

---

Subject: Re: TVIM display routine on 24-bit displays

Posted by [paul](#) on Thu, 11 Jan 2001 19:22:23 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

David Williams <d.williams@qub.ac.uk> writes:

> Hi, there.

>

> I've recently made the switch (as a user, not as sysadmin!) from a Sun  
> Ultra running Solaris to a PC running Red Hat 6.2. No big deal, perhaps,  
> but I'm having to come to terms with using a decent graphics card with  
> 24-bit colour when using IDL (and the various astronomy add-ons like CDS  
> and SolarSoft).

>

> My problem is this: I used to use a very nifty display routine by Paul  
> Ricchiazzi called TVIM.PRO, which always did the job brilliantly for me.  
> It still plots fine on a postscript device, but when you use it to  
> display an image on your 24-bit colour XWindow, the colour table doesn't  
> scale nicely. If you're using colour table 0 (greyscale), for example,  
> to view a 2-D image, then it doesn't scale from black (minimum data  
> value) to white (maximum data value) as it would under an 8-bit display.

>

> I know that the colour system under 24-bit isn't as straightforward as  
> under 8-bit, for which (I read) IDL is optimised.

>

> Because so many of my routines (and some of my colleagues') depend on  
> TVIM and all its nice keywords, does anyone know of a way to make TVIM  
> display things properly on a 24-bit XWindow? The only kludge I've been  
> able to get to work so far has been to get our sysadmin to cripple the  
> machine's display to 8-bit colour, which seems like a rather ham-fisted  
> way of doing things.

>

> Any help would be greatly appreciated,

>

> thanks for taking the time to read this,

>

> Dave Williams,

> Belfast.

Hi Dave,

You're in luck. I recently put 24-bit color into tvim. This new version is included in an updated version of the esrg idl library, available from:

<ftp.icess.ucsb.edu:/pub/esrg/idl>

and to make things easy I've also included it below.

