With J

Fast Fourier Transforms and Removing Motion Blur

—*by Cliff Reiter*
Department of Mathematics
Lafayette College

FAST FOURIER TRANSFORMS make it possible to convert back and forth between image space and frequency space. It is amazing that it is possible to remove of some of the blur caused by motion or improper focus [5]. J offers a powerful add-on package *fftw* based on work of Frigo and Johnson [2], that makes it easy to combine fast Fourier transforms with other array processing. This column closely follows the discussion of motion blur in [4].

Fourier transform on a small array

We will not discuss the mathematics of computing the Fourier transform. However, we remark that the Fourier transform of an argument array results in another array. While the input array is ordinarily a real array representing an image, any complex valued array is allowed. The result, typically a complex valued array, can be thought of as giving a real and imaginary part for each entry. Or, more important for applications, we can think of those complex entries as a magnitude and phase. Images of the magnitude essentially give diffraction patterns [1,3]. The *fftw* add-on package can be downloaded from *www.jsoftware.com*. We load the *fftw* add-on package, create a small example and compute the Fourier transform of that array as follows:

```
   require 'system\packages\fftw\fftw'

   ]a=:</~i.4
0 1 1 1
0 0 1 1
0 0 0 1
0 0 0 0

   fftw a
   6   _2j2 _2 _2j_2
2j_2      0  0  _2j2
   2      0 _2     0
 2j2 _2j_2  0      0
```

The real and imaginary parts may be computed:

```
   <"2 ] 9 11 o./ fftw a
┌──────────┬──────────┐
│6 _2 _2 _2│ 0  2 0 _2│
│2  0  0 _2│_2  0 0  2│
│2  0 _2  0│ 0  0 0  0│
│2 _2  0  0│ 2 _2 0  0│
└──────────┴──────────┘
```

The magnitude and phase may be computed:

```
   ,.<"2 ]7j3": 10 12 o./ fftw a
┌──────────────────────────────┐
│  6.000  2.828  2.000  2.828  │
│  2.828  0.000  0.000  2.828  │
│  2.000  0.000  2.000  0.000  │
│  2.828  2.828  0.000  0.000  │
├──────────────────────────────┤
│  0.000  2.356  3.142 _2.356  │
│ _0.785  0.000  0.000  2.356  │
│  0.000  0.000  3.142  0.000  │
│  0.785 _2.356  0.000  0.000  │
└──────────────────────────────┘
```

The magnitude also may be computed as follows:

```
   fftw^:_1 fftw a
0 1 1 1
0 0 1 1
0 0 0 1
0 0 0 0
```

The process is reversible:

```
   |fftw a
      6 2.82843 2 2.82843
2.82843       0 0 2.82843
      2       0 2       0
2.82843 2.82843 0       0
```

We find it useful to define two utility functions below. If the input array is thought of as an optical mask, the image of the magnitude of the fast Fourier transform would give the optical transform. However, the result of `fftw` is centered in such a way that in order to get classical diffraction patterns, we need to rotate rows and columns to move the corners to the center. The function HR defined below does that recentering. Also, when we want to view images of the magnitudes appearing in images, it is convenient to divide the data into coherent discrete regions where each discrete level has nearly equal area. The function `cile` allows us to accomplish that rescaling:

```
   hr=:-:@# |. ]

   HR=:hr"1@:hr
```

```
   i.4 4
 0  1  2  3
 4  5  6  7
 8  9 10 11
12 13 14 15

   HR i.4 4
10 11  8  9
14 15 12 13
 2  3  0  1
 6  7  4  5

   cile=:$@] $ ((/:@/:@] <.@:* (%#)),)

   3 cile i.7 7
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 1 1 1 1
1 1 1 1 1 1 1
1 1 1 1 1 2 2
2 2 2 2 2 2 2
2 2 2 2 2 2 2
```

We now turn to using Fourier transforms to remove motion blur.

Removing motion blur

Motion blur can, to an extent, be removed by dividing the Fourier transform of the blurred image by the Fourier transform of a line segment that defines the motion of the blur. In practice, difficulties are caused by division of entries near zero, so we use a twist by a constant K. That is, if B denotes the Fourier transform of the blurred image and L denotes the Fourier transform of the line segment, then $\frac{B}{L} = B\frac{\bar{L}}{L\bar{L}} \approx B\frac{\bar{L}}{K + L\bar{L}}$ when K is "small" and $\bar{L}$ denotes the complex conjugate of L; and thus, $L\bar{L}$ gives the square of the magnitude of L. We will use utilities `writebmp8` and `readbmp8` from the script *raster4.ijs* which is available from the link to the web-page for this column that can be found at *http://www.lafayette.edu/~reiterc/j/withj/index.html*. The blurred image and the motion removing line image can also be found there.

In our computation below, `fft_blur` denotes the Fourier transform of the blurred image, and `fft_line` denotes the Fourier transform of the line segment. The Fourier transform of the deblurred image is given by the inverse transform of `fft_blur*(+fft_line)%K+*:|fft_line` where `+fft_line` gives the complex conjugate of `fft_line`. By experimenting, we identified a line segment that corresponded to the motion blur. Notice the extent of the motion blur can be estimated by the barely visible edges of the gasoline cap cover in the blurred image. The spacing of the hot spots in the Fourier transform also is indicative of the extent of the motion. We found that the constant `K=:10*255^2` worked reasonably well. While that `K` isn't small in absolute terms, it is not large compared to the entries in the arrays. Figure 1 (on page 18) shows the motion blurred image, the line, the deblurred image and the Fourier transform of each. The expressions required to create these images from the blurred image and the line segment follow. The palette `w256b` is a white to black grayscale palette.

```
'w256b blur'=:readbmp8 'blur.bmp'

'w256b line'=:readbmp8 'line.bmp'

fft_blur=:fftw blur

(w256b;256 cile |fft_blur) writebmp8 'fft_blur.bmp'

fft_line=:fftw line

(w256b;256 cile |fft_line) writebmp8 'fft_line.bmp'

fft_conv=:fft_blur*(+fft_line)%(10*255^2)+*:|fft_line

(w256b;256 cile |fft_conv) writebmp8 'fft_conv.bmp'

deblur=:fftw^:_1 fft_conv

(w256b;HR 256 cile |deblur) writebmp8 'deblur.bmp'
```

