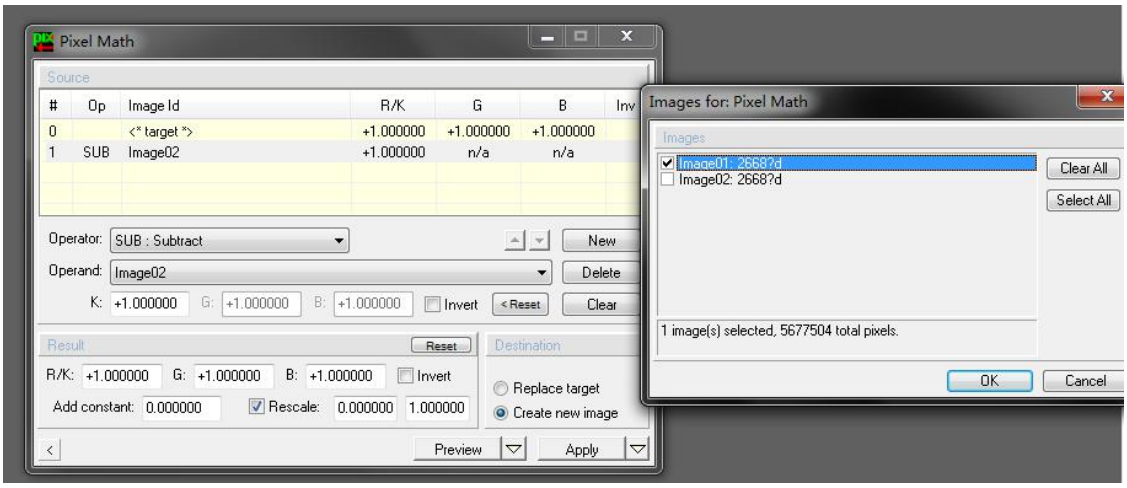
什么是 PixelMath？说白了就是两张照片之间的数学运算。其实我们做校准，叠加什么的本质上也是 PixelMath。

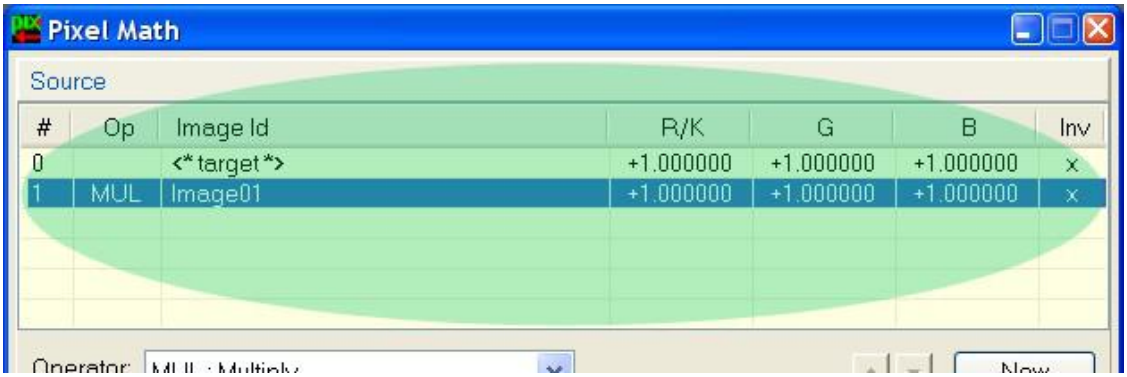
今天讲的 PixelMath 是校准叠加之外的。进行 PixelMath 的图像有两种，一种是目标（Target），另一个是动作源（Operand，动作源的说法是我胡扯出来的）。目标是被动方，动作源是主动方。比如一次减法运算，目标是被减数，动作源是减数（小学概念，别说不懂）。

