VCB-Studio 教程 14 Blur/柔化

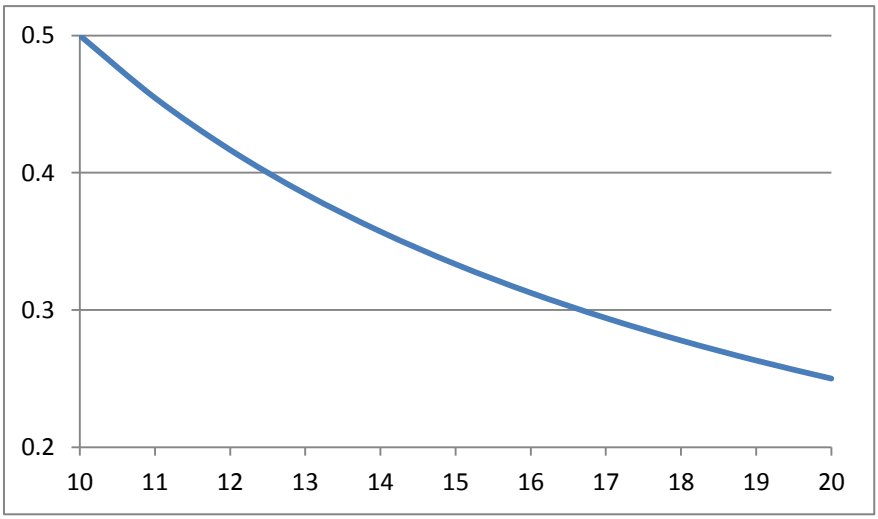
本教程旨在讲述 AverageBlur 和 MedianBlur 两种 Blur 思想，并详细讲解 RemoveGrain 的几种常见用法。

0. Blur 的原理

~~总结为一句话就是：允许你装逼，但是不要太过分。~~

数字图像中，图像的高频信息，比如线条，比如噪点，都是体现在数字比邻域大（或者小）。比如说我们来看一维的图像：

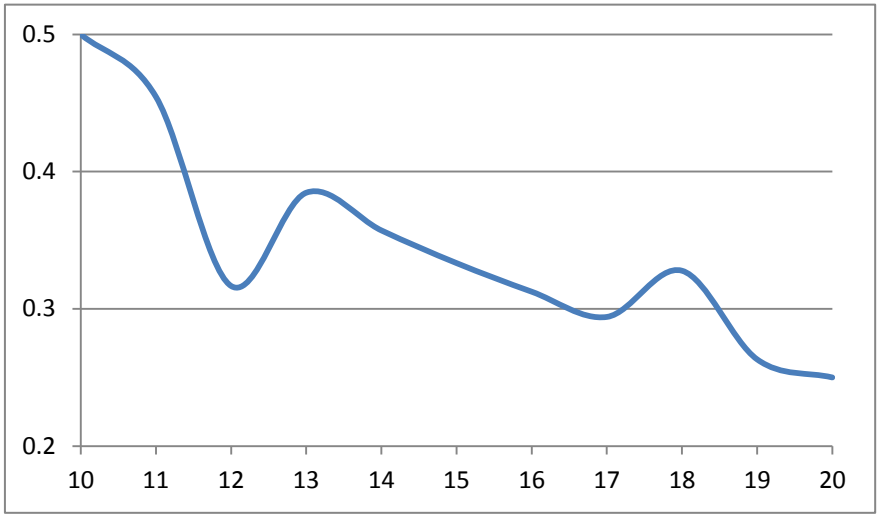
0.5
0.454545
0.416667
0.384615
0.357143
0.333333
0.3125
0.294118
0.277778
0.263158
0.25



