(12) UK Patent (19) GB (11) 2527588 (13) B

(45) Date of B Publication 18.05.2016

(54) Title of the Invention: Encoder and decoder

(51) INT CL: ***H03M 7/30*** (2006.01) ***G06F 3/06*** (2006.01) ***G06F 17/30*** (2006.01) ***H04L 29/06*** (2006.01) ***H04N 19/00*** (2014.01)

(21) Application No: **1411451.6**

(22) Date of Filing: **27.06.2014**

(43) Date of A Publication **30.12.2015**

(56) Documents Cited:

EP 2359233 A1	**EP 2256934 A1**
US 7643505 B1	**US 5434568 A**
US 20130315307 A1	**US 20100115137 A1**

(58) Field of Search:
As for published application 2527588 A viz:
INT CL **G06F, G06T, H03M, H04L, H04N**
Other: **WPI, EPODOC, INSPEC, TXTE**
updated as appropriate

(72) Inventor(s):
Tuomas Mikael Kärkkäinen
Ossi Kalevo

(73) Proprietor(s):
Gurulogic Microsystems Oy
Linnankatu 34, Turku 20100, Finland

(74) Agent and/or Address for Service:
Basck Ltd
16 Saxon Road, CAMBRIDGE, Cambridgeshire, CB5 8HS, United Kingdom

GB 2527588 B

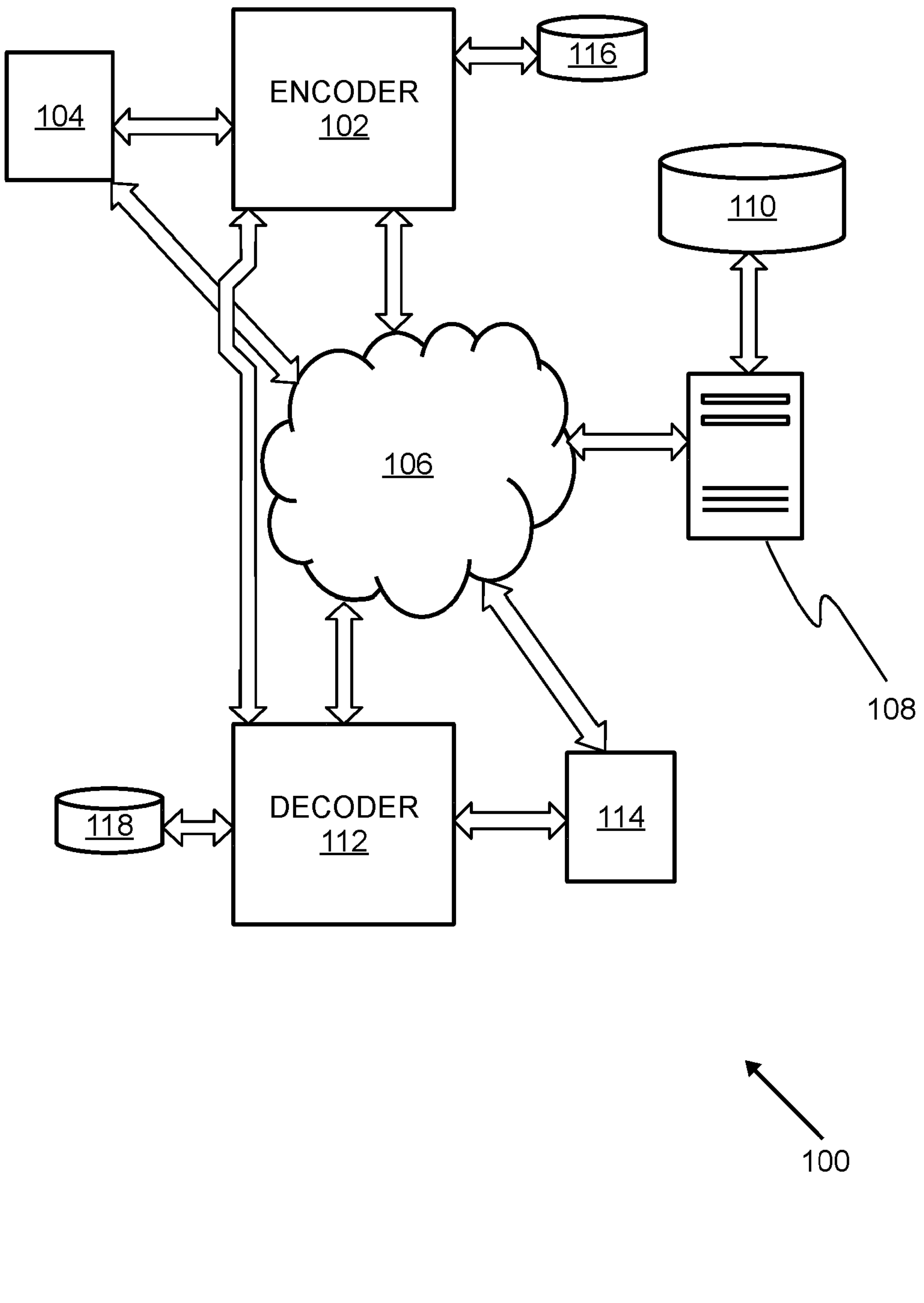

Fig. 1

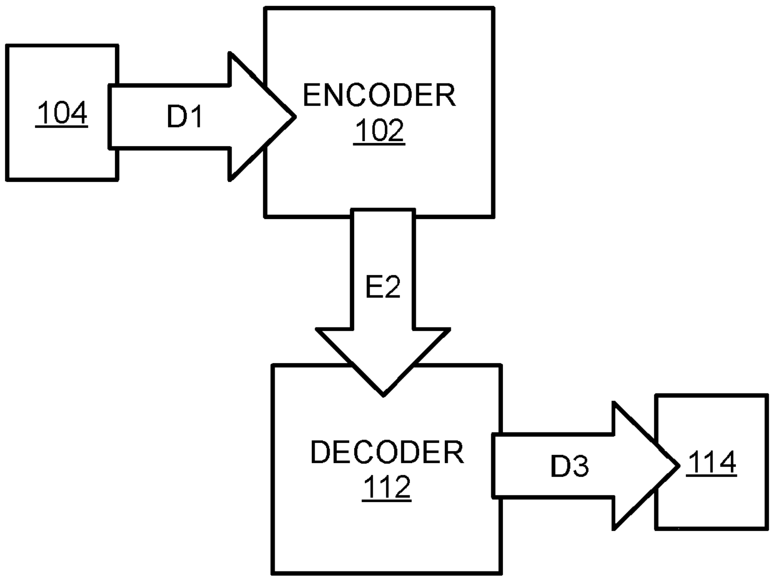

Fig. 2

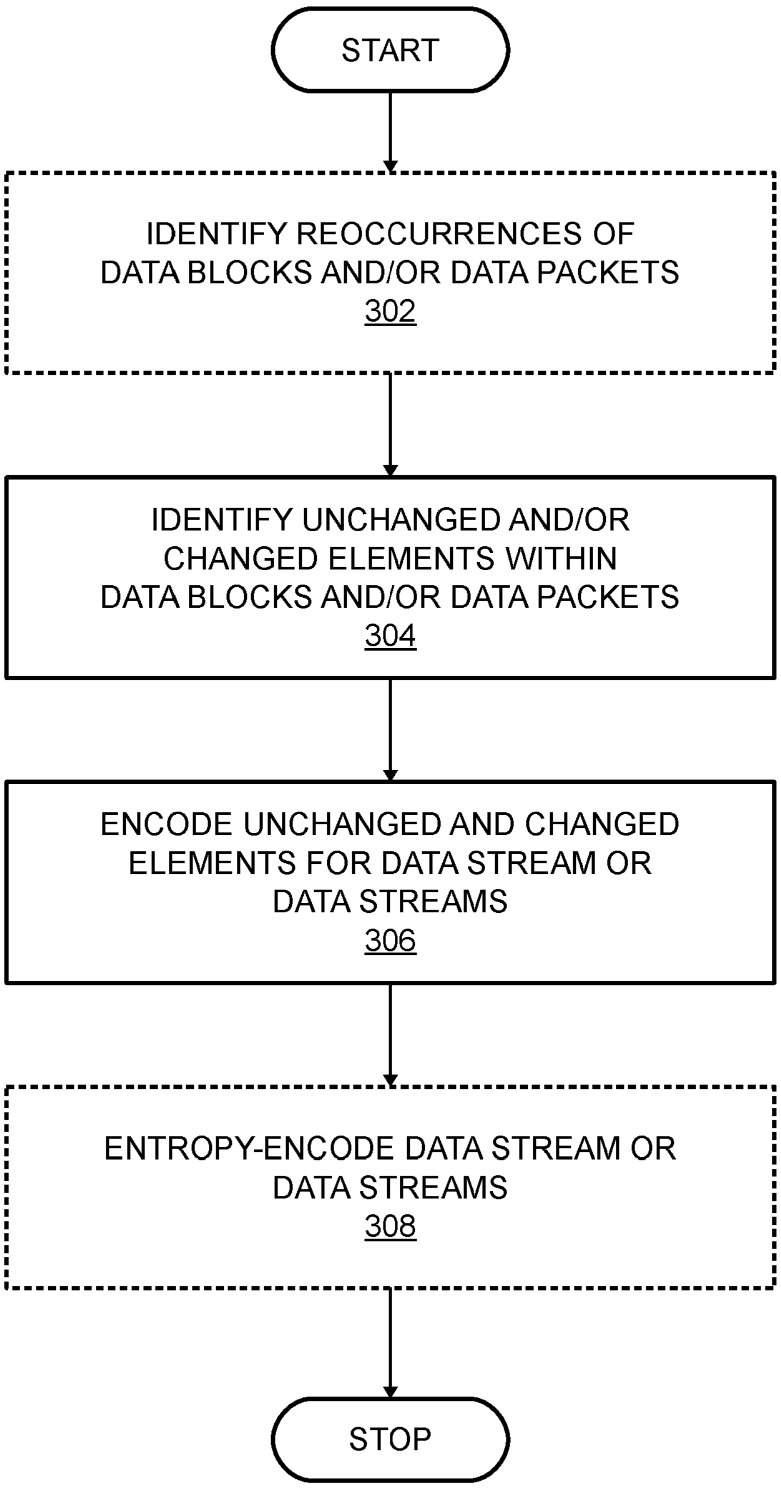
START
IDENTIFY REOCCURRENCES OF DATA BLOCKS AND/OR DATA PACKETS
302
IDENTIFY UNCHANGED AND/OR CHANGED ELEMENTS WITHIN DATA BLOCKS AND/OR DATA PACKETS
304
ENCODE UNCHANGED AND CHANGED ELEMENTS FOR DATA STREAM OR DATA STREAMS
306
ENTROPY-ENCODE DATA STREAM OR DATA STREAMS
308
STOP