In practice, one can only hope to remove a small amount of the blurring and undesirable artifacts are introduced. Noticed the vertical bands in the deblurred image. However, in applications, even removal of a small amount of the blurring can be extremely important. In our illustration it makes the difference between being able to read a serial number and not. ■

References

1. N. W. Allis, J. P. Dumont, F. J. Heiss, and C. A. Reiter, Fast Fourier Transforms, Diffraction Patterns, and J, *Vector*, 16 3(2000) 111–116
2. M. Frigo and S. Johnson, *FFTW Homepage*, http://www.fftw.org
3. G. Harburn, C. A. Taylor and T. R. Welberry, *Atlas of Optical Transforms*, G. Bell & Sons, Ltd, (1972)
4. C.A. Reiter, *Fractals, Visualization, and J, Second Edition*; to appear
5. J. C. Russ, *The Image Processing Handbook*, Second Edition, Boca Raton, CRC Press, 1994

***Cliff Reiter**, when he isn't wandering around the Adirondacks, might be found teaching math at Lafayette College. He can be reached via e-mail at "reiterc@lafayette.edu".*

See accompanying figures on the following page.

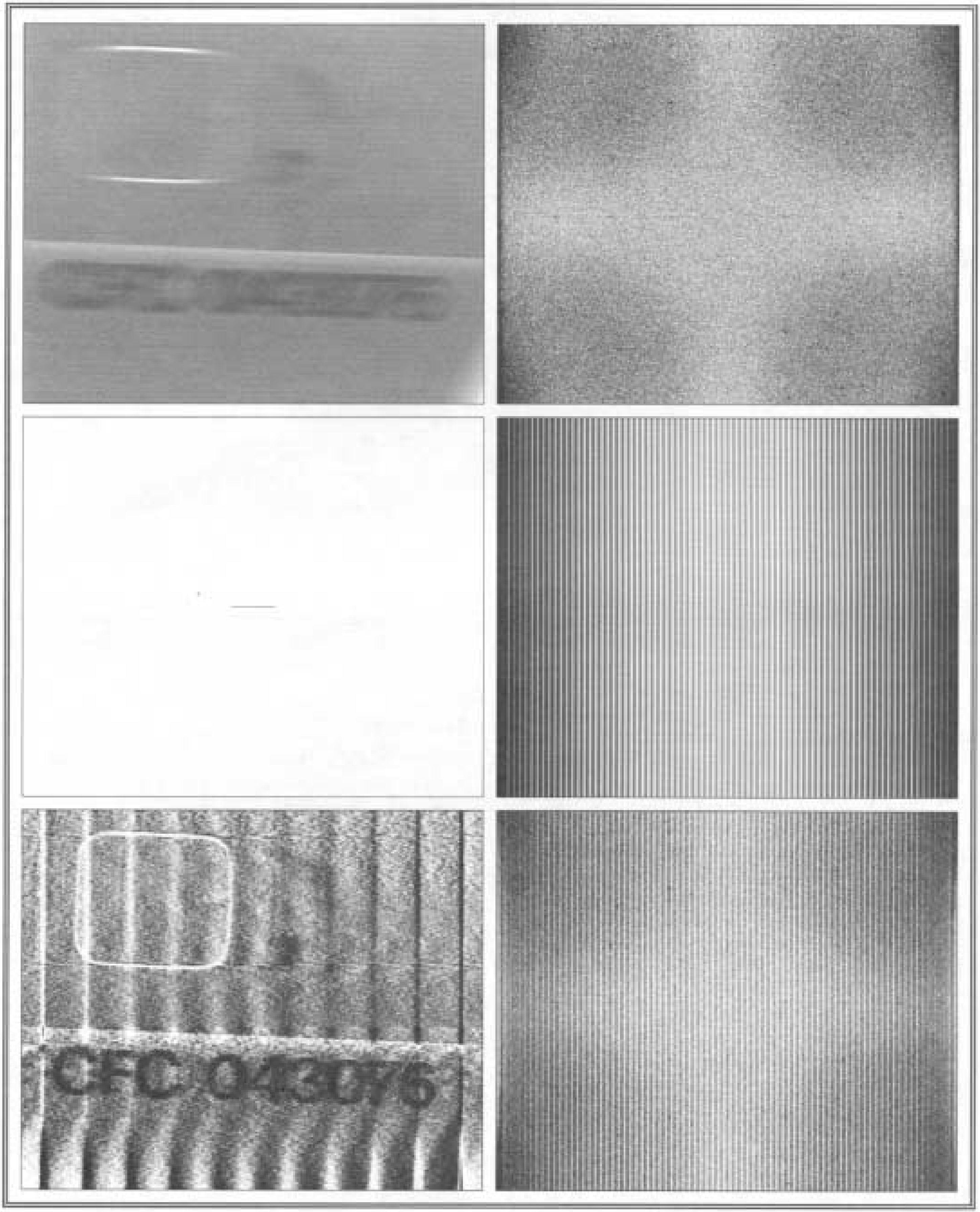

Figure 1: Motion-blurred image, line segment, deblurred image, and their Fourier transforms