Subject: Re: accessing large arrays quickly
Posted by davidf on Thu, 27 May 1999 07:00:00 GMT
View Forum Message <> Reply to Message

David Mattes (dmattes@u.washington.edu) writes:

> what would be nice though, is a routine similar to tvscl, but with all the
> flexibility of bytscl, and with a couple extra features. tvscl only
> allows specifying a top value. it would be nice to specify a bottom
> value, to make working with split color tables a lot easier. does anybody
> know of such a procedure???

Well, because you asked and because I didn't feel like doing anything
else today, here is a TVSCALE program that has the flexibility of BYTSCL
and a "couple of extra features".

You can use it like this:

```
TVScale, image, TOP=200, BOTTOM=100
```

and the image data will be scaled between 100 and 200. Note that
this is 101 pixel values, however. So your color table should be
loaded like this:

```
LoadCT, 5, NColors=101, Bottom=100
```

As for additional features. Here are a few:

1. It will erase the display if required.

   ```
   TVScale, image, /Erase
   ```

2. The image can be positioned in the window with the POSITION keyword.

   ```
   TVScale, image, Position=[0.1, 0.2, 0.9, 0.75]
   ```

3. The program is device independent. In other words, it works
   exactly the same way in the PostScript device as it does on the
   display. You don't have to size images two different ways depending
   upon which device you are in.

4. It works transparently with 8-bot or 24-bit images. You don't have
   to worry about setting the appropriate TRUE keyword, nor do you
   have to worry about whether you are using DECOMPOSED color or not.
   The program will set these keywords appropriately for the image.
   (Well, it will if you are using IDL 5.2, which has the ability to
   get and set these keywords.)

5. It will work with !P.MULTI.

   !P.Multi = [0, 2, 1]
   Plot, Histogram(image)
   TVScale, image, /Multi

6. By default it fills up the entire current graphics window.

7. If you prefer, you can choose to keep and preserve the aspect
   ratio of the image:

   TVScale, image, /Keep_Aspect

You can find the program here:

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

As some of you will realize, it's companion program is TVImage,
the one I prefer:

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

> also, if anyone can answer this question, you get serious kudos!

Oh, I *like* serious kudos! :-)

> i am
> defining several objects and structures in my programs. if i add a tag to
> a structure once this structure has existed in memory, i must restart IDL
> and recompile, in order for my changes to be accepted. i always perform
> garbage collection on all the heap variables, using heap_gc. how can i
> get IDL to forget about all previous structure definitions???

Exit IDL!!! (This was too easy.)

Uh, seriously. This is how this is *suppose* to work. If you
are changing your structure definitions all the time, you should
be using anonymous rather than named structures. Can you imagine
the confusion that might ensue if IDL allowed you to change
COMMON block or named structure definitions whenever you "felt
like it"?

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting

Subject: Re: accessing large arrays quickly
Posted by D. Mattes on Thu, 27 May 1999 07:00:00 GMT
View Forum Message <> Reply to Message

well, this question about accessing large arrays has proven a rather embarrasing entry to this prestigious newsgroup, as i have discovered the real root of my problem. i was using the bytscl operation on each slice before drawing it to the window, thus slowing things down quite a bit. the performance hit was only noticeable once the array size grew larger than say 256x256x30, so i felt sure something else was culprit. but now, if i spend the time to bytscl the entire volume first, accessing the slices isn't putting any crimp in my style. thanks to those who took the time to respond!

what would be nice though, is a routine similar to tvscl, but with all the flexibility of bytscl, and with a couple extra features. tvscl only allows specifying a top value. it would be nice to specify a bottom value, to make working with split color tables a lot easier. does anybody know of such a procedure???

also, if anyone can answer this question, you get serious kudos! i am defining several objects and structures in my programs. if i add a tag to a structure once this structure has existed in memory, i must restart IDL and recompile, in order for my changes to be accepted. i always perform garbage collection on all the heap variables, using heap_gc. how can i get IDL to forget about all previous structure definitions???

thanks!
david mattes

On Thu, 27 May 1999, D. Mattes wrote:

> hello idl gurus: i have a very large volume array out of which i extract
> 3 orthogonal 2-d slices and display these slices in three separate
> windows. i extract a slice by assignment:
>
> slice=data(*,*,zslice)
>
> then i scale the slice, and finally display it using tv. once the volume
> grows to larger than 10Meg, i suffer a performance hit on the array access
> times, and my image browser slows down considerably. how can i improve
> performance???
>

> some ideas i've had:
> 1. render the entire volume and specify cutting planes to just display
> the slice of interest.
> 2. use an external c function, like memcpy, to speed up the variable
> swapping when i assign 2-d array as a crosssection of the volume array.
> 3. store each possible slice separately, perhaps in a linked list.
>
> do you idl gurus out there have any suggestions or comments on my ideas???
>
> thank you in advance for your time.
>
> david mattes
>
>
>
>

Subject: Re: accessing large arrays quickly
Posted by David Foster on Thu, 27 May 1999 07:00:00 GMT
View Forum Message <> Reply to Message

D. Mattes wrote:
>
> hello idl gurus: i have a very large volume array out of which i extract
> 3 orthogonal 2-d slices and display these slices in three separate
> windows. i extract a slice by assignment:
>
> slice=data(*,*,zslice)
>
> then i scale the slice, and finally display it using tv. once the volume
> grows to larger than 10Meg, i suffer a performance hit on the array access
> times, and my image browser slows down considerably. how can i improve
> performance???

David -

I think the single most effective solution would be to add memory
to your system. What platform are you using? If things slow down
considerably as the array grows, then you are being limited by
the amount of the array that can be stored in memory simultaneously.