其实是 $y=\frac{5}{x}, x=10,11,...20$ 生成的。我们把它画成曲线，得到一条光滑的线。

现在，我们在第 3 个和倒数第 3 个数字的地方，分别添加-0.1 和 0.05 的噪点：

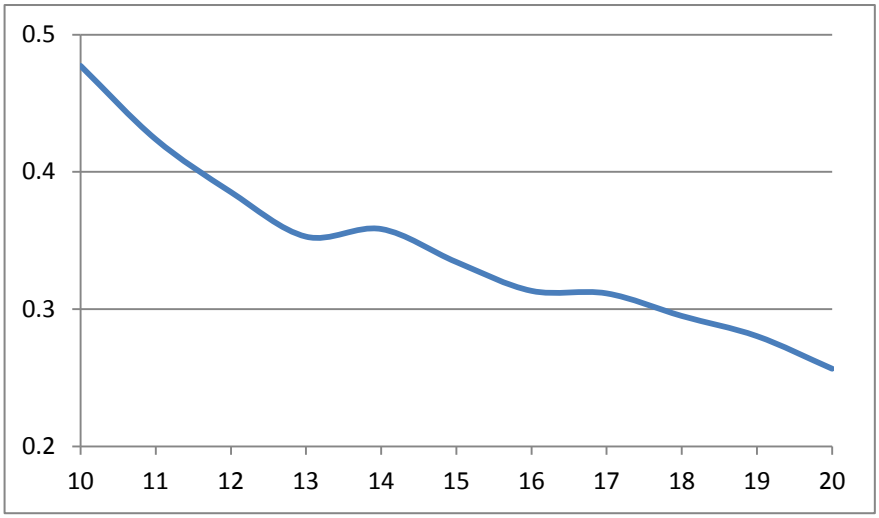
0.5
0.454545
0.316667
0.384615
0.357143
0.333333
0.3125
0.294118
0.327778
0.263158
0.25



可以看出，原本平滑的图像出现了峰谷。从数字上看，x=12 和 x=18 处，其数字明显与左右邻域大小不合，这就是高频信息产生的原因。

如果我们想平滑/模糊/blur 这个曲线，有两个办法：一个是每个数换成它邻域内的平均数，还有一个是换成领域内的中位数。我们先来尝试一下平均数：

0.477273
0.423737
0.385276
0.352808
0.358364
0.334325
0.313317
0.311465
0.295018
0.280312
0.256579

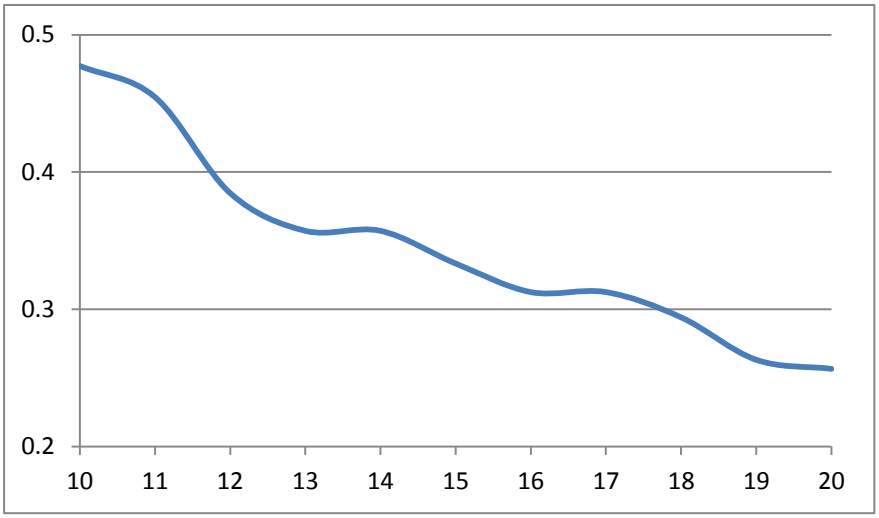


首末的值是（原来的值+原来旁边的值）/2，比如 0.477273=(0.5+0.454545)/2；

中间的值是（原来的值+原来上边的值+原来下边的值）/3，比如 0.423737=(0.5+0.454545+0.3166667)/3。从图上看，似乎 blur 的效果还是比较明显的，原来波动的效果平滑了很多。这种方式叫做 AverageBlur

我们再来试试中位数（如果首末，则取平均数。这也是两个数中位数计算方法）

0.477273
0.454545
0.384615
0.357143
0.357143
0.333333
0.3125
0.3125
0.294118
0.263158
0.256579