最基础的概念就是这个了。现在看看 Pixel Math 的窗口。

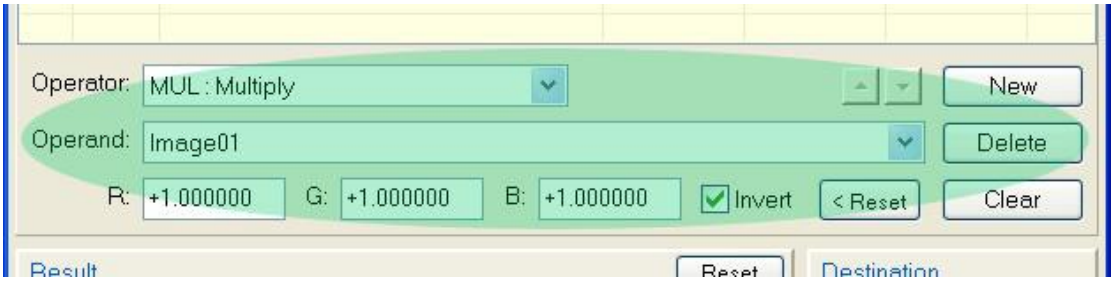


左边是 PixelMath 本体的窗口，右边的是 Apply 窗口，点击本体窗口右下角那个小三角形可以叫出来。

好了先围绕这个窗口说事。



最上面是这个东西。说实话没什么卵用，只是一个给你看参数的东西，等我之后讲完那些参数，也就能看懂了。不管他，往下看。



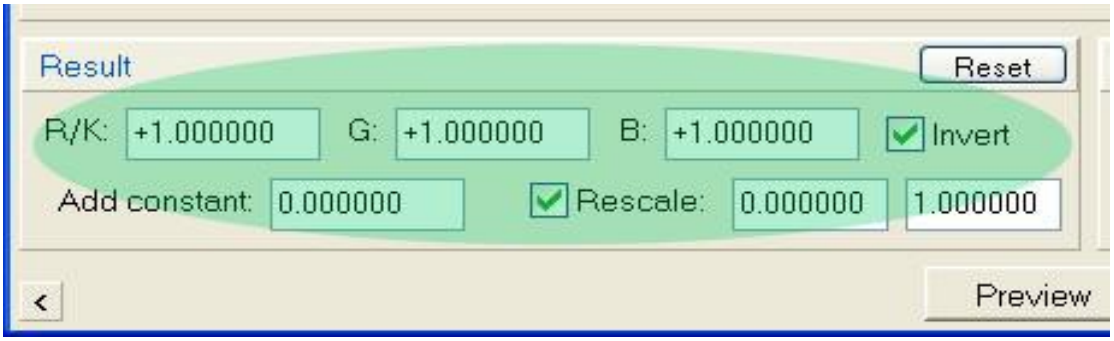
这个东西，是设置动作源图片参数的。点击 new 可以新建一个运算，Delete 用于删除当前运算。

Operator 是指使用的算法，比如加法（ADD）减法（SUB）什么的。PI LE 虽然说是阉割版，但还算非常良心的，提供了一大堆算法，这里不一一说明，大家可以自己去看看软件提供的 Help，之后我也会放一点实例出来。

Operand 就是动作源图像了。回想我们做 DBE 的时候，这个 Operand 中应该选择我们提取出来的背景，而运算是减法运算。那整个 PixelMath 的过程就是，以原图为被减数，提取出来的背景为减数，进行减法运算。

下面一行 R（K）、G、B 三个数值是类似于渐隐的概念。Operand 为彩色图片的话是 RGB，如果是灰度的那就只有一个 K。这个数值的意思是把 Operand（的各个通道）乘以这些数值之后再进行运算。比如说我有一张降过噪的图，觉得降噪力度有点大，背景太平了，想做个渐隐，那我可以把原图和降噪后的图混合，算法用 Mov（相当于 PS 中图层混合的“正常”），Operand 用降噪后的图。然后 RGB（假设为彩色图片）都填 0.7，那就是说把降噪后的图片的 70%与原图混合。

然后在最右边还有一个 Invert，如果勾选的话，那就会在运算前将 Operand 图像反色。



这个是设置运算结果的。上面一行 R（K）GB 的意思是把结果（的各个通道）乘以那些数值。而 Add constant 就是把运算结果加上填入的数值（0-1 之间）。注意，在 PixelMath 运算中所有像素值都用介于 0-1 的实数表达。Invert 选项的意思是是否把结果反色。

Rescale 非常重要，默认是勾选的，意思是把结果重新投影到[0,1]（事实上你也可以自行填入范围）上，防止运算产生的饱和与零值。假如没有勾选那最终结果应用 1 中的算法；勾选了的话就是用 2 中的算法。

r 是指各个像素的运算结果，r’是最终结果，m 是计算结果中所有像素的最小值，M 是计算结果中所有像素的最大值。

这个投影算法是线性的。

Rescale

When no rescaling is selected, resulting pixels are just truncated to the normalized range. For each pixel r of the result:

$$r' \leftarrow \text{Min}(\text{Max}(0, r), 1) \quad (1)$$

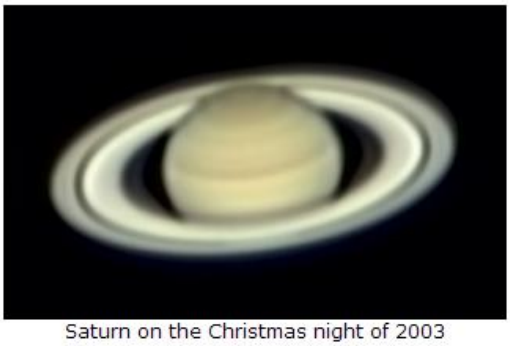
where r' represents the truncated resulting pixel value.

