2018.10.30

基于Android人脸识别

孙浩翔 夏启林

版本信息

|  |  |  |
| --- | --- | --- |
| 版本 | 日期 | 更改 |
| V1.0 | 2018年10月30日 | 创建文档 |

# 介绍

基于Android的脸部识别程序。经过C++验证的脸部识别算法，和杭州艾芯智能AXON M2官方提供的安卓平台相结合。实现基于深度图像的人脸识别。并实时显示检测结果。

实现目标：1.C++程序移植到android平台，生成.apk文件；

2.算法处理效果图形以及界面布局显示。

# 环境

操作系统：Windows 10

安卓SDk：最低版本 Android 4.0

目标版本 Android 4.3 （安卓平台NanoPC-T4 支持Android7.0）

编写工具：Android Studio 3.1.4

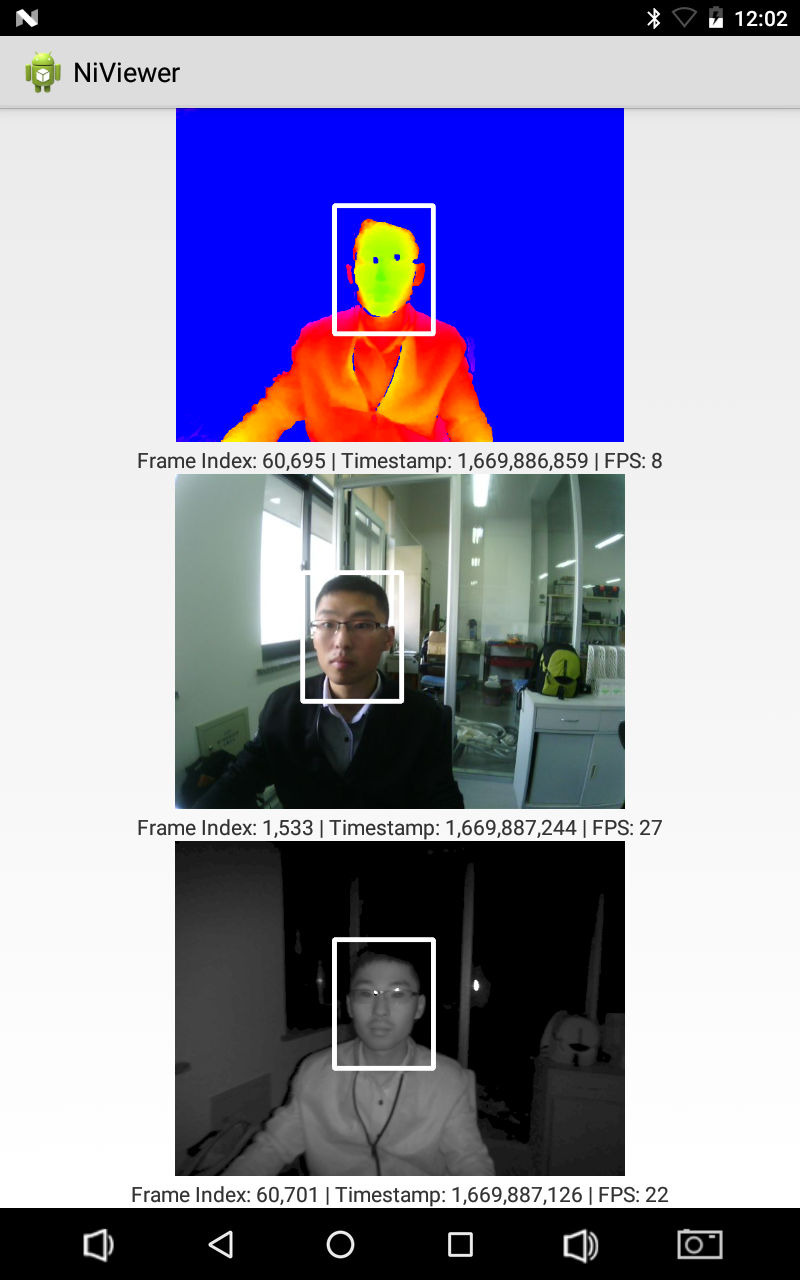
JRE版本: 1.8.0\_152-release-1024-b02 amd64