Ciao,

Paul Ricchiazzi

```
.....  
;+  
; ROUTINE: tvim  
;  
; USEAGE: tvim,a  
;  
; tvim,a,scale=scale,range=range,xrange=xrange,yrange=yrange, $  
; aspect=aspect,title=title,xtitle=xtitle,ytitle=ytitle, $  
; noaxis=noaxis,interp=interp,colors=colors,c_map=c_map, $  
; stitle=stitle,rmarg=rmarg,clip=clip,labels=labels, $  
; pcharsize=pcharsize,lcharsize=lcharsize,nbotclr=nbotclr, $  
; clevels=clevles,nodata=nodata,rgb_nodata=rgb_nodata, $  
; barwidth=barwidth,position=position,noframe=noframe,rct=rct  
;  
; PURPOSE: Display an image with provisions for  
;  
; 1. numbered color scale  
; 2. plot title  
; 3. annotated x and y axis  
; 4. simplified OPLOT capability  
; 5. manual or automatic scaling with smooth or discreet colors  
; 6. special treatment of "invalid data" pixels  
; 7. display true color images  
;  
; INPUT  
; a image quantity  
;  
; if A is a two dimensional array it is assumed to be an  
; array of a physical quantity in the same units as specified  
; in RANGE.  
;  
; if A is a three dimensional array it is assumed to be a  
; band-interleaved true color image in the form  
; (3,nx,ny) or (nx,ny,3) [either form will work], where  
; nx and ny are the number of pixels in the x and y  
; directions and the red, green and blue components are  
; specified by the third index. All keywords associated  
; with scaling the image to the color key (e.g., SCALE,  
; RANGE, CLEVELS, STITLE etc.) are disabled for true color  
; images.  
;
```

```
; Before display the image array is scaled to the 0-255
; range using bytscl. It is then routed through the
; color tables. The red color table array contains the
; intensity translation table for the red image, and so
; forth. Assuming that the color tables have been loaded
; with the vectors R, G, and B, a pixel with a color
; value of (r, g, b) is displayed with a color of (R[r],
; G[g], B[b]). As with other devices, a color table
; value of 255 represents maximum intensity, while 0
; indicates an absence of the color. To pass the RGB
; pixel values without change, load the red, green and
; blue color tables with a ramp with a slope of 1.0:
; TVLCT, INDGEN(256), INDGEN(256), INDGEN(256) or with
; the LOADCT procedure: LOADCT, 0 ; Load standard
; black/white table.
```

```
; True color images can be written to postscript using
```

```
; toggle,/color
; tvim,a
; toggle
```

```
; NOTE: To write a gif image you'll have to use color_quan to
; quantize the true color image into 256 colors. This
; procedure produces a pretty lousy result:
```

```
; v=color_quan(a,3,r,g,b,color=256)
; tvlct,r,g,b
; tvim,v
; gif_write,'junk.gif'
```

```
; NOTE: if you have a pc you can use a screen capture
; to cut and paste the image into a document ("ALT
; print-screen" copies the screen into the paste
; buffer).
```

```
; Optional keyword input:
```

```
; title
; plot title
;
; xtitle
; x axis title
;
; ytitle
; y axis title
```

```

;
; stitle
;   color key title (drawn to the right of color scale)
;
;
; rmarg
;   right margin expansion factor to provide more room for extra wide
;   color scale annotation (default=1)
;
;
; xrange
;   array spanning entire x axis range. (default = [0,x_dimension-1])
;
;
; yrange
;   array spanning entire y axis range. (default = [0,y_dimension-1])
;
;
;   NOTE:TVIM only uses min(XRANGE), max(XRANGE), min(YRANGE) and max(YRANGE).
;
;
; scale
;   if set draw color scale. SCALE=2 causes stepped color scale
;
;
; range
;   two or three element vector indicating physical range over which
;   to map the color scale. The third element of RANGE, if specified,
;   sets the step interval of the displayed color scale. It has no
;   effect when SCALE is not set. E.g., RANGE=[0., 1., 0.1] will
;   cause the entire color scale to be mapped between the physical
;   values of zero and one; the step size of the displayed color
;   scale will be set to 0.1.
;
;
; clip
;   specifies a set percentage of pixels which will be left outside
;   the color scale. Larger values of CLIP yield images with greater
;   contrast. For example if clip=2 then the color scale range will
;   be set to include 98% of the pixels and to exclude the brightest
;   1% and the dimmest 1% of the pixel brightness distribution.
;   Keyword RANGE overrides the effect of CLIP.
;
;
;   If CLIP is a two element vector, the upper and lower percentile
;   range can be specified. For example CLIP=[0,95] will exclude the
;   top 5%.
;
;
;   An alternative is to set CLIP to a single negative integer. This
;   causes the grey scale range to be obtained from the max and min
;   value of the image array after it is filtered by a 2-D median
;   filter. The radius of the median filter is set to -clip. Median
;   filtering is effective in removing "salt and pepper" noise,
;   (isolated high or low values). For example, setting clip=-1
;   causes the code to check the 9 element superpixel centered at
;   each pixel for median values. A single extremely large or small

```

```

; value does not affect the final value of the median at each pixel
; point.
;
; aspect
; the x over y aspect ratio of the output image. If not set, aspect
; is set to (size_x/size_y) of the input image.
;
; position
; specifies the lower left and upper right TVIM frame position in
; normal coordinates. When set POSITION overrides the setting of
; ASPECT, and disables screen blanking for new plots.
;
; noaxis
; if set do not draw x and y numbered axis around image. Instead
; just draw a box
;
; noframe
; if set no not draw anything around image, not even a box
;
; interp
; TVIM normally uses nearest neighbor sampling to resize the image
; to fill the data window area. If INTERP is set the resizing
; operation uses bilinear interpolation
;
; If INTERP = 2 the interpolation range is adjusted so that there
; is a half pixel border around the plot. This ensures that the
; smoothed image properly represents the cell center values of the
; pixels, but introduces an extrapolation on the outer edges. This
; option has noticeable effects for small (<20) image arrays.
;
; NOTE: When writing to a Postscript file the resizing operation is
; accomplished by changing the pixel size. Thus the INTERP
; parameter has no effect on Postscript output.
;
; pcharsize
; character size of numbers along x and y axis and the title
; (default is !p.charsize if non-zero, 1 otherwise)
;
; NOTE: The character size of the x and y-axis number scale
; annotation may be independently controlled by the !x.charsize or
; !y.charsize system variables. For example, the x and y axis
; number scale characters may be made invisible by setting these
; variables to a very small non-zero number before calling TVIM.
; Remember to restore normal character sizes by setting !x.charsize
; and !y.charsize to 1 or zero after the call to TVIM.
;
; lcharsize
; character size of color key number or labels (default is

