PyAstro for GIMP

Gimp Python Plug-ins for Processing Astro Images

Bill Smith 2018

Introduction

GIMP is a free image processing program that is freely available from its website at https://www.gimp.org/. It is developed by the Linux community and is often included in the default install of various flavours of the Linux operating system - particularly those derived from Ubuntu, which are currently very popular. Versions of the package is also available for Windows and Mac OS X at the GIMP website. It is said to rival Photoshop in capability and scope, though some dispute this. What is indisputable however, is that it is extremely powerful and versatile and, being free, represents exceptionally good value! The learning curve is a little steep, but there's a lot of free information on the web and the pay-off when it has been mastered is considerable.

As a believer in *free software* (having spent my career writing free scientific software for academic use) I have been using GIMP for over a decade now to process my astronomy images. These I obtained using a DSLR camera with several combinations of mounting, lens and telescope. In obtaining my images I have had to deal with usual hazards, such as poor visibility and light pollution (here in Cheshire in the UK), so I depend a great deal on image processing with GIMP to rescue my images from oblivion. It is a measure of the strength of GIMP that it allows me to carry out most of the processing I need to do, without me resorting to other packages.

My general approach involves taking multiple short exposures (< 30s) at high ISO (400-1600), which, as well as reducing the effect of light pollution, also means that tracking the camera accurately across the sky is less of an issue. Along with the astronomical images (`Lights'), I also take a few dark exposures (`Darks') for later use in correcting for in-camera noise. If I am feeling conscientious, I also take some flat frames (`Flats') to correct for vignetting and on-sensor dust. Otherwise, I try to handle these defects using GIMP.

In the past I worked with image files in JPEG format, since this was an easy option and I was able to improve the images quite a bit using standard methods. Dark subtraction, image stacking, colour correction and sharpening all helped a great deal. However, as time progressed I learned the limitations of this approach and switched to RAW image files. The main issue was that the sensor in my Nikon camera recorded colour to a depth of 12 bits per channel, and JPEG only gave me 8. Since many of the interesting things in the night sky are faint, these four extra bits can make a big difference in the quality of the final image. RAW files offered the possibility of reclaiming all 12 bits, but there were a couple of issues when using RAW files with GIMP.

Firstly, GIMP does not directly read RAW formats and so the image files had to be converted into another format, typically 16 bit TIFF files, beforehand. A number of free programs can do this, including IRIS, DCRaw and UFRaw which are arguably the best known[1]. (There is also a UFRaw based plug-in for GIMP called gimp-ufraw that allows RAW files to be accessed from within GIMP. However my experience with this is mixed; it's great when it works, but with new upgrades of GIMP I often struggled to get it

1 Note: Software for this purpose may be also available for free from your camera manufacturer, so check it out.

working again.) My personal choice is DCRaw, which works as a command line program on Windows, Linux and OS X platforms. On the Linux system I use (Ubuntu), the command line

dcraw -v -w -o 0 -q 3 -4 -T *.NEF

is enough to convert a whole directory of Nikon NEF files to 16 bit linear TIFF[2]. (A similar command works for other RAW formats.)

Secondly, most versions of GIMP to date can only read colour image files (even 16 bit TIFFs) to 8 bit colour depth, discarding whatever extra bits were available. With DCRaw I found a rough-and-ready antidote to this. Adding the key '-b 16' to the command line given above boosts the image brightness by the factor 16 – effectively shifting the 4 lowest bits into the accessible 8 bits of each channel. This has the effect of overexposing bright pixels, so stars become uniformly white, but faint objects show more detail. Fortunately however, since version 2.9 became available, GIMP is now capable of handling 16 bit colour and such *fudges* are not necessary. Ideally you should acquire this, or a later version, to maximise your image processing power. At the time of writing this, I am using version 2.10.6.

When processing astronomical images, I use GIMP for dark subtraction, image stacking, field flattening and image enhancement (in particular colour correction, brightness adjustment, noise reduction and sharpening).

Why Plug-Ins?

Processing images with GIMP can be a labour intensive task, particularly when you have many images to process or when an individual image has numerous blemishes that require a lot of work to eliminate. However, some image processing techniques, even complicated ones, are used repetitively, and these demand a more programmatic approach. Examples of these include dark frame subtraction, light pollution correction and image stacking. There are also many operations that have nothing to do with astronomical image processing as such, but are simply routine management tasks. Such techniques and operations are good candidates for GIMP plug-ins.

In using GIMP for processing astronomical images I identified a number of procedures that arose frequently in my work. Given that GIMP offers a robust and simple interface for user-developed plug-ins and lots of online tutorials on the subject, it was inevitable that I would eventually write these procedures in the form of GIMP plug-ins. It proved to be a worthwhile exercise because they reduced the tedium of my work quite considerably. For this reason I have made them available here for wider use, in the hope that others will also find them useful and perhaps be encouraged to write GIMP plug-ins of their own.

Collectively I call my plug-ins: “*PyAstro for GIMP*”, or “*PyAstro*” for short. They are written in the Python language and are easily incorporated into an installed GIMP package through the in-built Python interface. Having used them a great deal I have found them reasonably robust, though not infallible. Fortunately, GIMP is forgiving when things go wrong (it doesn't crash!) and I make much use of the GIMP Error Console to provide information on any problems the plug-ins identify. Since they are not perfect, users are advised always to work on *copies* of their images rather than the *originals* (which hopefully should be self evident!).