If the Rescale option is selected, then the resulting pixels are first rescaled to the normalized [0,1] range. This is done by:

$$r' \leftarrow (r - m) / (M - m) \quad (2)$$

最后就是，我们怎么把运算应用于图像呢？一种方法是用 Apply 按钮，先点击 apply 旁边的小三角，勾选目标图像，然后点 OK，再点击 Apply 就好了。或者直接把左下角的小于号拖到目标图像上。

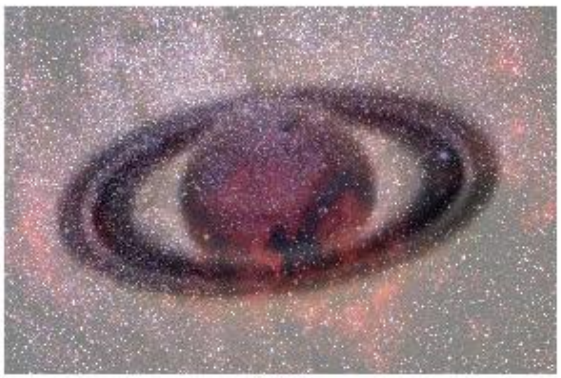
好了基础就是这些。很简单。下面我们看看一些例子。
首先是 PI LE 的 help 中的例子。一张土星和一张 NGC7000 之间应用各种不同算法产生的结果。



Saturn on the Christmas night of 2003

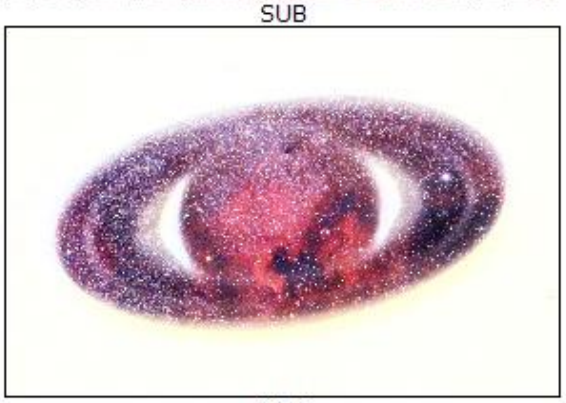
The North America and Pelican nebulae region. Three 1-hour shots on Kodak Royal Gold 400, taken on September 2000, when we were doing real astrophotography. Now we are always programming...

ADD



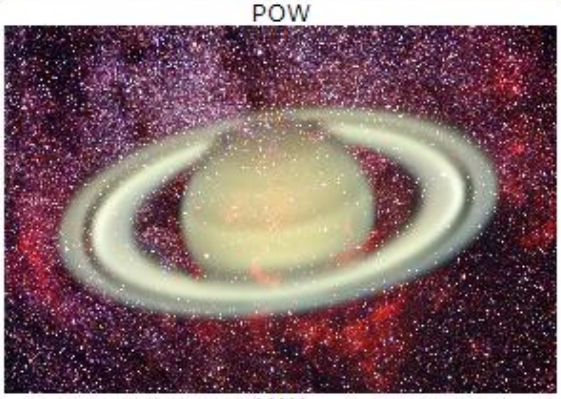
SUB

MUL



POW

MIN



MAX

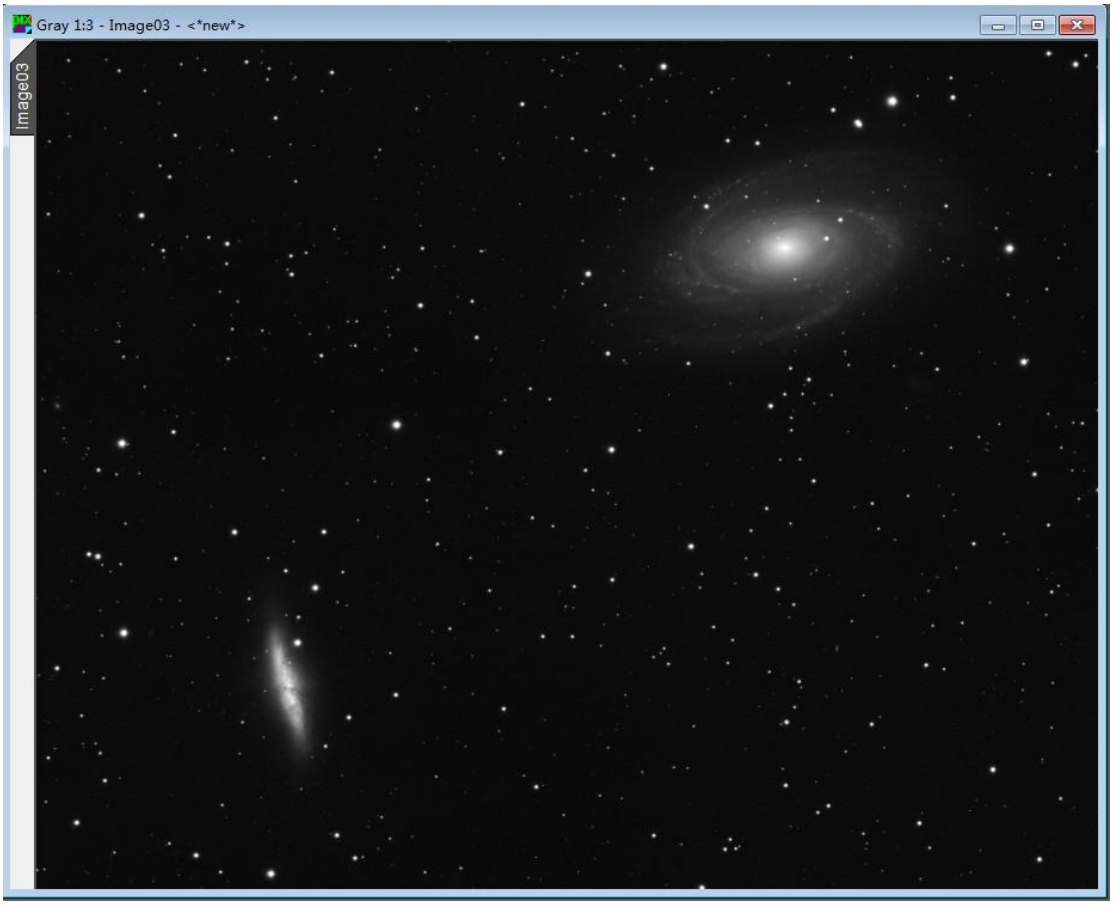
之后是我自己做的例子，用一个 HDR 和一个蒙版制作来让大家体会一下 PixelMath 的使用方法。这里用的是饭神拍摄的 M81&M82 的明度图像。

