WHAT IS FFT ? (PART I)

Moo K. CHUNG
Chung@math.toronto.edu

§ 1. Introduction.

A digital image is a function of two variables.

$$f : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$$

At each pixel $(i,j) \in \mathbb{Z}^2$, its grey-value is denoted by $f(i,j)$.

In practice,

$$f : \{0, \cdots, 2^{n+1}\}^2 \to \{0, \cdots, 255\}$$

Images can be implemented as matrices. Typically, an image is blocked into sub $2^m \times 2^m$ square blocks, and each block is transformed in such a manner that allows compression of the number of data entries. Compression is lossless if there is no loss of information and inversion yields the exact image that was compressed. If some amount of information is lost upon compression, then the coding is lossy and the image recovered by inversion will be an estimate of the original.

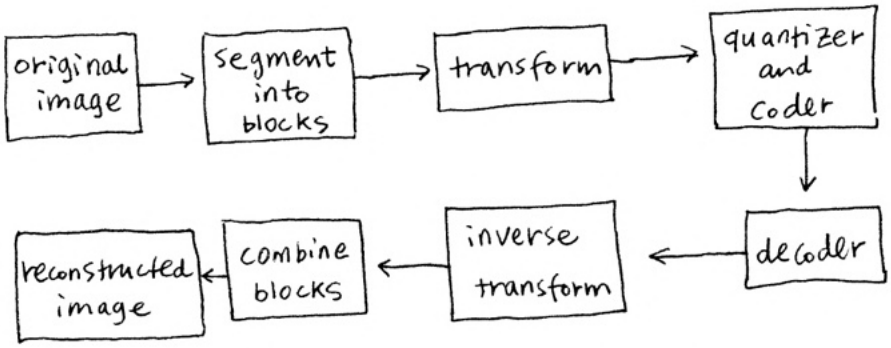

Fig. 1.1. Diagram for transform coding

§2. Hadamard Transform

An $N \times N$ Hadamard matrix H_N can be defined recursively by the Kronecker product.

$H_N = H_2 \otimes H_{N/2}$, where recursion is initiated by

$$H_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Thus

$$H_N = \frac{1}{\sqrt{2}} \begin{bmatrix} H_{N/2} & H_{N/2} \\ H_{N/2} & -H_{N/2} \end{bmatrix} \quad , \quad N = 4, 8, \ldots$$

It is possible to generate the terms of the Hadamard matrix in an alternate method so called Paley construction which produces Hadamard matrices of order $p+1$ where p is any odd prime such that $p+1$ is a multiple of 4. For details see [1] Barnett.

For $N = 4$,

$$H_4 = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & -1 \\ 1 & -1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

The rows of H_N form a basis set of orthogonal vectors.

H_N is orthogonal and symmetric

$$H_N H_N^T = H_N^T H_N = I_N$$

This can be checked easily by induction on N

Since $H_N = H_N^T$,

$$H_N H_N = I_N$$

As an example, let $X^T = (12, 10, 6, 4)$. Then

$$H_4 X = (16, 6, 0, 2)^T$$

Compression is achieved by not transmitting X but transmitting only the lower components of $H_4 X$. The compression is lossy in this example. If X_0 is obtained from $H_4 X$ by setting the last two components of HX equal to 0, then only the first two components of X_0 needs to be transmitted.

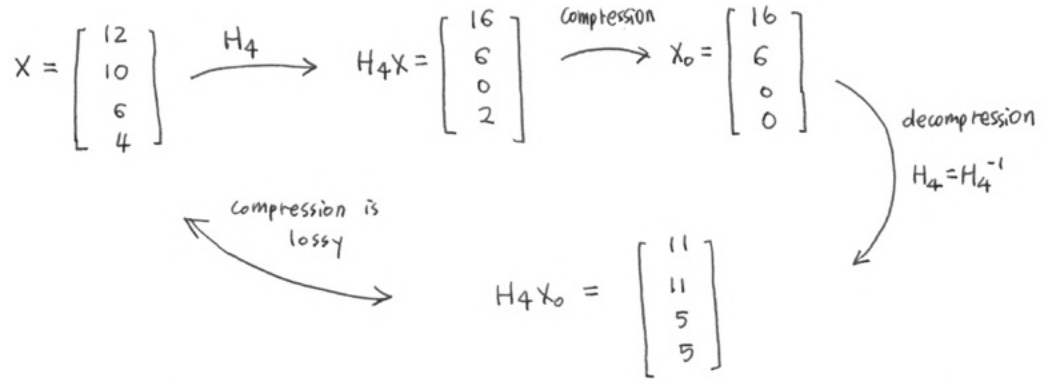

Fig 2.1. Compression by H_4

Adaptation of Hadamard compression to images is accomplished by blocking the image into 2^N by 2^N blocks and applying the transform to each block. If B_N is such a block and H_N is the Hadamard matrix of dimension 2^N, then the transformed block is

