Subject: Re: color_quan(...., Cube=6) makes white white, but ...
Posted by paul.krummel on Wed, 01 May 2002 02:46:34 GMT
View Forum Message <> Reply to Message

nobody@nowhere.com (Steve Smith<steven_smith>) wrote in message
news:<slrnacu5e9.e6b.nobody@pooh.nrel.gov>...
> On Tue, 30 Apr 2002 18:37:58 -0000, Steve Smith<steven_smith>
> <nobody@nowhere.com> wrote:
>
> <...snip...>
>
> I'm speculating that perhaps the thing to do is create a postscript document by
> printing the Word document to a file (and massaging out the non-postscript
> commands WinXX sticks in) and converiting that directly to PDF. Unfortunately,
> it doesn't fix the original problem: I need to give a figure that can be
> inserted into Word by a colleauge.

I thought I would just add what I do as I come across this all
the time while trying to share files or show some results. The
following is all done on a Windows 2000 PC.

As has already been said by a few people, if I am producing
a document that is to be printed on a PostScript printer,
then I use PostScript from IDL, turn it to an EPS and add
a preview with GSView. I then import this into Word and
I can rescale it etc and all prints very nicely.

However, if I am giving a presentation with say PowerPoint,
then I like to use images instead of EPS files. I have been
using PNG files with great success. I usually convert my PS
output to 300dpi PNG images using gsview. These import nicely
into Word or Powerpoint and at this resolution look very good
on screen and even when printed. In fact, this is how I give
plots and figures to other people in our group here as they
can easily use them. These images scale well in Word or
Powerpoint, and they have even been used in large posters
which look good printed.

I got sick of doing all this manually, so I now incorporate
the conversion to PNG and cropping the image into some of
my IDL code. Here is a simple example that makes a simple
plot in postscript, spawns gswin32c to convert to 300dpi
PNG file, then calls a routine I wrote called 'crop_image.pro'
to automatically crop the image.

