UDC 004.627
Serhiy USTENKO
ustenko.s.a@gmail.com
ORCID: 0000-0003-4968-1233
Serhiy LUKYANCHIKOV
lsd57@ukr.net
ORCID: 0000-0002-6837-2930
Mykolaiv
Ivan LUKYANCHIKOV
xman82@ukr.net
ORCID: 0000-0002-7031-4932
L'viv

A SOFTWARE FOR DATA COMPRESSION USING DOUBLE DIFFERENTIAL TRANSFORMATION OF SOUND SIGNAL

This work is dedicated to development of a software for lossless audio data compression using double differential transformation of sound signal.

Keywords: software, compression, decompression, software module, algorithm, lossless compression.

The software uses a method of lossless audio data compression using double differential transformation of sound signal [1, 2]. According to the functional requirements, the software should work in two modes:

1. Compression mode.
2. Decompression mode.

Since the software should, in fact, be an archiver for RIFF WAVE sound data files, the command line interface should be considered as the most natural one for this software.

In this case the actual working mode is being selected by command line keys with appropriate parameter values. Thus, two keys are needed — for compression and decompression modes respectively.

Parameter values for the compression key, according to the functional requirements, should be:

- number of bits for saving number of amplitude values in a chunk;
- sound data file name;
- archive file name.

Decompression keys should have only one parameter value — archive file name.

For better user experience it is nesessary to add one more special key without parameters for calling a brief help on how to use given software. The help should also be displayed in case if user haven't specified any command line keys.

Structure-wise, the software should contain 3 main functional modules:

- command line key analyzer module;
- sound data compression module;
- sound data decompression module.

Command line key analyzer module should be the main control module. It should call a functional module which implement desired working mode (compression, decompression) depending on the command line key. The functional diagram of software modules interaction is given by figure 1.

Sound data compression algorithm. Taking into account essence of sound data compression method using reduction of bits sufficient to represent sound signal amplitude values after double differential transformation, and according to given functional requirements for input/output data and software structure, the following sound data compression algorithm is proposed.

According to the mechanism of sound data compression using this method, it is proposed to use special coder buffer for better definition of optimal chunks of transformed sound signal amplitude values.

ISSN 2520-2820 (online), ISSN 2524-0978 (print)

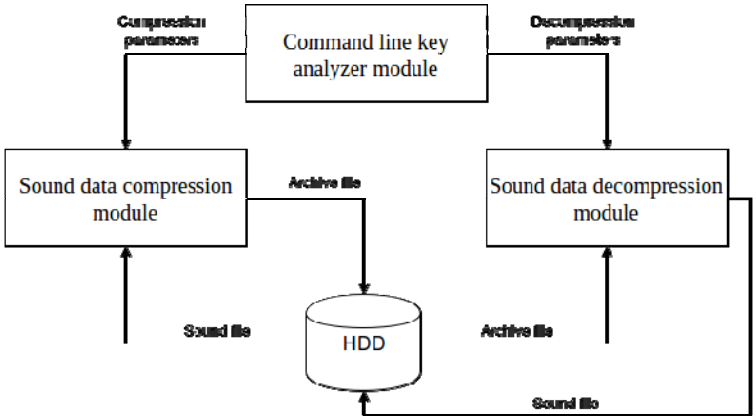

Figure 1. The functional diagram of software modules interaction

The mentioned coder buffer is needed as a temporary storage for transformed sound signal amplitude values from input sound file. Coder buffer should allow to store $l_{\max}$ of such ampitude values where $l_{\max}$ is a maximum number of amplitude values in a chunk, which is calculated depending on needed number of bits N allocated for its representation using formula:

$$l_{\max} = 2^N - 1. \tag{1}$$

Also, coder buffer should be organized as FIFO buffer: once older values are removed, all other values are shifted back, giving a free space of new ones.

According to the software specification, sound data compression algorithm should accept WAVE file name (RIFF WAVE format), archive file name and number of bits N for number of amplitude values in a chunk as input parameters.

An archive file should be generated by algoithm as an output.

Compression algorithm should have the following steps.

Step 1. Try to open WAVE file with given file name for reading. If file access error occurred, display error message and proceed to Step 9.

Step 2. Read WAVE file header and check if the file is mono, 16 bits per sample. If not — display a message about illegal file format and proceed to Step 9.