Fig. 3

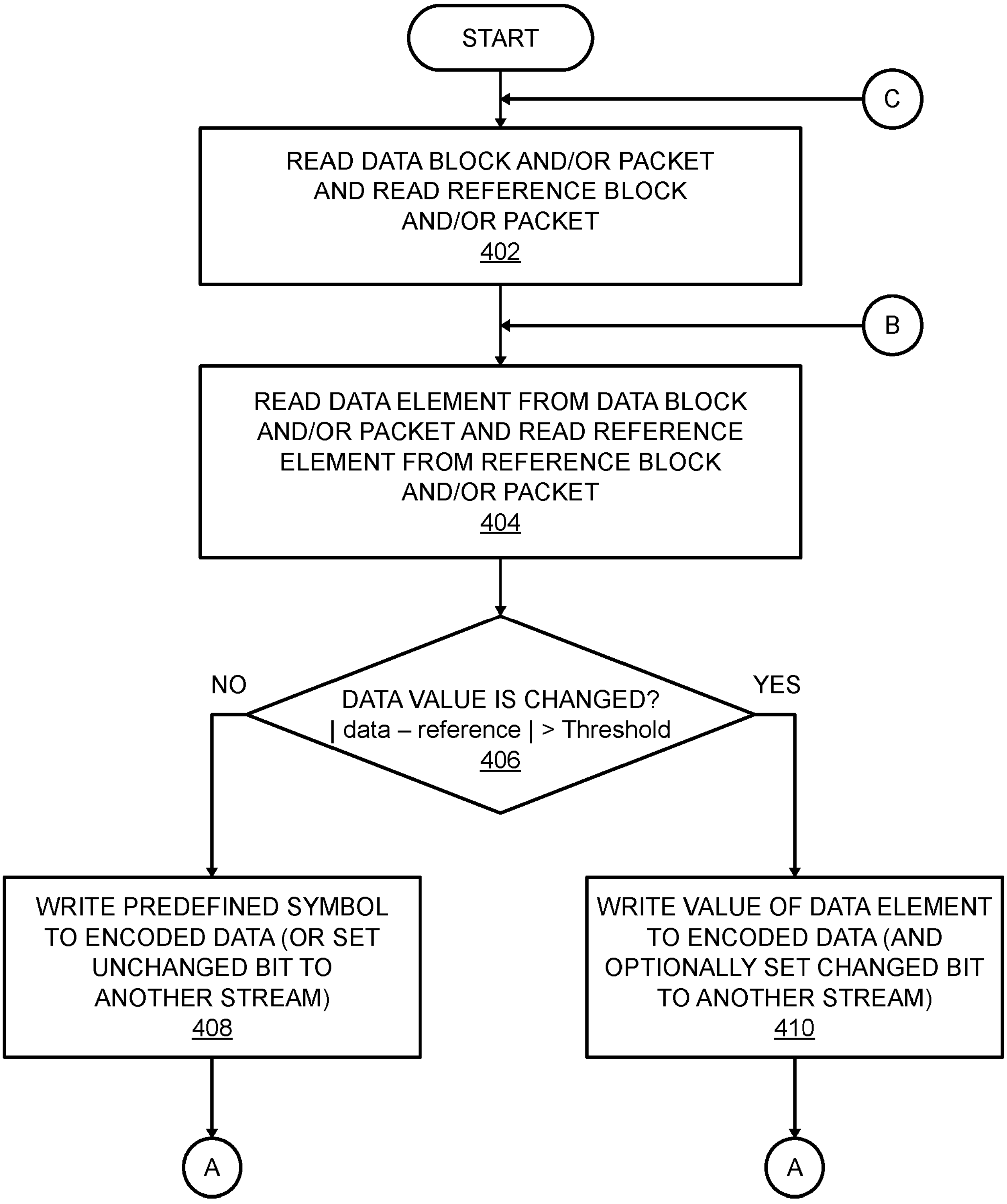
START
C
READ DATA BLOCK AND/OR PACKET AND READ REFERENCE BLOCK AND/OR PACKET
402
B
READ DATA ELEMENT FROM DATA BLOCK AND/OR PACKET AND READ REFERENCE ELEMENT FROM REFERENCE BLOCK AND/OR PACKET
404
NO
DATA VALUE IS CHANGED?
| data – reference | > Threshold
406
YES
WRITE PREDEFINED SYMBOL TO ENCODED DATA (OR SET UNCHANGED BIT TO ANOTHER STREAM)
408
WRITE VALUE OF DATA ELEMENT TO ENCODED DATA (AND OPTIONALLY SET CHANGED BIT TO ANOTHER STREAM)
410
A
A

Fig. 4A

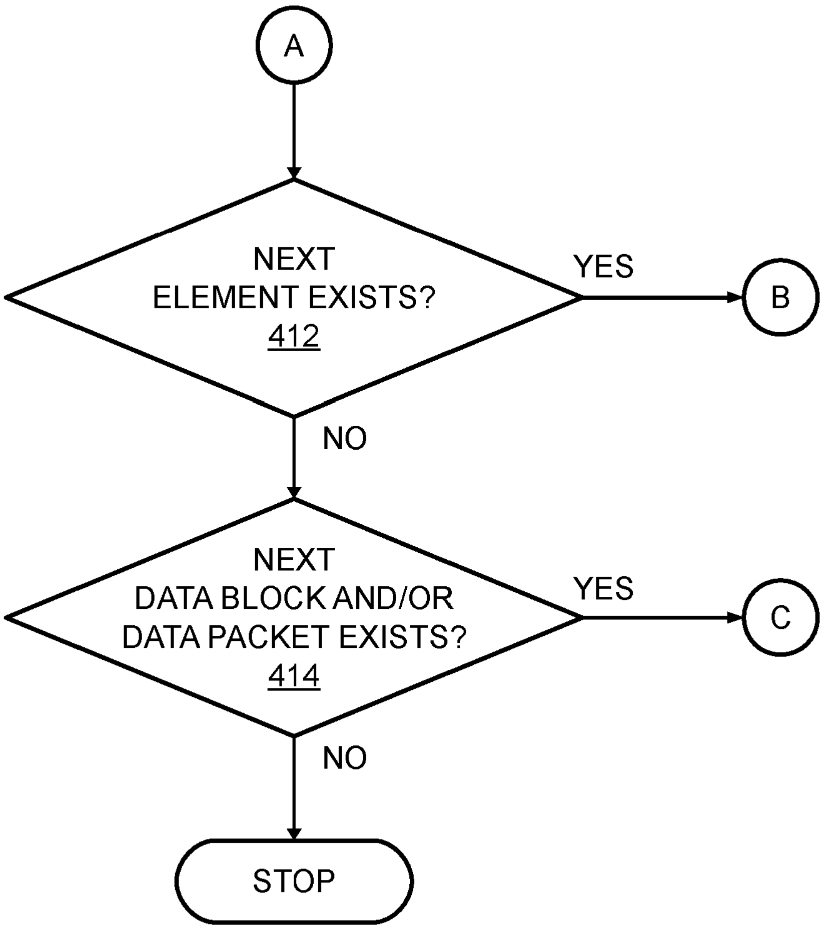
A
NEXT ELEMENT EXISTS? 412
YES
B
NO
NEXT DATA BLOCK AND/OR DATA PACKET EXISTS? 414
YES
C
NO
STOP

Fig. 4B

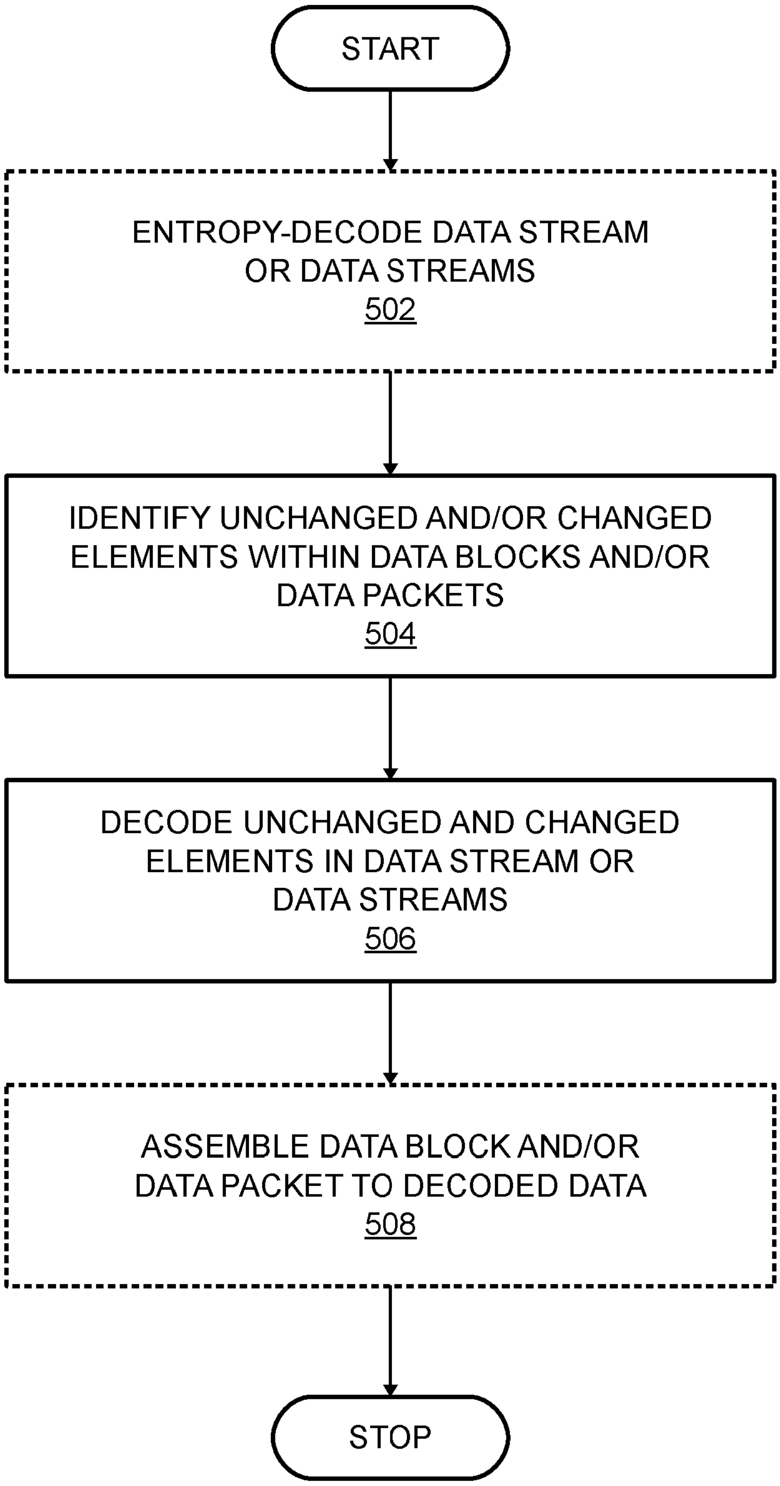
START
ENTROPY-DECODE DATA STREAM OR DATA STREAMS 502
IDENTIFY UNCHANGED AND/OR CHANGED ELEMENTS WITHIN DATA BLOCKS AND/OR DATA PACKETS 504
DECODE UNCHANGED AND CHANGED ELEMENTS IN DATA STREAM OR DATA STREAMS 506
ASSEMBLE DATA BLOCK AND/OR DATA PACKET TO DECODED DATA 508
STOP

Fig. 5

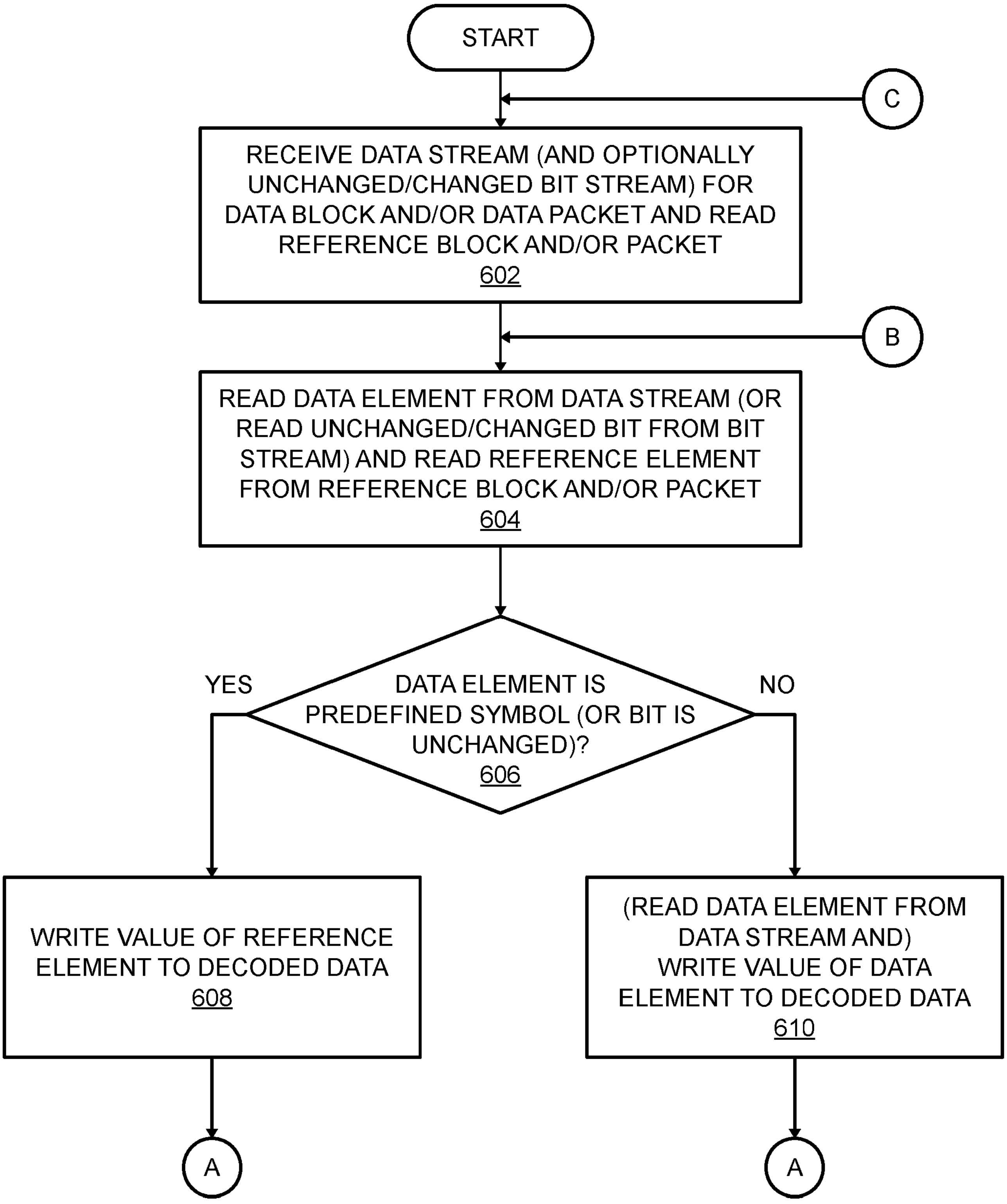
START
C
RECEIVE DATA STREAM (AND OPTIONALLY UNCHANGED/CHANGED BIT STREAM) FOR DATA BLOCK AND/OR DATA PACKET AND READ REFERENCE BLOCK AND/OR PACKET
602
B
READ DATA ELEMENT FROM DATA STREAM (OR READ UNCHANGED/CHANGED BIT FROM BIT STREAM) AND READ REFERENCE ELEMENT FROM REFERENCE BLOCK AND/OR PACKET
604
DATA ELEMENT IS PREDEFINED SYMBOL (OR BIT IS UNCHANGED)?
606
YES
NO
WRITE VALUE OF REFERENCE ELEMENT TO DECODED DATA
608
(READ DATA ELEMENT FROM DATA STREAM AND) WRITE VALUE OF DATA ELEMENT TO DECODED DATA
610
A
A