```

```

; !p.charsize if non-zero, 1 otherwise)
;
; barwidth
; width of color key which appears to right of image when SCALE is
; set. (default=1)
;
; labels
; a vector of strings used to label the color key levels. If not
; set the default color key number labels are used. If the number
; of elements in LABELS is the same as the number of elements in
; COLORS then the labels will appear in the middle of a given
; color band instead of at the boundary between colors. If COLORS
; is not set the number of elements in LABELS should be at least
; as large as the number of color key segments plus one.
;
; colors
; an array of color indicies. When the COLORS array is set TVIM
; will map values of A into the color values specified in COLORS.
; How physical values are assigned to color values depends on how
; the RANGE parameter is set, as shown in this table:
;
; RANGE      color_value
; -----
; 1. not set  COLORS(A)
; 2. [amin,amax]  COLORS((A-amin)/dinc)
; 3. [amin,amax,inc]  COLORS((A-amin)/inc)
;
; where amin, amax and inc are user specified elements of RANGE
; and dinc=(amax-amin)/n_elements(COLORS). In case 1, A is used
; directly as a subscript into COLORS. When RANGE is not set
; legend labels should be used to annotate the color values. In
; case 2, A is scaled so that each color defined in COLORS
; represents a proportionate fraction of the total dynamic range.
; In case 3, amin and inc (and not amax) are used to control which
; colors are associated with given physical values. See examples.
;
; c_map
; TVIM normally rescales the input image to span the entire color
; table range. Setting C_MAP disables this automatic re-scaling.
; This is useful when the image byte values correspond to a
; particular color mapping that could be destroyed by the
; rescaling process (E.G. a gif image).
;
; NOTE: When the number of IDL color table entries is less than
; 256 some colors may be lost. Use a private color map to avoid
; this. See examples.
;
; nbotclr

```

```

; number of reserved colors at the bottom of the color
; table. Color values between 0 and nbotclr-1 will not
; be used in the displayed image. This allows the
; bottom of the table to be set to special colors which
; will not be used in the displayed the image. This
; keyword is disabled when either the COLORS or NODATA
; options are used. (default=1)
;
; NOTE: by default the color table indices used within a TVIM
; image is limited to the range 1 to !d.n_colors-2. In most
; predefined color tables, color index 0 and !d.n_color-1
; correspond to pure black and white, respectively. TVIM uses
; these color indices as background and foreground colors for
; region blanking and axis annotation. Color tables which do not
; define black and white in these index locatations are not good
; choices for TVIM plots, unless the first and last color table
; entries are redefined as black and white using TVLCT, e.g.,
; tvlct,0,0,0,0 & tvlct,255,255,255,!d.n_colors-1
;
; NOTE: the procedure DCOLORS can be used to load a set
; of discreet colors in the bottom 11 color values
; (0-10). The original color table is resampled to fit
; between color value 11 and the !d.n_colors-1.
;
;
; nodata
; pixel values which are interpreted as invalid data. Pixels
; containing this data value are colored with color RGB_NODATA.
;
; rgb_nodata
; an RGB value (a 3 component vector) used to color pixels filled
; with invalid data. (default = [0,0,0])
;
; NOTE: NODATA and RGB_NODATA override the effect of NBOTCLR
;
; rct
; if set, reverse direction of color table. This keyword can be
; used on black & white printers to allow large field values to be
; represented by dark colors. Note, this feature is implemented
; by reversing the association between physical values and color.
; Though it reverses the colors on the color key, it does not
; affect the actual color table.
;
;
; KEYWORD OUTPUT:
; clevels
; physical values at the color key tic marks. Use this to set
; contour levels in a subsequent call to CONTOUR.

