BlurXTerminator

By Russell Croman, RC Astro, LLC

Deconvolves astronomical images using AI (neural network) methods. [more]

Categories: Deconvolution

Keywords: deconvolution, sharpening, point spread function, PSF, neural network, AI, SNR, sampling, aliasing, aberration, astigmatism, coma, chromatic aberration, guiding errors

Contents [hide]

1 Description [hide]

BlurXTerminator is an AI-based deconvolution tool designed specifically for astronomical images taken with equipment commonly used by amateur astrophotographers. Not all AI is created equal. AI-based sharpening tools

for general photography exist but, when applied to astronomical images, they are prone to "inventing" detail that does not exist. They also don't usually handle stars very well. Their neural networks were not trained on astronomical images, so they often make bad "guesses" as to what the original, unblurred scene looks like.

The design intent of BlurXTerminator is to recover as much detail as possible based on low-contrast information actually present in an image, without fabricating detail that does not in fact exist just for the sake of an image that *appears* sharper. Great care has been taken in the architecture and training of the neural network to ensure that its output is as faithful as possible to reality if it is properly used.

All deconvolution, including the classical algorithms developed by Richardson, Lucy, van Cittert, and others, fundamentally involves guesswork. Mathematically, deconvolution is said to be an *ill posed* problem: for a given blurry input image, there are many possible sharper images that, if re-blurred, would result in the same input image. Which one is correct, or at least a better guess?

The classical algorithms use knowledge of an image's *point spread function* (PSF) to help guide deconvolution, which can be made to work as long as the PSF supplied to the algorithm is accurate. The application of neural networks to deconvolution brings an additional source of information to guide the process: knowledge of the structures and patterns typically present in real, high-resolution astronomical images. BlurXTerminator's neural network was trained using extremely high-resolution images acquired by instruments such as the Hubble and James Webb space telescopes. It "understands" what astronomical structures actually look like at finer scales than can be resolved using amateur equipment.

The training methodology additionally includes a deep understanding of the common point spread functions that astronomical images are subject to, including variations caused by atmospheric turbulence, optical scattering, acquisition issues such as guiding errors, and optical distortions such as coma and chromatic aberration. There is no need to extract the PSF ahead of time: BlurXTerminator uses the stars in an image as PSF references. It analyzes and processes an image in one step, with no iteration required in most cases.

BlurXTerminator can apply different amounts of deconvolution to the stellar and nonstellar features of an image. Trying to recover all of the detail available in nonstellar, extended objects using the classical algorithms usually results in dark halos (*ringing)* around stars. With BlurXTerminator, more sharpening can be applied to the nonstellar parts of an image, bringing out more detail without producing ringing artifacts in most cases.

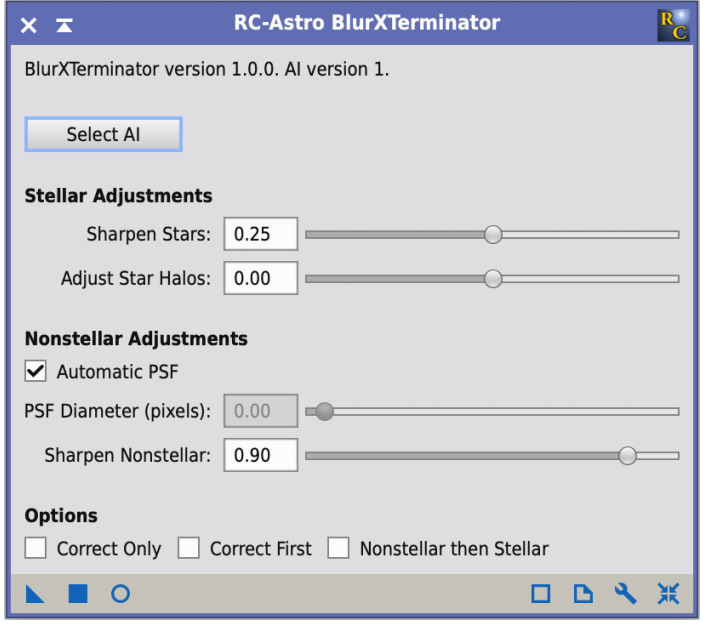
BlurXTerminator can additionally correct for other aberrations present in an image in limited amounts. Among those currently comprehended for most instruments are:

- Guiding errors
- Astigmatism
- Primary and secondary coma
- Chromatic aberration (color fringing)
- Varying star diameter (FWHM) and halos in each color channel

These aberrations are not assumed to be *stationary*: they can vary across the field of view. This is a major advantage over classical deconvolution algorithms that assume that the same PSF applies to the entire image. For example, stars with limited comatic profiles in the corners of an image will be made round and then sharpened, while stars in the center of the image that are already round will simply be sharpened. This correction can be applied to the nonstellar features of the image, too. Correction can be done as a separate step, or in combination with sharpening.

2 Usage [hide]

2.1 Controls



2.1.1 Stellar Adjustments

Sharpen Stars

Amount to reduce the diameter (in the FWHM sense) of stars. Higher values result in smaller, sharper stars, as small as half of their original diameter.

High values for this setting may not work well for all images acquired by all instruments. Choose a setting that results a reasonable amount of star diameter reduction without creating artifacts. If dark halos appear around stars, choose a lower value or compensate for this by increasing the Adjust Star Halos parameter.

