A Data Compression Method for Image Time Series

Mark S. Cohen*

UCLA Brain Mapping Division, Ahmanson-Lovelace Brain Mapping Center, Los Angeles, California

Edited by: Karl Friston, Associate Editor

Abstract: Image time series for functional studies frequently vary relatively little from one time point to the next. We describe here a method of lossless file compression that calculates the change in signal intensity as a char (one byte), as opposed to a short or long integer (usually two or four bytes), thus resulting in a compressed file size just over half the size of the original (a 2:1 compression ratio). The calculations and format are exceedingly simple; the compression time is comparable to the time required for a file copy, and is thus transparent to most users. The approach works equally well on any integer-based scanning modality, such as PET, CT, or DSA. Our purpose in this communication is to advocate it as a potential standard, to promote free interchange of scan data. *Hum. Brain Mapping* 12: 20–24, 2001. © 2001 Wiley-Liss, Inc.

Key words: compression; fMRI; PACS; Gzip; compression ratio; compression/expansion time

INTRODUCTION

The collection of large numbers of serial images is common in functional imaging studies, such as fMRI [Kwong et al., 1992; Ogawa et al., 1992], cerebral blood volume measurement [Rosen et al., 1989], DSA [Di Chiro et al., 1982; Kruger et al., 1983], MR mammography [Kaiser and Zeitler, 1989; Stack et al., 1990], and others. A typical functional neuroimaging study in our laboratory, for example, involves the collection of

about 500 Mbytes of digital data and causes considerable burden on the data storage, archiving, backup, and network systems. Commonly available lossless data compression algorithms such as gzip (Open Software Foundation, Cambridge, MA), compress (Digital unix 4.0, Compaq Computer, Houston, TX), or Stuffit (Aladdin Systems, Watsonville, CA) can offer compression ratios of as much as 2:1, but are relatively slow both in compression and in expansion, adding a significant time penalty to the entire data analysis stream. For statistical quantitation, that is common to most functional neuroimaging, lossy compression schemes are generally unacceptable.

The reason for the poor efficiency of the standard tools is that the image data typically contain noise and thus appear as essentially random numbers to the compression programs. By recognizing the repetitive structure of the imaging time series, it is easy to develop a much more efficient algorithm as described here.

Portions of this material were presented previously at the International Society for Magnetic Resonance in Medicine, April 2000. Contract grant sponsor: National Institute on Drug Abuse; grant number R01-DA13054-01.

*Correspondence to: Mark S. Cohen, Ph.D., UCLA Brain Mapping Division, Ahmanson-Lovelace Brain Mapping Center, 660 Charles E. Young Dr. South, Los Angeles, CA 90095-7085.

E-mail: mscohen@ucla.edu

Received for publication 31 May 2000; accepted 14 September 2000

METHODS

Algorithm

MR data, direct from the scanners, are usually stored as two-byte short integers, as this reflects the resolution of today's analog to digital convertors, and allow a signal to noise ratio of about 60,000:1—substantially better than the native MR signal. Typical functional data sets, however, track only the temporal changes in the signal intensity, which are usually much smaller than the base signal. In most cases, in our experience, the magnitude of the signal change can be described fully in one byte of the two-byte data. Even when the functionally significant changes are large, most of the pixel locations are either image background or changing very little. Were the actual data scaled such that the full range of the short integers was used, this would only allow us to track changes of about 0.4% (1 part in 256). In practice, however, the images are usually scaled to use only a fraction of the available dynamic range, using mean values of about 1,000 or 2,000 (arbitrary signal units) to accommodate the largest possible signals without digital saturation or wrap.

To compress such time series, therefore, we calculate a difference image that distinguishes each image from the one immediately preceding. Most of the values in the difference image can be expressed in one byte. For the few whose intensity difference exceeds one byte (absolute value >127), we list the location and create an array of two-byte differences. The final compressed file therefore contains a small descriptive header, and the first time-point, expressed as two-byte integers. For each successive time point, we save an array containing the differences that exceed one byte, and an image array of the single-byte differences for this time point. Figure 1 is a diagram of the file structure. Constructing the compressed file requires only addition, and is thus exceeding fast on today's computers. The method can be applied in the same way to structural images. When used in this way, successive images are compared to the adjacent slice plane. To the extent that the anatomy varies slowly (e.g., with thin sections) the method can be very efficient.

