Mathematical Theory and Modeling
ISSN 2224-5804 (Paper) ISSN 2225-0522 (Online)
Vol.3, No.6, 2013-Selected from International Conference on Recent Trends in Applied Sciences with Engineering Applications
www.iiste.org

The Fast Fourier Transform Algorithm and Its Application in Digital Image Processing

S.Arunachalam(Associate Professor)
Department of Mathematics, Rizvi College of Arts, Science & Commerce, Bandra (West), Mumbai – 400 050
Dr.S.M.Khairnar
Professor and Head, Department of Mathematics
MIT Academy of Engineering, Alandi, Pune-411 105
Dr.B.S.Desale
Department of Mathematics, North Maharashtra University, Jalgaon – 425 001

Abstract
Transforms are new image processing tools that are being applied to a wide variety of image processing problems. Fourier Transform and similar frequency transform techniques are widely used in image understanding and image enhancement techniques. Fast Fourier Transform (FFT) is the variation of Fourier transform in which the computing complexity is largely reduced. FFT is a mathematical technique for transforming a time domain digital signal into a frequency domain representation of the relative amplitude of different regions in the signal. The objective of this paper is to develop FFT based image processing algorithm. FFT can be computed faster than the Discrete Fourier Transform (DFT) on the same machine.
Key words: Fast Fourier Transform, Discrete Fourier Transform, Radix-2 FFT algorithm, Decimation in Time FFT, Time complexity.

1. Introduction:

DFT finds wide applications in linear filtering, correlation analysis and spectrum/transform analysis. Some special algorithms are developed for the easy implementation of DFT which result in saving of considerable computation time. Such algorithms are called FFT. By divide-and-conquer approach, the DFT which has a size N, where N is a composite number is reduced to the smaller DFT and computation is performed [1]. The computational algorithms are developed when the size of N is power of 2 and power of 4.
The FFT is used for the processing of images in its frequency domain rather than spatial domain. FFT is an important image processing tool which is used to decompose an image into its sine and cosine components [3]. The output of the transformation represents the image in the frequency domain, while the input image is the spatial domain equivalent. In the frequency domain image, each point represents a particular frequency contained in the spatial domain image. In this paper, to develop an algorithm to compute the FFT more efficiently and reduce the time it takes for calculation. This mathematical transform makes processing of images with larger data size practical. This paper is organized as follows: Section 2 describes DFT, Section 3 presents FFT, Section 4 shows the FFT algorithm, Radix-2 FFT algorithm, Radix-2 Decimation in Time FFT algorithm and Example, Section 5 describes the 2-D FFT in Image processing and Section 6 concludes the paper.

2. Discrete Fourier Transform (DFT):

The digital version of the Fourier transform is used in digital image processing [5] and it is referred as DFT. The one-dimension of DFT is given by

$$F(u) = \frac{1}{N}\sum_{x=0}^{N-1} f(x)e^{-j2\pi ux/N} \quad for\ u = 0,1,2,3,,\ldots\ldots\ldots\ldots\ldots,N-1 \qquad (1)$$

and the inverse DFT is given by

$$f(x) = \sum_{u=0}^{N-1} F(u)e^{j2\pi ux/N} \quad for\ x = 0,1,2,3,,\ldots\ldots\ldots\ldots\ldots,N-1 \qquad (2)$$

The DFT and inverse DFT is the foundation for the most frequency based image processing
Similarly, the DFT for two variables is called two-dimensional DFT and is given by

$$F(u,v) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi(ux/M+vy/N)} \qquad (3)$$

$for\ u = 0,1,2,\ldots\ldots\ldots,M-1\ and\ v = 0,1,2,\ldots\ldots\ldots,N-1.$
where $f(x,y)$ is a digital image of size M×N.
The inverse DFT is given by

$$f(x,y) = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)e^{j2\pi(ux/M+vy/N)} \qquad (4)$$

$for\ x = 0,1,2,\ldots\ldots\ldots,M-1\ and\ y = 0,1,2,\ldots\ldots,N-1.$

The DFT is frequently evaluated for each data sample and can be regarded as extracting particular frequency components from a signal.