With images acquired using long focal-length instruments, high values of this parameter may leave noticeable featureless areas around brighter stars: there is little to no information in the image with which to fill in these pixels with any detail due to clipping of the centers of the star profiles by sensor saturation.

Adjust Star Halos

Amount to adjust the "halos" of stars. Higher values result in brighter halos with larger extents, giving stars a softer appearance. Smaller values reduce halo brightness and extent, giving stars a sharper, harder-edged appearance.

Low values for this setting may leave dark halos around stars in images acquired with some instruments. Choose a setting that results in reasonable adjustment of the halos without artifacts.

Since the colors of stars with saturated (clipped) cores is visible only in their halos, increasing their halos while reducing their diameter can help retain or emphasize star color.

2.1.2 Nonstellar Adjustments

Automatic PSF

Determines the PSF (in the FWHM sense) for nonstellar sharpening automatically from the image.

Stars must be present in all parts of the image for this to be accurate. If stars are not present, BlurXTerminator will attempt to adaptively determine the PSF from nonstellar image features. This can still work well on many images in a qualitative sense, but smooth features that happen to have a naturally blurry appearance may become over-sharpened.

If Automatic PSF is selected, any aberration correction (coma, astigmatism, etc.) applied to stars will also be applied to nonstellar features. This is not true in manual mode (unless the Correct First option is selected): the PSF of nonstellar features will be assumed to be round.

Some images, particularly those taken at long focal lengths, may have areas with few or no stars. BlurXTerminator may over-sharpen the non-stellar features in these areas if the Automatic PSF option is selected. Images are processed in 512x512 pixel "tiles," with a 20% overlap between tiles to avoid artifacts. Individual tiles are processed semi-independently to allow for non-stationary PSFs. If a particular tile does not have enough stars in it, BlurXTerminator will revert to trying to deduce the PSF from non-stellar features, resulting in non-uniform sharpening.

In these cases, more accurate results can be obtained by switching to manual mode and setting the PSF Diameter to the known FWHM of stars in the image as a reasonable starting point.

PSF Diameter

The PSF diameter (FWHM) to be used when deconvolving nonstellar features.

For best accuracy, this should be set to the FWHM of stars in the image, or for lunar and planetary images, the FWHM of stars that would be produced by the same optical and camera system, using the same preprocessing pipeline.

Images with FWHM values greater than 8 pixels, the maximum value for this parameter, are oversampled by more than a factor of two and can be safely down-sampled by a factor of two. See Sampling below for more information.

Sharpen Nonstellar

The amount to sharpen nonstellar image features.

BlurXTerminator will attempt to reduce the size of the PSF of nonstellar image features by up to the full diameter specified. In other words, setting this to 1.00 means to attempt to reduce the nonstellar PSF to zero size — an ideal point PSF, and the maximum possible amount of sharpening. The *actual* amount of sharpening achieved will be limited by the recoverable low-contrast detail present in the image at fine pixel scales.

2.1.3 Options

Correct Only

BlurXTerminator can correct for limited amounts of motion blur (guiding errors), astigmatism, primary and secondary coma, unequal FWHM in color channels, slight chromatic aberration, and asymmetric star halos. Correction is always performed, but better results can sometimes be obtained by performing correction as a separate step, prior to any further sharpening.

This is a convenience option that is equivalent to selecting Automatic PSF and setting all other parameters to zero. Correction will be applied to nonstellar features as well as stars as long as stars are present in all parts of the image.

Technically speaking, BlurXTerminator will attempt to make the point spread function (PSF) found in the image *azimuthally symmetric* (round). The PSF need not be *stationary* — the aberrations can vary across the image. BlurXTerminator will attempt to correct for the local PSF in each part of the image, making it round while preserving its flux and centroid.

Correct First

This is a convenience option that processes the image twice: once as with the Correct Only option above, and then with other adjustments enabled. This will run faster than performing each step separately since processing overhead is shared between both operations.

Nonstellar then Stellar

Performs nonstellar sharpening first, followed by stellar sharpening.

This can be useful for some images, for example galaxies with faint, barely-resolved stars embedded in other structures such as HII regions. Stellar sharpening may not recognize these as stars at first, but if nonstellar sharpening is performed prior to stellar sharpening, these stars become resolved in the first step and then sharpened in the second step.

2.2 Input

2.2.1 Linear data

When first using a new tool, it is tempting to try it on images that have already been processed. BlurXTerminator, like classical deconvolution methods, works best on *linear* image data, ideally right after integration, channel combination, and perhaps color calibration and background flattening, but prior to any further processing. The one exception is that a reasonable amount of simple stretching with the HistogramTransformation tool will not harm performance. Any other stretching method, and indeed any other processing at all, will likely result in less accurate performance and artifacts: save those for after deconvolution.

Performing noise reduction, for example, prior to using BlurXTerminator is not recommended. BlurXTerminator's neural network has been trained to recover detail in the presence of noise. Most noise reduction techniques alter or destroy the low-contrast information at fine pixel scales needed for this.

Applying classical deconvolution algorithms prior to or after applying BlurXTerminator is also not recommended. The resulting image may *appear* sharper, but the likelihood that this apparent additional detail is not representative of reality is increased. Also, applying BlurXTerminator after traditional deconvolution usually results in any slight ringing, "curdling," or "worms" created by traditional algorithms being amplified.

