百度线下人脸采集 Android 开发手册

v1.0

1.简介

百度线下人脸采集解决方案**通过摄像头或平板进行视频采集，调用人脸检测技术和人脸 1：N 技术，验证摄像头或平板前的人是否在您注册的人脸库中，返回对应的人的信息，跟已有的系统打通。**基于该方案,开发者可以轻松的构建自己的线下人脸采集系统。

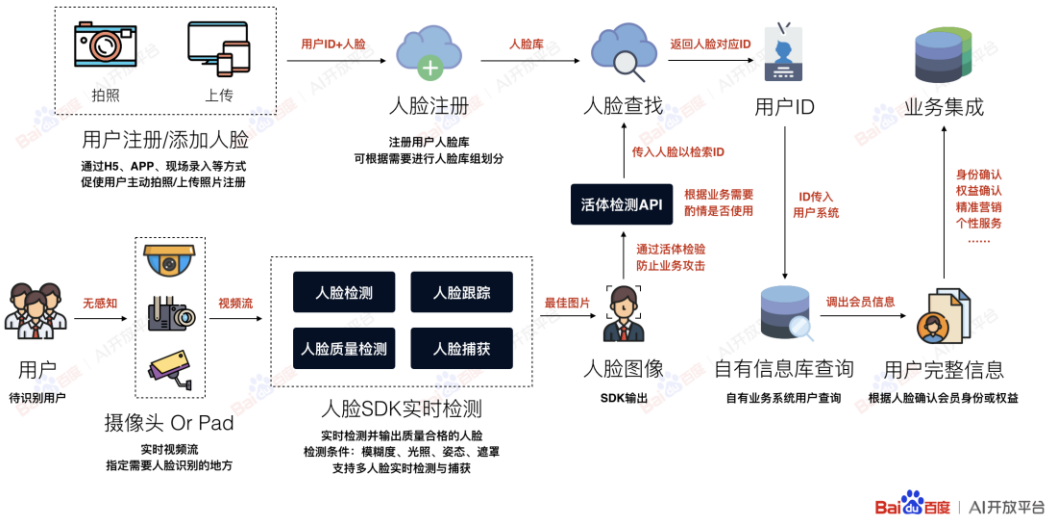
在您使用之前，我们首先为您介绍一下人脸技术及相关人脸能力，以便您能更方便使用人脸服务。

- **本地版活体检测**：通过让用户做出指定脸部交互动作，识别当前操作者是否为活体，此功能可离线使用，可设定指定动作是否使用及应用顺序。可有效抵御高清图片、3D 建模、视频等攻击。
- **在线版活体检测**：基于本地版有交互活体检测输出的质量合格的图片，请求在线活体检测接口，可以进一步判断用户上传的图片是否为活体，进一步抵御高精度视频、高精度建模等攻击。此接口建议与本地版活体检测结合使用，且输入的照片一定为二次采集图片（即限于产品前端拍照所得，不可为相册图片）。
- **人脸质量检测**：判断视频流中的图片帧中，哪些图片质量较佳，即人脸图像特征清晰（满足角度、姿态、光照、模糊度等校验）。
- **人脸图像采集**：通过本地 SDK 能力，采集人脸图像，同时经过人脸质量检测，确保采集到的人脸图像符合各条件校验（角度、姿态、光照、模糊度等），为设备前端获取有效可分析人脸的主要功能。
- **在线鉴权**：API 接口的权限判断，在后台获取 API Key 和 Secret Key 后生成的 token，用于每次请求时供服务器判断是否可通过此次请求。
- **离线授权**：SDK 的授权判断，授权介质也称为 license，在 SDK 使用中，需要通过 license 向授权服务器发起请求，判断 SDK 的使用合法性及使用有效期。
- **人脸识别**：人脸识别技术的泛称，通常指识别图片或视频中的人脸，提取人脸特征值，用于进一步的人脸对比、搜索等业务操作。
- **人脸检测接口**：分析并定位图片中的人脸及位置，是人脸识别技术的第一步分析内容。
- **人脸 1：1 接口**：目标分析人脸与已知身份人脸进行比对，对比两个图片中的人脸的相似度，根据分值判断两张人脸是否为同一人。
- **人脸 1：N 接口**：在一个已知身份的人脸集合中找到相似的人脸，返回内容为人脸集合中每一张脸与目标人脸的相似度分值，找出目标人脸是否属于哪一个已知身份的用户。
- **人脸属性接口**：通过人脸特征抽取，分析人脸的性别、年龄、种族、是否戴眼镜等属性值。

2 功能介绍

业务流程图

业务流程示意



线下人脸采集支持下列功能：

离线SDK功能完成人脸采集

- 人脸检测：检测当前镜头前是否存在人脸，并标记出人脸轮廓位置和关键点。
- IPCamera视频流采集：提供RTSPSdk采集IPCamera的视频流。
- 平板视频流采集：采集平板摄像头的视频流

