NEW ALGORITHMS FOR COMPUTING DIRECTIONAL DISCRETE FOURIER TRANSFORMS

Marios S. Pattichis[1], Ruhai Zhou[2], and Balaji Raman[1]

[1]Image and Video Processing and Communication Laboratory (ivPCL)
e-mails: {pattichis@eece.unm.edu, rbalaji@eece.unm.edu}, phone: (505) 277-0486
Department of Electrical and Computer Engineering
[2]Albuquerque High Performance Computing Center (AHPCC)
email: rzhou@ahpcc.unm.edu
The University of New Mexico, Albuquerque, NM 87131-1356, USA

ABSTRACT

New algorithms for computing the Discrete Fourier Transform (DFT) spectra along different directions are derived and implemented. For computing the DFT spectrum along any given direction (containing N DFT frequencies), a new algorithm is presented that requires $N(N-1)$ additions and a single 1-D FFT. As expected, for a single direction, the directional FFT algorithm is significantly faster than standard 2-D FFT algorithms that compute the entire spectrum (all results are compared against *FFTW* and *FFTPACK*). A scalable extension of the unidirectional algorithm for computing the entire DFT spectrum is also derived and implemented. The three most promising features of the new algorithm are that: (i) computation scales nearly linearly with the number of DFT frequencies computed, (ii) the algorithm uses a reduced number of multiplications (yet uses more additions), and (iii) it is more accurate.

1. INTRODUCTION

Multi-dimensional Discrete Fourier Transforms (DFTs) have found many important applications in scientific computing. Usually, whenever used, DFT computation accounts for the most significant portion of the computation time. The standard tensor-methods for computing Discrete Fourier Transforms requires a cascade of one-dimensional Fast Fourier Transforms (FFTs) which: (i) do not provide for a scalable spectrum computation, (ii) do not provide for efficient methods for processing objects of irregular shapes (such as in handling Video Object Planes (VOPs) in MPEG-4, (iii) do not take advantage of inherent spectral symmetries such as circular symmetry often present in filterbanks. To address these issues, we introduce a novel algorithm of the fast computation of multidimensional Discrete Fourier Transforms (DFTs) using directional DFTs.

Let $x(n_1, n_2, \ldots, n_M)$ denote an m-dimensional array. We will assume that we have the same number of elements in each array dimension: $0 \le n_i \le N-1$. We express the m-dimensional DFT $X(k_1, k_2, \ldots, k_M)$ as:

$$X(k_1, k_2, \ldots, k_m) = \sum_{n_1=0}^{n_1=N-1} \cdots \sum_{n_M=0}^{n_M=N-1} x(n_1, n_2, \ldots, n_M) \, W_N^{k_1 n_1 + \cdots + k_M n_M} \quad (1)$$

where $W_N = \exp\left[-2\sqrt{-1}\pi / N\right]$. To explain the directional approach, we first note that there are exactly N roots of unity that are admitted by this M-dimensional transform.

For the two-dimensional case, we let

$$s = k_1 n_2 + k_2 n_2 \mod N,$$

recognize that s can take all possible N values and thus rewrite (1) as

$$X(k_1, k_2) = \sum_{s=0}^{s=N-1} \left(\sum_{k_1 n_1 + k_2 n_2 = s \bmod N} x(n_1, n_2) \right) W_N^s .$$

Furthermore, we can show that

$$X(mk_1, mk_2) = \sum_{s=0}^{s=N-1} \left(\sum_{k_1 n_1 + k_2 n_2 = s \bmod N} x(n_1, n_2) \right) W_N^{ms} .$$

We rewrite the last equation as one-dimensional DFT that can be computed using an FFT algorithm:

0-7803-6725-1/01/$10.00 ©2001 IEEE

$$Y(m) = \sum_{s=0}^{s=N-1} y(s) W_N^{sm},$$

where the *DFT sums* are computed using

$$y(s) = \sum_{k_1 n_2 + k_2 n_2 = s \bmod N} x(n_1, n_2),$$

and we map the directional spectrum to the 2-d DFT using:

$Y(0)$ is mapped to $X(0,0)$

$Y(1)$ is mapped to $X(k_1, k_2)$

$Y(2)$ is mapped to $X(2k_1, 2k_2)$

$\vdots \qquad\qquad \vdots$

$Y(N-1)$ is mapped to $X((N-1)k_1, (N-1)k_2)$

Based on our discussion, the proposed *directional decomposition algorithm* can be summarized in 5 basic steps:

Step 1. Compute DFT sums for directions $(1, k)$.

Step 2. Compute FFTs along directions $(1, k)$.

Step 3. Compute DFT sums for directions $(2k, 1)$.

Step 4. Compute FFTs along directions $(2k, 1)$.

Step 5. Map directional spectrum to standard 2-D DFT spectrum.

