由于OpenCV Android版，需要先装OpenCV Manager，对用户相当不友善，并且泄及大量的代码重写。所以就采用JNI接口，从JAVA调用C来处理。

1.环境搭建

1.1 Android开发环境搭建，相当简单。网上有大量的文章，我就不详细写。

我用的是

jdk-8u25-windows-i586.exe

adt-bundle-windows-x86-20131030.zip

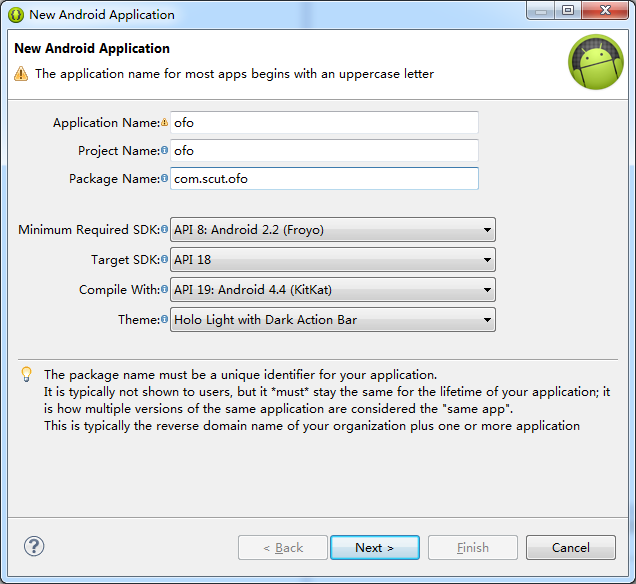
1.2 Android+JNI+OpenCV环境搭建

1.先下载并解压好

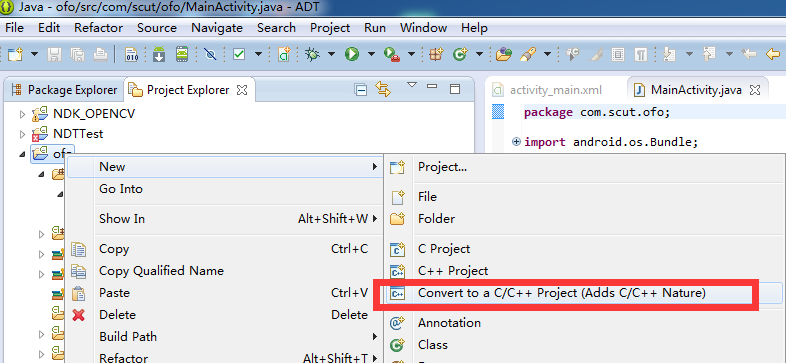
android-ndk-r10d-windows-x86.exe

OpenCV-3.1.0-android-sdk.zip

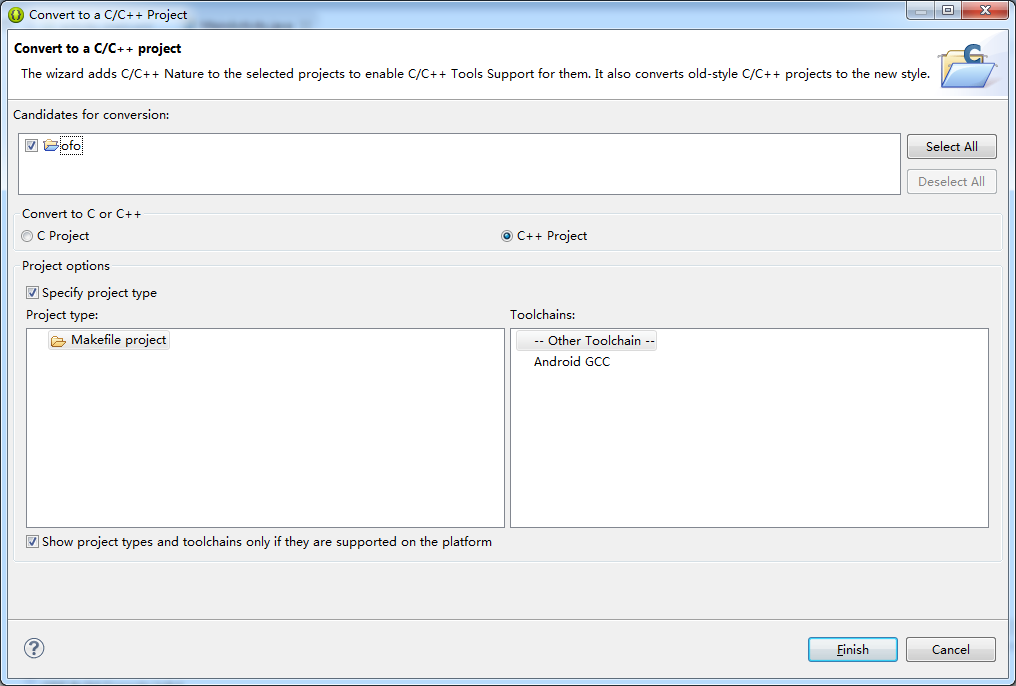
接着新建个Android项目



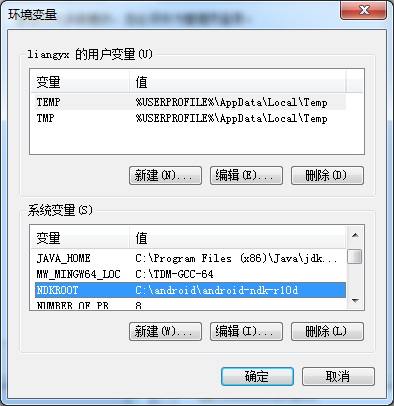
2.建好后，什么都不要操作，就要立即转成C++ Project，否则后面是选不到(太坑了！)



