Subject: Re: New Image Processing Routines
Posted by Marshall Perrin on Fri, 21 Apr 2006 17:24:01 GMT
View Forum Message <> Reply to Message

David Fanning <davidf@dfanning.com> wrote:
> I've since moved on to more pressing things, but maybe
> I'll have to revisit this. Let me know if a few hours of
> sleep leads you to some more ideas. :-)

Actually, it was while brushing my teeth that I had the epiphany.
The relelvant line of code is the scaling, of course:

```
 output =  Scale_Vector(ASinhScl_ASinh(alpha*beta*Temporary(output))/be ta, $
      minOut, maxOut, /NAN, Double=1)
```

The only way in which alpha affects things is as a multiplicative constant
times the input (also named 'output' above just to be tricky :-) We know that
it's the ratio of beta to the input that sets the amount of nonlinearity
in the data. Quoting from Lupton et al 1999, right after eq. 4:

 For x -> \infinity, \mu approaches m for any choice of \beta. On
 the other hand, when |x| <~ b, \mu is linear in x.

So it's the ratio of beta to x that matters, and since alpha just has the
effect of scaling x, it doesn't actually add an additional degree of freedom.
I'm fully convinced now that alpha doesn't give you anything new.

Next problem: the beta in your code (and in all the other IDL implementations
of asinh scaling, as far as I can tell!) is actually the *inverse* of the
parameter b in Lupton et al. Quoting now from Lupton et al. 2004, shortly
after Eq. 2:

 We take F = arcsinh(x/Beta), where the softening parameter Beta is
 chosen to bring out the desired details.

So they say they are dividing the input by Beta, while you're multiplying it.
Actually, they *say* they are dividing the input by Beta, but if you download
their code from http://cosmo.nyu.edu/hogg/visualization/ and look at
nw_asinh_fit.pro, the relevant line of code is

```
    val = asinh(radius*nonlinearity)/nonlinearity
```

The same approach is used in Dave Schlegel's djs_rgb_make available in his
IDLUTILS library. So it seems that they've defined nonlinearity = 1./beta .
Right? And thus your Beta is the inverse of theirs.

Does this actually matter? No, not really. Either way works fine. It's

just a slightly confusing state of affairs for anyone trying to match
up IDL code to published algorithms. I *think* the simplest solution
is to rename in your code "beta" to "nonlinearity" and put in a note
that nonlinearity = 1./b. But hey, who am I to tell you what to name
your variables? :-)

Another thing possibly worth adding to the documentation is that if
you have some estimate of the noise in your image, then setting
nonlinearity = 1./noiselevel (or equivalently, beta=noiselevel) does
a decent job at showing the full dynamic range of the image from the
peak down to the read noise. Actually I've found that some small
multiple of the noiselevel works best, say 5. That makes the asinh scaled
image linear in the range of 0 - 5 sigma, then logarithmic for
everything brighter.

Oh, this is so slick. I've *got* to start using this for all my plots.

 - Marshall