图像处理包括：

1. 浮点转定点 ：RGB2YCbCr
2. 图像增强算法：直方图均衡，对比度增强，GAMA映射
3. 图像降噪：均值滤波，中值滤波，高斯滤波，双边滤波
4. 图像二值化：全局二值化，局部二值化，sobel边缘检测，腐蚀膨胀，帧间差检测
5. 图像锐化：Robert锐化，sobel锐化，laplacian锐化
6. 图像缩放：最近邻插值，双线性插值，双三线性插值，深度学习缩放
7. Lenet5：lenet5FPGA实现

浮点转定点：

RGB模型 ：R : red, G : green, B : blue

YCbCr模型：Y:颜色的明亮度和浓度，Cb:蓝色浓度偏移量 ，Cr:红色浓度偏移量

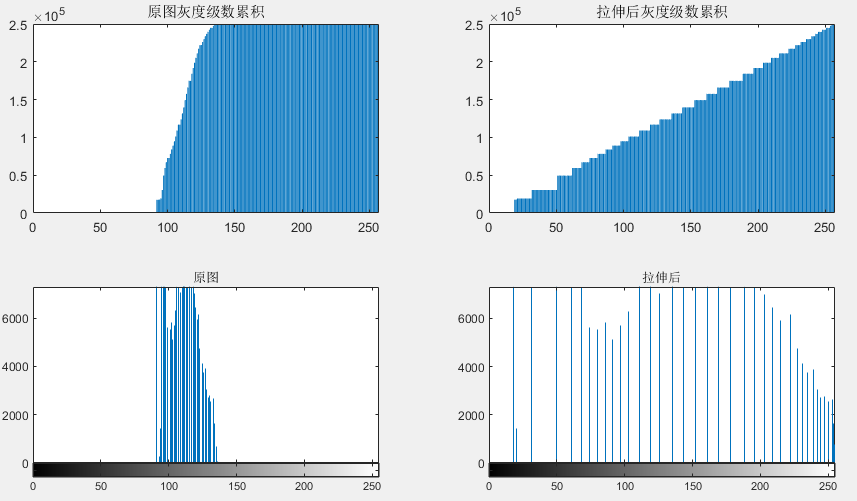
Y = (R\*76+G\*150+B\*29)>>8

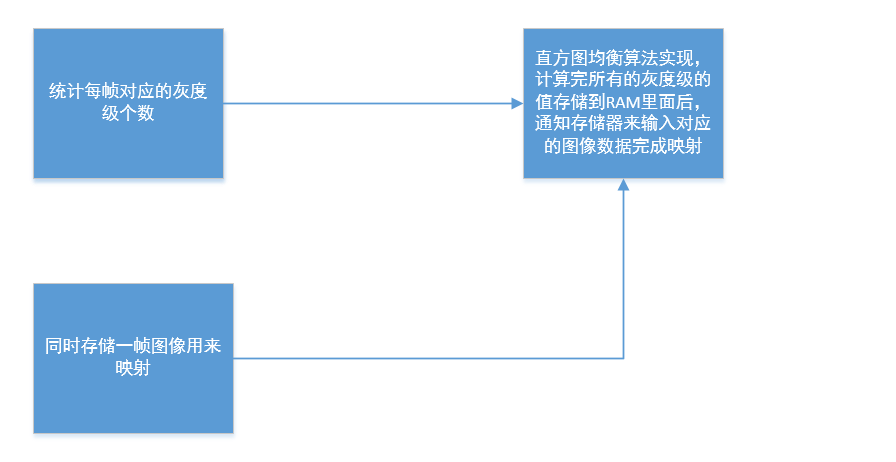
Cb = (-R\*43-G\*84+B\*128+32768)>>8

Cr = (R\*128-G\*107-B\*20+32768)>>8

图像增强算法

直方图均衡：

提高对比度来增强图像的辨识度，直方图均衡通过改变图像的直方图分布，来改变图像中各像素的灰度，用于增强动态范围偏小的图像(具体来说就是灰度值比较集中的图像，横坐标代表灰度级数，纵坐标代表累计数)的对比度，具体实现方法是



