Subject: Re: image contrast, bias a la DS9
Posted by JDS on Tue, 27 Apr 2010 18:27:13 GMT
View Forum Message <> Reply to Message

On Apr 22, 4:15 pm, Craig Markwardt <craig.markwa...@gmail.com> wrote:
> On Apr 22, 11:08 am, Gray <grayliketheco...@gmail.com> wrote:
>
>
>
>
>
>> On Apr 22, 10:35 am, David Fanning <n...@dfanning.com> wrote:
>
>>> Gray writes:
>>>> The image display/processing program DS9 has a feature where you can
>>>> interactively adjust the colormap of the image by dragging the mouse;
>>>> this changes the colormap's "contrast" (between 0 and 10) and
>>>> "bias" (between 0 and 1). I'd like to be able to reproduce that kind
>>>> of adjustment in IDL (not interactively - I want to be able to apply
>>>> the same adjustments to a number of images), but I'm not sure exactly
>>>> what it is they're doing. Can anyone give me guidance?
>
>>> I don't have any idea what they are doing, but this sounds
>>> suspiciously similar to "windowing and leveling" an image.
>>> That is, you select a range of image values in the image
>>> (the window) and you center that window at some value
>>> in the image (the level). In your case, contrast is
>>> the window and bias is the level, I would be willing to
>>> bet.
>
>>> http://www.dfanning.com/ip_tips/contrast.html
>
>>> Cheers,
>
>>> David
>
>>> --
>>> David Fanning, Ph.D.
>>> Fanning Software Consulting, Inc.
>>> Coyote's Guide to IDL Programming:http://www.dfanning.com/
>>> Sepore ma de ni thui. ("Perhaps thou speakest truth.")
>
>> That was my first thought, but then I realized that the contrast/bias
>> adjustment is different than setting a min/max for scaling the image,
>> which they do elsewhere. For example, I can set a min/max value to
>> -100/+100, and then a contrast of 1.3 and a bias of 0.5, and they all
>> do different things.
>

>> I did find this, however:
>
>> "Contrast refers to the rate of change of color with color level. At
>> low contrast, color changes gradually over many intensity levels,
>> while at high contrast it can change rapidly within a few levels.
>> Contrast adjustment works whether the image is in black and white, or
>> in color.
>
>> Bias refers to any offset added to the color levels before the color
>> map is applied. In other words, it determines where the color changes
>> start. Changing the bias corresponds to translating the color map with
>> respect to the intensity levels without changing the overall "look" of
>> the map. At low bias, low intensities (i.e., low pixel values) will
>> have non-zero color differences, while at high bias only high pixel
>> values will have non-zero differences."
>
>> I understand what all that means, I think, but I'm stuck on how to
>> implement it in IDL.
>
> I think DS9's quick color adjustments are equivalent to re-adjusting
> the top and bottom levels.
>
> For example, if you set your top/bottom levels to (-100,+100), and
> then do some quick DS9 adjustments, then the new levels will be, for
> example (-50, 50). That would correspond to a "magnification" of the
> color scale by a factor of 2x. You could have achieved the same
> effect by setting the top/bottom levels to (-50,+50) from the start.
> BYTSCL() is your friend.

DS9's methods, and the methods of many similar tools, date back to the "bad old days" when interactively rescaling the data between a minimum and maximum value was simply not feasible for large arrays. A simple trick, however, could be used to provide a form of interactivity, and almost for free. At the time, most video cards utilized writeable 8bit color tables. To change the contrast/bias, rather than rescale the (potentially large) data array, you simply rewrote the color table directly in the video card, which instantly updated the appearance of the image (and possibly caused all the background applications to look like bad Jackson Pollock paintings). A typical approach is, for example, to interactively set some bottom fraction of the color table black, and some top fraction white, and to map grayscale in between according to some contrast (e.g. power law).

Today, video cards do not utilize color tables in this way, so to produce this ancient effect, the data must be rescaled. This doesn't pose a CPU problem with most modern systems and even large arrays. On the other hand, the major limitation is still as it was then: by

rescaling the colorbar rather than the data themselves, you have reduced the number of independent colors used in displaying the image. In the old days, that was a reasonable trade-off, since it gave interactivity where otherwise none would exist. Today, it is not a good trade-off. If you are going to rescale, you might as well rescale in data space, and not color space. It will cost you the same compute cycles, but the former will result in a much better appearing image.

JD