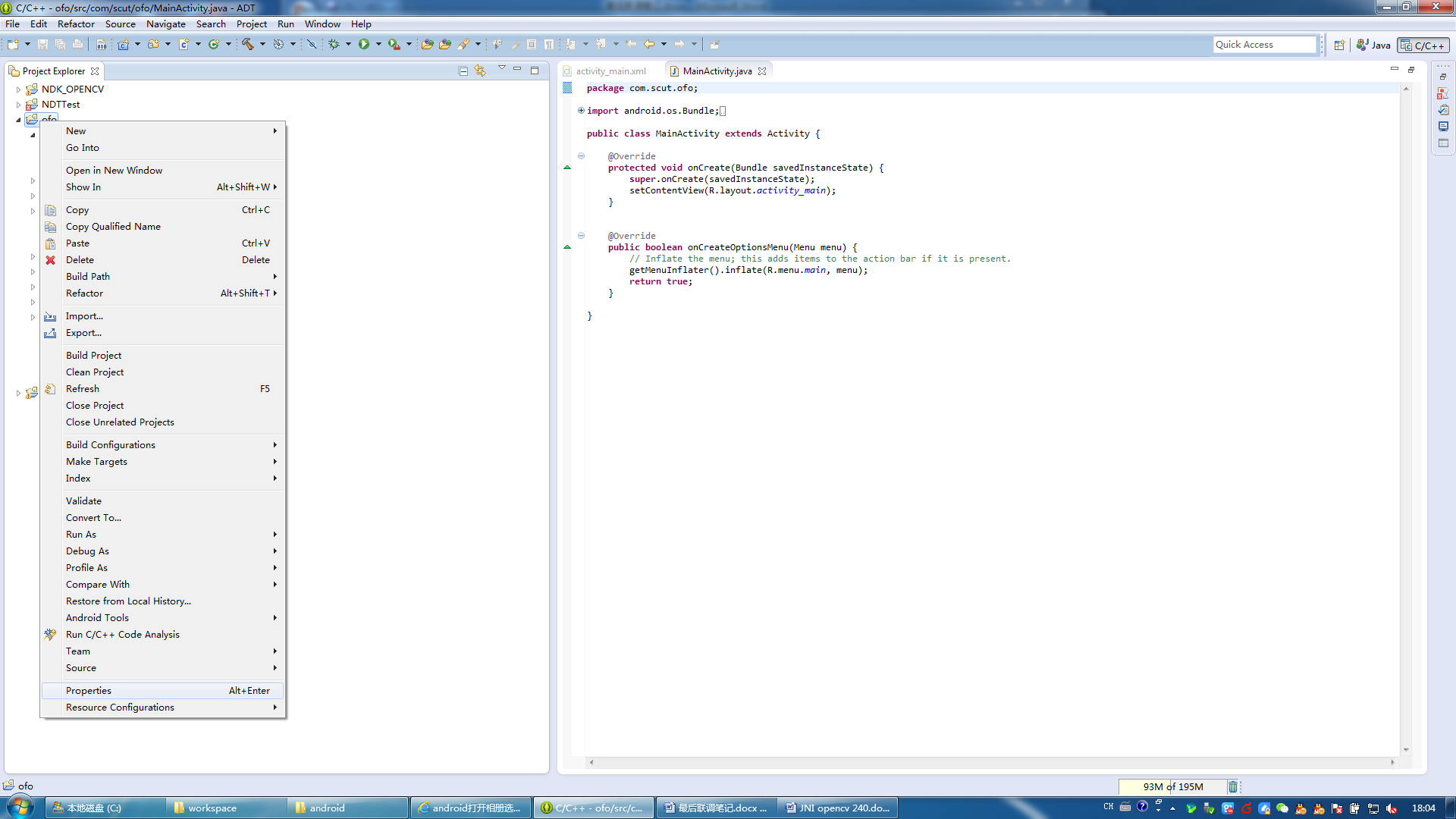
2.1见到这个可以转换时，就成功了10%



3.接下来在**windows环境变量**中设置NDK路径



4.1 回到eclipse，在项目属性中设好NDK相关信息

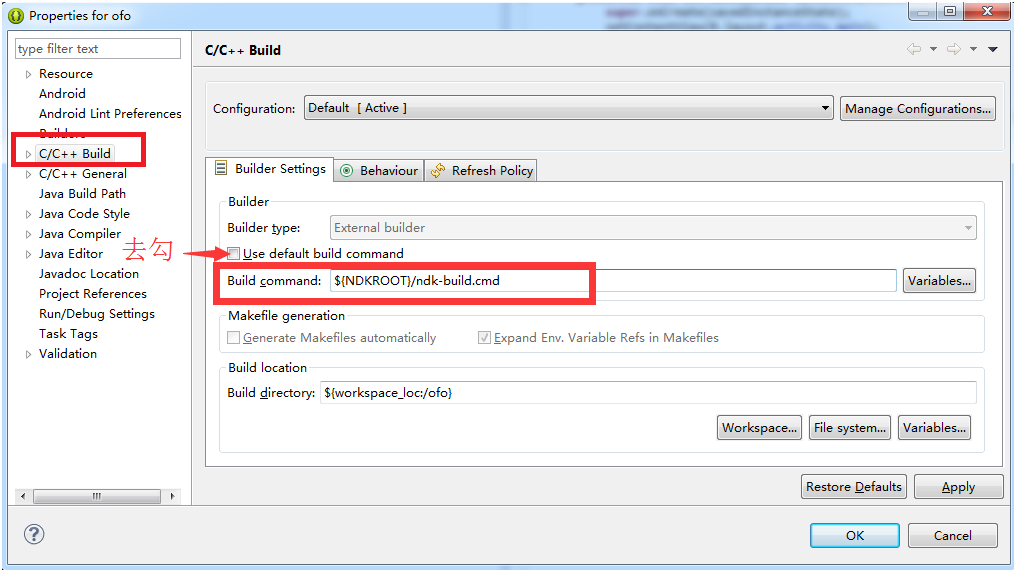


4.2然后在C/C++ Build中的Builder Settings中

将Use default build command：去勾

将Build command设为

|  |
| --- |
| ${NDKROOT}/ndk-build.cmd |

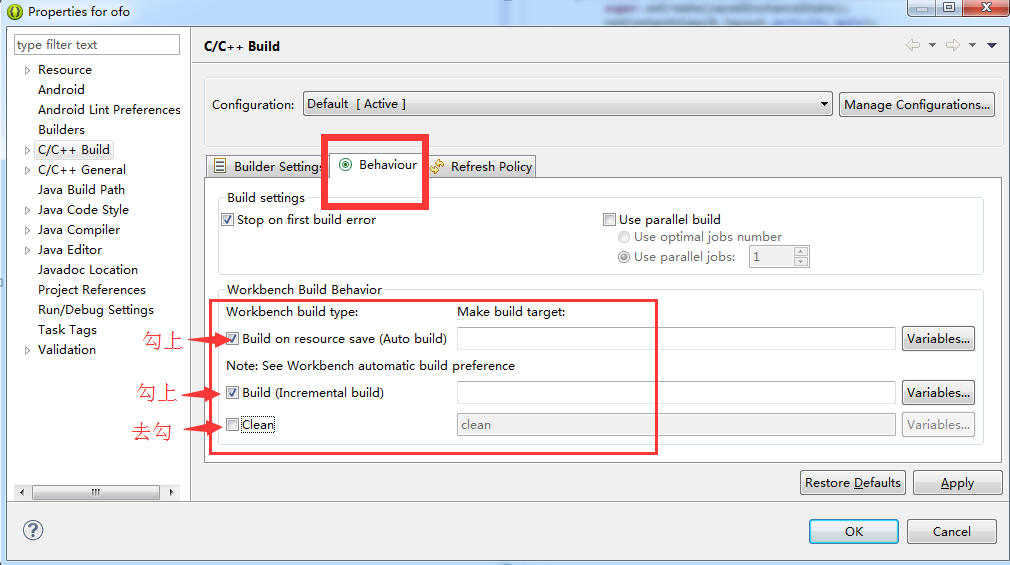


4.3 在Behaviour中

Build on resource save (Auto build)：勾上

