

**信号与信息处理综合实验**

**实验报告**

**学院： 电子信息工程学院**

**姓名： 高成鑫**

**学号： 17211270**

**小组成员： 李洺慧**

**指导教师： 陈新**

**北京交通大学**

**2020年 5 月28 日**

目录

[1 实验内容 3](#_Toc41663708)

[2 实验背景 3](#_Toc41663709)

[2.1 Android平台下的图像信息 3](#_Toc41663710)

[2.1.1 Mat对象 3](#_Toc41663711)

[2.1.2 Bitmap对象 3](#_Toc41663712)

[2.1.3 图像选择 3](#_Toc41663713)

[2.1.4 处理与显示 3](#_Toc41663714)

[2.2 图像处理 4](#_Toc41663715)

[2.2.1 卷积 4](#_Toc41663716)

[2.2.2 空间域滤波 4](#_Toc41663717)

[1.均值滤波 4](#_Toc41663718)

[2.高斯滤波 5](#_Toc41663719)

[3.统计排序滤波器（中值滤波、最大最小值） 5](#_Toc41663720)

[4.边缘保留滤波 6](#_Toc41663721)

[3 总体描述 6](#_Toc41663722)

[4 设计内容：介绍主要功能模块和设计方法 6](#_Toc41663723)

[4.1 准备工作 6](#_Toc41663724)

[4.1.1 加载OpenCV的本地库 7](#_Toc41663725)

[4.1.2 给按钮加上OnClickListner事件响应 7](#_Toc41663726)

[4.2 编写图像处理事件响应方法 8](#_Toc41663727)

[4.2.1 空间域滤波 8](#_Toc41663728)

[4.2.1.1 均值滤波 8](#_Toc41663729)

[4.2.1.2 高斯滤波 8](#_Toc41663730)

[4.2.1.3 统计排序滤波器滤波 9](#_Toc41663731)

[4.1.2.4 边缘保留滤波 10](#_Toc41663732)

[5. 总结与体会： 10](#_Toc41663733)

[6. 参考文献： 11](#_Toc41663734)

**多平台下二维图像信号的采集与处理实验（Android平台下的实验）**

# 1 实验内容

编程实现图像的读取与存储，并利用图像时频域分析方法确定图像中存在的噪声干扰，设计适用的滤波器实现图像的滤波处理（这里滤波处理建议采用自编函数实现）

（3）Android Studio平台下Java/C++编程、调用OpenCV实现（1）中实现的图像的分析和处理工程应用，并设计实现Android平台下的Activity应用界面；

（4）分别在Windows平台和嵌入式Firefly RK3399 Android平台下运行（2~3）实现的工程应用程序，对比分析不同环境下的图像处理的结果与性能。

# 2 实验背景

## 2.1 Android平台下的图像信息

### 2.1.1 Mat对象

Mat是OpenCV中用来存储图像信息的内存对象，当通过Imgcodecs.imread()方法从文件中读入一个图像文件时，imread方法就会返回Mat对象实例，或者通过Utils.bitmapToMat()方法由Bitmap对象转换而来。Mat对象中除了存储图像的像素数据外，还包括图像的其他属性，具体为宽、高、类型、维度、大小、深度等。当你需要这些信息时，还可以通过相关的API获取这些基本图像属性。

### 2.1.2 Bitmap对象

在Android系统中有一个和Mat对象相似的对象Bitmap。通过它可以获取图像的常见属性、像素数据，修改图像的像素数据，呈现出不同的图像显示效果，保存图像，等等。

### 2.1.3 图像选择

调用Android图片浏览功能，选择一张图片，自动返回图像并显示，代码如下：

代码1 图像选择程序

Code 1 Image selection program

Intent intent = new Intent();  
intent.setType("image/\*");  
intent.setAction(Intent.*ACTION\_GET\_CONTENT*);  
startActivityForResult(Intent.*createChooser*(intent,"选择图像..."), PICK\_IMAGE\_REQUEST);

### 2.1.4 处理与显示

调用OpenCV4Android的API对图像进行有目的的处理，处理之后返回处理后的图像并显示，这里是一个测试程序，使用OpenCV的imread功能来读取图像，完成灰度转换并显示。

