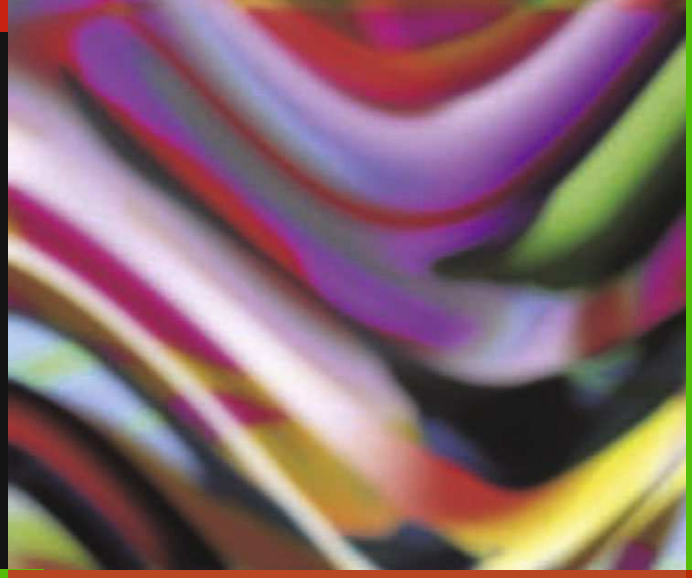
David Salomon

Data Compression

The Complete Reference

Fourth Edition

Springer



David Salomon

Data Compression

The Complete Reference

Fourth Edition

tion

Springer

Data Compression

Fourth Edition

David Salomon

With Contributions by Giovanni Motta and David Bryant

Data Compression

The Complete Reference

Fourth Edition

Springer

Professor David Salomon (emeritus)
Computer Science Department
California State University
Northridge, CA 91330-8281
USA

Email: david.salomon@csun.edu

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

Library of Congress Control Number: 2006931789

ISBN-10: 1-84628-602-6 e-ISBN-10: 1-84628-603-4
ISBN-13: 978-1-84628-602-5 e-ISBN-13: 978-1-84628-603-2

Printed on acid-free paper.

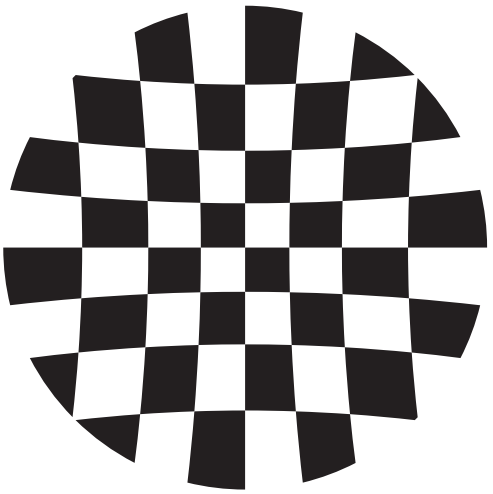
9 8 7 6 5 4 3 2 1

Springer Science+Business Media, LLC
springer.com

Data Compression

Fourth Edition

To Wayne Wheeler, an editor par excellence



Write your own story. Don't let others write it for you.
Chinese fortune-cookie advice

Preface to the Fourth Edition

I was pleasantly surprised when in November 2005 a message arrived from Wayne Wheeler, the new computer science editor of Springer Verlag, notifying me that he intends to qualify this book as a Springer major reference work (MRW), thereby releasing past restrictions on page counts, freeing me from the constraint of having to compress my style, and making it possible to include important and interesting data compression methods that were either ignored or mentioned in passing in previous editions.

> These fascicles will represent my best attempt to write a comprehensive account, but computer science has grown to the point where I cannot hope to be an authority on all the material covered in these books. Therefore I'll need feedback from readers in order to prepare the official volumes later.
>
> I try to learn certain areas of computer science exhaustively; then I try to digest that knowledge into a form that is accessible to people who don't have time for such study.
>
> —Donald E. Knuth, `http://www-cs-faculty.stanford.edu/~knuth/` (2006)

Naturally, all the errors discovered by me and by readers in the third edition have been corrected. Many thanks to all those who bothered to send error corrections, questions, and comments. I also went over the entire book and made numerous additions, corrections, and improvements. In addition, the following new topics have been included in this edition:

- Tunstall codes (Section 2.4). The advantage of variable-size codes is well known to readers of this book, but these codes also have a downside; they are difficult to work with. The encoder has to accumulate and append several such codes in a short buffer, wait until n bytes of the buffer are full of code bits (where n must be at least 1), write the n bytes on the output, shift the buffer n bytes, and keep track of the location of the last bit placed in the buffer. The decoder has to go through the reverse process. The idea of Tunstall codes is to construct a set of fixed-size codes, each encoding a variable-size string of input symbols. As an aside, the "pod" code (Table 7.29) is also a new addition.