Build (Incremental build)：勾上

Clean：去勾



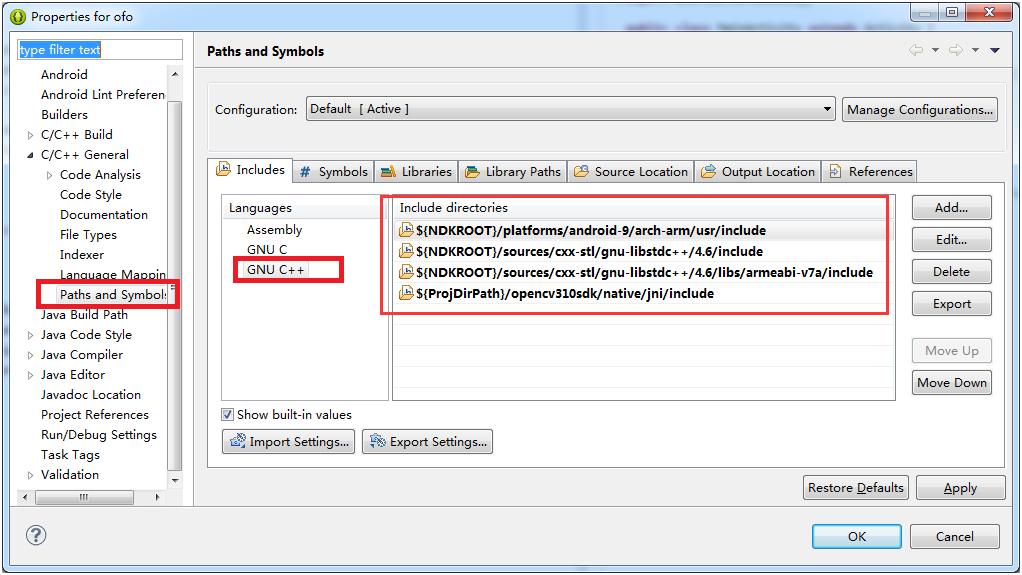
4.4 接着在C/C++ General→Paths and Symbols→GNU C++中设好

${NDKROOT}/platforms/android-9/arch-arm/usr/include

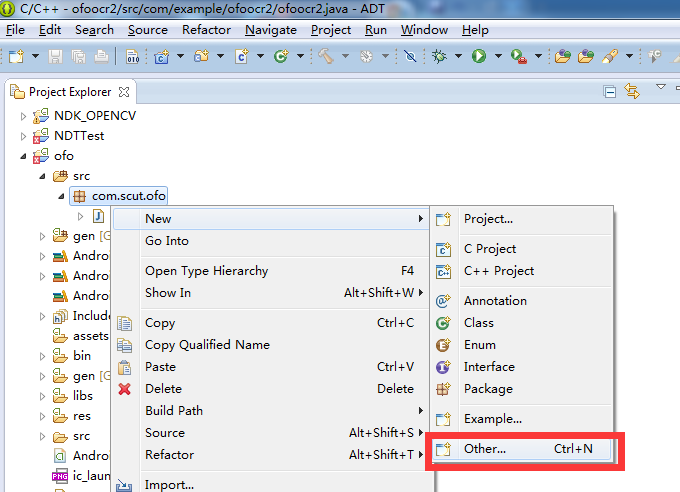
${NDKROOT}/sources/cxx-stl/gnu-libstdc++/4.6/include

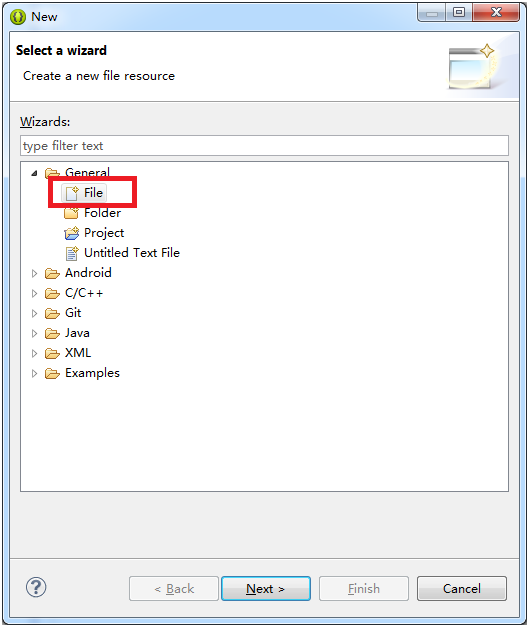
${NDKROOT}/sources/cxx-stl/gnu-libstdc++/4.6/libs/armeabi-v7a/include

