FacePass Android版SDK使用指导书

该文档旨在介绍FacePass Android的安装和部分常见操作，以帮助用户快速上手实践。

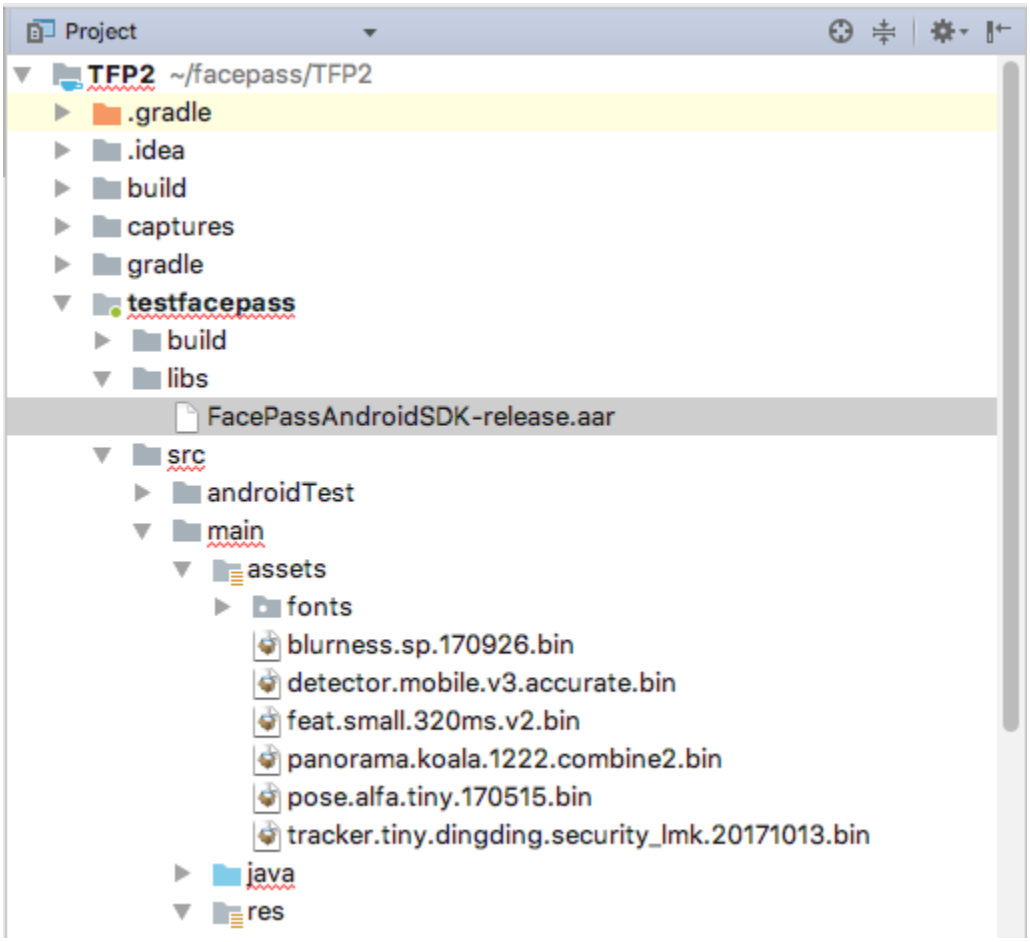
文档目录结构：

一.SDK部署

整个客户端SDK包含两部分：`FacePassAndroidSDK-release.aar`一个`aar`文件和多个`.bin`模型文件。

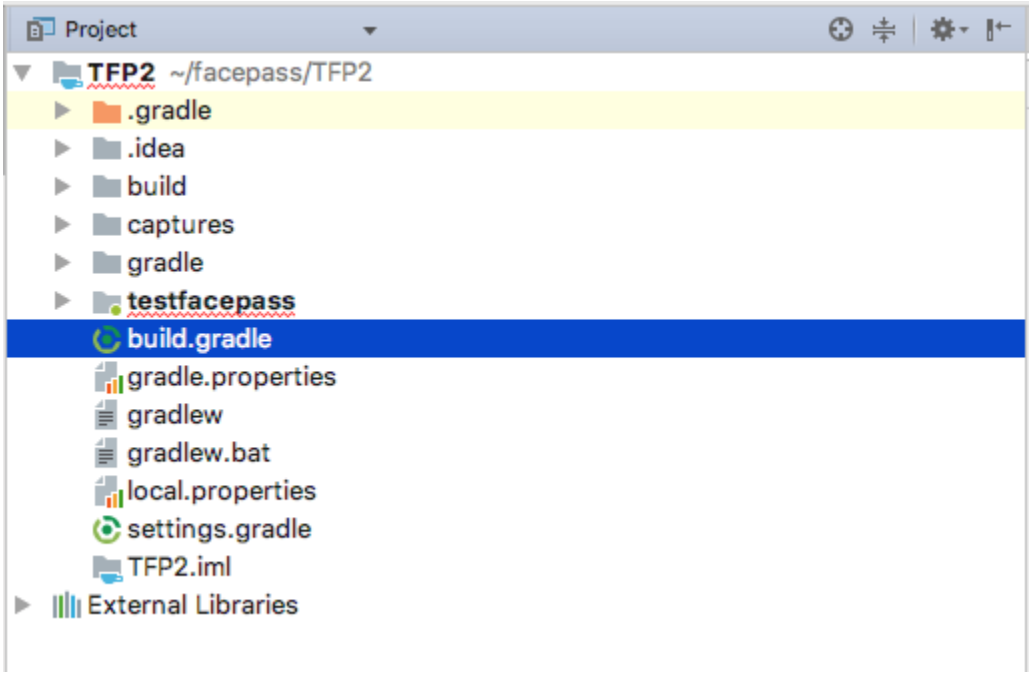
使用SDK开发安卓应用按照以下步骤（以使用Android Studio为例）：

1.将`aar`文件放入要开发的安卓应用的`libs`目录(在图中为testfacepass/libs)中，将模型文件(`*.bin`)放入`assets`目录中，如果没有目录请在对应位置新建。



2.修改工程根目录下的配置文件`build.gradle`，在`allprojects.repositories`下添加：
代码示例
?

```
flatDir { dirs 'libs' }
```



添加后如下：
代码示例
?

```
allprojects {
    repositories {
        jcenter()
        flatDir {
