Microsc. Microanal. 11, 9–17, 2005
DOI: 10.1017/S1431927605050014

Microscopy AND Microanalysis

Wavelet Compression of Three-Dimensional Time-Lapse Biological Image Data

H. Narfi Stefansson,[1] Kevin W. Eliceiri,[2,*] Charles F. Thomas,[2] Amos Ron,[1] Ron DeVore,[3] Robert Sharpley,[3] and John G. White[2]

[1]*Department of Mathematics and Computer Science, University of Wisconsin, Madison, WI 53706, USA*
[2]*Laboratory for Optical and Computational Instrumentation, University of Wisconsin, Madison, WI 53706, USA*
[3]*Department of Mathematics, University of South Carolina, Columbia, SC 29208, USA*

Abstract: The use of multifocal-plane, time-lapse recordings of living specimens has allowed investigators to visualize dynamic events both within ensembles of cells and individual cells. Recordings of such four-dimensional (4D) data from digital optical sectioning microscopy produce very large data sets. We describe a wavelet-based data compression algorithm that capitalizes on the inherent redunancies within multidimensional data to achieve higher compression levels than can be obtained from single images. The algorithm will permit remote users to roam through large 4D data sets using communication channels of modest bandwidth at high speed. This will allow animation to be used as a powerful aid to visualizing dynamic changes in three-dimensional structures.

Key words: wavelet image data compression, digital microscopy

Introduction

Much of our knowledge of the biological world has been obtained through the study of images. Research scientists use images from microscopes to understand the workings of a cell, the development of an organism, or the pathological state of a tissue. The use of microscopy to study living specimens has gained considerable momentum recently. This is primarily the result of the combined use of genome sequence data and fluorescent protein reporter technology, which has allowed the distribution of any protein to be observed in cells, tissues, or even embryos (Chalfie et al., 1994). Developments in microscopy have allowed optical sections to be visualized within intact tissue, thereby obviating the need for the microscopists' traditional strategy of cutting sections in order to reconstruct three-dimensional (3D) data.

Techniques for optical sectioning have been developed that can provide 3D data from living specimens. Nomarski imaging is a relatively simple technique that uses gradients of refractive index to produce contrast (Nomarski, 1955). It is the most benign of the optical sectioning techniques but has little specificity. On the other hand, fluorescence microscopy used in conjunction with fluorescent protein reporters can be used to reveal the distribution of a selected few molecular species out of the 100,000 components that typically make up a living organism. However, this technique can give rise to problems of phototoxicity. The newly developed optical sectioning technique of multiphoton fluorescence excitation (MPFE) imaging produces low levels of photoxicity and can obtain images considerably deeper into a specimen than the other commonly used fluorescence optical sectioning method, confocal imaging (Denk et al., 1990; Centonze & White, 1998). Using MPFE imaging, we have shown that it is possible to make continuous multifocal-plane, time-lapse recordings of a vertebrate embryo for 24 h without compromising viability or developmental potency (Squirrell et al., 1999).

To visualize dynamic processes in three dimensions using optical sectioning microscopy, multifocal-plane, time-lapse movies are often used. These movies are recorded digitally by a computer that is interfaced to the microscope. The computer controls microscope focus and image capture. Using multidimensional image viewing applications (Thomas et al., 1996; Eliceiri et al., 2002) an investigator can roam through four-dimensional (4D; 3D plus time) data sets in space and time using animation as an aid to visualizing dynamic processes occurring within the recording. This technique is proving to be a very powerful way of visualizing both developmental processes involving sheets of cells moving in three dimensions and intercellular processes involving dynamic changes to cytoarchitecture such as those that occur during cell division.

In the course of our studies on the early development of the nematode, *Caenorhabditis elegans*, we have assembled large numbers of 4D recordings depicting the early develop-

Received October 28, 2003; accepted January 15, 2004.
*Corresponding author. E-mail: eliceiri@wisc.edu

ment of both wild-type and mutant animals. These recordings are currently archived on CDs and are therefore not generally accessible. We are in the process of establishing an on-line database for these recordings that will allow remote users to access these data. We are also developing 4D browsers that can be used on client computers that can provide animated views of the specimen under study in both space and time. The major challenge in achieving this goal is that data have to be transmitted at high speed over communication lines of limited bandwidth in order to allow smooth animations. We have been investigating the potential of wavelet-based compression techniques applied to 4D data to allow a user to roam through large, multidimensional image data sets stored in a remote database via readily available, medium-speed data communication channels (e.g., 10 base T Ethernet).

