Inverting a 2D Fourier Transform

Biophysics.
Prof. Joshua Deutsch

Rafael Díaz
Ryan Hoffman
Joe Platzer

April 11,2013

All the members of the team discussed the interpretation of a Fourier Transform (FFT) and its meaning. Ryan Hoffman and Joe Platzer identified the images obtained from the program; Rafael Díaz wrote the required code and work on the mathematical part of the results interpretation.

Inverting a FFT of a 2D Image

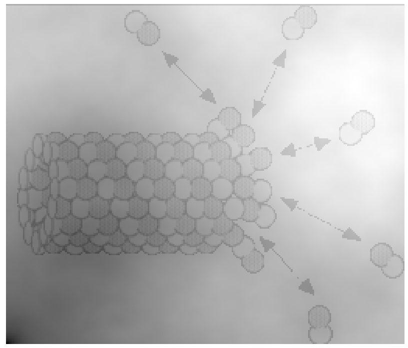
The code used for inverting the FFT's is shown below. A minor modification from the suggested one was implemented in order to process all the images automatically.

```
#You always will be importing a number of modules, or libraries initially
import numpy
from scipy import *
from pylab import *
import scipy
import os
import re


#In python, a function call always begins with "def".
def get_d(shp):
#   The argument to the function get_d in this case is a "tuple"
#   shp is the variable containing the shape of an array, that is
#   the dimensions (number of columns, and number of rows)

#   Suppose shp = (10,20), then below, you'd see below, that m=10, and n = 20
    m = shp[0]
    n = shp[1]
#   The next line creates an array "dsq" of dimensions shape, all initialized to 0:
    dsq = zeros(shp)
#   Below is the most common way to loop. range(m) creates a list of numbers [0,1,2,...,n-1]
#   We have two for loops, meaning that we'll be assigning values to every element dsq[i,j]
    for i in range(m):
         for j in range(n):
#   Here the R.H.S. calls another function called fold, defined below. "**" means "to the power of"
              dsq[i,j] = fold(i,m)**2 + fold(j,n)**2
#   It hands us back the array dsq filled up with the right values.
    return dsq
```

```
# This is another function that is useful when dealing with fourier transforms. As
# a function of x it goes up and then down again, like a triangle with a max at n/2
def fold(x,n):
    if x < n/2:
         return x
    else:
         return n-x


#  This finds the minimum value in an array of numbers
def mini(a):
     return a.flatten()[a.argmin()]

#  This finds the maximum value in an array of numbers
def maxi(a):
     return a.flatten()[a.argmax()]


num_images = 0

#  The next 3 lines iterate over all files that end in ".png"
#  With each one of these, we perform operations described below.
for rootdir, dirs, files in os.walk('encoded_images_0/'):
    for file in files:
        if re.search(".png",file):
#            Read in the image.
             image_read = imread(os.path.join(rootdir, file))
#            keep track of the number of images that we're processing
             num_images += 1
             print "processing image ", num_images, " called ", file

# read in an fft_image, call it fft_pic.png
             fft_image = image_read
# now subtract off the average value of the fft:
             ave = average(fft_image)
             fft_image -= ave
             dsq_array = get_d(fft_image.shape)
# now divide fft_image by dsq_array
             fft_image /= dsq_array # /= ?

# now we've just divided by zero so
             fft_image[0,0] = 0.0
# now take the inverse fourier transform (ifft2) and the real part of that (real)
             image  = real(ifft2(fft_image))
# Now show the image:
             colormap = cm.gist_gray


# The following lines are needed to obtain an image with the right orientation
             shp = shape(image)
             m = shp[0]/2
             n = shp[1]/2
             misc.imsave("decoded images/inverse_fft"+str(num_images)+".png",image[m:0:-1,0:n])
```

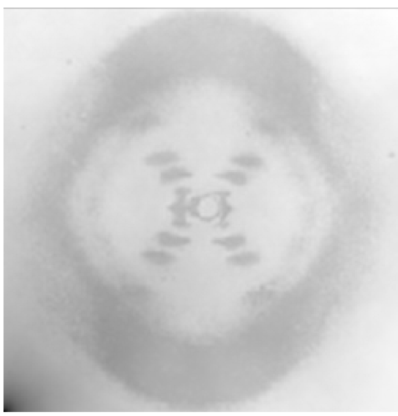
The images obtained using the code above are shown in Fig. 1.

(a) The telomeric end of a piece of linear DNA.