3. The Fast Fourier Transform (FFT):

The FFT is an efficient implementation of DFT and is used in digital image processing. FFT is applied to convert an image from the spatial domain to the frequency domain. Applying filters to images in frequency domain is computationally faster than to do the same in the spatial domain [4].

The computation cost of the DFT is very high and hence to reduce the cost, the FFT was developed. With the introduction of the FFT the computational complexity is reduced from N^2 to $\log_2 N$. For instance, for an image of size 256×256 pixels the processing time required is about two minutes on a general purpose computer. The same machine would take 30 times longer (60 minutes) to compute the DFT of the same image of size 256×256 [7].

In equation (1), for each of the N values of u, the expansion of the summation requires N complex multiplication of $f(x)$ by $e^{-j2\pi ux/N}$ and $N-1$ additions give the Fourier coefficient $F(0)$.

For the computation of N Fourier coefficients, the number of complex multiplications and additions required is proportional to N^2. The computational complexity in the implementation of equation (1) can be reduced from N^2 to $N\log_2 N$ by a decomposition procedure. This procedure is called FFT algorithm.

For example, when N = 512, the direct DFT computational complexity proportional to
$N^2 = 262144$, whereas the FFT computational complexity is proportional to $N\log_2 N = 2048$.
This means FFT is 32 times faster than DFT. [262144/2048 = 32].

4. FFT Algorithm:

For spectral analysis of discrete signals, DFT approach is a very straightforward one. For larger values of N, DFT becomes tedious as it requires the huge number of mathematical operations to be performed [8]. The computational problem for the DFT is to compute the sequence {X(k)} of N complex-valued numbers given another sequence of data {X(n)} of length N, according to the formula

$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn} \quad for\ 0 \le k \le N-1$ (5)

Where $W_N = e^{-2\pi j/N}$ (6)

In general, the data sequence $x(n)$ is also assumed to be complex-valued.

To improve the efficiency in computing the DFT, some properties of W_N^{kn} are exploited [3].

Symmetric property: $W_N^{nk+N/2} = -W_N^{nk}$ (7)

Periodicity property: $W_N^{nk} = W_N^{nk+N} = W_N^{nk+2N} = \cdots$ (8)

Recursion property: $W_{N/2}^{nk} = W_N^{2nk}$ (9)

These properties arise from the graphical representation of the twiddle factor or phase factor $W_N^{nk} = e^{-2\pi jnk/N}$ by the rotational vector for each nk value.

Consider the following DFT where N = 8.

$X(k) = \sum_{n=0}^{7} x(n) e^{-jk2\pi n/8} \quad for\ k = 0,1,2,3,4,5,6,7$ (10)

Substituting $K = (\frac{k2\pi}{8})$ in equation (7), we obtain

$$X(k) = x(0)e^{-jK(0)} + x(1)e^{-jK(1)} + x(2)e^{-jK(2)} + x(3)e^{-jK(3)} + x(4)e^{-jK(4)} + x(5)e^{-jK(5)} + x(6)e^{-jK(6)} + x(7)e^{-jK(7)} \ldots\ldots\ldots (11)$$

Equation (11) has eight terms on the right hand side in which each term contains a multiplication of a real term with complex exponential.

For example, $x(1)e^{-jK(1)} = x(1)[cosK(1) - jsinK(1)]$ requires two multiplications and one addition for each value of K where $K = (\frac{k2\pi}{8})$, $k = 0,1,2,\ldots\ldots,7$.

In equation (11), each term in the right hand side requires eight complex multiplications and seven additions. The 8-point DFT therefore requires $8\times8 = 8^2 = 64$ complex multiplications and $8\times7 = 8(8-1) = 56$ additions.

In general, for an N-point DFT, N^2 multiplications and N (N – 1) additions are required. The algorithm developed by J.W. Cooley and J.W. Tukey in 1965 is the most efficient algorithm.