The wavelet algorithms that we have been developing capitalize on the redundancy present in 4D image data sets to obtain much higher degrees of data compression than can be achieved using JPEG or MPEG algorithms. The goal of this project is to allow investigators on client terminals to roam through 4D image data sets of living specimens. These data sets can be very large, typically 5 Gbyte (512×512 pixels $\times$ 20 sections $\times$ 1000 time points). To achieve reasonably smooth animation (say, 10 frames per second), a data rate of around 20 Mbit/s is required. However, these speeds are usually impractical on the Internet; hence compression techniques that capitalize on inherent redundancy in the data must be applied. We have found that we can compress monochrome images up to a maximum of a 20:1 ratio using discreet cosine transforms as used in the JPEG compression algorithms, bringing the rate down to around 1.6 Mbit/s, which, although a substantial improvement, is still a large bandwidth requirement for the Internet. Indeed, one should note that because our data are four-dimensional, the MPEG compression could do better than JPEG compression because it attempts to exploit the fact that adjacent frames in a video are similar; moreover, four-dimensional data sets have yet an extra dimension of redundancy, allowing even higher levels of compression. We are developing the algorithms for the real-time compression of the 4D image files generated by optical sectioning microscopes, including Nomarski, confocal, and multiphoton imaging systems. However these algorithms are applicable to any multidimensional data.

Our aim has been to develop compression algorithms that function well at high levels of compression. This is to enable remote users to roam through 4D data sets held on a server with reasonable animation rates. The widely used JPEG/MPEG algorithms are known to perform well for low rates of compression (usually at the range of 10:1 up to 20:1), but rapidly break down (i.e., lead to images with highly unpleasant blocky artifacts) at higher rates. In contrast, the more sophisticated wavelet-based techniques perform significantly better at these high rates and "die gracefully" at extreme compression ratios.

Image Compression: The Principle Behind the Wavelet Representation

Successful image processing, compression in particular, hinges on the selection of effective representation methods. For compression, it is essential to find a *sparse* representation of the image. The basic idea is simple: One creates a small blurred "thumbnail" of the original image, and records separately the thumbnail and the "variations" or "details" in the image, that is, the thumbnail and the difference between the true image and its thumbnail representation. Because the amount of detail in a typical image varies spatially, it is important that the above variations will be recorded locally: In this way one may simply disregard the details in the areas where they matter less, and record them more faithfully in areas where they are substantial.

The original representation of the image as an array of pixels does not conform with the above methodology, and is indeed disastrous for compression. The Fourier representation records the "details" in a global form; hence it is ineffective, too. The original JPEG algorithm divides the image into small blocks, and uses the Fourier representation on each block. The small image within a certain block is then registered with a bit budget that is proportional to the amount of detail found in the block. One of the main drawbacks of this strategy is that the block size is fixed (and is usually small). Thus, in the case where the image contains a large area with very little detail, the bits spent by JPEG to record the information in that area will be proportional to the number of blocks that overlap with that area, and will not be related any more to the amount of information there.

The wavelet representation overcomes the above difficulty by employing blocks of variable sizes. The smallest blocks are 2×2 and the largest block matches the size of the image. Each block registers local details that were not already captured by its parental, larger, blocks. The number of bits allotted to each block (in the uncompressed representation) is fixed, that is, is independent of the block size and its location in the image. In the wavelet representation, the bigger blocks almost always contain valuable information. However, the vast majority of the blocks are small, and those typically contain no significant information unless they overlap a finely detailed area (e.g., an edge in the image). Such representation lends itself naturally to effective compression: One ignores the blocks with little information and records only the information in the significant blocks. However, because the location and size of the significant blocks is image dependent, one must also register the *location and size* of the significant blocks. This turned out to be a subtle art and is known as *encoding*.

When the recipient of the compressed information decompresses it, the decompressed data set will only approximate, and not necessarily be equal to, the original data set, just as happens when using JPEG and MPEG. This is known as *lossy compression* and it is a consequence of the fact that one only records the information in the significant blocks.

Wavelets and Multidimensional Data

Wavelet compression of image data is beginning to be generally used. The recent JPEG2000 compression standard (Christopoulos et al., 2000; Skodras et al., 2001) uses wavelet methodology, but only in two dimensions. 3D imagery is acquired as a sequence of 2D sections. It is grossly inefficient to compress each of the 2D sections separately, because nearby sections contain strongly correlated information; specifically if a wavelet block in one 2D section contains substantial information, it is likely that the same is true for the same block in an adjacent section. Fortunately, the wavelet representation methodology applies to 3D and even higher dimensional images, thus allowing one to process the union of the 2D sections as a single, cohesive 3D one. Thus, the aforementioned corresponding blocks in the 2D sections are treated in this way as one 3D block. From a computational point of view, the 3D treatment of the image is almost free: The treatment of the union of 2D sections as one 3D image increases the computational complexity by a fixed percentage (that depends on the details of the chosen wavelet system) as compared to the separate processing of the 2D sections. However, if the correlations among the sections are weak (e.g., when the vertical distance between adjacent sections is substantial), the 3D approach may result in *lower* compression rates, and hence should sometimes be avoided.

The addition of *time* as a fourth dimension results in 4D imagery. Just as with the third spatial dimension, incorporating the time dimension into the compression can offer large gains in compression rates if there is temporal correlation in the data set, and, if there is little such correlation, it may result in lower compression rates. The basic wavelet compression technique does not incorporate motion compensation, that is, the attempt to improve compression by tracking the motion of an object.