首先是 HDR。

线性原图这样。

拉伸了一次以后就这样了，星系的核心已经爆掉了，而背景的暗云气还不够明晰（事实上这张图片在暗云气上的信噪比不很高，强行去拉肯定会出问题）。

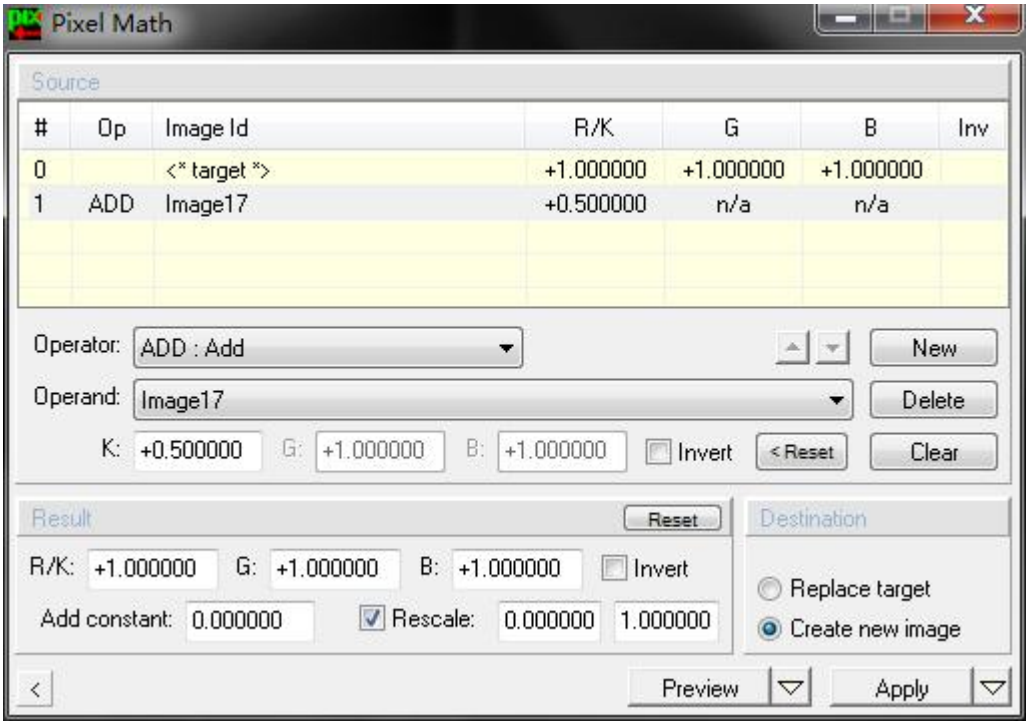
好那我们就来搞 HDR 吧。有些人局限地认为 HDR 必须要两张曝光不同的片子，事实上对于这种本身曝光没有任何问题的片子来说，我们只需要原图就行了。
分别进行不同程度的拉伸。



可以发现后者的亮部还没有饱和，细节仍然丰富。
于是我们就想到蒙版，亮部蒙版，把亮部蒙起来。

付诸实践后，蒙版长这样。

之后就很简单了，蒙上蒙版，进行 PixelMath 运算。一开始是想用 MOV，结果发现太过暴力。然后就选用 ADD 了。具体的操作窗口当时没有截图，大概是下图的意思。