人脸在线API功能完成人脸注册和人脸识别

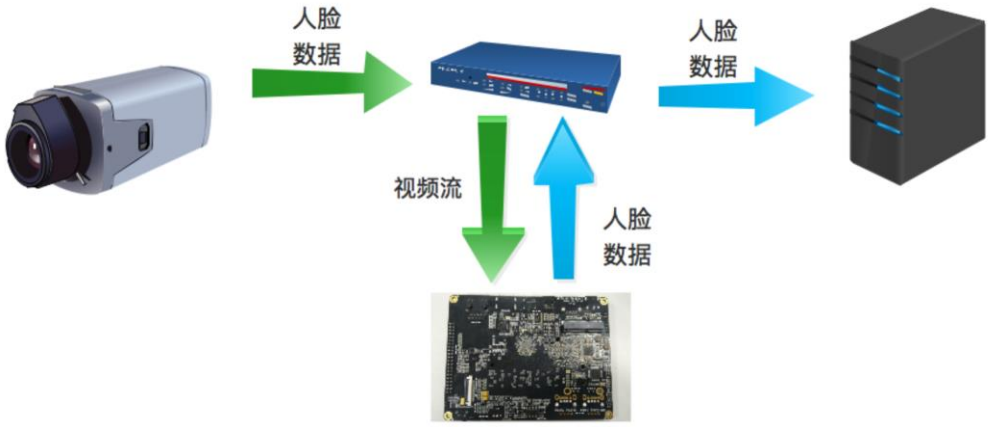
- 人脸注册：提供人脸注册功能，帮忙您建立人脸库。
- 人脸1：N：通过和人脸注册的人脸库进行对比，帮助你获取会员的ID。

3 硬件方案

3.1 网络摄像头方案

该方案适用于人到摄像头距离为 2-15 米的场景，主要设备如下表

项目	型号	备注
摄像头机身	海康 DS-2CD5026EFWD	
摄像头镜头	HV1140D-8MPIR	
人脸检测硬件	RK3399	



3.2 平板方案

该方案适用于柜台前的场景，可自行采购平板

3.3 USB 摄像头方案

该方案适用于柜台前的场景，可自行采购 usb 摄像头+rk3399

3.4 一体摄像头方案

该方案适用于人到摄像头距离为 2-15 米的场景，研发中。

4 开发包说明

文件/文件夹名	说明
/facelibrary	SDK 库及各平台的 so 库。so 包含以下几个平台如果关注包大小，请自行删减。armeabi/armeabi-v7a/arm64-v8a/x86，
线下人脸采集开发手册	本手册
/FaceMemeber/	DEMO 工程

5 集成指南

本章将进行 Step-By-Step 的讲解,如何快速的集成。一个完整的 Demo 请参考开发包中的示例程序 FaceMember。

5.1 准备工作

在正式集成前，需要做一些准备工作，完成一些账号、应用及配置，具体如下：

5.1.1 注册开发者

STEP1：点击百度 AI 开放平台导航右侧的控制台，页面跳转到百度云登录界面，登录完毕后，将会进入到百度云后台，点击「控制台」进入百度云控制台页面；您也可以在官网直接点击免费试用，登录完毕后将自动进入到百度云控制台。
STEP2：使用百度账号完成登录，如您还未持有百度账户，可以点击此处注册百度账户。
STEP3：进入百度云欢迎页面，填写企业/个人基本信息，注册完毕，至此成为开发者。注：(如您之前已经是百度云用户或百度开发者中心用户，STEP3 可略过。)
STEP4：进入百度云控制台，找到人工智能相关服务面板。
STEP5：**点击进入「人脸识别」模块。**

