**Assignment 4**

**Part 1 code**

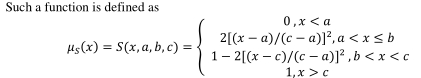
参见 fuzzy.cpp。

**Part 2 explanation**

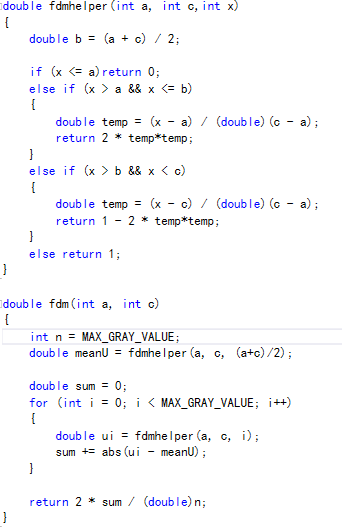
**(1)fdm formula**

这里主要参考的公式就是文章中提到的以下两条式子：





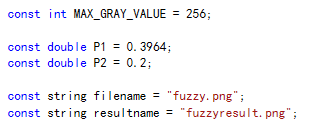
这里是具体实现的代码：



这个fdm函数算的实际上就是fuzzy densitity，用于后面阈值的划分。具体来说就是在Xleft和Xright之间取一个点Xi，用fdm来判断Xi更接近于左边还是右边。

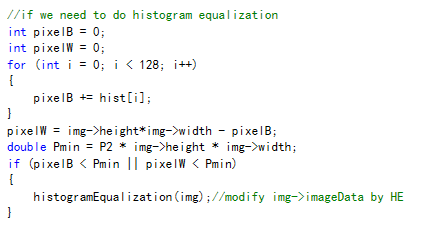
1. **constant variable**

这是代码中设置的常量，MAX\_GRAY\_VALUE表示最大灰度值，P1和P2是设置的百分比。filename和resultname分别是输入和输出图像。最后发现这里的P1和P2取值对结果的影响很大，因为他直接决定了factor、xleft和xright，感觉可以对图片的灰度分布进行分析之后自动的设置P1和P2的取值，可能会比人工设定常数效果好。

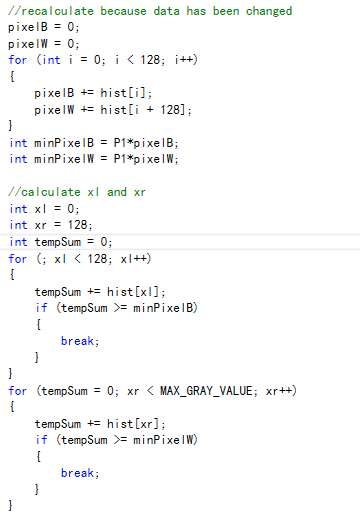


**(3)calculate threshold**

这是P1和P2具体的应用：根据0-128和128-256的灰度频数与P2\*M\*N作比较，来判断是否需要做直方图均衡化，是灰度分布更平均一些。



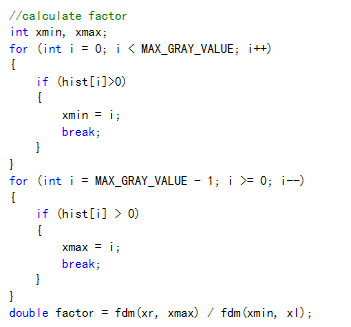
然后计算出minPixelB和minPixelW，用来确定xleft和xright：



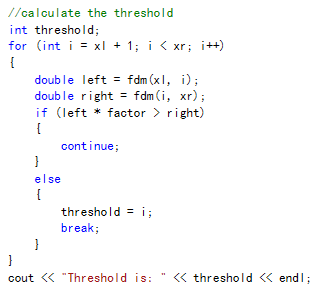
然后计算左/右的比例因子，我看原文的意思好像是直接算0到xleft的fdm1和xright到256的fdm2，然后factor = fdm1 / fdm2。但是这样计算的话，factor很容易偏大或者偏小，导致后面在取阈值的时候，阈值偏大或者偏小，产生的结果不尽人意。

这里我取得是xmin到xleft和xright到xmax的fdm值，xmin是整幅图出现过的最小灰度值，xmax是整幅图出现过的最大灰度值，这样产生的结果比较合理。

这是factor的计算过程：

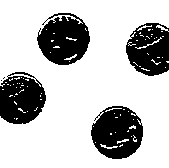


这是计算threshold的过程，从xleft到xright遍历：

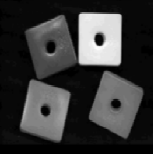
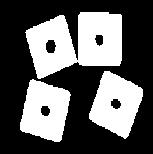


**Part 3 output**

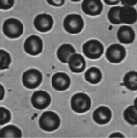
以下是对三幅图进行基于fuzzy set的阈值计算的结果：

（1）

（2）

（3）