Report

Cato Sandford*

September 27, 2012

Abstract

A sketch of my project with DWH and PJM: image deconvolution and colour-composition.

Contents

1 Introduction **3**
- 1.1 The Point-Spread Function (PSF) . . . 3
- 1.2 Bandpasses and Colour Images . . . 3

2 Deconvolution **4**
- 2.1 Source Identification . . . 4
- 2.2 Determining the PSF of an Image . . . 5
- 2.3 Generating an Idealised PSF . . . 5
- 2.4 Mapping to Ideal PSF . . . 6
  - 2.4.1 Interlude: Honesty Note . . . 6
- 2.5 Full Image Deconvolution . . . 7
- 2.6 Noise Properties . . . 8
- 2.7 Image Manipulations . . . 12

3 Colour Combination **12**

4 To do **13**

Appendix **15**

A Magain, Courbin & Sohy 1998 – "Deconvolution with Correct Sampling" **15**

B Lupton et al. 2004 – "Preparing Red-Green-Blue Images from CCD Data" **16**

*Department of Physics, New York University, USA; cato@nyu.edu

References **18**

1 Introduction

> I found I could say things with color and shapes that I couldn't say any other way – things I had no words for.
>
> – Georgia O'Keeffe

Astronomers continue to astonish the world with images of a Universe far richer and more magnificent than any ordinary eye could perceive or any mind conceive. Indeed, astronomy has always depended heavily on images**MORE**

- Sophistication of modern analysis
-
- The big picture: lens zoo, image presentation and combination.

PHIL: 'Basically we want to make PS1, DES, HSC and LSST data as informative as possible for astronomers who want to *look at images*. My own interest is in finding gravitational lenses, but basically anyone who wants to see what they have found in the object catalog database should be interested in this work.'

1.1 The Point-Spread Function (PSF)

1.2 Bandpasses and Colour Images

2 Deconvolution

As mentioned in the introduction, it is frequently possible and indeed desirable to reduce or control the bluriness of a telescope image. In this section, we describe how one might do this – the procedure is as follows:

1. Identify the astronomical sources in a FITS image.
2. Find the point sources, and from these estimate the PSF of the image.
3. Generate an idealised, symmetrical PSF for the image, using the dimensions and flux properties of the image PSF of point 2.
4. Create an object which maps the image PSF to the ideal PSF. This object is called the "kernel" and the mapping procedure is convolution.
5. Apply this procedure to the original image; this should correlate all the pixels in such a way as to rid the image of asymmetrical blurriness and replace it with a blurriness of known properties.

The reader may be confused at this point as to why, if we can correctly calculate the PSF of an image, we don't simply do away with the blurriness altogether – i.e. find a kernel which maps the PSF to a point. This would be a "hard deconvolution", a procedure which is compellingly discouraged by the work of Magain et al. (1998) (see appendix A for more discussion of this paper). Following this work, we may endeavour to obtain better knowledge of the sky by reducing the PSF, but we must avoid inadvertently violating the sampling theorem, which would certainly happen if we attempted a hard deconvolution.

Once we have made the PSF as small as possible within this restriction, we can enforce that it be symmetrical uniform throughout the whole image: this is a "soft deconvolution". The result of this will be that all point sources have the same pre-determined shape, and extended sources will be superpositions of this shape.

Mathematically, we can think of the measured PSF t as being composed of an idealised/reference PSF r convolved with a messy 2D function K:

$$t(\vec{x}) = r(\vec{x}) * K(\vec{x}). \tag{1}$$

In the following sections, we discuss how we determine the true PSF t, construct the reference PSF r, and find the convolution kernel K. Crucially for homogenising the image, we must also find the kernel k which governs the inverse transformation, i.e.

$$t(\vec{x}) * k(\vec{x}) = r(\vec{x}). \tag{2}$$

2.1 Source Identification

- Use SExtractor to identify the sources in an image
- How does it work?

- What else does it measure / output?

- SExtractor reads in a FITS image and catalogues all the astronomical sources it contains.
- My code automates this by generating catalogues of an entire directory of images, using some special settings.
- These default settings are (mostly) in the files `prepsfex.sex` (governs how the program runs) and `prepsex.param` (governs what information about the sources is recorded in the output file).
- The program outputs a file with extension `.cat` (by default, if the image is called `image.fits`, the output will be `image.cat`). This contains information like the position, elongation and flux of the source.

