Subject: Re: Reading Multiple DICOM Files
Posted by David Fanning on Sun, 08 Mar 2009 23:39:18 GMT
View Forum Message <> Reply to Message

Jye writes:

> Im currently putting together an application that needs to read the
> image data from numerous DICOM files, currently 900 but this will
> possible go up to about 2000.
>
> ATM I have a simple FOR loop :(which steps through each file, makes
> an object using Robbies GDLffDICOM and then reads the images data into
> a matrix. The object is then destroyed and the loop repeated. This is
> horribly slow as you would all imagine and takes about 90sec to read
> all of the images.

This is faster than I would have expected. Why does it
surprise you?

> Has anyone come across a fast way of reading numerous DICOM files? The
> problem would have easily been solved if only OBJARR could be used as
> a normal array :(

In what way can OBJARR not be used as a normal array?

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Reading Multiple DICOM Files
Posted by Robbie on Sun, 08 Mar 2009 23:42:18 GMT
View Forum Message <> Reply to Message

I think you'll find that all the frames in a CT or PET series often
have the same length and same offset for every DICOM tag. Some DICOM
packages use this 'feature' to optimise reading sequential files.
Evidently, GDLffDICOM and IDLffDICOM does not.

Rather than have 1000's of objects, it might be safer to sequentially
read the data into a contiguous 3D array. You'll find that IDL has a
limit of the number of files which can be open at one time.

Cheers,

Robbie

Subject: Re: Reading Multiple DICOM Files
Posted by Jye on Mon, 09 Mar 2009 00:17:55 GMT
View Forum Message <> Reply to Message

Hi Robbie and David,

I am currently reading the data into an array and this is where it
slowing down. Reading all of the files into an OBJARR is relatively
faster but this OBJARR must then have each objects image data read
individually eg I have not been able to do this;

Image_Data = oArray -> GetValue(REFERENCE = image_data_reference, /
no_copy)

Below is what Im currently using.

Cheers
Jye

```
PRO  Example

folder = 'D:\Dynamic\' ;Folder containing all of the DICOM files
files = file_search(string(folder, '*.DCM'), COUNT=nfiles)

dicom_obj = obj_new('GDLffDICOM')
result = dicom_obj -> Read(files[0])

image_Acq_Time_reference = dicom_obj -> GetReference('0018'x,'1242'x)
image_reference = dicom_obj -> GetReference('7FE0'x,'0010'x)

obj_destroy, dicom_obj

for i = 0, nfiles-1 do begin

 Slice_Number = i
 dicom_obj = obj_new('GDLffDICOM')
 result = dicom_obj -> Read(files[i])

 image_Acq_Time = dicom_obj -> GetValue
(REFERENCE=image_Acq_Time_reference,/no_copy)
 image_Acq_Time = *(image_Acq_Time[0])/1000.
 if i eq 0 then Acq_Time = image_Acq_Time else Acq_Time = [Acq_Time,
image_Acq_Time]

 image_slice = dicom_obj -> GetValue(REFERENCE=image_reference,/
no_copy)
 image_slice = rotate(*(image_slice[0]), 7) / image_Acq_Time

 if i eq 0 then All_Images = image_slice else All_Images =
[[[All_Images]], [[image_slice]]]

 obj_destroy, dicom_obj

endfor

END
```

Subject: Re: Reading Multiple DICOM Files
Posted by David Fanning on Mon, 09 Mar 2009 00:21:17 GMT
View Forum Message <> Reply to Message

Jye writes:

```
>  I am currently reading the data into an array and this is where it
>  slowing down. Reading all of the files into an OBJARR is relatively
>  faster but this OBJARR must then have each objects image data read
>  individually eg I have not been able to do this;
>
>  Image_Data = oArray -> GetValue(REFERENCE = image_data_reference, /
>  no_copy)
```

Yeah, I'd say you have been doing too much JAVA programming. :-)

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Reading Multiple DICOM Files
Posted by Robbie on Mon, 09 Mar 2009 03:11:14 GMT

It's just a sign that IDL has some inadequacies when it comes to OO. I
suspect that iterating method calls over an OBJARR() is always going
to be slow in IDL. The 'IDL way' would be come up with an object
called IDLffDICOMSet, which represents a set of files. Methods in
IDLffDICOMSet could accept array arguments, one array element for each
file. IDLffDICOMSet could be optimised to take advantage of common
file offsets of DICOM tags.

