Correspondence

Distributed Vector Processing of a New Local MultiScale Fourier Transform for Medical Imaging Applications

Robert A. Brown*, Hongmei Zhu, and J. Ross Mitchell

***Abstract*—The recently developed S-transform (ST) combines features of the Fourier and Wavelet transforms; it reveals frequency variation over both space and time. It is a potentially powerful tool that can be applied to medical image processing including texture analysis and noise filtering. However, calculation of the ST is computationally intensive, making conventional implementations too slow for many medical applications. This problem was addressed by combining parallel and vector computations to provide a 25-fold reduction in computation time. This approach could help accelerate many medical image processing algorithms.**

***Index Terms*—Biomedical image processing, discrete Fourier transforms, distributed computing, vector processing.**

I. Introduction

The S-transform (ST) [1], a generalization of the Fourier transform (FT), is a spectral localization transform that utilizes a frequency adapted Gaussian window to achieve optimum resolution at each frequency. Signals may be directly translated between the Fourier and S domains without loss of information, including preservation of phase information. The ST was originally developed for geophysics applications and was recently introduced for medical imaging and signal processing [2]–[5].

The classic Fourier transform provides information about the frequency content of an entire signal or image while the ST provides a local spectrum for each signal sample. This makes the ST useful for identifying changes in frequency content over time or space. The one-dimensional (1-D) ST, applied to a signal [such as time course functional magnetic resonance imaging (fMRI) data], can be used to localize and remove noise components and artifacts [3]. The two-dimensional (2-D) ST has proven useful for filtering and artefact removal in MR imaging [3], [4]. The 2-D ST also can provide local textural information that can be used to distinguish tissues and indicate disease activity [2], [4].

Unfortunately, calculation of the ST is computationally intensive. Currently, the time required to transform a typical MR image makes research difficult and application in a clinical setting often impractical. This paper describes techniques to accelerate the ST algorithm to allow application in clinically relevant time frames (<5 min to perform a 256 × 256 ST).

Manuscript received September 1, 2004; revised January 21, 2005. This work was supported in part by the Alberta Heritage Foundation for Medical Research (AHFMR), in part by the Multiple Sclerosis Society of Canada, and in part by the Natural Sciences and Engineering Research Council of Canada (NSERC). The Associate Editor responsible for coordinating the review of this paper and recommending its publication was P. Thévenaz. *Asterisk indicates corresponding author.*

*R. A. Brown is with the Department of Electrical and Computer Engineering, University of Calgary and the Seaman Family MR Research Centre, Foothills Medical Centre, 1403 29 St. N.W., Calgary, AB T2N 2T9, Canada (e-mail: brownr@ucalgary.ca).

H. Zhu was with the Seaman Family MR Research Centre, Calgary, AB T2N 259, Canada. She is now with the Department of Mathematics and Statistics, York University, Toronto, Toronto, ON M3J 1P3, Canada (e-mail: hzhu@yorku.ca).

J. R. Mitchell is with the Departments of Radiology and Clinical Neurosciences, Electrical and Computer Engineering, University of Calgary and the Seaman Family MR Research Centre, Foothills Medical Centre, Calgary, AB T2N 259, Canada (e-mail: rmitch@ucalgary.ca).

Digital Object Identifier 10.1109/TMI.2005.845320

II. Methods

The ST of an image $I(\mathrm{x},\mathrm{y})$ is defined by [1]

$$S(x,y,k_x,k_y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} G(\alpha,\beta;k_x,k_y)H(k_x-\alpha,k_y-\beta)\,\mathrm{d}\alpha\,\mathrm{d}\beta \quad (1)$$

where $H(k_x,k_y)$ is the FT of the source image and $G(\alpha,\beta;k_x,k_y)$ is a frequency adapted Gaussian window. This formulation of the ST allows us to build an algorithm that utilizes the fast Fourier transform (FFT) for more efficient computation [4].

1) Calculate the FFT of the image:

$$H(\alpha,\beta) = \mathrm{FFT}[I(x,y)].$$

FOR every frequency (k_x,k_y) DO:

2) Calculate the localizing 2-D Gaussian window at the current frequency (k_x,k_y)

$$G(\alpha,\beta;k_x,k_y) = e^{-\frac{2\pi^2\alpha^2}{k_x^2}}\cdot e^{-\frac{2\pi^2\beta^2}{k_y^2}}.$$

3) Shift the Fourier spectrum $H(\alpha,\beta)$ to $H(\alpha-k_x,\beta-k_y)$.
4) Let $L(\alpha,\beta;k_x,k_y) = H(\alpha-k_x,\beta-k_y)G(\alpha,\beta;k_x,k_y)$.
5) Inverse FT $L(\alpha,\beta;k_x,k_y)$ from the α–β plane into the x–y plane which gives the 2-D $S(*,*,k_x,k_y)$ at the current frequency (k_x,k_y). Note that $S(*,*,k_x,k_y)$ indicates the spatial locations where the frequency (k_x,k_y) occurs.

