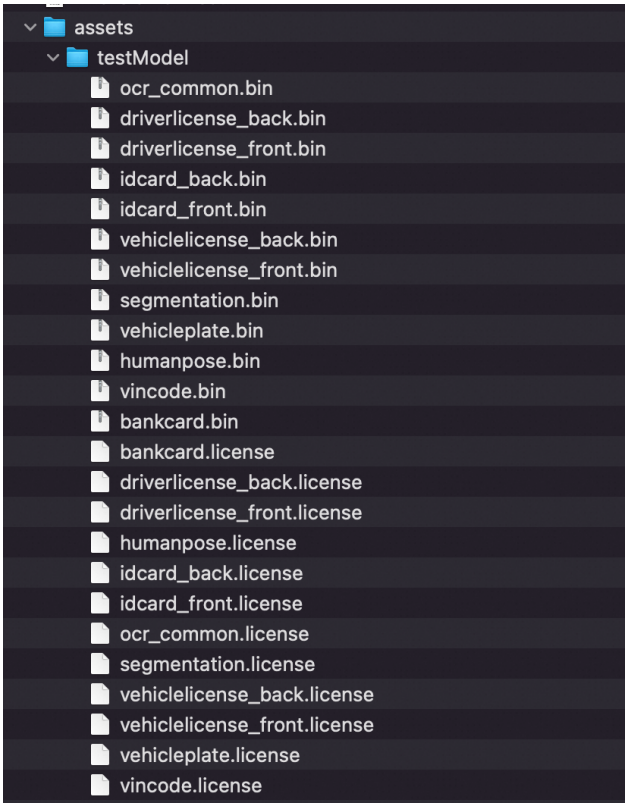
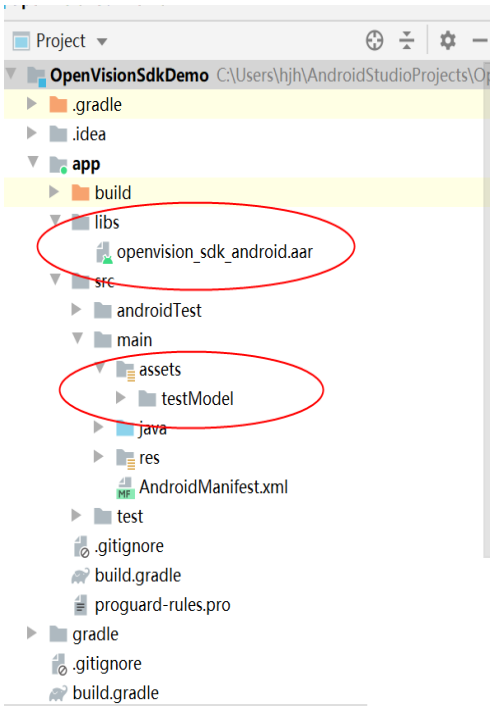
Android 端 openVisionSdk 集成说明

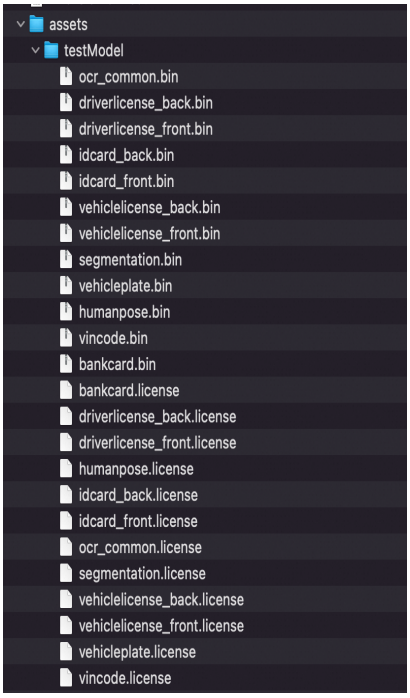
Android Studio 配置工程

1、获取相关资源压缩包（由阿里云相关人员提供下载链接）后，解压压缩包，可看到如下资源文件 aar 包及支持相关能力的 bin、license 文件。如下图：



2、把 aar 包拷贝到工程（此处截图以官方示例 Demo 为例子）的 libs 目录下，在工程的 assets 目录下新建文件夹，把剩余的 bin、license 文件拷贝到该文件夹下，如下图：