Subject: Re: Reading Multiple DICOM Files
Posted by rtk on Mon, 09 Mar 2009 18:07:55 GMT

On Mar 8, 6:17 pm, Jye <jye.sm...@gmail.com> wrote:
>   if i eq 0 then All_Images = image_slice else All_Images =
> [[[All_Images]], [[image_slice]]]

Forget too much Java, which, I agree, is a bad thing. I'd say you
have been doing too much Matlab programming. The vast majority of
your problem lies with the line above. If I run this code on my box:

```
pro ttt0
   s = systime(1)
   for i=0,899 do begin
      im = fix(dist(256))
      if (i eq 0) then begin
         all = im
      endif else begin
         all = [[[all]], [[im]]]
      endelse
   endfor
   print, 'runtime = ', systime(1)-s
   help, all
end
```

It takes 93 seconds. However, if I run this code:

```
pro ttt2
   all = intarr(256,256,900)
   s = systime(1)
   for i=0,899 do begin
      im = fix(dist(256))
      all[*,*,i] = im
   endfor
   print, 'runtime = ', systime(1)-s
   help, all
```

end

it takes 1.6 seconds. The only difference is that the output array is
allocated once, instead of 900 times with 899 ever-increasing array
copies to boot. Since you know the number of file in advance,
allocate All_Images and just assign into it.

Ron

Subject: Re: Reading Multiple DICOM Files
Posted by Jye on Mon, 09 Mar 2009 22:53:55 GMT
View Forum Message <> Reply to Message

On Mar 10, 4:07 am, rtk <oneelkr...@hotmail.com> wrote:
> On Mar 8, 6:17 pm, Jye <jye.sm...@gmail.com> wrote:
>
>   I'd say you have been doing too much Matlab programming.

And we have a winner :P

Thanks for the solution, this has dropped the time from 63.4 down to
21.6 seconds... still not terribly quick but getting there.

Subject: Re: Reading Multiple DICOM Files
Posted by Foldy Lajos on Tue, 10 Mar 2009 11:56:45 GMT
View Forum Message <> Reply to Message

On Mon, 9 Mar 2009, rtk wrote:

```
>  On Mar 8, 6:17 pm, Jye <jye.sm...@gmail.com> wrote:
>>         if i eq 0 then All_Images = image_slice else All_Images =
>>  [[[All_Images]], [[image_slice]]]
>
>  Forget too much Java, which, I agree, is a bad thing.  I'd say you
>  have been doing too much Matlab programming.  The vast majority of
>  your problem lies with the line above.  If I run this code on my box:
>
>  pro ttt0
>     s = systime(1)
>     for i=0,899 do begin
>         im = fix(dist(256))
>         if (i eq 0) then begin
>             all = im
>         endif else begin
>             all = [[[all]], [[im]]]
>        endelse
>     endfor
>     print, 'runtime = ', systime(1)-s
>     help, all
>  end
>
>  It takes 93 seconds.  However, if I run this code:
>
>  pro ttt2
>     all = intarr(256,256,900)
>     s = systime(1)
>     for i=0,899 do begin
>        im = fix(dist(256))
>        all[*,*,i] = im
>     endfor
>     print, 'runtime = ', systime(1)-s
>     help, all
>  end
>
>  it takes 1.6 seconds.  The only difference is that the output array is
>  allocated once, instead of 900 times with 899 ever-increasing array
>  copies to boot.  Since you know the number of file in advance,
>  allocate All_Images and just assign into it.
>
>  Ron
```