1：先统计每个灰度级的个数，输入的灰度级0-255，每来一个数对应的地址的个 数加1

2：进行像素灰度级数累积统计（输出给直方图均衡的算法处理模块，统计完一帧后的每个灰度级对应的频率）

3：对灰度值进行映射 = 归一化+扩大255倍（针对FPGA每帧的输入尺寸是固定的，因此可以提前定点化，例如720\*1280/255）

定点化算法利用MAATLAB实现：

根据以上算法确定 乘数和移位的值 Post\_data = (mg\_data \* m) >> Index;

clc;

a = (1280\*720)/255;

fprintf("a:%.20f\r\n",a);

b = 1/a;

fprintf("b:%.20f\r\n",b);

num = [];

for l=1:32

m = (2^l) / a ;

m = floor(m);

c = m / (2^l);

num(l) = c;

end

d = abs(num -b);

[~,Index] = min(d);

fprintf("Index : %d \n", Index);

m = (2^Index) / a;

m = floor(m);

fprintf("m:%.d\r\n",m);

fprintf("num(%d):%.20f\r\n",Index,num(Index));

a:3614.11764705882342241239

b:0.00027669270833333333

Index : 31

m:594193

num(31):0.00027669267728924751

也就是直方图转换公式 mult\_result= (mg\_data \* m);

将乘法和移位分成两个周期，在第一个周期中得到一个数据宽度为

整数部分：[Index + 7 : index] 小数部分：[Index -1 :0] 的mult\_result；

对其四舍五入：Post\_data = mult\_result[Index+7:index]+ mult\_result[Index-1];

缺点：

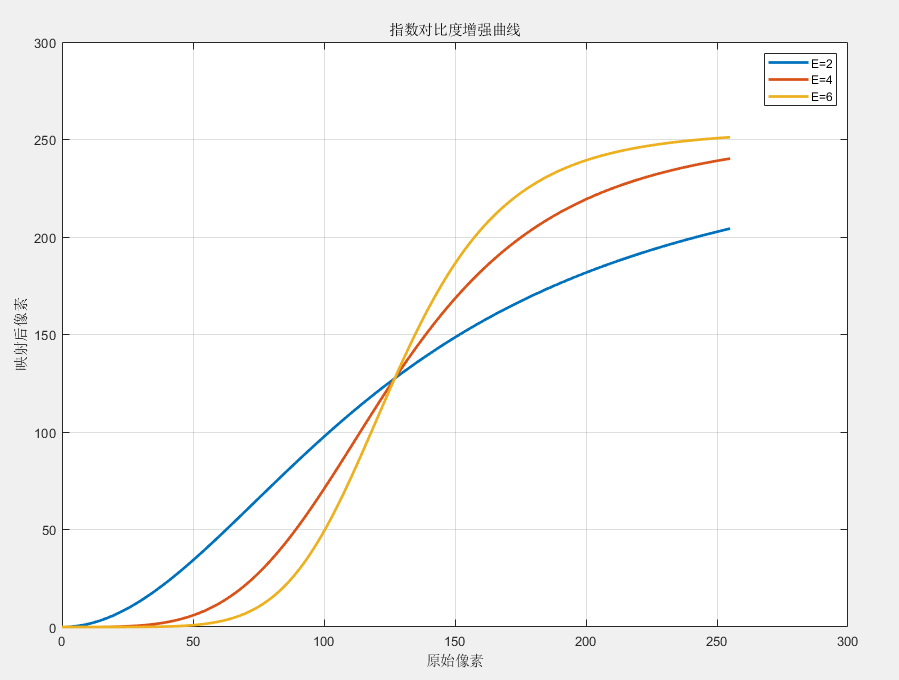
1．直方图均衡后，图像灰度级数减少，部分细节会丢失；

2．对于直方图有高峰时，对比拉伸后将出现对比度不自然的过分增强现象

对比度算法

对比度增强，目的时为了提高明暗之间的差异，从而提高图像对比度，改善主观视觉效果，指数对比度增强的方法有很多，但其核心只有一点，即以一定阈值为中心，提高阈值以上的亮度，降低阈值一下的亮度，白话来说就是让亮的更亮，暗的看起来更暗。

