日志@十七蝉

借助文字，一个人可以向众人说话，死者可以向生者说话，生者可以向未生者说话。

2014-03-20

OpenCV在Android环境下的使用方法

按照使用语言角度，在Android下使用OpenCV有以下几种方式：

1. 完全使用Java语言
2. 完全使用C++语言
3. 混合使用Java和C++语言

下面分别讲一下怎么做，并说明这样做可能需要注意的问题。

完全使用Java语言

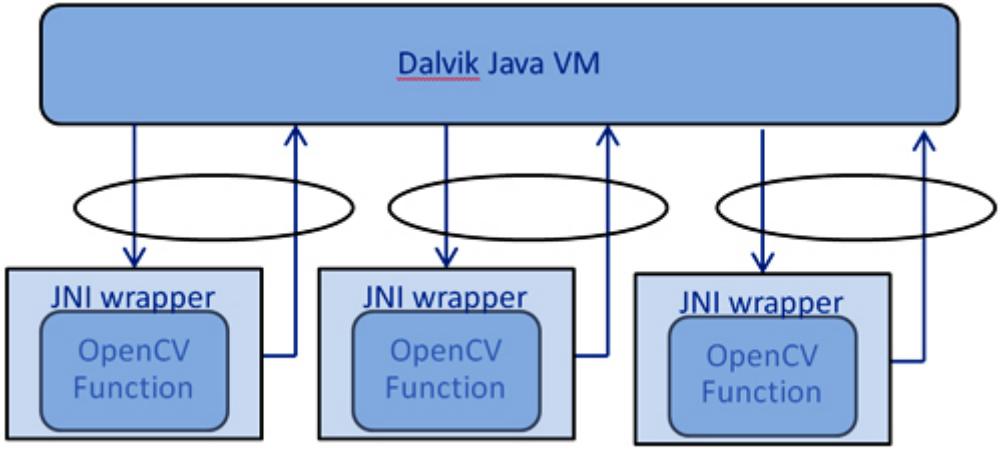
开始我是希望完全用Java语言开发的（也就是使用OpenCV4Android），主要好处是：

- 开发效率高，我指的是Java比C++生产代码的速度要快一些，至少在我的团队是这样
- 编译环境搭建比较简单，尤其是通过async initialization的情况下

但是，有得就有失，OpenCV是C++写的，虽然OpenCV组织也推出了供开发者使用的OpenCV4Android，但并不是所有的OpenCV C++功能都能在OpenCV4Android中找到。这也挺好理解的，人家是先写出C++的功能，然后再考虑在OpenCV4Android中实现。

所以，使用OpenCV4Android的问题是有些功能OpenCV里有，但它没有。

另外一个问题是，需要在架构上避免频繁的JNI操作，这样对性能不利。如下图：



该图的来源是：Developing OpenCV computer vision apps for the Android platform

实际上，OpenCV4Android只是在原来C++本地库的基础上做了个Java/JNI的包装（wrapper）。

在开发上，又有两种加载OpenCV4Android的方式：

- async initialization，这是官方文档里推荐的，你都不需要在自己Android项目里加入OpenCV的本地库，OpenCV提供了一个可在Google Play上下载的App，你只需要将OpenCV4Android的Android库项目（libraray project）加入到你Android项目中即可，官方文档说明了这个配置过程
- static initialization，官方文档建议只在开发阶段使用，和前者的不同在于，你需要将相关的本地库（so文件）部署到项目中来

如果我来选择，我可能希望正式环境下也使用static initialization的方式，原因是：

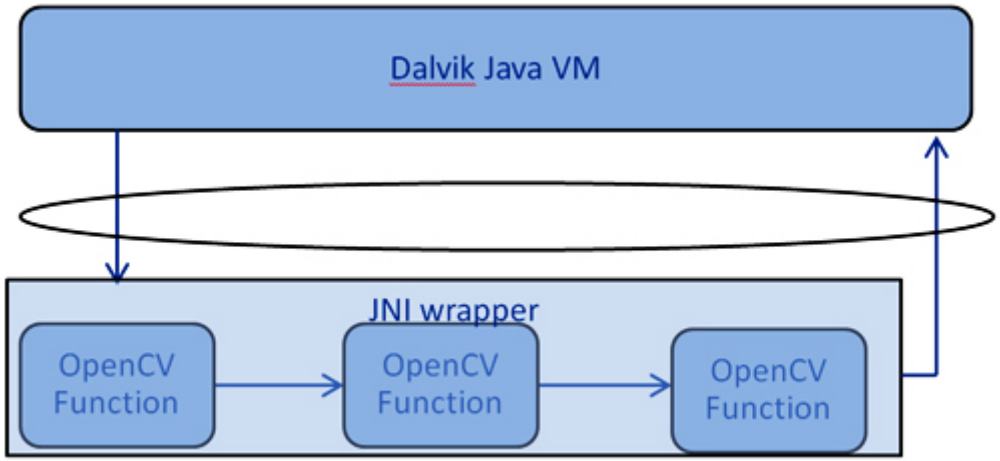
- 国内Android设备使用Google Play的并不多，虽然可以指定下载链接，但是增加了用户操作，不方便
- 可能存在这样的问题，通过Google Play下载的版本和开发使用的OpenCV4Android Java库版本不匹配