---

2 These 16 bit TIFF files provide only 12 bits per colour channel, limited by the camera!

Obviously PyAstro comes *without warranty or liability.* On the other hand you may freely redistribute the plug-ins or incorporate them into other packages if you think fit, provided no author names, copyrights, disclaimers *etc.* are removed. You may also modify them for new purposes and re-release them (for free) with appropriate acknowledgements. If anybody wants to convert them to C plug-ins - be my guest! I guess the pay-off there would be greater speed, though I don't find the Python originals prohibitively slow. PyAstro should work for 8 bit and 16 bit colour images, but not anything else beyond that. If anyone finds a bug, or better still, a fix, please let me know at stargazy(at)btinternet.com and I will undertake to repair them. (Please give details of what happened!) I will be pleased to incorporate contributed Python plug-ins, with appropriate acknowledgement, if donated. Please remember, if you don't like my stuff, you're not obliged to use it! Hostile feedback will be ignored.

Installing PyAstro for GIMP

To use PyAstro, you need to have both GIMP and Python installed on your computer. GIMP version 2.8 will do if you don't need 16 bit colour, but if you are serious about astrophotography go for version 2.9 or later. You can obtain Python from *https://www.**python**.org/*. *Yo*u need version 2.6 or above, but *not* Python 3, which GIMP doesn't recognise. GIMP should already have the necessary interfaces to Python. You can check this by looking for the *Python-Fu* entry under the GIMP *Filters* menu. (To be certain, you can even try running the *Python console*, but close it immediately afterwards - you won't need it for these plug-ins!). If *Python-Fu* is absent, you must install a later version of GIMP.

You should copy all the PyAstro plug-ins in the PyAstro.tar.gz archive file (available from my website http://stargazy.weebly.com) into the GIMP plug-ins directory. This directory can be hard to find, as it jumps around the directory structure with each different GIMP release, but you can find out where it is for your version through the GIMP *Edit* directory. First click on the *Preferences* entry to open the *Preferences* panel. Then expand the *Folders* selection on the panel and choose the *Plug-ins* option. The available plug-ins directories are listed. Note down the location of the plug-ins directory and then go there on your machine. Insert the PyAstro plug-ins in there. To do this unpack the PyAstro.tar.gz file using an archive program (usually, if you just double click on the file, the archiver automatically opens on Windows or Linux). Select all the .py files and copy them to the plug-ins directory. *Make sure they are marked as executable*, since GIMP will not pick them up if they aren't (this is a good way to disable broken plug-ins). Next restart GIMP and you should see a menu headed ***PyAstro***, with sub-menus: *Colour Tools*; *Effects*; *Layer Tools*; *Misc Tools; and Sharpen,* which are ready to use.

The PyAstro Plug-ins

In this section I describe the plug-ins available in *PyAstro for GIMP*. Firstly however, I should make some general comments. To begin with, the action of the plug-ins listed here can be undone or reversed in the usual GIMP manner. Also, any non-catastrophic problems the plug-ins encounter appear as error messages in the GIMP Error Console. If something odd (or indeed nothing) happens, take a look there. I recommend that PyAstro used with the 16 bit version of GIMP and that 8 bit images are converted into 16 bit using the *Precision* option on the GIMP *Image* menu. This is not an absolute requirement - they work fine with 8 bit images - but I find it more convenient, and you will too, probably.

Some of the plug-ins presented here are not specific to astronomical imaging and could, on that account, have been left out. Some even duplicate functionality already present in GIMP, but I have included them because they are educational (and mildly useful) and because, thinking ahead, they could be used as the basis of new Python plug-ins for other applications. The purely astronomical plug-ins probably don't have application elsewhere, but you never know. One or two of them have been lifted from elsewhere (with internal documentation intact!), but most are my own.

Here is a list of the plug-ins under each PyAstro sub-menu:

1. Colour Tools:

- Auto set black point
- Bayerize image
- Clip image brightness
- De-Bayerize image
- Enhance dynamic range
- Invert layer
- Merge colour channels
- Neutralise sky colour
- Pick out stars
- Purge red sky
- Scale image brightness
- Scale image darkness
- Split colour channels
- Undercut image brightness

2. Effects:

- Enhance star colours
- Enhance star luminance
- Enhance using SVD layers
- Round stars
- Set image dark sky
- Star haloes

3. Layer Tools:

- Divide layers by flat
- Merge all layers
- Save all layers
- Scale brightness of all layers
- Scale darkness of all layers
- Set all layer modes
- Stack image layers
- Stack star layers
- Stack sun layers
- Subtract dark from layers

4. Misc Tools:

- Image file converter
- Image list converter
- Image quick save
- Pixel peek
- Resample image

5. Sharpen:

- High pass filter
- LRGB sharpen
- Smart sharpen
- Unsharp mask

The entries on each sub-menu will be *greyed out* (i.e. inactive) when GIMP opens. They become bold (i.e. active) once an image is loaded into GIMP.

A Description of the PyAstro Plug-ins

This section describes each plug-in in alphabetical order. (Note: in what follows the term *maximum pixel brightness* means the number 255 for 8 bit colour channels or 65535 for 16 bit colour channels.)