${ProjDirPath}/opencv310sdk/native/jni/include

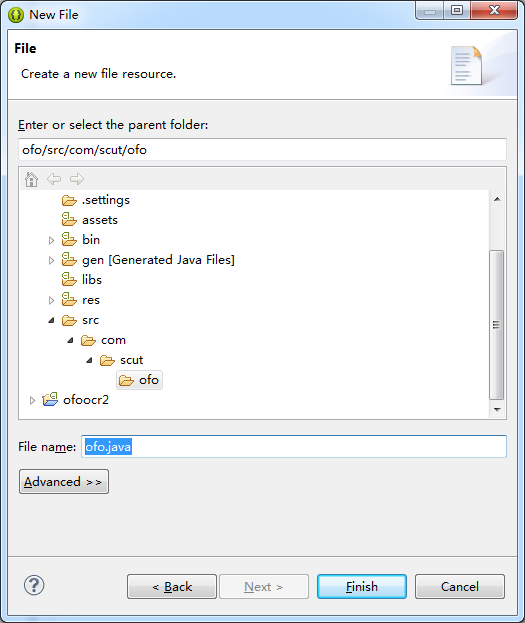


5.1 建一个JAVA文件，用于JAVA通过JNI交互

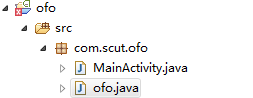




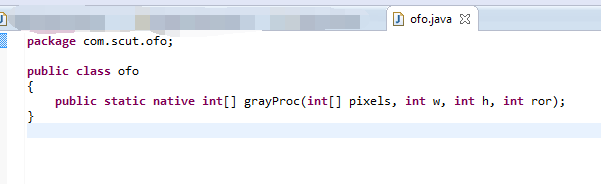
ofo.java



5.2 检查目录结构是否像这样

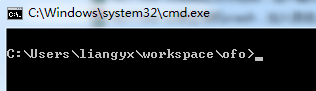


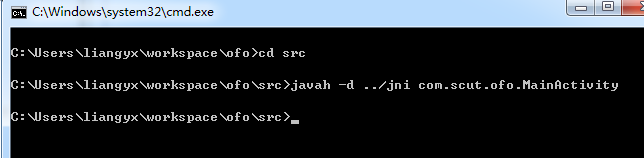
5.3 在ofo.java中输入交换的内容代码



|  |
| --- |
| **package** com.scut.ofo;  **public** **class** ofo  {  **public** **static** **native** **int**[] grayProc(**int**[] pixels, **int** w, **int** h, **int** ror);  } |