Fig. 6A

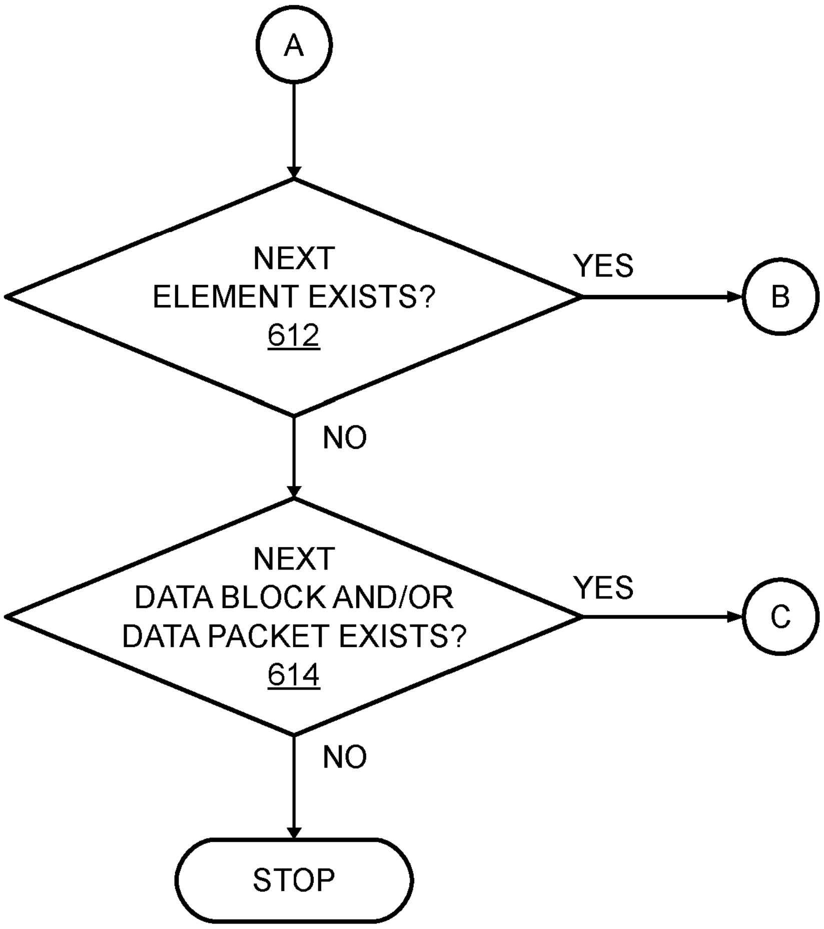
A
NEXT ELEMENT EXISTS? 612
YES
B
NO
NEXT DATA BLOCK AND/OR DATA PACKET EXISTS? 614
YES
C
NO
STOP

Fig. 6B

Application No. GB1411451.6 RTM Date :19 December 2014

The following terms are registered trade marks and should be read as such wherever they occur in this document:

WiMAX

ENCODER AND DECODER

TECHNICAL FIELD

The present disclosure relates generally to data compression, and more specifically, to encoders for encoding input data (D1) to generate corresponding encoded data (E2), and decoders for decoding the encoded data (E2) to generate corresponding decoded data (D3). Moreover, the present disclosure relates to methods of encoding input data (D1) to generate corresponding encoded data (E2), and methods of decoding the encoded data (E2) to generate corresponding decoded data (D3). Furthermore, the present disclosure also relates to computer program products comprising non-transitory (namely non-transient) computer-readable storage media having computer-readable instructions stored thereon, the computer-readable instructions being executable by a computerized device comprising processing hardware to execute the aforesaid methods.

BACKGROUND

Today, it has become a customary practice to compress data to reduce usage of resources, for example, during data storage and data communication. During data communication, a sequence of data blocks or packets is communicated from one device to another device. Data blocks or packets communicated later in the sequence are often changed in comparison to data blocks or packets that have been communicated earlier. However, changes in individual elements inside these changed data blocks or packets are considerably smaller than the original content of the data blocks or packets. In other words, most of the elements inside the changed blocks are unchanged in comparison to the earlier data blocks or packets. When such a sequence of data blocks or packets is compressed using conventional encoders, often only a small compression ratio is achieved.

One conventional encoder-decoder (hereinafter referred to as "*codec*") has been described in US patent document 20120219065 A1, titled "*Processing of Image*". The conventional codec processes entire data blocks of an image or data sequence, and compares the entire data blocks with previous data blocks. The conventional codec encodes an unchanged data block to a predefined colour value or data value, and also encodes a changed data block as it is. This means that all the original data values in a changed data block are coded, and so a compression efficiency performance provided by the conventional encoder-decoder is not as great as potentially achievable.

Moreover, another conventional codec (http://en.wikipedia.org/wiki/Delta_encoding) employs delta coding for processing data. In delta coding, a difference (namely, a delta value) between a current data element and a previous data element is written or transmitted. However, such delta values often generate new data values that are potentially not present in the data or otherwise enlarges the dynamic of the data values, and therefore, an increase entropy of the data thereby potentially arises.

Therefore, there exists a need for such a codec for compressing data that is more efficient in comparison to the conventional codecs.

In a published US patent application US 2013/0315307 A1 ("*Processing and Reproduction of Frames*"; Inventors: Tuomas Karkkainen, Ossi Kalevo; Applicant: Gurulogic Microsystems Oy), there is described a processor that is configured to receive input data, divide an individual frame into blocks, compare the blocks with corresponding blocks of a first prediction frame and identify changed blocks. The processor is also configured to include in generated intermediate data the identified changed blocks, and generate a change indicator indicating the positions in the individual frame of the identified changed blocks and identified unchanged blocks.

In a granted US patent US 5434568 A ("*Data Compression by Removing Repetition and Unnecessary Information*"; Inventor: Edward W. Moll; Applicant: Edward W. Moll), there is described a system in which repetitive data and non-repetitive data, including periods of no information, is encoded prior to transmission or storage in digital form. Repetition, partial repetition, and near repetition is encoded in a form that indicates the occurrence of repetition, its characteristics and its duration. The existence and size of repeated patterns in the data is dynamically determined. When repetition is detected, non-repetitive data is inserted into the data stream and repetitive data is removed from the data stream. To this non-repetitive data in the data stream are added a coded repeated pattern sample, an identification preamble signal, an instruction signal for decoding purposes, a period count signal, a mask signal, and a repeat count signal. All necessary data elements are combined and assembled to produce compressed data. A receiver utilizes these coded and uncoded data elements to regenerate complete original data.

In a granted US patent US 7643505 B1 ("*Method and System for Real Time Compression and Decompression*"; Inventor: Ian G. Colloff; Applicant: Qlogic Corporation), there are described a method and a system for compressing a data packet. The method includes receiving a data packet, comparing the data packet with content stored in a history module, generating a plurality of masks based on the comparisons, comparing the plurality of masks, selecting one of the plurality of masks based upon the mask comparisons, and generating a compression record, wherein the compression record includes size of a data packet, an address field, a mask field and data, and a data packet header includes a control bit indicating whether or not the data packet is compressed.

In a published EP patent application EP 2359233 A1 ("*Delta Compression after Identity Deduplication*"; Inventors: Mark Huang, Edward K Lee, Kai Li, Philip Shilane, Grant Wallace, Ming Benjamin Zhu; Applicant: Data Domain, Inc.), there are described a method and a system for processing data. The system includes a deduplicating system for determining that a first data segment is identical to a first previous data segment. The system also includes a delta compression system for determining that a second data segment, which is not identical to a second previous data segment, is similar to a third previous data segment.

In a published US patent application US 2010/0115137 A1 ("*Data Compression Method and Data Communication System Utilizing the Same*"; Inventors: Byeong Deok Kim, Sung Jo OH; Applicant: Samsung Electronics Co., Ltd.), there are described a data compression method and a data communication system utilizing the aforesaid method. The system includes a sender apparatus that compares input data with a previously sent data in storage, and produces, when a data item repeated in the input data and the previously sent data is found, delta data by excluding the repeated data item from the input data. The sender apparatus represents the delta data in a transport format, and transmits the delta data. The system also includes a receiver apparatus that receives data in the transport format, and adds, when delta data is present in the received data, a repeated data item to the delta data, and recovers the original data.

In a published EP patent application EP 2256934 A1 ("*Method and Apparatus for Content-aware and Adaptive Deduplication*"; Inventors: David G. Therrien, David Andrew Thompson; Applicant: Exagrid Systems, Inc.), there are described a method and a system for transmission of data across a network. In the method, a received data stream is analysed to determine a starting location and an ending location of each zone within the received data stream, and a zone stamp identifying that zone is generated. The zone stamp includes a sequence of contiguous characters representing at least a portion of data in the zone, wherein the order of characters in the zone stamp corresponds to the order of data in the zone. Moreover, in the method, a zone stamp of a given zone is compared with another zone stamp of another zone in any received data stream to determine whether or not the given zone is substantially similar to another zone. Subsequently, the method includes delta-compressing zones within any received data stream that have been determined to have substantially similar zone stamps, thereby deduplicating zones having substantially similar zone stamps within any received data stream, and transmitting the deduplicated zones across the network from one storage location to another storage location.

SUMMARY

The present disclosure seeks to provide an improved encoder for encoding input data (D1) to generate corresponding encoded data (E2).

The present disclosure also seeks to provide an improved decoder for decoding encoded data (E2) to generate corresponding decoded data (D3).

Moreover, the present disclosure seeks to provide an improved method of encoding input data (D1) to generate corresponding encoded data (E2).

Moreover, the present disclosure also seeks to provide an improved method of decoding encoded data (E2) to generate corresponding decoded data (D3).

In a first aspect, embodiments of the present disclosure provide an encoder including processing hardware for encoding input data (D1) to generate corresponding encoded data (E2), wherein the processing hardware is operable to process the input data (D1) as data blocks and/or data packets, characterized in that

the processing hardware is operable to:

(i) identify substantial reoccurrences of data blocks and/or data packets within at least a portion of the input data (D1), wherein the data blocks and/or data packets include a corresponding plurality of elements, wherein the elements include a plurality of bits;
(ii) identify where elements are unchanged within the substantially reoccurring data blocks and/or data packets, and/or where elements are changed within the substantially reoccurring data blocks and/or data packets;
(iii) encode unchanged elements in the encoded data (E2) by employing at least one corresponding symbol or at least one corresponding bit indicating an absence of change in the unchanged elements relative to corresponding elements in a reference data block and/or data packet; and
(iv) encode changed elements in the encoded data (E2).

Optionally, the input data (D1) is in a form of at least one of: text data, image data, video data, audio data, binary data, sensor data, measurement data, graphical data, multi-dimensional data and/or one-dimensional data, but not limited thereto.

Optionally, a reference data block and/or a reference data packet is a previous data block or data packet; a data block or a packet in a similar location in a previous frame, a view or a channel; a data block or a data packet described by a motion vector (namely motion compensation); a data block or a data packet described by symbol (namely deduplication); a data block or a data packet encoded with some coding method (for example using coding methods such as DC encoding, multilevel encoding, slide encoding, line encoding, discrete cosine transform (DCT) encoding, database encoding, vector quantization encoding, palette encoding, interpolation encoding, extrapolation encoding). Optionally, the at least one corresponding symbol is represented by a predetermined data value. Optionally, the predetermined data value is implemented as a zero data value. Optionally, the unchanged value, for example when the value of the bit is 0, and the changed value, for example when the value of the bit is 1, are described by bits in a separate data stream, and only the changed input data values are encoded into the data stream.

Moreover, optionally, the processing hardware is operable to encode at least a portion of the changed elements in a quantized manner in the encoded data (E2). Such a quantized manner of operation is capable of providing an enhanced degree of data compression.

When all the elements in a given data block or data packet are unchanged as compared to a reference data block or data packet, then, optionally, the data block or packet is set as unchanged and then there is no need to deliver any other information for that data block or data packet. Optionally, the data blocks and data packets with all values changed are also separated from partially changed data blocks or data packets.

