Subject: coherence test implementation
Posted by Mark Rehbein on Thu, 13 May 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Hi,

I'm doing some cloud detection on some rather large images using what some people call a coherence test. To do a coherence test on a pixel you take a 3x3 matrix around that pixel and assign that pixel the standard deviation of the 3x3 matrix. I have implemented this the following way:

```
for y=1, lines-1 do begin
  for x=1, pixels-1 do begin
    matrix=ch4(x-1:x+1, y-1:y+1)
    stats=moment(matrix, sdev=sdev)
    sddevimage(x,y)=sdev
  endfor
endfor
```

You might agree that the code above can be more efficient if I use array and matrix operations. I'm fairly new to IDL and haven't been able to successfully use array and matrix operations in this application.

I'd appreciate any help any of you could provide.

Please email mrehbein@aims.gov.au

Thanks

Mark

Subject: Re: coherence test implementation
Posted by Struan Gray on Thu, 13 May 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Craig Markwardt, craigmnet@cow.physics.wisc.edu writes:

> As the two posters graciously have demonstrated,
> they were able to partially vectorize the algorithm.
> However, because your procedure involves *overlapping*
> portions of the same array, it will never be
> possible to fully vectorize it.

True, but you can always use the built-in function CONVOL with a kernal full of ones to do the local-area averaging. My IDL is

installed elsewhere, so I can't check my code, but the following should give the right idea:

```
kernal = intarr(3,3)
kernal[*] = 1
squaremean = convol(image*image, kernal, total(kernal))
mean = convol(image, kernal, total(kernal))
standard_dev = sqrt(squaremean - mean*mean)
```

The nice thing about this method (in addition to running at native speeds) is that it is easily scaleable to larger kernals, and convol handles edge effects for you. The downside is that for big images it can soak up memory, especially if you convert the image to the next-largest data type in order to side-step data overflow problems.

Struan

Subject: Re: coherence test implementation
Posted by eddie haskell on Thu, 13 May 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Dick,

Thanks for your solution! I had just figured out how to do it with the other formulation of SD (i.e., sum((x-mean(x)^2))...) when your post appeared. It was subtracting the mean that was my mental block. Your method is about 20-25% faster than what I came up with (maybe I should go and get out those math books I paid so much for). Anyway, your solution, and the original question, prodded my memory and I can now go back and hopefully finish something I brick-walled on using a variable size submatrix window. Keep up the good work.

As an aside, I had a difficult time extracting the code you attached to your post. It could easily be my news reader but if others are using similar readers and you have other options for attachments, ...

Cheers,
eddie

----- ---- --- --- ---- --- -- --- --- -- -- - - - -
|\ A G Edward Haskell
|\ Center for Coastal Physical Oceanography
|\ Old Dominion University, Norfolk VA 23529
|\ Voice 757.683.4816 Fax 757.683.5550
|\ e-mail haskell*ccpo.odu.edu
----- ---- --- ---- --- --- --- --- -- -- -- - - - -

Subject: Re: coherence test implementation
Posted by Dick Jackson on Thu, 13 May 1999 07:00:00 GMT
View Forum Message <> Reply to Message

(quick, before they wake up in Australia! :-)

Hi again,

Dick Jackson wrote:
>
> You piqued my curiosity, since you're right, there had to be a more
> efficient way! My attached coherence.pro should do the trick, and I
> clock it at about 60 times faster with a 300x300 test.
> [...]
> Generalizing coherence.pro to allow variable 'width' [...]
> wouldn't be hard

I looked at my code again, and realized I was duplicating the function of the IDL CONVOL routine. That sure simplified things! (and in my 300x300 test, cut time in half yet again) Adding the 'width' idea was trivial. I think writing helpful comments was the worst part. :-)

And, as with any Good Thing in IDL, it can be written in one line:

```
coh = Sqrt((Convol(array ^ 2.0, Replicate(1, width, width)) $
          - (Convol(Float(array), Replicate(1, width, width))^2 $
            / (width ^ 2))) $
         / ((width ^ 2)-1))
```

(the attached .pro is much easier to read and understand)

Enjoy.

Cheers,
--
-Dick

Dick Jackson Fanning Software Consulting, Canadian Office
djackson@dfanning.com Calgary, Alberta Voice/Fax: (403) 242-7398
Coyote's Guide to IDL Programming: http://www.dfanning.com/

File Attachments

1) coherence.pro, downloaded 135 times

Subject: Re: coherence test implementation
Posted by Craig Markwardt on Thu, 13 May 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Mark Rehbein <mrehbein@aims.gov.au> writes:
>
> Hi,
>
> I'm doing some cloud detection on some rather large images using what
> some people call a coherence test. To do a coherence test on a pixel
> you take a 3x3 matrix around that pixel and assign that pixel the
> standard deviation of the 3x3 matrix. I have implemented this the
> following way:
>
> for y=1, lines-1 do begin
> for x=1, pixels-1 do begin
> ...

As the two posters graciously have demonstrated, they were able to partially vectorize the algorithm. However, because your procedure involves *overlapping* portions of the same array, it will never be possible to fully vectorize it.

Explicit FOR loops are not bad if the work done during one iteration takes longer than the overhead time with the loop. Of course, that rule of thumb is somewhat CPU dependent.

Craig

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

Subject: Re: coherence test implementation
Posted by Dick Jackson on Thu, 13 May 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Hi Mark,

Mark Rehbein wrote:
>
> Hi,
>
> I'm doing some cloud detection on some rather large images using what
> some people call a coherence test. To do a coherence test on a pixel
> you take a 3x3 matrix around that pixel and assign that pixel the
> standard deviation of the 3x3 matrix.

You piqued my curiosity, since you're right, there had to be a more

efficient way! My attached coherence.pro should do the trick, and I clock it at about 60 times faster with a 300x300 test.

From your code, just call:

sddevimage = Coherence(ch4)

I return an image of the same size and leave the outer ring of pixels as 0.0, is that reasonable?

I used the formula for the SD of a set of 9 values:

SD = Sqrt((Sum(X^2) - (Sum(X)^2 / 9)) / 8)

Generalizing coherence.pro to allow variable 'width' (not fixed at 3) is left as an exercise to the reader. :-) I guess it wouldn't be hard, changing all 'magic numbers' (3, 2, 9 and 8) to width, width-1, width^2 and width^2-1.

Cheers,
--
-Dick

Dick Jackson Fanning Software Consulting, Canadian Office
djackson@dfanning.com Calgary, Alberta Voice/Fax: (403) 242-7398
Coyote's Guide to IDL Programming: http://www.dfanning.com/

File Attachments

1) coherence.pro, downloaded 118 times