



(51) International Patent Classification:

G10L 19/00 (2013.01)

(21) International Application Number:

PCT/CA2020/050943

(22) International Filing Date:

07 July 2020 (07.07.2020)

(25) Filing Language:

English

(26) Publication Language:

English

(30) Priority Data:

62/871,253

08 July 2019 (08.07.2019)

US

(71) Applicant: VOICEAGE CORPORATION [CA/CA];

750 Lucerne Road, Suite 250, Town of Mount Royal,
Québec H3R 2H6 (CA).

(72) Inventor: EKSLER, Vaclav; Hlavni 145/110, 624 00 Brno

(CZ).

(74) Agent: PRINCE, Gaetan; 1100 Rene-Levesque boul.

West, 25th floor, Montreal, Québec H3B 5C9 (CA).

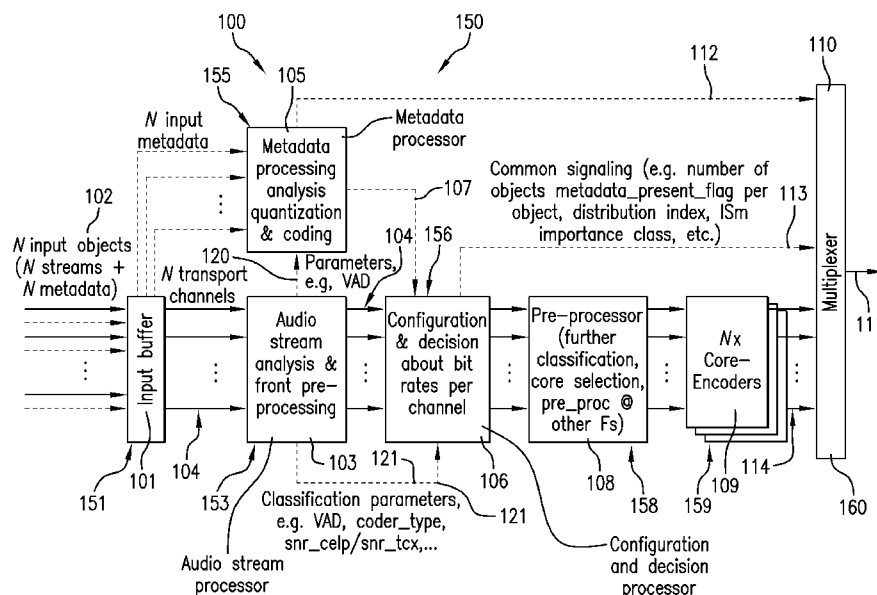
(81) Designated States (unless otherwise indicated, for every
kind of national protection available): AE, AG, AL, AM,AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ,
CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO,
DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN,
HR, HU, ID, IL, IN, IR, IS, IT, JO, JP, KE, KG, KH, KN,
KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD,
ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO,
NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW,
SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN,
TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.(84) Designated States (unless otherwise indicated, for every
kind of regional protection available): ARIPO (BW, GH,GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ,
UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ,
TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK,
EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV,
MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM,(54) Title: METHOD AND SYSTEM FOR CODING METADATA IN AUDIO STREAMS AND FOR FLEXIBLE INTRA-OBJECT
AND INTER-OBJECT BITRATE ADAPTATION

FIG. 1

(57) **Abstract:** A system and method code an object-based audio signal comprising audio objects in response to audio streams with associated metadata. In the system and method, an audio stream processor analyses the audio streams. A metadata processor is responsive to information on the audio streams from the analysis by the audio stream processor for coding the metadata. The metadata processor uses a logic for controlling a metadata coding bit-budget. An encoder codes the audio streams.



TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW,
KM, ML, MR, NE, SN, TD, TG).

Published:

- *with international search report (Art. 21(3))*
- *in black and white; the international application as filed
contained color or greyscale and is available for download
from PATENTSCOPE*

METHOD AND SYSTEM FOR CODING METADATA IN AUDIO STREAMS AND FOR FLEXIBLE INTRA-OBJECT AND INTER-OBJECT BITRATE ADAPTATION

TECHNICAL FIELD

[0001] The present disclosure relates to sound coding, more specifically to a technique for digitally coding object-based audio, for example speech, music or general audio sound. In particular, the present disclosure relates to a system and method for coding and a system and method for decoding an object-based audio signal comprising audio objects in response to audio streams with associated metadata.

[0002] In the present disclosure and the appended claims:

[0003] (a) The term “object-based audio” is intended to represent a complex audio auditory scene as a collection of individual elements, also known as audio objects. Also, as indicated herein above, “object-based audio” may comprise, for example, speech, music or general audio sound.

[0004] (b) The term “audio object” is intended to designate an audio stream with associated metadata. For example, in the present disclosure, an “audio object” is referred to as an independent audio stream with metadata (*ISm*).

[0005] (c) The term “audio stream” is intended to represent, in a bit-stream, an audio waveform, for example speech, music or general audio sound, and may consist of one channel (mono) though two channels (stereo) might be also considered. “Mono” is the abbreviation of “monophonic” and “stereo” the abbreviation of “stereophonic.”

[0006] (d) The term “metadata” is intended to represent a set of information describing an audio stream and an artistic intension used to translate the original or

coded audio objects to a reproduction system. The metadata usually describes spatial properties of each individual audio object, such as position, orientation, volume, width, etc. In the context of the present disclosure, two sets of metadata are considered:

- input metadata: unquantized metadata representation used as an input to a codec; the present disclosure is not restricted a specific format of input metadata; and
- coded metadata: quantized and coded metadata forming part of a bit-stream transmitted from an encoder to a decoder.

[0007] (e) The term “audio format” is intended to designate an approach to achieve an immersive audio experience.

[0008] (f) The term “reproduction system” is intended to designate an element, in a decoder, capable of rendering audio objects, for example but not exclusively in a 3D (Three-Dimensional) audio space around a listener using the transmitted metadata and artistic intension at the reproduction side. The rendering can be performed to a target loudspeaker layout (e.g. 5.1 surround) or to headphones while the metadata can be dynamically modified, e.g. in response to a head-tracking device feedback. Other types of rendering may be contemplated.

BACKGROUND

[0009] In last years, the generation, recording, representation, coding, transmission, and reproduction of audio is moving towards enhanced, interactive and immersive experience for the listener. The immersive experience can be described e.g. as a state of being deeply engaged or involved in a sound scene while the sounds are coming from all directions. In immersive audio (also called 3D audio), the sound image is reproduced in all 3 dimensions around the listener taking into account a wide range of sound characteristics like timbre, directivity, reverberation, transparency and accuracy of (auditory) spaciousness. Immersive audio is produced for given reproduction systems, i.e. loudspeaker configurations, integrated

reproduction systems (sound bars) or headphones. Then interactivity of an audio reproduction system can include e.g. an ability to adjust sound levels, change positions of sounds, or select different languages for the reproduction.

[0010] There are three fundamental approaches (also referred below as audio formats) to achieve an immersive audio experience.

[0011] A first approach is a channel-based audio where multiple spaced microphones are used to capture sounds from different directions while one microphone corresponds to one audio channel in a specific loudspeaker layout. Each recorded channel is supplied to a loudspeaker in a particular location. Examples of channel-based audio comprise, for example, stereo, 5.1 surround, 5.1+4 etc.

[0012] A second approach is a scene-based audio which represents a desired sound field over a localized space as a function of time by a combination of dimensional components. The signals representing the scene-based audio are independent of the audio sources positions while the sound field has to be transformed to a chosen loudspeakers layout at the rendering reproduction system. An example of scene-based audio is ambisonics.

[0013] A third, last immersive audio approach is an object-based audio which represents an auditory scene as a set of individual audio elements (for example singer, drums, guitar) accompanied by information about, for example their position in the audio scene, so that they can be rendered at the reproduction system to their intended locations. This gives an object-based audio a great flexibility and interactivity because each object is kept discrete and can be individually manipulated.

[0014] Each of the above described audio formats has its pros and cons. It is thus common that not only one specific format is used in an audio system, but they might be combined in a complex audio system to create an immersive auditory scene. An example can be a system that combines a scene-based or channel-based audio with an object-based audio, e.g. ambisonics with few discrete audio objects.

[0015] The present disclosure presents in the following description a framework to encode and decode object-based audio. Such framework can be a standalone system for object-based audio format coding, or it could form part of a complex immersive codec that may contain coding of other audio formats and/or combination thereof.

SUMMARY

[0016] According to a first aspect, the present disclosure provides a system for coding an object-based audio signal comprising audio objects in response to audio streams with associated metadata, comprising an audio stream processor for analyzing the audio streams; a metadata processor responsive to information on the audio streams from the analysis by the audio stream processor for coding the metadata, wherein the metadata processor uses a logic for controlling a metadata coding bit-budget for coding the metadata, and an encoder for coding the audio streams.

[0017] The present disclosure also provides a method for coding an object-based audio signal comprising audio objects in response to audio streams with associated metadata, comprising: analyzing the audio streams; coding the metadata using (a) information on the audio streams from the analysis of the audio streams, and (b) a logic for controlling a metadata coding bit-budget; and encoding the audio streams.

[0018] According to a third aspect, there is provided an encoder device for coding a complex audio auditory scene comprising scene-based audio, multi-channels, and object-based audio signals, comprising the above defined system for coding the object-based audio signals.

[0019] The present disclosure further provides an encoding method for coding a complex audio auditory scene comprising scene-based audio, multi-

channels, and object-based audio signals, comprising the above mentioned method for coding the object-based audio signals.