In steps 1 and 4, we note that the prime directions $(1, k)$ and $(2k, 1)$ are chosen so as to cover the entire 2-D spectrum [1,6]. The fact that these directions are sufficient for covering the DFT spectrum was first published in [6] for rectangular images. For two-dimensions, we note that only $3N/2$ one-dimensional FFTs are needed. Hence, in floating-point performance, the directional DFT approach rivals the vector-radix FFT (see [2] for a summary of the vector-radix FFT).

We note that it is clear from the directional spectrum definition that the different directions are independent of each other. This observation leads to a scalable algorithm summarized as:

Step 1. Transpose 2-D array for computing DFT sums for $(k, 1)$ directions.
(or distribute input matrix and its and transpose among processors).

Step 2. For $i = 0, 1, 2, \ldots, 2^q - 1$, either
Compute DFT sums for directions $(1, 2^q + i)$
Compute FFTs along directions $(1, 2^q + i)$

or

Compute DFT sums for directions $(2^{q+1} + 2i, 1)$
Compute FFTs along directions $(2^{q+1} + 2i, 1)$

Step 3. Map directional spectrum to standard 2-D DFT spectrum.

The algorithm emphasizes the fact that the scalable algorithm is very easy to implement on a parallel architecture. This is currently under investigation. Furthermore, we note that the directional spectrum computation is inherently very accurate, since a single FFT is used to compute each frequency component.

In Section 2, an efficient algorithm for computing the DFT sums is presented. After implementation, comparative results are presented in Section 3. A summary of ongoing research is presented in Section 4.

2. AN EFFICIENT ALGORITHM FOR COMPUTING DFT SUMS

The most fundamental challenge of the new approach is in computing the DFT sums. A direct implementation of the DFT sums will be very inefficient. It is possible to derive a very efficient implementation of the DFT sums, after we notice a recursive relationship that is summarized in Theorem 1.

The fundamental result of this Section is due to the following theorem (see [1] for a proof):

Theorem 1. If $k = Cp + i$ with

$s_k(m,n) = m + kn \mod N$, then

$$s_k(m,n) = s_k(m + (C-i)N/C, n + N/C). \quad (2)$$

In the theorem, $k = Cp + i$ refers to an equivalent class of two-dimensional "prime" directions denoted by $(1, k)$. Two "prime directions" $(1, k_1)$ and $(1, k_2)$ belong to the same equivalent class if they are both expressible in the form $k = Cp + i$. Hence, theorem 1 presents an efficient recursive relationship for computing the DFT sums for frequencies within the same partition. It is straightforward to see that this leads to an efficient algorithm for computing two-dimensional DFT Spectra for different partitions. It is possible to divide the sum computation in 2, 4, 8, 16, etc equivalent pieces. Clearly, for each partition, there is a corresponding number of one-dimensional FFTs as given in the introduction.

The effective algorithm is captured in (2). It describes how we can add the *n*-th column and the *(n+N/C)*-th column after shifting the latter column by *(C-i)N/C*. This allows us to compute a partial sum for the specific direction $k=C\backslash,p + i$, where *n* can be from *0* to some suitable integer. In other words, the resulting sums from each step are spilt into *C/2* equivalent classes, each equivalent class containing *2N/C* partial sums for the direction *C/2+i* for some *i* from *0* to *C/2-1*. In each group, we add the first half columns and the other half columns twice with row shifting *(C-i)N/C* and *(C-i)N/C-N/2*, respectively. After this step, we have added 2C columns of the original input for each direction.

Using this algorithm, we get that the number of additions in the algorithm described above is $N^2 \log N$. Naturally, these additions are performed using integer arithmetic.

3. TWO DIMENSIONAL DFT COMPUTATIONAL RESULTS

In this section, some computational results of the directional DFT algorithms are presented. The algorithms were implemented using FORTRAN90. Because one dimensional DFTs are needed in the directional DFT method, the performance of our method is dependent on the one dimensional FFT code used. In addition, none of the results that we are presenting were optimized for the target architecture. Such results will be reported in [8,9,10].

For the one-dimensional FFT code, we used *FFTPACK* and *FFTW* [3,4,5,11]. For the *FFTPACK*-based implementation, single precision arithmetic is used. For the *FFTW*-based implementation, double precision is used. Furthermore, in the following simulations, the input images were assumed to be of integer type. All computation was done using a 550-Mhz Pentium III machine of the Albuquerque High Performance Computing Center.

The computational results for spectrum computation are summarized in Tables 1 and 2. The times (in milliseconds) for the familiar tensor decomposition algorithm of computing FFTs along each row (or column) followed by matrix transpose, and computing FFTs along each column (or row) are given as the first rows in tables 1 and 2. The times for the directional decomposition algorithm are given in the 2nd rows For the directional-spectrum computation (listed as Dir Dec I), and in the 3rd rows (listed as Dir Dec II) for the standard FFT spectrum computation. In the directional decomposition algorithm described in the Introduction, the 2nd row timings correspond to the total time taken for Steps 1, 2, 3 and 4, while the 3rd row timings correspond to the total time taken for all the steps.