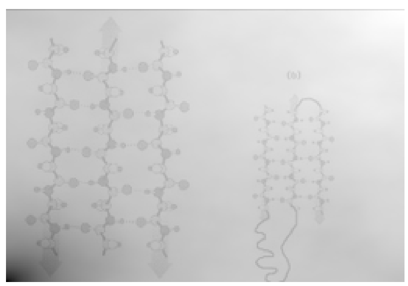


(b) Microtubule polymerization/depolymerization.

(c) A mitochondria.

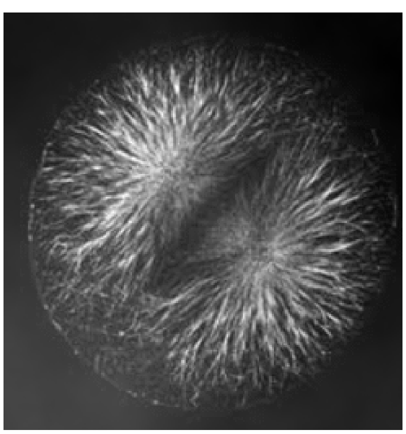


(d) X-ray diffraction pattern produced by the DNA double helix.

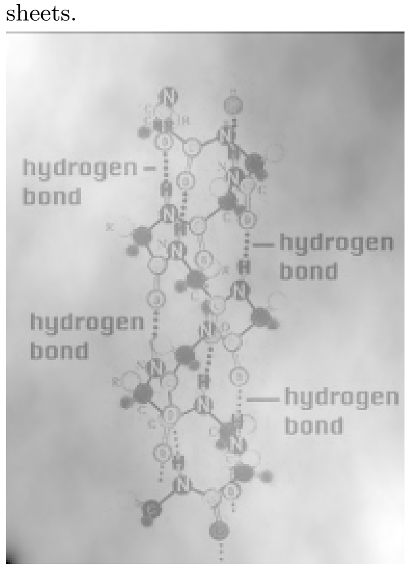


(e) Some antiparallel protein beta-sheets.

(f) Picture of Watson and Crick with their model DNA.



(g) A cell in metaphase of mitosis.



(h) A protein alpha-helix.

(i) A bacteriophage.

Figure 1: Images obtained inverting a 2D FFT

Interpretation of results

1. Inverse Fourier Transform of the Real Part of a FFT.

   Let us assume that $G(k)$ is the FFT of $g(x)$, *i.e.*

   $$G(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-ikx} g(x) \mathrm{d}x.$$

   Since $e^{i\alpha} = \cos(\alpha) + i\sin(\alpha)$, the real part of the last equation is (assuming $g(x)$ is real):

   $$Re(G) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \cos(kx) g(x) \mathrm{d}x. \tag{1}$$

   Now, in general, the Inverse Fourier Transform of any function $F(k)$ is

   $$\mathcal{F}^{-1}\{F(k)\} = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{ikx} F(k) \mathrm{d}k.$$

   Thus, plugging in Eq. (1) in the last expression (suppressing the integral limits for clarity) we get

   $$\begin{aligned} \mathcal{F}^{-1}\{Re(G(k))\} = & \frac{1}{2\pi} \int e^{ikx'} \left\{ \int \cos(kx) g(x) \mathrm{d}x \right\} \mathrm{d}k \\ = & \frac{1}{2\pi} \iint e^{ikx'} \left(\frac{e^{ikx} + e^{-ikx}}{2} \right) g(x) \mathrm{d}x \mathrm{d}k \\ = & \frac{1}{2} \iint \left(\frac{e^{ik(x'+x)} + e^{ik(x'-x)}}{2\pi} \right) g(x) \mathrm{d}x \mathrm{d}k \end{aligned} \tag{2}$$

   But, by definition,

   $$\delta(x - x') = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{ik(x-x')} \mathrm{d}k.$$

   Therefore, by reversing the order of integration in the last line of (2), the result is

   $$\begin{aligned} \mathcal{F}^{-1}\{Re(G(k))\} = & \frac{1}{2} \int g(x) \left[\delta(x + x') + \delta(x' - x) \right] \mathrm{d}x \\ = & \frac{g(x') + g(-x')}{2}. \end{aligned} \tag{3}$$

   What this show is that, by taking only the real part of a FFT and then inverting it, one will not obtain the original function. Rather a superposition of it is obtained.