We regularly do this type of array access, for computing orthogonal
slices, with integer arrays 256x256x124, or 16.2MB in size, and
we find this to be pretty fast. We're using Sun Sparc systems with >96MB
memory.

>

> some ideas i've had:
> 1. render the entire volume and specify cutting planes to just displa
> the slice of interest.

I wouldn't expect this to be faster than directly accessing the array.

> 2. use an external c function, like memcpy, to speed up the variable
> swapping when i assign 2-d array as a crosssection of the volume array.

I don't think this is your limitation.

> 3. store each possible slice separately, perhaps in a linked list.

Each slice along all three axes? That's a lot of slices! You would
run into the same memory limitations as before.

>
> do you idl gurus out there have any suggestions or comments on my ideas???
>
> thank you in advance for your time.
>
> david mattes

You might try playing around with the order in which you access
the three planes (XY, YZ, XZ), to try and optimize how you are
accessing the memory (XY will be least demanding).

Dave Foster
--

~~~~~~ ~~~~~~
David S. Foster Univ. of California, San Diego
Programmer/Analyst Brain Image Analysis Laboratory
foster@bial1.ucsd.edu Department of Psychiatry
(619) 622-5892 8950 Via La Jolla Drive, Suite 2240
La Jolla, CA 92037
~~~~~~ ~~~~~~

Subject: Re: accessing large arrays quickly
Posted by davidf on Thu, 27 May 1999 07:00:00 GMT
View Forum Message <> Reply to Message

David Mattes (dmattes@u.washington.edu) writes:

> hello idl gurus: i have a very large volume array out of which i extract
> 3 orthogonal 2-d slices and display these slices in three separate
> windows. i extract a slice by assignment:

>
> slice=data(*,*,zslice)
>
> then i scale the slice, and finally display it using tv. once the volume
> grows to larger than 10Meg, i suffer a performance hit on the array access
> times, and my image browser slows down considerably. how can i improve
> performance???
>
> some ideas i've had:
> 1. render the entire volume and specify cutting planes to just display
> the slice of interest.

Yeah. I'd get one of those machines that have a GByte of RAM. That
should help. :-)

> 2. use an external c function, like memcpy, to speed up the variable
> swapping when i assign 2-d array as a crosssection of the volume array.
> 3. store each possible slice separately, perhaps in a linked list.
>
> do you idl gurus out there have any suggestions or comments on my ideas???

If this was something I was doing often, I might think about
having three versions of the array on disk, stored so that the
plane of data I was interested in was contiguous in memory.
(I hesitate to mention this, but data is stored in row order
in memory.) Then you could easily use an associated
variable method to quickly access the plane of interest.

I have a feeling the ENVI guys do something like this when
they work with large data sets.

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: accessing large arrays quickly
Posted by David Foster on Fri, 28 May 1999 07:00:00 GMT
View Forum Message <> Reply to Message

D. Mattes wrote:

>
> well, this question about accessing large arrays has proven a rather
> embarrasing entry to this prestigious newsgroup, as i have discovered the
> real root of my problem.

David - (seems this is the "David" thread)

Believe me, we have ALL been there!

> i was using the bytscl operation on each slice
> before drawing it to the window, thus slowing things down quite a bit.
> the performance hit was only noticeable once the array size grew larger
> than say 256x256x30, so i felt sure something else was culprit. but now,
> if i spend the time to bytscl the entire volume first, accessing the
> slices isn't putting any crimp in my style.

Plus now you are working with 8-bit data, not 16-bit (I'm assuming).
If the time it takes to bytscl your data increases as the volume
increases in size, it's probably because the system is starting to
page fault. If you're using UNIX you can use vmstat to see this.

>
> what would be nice though, is a routine similar to tvscl, but with all the
> flexibility of bytscl, and with a couple extra features. tvscl only
> allows specifying a top value. it would be nice to specify a bottom
> value, to make working with split color tables a lot easier. does anybody
> know of such a procedure???

Since it looks like you are using medical images, you might want to
check out some of the routines I've made available at:

ftp://bial8.ucsd.edu/pub/software/idl/share

Many of the routines are for medical image apps, including a grayscale
routine that allows you to easily define a split color table, or even
split the color table into thirds. Also, the SHOW_IMG program allows
you to easily view medical images of many formats.

Hope this is useful.

Dave Foster

>
> On Thu, 27 May 1999, D. Mattes wrote:
>
>> hello idl gurus: i have a very large volume array out of which i extract
>> 3 orthogonal 2-d slices and display these slices in three separate
>> windows. i extract a slice by assignment:

>>
>> slice=data(*,*,zslice)
>>
>> then i scale the slice, and finally display it using tv. once the volume
>> grows to larger than 10Meg, i suffer a performance hit on the array access
>> times, and my image browser slows down considerably. how can i improve
>> performance???
>>
>> some ideas i've had:
>> 1. render the entire volume and specify cutting planes to just display
>> the slice of interest.
>> 2. use an external c function, like memcpy, to speed up the variable
>> swapping when i assign 2-d array as a crosssection of the volume array.
>> 3. store each possible slice separately, perhaps in a linked list.
>>
>> do you idl gurus out there have any suggestions or comments on my ideas???
>>
>> thank you in advance for your time.
>>
>> david mattes
>>
>>
>>
>>

--

~~~~~~ ~~~~~~~
David S. Foster Univ. of California, San Diego
Programmer/Analyst Brain Image Analysis Laboratory
foster@bial1.ucsd.edu Department of Psychiatry
(619) 622-5892 8950 Via La Jolla Drive, Suite 2240
La Jolla, CA 92037
~~~~~~ ~~~~~~~