Subject: Re: COLOR_QUAN question
Posted by davidf on Wed, 18 Aug 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Daniel Peduzzi (peduzzi@mediaone.net) writes:

> My question concerns the R, G, and B arrays returned by the COLOR_QUAN
> function. I've noticed that I don't receive the same RGB values if I call the function
> multiple times with the same input arguments. This isn't very noticeable upon visual
> inspection of the resulting images, unless the differences are exaggerated by color map
> operations such as histogram equalization.

Oh, oh. Hold on here. I think we may be fooling ourselves
a bit. First of all, in the examples that matter (Step 1
and Step 3) the first 31 colors are the gray scale colors
of the images. They appear to be identical in both color
tables. (I used my CINDEX program to view the color tables
after I loaded them.) Moreover, the resulting 2D images
only have values between 0 and 31, and *they* are
identical.

http://www.dfanning.com/programs/cindex.pro

Notice that your differences start *above* the values
that are really used in the images.

Although I can't really explain the differences that
don't matter, I do note that the algorithm is a
statistical method. To me this suggests some randomization
may be involved to get some kind of "seed" or something.
(I'm making this up, but I bet I'm right.)

Using a histogram equalization will certainly lead
to strange results, because the color tables returned
from COLOR_QUAN are *never* continuous in color. A pixel
value is assigned a particular color, but that color may
be COMPLETELY different from the pixel with an adjacent
value. There is no requirement, I don't think, that the
same value be assigned the same color in two different
instances. Only that the pixel values and the colors
fairly represent the colors in the 3D image.

> If I include the /MAP_ALL keyword with each call to COLOR_QUAN, the discrepancies
> disappear. However, the documentation indicates that /MAP_ALL should be used
> only if /GET_TRANSLATION is also present (which I don't think I need.)
>
> Should I expect to see the differences above, and is it safe to use the /MAP_ALL
> keyword to eliminate those differences?

I really think the differences are a non-issue. Forget about
the MAP_ALL keyword and cheerfully use COLOR_QUAN. :-)

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: COLOR_QUAN question
Posted by steinhh on Thu, 19 Aug 1999 07:00:00 GMT
View Forum Message <> Reply to Message

In article <37BC1A07.E6E4C022@ssec.wisc.edu>
Liam Gumley <Liam.Gumley@ssec.wisc.edu> writes:

> Have you looked at the ImageMagick API?
> http://www.wizards.dupont.com/cristy/www/api.html
>
> Seems like it could be interfaced to IDL by some enterprising
> individual.

Sure, no problem, I think. Other than how to find the Time...

Speaking of time, I just wrote a DLM wrapper for the one of the
functions in the FFTW library (Fastest Fourier Transform
in the West see http://www.fftw.org), to do real->complex and
complex->real transforms of some *large* datasets. Beats the
native IDL FFT by factors of about 6 - 15 on my platform!
Of course it only uses half the space, as well.... Ideal for
convolving fltarrs with other fltarrs!

In the spirit of sharing etc... here follows the DLM file and the
c code to be linked with the fftw libraries. This one only deals
with single precision transforms, mind you... Although the library
can be configured to produce both double and single precision
code, it's a bit difficult to mix transforms for different types
in one executable...

And no, I won't have time to answer questions about how to
get this working on your computer... But as a hint, my link

statement includes "-lidl -lsrfftw -lsfftw -lc -lm".

Use: To convolve a=fltarr(m,n,...) with b=fltarr(m,n,...), do
this:

```
    result = rfftwnd(rfftwnd(a)*rfftwnd(b),(size(b))(1))
```

Bye for now.

