Subject: Re: FFT and ROTATE
Posted by wheinz on Fri, 05 Sep 2008 21:04:11 GMT
View Forum Message <> Reply to Message

Thanks for the suggestions. I am still confused by a few things, but with Wox's code I think I can clarify my question. The problem is that although Wox's code shows that the FFT of the rotated image and the rotated FFT of the original image give you back images that look like the original when inverse -transfored, the values of the coefficients in those two FFTs are not the same. I added some print statements to Wox's code to show what is confusing me.

```
pro rotFFTtest3
  ;;load an image
  fn = filepath('md1107g8a.jpg',SUBDIRECTORY='examples/data')
  image=bytarr(251,251)
  image90=bytarr(251,251)
  image[0,0]= read_image(fn)
  image90[0,0] = rotate(read_image(fn),1)

  ;display the images
  window,0
  tvscl,image,0
  tvscl,image90,1
  n = size(image,/dim)
  nfreq=n/2+1 ; # positive freq in each dim
  nfreq_m=nfreq-1-(~(n mod 2)) ; # negative fequencies in each dim

  ;;take fft of image
  f = fft(image)

  ;;shift it
  f = shift(f,-nfreq[0],-nfreq[1])

  ;;rotate the fft 90 degrees
  f90_1 = rotate(f,1)

  ;;shift it back
  f90_1 = shift(f90_1,nfreq[0],nfreq[1])
  ;;take the fft of image90 -- the rotated image
  f90_2 = fft(image90)

;;**********new code to print sorted values of coefficients**********
  ;;get the real parts of the ffts
  fr =real_part(f)
  f90_1r = real_part(f90_1)
  f90_2r = real_part(f90_2)
```

```
  ;;now we look at the set of real coefficiencts of each fft
  sf = fr[sort(fr)]
  s1 = f90_1r[sort(f90_1r)]
  s2 = f90_2r[sort(f90_2r)]

  print,'The sorted list of real coefficients.'
  print,'  fft(image):',sf[0:4]
  print,' rotated fft:',s1[0:4]
  print,'fft(image90):',s2[0:4]

  ;;**********end new code to print sorted values of
coefficients**********


  tvscl,fft(f90_1,/inverse),2
  tvscl,fft(f90_2,/inverse),5

end
```

The program prints the following:

```
IDL> rotFFTtest2
The sorted list of real coefficients.
  fft(image):     -36.3499     -36.3499     -26.5806     -26.5806
-5.83106
 rotated fft:     -36.3499     -36.3499     -26.5806     -26.5806
-5.83106
fft(image90):     -36.3499     -36.3499     -26.6964     -26.6964
-5.65933
```

So, you can see that the values differ, and (to answer ken's question) the differences in the values are greater than the machine's precision. Here I only printed the real parts, but the same thing happens with the imaginary parts.

The fact that the inverse transformations return the original image suggests that there is a phase shift introduced somewhere in the transform. I can understand that if the array has an even number of elements in x or y, then a rotation forces a translation of the pixels. But an array with an odd number of elements in x and y should not, especially if we are only looking at 90 degree rotations.

Why aren't the sorted lists of the real parts of the coefficients from the rotated fft and the fft of the rotated image equal?

The article Vince linked to states that different implementations of the FFT use different indexing and normalization approaches. Is it possible that these differences could come from whatever scheme IDL

uses for its FFT?

Thanks,
Will