END FOR.

For reference, we implemented the above ST algorithm in the Interactive Data Language (IDL Version 6.0.1, Research Systems Inc., Boulder Co.), which is designed to allow rapid implementation of new algorithms. Our ST implementation uses the built-in IDL FFT and performs the ST of a 256 × 256 image in about 5600 s (≈1.5 h) (Dual 1-GHz Apple PowerMac G4 using one processor).

Three basic tasks are performed inside the loop in algorithm 1: a shift of $H(\alpha,\beta)$ to $H(\alpha-k_x,\beta-k_y)$, a point-wise matrix multiplication $H(\alpha-k_x,\beta-k_y)\cdot G(\alpha,\beta;k_x,k_y)$ and an inverse FT. For an $\mathrm{N}\times\mathrm{N}$ image $I(x,y)$ these tasks are $O(c)$ (accomplished with pointer operations), $O(N^2)$ and $O[N^2\log(N)]$ respectively. Steps two through five are repeated N^2 times, once for each frequency (k_x,k_y). This yields an overall complexity for the 2-D ST of $O[N^4\log(N)]$. The ST of a 2-D image produces a four-dimensional structure: $S(x,y,k_x,k_y)$. The ST's memory requirement grows as N^4. Thus, a 256 × 256 image requires 256^4 storage elements -32 GB of floating point values for example. These memory requirements pose problems not only for ultimate long-term storage but also for the execution of an ST. Note that the ST of a real-valued image is symmetric, like the FT. Thus, for real-valued images only half of the S-domain needs to be calculated. In this paper we have performed the entire ST calculation as our primary interest is analysis of MR images, which are acquired in the complex Fourier domain.

It is possible that further ST research will reveal a more efficient method of computation. However, in the absence of such an algorithm, we propose two approaches to increase computation speed. First, the ST algorithm can be modified to utilize vector computing hardware available in recent desktop workstations. Second, the ST algorithm can be parallelized and distributed over multiple processors since each loop iteration is fully independent—each can be calculated with only the original image as input.

The Macintosh G4 processor includes the Altivec vector processing unit—a powerful parallel vector processor to accelerate multimedia and signal processing tasks [6]. The Altivec processor performs operations on a 128-bit vector that can be flexibly divided into several elements. For example, the Altivec unit can perform an operation on four floating-point values simultaneously, each clock cycle. Many common signal processing operations have been optimized for the Altivec architecture.

Apple Computer's vecLib library includes FFT routines optimized for Altivec [6]. We wrapped the 2-D in-place FFT function for easy use and benchmarking in IDL and discovered the Altivec FFT is at least five times faster than the IDL function for matrix sizes of 128×128 and larger. Since the FFT is the highest order operation in our ST algorithm, the simple substitution of the optimized FFT function provides immediate benefit.

Profiling of the resulting code revealed that the majority of the remaining processing time was due to the application of the Gaussian window. This operation consists of simple element-wise multiplication of two matrices and also was a candidate for Altivec optimization. Unfortunately, for the Altivec unit to work efficiently, the data must be aligned on a 16-byte boundary. This is not a problem for the Gaussian window itself, but the data to which it is being applied is shifted at each iteration, so scalar code must be used in the case of misalignment.

Distributed algorithms may perform lengthy calculations by executing on several computers (nodes), connected by a network to form a computing cluster. Due to the inherently parallel nature of the ST algorithm it is conceptually easy to construct a distributed form. As the calculation for each (k_x, k_y) value is independent, each can be assigned to a separate node.

A distributed computing system was implemented in C++ using custom protocols built on raw socket communications. The cluster software consists of a controller that handles scheduling of jobs, a client application that requests jobs to be executed and a node program that performs the actual ST calculation. Each node is given the image and a list of k_y values it is responsible for calculating. Apart from the Altivec code, which is easily replaced by native functions on different architectures, each of the three components of the cluster software are portable and may be recompiled and executed on a wide range of platforms. This flexibility allows a powerful computing cluster to be assembled from common commodity hardware. Workstations already in the lab can be configured to participate in cluster computation whenever they are idle.

Results for this paper were obtained from a cluster of desktop workstations, all Apple PowerMac G4 machines with dual 1.0-GHz processors (only one processor per machine was used for benchmarking). The nodes were connected through a standard switch via the built in Gigabit Ethernet interfaces over either copper or fiber optic cable.