2.2 Determining the PSF of an Image

- Use PSFEx to compute PSF (arbitrary shape, noisy)
- How does it work? Use that graph in the docs.
- What else does it measure / output?

- PSFEx uses the `.cat` file from SExtractor, and estimates the PSF in the image.
- There are a considerable number of options for how this estimation is done and what form the outputs take. After flirting with some of the more sophisticated options, it turns out that the key thing for my purpose is an integrated image of the average PSF for the image. This is saved as FITS.
- I have automated the PSFEx step to process all the image catalogues in a directory with some default settings imposed (these are kept in `default.psfex`).

2.3 Generating an Idealised PSF

- Find moments of PSFEx PSF (2D)
- Find total flux
- Size of PSF image
- Plug into a 2D Gaussian
- Generate FITS files

Some comments on generating

- We wish to deconvolve the image such that it has a constant, symmetrical PSF of finite width.

- My code reads in the FITS file containing the PSF; it calculates the total flux, and the width of the PSF in the y and x directions. Then it generates the image of a 2D Gaussian which shares these properties. This is the "ideal" PSF.

2.4 Mapping to Ideal PSF

Find the convolution kernels which map between these two versions of the PSF.

- In order to deconvolve the entire image to this "ideal" PSF, we must find an object which, when convolved with the "true" PSF gives the "ideal" one. This object is the convolution kernel, which is we retrieve by casting the deconvolution procedure as a linear algebra problem.

Say we have two similar images: $\mathcal{A}$, which is a picture taken by a real telescope, and $\mathcal{B}$ which is the same image but deconvolved to a constant, ideal PSF $r(\vec{x})$. We need to deconvolve the image $\mathcal{A}$ by some kernel k in order to leave $\mathcal{B}$ with a clean PSF of $r(\vec{x})$. Since convolution is a linear operation, we can cast our search for the convolution kernel in terms of a linear algebra problem. Specifically, we want to solve (for $\vec{k}$) an equation of the form

$$\vec{b} = A\vec{k}, \tag{3}$$

where A and $\vec{b}$ encode the original and deconvolved image, and $\vec{k}$ represents the convolution kernel (also an image). When written in this form, our problem of finding $\vec{k}$ becomes a traditional vector-equation–solve.

The vectors $\vec{b}$ and $\vec{k}$ are easy to construct – just flatten the respective images. If $\mathcal{B}$ is an image of $n \times m$ pixels, then $\vec{b}$ will have nm elements. A is less strightforward, and we must think carefully about the convolution process to understand what's going on here.

MORE

A is then a matrix with size(b) rows and size(k) columns. The vector problem in equation 3 is overdetermined, so we must use some optimisation procedure (typically least-squares minimisation) to find $\vec{k}$.

2.4.1 Interlude: Honesty Note

When we convolve two images, we produce a new image pixel by taking a weighted sum of the surrounding pixels. For instance, say we convolve a large image $\mathcal{A}$ with a 3×3-pixel image to get $\mathcal{B}$. The first (i.e. top-left) pixel of $\mathcal{B}$ that we can properly determine is not the same as the first pixel of $\mathcal{A}$, because **MORE**.

Thus, the honest thing to do is to throw away some information by making image $\mathcal{B}$ smaller than $\mathcal{A}$ – in this example with a 3×3 image, we shave off the four outer edges, so if $\mathcal{A}$ is $N \times M$, $\mathcal{B}$ is $(N-2) \times (M-2)$. A bigger convolving image would require more pixels to be lost from the result.

This contrasts with the traditional solution to the problem, which is to "pad" the original image with zeroes (see figure **??**) such that the final image is the same size as the (pre-padded) original was. This seems to introduce spurious information – how can we possibly know that there are zeros at the border of an image? Often this is patently not the case, even for astronomical images which can be mostly black. With this in mind, it may seem preferable to adopt the "lossy" procedure outlined in the last paragraph.

DWH has argued semi-convincingly that we needn't make such a sacrifice in practice, because there is enough information in the border pixels to mitigate grave data-fabrication. Hmm...