OpenCV： 2.4.3

OpenNI2

编译构建工具：Gradle 3.1.4 （需要联网下载）

# 软件界面



说明：1.通过USB连接相机，HDMI外接显示器以及USB鼠标，安装APP；

2.如上图，三张显示图片，第一张是相机实时采集的深度图像经过算法处理后的显示效果图，第二张是对应的RGB图，第三张是红外图；

# 文件结构

Project 工程文件夹显示下 对应文件结构；

app

├ build

├ src

│ └ main

│ ├ assets

│ ├ cpp

│ │ ├ include // CPP文件包含目录

│ │ │ （opencv库及外部头文件）

│ │ └ native-lib.cpp // CPP文件

│ ├ java

│ │ └ org.openni

│ │ ├ andoird

│ │ │ ├ tools.niviewer

│ │ │ │ ├ DeviceSelectDialog

│ │ │ │ ├ NiViewerActivity // 启动Activity（入口）

│ │ │ │ └ StreamView // 数据流处理类

│ │ │ ├ OpenNIHelper

│ │ │ └ OpenNIView

│ │ ├ CamParam

│ │ ├ …

│ │ └ VideoStream

│ ├ jni

│ ├ jniLibs

│ │ ├ armeabi-v71 // OpenCV、OpenNI的so文件

│ │ └ libs

│ ├ res

│ │ ├ drawable-hdpi

│ │ ├ drawable-mdpi

│ │ ├ drawable-xhdpi

│ │ ├ drawable-xxhdpi

│ │ ├ layout

│ │ │ ├ activity\_niviewer.xml // 主窗口的布局文件

│ │ │ └ stream\_view.xml // 数据流处理布局文件

│ │ ├ menu

│ │ └ values

│ └ AndroidManifest.xml // 启动配置文件

├ build.gradle // gradle脚本

└ CMakeLists.txt // CMakeLists文件

# 运行流程

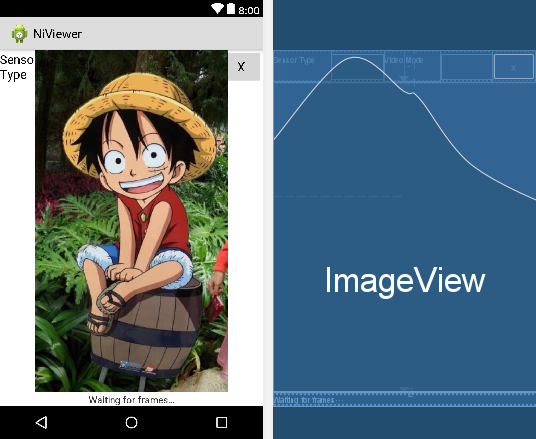
## Java部分



1. 程序启动，根据AndroidManifest.xml配置，首先激活NiViewerActivity类，进行界面初始化。初始化界面只有一个水平布局控件，后续会调用StreamView类添加图像显示内容。初始化OpenNIHelper类。
2. 实现OpenNIHelper.DeviceOpenListener的onDeviceOpened接口。当应用从后台切换到前台显示，外部设备打开的时候会调用监听器的onDeviceOpened函数，将显示的StreamView控件的设备修改为打开的设备。
3. 如果没有StreamView控件存在，调用addStream函数创建StreamView类，并添加到主显示窗口中。StreamView类主要负责数据流的读取、解析和显示。初始化StreamView类的时候，依次调用setDevice函数设置控件显示的图片类型（Depth，Color，IR）。
4. 对设备的选择触发监听器调用onSensorSelected函数，根据传入的序号通知VideoStream类读取不同类型的图片数据流。根据选择的不同类型，更新视频模式下拉框，并默认选择第一个。
5. 选择视频模式会触发监听器调用onVideoModeSelected函数。设置VideoStream类视频模式，然后开启子线程。
6. 新线程循环读取视频数据流，经过格式转换后交给C++程序进行人脸识别并在图片中标识识别结果，经过格式转换后交给Java，在界面中展示识别结果。
7. Android中相关的view和控件不是线程安全的。通过Handler传递消息，实现异步UI更新。

注意：

VideoStream类的窗口布局包含了一个水平布局的control\_panel控件，包含两个TextView，两个spinner和一个Button控件。显示的时候将其高度设置为1px，目的是为了隐藏功能，同时可以保证原有的程序框架不做大的改动。





## C++部分

略

# 配置过程

## NDK配置

NDK 版本18.1.5063045

JNI：是Java Native Interface的缩写，它提供了若干的API实现了Java和其他语言的通信（主要是C&C++）。JNI是一套编程接口，用来实现Java代码与本地的C/C++代码进行交互。

