

Preliminary Study for the Development of an IUE High Dispersion Line-By-Line File

Muriel A. Taylor
Astronomy Programs
Computer Sciences Corporation

January 31, 1989

1 Introduction

A high dispersion line-by-line file, in the format of a rotated image, would allow more flexibility in image analysis to the scientist wishing to work with IUE spectral images than is currently available. This preliminary study will examine the question of how to produce a rotated line-by-line file to make the most effective use of specialized data extraction techniques such as Gaussian extraction.

To be practical for generating such files for all high dispersion images the method selected must meet certain criteria. The rotated data should closely resemble the input with minimum smoothing. The method should not require large amounts of computer time. The methods that will be tested and considered are discussed below, with an emphasis on Fourier and Fourier-like transformations.

There are many ways to generate a rotated image. Approaches which use interpolation or extrapolation presume that the data varies smoothly. In most instances this results in, at best, an average fit to the data, or, at worst, a smoothing of the features in the data. If the data really is already a smooth function, it is better approximated by a high order or stiff method, such as a spline, cubic spline, etc. Smoothing the data is sometimes desirable in spectral analysis. However, in this particular instance,

sampling an image which has already been sampled and re-sampled in the data reduction process cannot be considered a good thing. In particular, resampling images which have low signal-to-noise ratios would be detrimental to the data.

A function with sharp changes, such as an IUE spectral image, is less accurately approximated by stiff methods. Bilinear interpolation is a common choice, and a method that produces average results. From the class of polynomial fits, the Lagrange interpolation might be a good choice. The Lagrange formula is difficult to code and gives no error estimate. A better algorithm is Neville's algorithm; it does the same thing as Lagrange - constructing a unique interpolating polynomial. As before, one has to consider how well the function is approximated. LaGrangian interpolation can be described in vector-matrix format, and can be imbedded in some image transformation techniques (discussed later in this text). Rotations done in this format might preserve the structure of the data better than other standard interpolation techniques. For a discussion of Lagrange interpolation and Neville's algorithm, see Press, et al., 1986.

Rational function interpolation and extrapolation methods can be superior to polynomial interpolation/extrapolation because of their ability to model functions with poles. Rational function interpolation or extrapolation can give the capability of globally approximating the function in question. The Bulirsch-Stoer algorithm (Press, et al., 1986) performs rational function extrapolation on tabulated data. This algorithm produces a diagonal rational function. Its recurrence relation is analogous to a polynomial approximation. The main question here is how well can you rotate the data in this form, and still retain the data structure. The amount of computer time required for the above procedures, as well as the time needed to produce an equivalent wavelength matrix, has not yet been determined.

A better method, in theory, for sampling discrete or continuous data, is through image transformations, such as those used in digital signal processing. Fast Fourier Transforms, and other equivalent transforms, are capable of producing optimal results under practically any sort of initial criteria. One existing theory, the Nyquist sampling theory, says that if the sampling is done at the Nyquist Frequency, the function (or data) is completely determined. This is not necessarily true if the sampling is done in the spatial domain, but if the sampling is in the frequency domain with frequency sam-

pling, the structure of the data is well preserved. Interesting applications of Fourier Transform theory can be found in Electrical Engineering. A class of transforms called Discrete Orthogonal Transforms do FFTs in the form of equivalent vector-matrix transformations. This class of discrete transforms has some useful properties in the area of data (or matrix) manipulation.

Matrix factorization is a simple technique for developing FFT equivalents. These techniques can lead to methods of greatly reduced arithmetic operations. This depends on the number of points in the transform, the system of indexing, loading, and storing of the data, and whether the data is initially real or complex. IUE-specific data has several inherently positive points for developing an optimized method for image transformation. The fixed size and type of the data are but two of these. One initial test has been performed and shows that for a matrix 768 by 768 composed of integer data, a forward FFT in IDL took 50 CPU seconds on the VAX 8350. For a matrix 768 by 768 composed of real data, a forward FFT in IDL took three minutes. Inverse FFTs take similar times. The IDL FFT was the one currently available on the system. These times are not controlled benchmarks, but quick timing tests.