Stein Vidar

rfftwnd.dlm---------- ---

```
# $Id: rfftwnd.dlm,v 1.1 1999/08/16 10:59:04 steinhh Exp steinhh $
MODULE RFFTWND
DESCRIPTION N-dimensional Real fftw
VERSION $Revision: 1.1 $
BUILD_DATE $Date: 1999/08/16 10:59:04 $
SOURCE S.V.H.HAUGAN
FUNCTION RFFTWND    1 2
```

---

rfftwnd.c---------- ---

```
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "export.h"
#include "srfftw.h"

#define NULL_VPTR ((IDL_VPTR) NULL)

#define GETVARADDR(v) ((v->flags&IDL_V_ARR) ? (void*)v->value.arr->data \
    : (void*) &v->value.c)

#define GETVARDATA(v,n) ((v->flags&IDL_V_ARR) \
    ? (*(n) = v->value.arr->n_elts, v->value.arr->data) \
    : (*(n) = 1, & v->value.c ) )


FILE *rfftw_wisdom_file(char *mode)
{
  char *name;
  FILE *f;

  name=getenv("IDL_FFTW_WISDOM");

  if (name==(char*)NULL) {
    name = calloc(1024,sizeof(char));
```

```
    if (name==(char*)NULL) return (FILE*) NULL;

    strncpy(name,getenv("HOME"),1023);
    strncat(name,"/.idl_rfftw_wisdom.",1024-strlen(name));
    gethostname(name+strlen(name),1024-strlen(name));
  }
  f=fopen(name,mode);
  free(name);
  return f;
}

static char *wisdom=(char*)NULL;

void rfftw_init(void)
{
  FILE *f;

  static int done=0;

  if (done) return;

  done=1;
  f = rfftw_wisdom_file("r");
  if (f==(FILE*)NULL) {
    IDL_Message(IDL_M_NAMED_GENERIC,IDL_MSG_INFO,"Can't read wisdom file.");
    return;
  }
  if (fftw_import_wisdom_from_file(f)!=FFTW_SUCCESS) {
    IDL_Message(IDL_M_NAMED_GENERIC,IDL_MSG_INFO,"Bad wisdom data?");
  }
  fclose(f);

  wisdom=fftw_export_wisdom_to_string();
  IDL_Message(IDL_M_NAMED_GENERIC,IDL_MSG_INFO,"Imported wisdom from file.");
}


void rfftw_finish(void)
{
  FILE *f;
  static char *new_wisdom;

  new_wisdom=fftw_export_wisdom_to_string();
  if (new_wisdom==(char*)NULL) return;

  if (wisdom!=(char*)NULL && !strcmp(wisdom,new_wisdom)) {
    fftw_free(new_wisdom);
    return;
```

```
  }

  if (wisdom!=(char*)NULL) fftw_free(wisdom);

  wisdom=new_wisdom;

  f = rfftw_wisdom_file("w");
  if (f==(FILE*)NULL) {
    IDL_Message(IDL_M_NAMED_GENERIC,IDL_MSG_INFO,"Couldn't write wisdom file");
    return;
  }
  fftw_export_wisdom_to_file(f);
  fclose(f);
  IDL_Message(IDL_M_NAMED_GENERIC,IDL_MSG_INFO,"Wrote wisdom to file");
}

/* Forward is REAL -> COMPLEX, use input dimensions for plan */
void rfftwnd_forward(IDL_VPTR in, IDL_VPTR out)
{
  fftw_real *src;
  fftw_complex *Fsrc;
  rfftwnd_plan plan;

  rfftw_init();

  src = (fftw_real*) in->value.arr->data;
  Fsrc = (fftw_complex*) out->value.arr->data;

  plan=rfftwnd_create_plan(in->value.arr->n_dim,in->value.arr- >dim,
      FFTW_REAL_TO_COMPLEX,FFTW_MEASURE|FFTW_USE_WISDOM);

  rfftwnd_one_real_to_complex(plan,src,Fsrc);
  rfftwnd_destroy_plan(plan);
  rfftw_finish();
}


/* Backward is COMPLEX -> REAL, use OUTPUT dimensions for plan! */
void rfftwnd_backward(IDL_VPTR in, IDL_VPTR out)
{
  fftw_complex *src;
  fftw_real *Fsrc;
  rfftwnd_plan plan;

  rfftw_init();

  src = (fftw_complex*) in->value.arr->data;
  Fsrc = (fftw_real*) out->value.arr->data;
```

```
  plan=rfftwnd_create_plan(out->value.arr->n_dim,out->value.arr- >dim,
     FFTW_COMPLEX_TO_REAL,FFTW_MEASURE|FFTW_USE_WISDOM);

  rfftwnd_one_complex_to_real(plan,src,Fsrc);
  rfftwnd_destroy_plan(plan);
  rfftw_finish();
}


IDL_VPTR RFFTWND(int argc, IDL_VPTR argv[])
{
  int i=0; /* Note it's use in indexing argv[i++]                  */
           /* This simplifies taking away extra arguments,         */
           /* typically those specifying the number of elements    */