Materials and Methods

The Main Components of Wavelet Compression

- Wavelet transform: Much like the Fast Fourier Transform, which represents a data set via its Fourier coefficients, the wavelet transform represents a data set via its wavelet coefficients. A data set with N points is represented with the aid of N wavelet coefficients. The computational effort of the wavelet transform is of order N, whereas the FFT is of the order $N \log N$. The wavelet coefficients can be naturally thought of as several arrays of matrices, with a total of N entries. Each matrix corresponds to frequency contributions, with the position within the matrix identifying the spatial/temporal location.
- Thresholding/quantization: Typically the wavelet coefficients are computed and represented via floating point arithmetic (as opposed to only computing with signed integers or with rational numbers). In lossy compression, one chooses a single floating point number, called a "quantum" or a "threshold," and approximates the wavelet coefficients as integer multiples of that single floating point number. Most of the coefficients are insignificant and are rounded to zero. Thus, one is left with the single floating point number, the "quantum," and arrays of sparse integer matrices.
- Encoding: The arrays of sparse integer matrices must be written efficiently to a file. Because there are correlations between the various matrices in each array, this can be done very effectively.
- Lossless compression: The output from the encoding step is not free of correlations, so a general-purpose lossless compressor can compress it even further.

Wavelet Transform—Preliminaries

Before going into details, we begin with an overview. We assume, for simplicity, that the given d-dimensional image is an array of $2^M \times 2^M \times \ldots \times 2^M$ pixels. The wavelet decomposition is an iterative procedure, which produces at each iteration j, a "thumbnail" of the data "zoomed out" (by magnification 2^j) to scale j and the detail at that level required to reconstruct the original data. The input for each iteration is a function $F_{0,j}$ (with j a positive integer), which is defined on a given array of blocks of the original data. Each block in this array is a cube with edge size 2^j. The total number of blocks in the array is, thus, $2^{d(M-j)}$. The value $F_{0,j}(m)$ of the function at the block indexed by m represents the "intensity" of this block at scale j.

The output of the iteration is a sequence of 2^d arrays of blocks, which are indexed by $(i, j+1)$ with $i = 0, \ldots, 2^d - 1$. Each block in the $(i, j+1)$-array has an edge size 2^{j+1} (hence the total number of blocks in each array is $2^{d(M-j-1)}$). Each array $(i, j+1)$ is associated now with a new intensity function $F_{i,j+1}$, that is, $F_{i,j+1}(m)$ describes the intensity value of the block m in the $(i, j+1)$-array at scale $j+1$. The array $(0, j+1)$ is our new thumbnail array and the arrays $(i, j+1)$, $i \geq 1$ store the details, that is, the difference between the $(0, j)$ array and the $(0, j+1)$ array. We "save" all the information associated with the "detail" arrays $(i, j+1)$, $i \geq 1$, and keep iterating on the new thumbnail array $(0, j+1)$. We note that the blocks in the output arrays $(i, j+1)$ have longer edges than the blocks in the input array and that there are fewer blocks in each array. It follows that as we continue the interations over j, the arrays $(i, j+1)$ contain progressively coarser views of the original data set. A more detailed description follows.

A d-dimensional sequence a is indexed by elements in $\mathbb{Z}^d$ (i.e., a vector of length d, *containing signed integers*) denoted by $(a(m))_{m \in \mathbb{Z}^d}$. (The notation $m \in \mathbb{Z}^d$ should be read as: m is in the set $\mathbb{Z}^d$). If $(a(m))_{m \in \mathbb{Z}^d}$ and $(b(m))_{m \in \mathbb{Z}^d}$ are two sequences, then one defines their convolution, denoted by $a * b$, as

$$(a * b)(m) := \sum_{n \in \mathbb{Z}^d} a(n-m)b(n),\ m \in \mathbb{Z}^d.$$

Define the downsampling of a, denoted by $a\downarrow$, as $a\downarrow(m) := a(2m)$, that is, drop all the noneven entries from a to obtain the subsampled $a\downarrow$. In a similar manner, define the upsampling of a, denoted by $a\uparrow$, by $a\uparrow(2m) := a(m)$ for the even multi-indices and 0 otherwise.

The data sets under consideration are d-dimensional, but they are also of a finite size. For the purposes of the following discussion, regard them as d-dimensional sequences (and thereby infinite) by extending them by zeros.

Denote the d-dimensional data set by $F_{0,0}$. The wavelet transform at each iteration is implemented through convolution with 2^d fixed sequences x_i, $i = 0,\ldots,r := 2^d - 1$, which are referred to hereafter as *filters*. These filters correspond to a discretization of the chosen wavelet basis. We refer to x_0 as a low-pass (thumbnail) filter and $x_i, i > 0$ as high-pass (detail) filters.

The Wavelet Transform—The Algorithm Itself

Wavelet Decomposition

Assume that $F_{0,0}$ is given

```
for j = 1,2,...,j0
   for i = 0,...,r
      F_{i,j} ← (x_i * F_{0,j-1})↓
   end
end
```

At this point $(F_{i,j})_{i=1,j=1}^{r,j_0}$ and F_{0,j_0} form the decompositions.