Orthogonal transformations which are equivalent to FFTs are widely used in engineering sciences. The analogy between digital signal processing and digital image processing is a strong one. Of particular interest here are some of the properties of orthogonal transformations, such as the fact that both symmetric and antisymmetric properties are preserved in the transform space, and periodicity is directly equivalent. Some of the byproducts of doing image transformations are interesting data products themselves, such as the power spectrum or the phase (position) spectrum. Another application of Fourier Transform methods is in the area of geometric corrections. An image rotation and geometric correction could be done at the same time using orthogonal transformation techniques, again under the initial criteria of not over-smoothing the data and not consuming huge amounts of computer time. However, these type of techniques would need extensive testing.

Matrix factorization gives us some unique tools to work with. This type of transform is a subset of a Fast Generalized Transform and Fast Generalized Transform matrices follow directly from that. See Appendix II for a discussion of a Discrete Fourier Transform in the form of a fast

Discrete Cosine Transform, and it's application to image rotations.

A Appendix I: A Short Derivation of a Fourier Transform

Fourier Transforms (FTs) are a general numerical and system analysis tool. Fourier Series are used to decompose periodic signals into the sum of sinusoids. The FT provides a frequency domain analysis of signals that can be represented by a series, or signals that have a continuous spectrum. The input function $x(t)$ is the periodic or driving function of a linear, time invariant system. Sinusoids have orthogonality properties - for example, over an interval from $-P/2 \leq t \leq P/2$, where P is the period, the exponential functions are also orthogonal. Since $x(t)$ can be represented by a Fourier expansion of

$$x(t) = \frac{a_0}{2} + \sum_{l=1}^{\infty} \left[\frac{a_L \cos 2\pi Lt}{P} + \frac{b_L \sin 2\pi Lt}{P} \right], \quad (1)$$

where P is the period, $L = 0, 1, 2, \dots$ integer number of cycles in P , L/P is the frequency, a 's and b 's are the Fourier series coefficients with $X(k)$ complex coefficients $k = \pm 1, \pm 2, \dots$, we can now give the evaluation formula, after some manipulation, as

$$X(k) = \frac{1}{P} \int_{-P/2}^{P/2} x(t) e^{-j2\pi kt/P} dt. \quad (2)$$

The derivation of the Fourier Transform begins by multiplying both sides by P so that

$$PX(k) = \int_{-P/2}^{P/2} x(t) e^{-j2\pi kt/P} dt. \quad (3)$$

If we then assume that the frequency approaches a continuous variable, we can define the frequency as

$$f = \lim_{P \rightarrow \infty} k/P. \quad (4)$$

If we then assume that the $P(X(k))$ term is meaningful for all P , we can say that

$$X(f) = \lim_{P \rightarrow \infty} P(X(k)). \quad (5)$$

Combining (3) and (5) leads to

$$X(f) = \int_{-\infty}^{\infty} x(t)e^{-j2\pi ft} dt \quad (6)$$

Equation (6) is then the Fourier Transform of $x(t)$. The function can be real or complex and will be called the spectrum in frequency space of the input signal(or data) $x(t)$. The signal can be recovered from it's spectrum $X(f)$ using the Inverse Fourier Transform(IFT). The derivation of this leads to the equation for $X(f)$

$$x(t) = \int_{-\infty}^{\infty} X(f)e^{j2\pi ft} df \quad (7)$$

The signal $x(t)$ recovered from it's spectrum $X(f)$ can be a real or complex function. The fact that it can be real will be an important consideration in terms of CPU time needed to implement a transformation.

