VCB-Studio 教程 20: MaskTools (1)

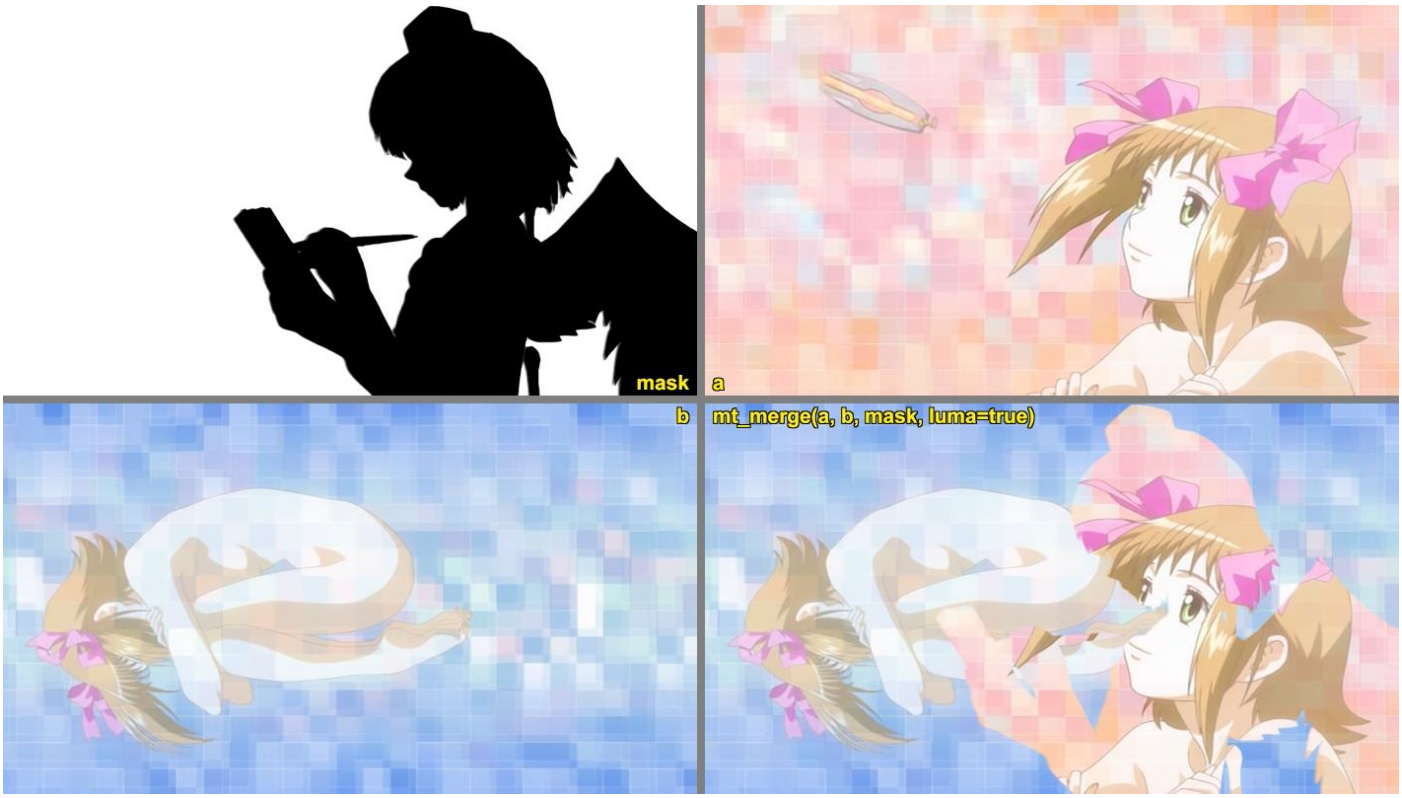
本教程旨在讲述 masktools 最基本的思路和用法。

1. MaskTools 的基本用法

MaskTools 的基本用法是 mt_merge(clipa,clipb,bweight)，它通过 bweight 作为权值，来把 clipa 和 clipb 做像素级别的融合：