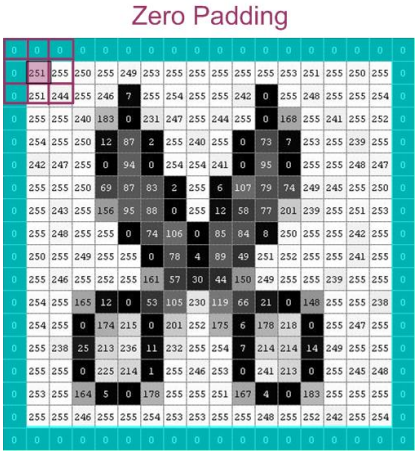


Figure 1: An image padded with zeroes. Convolution with the 3×3 stencil will now preserve image size; but at what cost?

2.5 Full Image Deconvolution

In section 2.4 we found the kernel k which would transform a messy PSF image into a nice symmetrical one with known properties. This same kernel can be used to transform an entire image with the messy PSF, $\mathcal{C}$, into the same image[1] with a nice PSF, $\mathcal{D}$. To do this, we must perform the operation

$$\mathcal{D} = k * \mathcal{C}, \tag{4}$$

or, in the language of linear algebra,

$$\vec{d} = \underline{\underline{K}}\vec{c}. \tag{5}$$

The image vectors $\vec{d}$ and $\vec{c}$ have the same form as discussed earlier – they are simply flattened versions of the pixel-arrays. The kernel matrix $\underline{\underline{K}}$ is more tricky; but after some refelxion, it transpires that $\underline{\underline{K}}$ is a sparse, upper-diagonal matrix where every row is identical, but shifted right by one element with respect to the row above. (Note that in practice it is unweildy and prohibitively expensive to store all the zero-entries of the $\underline{\underline{K}}$ matrix – we have to be a little bit clever about how to store and manipulate the relevant information.)

[1] Modulo concerns raised in section 2.4.1.

Once we have constructed an appropriate kernel matrix and multiplied it by the (messy-PSF) image vector, we are done.

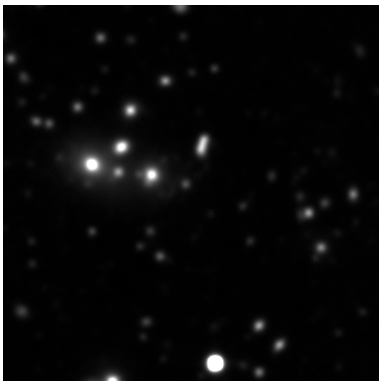
Alternatively, we can simply use a pre-existing module which performs the convolution for us: `scipy.signal.fftconvolve` does the job nicely. We can even tell this module to reduce the size of the convolved image (`mode="valid"`).

However, it would be naive to simply place our trust in SciPy – their convolution method may be subtly different from the one we wish to use for scientific inference. Thus, we cannot escape having to implement the concolution procedure of equation 5. Figure 2 shows a comparison, with residuals. Leaving aside the image artefacts from using a small convolution kernel (**MORE** FIX!), we can see that the two methods do not agree completely. This is worrying, but according to DWH the problem is almost certainly with my implementation rather than SciPy's, and I am inclined to agree (it might be a simple indexing error). Combined with the fact that SciPy is around 15 times faster, it makes little sense to labour on with my code; so we adopt the SciPy convolution module.

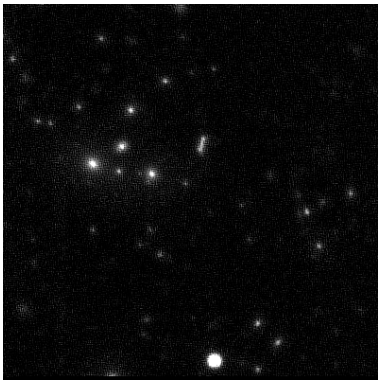
2.6 Noise Properties

Here I've applied a few different stretches to the original image and to the deconvolved image. The left-hand side images are all stretched originals and the right-hand side are all stretched deconvolutions. The stertch gets more sever as you go down.