注一：如果您通过百度 AI 开放平台登录到后台，AI 相关服务模块入口，如下图所示：

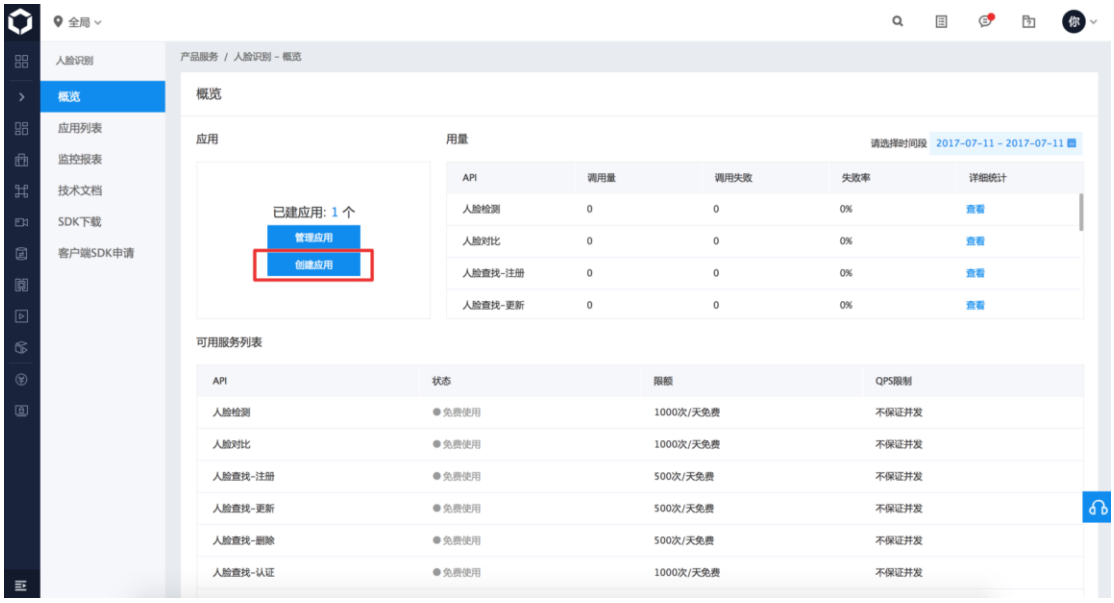
注二：如果您通过百度云直接登录后台，AI 相关服务模块入口，则如下图红框部分所示：

两种入口只是展现形式不同，相关 AI 服务模块内容完全一样。

5.1.2 创建应用（用于调用在线api）

创建好账号后，在正式调用 AI 能力之前，需要您创建一下应用，这个是调用服务的基础能力单元

选择「人脸识别」服务，首先见到此服务的控制台概览页，如下图所示：



如上图所示，点击「创建应用」，即可进入应用创建界面，如下图所示：

5.1.3 获取密钥

在您创建完毕应用后，平台将会分配给您此应用的相关凭证，主要为 **AppID、API Key、Secret Key**，以上三个信息是您应用实际开发的主要凭证，每个应用之间各不相同，请您妥善保管。如上图所示。该 AK/SK 用于调用在线 API 如：注册、识别。和 SDK 创建应用不是一个概率。

注：开发中请注意区分多份 AKSK（API Key、Secret Key）

5.1.4 生成token

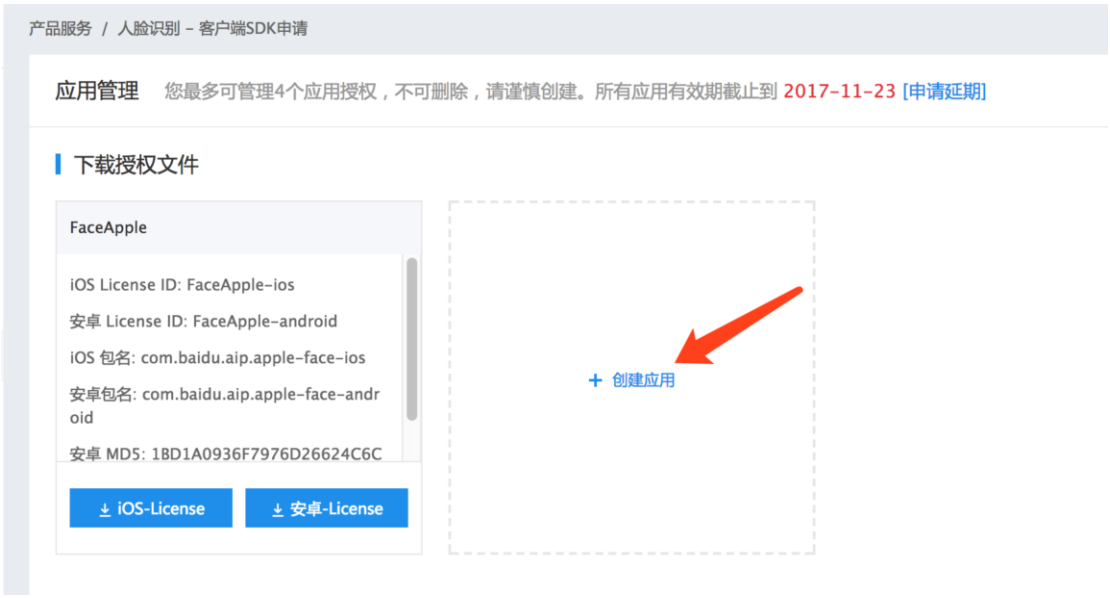
刚才所创建的应用在调用开放平台 API 之前，首先需要获取 Access Token（用户身份验证和授权的凭证）您需要使用创建应用所分配到的 AppID、API Key 及 Secret Key，进行 Access Token 的生成，方法详见 Access Token 获取，我们为您准备了几种常见语言的请求示例代码。

