Subject: Re: efficient kernel or masking algorithm ?
Posted by John-David T. Smith on Wed, 29 Nov 2000 08:00:00 GMT
View Forum Message <> Reply to Message

Richard Tyc wrote:
>
> WOW, I need to look at these equations over about a dozen times to see what
> is going on ?
>
> I have been struggling with the variance of an nxn window of data, INCLUDING
> central pixel
>
```
>    ;mean of the neighboring pixels (including central)
>    mean=smooth(arr,n)
>    ;square deviation from that mean
>    sqdev=(arr-mean)^2
>    ;variance of an nxn window of data, INCLUDING central pixel
>    var=(smooth(sqdev,n)*n^2-sqdev)/(n^2-1)
>
```

Almost right. Try:

```
var=smooth(sqdev,n)*n^2/(n^2-1)
```

but this still won't yield exactly what you're after, but maybe you're
after the wrong thing ;)

What this computes is a smoothed box variance, not a true box variance,
since the mean you are using changes over the box (instead of
subtracting the mean value at the central pixel from each in the box, we
subtract the box mean value at *that* pixel). Usually, this type of
variance is a more robust estimator, e.g. for excluding outlier pixels,
etc. (in which case you probably should exclude the central pixel after
all to avoid the chicken and egg problem with small box sizes). If you
really want the true variance, you're probably stuck with for loops,
preferrably done in C and linked to IDL.

This reminds me of a few things I've been thinking about IDL recently.
Why shouldn't *all* of these smooth type operations be trivially
feasible in IDL. Certainly, the underlying code required is simple.
Why can't we just say:

```
a=smooth(b,n,/VARIANCE)
```

to get a true box variance, or

```
a=smooth(b,n,/MAX)
```

to get the box max. Possibilities:
*MEAN (the current default)
*TOTAL (a trivial scaling of mean),
*VARIANCE
*MEDIAN (currently performed by the median function, in a addition to
its normal duties. To see why this is strange, consider that total()
doesn't have an optional "width" to perform neighborhood filtering).
*MIN
*MAX
*MODE
*SKEW
etc.

To be consistent, these should all operate natively on the input data
type (float, byte, long, etc. -- like smooth() and convol() do, but like
median() does not!), and should apply consistent edge conditions
activated by keywords. These seem like simple enough additions, and
would require much reduced chicanery.

While I'm on the gripe train, why shouldn't we be able to consistently
perform operations along any dimension of an array we like with relevant
IDL routines. E.g., we can total along a single dimension. All due
respect to Craig's CMAPPLY function, but some of these things should be
much faster. Resorting to summed logarithms for multiplication is not
entirely dubious, but why shouldn't we be able to say:

```
col_max=max(array,2,POS=mp)
```

and have mp be a list of max positions, indexed into the array, and
rapidly computed? While we're at it, why not

```
col_med=median(array,2,POS=mp)
```

IDL is an array based language, but it conveniently forgets this fact on
occassion. Certainly there are compatibility difficulties to overcome
to better earn this title, but that shouldn't impede progress.

JD

--
J.D. Smith | WORK: (607) 255-6263
Cornell Dept. of Astronomy | (607) 255-5842
304 Space Sciences Bldg. | FAX: (607) 255-5875
Ithaca, NY 14853 |