From the results, it is clear that the directional decomposition algorithm computes the directional spectrum faster than the standard (tensor) spectrum. Furthermore, for computing the standard FFT spectra, the directional algorithm is slightly slower.

Algorithm	N=1024	N=512	N=256	N=128
Tensor Dec	1618	359	61	12
Dir. Dec I	1311	301	58	13
Dir. Dec II	1769	404	80	18

Table 1. Computational Times for the Directional DFT using 1-D FFT provided by *FFTPACK*. Times are measured in milliseconds.

Algorithm	N=1024	N=512	N=256	N=128
Tensor Dec	1677	378	87	15
Dir. Dec I	1653	363	77	16
Dir. Dec II	2281	511	111	23

Table 2. Computational Times for the Directional DFT using 1-D FFT provided by *FFTW*. Times are measured in milliseconds.

4. ONGOING RESEARCH

Ongoing research will focus on two distinct areas: (i) a more efficient implementation of the directional and standard DFT for the SIMD and SIMD2 extensions of the Pentium III and Pentium 4 architectures [8,9], and (ii) an extension of the directional DFT for computing multidimensional DFTs of irregularly-shaped objects, and for circularly-symmetric filters.

Many of the results in our paper can be improved by using a one-dimensional FFT that is optimized for the Pentium III and Pentium 4 architectures. These results will be presented in [8,9]. To the best of our knowledge, the fastest one-dimensional FFT implementation on the Pentium Architecture will be presented in [9]. For the Pentium-specific implementations, all the additions and the following one-dimensional FFT algorithms have been adopted to take advantage of the Streaming SIMD Extensions (SSE), allowing us to execute additions and multiplications on at-least 4 elements at a time.

Most of the promise in the proposed algorithm is in its ability to compute efficient directional spectra of irregularly shaped objects. This is primarily due to the fact that the zeros can be efficiently handled in the additions

part of the algorithm. Furthermore, for circularly-symmetric filters, the directional DFT is ideal in that it need only compute 1/8th of the entire 2-D spectrum.

The algorithms presented here have also been extended to rectangular (non-square) images and also 3-D images. The results will be presented in [10].

5. REFERENCES

[1] M.S. Pattichis and R. Zhou, "A Novel Directional Approach for the Scalable, Accurate and Efficient Computation of Two-Dimensional Discrete Fourier Transforms," AHPCC2000-019, Albuquerque High Performance Computing Center, The University of New Mexico, 2000.

[2] D.E. Dudgeon & R.M. Mersereau, *Multidimensional Digital Signal Processing*, Prentice-Hall, New Jersey 1984.

[3] M. Frigo, "A Fast Fourier Transform Compiler," *Proceedings of the 1999 ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, Atlanta, 1999.

[4] M. Frigo & S.G. Johnson, *The Fastest Fourier Transform in the West*, MIT-LCS-TR-728, Massachusetts Institute of Technology, 1997.

[5] M. Frigo & S.G. Johnson, "FFTW: An Adaptive Software Architecture for the FFT," *ICASSP Conference proceedings*, vol. 3, 1998.

[6] M.S. Pattichis, "Novel Algorithms for Accurate, Efficient, and Parallel Computation of Multidimensional, Regional Discrete Fourier Transforms," *Proceedings of the 10th Mediterranean Electrotechnical Conference*, Limassol, Cyprus, May 29-31, 2000.

[7] M.S. Pattichis, A.C. Bovik, J.W. Havlicek, and N.D. Sidiropoulos, "Multidimensional Orthogonal FM Transforms," *IEEE Trans. on Image Processing*, vol. 10, no. 3, pp. 448-464, Mar. 2001.

[8] Paul Rodríguez V. and Marios S. Pattichis, "Real time Computation of the Directional Discrete Fourier Transform on the Pentium 4," submitted to the *2001 Workshop on Multimedia Signal Processing*, Cannes, France, Oct. 3-5, 2001.

[9] Paul Rodríguez V., "A novel approach for the radix-2 FFT algorithm using the Streaming SIMD Extensions (SSE) architecture," to be submitted to *Signal Processing Letters*.

[10] Marios S. Pattichis, Ruhai Zhou, and Paul Rodríguez V., "New Results on Computing Directional Discrete Fourier Transforms," to be submitted to *IEEE Transactions on Image Processing*.

[11] P.N. Swarztrauber, "Vectorizing the FFTs," in *Parallel Computations*, G. Rodrigue, ed., Academic Press, pp. 51-83, 1982, http://www.netlib.org/fftpack.