Auto set black point

This is not the same as the GIMP Colours/Levels *black point picker*, which sets an absolutely black point (i.e. colour (0,0,0)) at the sample position chosen and adjusts the whole image brightness accordingly. Instead this utility scans the whole image and calculates the average brightness of each colour channel and uses this to determine what colour to use as the black point for the image. The intention is to reduce the background glare in an astro image in an undramatic way. The effect is similar to shifting the left hand marker on the Levels *input levels* slider to the right towards the main peak in the histogram except of course this is automatic and requires no user input. The result should look less artificial than using the Levels picker. It is a bit slow however and it sometimes decides that the image is not worth processing since the background is dark enough already (watch the Error Console!). Since it needs no user input it is useful as a Python callable routine. No panel is opened with this plug-in.

Bayerize image

This option is located on the *Colour Tools* sub-menu. Its function is to Bayerize an RGB image, as obtained from a DSLR colour sensor. The plug-in opens the panel *python_image_bayerize*, which has one selectable user control: the Bayer mask, which is chosen from a list of options shown as:[BG][GR], [GB][RG], [GR][BG] and [RG][GB]. You should select the one appropriate for your camera. (You can find out which of these applies to your camera by using the free ExifTool program, which lists the Exif data in the image file, which includes the Bayer mask.)

In astronomical imaging, a linear Bayerized image is used to make dark subtraction optimally accurate. If both the sky image and the dark image are Bayerized the process accurately corrects for sensor noise *before* the interpolation of the colour matrix is applied (a process known as de-Bayerization). After dark subtraction the image can be de-Bayerized to get an accurate noise-free image. Note however that recreating the Bayerized image in this way assumes that the in-camera de-Bayerization has not altered the pixel values of the original Bayerized image. This can be ensured using the DCRaw program, when converting from RAW to 16 bit TIFF format. For example, for Nikon NEF files, use the command line:

```
dcraw -v -w -4 -T -d *.NEF
```

This produces a 16 bit greyscale TIFF files, which on loading into GIMP, must be converted to RGB using the *Mode/RGB* option from the *Image* menu. Applying the

plug-in to the converted image, should then produce a true Bayerized image.

Note that the plug-in will produce a Bayerized image whatever Bayer mask you choose, but you must use the correct one in your astronomy work, if it is to have any real benefit.

Clip image brightness

This plug-in option is located on the *Colour Tools* sub-menu. Its function is to limit the individual colour channels of every pixel in the image to some chosen percentage of the maximum. Selecting this option opens the *python_clip_image_brightness* panel, which which features three sliders for the red, green and blue channels respectively. Each slider ranges for 0 to 100%, and the actual value sets a ceiling on the magnitude of the corresponding colour channel for every pixel of the image. So, for example, a value of 25% sets the limiting magnitude of an 8 bit colour channel to 64 and for a 16 bit channel to 16384. Each colour channel may be set to a different limit. An example use of this plug-in is to selectively clip the brightness of the stars, to use as a mask for selection purposes.

De-Bayerize image

This option is located on the *Colour Tools* sub-menu. Its function is to deBayerize the image produced by the *Bayerize image* plug-in described above. This plug-in opens the *python_image_debayerize* panel, from which the required Bayer mask can be chosen. The options are: [BG][GR], [GB][RG], [GR][BG] and [RG][GB]. You should select the one appropriate for your image and it should be the same as that used when the image was originally Bayerized. See the the Bayerize image section above for more details.

Note that this plug-in will produce a de-Bayerized image whatever Bayer mask is chosen, so be careful to use the correct one. It should also be noted that this plug-in uses a simple linear interpolation, which may not be optimal for some users.

Divide layers by flat

This is located in the *Layer Tools* sub-menu. Its function is to divide all the layers of an image stack by a suitable `Flat', to obtain an image that is corrected for both vignetting and dust on the camera sensor. Selecting this option opens the panel *python_layer_divide_flat*, which hosts a file selector allowing the choice of the image to use as the Flat. This Flat is then divided into *all* layers in the image stack. It also works with images that have only one layer.

Enhance dynamic range

This plug-in is found in the *Colour Tools* sub-menu. It is an implementation of a method accredited to astrophotographer Jerry Lodriguss and is concerned with the construction of composite astro images that exhibit the full dynamic range of the subject. This approach is needed for subjects like M52 in Orion, for which a short exposure captures the details at the heart of the nebula and a long exposure captures the fainter details of the halo, but no single exposure can capture both. Lodriguss' method blends together two such exposures using a *layer mask*, so that the essential details of both shots can show through in the final image. This plug-in was developed to make the procedure simpler.

The plug-in can be used in two ways. Method 1 starts with just one astro image, which

is a short exposure of the target constructed from a stack of many shots, (and is therefore of high quality). The plug-in duplicates the shot in the layer stack and multiplies the brightness of the first copy by a factor chosen by the user. This becomes the *effective* long exposure shot, while the second copy remains short exposure. The two are then blended using the Lodriguss method. The result is a layer stack of two images. If you like the result as it is you can flatten the stack to obtain a final image. Most likely however, you will want to tweak it some more. I recommend you use the GIMP *Colour/Levels* panel for this, tweaking first the layer mask of the short exposure image to adjust its transparency, then afterwards use Levels to fine-adjust the brightness of either image to bring out the details.