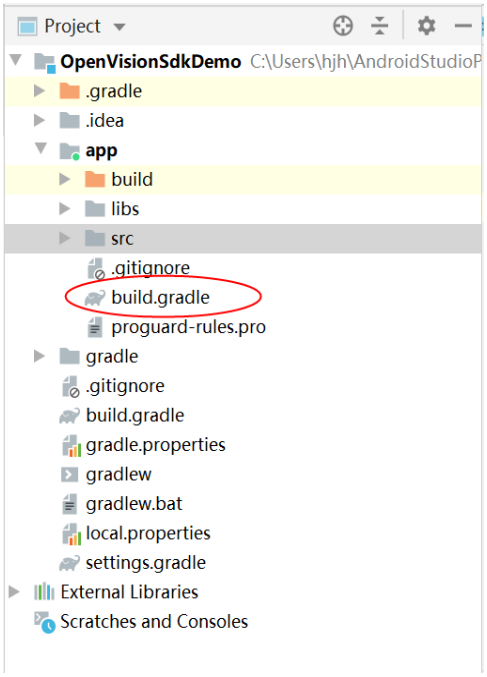


3、

(1)在配置文件 AndroidManifest 中增加相机和读写相关权限，Android API23（即 Android 6.0）及以上需要添加动态访问权限：

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

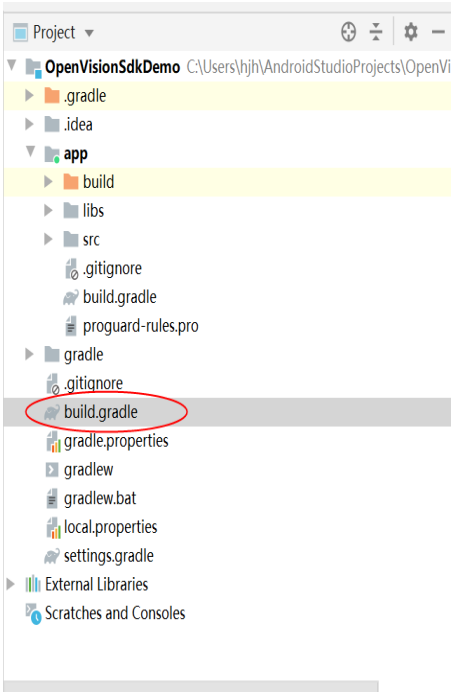
(2)在主工程的 build.gradle 文件相关配置设置及 dependencies 依赖，主工程的 build.gradle 文件在 Project 目录中位置如下图：



相关配置 minSdkVersion 最小为 18，targetSdkVersion 和 compileSdkVersion 可以按需配置：

```
  android {
   defaultConfig {
       ndk {
           //设置支持的 SO 库架构（开发者可以根据需要，选择一个或多个平台的 so）
           abiFilters "armeabi-v7a", "arm64-v8a"
           }
        }
  }
  dependencies {
    implementation fileTree(dir: "libs", include: ["*.jar"])
    //相关 aar 包依赖
    implementation(name: 'openvision_sdk_android', ext: 'aar')
}
```

（3）在 Project 的 build.gradle 文件中配置 repositories，添加 maven 或 jcenter 仓库地址，Android Studio 默认会在 Project 的 build.gradle 为所有 module 自动添加 jcenter 的仓库地址，如果已存在，则不需要重复添加。Project 的 build.gradle 文件在 Project 目录中位置如图所示：



配置如下：

```
allprojects {
  repositories {
    google()
    jcenter() // 或者 mavenCentral()

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

4、相关功能实现，通过相机预览或相册选择图片实现银行卡、身份证（正、反面）、驾驶证（正、反面）、行驶证（正、反面）、车牌的信息识别。（注：相关功能的实现依赖于 aar 包是否包含该功能及项目中是否有对应的 bin 和 license 文件）。

声明相机类对象 `CVCameraView`、算法类对象 `CVEngine`、及算法结果回调 `CVEngineCallback`：