           /* in input arrays (available as var->value.arr->n_elts) */

  IDL_VPTR a=argv[i++];    /* Input array */
  IDL_VPTR odim=argv[i++], /* Output leading dim. for backward transform */

    call[1], /* For use when calling conversion routines */
    tmp;

  IDL_MEMINT dim[IDL_MAX_ARRAY_DIM], tmpi;
  int ndim,dir;

  char odimreq[]=
    "2nd arg (leading output dimension) required for backward transform";
  char odimwrong[]=
    "2nd arg (leading output dimension) has an inappropriate value";

  /* TYPE CHECKING / ALLOCATION SECTION */

  IDL_EXCLUDE_STRING(a);
  IDL_ENSURE_ARRAY(a);
  IDL_ENSURE_SIMPLE(a);

  ndim = a->value.arr->n_dim; /* Shorthand */

  /* If we're doing a forward transform, convert to Float,              */
  /* if baclwards, convert to Complex, and make sure we have the leading */
  /* dimension size, and convert it to long                             */

  call[0] = a;
  if (a->type!=IDL_TYP_COMPLEX && a->type!=IDL_TYP_DCOMPLEX) {

    dir = -1; /* We're doing a forward transform */
    a = IDL_CvtFlt(1,call); /* May cause a to be tmp */
```

```
} else {

  /* Require 2 arguments */
  if (argc!=2) IDL_Message(IDL_M_NAMED_GENERIC,IDL_MSG_LONGJMP,odimreq);

  dir = 1;    /* Backward transform */
  a = IDL_CvtComplex(1,call); /* May cause a to be tmp */

  IDL_EXCLUDE_UNDEF(odim); /* Output leading dimension */
  IDL_ENSURE_SIMPLE(odim);
  IDL_EXCLUDE_STRING(odim);
  IDL_ENSURE_SCALAR(odim);

  call[0] = odim;
  odim = IDL_CvtLng(1,call);

  /* Check validity of the output leading dimension */
  tmpi = 2*a->value.arr->dim[0] - odim->value.l;
  if (tmpi != 1 && tmpi != 2)
    IDL_Message(IDL_M_NAMED_GENERIC,IDL_MSG_LONGJMP,odimwrong);

  /* Give warning about overwritten input arg. */
  if (ndim>1 && !(a->flags&IDL_V_TEMP))
    IDL_Message(IDL_M_NAMED_GENERIC,IDL_MSG_INFO,"Input overwritten!");
}

/* Swap dimensions to row major */
for (i=0; i<ndim/2; i++) {
  tmpi=a->value.arr->dim[i];
  a->value.arr->dim[i] = a->value.arr->dim[ndim-i-1];
  a->value.arr->dim[ndim-i-1] = tmpi;
}

/* Copy dimensions */
for (i=0; i<a->value.arr->n_dim; i++) dim[i] = a->value.arr->dim[i];

/* Correct dimensions - for allocation */
if (dir==-1) dim[ndim-1] = 2*(dim[ndim-1]/2+1);
else         dim[ndim-1] = 2*dim[ndim-1];  /* Complex takes 2x FLOAT */

/* Storage for result */
IDL_MakeTempArray(IDL_TYP_FLOAT,a->value.arr->n_dim,dim,
    IDL_ARR_INI_INDEX,&tmp);

/* Correct output leading dimension - to make correct plan */
if (dir==1) {
  tmp->value.arr->n_elts /= tmp->value.arr->dim[ndim-1];
```

```
    tmp->value.arr->dim[ndim-1] = odim->value.l;
    tmp->value.arr->n_elts *= tmp->value.arr->dim[ndim-1];
  }

  if (dir==FFTW_REAL_TO_COMPLEX) rfftwnd_forward(a,tmp);
  else                           rfftwnd_backward(a,tmp);

  /* Swap dimensions back! */
  for (i=0; i<ndim/2; i++) {
    tmpi=a->value.arr->dim[i];
    a->value.arr->dim[i] = a->value.arr->dim[ndim-i-1];
    a->value.arr->dim[ndim-i-1] = tmpi;
    tmpi=tmp->value.arr->dim[i];
    tmp->value.arr->dim[i] = tmp->value.arr->dim[ndim-i-1];
    tmp->value.arr->dim[ndim-i-1] = tmpi;
  }

  i=0;

  if (a!=argv[i++]) IDL_DELTMP(a);
  if (odim!=argv[i++]) IDL_DELTMP(odim);

  if (dir==-1) {
    tmp->type = IDL_TYP_COMPLEX;  /* Change type, halve the size */
    tmp->value.arr->dim[0] /= 2;
    tmp->value.arr->n_elts /= 2;
  } else {
  }
  return tmp;
}

int IDL_Load(void)
{
  static IDL_SYSFUN_DEF func_def[] = {
    {(IDL_FUN_RET) RFFTWND,"RFFTWND",1,2}
    };
  return IDL_AddSystemRoutine(func_def,TRUE,1);
}
```