The method described in this disclosure is beneficially used for encoding the partially changed data blocks or packets. Partially changed data blocks or packets contain both changed and unchanged data values. Optionally, the method is also operable to encode changed data blocks or packets. Optionally, the method is also operable to encode unchanged data blocks or packets.

Moreover, optionally, the processing hardware is operable to apply a compression algorithm, for example Range coding, SRLE (split run length encoding), Delta coding, ODelta coding, EM (entropy modifying encoding), Arithmetic coding, Huffman coding, but not limited thereto, to compress the encoded data (E2) to generate compressed data (C4) which is included into the encoded output data (E2). Such additional compression provided by the compression algorithm is capable of further compressing the encoded data (E2) relative to the input data (D1).

In a second aspect, embodiments of the present disclosure provide a decoder including processing hardware for decoding encoded data (E2) to generate corresponding decoded data (D3), wherein the processing hardware is operable to process the encoded data (E2) as data blocks and/or data packets, characterized in that

the processing hardware is operable to:

(i) decode the encoded data (E2) to generate data for changed elements, the changed elements being elements that are changed within substantial reoccurrences of data blocks and/or data packets within the encoded data (E2);

(ii) decode the encoded data (E2) to generate data for unchanged elements, the unchanged elements being elements that are unchanged within the substantial reoccurrences of data blocks and/or data packets within the encoded data (E2), wherein the unchanged elements are represented by at least one corresponding symbol or at least one corresponding bit indicating an absence of change in the unchanged elements relative to corresponding elements in a reference data block and/or data packet; and

(iii) assemble the data generated for the changed and unchanged elements in (i) and (ii) into data blocks and/or data packets to generate the decoded data (D3), wherein the data blocks and/or data packets include a corresponding plurality of elements, wherein the elements include a plurality of bits.

Optionally, the processing hardware is operable to decode at least a portion of the changed elements in a quantized manner in the decoded data (D3).

Optionally, the processing hardware is operable to apply a decompression algorithm to decompress compressed data (C4) to generate the encoded data (E2) for decoding the encoded data (E2) to generate the data for the changed and unchanged elements.

Optionally, the decoded data (D3) is in a form of at least one of: text data, image data, video data, audio data, binary data, sensor data, measurement data, graphical data, multi-dimensional data and/or one-dimensional data, but not limited thereto.

In a third aspect, embodiments of the present disclosure provide a codec including the aforementioned encoder and the aforementioned decoder. Optionally, the codec is in a form of at least one of: a video codec, an audio codec, an image codec and/or a data codec, but not limited thereto.

Moreover, optionally, the encoder and the decoder are operable to implement chunked transfer encoding for Hypertext Transfer Protocol (HTTP) and/or Real-Time

Messaging Protocol (RTMP). Optionally, the HTTP and/or RTMP employ fixed-size data blocks and/or data packets inside requests and responses to the requests.

In a fourth aspect, embodiments of the present disclosure provide a method of encoding input data (D1) to generate corresponding encoded data (E2), wherein the method includes processing the input data (D1) as data blocks and/or data packets, characterized in that

the method includes:

(i) identifying substantial reoccurrences of data blocks and/or data packets within at least a portion of the input data (D1), wherein the data blocks and/or data packets include a corresponding plurality of elements, wherein the elements include a plurality of bits;
(ii) identifying where elements are unchanged within the substantially reoccurring data blocks and/or data packets, and/or where elements are changed within the substantially reoccurring data blocks and/or data packets;
(iii) encoding unchanged elements in the encoded data (E2) by employing at least one corresponding symbol or at least one corresponding bit indicating an absence of change in the unchanged elements relative to corresponding elements in a reference data block and/or data packet; and
(iv) encoding changed elements in the encoded data (E2).

In a fifth aspect, embodiments of the present disclosure provide a computer program product comprising a non-transitory (namely non-transient) computer-readable storage medium having computer-readable instructions stored thereon, the computer-readable instructions being executable by a computerized device comprising processing hardware to execute the aforementioned method.

In a sixth aspect, embodiments of the present disclosure provide a method of decoding encoded data (E2) to generate corresponding decoded data (D3), wherein the method includes processing the encoded data (E2) as data blocks and/or data packets, characterized in that

the method includes:

(i) decoding the encoded data (E2) to generate data for changed elements, the changed elements being elements that are changed within substantial reoccurrences of data blocks and/or data packets within the encoded data (E2);

(ii) decoding the encoded data (E2) to generate data for unchanged elements, the unchanged elements being elements that are unchanged within the substantial reoccurrences of data blocks and/or data packets within the encoded data (E2), wherein the unchanged elements are represented by at least one corresponding symbol or at least one corresponding bit indicating an absence of change in the unchanged elements relative to corresponding elements in a reference data block and/or data packet; and

(iii) assembling the data generated for the changed and unchanged elements in steps (i) and (ii) into data blocks and/or data packets to generate the decoded data (D3), wherein the data blocks and/or data packets include a corresponding plurality of elements, wherein the elements include a plurality of bits.

In a seventh aspect, embodiments of the present disclosure provide a computer program product comprising a non-transitory (namely non-transient) computer-readable storage medium having computer-readable instructions stored thereon, the computer-readable instructions being executable by a computerized device comprising processing hardware to execute the aforementioned method.

Embodiments of the present disclosure substantially eliminate, or at least partially address, the aforementioned problems in the prior art, and enable lossless or near-lossless data compression of one-dimensional image data or multi-dimensional image data, video data, audio data and any other type of data with a high compression ratio.

Additional aspects, advantages, features and objects of the present disclosure are made apparent in the drawings and the detailed description of the illustrative embodiments construed in conjunction with the appended claims that follow.

It will be appreciated that features of the present disclosure are susceptible to being combined in various combinations without departing from the scope of the present disclosure as defined by the appended claims.

DESCRIPTION OF THE DRAWINGS

The summary above, as well as the following detailed description of illustrative embodiments, is better understood when read in conjunction with the appended drawings. For the purpose of illustrating the present disclosure, exemplary constructions of the disclosure are shown in the drawings. However, the present disclosure is not limited to specific methods and apparatus disclosed herein. Moreover, those in the art will understand that the drawings are not to scale. Wherever possible, like elements have been indicated by identical numbers.

Embodiments of the present disclosure will now be described, by way of example only, with reference to the following diagrams wherein:

Fig. 1 is a schematic illustration of an example network environment that is suitable for practicing embodiments of the present disclosure;

Fig. 2 is an illustration of an example data flow, in accordance with an embodiment of the present disclosure;

Fig. 3 is an illustration of steps of a method of encoding input data (D1) to generate corresponding encoded data (E2), in accordance with an embodiment of the present disclosure;

Figs. 4A and 4B collectively are an illustration of steps of an encoding processing, in accordance with an embodiment of the present disclosure;

Fig. 5 is an illustration of steps of a method of decoding the encoded data (E2) to generate corresponding decoded data (D3), in accordance with an embodiment of the present disclosure; and

Figs. 6A and 6B collectively are an illustration of steps of a decoding processing, in accordance with an embodiment of the present disclosure.

In the accompanying drawings, an underlined number is employed to represent an item over which the underlined number is positioned or an item to which the underlined number is adjacent. A non-underlined number relates to an item identified by a line linking the non-underlined number to the item. When a number is non-underlined and accompanied by an associated arrow, the non-underlined number is used to identify a general item at which the arrow is pointing.

DETAILED DESCRIPTION OF EMBODIMENTS

The following detailed description illustrates embodiments of the present disclosure and ways in which they can be implemented. Although the best mode of carrying out the present disclosure has been disclosed, those skilled in the art would recognize that other embodiments for carrying out or practicing the present disclosure are also possible.

Embodiments of the present disclosure provide an encoder including processing hardware for encoding input data (D1) to generate corresponding encoded data (E2). The processing hardware is operable to process the input data (D1) as data blocks and/or data packets. Optionally, the input data (D1) is in a form of at least one of: text data, image data, video data, audio data, binary data, sensor data, measurement data, graphical data, multi-dimensional data and/or one-dimensional data, but not limited thereto.

The processing hardware is operable to identify substantial reoccurrences of data blocks and/or data packets within the input data (D1). The processing hardware is then operable to identify where elements are unchanged within the substantially reoccurring data blocks and/or data packets, and/or where elements are changed within the substantially reoccurring data blocks and/or data packets.

Subsequently, the processing hardware is operable to encode unchanged elements in the encoded data (E2) by employing at least one corresponding symbol, or at least one corresponding bit, for example a single bit, indicating an absence of change in the unchanged elements relative to corresponding elements in a reference data block and/or data packet. Optionally, the at least one corresponding symbol is represented by a predetermined data value. Optionally, the predetermined data value is implemented as a zero data value. Optionally, the unchanged value, for example when the bit value is 0, and the changed value, for example when the bit value is 1, are described by bits in a separate bit stream and only the changed input data values are encoded to the data stream. By “channel” is meant at least one of: a channel-defined portion of the encoded data (E2), a channel-defined separate stream of data, a channel defined separate data file.

Moreover, the processing hardware is operable to encode changed elements in the encoded data (E2). Optionally, the processing hardware is operable to encode at least a portion of the changed elements in a quantized manner in the encoded data (E2).

Moreover, optionally, the processing hardware is operable to apply a compression algorithm, for example Range coding, SRLE, Delta coding, ODelta coding, EM (entropy modifying) coding, Arithmetic coding, Huffman coding, but not limited thereto, to compress the encoded data (E2) to generate compressed data (C4) which is included into the encoded output data (E2).

Furthermore, embodiments of the present disclosure also provide a decoder including processing hardware for decoding encoded data (E2) to generate corresponding decoded data (D3). The processing hardware is operable to process the encoded data (E2) as data blocks and/or data packets.

The processing hardware is operable to decode the encoded data (E2) to generate data for elements that are changed (hereinafter referred to as "*changed elements*") within substantial reoccurrences of data blocks and/or data packets within the encoded data (E2). Optionally, the processing hardware is operable to decode at least a portion of the changed elements in a quantized manner in the decoded data (D3). By "*quantized manner*" is meant that the changed elements are decoded into data, wherein the decoded data is selected from a finite number of possible values for the data, namely the decoded data changes in a non-continuous manner.

The processing hardware is operable to decode the encoded data (E2) to generate data for elements that are unchanged (hereinafter referred to as "*unchanged elements*") within the substantial reoccurrences of data blocks and/or data packets within the encoded data (E2). In the encoded data (E2), the unchanged elements are represented by at least one corresponding symbol, or at least one corresponding bit, for example a single bit, indicating an absence of change in the unchanged elements relative to corresponding elements in a reference data block and/or data packet.

Optionally, the processing hardware is operable to apply a decompression algorithm to decompress compressed data (C4) present in the encoded data (E2) for use in decoding the encoded data (E2) to generate the data for the changed and unchanged elements.

Moreover, the processing hardware is operable to assemble the data generated for the changed and unchanged elements into data blocks and/or data packets to generate the decoded data (D3).

Optionally, the decoded data (D3) is in a form of at least one of: text data, image data, video data, audio data, binary data, sensor data, measurement data, graphical data, multi-dimensional data and/or one-dimensional data, but not limited thereto.

Furthermore, embodiments of the present disclosure also provide a codec including the aforementioned encoder and the aforementioned decoder. Optionally, the codec is in a form of at least one of: a video codec, an audio codec, an image codec and/or a data codec, but not limited thereto.

Moreover, optionally, the encoder and the decoder are operable to implement chunked transfer encoding for Hypertext Transfer Protocol (HTTP) and/or Real-Time Messaging Protocol (RTMP). Optionally, the HTTP and/or RTMP employ fixed-size data blocks and/or data packets inside requests and responses to the requests.

Referring now to the drawings, particularly by their reference numbers, Fig. 1 is a schematic illustration of an example network environment **100** that is suitable for practicing embodiments of the present disclosure. The network environment **100** includes an encoder **102** and one or more electronic devices, depicted as an electronic device **104** in Fig. 1. The network environment **100** also includes a communication network **106**, and one or more data servers and/or data storages and one or more databases, depicted as a data server and/or data storage **108** and a database **110** in Fig. 1. Additionally, the network environment **100** includes a decoder **112** and one or more computerized devices, depicted as a computerized device **114** in Fig. 1. Optionally, the network environment **100** includes one or more databases and/or one or more local data memories (**116, 118**) which are spatially local to devices of the network environment **100**.

The network environment **100** is optionally implemented in various ways, depending on various possible scenarios. In one example scenario, the network environment **100** is optionally implemented by way of a spatially collocated arrangement of the data server and/or data storage **108** and the database **110** coupled mutually in communication via a direct connection, for example, as shown in Fig. 1. In another example scenario, the network environment **100** is optionally implemented by way of a spatially distributed arrangement of the data server and/or data storage **108** and the database **110** coupled mutually in communication via a communication network, such as the communication network **106**. In yet another example scenario, the data server and/or data storage **108** and the database **110** are optionally implemented via cloud computing services. Optionally, the network environment **100** is implemented in a distributed peer-to-peer (P2P) manner.