[0020] The foregoing and other objects, advantages and features of the system and method for coding an object-based audio signal and the system and method for decoding an object-based audio signal will become more apparent upon reading of the following non-restrictive description of illustrative embodiments thereof, given by way of example only with reference to the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

[0021] In the appended drawings:

[0022] Figure 1 is a schematic block diagram illustrating concurrently the system for coding an object-based audio signal and the corresponding method for coding the object-based audio signal;

[0023] Figure 2 is a diagram showing different scenarios of bit-stream coding of one metadata parameter;

[0024] Figure 3a is a graph showing values of an absolute coding flag, $flag_{abs}$, for metadata parameters of three (3) audio objects without using an inter-object metadata coding logic, and Figure 3b is a graph showing values of the absolute coding flag, $flag_{abs}$, for the metadata parameters of the three (3) audio objects using the inter-object metadata coding logic, wherein arrows indicate frames where the value of several absolute coding flags equal to 1;

[0025] Figure 4 is a graph illustrating an example of bitrate adaptation for three (3) core-encoders;

[0026] Figure 5 is a graph illustrating an example of bitrate adaptation based on an *ISm* (Independent audio stream with metadata) importance logic;

[0027] Figure 6 is a schematic diagram illustrating the structure of a bit-stream

transmitted from the coding system of Figure 1 to the decoding system of Figure 7;

[0028] Figure 7 is a schematic block diagram illustrating concurrently the system for decoding audio objects in response to audio streams with associated metadata and the corresponding method for decoding the audio objects; and

[0029] Figure 8 is a simplified block diagram of an example configuration of hardware components implementing the system and method for coding an object-based audio signal and the system and method for decoding the object-based audio signal.

DETAILED DESCRIPTION

[0030] The present disclosure provides an example of mechanism for coding the metadata. The present disclosure also provides a mechanism for flexible intra-object and inter-object bitrate adaptation, i.e. a mechanism that distributes the available bitrate as efficiently as possible. In the present disclosure, it is further considered that the bitrate is fixed (constant). However, it is within the scope of the present disclosure to similarly consider an adaptive bitrate, for example (a) in an adaptive bitrate-based codec or (b) as a result of coding a combination of audio formats coded otherwise at a fixed total bitrate.

[0031] There is no description in the present disclosure as to how audio streams are actually coded in a so-called “core-encoder.” In general, the core-encoder for coding one audio stream can be an arbitrary mono codec using adaptive bitrate coding. An example is a codec based on the EVS codec as described in Reference [1] with a fluctuating bit-budget that is flexibly and efficiently distributed between modules of the core-encoder, for example as described in Reference [2]. The full contents of References [1] and [2] are incorporated herein by reference.

1. Framework for coding of audio objects

[0032] As a non-limitative example, the present disclosure considers a

framework that supports simultaneous coding of several audio objects (for example up to 16 audio objects) while a fixed constant *ISm* total bitrate, referred to as *ism_total_brake*, is considered for coding the audio objects, including the audio streams with their associated metadata. It should be noted that the metadata are not necessarily transmitted for at least some of the audio objects, for example in the case of non-diegetic content. Non-diegetic sounds in movies, TV shows and other videos are sound that the characters cannot hear. Soundtracks are an example of non-diegetic sound, since the audience members are the only ones to hear the music.

[0033] In the case of coding a combination of audio formats in the framework, for example an ambisonics audio format with two (2) audio objects, the constant total codec bitrate, referred to as *codec_total_brake*, then represents a sum of the ambisonics audio format bitrate (i. e. the bitrate to encode the ambisonics audio format) and the *ISm* total bitrate *ism_total_brake* (i.e. the sum of bitrates to code the audio objects, i.e. the audio streams with the associated metadata).

[0034] The present disclosure considers a basic non-limitative example of input metadata consisting of two parameters, namely azimuth and elevation, which are stored per audio frame for each object. In this example, an azimuth range of $[-180^{\circ}, 180^{\circ}]$, and an elevation range of $[-90^{\circ}, 90^{\circ}]$, is considered. However, it is within the scope of the present disclosure to consider only one or more than two (2) metadata parameters.

2. Object-based coding

[0035] Figure 1 is a schematic block diagram illustrating concurrently the system 100, comprising several processing blocks, for coding an object-based audio signal and the corresponding method 150 for coding the object-based audio signal.

2.1 Input buffering

[0036] Referring to Figure 1, the method 150 for coding the object-based audio signal comprises an operation of input buffering 151. To perform the operation 151 of

input buffering, the system 100 for coding the object-based audio signal comprises an input buffer 101.

[0037] The input buffer 101 buffers a number N of input audio objects 102, i.e. a number N of audio streams with the associated respective N metadata. The N input audio objects 102, including the N audio streams and the N metadata associated to each of these N audio streams are buffered for one frame, for example a 20 ms long frame. As well known in the art of sound signal processing, the sound signal is sampled at a given sampling frequency and processed by successive blocks of these samples called “frames” each divided into a number of “sub-frames.”

2.2 Audio streams analysis and front pre-processing

[0038] Still referring to Figure 1, the method 150 for coding the object-based audio signal comprises an operation of analysis and front pre-processing 153 of the N audio streams. To perform the operation 153, the system 100 for coding the object-based audio signal comprises an audio stream processor 103 to analyze and front pre-process, for example in parallel, the buffered N audio streams transmitted from the input buffer 101 to the audio stream processor 103 through a number N of transport channels 104, respectively.

[0039] The analysis and front pre-processing operation 153 performed by the audio stream processor 103 may comprise, for example, at least one of the following sub-operations: time-domain transient detection, spectral analysis, long-term prediction analysis, pitch tracking and voicing analysis, voice/sound activity detection (VAD/SAD), bandwidth detection, noise estimation and signal classification (which may include in a non-limitative embodiment (a) core-encoder selection between, for example, ACELP core-encoder, TCX core-encoder, HQ core-encoder, etc., (b) signal type classification between, for example, inactive core-encoder type, unvoiced core-encoder type, voiced core-encoder type, generic core-encoder type, transition core-encoder type, and audio core-encoder type, etc., (c) speech/music classification, etc.). Information obtained from the analysis and front pre-processing operation 153 is

supplied to a configuration and decision processor 106 through line 121. Examples of the foregoing sub-operations are described in Reference [1] in relation to the EVS codec and, therefore, will not be further described in the present disclosure.

2.3 Metadata analysis, quantization and coding

[0040] The method 150 of Figure 1, for coding the object-based audio signal comprises an operation of metadata analysis, quantization and coding 155. To perform the operation 155, the system 100 for coding the object-based audio signal comprises a metadata processor 105.

2.3.1 Metadata analysis

[0041] Signal classification information 120 (for example VAD or *localVAD* flag as used in the EVS codec (See Reference [1]) from the audio stream processor 103 is supplied to the metadata processor 105. The metadata processor 105 comprises an analyzer (not shown) of the metadata of each of the N audio objects to determine whether the current frame is inactive (for example $VAD = 0$) or active (for example $VAD \neq 0$) with respect to this particular audio object. In inactive frames, no metadata is coded by the metadata processor 105 relative of that object. In active frames, the metadata are quantized and coded for this audio object using a variable bitrate. More details about metadata quantization and coding will be provided in the following Sections 2.3.2 and 2.3.3.

2.3.2 Metadata quantization

[0042] The metadata processor 105 of Figure 1 quantizes and codes the metadata of the N audio objects, in the described non-restrictive illustrative embodiments, sequentially in a loop while a certain dependency can be employed between quantization of audio objects and the metadata parameters of these audio objects.

[0043] As indicated herein above, in the present disclosure, two metadata

parameters, azimuth and elevation (as included in the N input metadata), are considered. As a non-limitative example, the metadata processor 105 comprises a quantizer (not shown) of the following metadata parameter indexes using the following example resolution to reduce the number of bits being used:

- Azimuth parameter: A 12-bit azimuth parameter index from a file of the input metadata is quantized to B_{az} -bit index (for example $B_{az} = 7$). Giving the minimum and maximum azimuth limits (-180° and $+180^\circ$), a quantization step for a ($B_{az} = 7$)-bit uniform scalar quantizer is 2.835° .
- Elevation parameter: A 12-bit elevation parameter index from the input metadata file is quantized to B_{el} -bit index (for example $B_{el} = 6$). Giving the minimum and maximum elevation limits (-90° and $+90^\circ$), a quantization step for a ($B_{el} = 6$)-bit uniform scalar quantizer is 2.857° .

[0044] A total metadata bit-budget for coding the N metadata and a total number quantization bits for quantizing the metadata parameter indexes (i.e. the quantization index granularity and thus the resolution) may be made dependent on the bitrate(s) *codec_total_brate*, *ism_total_brate* and/or *element_brate* (the latter resulting from a sum of a metadata bit-budget and/or a core-encoder bit-budget related to one audio object).

[0045] The azimuth and elevation parameters can be represented as one parameter, for example by a point on a sphere. In such a case, it is within the scope of the present disclosure to implement different metadata including two or more parameters.

2.3.3 Metadata coding

[0046] Both azimuth and elevation indexes, once quantized, can be coded by a metadata encoder (not shown) of the metadata processor 105 using either absolute or differential coding. As known, absolute coding means that a current value of a parameter is coded. Differential coding means that a difference between a current

value and a previous value of a parameter is coded. As the indexes of the azimuth and elevation parameters usually evolve smoothly (i.e. a change in azimuth or elevation position can be considered as continuous and smooth), differential coding is used by default. However, absolute coding may be used, for example in the following instances:

- There is too large a difference between current and previous values of the parameter index which would result in a higher or equal number of bits for using differential coding compared to using absolute coding (may happen exceptionally);
- No metadata were coded and sent in the previous frame;
- There were too many consecutive frames with differential coding. In order to control decoding in a noisy channel (Bad Frame Indicator, BFI = 1). For example, the metadata encoder codes the metadata parameter indexes using absolute coding if a number of consecutive frames which are coded using differential is higher than a maximum number of consecutive frames coded using differential coding. The latter maximum number of consecutive frames is set to β . In a non-restrictive illustrative example, $\beta = 10$ frames.