When both integrals exist we say that $x(t)$ and $X(f)$ make up a Fourier Transform pair, represented by

$$x(t) \longleftrightarrow X(f) \quad (8)$$

A.1 Matrix Representation of a Discrete Fourier Transform

A Discrete Fourier Transform (DFT) can be represented by a matrix, and re-arrangement of this matrix results in matrix factorization leading to the FFT. The input to the DFT is the data sequence contained in vector x such that

$$x = [x(0), x(1), \dots, x(N - 1)]^T .$$

where T is the Transpose, N is the size of the array, in each case. The output of the DFT is the transform sequence contained in the vector X such that

$$X = [X(0), X(1), \dots, X(N - 1)]^T .$$

where T is the Transpose. All operations combine into a matrix form given by:

$$X = \left(\frac{1}{N} W^E \right) * x. \quad (9)$$

where W^E is the DFT matrix with row numbers $k = 0, 1, 2, \dots, N - 1$ and column numbers $n = 0, 1, 2, \dots, N - 1$, E is the matrix of exponents so that $W^{E(k,n)}$ is in row k and column n . Equation (9) is the vector-matrix equation for the DFT. Each entry is the product of the k^{th} value for the row and the n^{th} value for the column (computed mod n). The DFT matrix is a square matrix $N \times N$.

A.2 DFT Inversion- the IDFT

The DFT resulted from approximating an integral equation with a summation. The IDFT is also a summation. It is similar to the Fourier series representation of $x(t)$ given by (1) This is done with only the first N coefficients, because N points allow us to solve for N unknown transform sequence values. Equation (1) rewritten with N coefficients, $-N/2 \leq k < N/2$, gives

$$x(n) = \sum_{k=-N/2}^{N/2-1} X(k)e^{j2\pi kn/N}. \quad (10)$$

The IDFT is then

$$\begin{aligned} x(n) &= \sum_{k=0}^{N-1} X(k)e^{j2\pi kn/N} \\ &= \sum_{k=0}^{n-1} X(k)W^{-kn}, \text{ for } n = 0, 1, 2, \dots, N - 1, \end{aligned} \quad (11)$$

and

$$W^{-1} = \exp(j2\pi/N).$$

The N equations can be written in vector-matrix notation as

$$x = W^{-E} \times X \quad (12)$$

where x and X are vectors of data samples and DFT coefficients, W^{-E} is the IDFT matrix, $n = 0, 1, 2, \dots, N - 1$ are the row numbers, and $k = 0, 1, 2, \dots, N - 1$ are the column numbers, so that entry $W^{-E(n,k)}$ is in row n , column k .

A.3 The DFT and IDFT - Unitary Matrices

If an $N \times N$ matrix U is unitary (* indicates unitary), its inverse is the complex conjugate of U transposed

$$U^{-1} = (U^*)^T = (U^T)^*. \quad (13)$$

It can be shown that if the time and frequency tags are in natural order (the data sequencing numbers can be defined on all matrices in a natural order of $0, 1, 2, \dots, N-1$, and are often called time and frequency tags) then the scaled DFT matrix $W^{E(\sqrt{N})}$ and the scaled IDFT matrix $(W^E)^{-1/\sqrt{N}}$ satisfy unitary matrix conditions leading to

$$W^{-E} = ((W^*)^E)^T = ((W^E)^*)^T. \quad (14)$$

A fast and easy way to derive FFT algorithms is to manipulate W^E into a product of matrices. Where E is the corresponding exponential matrix that follows from (2) when converted to matrix notation. The factorization of W^E into sub-matrices W^{E_1} and W^{E_2} is such that only the nonzero entry per row of W^{E_2} is multiplied by a nonzero entry of a column of W^{E_1} . This is the row times column rule of matrix multiplication. The matrix multiplication becomes addition when applied to the exponents, and each entry in E is the sum of two exponents. The preceding result is true of all FFT matrices in their factored form. When both $x(n)$ and $X(k)$ are defined this way, they constitute a DFT pair.