            dirs 'libs'
        }
    }
}
```

MyApplication

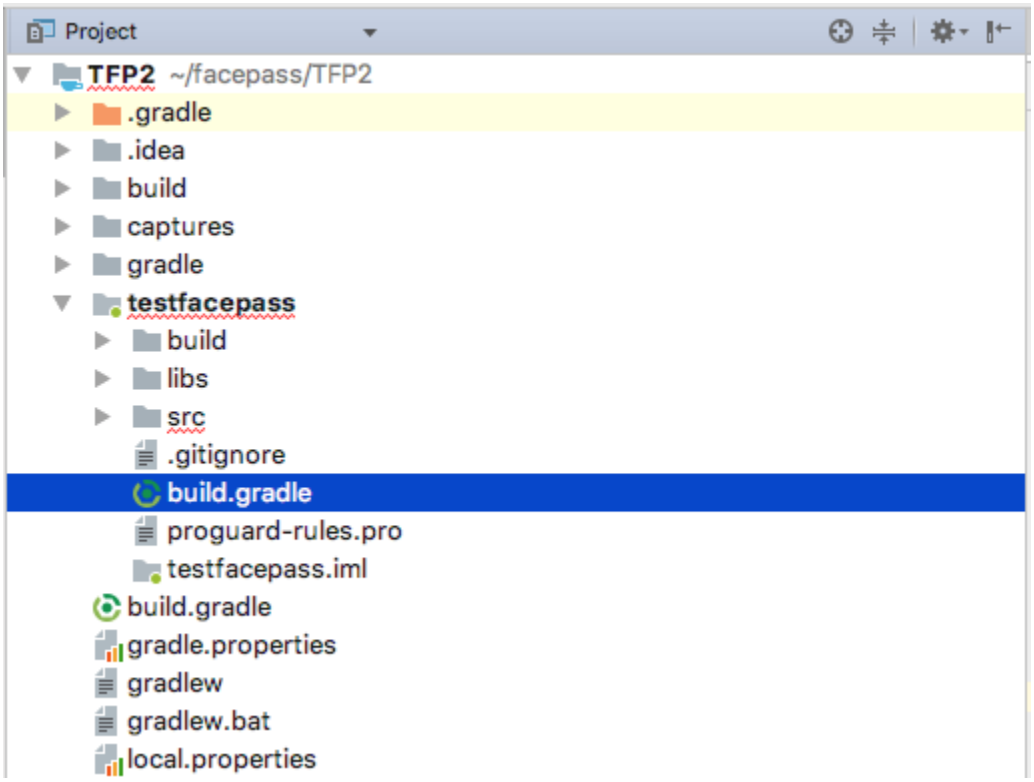
```
// Top-level build file where you can add configuration options common to all sub-projects/modules.

buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:2.3.3'

        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files
    }
}

allprojects {
    repositories {
        jcenter()
        flatDir {
            dirs 'libs'
        }
    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

3.修改工程app目录(在图中为testfacepass)下的配置文件build.gradle，在dependencies下添加:

代码示例

?