Anyway, for what it is worth, this is what I do!
Cheers, Paul

```
pro ps_image_crop
;
; Quick routine to make a PS file,
; convert it to PNG and crop the
; image.
; PBK 1 May 2002.
; ++++
; Setup a test PS file
ps_file='c:\krum\gaslab\muck\test.ps'
png_file='c:\krum\gaslab\muck\test.png'
png_crop='c:\krum\gaslab\muck\crop.png'
;
; Create some data
x=findgen(200)*0.1
y=sin(x)
;
; ++++
; Setup plotting
!p.multi=0
set_plot,'ps'
device,file=ps_file,/helvetica,/color,bits=8
!p.font=0
!p.thick=4
!x.thick=4
!y.thick=4
;
; load rainbow colour table
loadct,13
;
plot, x, y, xtitle='X', ytitle='sin(X)', $
 title='TEST2 - abc ijk xyz',/nodata
oplot,x,y,color=255
oplot,x,y,color=64,psym=1
;
; close ps device
device,/close
;
; ++++
; convert the ps file into PNG files
spawn,'"c:\program files\gs\gs7.04\bin\gswin32c"'+ $
 ' -sDEVICE=png256 -r300 -q -dNOPAUSE -dBATCH '+ $
 '-sPAPERSIZE=a4 -sOutputFile='+png_file+' '+ps_file, $
 /log_output
;
; Note version of gostscript and path may need to be
; changed depending on what you have installed.
; Note usage of 'gswin32c' which is the command line
; version of gswin32. Here is a quick explanation
; of some of the command line options:
; -sDEVICE=png256
; convert to PNG with 256 colours or 16 million (16m)
;
; -r300  -> convert at 300 dpi see documentation for
; available resolutions.
;
; -q -dNOPAUSE -dBATCH
; various options for batch processing and being quiet
;
; -sPAPERSIZE=a4  -> paper size
;
; -sOutputFile=png_file
; output file (note for multiple pages you can set
; something like this for the file name 'out_page%d.png'
; where %d will be the page number, so if there are
; 4 pages in the PS file then there will be 4 image files
; called out_page1.png, out_page2.png, out_page3.png
; and out_page4.png.
;
; ++++
; Now crop the image automatically with a 2% buffer around it
; and write out to new file.
crop_image,png_file,out_file=png_crop
; or crop it and overwrite the file
;crop_image,png_file
; or if in landscape, crop it and rotate it
;crop_image,png_file,rot=1
;
; ++++
;
beep
end

*******************************************
;+
; NAME:
; CROP_IMAGE
;
; PURPOSE:
; This procedure performs an autocrop on an image in
; IDL based on white space or white background. It
; will work with 8-bit (256 colours) or 24-bit (true
; colour) images. It can also rotate images if required.
;
; CATEGORY:
; Image manipulation
;
```

```
; CALLING SEQUENCE:
; CROP_IMAGE, Image_File, OUT_FILE=Out_File, COORDS=CoOrds, ROT=Rot
;
; INPUTS:
; Image_File: This is the path and filename of the image
;   file to crop. Must be a string.
;
; KEYWORD PARAMETERS:
; OUT_FILE: Set this keyword to the desired output filename. If
;   this keyword is not set then the default is to overwrite
;   the input image file (Image_File above) with the cropped
;   image. Currently the output image will be in the same
;   format as the input image. Must be a string.
;
; COORDS:  Set this keyword to a four element array containing
;   the min and max pixel locations where you want the image
;   to be cropped eg. [x_min,y_min,x_max,y_max]. If this
;   keyword is not set, the default is to crop the image to
;   allow a 2% buffer of the total image pixel size on all
;   sides of the image from the first non white pixels.
;   NOTE: In some image formats the first pixel row starts at
;   the top of the image while in others it is at the bottom
;   of the image.
;
; ROT:  Set this keyword to the rotation (direction) that
;   is required based on the following table:
;  Direction  Transpose?  Rotation Counterclockwise  X1  Y1
;   0          No       None                  X0 Y0
;   1          No        90ï¿½                  -Y0  X0
;   2          No       180ï¿½                  -X0  -Y0
;   3          No       270ï¿½                   Y0  -X0
;   4          Yes      None                  Y0  X0
;   5          Yes       90ï¿½                  -X0  Y0
;   6          Yes      180ï¿½                  -Y0  -X0
;   7          Yes      270ï¿½                   X0  -Y0
;
;
; OUTPUTS:
; Writes an image to file, see OUT_FILE above.
;
; RESTRICTIONS:
; Only works for 8- or 24-bit images. Requires IDL 5.4 or higher.
;
; PROCEDURE:
; Straightforward image reading and cropping (sub array extraction)
; based on COORDS OR simple checking for first non-white pixel
; locations (if COORDS keyword not set) and taking 2% buffers
; around these.
```

```