代码2 图像灰度化处理程序

Code 2 Image graying processing program

private void convertGray() {  
 Mat src = new Mat();  
 Mat temp = new Mat();  
 Mat dst = new Mat();  
 Utils.*bitmapToMat*(selectbp, src);  
 Imgproc.*cvtColor*(src, temp, Imgproc.*COLOR\_BGRA2BGR*);  
 Log.*i*("CV", "image type:" + (temp.type() == CvType.*CV\_8UC3*));  
 Imgproc.*cvtColor*(temp, dst, Imgproc.*COLOR\_BGR2GRAY*);  
 Utils.*matToBitmap*(dst, selectbp);  
 myImageView.setImageBitmap(selectbp);   
 //少了对src、temp、dst的release;  
}

## 2.2 图像处理

### 2.2.1 卷积

卷积常用于图像模糊，对于二维的图像卷积，可以表示为

(1)

其中F是卷积核/卷积操作数

### 2.2.2 空间域滤波

#### 1.均值滤波

对于上述卷积核，对于常见的均值滤波，F为

(2)

均值滤波最主要的作用是可以降低图像的噪声、模糊图像、降低图像的对比度。在OpenCV对应的模糊函数在Imgproc模块下定义如下：

代码3 OpenCV库中的均值滤波的模糊函数

Code 3 Fuzzy function of mean filtering in OpenCV library

public static void blur(Mat src, Mat dst, Size ksize, Point anchor, int borderType)

其中，src表示输入图像，dst表示卷积模糊后输出的图像，ksize表示图像卷积核的大小，anchor表示卷积核的中心位置，borderType边缘填充的类型，默认情况下为BORDER\_DEFAULT方式。可根据该函数编写均值滤波的处理代码。

#### 2.高斯滤波

在均值滤波中，卷积核所有的系数都相等，但另一种常见的方式是根据空间相对位置的不同，卷积核中的每个系数具有不同的权重，因为我们经常使用高斯正态分布函数来生成权重系数，因此这种模糊又称为高斯模糊。用来生成权重系数的高斯公式通常为：

(3)

对于整个卷积核来说，越是在卷积核中心地方，其系数应该越高，越靠近边缘的地方，其系数应该越低，由此即可得知基于高斯的卷积核权重与空间位置的关系。同时，也可以看出高斯模糊与参数σ有关，σ越大，模糊程度越大。OpenCV中高斯模糊API处于Imgproc模块中，定义如下：

代码4 OpenCV库中的高斯滤波的API

Code 4 Gaussian filtering API in OpenCV library

public static void GaussianBlur(Mat src, Mat dst, Size ksize, double sigmaX, double sigmaY, int borderType)

其中：src为输入图像，常见为CV\_8UC3/CV\_8UC4类型。dst表示高斯模糊后输出图像，类型通常与输入图像一致。ksize表示卷积核大小，当ksize=new Size(0,0)时候表示从sigmaX计算得到。sigmaX表示X方向上的模糊程度，sigmaY表示Y方向的模糊程度，当sigmaY不填写时，表从sigmaX计算得到。boderType表示边缘填充方式，默认为BORDER\_DEFAULT。

#### 3.统计排序滤波器（中值滤波、最大最小值）

之前的两种图像滤波操作都属于图像的线性滤波，而非线性滤波器在特定场合与应用场景下也有许多的应用，其中主要是几种基于统计排序的滤波器，他们是中值滤波、最大最小值滤波，他们也常用于消除图像噪声或者抑制图像像素极小值与极大值。

中值滤波是将对应的像素值排序，取中间值作为输出，相关API在Imgproc包中。

代码5 OpenCV库中的中值滤波API

Code 5 Median filtering API in OpenCV library

public static void medianBlur(Mat src, Mat dst, int ksize)

ksize表示排序模板的大小，常见为3、5，模板大小必须为奇数并且必须大于1.

最大最小值滤波与中值滤波相似，唯一不同就是排序后使用像素的最大最小值来作为输出。OpenCV没有以max或min单词为开头来命名的最大或最小值滤波函数，是通过两个形态学操作函数来替代实现最大值与最小值滤波，他们是dilate与erode。