[0047] The metadata encoder produces a 1-bit absolute coding flag, $flag_{abs}$, to distinguish between absolute and differential coding.

[0048] In the case of absolute coding, the coding flag, $flag_{abs}$, is set to 1, and is followed by the B_{az} -bit (or B_{el} -bit) index coded using absolute coding, where B_{az} and B_{el} refer to the above mentioned indexes of the azimuth and elevation parameters to be coded, respectively.

[0049] In the case of differential coding, the 1-bit coding flag, $flag_{abs}$, is set to 0 and is followed by a 1-bit zero coding flag, $flag_{zero}$, signaling a difference Δ between the B_{az} -bit indexes (respectively the B_{el} -bit indices) in the current and previous frames equal to 0. If the difference Δ is not equal to 0, the metadata encoder continues coding

by producing a 1-bit sign flag, $flag_{sign}$, followed by a difference index, of which the number of bits is adaptive, in a form of, for example, a unary code indicative of the value of the difference Δ .

[0050] Figure 2 is a diagram showing different scenarios of bit-stream coding of one metadata parameter.

[0051] Referring to Figure 2, it is noted that not all metadata parameters are always transmitted in every frame. Some might be transmitted only in every y^{th} frame, some are not sent at all for example when they do not evolve, they are not important or the available bit-budget is low. Referring to Figure 2, for example:

[0052] - in the case of absolute coding (first line of Figure 2), the absolute coding flag, $flag_{abs}$, and the B_{az} -bit index (respectively the B_{er} -bit index) are transmitted;

[0053] - in the case of differential coding with the difference Δ between the B_{az} -bit indexes (respectively the B_{er} -bit indexes) in the current and previous frames equal to 0 (second line of Figure 2), the absolute coding flag, $flag_{abs}=0$, and the zero coding flag, $flag_{zero}=1$ are transmitted;

[0054] - in the case of differential coding with a positive difference Δ between the B_{az} -bit index (respectively the B_{er} -bit indexes) in the current and previous frames (third line of Figure 2), the absolute coding flag, $flag_{abs}=0$, the zero coding flag, $flag_{zero}=0$, the sign flag, $flag_{sign}=0$, and the difference index (1 to $(B_{az}-3)$ -bits index (respectively 1 to $(B_{er}-3)$ -bits index)) are transmitted; and

[0055] - in the case of differential coding with a negative difference Δ between the B_{az} -bit indexes (respectively the B_{er} -bit indexes) in the current and previous frames (last line of Figure 2), the absolute coding flag, $flag_{abs}=0$, the zero coding flag, $flag_{zero}=0$, the sign flag, $flag_{sign}=1$, and the difference index (1 to $(B_{az}-3)$ -bits index (respectively 1 to $(B_{er}-3)$ -bits index)) are transmitted.

2.3.3.1 Intra-object metadata coding logic

[0056] The logic used to set absolute or differential coding may be further extended by an intra-object metadata coding logic. Specifically, in order to limit a range of metadata coding bit-budget fluctuation between frames and thus to avoid too low a bit-budget left for the core-encoders 109, the metadata encoder limits absolute coding in a given frame to one, or generally to a number as low as possible of, metadata parameters.

[0057] In the non-limitative example of azimuth and elevation metadata parameter coding, the metadata encoder uses a logic that avoids absolute coding of the elevation index in a given frame if the azimuth index was already coded using absolute coding in the same frame. In other words, the azimuth and elevation parameters of one audio object are (practically) never both coded using absolute coding in a same frame. As a consequence, the absolute coding flag, $flag_{abs.ele}$, for the elevation parameter is not transmitted in the audio object bit-stream if the absolute coding flag, $flag_{abs.azi}$, for the azimuth parameter is equal to 1.

[0058] It is also within the scope of the present disclosure to make the intra-object metadata coding logic bitrate dependent. For example, both the absolute coding flag, $flag_{abs.ele}$, for the elevation parameter and the absolute coding flag, $flag_{abs.azi}$, for the azimuth parameter can be transmitted in a same frame if the bitrate is sufficiently large.

2.3.3.2 Inter-object metadata coding logic

[0059] The metadata encoder may apply a similar logic to metadata coding of different audio objects. The implemented inter-object metadata coding logic minimizes the number of metadata parameters of different audio objects coded using absolute coding in a current frame. This is achieved by the metadata encoder mainly by controlling frame counters of metadata parameters coded using absolute coding chosen from robustness purposes and represented by the parameter β . As a non-

limitative example, a scenario where the metadata parameters of the audio objects evolve slowly and smoothly is considered. In order to control decoding in a noisy channel where indexes are coded using absolute coding every β frames, the azimuth B_{az} -bit index of audio object #1 is coded using absolute coding in frame M , the elevation B_{el} -bit index of audio object #1 is coded using absolute coding in frame $M+1$, the azimuth B_{az} -bit index of audio object #2 is encoded using absolute coding in frame $M+2$, the elevation B_{el} -bit index of object #2 is coded using absolute coding in frame $M+3$, etc.

[0060] Figure 3a is a graph showing values of the absolute coding flag, $flag_{abs}$, for metadata parameters of three (3) audio objects without using the inter-object metadata coding logic, and Figure 3b is a graph showing values of the absolute coding flag, $flag_{abs}$, for the metadata parameters of the three (3) audio objects using the inter-object metadata coding logic. In Figure 3a, the arrows indicate frames where the value of several absolute coding flags is equal to 1.

[0061] More specifically, Figure 3a shows the values of the absolute coding flag, $flag_{abs}$, for two metadata parameters (azimuth and elevation in this particular example) for the audio objects without using the inter-object metadata coding logic, while Figure 3b shows the same values but with the inter-object metadata coding logic implemented. The graphs of Figures 3a and 3b correspond to (from top to bottom):

- audio stream of audio object #1;
- audio stream of audio object #2;
- audio stream of audio object #3,
- absolute coding flag, $flag_{abs,azi}$, for the azimuth parameter of audio object #1;
- absolute coding flag, $flag_{abs,ele}$, for the elevation parameter of audio object #1;

- absolute coding flag, $flag_{abs,azi}$, for the azimuth parameter of audio object #2;
- absolute coding flag, $flag_{abs,elev}$, for the elevation parameter of audio object #2;
- absolute coding flag, $flag_{abs,azi}$, for the azimuth parameter of audio object #3;
and
- absolute coding flag, $flag_{abs,elev}$, for the elevation parameter of audio object #3.

[0062] It can be seen from Figure 3a that several $flag_{abs}$ may have a value equal to 1 (see the arrows) in a same frame when the inter-object metadata coding logic is not used. In contrast, Figure 3b shows that only one absolute flag, $flag_{abs}$, may have a value equal to 1 in a given frame when the inter-object metadata coding logic is used.

[0063] The inter-object metadata coding logic may also be made bitrate dependent. In this case, for example, more than one absolute flag, $flag_{abs}$, may have a value equal to 1 in a given frame even when the inter-object metadata coding logic is used, if the bitrate is sufficiently large.

[0064] A technical advantage of the inter-object metadata coding logic and the intra-object metadata coding logic is to limit a range of fluctuation of the metadata coding bit-budget between frames. Another technical advantage is to increase robustness of the codec in a noisy channel; when a frame is lost, then only a limited number of metadata parameters from the audio objects coded using absolute coding is lost. Consequently, any error propagated from a lost frame affects only a small number of metadata parameters across the audio objects and thus does not affect the whole audio scene (or several different channels).

[0065] A global technical advantage of analyzing, quantizing and coding the metadata separately from the audio streams is, as described hereinabove, to enable processing specially adapted to the metadata and more efficient in terms of metadata coding bitrate, metadata coding bit-budget fluctuation, robustness in noisy channel,

and error propagation due to lost frames.

[0066] The quantized and coded metadata 112 from the metadata processor 105 are supplied to a multiplexer 110 for insertion into an output bit-stream 111 transmitted to a distant decoder 700 (Figure 7).

[0067] Once the metadata of the N audio objects are analyzed, quantized and encoded, information 107 from the metadata processor 105 about the bit-budget for the coding of the metadata per audio object is supplied to a configuration and decision processor 106 (bit-budget allocator) described in more detail in the following section 2.4. When the configuration and bitrate distribution between the audio streams is completed in processor 106 (bit-budget allocator), the coding continues with further pre-processing 158 to be described later. Finally, the N audio streams are encoded using an encoder comprising, for example, N fluctuating bitrate core-encoders 109, such as mono core-encoders.

2.4 Bitrates per channel configuration and decision

[0068] The method 150 of Figure 1, for coding the object-based audio signal comprises an operation 156 of configuration and decision about bitrates per transport channel 104. To perform the operation 156, the system 100 for coding the object-based audio signal comprises the configuration and decision processor 106 forming a bit-budget allocator.

[0069] The configuration and decision processor 106 (herein after bit-budget allocator 106) uses a bitrate adaptation algorithm to distribute the available bit-budget for core-encoding the N audio streams in the N transport channels 104.