In Method 2 the user supplies both the short and long exposure shots (again, hopefully, good quality, multi stacked images). The long exposure shot should be opened first and the short exposure shot opened as a second layer on top of it in the layer stack. You can brighten the long exposure a little if you wish (the plug-in allows this), but otherwise the Lodrigus procedure can applied immediately. Once again you are left with two layers for further tweaking.

Note that once the final image is obtained by either method it is possible to go through the procedure again using Method 1. This may result in a further increase in dynamic range, provided the long exposure image has not been brightened too much in the first round. As always, it is best to experiment.

When this plug-in is selected it opens the *python_enhance_dynamic_range* panel with three controls. The first is the *brightness factor* slider. This is the factor by which the brightness of the *long exposure* image will be increased. This can be anything from 1 to 10 for Method 1 (values 4~5 are recommended). For Method 2 it is probably best to start with a brightness factor of 1, since the image is supposedly bright enough already. The second control is the *Gaussian blur* slider, which selects the size of the blur to be applied to the layer mask of the short exposure image. Blurring is necessary to smooth the blending of the two images in the stack. Values between 25 and 75 pixels are recommended – experiment! The last control is an option to set an *automatic black point* on the long exposure image, in order to diminish the amount of background glare that multiplication of the brightness can introduce. If the background is suitably subdued anyway, this is not necessary, and can in fact be done after the Lodriguss procedure is complete (by other means) but before flattening the image.

Users are advised that while this does not offer a fully automatic process, the plug-in does get you quickly to a point where your own adjustment of the image quality can be applied.

Enhance Star Colours

This plug-in is found under the *Effects* sub-menu and is experimental in nature. It is meant to do what it says – enhance the colours of stars in an astro-image. However, this may not be too successful if there is light pollution (e.g. sodium lamps) or if there is coloured nebulosity in the shot (which will also be enhanced). It’s worth a try though, if your star images look a bit pale.

The plug-in opens a panel with a single slider defining the degree of colour enhancement.

Enhance Star Luminance

Another experimental plug-in found on the *Effects* sub-menu. In this case it meant to enhance the luminance of the stars in the photograph. As with *Enhance Star Colours*

plug-in above, it does not affect just stars, so it should be used selectively. (It is arguably less destructive when inappropriately applied however.)

The plug-in opens a panel with a single slider setting the width of a Gaussian blur that is part of the enhancement process. Ignore this definition and regard it as a parameter for enhancing the luminance: larger values make brighter stars.

Enhance Using SVD

This plug-in is another experimental offering from the *Effects* sub-menu. It has been found to enhance the colour and brightness of some of my star pictures. Though its effects can be unexpected it is worth a try on pictures that are lacking in vibrancy. It was an accidental discovery when I was trying to do something else, which shows the value of just playing with GIMP!

Selection of this opens a panel with a slider mysteriously called the *dodge layer opacity*. This is best thought of as a parameter scaling the magnitude of the overall effect: the higher the value, the more striking the result.

High Pass Filter

This is found under the *Sharpen* sub-menu. A standard way to to sharpen an image is to make a high pass filter, which is derived from the original image and only contains information about the edges of objects, while the rest of the filter is uniformly gray. Blending the filter with the original image using *overlay* mode results in an image with enhanced edges, which thus looks sharper.

Selecting this plug-in opens a panel where the user specifies a single *Gaussian blur parameter* using a slider. This parameter is used to ‘soften’ the background of the filter and help make the edges on it more prominent.

Image file converter

Obtained via the *Misc Tools* sub-menu, this `no-frills' plug-in allows the quick writing of an image file in a range of chosen formats, in the same location as the original image. The option opens the panel python_image_file_converter to allow the user to select the required format from a presented list. Raw images are not an option. To prevent accidental file overwrite, the input and output files cannot be of the same type.

Image list converter

Located on the *Misc Tools* sub-menu, this plug-in allows the quick conversion of a sequential list of image files of the same type into some other image file format. All the files to be converted must reside in the same directory and numbered sequentially, with names of the form xxxxnnn.typ, where xxxx is the same character string for all the files, nnn is a sequential number and .typ is the image file type (e.g. .JPG).

Note that the first file to be converted must be loaded into GIMP, to provide a template for the additional files. Selecting *Image list converter* opens the panel *python_image_file_converter* and the number of files to be converted is entered into a text box and the required file format selected from a menu. The plug-in will then proceed to read and write the sequence of image files. The new image files are written in the same directory as the originals.

The number of files entered on the panel should equal the total number *assuming the*

sequence is complete. If you enter the true number of files to convert and ignore those missing from the input sequence, you may not successfully convert all the files you expect to.

Input and output files cannot be of the same type. Raw images are not an option.

Image quick save

Found under the *Misc Tools* sub-menu, clicking on this option results in an immediate write of the current image back to its source directory. It opens no panel. This is probably unnecessary, given that the *Overwrite* option on the GIMP *File* menu does the same thing. However, you may have use for a version in Python if you are a plug-in writer.

Invert layer

This plug-in is found on the *Colour Tools* sub-menu. This option simply inverts the colours of the selected GIMP layer (with no intervening panel). It is functionally identical to the *Invert* option on the GIMP Colours menu and is arguably redundant. However it is useful as a simple template for a Python plug-in.