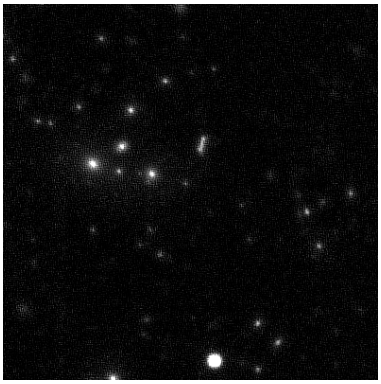
Specific notes:
* I used a 9x9 kernel for these images.
* To change the strength of the stretch, I simply divided the upper bound (you called it "vmax") by 2 and 10.



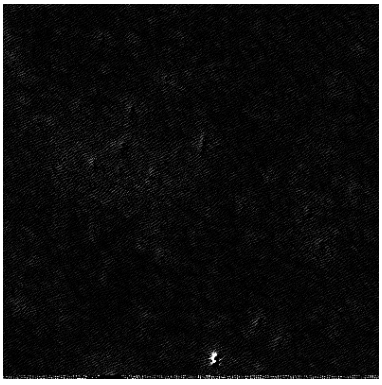
(a) Original image



(b) Deconvolved using my procedure and a 3x3 kernel.

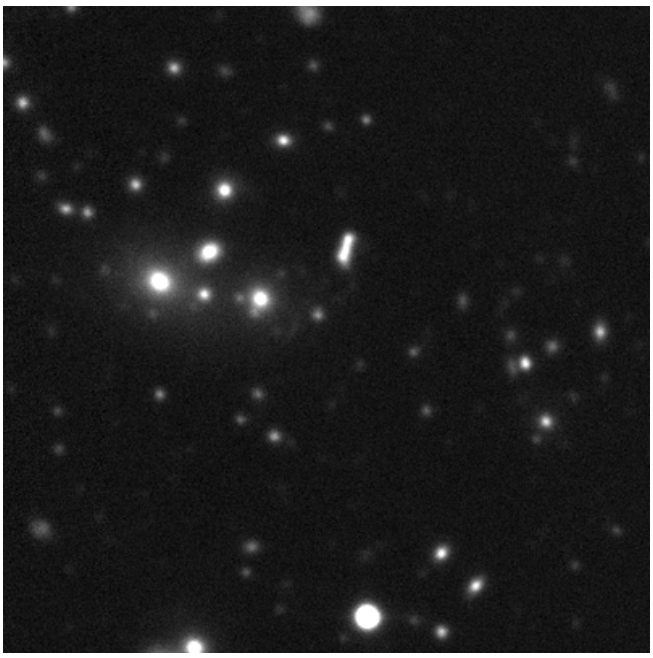


(c) Deconvolved using SciPy's procedure and a 3x3 kernel.

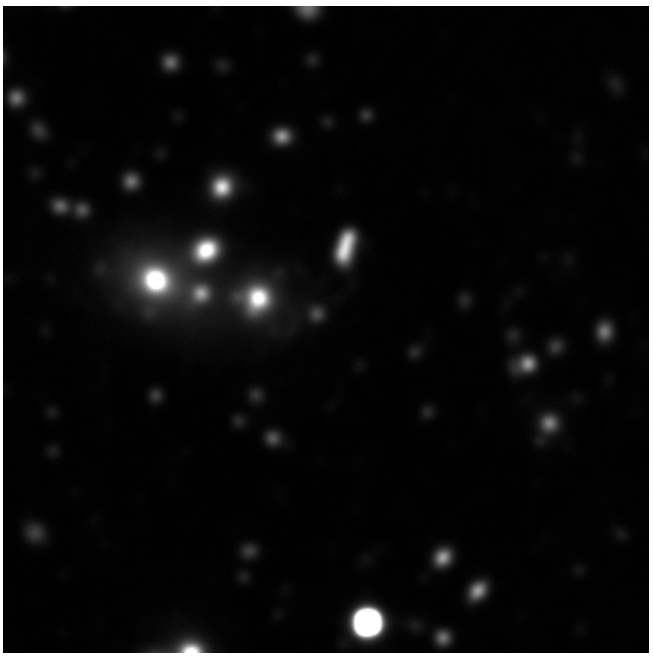


(d) Residuals.

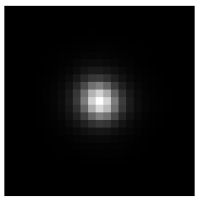
Figure 2: Deconvolution procedure – a comparison of my home-made convolution algorithm and SciPy's optimised one. SciPy takes about 1/15 of the time. Note that a 3x3 kernel was used for this test, and there are obvious artefacts introduced around bright points of the image.