搞出来大概是这样子。

似乎有点太暴力了……那就加以调整咯。多次调整参数，并使用三张图片进行两次 HDR 以后产生如下结果。

星点看起来好像略有些奇诡。HDR 的后遗症。

这不重要，因为解决这个问题只需要一个蒙版——把蒙版中的星点扣掉不让星点参与 HDR 就行了。在此我就不再管这个问题，大家在下一个例子中可以学到如何预防这个问题。

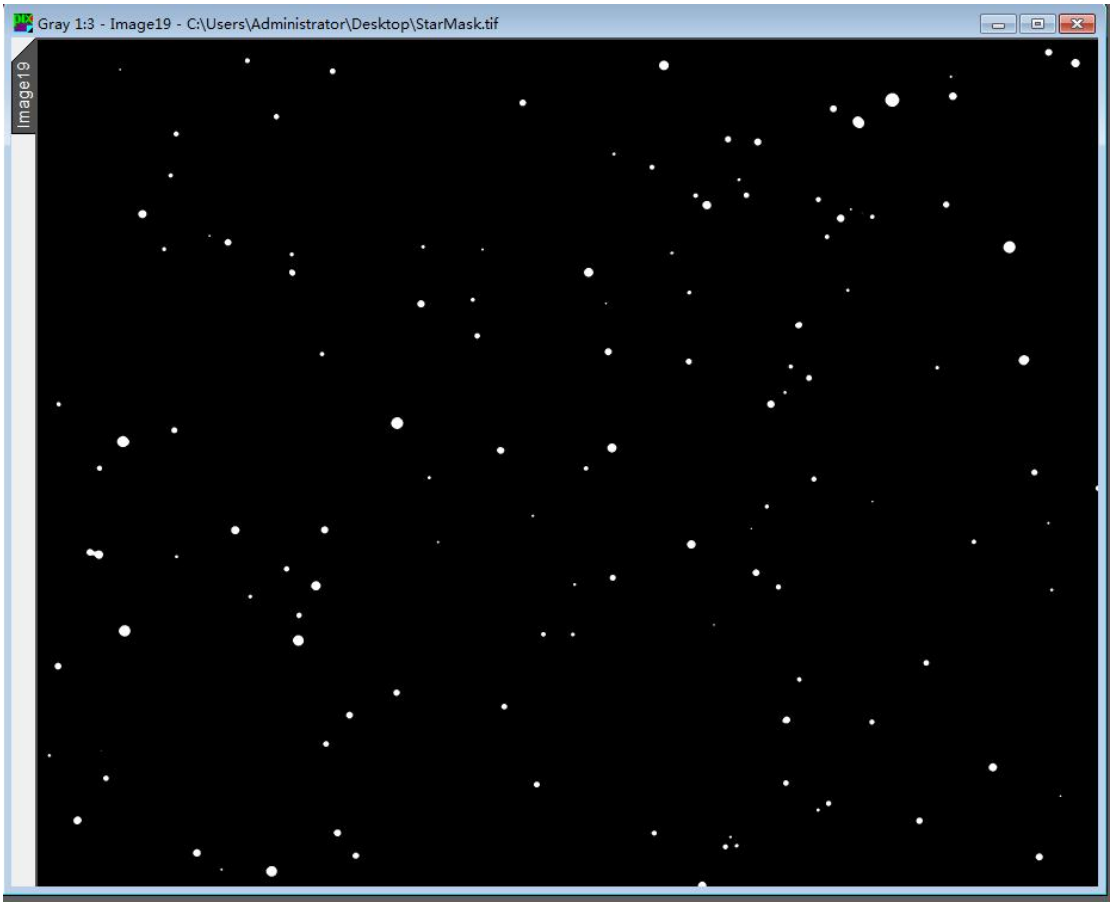
第二个例子，根据所需制作混合蒙版。

假如说我们想对这张 M81&M82 进行结构强化（在这里我就懒得花时间降噪了，但大家得注意强化最好在降噪之后进行）。