注：Access Token 的有效期为 30 天（以秒为单位），请您集成时注意在程序中定期请求新的 token，或每次请求都获取一次 token。

5.1.5 SDK license（重要！重要！重要！）

人脸 SDK License： 此 license 用于 SDK 离线功能使用，在您的申请人脸 SDK 的后台页面，全局->产品服务->人脸识别->客户端 SDK 申请

点击客户端 SDK 管理，弹出如下图：创建应用（这里创建应用是为了使用离线 SDK，上面创建应用为了使用人脸在线接口，如注册、识别等）



在弹出的框中输入授权标识，选择应用类型，应用系统，以及包名、MD5 签名，详情请查看输入框右边提示

5.1.6 下载登录示例工程（重要！重要！重要！）

本文档为人脸登录文档，您不需要下载 SDK，直接下载人脸登录的示例工程即可，示例工程有登录的完整流程。

点击下载示例工程，弹出下图，您可以选择管理应用创建的应用，点击下载示例工程，这些下载下来的示例工程将是已经完成离线授权配置的示例工程（在线接口使用的 sk，sk 还是需要您自己申请，参见前面的创建应用生成 aksk），将会减少您在集成文档中不必要的问题（推荐）

5.2 集成逻辑

5.2.1 流程图

参见功能介绍

5.2.2 阈值选择

参数	名称	默认值	取值范围
brightnessValue	图片爆光度	40f	
blurnessValue	图像模糊度	0.7f	0~1.0f
occlusionValue	人脸遮挡阀值	0.5f	0~1.0f
headPitchValue	低头抬头角度	15	0~45
headYawValue	左右角度	15	0~45
headRollValue	偏头角度	15	0~45
cropFaceValue	裁剪图片大小	600	
minFaceSize	最小人脸检测值 小于此值的人脸将 检测不出来。最小值 为80.	200	
notFaceValue	人脸置信度	0.8f	0~1.0f

isCheckFaceQuality	是否检测人脸质量	true	true/flase

5.2.3 代码集成

5.2.3.1 自动配置授权信息集成

如果您是通过自动配置授权信息下载的示例工程，只需要配置 Config 类和配置签名，您的集成将少很多步骤。

1、修改 Config 类中参数,自动配置已经为您修改好了 licenseID 和 license 文件，您只需要修改 apiKey、secretKey（即 ak/sk），groupId 是自己定义的，用于人脸注册和人脸识别等接口使用。保证注册的人脸和查找的人脸在同一个 groupId 即可。

```
public class Config {

    // 为了apiKey,secretKey为您调用百度人脸在线接口的，如注册，识别等。
    // 为了的安全，建议放在您的服务端，端把人脸传给服务器，在服务端端进行人脸注册、识别放在示例里面是为了您快速看到效果
    public static String apiKey = 替换为你的apiKey(ak);
    public static String secretKey = 替换为你的secretKey(sk);
    public static String licenseID = 替换为你的licenseID,后台SDK管理界面中，已经生成的licenseID,如:test-face-android;
    public static String licenseFileName = "idl-license.face-android";


    /**
        groupId，标识一组用户（由数字、字母、下划线组成），长度限制128B，可以自行定义，只有注册和识别都是同一个组。

        每个开发者账号只能创建一个人脸库；groupID用于标识人脸库
        每个人脸库下，用户组（group）数量没有限制；
        如闸机场景，可以给一个公司生成一个groupID;用于区分不同公司间的人脸库。
        详情见 http://ai.baidu.com/docs#/Face-API/top

        人脸识别 接口 https://aip.baidubce.com/rest/2.0/face/v2/identify
        人脸注册 接口 https://aip.baidubce.com/rest/2.0/face/v2/faceset/user/add
     */

    public static String groupID = 替换为groupID;

}
```

2、 配置签名（申请 license 时的 md5 为打包签名的文件，所以必须用申请 license 的签名文件）

因为 SDK 运行时会对比 license 里面的 md5 和签名文件的 md5 ,为了能 debug 也能使用人脸，所以需要进行下面的配置。实际发布时只要使用申请时关联的签名文件即可，没有下面的配置也行。

app->build.gradle->android->signingConfigs

keyAlias 为你创建的打包签名文件的别名

```
signingConfigs {
    debug {
        keyAlias [redacted]
        keyPassword [redacted]
        storeFile file('../[redacted].jks')
        storePassword([redacted])
    }
    release {
        keyAlias [redacted]
        keyPassword [redacted]
        storeFile file('../[redacted].jks')
        storePassword([redacted])
    }
}
```