You can get some more speedup by replacing 'all[*,*,i] = im' with
'all[0,0,i] = im':

```
; cut here, ttt2.pro
pro ttt2
    all = intarr(256,256,900)
    im = fix(dist(256))

    s = systime(1)
    for i=0,899 do begin
       all[*,*,i] = im
    endfor
    print, 'runtime = ', systime(1)-s

    s = systime(1)
    for i=0,899 do begin
       all[0,0,i] = im
    endfor
    print, 'runtime = ', systime(1)-s
end
; cut here
```

```
IDL> ttt2
% Compiled module: TTT2.
% Compiled module: DIST.
runtime =       0.20675302
runtime =      0.047688007
```

(This is not the bottleneck for the original problem, though.)

regards,
lajos

Subject: Re: Reading Multiple DICOM Files
Posted by Tibor47 on Mon, 23 Mar 2009 17:26:05 GMT

View Forum Message <> Reply to Message

```
On Mar 10, 7:56 am, FÖLDY Lajos <fo...@rmki.kfki.hu> wrote:
>  On Mon, 9 Mar 2009, rtk wrote:
>>  On Mar 8, 6:17 pm, Jye <jye.sm...@gmail.com> wrote:
>>>         if i eq 0 then All_Images = image_slice else All_Images =
>>>  [[[All_Images]], [[image_slice]]]
>
>>  Forget too much Java, which, I agree, is a bad thing.  I'd say you
>>  have been doing too much Matlab programming.  The vast majority of
>>  your problem lies with the line above.  If I run this code on my box:
>
>>  pro ttt0
>>     s = systime(1)
>>     for i=0,899 do begin
>>        im = fix(dist(256))
>>        if (i eq 0) then begin
>>           all = im
>>        endif else begin
>>           all = [[[all]], [[im]]]
>>        endelse
>>     endfor
>>     print, 'runtime = ', systime(1)-s
>>     help, all
>>  end
>
>>  It takes 93 seconds.  However, if I run this code:
>
>>  pro ttt2
>>     all = intarr(256,256,900)
>>     s = systime(1)
>>     for i=0,899 do begin
>>        im = fix(dist(256))
>>        all[*,*,i] = im
>>     endfor
>>     print, 'runtime = ', systime(1)-s
>>     help, all
>>  end
>
>>  it takes 1.6 seconds.  The only difference is that the output array is
>>  allocated once, instead of 900 times with 899 ever-increasing array
>>  copies to boot.  Since you know the number of file in advance,
>>  allocate All_Images and just assign into it.
>
>>  Ron
>
>  You can get some more speedup by replacing 'all[*,*,i] = im' with
>  'all[0,0,i] = im':
>
>  ; cut here, ttt2.pro
>  pro ttt2
>      all = intarr(256,256,900)
>      im = fix(dist(256))
>
>      s = systime(1)
>      for i=0,899 do begin
>          all[*,*,i] = im
>      endfor
>      print, 'runtime = ', systime(1)-s
>
>      s = systime(1)
>      for i=0,899 do begin
>          all[0,0,i] = im
>      endfor
>      print, 'runtime = ', systime(1)-s
>  end
>  ; cut here
>
>  IDL> ttt2
>  % Compiled module: TTT2.
>  % Compiled module: DIST.
>  runtime =       0.20675302
>  runtime =      0.047688007
>
>  (This is not the bottleneck for the original problem, though.)
>
>  regards,
>  lajos- Hide quoted text -
>
>  - Show quoted text -
```

Hey I noticed you're also rotating each slice individually before

adding it to the group. It may work faster if you rotate the full
volume in one step at the end.
Of course IDL doesn't have anything to do this. But you can find a
link to the code that does it here. http://www.dfanning.com/math_tips/rotvolume.html.

Try building your volume first then rotating the whole thing in one
step. May or may not save you some time.