4.1 Radix-2 FFT Algorithm:

For efficient computation of DFT several algorithms have been developed based on divide and conquer methods. The method is applicable for N being a composite number [2].

Consider the case when $N = r_1 r_2 \ldots\ldots\ldots\ldots r_v$ where the $\{r_j\}$ are prime.

If $r_1 = r_2 = \cdots\ldots\ldots\ldots\ldots\ldots\ldots = r$ then $N = r^v$. The DFT are of size r. The number r is called the radix of the FFT algorithm. The most widely used FFT algorithms are radix-2 and radix-4 algorithms. For performing radix-2 FFT, the value of $N = 2^m$. Here the decimation can be performed m times where $m = \log_2 N$.

In direct computation of N-point DFT, the total number of complex additions are N (N – 1) and total number of complex multiplications are N^2. In radix-2 FFT, the total number of complex additions are reduced to $N\log_2 N$

and total number of complex multiplications are reduced to (N/2) $\log_2$N. Comparison of number of computations by DFT and FFT is shown in Table 1.

Table 1: Comparison of number of computations by DFT and FFT.

No. of points N	DFT Direct Computation		Radix-2 FFT	
	Addition N(N-1)	Multiplication N^2	Addition N$\log_2$N	Multiplication (N/2) $\log_2$N
4	12	16	8	4
8	56	64	24	12
16	240	256	64	32
32	992	1024	160	80
64	4032	4096	384	192
128	16256	16384	896	448
256	65280	65536	2048	1024

For example, let us calculate the percentage saving in calculations of N = 1024 point radix-2 FFT when compared to direct DFT.

Direct computation of DFT:

Number of complex additions = N (N – 1) = 1024 × (1024 – 1) = 1047552

Number of complex multiplications = N^2 = 1024 × 1024 = 1048576

Radix-2 FFT:

Number of complex additions = N $\log_2$N = 10240

Number of complex multiplications = (N/2) $\log_2$N = 5120

Percentage saving in additions = $100 - [\frac{Number\ of\ additions\ in\ radix-2\ FFT}{Number\ of\ additions\ in\ direct\ DFT}] \times 100$

$= 100 - [\frac{10240}{1047552}] \times 100 = 99.02\%$

Percentage saving in multiplications = $100 - [\frac{Number\ of\ multiplications\ in\ radix-2\ FFT}{Number\ of\ multiplications\ in\ direct\ DFT}] \times 100$

$= 100 - [\frac{5120}{1048576}] \times 100 = 99.51\%$

Hence, we observe that FFT algorithm reduces the number of complex multiplication and complex addition operation and computation time required to compute DFT.

4.2 Radix-2 Decimation in Time (DIT) FFT Algorithm:

The basic idea of the FFT is to decompose the DFT of a time domain sequence of length N into successively smaller DFTs whose calculations require less arithmetic operations [4]. This is known as a divide-and-conquer strategy. Decomposition into shorter DFTs may be performed by splitting an N-point input data sequence $x(n)$ into two (N/2)-point data sequences $a(m)$ and $b(m)$ corresponding to the even-numbered and odd-numbered samples of $x(n)$ respectively. That is,

$a(m) = x(2m)$, that is, sample of $x(n) for\ n = 2m$

$b(m) = x(2m+1)$, that is, sample of $x(n) for\ n = 2m+1, where\ 0 \le m < N/2$

This process of splitting a time domain sequence into even and odd samples is called decimation in time algorithm. Thus $a(m)$ and $b(m)$ are obtained by decimating $x(n)$ by a factor of 2, hence the resulting FFT algorithm is called radix-2 FFT algorithm. This is the simplest and most common form of the Cooley-Tukey algorithm.