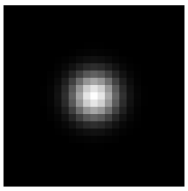
CFHTLS_03_g_sci_rescaled.png



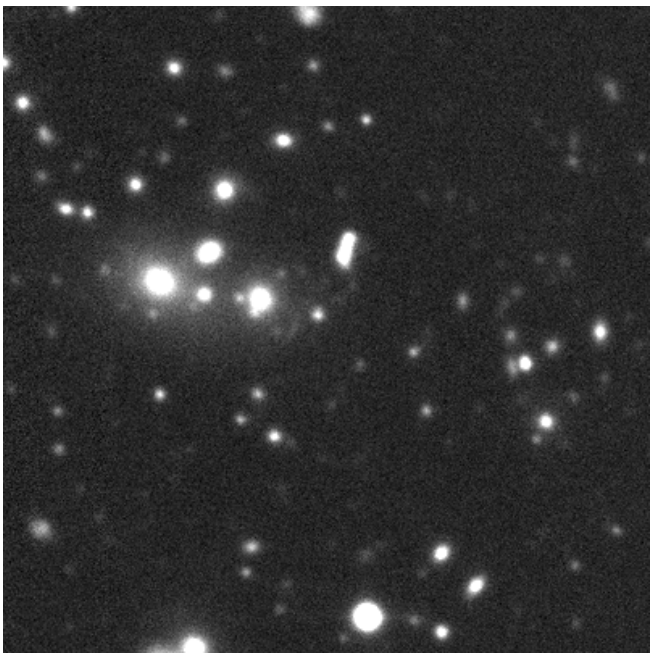
gsc_CFHTLS_03_g_sci_rescaled.png



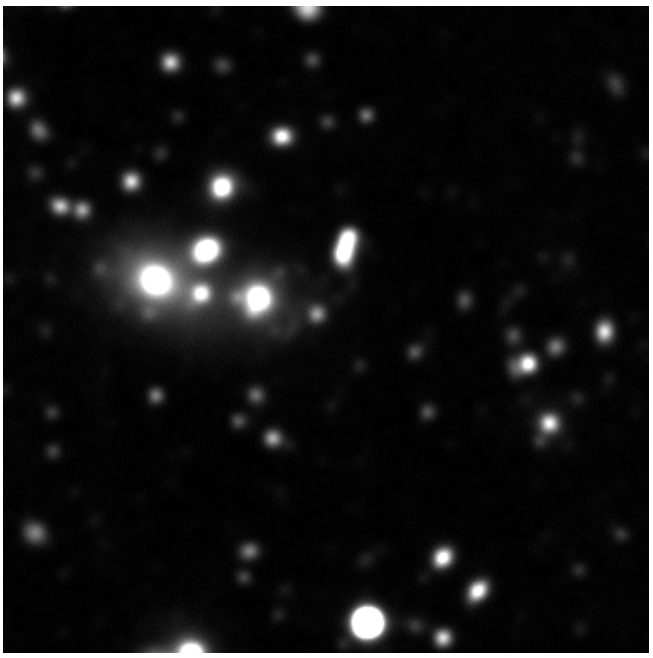
snap_n1_d0_CFHTLS_03_g_sci_z8.png



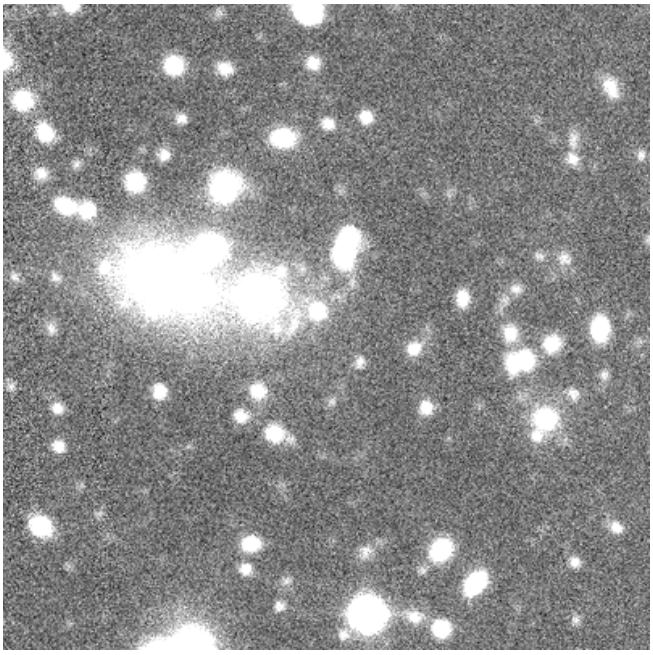
snap_n1_d0_CFHTLS_03_g_sci_Gauss_z8.png



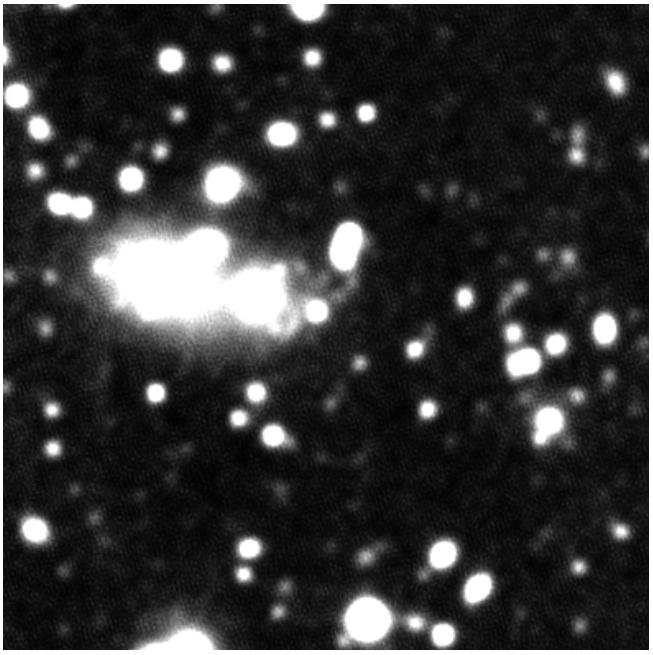
CFHTLS_03_g_sci_rescaledl.png



gsc_CFHTLS_03_g_sci_rescaledl.png



CFHTLS_03_g_sci_rescaledll.png



gsc_CFHTLS_03_g_sci_rescaledll.png

2.7 Image Manipulations

- Making them viewable by humans – stretch
- .psf file conversion
- FITS conversion

3 Colour Combination

Telescope data is often given in "bands" – we do not record the total flux coming from a point in the sky, but the flux in a certain wavelength range. This information allows us to investigate the properties of astronomical objects, such as temperature and chemical composition. However to make a readily-interprable image which contains maximum information in one hit, we need to combine the bands to make a colour-composite image.

Many methods had been developed to do this in accordance with researchers' individual aesthetic preference. But the full richness of possibility was still unexploited, until Lupton et al. (2004) demonstrated just how much detail one could extract from data when colour was given its deserved treatment (see