Image data should be in floating-point format, even if stretched. If a stretched image is converted to 16-bit integer format prior to applying BlurXTerminator, very bright areas may develop *posterization,* a step-wise quantization of the brightness and/or color in the image.

2.2.2 Stars

BlurXTerminator uses the stars in an image as a reference to comprehend how it has been blurred, and how this varies across the field of view. If you plan to remove the stars from the image, it is best to perform this operation after applying BlurXTerminator.

If stars are removed prior to applying BlurXTerminator, it is recommended to de-select the Automatic PSF option, and instead manually set the PSF Diameter for nonstellar sharpening to the FWHM value of the stars prior to their removal. If the automatic mode is selected, BlurXTerminator will adaptively try to determine the local PSF from nonstellar features in the image. While this can produce a result that "looks nice" in a qualitative sense, it will usually result in over-sharpening of smooth features that in reality lack detail.

If there are parts of an image that have few or no stars, the same thing may happen: over-sharpening of smooth features. In this case, use the manual PSF Diameter parameter, and set it to the FWHM value of stars in the image as a starting point. This will help ensure that the whole image is sharpened uniformly.

For images with no stars, such as lunar and planetary images, the manual PSF Diameter mode is recommended for best accuracy. Set it to the FWHM value of stars that the same imaging system and preprocessing pipeline would produce, at least as a starting point.

2.2.3 Color

Best results will most likely be obtained by running BlurXTerminator on RGB images as opposed to each monochrome channel independently. The neural network is trained to recognize not only the PSF in each channel, but also the relationships between the channels. Slight color fringing due to chromatic aberration, for example, is not visible in the data of a single channel, and BlurXTerminator can't correct it unless it sees all three channels simultaneously.

When combining narrow-band data into a color image, it is best to avoid "mixing" color channels, as is done in some advanced color palette techniques, prior to deconvolution. If channels are mixed prior to deconvolution, especially with very different gains, star profiles can be significantly altered, perhaps so much as to make them unrecognizable as stars to the neural network. Do a simple SHO color combination, run BlurXTerminator, then perform any mixing between channels afterwards.

For a similar reasons, when processing LRGB images it may be best to deconvolve the L and RGB images separately, and then combine. If star profiles are significantly different in L than in RGB, LRGB combination may result in star profiles that the neural network has difficulty recognizing.

2.2.4 Noise

More detail can be recovered from images with higher SNR. Take and integrate more exposures to increase SNR.

A fundamental limit for any information recovery technique such as deconvolution is the amount of signal present compared to the amount of noise, the *signal-to-noise ratio* (SNR). Neither BlurXTerminator nor any other processing tool or technique can overcome this limitation, which is a basic principle of Information Theory. In a photon-limited practice such as astrophotography, the *only* way to increase SNR, all else being equal, is to collect more light, which is to say take more exposures. This assumes that the noise in individual exposures is indeed limited by *shot noise*, the noise inherent in measuring a signal in the form of discrete events (in this case, the arrival of individual photons).

2.2.5 Sampling

You will notice that the maximum PSF Diameter parameter value is 8 pixels. This is the maximum PSF diameter (in a FWHM sense) that BlurXTerminator has been developed to deconvolve. What about images that have larger stars, and thus larger amounts of blur?

It is mathematically straightforward to demonstrate that images that have PSF FWHM values greater than 8 pixels are oversampled by at least a factor of two: they contain no significant information at scales finer than two pixels. These images can therefore be safely down-sampled (using the IntegerResample process, for example) by a factor of two without any detectable loss of information. SNR will be higher due to averaging noise across multiple pixels, subsequent processing will run ~4x faster, and 4x less disk space will be consumed.

For those technically inclined wanting a more detailed explanation, see the analysis below.

2.2.6 Better data

While BlurXTerminator can partially correct for limited sins of acquisition — guiding errors, focal plane tilt, etc. — it will always be true that better input data will produce better output images. We spend hours and hours collecting our precious few photons. Spend a few more hours making sure that optical and mechanical issues don't get in the way of a quality result.

2.2.7 Space telescope images

BlurXTerminator has not been trained on stellar profiles from instruments such as the Hubble and James Webb space telescopes. It will thus very likely not recognize many stars in these images as such — particularly bright stars in JWST data — due to their unique diffraction patterns. Faint stars will likely be recognized and can be adjusted with the stellar parameters. Nonstellar sharpening can be performed, but this should be done with a manual nonstellar PSF diameter.

2.2.8 High dynamic range images on MacOS

On MacOS, many neural network computations are performed using 16-bit floating point arithmetic by the "CoreML" software library provided by Apple. For most images this is sufficient precision. In certain cases, notably images containing extremely high dynamic range objects — the core of M42, certain galaxies with very bright central cores, etc. — this can cause slight posterization in the brightest areas. In these cases, better results may be obtained by performing a HistogramTransformation stretch, applying some *mild* HDR compression (using HDRMultiscaleTransform) to reduce the dynamic range, and then applying BlurXTerminator. Stars should be masked prior to applying HDRMT to avoid significantly altering their profiles.

3 Discussion [hide]

3.1 How does it work?

All raw astronomical images are a result of two things: the actual scene that has been photographed, and a *point spread function* (PSF) by which the light from this scene has been *convolved,* or blurred.

