Subject: Re: Rapid "moving windows" access in IDL?
Posted by mchinand on Fri, 30 Jan 2004 04:33:19 GMT
View Forum Message <> Reply to Message

In article <BC3EF6CE.18DB4%greenberg@ucdavis.edu>,
Jonathan Greenberg <greenberg@ucdavis.edu> wrote:
> I'm trying to code a semivariance measure to analyze an image in IDL. Since
> the call requires extracting data from a matrix a certain distance away from
> the center pixel, is there a fast way of coding IDL to extract these
> locations, or is am I doomed to have a very slow algorithm. Basic gist:
>
> For a given pixel/matrix location, subtract off the value of the pixels
> surrounding that pixel:
>
> For pixel at A is at x,y
>   B at x-1,y
>   C at x+1,y
>
> Semivariance = ((A-B)^2 + (A-C)^2)/4
>
> Any suggestions?
>
> --j
>

I'm not sure if this will be faster, but one option is to create B and C
arrays using SHIFT:

B=shift(A,-1,0)
C=shift(B,1,0)

sv= ((A-B)^2 + (A-C)^2)/4

Obviously, this method is more memory intensive than looping through
A. The edge values probably won't be what you want either.

--Mike

--
Michael Chinander
m-chinander@uchicago.edu
Department of Radiology
University of Chicago

Subject: Re: Rapid "moving windows" access in IDL?
Posted by Craig Markwardt on Fri, 30 Jan 2004 06:01:42 GMT
View Forum Message <> Reply to Message

Jonathan Greenberg <greenberg@ucdavis.edu> writes:
> I'm trying to code a semivariance measure to analyze an image in IDL. Since
> the call requires extracting data from a matrix a certain distance away from
> the center pixel, is there a fast way of coding IDL to extract these
> locations, or is am I doomed to have a very slow algorithm. Basic gist:
>
> For a given pixel/matrix location, subtract off the value of the pixels
> surrounding that pixel:
>
> For pixel at A is at x,y
> B at x-1,y
> C at x+1,y
>
> Semivariance = ((A-B)^2 + (A-C)^2)/4
>
> Any suggestions?

Yes, how about

```
N    = n_elements(A(*,0))  ;; Number of pixels per row
B    = A(0:N-3,*)   ; "left" pixel
Amid = A(1:N-2,*)   ; "middle" pixels
C    = A(2:N-1,*)   ; "right pixels

semivariance = ((Amid-B)^2 + (Amid-C)^2)/4
```

Of course this array will have an row size of two less than the original. People who know me, also know that you can avoid some of the steps above by using a "trick" where IDL will automatically truncate arrays if they are too big, which means you can write the whole thing as:

```
Amid = A(1:*,*)
semivariance = ((Amid-A)^2 + (Amid-A(2:*,*))^2)/4
```

which is more cryptic, but it saves me from the tragic pain of figuring out what "N" is.

Happy semi-variancing,
Craig

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

---------- ----------

Subject: Re: Rapid "moving windows" access in IDL?
Posted by Jonathan Greenberg on Fri, 30 Jan 2004 11:15:53 GMT
View Forum Message <> Reply to Message

Ok, now to make this a bit more confusing -- in practice, what I will actually be doing is selecting all pixels at about a given distance from the center point, creating a ring of pixels that will be used for the semivariogram. Getting back to an earlier question, is the array subscripting an inherently slow process, and are there "better" and "worse" ways of accessing array elements given an x,y coordinates?

More thoughts?

--j

On 1/29/04 10:01 PM, in article onisiu9dq1.fsf@cow.physics.wisc.edu, "Craig Markwardt" <craigmnet@REMOVEcow.physics.wisc.edu> wrote:

>
> Jonathan Greenberg <greenberg@ucdavis.edu> writes:
>> I'm trying to code a semivariance measure to analyze an image in IDL. Since
>> the call requires extracting data from a matrix a certain distance away from
>> the center pixel, is there a fast way of coding IDL to extract these
>> locations, or is am I doomed to have a very slow algorithm. Basic gist:
>>
>> For a given pixel/matrix location, subtract off the value of the pixels
>> surrounding that pixel:
>>
>> For pixel at A is at x,y
>> B at x-1,y
>> C at x+1,y
>>
>> Semivariance = ((A-B)^2 + (A-C)^2)/4
>>
>> Any suggestions?
>
> Yes, how about
>
> N = n_elements(A(*,0)) ;; Number of pixels per row
> B = A(0:N-3,*) ; "left" pixel
> Amid = A(1:N-2,*) ; "middle" pixels
> C = A(2:N-1,*) ; "right pixels
>
> semivariance = ((Amid-B)^2 + (Amid-C)^2)/4
>

> Of course this array will have an row size of two less than the
> original. People who know me, also know that you can avoid some of
> the steps above by using a "trick" where IDL will automatically
> truncate arrays if they are too big, which means you can write the
> whole thing as:
>
> Amid = A(1:*,*)
> semivariance = ((Amid-A)^2 + (Amid-A(2:*,*))^2)/4
>
> which is more cryptic, but it saves me from the tragic pain of
> figuring out what "N" is.
>
> Happy semi-variancing,
> Craig

Subject: Re: Rapid "moving windows" access in IDL?
Posted by Chris Lee on Fri, 30 Jan 2004 14:09:14 GMT
View Forum Message <> Reply to Message