NDK: NDK是Google开发的一套开发和编译工具集，可以生成动态链接库，主要用于Android的JNI开发。

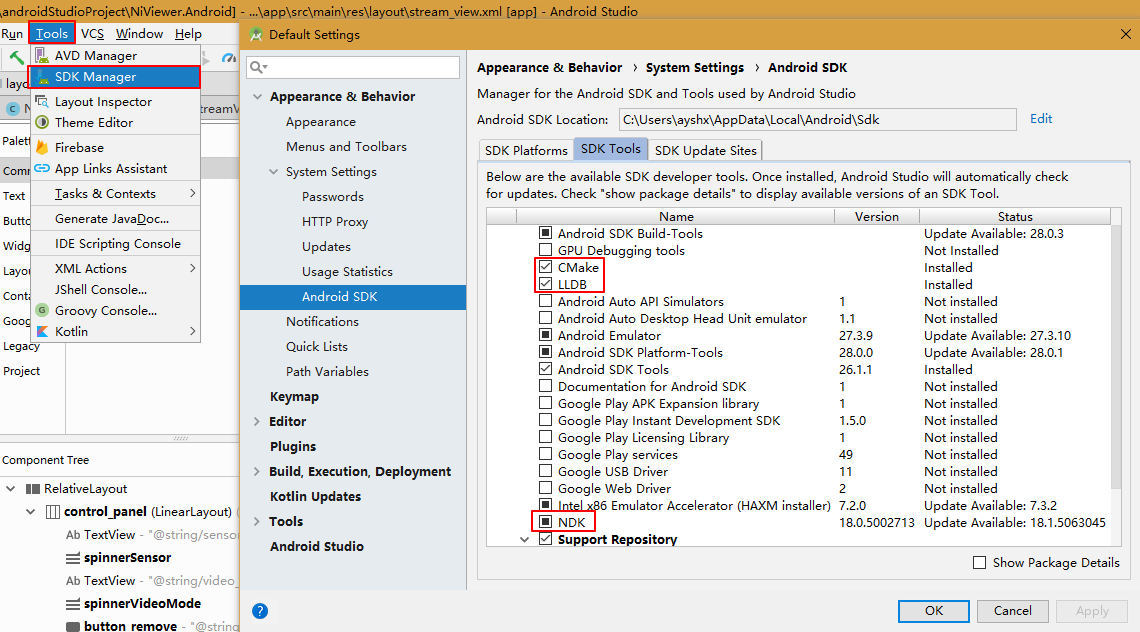
要为应用编译和调试原生代码，需要安装以下组件：

Android原生开发工具包 (NDK)：这套工具集允许我们为Android使用C和C++代码，且其提供众多平台库让我们可以管理原生Activity和访问物理设备组件，例如传感器和触摸输入。

CMake：一款外部构建工具，可与Gradle搭配使用来构建原生库。如果只计划使用ndk-build（需要编写Android.mk文件和Application.mk文件），则不需要此组件，推荐使用Cmake,编写CMakLists.txt.

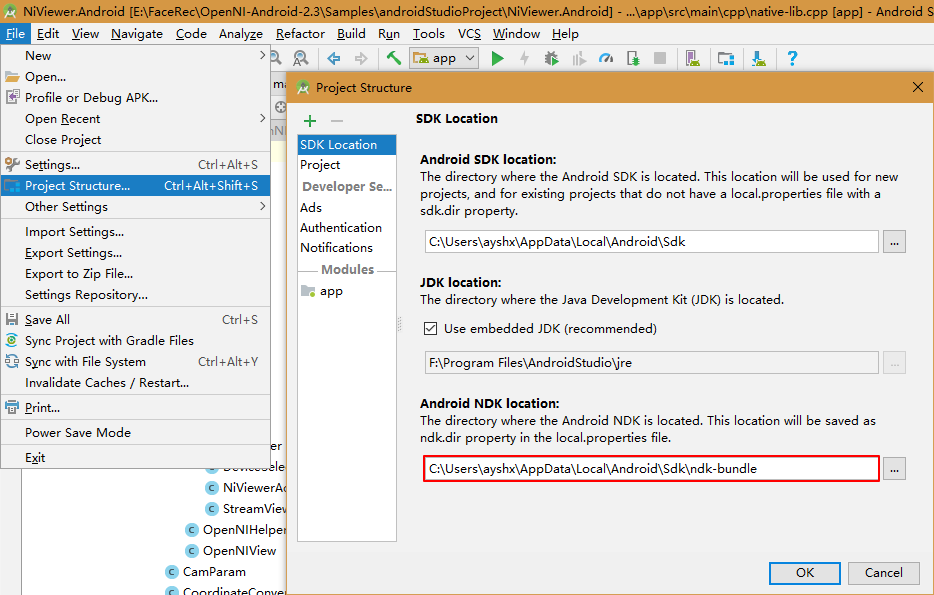
LLDB：一种调试程序，Android Studio使用它来调试原生代码。

Android Studio内提供了安装选项，tools->SDK Manager->Android SDK->SDK Tools，勾选CMake、LLDB和NDK后点击确定自动安装，根据网络环境大概需要十到二十分钟。