Wavelet Reconstruction

Assume that $(F_{i,j})_{i=1,j=1}^{r,j_0}$ and F_{0,j_0} are given

```
for j = j0,...,2,1
   F_{0,j-1} ← Σ_{l=1}^{r} x̄_i * (F_{i,j}↑)
end
```

At this point $F_{0,0}$ is the reconstruction where the filter $\bar{x}_i$ is defined by $\bar{x}_i(m) := x_i(-m)$. In the description above, the initial assumption was that $F_{0,0}$ contained the original image and was zero outside of the image boundary. However, this abrupt drop to zero introduces unnecessary edges and would lead to inefficient compression and artifacts. There are other ways in which one can extend the original image and avoid this artifact; symmetric extension is most commonly used (Brislawn, 1996). We note that if the boundary conditions are dealt with properly, then all the objects $F_{i,j}$ can be regarded as finite arrays instead of infinite sequences and a d-dimensional data set with N points is mapped into a total of N coefficients, stored in $(F_{i,j})_{i=1,j=1}^{r,j_0}$ and F_{0,j_0}. Furthermore, $F_{i,j+1}$ contains approx. $1/2^d$ as many points as $F_{i,j}$.

In the decomposition algorithm, effectively $F_{i,j}$ is computed by convolving the original data set with x_0 and downsampling $j+1$ times, followed by a convolution with x_i and downsampling. Because x_0 is a low-pass filter, it is clear that with $i > 0$ fixed, $F_{i,j}$ contains progressively coarser information about the original data set as j increases.

The filters we employ were selected based on simple criteria: In the 2D sections, we must use filters that perform well in image compression; in the third spatial dimension and in the time dimension, we must use short filters because we may not have many samples in those directions. Furthermore, the wavelet decomposition followed by the wavelet reconstruction must, of course, yield the original data set. Thus, one is left with only a few choices, and the filters we chose are based on the so-called (univariate) 6/10 biorthogonal system in the 2D sections and the Haar filter in the third dimension and the time dimension. This implies that a typical filter x_i is a 4D array of size $m_1 \times m_2 \times 2 \times 2$, with $m_1, m_2 \in \{6,10\}$ (Daubechies, 1992; Strang & Nguyen, 1996).

Thresholding/Quantization

In compression the wavelet transform is followed by the thresholding step, also known as the quantization step. One chooses a threshold $t > 0$ and approximates all the wavelet coefficients calculated in the decomposition by the nearest integer multiple of t, that is, we approximate a wavelet coefficient $F_{i,j}(m)$ by $tI_{i,j}(m)$ where $I_{i,j}(m)$ is an integer.

Most of the wavelet coefficients are small, so with a moderately large value of t, most of the $I_{i,j}(m)$ are 0.

The current approach is to fully decompose the original data set so that F_{0,j_0} only contains a single number, which is not quantized, but written to file with full precision.

Encoding the Wavelet Representation

The matrices $I_{i,j}$ obtained in the previous step are sparse integer matrices and it remains to find an efficient way of writing them to file, that is, convert to a string of 0s and 1s. To achieve this, we apply the encoder developed by Gao in his Ph.D. thesis (Gao et al., 1997) and start by splitting $I_{i,j}$ into three parts. Let v_i contain the absolute values of the nonzero values in $I_{i,j}$ (in lexicographical order), s_i contain their signs, and $S_{i,j}(m)$ be the Boolean indicator: 1 if $I_{i,j}(m)$ is nonzero, 0 if $I_{i,j}(m)$ is zero. One encodes the Boolean matrices $S_{i,j}$ as follows:

```
Let i > 0 be given and define S^F_{i,1} := S_{i,1}.
Let f_i, l_i and h_i be empty binary strings.
for j = 1,2,... j0
   let S^0_{i,j} := S^F_{i,j}
   for k = 1,...,d
      encode spatial correlations of S^{k-1}_{i,j} in the kth dimension, appending to h_i and l_i and storing the output matrix in S^k_{i,j}
   end
   encode frequency correlations between S^d_{i,j} and S_{i,j+1}, appending to f_i and l_i and storing the output matrix in S^F_{i,j+1}
end
```

Table 1. Spatial correlation encoding table

$S_{i,j}^{k-1}(m_1)$	$S_{i,j}^{k-1}(m_2)$	$S_{i,j}^{k}(m_3)$	h_i	l_i
0	0	0		
0	1	1	0	0
1	0	1	0	1
1	1	1	1	

Table 2. Frequency correlation encoding table

$S_{i,j}^{d}(m)$	$S_{i,j+1}(m)$	$S_{i,j+1}^{F}(m)$	f_i	l_i
0	0	0		
0	1	1	0	
1	0	1	1	1
1	1	1	1	0

At this point l_i, h_i, f_i, and S_{i,j_0}^{F} store all the information from $(S_{i,j})_{j=1}^{j_0}$.

Spatial correlations in dimension k are encoded as follows: Let e_k be the k-th unit vector, in d dimensions $m \in \mathbb{Z}^d$ be such that $m(k) = 0$ and l be such that $m_1 := m + 2le_k$ and $m_2 := m + (2l + 1)e_k$ are valid indices into $S_{i,j}^{k-1}$. Then we use Table 1 to determine $S_{i,j}^{k}(m_3)$, $m_3 := m + le_k$ and what, if anything, to append to h_i and l_i.

