**基于Android图片处理**

# 一、介绍

本设计主要是基于Android对图片进行增强、评价和编辑等处理。对图片的增强采用OpenCV库进行处理，使用C++语言编写算法；评价功能采用Python语言编写算法；而图片编辑则是采用Android SDK处理。

由于本设计采用多种语言进行开发设计，因此将各大功能“模块化”，以提高程序的内聚性，降低耦合性。

# 二、图片增强模块

图片增强使用C++语言编写算法，Google对此提供了NDK，Android NDK是一套允许开发者使用原生语言（例如C和C++）实现部分应用的工具集。当然，NDK只是开发环境，而如果要让Java调用C++代码，还需要另一个工具——JNI，JNI是Java Native Interface（Java本地接口）的缩写。JNI作为Java和操作系统间的一个直接接口，可以通过JNI使得Java直接调用操作系统的资源。目前JNI只能通过C/C++实现，因为JNI只是对操作系统资源调用的一个桥接过程。Java代码编译之后是运行在一个Jvm里，所以Java的任何操作对操作系统而言都是隔着一层虚拟机外壳，这点也正式Java的优点，帮助Java实现了“Write Once, Run Everywhere”的可移植性。但是使用了JNI之后必须要明白这个“Write Once, Run Everywhere”要被打破，必须要实现不同的操作系统的各种JNI版本。

使用JNI调用C++代码时对C++代码的方法和类型有一些要求，具体请看本目录下的“readme.md”文件。

# 三、图片评价模块

图片评价模块采用Python语言编写，由于Android 不能直接调用Python，必须先加载Python解释器，然后Python解释器再加载Python文件执行Python代码。这样一来Python代码并不是被编译到App中去，而是以文件的形式打包进去，再由Python解释器来读取加载，这种方式比较麻烦，而且不方便程序调试，更重要的Python代码包含的第三方库也只能是纯Python编写的。还有一种方法可以调用Python，那就是Java通过JNI调用C++代码，C++再调用Python代码。不过这样一来就又需要编写额外的C++代码了，更重要的是，这种方式只能调用纯Python语言代码。也就是说，该Python代码的所有实现都必须是Python代码，包括第三方库。而opencv-python虽然是Python的第三方库，但其确实以C/C++为核心库，再打包成Python库。因此使用Java通过JNI调用C++，C++再调用Python的方案也不可取。

目前能想到最好的方案就是构建一个完整的Python运行环境，参考网上的方法，引入“chaquo”第三方库，可以构建Python运行环境。而引入Python第三方库也是使用pip程序下载引入，方法如图3-1所示：

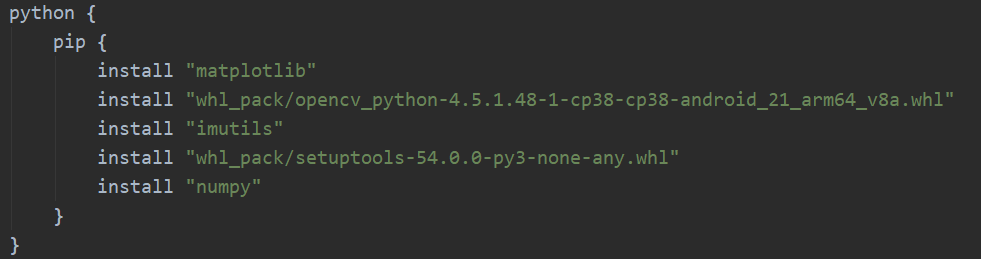


图3-1 导入python第三方库

图中有两个第三方库是本地文件，即“\*.whl”。这是由于pip程序无法从互联网下载该库，所以手动下载这两个库文件，放到项目中，本地加载。

由于“opencv-python”库使用“Mat”类读取图片，因此Java代码无法直接传递图片参数给Python，而Python代码也无法直接返回图片参数给Java代码。因此在传递图片时需要将图片数据转换为Python和Java都有的基本类型——String。所以将图片按“Base64”规则对图片进行编码从而转为String类型，当Python代码接收到该String之后再进行Base64解码得到Mat类型的图片，返回时也需要对图片进行Base64编码，Java接收到返回值之后再次进行Base64解码得到处理后的图片。

## 四、图片编辑模块

本设计的图片编辑主要是对图片进行裁剪，其中裁剪又需要实现两种不同的裁剪方案，第一种是对图片进行“九宫格”式裁剪，除了九宫格式裁剪，本设计还实现了多种裁剪方式，例如将一张图片裁剪为2、3、4、6、8、9、10、12、15、16等份的图片；而另一种裁剪方案则是选定一个区域对图片进行裁剪。事实上两种裁剪方案的原理都一样，只是裁剪结果不一样而已。下文将详细介绍“等份裁剪”原理：

1. 创建一个大小为图片宽高之积的int类型的一维数组；
2. 将Bitmap图片转为像素点并存放到该数组中；
3. 根据要裁剪的份数计算横向和纵向的张数；
4. 根据要裁剪的份数计算裁剪后图片的宽高；
5. 使用双循环对图片进行等份裁剪；
6. 返回裁剪结果。

