Subject: "Max()" filter?
Posted by Dick Jackson on Wed, 06 Mar 2002 18:30:05 GMT
View Forum Message <> Reply to Message

Hi all,

Given these well-known filters for 2D array 'x':

Smooth(x, n)
Median(x, n)
Convol(x, [nxn kernel])

All of them calculate for each element in x (in general) some function of an nxn set of elements around that element. For the three, the function is

Mean()
Median()
WeightedSum()

I'd like a filter that uses the Max() function, so that this array:

1 4 2 4 3
3 5 2 1 4
5 7 4 3 1
3 5 1 1 2
3 1 2 3 1

filtered with width 3 gives:

5 5 5 4 4
7 7 7 4 4
7 7 7 4 4
7 7 7 4 3
5 5 5 3 3

(I think! :-)

Even better would be to supply any function in general. Is there a name for this? Is there any free IDL code for this out there? :-) Google tells me that something called "local max filter" was discussed here once, but that's not really what I want.

Many thanks for any leads.

Cheers,
--
-Dick

Dick Jackson / dick@d-jackson.com
D-Jackson Software Consulting / http://www.d-jackson.com
Calgary, Alberta, Canada / +1-403-242-7398 / Fax: 241-7392

Subject: Re: "Max()" filter?
Posted by Craig Markwardt on Fri, 08 Mar 2002 18:30:01 GMT
View Forum Message <> Reply to Message

"Dick Jackson" <dick@d-jackson.com> writes:
> "trouble" <the_cacc@hotmail.com> wrote in message
> news:5f9f0a23.0203070323.4c32551b@posting.google.com...
>> You could code this up quicker than it took to write the message...
>
> trouble,
>
> Certainly a brute-force method with loops that go over every array element
> and do a Max operator over the appropriate nxn (or less) kernel is easy, but
> I think that it is somewhat inefficient for the 1024x1024 datasets I have in
> mind where the kernel might be around 30x30. In trying it, it runs in about
> 14 seconds.
>
> Perhaps I should have added the word "efficient" somewhere, where I would
> like performance on the same order as the Smooth function, which takes 0.151
> seconds. I believe Smooth saves huge amounts of time by overlapping
> calculations. I don't see how an efficient local max filter like this would
> be trivial to write in IDL, that's why I was asking.

Hi Dick--

These are the kinds of things that JD Smith, Wayne Landsman (?) and I
have wished for before. I believe it's nontrivial to do in IDL as it
exists today.

Craig

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

Subject: Re: "Max()" filter?
Posted by JD Smith on Sat, 09 Mar 2002 02:36:00 GMT
View Forum Message <> Reply to Message

Craig Markwardt wrote:
>
> "Dick Jackson" <dick@d-jackson.com> writes:
>> "trouble" <the_cacc@hotmail.com> wrote in message
>> news:5f9f0a23.0203070323.4c32551b@posting.google.com...
>>> You could code this up quicker than it took to write the message...
>>
>> trouble,
>>
>> Certainly a brute-force method with loops that go over every array element
>> and do a Max operator over the appropriate nxn (or less) kernel is easy, but
>> I think that it is somewhat inefficient for the 1024x1024 datasets I have in
>> mind where the kernel might be around 30x30. In trying it, it runs in about
>> 14 seconds.
>>
>> Perhaps I should have added the word "efficient" somewhere, where I would
>> like performance on the same order as the Smooth function, which takes 0.151
>> seconds. I believe Smooth saves huge amounts of time by overlapping
>> calculations. I don't see how an efficient local max filter like this would
>> be trivial to write in IDL, that's why I was asking.
>
> Hi Dick--
>
> These are the kinds of things that JD Smith, Wayne Landsman (?) and I
> have wished for before. I believe it's nontrivial to do in IDL as it
> exists today.
>

Yes, Craig is correct. I even got so bothered by a similar problem that
I coded up a DLM I call "REDUCE" which implements dimensional reduction
operators similar to cmapply, but all in C (fast). This is an expanded
form of the TOTAL() functionality which allows a dimension over which to
total to be specified (functionality now shared, in IDL v5.5, by MIN()
and MAX()). Currently I've got:

MAX
MIN
MEAN
TOTAL
MULTIPLY
MEDIAN

as in

a=reduce(randomu(sd,128,128,128),2,/MEDIAN)

would yield a 128x128 array with the middle dimension collapsed. Maybe
I should have called it COLLAPSE.

I never got it into releaseable shape, since it appears there is really
no way to code for generic numerical type without profound ugliness.
One of my criteria was always to preserve type (float, int, long, etc.),
unless absolutely necessary (i.e. MEAN,TOTAL,MULTIPLY, where buffer
overflows would be common), and to give flexibility of choosing float or
double results in these cases.

It could easily be adapted to an n x m(x p x q x...) general purpose box
kernel tool, with possible functions:

MIN
MAX (does IDL do this?)
MEAN (IDL does this)
MEDIAN (IDL does this)
TOTAL (IDL sort of does this)
MULTIPLY

Any other suggestions? Anyone interested in such a project? I know,
DLM's are a pain, but sometimes there's no other way.

JD