Generally, observations indicate that pairs of zeros will be dominant and so nothing is appended to h_i and l_i. Pairs of ones are second most likely, in which case only one number is appended to h_i and nothing to l_i in this highest probability case. The other two cases are the least likely and result in one symbol being added to each of h_i and l_i.

The frequency encoding of $S_{i,j}^{d}$ and $S_{i,j+1}$ is similar in that one computes an output matrix $S_{i,j+1}^{F}$, and values are possibly appended to f_i and l_i (Table 2).

Lossless Compression

The final steps in the encoding are to write S_{i,j_0}^{F}, l_i, h_i, f_i, s_i, and v_i from the previous step to file, and use a lossless compressor to compress that file. The authors have found that gzip (http://www.gzip.org) has the distinct advantages of being patent-free, fast, and performing very well with this particular kind of data.

Results and Discussion

Encoding into an Application

As a proof of concept and for testing purposes the wavelet code was incorporated into a test application, Wavelet Processor. This cross-platform JAVA application provided a graphical user interface for harnessing the multidimensional wavelet compression algorithm. The Wavelet Processor application does not fully exploit the full capabilities of the wavelet code in that it can only handle dimensions of time and space. However it shows promise as a simple tool for storing 4D data sets in a manner that significantly improves on more conventional JPEG algorithms. Whether it is the Wavelet Processor application or the wavelet code itself, there are three key elements (discussed below): data compression, threshold estimation, and data decompression.

Data Compression

As explained above, the spatial and temporal correlations in the data sets determine whether or not it is advantageous to process them as 2D, 3D, or 4D images. Furthermore, it may not be possible to fit an entire 4D data set into memory. For these reasons, the Wavelet Processor allows users to specify upper limits g_3 and g_t such that when a 4D data set is split into smaller blocks, each block contains no more than g_3 sections along the third dimension and g_t sections along the time axis. Each block is compressed independently of the others, so should the user choose both g_3 and g_t greater than 1, a true 4D compression takes place; otherwise either a 2D compression ($g_3 = g_t = 1$) or a 3D compression is performed on each block.

In addition to controlling the block size through g_3 and g_t, the user also controls the two parameters: threshold t and metric p. The threshold t is the basic parameter that controls the compression ratio and is described elsewhere in this article. The value of p indicates in what p norm one wishes to minimize the error of the decompressed images (Devore et al., 1992). The p norm of a sequence $(a(m))_{m\in\mathbb{Z}^d}$ is given by $(\sum_{m\in\mathbb{Z}^d}|a(m)|^p)^{1/p}$ when $0 < p < \infty$ and by $\sup_{m\in\mathbb{Z}^d}|a(m)|$ when $p = \infty$. Much more intuitively, the value of p describes one's preference for high frequency features in the images versus low frequency features as p increases. The default value of 2 is used to preserve energy in the standard L^2 sense.

Threshold Estimation

Compression involves a clear trade-off between the fidelity of the compressed image on the one hand and the rate of compression on the other hand. Neither of the two is controlled directly in the wavelet compression algorithm. Instead, the controlling parameter is threshold. Qualitatively, raising the threshold results in encoding the information in fewer wavelet blocks, leading to lower fidelity and a higher compression rate. However, the the exact improvement in the compression rate cannot be determined exactly in advance. Most users would like instead to control directly the compression rate. We use the following heuristic to estimate the threshold t such that the compression ratio with that threshold, call it $cr(t)$, produces a user-specified desired compression ratio c.

1. Given a desired compression ratio c, we want to find the threshold t such that the compression rate approximately equals c, that is, $c = cr(t)$.
2. Assume that the cost of encoding each nonzero wavelet coefficient is 6 bits, and find the threshold that achieves the given compression ratio per that assumption. Because of our assumptions, we can find this threshold by only doing repeated scans of the wavelet coefficients, and we do not need to perform any encoding. This gives us a threshold t_1.
3. Perform a full thresholding and encoding of the current wavelet coefficients using t_1 and measure the compression rate achieved, that is, calculate $cr(t_1)$. If we are below the target compression ratio, let t_2 be $1.5t$, otherwise let t_2 be $t_1/1.5$. Calculate $cr(t_2)$.
4. Until we have achieved convergence, use weighted linear interpolation with the weights $w_i := (c - cr(t_i))^{-2.5}$ to model the relationship between thresholds and compression ratio. Let the next threshold be the one that the model predicts to achieve the desired compression ratio.

Data Decompression

The compressed file stores the settings necessary to decompress the data set, that is, g_3, g_t, p and t. In ordinary use, the user only needs to decide on where to store the decompressed images before the decompression can start. However, the wavelet code also provides the option to decompress into reduced size images that could, for example, serve as thumbnails for an interactive browser through the compressed data sets.