Every star in an astrophoto taken with amateur (and most professional) equipment approximates an ideal *point source*: a spot of light with zero width, but with some brightness and color. If we had perfect imaging systems — extremely large optics, extremely small pixels, and no atmospheric seeing effects, guiding errors, optical distortions, etc. — all of the stars in our images would appear as infinitesimally small dots of varying brightness and color. Instead, we get stars with a profile that has a non-zero width and some particular shape. The ideal point-source star profile has been altered — blurred — by the sum total of the acquisition process. This process applies to every point of light in the image, not just the stars.

Imagine taking a single point of light in the actual scene, say the light from a single star, and spreading it out over multiple surrounding camera pixels, perhaps extending out to a great distance from the point's location. The light that should have ideally landed on just one of the camera's pixels might in some small part land on pixels tens, hundreds, or even thousands of pixels away. Repeat this process for every point of light in the actual scene in a very consistent and systematic way, using the same method of spreading the light out. If we then recombine the results of doing this to every point in the scene into a composite image, we have what we capture in an astrophotograph: a blurred version of reality. Technically, the exact way that the light from each point has been spread out is called the *point spread function*, or PSF. Performing this operation to every point of light in an image is called *convolution.* Attempting to reverse this process is called *deconvolution.*

The primary sources of the blurring function, the PSF, are the finite size of our optics and the turbulent nature of our atmosphere. The wave nature of light, combined with optics of limited size, gives rise to a *diffraction pattern*, the first source of blur. When referred to angular resolution on the sky, optics with smaller diameters produce larger diffraction patterns, which is to say more blur.

Earth-bound telescopes can collect light only after it has passed through our atmosphere. Turbulence combined with temperature differences in different parcels of air through which the light passes give rise to the second source of blur: *seeing.*

Life as an astrophotographer would be simpler if it stopped here, but there are many other sources of blur. Errors in guiding smear light into neighboring pixels, often mainly in the direction of the right ascension axis, leading to oval stars. Modern telescope optics are quite good, but there can be optical distortions such as chromatic aberration, astigmatism, field curvature, coma, etc. Diffraction spikes around bright stars, created by light interacting with secondary mirror support structures in telescopes that have them, are part of the PSF that can extend very far from its center. The "halos" of light surrounding stars would ideally be perfectly smooth and symmetric, but small variations in the optics can give rise to "rays" in these halos. Uneven obstruction of the light path by structures such as baffles and aperture stops can give rise to asymmetry in the halos.

Each of these mechanisms contribute to the total PSF of the system. Each is one additional way that the image gets blurred. It gets even worse: many of these mechanisms are what is technically referred to as *non-stationary*: they vary across the field of view. Stars in the corners of an image, for example, are rarely as sharp as stars in the center.

BlurXTerminator has the aim of reversing this blurring action using a process called *deconvolution.* Though it takes a completely different approach, it has the same goal as classical deconvolution algorithms such as those devised by Richardson, Lucy, van Cittert, and others. Most implementations of those algorithms suffer from the following limitations:

- They assume that the PSF is stationary (constant) across the field of view
- They require knowledge of the PSF prior to performing deconvolution
- They require many iterations to produce a result
- They assume the input data is entirely linear
- They assume a PSF of limited extent

BlurXTerminator, on the other hand:

- Does not assume that the PSF is stationary: it can handle a PSF that varies across the field
- Requires no *a priori* knowledge of the PSF: it determines the PSF on-the-fly from the stars in an image
- Produces a final result in one shot in most cases
- Can comprehend the nonlinearity of detector saturation (clipping)
- Does not assume a PSF of limited extent

The last two points are very important in handling bright stars. Faint stars significantly affect only a small number of pixels around their centroids. Bright stars begin to show the extended "wings" of the PSF, resulting in larger (sometimes much larger) visible halos as the light from these wings becomes brighter than the background. In addition, bright stars very often saturate (clip) the camera sensor: the PSF is "chopped off" at the top. These stars appear to have larger diameters and halos with greater extent, when in reality their profiles stem from the exact same PSF as fainter stars.

BlurXTerminator has been trained to deal with this situation. It "understands" that a bright, saturated star's actual peak value is "off the top" of the image due to clipping. It will reduce that star's diameter, and adjust the brightness and extent of its halo, as if the clipping hadn't happened. It processes an image at multiple scales simultaneously, so it can handle point spread functions that extend very far from their centroids.

It is useful to think of the action that BlurXTerminator performs on an image as translating its existing PSF into a different (smaller) PSF. The well-known *Moffat* PSF, for example, has two parameters, *sigma* and *beta.* The *sigma* parameter controls the overall size of the PSF, which is related to its FWHM diameter. The *beta* parameter controls the "wings" of the PSF — the faint outer extents that give bright stars their extended "halos." BlurXTerminator provides separate controls to modify both the diameter and the halos of the stellar PSF, effectively making adjustments to the *sigma* and *beta* parameters of the PSF independently. BlurXTerminator's neural network analyzes the existing PSF in the image and translates it to a different PSF with new *sigma* and *beta* parameters, effectively replacing the PSF with a new one.

No explict PSF extraction is performed by BlurXTerminator — it is done "on the fly" during processing. While it is technically possible to extract the PSF, it would complicate the neural network architecture and slow down processing.

3.2 Deconvolution? Really?

Can a neural network actually perform deconvolution in the formal sense?

Yes.