完成后点击“Finish”即可。

可以在工程配置中看到已经配置好的NDK目录。

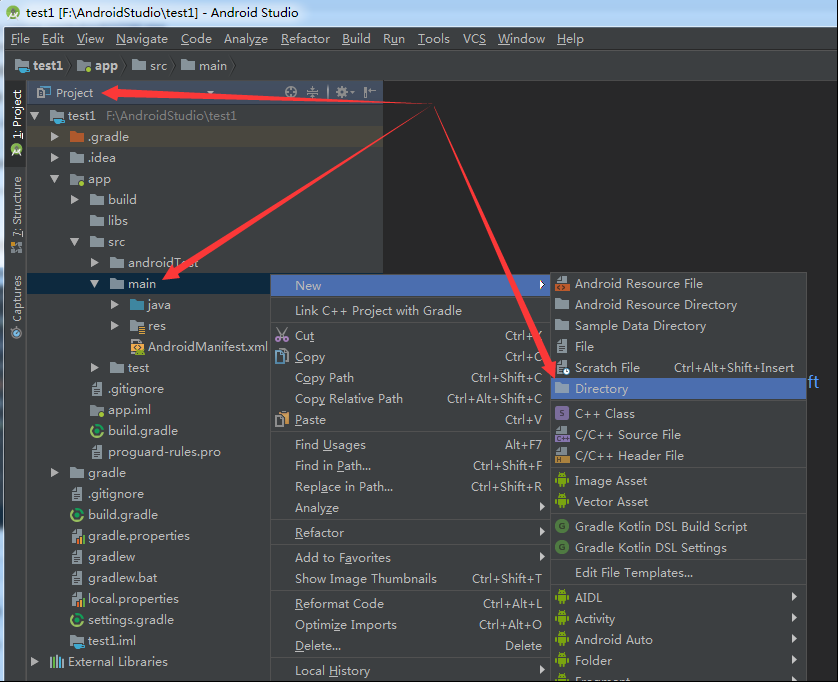


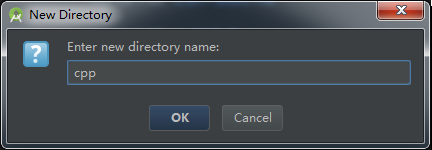
## 加入C++支持

### 创建CPP文件

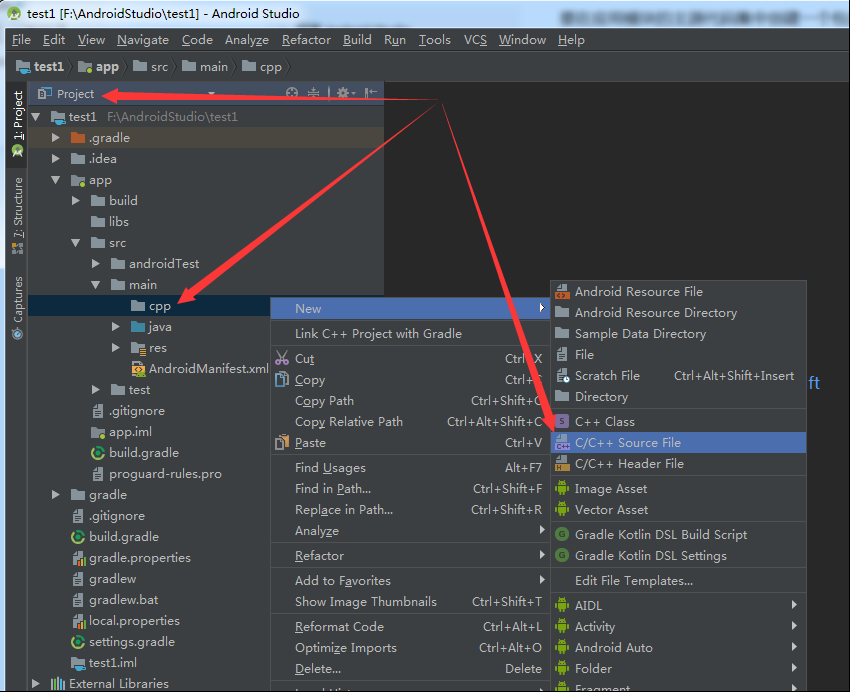
相机官方提供了一个Android工程，现对工程进行修改使其支持C++代码的调用。