```
//声明相机类对象
private CVCameraView mCameraView;
//声明算法类对象
private CVEngine mCurEngine;
//算法结果回调
```

```
6.  private CVEngineCallback mEngineCallback = new CVEngineCallback() {
7.      @Override
8.      public void onResult(CVEngine engine, CVResult result) {
9.          //result 对象可转换为 json 字符串，按需获取需要数据，需切回 UI 线程展示
10.         Log.i(TAG, engine.getClass().getSimpleName() + " result: " + JSON.toJSONString(result));
11.         //如果 result.error.code 等于 0 即为处理成功
12.         if (result.error.code == Error.ERROR_NONE && result.results != null) {
13.         //可获得算法处理过的图片以 Bitmap 形式返回
14.          if (result.image != null) {
15.
16.          }
17.     }
18. };
```

实现相机内矩形框，目的是为了把相关证件、卡片放矩形框内算法更准确识别，可自己实现也可参考 Demo，不添加识别框有可能会降低识别准确率。实例化相机 View，XML 文件中添加父布局加载相机 View 及矩形框：

```
1.  //实例化相机类
2.  float[] roi = new float[]{0.02f,0.218f,0.96f,0.294f};
3.  mCameraView = new CVCameraView(this);
4.  mCameraView.setRoi(roi);
5.  //画类似银行卡、身份证的矩形框,目的算法可以更好识别信息，可自己实现，也可参考 Demo
6.  ScanFrameView mScanRect = new ScanFrameView(this);
7.  mScanRect.setRoi(roi);
8.  //添加父布局加载相机 View 及矩形识别框
9.  RelativeLayout.LayoutParams lp = new RelativeLayout.LayoutParams(
10.     ViewGroup.LayoutParams.MATCH_PARENT, ViewGroup.LayoutParams.MATCH_PARENT);
11. RelativeLayout container = (RelativeLayout) findViewById(R.id.container);
12. container.addView(mCameraView, lp);
13. container.addView(mScanRect, lp);
```

启动相机：

```
1.  mCameraView.startPreview();
```

启动算法，加载对应的算法库：

```
1.  private void switchEngine(String engineName) {
2.          if (TextUtils.isEmpty(engineName)) {
3.              return;
4.          }
5.          if (mCurEngine != null) {
6.              mCurEngine.release();
7.              mCameraView.removeEngine(mCurEngine);
8.          }
9.          //正反面设置,只有身份证、驾驶证、行驶证可设置
```

```
        boolean isFace = true;

        CVConfig config = new CVConfig();
        switch (engineName) {
            case Engines.BANKCARD://银行卡
                config.license = MiscUtil.getAssetFileContent(this, "testModel/bankcard.license");  // getAssetFileContent 方法可参考 Demo 也可自己实现
                config.model = MiscUtil.getAssetFilePath(this, "testModel/bankcard.bin");
                mCurEngine = new Bankcard(this);
                break;
            case Engines.IDCARD://身份证
                if (isFace) {//正面
                    config.license = MiscUtil.getAssetFileContent(this, "testModel/idcard_front.license");
                    config.model = MiscUtil.getAssetFilePath(this, "testModel/idcard_front.bin");
                } else {//反面
                    config.license = MiscUtil.getAssetFileContent(this, "testModel/idcard_back.license");
                    config.model = MiscUtil.getAssetFilePath(this, "testModel/idcard_back.bin");
                }
                mCurEngine = new IDCard(this);
                break;
            case Engines.DRIVER_LICENSE://驾驶证
                if (isFace) {//正面
                    config.license = MiscUtil.getAssetFileContent(this, "testModel/driverlicense_front.license");
                    config.model = MiscUtil.getAssetFilePath(this, "testModel/driverlicense_front.bin");
                } else {//反面
                    config.license = MiscUtil.getAssetFileContent(this, "testModel/driverlicense_back.license");
                    config.model = MiscUtil.getAssetFilePath(this, "testModel/driverlicense_back.bin");
                }
                mCurEngine = new DriverLicense(this);
                break;
            case Engines.VEHICLE_LICENSE://行驶证
                if (isFace) {//正面
                    config.license = MiscUtil.getAssetFileContent(this, "testModel/vehiclelicense_front.license");
```