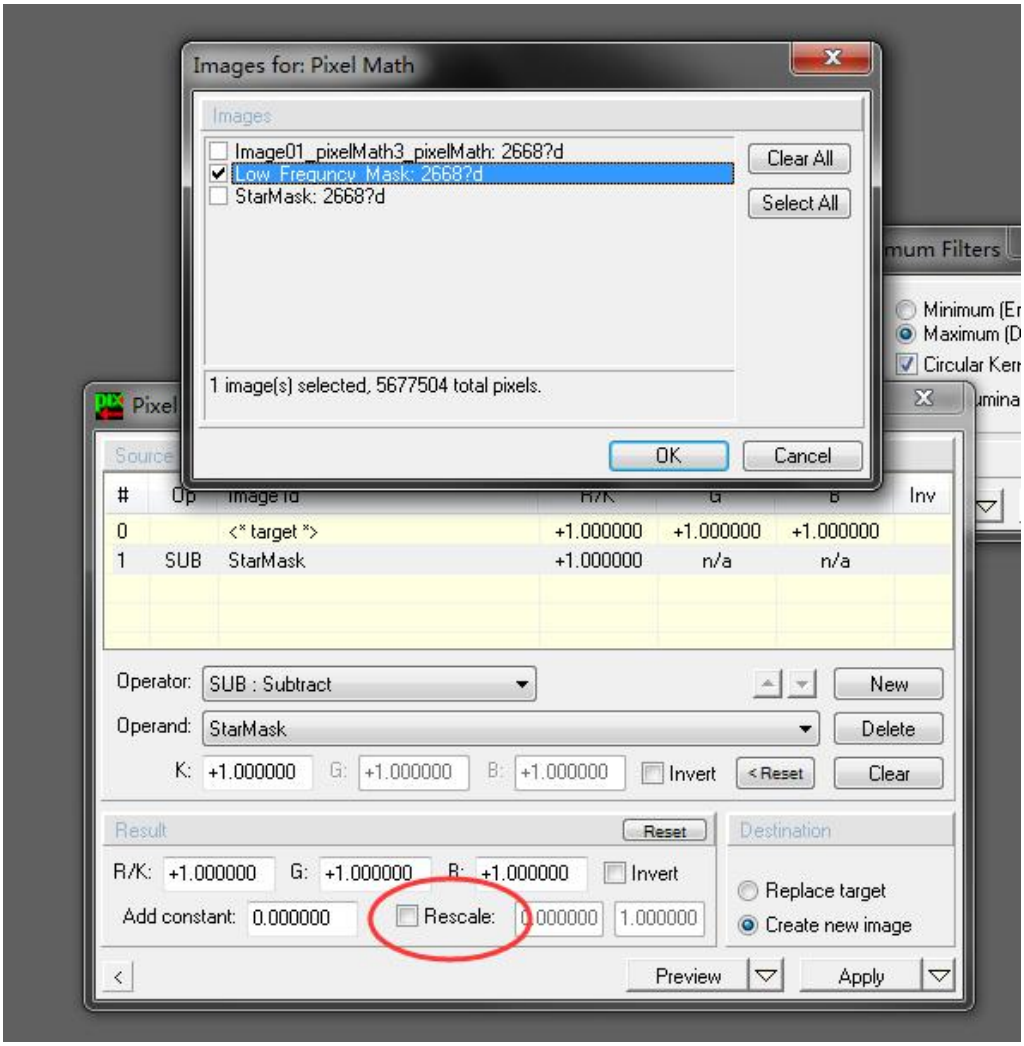
直接用 ATWT 增加 Bias，这样搞肯定不行。一方面 Ringing 效应吓死人，另一方面容易强化噪声。那还是得用蒙版。但是普通的低频蒙版不能排除星点，我们需要一个混合蒙版。

先做一个低频蒙版。这里直接用之前那个。

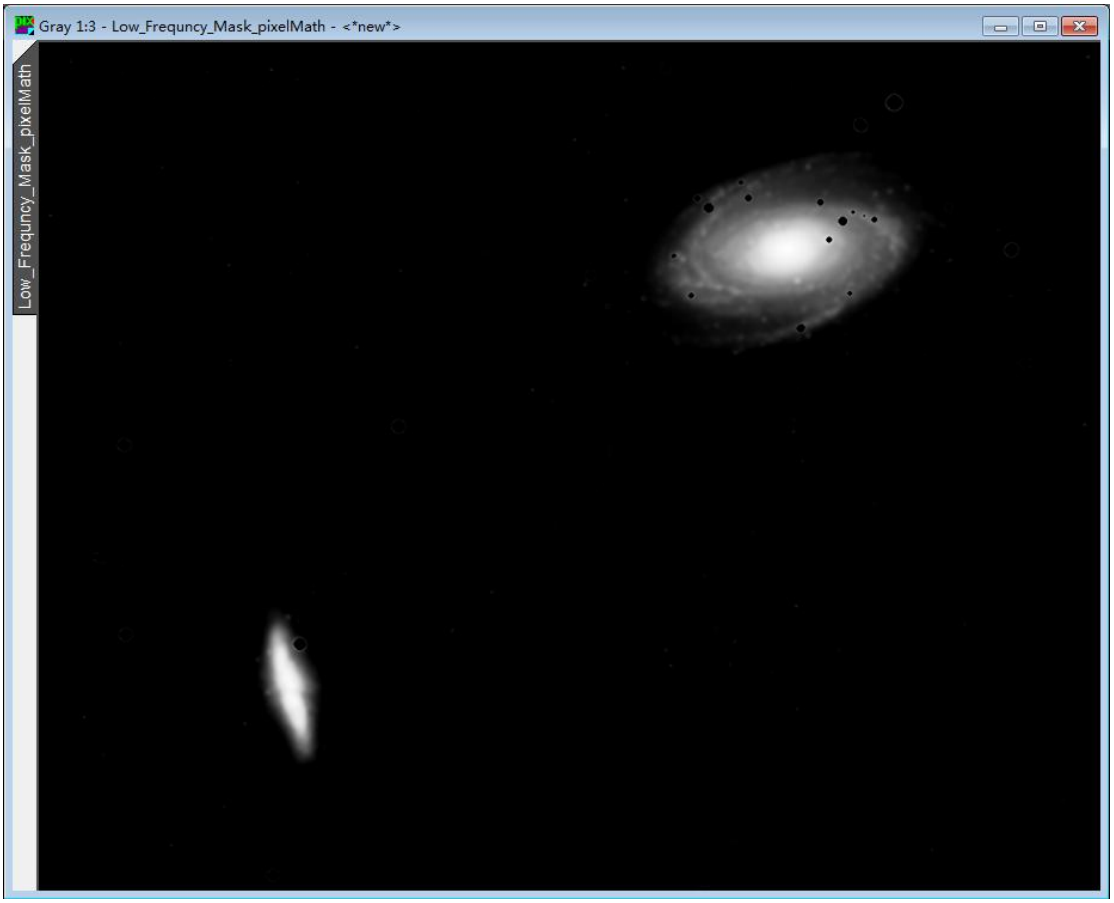
然后用 ATWT 提取出蒙版中的星点。之所以不在原图中提取是因为在蒙版中方便提取出亮星。