LRGB Sharpen

Found in the *Sharpen* sub-menu, this sharpening method is based on a technique devised (independently) by Okano and Dalby, which works by sharpening the luminance layer of an image while simultaneously softening the coloured layers by blurring. Blending the sharpened luminance layer with the blurred RGB layers, results in an image with enhanced edges and smoothed areas of colour. Thus the method lacks the noise that would otherwise appear in coloured areas if the whole image was sharpened. Though intended for advanced astrophotography employing separate L, R, G and B exposures, it is applied to RGB digital camera images by constructing a luminance layer from a colour desaturated copy of the original image.

Selecting this option opens a panel from which the user may select the method of colour desaturation (luminance, average, lightness or luma), the radius and strength of the unsharp mask used to sharpen the luminance layer, the blur radius of the Gaussian blur applied to the RGB layers and finally, a brightness factor. This last factor is needed because the method tends to darken the image and this has to be compensated for.

Merge all layers

Located on the *Layer Tools* sub-menu, the function of this plug-in is to merge all the *visible* layers in an image into a flat image. The blending mode used is *Normal* and the opacity of each layer is set to 100/(n+1) percent, where n is the layer number (starting from the bottom of the stack with n=0). There are no user defined parameters in this plug-in so it does not open a panel.

Merge colour channels

This plug-in option is found in the *Colour Tools* sub-menu. Its function is to reverse the operation of the *Split colour channels* plug-in described below, which separates out the colour channels of an image as distinct layers. Together, the *Split colour channels* and *Merge colour channels* resemble the GIMP *Decompose* and *Recompose* options in the *Colours/Components* menu, but work with RGB layers rather than greyscale, for when coloured layers are more desirable. See the *Split colour channels* section below for

further comments. There are no user defined parameters associated with this plug-in, so no panel appears.

Neutralise sky colour

An option in the *Colour Tools* sub-menu, this is designed to ensure the image sky is a neutral colour. It works best when there is a single dominant peak on the left of the image histogram, otherwise it may affect the colours of stars unduly. (Consider the alternative: *Purge red sky*, described below.) It works by ensuring the histograms of the red, green and blue components of the image are centred in the same place in the histogram and with the same width. Selecting this plug-in does not open a panel as there are no user parameters to input.

Pick out stars

This option appears under the *Colour Tools* sub-menu. Selecting it opens the *python_pick_stars* panel, which hosts three sliders defining the red, green and blue brightness limits for star selection. Stars with brightnesses less than the set limits will not be selected. Brightnesses are specified as a percentage of the maximum pixel brightness. The resulting image shows only the selected stars against a black background. This is useful as a selection mask for image processing specific to the brightest stars.

Pixel peek

Pixel peek is found under the *Misc Tools* sub-menu. Its function is to list the contents of all the bytes belonging to a specific pixel. This is not, frankly, much use to astronomers, but it is useful when checking plug-ins that operate on individual pixels. The option opens the panel *python_pixel_peek*, where the user is chooses the coordinates of the pixel of interest (default: (0,0)). (The GIMP image panel, on the bottom left, indicates the coordinates of the cursor location, if a specific location is required.) The byte contents of the required pixel and the number of bytes per pixel (BPP) will be listed in the GIMP Error Console when the OK button is clicked. This resembles the GIMP Pointer Information, except that the byte contents are displayed, not the colour percentages.

Purge red sky

This option appears under the *Colour Tools* sub-menu. As its name implies, it is used to remove the red colour arising from light pollution from astronomical images. However it can correct for other sky colours (should they ever occur!) and can, simultaneously, correct for vignetting. The method is based on the well-known trick of subtracting a blurred copy of an image from its original, but with a few refinements. Here the image copy is clipped before blurring, so the bright stars are less apparent in the blurred image, and the image subtraction is moderated by adjusting the opacity of the blurred image to control the strength of the colour correction. This works well when the image sky shows a strong colour tint, particularly when the image histogram shows more than one major peak. If there is a single peak, the *Neutralise sky colour* option (above) may be the better choice.

Selecting this option opens the panel *python_purge_red_sky*, which hosts three sliders and a toggle button. The sliders set three parameters: the *red sky level*; the *Gaussian blur*; and the *strength*. The first is a limit on the brightness of the copied image, the effect of which is to reduce the brightness of stars in the blurred image. It should be set just beyond the main peak (or peaks) on the left of the image histogram. Note its value is a *percentage* of the maximum pixel brightness. The second parameter is the

radius of the Gaussian blur, in pixels, used when blurring the image copy. The default is 75 pixels. The third parameter is the strength of the blending of the original image with the blurred copy in the final result, defined so that 100(%) represents a full weighting of the copy. (This is equivalent to altering the opacity of the blurred copy at the subtraction stage.) The default is 75%.

The toggle button allows disabling of the image flattening at the end of the process. If this is disabled it becomes possible to experiment with the opacity, hopefully to get a more desirable result before the image is finally flattened.

If the final result of this plug-in is not to your liking, you are encouraged to experiment with the parameters to get a better result.

Resample image

An entry on the *Misc Tools* sub-menu, this plug-in will resample the image by a specified factor ranging from 2 to 6. So, for example, setting a factor 5, will reduce the area of the image (and file hence file size) by a factor of 25. This is useful for combining neighbouring pixels to reduce noise and for shrinking images for transport over the internet. This option opens the *python_resample_image* panel, where the re-sampling factor can be set. It is applicable beyond astronomical imaging.