```
                    config.model = MiscUtil.getAssetFilePath(this, "testModel/vehiclelicense_front.bin");
                } else {//反面
                    config.license = MiscUtil.getAssetFileContent(this, "testModel/vehiclelicense_back.license");
                    config.model = MiscUtil.getAssetFilePath(this, "testModel/vehiclelicense_back.bin");
                }
                mCurEngine = new VehicleLicense(this);
                break;
            case Engines.VEHICLE_PLATE://车牌
                config.license = MiscUtil.getAssetFileContent(this, "testModel/vehicleplate.license");
                config.model = MiscUtil.getAssetFilePath(this, "testModel/vehicleplate.bin");
                mCurEngine = new VehiclePlate(this);
                break;
             case  Engines.COMMON_OCR://通用文字识别
config.license = MiscUtil.getAssetFileContent(this,
"testModel/ocr_common.license");
config.model = MiscUtil.getAssetFilePath(this, "testModel/ocr_common.bin");
mCurEngine = new CommonOcr(this);
                break ;
            default:
                XLog.e(TAG, "unknown engine: " + engineName);
                return;
        }
        Error error = mCurEngine.init(config);
        //非 0 返回，错误码及错误提示
        if (error.code != Error.ERROR_NONE) {
            final String prefix = mCurEngine.getClass().getSimpleName() + " Error";
            JSON.toJSONString(error);  //此处用的阿里 fastjson 库解析，也可用其他解析，得到错误码及错误信息。
        } else {
            mSelectedEngineName = engineName;
            mCameraView.addEngine(mCurEngine, mEngineCallback);
        }
    }
```

在需要调用起算法时直接调用：

```
switchEngine(Engines.BANKCARD);
```

Engines 类说明：

```
public class Engines {
    public static final String BANKCARD = "Bankcard";//银行卡
    public static final String IDCARD = "IDCard";//身份证
    public static final String DRIVER_LICENSE = "DriverLicense";//驾驶证
    public static final String VEHICLE_LICENSE = "VehicleLicense";//行驶证
    public static final String VEHICLE_PLATE = "VehiclePlate";//车牌
    public static final String POSE_DETECT = "PoseDetect";//人体姿态
    public static final String SEGMENTATION = "Segmentation";//人脸分割
    public static final String COMMON_OCR = "CommonOcr";//通用文字识别
}
```

拍照单张图片处理：

```
findViewById(R.id.take_picture).setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            mCameraView.takePicture(mCurEngine, new CVEngineCallback() {
                @Override
                public void onResult(CVEngine engine, CVResult result) {
                    Log.d(TAG, engine.getClass().getSimpleName() + " takePicture "
                            + " result: " + JSON.toJSONString(result));
                }
            });
        }
    });
```

`Activity` 各生命周期管理：

```
@Override
    protected void onResume() {
        super.onResume();
        mCameraView.startPreview();
    }
 @Override
    protected void onPause() {
        super.onPause();
        mCameraView.stopPreview();
    }
```

```
@Override
    protected void onDestroy() {
        super.onDestroy();
        if (mCurEngine != null) {
            mCurEngine.release();
            mCameraView.removeEngine(mCurEngine);
        }
```

```
        mCameraView.releaseCamera();
    }
```

5、相关功能实现,人脸分割，实现把人像抠出来的效果。

声明相机类对象 CVCameraView、算法类对象 SegmentationEngine：

```
//相机类
private CVCameraView mCameraView;
//算法处理类
private SegmentationEngine mSegmentation;
```

初始化 View，在 XML 文件中添加父布局（RelativeLayout、LinearLayout 、ConstraintLayout 等均可），加载相机 View 进父布局：

```
mCameraView = new CVCameraView(this);
RelativeLayout.LayoutParams lp = new RelativeLayout.LayoutParams(
        ViewGroup.LayoutParams.MATCH_PARENT, ViewGroup.LayoutParams.MATCH_PARENT);
RelativeLayout container = (RelativeLayout) findViewById(R.id.container);
container.addView(mCameraView, lp);
```

打开相机：

