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

oops! I just noticed that you need a few extra routines from my library
in order to run the demo I just sent. Here they are (and I hope David
won't mind that I attach his TVIMAGE for completeness):

Martin.

```
 ;------------------------------------------------------------ --
;+
; NAME:
;       MYCT
;
; PURPOSE:
;       load a color table and define the first 16 colors for
;       drawing colors (white,black,red,green,blue,yellow,magenta,
;       lightblue,lightred,lightgreen,purple,black,85%grey,67%grey,
;       50%grey,33%grey,15%grey).
;
; CATEGORY:
;       color table manipulation
;
; CALLING SEQUENCE:
;       MYCT
;
; INPUTS:
;       TABLE --> [optional] number of the color table to be used
;             default is EOS-B (number 27)
;
; KEYWORD PARAMETERS:
;
; OUTPUTS:
;
; SUBROUTINES:
;
; REQUIREMENTS:
;
; NOTES:
;       It is recommended to use the COLOR= keyword whenever possible
;       This will ensure correct colors on (hopefully) all devices.
;       In order to get 256 colors on a postcript printer use
;       DEVICE,/COLOR,BITS_PER_PIXEL=8
;
; EXAMPLE:
;
; MODIFICATION HISTORY:
```

```
;       mgs, 06 Feb 1997: VERSION 1.00
;       mgs, 03 Aug 1997: added input parameter and template
;
;-
; Copyright (C) 1997, Martin Schultz, Harvard University
; This software is provided as is without any warranty
; whatsoever. It may be freely used, copied or distributed
; for non-commercial purposes. This copyright notice must be
; kept with any copy of this software. If this software shall
; be used commercially or sold as part of a larger package,
; please contact the author to arrange payment.
; Bugs and comments should be directed to mgs@io.harvard.edu
; with subject "IDL routine myct"
 ;--------------------------------------------------------- --


pro myct,table

; loads colortable (default EOS-B) and modifies first entries:
;   color 0 becomes whitefor background
;   colors 1..10 become brilliant plot colors
;   colors 11..16 become grey shadings, beginning with black
;   the rest is unaltered

if (n_params() le 0) then table = 27

loadct, table

red  =[  255,  0,255,  0,  0,255,255,  0,255,127,127,0,62,98,172,200,232,255]
green=[  255,  0,  0,255,  0,255,  0,255,127,255,127,0,62,98,172,200,232,255]
blue =[  255,  0,  0,  0,255,  0,255,255,127,127,255,0,62,98,172,200,232,255]
; red  =[  255,  0,255,  0,  0,255,255,  0,255,127,127,0,42,85,128,170,212,255]
; green=[  255,  0,  0,255,  0,255,  0,255,127,255,127,0,42,85,128,170,212,255]
; blue =[  255,  0,  0,  0,255,  0,255,255,127,127,255,0,42,85,128,170,212,255]

TVLCT, red, green, blue
end

 ;--------------------------------------------------------- --
;+
; NAME:
;       OPEN_DEVICE
;
; PURPOSE:
;       If hard copy is to be generated, OPEN_DEVICE opens the
;       PostScript device.  Otherwise OPEN_DEVICE opens an Xwindow.
;
; CATEGORY:
```

```
;       Input/Output
;
; CALLING SEQUENCE:
;       OPEN_DEVICE, OLD_DEVICE, [,keywords]
;
; INPUTS:
;
; KEYWORD PARAMETERS:
;       PS       (int)  -> will send PostScript file to printer
;
;       COLOR    (int)  -> will enable PostScript color mode
;
;       LANDSCAPE (int)  -> will enable PostScript landscape mode
;
;       PORTRAIT  (int)  -> will enable PostScript portrait mode
;
;       FILENAME  (str)  -> The name to be given the PostScript file.
;               Default: idl.ps
;
;       WINPARAM  (int)  -> An integer vector with 3 elements:
;                       WINPARAM(0) = window number
;                       WINPARAM(1) = X dimension of window in pixels
;                       WINPARAM(2) = Y dimension of window in pixels
;
;       _EXTRA -> additional keywords that are passed to the call to
;               the DEVICE routine
;
; OUTPUTS:
;       OLD_DEVICE (str) -> stores the previous value of !D.NAME
;
; SUBROUTINES:
;
; REQUIREMENTS:
;
; NOTES:
;       If PS=0 then
;          Open Xwindow WINPARAM(0), which is WINPARAM(1) pixels wide
;          in the X-direction, and WINPARAM(2) pixels wide in the
;          Y-direction.
;
;       If PS=1 then
;         depending on /PORTRAIT or /LANDSCAPE and /COLOR