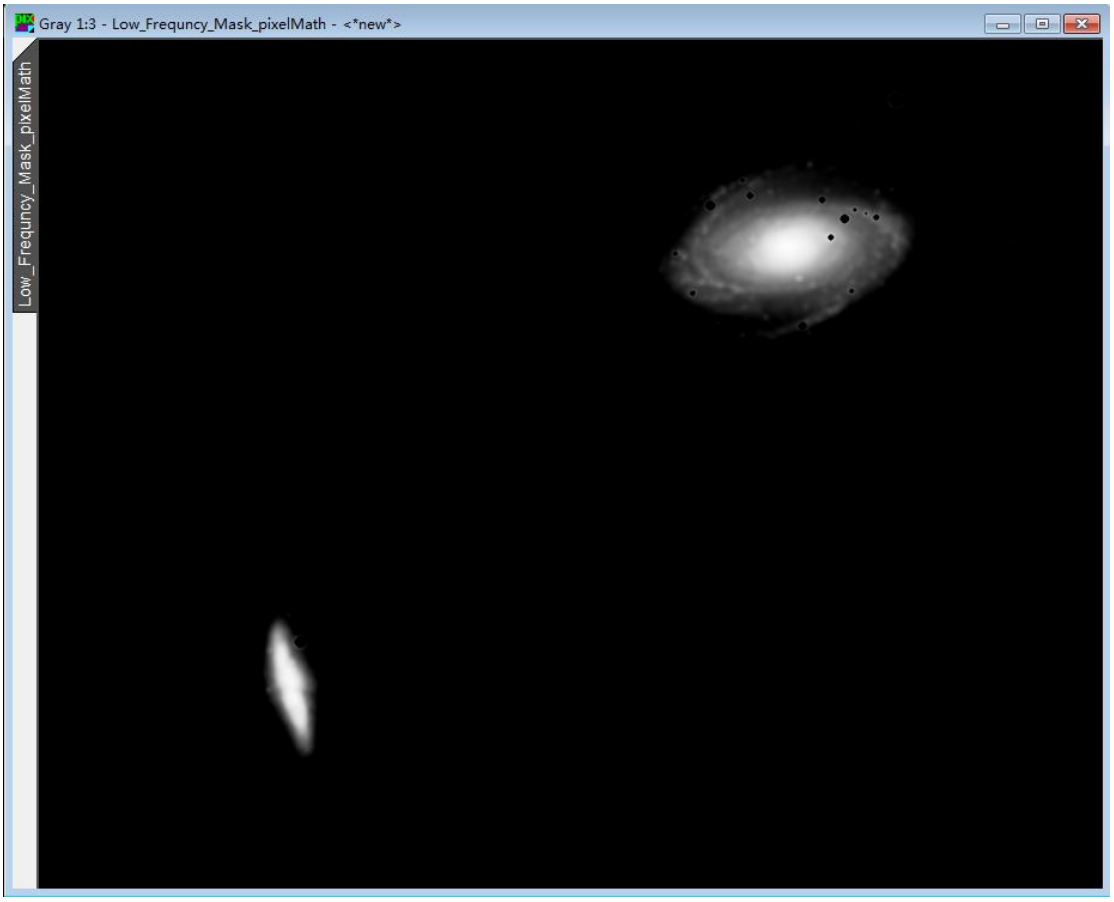
这个星点蒙版包含的是亮星。现在我们要把这个星点蒙版从之前的低频蒙版中扣除，以排除星点。