```
compile(name: 'FacePassAndroidSDK-release', ext: 'aar')
```



```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 25
    buildToolsVersion "26.0.0"

    defaultConfig {
        applicationId "megvii.testfacepass"
        minSdkVersion 16
        targetSdkVersion 25
        versionCode 1
        versionName "1.0"
        multiDexEnabled true
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"

    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
    packagingOptions {
        exclude 'META-INF/DEPENDENCIES'
        exclude 'META-INF/NOTICE'
        exclude 'META-INF/NOTICE.txt'
        exclude 'META-INF/LICENSE'
        exclude 'META-INF/LICENSE.txt'
        exclude 'META-INF/maven/org.java-websocket/Java-WebSocket/pom.xml'
    }
}


dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile(name: 'FacePassAndroidSDK-release', ext: 'aar')
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })
    compile('org.apache.httpcomponents:httpmime:4.3') {
        exclude module: "httpclient"
    }
    compile 'com.android.volley:volley:1.0.0'
    compile 'com.jakewharton:butterknife:8.5.1'
    compile 'org.apache.httpcomponents:httpcore:4.3.3'
    compile 'com.android.support:appcompat-v7:25.3.1'
    compile 'com.android.support.constraint:constraint-layout:1.0.2'
    testCompile 'junit:junit:4.12'
    annotationProcessor 'com.jakewharton:butterknife-compiler:8.5.1'
}

```

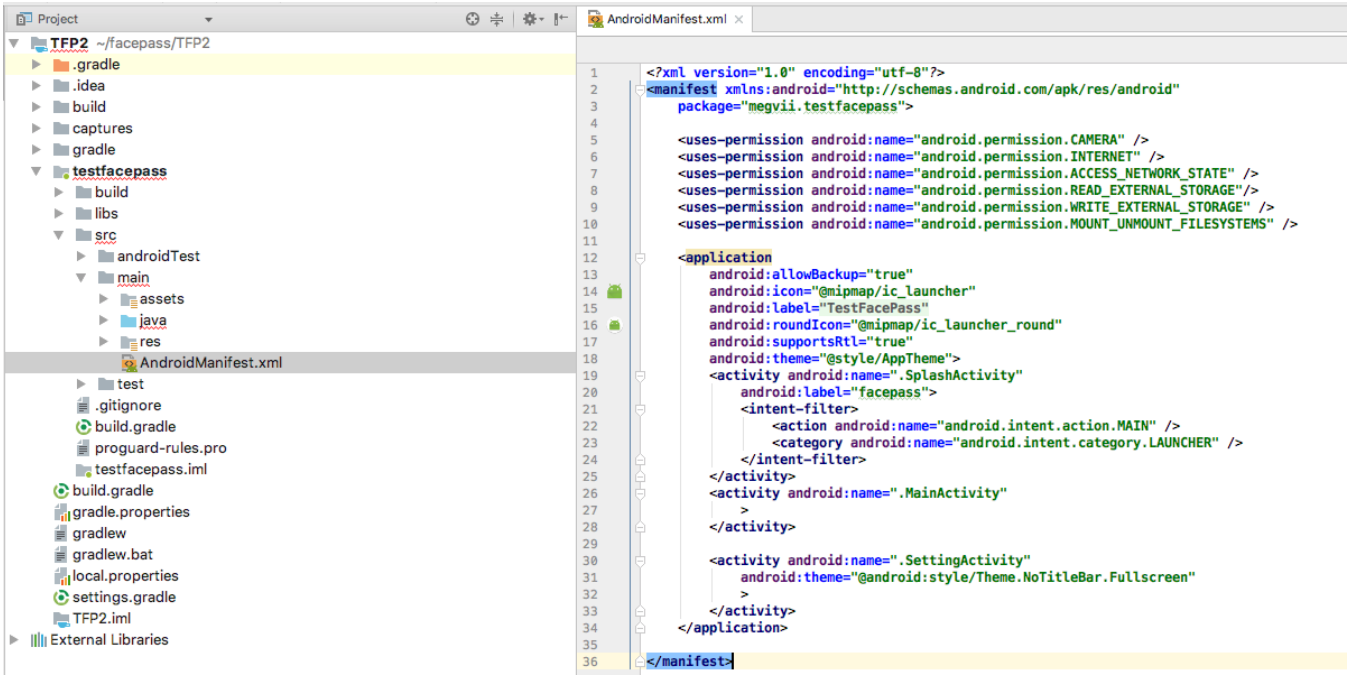
4.修改AndroidManifest.xml文件，添加:

代码示例

?

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

注：Android 6.0之后的系统中需要通过其他方式添加这些权限，需要的权限有相机权限、网络权限、获取网络状态权限、读写文件权限、创建删除文件的权限，详见示例代码。



5.使用SDK进行开发。

6.如果需要运行Demo进行体验，请先修改src/main/java/megvii/testfacepass/MainActivity.java中的相关配置：

API

?