2. Fourier Transform of $\cos(ax)$ Analytically, the FFT of $\cos(ax)$ is a sum of two δ-functions, centered at a and $-a$:

   $$\begin{aligned} F(k) = \mathcal{F}\{\cos(ax)\} = & \frac{1}{\sqrt{2\pi}} \int e^{-ikx} \cos(ax) \mathrm{d}x \\ = & \frac{1}{2} \frac{1}{\sqrt{2\pi}} \int \left(e^{ix(k+a)} + e^{-ix(k+a)} \right) \mathrm{d}x \\ = & \sqrt{\frac{\pi}{2}} \left[\delta(a - k) + \delta(a + k) \right] \end{aligned} \tag{4}$$

On the other hand, using the code provided in the course web page, the resultant plot is depicted in Fig. 2. As can be seen in this figure, the two "δ peaks" expected do appear. However, the one at $-a$ is translated, just as happened in the examples during the lecture.

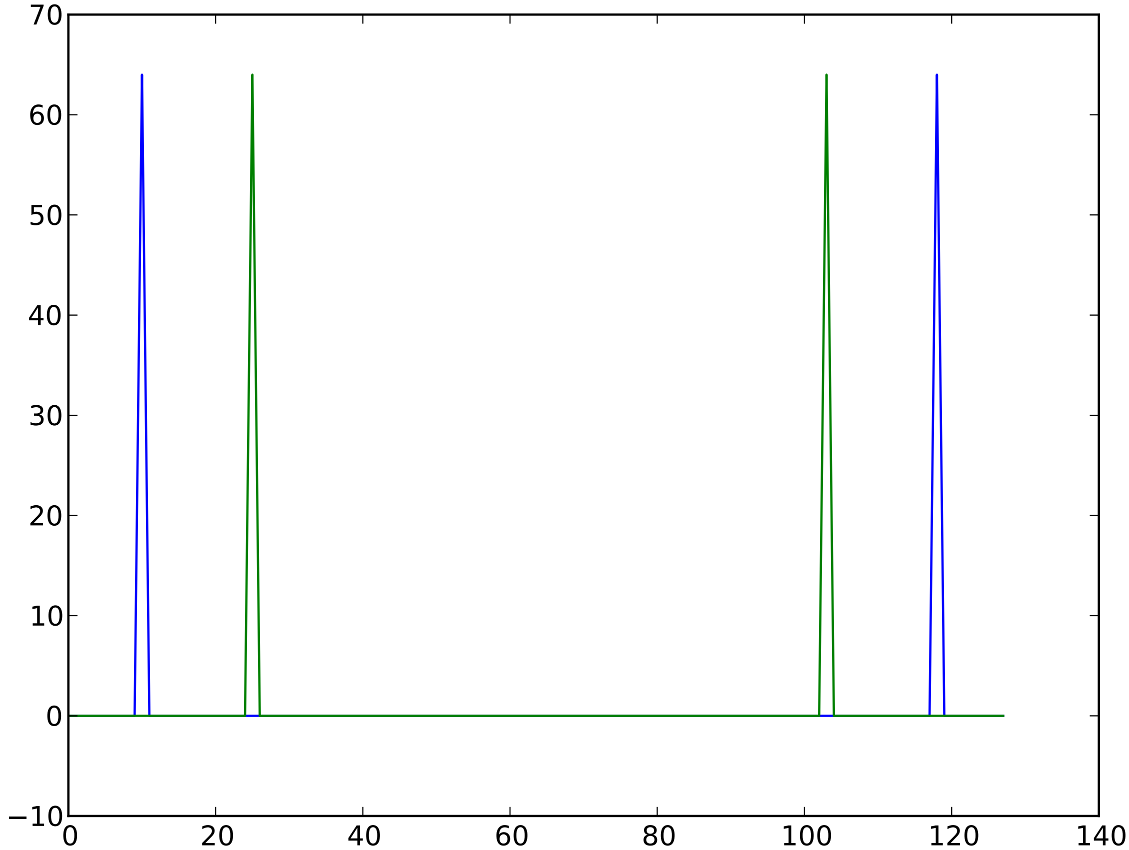


Figure 2: FFT of $\cos 10x$ (blue line) and $\cos 25x$ (green line).

Appendix

In this section, we present various screen shots to show that all the members of the team were able to install Python properly.