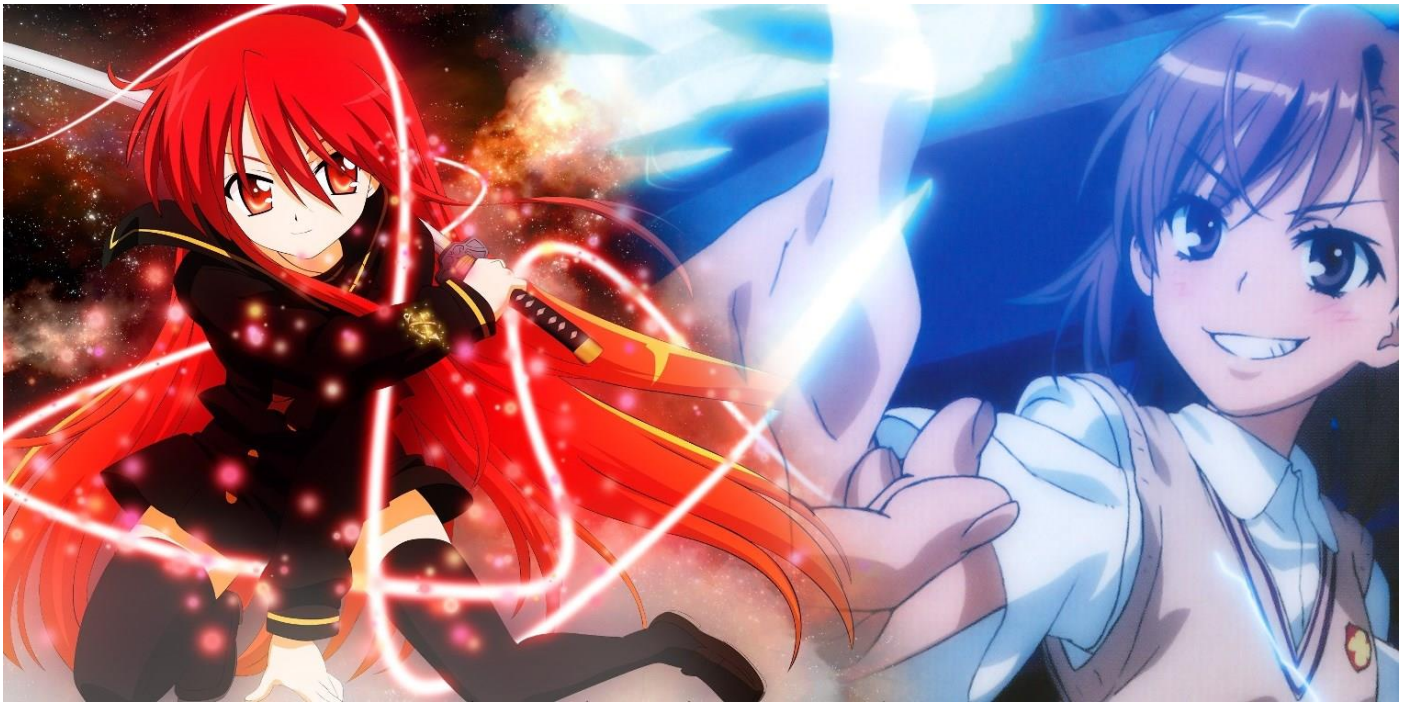
clipa, b 是两个待融合的 clip，比如下文的 a 和 b，成品的每一个像素，都由 a 和 b 对应位置像素加权融合；

bweight 每个像素，决定了输出对应位置，clipb 的权值。在 8bit 下，这个权值就是 bweight/256；clipa 的权值是 1-bweight/256。为了简化理解，我们不妨先来看 bweight 是 0/255 两种情况，这样加权出来只有两种可能——要么完全是 clipb，要么（几乎）完全是 clipa：



mask 那图，黑色部分，即射命丸文的框架，数值为 0，这部分 clipb 的权值是 0，在成品中完全取自 clipa；否则，白色部分，对应值是 255，clipb 的取值是 255/256≈1，图像中取自 clipb。