;
; EXAMPLE:
; Read in the image CapeGrim_Map.png, auto crop it and write out
; to same file name:
;  IDL> crop_image,'c:\krum\gaslab\map\CapeGrim_Map.png'
; OR also rotate it 90 degress counter clockwise
;  IDL> crop_image,'c:\krum\gaslab\map\CapeGrim_Map.png', rot=1
;
; MODIFICATION HISTORY:
;  Written by: Paul Krummel, CSIRO Atmospheric Research, 28 Febuary 2001.
;  Modified by Paul Krummel, CAR, 25 April 2001. Added COORDS keyword.
; Modified by Paul Krummel, CAR, 23 August 2001. Added ROT keyword.
; Modified by Paul Krummel, CAR, 17 January 2002. Added file checking
;  to see if input image file variable (im_file) is of type string
;  and if input image file exists or not.
; Modified by Paul Krummel, CAR, 31 January 2002. Fixed oversight (bug)
;  with rotating the image when it is 24 bit!! ROT now works for
;  24-bit images.
;-

PRO CROP_IMAGE, Im_File, OUT_FILE=Out_File, COORDS=CoOrds, ROT=Rot
;
; =====>> HELP
;
on_error,2
if (N_PARAMS(0) NE 1) or keyword_set(help) then begin
   doc_library,'CROP_IMAGE'
   if N_PARAMS(0) NE 1 and not keyword_set(help) then $
              message,'Incorrect number of parameters, see above for usage.'
   return
endif
;
; ++++
; THINGS STILL TO DO:
; Add options for output image format
; Allow passing in an image array and passing image back again i.e.
; no reading/writing from file on disk.
;
; ++++
;
; Make sure input image file variable is a string
if size(im_file,/type) ne 7 then message,'"image_file" must be a string!!'
;
; Check that the image files exists and is readable!
if not file_test(im_file,/read) then message,'Bugger ... '+im_file+ $
     ' does not exist or is not readable!!'
;
; Read in the image
```

```
ok=query_image(im_file,info)
if not ok then message,'Oh BUGGER ... '+im_file+ $
   ' is not a proper or supported image file!'
image = READ_image ( im_File, R, G, B)
;
; ++++
; Check to see if coords were passed in, if so use them!
if n_elements(coords) gt 0 then begin
 c_min=coords[0] & c_max=coords[2]
 r_min=coords[1] & r_max=coords[3]
;
; ++++
; If not then autocrop
ENDIF ELSE BEGIN
;
; ++++
; Check if it is an 8 bit or 24 bit image
 case info.channels of
  1: BEGIN ; 8-bit
;
;   find the min colour used in the image
   i_min=min(image)
;
;    Check for white (r,g,b=255), just a test!
;   White is usually ALWAYS at 255 which usually is
;    r=255, g=255, b=255, but just check to make sure!
   rgb=fix(r)+fix(g)+fix(b)
   wh=where(rgb[i_min:255] eq 765)+i_min
;   white can occur more than once!
;   Is usually 255 though, just use the max
   wh=max(wh)
;   Find where image is not white
   not_white=where(image ne wh, cnt_nw)
;
    END
;
  3: BEGIN ; 24-bit
;
;   24 bit image -> 16.7 million colours
   rgb=total(image,1)
;   White is 765
   wh=765
;   Find where image is not white
   not_white=where(rgb ne wh, cnt_nw)
;
    END
 endcase
;
```

```
;  Convert this to columns and rows
 ncol = info.dimensions[0]
 nrow = info.dimensions[1]
 col = not_white MOD ncol
 row = not_white / ncol
;
;  Find the min and max of the col and row
 c_min=min(col,max=c_max)
 r_min=min(row,max=r_max)
;
;  Find average of col and row sizes and take 2% of it
 buff=ceil((((c_max-c_min)+(r_max-r_min))/2)*0.02)
;
;  Now add this to make buffer around image and find new dimensions
 c_min= c_min-buff > 0
 c_max= c_max+buff < ncol-1
 r_min= r_min-buff > 0
 r_max= r_max+buff < nrow-1
;
; ++++
ENDELSE
;
; ++++
; resize the image, with 'buff' pixels added on all sides
case info.channels of
 1: image=image[c_min:c_max,r_min:r_max] ; 8 bit
 3: image=image[*,c_min:c_max,r_min:r_max] ; 24 bit
endcase
;
; ++++
; if requested, rotate the image
if n_elements(rot) gt 0 then begin
 case info.channels of
  1: image=rotate(image,rot) ; 8 bit image
  3: begin   ; 24 bit image
;   Rotate the individual channels of the image by requested amount
   ir=rotate(reform(image[0,*,*]),rot) ; 24 bit red
   ig=rotate(reform(image[1,*,*]),rot) ; 24 bit green
   ib=rotate(reform(image[2,*,*]),rot) ; 24 bit blue
;          Reassign the individual channels to 3D and delete the variable
   sz=size(ir,/dimension)
   image=bytarr(3,sz[0],sz[1])
   image[0,*,*]=temporary(ir)
   image[1,*,*]=temporary(ig)
   image[2,*,*]=temporary(ib)
    end
 endcase
endif
```

```
;
; ++++
; Write out the image
if keyword_set(out_file) then o_file=out_file $
else o_file=im_file
WRITE_image, o_file, info.type, Image, R, G, B
;
; ++++
end
```