TODO: 提供一个这样的github项目示例

完全使用C++

在前一阶段的开发，我们使用的是这种方式。

希望能尽量减少混合编程带来的复杂性。借图说明：



但是，项目中业务的复杂性，需要考虑面向对象和它们的生命周期，这样可以让架构简明，而使用纯粹的C++解决方案，Java无法管理到OpenCV的对象，比如矩阵（Mat）。我们对C++的控制能力有限，担心如果设计复杂的对象和生命周期的结构，会带来错误和调试上的效率问题。

TODO: 提供github项目示例

混合使用Java和C++语言

这部分，写了个示例，见：https://github.com/MarshalW/OpenCvProto

使用上，用矩阵（Mat）的操作举例。

Java创建矩阵

创建矩阵，将Android api中的Bitmap转为矩阵：

```
1 Mat bitmapMat = new Mat();
2 Utils.bitmapToMat(bitmap, bitmapMat);
```

如何将Java创建矩阵对象传递到C++

将矩阵对象传递给C++代码，实际上Mat就是C++在内存中生成的对象，Java只需传递该对象的地址就行了。

```
1 long address=bitmapMat.getNativeObjAddr();
```

然后Java本地方法类似这样：

```
1 private native long generateHistogram(long bitmap);
```

那么通过javah生成的头文件中方法类似这样：

```
1 JNIEXPORT void JNICALL Java_marshal_opencvproto_Detector_detect(JNIEnv *env,
2                 jobject thiz, jlong matPtr);
```

在C++代码中这样生成Mat对象：

```
1 Mat *bitmpaMat = (Mat*) bitmapMatPtr;ddd
```

如何将C++的矩阵对象传递给Java

要根据生命周期来分析，可以有两种方式：

1. 可在Java中创建，然后传递到C++中做处理，不需要作为返回值，因为引用的是同一个对象
2. 在C++中创建，然后作为方法的返回值

对于后者，在C++中只需要做一次强制转型就可将矩阵对象指针传递回Java：

```
1 Mat *hist = new Mat();
2
```

```
3 // ...
4
5 return (jlong) hist;
```

回传到Java需要做的处理：

```
1 long address = generateHistogram(bitmapMat.nativeObj);
2 Mat histogram = new Mat(address);
```

即，直接用地址的long值创建矩阵对象即可。

C++和Java混合编程需要注意的问题

环境搭建

环境搭建中的步骤类似官方文档中static initialization的步骤。

要使用文档中说的有JNI部分的方式，而不是简单的复制本地库到libs目录下。所以重点是配置Android.mk文件，以下是我的：

```
1  LOCAL_PATH := $(call my-dir)
2
3  include $(CLEAR_VARS)
4
5  OPENCV_INSTALL_MODULES:=on
6
7  include /opt/OpenCV-2.4.6-android-sdk/sdk/native/jni/OpenCV.mk
8
9  LOCAL_MODULE    := Detector
10 LOCAL_SRC_FILES := Detector.cpp
11
12 LOCAL_LDLIBS :=-llog
13
14 include $(BUILD_SHARED_LIBRARY)
```

如何加载OpenCV库

加载OpenCV库，如果按照官方文档，也就是这样：

```
1 OpenCVLoader.initDebug();
```

会有一个error日志，不过不影响使用：

```
1 OpenCV error: Cannot load info library for OpenCV
```

查了一下OpenCV源代码，是加载opencv_info.so出错造成的，我没来得及看如何在Android.mk中设置将它加进来。

但可以直接这样直白的加载OpenCV库：

```
1 system.loadLibrary("opencv_java");
```

就没有问题了。

使用OpenCVLoader.initDebug()加载的好处是，日志会显示很详细的OpenCV加载信息，便于你排查问题。

未解决的问题

目前有一个，就是怎样在Java中保存矩阵对象数据，以后还能根据这个数据恢复矩阵对象。

OpenCV提供了写入文件的办法，使用YAML格式，但是在Android环境下，未提供相应的wrapper API。

目前我有个临时的解决办法，见这里怎样在Android下保存OpenCV矩阵。

android, opencv

搜索

微信公众号

标签

- AFNetworking1
- Android1
- BLE3
- BlueTooth3
- Browserify1
- CSS32
- Docker5
- EasyRTC1
- Flux1
- HTML54
- Hexo2
- Mac1
- OHHTTPStubs1
- RSA1
- React5
- SDWebimage1
- Swift4
- Translation1
- Ubuntu Server1
- Unity2
- WeChat1
- WebRTC2
- android2
- arp1

- bower1
- canjs1
- crypto1
- docker1
- gulp2
- iOS7
- momentjs1
- mqtt1
- nodejs3
- numpy1
- opencv2
- python1
- qiniu1
- qrcode1
- requirejs1
- rsync1
- sfnttool1
- stats1