5.2.3.2 未使用自动配置授权信息的集成

1、把申请的 license（idl-license.face-android"）放到到项目中 assets 目录中

2、修改 Config 类中的参数

licenseID 信息请直接查看后台授权信息列表即可。

GroupId 是自己定义的，用于人脸注册和人脸识别等接口使用。保证注册的人脸和查找的人脸在同一个 groupId 即可。

```
public class Config {

    // 为了apiKey,secretKey为您调用百度人脸在线接口的，如注册，识别等。
    // 为了的安全，建议放在您的服务端，端把人脸传给服务器，在服务端端进行人脸注册、识别放在示例里面是为了您快速看到效果
    public static String apiKey = 替换为你的apiKey(ak);
    public static String secretKey = 替换为你的secretKey(sk);
    public static String licenseID = 替换为你的licenseID,后台SDK管理界面中，已经生成的licenseID,如:test-face-android;
    public static String licenseFileName = "idl-license.face-android";


    /**
        groupId，标识一组用户（由数字、字母、下划线组成），长度限制128B，可以自行定义，只有注册和识别都是同一个组。

        每个开发者账号只能创建一个人脸库；groupID用于标识人脸库
        每个人脸库下，用户组（group）数量没有限制；
        如闸机场景，可以给一个公司生成一个groupID;用于区分不同公司间的人脸库。
        详情见 http://ai.baidu.com/docs#/Face-API/top

        人脸识别 接口 https://aip.baidubce.com/rest/2.0/face/v2/identify
        人脸注册 接口 https://aip.baidubce.com/rest/2.0/face/v2/faceset/user/add
     */

    public static String groupID = 替换为groupID;

}
```

3、配置签名（申请 license 时的 md5 为打包签名的文件，所以必须用申请 license 的签名文件）

因为 SDK 运行时会对比 license 里面的 md5 和签名文件的 md5，为了能 debug 也能使用人脸，所以需要进行下面的配置。实际发布时只要使用申请时关联的签名文件即可，没有下面的配置也行。

app->build.gradle->android->signingConfigs

keyAlias 为你创建的打包签名文件的别名

keyPassword 为你创建的打包签名的别名密码

storeFile 为签名文件的路径

storePassword 为签名文件的密码

```
    signingConfigs {
        debug {
            keyAlias [redacted]
            keyPassword [redacted]
            storeFile file('../[redacted].jks')
            storePassword('[redacted]')
        }
        release {
            keyAlias [redacted]
            keyPassword [redacted]
            storeFile file('../[redacted].jks')
            storePassword('[redacted]')
        }
    }
```

4、修改包名 app->build.gradle->android->defaultConfig ->您申请 license 时填的包名

```
android {
    compileSdkVersion 25
    buildToolsVersion "25.0.3"
    defaultConfig {
        applicationId "[redacted]"
        minSdkVersion 19
        targetSdkVersion 25
        versionCode 1
        versionName "1.0"

        ndk{
            abiFilters 'armeabi-v7a'
        }

    }
```

```
manifest | application | activity
<?xml version="1.0" encoding="utf-8"?>
<!--
  ~ Copyright (C) 2017 Baidu, Inc. All Rights Reserved.
  -->
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
          package="[redacted]">

    <uses-permission android:name="android.permission.CAMERA"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>

    <uses-feature android:name="android.hardware.camera"/>
    <uses-feature android:name="android.hardware.camera.autofocus"/>
```