- Recursive range reduction (3R) (Section 1.7) is a simple coding algorithm due to Yann Guidon that offers decent compression, is easy to program, and its performance is independent of the amount of data to be compressed.

- LZARI, by Haruhiko Okumura (Section 3.4.1), is an improvement of LZSS.

- RAR (Section 3.20). The popular RAR software is the creation of Eugene Roshal. RAR has two compression modes, general and special. The general mode employs an LZSS-based algorithm similar to ZIP Deflate. The size of the sliding dictionary in RAR can be varied from 64 Kb to 4 Mb (with a 4 Mb default value) and the minimum match length is 2. Literals, offsets, and match lengths are compressed further by a Huffman coder. An important feature of RAR is an error-control code that increases the reliability of RAR archives while being transmitted or stored.

- 7-z and LZMA (Section 3.24). LZMA is the main (as well as the default) algorithm used in the popular `7z` (or `7-Zip`) compression software [7z 06]. Both `7z` and LZMA are the creations of Igor Pavlov. The software runs on Windows and is free. Both LZMA and `7z` were designed to provide high compression, fast decompression, and low memory requirements for decompression.

- Stephan Wolf made a contribution to Section 4.30.4.

- H.264 (Section 6.8). H.264 is an advanced video codec developed by the ISO and the ITU as a replacement for the existing video compression standards H.261, H.262, and H.263. H.264 has the main components of its predecessors, but they have been extended and improved. The only new component in H.264 is a (wavelet based) filter, developed specifically to reduce artifacts caused by the fact that individual macroblocks are compressed separately.

- Section 7.4 is devoted to the `WAVE` audio format. `WAVE` (or simply Wave) is the native file format employed by the Windows opearting system for storing digital audio data.

- FLAC (Section 7.10). FLAC (free lossless audio compression) is the brainchild of Josh Coalson who developed it in 1999 based on ideas from Shorten. FLAC was especially designed for audio compression, and it also supports streaming and archival of audio data. Coalson started the FLAC project on the well-known sourceforge Web site [sourceforge.flac 06] by releasing his reference implementation. Since then many developers have contributed to improving the reference implementation and writing alternative implementations. The FLAC project, administered and coordinated by Josh Coalson, maintains the software and provides a reference codec and input plugins for several popular audio players.

- WavPack (Section 7.11, written by David Bryant). WavPack [WavPack 06] is a completely open, multiplatform audio compression algorithm and software that supports three compression modes, lossless, high-quality lossy, and a unique hybrid compression mode. It handles integer audio samples up to 32 bits wide and also 32-bit IEEE floating-point data [IEEE754 85]. The input stream is partitioned by WavPack into blocks that can be either mono or stereo and are generally 0.5 seconds long (but the length is actually flexible). Blocks may be combined in sequence by the encoder to handle multichannel audio streams. All audio sampling rates are supported by WavPack in all its modes.

- Monkey's audio (Section 7.12). Monkey's audio is a fast, efficient, free, lossless audio compression algorithm and implementation that offers error detection, tagging, and external support.

- MPEG-4 ALS (Section 7.13). MPEG-4 Audio Lossless Coding (ALS) is the latest addition to the family of MPEG-4 audio codecs. ALS can input floating-point audio samples and is based on a combination of linear prediction (both short-term and long-term), multichannel coding, and efficient encoding of audio residues by means of Rice codes and block codes (the latter are also known as block Gilbert-Moore codes, or BGMC [Gilbert and Moore 59] and [Reznik 04]). Because of this organization, ALS is not restricted to the encoding of audio signals and can efficiently and losslessly compress other types of fixed-size, correlated signals, such as medical (ECG and EEG) and seismic data.

- AAC (Section 7.15). AAC (advanced audio coding) is an extension of the three layers of MPEG-1 and MPEG-2, which is why it is often called `mp4`. It started as part of the MPEG-2 project and was later augmented and extended as part of MPEG-4. Apple Computer has adopted AAC in 2003 for use in its well-known iPod, which is why many believe (wrongly) that the acronym AAC stands for apple audio coder.

- Dolby AC-3 (Section 7.16). AC-3, also known as Dolby Digital, stands for Dolby's third-generation audio coder. AC-3 is a perceptual audio codec based on the same principles as the three MPEG-1/2 layers and AAC. The new section included in this edition concentrates on the special features of AC-3 and what distinguishes it from other perceptual codecs.

- Portable Document Format (PDF, Section 8.13). PDF is a popular standard for creating, editing, and printing documents that are independent of any computing platform. Such a document may include text and images (graphics and photos), and its components are compressed by well-known compression algorithms.

- Section 8.14 (written by Giovanni Motta) covers a little-known but important aspect of data compression, namely how to compress the differences between two files.

