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。

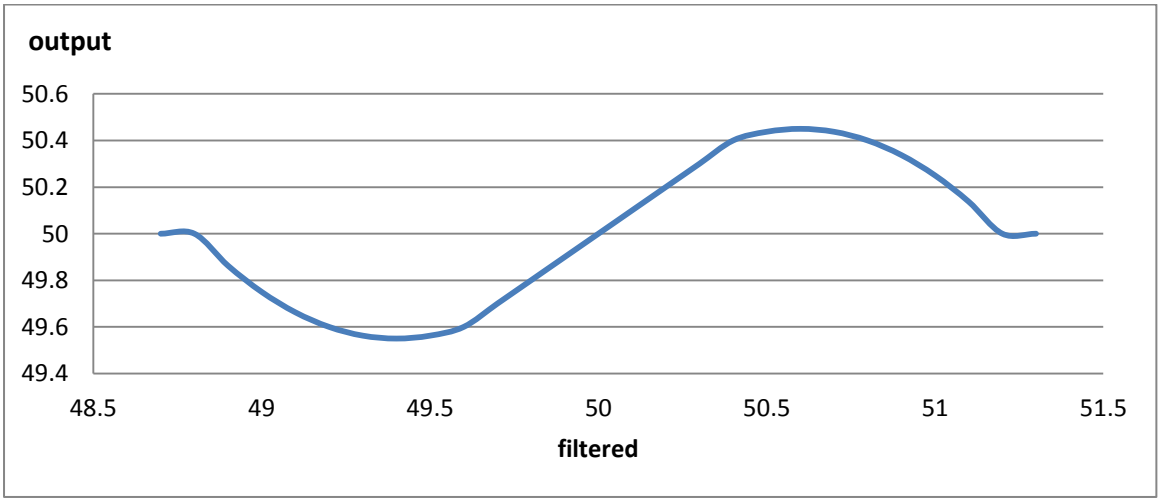
说的简单些，不妨假定 ref=src，即只输入 flt 和 src，并且对于某个像素，src=50。我们规定 thr=0.5, elast=2.0。那么无论什么情况，调整后的 clip，与 src 的差距，不会超过 0.5。(**注意，这个结论只在 elast<=2.0 的时候成立)**

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：

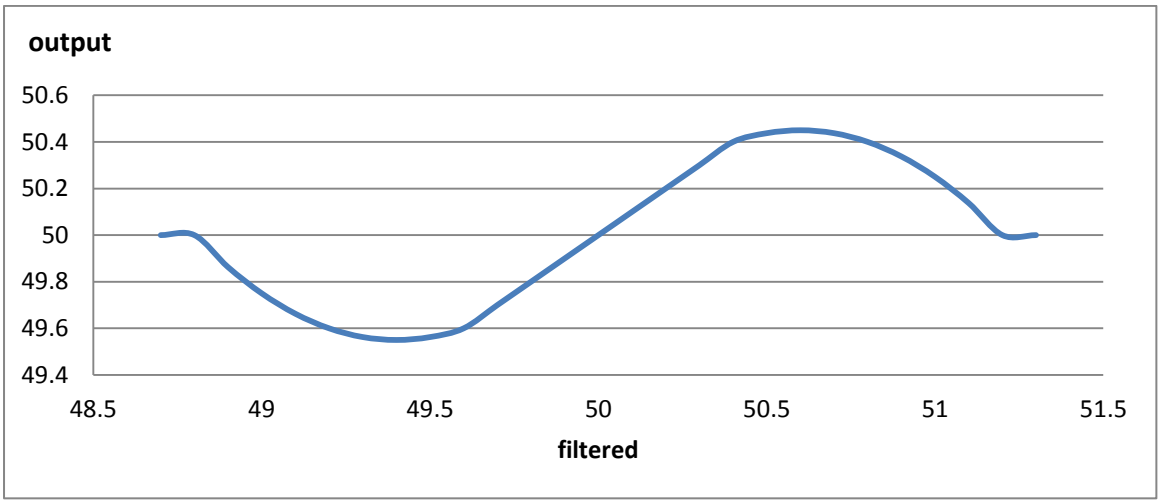
令$\mathrm{dif} = \mathrm{flt} - \mathrm{src}, \mathrm{dif}_{\mathrm{ref}} = \mathrm{flt} - \mathrm{ref}$, 即 flt 分别和 src 与 ref 做差；

$\mathrm{thr}' = \mathrm{thr} * \mathrm{elast}$, 即上下最多允许的调整范围

如果$|\mathrm{dif}_{\mathrm{ref}}| \le \mathrm{thr}$, 则$\mathrm{res} = \mathrm{flt}$；如果$|\mathrm{dif}_{\mathrm{ref}}| \ge \mathrm{thr}$, 则$\mathrm{res} = \mathrm{src}$；否则$\mathrm{res} = \mathrm{src} + \mathrm{dif} \times (\mathrm{thr}' - |\mathrm{dif}_{\mathrm{ref}}|)/(\mathrm{thr}' - \mathrm{thr})$

思考题：证明，当$\mathrm{src} = \mathrm{ref}$的前提下，$|\mathrm{res} - \mathrm{src}| > \mathrm{thr}$ 是 $\mathrm{elast} > 2$ 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。