6 功能使用

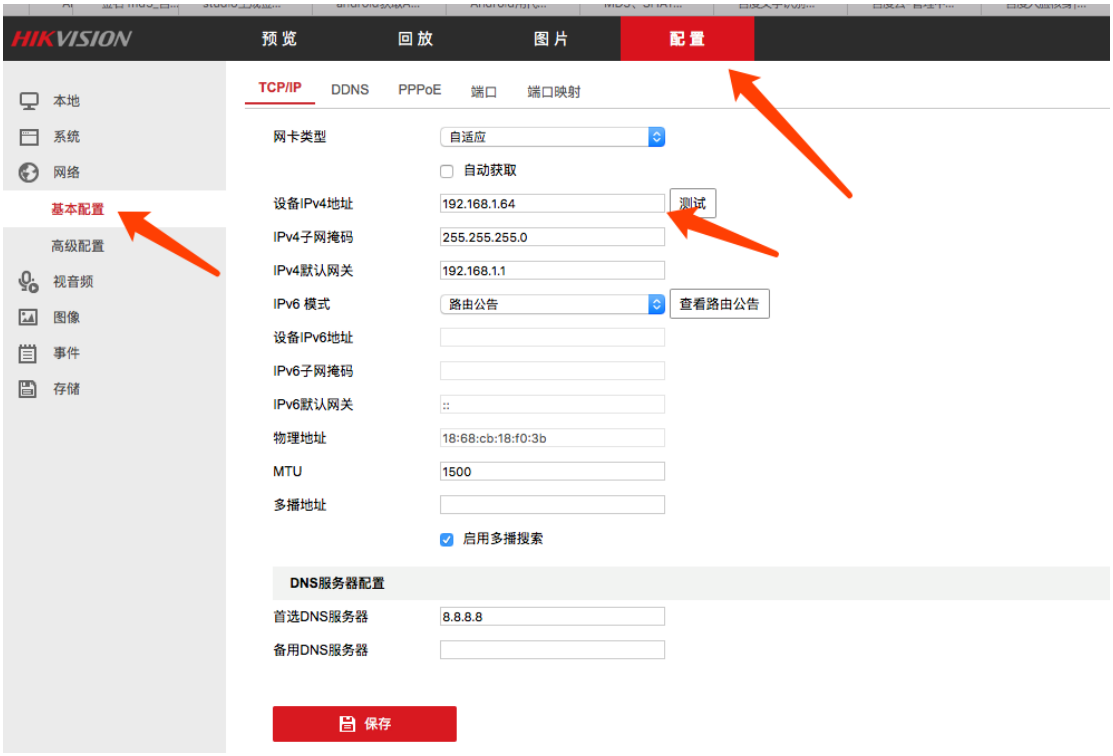
6.1 IPCamera方案

6.1.1 硬件连接和设置

1、 海康摄像头ip默认为192.168.1.64。若需要修改，有两种方案

A. 电脑修改ip为该192.168.1.x网段，把摄像头与电脑相连。第一次进入是需要设置摄像头用户名密码（后面人脸检测需要用到改用户名和密码）。

B. 路由器ip为该192.168.1.x网段，把摄像头与电脑相连。第一次进入是需要设置摄像头用户名密码（后面人脸检测需要用到改用户名和密码）。


HIKVISION
预览
回放
图片
配置
本地
系统
网络
基本配置
高级配置
视音频
图像
事件
存储
TCP/IP
DDNS
PPPoE
端口
端口映射
网卡类型
自适应
自动获取
设备IPv4地址
192.168.1.64
测试
IPv4子网掩码
255.255.255.0
IPv4默认网关
192.168.1.1
IPv6 模式
路由公告
查看路由公告
设备IPv6地址
IPv6子网掩码
IPv6默认网关
::
物理地址
18:68:cb:18:f0:3b
MTU
1500
多播地址
启用多播搜索
DNS服务器配置
首选DNS服务器
8.8.8.8
备用DNS服务器
保存

2、 摄像头视频设置


HIKVISION
预览
回放
图片
配置
本地
系统
网络
视音频
图像
事件
存储
视频
音频
码流信息叠加
区域裁剪
码流类型
主码流（定时）
视频类型
视频流
分辨率
1280*720P
码率类型
变码率
图像质量
中
视频帧率
18
fps
码率上限
12288
Kbps
平均码率
5232
Kbps
视频编码
H.264
Smart264
开启
编码复杂度
中
I帧间隔
5
SVC
关闭
保存

3、 摄像头图像设置

4、 RK3399接入摄像头同一网段。

6.1.2 软件设置

1、 初始化SDK

```
// 初始化人脸库
FaceDetector.init(this, Config.licenseID, Config.licenseFileName);
// 设置最小人脸，小于此值的人脸不会被识别，可设置范围100-200
FaceDetector.getInstance().setMinFaceSize(120);
FaceDetector.getInstance().setCheckQuality(true);
// 头部的欧拉角，大于些值的不会被识别
FaceDetector.getInstance().setEulerAngleThreshold(45, 45, 45);
FaceDetector.getInstance().setVerifyLiveness(false);
```

设置视频流来源为 IPCamera，参见 RtspTestActivity

```
faceDetectManager.setImageSource(rtspImageSource);
```

2、 设置IPCamera的url，admin和Aa123456为摄像头的用户名密码，在浏览器里第一次访问摄像头时会提示你设置（默认ip为192.168.1.64）参见RtspTestActivity

```
@Override
protected void onStart() {
    super.onStart();
    // rtsp网络摄像头地址。具体格式见，摄像头说明书。
    String ip = PreferencesUtil.getString("ip", "");
    if (TextUtils.isEmpty(ip)) {
        Intent intent = new Intent(this, SettingsActivity.class);
        startActivityForResult(intent, 100);
        return;
    }
    String url = String.format(Locale.ENGLISH,
            "rtsp://admin:Aa123456@%s:554/h264/ch1/main/av_stream", ip);
    rtspImageSource.setUrl(url);
    faceDetectManager.start();
}
```