$$B_N' = H_N B_N H_N^T = H_N B_N H_N$$

Inversion is given by

$$B_N = H_N^T B_N' H_N = H_N B_N' H_N$$

For images, compression is accomplished by keeping only some set of upper left components of B_N'. If B_N'' is such a matrix, decompressed image is given by

$$H_N B_N'' H_N$$

which is slightly different from the original image $B_N = H_N B_N' H_N$.

An advantage of the Hadamard transform over other transform is that multiplication by the Hadamard matrix can be done without arithmetic multiplications. It requires only addition and subtraction. However, the amount of computation can be further reduced by factorizing H_N into the matrix product of a set of sparse matrices. A fast Hadamard transform is achieved by such factorization.

$$H_8 = A \cdot A \cdot A$$

, where

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix}$$

Applying H_8 directly requires 49 additions or subtractions. But applying A requires 8 additions or subtractions. So the factorization results in a total of 24 additions or subtractions. Generally, for H_N, the factorization requires $N \log_2 N$ additions or subtractions.

§.3. Discrete Fourier Transform

Given an N-th order sequence of values $\{f_0, f_1, \ldots, f_{N-1}\}$, the corresponding Nth order discrete fourier transform is the sequence $\{F_0, F_1, \ldots, F_{N-1}\}$ given by

$$F_j = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} f_k \, e^{-2\pi i k j / N} \quad , \quad j = 0, \ldots, N-1$$

The inverse transform is given by

$$f_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} F_j \, e^{2\pi i k j / N} \quad , \quad k = 0, \ldots, N-1$$

Define the weighting kernel W_N as

$$W_N = e^{2\pi i / N}$$

Then

$$F_j = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} f_k \, W_N^{-kj} \quad ,$$

$$f_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} F_j \, W_N^{kj}$$

Let $f = (f_0, f_1, \ldots, f_{N-1})^T$ and

$F = (F_0, F_1, \ldots, F_{N-1})^T$. Then the $N \times N$ Fourier matrix $[\mathcal{F}_N]$ is defined by

$$F = [\mathcal{F}_N] f$$

Note that

$$[\mathcal{F}_N] = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W_N^{-1} & W_N^{-2} & \cdots & W_N^{-(N-1)} \\ 1 & W_N^{-2} & W_N^{-4} & \cdots & W_N^{-2(N-1)} \\ \vdots & \vdots & & \ddots & \vdots \\ 1 & W_N^{-(N-1)} & W_N^{-2(N-1)} & \cdots & W_N^{-(N-1)^2} \end{bmatrix} = \frac{1}{\sqrt{N}} \left[W_N^{-(r-1)(s-1)} \right], \; r, s = 1, \ldots, n$$

Since $W_N^N = 1$, it follows that there are only N distinct elements in $[F_N]$. For example

$$[F_4] = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 \\ 1 & \omega^2 & 1 & \omega^2 \\ 1 & \omega^3 & \omega^2 & \omega \end{bmatrix}, \quad \text{where} \quad \omega = W_4^{-1} = e^{-\pi i/2}$$

$N \times N$ Fourier matrix has many interesting properties. See [1], Barnett for details. The matrix is a particular case of the Vandermonde matrix. If $N \times N$ circulant matrix is defined by

$$C_N = \text{circ}(c_1, c_2, \ldots, c_N) = \begin{bmatrix} c_1 & c_2 & c_3 & \cdots & c_N \\ c_N & c_1 & c_2 & \cdots & c_{N-1} \\ c_{N-1} & c_N & c_1 & \cdots & c_{N-2} \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ c_2 & c_3 & c_4 & \cdots & c_1 \end{bmatrix}$$

Then $[F_N]$ diagonalizes C_N. The explicit diagonalization is given by

$$[F_N]\, C_N\, [F_N]^* = \text{diag}\left[c(1), c(W_N), c(W_N^2), \ldots, c(W_N^{N-1})\right]$$

