Subject: Re: Using the most colors in tvscl
Posted by Peter Mason on Mon, 05 Aug 1996 07:00:00 GMT
View Forum Message <> Reply to Message

> Since there appear to be quite a few color experts out there, I was
> wondering if anyone has addressed this problem. I'm running IDL 4.0.1
> on a Windows 95 platform. The system variable !D.N_COLORS=65536, which
> I assume means that this is the maximum number of displayable colors. I
> have some 12-bit data (0-4095) that would ideally be scaled into 12-bit
> color data--however, when using tvscl, although this is a "true-color
> device", as indicated by help,/device, only 256 colors are displayed. I
> tried making up a table via a=indgen(4096), tvlct,a,a,a, which should
> give a grey-scale intensity image from 0-4095, but it does not appear to
> affect the display (i.e. changing it to a=indgen(10) still gives the
> same picture for dist(400)).
>
> Would using the tvrd procedure to read back the data in its original
> form be possible, since supposedly all those colors can be displayed?
>
> Also, is there any way to get more than 255 colors out of plots, using
> the color keyword?

You're using a 16-bit colour mode in Windows. This functions like a 24-bit mode.
A 24-bit colour value (in RGB space) consists of 8 bits for red, 8 for green, and 8 for blue. (There are 3 planes: red, green and blue.)
Hidden from you is the fact that, for 16-bit colour, your [R,G,B] bits are the most significant [5,6,5] (or such) of a full 24-bit-mode's [8,8,8].

!D.N_COLORS shows the number of different colours your screen can display simultaneously (2^16 == 65536). For "hi-color" displays, this is not the same as the number of colour indices - which is given in !D.TABLE_SIZE, and is usually 256. (It's my understanding that it's never more than 256 in IDL.)

There is still a colour table associated with 24-bit modes, but it is used like three 8-bit-wide tables to modify R,G and B independently, rather than a means of fattening 8-bit colour indices into 24-bit colour values as in 8-bit modes.
e.g., You can change the LUT to apply a stretch to a 24-bit image without changing the image data. (Unfortunately you still have to redisplay the image in MS Windows as the LUT is managed in IDL rather than written directly to the graphics card's registers.)

So how do you get the most out of your 12-bit image?
Since each colour plane is only 8 bits wide in (24-bit mode), if you want to display your image in grey (where red==green==blue), you have only 256 different shades of grey to use. (In fact, in 16-bit mode, you'll have only

2^5==32 shades of grey, and you'd be better off using an 8-bit screen mode, where you'd get 2^6 or 2^8 shades of grey, depending on your card.)

You might consider using some artificial colour scheme.   There are many possibilities, none of them easy.   (Essentially, you have to select 4096 colours out of your 65536 in some reasonable way.)
As a first attempt, you might try something like the following (and I'm not sure how many different colours it will give you):

```
  ;Let's say your image is 614*512*12-bit.
  ;Set up your "LUT", which you must manage yourself...
  v=fltarr(4096)+1.0 &s=v &h=findgen(4096)*(360.0/4095.0)
  color_convert,h,s,v,r,g,b,hsv_rgb ;a set of rainbow colours
  ;Assemble a 24-bit image by "manually" putting your image through your LUT...
  img3=bytarr(614,512,3)
  img3(0,0,0)=r(your_img)
  img3(0,0,1)=g(your_img)
  img3(0,0,2)=b(your_img)
  ;display the image...
  tv,img3,true=3
```

Reading back an image:
If you use TVRD() without any arguments, you'll get back a single-band byte image (the intensity).   If you use the TRUE keyword (e.g., TVRD(TRUE=3)), you'll get back a 3-band (24-bit) image.
If you experiment with regular LUTs (e.g., with LOADCT, TVLCT, ...) while displaying and reading back images, you'll gain an understanding of how IDL uses LUTs in Windows hi-color modes: It puts the image through the LUT to get a temporary image, then copies this to the screen memory - i.e., it does its LUT work all in software.   So when you TVRD() back an image, you get back something which has been modified by the current LUT.   (You normally wouldn't notice this as the default LUTs are 1:1.)

Plotting with many colours:
You've probably noticed that some IDL graphics commands - like PLOT - appear to work strangely on Windows hi-color displays - they plot in red, if at all.
This is because these commands don't put the specified colour index through the LUT, they just blast it out to the screen.   As a colour index is normally 8-bit, it normally affects only the red plane.
To get proper colours, use a 24-bit colour value instead of an 8-bit colour index.
e.g.,

```
  tvlct,r,g,b,/get ;load the current LUT
  mylut=long(r)+long(g)*256L+long(b)*65536L
  plot,my_x,my_y,color=mylut(my_intended_color),...
```

Since you're using a 24-bit colour value here, you can plot in as many colours as you like.   (I've just used LUTs here to show how to get the plot to work as it does in 8-bit mode.)

I hope you can make sense of all this (I'm not sure I can :)
Peter Mason