```
/* IP */
private static final String serverIP_offline = "10.104.44.50";
private static final String serverIP_online = "10.199.1.14";
private static final String authIP = "47.94.181.237";

/* Group */
private static final String group_name = "face-pass-test-x";
```

二.快速搭建步骤

请阅读《FacePass客户端SDK文档》，本文档仅介绍核心逻辑，不对具体API展开叙述。

1.SDK初始化示例

同1.4

?

```
/*  */
if (!hasPermission()) {
    requestPermission();
}

/* FacePass SDKassets */
FacePassModel trackModel = FacePassModel.initModel(getApplicationContext().getAssets(), "tracker.tiny.dingding.
security_lmk.20171013.bin");
FacePassModel poseModel = FacePassModel.initModel(getApplicationContext().getAssets(), "pose.bin");
FacePassModel blurModel = FacePassModel.initModel(getApplicationContext().getAssets(), "blurness.sp.170926.bin");
```

```
FacePassModel searchModel = FacePassModel.initModel(getApplicationContext().getAssets(), "feat.small.v3.bin");
FacePassModel livenessModel = FacePassModel.initModel(getApplicationContext().getAssets(), "panorama.mobile.bin"
);
FacePassModel detectModel = FacePassModel.initModel(getApplicationContext().getAssets(), "detector.mobile.v3.
accurate.bin");

/*  */
/* "apiKey""apiSecret"StringFace++ */
String authURL = "www.megvii-inc.com";
String apiKey = "xxxxxxx";
String apiSecret = "yyyyyyyyyy";

FacePassHandler.getAuth(authURL, apiKey, apiSecret);

/* isAuthorized */
if (FacePassHandler.isAuthorized()) {
    //xxx;
}

/* SDK */
FacePassHandler.initSDK(getApplicationContext());

/* SDKSDK Handler */
FacePassHandler handler;
new Thread() {
    @Override
    public void run() {
        while (isFinishing()) {
            /* SDK */
            if (FacePassHandler.isAvailable()) {
                /* SDK  */
                float searchThreshold = 75f;
                float livenessThreshold = 30f;
                boolean livenessEnabled = true;
                int faceMinThreshold = 150;
                FacePassPose poseThreshold = new FacePassPose(30f, 30f, 30f);
                float blurThreshold = 0.2f;
                float lowBrightnessThreshold = 70f;
                float highBrightnessThreshold = 210f;
                float brightnessSTDThreshold = 60f;
                int retryCount = 2;
                int rotation = FacePassImageRotation.DEG0;
                String fileRootPath = Environment.getExternalStorageDirectory().getPath() + "/";