指数对比增强，无论时指数函数，还是各类曲线其本质上就是一种像素映射操作，因为其计算非常耗时，而且在FPGA上也很难实现浮点数的操作，但是当选定好参数后，其结果是固定的，所以可以采用映射的方式来实现。



GAMMA映射

为了适应人眼的亮度变化曲线，对图像传感器产生的图像进行GAMMA变换，来提升图像的辨识度。

在图像处理中 ，像素始终在0-255，所以对公式进行变形

值没有标准也没有对错，不同的LCD屏幕响应曲线各不相同，最正确的方式是通过灰阶找到最佳的GAMMA值，目前经典的gamma值就是2.2

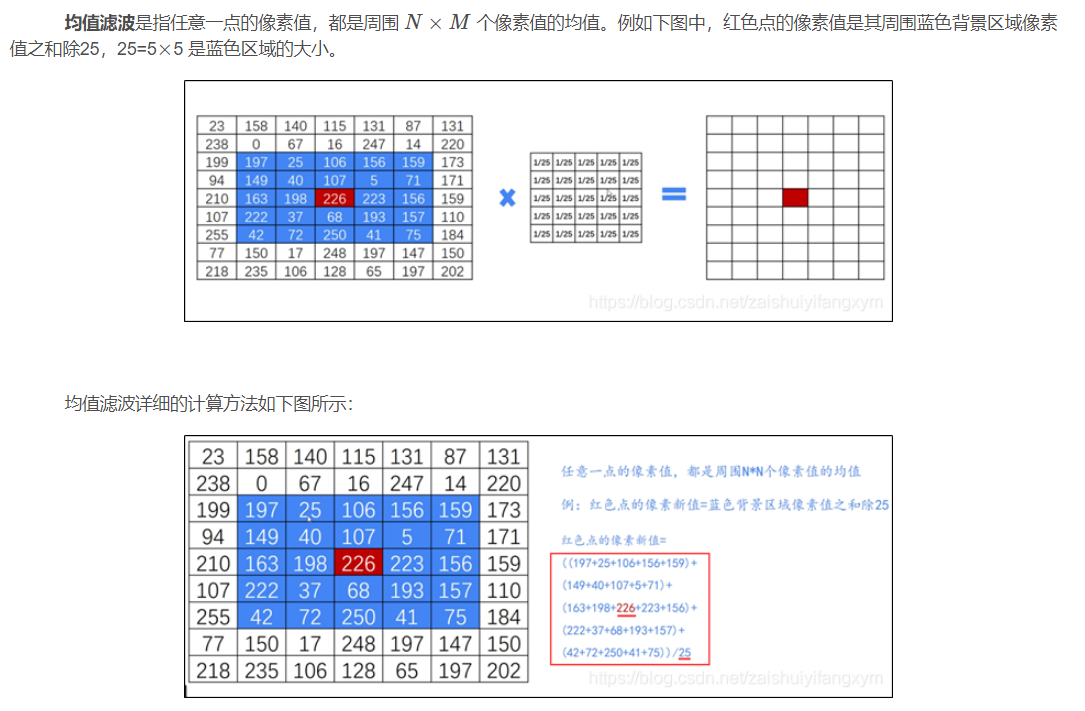
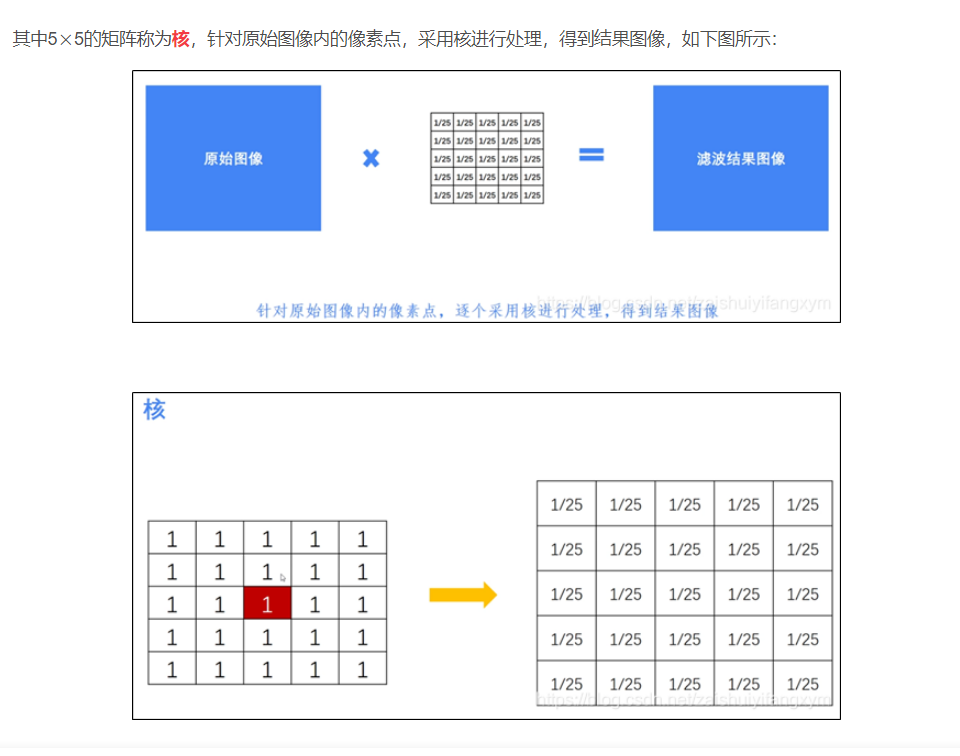
常用图像降噪处理算法