$$x(n) \leftrightarrow X(k). \quad (15)$$

The notation $X(k) = DFT[x(n)]$ and $x(n) = IDFT[X(k)]$ means that the DFT and its inverse are defined by the N - point sequences $x(n)$ and $X(k)$ respectively. The utility of the DFT lies in its ability to estimate a spectrum using numerical methods. The DFT coefficients correspond to the spectrum determined using the Fourier Transform. The FFT uses a greatly reduced number of arithmetic operations and an easy way to visualize the procedure for generating an FFT equivalent algorithm results from matrix factorization, specifically, by reordering rows and/or columns of the DFT matrix W^E such that it factors into a product of matrices. The easiest case to work with is for $N = 2^L$ where L is the number of

integral factors of N . The more general case is for N having L integral factors such that $N = N_L, N_{L-1}, \dots, N_1$. FFTs for this case are called mixed- (or split-) radix transforms, and this particular class of transforms have some interesting applications in digital signal processing that shows some analogy to image processing. See Appendix II for a discussion of the split-radix Transform. Matrix manipulation methods used to develop FFT and IFFT algorithms include matrix transpose, using IFFTs to deduce FFTs, and inserting a factored identity matrix into a factored FFT. These additional FFT methods are developed for several purposes. They provide software or hardware engineers with enhanced flexibility in a particular application. They also show techniques to apply to the derivation of other FFTs and FFT-like Transforms.

A.4 Applications to Image Rotations

A sample image can be represented in the form of an N^2 -dimensional vector x_i , where x_{ij} would be the j -th component of the vector x_i . A Covariance matrix can then be defined consisting of the x vectors such that

$$C_x = E[(x - m_x)(x - m_x)'], \quad (16)$$

where $m_x = E[x]$ is the mean vector, E is the expected value, and the prime indicates transposition. The covariance equation can be approximated from the samples to an equivalent form

$$C_x = \frac{1}{M} \left[\sum_{i=1}^M x_i x_i' \right] - m_x m_x'. \quad (17)$$

The dimension of the mean vector is N^2 and C_x is an $N^2 \times N^2$ matrix. If e_i and λ_i are the eigenvectors and eigenvalues of C_x , then we can make an equivalent transformation matrix whose rows are the eigenvectors of C_x . Call this matrix A . We can then multiply a central image vector $(x - m_x)$ by A to make a new image vector y , such that

$$y = A(x - m_x). \quad (18)$$

The equivalent covariance matrix would then be

$$C_y = E[(y - m_y)(y - m_y)']. \quad (19)$$

It can be shown that C_y is a diagonal matrix with elements equal to the eigenvalues of C_x . This property is important, since the elements off the diagonal are zero, the y 's are uncorrelated. Also, each eigenvalue λ_i is equal to the variance of the i -th element of y along eigenvector e_i . In the Transform, if the choice of the basis equations (vectors) is such that these vectors all point in the direction of optimum variance of the data, then the bases are the eigenvectors of the covariance matrix. (All vectors must, of course, be mutually orthogonal and the transformed components be uncorrelated.) This required orientation of the eigenvectors leads to optimal properties needed for image rotations. If we consider the coordinate system z_1 and z_2 , it then follows from elementary trigonometry that the new and old axes are related by

$$z_1 = z_1 \cos\theta + z_2 \sin\theta \quad (20)$$

$$z_2 = -z_1 \sin\theta + z_2 \cos\theta. \quad (21)$$

The original coordinates of each pixel can then be interpreted as a two dimensional random variable with the mean

$$m_x = \frac{1}{P} \sum_{i=1}^P P x_i, \quad (22)$$

and the covariance matrix

$$C_x = \frac{1}{P} \left[\sum_{i=1}^P P x_i x_i' \right] - m_x m_x'. \quad (23)$$

Where P is the number of pixels in the object to be rotated and x_i is the vector composed of the coordinates of the i -th pixel. m_x is a two dimensional vector.