3、 修改人脸上传服务器地址及上传参数（建议识别接口调用在您的服务端进行，终端只负责上传人脸）

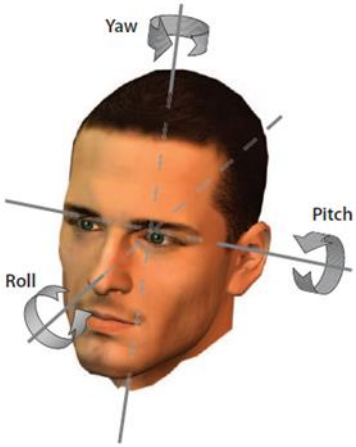
trackId：同一个人进入摄像头，在离开前trackId都相同

FaceEvent.OnEnter：为人脸进入、离开、更新摄像头的事件；

FaceEvent.OnUpdate：为人脸更新事件

FaceEvent.OnLeave：为人脸离开摄像头的事件；

Pitch、roll、yaw 为人脸的三个角度，可以用来过滤人脸，人脸越正，分数越高



```
private void uploadToCella(FaceCache cache, final File file, int faceEvent) {
    HashMap<String, String> params = new HashMap<>();
    switch (faceEvent) {
        case FaceEvent.OnEnter:
            params.put("event", "onEnter");
            break;
        case FaceEvent.OnUpdate:
            params.put("event", "onUpdate");
            break;
        case FaceEvent.OnLeave:
            params.put("event", "onLeave");
            break;
        default:
            break;
    }
    params.put("trackId", cache.getTrackId());
    float pitch = cache.faceInfo.headPose[0];
    float roll = cache.faceInfo.headPose[1];
    float yaw = cache.faceInfo.headPose[2];
    params.put("rotation", String.format(Locale.ENGLISH, "%f,%f,%f", pitch, roll

    APIService.getInstance().upload(new OnResultListener<Integer>() {
        @Override
        public void onResult(Integer result) {
            LogUtil.i("wtf", "upload ");
            if (file != null) {
                file.delete();
            }
        }

        @Override
        public void onError(FaceError error) {
            LogUtil.i("wtf", "upload error");
            if (file != null) {
                file.delete();
            }
        }
    }, file, params);
}
```

4、 运行

6.2 平板方案

6.2.1 硬件连接和设置

1、 平板接入网络

6.2.2 人脸检测

1、 初始化SDK

```
// 初始化人脸库
FaceDetector.init(this, Config.appName, Config.licenseFileName);
//        FaceSDKManager.getInstance().initialize(this, Config.appName, Config.licenseFileName);
// 设置最小人脸，小于此值的人脸不会被识别
FaceDetector.getInstance().setMinFaceSize(120);
//        FaceSDKManager.getInstance().getFaceTracker().set_min_face_size(300);
FaceDetector.getInstance().setCheckQuality(true);
//        FaceSDKManager.getInstance().getFaceTracker().set_isCheckQuality(true);
// 头部的欧拉角，大于些值的不会被识别
FaceDetector.getInstance().setEulerAngleThreshold(45, 45, 45);
//        FaceSDKManager.getInstance().getFaceTracker().set_isVerifyLive(false);
```

2、 设置视频流来源为CameraImageSource。具体示例参见CameraDetectActivity

3、 修改人脸上传服务器地址

4、 运行

6.3 人脸注册

示例代码里提供人脸注册功能，可以根据自己的需求放到其他应用来进行人脸注册。人脸注册调用的百度人脸云服务，调用此服务需要token，可以通过创建应用产生的ak/sk来获取。注册生成的人脸库用于后面的人脸1 :N。所以人脸注册和人脸ak ,sk使用的使用同一套ak/sk。为了防止ak/sk泄露，推荐把获取token的逻辑放在服务器端。当然也可以把人脸发给服务器进行人脸注册和人脸1：N。人脸注册提供人脸自动检测注册和相册选取注册两种方式

参见RegActivity和RegDetectActivity

6.3.1 人脸自动检测注册

1、 FaceSDK初始化及配置。

```
// 初始化人脸库
FaceDetector.init(this, Config.appName, Config.licenseFileName);
//        FaceSDKManager.getInstance().initialize(this, Config.appName, Config.licenseFileName);
// 设置最小人脸，小于此值的人脸不会被识别
FaceDetector.getInstance().setMinFaceSize(120);
//        FaceSDKManager.getInstance().getFaceTracker().set_min_face_size(300);
FaceDetector.getInstance().setCheckQuality(true);
//        FaceSDKManager.getInstance().getFaceTracker().set_isCheckQuality(true);
// 头部的欧拉角，大于些值的不会被识别
FaceDetector.getInstance().setEulerAngleThreshold(45, 45, 45);
//        FaceSDKManager.getInstance().getFaceTracker().set_isVerifyLive(false);
```

2、 创建CameraImageSource，设置预览PreviewView，开启预览。

3、 在屏幕用黄线框，框出人脸检测区域。

4、 FaceDetectManager设置ImageSource（相机），设置屏幕检测区域与相机预览图的映射。

5、 人脸检测回调数据，进行人脸质量判断（人脸角度，光线，在检测区域的位置）

6、 返回合适的人脸，调用FaceCropper.crop方法裁剪人脸区域，（非必须，也可以使用整张图片进行注册，但裁剪再压缩可以节省流量，同时减少网络传输时间）调用人脸注册接口进行注册（注册前需要获取注册的token）。注册参数为uid（可以自行设定，可以为自己系统的帐号的uid），groupid（人脸1：n时在此组中进行识别），userinfo（用户可以自行添加已经信息）。具体可以参见人脸注册接口：http://ai.baidu.com/docs#/Face-API/top