在图像传感器成像的过程中，光电转换及数模放大时，不可避免的会产生噪声，在图像传输过程中也会二次引入噪声。

图像降噪主要分为2D(空域)和3D（时域/多帧）降噪，2D降噪的大部分算法都是基于窗口的卷积运算

均值滤波：

所有滤波算法都是通过当前像素周边的像素，以一定的权重加权计算滤波后的结果。

中值滤波

领域像素有一定的相似性与连续性，那么添加噪声后的像素一定会偏离原值，所以取中间值可以有效的将异常偏离像素过滤掉

与均值滤波相比，均值滤波只是将周围像素进行平均，而中值滤波可以很好的过滤掉异常点

中值滤波的实现方法：

图像进行滑窗处理，获得3x3的窗口

1. 分别对3x3窗口的每行3个像素进行比较，得到3个像素的最大值，中间值和最小值
2. 每行的最大值，中间值，最小值分别比较选出，3行最大值里的最小值，中间值里的中间值，最小值里的最大值
3. 求上一步得出的三个值的中间值，得到中值滤波的结果
4. 判断窗口是否是位于图像边沿，如果是则输出原来的值，否则输出滤波的值

高斯滤波 ：

均值/中值滤波。对于滤波窗口内每个像素的权重都是一样的。但是噪声再图像当中往往比较突出，那么他必然不是平均分布的。

高斯滤波实现方法：

1. 将生成的5\*5的窗口值乘以高斯模板
2. 计算每行5个数值的累加值
3. 计算5行的累加值
4. 四舍五入
5. 判断是否位于图像那个边界

|  |  |  |  |
| --- | --- | --- | --- |
| 算法 | 实现过程 | 优点 | 缺点 |
| 均值滤波 | 窗口内权重相同，直接累加后求平均值 | 计算简单 | 噪声被平均了，边缘被抹平了，清晰度下降 |
| 中值滤波 | 取窗口的中间值作为结果 | 对椒盐噪声等异常突兀噪声处理效果好 | 容易把边缘滤掉。但比均值滤波好点 |
| 高斯滤波 | 窗口内权重按照高斯分布，卷积后得到结果 | 考虑了正态分布的噪声影响，受噪声影响较小 | 没有考虑噪声的相干性，对边缘保护仍然不佳 |

双边滤波：

双边滤波是一种非线性滤波器，它既可以达到降噪平滑，同时又保持边缘的效果，和其他滤波的原理一样，双边滤波也是采用加权平均的方法，用周边像素亮度值的加权平均来代表某个像素的强度，所用的加权平均也是基于高斯分布的