In article <BC3F7CE8.18DCE%greenberg@ucdavis.edu>, "Jonathan Greenberg" <greenberg@ucdavis.edu> wrote:

> Ok, now to make this a bit more confusing -- in practice, what I will
> actually be doing is selecting all pixels at about a given distance from
> the center point, creating a ring of pixels that will be used for the
> semivariogram. Getting back to an earlier question, is the array
> subscripting an inherently slow process, and are there "better" and
> "worse" ways of accessing array elements given an x,y coordinates? More
> thoughts?
> --j

Why not make a ring of 1s and 0s..e.g.

```
nx=100
ny=100

dist=shift(dist(nx,ny),nx/2, ny/2)
mask=fltarr(nx,ny)
mask[*]=0.0
outer=25
inner=23

mask[where(dist lt outer)]=1.0
mask[where(dist lt inner)]=0.0

;mask contains a ring of 1s
```

ring = mask * data .
OR
ring = data[where(mask eq 1)]

Chris.

Subject: Re: Rapid "moving windows" access in IDL?
Posted by Craig Markwardt on Fri, 30 Jan 2004 22:55:26 GMT
View Forum Message <> Reply to Message

"Christopher Lee" <cl@127.0.0.1> writes:
> In article <BC3F7CE8.18DCE%greenberg@ucdavis.edu>, "Jonathan Greenberg"
> <greenberg@ucdavis.edu> wrote:
>
>
>> Ok, now to make this a bit more confusing -- in practice, what I will
>> actually be doing is selecting all pixels at about a given distance from
>> the center point, creating a ring of pixels that will be used for the
>> semivariogram. Getting back to an earlier question, is the array
>> subscripting an inherently slow process, and are there "better" and
>> "worse" ways of accessing array elements given an x,y coordinates? More
>> thoughts?
>> --j
>
> Why not make a ring of 1s and 0s..e.g.
>
> nx=100
> ny=100
>
> dist=shift(dist(nx,ny),nx/2, ny/2)
> mask=fltarr(nx,ny)
> mask[*]=0.0
> outer=25
> inner=23
>
> mask[where(dist lt outer)]=1.0
> mask[where(dist lt inner)]=0.0

This is a good idea. However, for large images you will run into a
problem where you are wasting a lot of time multiplying by zero.

One way around this is to not go as far as Chris said, and preserve
the "where" output, then use that as an offset.

Example:
;; Find center-point of the image

```
  wh0 = where(dist EQ min(dist))
  ;; Find the points around the center which are between inner and outer radius
  wh = where((dist LT outer) AND (dist GE inner))

  ;; Compute locations of the ring as a *offset* from the center pixel
  wh = wh - wh0
```

Now you can use this list of offsets, WH, to index into the original
array wherever you want.

Example: You are interested in the ring of points around the position
 A[20,30]. Then you can do this:

```
 ind_1d = 20 + 30*nx   ;; Compute the 1D index into the array

 A_ring = A[ind_1d + wh]
```

Now A_RING contains all the elements within the ring surrounding
20,30, and you can compute the variance of those elements or whatever.

If you want to compute the whole image, well, then you will have to
loop, but at least the innermost loop(s) are vectorized.

Craig

--

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

Subject: Re: Rapid "moving windows" access in IDL?
Posted by Jaco van Gorkom on Sun, 01 Feb 2004 22:20:51 GMT
View Forum Message <> Reply to Message

"Christopher Lee" <cl@127.0.0.1> wrote in message
 news:20040130.140913.771151432.24672@buckley.atm.ox.ac.uk...
> In article <BC3F7CE8.18DCE%greenberg@ucdavis.edu>, "Jonathan Greenberg"
> <greenberg@ucdavis.edu> wrote:
>
>> Ok, now to make this a bit more confusing -- in practice, what I will
>> actually be doing is selecting all pixels at about a given distance from
>> the center point, creating a ring of pixels that will be used for the
>> semivariogram. Getting back to an earlier question, is the array
>> subscripting an inherently slow process, and are there "better" and
>> "worse" ways of accessing array elements given an x,y coordinates? More
>> thoughts?

>> --j
>
> Why not make a ring of 1s and 0s..e.g.
> nx=100
> ny=100
> dist=shift(dist(nx,ny),nx/2, ny/2)
> mask=fltarr(nx,ny)
> mask[*]=0.0
> outer=25
> inner=23
> mask[where(dist lt outer)]=1.0
> mask[where(dist lt inner)]=0.0
>
> ;mask contains a ring of 1s
>
> ring = mask * data .
> OR
> ring = data[where(mask eq 1)]

There is no really efficient way to acces the elements in a moving window like you need. There is however a fast built-in routine for accessing the (weighted) sum of these elements: CONVOL() . And that might just be enough for you, since your semivariance can probably be rewritten to depend only on the square of the sum over the elements, and on the sum of their squares.

To get an array of the sum over the ring elements for each centre point:
sums = convol(data, mask)
The sums of their squares:
sumsq = convol(data^2, mask)

There is also a very useful program and discussion on David's website: http://www.dfanning.com/math_tips/variance.html . The program by Martin Downing on that page uses SMOOTH() for square "moving windows". By plugging in CONVOL() and its keywords you should be able to use rings of elements, or whatever else you might want.

Hope this helps,
Jaco