Results of the Multidimensional Wavelet Compression

The multidimensional wavelet code has been evaluated extensively on multidimensional light microscopy data from two distinct sources: Nomarski (DIC) and laser scanning microscopy (confocal and multiphoton). As these data sets have different fundamental attributes due to the microscopy technique itself, we ran extensive sets experimenting with groupings and compression testings. These tests revealed no substantial gains in one setting over another, and instead showed that across the board the wavelet processor yielded excellent compression with both data types using the same settings. We found that, as expected, both wavelet 2D and 3D performed far better than JPEG, with wavelet 3D performing the best of all (Fig. 1). Moreover, testing revealed significant increases in compressor performance when working with higher dimensional data (Fig. 2).

Benchmarks of the Multidimensional Wavelet Code

To benchmark the multidimensional wavelet code and at the same time make full use of its flexibility, we compressed two data sets at a compression rate of 1:50, using 2D, 3D, and 4D compression. We then decompressed them again, once into full-size images and once into half-size images. The benchmarks were performed on a 3.06 GHz Pentium IV with 1 GB of memory, running Red Hat Linux version 7.2. We used two different Java run-time environments, the IBM 32-bit SDK for Linux on Intel, version 1.4.1 and the Sun JDK, version 1.4.2 and they were allowed to use up to 400 MB of memory.

Table 3. Data set 1, wt-data set

	Compression		Decompression into original size		Decompression into half size	
	2D	3D	2D	3D	2D	3D
IBM1.4.1	24.6	25.6	31.7	61.7	78.7	205.2
Jdk1.4.2	9.6	7.8	14.2	38.6	41.3	171.3

Table 4. Data set 2, dub-data set

	Compression		Decompression into original size		Decompression into half size	
	2D	4D	2D	4D	2D	4D
IBM1.4.1	10.1	11.0	11.7	9.9	42.3	63.0
Jdk1.4.2	2.8	3.7	5.6	6.3	23.9	50.8

Data set 1, wt-data set

The original data set had 29 time points in it and each of them had 15 2D slices. Each slice was of the size 452×340 pixels. The thumbnails were of the size 226×170. Table 3 shows the number of 2D slices processed per second. The 2D compression used $g_3 = 1$, $g_t = 1$, the 3D compression used $g_3 = 15$, $g_t = 1$.

Data set 2, dub-data set

We processed four time points, each had 32 2D slices of size 768×512 pixels. The thumbnails were of the size 384×256 pixels. The 2D compression used, the 4D compression used $g_3 = 4, g_t = 4$ (Table 4).

We note that although the multidimensional wavelet code can decompress into thumbnails or half-sized images, the Wavelet Processor cannot yet take advantage of that functionality.

Conclusions

This type of multidimensional wavelet compressor/encoder has previously only been used on simulated geophysical