双边滤波实现步骤：

提前：生成高斯模板和相似度权重的查找表（什么是相似度权重？）简单来说就是3\*3的窗口，其他值同中心窗口的差值，根据这个差值利用高斯分布产生的一个相同窗口大小的权重，这个权重和高斯模板的权重是类似的，不同之处高斯模板考虑的是像素空间位置差异对中间值的影响，而相似度权重考虑的是像素值之间的差异对中间值的影响，比如窗口中间值为8f ,那么窗口1.1的值如果为01，其相似度权重就小，如果为90其相似度权重就大 ，由于FPGA不适合浮点和平方运算，所以可以确定好高斯分部的强度后，先计算好对应差值(两个像素的差值肯定是在（0 – 255 ）之间)的相似度权重，然后利用查找表的方式来映射出对应的相似度权重，将相似度权重和高斯空间权重相乘（这里涉及到矩阵的乘法c[I][j] = sum(a [i][k] \*b[k][j] for k = 1 – n ;n为矩阵a的列数,矩阵乘法只有 a的列== b的行数 才能相乘）

1.先生成滑动窗口

2.计算以当前像素为中心的3\*3窗口的内的相似度权重；先分别计算3\*3窗口内的像素与像素中心的差值的绝对值，然后通过查找表获得相似度权重

3.将高斯空间权重和相似度权重相乘（矩阵）得到双边滤波权重，

4.对双边滤波权重做归一化处理

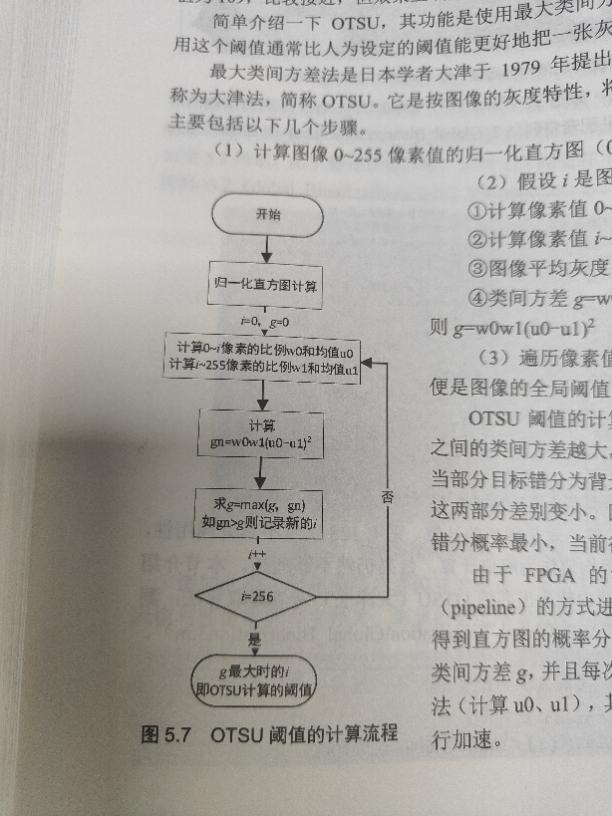
5.将像素窗口同双边滤波权重进行卷积并累加

6.判断是否处于图像边界，选择输出

常用二值化算法

全局二值化

比较简单的就是采用128作为阈值，大于阈值的输出为255，小于阈值的作输出为0，

OTSU（大津法）

局部阈值二值化方法：

1. 计算5\*5窗口中每行5个像素的累加和
2. 计算窗口所有像素的累加和
3. 计算窗口内的均值（直接用均值效果不是很好，乘一个置信度也就是允许多大范围内的浮动），取整后化为二值化阈值
4. 将窗口中心值同阈值做比较，如果小于阈值就输出0，否则输出1（或255）输出1可以减小后续的计算量

Sobel边缘检测：

1.利用sobel算子分别计算水平方向梯度GX\_DATA和垂直方向梯度GY\_DATA

2.将水平方向梯度DX\_dATA和垂直方向GY——DATA合并取模得到最终的梯度G\_dATA;

3.将梯度G\_DATA与阈值THRESH 做比较，大于thresh输出1，否则输出0,图像边沿输出0