The data server and/or data storage **108** is coupled in communication with the encoder **102** and the decoder **112**, via the communication network **106** or via a direct connection . Moreover, the encoder **102** is coupled in communication with the decoder **112**, via the communication network **106** or via a direct connection.

The communication network **106** is optionally a collection of individual networks, interconnected with each other and functioning as a single large network. Such individual networks are optionally wired, wireless, or a combination thereof. Examples of such individual networks include, but are not limited to, Local Area Networks (LANs), Wide Area Networks (WANs), Metropolitan Area Networks (MANs), Wireless LANs (WLANs), Wireless WANs (WWANs), Wireless MANs (WMANs), the Internet, second generation (2G) telecommunication networks, third generation (3G) telecommunication networks, fourth generation (4G) telecommunication networks, and Worldwide Interoperability for Microwave Access (WiMAX) networks.

The electronic device **104** provides the encoder **102**, either directly or through communication network **106**, which has input data (D1) as an input thereto. Optionally, the input data (D1) is in a form of at least one of: text data, image data, video data, audio data, binary data, sensor data, measurement data, graphical data, multi-dimensional data and/or one-dimensional data, but not limited thereto. The input data (D1) is optionally modified data, a part of an entire data sequence, or a combination of various types of data. Optionally, the input data (D1) is received as a stream or as a file.

The encoder **102** includes processing hardware that is operable to execute computer-readable instructions stored on a non-transitory (namely non-transient) computer-readable storage medium for encoding the input data (D1) to generate corresponding encoded data (E2). Alternatively, or additionally, the processing hardware is hardwired, for example implemented by way of an application-specific integrated circuit (ASIC), a state-variable machine or similar.

Optionally, the encoder **102** is implemented as a part of the electronic device **104**. In this case, the processing hardware of the encoder **102** is included in the electronic device **104**. In an example, the electronic device **104** is an image and/or video capturing device that generates in operation large quantities of image and/or video data, wherein a lossless compression is desired so as to preserve fine information in the image and/or video data, whilst rendering the quantities of the image and/or video data manageable for data storage purposes. Examples of such image and/or video capturing devices include, but are not limited to, surveillance cameras, video recorders, X-ray devices, Magnetic Resonance Imaging (MRI) scanners, and ultrasound scanners. The electric device **104** is beneficially implemented using Reduced Instruction Set Computing (RISC) processors that are capable of performing data manipulations associated with methods of the present disclosure in a highly efficient manner, while simultaneously being very energy efficient.

Alternatively, optionally, the encoder **102** is implemented independently, for example, using a computerized device that includes the processing hardware of the encoder **102**.

Upon receiving the input data (D1), the processing hardware of the encoder **102** is operable to process the input data (D1) as data blocks and/or data packets. Optionally, these data blocks and/or data packets have a fixed size. Alternatively, the data blocks and/or data packets have a variable size; optionally, the variable size is determined as a function of content and/or format of the input data (D1). Optionally, the content is automatically analyzed by employing a combination of spatial Fourier analysis and temporal Fourier analysis for computing one or more parameters for determining the variable size of the data blocks and/or data packets. Fourier transforms, for example Fast Fourier Transform (FFT), are known to a person skilled in the art, and are optionally implemented in a recursive manner, for example using one or more RISC processors.

The processing hardware of the encoder **102** is operable to identify substantial reoccurrences of data blocks and/or data packets within the input data (D1), namely, substantially similar data blocks and/or data packets within the input data (D1). The processing hardware of the encoder **102** is then operable to identify where elements are unchanged within the substantially reoccurring data blocks and/or data packets, and/or where elements are changed within the substantially reoccurring data blocks and/or data packets.

Subsequently, the processing hardware of the encoder **102** is operable to encode, in the encoded data (E2), unchanged elements within each data block and/or data packet by employing at least one corresponding symbol or one or more corresponding bits, for example a single bit, indicating an absence of change in the unchanged elements within that data block and/or data packet relative to corresponding elements within its corresponding reference data block and/or data packet. In this regard, a "*MemoryCompare*" functionality is optionally used to compare elements of a given data block and/or data packet with elements of a reference data block and/or data packet.

Optionally, the reference data block and/or data packet is fixed. In an example, the reference data block and/or data packet is similar for all of the data blocks and/or data packets of the input data (D1). In another example, the reference data block and/or data packet is similar for at least a subset of the data blocks and/or data packets of the input data (D1).

Alternatively, optionally, the reference data block and/or data packet changes, based on certain criteria, as will be elucidated below. In an example, when a number of changed elements within a given data block and/or data packet and another block and/or data packet is less than a predefined threshold number, then the given data block and/or data packet can be taken as a reference data block and/or data packet for any data block and/or data packet in the input data (D1). In another example, a previous data block and/or data packet is taken as a reference data block and/or data packet for a current data block and/or data packet in the input data (D1). Optionally, the previous data block and/or data packet is a data block and/or data packet after which the current data block consecutively followed. Alternatively, optionally, the previous data block and/or data packet is a data block and/or data packet that occurred in a previous frame, view or channel. Yet alternatively, optionally, the previous data block and/or data packet is a data block and/or data packet that was known or selected beforehand. Yet alternatively, optionally, the information of the selected previous data block and/or data packet is delivered with a symbol or reference number describing it or with, for example, a motion vector describing the selected data block and/or data packet.

Optionally, the at least one corresponding symbol is represented by a predetermined data value. Optionally, the predetermined data value is determined based on an analysis of data values of changed elements of the data blocks and/or data packets of the input data (D1); such analysis is performed, for example by performing a statistical analysis of element values within data blocks present in the input data (D1). Optionally, the predetermined data value is implemented as a zero data value. Alternatively, optionally, bits are used to describe changed and unchanged data values in a data block or packet and are sent to another stream, and in such a case, only the changed data values are optionally encoded to the data stream.

Moreover, the processing hardware of the encoder **102** is operable to encode the changed elements in the encoded data (E2). Optionally, for a lossless operation, the processing hardware of the encoder **102** is operable to encode the changed elements in their original form in the encoded data (E2). Alternatively, optionally, for a lossy operation, the processing hardware of the encoder **102** is operable to encode at least a portion of the changed elements in a quantized manner in the encoded data (E2).

Yet alternatively, optionally, for a near-lossless operation, the processing hardware of the encoder **102** is operable to encode at least some portion of the changed elements in a quantized manner in the encoded data (E2). For this purpose, the processing hardware of the encoder **102** is optionally operable to quantize only some portions of the changed elements, based on an analysis of content, type and/or composition of the input data (D1); such an analysis is operable, for example to determine an occurrence of regions of interest in the input data (D1) which are beneficially encoded with better or worse quality than other regions of interest. Optionally, if the input data is medical, military, binary, text or similar data, then it beneficially must be coded losslessly, but if the input data is audio, video, still images and so forth, then lossy coding is beneficially allowed as well. Of course, there potentially occur regions of interest in the input data (D1) which are beneficially encoded with better or worse quality than other regions of interest. Consequently, the encoder **102** is capable of adaptively varying a compression ratio of lossily encoded data by changing the quantization between the input data (D1) and the encoded data (E2).

Moreover, optionally, the processing hardware of the encoder **102** is operable to apply a compression algorithm to compress the encoded data (E2) to generate compressed data (C4). In this regard, the encoder **102** is beneficially useable with any contemporary entropy encoders; for example encoders utilizing Range coding, SRLE, Delta coding, ODelta coding, EM, Arithmetic coding, Huffman coding.

Furthermore, the encoder **102** is operable to communicate the encoded data (E2) to the data server and/or data storage **108** for storing in the database **110**. The data server and/or data storage **108** is arranged to be accessible to the decoder **112,** either via the communication network or via a direct connection, which is beneficially compatible with the encoder **102**, for subsequently decoding the encoded data (E2). In an example where the compression algorithm is applied to compress the encoded data (E2) to generate the compressed data (C4), the encoder **102** communicates the compressed data (C4) in the encoded data (E2) to the data server and/or data storage **108** either via a communication network or via a direct connection for storing in the database **110**.

In some examples, the decoder **112** is optionally operable to access the encoded data (E2) or the compressed data (C4) from the data server and/or data storage **108**. In alternative examples, the encoder **102** is optionally operable to stream the encoded data (E2) or the compressed data (C4) to the decoder **112**, either via the communication network **106** or via a direct connection. Optionally, in addition, a data file can be produced as output of the encoder **102** and used as input for the decoder

112. Moreover, it will be appreciated that a device equipped with a hardware or software encoder can also communicate directly with another device equipped with a hardware or software decoder. In yet other alternative examples, the decoder **112** is optionally implemented so as to retrieve the encoded data (E2) or the compressed data (C4) from a non-transitory (namely non-transient) computer-readable storage medium, such as a hard drive and a Solid-State Drive (SSD).

The decoder **112** includes processing hardware that is operable to execute computer-readable instructions stored on a non-transitory (namely non-transient) computer-readable storage medium for decoding the encoded data (E2) to generate corresponding decoded data (D3).

Optionally, the decoder **112** is implemented as a part of the computerized device **114**. In this case, the processing hardware of the decoder **112** is included in the computerized device **114**. Examples of the computerized device **114** include, but are not limited to, a mobile phone, a smart telephone, a Mobile Internet Device (MID), a tablet computer, an Ultra-Mobile Personal Computer (UMPC), a phablet computer, a Personal Digital Assistant (PDA), a web pad, a Personal Computer (PC), a handheld PC, a laptop computer, a desktop computer, a large-sized touch screen with an embedded PC, and an interactive entertainment device, such as a game console, a video player, a Television (TV) set and a Set-Top Box (STB).

Alternatively, optionally, the decoder **112** is implemented independently, for example, using another computerized device that includes the processing hardware of the decoder **112**.

When required, the processing hardware of the decoder **112** is operable to decode the encoded data (E2) to generate the corresponding decoded data (D3). In this regard, the processing hardware of the decoder **112** is operable to process the encoded data (E2) as data blocks and/or data packets.

The processing hardware of the decoder **112** is operable to decode the encoded data (E2) to generate data for the changed elements within the substantially reoccurring data blocks and/or data packets within the encoded data (E2). Optionally, for a lossless operation, the processing hardware of the decoder **112** is operable to decode the changed elements to their original form in the decoded data (D3).

Alternatively, optionally, for a lossy operation, the processing hardware of the decoder **112** is operable to decode at least a portion of the changed elements in a quantized manner in the decoded data (D3); optionally, the quantized manner of decoding employs a quantization which is variable, for example depending upon a format or type of data present in the encoded data (E2), wherein such variable quantization is capable on reducing power consumption in decoders, which is important for low-power portable devices, for example battery-powered devices. Yet alternatively, optionally, for a near-lossless operation, the processing hardware of the decoder **112** is operable to decode at least some portion of the changed elements in a quantized manner in the decoded data (D3).

Moreover, the processing hardware of the decoder **112** is operable to decode the encoded data (E2) to generate data for the unchanged elements within the substantially reoccurring data blocks and/or data packets within the encoded data (E2). In the encoded data (E2), the unchanged elements are represented by the at least one corresponding symbol, or at least one corresponding bit, for example a single bit, indicating the absence of change in the unchanged elements relative to corresponding elements in a reference data block and/or data packet, as described earlier. Accordingly, optionally, the processing hardware of the decoder **112** is operable to identify where the at least one corresponding symbol, or at least one corresponding bit, for example a single bit, has occurred within a given data block and/or data packet of the encoded data (E2), and to replace the at least one corresponding symbol, or alternatively, to set at least one data value to the position of the corresponding bit in the data block and/or data packet with corresponding elements in the reference data block and/or data packet.

In an example where the decoder **112** is provided with the compressed data (C4), the processing hardware of the decoder **112** is operable to apply a decompression algorithm to decompress the compressed data (C4) to generate the encoded data (E2) for decoding the encoded data (E2) to generate the data for the changed and unchanged elements.

Moreover, the processing hardware of the decoder **112** is operable to assemble the data generated for the changed and unchanged elements into data blocks and/or data packets to generate the decoded data (D3). Optionally, the decoded data (D3) so generated is in a form of at least one of: text data, image data, video data, audio data, binary data, sensor data, measurement data, graphical data, multi-dimensional data and/or one-dimensional data, but not limited thereto.

Subsequently, optionally, the decoder **112** is operable to send the decoded data (D3) to the computerized device **114**.

Furthermore, optionally, the encoder **102** and the decoder **112** are arranged to be implemented in a codec. Optionally, the codec is in a form of at least one of: a video codec, an audio codec, an image codec and/or a data codec, but not limited thereto. The codec optionally is used in portable electronic devices such as digital cameras, mobile telephones, smart phones, surveillance equipment and such like.

Moreover, optionally, the encoder **102** and the decoder **112** are operable to implement chunked transfer encoding for Hypertext Transfer Protocol (HTTP) and/or Real-Time Messaging Protocol (RTMP). Optionally, the HTTP and/or RTMP employ fixed-size data blocks and/or data packets inside requests and responses to the requests. Typically, the RTMP resets a size of the data block and/or data packets during a communication session periodically, based on a response time and capacity of a communication network.