Step 3. Create archive file, write archive header and WAVE file header.

Step 4. Fill coder buffer with sound signal amplitude values from WAVE file being transformed using formula (3).

Step 5. Define optimal number of transformed amplitude values from coder buffer to represent as a chunk being transcoded using bit structure. Number of values in a chunk should be considered as optimal one if it gives maximum compression rate for this chunk; compression rate is calculated as:

$$K = \frac{m \cdot l}{N + 5 + n \cdot l}, \tag{2}$$

where m – initial number of bits per sound amplitude value before transformation.

Step 6. Remove selected chunk of transformed amplitude values from coder buffer, encode it as a bit structure and append to the archive file data.

Step 7. If end of WAVE file is not reached, proceed to Step 4.

Step 8. Display a message about successful WAVE file compression.

Step 9. End of algorithm.

Sound data decompression algorithm. For data decompression, being compressed using given method, there is no need to use coder buffer which is needed for compression stage. Decompression algorithm is much more trivial than one used for compression.

Algorithm needs only sound data archive file name as input data. Similar to compression algorithm, let's describe decompression algorithm step by step.

Step 1. Try to open archive file with given name for reading. If file access error occurred, display error message and proceed to Step 9.

Step 2. Read archive header. If archive identifier is not "DD_ARCHIVE" string, display a message about illegal archive format and proceed to Step 9.

Step 3. Read RIFF WAVE header and create corresponding WAVE file with the same name as compressed WAVE file has. Write 2 first amplitude values from archive header to the created WAVE file.

Step 4. Read a chunk of transformed amplitude values as a bit structure from the archive data.

Step 5. Decode a structure and perform reverse transformation of saved amplitude values using formula:

$$S_i = dd_i + 2S_{i-1} - S_{i-2}, \quad (3)$$

where S_i – audio amplitude value to restore; i – index of this value, $i = 3, 4, 5, ...$; dd_i – audio amplitude value from the archive (finite second order difference); S_{i-1}, S_{i-2} – previous restored amplitude values.

Step 6. Write restored amplitude values to WAVE file.

Step 7. If end of archive file not reached, proceed to Step 4.

Step 8. Display a message about successful decompression.

Step 9. End of algorithm.

As defined in requirements for the archive file structure, it should consist of the following items:

1) archive header;
2) header of WAVE file being compressed;
3) data area.

According to the archive file structure, archive header should contain following data:

1) archive identifier;
2) name of WAVE file being compressed;
3) number of bits N for number of amplitude values in a chunk;
4) two first sound amplitude values from WAVE file.

According to the requirements, archive identifier should be 10 byte string "DD_ARCHIVE", which is needed to distinguish if file is a sound data archive being compressed using given compression method.

Let's allocate 256 bytes for archived WAVE file name since this is the supported file name length for the most of modern file systems. Let's use 1 byte for the number N for representation of number of amplitude values, taking to account $4 \le N \le 8$. For each of first two amplitude values, let's use 2 bytes (16 bits).

Compresed WAVE file header (RIFF WAVE header) is a 44-byte structure which is common and standartized, thus we do not demonstrate this one in a current work.

Archive data area, according to requirements, is a set of chunks of transformed sound amplitude values. Each of these chunks is represented using bit structure.

For practical implementation we have selected C programming language. C is a powerful standartized high-level programming language which provides high performance of compiled code compared to low-level programming languages. Compilers and environments for C programming languages are implemented for huge variety of hardware platforms and operation systems.

The following test set of sound files (RIFF WAVE, 44kHz, 16 bit/sample, mono) has been selected for testing the software:

– gamlet.wav (audio version of “Hamlet” stage play by Les Podervyanskyi) – speech audio recording, 44596268 bytes;

– moon_sonate.wav ("Moon sonate" by Beethoven, part 1) – classic music audio recording, 13291820 bytes;

– voodoo_child.wav (Jimi Hendrix, "Voodoo child") – музичний аудиозапис у стилі rock-n-blues, 27857708 bytes;

– bluesfriend.wav ("Moy drug luchshe vsekh igrayet blues" by Mashina Vremeni) – rock music audio recording, 22040108 bytes;

– oskolok_lda.wav ("Oskolok L'da" by ARIA) – heavy metal music audio recording, 28726316 bytes.