Now the N-point DFT can be expressed in terms of DFTs of the decimated sequence as follows:

$$X(k) = \sum_{n=0}^{N-1} x(n)\, W_N^{nk}$$

$$= \sum_{m=0}^{N/2-1} x(2m)\, W_N^{2mk} + \sum_{m=0}^{N/2-1} x(2m+1)\, W_N^{(2m+1)k}$$

$$= \sum_{m=0}^{N/2-1} x(2m)\, W_N^{2mk} + W_N^k \sum_{m=0}^{N/2-1} x(2m+1)\, W_N^{(2m)k} \quad (12)$$

Using equation (9) in equation (12), we obtain

$$= \sum_{m=0}^{N/2-1} x(2m)\, W_{N/2}^{mk} + W_N^k \sum_{m=0}^{N/2-1} x(2m+1)\, W_{N/2}^{mk}$$

$$X(k) = A(k) + W_N^k B(k), 0 \le k < n \quad (13)$$

These two summations represent the (N/2)-point DFTs of the sequence $a(m)$ and $b(m)$ respectively. Thus DFT $[a(m)] = A(k)$ for even-numbered samples and DFT $[b(m)] = B(k)$ for odd-numbered samples. The outputs for $\frac{N}{2} \leq k < N$ from a DFT of length N/2 are identical to the outputs for $0 \leq k < N/2$ by using equation (8).

That is, $A\left(k+\frac{N}{2}\right) = A(k)\ and\ B\left(k+\frac{N}{2}\right) = B(k)\ for\ 0 \leq k < N/2$

By symmetrical property equation (7), $W_N^{k+N/2} = -W_N^k$

Thus, the whole DFT can be calculated as follows:

$$X(k) = A(k) + W_N^k B(k), \quad 0 \leq k < N/2$$

$$X(k+N/2) = A(k) - W_N^k B(k), \qquad 0 \leq k < \frac{N}{2} \qquad (14)$$

This result, expressing the DFT of length N recursively in terms of two DFTs of size N/2 is the core of the radix-2 DIT FFT. The final outputs of $X(k)$ are obtained by a combination of $A(k) and\ W_N^k B(k)$ which is simply a size 2 DFT. These combinations can be demonstrated by a simply oriented graph, called a butterfly graph. In each butterfly one complex multiplication and two complex additions are performed [6].

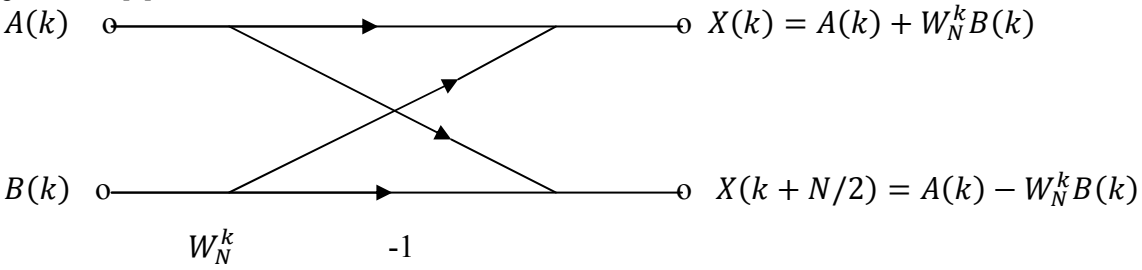
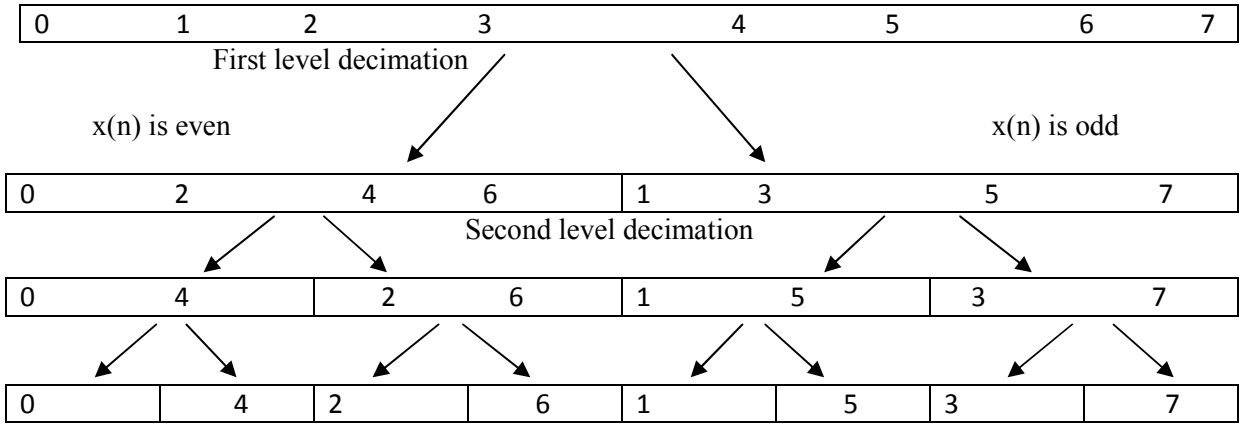


