Subject: Re: Wanted: colour table good for high contrast grayscale output
Posted by Martin Schultz on Fri, 06 Mar 1998 08:00:00 GMT
View Forum Message <> Reply to Message

Iarla Kilbane-Dawe wrote:
>
> Hi,
>
> Does anybody know of a colour table that prints well with high contrast
> between steps when output to a grayscale printer?
>
> I'm generating image from atmospheric data and would like to use a table
> that allows me to easily distinguish steps in the colour scale when
> printed on our monochrome laser printer. To make matter worse, the images
> contain a lot of detail.
>
> Does anyone have any advice? Thanks is advance.
>
> Iarla.
>
> Iarla@ozone-sec.ch.cam.ac.uk.no.spam

in my oppinion this is simply an issue of finding the optimum number of grey shadings so that you get as much "resolution" as possible but with sufficient "contrast" between the steps as you say. You probably know that you can load your b/w color table into a limited range of the actual color table

```
LOADCT,0,BOTTOM=xx,NCOLORS=nn
```

and that you can "scale" your image to ly in this region by

```
TMP = BYTSCL(image,TOP=nn)+xx
```

and then get it onto the device with David F.'s TVImage routine (which produces especially good postscript results)
So now you have to experiment with the number of colors (nn) that you want, I would think that 16 is reasonable.

And now we get to the harder part: how to fool the printer and your eye: In my experience if you use constant increase between the grey values, the plot will look too dark, so you may want to manipulate the color table and shift everything to lighter values (higher numbers). This can be done in the following way (assume you have done the LOADCT step above):

```
TVLCT,r,g,b,/get   ; get entries from color table
; we only need to manipulate one vector from the RGB triple
; because grey values are identical in r,g, and b
r = r(xx:xx+nn-1)  ; extract portion that we want to manipulate
```

```
x = (255-r)/255.   ; get coordinate from 0 to 1
y = x*x            ; apply non linear function which produces
                   ; values from 0 to 1
r = byte(255.*(1-y) < 255)     ; transform back
                   ; the <255 is meant for safety reasons if
                   ; you are playing with other functions

TVLCT,r,r,r,xx     ; store color values back in table

; now plot ...
```

Of course you could have generated the r array simply by

```
r = indgen(nn)
```

but I wanted to point out this more general way of manipulating color tables.

Hope it helps,
Martin.

BTW: just yesterday I played around with a similar thing: I tried to produce a colored contour plot using intensity as an indicator of the value of a second variable (e.g. you plot relative differences in O3 concentrations from 2 model runs, and you want to de-emphasize those large deviations where you have very low concentrations). This program uses the same technique in a somewhat more sophisticated way. Please find it attached. So far it is just a demo, but I will produce a working version soon.

--

---------- -------
Dr. Martin Schultz
Department for Earth&Planetary Sciences, Harvard University
186 Pierce Hall, 29 Oxford St., Cambridge, MA-02138, USA

phone: (617)-496-8318
fax : (617)-495-4551

e-mail: mgs@io.harvard.edu
IDL-homepage: http://www-as.harvard.edu/people/staff/mgs/idl/
---------- -------


```
pro demo,ps=ps

  ; demonstration of color intensity use
```

```
; create bogus data array
a = dist(40,40)

; create shading array
b = findgen(40) # (fltarr(40)+1)


; open device (if /ps, produce postscript file)
open_device,ps=ps,filename='idl.ps',/color

if (!d.name ne 'PS') then erase
myct          ; load color table and set lowest colors for line plots

; load color table EOS-B into 16 fields beginning from 20
loadct,27,bottom=20,ncolors=16

; retrieve color vectors and blow them up to 160 elements
tvlct,rcol,gcol,bcol,/get
; delete first 20 entries (those are used for line graph colors)
rcol = rcol(20:35)   ; 16 elements remaining
gcol = gcol(20:35)
bcol = bcol(20:35)

rnew = fltarr(160)
gnew = rnew
bnew = rnew
for i=0,15 do begin
   rnew(10*i:10*i+9) = rcol(i)
   gnew(10*i:10*i+9) = gcol(i)
   bnew(10*i:10*i+9) = bcol(i)
endfor

; "fade" offset colors
; this is done by increasing the color values of r, g, and b
; proportionally
; note that we scale all 16 colors at once
dum = indgen(16)
for i=0,9 do begin
   tmp = (255.-rnew(dum*10+i))/10.
   rnew(dum*10+i) = rnew(dum*10+i) + fix(tmp*i) < 255
   tmp = (255.-gnew(dum*10+i))/10.
   gnew(dum*10+i) = gnew(dum*10+i) + fix(tmp*i) < 255
   tmp = (255.-bnew(dum*10+i))/10.
   bnew(dum*10+i) = bnew(dum*10+i) + fix(tmp*i) < 255
endfor

; put those vectors into color table beginning on entry 20
```

```
   tvlct,rnew,gnew,bnew,20


   ; now generate pseudo data that corresponds to color entries
   ; from arrays a and b
   ; first bytscl the values of a to range from 0 to 150
   ; blow array up to 400x400 for display, use interpolation
   c = bytscl(congrid(a,400,400,/interp),top=15)*10

   ; now add values of b, reduced to the range from 0 to 9
   cb = bytscl(congrid(b,400,400,/interp),top=9)

   cdisp = c + cb

   ; use the c array as an image and display it
;  tv,cdisp+20

   p = [ 0.1, 0.1, 0.9, 0.9 ]
   TVimage,cdisp+20,position=p,/keep_aspect
   ; better to use TVImage routine from D. Fanning, because that allows
   ; same output in ps file and overlaying of line graphs


   ; dummy overlay of some contour lines from original A array
   !p.position = p   ; returned from TVimage

   contour,a,level=findgen(10)*3,color=1,/noerase




   close_device

return
end
```

File Attachments

1) demo.pro, downloaded 110 times