Subject: Re: 2D FFt
Posted by Peter Mason on Tue, 22 Oct 1996 07:00:00 GMT
View Forum Message <> Reply to Message

On Mon, 21 Oct 1996, Walid Atia wrote:
> Does anyone out there know of a routine which takes the 2d FFT of an
> image, much like that presented in the IDL demo? I know this can be
> done using the 1D FFT routine for each line, but has anyone actually
> written a routine to do this, or know of how it was done in the demo?

IDL's FFT routine does work on 2D (or higher) arrays.
For the record, you could get an FFT of a 2D array by first doing the 1D
FFT down each column, then doing the FFT across each row (of the column-FFT
product). Or vice-versa (same result).

> Also, how does one then apply a filter, as was done in the IDL demo, to
> the FFT data (obviously, by multiplying in the frequency domain, and
> then inverse FFTing the data, but how does one create the filter
> initially?).

There's a lot to this subject, and I only know a tiny fraction of it.
Anyway, I hope that this note will be of some help to you.

Let's call the FFT output F, and say it has dimensions NS and NL.
i.e., F is a COMPLEXARR(ns,nl).

Firstly, it'll be easier on you if you centre the FFT array. IDL outputs a
2D FFT with the zero-frequency element at F(0,0). The FFT has both
positive and negative frequencies (it has symmetrical frequencies). You may
find things more intuitive if you shift the FFT so that its zero frequency is
in the centre if the array. Also, the FFT of a real (datatype, as opposed
to complex) array is symmetrical, and it's arguably more useful to view it
centred like this.
So do:

```
  NS2=NS/2+1 & NL2=NL/2+1
  F1 = SHIFT(F,NS2,NL2)
```

As regards filters... There are many "standard" kinds. RSI used to have
some handy documentation on FFTs and filtering, but I can't find it in the
current user and reference manuals.
Anyway, the easy way to do it is to set up an array, say FILT=COMPLEXARR(NS,NL)
with values ranging from (>=0.0) to (<=1.0) (according to the filter) and do
F=TEMPORARY(F)*J to execute the filter.
Once you're done viewing F (or whatever), transform it back to the "time"
domain with: DATA=FFT(SHIFT(F,-NS2,-NL2)). If your original data was
real and your filter was symmetrical w.r.t. frequency, DATA's imaginary
component will be zero and you should do DATA=FLOAT(TEMPORARY(DATA)) to

get rid of it.

e.g., Let's say you wanted to use a "low-pass" exponential filter of order 3 with a "cutoff frequency" of N/4 of an image IMG, a FLTARR(N,N). ("Low pass" means that higher frequencies will be attenuated by the filter.   The exponential low-pass filter doesn't attenuate at all at the zero frequency, attenuates by a factor of 0.5 at its "cutoff frequency" and attenuates more strongly at higher frequencies.)

1] Get the FFT:
```
  F=FFT(IMG,-1)
```

2] Centre it:
```
  N2=N/2+1 &F=SHIFT(TEMPORARY(F),N2,N2)
```

3] At this point you might want to view the FFT result.   Normally the "power "spectrum" is viewed.   This is approx. == (ABS(F))^2.   Typically ALOG(ABS(F)) is viewed.

4] Set up the filter (one row at a time):
```
  ORD=3     ;filter order
  CUT2=(N/4)^2    ;(cutoff freq)^2
  FILT=COMPLEXARR(N,N)
  X=(FINDGEN(N)-N2)^2   ;N2 as in centring
  FAC=FLTARR(N)    ;we'll build it here, 1 line at a time
  FOR LINE=0L,N-1L DO BEGIN
    FAC(0)=EXP( -( (X+X(LINE))/CUT2 )^ORD )
    FILT(0,LINE)=COMPLEX(FAC,FAC) ;FILT has real==imaginary values
  ENDFOR
```

5] Filter:
```
  F=TEMPORARY(F)*FILT
```

6] Back to the time domain:
```
  FILTERED_IMG = FLOAT( FFT( SHIFT(F,-N2,-N2), -1) )
```

Peter Mason