, where $c(\lambda) = c_1 + c_2\lambda + \cdots + c_N\lambda^{N-1}$

and $[F_N]^*$ is the conjugate transpose of $[F_N]$

$[F_N]$ is unitary and symmetric

$$[F_N][F_N]^* = [F_N]^*[F_N] = I_N$$

For images, blocking is done as in the case of the Hadamard transform. For a block B_N, the forward and inversion transform is given by

$$B_N' = [F_N]\, B_N\, [F_N]^T = [F_N]\, B_N\, [F_N]$$

$$B_N = [F_N]^*\, B_N'\, [F_N]^*$$

Compression is achieved by only taking the lower-order spatial frequencies in the upper left corner of B'. As in the case of the Hadamard transform, there is a <u>fast Fourier transform</u> (FFT) factorization. For $N = 8$, ignoring the $\sqrt{N}$ scaling factor DFT matrix has the factorization

$$\sqrt{8}\,[\,\mathcal{F}_8\,] = A_3 \cdot A_2 \cdot A_1$$

, where $A_1 = \begin{bmatrix} I_4 & I_4 \\ I_4 & -I_4 \end{bmatrix}$

$$A_2 = \begin{bmatrix} I_2 & I_2 & 0 & 0 \\ I_2 & -I_2 & 0 & 0 \\ 0 & 0 & I_2 & W_8^{-2} I_2 \\ 0 & 0 & I_2 & -W_8^{-2} I_2 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & W_8^{-2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -W_8^{-2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & W_8^{-1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -W_8^{-1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & W_8^{-3} \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -W_8^{-3} \end{bmatrix}$$

Given an input block B_8, this composition results in the computation

$$B_8 \rightarrow A_1 B_8 \rightarrow A_2 A_1 B_8 \rightarrow A_3 A_2 A_1 B_8$$

In comparison to the original DFT matrix, there are fewer multiplications.

§4. Cooley-Tukey FFT Algorithms

In 1965 J.W. Tukey and J.W. Cooley published an algorithm, which is considered one of the most important in numerical analysis. See [3] Cooley for the original paper.

$$F_j = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} f_k W_N^{-kj}$$

$$= \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} f_k W_N^{-k(j+N)}$$

$$= F_{j+N}$$

F_j was defined for $j=0,\dots,N-1$ but we can extend the definition to $\forall j \in \mathbb{Z}$. Also f_k can be extended for $\forall k \in \mathbb{Z}$.

$$f_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} F_j W_N^{kj}$$

$$= \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} F_j W_N^{(k+N)j}$$

$$= f_{k+N}$$

Hence F_j and f_k are periodic with periodicity N.

Let $N = 2M$, $M \in \mathbb{N}$.

Define

$$f^1_k = f_{2k},$$

$$f^2_k = f_{2k+1} \qquad k=0,\dots,M-1$$

Then

$$f^1_{k+M} = f_{2(k+M)} = f_{2k+N} = f_{2k} = f^1_k$$

$$f^2_{k+M} = f^2_k$$

Since $\{f_k^1\}$ and $\{f_k^2\}$ are Mth order periodic sequences with periodicity M, we can determine their discrete Fourier transforms:

$$F_j^1 = \frac{1}{\sqrt{M}} \sum_{k=0}^{M-1} f_k^1 W_M^{-kj}$$

$$F_j^2 = \frac{1}{\sqrt{M}} \sum_{k=0}^{M-1} f_k^2 W_M^{-kj} \qquad j = 0, \ldots, M-1$$

The discrete Fourier transform of the Nth order sequence $\{f_k\}$:

$$F_j = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} f_k W_N^{-kj}$$

$$= \frac{1}{\sqrt{N}} \sum_{k=0}^{M-1} f_{2k} W_N^{-2kj} + \frac{1}{\sqrt{N}} \sum_{k=0}^{M-1} f_{2k+1} W_N^{-(2k+1)j}$$

Note that

$$W_N^{-2kj} = e^{-2\pi i/N \cdot 2kj} = W_M^{-kj}$$

$$W_N^{-(2k+1)j} = W_M^{-kj} W_N^{-j}$$

Therefore,

$$F_j = \frac{1}{\sqrt{N}} \sum_{k=0}^{M-1} f^1 W_M^{-kj} + \frac{W_N^{-j}}{\sqrt{N}} \sum_{k=0}^{M-1} f^2 W_M^{-kj}$$

$$= \frac{1}{\sqrt{2}} F_j^1 + \frac{W_N^{-j}}{\sqrt{2}} F_j^2$$

Also

$$F_{j+M} = \frac{1}{\sqrt{2}} \left(F_j^1 - F_j^2 W_N^{-j} \right) \qquad , \; j = 0, \ldots, M-1$$

To compute the discrete Fourier transform of $\{f_k\}$ requires $2N^2 - N$ complex operations, whereas it has been reduced to $N^2 + N$ complex operations in Cooley-Tukey FFT algorithm.