Deconvolution has a precise mathematical definition. At first glance, there is no simple, closed-form expression for what mathematical function a neural network with tens of millions of parameters and a bevy of nonlinear operations is performing. What can be expressed precisely, however, is the method employed to train it, and specifically whether or not this method is in keeping with the universal approximation theorem, which states that a neural network can provably approximate *any* continuous mathematical function, with arbitrary precision, provided that certain conditions are met.

All neural networks are trained as follows:

- Feed them an input,
- let them process it using initially randomized weights,
- "score" their performance by means of a suitable *loss function* which compares the output to some "ground truth,"
- use the chain rule from calculus to compute the gradient of the loss function with respect to every weight in the network,
- update the network weights to descend this gradient, and finally
- iterate to minimize the loss function using a suitable optimization algorithm.

Neural networks will learn to do whatever they are trained to do. If we wanted to train one to make astrophotos look like they were painted by Van Gogh, for example, we would present it with amateur astrophotos at the input, use Starry Night renditions as "ground truth," and choose a suitable loss function.

To train a neural network to approximate solutions to deconvolution, we must start with the training data. The training input images must be related to the "ground truth" images — the result that the neural network will learn to produce — only by deconvolution. In other words, the "ground truth" images must be perfect deconvolutions of the input images. The details of how this is accomplished are proprietary, but BlurXTerminator's training method satisfies this requirement. Through training, the network converges toward ideal deconvolved results in the presence of noise.

The loss function employed is critically important. BlurXTerminator's training strictly uses a loss function that is in keeping with the available proofs of the theorem. This is in stark contrast to many contemporary training methods such as so-called *Generative Adversarial Networks* (GANs). In GAN training, the loss function is the output of a second neural network (the "discriminator") which learns to declare the output from the first network "real" or "fake." As the main network gets better at "fooling" the discriminator, the discriminator progressively gets better at spotting the "fakes." "Real" and "fake" can be anything at all — whatever the discriminator is trained for.

Neither this nor any similar generative methods were used to train BlurXTerminator's network work because 1) they lack a rigorous closed-form loss function that will drive the network toward approximating a known mathematical function, and 2) they can very easily teach the main network to become inventive. This last property is in fact why GANs are used in many "style transfer," artificial face generation, and similar networks for which inventive behavior is the goal. They can be fantastically good at *fabricating* detail, and there is evidence that many popular photography sharpening tools were trained using methods such as these, but that is not what we want in a deconvolution tool.

The final requirement is that the internal construction of the neural network must only use the linear and nonlinear operations specified by the available proofs of the theorem. This includes the "activation functions" of the individual nodes within the network. While the details are again proprietary, all of the operations within BlurXTerminator's neural network satisfy these requirements.

In short, yes, BlurXTerminator produces results that approximate deconvolution to high precision. Since all implementations of deconvolution algorithms are fundamentally approximations, this places BlurXTerminator on par with the classical solutions to this problem in a formal sense.

3.3 Perfection?

No deconvolution tool will ever be perfect. As mentioned before, deconvolution is an *ill posed* problem, and necessarily involves guesswork on the part of whatever algorithm is doing the deconvolving. Even the classical algorithms are discussed in terms of *maximum liklihood* estimations — they produce *good guesses* at what the original scene looked like. BlurXTerminator attempts to do the same.

Compared to classical deconvolution methods, a neural network based approach is an entirely different animal. Internally they use some of the same mathematical operations (convolutions), but they are also full of nonlinear

operations that classical approaches tend to avoid. This is in fact part of what gives neural networks their power and versatility, but it can also sometimes cause odd or otherwise incorrect results.

Another major difference is that it is generally not possible to fully trace the reason why a neural network produces a particular result in a particular circumstance: neural networks are said to be *opaque.* Networks such as the one at the heart of BlurXTerminator perform hundreds of millions to billions of computations on a single image, using tens of millions of learned parameter values. Decisions about this or that feature or pixel value are spread across these computations in a "holographic" fashion. Tracing the exact path that led to a particular result is thus quite difficult, and rarely useful. Attempting to hand-modify parameter values to produce a better result in one case will inevitably cause worse results in any number of other (unpredictable) circumstances. Better training data and methods are the standard route to improving a given neural network's performance.

Great care has been taken in the design and development of the neural network architecture, and especially in the training methods employed, to try to ensure that BlurXTerminator produces results that are faithful to reality and not "inventive." It is not and never will be perfect in this sense for all images produced by all instruments. Star centroids may not be perfectly maintained. Total flux may not be perfectly conserved. Its "best guess" at what detailed structure underlies the original blurred image may not be perfectly faithful to reality in all cases.

A consequence of this is that BlurXTerminator should never be used for quantitative or scientific applications. Neural networks and other forms of "machine learning" *are* indeed finding applications in scientific astronomy, but most often in the form of tools fine-tuned for particular instruments, not general-purpose solutions like BlurXTerminator. Their results are still looked at with a bit of a jaundiced eye, and surprising or potentially important results are always cross-checked using traditional, more transparent methods.

3.4 Avoid over-processing

BlurXTerminator has been designed to avoid "over-sharpening" as much as possible if used properly. It nevertheless can be done, especially if the PSF Diameter parameter is set to a fictitious value, or if it is applied multiple times. There is nothing that prevents this, and it can produce some visually pleasing results, but it is no longer deconvolution.

The same is true of the classical deconvolution algorithms if, for example, they are supplied with a fictitious point spread function. Any tool can be used for purposes other than that for which it was intended. A scalpel can be a precision, life-saving tool or a murder weapon, depending on how it is wielded.