同样，平滑的效果很明显。这种操作称为 MedianBlur。

总结以上两种 Blur 的作用，就是将素数值经过一番变换，使得在邻域内很突出（值很高，或者很低）的像素，值不再那么突出。这样，图像可以变得平滑、柔和、模糊。

~~某像素：我不是针对谁，我是说在我旁边的各位，都太 low 了~~

~~Blur 滤镜：允许你装逼，但是不要太过分~~

1. AverageBlur 的三种实现——RemoveGrain(11,19 和 20)

回到二维平面 ,我们假定对 YUV/RGB 三个平面采用统一的策略。RemoveGrain 作为一个泛用性的空间 Blur 滤镜，实现了很多种 Blur 策略，通过不同的 mode 指定。一种常用的 Blur 策略是，每一个像素，换成 3x3 领域内（它本身和周边 8 个像素）的平均值。这就是 RemoveGrain(20)：

用法（以 avs 16bit 为例）:

LWLibavVideoSource("pv.mkv",threads=1,format="yuv420p16",stacked=true)

Dither_removeGrain16(20) #输入输出都是 yuv420p16

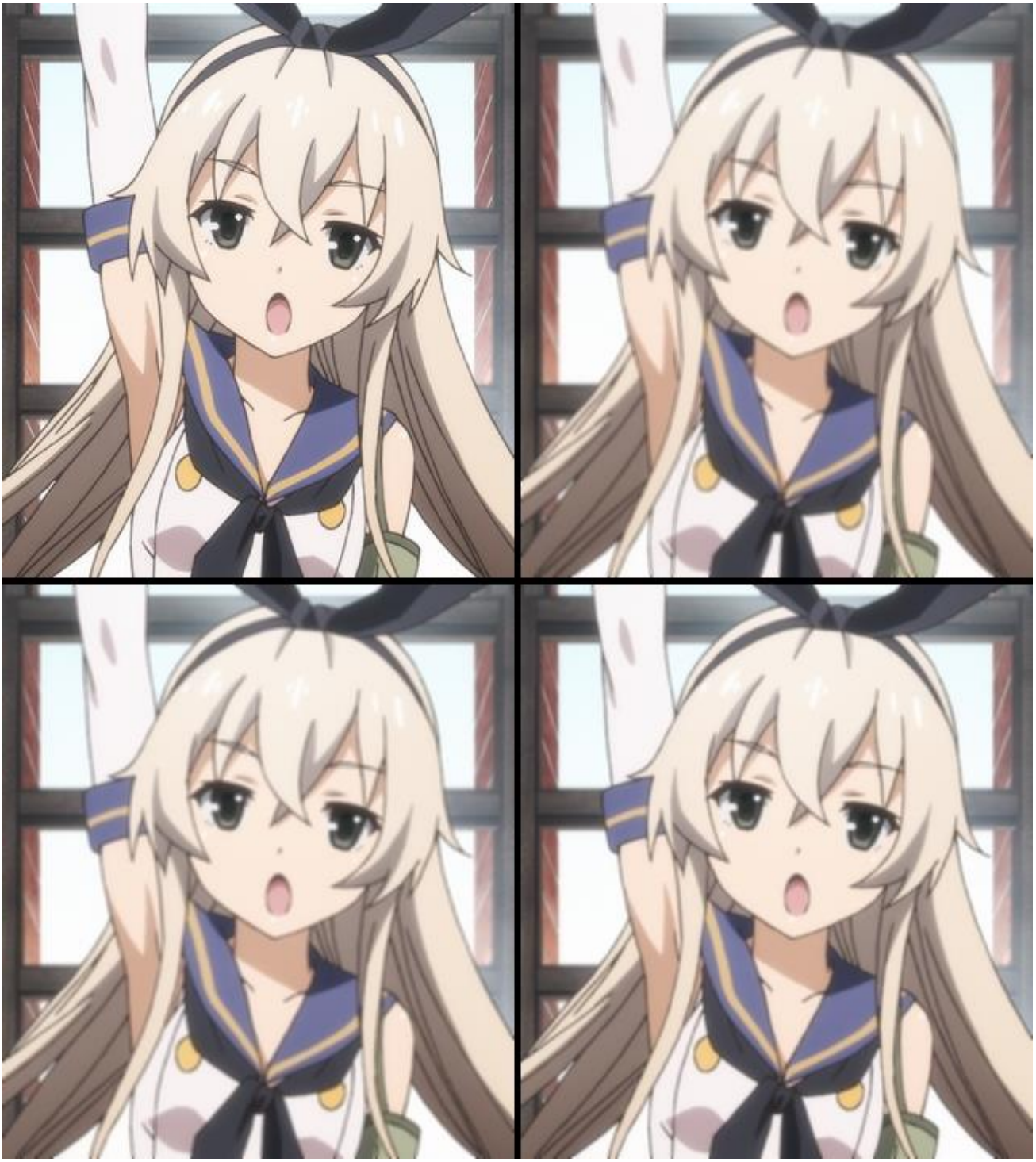
或许有人觉得这个 Blur 的力度太强了。假设中心像素是 9，周围所有像素都是 0，那么中心像素将被调整到 1，这损失还是很大的。一种调整方法是，不采用平均数，而是加权平均，让中心像素取得值大一些，离中心像素远的就小一些。比如说中心像素权值是 1/4，上下左右四个点权值是 1/8，四个角权值是 1/16。容易验证总权值 =1/4+4*1/8*4+4*1/16=1，所以这是一个可行的加权平均方法。这个方案就是 RemoveGrain(11):