如果 clipb 的值介于 0-255 中间，那么图像就会根据 ab 加权平均。如果权值变动从 0-255 是平滑渐变的，这样可以起到自然地过渡的效果，比如下图的中间部分就是典型的加权融合：



如果让夏娜那张作为 clipb，炮姐那张作为 clipa，那么 bweight 大概是这样：

mt_merge 除了这三个参数外，还有 Y/U/V 三个参数来控制三个平面的计算。vs 中这个通过 planes=[0,1,2]来控制。avs 版还有个参数 luma，当 luma=true 的时候，强制 Y=U=V=3，且三个平面均使用 bweight 的 luma 平面来计算权值。vs 版对应的参数是 first_plane = true，表示是否使用 bweight 的第一个平面计算权值。不过 vs 中开启 first_plane = true 并不会强制三个平面都计算，而是每一个参与运算的平面都适用统一的权值。

avs 中，16bit 版本的是 Dither_merge16 和 Dither_merge16_8，后者表示 mask 是 8bit。

2. MaskTools 的计算细节(选读)

假设在 8bit 精度下，MaskTools 的计算过程是这样的：

res = round[(256-bweight)/256 * clipa + bweight/256 * clipb]

实际上，masktools 是做不到让 clipa 的权值 0%的。不过这不是什么问题：clipa 的权值最少可以是 1/256；clipa 的最值是 255，那么造成的计算误差<=255/256<1，rounding 后最多产生 1 的误差，不是什么大问题。

ps：如果 clipa 全是 x，clipb 全是 0。我希望 clipa 的权值是 0%，即图像全部取 clipb=0。实际上 clipa 权值是 1/256,那么 clipa 是多少时候，图像结果出来会是 1 呢？

答：round(x/256) = 1， x>=128。也就是当某个像素，|clipa-clipb|≥128 的时候，才有可能造成 1 的误差。

浮点数运算是很拖累效率的；我们把公式变动一下：

res = round{[(256-bweight) * clipa + bweight * clipb]/256}

这样，计算(256-bweight) * clipa + bweight * clipb，纯粹是整数运算。

接下来，我们把四舍五入(rouding)转为截位（truncating），不过我们需要手动给结果+0.5，因为 round(a) = trunc(a+0.5):

res = trunc{[(256-bweight) * clipa + bweight * clipb]/256 + 0.5}

再变形一下：

res = trunc{[(256-bweight) * clipa + bweight * clipb + 128]/256}

现在我们有个很满意的结果： 对于整数 a 和 b，c=a/b 记录精确结果，并 trunc(c)，结果就是多数编程语言中的 a/b (整数除法)

所以，我们可以把 trunc 彻底拿去了：

res = [(256-bweight) * clipa + bweight * clipb + 128]/256

这就是 taro 的教程中给的表达式。我们不妨验证一下上文的问题：

如果 clipa 全是 x，clipb 全是 0。我希望 clipa 的权值是 0%，即图像全部取 clipb=0。实际上 clipa 权值是 1/256,那么 clipa 是多少时候，图像结果出来会是 1 呢？

[1 * x + 128]/256=1

x+128 >= 256, x >= 128

因此，前后结果是完全一致的。

3. MaskTools 用于 edge 和 nonedge 融合

显然，MaskTools 在预处理中的作用可不是用来融合两张毫不相干的图片的。MaskTools 的作用主要是区分 edge 和 nonedge，然后把它们融合。简单说：

edge = src.sharp() #锐化

nonedge = src.deband() #去色带

edgemask = src.edgemask() #edgemask 假设是一个可以生成 mask 的函数，结果在线条区域为 255，否则为 0

mt_merge(nonedg,edge,edgemask) # nonedg,edge,edgemask 记住顺序，这样使用的时候你不用去多想这三个 clip 的顺序，而是很自然的记住逻辑：平面，线条，线条 mask