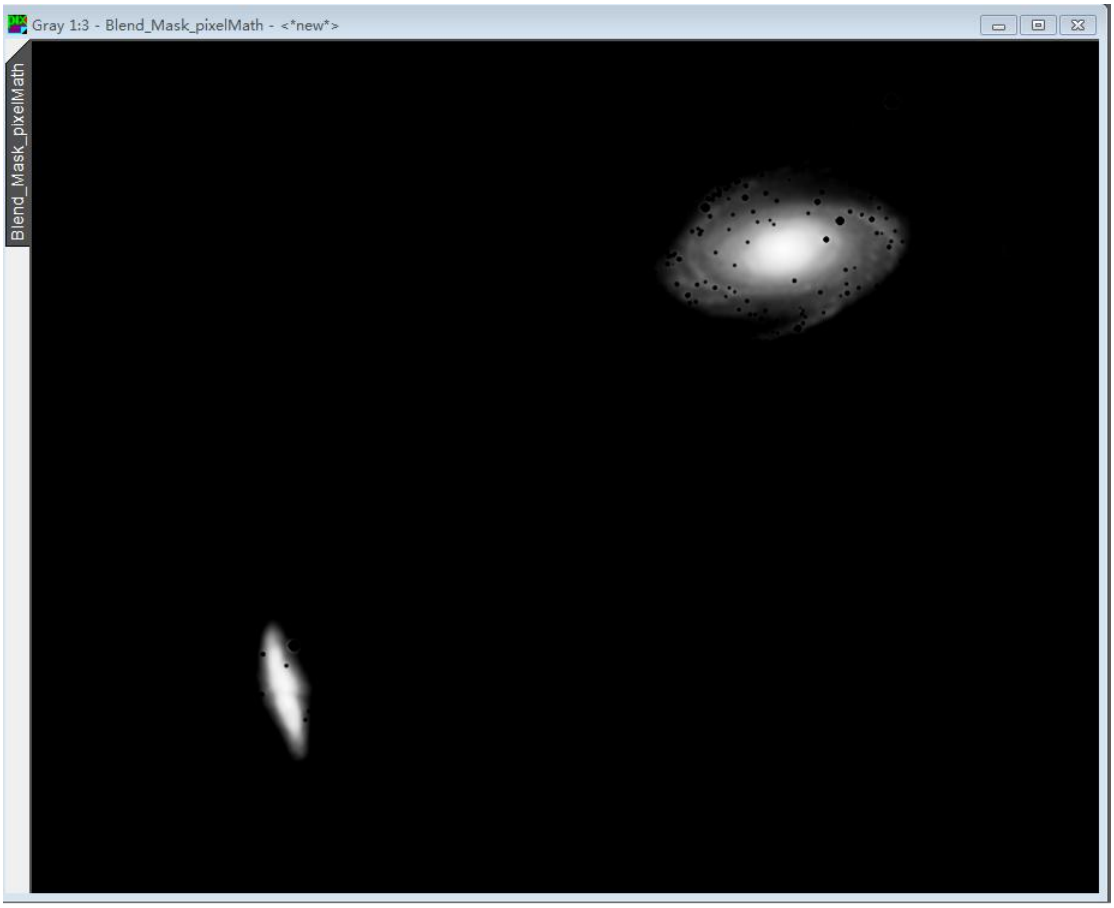
注意这个 Rescale。如果勾选了 Rescale 的话，那软件会把运算结果重新投影到[0,1]上，具体会产生什么，建议大家试试，这样对理解 Rescale 非常有帮助。
产生的结果是这样。



亮星被扣除了。发现扣得不是很干净，边缘有亮圈，做个色阶变换把它们切掉。



然后再用原图做 ATWT 提取出暗星，如法炮制，最终产生下图结果。



如此我们的混合蒙版就做好了。

接下来就是进行强化。用的还是 ATWT。随手乱搞了一下，搞成这样。

自己还是挺满意的。放大 M81 看看。

基本上就是这些东西。当然这个相比于我之前发的教程，比较粗枝大叶，讲得不细（事实上该讲的都讲了，只是没有涉及多少实际操作）。大家应该从中吸取思想，而不是机械地照搬。

两个例子包含了 PixelMath 的基本思想。其实 PixelMath 能做的多了去了。比如我们可以用某种手段（比如 PI1.8 的 GHDRC 和 DSE）提取出云气的细节，然后把它与原图做某种运算（最常用的是 ADD），以此强调某些成分。比如之前说到的渐隐。再比如各种复杂的空间结构运算（具体可以参考《星野摄影》一书）。甚至有人搞过用 PixelMath 完成所有后期步骤，因为整个数字影像的处理说穿了都是 PixelMath。

上面我自己做的例子完全就是自己开的脑洞。我们搞天文摄影的后期，灌注自己的理解实在是非常重要。只有自己真正理解了各个工具可以干什么，它是怎么运算的，那才能真正用活这个工具。

----------完----------