代码6 OpenCV库中的最大最小值滤波API

Code 6 Maximum and minimum filtering API in OpenCV library

public static void dilate(Mat src, Mat dst, Mat kernel); //膨胀（最大值滤波）

public static void erode(Mat src, Mat dst, Mat kernel) //腐蚀（最小值滤波）

kernel表示结果元素或者卷积核，它可以是任意形状。

#### 4.边缘保留滤波

边缘保留滤波也是一类非线性滤波器，OpenCV中已经实现的边缘保留滤波有高斯双边滤波、金字塔均值迁移滤波，他们有类似人脸美化或者图像美化的效果。

高斯双边滤波是在高斯滤波基础上进一步扩展与延申出来的图像滤波方法，相较于高斯滤波，双边滤波器(Bilateral Filter)可以在很好地保留边缘的同时，抑制平坦区域图像的噪声。双边滤波器能做到这些的原因在于它不像普通的高斯/卷积低通滤波，其不仅考虑了位置中心像素的影响，还考虑了卷积核中像素与中心像素之间相似程度的影响，据说，Adobe Photoshop的高斯磨皮功能就是应用了此技术。

高斯双边滤波的函数为：

public static void bilateralFilter(Mat src, Mat dst, int d, double sigmaColor, double sigmaSpace)

各个参数的解释如下：d表示用来过滤的卷积核直径大小，一般取0，意思是从sigmaColor参数自动计算。sigmaColor颜色权重计算时候需要用到的参数，sigmaSpace空间权重计算时需要用到的参数。通常情况下，sigmaColor的取值范围一般在100~150左右，sigmaSpace的取值范围在10~25之间的时候，双边滤波的效果比较好，调用函数时的速度也会比较快。

均值迁移滤波主要是通过概率密度估计与中心迁移的方式来实现图像边缘保留滤波，其基本原理是通过创建大小指定的卷积核窗口，搜索并计算该窗口中心像素P(x,y)范围内所有满足条件的像素，计算他们的中心位置，然后基于新中心位置再次更新计算，直至中心位置不再变化或者两次变化的钟信的距离满足置定的收敛精度值为止。

public static void pyrMeanShiftFiltering(Mat src, Mat dst, double sp, double sr, int maxLevel, TermCriteria termcrit)

# 3 总体描述

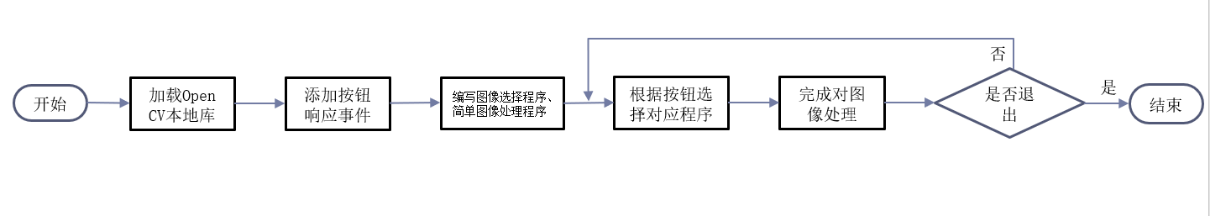
 本次实验的主要流程如下图所示

图 1 Android平台下二维图像信号的采集与处理实验流程图

Fig. 1 Experimental flow chart of the acquisition and processing of two-dimensional image signals under the Android platform

# 4 设计内容：介绍主要功能模块和设计方法

## 4.1 准备工作

### 4.1.1 加载OpenCV的本地库

代码7 加载OpenCV的本地库

Code 7 Load OpenCV native library

private void staticLoadCVLibraries(){  
 boolean load = OpenCVLoader.initDebug();  
 if(load) {  
 Log.i("CV", "Open CV Libraries loaded...");  
 }  
 else{  
 Toast.makeText(this.getApplicationContext(),"WARNING:Could not load OpenCV Libraries!",  
 Toast.LENGTH\_LONG).show();  
 }  
}

### 4.1.2 给按钮加上OnClickListner事件响应

代码8 给按钮加上事件响应

Code 8 Add event response to the button

Button selectImageBtn = (Button)findViewById(R.id.select\_btn);  
selectImageBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 makeText(MainActivity.this.getApplicationContext(), "start to browser image",  
 Toast.LENGTH\_SHORT).show();  
 selectImage();  
 }  
});  
  
Button processBtn = (Button)findViewById(R.id.process\_btn);  
processBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 makeText(MainActivity.this.getApplicationContext(), "hello, image process",  
 Toast.LENGTH\_SHORT).show();  
 convertGray();  
 }  
});