Round stars

This option is found on the *Effects* sub-menu. Its function is to make the images of prominent stars, that are distorted by poor tracking or optical effects, look more spherical, at the expense of making them a little larger in the image. It works by creating several copies of the starting image, rotating the stars in their positions by a different amount in each copy, and merging all the copies down usng the *Darken only* blending mode. The quality of the final image largely depends on the number of image copies made.

Selecting this option opens the *python_round_stars* panel, which hosts three sliders for the user to set the control parameters. The first slider sets the *Star minimum*, as a percentage of the maximum pixel brightness. The default is 85%. Any star less bright than this will not be rounded. The second slider sets the *Star scale factor*, which determines how large the final star will appear in the image. This should be set higher than the 1.5 default if the star is very non spherical. The last slider sets the *Star roundness* parameter., which can range from 4 to 10, with 4 as the default. This represents the number of image copies used to construct the final image. Use a higher number than the default if the resulting sphericity of the stars is not convincing.

Note you are likely to run out of memory (and crash) if you push the control parameters too far. I find this plug-in works best if only the brightest stars are processed (i.e. the star minimum is set high). You may find applying a tiny amount of Gaussian blurring to the final image will improve the result, particularly the appearance of the fainter stars, which are not processed by this plug-in.

Save all layers

Located on the *Layer Tools* sub-menu, this option allows the quick save of *all* the layers in an image to a sequentially numbered series of separate files of a chosen image type. All the files produced have the names STACKnn.ttt, where nn is a sequential number and ttt is one of JPG, JPEG, PNG, BMP, GIF, TIF or TIFF. Selecting this option opens the panel *python_save_layer_stack*, where the choice of image file type is selected. It is useful for saving a specific stack in a form that can be picked up again

by GIMP or some other package.

Scale brightness of all layers

This is found on the *Layer Tools* sub-menu. Its function is to *multiply* the pixel values of each layer in an image by a user defined factor between 1 and 5, with truncation of the pixel values if they exceed the maximum pixel brightness. This option opens the panel *python_scale_layers_brightness*, where the brightness factor may be set with a slider. The plug-in is the inverse of *Scale darkness of all layers,* also described below.

Scale darkness of all layers

Also found on the *Layer Tools* sub-menu, the function of this plug-in is to *divide* the pixel values of the layers of an image by a chosen factor between 1 and 5. Selection of this option opens the panel *python_scale_layers_darkness*, where the darkness factor may be set with a slider. This plug-in is effectively the inverse of *Scale brightness of all layers* described above. Be aware however that applying *Scale brightness of all layers* and *Scale darkness of all layers* successively may not restore the original layers exactly, due to the effect of the brightness truncation that can occur with layer brightening.

Scale image brightness

This is found on the *Colour Tools* sub-menu. Its function is to *multiply* the pixel values of the selected image by a chosen factor between 1 and 5, with truncation of pixel values if they exceed the maximum pixel brightness. Selection of this option opens the *python_scale_image_brightness* panel, where the brightening factor may be set with a slider. This can be useful to boost faint star images (at the expense of overexposing the bright stars). This plug-in is the inverse of *Scale image darkness* described below.

Scale image darkness

Also found on the *Colour Tools* sub-menu, the function of *Scale image darkness* is to *divide* the pixel values of the selected image by a chosen factor between 1 and 5. Selection of this option opens the panel *python_scale_image_darkness*, where the darkening factor may be set with a slider. This plug-in is the inverse of *Scale image brightness* described above. Be aware however that applying *Scale image brightness* and *Scale image darkness* successively may not restore the original image exactly, due to the effect of brightness truncation in the first operation.

Set all layer modes

This option is located in the *Layer Tools* sub-menu. Its function is to reset the blending mode of all layers in an image stack to a given mode. The plug-in opens the *python_set_layer_modes* panel, where the choice of blending mode can be made. Any of the GIMP blending modes may be chosen. Note that the plug-in always sets the mode of the bottom layer of the stack to *Normal.*

Smart Sharpen

This sharpening option is found under the *Sharpen* sub-menu. It works by first creating a sharpening mask from a copy of the original image. The edges in the copy are isolated (by a rather complicated procedure) and the image converted to grey to make the mask. Next the original image is converted to the LAB colour representation and the prepared mask is added to luminance layer (L) as a *channel selection*. Sharpening is then applied to the luminance layer, but due to the mask, only the edge regions are

sharpened. Meanwhile the colour layers (A,B) of the image are blurred to smooth out any noise. Finally, the LAB image is converted back to a regular RGB image, in which the edges are sharpened and the coloured areas are smoothed. This rather elaborate procedure produces good quality sharpening with low noise and is particularly good for polishing images of the Sun, Moon, planets and images featuring nebulosity. It is also good for general photography. Its main disadvantage is that it is slower than the other options.