;         postscript is enabled in either portrait or landscape
;         mode as color or b/w plot
;
;       The key parameter which determines whether to open a postscript
;       file or a screen window is PS. Therefore, e.g. in a widget
;       application, you can pass a standard set of parameters for both,
```

```
;       postscript and screen, to this routine, and determine the device
;       to be chosen by a button state or checkbox which is passed into PS.
;
;
; EXAMPLE:
;       OPEN_DEVICE, WINPARAM=[0,800,800]
;          opens a screen window of size 800x800 pixels
;
;       OPEN_DEVICE, OLD_DEVICE, /LANDSCAPE, FILENAME='myplot.ps'
;          opens a postscript file named myplot.ps in b/w and landscape
;          orientation
;
;       OPEN_DEVICE, OLDDEV, PS=PS, /PORTRAIT, /COLOR, WIN=2
;          depending on the value of PS either a color postscript file
;          named idl.ps is opened or screen window number 2 in default
;          size.
;
;
; MODIFICATION HISTORY:
;       bmy  15 Aug 1997: VERSION 1.00
;       bmy, 19 Aug 1997: VERSION 1.01
;       mgs, 20 Aug 1997: VERSION 1.02
;
;-
; Copyright (C) 1997, Bob Yantosca, Harvard University
; This software is provided as is without any warranty
; whatsoever. It may be freely used, copied or distributed
; for non-commercial purposes. This copyright notice must be
; kept with any copy of this software. If this software shall
; be used commercially or sold as part of a larger package,
; please contact the author to arrange payment.
; Bugs and comments should be directed to bmy@io.harvard.edu
; with subject "IDL routine open_device"
 ;-------------------------------------------------------------- --


pro open_device, OLD_DEVICE,                          $
              PS=PS, COLOR=COLOR, FILENAME=FILENAME,   $
              LANDSCAPE=LANDSCAPE, PORTRAIT=PORTRAIT,  $
              WINPARAM=WINPARAM, _EXTRA=E

     on_error, 2   ; return to caller

     OLD_DEVICE = !D.NAME    ; retrieve current device


     if (not keyword_set(FILENAME)) then FILENAME = 'idl.ps'
     if (not keyword_set(COLOR))    then COLOR    = 0
```

```
    if (not keyword_set(PORTRAIT)) then LANDSCAPE = 1 ; default

    if (keyword_set(PS)) then begin
       set_plot, 'PS'

       if (keyword_set(LANDSCAPE)) then begin       ;Landscape mode
          device, /landscape, color=COLOR, $
             bits=8, filename=FILENAME, _EXTRA=e

       endif else begin    ; Portrait mode
          device, color=COLOR, bits=8, /portrait,   $
             /inches, xoffset=0.25, yoffset=0.25,   $
             xsize=8.0, ysize=10, filename=FILENAME, _EXTRA=e
       endelse


    endif else begin      ; no postscript desired
                     ; only action if winparam given

       if (n_elements(WINPARAM) gt 0) then begin      ;Open Xwindow
; if winparam is 3 element vector then open window of desired size
;  else open window with standard size
          if(n_elements(winparam) eq 3) then $
             window, WINPARAM(0), xsize=WINPARAM(1), ysize=WINPARAM(2) $
          else  $
             window, winparam(0)
       endif

    endelse


return
end


 ;------------------------------------------------------------ --
;+
; NAME:
;       CLOSE_DEVICE
;
; PURPOSE:
;       CLOSE_DEVICE closes the PostScript device and spawns
;       a print job to the printer specified by the user or
;       it closes a graphics window.
;
; CATEGORY:
;       Input/Output
;
```

```
; CALLING SEQUENCE:
;       CLOSE_DEVICE, OLD_DEVICE, [keywords]
;
; INPUTS:
;       OLD_DEVICE (str) -> Content of !D.NAME before call to OPEN_DEVICE
;           If omitted, 'X' will be used as a default
;
; KEYWORD PARAMETERS:
;       PRINTER    (str) -> name of the printer to send output to
;           Default is 'none', i.e. the postscript file will only be closed
;           and can thenbe manually printed e.g. using the Unix lpr command.
;
;       FILENAME   (str) -> name of the PostScript file
;           Default is 'idl.ps'. NOTE: If a FILENAME was given with
;           OPEN_DEVICE, the same name must be supplied here if the output
;           shall be sent to the printer!
;
;       WINDOW     -> window number to be closed (or -1 if current)