这里有两个响应程序，selectImage()选择图片，具体实现如2.3中所述，covertGray()对图像的灰度化，实现如2.4所述。

## 4.2 编写图像处理事件响应方法

在4.1.2中processBtn按钮里面增加滤波处理，改善图片中的噪声影响。

### 4.2.1 空间域滤波

#### 4.2.1.1 均值滤波

根据代码3中函数，可以写处均值滤波处理代码如下，这里选择的卷积核大小为5x5。

代码9对图像进行均值滤波处理

Code 9 mean filtering on the image

//均值滤波  
private void meanFilteringpImg(){  
 Mat src = new Mat();  
 Mat dst = new Mat();  
 Utils.*bitmapToMat*(selectbp, src);  
  
 if (src.empty()){  
 Log.*e*("Error", "Load bitmap failed.");  
 return;  
 }  
  
 Imgproc.*blur*(src, dst, new Size(5,5), new Point(-1,-1), Core.*BORDER\_DEFAULT*);  
 Log.*i*("Success", "blur is ok.");  
  
 Utils.*matToBitmap*(dst, selectbp);  
 myImageView.setImageBitmap(selectbp);  
}

处理前后的图片如下图所示：

图 2 均值滤波下的图像（左：原图、右：处理后）

Figure 2 Image under mean filtering (left: original image, right: after processing)



均值滤波可以去除掉图像中的一些突出的噪声点，但是整个图像变得模糊，因为每个像素点的值被平均。

#### 4.2.1.2 高斯滤波

高斯滤波的代码只需要对均值滤波的API接口做改变即可。

代码10 高斯滤波

Code 10 Gaussian filtering

//均值滤波  
//Imgproc.blur(src, dst, new Size(5,5), new Point(-1,-1), Core.BORDER\_DEFAULT);  
//高斯滤波  
Imgproc.*GaussianBlur*(src, dst, new Size(0,0), 5, 0,  
 Core.*BORDER\_DEFAULT*);

处理前后的图片如下页图3所示。

根据图中结果，我们可以看出图像均值滤波，会导致图像出现轮廓和边缘消失的现象，而高斯模糊会产生类似毛玻璃的效果，导致边沿扩展效应明显、图像边缘细节丢失的问题。

图 3 使用高斯滤波的图像（左：原图 右：滤波后）

Figure 3 Image using Gaussian filtering (left: original image right: after filtering)



#### 4.2.1.3 统计排序滤波器滤波

代码11 统计排序滤波器

Code 11 Statistical ranking filter

//中值滤波  
Imgproc.*medianBlur*(src, dst, 5);

//最大最小值滤波（膨胀与腐蚀）  
Mat kernel = Imgproc.*getStructuringElement*(Imgproc.*MORPH\_RECT*, new Size(3,3));  
Imgproc.*dilate*(src, dst, kernel); //最大值滤波  
Imgproc.*erode*(src, dst, kernel); //最小值滤波

处理后对比如下图：



图 4 中值滤波对图像处理（左上：原图，右上：中值滤波，

左下：最大值滤波，右下：最小值滤波）

中值滤波对于图像的椒盐噪声有很明显的抑制作用，是一个很好的图像降噪的滤波函数。

最大最小值滤波除了处理少部分特定类型的噪声，一般用于形态学处理更多而不是图像滤波处理。

#### 4.1.2.4 边缘保留滤波

代码12 边缘保留滤波

Code 12 Edge retention filtering

//高斯双边滤波（程序崩溃了）  
Imgproc.*bilateralFilter*(src, dst, 0 , 100, 15);

//均值迁移滤波(程序崩溃了)  
Imgproc.*pyrMeanShiftFiltering*(src, dst, 10, 50);

程序崩溃了，因此没有得到输出的图像。

# 5. 总结与体会：

本次实验是使用了各种图像的空间滤波算法对图像进行滤波分析，由于没有合适的噪声图像，对普通正常图像的滤波看不出对其中噪声点的抑制，反而由于图像变得模糊而影响原来的效果。

空间域的滤波算法有许多，其中的区别在图中也可见一二，例如均值滤波让图像失去了原本的轮廓，和周围的区别变得不明显，高斯滤波仿佛是给图像蒙上了一层毛玻璃，中值滤波对于图像中变化较快的有明显的抑制作用等，在不同情况下应该是使用不同的滤波算法，就能给图像处理的工作带来便利。

# 6. 参考文献：

[1]郭霖.第一行代码——Android（第2版）[M].人民邮电出版社:北京,2016.12:1-75

[2]贾志刚.OpenCV Android开发实战[M].机械工业出版社:北京,2018.6:6-67