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。

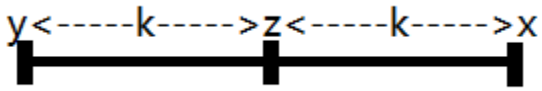
VS 中这个函数的实现方式是 std.Merge()，默认是 50%的融合，即取平均数。

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…2^8-1=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（2^7，16bit 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。