appendix B for more detail). Using their method, we can combine images from different bands in a way which enhances hidden or faint features without allowing bright objects to dominate. It has the further advantage that the brightness of an object in the image is decoupled from its colour. The method is implemented in PJM's HumVI (formerly ColorJPG).

A further improvement to the method was implemented by Wherry (**MORE** describe difference). We translated Wherry's improvements from IDL to Python, and integrated this module with the HumVI code. Some additional improvements are also made, such as increased versatility with rebinning.

A summary of this part of the project (with pseudocode) follows.

1. Translated (and improved?) the Wherry algorithm from IDL to Python (`wherry.py`). This is what it does:
   (a) `RGB = [readin(R_data),readin(G_data),readin(B_data)] ## *_data can be filename, array of data, or a channel instance`
   (b) `rescale(RGB, scalefactors) ## multiply each band by a given number`
   (c) `rebin(RGB, xrebinfactor,yrebinfactor) ## re-sample images`
   (d) `kill_noise(RGB, cutoff) ## sets all pixels below a threshold to 0`
   (e) `arsinh_stretch(RGB, nonlinearity)`
      - `rad = R_array+G_array+B_array ## collapse images onto each other`
      - `if rad[i,j]==0:  rad[i,j]=1`
      - `factor = arsinh(nonlin*rad)/(nonlin*rad)`

