You, too, can make useful and beautiful astronomical images at Mees: Lesson 4

Sharpening, stretching, scaling and color-combining

Useful references, besides Lessons 1-3:

- Rob Gendler (ed.) 2013, *Lessons from the Masters* (New York: Springer-Verlag), especially the articles by Adam Block (p. 159) and Johannes Schedler (p. 193).

Software:

- CCDSoft v. 5.
- CCDStack v. 2.9
- FITS Liberator v. 3
- Photoshop CC

Work in progress: NGC 5985 (left) and NGC 5982 (right) in Draco, LRGB, with three times as much L data as last week: now 36x5 minutes in L and 4x5 minutes each in R, G and B (4 hours total). It was moonless and "clear" on Saturday but the atmospheric transmission was poor. I had trouble autoguiding in R, G and B because of this, so I just took more L data.

Significant improvement over last week but there's still quite a way to go, especially in RGB which is still quite noisy in parts of the image where L is bright.

Sharpening images: deconvolution

If your L images have very high signal-to-noise, you can recover some of the angular resolution you lose from our normally not-very-good seeing.

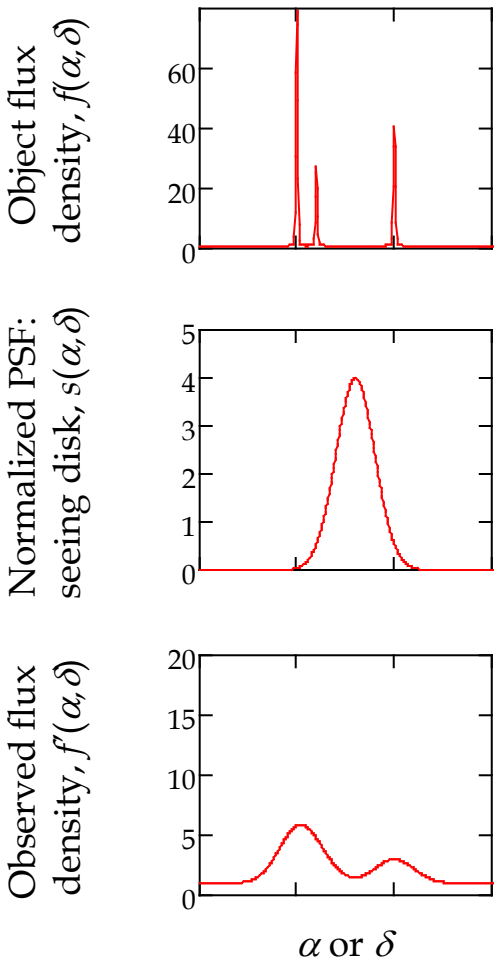
- Atmospheric turbulence broadens what should look like points, in the manner of convolution by a Gaussian:

$$f'(\alpha,\delta) = \iint f(x,y)\,s(\alpha-x,\delta-y)\,dxdy;$$

$$\equiv f(\alpha,\delta) * s(\alpha,\delta) \quad ;$$

$$s(x,y) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \quad .$$

Pont-spread function (PSF), a.k.a. "seeing disk."



Deconvolution (continued)

If one knows exactly what the point-spread function s is, and if there were no such thing as noise or systematic error, then one can determine the object's flux density unambiguously, because of the **convolution theorem**:

> If $f'(\alpha,\delta) = f(\alpha,\delta) * s(x,y)$, and F', F and S are respectively the Fourier transforms of f', f and s, then $F'(u,v) = F(u,v)S(u,v)$.

- Thus one would Fourier-transform the observations (f') and the point-spread function (s), divide the two results to produce F, then Fourier-transform F to produce the object's real flux density f, unsmeared by seeing.

Deconvolution (continued)

However, life is rarely so simple. There is noise and systematic error, and it messes everything up. Separating the measurable functions into noiseless/error-free terms and noise/error, as

$$f'(\alpha,\delta)+\Delta f'(\alpha,\delta)=f(\alpha,\delta)*\left[s(\alpha,\delta)+\Delta s(\alpha,\delta)\right],$$

one sees that the convolution theorem could still help, but additional constraints on the noise terms would be necessary.

- There are rarely enough constraints to do this algebraically, e.g. N equations in N unknowns solvable by linear-algebraic techniques.
- If the PSF is known essentially noiselessly, as in radio astronomy, procedural deconvolvers like the CLEAN algorithm work very well, but in ground-based visible/infrared astronomy (without AO) the PSF is rarely determined sufficiently noiselessly.

Deconvolution (continued)

- If there aren't enough constraints for an algebraic solution, one determines the *range* of solutions for *f* consistent with what constraints there are, and then selects among the range for the solution which is most probable.
- This devolves the question to: how does one rank the solutions by probability? There is no best way to do this, nor a way so far which can be proven to converge on the exact solution for *f*.
- Generally one proceeds by defining a measureable property of the image related to sharpness, and finding the solution corresponding to the maximum or minimum of that property. Routines like this include
  - **Maximum-likelihood (Lucy-Richardson) deconvolution**. Related to CLEAN, as it decomposes objects into a sum of PSFs, but searches for the most likely coefficients in the PSF sum under the assumption that they are Poisson-distributed, like shot noise.

Deconvolution (continued)

- **Positivity-constrained deconvolution.**
  Like Lucy-Richardson, but rectifies the results as one goes along to prevent the final answer from having unphysically-negative flux densities.