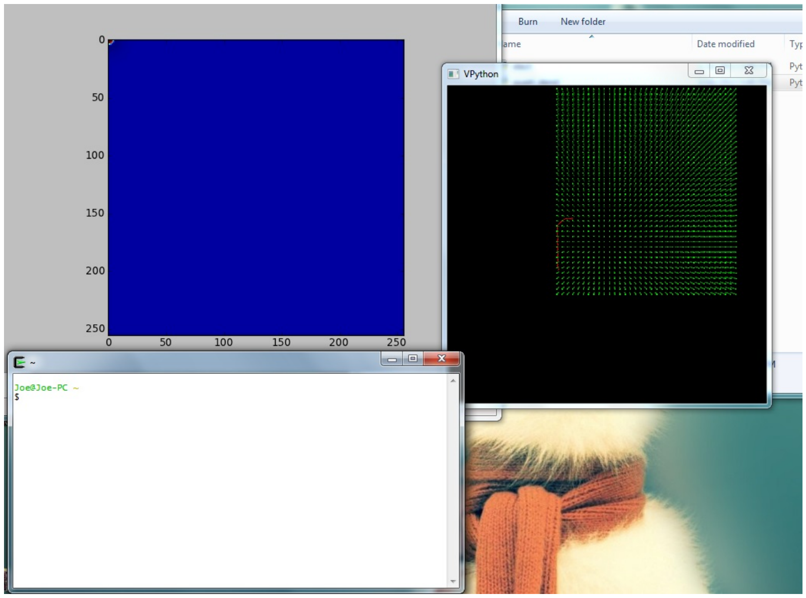
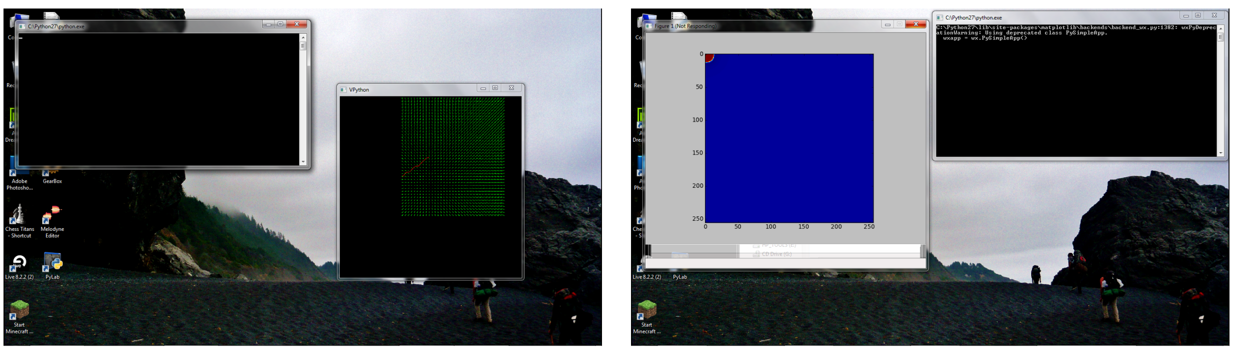


Figure 3: Joe Platzer's screen shot.



(a) Screen shot of `elect.py`.

(b) Screen shot of`quart-dend.py`.

Figure 4: Ryan Hoffman's screen shots

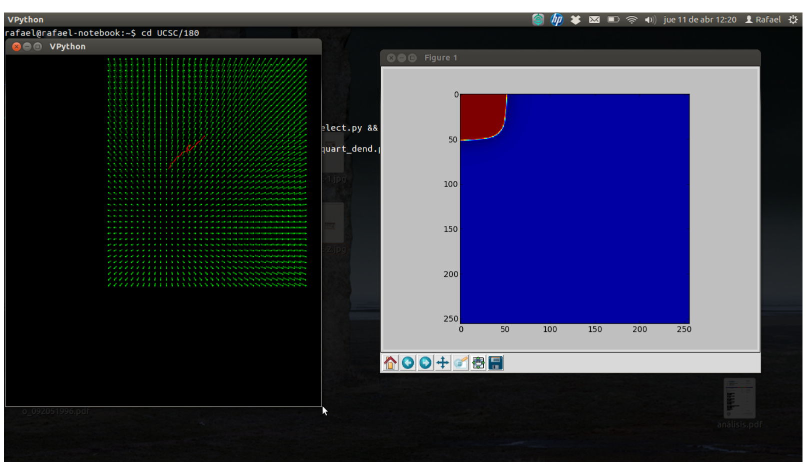


Figure 5: Rafael Díaz's screen shot.