We made benchmark measurements using Stuftit (Aladdin Systems, Watsonville, CA), gzip, compress, and MacGzip (Jean-loup Gailly; MacSPD@ivo.cps.unizar.es). These programs were selected because they are widespread, publicly available, and the most commonly used tools in the functional imaging community. Compression/expansion times are reported to the nearest second. We also measured the time re-

Figure 1.

SmallTime compressed file format. The compressed file has an initial binary header that describes overall size characteristics and contains basic information on the input image. This is followed by the original image header so that it can be restored completely. The first image in the time series is saved uncompressed (two bytes/pixel). Subsequent images are saved as variable length compressed structures. The first four bytes indicate the number of pixels whose intensity difference exceeds one byte. This is followed by an array of the locations and intensity differences for the over range intensity differences. Finally, we save the intensity difference from the prior image as an array of single bytes.

quired for a local file copy, to determine what fraction of the time was spent on disk I/O versus calculation. Finally, we express the compression efficiency as: Original File Size : Final File Size

Test data

To verify the compression efficiency and the compression and expansion times, we have used a variety of functional neuroimaging data, including primarily echo-planar data sets of 100 or more time points and 10 or more image locations using 64×64 , 128×64 , or 128×128 images. The performance data reported in the following tables are from a *typical* functional MRI data set consisting of 120 time points, each containing 16 images with a 64×64 pixel imaging matrix. This data set is 15,728,640 bytes in length. Specifically, these data study the neural activation associated with a simple hand motor task. It is important to note that the efficiency of the file compression is data dependent: the greater the signal changes, and the greater their extent, the less efficient the algorithm will be.

TABLE I. Comparison of compression results using SmallTime, and a variety of publicly available compression algorithms.

Program	Compressed size	Compression ratio	Compression time (sec)	Expansion time (sec)
gzip (unix)	10,299,583	1.53:1	49	5
gzip (Mac)	10,297,761	1.53:1	41	9
Compress (unix)	11,109,055	1.42:1	15	26
Stuffit (Mac)	8,058,127	1.95:1	69	134
SmallTime (unix)	8,045,416	1.95:1	<2	<2
SmallTime (Mac)	8,045,416	1.95:1	8	6

The original input file is 15,728,640 bytes in length. The compressed size is the file size after compression. Compression and expansion times are listed in seconds.

Though designed originally for time series, we also tested the efficacy of SmallTime on structural data sets. For this purpose, we report the results on a 3D volume data set consisting of 124 contiguous 256×256 slices acquired with a SPGR imaging sequence.

Computers

The results reported here were obtained on two computer platforms: an Apple Macintosh® 333 MHz PowerBook G3 laptop (Apple Computer, Cupertino, CA), and a Digital Alpha 500/266 Workstation (Compaq computer, Houston, TX), each with 192 Mbyte of RAM. The Macintosh versions of the program were compiled under MetroWerks Code Warrior version 5, and the digital versions were compiled under Digital UNIX 4.0.

RESULTS

Table I shows the compressed file size, compression time, and expansion time using each of the compression programs on the Apple PowerBook and the Digital Workstation. In the table, the best results are indicated in bold. The fastest compression time was achieved using our algorithm (which we call “SmallTime”). Aladdin Systems’ “Stuffit” produced almost equal file sizes but required the longest compression and expansion times by far. Based on expansion and compression times, the best unix program was “compress,” but the efficiency was very poor; the files were only reduced in size by 29.4%. Figure 2A and 2B show the same information graphically.

Using our 3D volume data, SmallTime achieved a compression ratio of 1.94:1. Compressing the 124-slice volume took 4.3 sec and decompression required 3.6 sec on the Macintosh platform.

To understand better the time efficiencies possible with SmallTime, we compared the compression and

expansion times to simple file copies using a 141,557,760 byte file on our Alphastation. In this case, file copy (using cp) took 42 sec, whereas compression

Figure 2.

Comparison of compression results using SmallTime and a variety of publicly available compression algorithms. (A) final compression ratio, (B) compression time.

required only 30 sec. Expansion time using SmallTime was 25 sec and the compression ratio was 1.99:1.

CONCLUSIONS

We have demonstrated that very rapid and efficient compression is possible using a simple algorithm that takes into account basic information about the structure of digital image data that changes in time.

The algorithm requires that at least one 2-byte image be included in the files. Therefore, for an image with dimensions of M pixels $\times$ N pixels $\times$ T time points, the theoretical maximum compression ratio is

$$2MNT:MN(T + 1),$$

which approaches 2:1 as T becomes infinite.