破坏力就不像之前那么大了。

AverageBlur 不可避免的造成图像变得模糊和柔和，但是细节、线条基本会被保留，噪点也很难彻底的去除。因为任何偏高偏低的数字，在局部取平均数只能削弱它偏高偏低的程度，并不能完全抹平。

RemoveGrain 中还有一种模式是，中心像素权值为 0，周围 8 个像素权值都是 1/8。这样 Blur 的力度更高：

所以，中心像素权值越低，四周像素权值越高，Blur 的力度就越大。4 幅图放在一起，你能看出 3 张 Blur 的图分别是哪种模式造成的么？



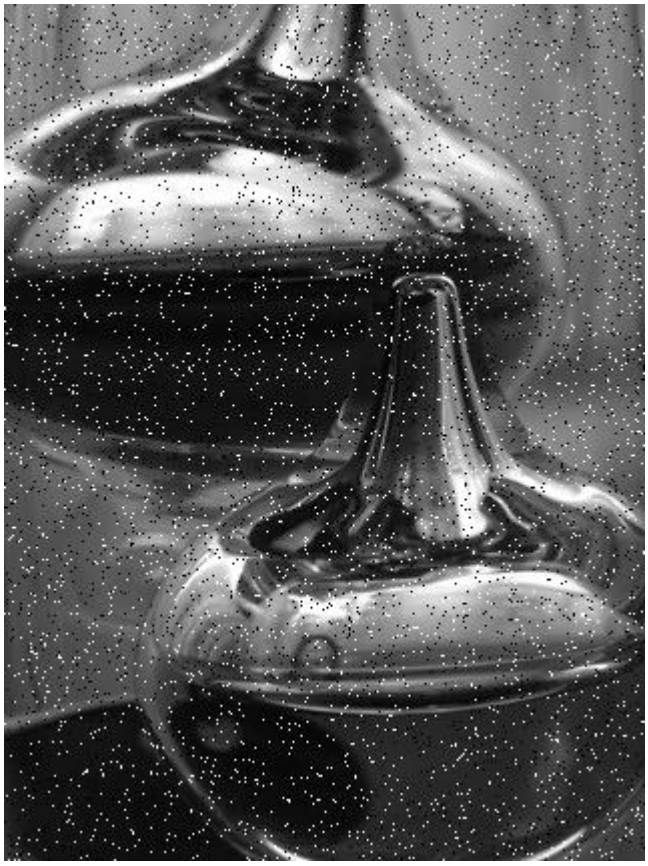
答案：$\begin{pmatrix} 源 & 19 \\ 20 & 11 \end{pmatrix}$

2. MedianBlur 的四种实现——RemoveGrain(1,2,3,4)

除了 AverageBlur，RemoveGrain 还可以实现 3x3 的 MedianBlur。这个模式是 RemoveGrain(4)，作用是将中心像素替换成九个数的中位数（第五大的数字）：

MedianBlur 的 blur 效果和 AverageBlur 有着非常不同的表现方式。图像不会出现之前模糊的效果，但是细线条之类的会有明显的杀伤。通常表现为锯齿、虚线、断层等效果。

在数字图像处理中，有一种噪点叫做椒盐噪点，它是一种随机出现的白点或者黑点。椒盐噪点的成因可能是影像讯号受到突如其来的强烈干扰而产生、类比数位转换器或位元传输错误等。例如失效的感应器导致像素值为最小值，饱和的感应器导致像素值为最大值。比如原图（左）和加入椒盐噪声（右）：