Figure.1 Butterfly computation in the DIT FFT Algorithm.

4.3 8-point DFT Using Radix-2 DIT FFT:

Consider the computation of an N = 8 point DFT. Here N = 8 = 2^3, and therefore the number of stages of computation is equal to 3. The given 8-point sequence is decimated to 2-point sequences. For each 2-point sequence, the 2-point DFT is computed. From the result of 2-point DFT, 4-point DFT can be computed. From the result of 4-point DFT, 8-point DFT can be computed. The block diagram of radix-2 DIT FFT for N = 8 is shown in Figure 2.

Before decimation the sequence are arranged in bit reversal order. Consider the sequence $x(n) = \{0,1,2,3,4,5,6,7\}$. In the first level decimation we have the sequence $x(0), x(2), x(4), x(6), x(1), x(3), x(5), x(7)$ and in the second level decimation, the sequence is $x(0), x(4), x(2), x(6), x(1), x(5), x(3), x(7)$.



One signal of 8-points:	0	1	2	3	4	5	6	7
Eight signals of 1-point:	0	4	2	6	1	5	3	7

Figure.2 Eight-point sequence decimation by Radix-2.

The shifting of the input data sequences is to be arranged in well-defined order. The index n of $x(n)$ is expressed in binary form. The data point $x(4)$ is expressed in binary form as $x(100)$ and is placed in position m

= 001 or m = 1 in the decimal array. The data $x(n)$ after decimation is stored in bit reversed order as shown in Figure.3