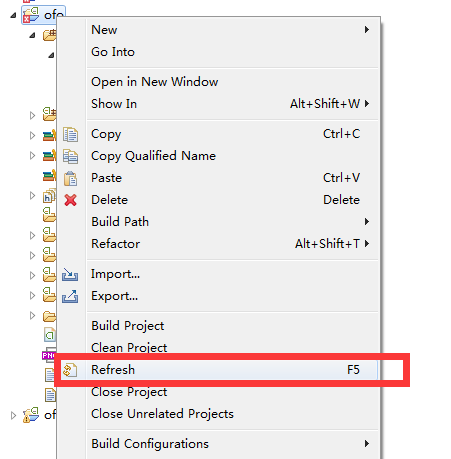
6.1 .在命令行中生成.h文件

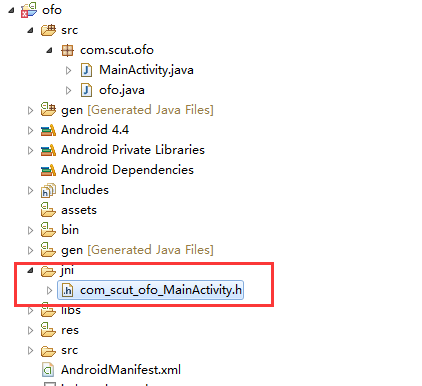




|  |
| --- |
| cd src  javah -d ../jni com.scut.ofo.MainActivity |

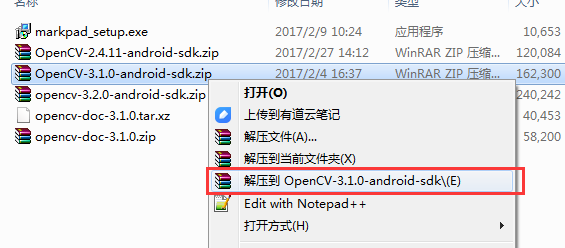
6.2在eclipse.exe中刷新一下，看看是否己经在项目中



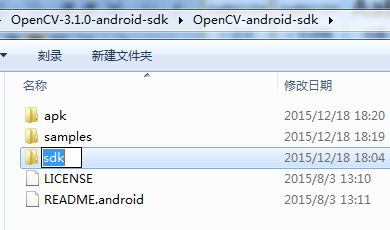


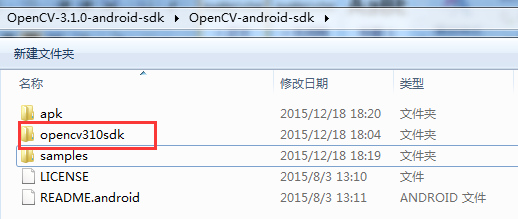
7.将OpenCV 3.1放进项目中

7.1先解压

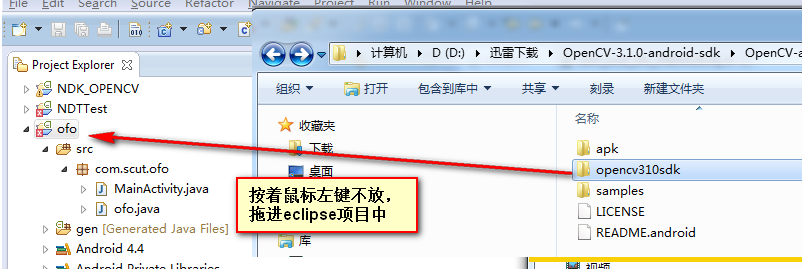


7.2将OpenCV-android-sdk/sdk目录重命名为opencv310sdk

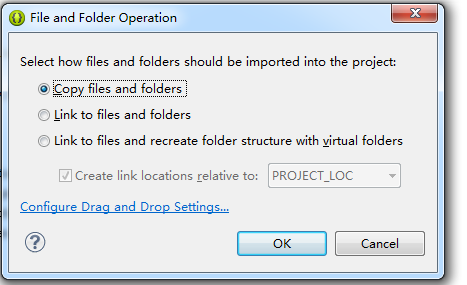




7.3然后拖进eclipse项目中



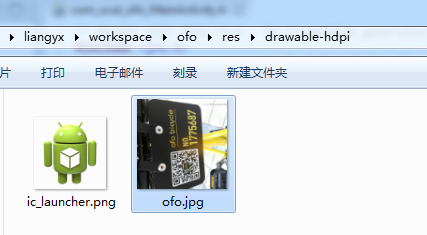
7.4这时会问你导入到项目，选Copy files and folders



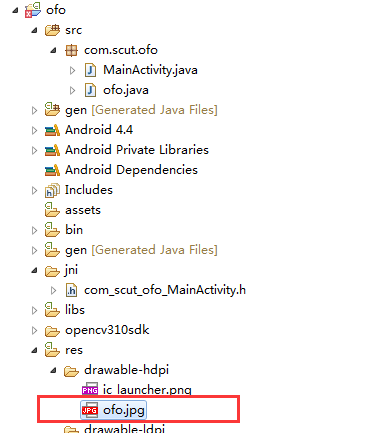
这样Android+JNI+OpenCV环境就准备好了。

接下来验证

将一张正方形旋转的ofo照片放进res/drawable-hdpi目录中



刷新一下



接着就帖代码

**注意Android版OpenCV在SVM阶段，要将Feature为0的值重新赋值为0.01，否则预测的值会全部相同。**

|  |
| --- |
| activity\_main.xml |
| <RelativeLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*  xmlns:tools=*"http://schemas.android.com/tools"*  android:layout\_width=*"match\_parent"*  android:layout\_height=*"match\_parent"*  android:paddingBottom=*"@dimen/activity\_vertical\_margin"*  android:paddingLeft=*"@dimen/activity\_horizontal\_margin"*  android:paddingRight=*"@dimen/activity\_horizontal\_margin"*  android:paddingTop=*"@dimen/activity\_vertical\_margin"*  tools:context=*".MainActivity"* >  <Button  android:id=*"@+id/btn\_gray\_process"*  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  android:text=*"识别7位ofo车牌"* />  <ImageView  android:id=*"@+id/image\_view"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:layout\_alignLeft=*"@+id/btn\_gray\_process"*  android:layout\_below=*"@+id/btn\_gray\_process"* />    <TextView  android:id=*"@+id/text\_view"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:layout\_below=*"@+id/image\_view"* />  </RelativeLayout> |

|  |
| --- |
| AndroidManifest.xml |
| <?xml version=*"1.0"* encoding=*"utf-8"*?>  <manifest xmlns:android=*"http://schemas.android.com/apk/res/android"*  package=*"com.scut.ofo"*  android:versionCode=*"1"*  android:versionName=*"1.0"* >  <uses-sdk  android:minSdkVersion=*"8"*  android:targetSdkVersion=*"18"* />  <application  android:allowBackup=*"true"*  android:icon=*"@drawable/ic\_launcher"*  android:label=*"@string/app\_name"*  android:theme=*"@style/AppTheme"* >  <activity  android:name=*"com.scut.ofo.MainActivity"*  android:label=*"@string/app\_name"* >  <intent-filter>  <action android:name=*"android.intent.action.MAIN"* />  <category android:name=*"android.intent.category.LAUNCHER"* />  </intent-filter>  </activity>  </application>  <uses-permission android:name=*"android.permission.CAMERA"* />  <uses-permission android:name=*"android.permission.FLASHLIGHT"* />  <uses-permission android:name=*"android.permission.WRITE\_EXTERNAL\_STORAGE"* />  <uses-feature android:name=*"android.hardware.camera"* />  <uses-feature android:name=*"android.hardware.camera.autofocus"* />    </manifest> |

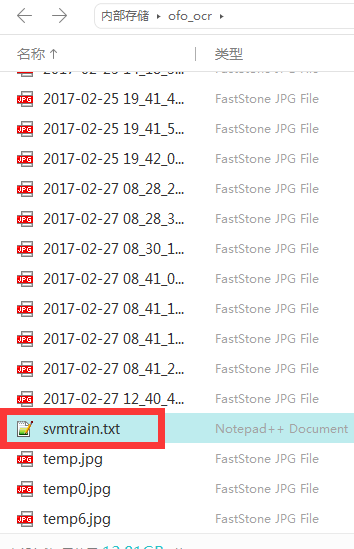
|  |
| --- |
| Android.mk |
| LOCAL\_PATH := $(call my-dir)  include $(CLEAR\_VARS)  OPENCV\_CAMERA\_MODULES:=on  OPENCV\_INSTALL\_MODULES:=on  OPENCV\_LIB\_TYPE:=SHARED  include opencv310sdk\native\jni\OpenCV.mk  LOCAL\_MODULE := ofo  LOCAL\_SRC\_FILES := ofo.cpp  LOCAL\_LDLIBS := -llog  include $(BUILD\_SHARED\_LIBRARY) |

|  |
| --- |
| Application.mk |
| APP\_STL := gnustl\_static  APP\_CPPFLAGS := -frtti -fexceptions  APP\_ABI := armeabi-v7a |