- `(R_array,G_array,B_array) *= factor`

(f) `if staturatetowhite==False:  box_scale(RGB)`

- `maxpixel[i,j] = max(R[i,j],G[i,j],B[i,j]) ## i.e.  find the maximum pixel value of the three arrays`
- `if maxpixel[i,j] < 1:  maxpixel[i,j]=1`
- `(R_array,G_array,B_array) /= maxpixel`
- `(Also translates origin of image if required)`

(g) `overlay/underlay ## not entirely sure what these are for`

(h) `scipy.misc.imsave(RGB)`

- Note: when treating `wherry.py` as a standalone code for making composite images, the user can choose which bands to use for R, G and B.
- Also, in the IDL version of the code, there is a function devoted to transforming the image data into bytes. When I was translating to Python, I decided that this was an IDL-specific step, and it was unnecessary to implement it as Python does it automatically / there is a way around. Perhaps I didn't fully understand the IDL.

2. Integrated with PJM's HumVI code, so that choice of Lupton/Wherry procedure is an option.
   - Lupton is default. Wherry requires command-line keyword `--wherry` or simply `-w`.
   - Again, the user can choose which bands to use for R, G and B – it just depends on the order of the three filenames.
     - After some initial misunderstanding, I now use R=i, G=r, B=g.
   - Of course, this bands→colour map is unchanged when when `--wherry` is specified.

4 To do

Document

- Use "target" PSF rather than "ideal". "Observed" rather than "true".

Plots

- Invert plots to save ink!

Colour

- Make some side-by-side composites showing Wherry and Lupton
- Wherry shold be default
- Rename fits2colorjpg as compose.py
- Scales should be input ourselves

Deconvolution

- Use larger kernels
- Treat all three bands of a certain image with the same kernel – should be the most conservative.
- User-specified PSF width. Keep total flux.
- Homogenise PSF across different fields. **TODO: What is the best way to do this?** Use largest PSF $\rightarrow$ no danger of violating sampling theorem; but this may be way too blurry to be useful. Pickle.
- PSFEx failure modes.
- Output FITS

Combining C&D

- Make pipeline for deconvolution to common $r(x)$ then combination. (The deconvolution and colour composition procedures are not integrated into a pipeline. There are some small challenges in integrating them, but it shouldn't be too difficult. The main danger is that the code becomes too cumbersome and dependent on my directory structure.)

Maths

- PHIL: 'I would like to see a derivation of this procedure, which involves a matrix acting on a noisy vector. Do you start by writing down the principled probabilistic inference of a model image given noisy data, and end up showing that this boils down (under certain assumptions) to the matrix operations you perform? Any improvements we make to your code will probably be of the form "assign a different prior to the pixel values of the kernel/final image", so it'd be good to see how that propagates through into new matrix operations.'

APPENDIX

A Magain, Courbin & Sohy 1998 – "Deconvolution with Correct Sampling"

Key Ideas – We shouldn't pretend to derive infinite resolution images from discrete data. A more honest approach can mitigate the appearance of artefacts.

– Bear in mind that correlations in astrnomical images are local. Global treatments and techniques are inappropriate.

Background – Ground-based telescopes suffer from aperture diffraction and from atmospheric inhomogeneities which distort light. One (post hoc) way of correcting for this is to infer the point spread function from a puative point source in your image; if we consider the data to represent "reality" or the "scene" convolved with this PSF, then we can in principle deconvolve the data from the PSF to retrieve the scene.

– There will be many scenes compatible with the (uncertain) data, so we must then pose the problem as an optimisation problem: we wish to find the scene, compatible with the data, that minimises some cost function to be devised. A typical procedure is to minimise the chi-squared function (between data and model).

– Also want solution to be smooth, so introduce a Lagrange function which enforces this. A common procedure is to maximise the entropy of the image (using the flux distribution as the information). This has the benefit of requiring positive flux values.

– So far we ignore noise in the image.

Problems – Two problems emerge with this way of doing things: 1) often find image artefacts (from improper sampling, as we shall see); 2) it doesn't preserve the global intensity scale.