[0070] The bitrate adaptation algorithm of the configuration and decision operation 156 comprises the following sub-operations 1-6 performed by the bit-budget allocator 106:

[0071] 1. The ISm total bit-budget, $bits_{ism}$, per frame is calculated from the ISm

total bitrate ism_total_brate (or the codec total bitrate $codec_total_brate$ if only audio objects are coded) using, for example, the following relation:

$$bits_{ism} = \frac{ism_total_brate}{50}$$

The denominator, 50, corresponds to the number of frames per second, assuming 20-ms long frames. The value 50 would be different if the size of the frame is different from 20 ms.

[0072] 2. The above defined element bitrate $element_brate$ (resulting from a sum of the metadata bit-budget and core-encoder bit-budget related to one audio object) defined for N audio objects is supposed to be constant during a session at a given codec total bitrate, and about the same for the N audio objects. A “session” is defined for example as a phone call or an off-line compression of an audio file. The corresponding element bit-budget, $bits_{element}$, is computed for the audio streams objects $n = 0, \dots, N-1$ using, for example, the following relation:

$$bits_{element}[n] = \left\lfloor \frac{bits_{ism}}{N} \right\rfloor$$

where $\lfloor x \rfloor$ indicates the largest integer smaller than or equal to x . In order to spend all available ISM total bit-budget $bits_{ism}$ the element bit-budget $bits_{element}$ of, for example, the last audio object is eventually adjusted using the following relation:

$$bits_{element}[N-1] = \left\lfloor \frac{bits_{ism}}{N} \right\rfloor + bits_{ism} \bmod N$$

where “mod” indicates a remainder modulo operation. Finally, the element bit-budget $bits_{element}$ of the N audio objects is used to set the value $element_brate$ for the audio objects $n = 0, \dots, N-1$ using, for example, the following relation:

$$element_brate[n] = bits_{element}[n] * 50$$

where the number 50, as already mentioned, corresponds to the number of frames per second, assuming 20-ms long frames.

[0073] 3. The metadata bit-budget $bits_{meta}$, per frame, of the N audio objects is

summed, using the following relation:

$$bits_{meta_all} = \sum_{n=0}^{N-1} bits_{meta}[n]$$

and the resulting value $bits_{meta_all}$ is added to an ISm common signaling bit-budget, $bits_{ISm_signalling}$, resulting in the codec side bit-budget:

$$bits_{side} = bits_{meta_all} + bits_{ISm_signalling}$$

[0074] 4. The codec side bit-budget, $bits_{side}$, per frame, is split equally between the N audio objects and used to compute the core-encoder bit-budget, $bits_{CoreCoder}$, for each of the N audio streams using, for example, the following relation:

$$bits_{CoreCoder}[n] = bits_{element}[n] - \left\lfloor \frac{bits_{side}}{N} \right\rfloor$$

while the core-encoder bit-budget of, for example, the last audio stream may eventually be adjusted to spend all the available core-encoding bit-budget using, for example, the following relation:

$$bits_{CoreCoder}[N-1] = bits_{element}[N-1] - \left\lfloor \frac{bits_{side}}{N} \right\rfloor + bits_{side} \bmod N$$

The corresponding total bitrate, $total_brate$, i.e. the bitrate to code one audio stream, in a core-encoder, is then obtained for $n = 0, \dots, N-1$ using, for example, the following relation:

$$total_brate[n] = bits_{CoreCoder}[n] * 50$$

where the number 50, again, corresponds to the number of frames per second, assuming 20-ms long frames.

[0075] 5. The total bitrate, $total_brate$, in inactive frames (or in frames with very low energy or otherwise without meaningful content) may be lowered and set to a constant value in the related audio streams. The so saved bit-budget is then redistributed equally between the audio streams with active content in the frame. Such redistribution of bit-budget will be further described in the following section 2.4.1.

[0076] 6. The total bitrate, *total_brate*, in audio streams (with active content) in active frames is further adjusted between these audio streams based on an *ISm* importance classification. Such adjustment of bitrate will be further described in the following section 2.4.2.

[0077] When the audio streams are all in an inactive segment (or are without meaningful content), the above last two sub-operations 5 and 6 may be skipped. Accordingly, the bitrate adaptation algorithms described in following sections 2.4.1 and 2.4.2 are employed when at least one audio stream has active content.

2.4.1 Bitrate adaptation based on signal activity

[0078] In inactive frames ($VAD = 0$), the total bitrate, *total_brate*, is lowered and the saved bit-budget is redistributed, for example equally between the audio streams in active frames ($VAD \neq 0$). The assumption is that waveform coding of an audio stream in frames which are classified as inactive is not required; the audio object may be muted. The logic, used in every frame, can be expressed by the following sub-operations 1-3:

[0079] 1. For a particular frame, set a lower core-encoder bit-budget to every audio stream n with inactive content:

$$bits_{CoreCoder}'[n] = B_{VAD0} \quad \forall n \text{ with } VAD=0$$

where B_{VAD0} is a lower, constant core-encoder bit-budget to be set in inactive frames; for example $B_{VAD0} = 140$ (corresponding to 7 kbps for a 20 ms frame) or $B_{VAD0} = 49$ (corresponding to 2.45 kbps for a 20 ms frame).

[0080] 2. Next, the saved bit-budget is computed using, for example, the following relation:

$$bits_{diff} = \sum_{n=0}^{N-1} (bits_{CoreCoder}'[n] - bits_{CoreCoder}[n])$$

[0081] 3. Finally, the saved bit-budget is redistributed, for example equally

between the core-encoder bit-budgets of the audio streams with active content in a given frame using the following relation:

$$bits_{CoreCoder}'[n] = bits_{CoreCoder}[n] + \left\lfloor \frac{bits_{diff}}{N_{VAD1}} \right\rfloor \quad \forall n \text{ with VAD}=1$$

where N_{VAD1} is the number of audio streams with active content. The core-encoder bit-budget of the first audio stream with active content is eventually increased using, for example, the following relation:

$$bits_{CoreCoder}'[n] = bits_{CoreCoder}[n] + \left\lfloor \frac{bits_{diff}}{N_{VAD1}} \right\rfloor + bits_{diff} \bmod N_{VAD1}, \quad \forall n \text{ A first VAD}=1 \text{ stream}$$

The corresponding core-encoder total bitrate, *total_brate*, is finally obtained for each audio stream $n = 0, \dots, N-1$ as follows:

$$total_brate'[n] = bits_{CoreCoder}'[n] * 50$$

[0082] Figure 4 is a graph illustrating an example of bitrate adaptation for three (3) core-encoders. Specifically, In Figure 4, the first line shows the core-encoder total bitrate, *total_brate*, for audio stream #1, the second line shows the core-encoder total bitrate, *total_brate*, for audio stream #2, the third line shows the core-encoder total bitrate, *total_brate*, for audio stream #3, line 4 is the audio stream #1, line 5 is the audio stream #2, and line 4 is the audio stream #3.

[0083] In the example of Figure 4, the adaptation of the total bitrate, *total_brate*, for the three (3) core-encoder is based on VAD activity (active/inactive frames). As can be seen from Figure 4, most of the time there is a small fluctuation of the core-encoder total bitrate, *total_brate*, as a result of the fluctuating side bit-budget $bits_{side}$. Then, there are infrequent substantial changes of the core-encoder total bitrate, *total_brate*, as a result of the VAD activity.

[0084] For example, referring to Figure 4, instance A) corresponds to a frame where the audio stream #1 VAD activity changes from 1 (active) to 0 (inactive). According to the logic, a minimum core-encoder total bitrate, *total_brate*, is assigned

to audio object #1 while the core-encoder total bitrates, *total_brate*, for active audio objects #2 and #3 are increased. Instance B) corresponds to a frame where the VAD activity of the audio stream #3 changes from 1 (active) to 0 (inactive) while the VAD activity of the audio stream #1 remains to 0. Accordingly to the logic, a minimum core-encoder total bitrate, *total_brate*, is assigned to audio streams #1 and #3 while the core-encoder total bitrate, *total_brate*, of the active audio stream #2 is further increased.

[0085] The above logic of section 2.4.1 can be made dependent from the total bitrate *ism_total_brate*. For example, the bit-budget B_{VAD0} in the above sub-operation 1 can be set higher for a higher total bitrate *ism_total_brate*, and lower for a lower total bitrate *ism_total_brate*.

2.4.2 Bitrate adaptation based on *ISm* importance

[0086] The logic described in previous section 2.4.1 results in about a same core-encoder bitrate in every audio stream with active content ($VAD = 1$) in a given frame. However, it may be beneficial to introduce an inter-object core-encoder bitrate adaptation based on a classification of *ISm* importance (or, more generally, on a metric indicative of how critical coding of a particular audio object in a current frame to obtain a given (decent) quality of the decoded synthesis is).

[0087] The classification of *ISm* importance can be based on several parameters and/or combination of parameters, for example core-encoder type (*coder_type*), FEC (Forward Error Correction), sound signal classification (*class*), speech/music classification decision, and/or SNR (Signal-to-Noise Ratio) estimate from the open-loop ACELP/TCX (Algebraic Code-Excited Linear Prediction/Transform-Coded eXcitation) core decision module (*snr_celp*, *snr_tcx*) as described in Reference [1]. Other parameters can possibly be used for determining the classification of *ISm* importance.