Our results suggest that the method is actually faster than file copies for both compression and decompression. As a result, it can be deployed as a regular part of the data transfer process from the scanner to any choice of archive media without significant time impact. Once the data are compressed, of course, all further file operations will be much improved.

The times listed in Table I may appear too short to be of much interest, even for the worst case. However, a 15-Mbyte set such as this represents only about 1/20th of our usual data collection. Thus, compression with gzip adds almost 20 min of compute time to the study. Further, the ultimate file size is a major factor in the time required for archiving, network transfer, and so on. Our lab, for example, routinely uses ftp to move data from the scanners to our archiving computer (which takes about 5 min for the uncompressed files). The process of archiving the data to compact discs requires about an additional 20 min. We estimate conservatively that the proposed compression method shortens processing time by at least 30 min for each study, and by much more if the data are routinely moved, compressed, and expanded during future analyses.

While this report considers primarily image time series, we obtain similar results with multislice volume images. This is due, of course, to the relatively small differences between the signal in successive image slices and the fact that a large fraction of the image field of view is typically background. SmallTime is thus a reasonable alternative for compression of conventional (single time point) data sets.

We have tested a much more efficient approach that calculates a noise threshold based on statistics of the

image intensity distribution and saves information only on the supra-threshold pixels. This method is, of course, no longer lossless, but does reduce file sizes to less than one-quarter their original size for typical brain images, depending on the field of view. Unfortunately, this approach is not acceptable for functional neuroimaging, as it does not properly handle changes in head position during the scan series. Finally, it is worth noting that publicly available lossy compression systems such as MPEG and JPEG are unsuitable for quantitative statistical analyses as commonly performed for functional imaging studies [Breeuwer et al., 1995].

We believe that a rapid, simple, and efficient means of lossless data compression will be of value to the imaging community. We have therefore made the source and executables available at our web site (<http://www.brainmapping.org/SmallTime/>) for public download. At the moment, we have source code available for Macintosh and console driven ANSI C environments that will support a variety of standard image formats including Mayo Clinic "Analyze," Advanced NMR Systems APD2, and the ".bshort" file type developed at Massachusetts General Hospital. To the extent that these formats save volume data, SmallTime can be used as is to compress along either temporal or spatial dimensions. The option is included to save as differences either between successive time points, or from the first image or time point. While the latter may be offer the advantage that a randomly accessed time point could be retrieved without reconstructing earlier images, it is less efficient in compressing spatial data or in accounting for instrumental or experimental drifts in signal intensity. We encourage interested users to download, modify, and send back to us any improved versions of the source and any desired extensions to other formats. Further, the best efficiencies in overall data handling will be achieved when compression and decompression are built directly into the image analysis tools.

ACKNOWLEDGMENTS

We acknowledge the generous support of the Brain Mapping Medical Research Foundation, the Ahmanson Foundation, the Pierson-Lovelace Foundation, the Jennifer Jones Simon Foundation and the Tamkin Foundation, and the UCLA Council on Research.

REFERENCES

- Breeuwer M, Heusdens R, et al (1995): Data compression of x-ray cardio-angiographic image series. *Int J Card Imaging* 11(Suppl 3):179-186.

- Di Chiro G, Rieth KG, et al (1982): Digital subtraction angiography and dynamic computed tomography in the evaluation of arteriovenous malformations and hemangioblastomas of the spinal cord. *J Comput Assist Tomogr* 6:655–670.
- Kaiser WA, Zeitler E (1989): MR imaging of the breast: fast imaging sequences with and without Gd- DTPA. Preliminary observations. *Radiology* 170(Pt 1):681–686.
- Kruger RA, Nelson JA, et al (1983): Dynamic tomographic digital subtraction angiography using temporal filtration. *Radiology* 147:863–867.
- Kwong KK, Belliveau JW, et al (1992): Dynamic magnetic resonance imaging of human brain activity during primary sensory stimulation. *Proc Natl Acad Sci U S A* 89:5675–5679.
- Ogawa S, Tank DW, et al (1992): Intrinsic signal changes accompanying sensory stimulation: functional brain mapping with magnetic resonance imaging. *Proc Natl Acad Sci U S A* 89:5951–5955.
- Rosen B, Belliveau J, et al (1989): Perfusion imaging by nuclear magnetic resonance. *Magn Res Q* 5:263–281.
- Stack JP, Redmond OM, et al (1990): Breast disease: tissue characterization with Gd-DTPA enhancement profiles. *Radiology* 174:491–494.