III. Results

As the FFT is the highest order operation in the ST algorithm, FFT performance is very important to the ST as a whole, especially with larger images. The IDL ST using the Altivec FFT was about 3.5 times faster than the pure IDL implementation for a 256×256 image. When combined with Altivec optimization of the code that applies the Gaussian window (the second highest order operation), implementation in C and hand tuning, performance increases of almost 7 times were obtained for the ST of 256×256 images. This brings calculation of the ST for this image size to less than 14 minutes, compared to over 90 minutes for the native IDL implementation.

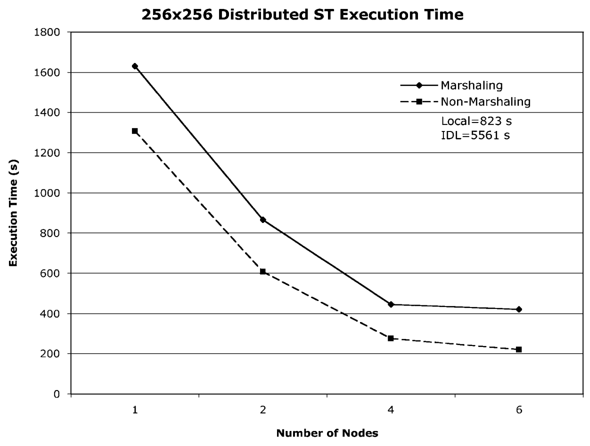

Fig. 1. Computing the ST using a cluster of 1-GHz G4 Powermacs with gigabit Ethernet interconnections. Distributing calculation introduces some overhead, but a 2-node cluster is almost as fast as an entirely local calculation. If additional analysis (such as texture analysis) can be done in a distributed fashion and the full ST results do not need to be transmitted to one machine (nonmarshalling mode), a 2-node cluster is faster than the local ST. The distributed ST performance continues to scale to all cluster sizes tested (up to 6-node).

We tested two configurations of the distributed ST. In the first, results from each node are returned to a receiver node (marshaled) and merged into one data structure. This approach is required for analysis that requires the full ST and cannot itself be distributed. In the second case, individual nodes do not return their results to a common node (nonmarshalled). This configuration eliminates a significant amount of network transmission and is useful for analysis algorithms that can be at least partially distributed along with the ST calculation (texture analysis for example [2]).

Fig. 1 shows the performance of the two distributed versions of the ST and the local ST for comparison. The distributed ST runs more slowly on one node than the local version due to overhead associated with scheduling and network communications, but a 2-node cluster that does not marshal its results outperforms the local ST by a factor of 1.4. A 4-node cluster is faster than the local ST for both marshaling and nonmarshaling configurations. The nonmarshalling version using four nodes calculates the ST of a 256×256 image in less than five minutes, a speed that is potentially clinically acceptable.

IV. Conclusion

The ST has been shown to be useful in a variety of medical image processing tasks. Unfortunately, research is difficult, and clinical relevance of techniques is limited because of the long processing times. We have demonstrated a method for vector computation of the ST, resulting in a speed improvement of 6.8 times for a 256×256 pixel image, and a method for distributed computation of the ST that scales acceptably with increasing cluster size. Both techniques were implemented using standard desktop computers and networking hardware. While our single processor, unoptimzed reference implementation took over an hour and a half (5561 s) to calculate the ST of a 256×256 pixel image, using a four node vector-computing cluster we were able to perform the same transform in less than four minutes (221 s), over 25 times faster. Future research will focus on investigating how the ST might be accelerated further by leveraging redundancy in the S-domain. Ultimately, calculation of the ST in minutes instead of hours makes techniques for filtering, image enhancement and feature detection based on the ST practical for acute clinical applications.

References

[1] L. Mansinha, R. G. Stockwell, and R. P. Lowe, "Pattern analysis with two-dimensional spectral localization: Applications of two-dimensional S transforms," *Physica A*, vol. 239, pp. 286–295, 1997.

[2] H. Zhu, G. Mayer, L. Mansinha, L. A. Law, C. J. Archibald, M. Luanne, and J. R. Mitchell, "Space-local spectral texture map based on MR images of MS patients," *MS: Clin. Lab. Res.*, 2001.

[3] B. G. Goodyear, H. Zhu, R. A. Brown, and J. R. Mitchell, "Removal of phase artifacts from fMRI data using a stockwell transform filter improves brain activity detection," *Magn. Reson. Med.*, vol. 51, pp. 16–21, 2004.

[4] H. Zhu, B. G. Goodyear, R. A. Brown, G. Mayer, A. G. Law, L. Mansinha, and J. R. Mitchell, "A new local multiscale Fourier analysis for medical imaging," *Med. Phys.*, vol. 30, pp. 1134–1141, 2003.

[5] R. Crandall and J. Klivington. (2000) Supercomputer-style FFT library for Apple G4. [Online]. Available: http://images.apple.com/acg/pdf/g4fft.pdf