- **Maximum-entropy deconvolution.**
  Maximizes the "image entropy" $S = -\sum_i p_i \log_2 p_i$, where p_i is the probability that the difference in DNs between adjacent pixels has the value i. The function has nothing to do with physical entropy, $S = -k_B \ln \Omega$; it's named for the resemblance to this formula as the Stirling approximation applies to it.

- CCDSoft v.5 does Lucy-Richardson deconvolution; CCDStack v.2.9 does positivity-constrained and maximum-entropy deconvolution; and Photoshop CC has all three, and several more in there somewhere. Look under Filter > Sharpen.

Deconvolution (continued)

Caveats:

- None of these methods will improve the resolution of images by much more than a factor of two, and that only at very high signal-to-noise.
- The resolution will be seen to vary across the image, being better (sharper features, smaller stars) where S/N is higher – unlike the original.
- Maximum entropy deconvolution does not conserve energy: the resulting image will have a different total flux density than the original. (L-R and positivity deconvolution are booby-trapped against that.)
- So **NEVER** use them, particularly maximum entropy, on images with which you want to do science.

Maximum-entropy demonstration

For pretty pictures, any may be OK; some images may be better with one method than another.

- Adam Block uses CCDStack's positivity-constrained deconvolution, though it seems to lead to more work in Photoshop.
- I have been getting better results so far with CCDStack's maximum entropy routine.

M101 in L: as observed (top) and maximum-entropy deconvolved (bottom).

Stretching images

CCDs have DN linearly proportional to power collected; each pixel's signal is sent out as a 16-bit floating point number, and stored in the computer as a 32-bit number. None of this matches displays very well.

- Computer monitors only display eight bits of brightness (0-256), for the very good reason that eyes can only resolve that many shades of gray. Printing on paper is similar.
- Thus one must **stretch** (or compress, really) the huge range of astronomical brightness within an eight-bit range for display.
- Photographic emulsion has a useful compressive feature built in to it, called **reciprocity failure**:
  - Hypersensitized emulsion produces image density linear in power at low light levels, but the response becomes more like logarithmic with brighter lights.

Stretching images (continued)

Each of the image analysis programs we use has a variety of ways to stretch images, notably these:

- **Gamma.** This maps power exponentially into signal:

$$DN_{\text{display}} = A\left(DN_{\text{image}}\right)^{\gamma} + B$$

  $\gamma = 1$ is of course the same as the original image, but $\gamma < 1$ compresses the display brightness into a smaller range. A ("brightness") and B ("background") can be set separately.

- **Logarithmic stretch:**

$$DN_{\text{display}} = A\log\left(DN_{\text{image}}\right) + B.$$

  An attempt to mimic what the eye itself does, but it has obvious problems if it's possible for the pixels to have DN values of zero.

Stretching images (continued)

- **Digital development process (DDP).** This is an algorithm, developed by a professional physicist who happened to be an amateur astronomer, which is meant to mimic the useful features of reciprocity failure. The algorithm is iterative and replicates the stages one would see in an exposed photographic plate immersed in developer.
- **Arcsinh stretch.** At large values, arcsinh(x) converges to ln(x); at small values it's linear (like ln(x)) but arcsinh(0) = 0 (unlike ln(x)). So it satisfies all the constraints and embodies several useful properties.
  - It also behaves very much like reciprocity failure in hypersensitized emulsion. The DDP stretch is very similar to the arcsinh stretch, as was first noticed long before the invention of DDP by fitting functions to the response of "analog" developed emulsion: Bunsen & Roscoe 1862, Schwarzschild 1899, Kron 1913. (Yes, that Schwarzschild.)

Demonstration of stretching

Linear

DDP

Demonstration of stretching

arcsinh

DDP

Color composition

CCDSoft, CCDStack, and FITS Liberator all have good ways to take individual R, G and B images and make an RGB color composite. Here are the steps.

- If your L images are binned 1x1 and your RGB 2x2, according to our recommendations in Lesson 2, then load L, R, G and B with L as the reference image, and register (align) the others to it. This will result in an interpolation of R, G and B to 1x1 binning, as well as lining up all the stars. Then close the L image.
- Go to Color Combination (similar names in all the programs), specify the images to combine if necessary, and enter the scaling factors.
  - With A0V stars as reference and images with the same exposure time: R:G:B = 1:1.2:1.5.
  - Or with G2V stars: R:G:B = 1:1.3:2.

(see Lesson 3, Example 1)

Color composition (continued)

Then click Apply (or whatever) and you will get an RGB composite.

- For scientific images this, and the component R G and B, may be your desired results. But if it looks pretty you should still proceed to LRGB composition.
- If you are going to try to make it prettier, don't tweak the display parameters just yet. You'll need to make it a bit duller, and prettiness can be retrieved in Photoshop.

M101, RGB, full frame, DDP stretch

On to Photoshop

For extra pretty images we need scaled versions of L and RGB in tagged image file format (TIFF).

- Unlike FITS, TIFF has a permanent scaling involved, so the image display needs to be slightly modified before creating the TIFFs.
- Namely: adjust the display for each, so that the brightest spot you're interested in has pixel values of 200 DN or less – so that Photoshop can adjust it up as well as down.
  - Best to do by decreasing gamma in a DDP or arcsinh stretch.

Next lesson (the last): how to make LRGB composites, and the most useful features of Photoshop for tweaking and retouching pretty astronomical pictures.

Then we consider Field Trips.