VCB-Studio教程17 LimitDiff的用法与nr-deband

本教程旨在讲述LimitDiff的用法，和nr-deband。

1. LimitDiff的原理

LimitDiff，最早来自avs的Dither Package中的Dither\_limit\_dif16，是一个很实用的工具性滤镜。Dither\_limit\_dif16的用法为：

Dither\_limit\_dif16 (clip flt, clip src, clip ref (undefined), float thr (0.25), float elast (3.0), int y (3), int u (3), int v (3))

通常它的用法是比较flt和src (默认ref设置为src)，将flt限制在src周边的范围内，并且有弹性的调整。thr规定这个范围大小(8bit尺度下)，elast规定这个弹性高低。

**说的简单些**，不妨假定ref=src，即只输入flt和src，并且对于某个像素，src=50。我们规定thr=0.5, elast=2.0。那么无论什么情况，调整后的clip，与src的差距，不会超过0.5。(**注意，这个结论只在elast<=2.0的时候成立)**

如果|flt-src|<=thr，直接保留。比如说flt=49.8或者50.4，那么不做调整；

如果|flt-src|>thr\*elast，直接取值src。比如说flt=48.9或者51.7，那么直接设置为50；

否则，结果会被弹性的调整到[src-thr, src+thr]之间。比如flt=49.1，调整结果大约为49.8，flt=49.3，调整结果大约为49.6，flt=50.6，调整结果大约为50.5，flt=50.9，调整结果大约为50.2。

整体上，图像显示出这样的趋势（注意49和51处拐角是excel插值出的"ringing"（overshot），而曲线最大值和最小值为50.5和49.5）：

接下来我们来**严格**地说一下Dither\_limit\_dif16的计算原理：

令$dif=flt-src, dif\_{ref}=flt-ref$, 即flt分别和src与ref做差；

$thr^{'}=thr\*elast$, 即上下最多允许的调整范围

如果$\left|dif\_{ref}\right|\leq thr$, 则$res=flt $; 如果$\left|dif\_{ref}\right|\geq thr$, 则$res=src $; 否则$res=src+dif×(thr^{'}-\left|dif\_{ref}\right|)/(thr^{'}-thr)$

思考题：证明，当$src=ref $的前提下，$\left|res-src\right|>thr$ 是 $elast>2 $的充分条件。

如果我们将thr设置为0.4，elast设置为3，上面的曲线变为这样：

可见，调整后的范围，其实是可以略微超过0.4的。这个允许超过的幅度，随着elast增加而增加。

vs中的写法是mvf.LimitFilter()

2. LimitDiff用于限制线条处理

LimitDiff本身就是为了限制处理后的clip，与src相比差异太大而产生的。我们可以用它来限制一系列可能力度过大的操作，比如说UnsharpMask(11)，我们接一个LimitDiff(thr=3.0, elast=4.0):



结果是非常让人满意的。主要锐利的线条都避开了锐化，锐化只有在眼睛，头巾，窗户等地方作用明显，即达到对纹理锐化的效果，在没有改变片源画风、不引入大量瑕疵的情况下，优化了目视观感。这是除了Moderate Sharpening使用Repair之外，另一种更好、更有效的限制Unsharp Mask副作用的方法。

类似的，LimitDiff还可以用来限制其他线条处理，比如说收线、加黑等，也可以用于aa/dering之后的控制破坏力度。通常，用于线条的时候，thr和elast取值都会比较大。

3. LimitDiff用于限制平面处理——nr-deband

除了用于限制线条，LimitDiff也用来被限制非线条。事实上，它在Dither Package中的作用，就是作为deband滤镜SmoothGrad和GradFun3的限制。SmoothGrad的做法就是：

1. 大范围的16bit Smooth，连细节带色带一并给抹了

2. 通过Dither\_limit\_dif16，把抹平了的线条给救回来

GradFun3的做法更细致一些，效果更好，破坏力也小得多。GradFun3中，默认thr=0.35，elast=3.0，已经是比较强的deband了。

除了GradFun3，另一个常用的deband滤镜是flash3kyuu\_deband，简称f3kdb（doc: <https://f3kdb.readthedocs.io/en/latest/usage.html>）