Subject: Re: COLOR_QUAN question
Posted by Liam Gumley on Thu, 19 Aug 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Struan Gray wrote:
> Seriously though, I once asked RSI if they had considered writing
> a routine to use Photoshop plug-ins. There are some good 3rd party
> processing and masking tools that it would be very nice to be able to

> use in IDL, as well file format translators and image capture hardware
> like scanners and cameras. It would also be a good and relatively
> safe way of interfacing to user-written compiled code. No joy, but
> perhaps now they are becoming more PC-oriented it's time to try again.

Have you looked at the ImageMagick API?
http://www.wizards.dupont.com/cristy/www/api.html

Seems like it could be interfaced to IDL by some enterprising
individual.

Cheers,
Liam.

--
Liam E. Gumley
Space Science and Engineering Center, UW-Madison
http://cimss.ssec.wisc.edu/~gumley

Subject: Re: COLOR_QUAN question
Posted by Struan Gray on Thu, 19 Aug 1999 07:00:00 GMT
View Forum Message <> Reply to Message

David Fanning, davidf@dfanning.com writes:

>> My wish list for IDL includes making it do everything
>> Photoshop, Illustrator, Visual Basic, and MatLab can
>> do. :-)
>
> Oh, did I mention Word and Powerpoint? Those too.

Don't forget that Lego robotics software.

Seriously though, I once asked RSI if they had considered writing a routine to use Photoshop plug-ins. There are some good 3rd party processing and masking tools that it would be very nice to be able to use in IDL, as well file format translators and image capture hardware like scanners and cameras. It would also be a good and relatively safe way of interfacing to user-written compiled code. No joy, but perhaps now they are becoming more PC-oriented it's time to try again.

Struan

Subject: Re: COLOR_QUAN question

Posted by Daniel Peduzzi on Thu, 19 Aug 1999 07:00:00 GMT
View Forum Message <> Reply to Message

David Fanning wrote in message ...
> Daniel Peduzzi (peduzzi@mediaone.net) writes:
>
>> My question concerns the R, G, and B arrays returned by the COLOR_QUAN
>> function. I've noticed that I don't receive the same RGB values if I call the function
>> multiple times with the same input arguments. This isn't very noticeable upon visual
>> inspection of the resulting images, unless the differences are exaggerated by color map
>> operations such as histogram equalization.
>
> Oh, oh. Hold on here. I think we may be fooling ourselves
> a bit. First of all, in the examples that matter (Step 1
> and Step 3) the first 31 colors are the gray scale colors
> of the images. They appear to be identical in both color
> tables. (I used my CINDEX program to view the color tables
> after I loaded them.) Moreover, the resulting 2D images
> only have values between 0 and 31, and *they* are
> identical.
>

Thanks...everything you said made perfect sense. However, I'm still a bit confused about the purpose of the COLORS keyword. In STEP 1-3 of my example, I specified COLORS=256 in each call to COLOR_QUAN. As you pointed out, the digital values of each resulting image lie in the range [0,31]. If the COLORS keyword doesn't have any influence on the values in the image returned by COLOR_QUAN, then what is it used for?