;
;       /TIMESTAMP  -> add a label with filename and system time to the plot
;
; OUTPUTS:
;       If postscript device is active, a *.ps file will be created and/or
;       sent to the printer.
;
; SUBROUTINES:
;
; REQUIREMENTS:
;
; NOTES:
;       The print command (lpr -P) is specific to Unix systems.  Users
;       who are running other operating systems will have to modify this.
;
;       The WINDOW keyword is only evaluated if the current device supports
;       windows [!D.FLAGS AND 256) GT 0]. If you only want to close a
;       postscript file and don't fuss around with your screen windows
;       then simply don't set this keyword.
;
;       The OLD_DEVICE parameter can be misused to set the output device
;       to anything ! Therefore, if you are working on a Unix system it's
;       probably safest to not use it and stick with the 'X' default.
;
;
; EXAMPLE:
;       CLOSE_DEVICE, OLD_DEVICE, PRINTER='pro', FILENAME='myplot.ps'
;          If current device name is PS then 'myplot.ps' will be closed
;          and spooled to printer 'pro'
;
```

```
;       CLOSE_DEVICE, OLD_DEVICE, WIN=-1
;          If current device name is PS then the postscript file will
;          be closed. If the current device is a screen device (that
;          supports windows), then the active window will be deleted.
;
;       CLOSE_DEVICE
;          restores the plotting device to 'X'
;
; MODIFICATION HISTORY:
;       bmy, 18 Aug 1997: VERSION 1.00
;       bmy, 19 Aug 1997: VERSION 1.01
;       mgs, 20 Aug 1997: VERSION 1.02
;
;-
; Copyright (C) 1997, Bob Yantosca, Harvard University
; This software is provided as is without any warranty
; whatsoever. It may be freely used, copied or distributed
; for non-commercial purposes. This copyright notice must be
; kept with any copy of this software. If this software shall
; be used commercially or sold as part of a larger package,
; please contact the author to arrange payment.
; Bugs and comments should be directed to bmy@io.harvard.edu
; with subject "IDL routine close_device"
 ;-------------------------------------------------------------- --


pro close_device, OLD_DEVICE, PRINTER=PRINTER, FILENAME=FILENAME, $
            WINDOW=WINDOW, TIMESTAMP=TIMESTAMP

     on_error, 2

     if (n_params() le 0) then OLD_DEVICE = 'X'

     if (not keyword_set(PRINTER))  then PRINTER  = 'none'
     if (not keyword_set(FILENAME)) then FILENAME = 'idl.ps'

     if (!d.name eq 'PS') then begin   ; if postscript device active

; add timestamp if desired
         if(keyword_set(TIMESTAMP)) then begin  ; draw timestamp if desired
             timelabel = systime(0)
             xyouts,0.98,0.007,filename+'  '+timelabel,color=1,   $
                  align=1,/norm,charsize=0.4
         endif



    device, /close      ; close postscript file
```

```
        if (PRINTER ne 'none') then begin   ; spawn postscript file to printer
         TRIM_PRINTER = strtrim(PRINTER,2)
           print, 'Sending output to printer ' + TRIM_PRINTER
           spawn, 'lpr -P ' + TRIM_PRINTER + ' ' + FILENAME
        endif

    endif else begin     ; no postscript device active

        ; check if device supports windows and if a window shall be closed
        if(n_elements(window) gt 0) then begin
           if(window lt 0) then window = !D.WINDOW
           if( (!D.FLAGS AND 256) GT 0 AND window ge 0) then $
              wdelete,window
        endif

    endelse

    set_plot, OLD_DEVICE

return
end



;+
; NAME:
;    TVIMAGE
;
; PURPOSE:
;    This purpose of TVIMAGE is to allow you to display an image
;    on the display or in a PostScript file in a particular position.
;    The position is specified by means of the POSITION keyword. In
;    this respect, TVIMAGE works like other IDL graphics commands.
;    Moreover, the TVIMAGE command works identically on the display
;    and in a PostScript file. You don't have to worry about how to
;    "size" the image in PostScript. The output on your display and
;    in the PostScript file will be identical. The major advantage of
;    TVIMAGE is that it can be used in a natural way with other IDL
;    graphics commands in resizeable IDL graphics windows. TVIMAGE