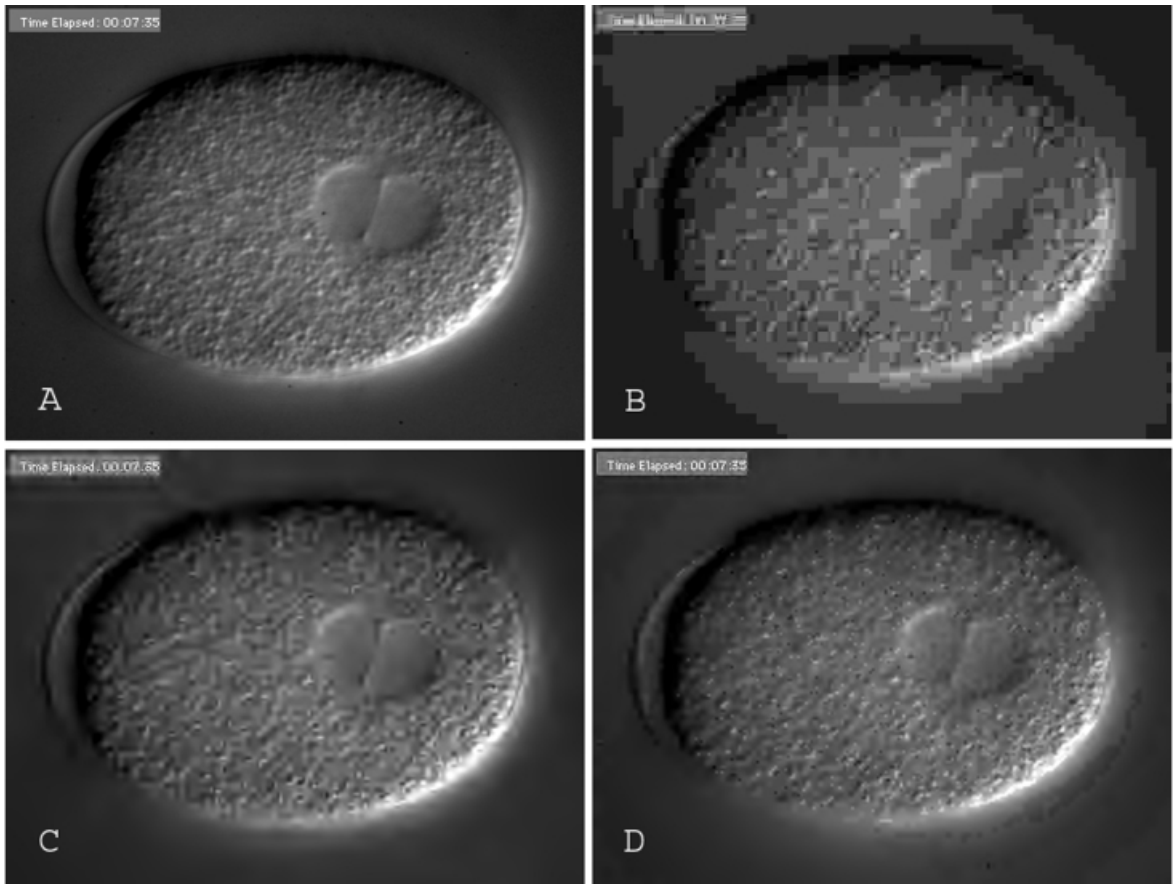

Figure 1. Comparison between uncompressed (**A**), JPEG 6.0 110× compressed (**B**), wavelet 110× compressed using 2D algorithm (**C**), and wavelet 110× compressed using 3D algorithm (**D**). Notice that both wavelet approaches perform significantly better than JPEG, with wavelet 3D performing the best of all. The image is taken from a 4D data set of a live one-cell *C. elegans* embryo recorded with Nomarski optics. The original data set was approximately 338.6 MB; at 110× compression it would be approximately 3.1 MB.

data to be able to monitor a supercomputer simulation over the internet in order to steer the "smooth," almost noise-free computation. This multidimensional method has not been previously used (to our knowledge) to process or analyze *real data* in any context, not just in biological applications as described here.

The multidimensional wavelet code that has been developed allows for a much greater flexibility than any other wavelet code of which the authors are aware and the Wavelet Processor makes good use of much of this functionality. The Wavelet Processor is based on the current needs for data compression and provides access to 2D, 3D, and 4D compression, whereas the code itself allows for compression in an arbitrary number of dimensions. The correlations in the data and the available computer memory for processing are the only limits on whether compression of higher dimensionalities would be feasible and beneficial.

Wavelet representations may be advantageous for some image processing task that are commonly applied to images from microscopes. Images of *in vivo* fluorescence are typically weak and suffer from photon noise. The noise can lead to poor compression and is often removed by spatial filtering (usually by convolving the image with a Gaussian mask). However these strategies lead to blurring of edges within the image. Wavelet-based denoising strategies have been described that can preserve edges in noisy confocal images (Boutet de Monvel et al., 2001).

The multidimensional wavelet code is written in Java and may therefore suffer the performance penalty that sometimes comes with this language on some platforms. However, we have demonstrated that, given an optimum choice of Java run-time environment (JRE), the multidimensional wavelet code has been shown to perform well enough to allow for interactive browsing through data sets.

We have shown that wavelet compression of multidimensional image data will allow libraries of 4D image data to be accessed remotely at a speed sufficient to allow animations to be used as a visualization aid by the investigator. Recently, a pioneering study used RNA interference to systematically suppress all the identified genes that are