生成 edgemask，一个简单的方法是：

mask = src8.tcannyMod(mode=1,sigma=1.2,sobel=true)

mask = mask.mt_binarize(3).mt_expand().mt_expand().mt_inpand().removeGrain(20,-1)

tcannyMod(mode=1,sigma=1.2,sobel=true)，是先用 GaussianBlur(1.2)对图像做一个 Blur，然后结合 sobel 算子，用梯度判断线条。出来的图像每个像素介于 0-255 之间，越高表明“线条指数”越强。这个计算只涉及 Y 平面，因为 UV 的信息一般较少，不如直接用 Y 的信息来引导 UV。

随后，我们给定一个阀值，比如 3，大于这个阀值的我们判定为 edge，否则判定为 nonedge。这样，我们通过 Binarize 这个操作，将图像二值化（0/255）。avs 中的 mt_binarize(x)是将图像≥ x的地方设置为 255，而 vs 中 std.Binarize(x)则是将图像>x 的地方设置为 255，还是有点区别的。有兴趣的不妨这时候输出看看，这时候输出的 mask，通常高亮于线条的两边，线条的中间反而是空的。这是因为出现坡度的地方其实不是线条中心。所以，我们通过 expand 操作和 blur 操作来后续加工：

expand：图像中每个像素，替换为周遭 9 个像素的最大值。用在 mask 上的效果就是 mask 向外扩张一个像素，很适合用来填补“空心线”。它的逆操作是 inpand，每个像素替换为最小值。在多次 expand 基本填满空心线条之后，就用 inpand 给外围长得过肥的 mask“瘦身”

类似的滤镜有 mt_infate 和 mt_deflate ,不过这个是把中心像素替换为周遭像素的平均值 ,如果平均值更大。deflate 则是当平均值更小的时候替换。

RemoveGrain(20)作 blur 用，将 edge 和 nonedge 的边缘 smooth 一下。

这就是一个简单的 0/1 mask 设计（0/1 表示 edge 的判定基本上只有 edge 和非 edge 两种判断，而不带太多“过渡”）。一般来说，如果你想调整框出来多少，你可以调整 tcannymod 里 sigma，可以调整阀值，还可以通过 expand/inpand/blur 的后加工来实现。

更多 mask 设计的方法和思路我们以后还会说。

用 avs 的 16bit 实现，将上述表达出来：

```
mask = src8.tcannyMod(mode=1,sigma=1.2,sobel=true)
mask = mask.mt_binarize(3).mt_expand().mt_expand().mt_inpand().removeGrain(20,-1)
nr16 = Dither_removegrain16(src16,20,11)
```

diff = dither_sub16(src16, nr16, dif=true) # nr16 和 diff 分别代表低频和高频信息
edge = dither_add16(src16,diff, dif=true)
edge = dither_limit_dif16(edge,src16,thr=4.0,thrc=3.0,elast=3.0) #unsharp mask 后接 limitdiff
nr16.f3kdb(8, 48 32, 32, 0, 0, input_mode=1, output_mode=1)
f3kdb(16, 32, 24, 24, 0, 0, input_mode=1, output_mode=1)
nonedge = Dither_limit_dif16(src16, thr=0.35, elast=2.5) #nr-deband 后接 limitdiff
dither_merge16_8(nonedge,edge,mask,luma=true) #合并