The stewardship of our photographic data is in our hands — it is up to each of us whether our final images are faithful representations of reality or grotesque, over-processed messes.

3.5 Future improvement

Like other AI-based tools from RC Astro, BlurXTerminator's neural network will continue to be improved. Targets for future versions include:

- Stellar *aureoles:* diffuse glow around very bright stars produced by some instruments or under some observing conditions (e.g., thin cirrus clouds)
- Field curvature (non-stationary defocus) — this already may be much less noticeable after applying BlurXTermintor, but field curvature is not currently explicitly addressed
- Additional coma and astigmatism profiles
- MacOS: Better/automatic handling of very high dynamic range regions of certain images to eliminate occasional slight posterization due to float16 arithmetic
- Handling of white image borders — this sometimes happens during overscan calibration, and currently causes the neural network to overflow and generate a white output for any tiles that include these borders. Crop white borders away as a workaround.

3.6 Support

BlurXTerminator support can be obtained from the RC Astro web site.

3.7 A technical analysis of resolution and sampling

Much has been written and said about under-sampling and over-sampling in astronomical images. Lively debate erupts periodically on this topic. Indeed, it seems popular at the present time to apply 2x drizzle integration to

sub-exposures that are already over-sampled, regardless of whether it makes any sense to do so. There seems to be confusion as to what is even meant by the terms *sampling, under-sampling, over-sampling,* and most importantly, *resolution.* One suspects that any debate and disagreement stem mostly from a lack of precision and clarity in the usage of these terms. Therefore let us begin by clearly defining them:

Resolution: the minimum *angular feature size* that can be distinguished (*resolved)* by an imaging system. The technical definition of this term as compared to its commonly-used meaning is a primary source of confusion — the common meaning can fail us when trying to understand the imaging process. For example, cameras are often spoken of as having "60 megapixel resolution" or some such. The number of pixels on a camera's sensor has exactly nothing to do with an imaging system's angular resolution.

A given optical system has a minimum angular resolution that is fundamentally limited by diffraction. Atmospheric seeing and the quality of the optics may limit it further. This is completely described by the *point spread function* of the optical system. How much of this resolution is actually captured in the imaging process is a function of the camera sensor's *pixel size* (not its total number of pixels) in relation to the size of the system's point spread function.

Sampling: the process of converting a signal that varies continuously at every point into a set of discrete values, or *samples.* In our case, the signal is light, and the samples are pixels. Any meaningful variation of light across the area of a single pixel is lumped into that one pixel, and the information about that variation is lost.

Under-sampling: not having small enough individual samples (pixels) to adequately gather all of the meaningful information present in an instrument's point spread function. Information (fine detail) is lost when under-sampling. This isn't necessarily a disaster: often a trade-off is being made, perhaps between resolution on one hand, and field of view or available/affordable camera technology on the other.

Over-sampling: having individual sample sizes (pixels) that more than adequately capture all of the meaningful variation in an instrument's point spread function. No information is lost when over-sampling, but excessive over-sampling does not add any meaningful information, either: it just consumes disk space, increases image processing time, and unnecessarily burdens algorithm developers.

As will be demonstrated shortly, there is not a sharp dividing line between under-sampled and over-sampled. As an image becomes slightly under-sampled, some fine-scale information begins to be lost, but image quality does not "fall off a cliff." If the choice is available, it is better to be slightly over-sampled than slightly under-sampled.

The remainder of this section is a simple analysis using basic signal processing concepts that will hopefully further illuminate this topic. This should be straightforward to follow for anyone with a reasonable grounding in signal processing theory. For those not having this background, here are the conclusions:

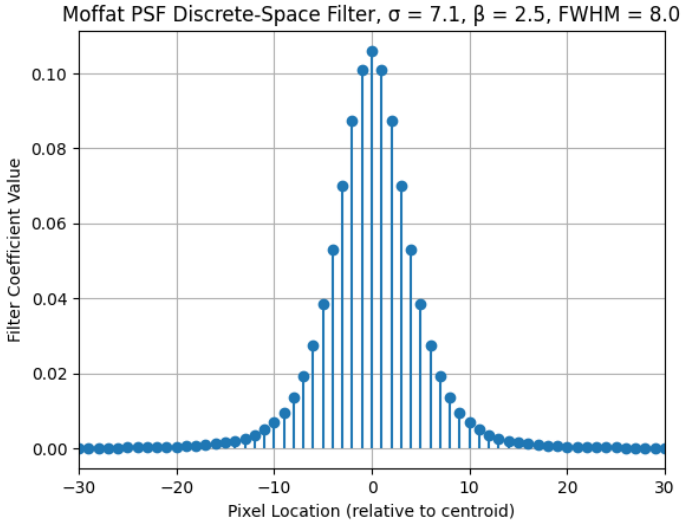
- Images with PSF FWHM values (star diameters) of 4 pixels are generously sampled.
- Images with 4-8 pixels FWHM are over-sampled, but probably not enough to justify down-sampling by 2x unless they are quite noisy.
- Images with greater than 8 pixels FWHM are more than 2x oversampled, and can be down-sampled by 2x with no significant loss of information. IntegerResample in Average mode should be used to minimize aliasing of noise and maximize SNR in the down-sampled image.
- Performing 2x drizzle integration of sub-exposures to recover resolution is only meaningful with under-sampled data (FWHM values less than 4 pixels). 1x drizzle integration of over-sampled data may make sense to eliminate the interpolation artifacts produced by other registration methods, provided there are enough sub-exposures with sufficient dithering to avoid drizzle artifacts (dry pixels).