I guess I expected to see 256 entries in the output palette USED by the returned image.

Dan Peduzzi
peduzzi@mediaone.net

Subject: Re: COLOR_QUAN question
Posted by davidf on Thu, 19 Aug 1999 07:00:00 GMT
View Forum Message <> Reply to Message

> My wish list for IDL includes making it do everything
> Photoshop, Illustrator, Visual Basic, and MatLab can
> do. :-)

Oh, did I mention Word and Powerpoint? Those too.

--

David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: COLOR_QUAN question
Posted by davidf on Thu, 19 Aug 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Struan Gray (struan.gray@sljus.lu.se) writes:

> My optimal solution? Use Photoshop. :-)

My wish list for IDL includes making it do everything Photoshop, Illustrator, Visual Basic, and MatLab can do. :-)

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: COLOR_QUAN question
Posted by Struan Gray on Thu, 19 Aug 1999 07:00:00 GMT
View Forum Message <> Reply to Message

David Fanning, davidf@dfanning.com writes:

> But, again, I'm not sure this is a big deal. Color_Quan
> is used to produce images that *LOOK* the same. That is
> to say, it is a *display* routine, not a *processing*
> routine.

It's not exactly a blight on my life, but I wanted to save GIFs in web-specific colour tables (those with colours that are 'safe' for a wide variety of displays). If COLOR_QUAN had returned the 'correct' table, shuffling the colour indicies to get back the ordering of the web-specific table would have been a one-liner. As it is, you have to

step through the table looking for closest matches, and if you're
going to do that you might as well do it from the start.

My optimal solution? Use Photoshop. :-)

Struan

Subject: Re: COLOR_QUAN question
Posted by davidf on Thu, 19 Aug 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Struan Gray (struan.gray@sljus.lu.se) writes:

> A while back someone asked for a way to force COLOR_QUAN to use a
> particular color table. I tried seeding it with a dummy image
> containing 256 pixels and a simple ramp through the desired table,
> hoping to use /MAP_ALL and /GET_TRANSLATION to force subsequent calls
> to use the same colour map.
>
> I gave up because the colour map returned by the first call was
> always different from that used to generate the dummy image. That is,
> even if you feed COLOR_QUAN an RGB image with only 256 pixels (which
> it should be able to reproduce exactly with an 8-bit colour table) the
> RGB values of the colour map it generates are different from the
> pixels' RGB values.

I just tried a similar experiment. The image has 256 values
(a 3D image constructed of a 2D image created like this:
image2D = BytScl(Findgen(256)#Findgen(256))).

Oddly enough, the resulting 2D image has only 155
values. And even though the returned color tables
differ in the pixel values above 154, they appear
identical in those values from 0-154. The
images *appear* identical on the display, though,
of course, they can't be.

But, again, I'm not sure this is a big deal. Color_Quan
is used to produce images that *LOOK* the same. That is
to say, it is a *display* routine, not a *processing*
routine. Performing image processing steps on the
resulting image, or using the pixel values to mean
something, is just as big a mistake as using the
byte values of any image on the display, rather than
the original image data.

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: COLOR_QUAN question
Posted by Struan Gray on Thu, 19 Aug 1999 07:00:00 GMT
View Forum Message <> Reply to Message

David Fanning, davidf@dfanning.com writes:

> There is no requirement, I don't think, that the
> same value be assigned the same color in two different
> instances. Only that the pixel values and the colors
> fairly represent the colors in the 3D image.

A while back someone asked for a way to force COLOR_QUAN to use a particular color table. I tried seeding it with a dummy image containing 256 pixels and a simple ramp through the desired table, hoping to use /MAP_ALL and /GET_TRANSLATION to force subsequent calls to use the same colour map.

I gave up because the colour map returned by the first call was always different from that used to generate the dummy image. That is, even if you feed COLOR_QUAN an RGB image with only 256 pixels (which it should be able to reproduce exactly with an 8-bit colour table) the RGB values of the colour map it generates are different from the pixels' RGB values.

Struan
