Subject: using TVRD(true=0) with a 24-bit image and decomposed=0
Posted by odell on Mon, 29 Mar 2004 20:18:30 GMT
View Forum Message <> Reply to Message

To all IDL color-masters,

I am trying to simply read an image that was originally plotted like
this:

tv, image

where image is a 2D byte-array of colors IN MY CURRENT COLORTABLE; ie,
numbers from 0 to 255. However, I am on a 24-bit display. I have
device, decomposed=0
and backing-store set on retain=2.

If i do this:

im2 = tvrd()
tv, im2

I get way more color=255 than I should; i do not get back the precise
original image. I understand that I can get the RGB triples by doing
tvrd(true=1), but *I don't want the RGB triples*. I just want the
simple numbers from 0 to 255 that I originally plotted. How do I get
them using tvrd()?

Thanks anyone who can help with this!
Chris O'Dell

Subject: Re: using TVRD(true=0) with a 24-bit image and decomposed=0
Posted by Karl Schultz on Wed, 31 Mar 2004 21:42:39 GMT
View Forum Message <> Reply to Message

Sure, that will work fine as long as you constrain the conditions enough.
The general case just cannot be handled because that would involve
representing all possible 24-bit colors in an 256-entry color table, which
clearly cannot be done for reasons I've already mentioned.

I can think of two conditions that can cause problems with your approach.

1) I think that if your application set decomposed=1 and started writing
colors to the window as RGB (24-bit) values instead of using one of the
color table values, some of your output image pixels won't match and will
get left set to the wrong values. Of course, you can easily not do this in
your app, and things will be fine as long as you assure that all drawing
goes through the color table.

2) If you tried this on a 16-bit display, it probably won't work unless you set up your color table carefully. Let's assume that you have 5 bits per channel (for easing this discussion). If you have an entry in your color table of [192, 64, 66], this color will get stored in the frame buffer as [192, 64, 64] because the lower three bits get tossed. When you tvrd it back, you will get [192, 64, 64] and that color won't match your [192,64,66] in the color table. You can work around this by setting the color table so that the lower three bits are always zero - this won't affect the display since the bits can't fit in the frame buffer anyway.

So, your approach will work fine as long as you don't let any colors into the frame buffer that are not in the color table and you can really match the bits read back from the frame buffer with the contents of the color table.

Karl

"Chris ODell" <odell@aos.wisc.edu> wrote in message news:2fafbdc3.0403310941.bd2f45@posting.google.com...
> Thanks Karl...that makes a lot of sense about how IDL works. BTW, I
> wrote a simple replacement for TV that does what I want; it is a bit
> slower but it hasn't failed yet:
>
> function TV_CT, Xo, Yo, Nx, Ny, _extra=_extra
>
> if n_elements(Xo) eq 0 then Xo = 0
> if n_elements(Yo) eq 0 then Yo = 0
> if n_elements(Nx) eq 0 then Nx = !d.x_size - Xo
> if n_elements(Ny) eq 0 then Ny = !d.y_size - Yo
> im = tvrd(Xo, Yo, Nx, Ny, _extra = _extra, true=1)
> imc = reform(im[0,*,*] + im[1,*,*]*256L + im[2,*,*]*256L^2)
> tvlct, r, g, b, /get
> ct = r + g*256L + b*256L^2
> out = byte(imc)
>
> for c = 0, 255 do begin
> w = where(imc eq ct[c])
> if w[0] ne -1 then out[w] = c
> endfor
>
> return, out
> END
>
> Chris
>
> "Karl Schultz" <kschultz_no_spam@rsinc.com> wrote in message news:<106h6sil1087e38@corp.supernews.com>...

>> "Chris ODell" <odell@aos.wisc.edu> wrote in message
>> news:2fafbdc3.0403291218.95f4491@posting.google.com...
>>> To all IDL color-masters,
>>>
>>> I am trying to simply read an image that was originally plotted like
>>> this:
>>>
>>> tv, image
>>>
>>> where image is a 2D byte-array of colors IN MY CURRENT COLORTABLE; ie,
>>> numbers from 0 to 255. However, I am on a 24-bit display. I have
>>> device, decomposed=0
>>> and backing-store set on retain=2.
>>>
>>> If i do this:
>>>
>>> im2 = tvrd()
>>> tv, im2
>>>
>>> I get way more color=255 than I should; i do not get back the precise
>>> original image. I understand that I can get the RGB triples by doing
>>> tvrd(true=1), but *I don't want the RGB triples*. I just want the
>>> simple numbers from 0 to 255 that I originally plotted. How do I get
>>> them using tvrd()?
>>
>> The IDL docs for TVRD say:
>>
>> "If the display is a 24-bit display, and both the CHANNEL and TRUE
>> parameters are absent, the maximum RGB value in each pixel is returned."
>>
>> This probably explains why you don't get back exactly what you put in
and
>> why you see more 255 values.
>>
>> If you do a TV with 8-bit image data and a color table on a 24-bit
display,
>> IDL is going to translate your 8-bit data to 24 bits via the color table
and
>> write the 24-bit data to the screen. There is not much else IDL can do
>> here. It needs to set all 24 bits to get the right picture on the
screen.
>>
>> Once the image is stored in the 24-bit frame buffer, the 8-bit data is
lost,
>> from the point of view of the TV routine. Your IDL program could save
it
>> someplace if it is needed later.
>>

>> When IDL does the TVRD from the 24-bit frame buffer in this situation,
it
>> needs to translate it back to 8 bits somehow. Clearly, there are some
>> difficulties here. One approach is to take the current color table and
try
>> to match every 24-bit pixel from the display to one of the color table
>> entries. This would be extremely time consuming and may not always
"work".
>> The closest match may not be good enough, or if the color table contains
two
>> entries with the same color, which of the two should be used? So, IDL
>> simply behaves as stated above.
>>
>> Now, if you really wanted to, you could read all three channels with the
>> TRUE kwd, and then use COLOR_QUAN to get a color array and palette back
out
>> of your 24-bit image. This is essentially the approach I mentioned
above,
>> which you can go ahead and do. But I think you'd be better off saving
the
>> original 8-bit data somehow.
>>
>> Karl