– In practice, telescope cameras are constructed so that their data just satisfy the sampling theorem – the pixel-spacing is 2x the maximum frequency expected from objects. Upon deconvolution, where the fuzziness is taken out, the sampling theorem will be violated. Theoretically, deconvolution can introduce point-sources/Dirac-deltas (i.e. stars), so an infinitely small sampling interval would have to be used.

– Deconvolution therefore leads to artefacts when there is a sharp discontinuity in the scene – e.g. a star on a black background shows ringing. (Can think of this as a window in frequency space (i.e. a cutoff at some maximum frequency) leading to a sinc function in position- space: the result of deconvolving a point source will be delta*sinc.)

– In traditional methods, riniging is mitigated by the positivity constraint, which damps down the lobes of oscillations. But this depends crucially on the zero-level, and accurate subtraction of sky noise is necessary for the methods to wrok well.

– Image artefacts steal flux and bias photometry. Also, maximising entropy makes the image as smooth (uniform) as possible, which tends to spread out point sources; peak intensity is thus undersetimated.

Proposal – Do not do a full deconvolution: do a "light" deconvolution where point sources are given as extended objects of know size and flux distribution. These objects are chosen such that they satisfy the sampling theorems. In other words, reconstruct the image you would get if you had a better instrument (rather than a perfect instrument).

– So now the image has a constant PSF, which M++ call r(x). This introduces a length scale over which the image must be smooth (?). This applies to point sources (which have shape r(x)) and extended sources. From the solution space of lightly-deconvolved scenes, we should choose the one which gives maximum smoothness on this local scale.

– Specifically, for each pixel we take the difference of the "background" (everything which isn't delta) from the "reconstrcuted background" (the fixed PSF convolved with the scene); then sum over pixels and minimise (equation 7). This procedure discards high-frequecy information, but is consistent with the adopted sampling and the frequencies of r(x).

– Artefacts not stealing flux AND no smoothing of point sources -¿ photometry possible.

– Requires no positivity constraint.

Usage – Using simulated and real astronomical images, with finite resolution and noise, the new procedure is compared with other standard procedures and does (stupidly) well. They are able to recover fluxes and positions to high accuracy, and they avoid exacerbating noise / artefacts in the image.

– Image combination is also demonstrated – deconvolution of many images to the same PSF before combining them yields high-resolution final image.

Further Work – Devise a more robust optimisation that finds minimum even in populated images.

B Lupton et al. 2004 – "Preparing Red-Green-Blue Images from CCD Data"

- 'sheer drama and beauty of the night sky'

KEY IDEA ——

- There is a lot of information in the colouration of an image. This often helps us distinguish features / phenomena and classify astronomical objects.

- Hitherto, focus has been on intensity differences.

BACKGROUND ———-

- We apply stretches to images in order to coax faint objects into observability. But we must strike a balance between this objective and the saturation of bright parts.

- Stretch is a re-scale, bringing all objects to within a brightness cutoff range. Re-scale can be linear, ln, sqrt, depending on preference and the diversity of images.

- Tuning parameters is not always straightforward. Any object above maximum brightness ends up bleached and obese.

- Furthermore, there is degeneracy between brightness and colour in traditional stretching procedures.

SOLUTION ———

- Using a different stretching procedure, (equation 2), can discard uninteresting intensity information in favour of colour information. This works by comparing the individual colour-intensities to the total intensity (i.e. compare the colours amongst themselves), and comparing the total instensity to the two cutoff intensities which define the brightness scale.

- NB the colours are unique – no degeneracy with intensity. So we can draw unambigouous conclusions from looking at colour differences.

- arsinh stretch magnifies faint objects (linear regime) and avoids bleaching bright objects (logarithmic regime). (But this could be achieved with other functions too).

EXAMPLES ———

- Some examples are given where the standard technique loses an embarrassing amount of detail compared with the new idea. By eye, we clearly distinguish differences between objects which are otherwise just rendered as white blobs.

References

R. Lupton, M. R. Blanton, G. Fekete, D. W. Hogg, W. O'Mullane, A. Szalay, and N. Wherry. Preparing Red-Green-Blue Images from CCD Data. PASP, 116:133–137, February 2004. doi: 10.1086/382245. 3

P. Magain, F. Courbin, and S. Sohy. Deconvolution with Correct Sampling. ApJ, 494:472, February 1998. doi: 10.1086/305187. 2