椒盐噪声就是类似四周是 0，中间是 255（最大值）这种存在。AverageBlur（左）在处理上就不如 MedianBlur（右）好使：

不过在日常处理中，直接取中位数的杀伤力一般还是太大。假设在 3x3 的邻域进行类似处理，且中央像素的值≥中位数，我们希望改一下算法：

1. 设定一个阀值，M，M=1,2,3 或者 4

2. 如果中央像素的值大于周边 8 个数字中，第 M 大的数，则把中央像素的值替换为这个数。

如果中央像素的值<中位数，做法类似：如果中央像素的值小于周边 8 个数字中，第 M 小的数，则把中央像素的值替换为这个数。

注意，M=4 的情况就是 MedianBlur：

如果中央像素就是中位数，那么周边 8 个数字一定正好有 4 个数≥它，4 个数≤它，不做改动；

如果中央数字大于中位数，那么 9 个数字中的中位数，是包括中央像素在内第 5 大的数，也是排除掉中央像素、8 个周边像素中第四大的数字，把中央像素改为第四大的数字，就是改为 9 个数的中位数。

如果中央数字小于中位数，情况类似。

如果 M=1，那么只有当中央像素是 9 个像素中最大或者最小的，我们才对它改动，改为它周边 8 个数字中最大或者最小的。这就是 RemoveGrain(1)，又有个 Undot()滤镜是做同样的事情。

如果 M=2，那么只有当中央像素是 9 个像素中最大/次大，或者最小/次小，我们才对它改动，改为周边 8 个像素中次大或者次小的。这就是 RemoveGrain(2)，又是 RemoveGrain()默认的模式。

如果 M=3，只要中间像素不是 中位数 / 中位数之前一个 /中位数之后一个，我们都改为周边 8 个像素中第三大或者第三小的。这就是 RemoveGrain(3)。

M 从 1 到 4，破坏力是越来越强的。一般 M=2/3 很适合用来做一般向的快速降噪（比如下图选择 M=3）：

3. RemoveGrain 小结

RemoveGrain 是最常用的 Blurring 滤镜，主要手段是以 MedianBlur 为核心的 mode=1~4，和以 AverageBlur 为核心的 11,20（偶尔也会用 19）。其中，Mode=4 是 MedianBlur(radius=1)。

RemoveGrain 还有很多其他 mode，绝大多数无人问津。

RemoveGrain 可以通过设置多个 mode，来指定 YUV/RGB 平面不同的处理方式。比如说 RemoveGrain(3,2,2)就是 Y 平面用 mode=3，U 和 V 平面用 mode=2。在指定的 mode 数量不足 3 个的时候，后面的平面自动用前面的平面使用的 mode。所以 RemoveGrain(3,2,2)跟 RemoveGrain(3,2)等价。RemoveGrain(20)就是 Y 用 mode=20，不指定的 UV 跟着 Y 用 20。

mode=0 表示不做处理，直接复制该平面的信息；mode=-1 则表示复制都去不复制，直接出来是垃圾数据（可能是任何值）

RemoveGrain 可以叠加使用。比如 RemoveGrain(20).RemoveGrain(20) .RemoveGrain(20)可以作为很强力的 blurring 滤镜。

在使用 averageblur 的模式时候，如果你的源是 8bit，转为 16bit 计算有助于提升精度，因为涉及到四则运算。反之，如果是 mode=1..4 这种 Min/Max/Median 的操纵，升到 16bit 没有什么用，因为只是比较大小和从现有数字中选取替换。

4. 特殊的 AverageBlur——GaussianBlur/高斯模糊

averageblur 在使用的时候，有个问题就是每个像素的权值给多少。这在扩展到更大范围 Blur 的时候尤其需要精心设计。我们希望每个像素的权值，随着离中心像素的距离疏远而减少。也就是说：