;    is a replacement for TV and assumes the image has been scaled
;    correctly when it is passed as an argument.
;
; AUTHOR:
;      FANNING SOFTWARE CONSULTING:
;      David Fanning, Ph.D.
;      2642 Bradbury Court
;      Fort Collins, CO 80521 USA
```

```
;       Phone: 970-221-0438
;       E-mail: davidf@dfanning.com
;       Coyote's Guide to IDL Programming: http://www.dfanning.com
;
; CATEGORY:
;    Graphics display.
;
; CALLING SEQUENCE:
;
;    TVIMAGE, image
;
; INPUTS:
;    image:    A 2D or 3D image array. It should be byte data.
;
; KEYWORD PARAMETERS:
;    _EXTRA:   This keyword picks up any TV keywords you wish to use.
;
;    KEEP_ASPECT_RATIO: Normally, the image will be resized to fit the
;          specified position in the window. If you prefer, you can
;          force the image to maintain its aspect ratio in the window
;          (although not its natural size) by setting this keyword.
;          The image width is fitted first. If, after setting the
;          image width, the image height is too big for the window,
;          then the image height is fitted into the window. The
;          appropriate values of the POSITION keyword are honored
;          during this fitting process. Once a fit is made, the
;          POSITION coordiates are re-calculated to center the image
;          in the window. You can recover these new position coordinates
;          as the output from the POSITION keyword.
;
;    MINUS_ONE: The value of this keyword is passed along to the CONGRID
;          command. It prevents CONGRID from adding an extra row and
;          column to the resulting array.
;
;    POSITION: The location of the image in the output window. This is
;          a four-element floating array of normalized coordinates of
;          the type given by !P.POSITION or the POSITION keyword to
;          other IDL graphics commands. The form is [x0, y0, x1, y1].
;          The default is [0.15, 0.15, 0.85, 0.85]. Note that this can
;          be an output parameter if the KEEP_ASPECT_RATIO keyword is
;          used.
;
; OUTPUTS:
;    None.
;
; SIDE EFFECTS:
;    Unless the KEEP_ASPECT_RATIO keyword is set, the displayed image
;    may not have the same aspect ratio as the input data set.
```

```
;
; RESTRICTIONS:
;    If the POSITION keyword and the KEEP_ASPECT_RATIO keyword are
;    used together, there is an excellent chance the POSITION
;    parameters will change. If the POSITION is passed in as a
;    variable, the new positions will be returned as an output parameter.
;
; EXAMPLE:
;    To display an image with a contour plot on top of it, type:
;
;       filename = FILEPATH(SUBDIR=['examples','data'], 'worldelv.dat')
;       image = BYTARR(360,360)
;       OPENR, lun, filename, /GET_LUN
;       READU, image
;       FREE_LUN, lun
;
;       TVIMAGE, image, POSITION=thisPosition, /KEEP_ASPECT_RATIO
;       CONTOUR, image, POSITION=thisPosition, /NOERASE, XSTYLE=1, $
;          YSTYLE=1, XRANGE=[0,360], YRANGE=[0,360], NLEVELS=10
;
; MODIFICATION HISTORY:
;     Written by:     David Fanning, 20 NOV 1996.
;     Fixed a small bug with the resizing of the image. 17 Feb 1997. DWF.
;     Removed BOTTOM and NCOLORS keywords. This reflects my growing belief
;       that this program should act more like TV and less like a "color
;       aware" application. I leave "color awareness" to the program
;       using TVIMAGE. Added 24-bit image capability. 15 April 1997. DWF.
;     Fixed a small bug that prevented this program from working in the
;        Z-buffer. 17 April 1997. DWF.
;     Fixed a subtle bug that caused me to think I was going crazy!
;        Lession learned: Be sure you know the *current* graphics
;        window! 17 April 1997. DWF.
;     Added support for the PRINTER device. 25 June 1997. DWF.
;     Extensive modifications. 27 Oct 1997. DWF
;        1) Removed PRINTER support, which didn't work as expected.
;        2) Modified Keep_Aspect_Ratio code to work with POSITION keyword.
;        3) Added check for window-able devices (!D.Flags AND 256).
;        4) Modified PostScript color handling.
;     Craig Markwart points out that Congrid adds an extra row and column
