Subject: FFT's and images: or My 4th day on what I thought would take me only 6hrs
Posted by tbowers0 on Fri, 17 Aug 2001 02:01:52 GMT
View Forum Message <> Reply to Message

Ok, 4 days is quite enough time to spend hacking over a problem before
I post. So here goes...
All I'm tryin' to do is MTF (modulation transfer function) blur an
image then restore it with:

```
blurredImage = fft(fft(originalImage,-1) * fft(pntSpreadFn,-1) ,1)
```

then reverse it for check...

```
reconstructedImage = fft(fft(blurredImage,-1) / fft(pntSpreadFn,-1)
,1)
```

1) How, oh HOW in the world do I create a *general* 2D gaussian to use
as my PSF to start off with. I.e, if my image is [sizeX,sizeY] and
sizeX does not necessarily equal sizeY, how do I create a symmetric 2D
gaussian?? The best I can do is a 2D gaussian with (sizeX eq sizeY)
always. Images are rarely the same in x and y.

2) Could anyone also give my the scoop on how to take any 1D psf(r or
psi) and create a 2D psf from it? E.g. I have psf for all angles:

```
angle=indgen(0,181)
psf=[p0,p1,p2,...,p180]
```

how do I set this up for fft'ing to apply as the filter for my 2D
image? Doesn't this also need to be done for general [sizeX,sizeY]?

Learning this fft() stuff is definately tricky. I've done alot of
internet searching and can't seem to find the info I need. Many times
I've seen examples of people 'hand-coding' the filter and applying
that to the fft image, like:

```
fakeFFT_pntSpreadFn = 1.0 / (1.0d + DIST(iXSIZE)/15.0)^2
blurredImage = fft(fft(originalImage,-1) * fakeFFT_pntSpreadFn ,1)
```

But aha! They never actually fft the filter, they just create it in a
way that makes it look fft'd! (peaks at the edges and all) Sneaky...
Yes, I'm aware that the fft of gaussian is a gaussian, but I need to
actually do it!

3) Should I normalize (make range 0 to 1.0) the psf? The fft of the
psf? The fft of the image? Anything? I ask because in my trials and
mostly errors I sometimes get a blurry image that has pixel values
magnitudes off of the original image. E.g. originalImage may range

[0,255] and resulting blurredImage ranges [5e-12,8e-08] or some other
variant. I *finally* realized that the forward fft was dividing by n
(at least, I think) and the reverse wasn't, so I thought this was my
problem. But, after testing, I'm not sure.

4) Lastly, if the psf has 0's in it anywhere, won't this screw up my
division when I try to reconstruct the image? I ask because in my
futile efforts at creating a gaussian psf I often manifest some at
the edges. Of course, when I try to add a very small offset to
where(psf eq 0.0), e.g. 1e-10, just to eliminate 'em, I sometimes get
more strange results.

My sanity is in your hands. Many, many thanks in advance.
todd