Invoking this option produces a panel with sliders for five tunable parameters. The first of these is the width of the edges the user wishes to apply. Secondly, a Gaussian width is required, for the several blurring operations the method requires. (In principle several Gaussian parameters are required, for several operations, but I have limited these to the same for simplicity.) The third parameter is the sharpening radius to be applied to the luminance channel. The fourth is the strength of the sharpening and the fifth the threshold at which sharpening is applied. Parameters 1 to 3 are expressed in pixels, parameters 3 to 5 are those required for unsharp mask sharpening. Parameter 5 can be set to zero for most applications.

The default parameters this plug-in employs,were found to be suitable for images of dimension 3000 x 2000 pixels. I suggest you start with these and experiment to get the better results. If you settle on values specific to your circumstances, you might want to amend the defaults in the Python code to suit your purposes – as you are free to do!

Set image dark sky

This option occurs on the *Effects* sub-menu and it provides a means to create an artificial sky in the background of a real sky image. This is *not* true image processing as such; it merely is a way of making star images more presentable by removing a chaotic background, which will include faint stars and deep sky objects - so beware! Like many techniques, it works best when the starting image is of reasonable quality to begin with, then the changes made are less drastic.

Invoking this plug-in brings up the *python_set_dark_sky* panel, which has four user options: the *Current dark sky level*, which is set as a percentage of maximum pixel brightness and is the level of sky brightness below which pixels are removed and replaced; the *New sky dark level*, which is the mean brightness of the reconstructed sky (again as a percent of maximum pixel brightness); the *Add random noise* option which, if selected, introduces a measure of random noise in the new background sky to improve realism; and finally the *Set noise level*, which is the RMS variance of the random noise added, defined as a percentage of the *New sky dark level*. The default values of the plug-in are merely typical, rather than optimal, and you will need to experiment to find out what's best for a particular image.

Note that the current dark sky level of an image can be found from the image histogram – choose a (percentage) level just off the edge of the dominant peak(s) on the left. The final image can be improved by the application of a tiny degree of Gaussian blurring. The technique does not correct for any sky colour bias, so this is best fixed beforehand.

Split colour channels

This plug-in is accessible from the *Colour Tools* sub-menu. Its function is to separate out the colour channels of an image into separate layers. It resembles the GIMP

Colours/Components/Decompose facility in operation, but instead of producing layers composed as greyscale images, it produces layers that are coloured RGB images. One purpose of this is to allow correction of poor colour superposition by shifting the layers with respect to each other to minimise the effect. Afterwards the layers may be re-merged using the *Merge colour channels* plug-in described above. There are no user defined parameters and hence no panel associated with this plug-in.

Stack star layers

This plug-in is found on the *Layer Tools* sub-menu. As its name implies, it performs the stacking operation that is commonly used in astrophotography to improve the signal-to-noise ratio of an astronomical image, resulting in a smoother and more defined image. It is an indispensable tool for astro-imagers. This implementation requires you first to pre-load a series of (nominally identical) star images as a stack of layers using the *Load as layers* option on the GIMP *File* menu and then to nominate one of the layers as the *reference image* by clicking on it in the layer stack. Clicking on the *Stack star layers* option then opens the *python_stack_star_layers* panel, where the user can define the required control parameters.

Five control options are available. The first is the *Star minimum* slider, which defines the minimum brightness of a pixel to be considered as belong to a star. It is expressed as a percentage of the maximum pixel brightness. The default is 85%. The second parameter is the *Search box width* slider, which is expressed in pixels and defines the width of a square search area over which the plug-in searches to match up identical stars in the different layers of the stack. The third parameter is the *Layer mode* option box, which specifies the layer blending mode of the final stack. The three options are Normal, Addition and Screen. The default option is Normal, but the others may be preferable if the images are faint. The fourth parameter is the *Flatten image* toggle switch, which controls the option to flatten the final image. The default is to flatten, but you may wish to check out some issues before doing this (such as an airplane intrusion) so you have the option to skip this. Last is the *Autocrop image* toggle switch, which as a default applies an automatic image crop when the image is flattened. Using this switch you can chose to cancel the *Autocrop*, but it is ignored anyway if you have deactivated the *Flatten image* option.

Some additional comments are in order:

Firstly, the algorithm employed here does not allow for image rotation. Hence it is of little use with alt-az tracking. With equatorial tracking, if your polar axis is accurately aligned, you should be all right as the algorithm will tolerate a linear displacement of successive star star images, such as occurs when the RA drive does not keep up with the sky between shots. It can even work if the star images are a little stretched (but not too much!) Obviously, if they *are* stretched, they will remain so in the final image.

Secondly, please do display the GIMP *Error Console* while the plug-in is working. It will tell you how many stars it detects in each image and what the relative displacements of the images are with respect to the chosen reference image. If the number of stars is zero, it means you need to lower the *Minimum star* parameter. Beware if it ever indicates an image displacement of 0,0, as then it is possible that it has not found a matching star in the search box area. However, this can also happen if your tracking is very good! The truth will be revealed in the final image – stars will appear as a trail of dots if matching has failed. If this happens repeat the exercise with a larger search box. In this context it is useful to select a reference image that is nearest to the middle of the time period between the first and last images, since this requires the smallest search box, which in turn results in a faster stacking.

Thirdly, stacking does not remove intruders like airplanes or satellites from the final image. These need to be dealt with by special measures. If you stack with the *Flatten image* option deactivated, then change the blend mode of the offending layer to *Darken only* and then from the menu obtained by right clicking the layer use *Merge Down*. Finally, use the *Merge all layers* plug-in (described above) to redefine the layer weighting of the stack and flatten the image. Alternatively you can simply set the offending layer to invisible and it will be ignored.