|  |
| --- |
| ofo.cpp |
| **#include** <jni.h>  **#include** <com\_scut\_ofo\_MainActivity.h> //请改成自己的  **#include** <opencv2/opencv.hpp>  **#include** <opencv2/core/core.hpp>  **#include** <opencv2/highgui/highgui.hpp>  **#include** <opencv2/ml/ml.hpp>  **#include** <string>  **#include** <vector>  **#include** <sstream>  **#include** <android/log.h>  **using** **namespace** cv;  **using** **namespace** std;  **static** **float** feature2Extract[7][32/8\*72/8];  **static** **int** resultOfoOcr[7];  **int** **img2feature**(Mat lastImg,**int** ror)  {  //s1-01:resize  cv::Mat resizeImg;  cv::resize(lastImg,resizeImg,Size(800,800));  resizeImg.copyTo(lastImg);  //s1-02:rotate  cv::Mat rotateImg;  //先设置中心点  cv::Mat M = cv::getRotationMatrix2D(Point2f(lastImg.cols/2,lastImg.rows/2),ror,1);  //再用warpAffine旋转  cv::warpAffine(lastImg,rotateImg,M,rotateImg.size());  rotateImg.copyTo(lastImg);  //s1-03 rgb2gray  Mat grayImg;  cv::GaussianBlur(lastImg,grayImg,Size(11,11),0,0); //高斯  cv::cvtColor(grayImg,grayImg,*COLOR\_RGB2GRAY*);  grayImg.copyTo(lastImg);  //s1-04:bw  Mat bwImg;  cv::Laplacian(lastImg,bwImg,-1,3); //边缘化  cv::threshold(bwImg,bwImg,0,255,*CV\_THRESH\_OTSU*+*CV\_THRESH\_BINARY*);  cv::fastNlMeansDenoising(bwImg,bwImg,30,15,31); //去燥  bwImg.copyTo(lastImg);  //s1-05:第一次膨胀，去掉数字中的燥点  Mat dilateImg1;  Mat element1 = cv::getStructuringElement(*MORPH\_RECT*,Size(3,3));  cv::dilate(lastImg,dilateImg1,element1);  cv::erode(dilateImg1,dilateImg1,element1);  dilateImg1.copyTo(lastImg);  //s1-06:绘制外接矩形，以寻找QR(7位)  Mat rectQRFindImg;  **int** QRFinded = 0;  **int** QRx,QRy,QRwidth,QRheight;  cv::Point2f QRFindRotatePoint[4];//用于旋转  lastImg.copyTo(rectQRFindImg);  cv::threshold(rectQRFindImg,rectQRFindImg,0,255,*CV\_THRESH\_OTSU*+*CV\_THRESH\_BINARY*);  std::vector< std::vector< cv::Point> > contoursQRFind;  cv::findContours(rectQRFindImg,contoursQRFind,*CV\_RETR\_EXTERNAL*,*CV\_CHAIN\_APPROX\_NONE*);  **for** (**int** i=0; i<contoursQRFind.size(); i++)  {  //绘制最小外接矩形  cv::Rect rectQRFind = cv::boundingRect(Mat(contoursQRFind[i]));  //判断是否为QR  **if**(  ((**float**(rectQRFind.width)/rectQRFind.height)>0.7)&&  ((**float**(rectQRFind.width)/rectQRFind.height)<1.3)&&  ((rectQRFind.width)>200)&&  ((rectQRFind.width)<400)&&  ((rectQRFind.x>10))&&  ((rectQRFind.x<200))&&  ((rectQRFind.y>50))&&  ((rectQRFind.y<400))  // (Mat(contoursQRFind[i]).size())>800)  )  {  cv::rectangle(rectQRFindImg,rectQRFind,Scalar(255,255,255));  **\_\_android\_log\_print**(*ANDROID\_LOG\_INFO*, "ofoocr", "s1-05: %d \* %d", rectQRFind.width,rectQRFind.height);  QRx = rectQRFind.x;  QRy = rectQRFind.y;  QRwidth = rectQRFind.width;  QRheight = rectQRFind.height;  QRFinded++;  //旋转  cv::RotatedRect rectQRFindRotate = cv::minAreaRect(Mat(contoursQRFind[i]));  rectQRFindRotate.points(QRFindRotatePoint);  **for** (**int** j=0; j<=3; j++)  {  line(rectQRFindImg,QRFindRotatePoint[j],QRFindRotatePoint[(j+1)%4],Scalar(255,255,255),10);  }  }  }  rectQRFindImg.copyTo(lastImg);  **if**(QRFinded != 1)  {  **\_\_android\_log\_print**(*ANDROID\_LOG\_INFO*, "ofoocr", "ERROR: Can't Find correct QR");  **return** 1;  }  //s1-08利用QR旋转  cv::Mat rotateOfoImg;  rotateImg.copyTo(rotateOfoImg); //不知为什么用lastImg会调用rotateImg  **double** angelOfo;  //需要补一个寻找左上角和右上角点。想到的思路是x+y最小的为左上角，而接下来的是右上角  **int** findLeft,findLeftXY = 800\*800;  **for** (**int** j=0; j<=3; j++)  {  **if** ((QRFindRotatePoint[j].x + QRFindRotatePoint[j].y) < findLeftXY)  {  findLeftXY = QRFindRotatePoint[j].x + QRFindRotatePoint[j].y;  findLeft = j;  }  }  // 用arctan来求旋转角度，公式(y/x)\*180/3.14  angelOfo = atan ((QRFindRotatePoint[findLeft+1].y - QRFindRotatePoint[findLeft].y) / (QRFindRotatePoint[findLeft+1].x - QRFindRotatePoint[findLeft].x)) \* 180 / 3.14;  //先设置中心点  cv::Mat M2 = cv::getRotationMatrix2D(Point2f(rotateOfoImg.cols/2,rotateOfoImg.rows/2),angelOfo,1);  //再用warpAffine旋转  //参数 源，目标，角度，尺寸  cv::warpAffine(rotateOfoImg,rotateOfoImg,M2,rotateOfoImg.size());  rotateOfoImg.copyTo(lastImg);  //s1-09 粗略将车牌连QR一起crop出来  Mat cropOfo7CharImg;  rotateOfoImg.copyTo(cropOfo7CharImg);  //二点间距离公式  // sqrt(pow((x2-x1),2)+pow((y2-y1),2))  **float** distanceQRx = sqrt(pow((QRFindRotatePoint[findLeft+1].x - QRFindRotatePoint[findLeft].x),2) + pow((QRFindRotatePoint[findLeft+1].y - QRFindRotatePoint[findLeft].y),2));  **float** distanceQRy = sqrt(pow((QRFindRotatePoint[findLeft].x - QRFindRotatePoint[findLeft-1].x),2) + pow((QRFindRotatePoint[findLeft].y - QRFindRotatePoint[findLeft-1].y),2));  cout<<"distanceX: "<<distanceQRx<<" Y: "<<distanceQRy<<**endl**;  **int** maskOfo7CharX = distanceQRx\*1.4;  **if**((maskOfo7CharX + QRx + distanceQRx) > 800)  {  maskOfo7CharX = 800-(QRx + distanceQRx);  }  cv::Rect maskOfo7Char(QRx + distanceQRx, QRy + (distanceQRy\*0.6), maskOfo7CharX, distanceQRy\*0.5);  cv::Mat croppedOfo7CharImg(cropOfo7CharImg, maskOfo7Char);  lastImg.empty();  croppedOfo7CharImg.copyTo(lastImg);  //s2-01 用小矩阵，再次进行中值滤波、灰度、二值化、膨胀  Mat ofo7CharReProcessImg;  **int** Ofo7CharFind = 0;  **int** Ofo7CharX,Ofo7CharY,Ofo7CharWidth,Ofo7CharHeight;  croppedOfo7CharImg.copyTo(ofo7CharReProcessImg);  cv::medianBlur(ofo7CharReProcessImg,ofo7CharReProcessImg,11);//中值滤波  cv::cvtColor(ofo7CharReProcessImg,ofo7CharReProcessImg,*COLOR\_BGR2GRAY*);  cv::Canny(ofo7CharReProcessImg,ofo7CharReProcessImg,50,150,3);//取边缘  Mat elementOfo7CharReProcess = cv::getStructuringElement(*MORPH\_RECT*,Size(25,1));//设置膨胀参数  cv::dilate(ofo7CharReProcessImg,ofo7CharReProcessImg,elementOfo7CharReProcess); //膨胀  cv::threshold(ofo7CharReProcessImg,ofo7CharReProcessImg,0,255,*CV\_THRESH\_OTSU*+*CV\_THRESH\_BINARY*); //再次二值化，为绘制矩形做准备  std::vector< std::vector< cv::Point> > contoursOfo7CharFind; //存放指针  cv::findContours(ofo7CharReProcessImg,contoursOfo7CharFind,*CV\_RETR\_EXTERNAL*,*CV\_CHAIN\_APPROX\_NONE*); //找轮廓  **for** (**int** i=0; i<contoursOfo7CharFind.size(); i++)  {  //绘制最小外接矩形  cv::Rect rectOfo7CharFind = cv::boundingRect(Mat(contoursOfo7CharFind[i]));  //判断是否为7个字符  **if**(  (((**float**(rectOfo7CharFind.width))/rectOfo7CharFind.height)>3.2)&&  (((**float**(rectOfo7CharFind.width))/rectOfo7CharFind.height)<4.8)&&  ((rectOfo7CharFind.width>distanceQRx)||(rectOfo7CharFind.width>distanceQRy))  )  {  cv::rectangle(ofo7CharReProcessImg,rectOfo7CharFind,Scalar(255,255,255));  cout<<"Draw7Char: "<< rectOfo7CharFind.width <<"\*"<<rectOfo7CharFind.height<<**endl**;  Ofo7CharX=rectOfo7CharFind.x;  Ofo7CharY=rectOfo7CharFind.y;  Ofo7CharWidth=rectOfo7CharFind.width;  Ofo7CharHeight=rectOfo7CharFind.height;  Ofo7CharFind++;  }  }  **if**(Ofo7CharFind!