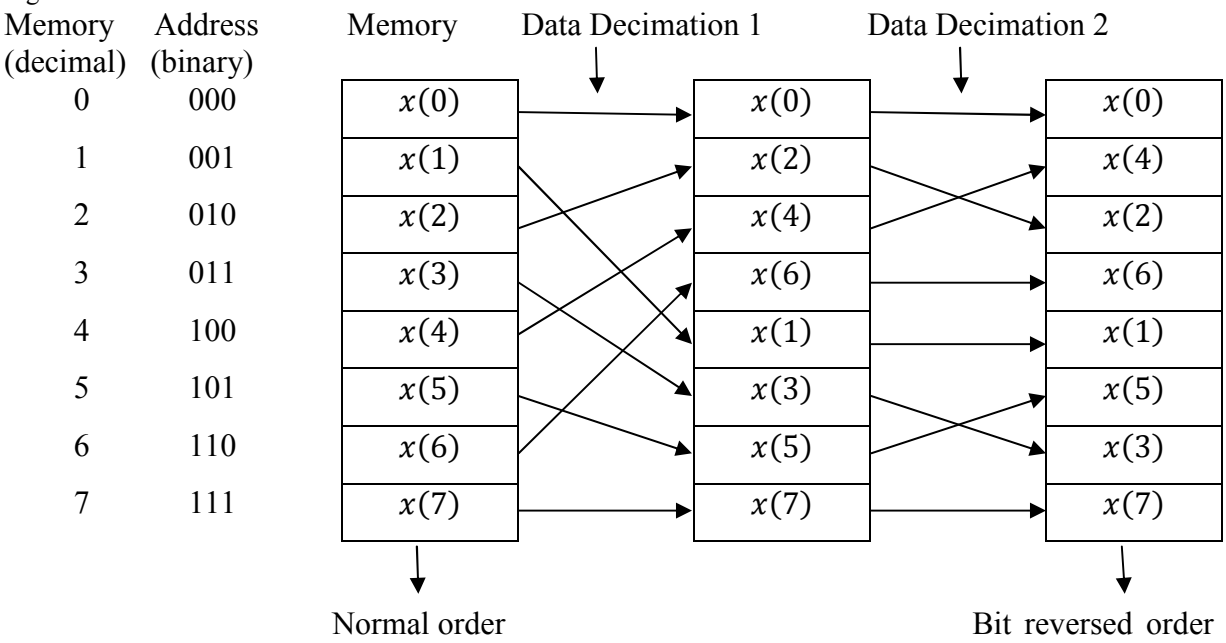


Figure.3 Shifting of data and bit reversal

The following Table 2 shows the input data $x(n)$ in normal order and its bit reversed order.

Table 2: Bit reversal with an 8-point input sequence

Sample Numbers in Normal Order		Sample Numbers in Bit Reversed Order	
Decimal	Binary	Decimal	Binary
0	000	0	000
1	001	4	100
2	010	2	010
3	011	6	110
4	100	1	001
5	101	5	101
6	110	3	011
7	111	7	111

The procedure of computing the discrete series of an N-point DFT into two (N/2)-point DFTs, each (N/2)-point sequence should be divided into two subsequences of even and odd terms and computing their DFTs consecutively. The decimation of the data sequence can be repeated again and again until the resulting sequence is reduced to one basic DFT.

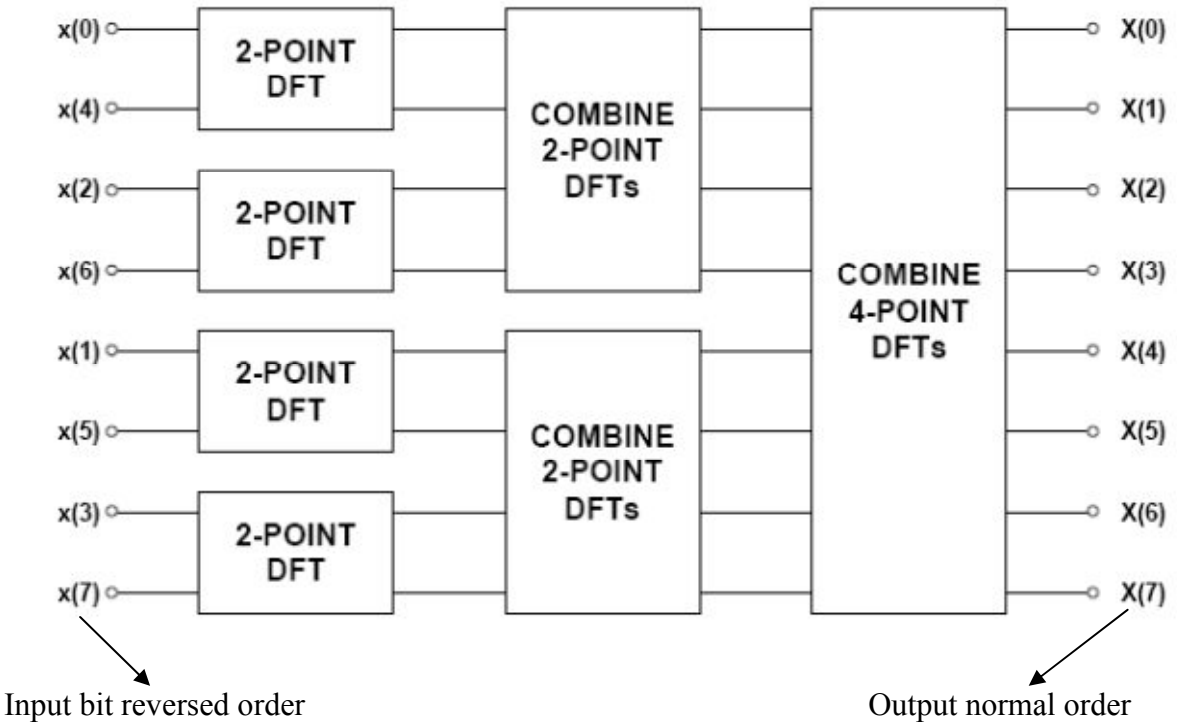


Figure 4: Block diagram of an 8-point DFT.