注：可以根据上述步骤画裁剪流程图。

下文将结合代码对上述步骤做进一步介绍：

### 4.1 等份裁剪图片原理

图4-1对应上文1~2两个步骤。这两个步骤主要解释

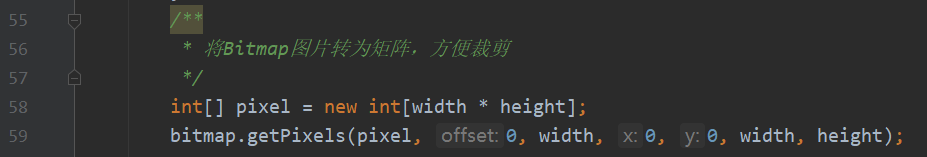


图4-1 Bitmap转矩阵图片代码

getPixels方法，该方法是Android SDK提供的图片裁剪方法，其原型如图4-2所示。

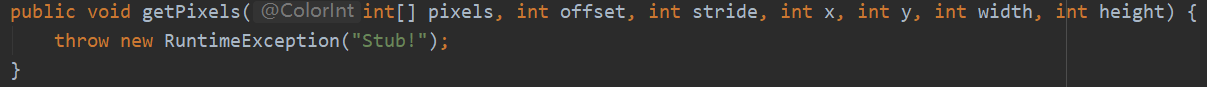


图4-2 getPixels方法原型

第一个参数是int类型数组，表示Bitmap转矩阵后存放的位置；

第二个参数从名字上看就知道是偏移量，Google官方对该参数的原话解释：“The first index to write into pixels[]”。即从数组的哪个位置开始写入；

第三个参数表示隔多少个像素再写下一行；

第四、五个参数则是从Bitmap读取的第一个像素的x、y坐标；

第六、七个参数分别表示一行要读取多少个像素和读取多少行。

对该方法做一个总结，第二、三个参数是告诉程序如何将数据存放到目标数组中去，而第四到七个参数是告诉程序如何读取原图，也就是说该方法只是根据参数来读取图片的一部分或全部像素到目标数组中去，并不是对图片进行裁剪。本设计是要对图片进行裁剪，因此需要读取图片的所有像素。

图4-3对应上文3~4步骤，图中可以看到，如果要将原图片裁剪

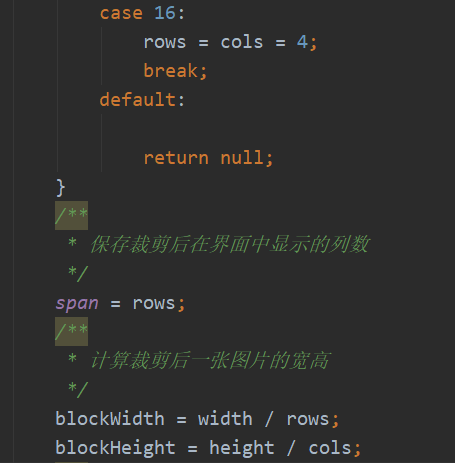


图4-3 计算裁剪图片的份数和宽高代码

为16等份的图片，则需要将图片的宽高分别4等份，而新图的宽高则为原图的宽高除以4。

图4-4对应上文5~6步骤，代码中“crop”方法各参数作用请看图4-4中注释部分。其中“crop”方法用到了Android SDK的一个方



图4-4 n等份图片裁剪代码

法：“createBitmap”，顾名思义，该方法的作用是生成位图。该方法的原型如图4-5所示。该方法的参数和“getPixels”方法的参数差

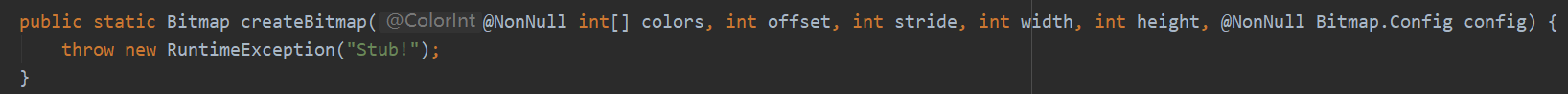


图4-5 createBitmap方法原型

不多，无非就是反过来而已。例如第一个参数就是存有像素点的数组，第二个参数是从数组的那个位置开始读取，第三个参数是读取多少个像素换行，第四、五个参数是新图的宽高，最后一个参数是图片的配置。

使用上文所述两个方法方可对图片进行裁剪。下文将举一个例子说明整个裁剪的过程。

现假设要对一张宽高均为100（像素）的图片进行4等份裁剪，经计算每张图片的宽高均为50（像素）。通过图4-1中的代码将图片转为数组pixel之后，pixel是这样存放原图的：0~99存放的是原图的第一行数据，100~199存放的是原图第二行的数据，以此类推。图4-4进行2\*2内外双循环对图片进行裁剪：

第一次循环，i=j=0，即offset为0，表示从pixel[0]开始读取，读取到width位置再读取下一行，其中width为原图的宽度，读完原图的第一行（即0~99）之后，由于cWidth为50，因此截取pixel[0]~pixel[49]作为新图的第一行，而cHeight为50，一共读取原图的50行，每次都将pixel[0]~pixel[49]填充到新图中，当第一轮循环完成之后，就得到了原图中坐标为(0,0)~(49,49)的部分，即左上角1/4的图；

第二次循环，i=0，j=1，即offset=1\*50+0\*50\*100=50，也就是说第二次循环从pixel[50]开始读取，当第二轮循环结束后得到的是原图右上角1/4的图片；

第三次循环，i=1,j=0，即offset=0\*50+1\*50\*100=5000,也就是说第三次循环从pixel[5000]开始读取，当第三轮循环结束后得到的是原图左下角1/4的图；

第四次循环，i=j=1，即offset=1\*50+1\*50\*100=5050，也就是说第四次循环从pixel[5050]开始读取，当第四轮循环结束后得到的是原图右下角1/4的图。