```
mCameraView.startPreview();
```

实例化算法，并添加相机流结果回调：

```
CVConfig segmentationConfig = new CVConfig();
segmentationConfig.license = MiscUtil.getAssetFileContent(this, "testModel/segmentation.license");
segmentationConfig.model = MiscUtil.getAssetFilePath(this, "testModel/segmentation.bin");
mSegmentation = new SegmentationEngine(this);
Error error = mSegmentation.init(segmentationConfig);
if (error.code != Error.ERROR_NONE) {
final String prefix = mSegmentation.getClass().getSimpleName() + " Error";
    String message = JSON.toJSONString(error)  //此处用的阿里 fastjson 库解析，也可用其他解析，得到错误码及错误信息。
    return;
}
//算法结果回调
mCameraView.addEngine(mSegmentation, new CVEngineCallback() {
    @Override
    public void onResult(CVEngine engine, CVResult result) {
        Log.i(TAG, "Segmentation result: " + result);

        SegmentationResult segmentationResult = (SegmentationResult) result;
       if (segmentationResult.error.code == Error.ERROR_NONE && segmentationResult.outputFrame != null) {
```

```
            //相机流获取的 bitmap 界面展示需在主线程操作
            if (segmentationResult  != null && segmentationResult .outputFrame != null) {
convertARGBFrameToBitmap(segmentationResult.outputFrame);
            }
        }
    }
});
```

convertARGBFrameToBitmap 类：

```
private static Bitmap convertARGBFrameToBitmap(SegmentationResult.FrameData frame) {
    return Bitmap.createBitmap(frame.data, frame.width, frame.height, Bitmap.Config.ARGB_8888);
}
```

从相册选择图片进行处理：

```
private void chooseImage() {
    Intent getIntent = new Intent(Intent.ACTION_GET_CONTENT);
    getIntent.setType("image/*");

    Intent pickIntent = new Intent(Intent.ACTION_PICK, android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
    pickIntent.setType("image/*");

    Intent chooserIntent = Intent.createChooser(getIntent, "Select Image");
    chooserIntent.putExtra(Intent.EXTRA_INITIAL_INTENTS, new Intent[] {pickIntent});

startActivityForResult(chooserIntent, PICK_IMAGE);  //   private static final int PICK_IMAGE = 1;

}
```

相册返回之后的回调：

```
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == PICK_IMAGE && resultCode == Activity.RESULT_OK) {
        if (data == null || data.getData() == null) {
            return;
        }
        try {
            InputStream is = getContentResolver().openInputStream(data.getData());
            Bitmap bitmap = BitmapFactory.decodeStream(is);
            if (bitmap != null) {
                SegmentationResult result = (SegmentationResult) mSegmentation.run(bitmap, null);
                if (result.error.code != Error.ERROR_NONE ) {
                    final String prefix = mSegmentation.getClass().getSimpleName() + " Error";
```

```
                String message = JSON.toJSONString(result.error) ; //此处用的阿里 fastjson 库解析，也可用其他解析，得到错误码及错误信息。
            } else {
                SegmentationResult segmentationResult = (SegmentationResult) result;
        if (segmentationResult.error.code == Error.ERROR_NONE && segmentationResult.outputFrame != null) {
                SegmentationResult.FrameData frameData = segmentationResult.outputFrame;
                //选择相册图片处理之后的 bitmap
                Bitmap bitmap1 = Bitmap.createBitmap(frameData.data, frameData.width, frameData.height, Bitmap.Config.ARGB_8888);
            }
          } }
        } catch (Exception e) {
            Log.e(TAG, "failed to open file", e);
        }
    }
  }
```

Activity 各生命周期管理：

```
@Override
protected void onResume() {
    super.onResume();
    mCameraView.startPreview();
}
```

```
@Override
protected void onPause() {
    super.onPause();
    mCameraView.stopPreview();
}
```

```
@Override
protected void onDestroy() {
    super.onDestroy();
    if (mSegmentation != null) {
        mSegmentation.release();
    }
    mCameraView.removeEngine(mSegmentation);
    mCameraView.releaseCamera();
}
```

6、相关功能实现,人体姿态，获取人体描边的位置点。

声明相机类对象 CVCameraView、算法类对象 PoseDetectEngine：