```

```

;
; SIDE EFFECTS:
;   Setting SCALE=2 changes the color scale using the STEP_CT
;   procedure. The color scale may be returned to its original
;   state by the command, STEP_CT,/OFF
;
;
; PROCEDURE:
;   TVIM first determines the size of the plot data window with a
;   dummy call to PLOT. When the output device is "X", CONGRID is
;   used to scale the image to the size of the available data
;   window. Otherwise, if the output device is Postscript,
;   scaleable pixels are used in the call to TV.
;
;
;   If the input image quantity is not of type byte, TVIM converts
;   them to byte by scaling the input between 0 and !d.n_colors-1.
;   If the input is already a byte array and if either the number of
;   color table entries is 256 or if C_MAP is set, then no scaling
;   is done. This is to ensure consistency with specially defined
;   private color tables.
;
;
;   After the the image is drawn TVIM calls PLOT again to draw the x
;   and y axis and titles. Finally if scale is set, TVIM calls the
;   COLOR_KEY procedure to draw the color scale just to the right of
;   the data window.
;
;
; RESTRICTIONS:
;   A stepped color scale (SCALE = 2 option) is not available when
;   writing to a black & white postscript file (i.e., when the
;   DEVICE color option is not enabled).
;
;
; DEPENDENCIES: COLOR_KEY, STEP_CT, PRANK
;
;
; EXAMPLES:
;
;::: Plot a GIF image with its own preset RGB
;::: table (perhaps from a digitized video image)
;
; window,colors=256      ; set up a private color map
; read_gif,'/home/paul/gif/el-nino.gif',im,r,g,b
; tvlct,r,g,b
; tvim,im,/c_map
;:::
;:::
;::: Plot an image with descreet greyscale values to a printer
;::: which has its own hard-wired color table. check documentation of gray.pro
;
;
; im=randata(128,128,s=4)
; immx=max(im) & immn=min(im)

```



```

; !p.multi=[0,2,2]
; loadct,5
; tvim,im,/scale,range=[immn,immx],color=gray(4)
; tvim,im,/scale,range=[immn,immx],color=gray(5)
; tvim,im,/scale,range=[immn,immx],color=gray(8)
; tvim,im,/scale,range=[immn,immx],color=gray(10)
...
...
...
;;; Display a map of surface types (green=veg, blue=water, red=sand)
;;; Notice how RANGE is used to associate physical values with colors.
;;; (use im from the previous example)
;
; im=randata(128,128,s=4)
; immx=max(im) & immn=min(im) & delim=immx-immn
;
; tvlct,[0,0,0,255],[0,255,0,0],[0,0,255,0]
; !p.multi=[0,2,2]
; colors=[1,2,3] & labels=[' veg',' water',' sand']
; range=[immn,immx,delim/3]
;
; tvim,im,/scale
;
; tvim,im,colors=colors,range=range,/scale
;
; tvim,im,colors=colors,range=range,/scale,labels=labels,lch=2
;
; !p.multi=0
; range=[immn-.25*delim,immx+.25*delim,delim/2]
; lbl=[' veg!c region 1',' water!c region 2',' sand!c region 3']
; tvim,im,colors=colors,range=range,/scale,labels=lbl,lch=2,rm arg=2,pchar=1.2
...
;;; Display the image from previous example and overlay a contour plot
...
...
; !p.multi=0
; loadct,5
; im=randata(128,128,s=4)
; immx=max(im) & immn=min(im) & delim=immx-immn
; xrange=[-120,-100]
; yrange=[20,50]
; range=[immn,immx]
; xx=findrng(xrange,128)
; yy=findrng(yrange,128)
;
; tvim,im,/scale,xrange=xrange,yrange=yrange,clevels=clevels,range=range
; contour,im,xx,yy,levels=clevels,/overplot
;
;
;
;;; NOTE: You might prefer the results obtained from procedure CONFILL.