;        onto an array. When viewing small images (e.g., 20x20) this can be
;        a problem. Added a Minus_One keyword whose value can be passed
;        along to the Congrid keyword of the same name. 28 Oct 1997. DWF
;-

PRO TVIMAGE, image, KEEP_ASPECT_RATIO=keep, POSITION=position, $
  MINUS_ONE=minusOne, _EXTRA=extra

ON_ERROR, 1
```

```
   ; Check for image parameter.

np = N_PARAMS()
IF np EQ 0 THEN MESSAGE, 'You must pass an image argument.'

   ; Check image size.

s = SIZE(image)
IF s(0) LT 2 OR s(0) GT 3 THEN $
   MESSAGE, 'Argument does not appear to be an image. Returning...'

   ; 2D image.

IF s(0) EQ 2 THEN BEGIN
   imgXsize = FLOAT(s(1))
   imgYsize = FLOAT(s(2))
   true = 0
ENDIF

   ; 3D image.

IF s(0) EQ 3 THEN BEGIN
IF (s(1) NE 3L) AND (s(2) NE 3L) AND (s(3) NE 3L) THEN $
   MESSAGE, 'Argument does not appear to be a 24-bit image. Returning...'
   IF s(1) EQ 3 THEN true = 1 ; Pixel interleaved
   IF s(2) EQ 3 THEN true = 2 ; Row interleaved
   IF s(3) EQ 3 THEN true = 3 ; Band interleaved
   CASE true OF
      1: BEGIN
         imgXsize = FLOAT(s(2))
         imgYsize = FLOAT(s(3))
         END
      2: BEGIN
         imgXsize = FLOAT(s(1))
         imgYsize = FLOAT(s(3))
         END
      3: BEGIN
         imgXsize = FLOAT(s(1))
         imgYsize = FLOAT(s(2))
         END
   ENDCASE
ENDIF

   ; Check for keywords.

IF N_ELEMENTS(position) EQ 0 THEN position = [0.15, 0.15, 0.85, 0.85] $
   ELSE position = FLOAT(position)
```

```
minusOne = Keyword_Set(minusOne)

   ; Maintain aspect ratio (ratio of height to width)?

IF KEYWORD_SET(keep) THEN BEGIN

      ; Find aspect ratio of image.

   ratio = FLOAT(imgYsize) / imgXSize

      ; Find the proposed size of the image in pixels without aspect
      ; considerations.

   xpixSize = (position(2) - position(0)) * !D.X_VSize
   ypixSize = (position(3) - position(1)) * !D.Y_VSize

      ; Try to fit the image width. If you can't maintain
      ; the aspect ratio, fit the image height.

   trialX = xpixSize
   trialY = trialX * ratio
   IF trialY GT ypixSize THEN BEGIN
      trialY = ypixSize
      trialX = trialY / ratio
   ENDIF

      ; Recalculate the position of the image in the window.

   position(0) = (((xpixSize - trialX) / 2.0) / !D.X_VSize) + position(0)
   position(2) = position(0) + (trialX/FLOAT(!D.X_VSize))
   position(1) = (((ypixSize - trialY) / 2.0) / !D.Y_Size)  + position(1)
   position(3) = position(1) + (trialY/FLOAT(!D.Y_VSize))

ENDIF

   ; Calculate the image size and start locations.

xsize = (position(2) - position(0)) * !D.X_VSIZE
ysize = (position(3) - position(1)) * !D.Y_VSIZE
xstart = position(0) * !D.X_VSIZE
ystart = position(1) * !D.Y_VSIZE

   ; Display the image. Sizing different for PS device.

IF (!D.NAME EQ 'PS') THEN BEGIN

      ; Need a gray-scale color table if this is a true
      ; color image.
```

```
   IF true GT 0 THEN LOADCT, 0, /Silent
   TV, image, xstart, ystart, XSIZE=xsize, $
     YSIZE=ysize, _EXTRA=extra, True=true

ENDIF ELSE BEGIN

     ; If the image is 24-bit but the display is 8-bit
     ; then COLOR_QUAN processing is required.

   IF (!D.Flags AND 256) GT 0 THEN BEGIN
     thisWindow = !D.Window
     Window, XSize=10, YSize=10, /Free, /Pixmap
     WDelete, !D.Window
     WSet, thisWindow
   ENDIF
   ncolors = !D.N_Colors
   IF ncolors LE 256 AND true GT 0 THEN BEGIN
     TV, Congrid(COLOR_QUAN(image, true, red, green, blue, $
       Colors=!D.N_Colors), CEIL(xsize), CEIL(ysize), $
       MINUS_ONE=minusOne), xstart, ystart, _EXTRA=extra
     TVLCT, red, green, blue
     RETURN
   ENDIF

   CASE true OF
     0: TV, CONGRID(image, CEIL(xsize), CEIL(ysize), /INTERP, $
          MINUS_ONE=minusOne), xstart, ystart, _EXTRA=extra
     1: TV, CONGRID(image, 3, CEIL(xsize), CEIL(ysize), /INTERP, $
          MINUS_ONE=minusOne), xstart, ystart, _EXTRA=extra, True=1
     2: TV, CONGRID(image, CEIL(xsize), 3, CEIL(ysize), /INTERP, $
          MINUS_ONE=minusOne), xstart, ystart, _EXTRA=extra, True=2
     3: TV, CONGRID(image, CEIL(xsize), CEIL(ysize), 3, /INTERP, $
          MINUS_ONE=minusOne), xstart, ystart, _EXTRA=extra, True=3
  ENDCASE
ENDELSE
END
```

