Subject: Re: Finding the mean of a set of images
Posted by Jaco van Gorkom on Tue, 22 Oct 2002 17:48:51 GMT
View Forum Message <> Reply to Message

"David Oesch" <oesch@giub.unibe.ch> wrote in message
news:3DB56D82.2090407@giub.unibe.ch...
> Craig Markwardt wrote:
>> David Oesch <oesch@giub.unibe.ch> writes:
>>> ...
>>> Does anyone have an algorithm for finding the mean/standardeviation etc
>>> at each pixel position for a set of equal size 2-D images? Currently the
>>> only way I have to do this is to extract all the values for a given
>>> pixel position into a 1-D array and find the mean/standardeviation etc
>>> on that. Doing it pixel by pixel like this is inefficient in IDL so I am
>>> looking for an *array* based algorithm that would find all
>>> the mean/standardeviation etc in parallel.
>>
>> Sure, if you stack your image into a 3D image cube, then you would
>> have something like IMAGE = FLTARR(NX, NY, NIMAGES)
>>
>> Then the mean image is:
>>
>>   mean = total(image,3)/nimages
>>
>> ...
> Yes, I got that one with TOTAL, the problem is, it only works with
> imagesets where all datalayers consists of 100% valid pixels...but this
> is not always the case...
> to eliminate those values, I came as far as
>
> mean = total((image NE -9999),3)/nimages
>
> but the problem is, nimages needs to be changed to...

Yes. I suppose you mean total(image * (image NE -9999),3....

How about

```
  valid = image NE -9999
  mean = TOTAL(image * valid, 3) / TOTAL(valid, 3)
```

Or, if you would use NaN (Not-a-Number, floating point!) as missing value:

```
  mean = ( SMOOTH(image, [1,1,Nimages], /NAN) )[*, *, Nimages/2]
```

For that to work you need to enforce an odd number of images.

cheers,
 jaco