Fig. 1 is merely an example, which should not unduly limit the scope of the claims herein. It is to be understood that the specific designation for the network environment **100** is provided as an example and is not to be construed as limiting the network environment **100** to specific numbers, types, or arrangements of encoders, electronic devices, decoders, computerized devices, data servers and/or data storages, databases, and communication networks. A person skilled in the art will recognize many variations, alternatives, and modifications of embodiments of the present disclosure.

Fig. 2 is an illustration of an example data flow, in accordance with an embodiment of the present disclosure. For illustration purposes, an example will be described wherein the electronic device **104** is an Internet Protocol (IP) camera that has been employed at a facility for implementing a remote surveillance system, for example, for detecting intruders and/or for detecting hazardous events, such as fire, flooding, and the like.

The IP camera is operable to provide the encoder **102** with sensor data as sensed by one or more image sensors included within the IP camera. In this example, the sensor data includes one-dimensional image data or-multi-dimensional image data and/or video data and/or other types of data.

Moreover, in operation, a video captured by the electronic device **104** is streamed to the computerized device **114** so as to be viewed by a user associated therewith.

In the example data flow, the input data (D1) is an original video captured by an IP camera. The input data (D1) is typically large in size; and therefore, requires a large data storage space for storing it in the database **110** and a large network bandwidth for transferring it via the communication network **106** or via a direct connection.

In order to encode the input data (D1) to the encoded data (E2), the processing hardware of the encoder **102** is operable to analyze content, type and/or composition of the input data (D1). Based on the analysis, the processing hardware of the encoder **102** is operable to divide image frames, views, or channels of the video into a plurality of data blocks. The analysis beneficially involves, for example, computing a spectrum of changes occurring in the input data (D1), for example a spectrum of temporal rates of change in the input data (D1), and also an analysis of spatial portions of temporal sequences of images and their associated rates of change in the input data (D1). (Beneficially, the analysis is operable to find periodicities in the data, namely recurring configurations of data, and where such periodicity is found, then it is advantageous to conduct the division of the data to blocks or packets based on such periodicity.

Optionally, each image frame, view or channel is divided into the data blocks in a similar manner. This is particularly beneficial for enabling selection of reference data blocks in previous image frames, views or channels.

This method is optionally used with a deduplication method or a block encoder, and a corresponding block decoder, that are operable to detect reoccurrences, namely selecting a suitable reference block, and to store and update the reference data blocks and/or data packets. These methods further are operable to split the data into suitable data blocks and/or data packets, and to combine the data blocks and/or data packets back to generate the decoded data (D3). Optionally, a block encoder is used to generate data streams that are then encoded with a deduplication method, for example as described in the foregoing.

Optionally, the data blocks are rectilinear in relation to areas of data frames represented by these data blocks, for example, 64 x 64 elements, 32 x 16 elements, 4 x 20 elements, 10 x 4 elements, 1 x 4 elements, 3 x 1 elements, 8 x 8 elements, 1 x 1 element and so on. However, it will be appreciated here that other shapes of data blocks can alternatively be employed, for example, such as triangular, hexagonal, elliptical and circular. In one example, the input data (D1) corresponds to an image of billowing smoke or flames, or a turbulent water flow that includes multiple curved image components that are inefficiently represented by rectilinear data blocks, but map efficiently onto elliptical and circular data blocks, thereby providing potentially a higher degree of data compression. Moreover, the term '*data block*' optionally refers to a data block as well as data segments included within the data block, throughout the present disclosure.

Optionally, the data blocks have a predefined size. The predefined size is optionally either user-defined or system-defined by default. Optionally, the predefined size is defined by the encoder **102** based on the analysis of the content, type and/or composition of the input data (D1), as described in the foregoing. Optionally, the data blocks have a fixed size. Therefore, the size of the data blocks is either known to the decoder **112** or transmitted only once to the decoder **112**. When the data blocks have a fixed size, header information describing the size of the data blocks is not required to be written or transmitted, thereby enabling greater data compression to be achieved.

Moreover, optionally, the processing hardware of the encoder **102** is operable to define reference data blocks of a reference frame corresponding to data blocks of a first image frame, or data block or data packet, of the video. For this purpose, the processing hardware of the encoder **102** is operable to reset data values of elements of the reference data blocks or data packets to a predetermined data value. Optionally, the predetermined data value is implemented as a zero data value.

Alternatively, optionally, the processing hardware of the encoder **102** is operable to write or transmit the first image frame, or data block or data packet, as it is, and to define the data blocks of the first image frame as the reference data blocks for a next image frame. In an example, for a given data block in a current image frame, a reference data block is selected to be a data block that is at a position in a previous image frame that is similar to a position of the given data block in the current image frame. In this example, data values of the previous image frame are required to be stored in memory, instead of data values of the reference data block.

For a given data block, the processing hardware of the encoder **102** is operable to process through data values of all elements of the given data block, and to compare the data values of the elements of the given data block with data values of corresponding elements of its corresponding reference data block.

Optionally, for a lossless operation, the data values are read without quantization. If a data value of a particular element of the given data block is different from a data value of a corresponding element of the reference data block, that particular element is considered a changed element. Otherwise, if the data value of the particular element of the given data block is identical, or optionally substantially identical, to the data value of the corresponding element of the reference data block, that particular element is considered an unchanged element.

Alternatively, optionally, for a lossy operation, the data values are quantized before the comparison is made. If a difference between a quantized data value of a particular element of the given data block and a quantized data value of a corresponding element of the reference data block is greater than a predefined threshold value, that particular element is considered a changed element. Otherwise, if the difference between the quantized data values is smaller or equal to the predefined threshold value, that particular element is considered an unchanged element. Optionally, the predefined threshold value is based on a quality level set for the lossy operation. The better the quality that is required, the smaller threshold value that is used. The predefined threshold is zero for lossless operation.

Subsequently, the processing hardware of the encoder **102** is operable to encode, in the encoded data (E2), unchanged elements of the given data block using at least one corresponding symbol, or at least one corresponding bit, for example a single bit, as described earlier. Additionally, the processing hardware of the encoder **102** is operable to encode the changed elements in the encoded data (E2), as described earlier.

Additionally, optionally, the processing hardware of the encoder **102** is operable to write data values of the changed elements to the current or new reference data block. Thus, the reference data block is optionally also used for a next data block, or next data blocks.

In this manner, the processing hardware of the encoder **102** is operable to encode the data blocks of the input data (D1) in the encoded data (E2). The encoder **102** then communicates the encoded data (E2) to the decoder **112**, as shown in Fig. 2.

Next, in order to decode the encoded data (E2) to generate the corresponding decoded data (D3), the processing hardware of the decoder **112** is operable to identify occurrences of the at least one corresponding symbol or one corresponding bit within a given data block within the encoded data (E2), and to replace the at least one corresponding symbol or set the data value in the position of the corresponding bit in a data block and/or data packet with corresponding elements in the reference data block.

Subsequently, the processing hardware of the decoder **112** is operable to assemble data generated for the changed and unchanged elements, to generate the decoded data (D3).

Subsequently, the decoder **112** sends the decoded data (D3) to the computerized device **114**. Continuing from the aforementioned example of the data flow where the input data (D1) is the original video, the user is presented the video on a display screen of the computerized device **114**.

Moreover, the encoder **102** optionally streams the encoded data (E2) to the decoder **112**, whilst concurrently encoding the input data (D1) in real time. This is particularly beneficial in a situation where source data is encoded at a multimedia server in real time for streaming to users, for example, for Internet-delivered multimedia services.

Fig. 2 is merely an example, which should not unduly limit the scope of the claims herein. A person skilled in the art will recognize many variations, alternatives, and modifications of embodiments of the present disclosure.

In another example, the encoder **102** and/or the decoder **112** are implemented in a similar manner to encode audio data, wherein the audio data may be divided into a plurality of data packets and/or data sections. The term ‘data packet and/or data section’ is synonymous with the term ‘data block and/or data packet’, but pertains to audio data rather than image and/or video data. Optionally, the processing hardware of the encoder **102** is operable to concurrently encode audio data along with image and/or video data. When audio signals include tones which are sustained over many cycles of audio signal and/or the music is rhythmic in nature with substantially repetitive rhythmical patterns, such audio signals are efficiently compressed using encoding methods of the present disclosure. Most contemporary popular music tends to be repetitively rhythmic in nature.

However, it will be appreciated that the encoder **102** can be used to encode other types of data in a similar manner, for example, including at least one of: economic data, measurement data, seismographic data, analog-to-digital converted data, biomedical signal data, textural data, calendar data, mathematical data, and binary data, but not limited thereto.

18 01 16

The encoder **102** and the decoder **112** are suitable for various types of data, because a majority of data being processed or encoded has been produced in a machine language by machines, and therefore, is divisible to data blocks and/or data packets.

Moreover, principal traffic in communication networks includes transmitting requests and responses to requests. This means data bytes are sent back and forth. These data bytes mostly contain partially or entirely similar IP packet data. Therefore, the encoder **102** and the decoder **112** are well-suited for communication protocols used in transferring data packets.

For illustration purposes only, there will next be considered an example wherein the input data (D1) includes five Transmission Control Protocol/Internet Protocol (TCP/IP) frames, represented as following:

```
00 0c 29 9d b4 1d 00 10 f3 2a 2a ac 08 00 45 00
00 28 78 fa 40 00 76 06 ce 63 3e f1 c1 34 ac 10
11 3c 22 c9 c0 dc bd 1f b7 03 1f fe c0 92 50 10
00 fe b9 09 00 00
```

```
00 0c 29 9d b4 1d 00 10 f3 2a 2a ac 08 00 45 00
00 28 79 0d 40 00 76 06 ce 50 3e f1 c1 34 ac 10
11 3c 22 c9 c0 dc bd 1f b7 03 1f fe c9 cc 50 10
01 02 af cc 00 00

00 0c 29 9d b4 1d 00 10 f3 2a 2a ac 08 00 45 00
00 28 79 11 40 00 76 06 ce 4c 3e f1 c1 34 ac 10
11 3c 22 c9 c0 dc bd 1f b7 03 1f fe d1 41 50 10
01 02 a8 57 00 00

00 0c 29 9d b4 1d 00 10 f3 2a 2a ac 08 00 45 00
00 28 79 19 40 00 76 06 ce 44 3e f1 c1 34 ac 10
11 3c 22 c9 c0 dc bd 1f b7 03 1f fe dd a4 50 10
01 02 9b f4 00 00

00 0c 29 9d b4 1d 00 10 f3 2a 2a ac 08 00 45 00
00 28 79 43 40 00 76 06 ce 1a 3e f1 c1 34 ac 10
11 3c 22 c9 c0 dc bd 1f b7 03 1f fe e9 95 50 10
01 02 90 03 00 00
```

In the example, the five TCP/IP frames have been randomly selected from numbers #185 to #198. These TCP/IP frames require a total of 270 bytes (= 2160 bits) for communicating over a communication network.

In operation, the processing hardware of the encoder **102** processes the TCP/IP frames as individual data blocks, and encodes them in the encoded data (E2), represented as following:

```
00 0c 29 9d b4 1d 00 10 f3 2a 2a ac 08 00 45 00
00 28 78 fa 40 00 76 06 ce 63 3e f1 c1 34 ac 10
11 3c 22 c9 c0 dc bd 1f b7 03 1f fe c0 92 50 10
00 fe b9 09 00 00

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 79 0d 00 00 00 00 00 50 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 c9 cc 00 00
01 02 af cc 00 00

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 11 00 00 00 00 00 4c 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 d1 41 00 00
00 00 a8 57 00 00

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 19 00 00 00 00 00 44 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 dd a4 00 00
00 00 9b f4 00 00
```

```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 43 00 00 00 00 00 1a 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 e9 95 00 00
00 00 90 03 00 00
```

In this example, the data value ‘00’ represents at least one corresponding symbol indicating an absence of change in unchanged elements relative to corresponding elements in a reference data block. Although the data value ‘00’ is also present in the input data (D1), it does not cause any problems in this example, because no changed data value is represented by ‘00’. Often, there is a need to use a symbol for unchanged data elements that is not present in the input data (D1). Optionally, unchanged/changed decision bits can be delivered with the changed data values instead of delivering the predefined symbols for unchanged data and the changed data values.

It is evident that the encoded data (E2) has low entropy, in comparison to the input data (D1). When the encoded data (E2) is entropy-coded with an advanced range coding method that is based on arithmetic compression, compressed data (C4) so generated requires only 113 bytes (= 904 bits) for communicating over the communication network. Correspondingly, when the input data (D1) is entropy-coded in a similar manner, compressed input data so generated requires 253 bytes (= 2024 bits). Thus, an amount of data to be communicated over the communication network is reduced by 140 bytes (253 – 113), namely, by 55.3 %. The original amount of input data (D1) without any compression is 270 bytes.