[0088] In a non-restrictive example, a simple classification of *ISm* importance is

based on the core-encoder type as defined in Reference [1] is implemented. For that purpose, the bit-budget allocator 106 of Figure 1 comprises a classifier (not shown) for rating the importance of a particular *ISm* stream. As a result, four (4) distinct *ISm* importance classes, $class_{ISm}$, are defined:

- No metadata class, *ISM_NO_META*: frames without metadata coding, e.g. inactive frames with $VAD = 0$;
- Low importance class, *ISM_LOW_IMP*: frames where $coder_type = UNVOICED$ or *INACTIVE*;
- Medium importance class, *ISM_MEDIUM_IMP*: frames where $coder_type = VOICED$;
- High importance class *ISM_HIGH_IMP*: frames where $coder_type = GENERIC$.

[0089] The *ISm* importance class is then used by the bit-budget allocator 106, in the bitrate adaptation algorithm (See above Section 2.4, sub-operation 6) to assign a higher bit-budget to audio streams with a higher *ISm* importance and a lower bit-budget to audio streams with a lower *ISm* importance. Thus for every audio stream n , $n = 0, \dots, N-1$, the following bitrate adaptation algorithm is used by the bit-budget allocator 106:

1. In frames classified as $class_{ISm} = ISM_NO_META$, the constant low bitrate B_{VAD0} is assigned.
2. In frames classified as $class_{ISm} = ISM_LOW_IMP$, the total bitrate, $total_brate$, is lowered for example as:

$$total_brate_{new}[n] = \max(\alpha_{low} * total_brate[n], B_{low})$$

where the constant α_{low} is set to a value lower than 1.0, for example 0.6. Then

the constant B_{low} represents a minimum bitrate threshold supported by the codec for a particular configuration, which may be dependent upon, for example, the internal sampling rate of the codec, the coded audio bandwidth, etc. (See Reference [1] for more detail about these values).

3. In frames classified as $class_{ISM} = ISM_MEDIUM_IMP$: the core-encoder total bitrate, $total_brate$, is lowered for example as

$$total_brate_{new}[n] = \max(\alpha_{med} * total_brate[n], B_{low})$$

where the constant α_{med} is set to a value lower than 1.0 but higher than α_{low} , for example to 0.8.

4. In frames classified as $class_{ISM} = ISM_HIGH_IMP$, no bitrate adaptation is used;
5. Finally, the saved bit-budget (a sum of differences between the old ($total_brate$) and new ($total_brate_{new}$) total bitrates) is redistributed equally between the audio streams with active content in the frame. The same bit-budget redistribution logic as described in section 2.4.1, sub-operations 2 and 3, may be used.

[0090] Figure 5 is a graph illustrating an example of bitrate adaptation based on ISM importance logic. From top to bottom, the graph of Figure 5 illustrates, in time:

- An active speech segment of the audio stream for audio object #1;
- An active speech segment of the audio stream for audio object #2;
- The total bitrate, $total_brate$, of the audio stream for audio object #1 without using the bitrate adaptation algorithm;

- The total bitrate, *total_brake*, of the audio stream for audio object #2 without using the bitrate adaptation algorithm;
- The total bitrate, *total_brake*, of the audio stream for audio object #1 when the bitrate adaptation algorithm is used; and
- The total bitrate, *total_brake*, of the audio stream for audio object #2 when the bitrate adaptation algorithm is used.

[0091] In the non-limitative example of Figure 5, with two audio objects ($N=2$) and a fixed constant total bitrate, *ism_total_brake*, equal to 48 kbps, the core-encoder total bitrate, *total_brake*, in active frames of audio object #1 fluctuates between 23.45 kbps and 23.65 kbps when the bitrate adaptation algorithm is not used while it fluctuates between 19.15 kbps and 28.05 kbps when the bitrate adaptation algorithm is used. Similarly, the core-encoder total bitrate, *total_brake*, in active frames of audio object #2 fluctuates between 23.40 kbps and 23.65 kbps without using the bitrate adaptation algorithm and between 19.10 kbps and 28.05 kbps with the bitrate adaptation algorithm. A better, more efficient distribution of the available bit-budget between the audio streams is thereby obtained.

2.5 Pre-processing

[0092] Referring to Figure 1, the method 150 for coding the object-based audio signal comprises an operation of pre-processing 158 of the N audio streams conveyed through the N transport channels 104 from the configuration and decision processor 106 (bit-budget allocator). To perform the operation 158, the system 100 for coding the object-based audio signal comprises a pre-processor 108.

[0093] Once the configuration and bitrate distribution between the N audio streams is completed by the configuration and decision processor 106 (bit-budget allocator), the pre-processor 108 performs sequential further pre-processing 158 on

each of the N audio streams. Such pre-processing 158 may comprise, for example, further signal classification, further core-encoder selection (for example selection between ACELP core, TCX core, and HQ core), other resampling at a different internal sampling frequency F_s adapted to the bitrate to be used for core-encoding, etc. Examples of such pre-processing can be found, for example, in Reference [1] in relation to the EVS codec and, therefore, will not be further described in the present disclosure.

2.6 Core-encoding

[0094] Referring to Figure 1, the method 150 for coding the object-based audio signal comprises an operation of core-encoding 159. To perform the operation 159, the system 100 for coding the object-based audio signal comprises the above mentioned encoder of the N audio streams including, for example, a number N of core-encoders 109 to respectively code the N audio streams conveyed through the N transport channels 104 from the pre-processor 108.

[0095] Specifically, the N audio streams are encoded using N fluctuating bitrate core-encoders 109, for example mono core-encoders. The bitrate used by each of the N core-encoders is the bitrate selected by the configuration and decision processor 106 (bit-budget allocator) for the corresponding audio stream. For example, core-encoders as described in Reference [1] can be used as core-encoders 109.

3.0 Bit-stream structure

[0096] Referring to Figure 1, the method 150 for coding the object-based audio signal comprises an operation of multiplexing 160. To perform the operation 160, the system 100 for coding the object-based audio signal comprises a multiplexer 110.

[0097] Figure 6 is a schematic diagram illustrating, for a frame, the structure of

the bit-stream 111 produced by the multiplexer 110 and transmitted from the coding system 100 of Figure 1 to the decoding system 700 of Figure 7. Regardless whether metadata are present and transmitted or not, the structure of the bit-stream 111 may be structured as illustrated in Figure 6.

[0098] Referring to Figure 6, the multiplexer 110 writes the indices of the N audio streams from the beginning of the bit-stream 111 while the indices of ISm common signaling 113 from the configuration and decision processor 106 (bit-budget allocator) and metadata 112 from the metadata processor 105 are written from the end of the bit-stream 111.

3.1 ISm common signaling

[0099] The multiplexer writes the ISm common signaling 113 from the end of the bit-stream 111. The ISm common signaling is produced by the configuration and decision processor 106 (bit-budget allocator) and comprises a variable number of bits representing:

[00100] (a) a number N of audio objects: the signaling for the number N of coded audio objects present in the bit-stream 111 is in the form of, for example, a unary code with a stop bit (e.g. for $N = 3$ audio objects, the first 3 bits of the ISm common signaling would be “110”).

[00101] (b) a metadata presence flag, $flag_{meta}$: The flag, $flag_{meta}$, is present when the bitrate adaptation based on signal activity as described in section 2.4.1 is used and comprises one bit per audio object to indicate whether metadata for that particular audio object are present ($flag_{meta} = 1$) or not ($flag_{meta} = 0$) in the bit-stream 111, or (c) the ISm importance class: this signaling is present when the bitrate adaptation based on the ISM importance as described in section 2.4.2 is used and comprises two bits per audio object to indicate the ISm importance class, $class_{ISm}$ (ISM_NO_META , ISM_LOW_IMP , ISM_MEDIUM_IMP , and ISM_HIGH_IMP), as defined in section 2.4.2.

[00102] (d) an *ISm* VAD flag, $flag_{VAD}$: the *ISm* VAD flag is transmitted when $flag_{meta} = 0$, respectively $class_{ISm} = ISM_NO_META$, and distinguishes between the following two cases:

- 1) input metadata are not present or metadata are not coded so that the audio stream needs to be coded by an active coding mode ($flag_{VAD} = 1$); and
- 2) input metadata are present and transmitted so that the audio stream can be coded by an inactive coding mode ($flag_{VAD} = 0$).

3.2 Coded metadata payload

[00103] The multiplexer 110 is supplied with the coded metadata 112 from the metadata processor 105 and writes the metadata payload sequentially from the end of the bit-stream for the audio objects for which the metadata are coded ($flag_{meta} = 1$, respectively $class_{ISm} \neq ISM_NO_META$) in the current frame. The metadata bit-budget for each audio object is not constant but rather inter-object and inter-frame adaptive. Different metadata format scenarios are shown in Figure 2.

[00104] In the case that metadata are not present or are not transmitted for at least some of the N audio objects, the metadata flag is set to 0, i.e. $flag_{meta} = 0$, respectively $class_{ISm} = ISM_NO_META$, for these audio objects. Then, no metadata indices are sent in relation to those audio objects, i.e. $bits_{meta}[n] = 0$.

3.3 Audio streams payload

[00105] The multiplexer 110 receives the N audio streams 114 coded by the N core encoders 109 through the N transport channels 104, and writes the audio streams payload sequentially for the N audio streams in chronological order from the beginning of the bit-stream 111 (See Figure 6). The respective bit-budgets of the N audio streams are fluctuating as a result of the bitrate adaptation algorithm described in section 2.4.

4.0 Decoding of audio objects

[00106] Figure 7 is a schematic block diagram illustrating concurrently the system 700 for decoding audio objects in response to audio streams with associated metadata and the corresponding method 750 for decoding the audio objects.