```

```

;
; confill,im,xx,yy,/asp,levels=clevels
...
;;;
...
;;;
...
Display a grey scale image and annotate certain points in color
...
...
; loadct,0
; dcolors
; tvim,im,/scale,nbotclr=11          ; reserve some colors at bottom
; xx=interpolate([40,80],randomu(iseed,10))
; yy=interpolate([40,80],randomu(iseed,10))
; plots,xx,yy,psym=2,color=1+indgen(10)
;
;
...
;;;
...
inset an image onto a line plot
...
...
;
; !p.multi=0
; loadct,0
; xy=randata(100,100,s=3)
; plot,xy[*],40]
; tvim,xy,pos=boxpos(/cur),title='transect',pcharsize=.7
; oplot,[0,99],[40,40]
;
;
...
;;;
...
...
Display a grey scale image and show nodata values in red
...
...
; loadct,0
; im=randata(100,100,s=4.5)
; im(randomu(iseed,50)*9999)=-999.
; tvim,im,/scale,nodata=-999.,rgb_nodata=[255,0,0]
;
;
...
...
...
...
Postscript output with a reversed color scale. Because the
...
background color is always RGB=(255,255,255) you must set the
...
default color, !p.color, to a dark color if you want good contrast
...
between the axis labels and the white paper. According to
...
Didier's reversed color table a color index of 255 should produce
...
black, but for some reason !p.color=255 doesn't work right.
...
I think IDL is interpreting !p.color=255 in some special way.
...
So instead use !p.color=254; this seems to work fine.
...
...
; toggle,/color
; loadct,28
; !p.color=254          ; don't use 255, don't ask me why
; tvim,dist(20),/scale

```

```

; toggle
;
;
;
; display data defined on a regular LAT-LON grid onto a given map
; projection. USE MAP_SET and MAP_IMAGE to create the map projection
; and to warp the image. Then use BOXPOS to position the TVIM frame
; to correctly register the map and image
;
;
; w8x11
; IMAGE = sin(!pi*findrng(0,24,200))#sin(!pi*findrng(0,12,200))
; !p.multi=[0,1,2]
; map_set,45,0,/ortho,/advance,pos=boxpos(/aspect)
; newimage=map_image(image,/bilin,latmin=-90,latmax=90,lonmin=-180,lonmax=180)
; tvim,newimage,title='Warped data',pos=boxpos(/get),/scale
; map_set,45,0,/ortho,pos=boxpos(/get),/grid,/cont,/noerase ; draw map
; tvim,image,xrange=[-180,180],yrange=[-90,90],/scale, $
; title='Unwarped data',xtitle='Longitude',ytitle='Latitude'
; map_set,0,0,/cyl,pos=boxpos(/get),/grid,/cont,/noerase ; draw map
;
;
; use MAP_SET2 to mask out land areas. Note that the example below is
; unusual. MAP_SET2 doesn't always produce such nicely filled land areas.
; Typically one must run MAP_SET2 with /SAVE
; to create a ascii file containing the continental boundary lat-lon
; coordinates. Then use an editor to group the continental coordinates
; line segments to form closed contours which POLYFILL can understand
; (the call to POLYFILL is enabled by setting con_color).
;
;
;
; image=randata(256,256,s=2.5)
; tvim,image,yrange=[-65.1,-64],xrange=[-64.5,-62]
; map_set5,-64,-64,/cont,limit=[-65.1,-64.5,-64,-62],/pos,/noerase
; map_set,-64,-64,/cont,limit=[-65.1,-64.5,-64,-62],pos=boxpos (/get),/noerase
;
;
; display a true color image
;
;
; a=fltarr(100,100,3)
; g=bytsc(1,100)#(findgen(100)>30))
; a(*,*,0)=rotate(g,1)
; a(*,*,1)=g
; a(*,*,2)=rotate(g,3)
; !p.multi=[0,2,2]
; loadct,0
; tvim,a,xrange=[100,300],yrange=[0,10],title='true color image'
; tvim,a(*,*,0),range=[0,255],title='red component'
; tvim,a(*,*,1),range=[0,255],title='green component'
; tvim,a(*,*,2),range=[0,255],title='blue component'
;
;
; other interleaving possibility