This method is subject to many interpretations (applications) which are not at all clear in this synopsis of the mathematics. It is merely an opinion that some method can be found that will be an optimum method specific to IUE's vidicon images. Since this is a preliminary report, much testing is yet needed.

See the references section of this paper for a list of the textbooks from which these condensed derivations were taken.

B Appendix II

In this appendix we will take a quick look at two examples of how to implement a fast transform. References are given for those interested in pursuing the topic further.

Example 1: The Fast Hartley Transform resulting from application of the mixed- (or split-)radix transform.

The fast Hartley Transform (FHT) method is a type suggested in Bracewell, 1965. This type of transform sets up the forward and inverse transforms in the same way. The FHT type of transform also employs only real values, which simplifies the computations. Others have established that a split-radix FHT is twice as fast as a split-radix FFT. A split-radix FHT was used to implement a discrete Hartley Transform and was shown to be faster than other known algorithms. This technique not only is faster to compute, but also results in improved accuracy over comparable techniques. Results from a paper by Agbinya, 1987, show that this method of doing a FHT leads to smaller mean square errors than the FFT. The sampling rate here is higher than the Nyquist rate. See the reference for the article by Agbinya, 1987 for a mathematical discussion of this example. For a general discussion on split-radix transforms see the reference of an article by Pei, 1986.

Example 2:

Discrete Cosine Transforms (DCT) have applications in image processing. The traditional method for implementing the DCT has been with complex arithmetic throughout. Researchers have reported recent successes in developing fast DCT algorithms which make this transform more useful than its traditional form. The faster methods for utilizing DCTs involve using real instead of complex arithmetic. The DCT and fast DCT equivalents are orthogonal transforms and compare well with the optimal orthogonal Karhunen-Loeve Transform criteria: a) that it will completely decorrelate any sequence in the transform domain. b) that it packs the most energy into a minimum number of transform coefficients. c) that the mean square error is minimized between the original data and the reconstructed data for any specific bandwidth reduction. d) that the total entropy of the sequence is also minimized.

For a complete discussion and derivation of discrete orthogonal trans-

forms, and their applications see the reference for Elliott and Rao, 1982.

This work was supported by NASA grant number NAS5 - 28749 to the Computer Sciences Corporation.

REFERENCES

- Bracewell, R. 1965 *The Fourier Transform and Its Applications* McGraw-Hill, Inc.
- Brigham, E. 1974 *The Fast Fourier Transform* Prentice-Hall, Inc.
- Castleman, K. 1979 *Digital Image Processing* Prentice-Hall, Inc.
- Elliott, D., Rao, K. 1982 *Fast Transforms Algorithms, Analyses, Applications* Academic Press, Inc.
- Gonzalez, R., Wintz, P. 1987 *Digital Image Processing* Addison, Wesley, Inc.
- Oppenheim, A., Schafer, R. 1975 *Digital Signal Processing* Rainbow Bridge Book Co., Ltd.
- Press, W., Flannery, B., et al. 1987 *Numerical Recipes* Cambridge University Press.
- Agbinya, J. 1987 "Fast Interpolation Algorithm Using Fast Hartley Transform" *Proceedings of the IEEE* Vol. 75, No 4.
- Ahmed, N., et al. 1974 "Discrete Cosine Transform" *IEEE Transactions on Computers* pp. 90 - 93.
- Chen, W., et al. 1977 "A Fast Computational Algorithm for the Discrete Cosine Transform" *IEEE Transactions on Computers* pp. 186 - 192.
- Kamangar, F., Rao, K. 1982 "Fast Algorithms for the 2-D Discrete Cosine Transform" *Proceedings from the IEEE* Vol. 62, No. 6.
- Pei, S.C., Wu, J.L. 1986 "Split-radix fast Hartley transform" *Electron Letters* pp. 26 - 27.