What is FFT ? (PART II)

Moo K. Chung
Chung @ math.toronto.edu

§5. Asymptotic Behavior of Cooley-Tukey FFT

For a single value of j,

$$F_j = \frac{1}{\sqrt{N}} \left(f_0 \cdot W_N^{-j} + \cdots + f_{N-1} \cdot W_N^{-(N-1)j} \right)$$

we can disregard the scaling factor $\sqrt{N}$. The numbers $W_N^{-j}, \ldots, W_N^{-(N-1)j}$ are given. Let $\phi_0(X)$ the number of complex operations required to evaluate X. To evaluate F_j, N complex multiplications and $N-1$ complex additions. Hence,

$$\phi_0(F_j) = 2N-1 \qquad j = 0, 1, \ldots, N-1$$

Since $\phi_0(F_0) = \phi_0(F_1) = \cdots = \phi_0(F_{N-1})$,

$$\phi_0(\{F_j\}) = 2N^2 - N$$

If we divide $\{f_k\}$ into even and odd sequences and apply Cooley-Tukey algorithm, then we can reduce the number of complex operations by almost half when N gets large.

$$F_j = \frac{1}{\sqrt{2}} \left(F_j^1 + F_j^2 \cdot W_N^{-j} \right)$$

$$F_{j+\frac{N}{2}} = \frac{1}{\sqrt{2}} \left(F_j^1 - F_j^2 \, W_N^{-j} \right) \qquad j = 0, \ldots, \frac{N}{2}-1$$

Let ϕ_1 be the number of complex operations required when Cooley-Tukey algorithm is applied once. Then

$$\phi_1(\{F_j\}) = \phi_0(\{F_j^1\}) + \phi_0(\{F_j^2\}) + 2N$$

$2N$ comes from the fact that there are N complex multiplications and N complex additions. Since $\{F_j^1\}$ is $\frac{N}{2}$th order DFT

$$\phi_0(\{F_j^1\}) = \phi_0(\{F_j^2\}) = 2\left(\frac{N}{2}\right)^2 - \left(\frac{N}{2}\right)$$

$$= \frac{N^2-N}{2}$$

Hence

$$\phi_1(\{F_j\}) = N^2 + N$$

Comparing with $\phi_0(\{F_j\}) = 2N^2 - N$, we see that the number of complex operation has been decreased almost by half.

In general, if N is divisible by 2^p for some $p \in \mathbb{N}$, then divide $\{f_k\}$ into 2^p subsequences of the same order $\frac{N}{2^p}$ and apply Cooley-Tukey FFT recusively p times.

For instance when $\{f_k\}$ is 8th order sequence, we can use Cooley-Tukey FFT 3 times. (See. Fig. 5.1.). The recursion is initiated by applying Cooley-Tukey FFT to 2nd order sequences. For $\{f_0, f_1\}$, DFT by Cooley-Tukey FFT is given by

$$F_0 = \frac{1}{\sqrt{2}}\left(f_0 + f_1 \cdot W_2^0\right)$$

$$F_1 = \frac{1}{\sqrt{2}}\left(f_0 - f_1 \cdot W_2^{-1}\right)$$

Notice that these equations are exactly DFT of $\{f_0, f_1\}$.

So for the 2nd order sequence $\{f_0, f_1\}$

$$\phi_1(\{F_0, F_1\}) = \phi_0(\{F_0, F_1\})$$

Therefore, when $\{f_k\}$ is 2^p th order sequence, $\{F_j\}$ can be computed with Cooley-Tukey FFT algorithm only without invoking DFT.