```

```

;
; a=fltarr(3,100,100)
; g=bytsc1(replicate(1,100)#(findgen(100)>30))
; a(0,*,*)=rotate(g,1)
; a(1,*,*)=g
; a(2,*,*)=rotate(g,3)
; !p.multi=[0,2,2]
; loadct,0
; tvim,a,xrange=[100,300],yrange=[0,10],title='true color image'
; tvim,a(0,*,*),range=[0,255],title='red component'
; tvim,a(1,*,*),range=[0,255],title='green component'
; tvim,a(2,*,*),range=[0,255],title='blue component'
;
;
;
;;; display a sparse data field superposed on a background topography.
;;; topography shown with a gray scale, data filled areas shown with
;;; color table 5
;
; minval=.1           ; minimum physical value of interest
; nbotclr=30         ; number of colors assigned to gray scale
; maxgray=240        ; max gray scale intensity (240 out of 255)
; topo=randata(200,200,s=4) ; background topography variable (just for looks)
; data=randata(200,200,s=2) ; data variable
; loadct,5           ; color table 5
;
;
; topo=topo-max(topo)
; rngdata=max(data)-minval
; rngtopo=-min(topo)
; topo=nbotclr/(!d.n_colors<256.-nbotclr)*rngdata*topo/rngtopo
; topo=topo+minval
; ii=where(data lt minval)
; data[ii]=topo[ii]
; r=findrng(0,maxgray,nbotclr) & g=r & b=r
; tvlct,r,g,b,0
; tvim,data,rmarg=1
; color_key,range=data>minval,nbotclr=nbotclr
;
;
;;;
; AUTHOR:   Paul Ricchiazzi   oct92
;           Earth Space Research Group, UCSB
;
;
; REVISIONS:
; 28jan93: switched from PUT_COLOR_SCALE to COLOR_KEY
; 11feb93: added INTERP keyword
; 11feb93: added C_MAP keyword
; 11feb93: added STITLE keyword
; 20apr93: added RMARG keyword, centered image in plot window
; 23apr93: call COLOR_KEY before TV. Now stepped color scales copy to PS file

```

```

; 10sep93: deal with perfectly flat images. No more math error when a=0
; 14sep93: added CLIP keyword
; 11feb94: added PCHARSIZE,LCHARSIZE,LABELS, and COLORS keywords
; 18feb94: added NBOTCLR keyword
; 16mar94: added INTERP=2 option and adjusted centering of pixels
; 08Jul94: added NODATA and RGB_NODATA keywords
; 04Aug94: added BARWIDTH keyword
; 04Aug94: defaulted PCHARSIZE and LCHARSIZE to !P.CHARSIZE if non-zero
; 02Mar95: added POSITION keyword
; 31Mar95: added NOAXIS keyword
; 24may95: initial sizing includes allowance for PCHARSIZE > 1
; 06sep95: color indices used in plot now run from 1 to !d.ncolors-2
; 24jul96: do a REFORM(a) to allow tvim,a(1,*,*) to work with no complaints
; 01sep96: added the RCT keyword
; 17may00; treat true color images
; 01dec00; when position keyword is set, add to current page
;-

```

```

pro tvim,a,scale=scale,range=range,xrange=xrange,yrange=yrange,$
  aspect=aspect,title=title,xtitle=xtitle,ytitle=ytitle,$
  noframe=noframe,noaxis=noaxis,interp=interp,colors=colors,c_map=c_map,$
  stitle=stitle,rmarg=rmarg,clip=clip,labels=labels,clevels=clevels,$
  pcharsize=pcharsize,lcharsize=lcharsize,nbotclr=nbotclr,nodata=nodata,$
  rgb_nodata=rgb_nodata,barwidth=barwidth,position=position,rct=rct

```

```

if n_params() eq 0 then begin
  xhelp,'tvim'
  return
endif

```

```

if keyword_set(position) then begin
  psave=!p.multi
  !p.multi=[1,1,1]
endif

```

```

if keyword_set(pcharsize) eq 0 then begin
  if !p.charsize eq 0 then pcharsize=1 else pcharsize=!p.charsize
endif

```

```

if keyword_set(lcharsize) eq 0 then begin
  if !p.multi(1)>!p.multi(2) gt 2 then lcharsize=.5*pcharsize $
    else lcharsize=pcharsize
endif

```

```

if not keyword_set(interp) then interp=0

```

```

sz=size(reform(a))