Results of compression for the test files mentioned above using developed software versus WinZip and WinRAR archivers are given in tables 1–3 respectively.

As one can notice from given compression results, developed compression software allows to reach significantly higher compression rates than WinZip archiver. Sound files compression rates provided by developed software is comparable to the ones reached by WinRAR – one of the most effective modern generic archivers.

ISSN 2520-2820 (online), ISSN 2524-0978 (print)

Table 1

Compression results using ddpack software

Sound file	Size before compression, bytes	Size after compression, bytes	Compression rate	Compression time, seconds
gamlet.wav	44 596 268	23 627 636	1,89	17,95
moon_sonate.wav	13 291 820	4 239 409	3,14	4,35
voodoo_child.wav	27 857 708	17 703 263	1,57	11,57
bluesfriend.wav	22 040 108	15 993 601	1,38	9,80
oskolok_lda.wav	28 726 316	20 385 314	1,41	12,78

Table 2

Compression results using WinZip

Sound file	Size before compression, bytes	Size after compression, bytes	Compression rate	Compression time, seconds
gamlet.wav	44 596 268	34 608 521	1,29	10,02
moon_sonate.wav	13 291 820	10 168 040	1,31	3,35
voodoo_child.wav	27 857 708	24 177 235	1,15	5,83
bluesfriend.wav	22 040 108	19 760 072	1,12	5,16
oskolok_lda.wav	28 726 316	26 326 441	1,09	6,66

Table 3

Compression results using WinRAR

Sound file	Size before compression, bytes	Size after compression, bytes	Compression rate	Compression time, seconds
gamlet.wav	44 596 268	24 395 329	1,83	35,69
moon_sonate.wav	13 291 820	4 155 373	3,20	7,52
voodoo_child.wav	27 857 708	17 111 539	1,63	13,82
bluesfriend.wav	22 040 108	15 033 534	1,47	13,06
oskolok_lda.wav	28 726 316	19 112 927	1,50	16,79

Average compression time is almost twice longer as for WinZip and approximately 1.5–2 times faster as for WinRAR. Thus, developed software compresses in general 2 times longer than WinZip but 1.5–2 times faster than WinRAR.

Restoring (decompression) of sound data is performed approximately 2–2.5 times faster than related compression.

This allows to state that developed software provides significant advantage in compression time, comparing to WinRAR, providing compression rates similar to WinRAR for sound data and high decompression speed.

References

1. Лукьянчиков И. С. Анализ современных методов сжатия информации и их применения к сжатию звуковых данных // «Сучасні інформаційні технології в освіті та промисловості»: матеріали II міжнародної конференції. — Миколаїв : УДМТУ, 2003. — С. 75—76.
2. Лукьянчіков І. С. Спосіб безвтратного стиснення аудіоданих на основі перетворення кінцевими різницями другого порядку // Збірнік наукових праць НУК, 2006. — С. 158—161.

Сергей УСТЕНКО, Сергей ЛУКЬЯНЧИКОВ
г. Николаев
Иван ЛУКЬЯНЧИКОВ
г. Львов

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДЛЯ АРХІВАЦИИ С ПОМОЩЬЮ ДВОЙНОГО-ДИФЕРЕНЦИАЛЬНОГО ПРЕОБРАЗОВАНИЯ ЗВУКОВОГО СИГНАЛА

Работа посвящена разработке программного обеспечения для сжатия аудио данных без потерь с помощью двойного-дифференциального преобразования звукового сигнала.

Ключевые слова: программа, архивация, разархивирование, программный модуль, алгоритм, сжатие без потерь.

Сергій УСТЕНКО, Сергій ЛУКЬЯНЧІКОВ
м. Миколаїв
Іван ЛУКЬЯНЧІКОВ
м. Львів

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ АРХІВАЦІЇ ЗА ДОПОМОГОЮ ДВІЧІ-ДИФЕРЕНЦІАЛЬНОГО ПЕРЕТВОРЕННЯ ЗВУКОВОГО СИГНАЛУ

Робота присвячена розробці програмного забезпечення для безвтратного стиснення аудіо даних за допомогою двічі-диференціального перетворення звукового сигналу.

Ключові слова: програма, архівація, розархівація, програмний модуль, алгоритм, стиснення без втрат.

Стаття надійшла до редколегії 12.04.2017