Subject: Re: color value interpolation from colorbar
Posted by Jeremy Bailin on Fri, 05 Dec 2008 01:00:50 GMT
View Forum Message <> Reply to Message

On Dec 4, 4:35 pm, Paolo <pgri...@gmail.com> wrote:
> So your first step is to "digitize" the color bar:
>
> Create 3 arrays of r,g,b with the colors as a function of pixel size
> and an array x of the pixel index number.
>
> Then, show us a plot of r(x), g(x), b(x)
>
> Ciao,
> Paolo
>
> j.coe...@gmail.com wrote:
>> Thanks. The red-to-yellow colorbar is already on the image -- I don't
>> create it. The map of gradient values to the 3D colorspace would have
>> to be determined from this colorbar, which also has errors in it
>> (garbage in, as Dr. Fanning says.) I suppose this would be an
>> irregular grid of gradient values within the 3D colorspace.
>
>> Basically, there's a color-coded image with a colorbar *on* the
>> image. The colors, even those in the bar, often have errors because
>> they were digitized from analog tape recordings. Still, it is
>> possible to eyeball the images to determine color levels. I want the
>> computer to do something more quantitative than the eyeball analysis.
>
>

How many points do you have on the colour bar? If it's sampled finely enough, it might be okay to assume a straight line through RGB space between them. Then you could calculate the minimum distance of any desired colour from each of those interpolated segments in RGB space to find the closest segment, and then use its component in the direction of the segment.

That's only going to work if the input isn't too garbagy - you'll need points on the colour bar that sample the gradiant quite finely, and the colours on the image can't be too distant in RGB space from the locus that the colour bar defines. But if those conditions are met, this might work.

I'm imagining something like this. You have cbar_r, cbar_g and cbar_b, each of which contain nbar points and represent data values cbar_data. To find the best data value for a pixel with colour image_r, image_g and image_b (untested and unwieldy - can easily be vectorized to work on the image at once, but I'm too hungry):

```
; find the closest colourbar point
cbardist2 = (cbar_r-image_r)^2 + (cbar_g-image_g)^2 + (cbar_b-image_b)
^2
cbarmin = min(cbardist2, segment1)
case segment1 of
  0: segment2=1
  nbar-1: segment2=nbar-2
  else: segment2 = (cbardist2[segment1-1] gt cbardist2[segment1+1) ?
(segment1+1) $
        : (segment1-1)
endcase

; find component along the segment
from1toi = [image_r-cbar_r[segment1],image_g-cbar_b[segment1],image_b-
cbar_b[segment1]]
from1to2 = [cbar_r[segment2]-cbar_r[segment1],cbar_g[segment2]-cbar_g
[segment1], $
   cbar_b[segment2]-cbar_b[segment1]]
projcomp = total(from1toi * from1to2) / (norm(from1to2) * norm
(from1toi))

; interpolate data value
imagedata = cbar_data[segment1]*(1.-projcomp) + cbar_data[segment2]
*projcomp
```

-Jeremy.
