VCB-Studio教程15 Clip加减运算与Unsharp Mask

本教程旨在讲述简单的clip运算方法，以及unsharp mask等基于clip加减运算的手段的处理方法。

1. clip的平均数操作

avs中有这么个函数：mt\_average，其作用是将两个clip求“平均数”，即结果的YUV数值，等于两个输入的YUV数值算术平均数：

src = LWLibavVideoSource("pv.mkv")

blur = RemoveGrain(src,20)

mt\_average(src,blur, u=3, v=3) #u=v=3是表示处理UV，否则默认不处理

其效果是把源和RemoveGrain(20)做了个平均数，等于做了个中心像素5/9，周边像素1/18的Blur。（思考题：这个比例怎么得来的？）

另一个模式，RemoveGrain(11)的中心像素权值是1/4，小于5/9，所以如果我们把图像拿来比一比，不难发现其柔化程度（左）是轻于RemoveGrain(11)的（右）：



而RemoveGrain(20)本身比这俩模糊，图就不贴了。

所以数字图像处理中，如果你想得到一个介于a和b之间的效果，最简单的方法之一就是两个clip求一个平均数。

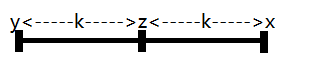
VS中这个函数的实现方式是std.Merge()，默认是50%的融合，即取平均数。

2. Clip的加减运算

如果给大家y和z=(x+y)/2，大家推算x，这算术题谁都会算：

k = z-y, x = z+k = 2\*z-y。

其中，k可以看做是平均数z到y的差距。如果我们把y加上这个差距，我们可以得到平均数z；如果我们在平均数z的基础上继续加上这个差距，我们就得到了x：



在clip操作中，也会有类似问题：如果给你blur=RemoveGrain(src,20)，以及ave=mt\_average(src,blur)，你如何（大致）还原src？答案是通过clip运算：

diff = ave - blur

src = ave + diff

clip运算中，减法其实不太好做。因为avs/vs储存clip的格式，都是无符号整数，或者说是自然数。比如8bit的clip，取值只能是{0,1,2…28-1=255}。而减法很容易做出负数。所以一个补救方式是，做减法的时候，结果加上128（27，16bit下就是32768）：

18-18=128， 18-16 = 130， 18-30 = 116

这样可以很好的解决相差不大的时候，clip的做差。如果clip做差相差很大，加上128之后还是<0或者>255，那么就clamp到0或者255。反正两个相差极大的clip做差一般也没意义，所以这个限制几乎不影响日常使用。

为了配合这个减法，那么加法在求和后减去128也就理所当然了，18+128=16+130=30+116=18。这样保证了下文avs中做法可以有效的实现上文思路：

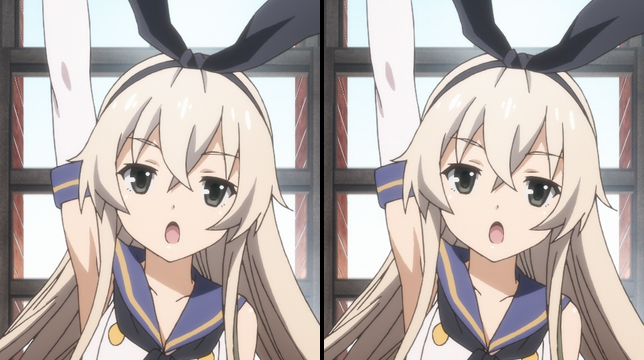
diff = mt\_makediff(ave,blur,u=3,v=3)

src = mt\_adddiff(ave,blur,u=3,v=3)

16bit的实现依赖于dither\_add16和dither\_sub16。注意这两个有个dif参数；默认false，只有在开启=true的时候，才会对结果做offset，否则就是直接做加减（一般我们不期望如此，因为很可能减出负数然后被clamp到0）

vs中使用的是std.MakeDiff和std.MergeDiff

以下是真·原视频和用上述手段还原的。不要问我为啥不说哪个是左边，那个是右边：



3. Unsharp Mask的原理和实现

上文中，我们从一个模糊的图像(Blur)，和一个不怎么模糊的图像(ave)，运算出一个清晰的图像(src)