中心像素权值最高，越远离它，权值越低；

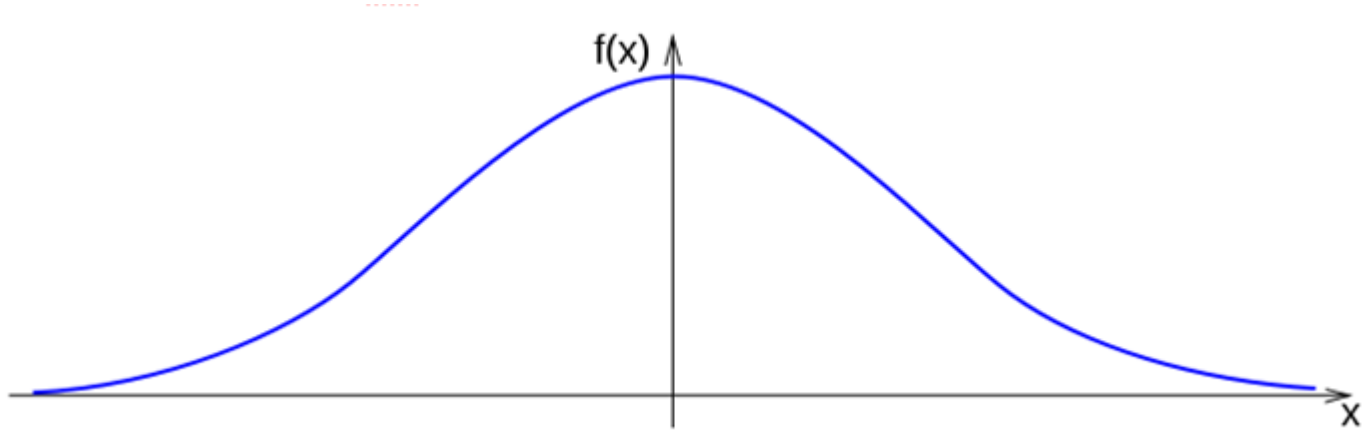
距离相同的点，权值相同；

距离无限远，权值趋近于 0；

同时，这些权值加起来等于 1。

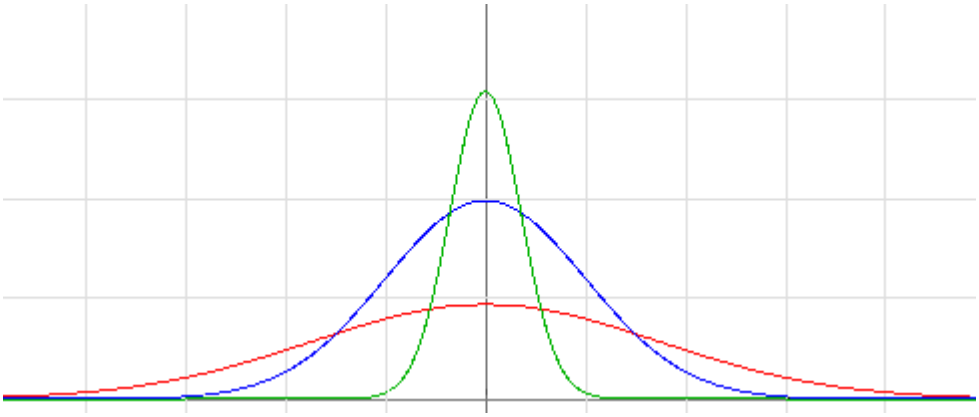
抽象到一维数学：有哪个曲线 $y=f(x)$，在 $f(0)$ 值最高；离 $x=0$ 越远值越小；距离为 a 的时候，$f(a)=f(-a)$；$\lim_{|x|\to\infty} f(x)=0$；且 $...+f(-2)+f(-1)+f(0)+f(1)+f(2)...\approx\int_{-inf}^{inf} f(x)dx=1$？

答案是：高斯曲线，又可以被看做正态分布 $N(0,\sigma^2)$ 的概率密度函数：



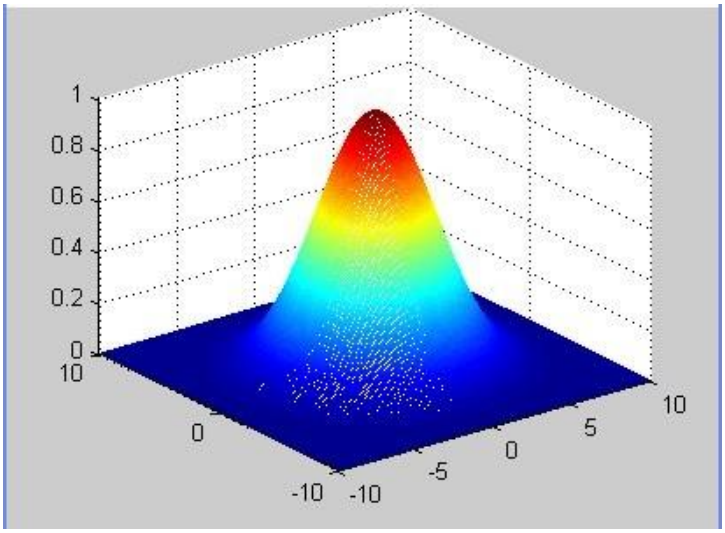
高斯曲线的表达式是 $y=f(x;\sigma)=\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{x^2}{2\sigma^2}}$。