4.1 Demultiplexing

[00107] Referring to Figure 7, the method 750 for decoding audio objects in response to audio streams with associated metadata comprises an operation of demultiplexing 755. To perform the operation 755, the system 700 for decoding audio objects in response to audio streams with associated metadata comprises a demultiplexer 705.

[00108] The demultiplexer receive a bit-stream 701 transmitted from the coding system 100 of Figure 1 to the decoding system 700 of Figure 7. Specifically, the bit-stream 701 of Figure 7 corresponds to the bit-stream 111 of Figure 1.

[00109] The demultiplexer 110 extracts from the bit-stream 701 (a) the coded N audio streams 114, (b) the coded metadata 112 for the N audio objects, and (c) the ISm common signaling 113 read from the end of the received bit-stream 701.

4.2 Metadata decoding and dequantization

[00110] Referring to Figure 7, the method 750 for decoding audio objects in response to audio streams with associated metadata comprises an operation 756 of metadata decoding and dequantization. To perform the operation 756, the system 700 for decoding audio objects in response to audio streams with associated metadata comprises a metadata decoding and dequantization processor 706.

[00111] The metadata decoding and dequantization processor 706 is supplied with the coded metadata 112 for the transmitted audio objects, the ISm common signaling 113, and an output set-up 709 to decode and dequantize the

metadata for the audio streams/objects with active contents. The output set-up 709 is a command line parameter about the number M of decoded audio objects/transport channels and/or audio formats, which can be equal to or different from the number N of coded audio objects/transport channels. The metadata decoding and de-quantization processor 706 produces decoded metadata 704 for the M audio objects/transport channels, and supplies information about the respective bit-budgets for the M decoded metadata on line 708. Obviously, the decoding and dequantization performed by the processor 706 is the inverse of the quantization and coding performed by the metadata processor 105 of Figure 1.

4.3 Configuration and decision about bitrates

[00112] Referring to Figure 7, the method 750 for decoding audio objects in response to audio streams with associated metadata comprises an operation 757 of configuration and decision about bitrates per channel. To perform the operation 757, the system 700 for decoding audio objects in response to audio streams with associated metadata comprises a configuration and decision processor 707 (bit-budget allocator).

[00113] The bit-budget allocator 707 receives (a) the information about the respective bit-budgets for the M decoded metadata on line 708 and (b) the ISm importance class, $class_{ISm}$, from the common signaling 113, and determines the core-decoder bitrates per audio stream, $total_brate[n]$. The bit-budget allocator 707 uses the same procedure as in the bit-budget allocator 106 of Figure 1 to determine the core-decoder bitrates (see section 2.4).

4.4 Core-decoding

[00114] Referring to Figure 7, the method 750 for decoding audio objects in response to audio streams with associated metadata comprises an operation of core-decoding 760. To perform the operation 760, the system 700 for decoding audio objects in response to audio streams with associated metadata comprises a

decoder of the N audio streams 114 including a number N of core-decoders 710, for example N fluctuating bitrate core-decoders.

[00115] The N audio streams 114 from the demultiplexer 705 are decoded, for example sequentially decoded in the number N of fluctuating bitrate core decoders 710 at their respective core-decoder bitrates as determined by the bit-budget allocator 707. When the number of decoded audio objects, M , as requested by the output set-up 709 is lower than the number of transport channels, i.e. $M < N$, a lower number of core-decoders are used. Similarly, not all metadata payloads may be decoded in such a case.

[00116] In response to the N audio streams 114 from the demultiplexer 705, the core-decoder bitrates as determined by the bit-budget allocator 707, and the output set-up 709, the core-decoders 710 produces a number M of decoded audio streams 703 on respective M transport channels.

5.0 Audio channel rendering

[00117] In an operation of audio channel rendering 761, a renderer 711 of audio objects transforms the M decoded metadata 704 and the M decoded audio streams 703 into a number of output audio channels 702, taking into consideration an output set-up 712 indicative of the number and contents of output audio channels to be produced. Again, the number of output audio channels 702 may be equal to or different from the number M .

[00118] The renderer 761 may be designed in a variety of different structures to obtain the desired output audio channels. For that reason, the renderer will not be further described in the present disclosure.

6.0 Source code

[00119] According to a non-limitative illustrative embodiment, the system and method for coding an object-based audio signal as disclosed in the foregoing

description may be implemented by the following source code (expressed in C-code) given herein below as additional disclosure.

[00120]

```

void ism_metadata_enc(
    const long ism_total_brat,          /* i : ISMs total bitrate          */
    const short n_ISMs,                 /* i : number of objects           */
    ISM_METADATA_HANDLE hIsmMeta[],    /* i/o: ISM metadata handles       */
    ENC_HANDLE hSCE[],                 /* i/o: element encoder handles    */
    BSTR_ENC_HANDLE hBstr,              /* i/o: bitstream handle           */
    short nb_bits_metadata[],           /* o : number of metadata bits     */
    short localVAD[]
)
{
    short i, ch, nb_bits_start, diff;
    short idx_azimuth, idx_azimuth_abs, flag_abs_azimuth[MAX_NUM_OBJECTS],
nbits_diff_azimuth;
    short idx_elevation, idx_elevation_abs, flag_abs_elevation[MAX_NUM_OBJECTS],
nbits_diff_elevation;
    float valQ;
    ISM_METADATA_HANDLE hIsmMetaData;
    long element_brat[MAX_NUM_OBJECTS], total_brat[MAX_NUM_OBJECTS];
    short ism_metadata_flag_global;
    short ism_imp[MAX_NUM_OBJECTS];

    /* initialization */
    ism_metadata_flag_global = 0;
    set_s( nb_bits_metadata, 0, n_ISMs );
    set_s( flag_abs_azimuth, 0, n_ISMs );
    set_s( flag_abs_elevation, 0, n_ISMs );

    /*-----*
    * Set Metadata presence / importance flag
    *-----*/

    for( ch = 0; ch < n_ISMs; ch++ )
    {
        if( hIsmMeta[ch]->ism_metadata_flag )
        {
            hIsmMeta[ch]->ism_metadata_flag = localVAD[ch];
        }
        else
        {
            hIsmMeta[ch]->ism_metadata_flag = 0;
        }

        if ( [ch]->hCoreCoder[0]->tcxonly )
        {
            /* at highest bitrates (with TCX core only) metadata are sent in every
frame */

```

```

        [ch]->ism_metadata_flag = 1;
    }
}

rate_ism_importance( n_ISms, hIsmMeta, hSCE, ism_imp );

/*-----*
 * Write ISm common signalling
 *-----*/

/* write number of objects - unary coding */
for( ch = 1; ch < n_ISms; ch++ )
{
    push_indice( hBstr, IND_ISM_NUM_OBJECTS, 1, 1 );
}
push_indice( hBstr, IND_ISM_NUM_OBJECTS, 0, 1 );

/* write ISm metadata flag (one per object) */
for( ch = 0; ch < n_ISms; ch++ )
{
    push_indice( hBstr, IND_ISM_METADATA_FLAG, ism_imp[ch],
ISM_METADATA_FLAG_BITS );

    ism_metadata_flag_global |= hIsmMeta[ch]->ism_metadata_flag;
}

/* write VAD flag */
for( ch = 0; ch < n_ISms; ch++ )
{
    if( hIsmMeta[ch]->ism_metadata_flag == 0 )
    {
        push_indice( hBstr, IND_ISM_VAD_FLAG, localVAD[ch], VAD_FLAG_BITS );
    }
}

if( ism_metadata_flag_global )
{
    /*-----*
     * Metadata quantization and coding, loop over all objects
     *-----*/

    for( ch = 0; ch < n_ISms; ch++ )
    {
        hIsmMetaData = hIsmMeta[ch];
        nb_bits_start = hBstr->nb_bits_tot;

        if( hIsmMeta[ch]->ism_metadata_flag )
        {
            /*-----*
             * Azimuth quantization and encoding
             *-----*/

            /* Azimuth quantization */

```

```

        idx_azimuth_abs = usquant( hIsmMetaData->azimuth, &valQ,
ISM_AZIMUTH_MIN, ISM_AZIMUTH_DELTA, (1 << ISM_AZIMUTH_NBITS) );
        idx_azimuth = idx_azimuth_abs;

        nbits_diff_azimuth = 0;

        flag_abs_azimuth[ch] = 0; /* differential coding by default */
        if( hIsmMetaData->azimuth_diff_cnt == ISM_FEC_MAX /* make
differential encoding in ISM_FEC_MAX consecutive frames at maximum (in order to
control the decoding in FEC) */
            || hIsmMetaData->last_ism_metadata_flag == 0 /* If last frame
had no metadata coded, do not use differential coding */
        )
        {
            flag_abs_azimuth[ch] = 1;
        }

        /* try differential coding */
        if( flag_abs_azimuth[ch] == 0 )
        {
            diff = idx_azimuth_abs - hIsmMetaData->last_azimuth_idx;

            if( diff == 0 )
            {
                idx_azimuth = 0;
                nbits_diff_azimuth = 1;
            }
            else if( ABSVAL( diff ) < ISM_MAX_AZIMUTH_DIFF_IDX ) /* when
diff bits >= abs bits, prefer abs */
            {
                idx_azimuth = 1 << 1;
                nbits_diff_azimuth = 1;

                if( diff < 0 )
                {
                    idx_azimuth += 1; /* negative sign */
                    diff *= -1;
                }
                else
                {
                    idx_azimuth += 0; /* positive sign */
                }

                idx_azimuth = idx_azimuth << diff;
                nbits_diff_azimuth++;

                /* unary coding of "diff */
                idx_azimuth += ((1<<diff) - 1);
                nbits_diff_azimuth += diff;

                if( nbits_diff_azimuth < ISM_AZIMUTH_NBITS - 1 )
                {
                    /* add stop bit - only for codewords shorter than
ISM_AZIMUTH_NBITS */