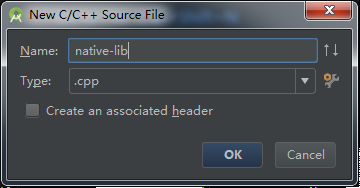
在src目录下创建一个新的文件夹cpp。



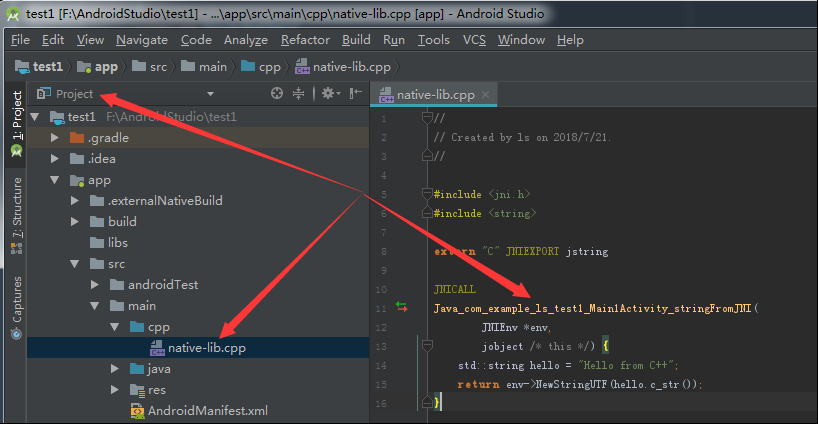


新建一个CPP文件native-lib.cpp。写一个测试函数。



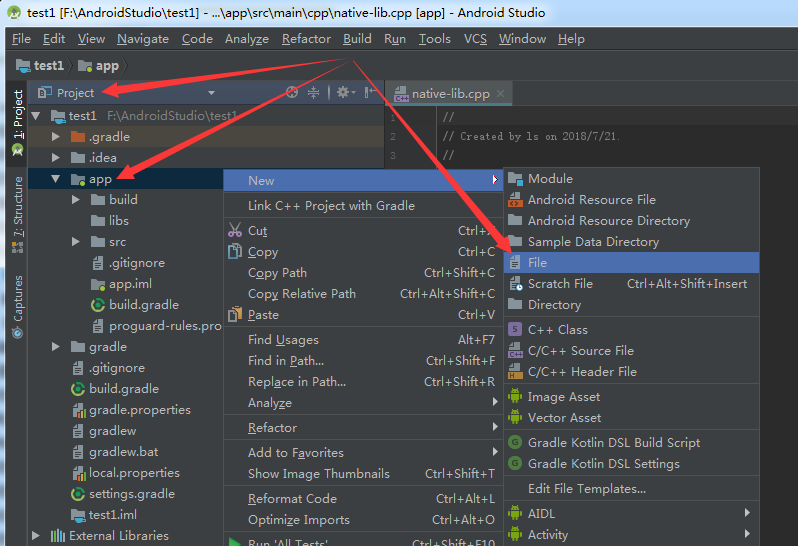


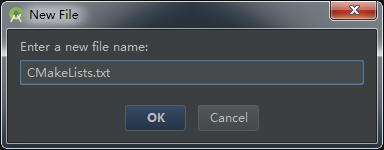
测试函数需要注意函数命名规则，Java\_包名\_类名\_函数名(JNIEnv \*env, jobject[, 其他参数])。



|  |
| --- |
| #include **<jni.h>** #include **<string>  using namespace** std;  **extern "C" JNIEXPORT** jstring  **JNICALL** Java\_org\_openni\_android\_tools\_niviewer\_NiViewerActivity\_stringFromJNI(  JNIEnv \*env,  jobject */\* this \*/*) {   std::string hello = **"Hello from C++"**;  **return** env->NewStringUTF(hello.c\_str());  } |

在app目录下创建CMakeLists.txt文件





### 创建CMake构建脚本

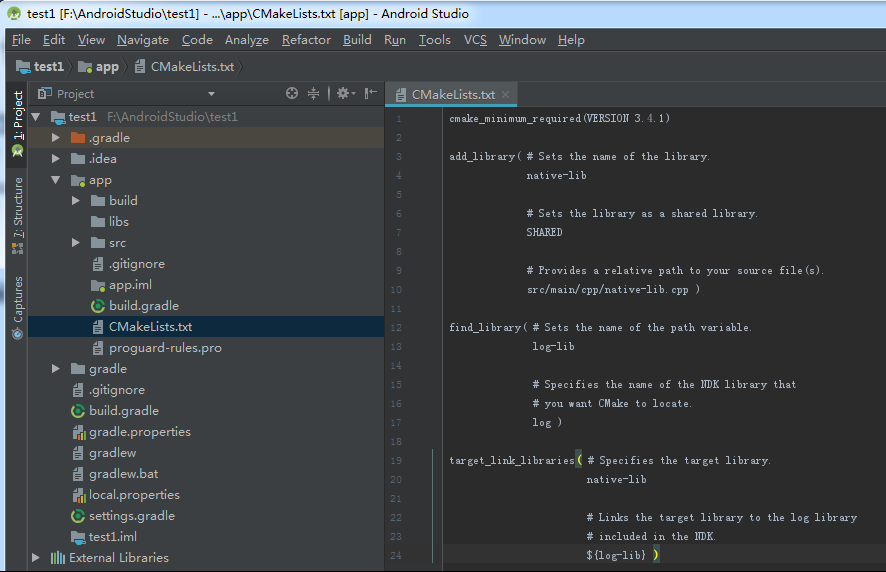
CMake是一种跨平台编译工具，比make更为高级，使用起来要方便得多。CMake主要是编写CMakeLists.txt文件，然后用cmake命令将CMakeLists.txt文件转化为make所需要的makefile文件，最后用make命令编译源码生成可执行程序或共享库。

Android Studio采用CMake脚本语法配置C编译器的环境，如果之前有过使用CMAKE的经验，或许这并非难题，但对于初学者而言，CMAKE的脚本语法，还是略过于生涩，而且AS对该文件的配置并不友好，居然没有代码提示，于是不得不查很多文档。但好在，NDK的开发大多不是大型的C++项目，也不太需要过于复杂的设置。