File Attachments

1) myct.pro, downloaded 112 times
2) open_device.pro, downloaded 135 times
3) close_device.pro, downloaded 122 times
4) tvimage.pro, downloaded 126 times

Subject: Re: Wanted: colour table good for high contrast grayscale output
Posted by Martin Schultz on Fri, 06 Mar 1998 08:00:00 GMT

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 117 times

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

i001@justwright.com (Iarla Kilbane-Dawe) wrote:

> 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.

I don't know if it will look nice, but you may want to consider
constructing a periodic color table, something like

  black, dark gray, light gray, white, light gray, dark gray, black

repeated N times... This would mimic the way the interference pattern
("Newton rings") of monochromatic light on an "oily water surface"
encodes the thickness of the oil layer...

No guarantees, though!

Regards,

Stein Vidar

Subject: Re: Wanted: colour table good for high contrast grayscale output
Posted by David Foster on Mon, 09 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 -

I would suggest modifying the grayscale. You can use IDL's STRETCH(),
but it would be easier to use my GRAYSCALE.PRO that allows you to
use sliders. This can be run by itself or as a popup from within
another module. You can download it from:

ftp://bial8.ucsd.edu pub/software/idl/share/idl_share.tar.gz

Hope this helps!

Dave
--

~~~~~~ ~~~~~~~
David S. Foster Univ. of California, San Diego
Programmer/Analyst Brain Image Analysis Laboratory
foster@bial1.ucsd.edu Department of Psychiatry
(619) 622-5892 8950 Via La Jolla Drive, Suite 2240
La Jolla, CA 92037
~~~~~~ ~~~~~~~