一般来说，f3kdb在使用的时候，如同f3kdb(16,48,32,32,0,0)这样连续输入6个数字，意义分别为：

range=16，这是smoothing的范围，越大破坏力越强

Y=48, Cb=32, Cr=32，这是不同平面banding检测阀值，或者说是deband强度。

GrainY=0，GrainC=0，这是加噪点力度。以前加噪点对保留deband效果非常重要，现在一般不推荐在deband的时候加噪点。

avs里，通常用input\_mode=1/output\_mode=1来指定输入输出是stacked 16bit；vs里只需要用output\_depth=16来指定输出精度（因为输入精度可以自动判断）

结合GradFun3的思想，我们可以利用两者的特长：用f3kdb优秀的smoothing kernel，配合LimitDiff有效的保护：

src16

f3kdb(8,64,48,48,0,0,input\_mode=1,output\_mode=1)

f3kdb(16,48,32,32,0,0,input\_mode=1,output\_mode=1)

Dither\_limit\_dif16(src16, thr=0.4, thrc=0.3, elast=3.0)

就是2pass f3kdb做smoothing，然后接Dither\_limit\_dif16做限制。一般来说，f3kdb强度往高给，limit\_dif的强度往低给，这样可以比较好的实现 平滑-限制 的组合机制。

虽然加噪可以防止色带，但是给你的源，有噪点不意味着没有色带——制作的时候，很可能是在有色带的源上加噪点，让噪点下面还盖着色带效果。这时候强行deband势必需要开极高的强度才可以抹平，而且deband后，线条破坏很大，噪点也损失殆尽，源和成品画风上产生较大差异。我们希望改善一下这种情况：先把噪点抠出来，对平面部分deband，再把噪点打回去。用vs写法如下：

nr16 = core.rgvs.RemoveGrain(src16,[20,11])

noise16 = core.std.MakeDiff(src16,nr16)

db = core.f3kdb.Deband(nr16,8,64,48,48,0,0,output\_depth=16)

db = core.f3kdb.Deband(dbed,16,48,32,32,0,0,output\_depth=16)

db = mvf.LimitFilter(db, src16, thr=0.4, thrc=0.3, elast=3.0)

res = core.std.MergeDiff(db, noise16)

上文的实现中，nr16是用RemoveGrain(20,11)做noise dumper的结果。使用的降噪通常只需要足够的强度，同时速度较快。另外，低精度或者MedianBlur性质的NoiseDumper并不适合，容易导致噪点层不平滑，导致噪点加回去后保留部分色带pattern。

比较合理的NoiseDumper包括小范围RemoveGrain/MinBlur/SBR/Bilateral。 KNLMeansCL不适合在重色带场景做noisedumper。排除掉噪点干扰，deband的强度就可以给的比较小了。

这种通过Blur滤镜（又称为LowPass滤镜，即低频部分才可以无伤通过）区分噪点层（高频信息）和降噪后（低频信息），然后分别对高频和低频处理的方法，在预处理中很常用。类似GSMC等滤镜都有相似的手段。

4. LimitDiff用于融合线条和非线条——nnedi3\_resize16

在avs中，一些针对线条处理的滤镜，比如nnedi3/toon等，运算精度只有8bit。处理线条时候这不是问题，但是处理平面，就有可能产生精度问题。我们希望它们的效果只作用于线条，平面部分采用原生高精度的方案。典型的比如用nnedi3做拉升：

src16 = …

src8 = src16.ditherpost()

edge = src8.upscale\_by\_nnedi3.U16() #用nnedi3做拉升，在线条区域表现良好，但是平面区域精度不足

nonedge = dither\_resize16 (taps=4) #用16bit spline64做拉升，线条区域表现不佳，但是平面区域精度优秀

如何把edge和nonedge各取所长的融合呢？

一种方式是利用masktools，计算出线条区域部分，然后根据这个信息进行融合；

另一种捷径是利用线条部分数值差异远大于平面部分：当edge和nonedge相差不大的时候，选nonedge，否则选edge:

Dither\_limit\_dif16(nonedge, edge, thr=1.0, elast=1.5)

thr越大，越多的地方被判定为nonedge，否则相反；1.0左右的值保证了当被判定为nonedge的时候，edge和nonedge数值差距在抖动噪声范围级别（<1.0）

elast越大，越多的地方将从edge和nonedge中融合获取，融合越平滑。1.5~2.0左右的值已经很合理。

能用Dither\_limit\_diff16的场合是，线条部分显著区别，平面部分除了精度几乎没有区别。事实证明这样做比mask/merge速度要快一些，写法和调节也相对简单。