ret);
    return ret;
}
```

下来可以通过调用 rockface_set_data_path 来设置数据文件(sdk/rockface-data)在设备的路径，示意代码如下所示：

```
rockface_set_data_path(face_handle, "/usr/share/rockface-data")
```

以上成功设置后，可以根据需要使用的模块来调用不同的初始化函数进行初始化，示意代码如下所示：

```
ret = rockface_init_detector(face_handle);
if (ret != ROCKFACE_RET_SUCCESS) {
    printf("Error: init detector error %d!", ret);
    return ret;
}
ret = rockface_init_recognizer(face_handle);
if (ret != ROCKFACE_RET_SUCCESS) {
    printf("Error: init recognizer error %d!", ret);
    return ret;
}
```

最后如果不需要继续使用，可以调用 rockface_release_handle 函数进行释放，示意代码如下：

```
rockface_release_handle(face_handle);
```

4.1.4 接口调用

SDK 所包含模块的接口函数如表 3-1 所示。

表 3-1 SDK 接口函数

函数	初始化函数	描述
rockface_detect	rockface_init_detector	人脸检测
rockface_track	rockface_init_detector	人脸跟踪
rockface_align	rockface_init_detector	人脸对齐
rockface_landmark	rockface_init_analyzer	人脸关键点
rockface_angle	rockface_init_analyzer	人脸角度
rockface_attribute	rockface_init_analyzer	人脸属性分析
rockface_feature_extract	rockface_init_recognizer	人脸特征提取
rockface_feature_compare	rockface_init_recognizer	人脸特征比对
rockface_liveness_detect	rockface_init_liveness_detector	活体检测

模块接口函数示例代码如下：

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

// detect face
ret = rockface_detect(face_handle, &input_image, &face_array);
if (ret != ROCKFACE_RET_SUCCESS) {
    printf("rockface_face_detect error %d\n", ret);
    return -1;
}
```

4.1.5 API 参考指南

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

5 性能指标

5.1 主要模块精度

5.1.1 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。

5.1.2 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。

2) 人脸比对使用欧式距离。

5.1.3 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。

5.1.4 人脸关键点定位

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。

5.2 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)
人脸检测	41	24
人脸矫正对齐	9	20
人脸跟踪	1	18
人脸关键点（68 点）	11	34
人脸角度	2	21
人脸属性分析	16	19
人脸识别特征提取	44	117

注：

1) 图中所测的内存值为峰值内存。