Subject: Re: Creating Colour Maps
Posted by JD Smith on Wed, 28 May 2003 02:32:31 GMT
View Forum Message <> Reply to Message

On Tue, 27 May 2003 08:44:22 -0700, David Fanning wrote:

> Kate (faeriepunk@aol.com) writes:
>
>> I'm relatively new to IDL and trying to create brain maps using
>> diffusion tensor magnetic resonance imaging and display these images
>> effectively in IDL. I have a map of my image and I want to specify the
>> colours of the pixels not only based on the intensity value of the
>> pixel, but also on the direction of the principle eigenvector
>> associated with each pixel, thus the colour will be associated with
>> direction and the intensity with magnitude. I have no idea how to do
>> this, I know how to create a colour map based on the intensity value
>> but I don't know how to take into account another factor that is not
>> associated with the image but with another dataset and use this to
>> determine the RGB mix. If someone could help me that would be
>> fantastic!
>
> I seriously doubt this is what you want to do. :-)
>
> It is not a problem to do it. Make your image a 24-bit image. The red
> plane values are scaled according to intensity, the green plane values
> are scaled according to vector magnitude, and the blue place values are
> kept constant.
>
> I just don't think this is going to communicate effectively to the
> viewer the important concepts you are trying to demonstrate. I think it
> would be better to use color for intensity and overlay a vector field on
> the image to indicate vector magnitude. Or, perhaps, two images, side by
> side showing the comparison. Or three images, with the last showing the
> combined information as above.
>
> But you need someway to communicate the essential idea and I don't think
> a simple pixel color is going to work.
>

I don't know, if the eigenvector direction doesn't deviate much it
might actually work. I'm imagining a mostly grayscale brain with
small splashes of color here and there. I would do it this way:

```
device, /decomposed
sz = 512L
intensity = dist(sz) &  intensity = intensity/max(intensity)
phase = randomn(sd, sz, sz, /NORMAL)* 10./!RADEG
```

s = sin(phase) & c = cos(phase)
blue = 256.*intensity
red   = blue*(c-s)
green = blue*(c+s)

tv, [[[red]], [[green]], [[blue]]], TRUE=3

i.e. use the phase angle to rotate the intensity vector (which would normally be the grayscale triple [x,x,x] if angle=0 everywhere) about the blue axis. Areas which are nearly grayscale (i.e. low saturation, washed out colors) have small directional offsets. High saturation colors mean large directional offsets, and bright/dark is as expected, since the length of the color vector is unchanged. In reality, your eye responds differently to red, green, and blue, so for highest fidelity you'd perform scaled rotations to preserve the "luminance", often defined as:

Y=0.30 R + 0.59 G + 0.11 B

which represents the eye's approximate response to different colors. So if you'd like to preserve not the mathematical length of the color vector as a surrogate for intensity, but the typical human's visual response to intensity, this is the relation to use.

JD