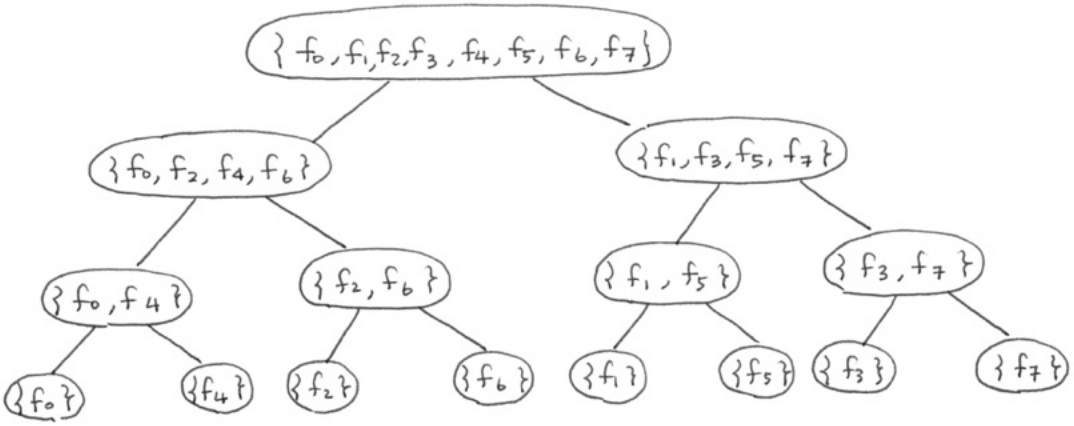

Fig. 5.1. Cooley-Tukey FFT for 8th order sequence

Let ϕ_p be the number of complex operations required when Cooley-Tukey FFT is used p times recursively. Let N be divisible by 2^p but not by 2^{p+1}. Then

$$\phi_p(\{F_0, \ldots, F_{N-1}\}) = \phi_{p-1}(\{F_0, F_2, \ldots, F_{N-2}\}) + \phi_{p-1}(\{F_1, F_3, \ldots, F_{N-1}\}) + 2N$$

$$= 2 \cdot \phi_{p-1}(\{F_0, F_2, \cdots, F_{N-2}\}) + 2N$$

Solving the recurrence relation, we get

$$\phi_p(\{F_0, \ldots, F_{N-1}\}) = \frac{N^2}{2^{p-1}} + (2p-1)N$$

We can check the above formular by induction on p. Note that when we increase p from $p+1$, we are also increasing N to $2N$.

When $N = 2^p$, Cooley-Tukey algorithm goes to complete reduction with

$$\phi_p = N + 2N \cdot \log_2 N$$

complex operations.

As N becomes large

$$\frac{\phi_p}{\phi_0} = \frac{N + 2N \cdot \log N}{2N^2 - N} = \frac{1 + 2\log_2 N}{2N - 1} \to \frac{\log_2 N}{N}$$

Hence

$$\frac{\phi_p}{\phi_0} = O\left(\frac{\log_2 N}{N}\right)$$

and $\phi_p = O(N \log_2 N)$

Although most sequences do not have such a convenient number of terms with $N = 2^p$, we can always artificially add zeros to the end of the sequence to reach such a value. This extra number of terms in the sequence is more than compensated by the tremendous savings afforded by Cooley-Tukey algorithm.

We have considered an algorithm for computing DFT of an Nth order sequence that contained an even number of terms. We can develope a similar FFT algorithm when N is divisible by 3^P. The number of complex operations required in this case is given by

$$\frac{\phi_p}{\phi_0} = O\left(\frac{3 \log_3 N}{N}\right)$$

, see, [2] Weaver

In an analogous way it is possible to generate DFT based on any prime number p_n.

It is also possible to combine these various methods into one algorithm. Suppose

$$N = 2^{q_1} \cdot 3^{q_2} \cdot 5^{q_3} \cdot 7^{q_4} \cdots p_n^{q_n}$$

To digitally obtain DFT of this Nth order sequence, subdivide the sequence q_n times using radix p, then q_{n+1} times using radix p_{n+1} and continue until q_1 times using radix 2.

This method is called mixed radix fast transform method

There is an iterative FFT implementation based on bit reversal. see [6], Cormen. Bit reversal of binary number $(b_k b_{k-1} \ldots b_0)_2$ is defined as $(b_0 b_1, \ldots, b_k)_2$

References

[1] Barnett, Matrices: Methods and Applications, 1990, Oxford Univ. Press

[2] Weaver H.J., Applications of Discrete and Continuous Fourier Analysis, 1983, John Wiley & Sons, Inc.

[3] Cooley, J.W. and J.W. Tukey, "An Algorithm for the Machine Calculation of Complex Fourier Series", Math. Computations 19, April, 1965

[4] Tolimieri, Algorithms for Discrete Fourier Transform and Convolution, 1989, Springer-Verlag

[5] Wade G, Signal Coding and Processing, 2nd. ed., 1994, Cambridge Univ. Press.

[6] Cormen, Leiserson, Rivest, Introduction to Algorithms, 1990, MIT Press