这里CMakeLists文件的目的就是告诉编译器需要编译的文件，文件和类之间的依赖关系，需要引入的动态库、头文件位置等编译信息。

写入以下内容：

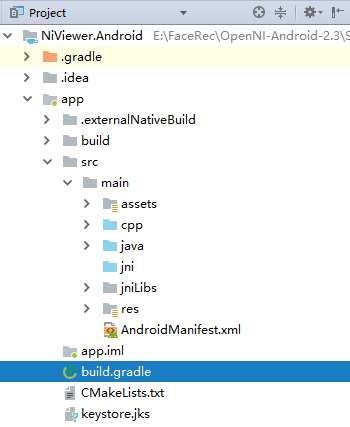
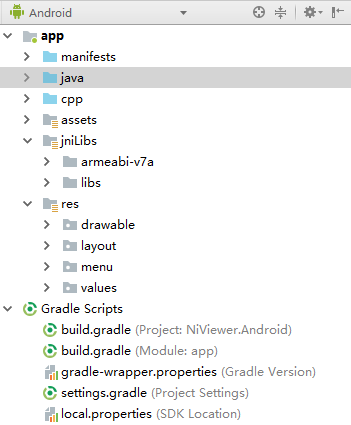
|  |
| --- |
| #规定CMake版本  cmake\_minimum\_required(VERSION 3.4.1)  add\_library( # Sets the name of the library.  native-lib  # Sets the library as a shared library.  SHARED  # Provides a relative path to your source file(s).  src/main/cpp/native-lib.cpp )  find\_library( # Sets the name of the path variable.  log-lib  # Specifies the name of the NDK library that  # you want CMake to locate.  log )  target\_link\_libraries( # Specifies the target library.  native-lib  # Links the target library to the log library  # included in the NDK.  ${log-lib} ) |



### 向gradle注册构建请求

简单的说，Gradle是一个构建工具，它是用来帮助我们构建app的，构建包括编译、打包等过程。我们可以为Gradle指定构建规则，然后它就会根据我们的“命令”自动为我们构建app。Android Studio中默认就使用Gradle来完成应用的构建。

Project是显示文件夹下所有文件，正确配置Gradle文件后，Android下会显示所有跟项目相关的文件结构和文件。

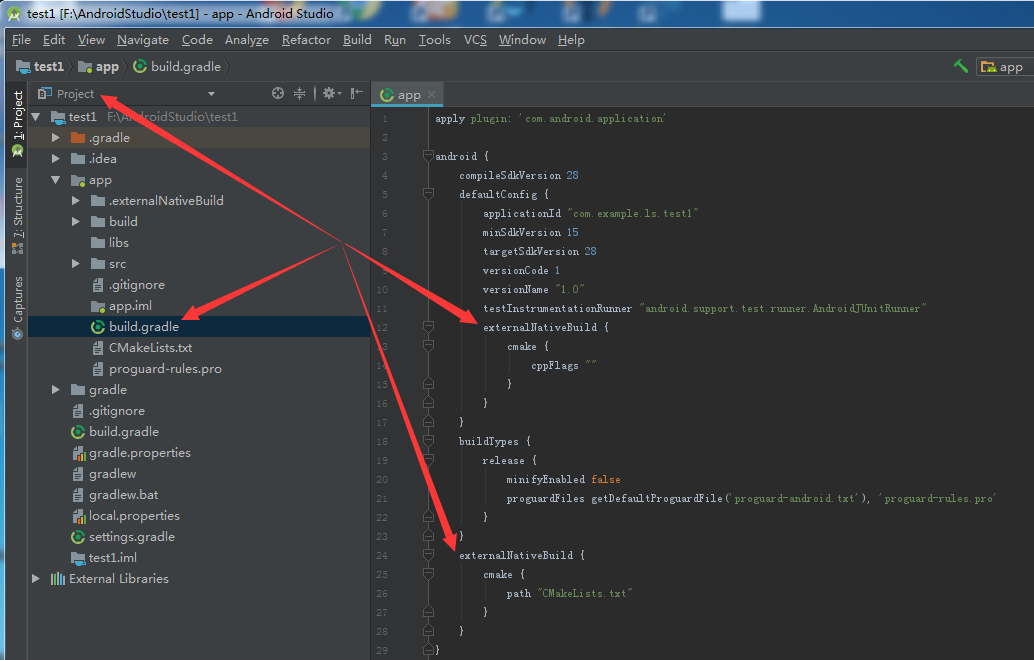
 

打开build.gradle向android/defaultConfig节区追加以下内容：

|  |
| --- |
| externalNativeBuild {  cmake {  cppFlags **"-std=c++11 -frtti -fexceptions"** abiFilters **'armeabi-v7a'** } } |

向android节区追加以下内容：

|  |
| --- |
| externalNativeBuild{  cmake{  path **"CMakeLists.txt"** } } |



### 测试

在NiViewerActivity类中加入库：

|  |
| --- |
| **static** {  System.*loadLibrary*(**"native-lib"**); } |

加入函数声明：

|  |
| --- |
| **public native** String stringFromJNI( ); |

程序中调用即可：

|  |
| --- |
| Toast.*makeText*(**this**,stringFromJNI(),Toast.***LENGTH\_LONG***).show(); |