In the Fig.4, we observe that the computation of N = 8 point DFT is performed in three stages. ($3 = \log_2 8$)

Step 1: Compute four 2-point DFTs

Step 2: Compute two 4-point DFTs

Step 3: Compute one 8-point DFT

In general, for an N-point FFT, the FFT algorithm decomposes the DFT into $\log_2 N$ stages, each of which consists of (N/2) butterfly computations.

5. The Two-Dimensional FFT in Image Processing:

Work in the frequency domain would not be practical if we had to implement the equations (3) and (4) directly. Implementation of these equation requires on the order of $(MN)^2$ summations and additions. For images of moderate size say 1024 × 1024 pixels, this means on the order of trillion multiplications and additions for just one DFT, excluding the exponentials, which could be computed once and stored in a look-up table. This would be a challenge even for super computers. The FFT reduces the computations to the order of $(MN)(\log_2 MN)$ multiplications and additions. For instance, computing the two-dimensional DFT of a 1024 × 1024 image would require on the order of 20 million multiplication and additions, which is a significant reduction from the one trillion computations mentioned above [5].

The computational advantage of the FFT over a direct implementation of the one-dimensional DFT is defined as

$$C(N) = \frac{N^2}{N log_2 N} = \frac{N}{log_2 N} \dots\dots\dots\dots\dots\dots\dots\dots\dots (15)$$

Since it is assumed that $N = 2^n$, we can write equation (9) in terms of n:

$$C(n) = C(2^n) = \frac{2^n}{log_2 2^n} = \frac{2^n}{n} \dots\dots\dots\dots\dots\dots\dots (16)$$

Table 3: Values of $C(n)$ for different values of n.

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
C(n)	2	2	2.67	4	6.40	10.7	18.3	32	56.9	102.4	186	341	630	1170	2185

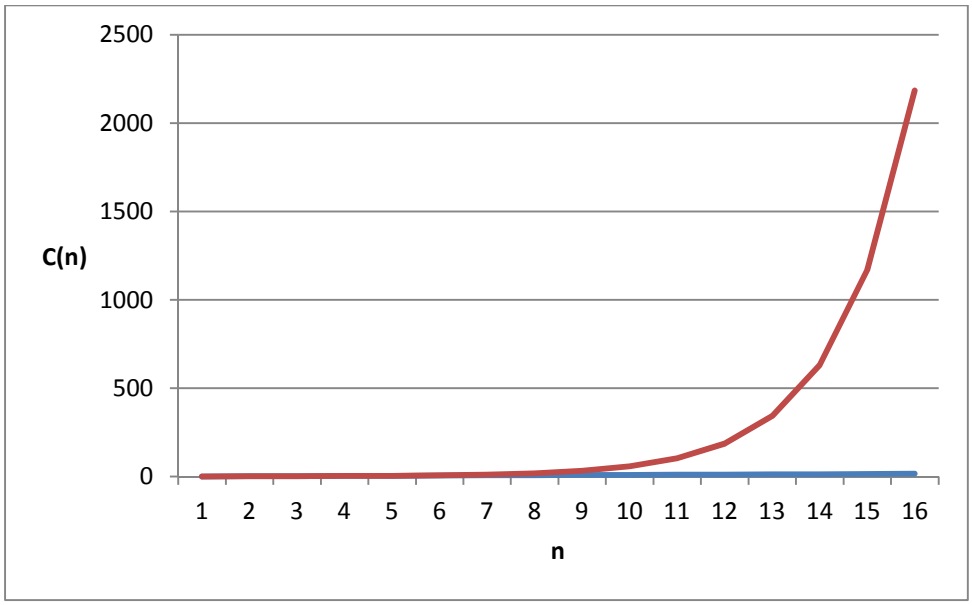


Figure 5: Computational advantage of the FFT over the direct implementation of the 1-D DFT.

