Subject: Re: efficient kernel or masking algorithm ?
Posted by Struan Gray on Thu, 30 Nov 2000 08:00:00 GMT
View Forum Message <> Reply to Message

J.D. Smith, jdsmith@astro.cornell.edu writes:

> If you really want the true variance, you're
> probably stuck with for loops,
> preferrably done in C and linked to IDL.

There was a thread back in August called 'Standard Deviation' where we discussed this a bit. The best i could come up with was to use the SHIFT function, which for a 3x3 kernal looks like this:

```
;*******************************

function smg_imageSD, image

  fimage = float(image)
  localmean = smooth(fimage, 3)
  sum = (fimage - localmean)^2
  sum = temporary(sum) + $
    shift((fimage - shift(localmean, 1, 1))^2,-1,-1)
  sum = temporary(sum) + $
    shift((fimage - shift(localmean, 0, 1))^2, 0,-1)
  sum = temporary(sum) + $
    shift((fimage - shift(localmean,-1, 1))^2, 1,-1)
  sum = temporary(sum) + $
    shift((fimage - shift(localmean, 1, 0))^2,-1, 0)
  sum = temporary(sum) + $
    shift((fimage - shift(localmean,-1, 0))^2, 1, 0)
  sum = temporary(sum) + $
    shift((fimage - shift(localmean, 1,-1))^2,-1, 1)
  sum = temporary(sum) + $
    shift((fimage - shift(localmean, 0,-1))^2, 0, 1)
  sum = temporary(sum) + $
    shift((fimage - shift(localmean,-1,-1))^2, 1, 1)

  sum = sqrt(temporary(sum)/8)

  dims = size(sum, /dim)
  sum[0,*] = 0.0
  sum[*,0] = 0.0
  sum[dims(0)-1,*] = 0.0
  sum[*,dims(1)-1] = 0.0

  return, sum
end
```

;*******************************

This handles the edges crudely, but a mixture of padding the original image
and using the right keywords when doing the initial smooth allows more complex
behaviours.

Obviously, it would be possible to generalise the shifting and final sqrt for
other kernal sizes, and to add all sorts of checking for different data types.

It's reasonably fast.

Struan