=1)  {  **\_\_android\_log\_print**(*ANDROID\_LOG\_INFO*, "foo", "ERROR:Can't Find OFO 7 Char position");  **return** 2;  }  ofo7CharReProcessImg.copyTo(lastImg);  //s2-02 crop出来  Mat cropReProcessImg;  croppedOfo7CharImg.copyTo(cropReProcessImg);  cv::Rect maskOfo7CharReProcessImg(Ofo7CharX, Ofo7CharY, Ofo7CharWidth,Ofo7CharHeight);  cv::Mat croppedOfo7CharReProcessImg(cropReProcessImg, maskOfo7CharReProcessImg);  croppedOfo7CharReProcessImg.copyTo(lastImg);  //s2-03 将7个字找出来  Mat char7findImg;  Mat char7findImgBw;  //charList结构：1标记,x,y,w,h  **int** charList[7][5] = {};  **int** charListCount = 0;  croppedOfo7CharReProcessImg.copyTo(char7findImg);  // cout<<char7findImg.cols<<"\*"<<char7findImg.rows<<endl;  cv::medianBlur(char7findImg,char7findImg,5);//中值滤波  cv::cvtColor(char7findImg,char7findImg,*COLOR\_BGR2GRAY*);  cv::threshold(char7findImg,char7findImg,0,255,*CV\_THRESH\_OTSU*+*CV\_THRESH\_BINARY*); //再次二值化，为绘制矩形做准备  char7findImg.copyTo(char7findImgBw);  std::vector< std::vector< cv::Point> > contourschar7findImg; //存放指针  cv::findContours(char7findImg,contourschar7findImg,*CV\_RETR\_EXTERNAL*,*CV\_CHAIN\_APPROX\_NONE*); //找轮廓  **for** (**int** i=0;i<contourschar7findImg.size();i++)  {  //绘制最小外接矩形  cv::Rect rectRotateReProcess=cv::boundingRect(Mat(contourschar7findImg[i]));  // cout<<"Char:"<<rectRotateReProcess.width<<"\*"<<rectRotateReProcess.height<<" x:"<<rectRotateReProcess.x<<" y:"<<rectRotateReProcess.y<<endl;  **if** (  (((**float**(rectRotateReProcess.height))/rectRotateReProcess.width) > 1.5)&&  (((**float**(rectRotateReProcess.height))/rectRotateReProcess.width) < 4)&&  (rectRotateReProcess.height > ((char7findImg.rows)\*0.67))  )  {  cv::rectangle(char7findImg,rectRotateReProcess,Scalar(255,255,255));  charList[charListCount][1] = rectRotateReProcess.x;  charList[charListCount][2] = rectRotateReProcess.y;  charList[charListCount][3] = rectRotateReProcess.width;  charList[charListCount][4] = rectRotateReProcess.height;  charListCount++;  }  }  **if**(charListCount!=7)  {  **\_\_android\_log\_print**(*ANDROID\_LOG\_INFO*, "ofoocr", "ERROR: Can't find ALL 7 char");  **return** 3;  }  char7findImg.copyTo(lastImg);  //s2-04排序  //冒泡排序  **int** charSortTemp[5];  **for** (**int** i=0; i<7; i++)  {  **for** (**int** j=i+1; j<7; j++)  {  **if**(charList[i][1] > charList[j][1])  {  **for**(**int** k=0; k<5; k++)  {  charSortTemp[k] = charList[i][k];  charList[i][k] = charList[j][k];  charList[j][k] = charSortTemp[k];  }  }  }  }  //s2-05 crop出来，并且resize 32\*72  Mat croppedCharListImg[7];  Mat croppedCharTempImg;  char7findImgBw.copyTo(croppedCharTempImg);  **for** (**int** i=0; i<7; i++)  {  cv::Rect maskCharList(charList[i][1],charList[i][2],charList[i][3],charList[i][4]);  cv::Mat croppedChar(croppedCharTempImg, maskCharList);  croppedChar.copyTo(croppedCharListImg[i]);  cv::resize(croppedCharListImg[i],croppedCharListImg[i],Size(32,72),0,0,*INTER\_CUBIC*);  //因为ADT用不了to\_string，只能先这样来debug~\_~  // if(i==0)  // {  // cv::imwrite("/sdcard/ofo\_ocr/temp0.jpg",croppedCharListImg[i]);  // } else if(i==6) {  // cv::imwrite("/sdcard/ofo\_ocr/temp6.jpg",croppedCharListImg[i]);  // }  }  //s2-06 将Mat矩阵转换成二维矩阵，因为Mat矩阵只能用一维指针来操作，太麻烦了  **int** imgArray[7][72][32];  **for**(**int** i=0; i<7; i++)  {  //先二值化  cv::threshold(croppedCharListImg[i],croppedCharListImg[i],0,255,*CV\_THRESH\_OTSU*+*CV\_THRESH\_BINARY*);  croppedCharListImg[i].convertTo(croppedCharListImg[i],CV\_8U);  cv::MatIterator\_<cv::Vec3b> it = croppedCharListImg[i].begin<cv::Vec3b>();  **int** t=0;  **for**(; it!=croppedCharListImg[i].end<cv::Vec3b>(); it++)  {  **if**((((\*it)[0])==255)&&(((\*it)[1])==255)&&(((\*it)[2])==255))  {  imgArray[i][t/32][t%32] = 1;  } **else** {  imgArray[i][t/32][t%32] = 0;  }  t++;  }  }  //s2-07 计算feature2  **for** (**int** i=0; i<7; i++)  {  cv::MatIterator\_<cv::Vec3b> it = croppedCharListImg[i].begin<cv::Vec3b>();  **int** arrayNum = 0;  **for**(**int** j=0; j<72; j=j+8)  {  **for** (**int** k=0; k<32; k=k+8)  {  **int** countBW = 0;  **for** (**int** lj=0; lj<8; lj++)  {  **for**(**int** lk=0; lk<8; lk++)  {  **if** ((imgArray[i][j+lj][k+lk]) == 1)  {  countBW++;  }  }  }  feature2Extract[i][arrayNum] = countBW;  arrayNum++;  }  }  }  //输出至log方便观察 (奇怪是每个数都比Qt算出来的少1)  **for** (**int** i=0; i<7; i++)  {  **for** (**int** j=0; j<32/8\*72/8; j++)  {  **if**(feature2Extract[i][j] == 0)  {  feature2Extract[i][j] = 0.01; //关键！！！所有为0的均赋值为0.01，否则预测计的值会全部相同  }  }  }  cv::imwrite("/sdcard/ofo\_ocr/temp.jpg",lastImg);  }  //JNImain  **extern** "C" {  JNIEXPORT jintArray JNICALL **Java\_com\_scut\_ofo\_ofo\_grayProc** (JNIEnv\* env, jclass obj, jintArray buf, jint w, jint h,jint ror) //请改成自己  {  jint \*cbuf;  cbuf = env->GetIntArrayElements(buf,JNI\_FALSE);  **if**(cbuf == NULL){  **return** 0;  }  Mat imgData(h, w, CV\_8UC4, (**unsigned** **char**\*)cbuf);  Mat lastImg;  imgData.copyTo(lastImg);  img2feature(lastImg,ror);  //训练 3.1.0  // using namespace cv::ml;  // Ptr<SVM> svm = SVM::create();  // svm->setType(SVM::C\_SVC);  // svm->setKernel(SVM::POLY);  // svm->setDegree(3);  // svm->setGamma(0.01);  // svm->setCoef0(2);  // svm->setTermCriteria(TermCriteria(TermCriteria::MAX\_ITER, 100, 1e-6));  // svm->train(tm,ROW\_SAMPLE, labelsMat);  // svm->save("/sdcard/ofo\_ocr/svmtrain1775687.txt");  // \_\_android\_log\_print(ANDROID\_LOG\_INFO, "foo", "train finish");  //预测SVM 3.1.0  **for** (**int** i=0; i<7; i++)  {  Mat trainningDataMat(1,32/8\*72/8,CV\_32FC1,feature2Extract[i]);  //用于训练  // int q = svm->predict(trainningDataMat);  //实战  cv::Ptr<cv::ml::SVM> mSvm2;  mSvm2 = Algorithm::*load<cv::ml::SVM>*("/sdcard/ofo\_ocr/svmtrain.txt");  **int** q2 = mSvm2->predict(trainningDataMat);  **\_\_android\_log\_print**(*ANDROID\_LOG\_INFO*, "foo", "%d ",q2);  resultOfoOcr[i] = q2;  }  jintArray jarr = env->NewIntArray(7);  jint \*arr = env->GetIntArrayElements(jarr,NULL);  **for** (**int** i=0; i<7; i++)  {  arr[i] = resultOfoOcr[i];  }  **return** jarr;  }  } |