In another example, there are processed two first frames of the previous example by using quantization and bits describing the unchanged/changed decision. Quantization is applied by using the divider value ‘2’, and dequantization by using the multiplier value ‘2’. The threshold for changed values is ‘1’, and the changed values are expressed with the bit value ‘1’, and the unchanged values with the bit value ‘0’. The original data frames are as follows:

```
00 0c 29 9d b4 1d 00 10 f3 2a 2a ac 08 00 45 00
00 28 78 fa 40 00 76 06 ce 63 3e f1 c1 34 ac 10
11 3c 22 c9 c0 dc bd 1f b7 03 1f fe c0 92 50 10
00 fe b9 09 00 00
```

```
00 0c 29 9d b4 1d 00 10 f3 2a 2a ac 08 00 45 00
00 28 79 0d 40 00 76 06 ce 50 3e f1 c1 34 ac 10
11 3c 22 c9 c0 dc bd 1f b7 03 1f fe c9 cc 50 10
01 02 af cc 00 00
```

The quantization is applied before the method, namely in reoccurrance detection, as described in the foregoing.

The first frame values after quantization are as follows:

```
00 06 14 4e 5a 0e 00 08 79 15 15 56 04 00 22 00
00 14 3c 7d 20 00 3b 03 67 31 1f 78 60 1a 56 08
08 1e 11 64 60 6e 5e 0f 5b 01 0f 7f 60 49 28 08
00 7f 5c 04 00 00
```

These values are delivered for the first frame, and they are set as a previous buffer for prediction purposes of the next frame.

The input data values for the second frame after quantization are as follows:

```
00 06 14 4e 5a 0e 00 08 79 15 15 56 04 00 22 00
00 14 3c 06 20 00 3b 03 67 28 1f 78 60 1a 56 08
08 1e 11 64 60 6e 5e 0f 5b 01 0f 7f 64 66 28 08
00 01 57 66 00 00
```

18 01 16

Now, the changed/unchanged bit stream can be generated and it is as follows:

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0   0 1 1 1 0 0
```

It contains 54 bits (7 ones and 47 zeros), and it can be efficiently compressed later by an entropy encoder. For example, RLE generates a stream of data 19, 1, 5, 1, 18, 2, 3, 3, 2, which can be delivered , for example, with nine five bits values (= 9 * 5 bits = 45 bits). Range coding with probabilities 0.13 and 0.87 is also useable for delivering the bits.

The seven changed values for the second frame are then as follows:

```
06 28 64 66 01 57 66
```

These seven changed values are optionally compressed with the first frame values (54), for example by using Range coding. Without entropy encoding, they require 61 *

7 bits = 428 bits. When the frames are decoded, then, after the dequantization, the values for the two frames are as follows:

```
00 0c 28 9c b4 1c 00 10 f2 2a 2a ac 08 00 44 00
00 28 78 fa 40 00 76 06 ce 62 3e f0 c0 34 ac 10
10 3c 22 c8 c0 dc bc 1e b6 02 1e fe c0 92 50 10
00 fe b8 08 00 00
```

