Subject: thresholding/color question
Posted by Patty Howell on Wed, 02 Feb 2000 08:00:00 GMT
View Forum Message <> Reply to Message

I want to display all values of an image with
amplitudes greater than a given threshold limit in
red, while leaving the rest of the image in
greyscale. I know there has to be a simple way to
do this, but I sure can't find it.

Subject: Re: thresholding/color question
Posted by davidf on Sat, 05 Feb 2000 08:00:00 GMT
View Forum Message <> Reply to Message

Patty Howell (p.a.howell@larc.nasa.gov) writes:

> I want to display all values of an image with
> amplitudes greater than a given threshold limit in
> red, while leaving the rest of the image in
> greyscale. I know there has to be a simple way to
> do this, but I sure can't find it.

I'd set up the colors like this:

```
LoadCT, 0, NColors=!D.Table_Size-3 ; Gray-Scale
TVLCT, 255, 0, 0, !D.Table_Size-2  ; Red pixel.
```

Then, I'd scale my image data like this:

```
scaledImage = BytScl(image, Top=!D.Table_Size-4)
```

I'd find the red pixels and assign them like this:

```
redPixels = Where(image GT threshold, count)
IF count GT 0 THEN scaledImage[redPixels] = !D.Table_Size-2
```

Display the data like this:

```
TV, scaledImage
```

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com

Subject: Re: thresholding/color question
Posted by davidf on Mon, 07 Feb 2000 08:00:00 GMT
View Forum Message <> Reply to Message

Alex Schuster (alex@pet.mpin-koeln.mpg.de) writes:

> Isn"t the usage of WHERE discouraged because of being slow?

If it is, it's news to me. :-)

> I always use direct matrix operation for this, like:
>
> redmask = scaledImage GT threshold
> scaledImage = scaled_Image * (1B-redmask) + byte(!D.Table_Size-2) *
> redmask

I don't doubt there are speed advantages in matrix operations,
but unless you had a huge image I doubt very much you
would notice. And even then, the increase in speed would probably
be more than offset by the additional memory usage.

Plus, I don't have a clue what this code is doing when I
read it! Simple things for simple folk, is my motto. :-)

Cheers,

David

P.S. If my memory serves, Alex, aren't you the one with
the 10 quadrillion Megs of RAM on your machine? Perhaps
matrix operations *are* the best thing for you. :-)
--
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: thresholding/color question
Posted by Alex Schuster on Mon, 07 Feb 2000 08:00:00 GMT
View Forum Message <> Reply to Message

David Fanning wrote:

> Then, I'd scale my image data like this:
>
> scaledImage = BytScl(image, Top=!D.Table_Size-4)
>
> I'd find the red pixels and assign them like this:
>
> redPixels = Where(image GT threshold, count)
> IF count GT 0 THEN scaledImage[redPixels] = !D.Table_Size-2

Isn"t the usage of WHERE discouraged because of being slow? I always use
direct matrix operation for this, like:

redmask = scaledImage GT threshold
scaledImage = scaled_Image * (1B-redmask) + byte(!D.Table_Size-2) *
redmask

Alex
--
Alex Schuster Wonko@weird.cologne.de PGP Key available
alex@pet.mpin-koeln.mpg.de

Subject: Re: thresholding/color question
Posted by Craig Markwardt on Mon, 07 Feb 2000 08:00:00 GMT
View Forum Message <> Reply to Message

Alex Schuster <alex@pet.mpin-koeln.mpg.de> writes:
> David Fanning wrote:
>
>> Then, I'd scale my image data like this:
>>
>> scaledImage = BytScl(image, Top=!D.Table_Size-4)
>>
>> I'd find the red pixels and assign them like this:
>>
>> redPixels = Where(image GT threshold, count)
>> IF count GT 0 THEN scaledImage[redPixels] = !D.Table_Size-2
>
> Isn"t the usage of WHERE discouraged because of being slow? I always use
> direct matrix operation for this, like:
>
> redmask = scaledImage GT threshold
> scaledImage = scaled_Image * (1B-redmask) + byte(!D.Table_Size-2) * redmask

I'm not sure the masking technique will always be faster. If you do
an operation count, then you can see that the masking operation costs

4*N operations, where N is the number of pixels. (operations are "GT",
"*" (twice) and "-" (once))

The WHERE technique involves somewhere between N and 2*N operations,
depending on how many pixels are above threshold (one to do the "GT"
operation, and then between 0 and N to do the scatter operation on
scaledImage). However, I will grant that the scatter operation is
probably slower than a flat-out matrix multiply.

Therefore, I would argue that for large images with few red pixels,
the WHERE operation would be superior (and as David says, use less
memory). Anybody care to bring this into the real world?

Craig

--
---------- ----------
Craig B. Markwardt, Ph.D. EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
---------- ----------

Subject: Re: thresholding/color question
Posted by Alex Schuster on Mon, 07 Feb 2000 08:00:00 GMT
View Forum Message <> Reply to Message

David Fanning wrote:

> Alex Schuster (alex@pet.mpin-koeln.mpg.de) writes:
>
>> Isn"t the usage of WHERE discouraged because of being slow?
>
> If it is, it's news to me. :-)

Hey! I think I read this here in this newsgroup.

>> I always use direct matrix operation for this, like:
>>
>> redmask = scaledImage GT threshold
>> scaledImage = scaled_Image * (1B-redmask) + byte(!D.Table_Size-2) *
>> redmask
>
> I don't doubt there are speed advantages in matrix operations,
> but unless you had a huge image I doubt very much you
> would notice. And even then, the increase in speed would probably
> be more than offset by the additional memory usage.

Okay, I just tried it out. Indeed, not much of a difference, both

algorithms take about the same time. It depends on how big your index list will get. Here's my test program:

```
pro test, dim

 b = bindgen( dim )
 starttime = systime( 1 )
 index = where( b ge 200B )
 if ( index[0] ne -1L ) then b[index] = 255B
 print, 'Time using index list:', systime( 1 ) - starttime,$
       format='(A,D8.4)'

 b = bindgen( dim )
 starttime = systime( 1 )
 mask = b ge 100B
 b = temporary( b ) * (1B-mask) + mask * 255B
 print, 'Time using mask array:', systime(1) - starttime,$
       format='(A,D8.4)'

end
```

An array of dimension 10,000,000 takes about two seconds to compute, with the index method being slightly faster here. When lowering the threshold, my method becomes faster.

> Plus, I don't have a clue what this code is doing when I
> read it! Simple things for simple folk, is my motto. :-)

Well, believe me or not, I think my approach _is_ simple :) Finding out all those red values, putting them together into a list and then assigning a new value to each pixel in this list, wow, that's a lot of work. So I just multiply some arrays :)
Okay, I must admit, it also makes the source code look _cooler_.

> P.S. If my memory serves, Alex, aren't you the one with
> the 10 quadrillion Megs of RAM on your machine? Perhaps
> matrix operations *are* the best thing for you. :-)

Not really, I must admit. Just tested it with a size of 100,000,000, and the mask method took 300 seconds, compared to 40 using the index list. Hm, strange. This was a machione with 400 MB memory. On an UltraSparc with 512 MB (not 10 quadrillion megs), it takes 32 / 37 seconds. Looks like a constant ratio, unless swapping begins.

test, 1000000000000LLL is still running, can't say anything about that yet... I will post the results to this newsgroup, if there still is such a thing as newsgroups when it's finished.

Alex

--

Alex Schuster Wonko@weird.cologne.de PGP Key available
alex@pet.mpin-koeln.mpg.de