- Hyperspectral data compression (Section 8.15, partly written by Giovanni Motta) is a relatively new and growing field. Hyperspectral data is a set of data items (called pixels) arranged in rows and columns where each pixel is a vector. A home digital camera focuses visible light on a sensor to create an image. In contrast, a camera mounted on a spy satellite (or a satellite searching for minerals and other resources) collects and measures radiation of many wavelegths. The intensity of each wavelength is converted into a number, and the numbers collected from one point on the ground form a vector that becomes a pixel of the hyperspectral data.

Another pleasant change is the great help I received from Giovanni Motta, David Bryant, and Cosmin Truţa. Each proposed topics for this edition, went over some of the new material, and came up with constructive criticism. In addition, David wrote Section 7.11 and Giovanni wrote Section 8.14 and part of Section 8.15.

I would like to thank the following individuals for information about certain topics and for clearing up certain points. Igor Pavlov for help with `7z` and LZMA, Stephan Wolf for his contribution, Matt Ashland for help with Monkey's audio, Yann Guidon

for his help with recursive range reduction (3R), Josh Coalson for help with FLAC, and Eugene Roshal for help with RAR.

> In the first volume of this biography I expressed my gratitude to those individuals and corporate bodies without whose aid or encouragement it would not have been undertaken at all; and to those others whose help in one way or another advanced its progress. With the completion of this volume my obligations are further extended. I should like to express or repeat my thanks to the following for the help that they have given and the premissions they have granted.
>
> Christabel Lady Aberconway; Lord Annan; Dr Igor Anrep; ...
>
> —Quentin Bell, *Virginia Woolf: A Biography* (1972)

Currently, the book's Web site is part of the author's Web site, which is located at `http://www.ecs.csun.edu/~dsalomon/`. Domain `DavidSalomon.name` has been reserved and will always point to any future location of the Web site. The author's email address is `dsalomon@csun.edu`, but email sent to ⟨*anyname*⟩`@DavidSalomon.name` will be forwarded to the author.

Those interested in data compression in general should consult the short section titled "Joining the Data Compression Community," at the end of the book, as well as the following resources:

- `http://compression.ca/`,
- `http://www-isl.stanford.edu/~gray/iii.html`,
- `http://www.hn.is.uec.ac.jp/~arimura/compression_links.html`, and
- `http://datacompression.info/`.

(URLs are notoriously short lived, so search the Internet).

> People err who think my art comes easily to me.
>
> —Wolfgang Amadeus Mozart

Lakeside, California David Salomon

Preface to the Third Edition

I was pleasantly surprised when in December 2002 a message arrived from the editor asking me to produce the third edition of the book and proposing a deadline of late April 2003. I was hoping for a third edition mainly because the field of data compression has made great strides since the publication of the second edition, but also for the following reasons:

Reason 1: The many favorable readers' comments, of which the following are typical examples:

First I want to thank you for writing "Data Compression: The Complete Reference." It is a wonderful book and I use it as a primary reference.

I wish to add something to the errata list of the 2nd edition, and, if I am allowed, I would like to make a few comments and suggestions.. . .

—Cosmin Truţa, 2002

---

sir,

i am ismail from india. i am an computer science engineer. i did project in data compression on that i open the text file. get the keyword (symbols,alphabets,numbers once contained word). Then sorted the keyword by each characters occurrences in the text file. Then store the keyword in a file. then following the keyword store the 000 indicator.Then the original text file is read. take the first character of the file.get the positional value of the character in the keyword. then store the position in binary. if that binary contains single digit, the triple bit 000 is assigned. the binary con two digit, the triple bit 001 is assigned. so for 256 ascii need max of 8 digit binary.plus triple bit .so max needed for the 256th char in keyword is 11 bits. but min need for the first char in keyworkd is one bit+three bit , four bit. so writing continuously o's and 1's in a file. and then took the 8 by 8 bits and convert to equal ascii character and store in the file. thus storing keyword + indicator + converted ascii char
can give the compressed file.

then reverse the process we can get the original file.
These ideas are fully mine.

(See description in Section 3.2).

Reason 2: The errors found by me and by readers in the second edition. They are listed in the second edition's Web site, and they have been corrected in the third edition.

Reason 3: The title of the book (originally chosen by the publisher). This title had to be justified by making the book a complete reference. As a result, new compression methods and background material have been added to the book in this edition, while the descriptions of some of the older, obsolete methods have been deleted or "compressed." The most important additions and changes are the following:

- The BMP image file format is native to the Microsoft Windows operating system. The new Section 1.4.4 describes the simple version of RLE used to compress these files.

- Section 2.5 on the Golomb code has been completely rewritten to correct mistakes in the original text. These codes are used in a new, adaptive image compression method discussed in Section 4.22.