Figure 5 shows the plot of the equation (16). It is seen that the computational advantage increases rapidly as a function of n.

For instance, when n = 15, then $C(n) = 2^{15}/15 = 32768/15 = 2184.53$

Thus we would expect that the FFT can be computed nearly 2200 times faster than the DFT on the same machine [5].

6. Conclusion:

We have presented the DFT, FFT, FFT algorithms with example. FFT algorithms, namely the divide and conquer approach shows the computational efficiency. The DFT needs N^2 complex multiplications and N (N - 1) complex additions, whereas the FFT takes only (N/2) ($\log_2 N$) complex multiplications and $N\log_2 N$ complex additions. Therefore, the ratio between the DFT computation and FFT computation for the same N is proportional to ($2N/\log_2 N$). If N is small, this ratio is not very significant. If N is large, this ratio gets very large. Hence the FFT is simply a fast way to calculate the DFT. The FFT algorithm is needed to compute DFT with reduced number of calculations. Therefore, FFT algorithm reduces the computation time required to compute DFT

References:

[1] James W.Cooley, "The Re-Discovery of the Fast Fourier Transform Algorithm", Mikrochimica Acta, Springer Verlag 1987, Vol. III, pp 33-45.
[2] James W.Cooley, Peter A.W.Lewis and Peter D.Welch, "The Fast Fourier Transform and its Applications" IEEE Transactions as Education, Vol. 12. No. 1, March,1969.
[3] Ludek Slosarcik, "FFT-Based Algorithm for Metering Applications", Freescale Semiconductor,Inc. Doc. No. AN4255, November,2011.
[4] Mohammad Nazmul Haque and Mohammad Shorif Uddin, "Accelerating Fast Fourier Transformation for Image Processing using Graphics Processing Unit", Journal of Emerging Trends in Computing and Information Sciences", Vol 2, No.8, August,2011.
[5] Rafael C.Gonzalez and Richard E.Woods, "Digital Image Processing", Third Edition, Pearson Education, Inc., 2009.
[6] John G.Proakis and Dimitris G.Manolakis, "Digital Signal Processing: Principles, Algorithms, and Applications", Fourth Edition, Pearson Education, Inc., 2007.
[7] S.Annadurai and R.Shanmugalakshmi, "Fundamentals of Digital Image Processing", Pearson Education, 2007.
[8] S.Palani and D.Kalaiyarasi, "Digital Signal Processing", First Edition, Ane Books Pvt. Ltd, New Delhi, 2010.

This academic article was published by The International Institute for Science, Technology and Education (IISTE). The IISTE is a pioneer in the Open Access Publishing service based in the U.S. and Europe. The aim of the institute is Accelerating Global Knowledge Sharing.

More information about the publisher can be found in the IISTE's homepage: http://www.iiste.org

CALL FOR PAPERS

The IISTE is currently hosting more than 30 peer-reviewed academic journals and collaborating with academic institutions around the world. There's no deadline for submission. **Prospective authors of IISTE journals can find the submission instruction on the following page:** http://www.iiste.org/Journals/

The IISTE editorial team promises to the review and publish all the qualified submissions in a **fast** manner. All the journals articles are available online to the readers all over the world without financial, legal, or technical barriers other than those inseparable from gaining access to the internet itself. Printed version of the journals is also available upon request of readers and authors.

IISTE Knowledge Sharing Partners

EBSCO, Index Copernicus, Ulrich's Periodicals Directory, JournalTOCS, PKP Open Archives Harvester, Bielefeld Academic Search Engine, Elektronische Zeitschriftenbibliothek EZB, Open J-Gate, OCLC WorldCat, Universe Digtial Library , NewJour, Google Scholar