HowTo_OpenCV_Native(C++)

In my previous work, I'm trying to use **OpenCV for Android or Java**, which is a big topic when configuring in Android Studio. One of the reason is that the configuration of studio is not so friendly as Eclipse(or maybe we're not familiar with it), it requires some coding of configuration for the gradle script; Another reason is that I want to use OpenCV as large as possible, which means that **OpenCV4Android** is not a good choice, it only contains part of android's processing and CV Class. To the best of my survey and knowledge, the complete version can be achieved by utilizing the module building in Android studio. I download the OpenCV sdk, then extract the java folder, import these files as a module into our current project. During this step, there should be some errors due to lack of definition of how to build it. Then I copy the build.gradle and modified it in opencv module, sync the project, which will give some error clues like APILevel, dependence, path, etc… After fixing them, remember to add it as a dependence of the application, which will helps the building the application.

In the above description, I just outlined the steps for integrating the OpenCV for Java, but it's not totally separated from Android. As I find that OpenCV4Android has done similar things but only limited to some CVLoader class or so, we have also import some core library like imgproc, our integrated part also contains the **android** folder for UI/operation, In this way we can utilize opencv using Java under Android circumstances.

However, I also tried to make the ndk work in Android studio, that will make us capable of directly compiling the C++ code and generating the necessary *.so/.a libarary, then in Java file we define some combined interface to call these C++ function and use them.

At the beginning of our project, this is our plan of quickly implementing the code. The steps here are as follows(partially refer to previous document **Android_Studio_with_OpenCV_and_NDK_support.pdf**, because it contains the plugin and some other setting, we depend on the them):

1) Create the prototype of how to call using the JNI-style:

```
JNIEXPORT jstring JNICALL Java_com_android_hacks_ndkdemo_MyActivity_hello
 (JNIEnv *, jobject);
```

2) Add ndk module which is our coded C++ native part:

```
  ndk{
        moduleName "hello"
    }
```

3) Define our own Android.mk and Application.mk, which includes the rules for how to build native code and its dependence(like OpenCV.mk, which will add link and source file):

```
LOCAL_PATH := $(call my-dir)
```

```
include $(CLEAR_VARS)

LOCAL_SRC_FILES := main.c
LOCAL_LDLIBS += -llog
LOCAL_MODULE := hello

include $(BUILD_SHARED_LIBRARY)
```

Applicatioin.mk:

```
APP_ABI := armeabi
APP_PLATFORM := android-16
```

4) Modify the build.gradle, add function for starting building using the above Android.mk, instead of using its own generated missing part.

```
// add begin
    sourceSets.main.jni.srcDirs = []

    task ndkBuild(type: Exec, description: 'Compile JNI source via NDK') {
        ndkDir =
project.plugins.findPlugin('com.android.application').getNdkFolder()
        commandLine "$ndkDir/ndk-build",
                'NDK_PROJECT_PATH=build/intermediates/ndk',
                'NDK_LIBS_OUT=src/main/jniLibs',
                'APP_BUILD_SCRIPT=src/main/jni/Android.mk',
                'NDK_APPLICATION_MK=src/main/jni/Application.mk'
    }

    tasks.withType(JavaCompile) {
        compileTask -> compileTask.dependsOn ndkBuild
    }
    // add end
```

5) Build, fix the bugs according to the information(mainly because of the sdk version, path, gradle definition).

In the above tutorial, I find that not all of them are correct due to the changes of syntax, especially the model definition (`    sourceSets.main.jni.srcDirs = []`), instead I tried a few times and find this one:

```
  // add begin
  android.sources {
      main {
          jni {
              source {
                  srcDirs = []
              }
          }
      }
  }
  // add ends
```

Many more details about gradle and setting can be find here(also issues):
https://github.com/deercoder/OpenCV-Demo

Some errors are quite dependant on the environment and version, remember to Google!!!(Not all as I recalled and record here!)

Comments: Even though above way can make sure that we use the OpenCV native libs, but for some reason, it still cannot make the opencv header passed with complex path(like opencv2/opencv.hpp, but can be OK opencv/core/core.hpp), I find that it's very time consuming if we locate each cv class's header file and add the #include, and also the header file has dependence or #include relationship(like in A.hpp, it has #include<B.hpp>), that situation make it very complex for compiling our native when we called libraries that has strong dependence...

Luckily, I have switched the way and tried the OpenCV for java in android studio, rewrite the implementation using Java with correct type and class, it seems good and reliable though. See here: https://github.com/deercoder/OpenCV-Demo/tree/master/backup1

Reference

(*)[1]
http://hujiaweibujidao.github.io/blog/2014/10/22/android-ndk-and-opencv-development-with-android-studio/
http://stackoverflow.com/questions/17767557/how-to-use-opencv-in-android-studio-using-gradle-build-tool/27356635#27356635
http://stackoverflow.com/questions/31507891/android-studio-1-3-gradle-plugin-returns-error-when-defining-jni-and-jnilibs-in
http://blog.gaku.net/ndk/

[1] This is the main reference for native C++, but it contains some error due to studio version/experimental plugin

http://stackoverflow.com/questions/31507891/android-studio-1-3-gradle-plugin-returns-error-when-defining-jni-and-jnilibs-in
https://codelabs.developers.google.com/codelabs/android-studio-jni/index.html?index=..%2F..%2Findex#3