简单的完整程序如下：

|  |
| --- |
| package com.example.myapplication;    import android.support.v7.app.AppCompatActivity;  import android.os.Bundle;  import android.widget.TextView;    public class MainActivity extends AppCompatActivity {    // Used to load the 'native-lib' library on application startup.  static {  System.loadLibrary("native-lib");  }    @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.activity\_main);    // Example of a call to a native method  TextView tv = (TextView) findViewById(R.id.sample\_text);  tv.setText(stringFromJNI());  }    /\*\*  \* A native method that is implemented by the 'native-lib' native library,  \* which is packaged with this application.  \*/  public native String stringFromJNI();  } |

运行后可以看到Toast消息框中显示**Hello from C++**。

## 加入OpenCV的支持

### 下载相关文件

OpenCV可以直接在官网上下，官方网址为：<https://opencv.org/releases.html>。选择Android pack版本即可，大约310MB。

ABOUT 
OpenCV 
Releases 
3.4.1 
2018-02-27 
3.4.1 
NEWS 
RELEASES 
PLATFORMS 
BOOI 
Documentation 
[D Sources 
Win pack 
i iOS pack 
Android pack 

### OpenCV相关文件

• include文件

在下载好的OpenCV压缩包中，打开路径下的.\opencv-2.4.3-android-sdk\OpenCV-android-sdk\sdk\native\jni 有一个include文件夹，把这个文件夹复制粘贴至我们的OpenCVTest项目中，路径为src/main/cpp

• jni文件

然后是动态库(.so文件)，打开路径下的.\opencv-2.4.3-android-sdk\OpenCV-android-sdk\sdk\native\libs\ armeabi-v7a文件夹，这个文件夹里面是arm版本的so文件。复制粘贴到我们的项目中，路径为src/main/jniLibs/ armeabi-v7a。和OpenNI的so文件放在一起。

路径是可以自己选择的，只是习惯上这样按类别放。

注意：由于c++部分调用的opencv版本是2.4.3，故建议保持一致性

### 配置文件

修改app文件夹下的build.gradle文件

|  |
| --- |
| apply **plugin**: **'com.android.application'** android {   compileSdkVersion 28  buildToolsVersion **"28.0.1"** defaultConfig {  applicationId **"org.openni.android.tools.niviewer"** minSdkVersion 14  targetSdkVersion 28  externalNativeBuild {  cmake {  cppFlags **"-std=c++11 -frtti -fexceptions"** abiFilters **'armeabi-v7a'** }  }  }  sourceSets{  main{   jniLibs.srcDirs = [**'src/main/jniLibs'**]  *//jniLibs.srcDirs = ['src/main/jniLibs/libs']* }  }  splits {  abi {  enable **true** reset()  include **'armeabi-v7a'** universalApk **true** }  }   buildTypes {  release {  minifyEnabled **false** proguardFiles getDefaultProguardFile(**'proguard-android.txt'**), **'proguard-rules.pro'** }  }   externalNativeBuild{  cmake{  path **"CMakeLists.txt"** }  } }  dependencies {  implementation fileTree(**dir**: **'libs'**, **include**: [**'\*.jar'**])  implementation **'com.android.support:appcompat-v7:28.0.0'** implementation **'com.android.support.constraint:constraint-layout:1.1.3'** testImplementation **'junit:junit:4.12'** androidTestImplementation **'com.android.support.test:runner:1.0.2'** androidTestImplementation **'com.android.support.test.espresso:espresso-core:3.0.2'** } |

修改CMakeLists文件

|  |
| --- |
| # For more information about using CMake with Android Studio, read the # documentation: https://d.android.com/studio/projects/add-native-code.html  # Sets the minimum version of CMake required to build the native library.  cmake\_minimum\_required(VERSION 3.4.1)  # 设置include文件夹的地址 include\_directories(${CMAKE\_SOURCE\_DIR}/src/main/cpp/include) include\_directories(${CMAKE\_SOURCE\_DIR}/src/main/cpp)  # 设置opencv的动态库 add\_library(libopencv\_java SHARED IMPORTED) set\_target\_properties(libopencv\_java PROPERTIES IMPORTED\_LOCATION ${CMAKE\_SOURCE\_DIR}/src/main/jniLibs/libs/armeabi-v7a/libopencv\_java.so)   # Creates and names a library, sets it as either STATIC # or SHARED, and provides the relative paths to its source code. # You can define multiple libraries, and CMake builds them for you. # Gradle automatically packages shared libraries with your APK.  add\_library( # Sets the name of the library.  native-lib   # Sets the library as a shared library.  SHARED   # Provides a relative path to your source file(s).  src/main/cpp/native-lib.cpp )  # Searches for a specified prebuilt library and stores the path as a # variable. Because CMake includes system libraries in the search path by # default, you only need to specify the name of the public NDK library # you want to add. CMake verifies that the library exists before # completing its build.  find\_library( # Sets the name of the path variable.  log-lib   # Specifies the name of the NDK library that  # you want CMake to locate.  log )  # Specifies libraries CMake should link to your target library. You # can link multiple libraries, such as libraries you define in this # build script, prebuilt third-party libraries, or system libraries.  target\_link\_libraries( # Specifies the target library.  native-lib libopencv\_java   # Links the target library to the log library  # included in the NDK.  ${log-lib} ) |