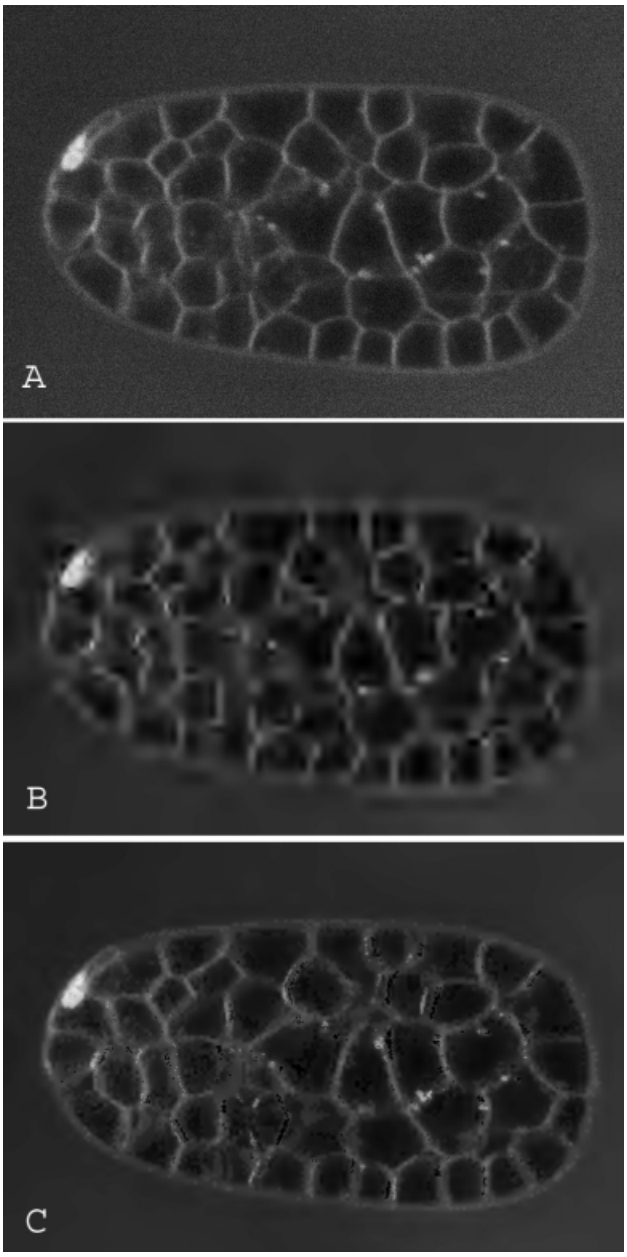

Figure 2. Illustration showing advantage of multidimensional wavelet compression at high compression rates. **A** is uncompressed, **B** is 800× compression with a 2D wavelet transform, and **C** is a 800× compression with a 4D wavelet transform. The 4D wavelet transform yields less compression artifacts and maintains details of the membrane structure. The image is taken from a 4D data set recorded on a multiphoton microscope of a live *C. elegans* embryo labeled with a fluorescent membrane probe—FM4-64. The original data set was approximately 1051.9 MB; at 800× compression it would be approximately 1.3 MB.

transcribed in *C. elegans* and observed the consequent phenotypes (Kamath & Ahringer, 2003). Simple Quicktime movie sequences of all embryos that exhibited embryonic defects were posted on an on-line database (http://nematoda.bio.nyu.edu/). Using multidimensional wavelet compression technology, it will be possible to make animations of 4D data sets available to remote investigators, thereby allowing full 3D visualization of the development of mutant embryos.

Availability of the Code

The multidimensional wavelet source code is freely available from our website (http://www.loci.wisc.edu/wavelet/) under the GNU public license. Our long-term plans are to integrate the wavelet code into a freely available database rather than as a stand-alone application. However, until then, the Wavelet Processor application is also available from the above website to interested parties.

Acknowledgments

The authors thank Scott Johnson and Curtis Rueden for their valuable input and discussion. In addition the authors thank Bill Mohler for providing multidimensional image data sets for testing purposes.

References

Boutet de Monvel, J., Le Calvez, S. & Ulfendahl, M. (2001). Image restoration for confocal microscopy: Improving the limits of deconvolution, with application to the visualization of the mammalian hearing organ. *Biophys J* **80**, 2455–2470.

Brislawn, C.M. (1996). Classification of nonexpansive symmetric extension transforms for multirate filter banks. *Applied and Computational Haromonic Analysis* **3**, 337–357.

Centonze, V.E. & White, J.G. (1998). Multiphoton excitation provides optical sections from deeper within scattering specimens than confocal imaging. *Biophys J* **75**, 2015–2024.

Chalfie, M., Tu, Y., Euskirchen, G., Ward, W.W. & Prasher, D.C. (1994). Green fluorescent protein as a marker for gene expression. *Science* **263**, 802–805.

Christopoulos, C.A., Skodras, A.N. & Ebrahimi, T. (2000). The JPEG2000 still image coding system: An overview. *IEEE Trans Consumer Electronics* **46**, 1103–1127.

Daubechies, I. (1992). *Ten Lectures on Wavelets*. CBMS Regional Conference Series in Applied Mathematics. Philadelphia, PA: SIAM.

Denk, W., Strickler, J.H. & Webb, W.W. (1990). Two-photon laser scanning fluorescence microscopy. *Science* **248**, 73–76.

Devore, R., Jawerth, B. & Lucier, B. (1992). Image compression through transform coding. *IEEE Proc Info Theory* **38**, 719–746.

Eliceiri, K.W., Rueden, C., Mohler, W.A., Hibbard, W.L. & White, J.G. (2002). Analysis of multidimensional biological image data. *Biotechniques* **33**, 1268–1273.

Gao, Z., Andreev, A. & Sharpley, R.C. (1997). Data compression and elementary encoding of wavelet coefficients. Technical Report, Columbia, SC: The University of South Carolina Math Department.

Kamath, R.S. & Ahringer, J. (2003). Genome-wide RNAi screening in *Caenorhabditis elegans*. *Methods* **30**, 313–321.

Nomarski, G. (1955). Microinterferometre differentiel a ondes polarisees. *J Phys Radium* **16**, 9–13.

Skodras, A.N., Christopoulos, C.A. & Ebrahimi, T. (2001). JPEG2000: The upcoming still image compression standard. *Patt Recog Lett* **22**, 1337–1345.

Squirrell, J.M., Wokosin, D.L., White, J.G. & Bavister, B.D. (1999). Long-term two-photon fluorescence imaging of mammalian embryos without compromising viability. *Nature Biotechnol* **17**, 763–767.

Strang, G. & Nguyen, B. (1996). *Wavelets and Filter Banks*. Wellesley, MA: Wellesley-Cambridge Press.

Thomas, C., DeVries, P., Hardin, J. & White, J. (1996). Four-dimensional imaging: Computer visualization of 3D movements in living specimens. *Science* **273**, 603–607.