```

```

if sz(0) eq 3 then begin
  true_color=1
  case 1 of
    sz(1) eq 3: true_index=1
    sz(3) eq 3: true_index=3
    else: message,'true color array must be of form (nx,ny,3) or (3,nx,ny)'
  end
  if true_index eq 1 then begin
    nx=sz(2)
    ny=sz(3)
  endif else begin
    nx=sz(1)
    ny=sz(2)
  endelse
endif else begin
  true_color=0
  nx=sz(1)
  ny=sz(2)
endelse

nxm=nx-1
nym=ny-1
plot, [0,1],[0,1],/nodata,xstyle=4,ystyle=4,charsize=pcharsize
px=!x.window*!d.x_vsize
py=!y.window*!d.y_vsize
xsize=px(1)-px(0)
ysize=py(1)-py(0)

if keyword_set(scale) or keyword_set(rmarg) then begin
  if keyword_set(rmarg) eq 0 then begin
    if(!d.name eq "WIN") then begin
      rmarg=2.
    endif else begin
      rmarg=1.
    endelse
  endif
  xmarg=!d.x_ch_size*(4+4*!charsize)
  if keyword_set(stitle) then xmarg=xmarg+2*!d.y_ch_size*!charsize
  xsize=xsize-rmarg*xmarg
endif
oxs=xsize
oys=ysize
if keyword_set(aspect) eq 0 then aspect=float(nx)/ny
if xsize gt ysize*aspect then xsize=ysize*aspect else ysize=xsize/aspect
if xsize lt oxs then px(0)=px(0)+.5*(oxs-xsize)
if ysize lt oys then py(0)=py(0)+.5*(oys-ysize)
px(1)=px(0)+xsize
py(1)=py(0)+ysize

```

```

if keyword_set(position) then begin
  px=position([0,2])*!d.x_vsize
  py=position([1,3])*!d.y_vsize
  xsize=px(1)-px(0)
  ysize=py(1)-py(0)
endif

pos=[px(0),py(0),px(1),py(1)]

if true_color then begin ; true color image

  if !d.name eq 'PS' then begin
    tv,bytsc1(a),px(0),py(0),xsize=xsize,ysize=ysize,/device,true_index
  endif else begin
    mone=interp
    if interp lt 2 then begin
      fx=(nx-1)*findgen(xsize)/(xsize-1)
      fy=(ny-1)*findgen(ysize)/(ysize-1)
    endif else begin
      fx=nx*findgen(xsize)/(xsize-1)-.5
      fy=ny*findgen(ysize)/(ysize-1)-.5
    endelse
    device,decomposed=1
    if true_index eq 3 $
      then tv,interpolate(bytsc1(a),fx,fy,indgen(3),/grid),px(0),py(0), true=3 $
      else tv,interpolate(bytsc1(a),indgen(3),fx,fy,/grid),px(0),py(0), true=1
    device,decomposed=0
  endelse

endif else begin

;
;max_color=!d.n_colors-1
max_color=(!d.n_colors-2) < 255
;
if keyword_set(title) eq 0 then title="

amax=float(max(a))
amin=float(min(a))

num_range=n_elements(range)

if n_elements(nodata) then begin
  if n_elements(rgb_nodata) eq 0 then rgb_nodata=[0,0,0]
  iinodata=where(a eq nodata,ncnd)
  iidata=where(a ne nodata,ncgd)
  if ncgd gt 0 then begin

```

```

    amin=float(min(a(iidata)))
    amax=float(max(a(iidata)))
    tvlct,rgb_nodata(0),rgb_nodata(1),rgb_nodata(2),1
    nbotclr=2
endif
endif else begin
    iinodata=-1
endif else
endif

if num_range eq 1 then message,'must specify either 2 or 3 elements of RANGE'

if num_range eq 0 then begin
if amin eq amax then begin
    case 1 of
        amax eq 0.: rng=[-1,1.]
        amax gt 0.: rng=[0,2*amax]
        amax lt 0.: rng=[2*amax,0]
    endcase
endif else BEGIN
    CASE n_elements(clip) OF
        1:if clip ge 0 then begin
            rng=prank(a,[clip,100-clip])
        endif else begin
            clp=-2*clip+1
            ix1=-clip & ix2=nx+clip
            iy1=-clip & iy2=ny+clip
            rng=[min((median(a,clp))(ix1:ix2,iy1:iy2),max=mx),mx]
        endelse
        2:rng=prank(a,clip)
        else:rng=[amin,amax]
    endcase
endif else
endif else begin
    rng=range(0:1)
    if num_range eq 3 then inc=range(2) else inc=0
endif else
endif

ncolors=n_elements(colors)

case 1 of

ncolors ne 0 : begin
    if keyword_set(rct) then message,'rct invalid with color specificaton'
    case num_range of
        0:begin
            aa=colors(fix(reform(a)))
            rng=[0,ncolors]
            inc=1