设定对称轴为 $x=0$ 时，高斯曲线由参数 σ 决定。σ 越小，$f(0)=\frac{1}{\sqrt{2\pi}\sigma}$ 越高，图像越窄，函数在远离 $x=0$ 时衰减越快。不同 σ 对图像影响如下（绿线 σ 最小，红线最大）：



高斯曲线 99.7%的信息量都集中在 $[-3\sigma,3\sigma]$ 的区间内，意味着我们可以无视更远距离的点，因为它们几乎都为 0。

把它扩展到二维平面，我们就得到了一个高斯曲面：



方程为 $z = g(x, y; \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} = f(x;\sigma) \times f(y;\sigma)$ ，就是一维的直接相乘。如果我们把图像格点相对距离看做 xy 平面，并且指定一个 σ ，我们就可以套用高斯曲线的公式，计算权重。比如说 $\sigma = 1$ ，我们计算 $[-3\sigma, 3\sigma] \times [-3\sigma, 3\sigma]$，也就是 7x7 范围内的权值：

1.9641E-05	0.00023928	0.00107238	0.00176805	0.00107238	0.00023928	1.9641E-05
0.00023928	0.00291502	0.01306423	0.02153928	0.01306423	0.00291502	0.00023928
0.00107238	0.01306423	0.05854983	0.09653235	0.05854983	0.01306423	0.00107238
0.00176805	0.02153928	0.09653235	0.15915494	0.09653235	0.02153928	0.00176805
0.00107238	0.01306423	0.05854983	0.09653235	0.05854983	0.01306423	0.00107238
0.00023928	0.00291502	0.01306423	0.02153928	0.01306423	0.00291502	0.00023928
1.9641E-05	0.00023928	0.00107238	0.00176805	0.00107238	0.00023928	1.9641E-05

这些数字加起来和是 0.99946，非常接近 1。我们把它用作权值进行 Blur，这就是 avs 中的 Gblur(1.0)或者 vs 中的 std.Gblur(1.0):

如果使用$\mathrm{Gblur}(\sigma=0.5)$，则 Blurring 的效果会弱很多：

所以，GaussianBlur 中，σ 决定了柔化的力度，这个值越高，柔化的力度越强。

在较低强度的 Blurring 中，GaussianBlur 很好用；较高强度的 Blurring，并不如使用（串联的）RemoveGrain 好使；因为这时候你只是想达成糊一脸的效果，RemoveGrain 计算要快很多，效果也没啥区别。

5. 其他的 Blur 手段介绍

avs 中的 Blur(x)和 vs 中 generic.Blur(x)都能做 averageblur，x 控制强度。avs 中和 vs 中强度并不对应。一般建议用 RemoveGrain 替换它。

avs 中的 MedianBlur()是做大范围 MedianBlur 的，比如说 MedianBlur(3,2)就是对 Y 做 7x7，UV 做 5x5 的 MedianBlur，威力巨大。

avs 中和 vs 中都有 Convolution，就是对图像做卷积，可以看做一个自定义参数的 averageblur。不过一般没有必要学习如何使用它。

MinBlur，avs 中 mawen 的各种 vs 脚本里面就有（比如 GSMC_MinBlur）,vs 中可以用 haf.MinBlur 调用，是一种结合 averageblur 和 medianblur 的做法，具体是分别按照给定的半径，用两种方法 blur，然后取杀伤力较小，即对像素改动较小的方式来处理。比如说 MinBlur(r=1)，也就是 3x3 的 MinBlur：

可见，其效果和破坏力的控制，兼具 RemoveGrain(4)和 RemoveGrain(11)的优点。