- Section 2.9.6 has been added to briefly mention an improved algorithm for adaptive Huffman compression.

- The PPM lossless compression method of Section 2.18 produces impressive results, but is not used much in practice because it is slow. Much effort has been spent exploring ways to speed up PPM or make it more efficient. This edition presents three such efforts, the PPM* method of Section 2.18.6, PPMZ (Section 2.18.7), and the fast PPM method of Section 2.18.8. The first two try to explore the effect of unbounded-length contexts and add various other improvements to the basic PPM algorithm. The third attempts to speed up PPM by eliminating the use of escape symbols and introducing several approximations. In addition, Section 2.18.4 has been extended and now contains some information on two more variants of PPM, namely PPMP and PPMX.

- The new Section 3.2 describes a simple, dictionary-based compression method.

- LZX, an LZ77 variant for the compression of cabinet files, is the topic of Section 3.7.

- Section 8.14.2 is a short introduction to the interesting concept of file differencing, where a file is updated and the differences between the file before and after the update are encoded.

- The popular Deflate method is now discussed in much detail in Section 3.23.

- The popular PNG graphics file format is described in the new Section 3.25.

- Section 3.26 is a short description of XMill, a special-purpose compressor for XML files.

- Section 4.6 on the DCT has been completely rewritten. It now describes the DCT, shows two ways to interpret it, shows how the required computations can be simplified, lists four different discrete cosine transforms, and includes much background material. As a result, Section 4.8.2 was considerably cut.

- An N-tree is an interesting data structure (an extension of quadtrees) whose compression is discussed in the new Section 4.30.4.

- Section 5.19, on JPEG 2000, has been brought up to date.

- MPEG-4 is an emerging international standard for audiovisual applications. It specifies procedures, formats, and tools for authoring multimedia content, delivering it, and consuming (playing and displaying) it. Thus, MPEG-4 is much more than a compression method. Section 6.6 is s short description of the main features of and tools included in MPEG-4.

- The new lossless compression standard approved for DVD-A (audio) is called MLP. It is the topic of Section 7.7. This MLP should not be confused with the MLP image compression method of Section 4.21.

- Shorten, a simple compression algorithm for waveform data in general and for speech in particular, is a new addition (Section 7.9).

- SCSU is a new compression algorithm, designed specifically for compressing text files in Unicode. This is the topic of Section 8.12. The short Section 8.12.1 is devoted to BOCU-1, a simpler algorithm for Unicode compression.

- Several sections dealing with old algorithms have either been trimmed or completely removed due to space considerations. Most of this material is available on the book's Web site.

- All the appendixes have been removed because of space considerations. They are freely available, in PDF format, at the book's Web site. The appendixes are (1) the ASCII code (including control characters); (2) space-filling curves; (3) data structures (including hashing); (4) error-correcting codes; (5) finite-state automata (this topic is needed for several compression methods, such as WFA, IFS, and dynamic Markov coding); (6) elements of probability; and (7) interpolating polynomials.

- A large majority of the exercises have been deleted. The answers to the exercises have also been removed and are available at the book's Web site.

I would like to thank Cosmin Truţa for his interest, help, and encouragement. Because of him, this edition is better than it otherwise would have been. Thanks also go to Martin Cohn and Giovanni Motta for their excellent prereview of the book. Quite a few other readers have also helped by pointing out errors and omissions in the second edition.

Currently, the book's Web site is part of the author's Web site, which is located at `http://www.ecs.csun.edu/~dsalomon/`. Domain `BooksByDavidSalomon.com` has been reserved and will always point to any future location of the Web site. The author's email address is `david.salomon@csun.edu`, but it's been arranged that email sent to ⟨*anyname*⟩`@BooksByDavidSalomon.com` will be forwarded to the author.

Readers willing to put up with eight seconds of advertisement can be redirected to the book's Web site from `http://welcome.to/data.compression`. Email sent to `data.compression@welcome.to` will also be redirected.

Those interested in data compression in general should consult the short section titled "Joining the Data Compression Community," at the end of the book, as well as the following resources:

- `http://compression.ca/`,
- `http://www-isl.stanford.edu/~gray/iii.html`,
- `http://www.hn.is.uec.ac.jp/~arimura/compression_links.html`, and
- `http://datacompression.info/`.

(URLs are notoriously short lived, so search the Internet).

> One consequence of the decision to take this course is that I am, as I set down these sentences, in the unusual position of writing my preface before the rest of my narrative. We are all familiar with the after-the-fact tone—weary, self-justificatory, aggrieved, apologetic—shared by ship captains appearing before boards of inquiry to explain how they came to run their vessels aground, and by authors composing forewords.
>
> —John Lanchester, *The Debt to Pleasure* (1996)

Northridge, California David Salomon