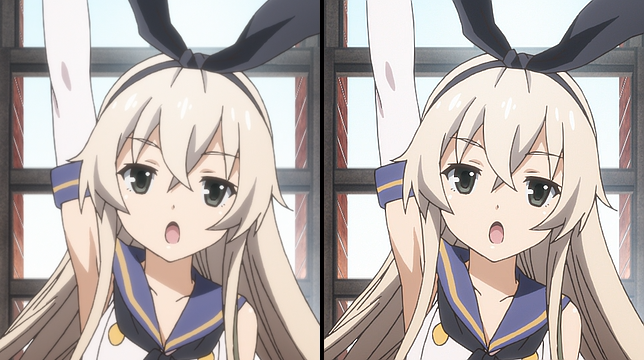
写成函数就是：src = sharp(blur,ave)

如果我们把ave换成src呢？

diff = src - blur

return src + diff

把源和blur的差距，叠加在源上。我们期望获得是一个比源（左）还锐利的图像（右）：



这就是锐化的基本手段：unsharp mask

unsharp，就是blur的近义词，先把画面变得柔和一些；

mask，意思是盖上去，这里指的是把源和柔化做差得到的差异，给叠加到源上。

avs写法：

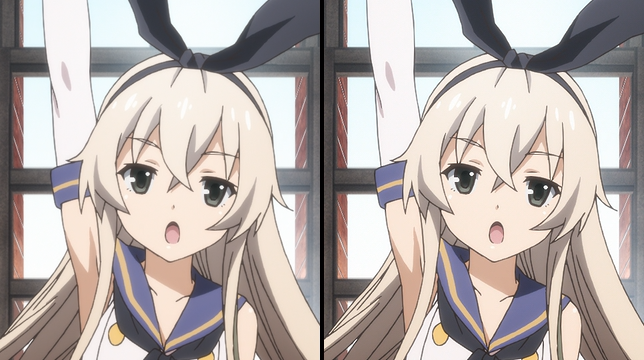
blur = src.removegrain(20)

diff = mt\_makediff(src,blur,u=3,v=3)

mt\_adddiff(src,diff,u=3,v=3)

试着用16bit和vs写写看。

换一个弱一些的blurring kernel，unsharp mask的强化幅度也会减弱，比如我们用GaussianBlur(0.5)做kernel：



Unsharp mask原理简单，实现粗暴，副效果也是很明显的：会带来很多很明显的锯齿和haloing，会把画面原有高频瑕疵放大到很难看。不怕瞎眼的不妨找个DVD源，做个unsharp mask看看效果。

所以实际压片中，几乎不推荐直接使用，除非源本身全局性的模糊，且没有什么线条瑕疵。不过unsharp mask作为几乎所有锐化的核心思路，回头我们还会不断地用到并改善它。

4. SBR的原理和实现

SBR（不要问具体名称，除了创始者Didée应该没人知道这三个字母代表啥）是一种降噪滤镜的设计原理，简单说，对噪点层降噪。表现为：

1. denoised=对src降噪；

2. diff = src - denoised，获得噪点层；

3. refined = 对diff降噪；

4. return denoised+refined，把处理后的噪点层打回去

在avs中，我们用MinBlur(2)作为Denoise Kernel：

denoised = GSMC\_MinBlur(src,r=2)

diff = mt\_makediff(src,denoised,u=3,v=3)

refined = GSMC\_MinBlur(diff,r=2)

mt\_adddiff(denoised, refined, u=3, v=3)

对比一下MinBlur(2)和SBR:



可见SBR是一个依靠破坏性denoiser，来设计保护性denoiser的方法。