|  |
| --- |
| MainActivity.java |
| **package** com.scut.ofo;  **import** java.io.File;  **import** java.io.FileInputStream;  **import** java.io.IOException;  **import** android.os.Bundle;  **import** android.app.Activity;  **import** android.graphics.Bitmap;  **import** android.graphics.Bitmap.Config;  **import** android.graphics.BitmapFactory;  **import** android.util.Log;  **import** android.view.Menu;  **import** android.view.View;  **import** android.view.View.OnClickListener;  **import** android.widget.Button;  **import** android.widget.ImageView;  **import** android.widget.TextView;  **public** **class** MainActivity **extends** Activity **implements** OnClickListener{    **private** **static** String *TAG* = "ofoocr";  **private** Button btnProc;  **private** ImageView imageview;  **private** Bitmap bmp;  **private** TextView tv;    //OpenCV类库加载并初始化成功后的回调函数，在此我们不进行任何操作  **static** {  Log.*i*(*TAG*, "TryloadLibrary");  System.*loadLibrary*("ofo");  Log.*i*(*TAG*, "loadLibrary");  }      @Override  **protected** **void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);  setContentView(R.layout.*activity\_main*);    imageview = (ImageView) findViewById(R.id.*image\_view*);  bmp = BitmapFactory.*decodeResource*(getResources(), R.drawable.*ofo*);  imageview.setImageBitmap(bmp);    tv = (TextView) findViewById(R.id.*text\_view*);    btnProc = (Button) findViewById(R.id.*btn\_gray\_process*);  btnProc.setOnClickListener(**new** Button.OnClickListener()  {  @Override  **public** **void** onClick(View arg0) {  // **TODO** Auto-generated method stub    **int** w = bmp.getWidth();  **int** h = bmp.getHeight();    //bmp --> rgb\_pixels\_data  **int**[] pixels = **new** **int**[w\*h];  bmp.getPixels(pixels, 0, w, 0, 0, w, h);    //rgb\_pixels\_data --> gray\_pixels\_data  **int**[] feature2 = ofo.*grayProc*(pixels, w, h,270);  String resultOfoOcr = "";  **for** (**int** i=0;i<7;i++)  {  Log.*i*(*TAG*, "return: "+ feature2[i]);  resultOfoOcr += Integer.*toString*(feature2[i]);  }  Log.*i*(*TAG*, "return: "+ resultOfoOcr);  tv.setText(resultOfoOcr);      //找了很久资源都找不到Mat转jintarray，只能用文件交互法  String filepatha = "/sdcard/ofo\_ocr/temp.jpg";  File file = **new** File(filepatha);  **if** (file.exists()) {  Bitmap bm = BitmapFactory.*decodeFile*(filepatha);  imageview.setImageBitmap(bm);  }  Log.*i*(*TAG*, "setImageBitmap");    }  });  }  @Override  **public** **boolean** onCreateOptionsMenu(Menu menu) {  // Inflate the menu; this adds items to the action bar if it is present.  getMenuInflater().inflate(R.menu.*main*, menu);  **return** **true**;  }  @Override  **public** **void** onClick(View arg0) {  // **TODO** Auto-generated method stub    }    } |

然后在手机中的/sdcard/目录中建立ofo\_ocr目录，并将svmtrain.txt传上去。



测试一下



成功！



接下来的步骤就相当简单，加个摄像头、做些优化就可以。