6.3.2 相册图片检测注册

1、 FaceSDK初始化及配置。

2、 调用相册读取一张照片。

3、 传入图片给FileImageSource ,调用FaceDetectManager.start()，进行人脸检测。

4、 调用人脸注册接口进行注册（注册前需要获取注册的token）。注册参数为uid（可以自行设定 ,可以为自己系统的帐号的uid）,groupid(人脸1 :n时在此组中进行）,userinfo（用户可以自行添加已经信息）。具体可以参见人脸注册接口：http://ai.baidu.com/docs#/Face-API/top 人脸注册

6.4 人脸识别（1：N）

人脸1：N在您的服务器调用的百度人脸云服务，调用此服务需要token，可以通过创建应用产生的ak/sk来获取。

1．服务端获取token

```
private void initAccessToken() {
    FaceMember.getInstance().init(getApplicationContext());
    FaceMember.getInstance().initAccessTokenWithAkSk(new OnResultListener<AccessToken>() {
        @Override
        public void onResult(AccessToken result) {
            if (result != null) {
                displayAccessTokenResult("获取token成功");
            } else {
                displayAccessTokenResult("获取token失败");
            }

            LogUtil.i("wtf", "AccessToken->" + result.getAccessToken());
        }

        @Override
        public void onError(OCRError error) {

        }
    }, getApplicationContext(), "", "");
}
```

2．设置上传人脸给服务器进行人脸1：N，并上传人脸。

```
private void uploadToCella(FaceCache cache, final File file, int faceEvent) {
    HashMap<String, String> params = new HashMap<>();
    switch (faceEvent) {
        case FaceEvent.OnEnter:
            params.put("event", "onEnter");
            break;
        case FaceEvent.OnUpdate:
            params.put("event", "onUpdate");
            break;
        case FaceEvent.OnLeave:
            params.put("event", "onLeave");
            break;
        default:
            break;
    }
    params.put("trackId", cache.getTrackId());
    float pitch = cache.faceInfo.headPose[0];
    float roll = cache.faceInfo.headPose[1];
    float yaw = cache.faceInfo.headPose[2];
    params.put("rotation", String.format(Locale.ENGLISH, "%f,%f,%f", pitch, roll, yaw));

    FaceMember.getInstance().upload(new OnResultListener<Integer>() {
        @Override
        public void onResult(Integer result) {
            LogUtil.i("wtf", "upload ");
            if (file != null) {
                file.delete();
            }
        }

        @Override
        public void onError(OCRError error) {
            LogUtil.i("wtf", "upload error");
            if (file != null) {
                file.delete();
            }
        }
    }, file, params);
}
```

3. 服务端进行调用百度人脸1：N服务，得到该人脸跟人脸库的已有人脸的相似度分数和用户信息。您可以根据分数值判断是否为通一个人，然后根据人脸新进行相应的业务逻辑

6.5 类及方法说明

FaceDetectManager 封装了图片采集到人脸检测流程。通过 ImageSource 获取图像数据，并将检测结果通过 OnFaceDetectListener 回调给使用者。

ImageSource 负责图像采集和渲染，目前有 CameraImageSource 用于从系统相机获取，FileImageSource 图片文件获取。以及 RtspImageSource 从网络视频流中获取。可根据自己的需求选择使用。

PreviewView 负责预览，因为相机的预览尺寸是固定尺寸的，为了适配屏幕会有缩放位移等操作。所以实际看到的图片和真实的图片是有差距。
mapToOriginalRect 方法提供了，View 坐标到原图坐标的转换。
mapFromOriginalRect 提供了从预览图到 View 坐标的转换，可用于实现绘制人脸区域，关键点等。

FaceProcessor 提供检测前对原始图像数据的处理回调。注册 demo 中的 DetectRegionProcessor 实现了边框位置图片裁剪，只检测，黄色边框中的内容。
FaceCropeer 提供了 从 argb 数组中裁剪人脸位置的数据。

7 常见问题

Q：license文件有什么用，该放在什么地方？

A:license文件需要申请，目的是作为sdk校验开发者的使用合法性，license文件放置位置不对或未放置license文件会导致没法使用sdk，一般应先申请license文件，并把申请得到的license文件，放置在assets目录下面。

Q：license文件失效了，不能用了怎么办？

A:icense文件申请时候有期限，如过期会导致校验失效，需要上官网申请延期。

Q：Android 申请时需要填入打包签名的 MD5,该 MD5 如何得来？ A:创建工程时生成签名文件 keystore.jks(该文件会用于最终发布是打包,请认真对待),在命令行工具 中使用 keytool -list -v -keystore keystore.jks 得到 md5,删除冒号。