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

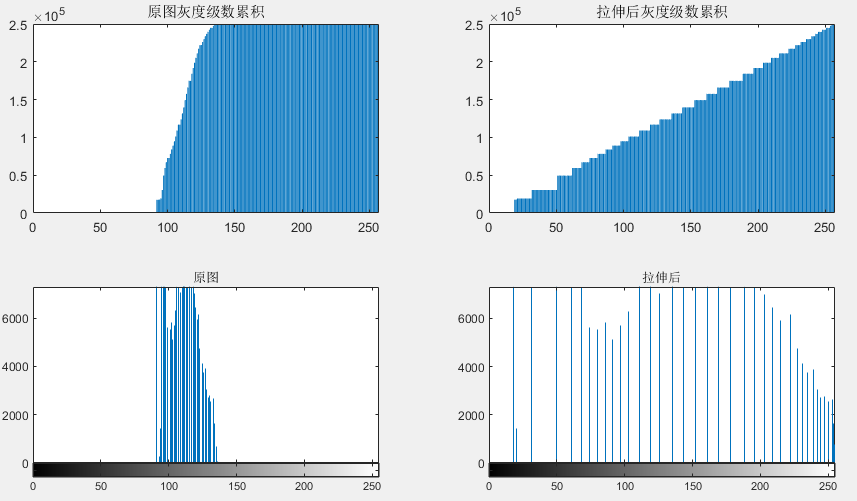
Y = (R\*76+G\*150+B\*29)>>8

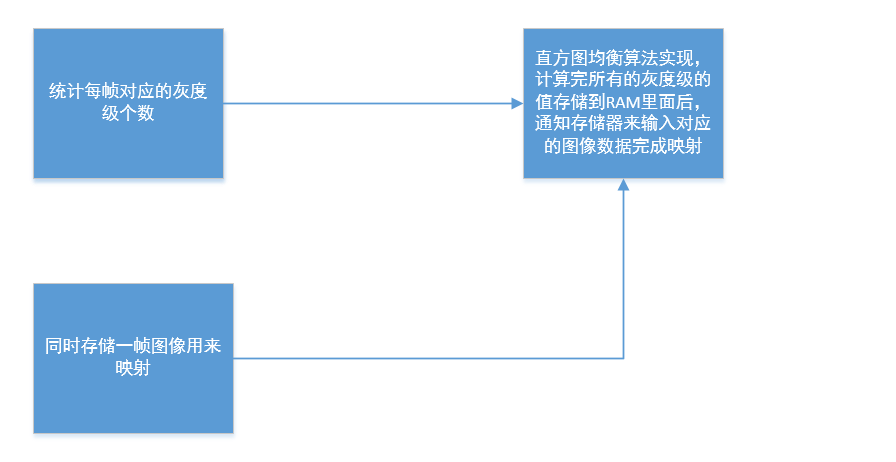
Cb = (-R\*43-G\*84+B\*128+32768)>>8

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：

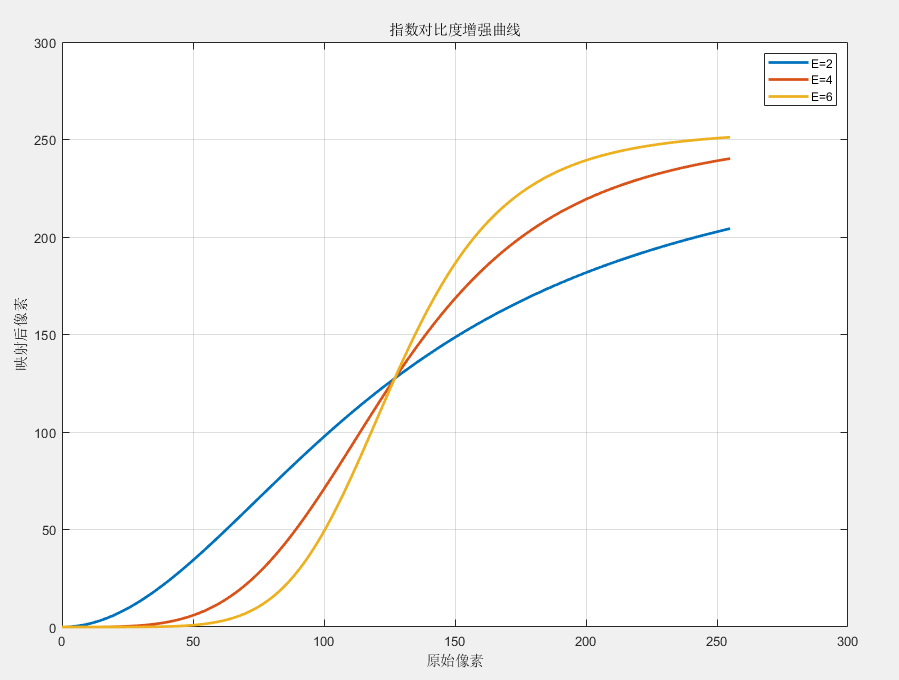
1．直方图均衡后，图像灰度级数减少，部分细节会丢失；

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。