Finally, remember the plug-in is intended to stack stars; it will not stack any other kind of image!

Stack image Layers

This is an alternative for the star stacker above and is also found on *Layers* sub-menu. It is an experimental version that seems to be a little quicker. It was also relatively simple to write. However, I have a hunch that it is less effective if the images contain too much detail (e.g. too many objects), but it's worth trying. Since it does not work by identifying the stars, it theoretically could be used on other kinds of image, such as the Sun and Moon. No rotation is permitted however.

Invoking this option opens a panel with the same parameters as the star stacker, though the *Minimum star* slider is replaced by a *Minimum brightness* slider, which is used to confine the alignment exercise to the brightest objects in the picture.

Stack sun layers

Located on the *Layer Tools* sub-menu this plug-in is designed to stack images of the sun. It is an adaptation of *Stack star layers* (above) and you are advised to read that section for further details. It is essentially the same method, slightly tweaked for one major star – the sun. The plug-in requires pre-loading a series of nominally identical sun images using the *Load as layers* option on the GIMP *File* menu and then nominating one of the images as the *reference image* **-** by clicking on it in the layer stack. Then clicking on the *Stack sun layers* option opens the panel *python_stack_sun_layers*, where the user can define the required control parameters.

This can also work for images of the full moon, and even other phases (try it!), or indeed, the sun in partial eclipse.

Note however, that the algorithm does not work for images that are rotating.

Star haloes

This option is located on the *Effects* sub-menu. Like the *Set image dark sky* plug-in described above, this is not image processing in its purest sense. Rather it is a tool for modifying images by introducing particular artefacts for aesthetic reasons. In this case, the artefacts are haloes and diffraction spikes, which some find pleasing, but by no means everyone!

Selecting this option opens the *python_star_halo panel*, with three sliders and a selection box to set the plug-in control parameters. The first slider is the *Star minimum*, which defines the minimum acceptable brightness of a star pixel, expressed as a percentage of the maximum pixel brightness. The default is 85%. The next slider defines the *Halo radius,* which is a measure of the size of the halo or diffraction spike in units of the star radius. The third slider controls the Halo brightness, in arbitrary units. Lastly the selection box offers four options: a circular halo; a plus (+) effect; a cross (x) effect and an asterisk (*) effect. Clicking the OK button runs the plug-in,

which produces a new layer above the original in the layers stack. In this location the halo effect may be added to the original image by appropriately resetting its opacity and/or blending mode and then merging the layer down on to the original image. If no halo appears when the plug-in is run, try reducing the star minimum, to ensure the stars are actually being picked up!

If more than one halo effect is required, such as a halo with diffraction spikes, this will require running the plug-in again. Remember not to merge down the first halo layer and make sure that you select the original image layer for processing, so you are not using the previous halo layer as your input. Ideally you shouldn't set circular halo as bright as the diffraction spikes, otherwise the spikes will not be easy to see.

Some further points. Using this plug-in requires some experience to get a satisfactory result. It pays to experiment with the with the halo brightness and size and later with the opacity and blending mode before merging down. In constructing a convincing effect, note that less can be more! Over bright halos detract rather than enhance. Be aware that double stars will not be correctly processed by this plug-in if they merge into each other in the original image. You may have to edit the image to separate the stars by a pixel or two beforehand.

Subtract dark from layers

This option is located in the *Layer Tools* sub-menu and its function is to subtract a chosen image `Dark' image from the stacked layers of an image, to obtain a final image that is corrected for noise from the camera sensor. Choosing this option opens the panel *python_layer_subtract_dark*, which hosts a file selector to choose the required Dark image. This Dark is then subtracted from all layers in the image stack. (This also works with images that have only one layer.) The calibrated layer stack can then be processed with the *Stack star layers* plug-in described above.

Undercut image brightness

This plug-in option is located on the *Colour Tools* sub-menu. Its function is to undercut the individual colour channels of every pixel in the image by some chosen percentage of the maximum. Selecting this option opens the *python_undercut_image_brightness* panel, which which features three sliders for the red, green and blue channels respectively. Each slider ranges for 0 to 100%, and the actual value selected sets the magnitude of the undercut for the corresponding colour channel for every pixel of the image. So, for example, a value of 10% sets the undercut magnitude of an 8 bit colour channel to 26 and for a 16 bit channel to 6554. The undercut for a given colour channel will be subtracted from the colour channel of each pixel. If a negative value results from this, it is reset to zero. Each colour channel may be set to a different limit. An example use of this plug-in is to remove a low level colour background from an image, by setting the undercut to a value beyond the maximum peak(s) on the left of the histogram of each colour channel.

Unsharp Mask

This sharpening option is found in the *Sharpen* sub-menu and is redundant inasmuch as GIMP already has a its own unsharp masking option available under the standard *Filters* menu. However, it occurs here as a Python implementation, which some users could conceivable adapt for other purposes. It is simple and works fine.

Invoking this option opens a panel with two sliders so that the user can select the Gaussian blur radius and the strength of the sharpening, both of which are required by the method. (Note I have chosen to leave out a threshold option, since a value of zero

for this is commonly acceptable.)