```



```

end
2:begin
  clrinc=(rng(1)-rng(0))/ncolors
  aa=colors(fix((reform(a)-range(0))/clrinc))
end
3:begin
  aa=colors(fix((reform(a)-range(0))/range(2)))
  inc=range(2)
  rng=[range(0),range(0)+ncolors*inc]
end
endcase
end

keyword_set(c_map) eq 1 : begin
  if keyword_set(rct) then message,'rct invalid with c_map specifcatorn'
  aa=byte(reform(a))
  rng(0)=0
  rng(1)=max_color
end

else: begin
; if keyword_set(nbotclr) eq 0 then nbotclr=0 else nbotclr=nbotclr
  if keyword_set(nbotclr) eq 0 then nbotclr=1 else nbotclr=nbotclr
  max_color=max_color-nbotclr
  aa=nbotclr+max_color*((float(reform(a))-rng(0))/(rng(1)-rng( 0)) >0.<1.)
  if keyword_set(rct) then aa=max_color+2*nbotclr-aa
end
endcase

;
posbar=[px(1)+!d.x_ch_size,py(0)]
if keyword_set(scale) then begin
  if scale eq 2 then step=1 else step=0
  if not keyword_set(barwidth) then barwidth=1
  if ncolors eq 0 then begin
    color_key,pos=posbar,ysize=ysize,range=rng,inc=inc,step=step,$
    title=stitle,charsize=lcharsize,labels=labels,nbotclr=nbotcl r,$
    clevels=clevels,barwidth=barwidth,rct=rct
  endif else begin
    color_key,pos=posbar,ysize=ysize,range=rng,inc=inc,title=sti tle,$
    charsize=lcharsize,colors=colors,labels=labels,clevels=cleve ls,$
    barwidth=barwidth,rct=rct
  endelse
endif
endif

if iinodata(0) ne -1 then aa(iinodata)=1

if !d.name eq 'PS' then begin

```

```

    tv,aa,px(0),py(0),xsize=xsize,ysize=ysize,/device
endif else begin
    mone=interp
    if interp lt 2 then begin
        tv,congrid(aa,xsize,ysize,interp=interp,minus_one=mone),px(0 ),py(0)
    endif else begin
        tv,interpolate(aa,nx*findgen(xsize)/(xsize-1)-.5,$
            ny*findgen(ysize)/(ysize-1)-.5,/grid),px(0),py(0)
    endelse
endelse

endelse
;
if keyword_set(xtitle) eq 0 then xtitle=""
if keyword_set(ytitle) eq 0 then ytitle=""

if not keyword_set(xrange) then xrng=[0,nxm] $
    else xrng=[min(xrange), max(xrange)]
if not keyword_set(yrange) then yrng=[0,nym] $
    else yrng=[min(yrange), max(yrange)]

case 1 of

    keyword_set(noframe):plot,[0,0],[0,0],xstyle=5,ystyle=5,position=pos,$
        title=title,/device, charsize=pcharsize, $
        xrange=xrng,yrange=yrng,/noerase,/nodata

    keyword_set(noaxis): plot,[0,0],[0,0],/xstyle,/ystyle,position=pos,$
        title=title,xtitle=xtitle,ytitle=ytitle,/device, $
        xrange=xrng,yrange=yrng,/noerase,/nodata,$
        charsize=pcharsize,xticks=1,yticks=1,$
        xtickname=[" "," "],ytickname=[" "," "]

    else:      plot,[0,0],[0,0],/xstyle,/ystyle,title=title,/nodata,$
        xtitle=xtitle,ytitle=ytitle,xrange=xrng,yrange=yrng,$
        position=pos,/noerase,/device,charsize=pcharsize

endcase

if keyword_set(position) then !p.multi=psave
;
end

```