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