Note that these conclusions do not depend on the aperture of the optics, the focal length, the focal ratio, the pixel size, nor the seeing at the observing location. These are important when selecting the components of an imaging system and choosing where to put it, but once a system is constructed it will produce whatever PSF it will produce, and that PSF will be sampled by the camera's pixels. The analysis below demonstrates that the achievable angular resolution of a given imaging system becomes limited by the point spread function, not the pixel size, once 4 or more pixels span the FWHM of the PSF.

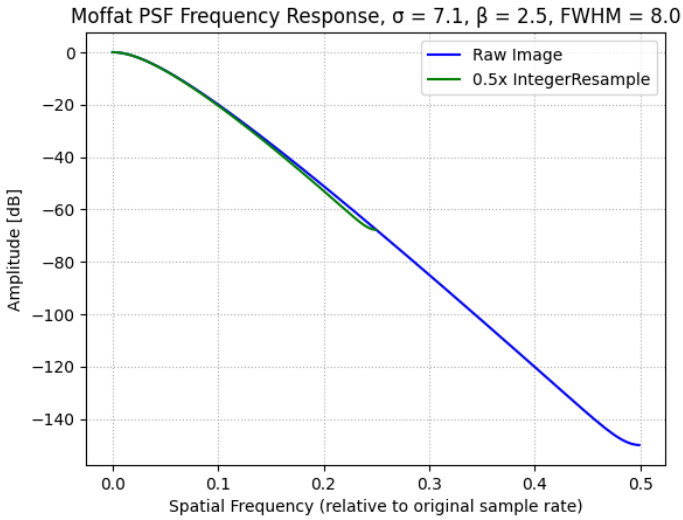
3.7.1 Analyzing resolution and sampling using signal processing concepts

Our images are the result of sampling a signal that has been convolved with a point spread function. This is easily modeled as a discrete-time (or rather discrete-space) filter, from which we can analyze the process in terms of the familiar signal processing concepts of frequency response and aliasing.

PSFs typically encountered in astronomical images are very well represented by a Moffat distribution. Here is a plot of a one-dimensional sampled Moffat PSF viewed as a discrete-space filter. This filter has Moffat distribution parameters chosen such that it has a FWHM diameter of 8 pixels.



Here is the spatial frequency response of this filter, along with the effective 0.5x IntegerResample frequency response, which is just two-sample box-car averaging followed by decimation. This response is referred to the original sample rate to make comparison easy:



As can be seen, the attenuation at the Nyquist spatial frequency by this PSF in the raw image is over 140dB, putting any signal at this frequency *well* below any imaginable real noise floor. Even at half of the Nyquist frequency the attenuation is around 70dB: information at this spatial frequency has been attenuated by a factor of roughly 3,000. This can be considered a reasonable anti-aliasing filter for decimation by a factor of two. (Anti-aliasing of noise between the half-Nyquist and Nyquist frequencies is effectively accomplished by the averaging operation in IntegerResample.)

Down-sampling an image with this PSF by 2x will have *no significant effect* on the actually-recoverable signal present in the image, and there will be no significant aliasing. Yes, it is *theoretically* possible to have an image with such high SNR that some minor loss of information or aliasing would occur, but in *practical* terms for us it is not: we can only dream that our raw images would be that free of noise.

The attenuation values above vary a bit with the *beta* parameter of the Moffat PSF, but not enough to make any difference in this conclusion. Even with an extreme example of a Lorentzian PSF (equivalent to Moffat with *beta* of 1.0), Nyquist attenuation is 100dB, and half-Nyquist is about 55dB.

This is why the maximum PSF Diameter value in BlurXTerminator is 8 pixels. Any image having stars with FWHM values greater than 4 pixels can already be considered over-sampled. Allowing for up to 8-pixel FWHM values provides 2x overhead for integer resampling without significant loss of information.

3.8 GPU acceleration

On MacOS, use of a GPU (or "neural engine" on Apple silicon based Macs) to accelerate neural network computations is automatic and handled by the "CoreML" library provided by Apple. This library makes the decision whether or not a particular hardware configuration is useable to accelerate BlurXTerminator's neural network. Most Macs of recent vintage benefit from acceleration with no additional configuration needed.

On Windows and Linux machines, the neural network computations are performed using the TensorFlow library provided by Google. A CPU-only version of this is installed by default with PixInsight on hardware that will support it. If your machine has a compatible NVIDIA GPU, it may be possible to dramatically accelerate BlurXTerminator and other neural network based tools.

Unfortunately, NVIDIA does not make it easy for small developers to license all of the additional software libraries needed to enable this. Accomplishing the acceleration is thus unfortunately a complex task that involves various downloads and installations, setting environment variables, etc. It can get even more confusing considering that the versions of all of the downloaded components must be compatible with one another.

For those who have the technical skills and feel up to this task, here is a brief guide for Windows machines. This is a simplified version of William Li's excellent guide to accelerating StarNet. Similar instructions are covered in this NVIDIA guide, which also has instructions for Linux machines.

If you have any doubt about being able to perform these steps successfully, find a tech-savvy friend to help. This procedure will likely upgrade the graphics driver for your GPU — if this might interfere with other applications that depend on a particular graphics driver version, consider backing up your system so you can revert if neeeded.