```

```

        idx_azimuth = idx_azimuth << 1;
        nbits_diff_azimuth++;
    }
}
else
{
    flag_abs_azimuth[ch] = 1;
}
}

/* update counter */
if( flag_abs_azimuth[ch] == 0 )
{
    hIsmMetaData->azimuth_diff_cnt++;
    hIsmMetaData->elevation_diff_cnt = min( hIsmMetaData-
>elevation_diff_cnt, ISM_FEC_MAX );
}
else
{
    hIsmMetaData->azimuth_diff_cnt = 0;
}

/* Write azimuth */
push_indice( hBstr, IND_ISM_AZIMUTH_DIFF_FLAG, flag_abs_azimuth[ch],
1 );

if( flag_abs_azimuth[ch] )
{
    push_indice( hBstr, IND_ISM_AZIMUTH, idx_azimuth,
ISM_AZIMUTH_NBITS );
}
else
{
    push_indice( hBstr, IND_ISM_AZIMUTH, idx_azimuth,
nbits_diff_azimuth );
}

/*-----*
 * Elevation quantization and encoding
 *-----*/

/* Elevation quantization */
idx_elevation_abs = usquant( hIsmMetaData->elevation, &valQ,
ISM_ELEVATION_MIN, ISM_ELEVATION_DELTA, (1 << ISM_ELEVATION_NBITS) );
idx_elevation = idx_elevation_abs;

nbits_diff_elevation = 0;

flag_abs_elevation[ch] = 0; /* differential coding by default */
if( hIsmMetaData->elevation_diff_cnt == ISM_FEC_MAX /* make
differential encoding in ISM_FEC_MAX consecutive frames at maximum (in order to
control the decoding in FEC) */
|| hIsmMetaData->last_ism_metadata_flag == 0 /* If last
frame had no metadata coded, do not use differential coding */

```

```

    )
    {
        flag_abs_elevation[ch] = 1;
    }

    /* note: elevation is coded starting from the second frame only (it
is meaningless in the init_frame) */
    if( hSCE[0]->hCoreCoder[0]->ini_frame == 0 )
    {
        flag_abs_elevation[ch] = 1;
        hIsmMetaData->last_elevation_idx = idx_elevation_abs;
    }

    diff = idx_elevation_abs - hIsmMetaData->last_elevation_idx;

    /* avoid absolute coding of elevation if absolute coding was already
used for azimuth */
    if( flag_abs_azimuth[ch] == 1 )
    {
        flag_abs_elevation[ch] = 0;

        if( diff >= 0 )
        {
            diff = min( diff, ISM_MAX_ELEVATION_DIFF_IDX );
        }
        else
        {
            diff = -1 * min( -diff, ISM_MAX_ELEVATION_DIFF_IDX );
        }
    }

    /* try differential coding */
    if( flag_abs_elevation[ch] == 0 )
    {
        if( diff == 0 )
        {
            idx_elevation = 0;
            nbits_diff_elevation = 1;
        }
        else if( ABSVAL( diff ) < ISM_MAX_ELEVATION_DIFF_IDX ) /* when
diff bits >= abs bits, prefer abs */
        {
            idx_elevation = 1 << 1;
            nbits_diff_elevation = 1;

            if( diff < 0 )
            {
                idx_elevation += 1; /* negative sign */
                diff *= -1;
            }
            else
            {
                idx_elevation += 0; /* positive sign */
            }
        }
    }

```

```

        idx_elevation = idx_elevation << diff;
        nbits_diff_elevation++;

        /* unary coding of "diff */
        idx_elevation += ((1 << diff) - 1);
        nbits_diff_elevation += diff;

        if( nbits_diff_elevation < ISM_ELEVATION_NBITS - 1 )
        {
            /* add stop bit */
            idx_elevation = idx_elevation << 1;
            nbits_diff_elevation++;
        }
    }
    else
    {
        flag_abs_elevation[ch] = 1;
    }
}

/* update counter */
if( flag_abs_elevation[ch] == 0 )
{
    hIsmMetaData->elevation_diff_cnt++;
    hIsmMetaData->elevation_diff_cnt = min( hIsmMetaData-
>elevation_diff_cnt, ISM_FEC_MAX );
}
else
{
    hIsmMetaData->elevation_diff_cnt = 0;
}

/* Write elevation */
if( flag_abs_azimuth[ch] == 0 )      /* do not write
"flag_abs_elevation" if "flag_abs_azimuth == 1" */ /* VE: TBV for VAD 0->1 */
{
    push_indice( hBstr, IND_ISM_ELEVATION_DIFF_FLAG,
flag_abs_elevation[ch], 1 );
}

if( flag_abs_elevation[ch] )
{
    push_indice( hBstr, IND_ISM_ELEVATION, idx_elevation,
ISM_ELEVATION_NBITS );
}
else
{
    push_indice( hBstr, IND_ISM_ELEVATION, idx_elevation,
nbits_diff_elevation );
}

/*-----*
* Updates

```

```

        *-----*/

        hIsmMetaData->last_azimuth_idx = idx_azimuth_abs;
        hIsmMetaData->last_elevation_idx = idx_elevation_abs;

        /* save number of metadata bits written */
        nb_bits_metadata[ch] = hBstr->nb_bits_tot - nb_bits_start;
    }
}

/*-----*
 * inter-object logic minimizing the use of several absolutely coded
 * indexes in the same frame
 *-----*/

i = 0;
while( i == 0 || i < n_ISms / INTER_OBJECT_PARAM_CHECK )
{
    short num, abs_num, abs_first, abs_next, pos_zero;
    short abs_matrice[INTER_OBJECT_PARAM_CHECK * 2];

    num = min( INTER_OBJECT_PARAM_CHECK, n_ISms - i *
INTER_OBJECT_PARAM_CHECK );
    i++;

    set_s( abs_matrice, 0, INTER_OBJECT_PARAM_CHECK * ISM_NUM_PARAM );

    for( ch = 0; ch < num; ch++ )
    {
        if( flag_abs_azimuth[ch] == 1 )
        {
            abs_matrice[ch*ISM_NUM_PARAM] = 1;
        }

        if( flag_abs_elevation[ch] == 1 )
        {
            abs_matrice[ch*ISM_NUM_PARAM + 1] = 1;
        }
    }
    abs_num = sum_s( abs_matrice, INTER_OBJECT_PARAM_CHECK * ISM_NUM_PARAM
);

    abs_first = 0;
    while( abs_num > 1 )
    {
        /* find first "1" entry */
        while( abs_matrice[abs_first] == 0 )
        {
            abs_first++;
        }

        /* find next "1" entry */
        abs_next = abs_first + 1;
        while( abs_matrice[abs_next] == 0 )

```

```

        {
            abs_next++;
        }

        /* find "0" position */
        pos_zero = 0;
        while( abs_matrice[pos_zero] == 1 )
        {
            pos_zero++;
        }

        ch = abs_next / ISM_NUM_PARAM;

        if( abs_next % ISM_NUM_PARAM == 0 )
        {
            hIsmMeta[ch]->azimuth_diff_cnt = abs_num - 1;
        }

        if( abs_next % ISM_NUM_PARAM == 1 )
        {
            hIsmMeta[ch]->elevation_diff_cnt = abs_num - 1;
            /*hIsmMeta[ch]->elevation_diff_cnt = min( hIsmMeta[ch]-
>elevation_diff_cnt, ISM_FEC_MAX );*/
        }

        abs_first++;
        abs_num--;
    }
}

/*-----*
 * Configuration and decision about bit rates per channel
 *-----*/

ism_config( ism_total_brat, n_ISMs, hIsmMeta, localVAD, ism_imp, element_brat,
total_brat, nb_bits_metadata );

for( ch = 0; ch < n_ISMs; ch++ )
{
    hIsmMeta[ch]->last_ism_metadata_flag = hIsmMeta[ch]->ism_metadata_flag;

    [ch]->hCoreCoder[0]->low_rate_mode = 0;
    if ( [ch]->ism_metadata_flag == 0 && [ch][0] == 0 &&
ism_metadata_flag_global )
    {
        [ch]->hCoreCoder[0]->low_rate_mode = 1;
    }

    hSCE[ch]->element_brat = element_brat[ch];
    hSCE[ch]->hCoreCoder[0]->total_brat = total_brat[ch];

    /* write metadata only in active frames */

```

```

        if( hSCE[0]->hCoreCoder[0]->core_brat > SID_2k40 )
        {
            reset_indices_enc( hSCE[ch]->hMetaData, MAX_BITS_METADATA );
        }
    }

    return;
}

void rate_ism_importance(
    const short n_ISms,          /* i : number of objects          */
    ISM_METADATA_HANDLE hIsmMeta[], /* i/o: ISM metadata handles      */
    ENC_HANDLE hSCE[],          /* i/o: element encoder handles    */
    short ism_imp[],            /* o : ISM importance flags        */
)
{
    short ch, ctype;

    for( ch = 0; ch < n_ISms; ch++ )
    {
        ctype = hSCE[ch]->hCoreCoder[0]->coder_type_raw;

        if( hIsmMeta[ch]->ism_metadata_flag == 0 )
        {
            ism_imp[ch] = ISM_NO_META;
        }
        else if( ctype == INACTIVE || ctype == UNVOICED )
        {
            ism_imp[ch] = ISM_LOW_IMP;
        }
        else if( ctype == VOICED )
        {
            ism_imp[ch] = ISM_MEDIUM_IMP;
        }
        else /* GENERIC */
        {
            ism_imp[ch] = ISM_HIGH_IMP;
        }
    }

    return;
}

void ism_config(
    const long ism_total_brat,    /* i : ISMs total bitrate         */
    const short n_ISms,          /* i : number of objects          */
    ISM_METADATA_HANDLE hIsmMeta[], /* i/o: ISM metadata handles      */
    short localVAD[],
    const short ism_imp[],        /* i : ISM importance flags        */
    long element_brat[],          /* o : element bitrate per object  */
    long total_brat[],            /* o : total bitrate per object    */
    short nb_bits_metadata[]      /* i/o: number of metadata bits    */
)