```
00 0c 28 9c b4 1c 00 10 f2 2a 2a ac 08 00 44 00
00 28 78 0c 40 00 76 06 ce 50 3e f0 c0 34 ac 10
10 3c 22 c8 c0 dc bc 1e b6 02 1e fe c8 cc 50 10
01 02 ae cc 00 00
```

Fig. 3 is an illustration of steps of a method of encoding input data (D1) to generate corresponding encoded data (E2), in accordance with an embodiment of the present disclosure. The method is depicted as a collection of steps in a logical flow diagram, which represents a sequence of steps that can be implemented in hardware, software, or a combination thereof.

Optionally, at a step **302**, substantial reoccurrences, namely repetitions, of data blocks and/or data packets within the input data (D1) are identified. Optionally, this step **302** is implemented with a deduplication method or with a block encoder. Usually, the step **302** also includes an operation of splitting the input data (D1) into new data blocks or data packets. Alternatively, the data split has already been carried out before the step **302**. Beneficially, only partially changed data blocks and/or data packets are delivered further to a step **304** for encoding, wherein the changed data blocks and/or data packets or unchanged data blocks and/or data packets are encoded with different methods, as aforementioned. Optionally, the changed data blocks and/or data packets and/or unchanged data blocks and/or data packets are also delivered to the step **304** for encoding them.

It will be appreciated that, if a certain given reference data block and/or packet is used, then it does not cause extra data to be transmitted. However, in some situations, the information on the selection of the reference data block and/or packet needs to be transmitted so that the decoder **112** will be aware of the reference data block and/or packet and use the same one. Correspondingly, when partially changed data blocks and/or packets are encoded, then if a portion of the data blocks and/or packets are not given to the method, then the network environment **100** needs to have some sort of method selection information available, which also needs to be delivered to the decoder **112**. Optionally, as mentioned above, it is a block encoder or a deduplication algorithm that takes care of delivering this piece of information, otherwise all data blocks and/or packets are processed with the method pursuant to the disclosure..

At the step **304**, it is identified where elements are unchanged within the substantially reoccurring data blocks and/or data packets, and/or where elements are changed within the substantially reoccurring data blocks and/or data packets.

Next, at a step **306**, changed and unchanged elements are encoded in the encoded data (E2) as one data stream. Optionally, two data streams are processed, wherein one data stream contains unchanged/changed decision bits and the other data stream contains changed data values that are to be used.

18 01 16

In accordance with the step **306**, the unchanged elements are encoded in the encoded data (E2) by employing at least one corresponding symbol, or at least one corresponding bit, for example a single bit, indicating an absence of change in the unchanged elements relative to corresponding elements in a reference data block and/or data packet. Optionally, the at least one corresponding symbol is represented by a predetermined data value, which is optionally implemented as a zero data value.

Optionally, for a lossless operation, the changed elements are encoded in their original form in the encoded data (E2). Alternatively, optionally, for a lossy operation, at least a portion of the changed elements is encoded in a quantized manner in the encoded data (E2). Yet alternatively, optionally, for a near-lossless operation, at least a portion of the changed elements is encoded in a quantized manner in the encoded data (E2).

The steps **304** and **306** are beneficially performed for partially changed data blocks and/or data packets. Optionally, the steps **304** and **306** are performed for each data block and/or data packet of the input data (D1). An encoding processing of the steps **304** and **306** has been described in conjunction with Figs. 4A and 4B.

Optionally, the method includes an additional step **308** at which a compression algorithm is applied to compress the encoded data (E2) to generate compressed data

(C4). The compressed data (C4) is relatively small in size in comparison to the input data (D1); and therefore, requires a small space for data storage and a small network bandwidth for data transfer over a communication network or over a direct connection. In the step **308**, it is beneficial that typically entropy encoding methods such as Range coding, SRLE, EM, ODelta coding and so forth are used there. The step **308** is typically executed with a deduplication method or with a block encoder.

It will be appreciated that compression is optional, but if it is carried out, then either a certain compression method is always used, or alternatively, information conveying the selected compression method needs to be delivered from the encoder **102** to the decoder **112**. Typically, the selection of compression method has been left as the responsibility of the block encoder or the deduplication method, but if necessary, the method pursuant to the present disclosure can express and deliver the information conveying information regarding the selected compression method, otherwise a certain compression method is always used, for example a default compression method.

18 01 16

Furthermore, the used reference data block and/or packet needs to be updated, or a new reference data block and/or reference packet needs to be generated. Such a step is also typically implemented with a deduplication method or with a block encoder, but optionally the update of a reference data block and/or packet is carried out with the steps **304** and **306** for reducing the amount of data copying.

Moreover, an amount of background memory allocated for the encoding processing needs to be only as large as an amount of elements in a current data block and/or data packet, times the amount of different reference data blocks and/or packets. If only one reference data block and/or packet is used, then the amount of background memory needed is the same as the size of the data block and/or packet. Moreover, a result of the encoding processing, namely, the encoded data (E2), is optionally written or transmitted directly into an original memory. This means that no separate transfer memories are required. Therefore, the method is capable of functioning as an in-place (*in situ*) operation, and is cost-effective.

The steps **302** to **308** are only illustrative and other alternatives can also be provided where one or more steps are added, one or more steps are removed, or one or more steps are provided in a different sequence without departing from the scope of the claims herein.

Figs. 4A and 4B collectively are an illustration of steps of the encoding processing, in accordance with an embodiment of the present disclosure. At a step **402**, the data block and/or packet is read, and the reference data block and/or packet is also read.

At a step **404**, a data value of an element of a given data block and/or data packet of the input data (D1) is read, and a data value of an element of a given reference data block and/or packet is also read.

Next, at a step **406**, it is determined whether or not the data value of the element has changed against a data value of a corresponding element in a reference data block and/or data packet. Beneficially, such determination detects that the value has changed when the absolute difference between the values is higher than the threshold value. Optionally, other methods can be used for determining changes of data values.

If, at the step **406**, it is determined that the data value of the element has not changed, a step **408** is performed. At the step **408**, a predetermined data value is written to the encoded data (E2) to indicate an absence of change, otherwise a decision bit indicating that the data value of the element has not changed is written to a separate stream.

However, if it is determined that the data value of the element has changed, a step **410** is performed. At the step **410**, the data value of the element is written to the encoded data (E2) without or with optional quantization, and additionally, optionally a decision bit indicating that the data value of the element has changed is written to a separate stream. Optional quantization is also performed during the reoccurrence detection.

Optionally, at the steps **406** and **408**, the data value of the element is also written to the new reference data block and/or data packet. Alternatively, optionally, the changed data value in the step **408** is updated also to the current reference data block and/or packet by using the original changed value with lossless processing and preferably decoded changed value with lossy processing. Alternatively, the reference values are generated or updated with a deduplication method or with a block encoder. The updated reference block and/or packet values are optionally used for next data blocks and/or data packets.

Next, at a step **412**, it is determined whether or not a next element exists in the given data block and/or data packet. If it is determined that a next element exists, the encoding processing restarts at the step **404**. Otherwise, if it is determined that no next element exists in the given data block and/or data packet, a step **414** is performed.

At the step **414**, it is determined whether or not a next data block and/or data packet exists in the input data (D1). If it is determined that a next data block and/or data packet exists, the encoding processing restarts at the step **402**. Otherwise, if it is determined that no next data block and/or data packet exists in the input data (D1), the encoding processing stops.

The steps **402** to **414** are only illustrative and other alternatives can also be provided where one or more steps are added, one or more steps are removed, or one or more steps are provided in a different sequence without departing from the scope of the claims herein.

Embodiments of the present disclosure provide a computer program product comprising a non-transitory (namely non-transient) computer-readable storage medium having computer-readable instructions stored thereon, the computer-readable instructions being executable by a computerized device comprising processing hardware to execute the method as described in conjunction with Figs. 3 and 4A-B. The computer-readable instructions are optionally downloadable from a software application store, for example, from an “App store” to a computerized device.

Fig. 5 is an illustration of steps of a method of decoding encoded data (E2) to generate corresponding decoded data (D3), in accordance with an embodiment of the present disclosure. The method is depicted as a collection of steps in a logical flow diagram, which represents a sequence of steps that can be implemented in hardware, software, or a combination thereof.

Optionally, at a step **502**, decoding of entropy-encoded data streams from the compressed data (C4) to generate the encoded data (E2) is applied. This step **502** optionally typically employs entropy decoding methods such as Range coding, SRLE, EM, ODelta coding and so forth. This step **502** is typically executed by employing a deduplication method or by employing a block decoder.

At a step **504**, the encoded data (E2) is searched to identify changed and unchanged elements within substantial reoccurrences of data blocks and/or data packets within the encoded data (E2).

At a step **506**, the encoded data (E2) is decoded to generate data for changed and unchanged elements within the substantial reoccurrences of data blocks and/or data packets within the encoded data (E2). In accordance with the step **504**, occurrences of at least one corresponding symbol or one corresponding bit indicative of an absence of change are identified, and replaced with the corresponding symbol, or set to the data value for position of the corresponding bit with corresponding elements in a reference data block and/or data packet. For changed blocks, the absence of change is not detected, the encoded value is used. Optionally, for a lossless operation, the changed elements are decoded to their original form in the decoded data (D3). Alternatively, optionally, for a lossy operation, at least a portion of the changed elements is decoded in a quantized manner, namely dequantization is carried out, in the decoded data (D3). Yet alternatively, optionally, for a near-lossless operation, a part of the changed elements is decoded in a quantized manner in the decoded data (D3).

The steps **504** and **506** are performed for partially changed data blocks and/or data packets. Optionally, the steps **504** and **506** are performed for each data block and/or data packet of the encoded data (E2). A decoding processing of the steps **504** and **506** has been described in conjunction with Figs. 6A and 6B.

Optionally, at a step **508**, the data generated for the changed and unchanged elements is assembled from the data blocks and/or data packets to generate the decoded data (D3). This step **508** typically employs a deduplication method or a block decoder.

Moreover, the used reference data block and/or packet needs to be updated or a new reference data block and/or packet needs to be generated. Such a step is also typically performed with a deduplication method or with a block decoder, but optionally, the updating of reference data block and/or packet is optionally carried out within the steps **504** and **506** for reducing the amount of data copying occurring during decoding operations.

The steps **502** to **508** are only illustrative and other alternatives can also be provided where one or more steps are added, one or more steps are removed, or one or more steps are provided in a different sequence without departing from the scope of the claims herein.

Figs. 6A and 6B collectively are an illustration of steps of the decoding processing, in accordance with an embodiment of the present disclosure.

At a step **602**, a stream of data values of a given data block and/or data packet of the encoded data (E2) is received. Optionally, the stream of bits describing unchanged/changed elements in the data block and/or packet are also received. Also, the reference data block and/or packet is read during the step 602.

Next, at a step **604**, an element of a given data block and/or packet is read. Alternatively, the unchanged/changed decision bit is read. An element of the reference data block or packet is also read during the step **604**.

Next, at a step **606**, it is determined whether or not a given element is unchanged. An unchanged element is detected if a predetermined data value has occurred. Alternatively, an unchanged element is detected, if the decision bit is unchanged.

If, at the step **606**, the unchanged element is detected, a step **608** is performed. At the step **608**, a data value of a corresponding element in a reference data block and/or data packet is written to the decoded data (D3).

Otherwise, if it is determined that the element has changed, a step **610** is performed. At the step **610**, the data value of the element is written to the decoded data (D3) without or with optional dequantization. If the detection of changed value was made based on the changed bit in decision bits, then the encoded value first needs to be read at this step, before it can be written to the decoded data (D3). Optional quantization can also be performed during the data assembly.

Optionally, at the steps **608** and **610**, the data value of the element is also written to the new reference data block and/or packet. Alternatively, optionally, the changed data value in the step **610** is also updated to the current reference data block and/or packet by using the decoded changed value with lossless and lossy processing. Alternatively, the reference values are generated or updated with a deduplication method or with a block decoder, for example as described in the foregoing. The updated reference data block and/or packet values is optionally used for next data blocks and/or packets.

Next, at a step **612**, it is determined whether or not a next element exists in the given data block and/or data packet. If it is determined that a next element exists, the decoding processing restarts at the step **604**. Otherwise, if it is determined that no next element exists in the given data block and/or data packet, a step **614** is performed.

At the step **614**, it is determined whether or not a next data block and/or data packet exists in the encoded data (E2). If it is determined that a next data block and/or data packet exists, the decoding processing restarts at the step **602**. Otherwise, if it is determined that no next data block and/or data packet exists in the encoded data (E2), the decoding processing stops.

The steps **602** to **614** are only illustrative and other alternatives can also be provided where one or more steps are added, one or more steps are removed, or one or more steps are provided in a different sequence without departing from the scope of the claims herein.

Embodiments of the present disclosure provide a computer program product comprising a non-transitory (namely non-transient) computer-readable storage medium having computer-readable instructions stored thereon, the computer-readable instructions being executable by a computerized device comprising processing hardware to execute the method as described in conjunction with Figs. 5, and Figs. 6A to 6B. The computer-readable instructions are optionally downloadable from a software application store, for example, from an "App store" to a computerized device, such as the computerized device **114**.

Furthermore, embodiments of the present disclosure provide a codec including at least one encoder as described in conjunction with Figs. 3, and Figs. 4A to 4B, and at least one decoder as described in conjunction with Figs. 5 and Figs. 6A to 6B.

Embodiments of the present disclosure are susceptible to being used for various purposes, including, though not limited to, enabling lossless or near-lossless data compression of one-dimensional image data or multi-dimensional image data, video data, audio data and any other type of data with a high compression ratio, in comparison to conventional codecs.

Modifications to embodiments of the present disclosure described in the foregoing are possible without departing from the scope of the present disclosure as defined by the accompanying claims. Expressions such as "including", "comprising", "incorporating", "consisting of", "have", "is" used to describe and claim the present disclosure are intended to be construed in a non-exclusive manner, namely allowing for items, components or elements not explicitly described also to be present. Reference to the singular is also to be construed to relate to the plural.

CLAIMS

We claim:

1. An encoder including processing hardware for encoding input data (D1) to generate corresponding encoded data (E2), wherein the processing hardware is operable to process the input data (D1) as data blocks and/or data packets, characterized in that

the processing hardware is operable to:

(i) identify substantial reoccurrences of data blocks and/or data packets within at least a portion of the input data (D1), wherein the data blocks and/or data packets include a corresponding plurality of elements, wherein the elements include a plurality of bits;

(ii) identify where elements are unchanged within the substantially reoccurring data blocks and/or data packets, and/or where elements are changed within the substantially reoccurring data blocks and/or data packets;

18 01 16

(iii) encode unchanged elements in the encoded data (E2) by employing at least one corresponding symbol or at least one corresponding bit indicating an absence of change in the unchanged elements relative to corresponding elements in a reference data block and/or data packet; and

(iv) encode changed elements in the encoded data (E2).

2. An encoder as claimed in claim 1, characterized in that the input data (D1) is in a form of at least one of: text data, image data, video data, audio data, binary data, sensor data, measurement data, graphical data, multi-dimensional data, uni-dimensional data.

3. An encoder as claimed in claim 1 or 2, characterized in that the at least one corresponding symbol is represented by a predetermined data value.

4. An encoder as claimed in claim 3, characterized in that the predetermined data value is implemented as a zero data value.

5. An encoder as claimed in any one of the preceding claims, characterized in that the processing hardware is operable to implement chunked transfer encoding for Hypertext Transfer Protocol (HTTP) and/or Real-Time Messaging Protocol (RTMP).

6. An encoder as claimed in claim 5, characterized in that the HTTP and/or RTMP employ fixed-size data blocks and/or data packets inside requests and responses.

7. An encoder as claimed in any one of the preceding claims, characterized in that the processing hardware is operable to encode at least a portion of the changed elements in a quantized manner in the encoded data (E2).

8. An encoder as claimed in any one of the preceding claims, characterized in that the processing hardware is operable to apply a compression algorithm to compress the encoded data (E2).

18 01 16

9. A method of encoding input data (D1) to generate corresponding encoded data (E2), wherein the method includes processing the input data (D1) as data blocks and/or data packets, characterized in that

the method includes:

(i) identifying substantial reoccurrences of data blocks and/or data packets within at least a portion of the input data (D1), wherein the data blocks and/or data packets include a corresponding plurality of elements, wherein the elements include a plurality of bits;

(ii) identifying where elements are unchanged within the substantially reoccurring data blocks and/or data packets, and/or where elements are changed within the substantially reoccurring data blocks and/or data packets;

(iii) encoding unchanged elements in the encoded data (E2) by employing at least one corresponding symbol or at least one corresponding bit indicating an absence of change in the unchanged elements relative to corresponding elements in a reference data block and/or data packet; and

(iv) encoding changed elements in the encoded data (E2).

10. A method as claimed in claim 9, characterized in that the method includes encoding the input data (D1) received in a form of at least one of: text data, image data, video data, audio data, binary data, sensor data, measurement data, graphical data, multi-dimensional data, uni-dimensional data.

11. A method as claimed in claim 9 or 10, characterized in that the method includes representing the at least one corresponding symbol by a predetermined data value.

12. A method as claimed in claim 11, characterized in that the predetermined data value is implemented as a zero data value.

13. A method as claimed in any one of claims 9 to 12, characterized in that the method includes implementing chunked transfer encoding for Hypertext Transfer Protocol (HTTP) and/or Real-Time Messaging Protocol (RTMP).

18 01 16

14. A method as claimed in claim 13, characterized in that the HTTP and/or RTMP employ fixed-size data blocks and/or data packets inside requests and responses.

15. A method as claimed in any one of claims 9 to 14, characterized in that the method includes encoding at least a portion of the changed elements in a quantized manner in the encoded data (E2).

16. A method as claimed in any one of claims 9 to 15, characterized in that the method includes applying a compression algorithm to compress the encoded data (E2) to generate corresponding compressed data (C4).

17. A computer program product comprising a non-transitory computer-readable storage medium having computer-readable instructions stored thereon, the computer-readable instructions being executable by a computerized device comprising processing hardware to execute a method as claimed in claim 9.

18. A decoder including processing hardware for decoding encoded data (E2) to generate corresponding decoded data (D3), wherein the processing hardware is

operable to process the encoded data (E2) as data blocks and/or data packets, characterized in that

the processing hardware is operable to:

(i) decode the encoded data (E2) to generate data for changed elements, the changed elements being elements that are changed within substantial reoccurrences of data blocks and/or data packets within the encoded data (E2);

(ii) decode the encoded data (E2) to generate data for unchanged elements, the unchanged elements being elements that are unchanged within the substantial reoccurrences of data blocks and/or data packets within the encoded data (E2), wherein the unchanged elements are represented by at least one corresponding symbol or at least one corresponding bit indicating an absence of change in the unchanged elements relative to corresponding elements in a reference data block and/or data packet; and

(iii) assemble the data generated for the changed and unchanged elements in (i) and (ii) into data blocks and/or data packets to generate the decoded data (D3), wherein the data blocks and/or data packets include a corresponding plurality of elements, wherein the elements include a plurality of bits.

18 01 16

19. A decoder as claimed in claim 18, characterized in that the decoded data (D3) is in a form of at least one of: text data, image data, video data, audio data, binary data, sensor data, measurement data, graphical data, multi-dimensional data, uni-dimensional data.

20. A decoder as claimed in claim 18 or 19, characterized in that the at least one corresponding symbol is represented by a predetermined data value.

21. A decoder as claimed in claim 20, characterized in that the predetermined data value is implemented as a zero data value.

22. A decoder as claimed in any one of claims 18 to 21, characterized in that the processing hardware is operable to implement chunked transfer encoding for Hypertext Transfer Protocol (HTTP) and/or Real-Time Messaging Protocol (RTMP).

23. A decoder as claimed in claim 22, characterized in that the HTTP and/or RTMP employ fixed-size data blocks and/or data packets inside requests and responses.

24. A decoder as claimed in any one of claims 18 to 23, characterized in that the processing hardware is operable to decode at least a portion of the changed elements in a quantized manner in the decoded data (D3).

25. A decoder as claimed in any one of claims 18 to 24, characterized in that the processing hardware is operable to apply a decompression algorithm to decompress compressed data (C4) to generate the encoded data (E2) for decoding the encoded data (E2) to generate the data for the changed and unchanged elements.

26. A method of decoding encoded data (E2) to generate corresponding decoded data (D3), wherein the method includes processing the encoded data (E2) as data blocks and/or data packets, characterized in that

the method includes:

(i) decoding the encoded data (E2) to generate data for changed elements, the changed elements being elements that are changed within substantial reoccurrences of data blocks and/or data packets within the encoded data (E2);

(ii) decoding the encoded data (E2) to generate data for unchanged elements, the unchanged elements being elements that are unchanged within the substantial reoccurrences of data blocks and/or data packets within the encoded data (E2), wherein the unchanged elements are represented by at least one corresponding symbol or at least one corresponding bit indicating an absence of change in the unchanged elements relative to corresponding elements in a reference data block and/or data packet; and

(iii) assembling the data generated for the changed and unchanged elements in steps (i) and (ii) into data blocks and/or data packets to generate the decoded data (D3), wherein the data blocks and/or data packets include a

18 01 16

corresponding plurality of elements, wherein the elements include a plurality of bits.

27. A method as claimed in claim 26, characterized in that the method includes generating the decoded data (D3) in a form of at least one of: text data, image data, video data, audio data, binary data, sensor data, measurement data, graphical data, multi-dimensional data, uni-dimensional data.

28. A method as claimed in claim 26 or 27, characterized in that the at least one corresponding symbol is represented by a predetermined data value.

29. A method as claimed in claim 28, characterized in that the predetermined data value is implemented as a zero data value.

30. A method as claimed in any one of claims 26 to 29, characterized in that the method includes implementing chunked transfer encoding for Hypertext Transfer Protocol (HTTP) and/or Real-Time Messaging Protocol (RTMP).

31. A method as claimed in claim 30, characterized in that the HTTP and/or RTMP employ fixed-size data blocks and/or data packets inside requests and responses.

32. A method as claimed in any one of claims 26 to 31, characterized in that the method includes decoding at least a portion of the changed elements in a quantized manner in the decoded data (D3).

33. A method as claimed in any one of claims 26 to 32, characterized in that the method includes applying a decompression algorithm to decompress compressed data (C4) to generate the encoded data (E2) for decoding the encoded data (E2) to generate the data for the changed and unchanged elements.

34. A computer program product comprising a non-transitory computer-readable storage medium having computer-readable instructions stored thereon, the computer-readable instructions being executable by a computerized device comprising processing hardware to execute a method as claimed in claim 26.

35. A codec including at least one encoder as claimed in claim 1 for encoding input data (D1) to generate corresponding encoded data (E2), and at least one decoder as claimed in claim 18 for decoding the encoded data (E2) to generate corresponding decoded data (D3).