                FacePassConfig config;
                try {

                    /*  */
                    config = new FacePassConfig(searchThreshold, livenessThreshold, livenessEnabled,
                                                faceMinThreshold, poseThreshold, blurThreshold,
                                                lowBrightnessThreshold, highBrightnessThreshold,
brightnessSTDThreshold,
                                                retryCount, rotation,fileRootPath,
                                                trackModel, poseModel, blurModel, searchModel, livenessModel,
                                                detectModel);
                    /* SDK */
                    handler = new FacePassHandler(config);
                } catch (FacePassException e) {
                    e.printStackTrace();
                    return;
                }
                return;
            }
            try {
                /* SDK */
                sleep(500);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
```

```
    }
}.start();
```

2.创建底库

通过FacePassHandler的createLocalGroup(groupName) 创建底库。

通过FacePassHandler的getLocalGroups()可查看本地所有底库。

具体参见:
代码示例
?

```
/* Group */
private static final String group_name = "face-pass-test-5";
/*  */
String[] localGroups = mFacePassHandler.getLocalGroups();
/*  */
boolean isSuccess = false;
try {
    isSuccess = mFacePassHandler.createLocalGroup(groupName);
} catch (FacePassException e) {
    e.printStackTrace();
}
```

3.客户端新增一张人脸

FacePass可以将bitmap插入本地底库之中。可以自由选择图片来源。返回值之中包含facetoken。

具体参见:
代码示例
?

```
try {
    FacePassAddFaceResult result = mFacePassHandler.addFace(bitmap);
    if (result != null) {
        if (result.result == 0) {
            toast("add face successfully");
            faceTokenEt.setText(new String(result.faceToken));
        } else if (result.result == 1) {
            toast("no face ");
        } else {
            toast("quality problem");
        }
    }
} catch (FacePassException e) {
    e.printStackTrace();
    toast(e.getMessage());
}
```

4.绑定底库和人脸

将group_name和上面返回的facetoken绑定。返回值为一个布尔值，代表是否绑定成功。

具体参见:
代码示例
?

```
try {
    boolean b = mFacePassHandler.bindGroup(groupName, faceToken);
    String result = b ? "success " : "failed";
    toast("bind  " + result);
} catch (Exception e) {
    e.printStackTrace();
    toast(e.getMessage());
}
```

5.添加相机帧

和1.5相同

客户端从相机帧生成FacePassImage对象，加到feedFrame。

客户端在完成各种初始化之后，需要根据从相机的返回函数中根据图片码流生成一个FacePassImage对象，然后调用FacePassHandler的feedFrame函数，并返回FacePassDetectionResult。

具体参见：
代码示例
?

```
/* SDKFacePassImage */
FacePassImage image = new FacePassImage(cameraPreviewData.nv21Data,
    cameraPreviewData.width,
    cameraPreviewData.height,
    FacePassImageRotation.DEG0,
    FacePassImageType.NV21);

/* FacePassImageSDK */
FacePassDetectionResult detectionResult = handler.feedFrame(image);
```

6.调用Recognize函数，处理结果

根据feedFrame接口返回的FacePassDetectionResult的message数组数据，FacePassHandler调用mFacePassHandler.recognize接口进行识别，需要传入group_name。由于recognize是耗时操作，需要单独开启一个线程来处理数据。建议用一个阻塞队列来维护需要处理的数据，通过RecognizeThread线程轮训的从队列中获取需要处理的数据。

具体参见：
代码示例
?

```
/*recognize*/
FacePassDetectionResult detectionResult = handler.feedFrame(image);
/*messageresultrecognize*/
if (detectionResult != null && detectionResult.message.length != 0) {
    FacePassRecognitionResult[] recognizeResult = mFacePassHandler.recognize(group_name, detectionResult.
message);
    if (recognizeResult != null && recognizeResult.length > 0) {
         for (FacePassRecognitionResult result : recognizeResult) {
                String faceToken = new String(result.faceToken);
                if (FacePassRecognitionResultType.RECOG_OK == result.facePassRecognitionResultType) {
                /* facetoken */
                    getFaceImageByFaceToken(result.trackId, faceToken);
                 }

            } catch (InterruptedException e) {
                e.printStackTrace();
            } catch (FacePassException e) {
                e.printStackTrace();
            }
       }
   }
}
```

至此，一次完整的纯人脸识别操作就完成了！

以上就是FacePass SDK的核心部署流程和实现第一次人脸识别的流程，希望能够帮助到您！

如有更多问题，请咨询技术支持人员。