```
1.  //相机类对象
2. private CVCameraView mCameraView;
3. //算法类对象
4. private PoseDetectEngine mPoseDetect;
```

初始化 View，在 XML 文件中添加父布局（RelativeLayout、LinearLayout、ConstraintLayout 等均可），加载相机 View 进父布局：

```
1.  mCameraView = new CVCameraView(this);
2. RelativeLayout.LayoutParams lp = new RelativeLayout.LayoutParams(
3.         ViewGroup.LayoutParams.MATCH_PARENT, ViewGroup.LayoutParams.MATCH_PARENT);
4. RelativeLayout container = (RelativeLayout) findViewById(R.id.container);
5. container.addView(mCameraView, lp);
```

打开相机：

```
1.  mCameraView.startPreview();
```

切换相机前后摄像头：

```
1. mCameraView.switchCamera();
```

实例化算法，并添加相机流结果回调：

```
1. CVConfig poseDetectConfig = new CVConfig();
2. poseDetectConfig.license = MiscUtil.getAssetFileContent(this, "testModel/humanpose.license");
3. poseDetectConfig.model = MiscUtil.getAssetFilePath(this, "testModel/humanpose.bin");
4. mPoseDetect = new PoseDetectEngine(this);
5. Error error = mPoseDetect.init(poseDetectConfig);
6. if (error.code != Error.ERROR_NONE) {
7.    String prefix = mPoseDetect.getClass().getSimpleName() + " Error";
8.    //此处用的阿里 fastjson 库解析，也可用其他解析，得到错误码及错误信息。
9.    String Messages = JSON.toJSONString(error) ;
10.   return;
11. }
12.
13. mCameraView.addEngine(mPoseDetect, new CVEngineCallback() {
14.    @Override
15.    public void onResult(CVEngine engine, CVResult result) {
16.       Log.i(TAG, "PoseDetect result: " + result);
17.        Log.d(TAG, "showResult: " + JSON.toJSONString(result));
18.       if (result != null && result.error.code == Error.ERROR_NONE) {
          PoseDetectResult mDetectResults;
          mDetectResults = ((PoseDetectResult) result);
19.          if (mDetectResults != null && mDetectResults.results != null) {
20.       for (Result[] results : mDetectResults.results) {
```

```
            for (Result result : results) {
                for (PointF pt : result.pos) { //节点获取后需在主线程操作
                    //通过 for 循环获取所有节点信息，在画布上绘制（可参考 Demo 示例）及用于其他操作

                }
            }
        }
    } else {
            //未获取到结果
    }
    }
});
```

`Activity` 各生命周期管理：

```
@Override
protected void onResume() {
    super.onResume();
    mCameraView.startPreview();
}
```

```
@Override
protected void onPause() {
    super.onPause();
    mCameraView.stopPreview();
}
```

```
@Override
protected void onDestroy() {
    super.onDestroy();
    if (mPoseDetect != null) {
        mPoseDetect.release();
    }
    mCameraView.removeEngine(mPoseDetect);
    mCameraView.releaseCamera();
}
```

7、其他：

文档中的 `JSON.toJSONString(result)`依赖解析库（也可用 `gson` 解析代替，混淆参考其官方文档）：

```
implementation 'com.alibaba:fastjson:1.2.70'
```

混淆保持：

```
#aar 包混淆保持
-keep class com.aliyun.openvision.**{ *; }
-keep class com.alipay.xmedia.**{ *; }
-keep class com.ant.phone.**{ *; }
#fastjson 混淆相关，用 gson 参考 gson 官方文档混淆
-dontwarn com.alibaba.fastjson.**
-dontskipnonpubliclibraryclassmembers
-dontskipnonpubliclibraryclasses
-keep class com.alibaba.fastjson.**{*;}
-keep class * implements java.io.Serializable { *; }
-keepattributes *Annotation
-keepattributes Signature
```

注：证件检测必须为所选证件项进行检测，包括正、反面的区别，否则会检测没反应或检测失败。