### 测试

这里提供已经简单的程序，适用于空工程。针对人脸识别的调用见具体代码实现。

* **MainActivity.java文件**

|  |  |
| --- | --- |
| |  | | --- | | package com.example.videomedicine.opencvtest;    import android.graphics.Bitmap;  import android.graphics.BitmapFactory;  import android.support.v7.app.AppCompatActivity;  import android.os.Bundle;  import android.view.View;  import android.widget.Button;  import android.widget.ImageView;  public class MainActivity extends AppCompatActivity implements View.OnClickListener{  // Used to load the 'native-lib' library on application startup.  static {  System.loadLibrary("native-lib");  }  private Button btn\_1;  private Button btn\_2;  private ImageView imageView;  private Bitmap bitmap;  @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.activity\_main);  btn\_1 = (Button)findViewById(R.id.button\_1);  imageView = (ImageView)findViewById(R.id.image);  bitmap = BitmapFactory.decodeResource(getResources(),R.drawable.luffy);  imageView.setImageBitmap(bitmap);  btn\_1.setOnClickListener(this);  btn\_2 = (Button)findViewById(R.id.button\_2);  btn\_2.setOnClickListener(this);  }  public void showImage(){  bitmap = BitmapFactory.decodeResource(getResources(),R.drawable.luffy);  imageView.setImageBitmap(bitmap);  }  public void gray(){  int w = bitmap.getWidth();  int h = bitmap.getHeight();  int[] piexls = new int[w\*h];  bitmap.getPixels(piexls,0,w,0,0,w,h);  int[] resultData =Bitmap2Grey(piexls,w,h);  Bitmap resultImage = Bitmap.createBitmap(w,h, Bitmap.Config.ARGB\_8888);  resultImage.setPixels(resultData,0,w,0,0,w,h);  imageView.setImageBitmap(resultImage);  }  @Override  public void onClick(View view){  switch(view.getId()){  case R.id.button\_1:showImage();break;  case R.id.button\_2:gray();break;  }  }  /\*\*  \* A native method that is implemented by the 'native-lib' native library,  \* which is packaged with this application.  \*/  public native int[] Bitmap2Grey(int[] pixels,int w,int h);    @Override  public void onResume(){  super.onResume();  }  } | |

* **native-lib.cpp文件**

|  |
| --- |
| #include <jni.h>  #include<opencv2/opencv.hpp>  #include<iostream>  using namespace cv;  using namespace std;    extern "C" JNIEXPORT jintArray    JNICALL  Java\_com\_example\_videomedicine\_opencvtest\_MainActivity\_Bitmap2Grey(  JNIEnv \*env,  jobject /\* this \*/,jintArray buf,jint w,jint h) {  jint \*cbuf;  jboolean ptfalse = false;  cbuf = env->GetIntArrayElements(buf, &ptfalse);  if(cbuf == NULL){  return 0;  }    Mat imgData(h, w, CV\_8UC4, (unsigned char\*)cbuf);  // 注意，Android的Bitmap是ARGB四通道,而不是RGB三通道  cvtColor(imgData,imgData,CV\_BGRA2GRAY);  cvtColor(imgData,imgData,CV\_GRAY2BGRA);    int size=w \* h;  jintArray result = env->NewIntArray(size);  env->SetIntArrayRegion(result, 0, size, (jint\*)imgData.data);  env->ReleaseIntArrayElements(buf, cbuf, 0);  return result;  } |

* **activity\_main.xml**

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:app="http://schemas.android.com/apk/res-auto"  xmlns:tools="http://schemas.android.com/tools"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:orientation="vertical"  tools:context=".MainActivity">    <LinearLayout  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:layout\_weight="1"  android:orientation="vertical">    <ImageView  android:id="@+id/image"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:layout\_weight="1"  app:srcCompat="@drawable/luffy" />  </LinearLayout>    <LinearLayout  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:layout\_weight="1"  android:orientation="horizontal">    <Button  android:id="@+id/button\_2"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:layout\_weight="1"  android:text="灰度图" />    <Button  android:id="@+id/button\_1"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:layout\_weight="1"  android:text="色图" />  </LinearLayout>    </LinearLayout> |

# 参考

* Android NDK学习笔记：Android Studio3.1+CMAKE+OpenCV3.4配置

<https://blog.csdn.net/cv_jason/article/details/79758823>

* Android Studio向项目添加C/C++原生代码教程

<http://www.cnblogs.com/lsdb/p/9337285.html>

* Android Handler消息传递机制

<https://www.cnblogs.com/zhaohongtian/p/6801596.html>