3.8.1 Compatibility

You will need an Intel/AMD x64 system running Windows 10 or later, and an NVIDIA GPU with CUDA compute capabilities of version 3.5 or higher. GPUs with less than 2GB of on-board RAM may not be sufficient. Check this NVIDIA page to verify your GPU's capabilities.

You will also need administrator privileges for your user account to make many of the changes. Windows will likely prompt for permission to perform a number of the actions.

3.8.2 Download and install the NVIDIA CUDA toolkit

"CUDA" stands for Compute Unified Device Architecture, NVIDIA's name for a set of software libraries that allow for general-purpose computing to be performed on many of their graphics processors (GPUs). The CUDA toolkit installer can be downloaded from this NVIDIA page. Select Windows, x86_64, your Windows version, "exe (local)," and finally click the download button. Run the installer and select Express installation to install all components.

This will also update the graphics driver for your GPU, ensuring that it is compatible with the version of the CUDA toolkit in the download.

The installer should also set a number of environment variables that will be needed later. The files comprising the toolkit should be installed in a location such as `C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v11.8`. The last bit is the toolkit version number, the latest of which is 11.8 at the time of this writing.

3.8.3 Download and install cuDNN files

The "cuDNN" moniker refers to yet more software libraries that provide for accelerating "deep neural network" computations on CUDA-enabled devices. Downloading this component requires an NVIDIA developer account, which can be created free of charge.

The cuDNN libraries can be downloaded from this NVIDIA page. Once you go through the process of creating an account and verifying your email address, you should be presented with a list of installers for various operating systems. Click the link labeled "Local Installer for Windows (Zip)."

Even though this is labeled as an installer, it is not. It is a collection of files in a compressed archive, only some of which are needed, and which must be copied manually to the required location. In the ZIP archive, locate the `bin` folder. Copy the contents of this folder to the `bin` folder in the CUDA toolkit installation from above, e.g., C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v11.8\bin.

None of the files from the `lib` or `include` folders are needed unless you will actually be developing your own GPU-accelerated neural network software.

3.8.4 Download and install the ZLIB compression library

There is a data compression software library that the above libraries depend on for some operations. It can be downloaded here. Decompress the downloaded archive and locate the `dll_x64` folder within. Copy the file named `zlibwapi.dll` to the CUDA toolkit's bin directory as above.

3.8.5 Download and install the GPU-enabled TensorFlow library

The TensorFlow project maintains different versions of a software library called `tensorflow.dll`. It is this library that BlurXTerminator and other neural network based tools use to perform computations. The GPU-enabled version of the `tensorflow.dll` library in turn depends on the CUDA and cuDNN libraries installed above.

The version of `tensorflow.dll` that is installed with PixInsight supports CPU operations only. A version that supports GPU acceleration can be downloaded from this TensorFlow page.

- Look for the entry labeled "Windows GPU only" and download that ZIP archive.
- Decompress it and look in the `lib` folder within to locate the `tensorflow.dll` file.
- Locate PixInsight's `bin` folder on your hard drive, usually `C:\Program Files\PixInsight\bin`
- Rename the `tensorflow.dll` file found there to something like `tensorflow_cpu.dll`. This is the CPU-only version of the TensorFlow library distributed with PixInsight. Renaming rather than replacing it allows you to easily revert to it if something goes wrong.
- Move the new `tensorflow.dll` file that was downloaded into PixInsight's `bin` folder.

3.8.6 Verify/set environment variables

Installation of the CUDA toolkit above should have set some environment variables that are needed so that the `tensorflow.dll` library can find all of the GPU acceleration goodies. Check these environment variables as follows:

- Launch the Windows environment variable editor in Control Panel
- Under "System variables", there should be a `CUDA_PATH` variable set to `C:\Program` Files\NVIDIA GPU Computing Toolkit\CUDA\v11.8, with perhaps a different version number depending on what you installed above.
- There should also be a `CUDA_PATH_V11_8` variable (or similar depending on your CUDA toolkit version) pointing to the same location.
- The system `Path` variable should include the CUDA toolkit `bin` and `libnvvp` folders: `C:\Program` Files\NVIDIA GPU Computing Toolkit\CUDA\v11.8\bin and `C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v11.8\libnvvp`.

One additional environment variable should be set to tell the TensorFlow library to only allocate as much GPU memory is needed, as opposed to its default behavior which is to allocate all GPU memory:

- Create a new System environment variable named `TF_FORCE_GPU_ALLOW_GROWTH`, and set it to `TRUE`

After closing the environment variable editor, check that the changes have taken effect. Launch a DOS Command Prompt and run the `set` command. This will list all the currently set environment variables and their values. If you don't see the changes that were made, it may be necessary to restart your machine for them to take effect.

3.8.7 Enjoy fast neural network processing

That should complete the setup for accelerating neural network based computations — launch PixInsight and enjoy. When you first execute BlurXTerminator on an image, it may take longer initially to get going — this delay is the software libraries above mapping the neural network operations to highly parallel GPU hardware operations. After this initial delay, processing should run much, much faster than before.

Any other applications or plug-ins on your machine that use TensorFlow to run neural nets should also be able to be accelerated by replacing the instances of the `tensorflow.dll` file that they load.

Generated by the PixInsight Documentation Compiler script version 1.6.9 on 2022-12-14 18:45:18 UTC