自己试着用 vs 写一写。

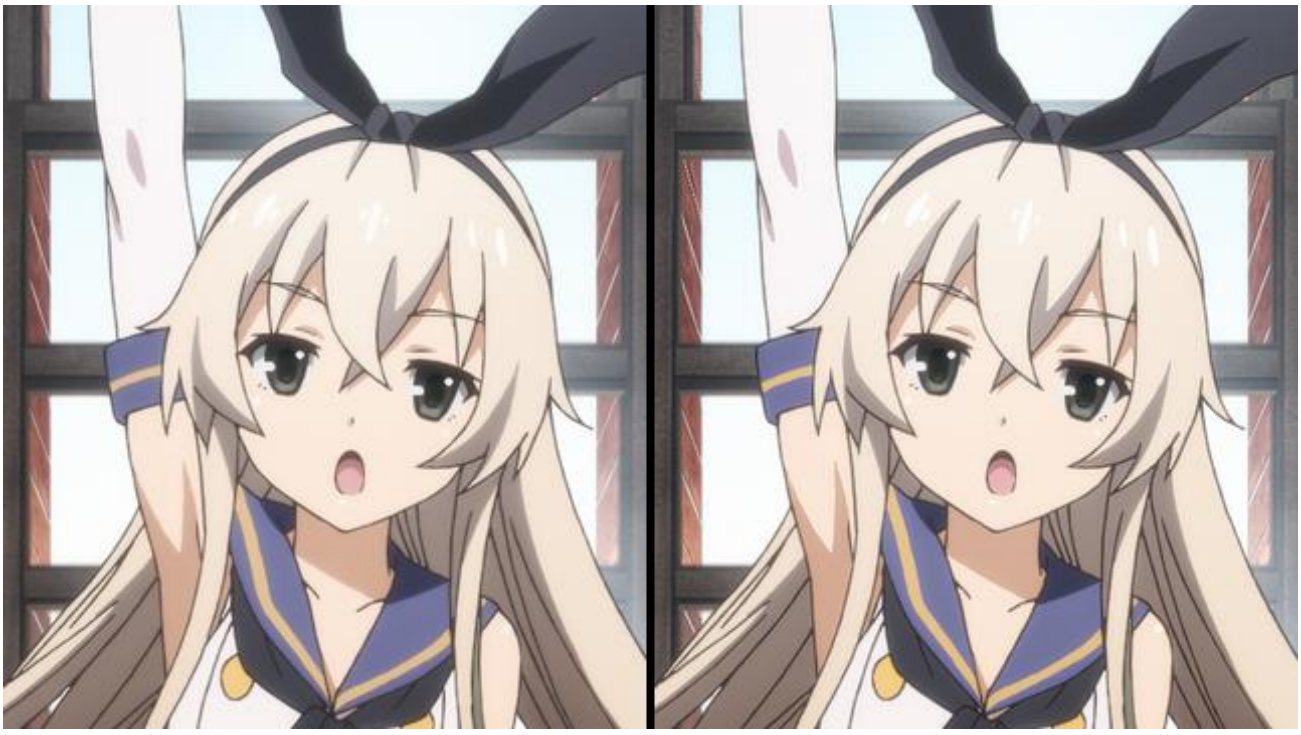
4. MaskTools 用于自适应 edge 处理

除了融合 edge 与 nonedge，masktools 还可以利用连续性 mask(区别于 0/1mask)传达的“锐利度”，来对图像做自适应的处理。比如说 unsharp mask ,我们希望限制它在最锐利的地方强度 ,除了之前说的 Repair 和 LimitDiff，还有一种方法就是利用 mask 限制：

利用 tcannymod 来评估图像中锐利程度，数值越高表明越锐利；
越锐利的地方，锐化的强度越小，也就是锐化后的图像比例越低。

```
mask = src8.tcannyMod(mode=1,sigma=0.5,sobel=true,chroma=1) #开启 chroma 处理
mask = mask.mt_expand().mt_expand().mt_inpand().removeGrain(20,-1) #这次不做二值化，而直接利用其连续性。
nr16 = Dither_removegrain16(src16,20,11)
diff = dither_sub16(src16, nr16, dif=true) # nr16 和 diff 分别代表低频和高频信息
edge = dither_add16(src16,diff, dif=true)
Dither_merge16_8(edge, src16, mask, luma=false) #YUV 三个平面各自用自己的评估来锐化。注意 ,这次 edge 放在最前面，因为设计中，mask 值越低的地方，锐化后的 edge 比例越高。
```

效果如下：



思考：为什么锐化的时候，不采用 luma 平面算 mask，然后用 luma=true，让三个平面都采用 luma 生成的 mask 信息？