```

```

)
{
    short ch;
    short bits_element[MAX_NUM_OBJECTS], bits_CoreCoder[MAX_NUM_OBJECTS];
    short bits_ism, bits_side;
    long tmpl;
    short ism_metadata_flag_global;

    /* initialization */
    ism_metadata_flag_global = 0;
    bits_side = 0;
    if( hIsmMeta != NULL )
    {
        for( ch = 0; ch < n_ISms; ch++ )
        {
            ism_metadata_flag_global |= [ch]->ism_metadata_flag;
        }
    }

    /* decision about bit rates per channel - constant during the session (at one
ism_total_brate) */
    bits_ism = ism_total_brate / FRMS_PER_SECOND;
    set_s( bits_element, bits_ism / n_ISms, n_ISms );
    bits_element[n_ISms - 1] += bits_ism % n_ISms;
    bitbudget_to_brate( bits_element, element_brate, n_ISms );

    /* count ISm common signalling bits */
    if( hIsmMeta != NULL )
    {
        nb_bits_metadata[0] += n_ISms * ISM_METADATA_FLAG_BITS + n_ISms;

        for( ch = 0; ch < n_ISms; ch++ )
        {
            if( hIsmMeta[ch]->ism_metadata_flag == 0 )
            {
                nb_bits_metadata[0] += ISM_METADATA_VAD_FLAG_BITS;
            }
        }
    }

    /* split metadata bitbudget equally between channels */
    if( nb_bits_metadata != NULL )
    {
        bits_side = sum_s( nb_bits_metadata, n_ISms );
        set_s( nb_bits_metadata, bits_side / n_ISms, n_ISms );
        nb_bits_metadata[n_ISms - 1] += bits_side % n_ISms;
        v_sub_s( bits_element, nb_bits_metadata, bits_CoreCoder, n_ISms );
        bitbudget_to_brate( bits_CoreCoder, total_brate, n_ISms );

        mvs2s( nb_bits_metadata, nb_bits_metadata, n_ISms );
    }

    /* assign less CoreCoder bit-budget to inactive streams (at least one stream
must be active) */

```

```

if( ism_metadata_flag_global )
{
    long diff;
    short n_higher, flag_higher[MAX_NUM_OBJECTS];

    set_s( flag_higher, 1, MAX_NUM_OBJECTS );

    diff = 0;
    for( ch = 0; ch < n_ISMs; ch++ )
    {
        if( hIsMMeta[ch]->ism_metadata_flag == 0 && localVAD[ch] == 0 )
        {
            diff += bits_CoreCoder[ch] - BITS_ISM_INACTIVE;
            bits_CoreCoder[ch] = BITS_ISM_INACTIVE;
            flag_higher[ch] = 0;
        }
    }

    n_higher = sum_s( flag_higher,          );

    if( diff > 0 && n_higher > 0 )
    {
        tmpL = diff / n_higher;
        for( ch = 0; ch < n_ISMs; ch++ )
        {
            if( flag_higher[ch] )
            {
                bits_CoreCoder[ch] += tmpL;
            }
        }

        tmpL = diff % n_higher;
        ch = 0;
        while( flag_higher[ch] == 0 )
        {
            ch++;
        }
        bits_CoreCoder[ch] += tmpL;
    }

    bitbudget_to_brake( bits_CoreCoder, total_brake, n_ISMs );

    diff = 0;
    for( ch = 0; ch < n_ISMs; ch++ )
    {
        long limit;

        limit = MIN_BRATE_SWB_BWE / FRMS_PER_SECOND;
        if( element_brake[ch] < MIN_BRATE_SWB_STEREO )
        {
            limit = MIN_BRATE_WB_BWE / FRMS_PER_SECOND;
        }
        else if( element_brake[ch] >= SCE_CORE_16k_LOW_LIMIT )
        {

```

```

        /*limit = SCE_CORE_16k_LOW_LIMIT;*/
        limit = (ACELP_16k_LOW_LIMIT + SWB_TBE_1k6) / FRMS_PER_SECOND;
    }

    if( ism_imp[ch] == ISM_NO_META && localVAD[ch] == 0 )
    {
        tmpL = BITS_ISM_INACTIVE;
    }
    else if( ism_imp[ch] == ISM_LOW_IMP )
    {
        tmpL = BETA_ISM_LOW_IMP * bits_CoreCoder[ch];
        tmpL = max( limit, bits_CoreCoder[ch] - tmpL );
    }
    else if( ism_imp[ch] == ISM_MEDIUM_IMP )
    {
        tmpL = BETA_ISM_MEDIUM_IMP * bits_CoreCoder[ch];
        tmpL = max( limit, bits_CoreCoder[ch] - tmpL );
    }
    else /* ism_imp[ch] == ISM_HIGH_IMP */
    {
        tmpL = bits_CoreCoder[ch];
    }

    diff += bits_CoreCoder[ch] - tmpL;
    bits_CoreCoder[ch] = tmpL;
}

if( diff > 0 && n_higher > 0 )
{
    tmpL = diff / n_higher;
    for( ch = 0; ch < n_ISMs; ch++ )
    {
        if( flag_higher[ch] )
        {
            bits_CoreCoder[ch] += tmpL;
        }
    }

    tmpL = diff % n_higher;
    ch = 0;
    while( flag_higher[ch] == 0 )
    {
        ch++;
    }
    bits_CoreCoder[ch] += tmpL;
}

/* verify for the maximum bitrate @12.8kHz core */
diff = 0;
for ( ch = 0; ch <          ; ch++ )
{
    limit_high = STEREO_512k / FRMS_PER_SECOND;

```

```

        if ( [ch] < SCE_CORE_16k_LOW_LIMIT ) /* replicate function
set_ACELP_flag() -> it is not intended to switch the ACELP internal sampling rate
within an object */
        {
            limit_high = ACELP_12k8_HIGH_LIMIT / FRMS_PER_SECOND;
        }

        tmpL = min( bits_CoreCoder[ch], limit_high );

        diff += bits_CoreCoder[ch] - tmpL;
        bits_CoreCoder[ch] = tmpL;
    }

    if ( diff > 0 )
    {
        ch = 0;
        for ( ch = 0; ch <          ; ch++ )
        {
            if ( flag_higher[ch] == 0 )
            {
                if ( diff > limit_high )
                {
                    diff += bits_CoreCoder[ch] - limit_high;
                    bits_CoreCoder[ch] = limit_high;
                }
                else
                {
                    bits_CoreCoder[ch] += diff;
                    break;
                }
            }
        }
    }

    bitbudget_to_brake( bits_CoreCoder, total_brake, n_ISms );
}

return;
}

```

7.0 Hardware implementation

[00121] Figure 8 is a simplified block diagram of an example configuration of hardware components forming the above described coding and decoding systems and methods.

[00122] Each of the coding and decoding systems may be implemented as a

part of a mobile terminal, as a part of a portable media player, or in any similar device. Each of the coding and decoding systems (identified as 1200 in Figure 8) comprises an input 1202, an output 1204, a processor 1206 and a memory 1208.

[00123] The input 1202 is configured to receive the input signal(s), e.g. the N audio objects 102 (N audio streams with the corresponding N metadata) of Figure 1 or the bit-stream 701 of Figure 7, in digital or analog form. The output 1204 is configured to supply the output signal(s), e.g. the bit-stream 111 of Figure 1 or the M decoded audio channels 703 and the M decoded metadata 704 of Figure 7. The input 1202 and the output 1204 may be implemented in a common module, for example a serial input/output device.

[00124] The processor 1206 is operatively connected to the input 1202, to the output 1204, and to the memory 1208. The processor 1206 is realized as one or more processors for executing code instructions in support of the functions of the various processors and other modules of Figures 1 and 7.

[00125] The memory 1208 may comprise a non-transient memory for storing code instructions executable by the processor(s) 1206, specifically, a processor-readable memory comprising non-transitory instructions that, when executed, cause a processor(s) to implement the operations and processors/modules of the coding and decoding systems and methods as described in the present disclosure. The memory 1208 may also comprise a random access memory or buffer(s) to store intermediate processing data from the various functions performed by the processor(s) 1206.

[00126] Those of ordinary skill in the art will realize that the description of the coding and decoding systems and methods are illustrative only and are not intended to be in any way limiting. Other embodiments will readily suggest themselves to such persons with ordinary skill in the art having the benefit of the present disclosure. Furthermore, the disclosed coding and decoding systems and methods may be customized to offer valuable solutions to existing needs and problems of encoding and decoding sound.

[00127] In the interest of clarity, not all of the routine features of the implementations of the coding and decoding systems and methods are shown and described. It will, of course, be appreciated that in the development of any such actual implementation of the coding and decoding systems and methods, numerous implementation-specific decisions may need to be made in order to achieve the developer's specific goals, such as compliance with application-, system-, network- and business-related constraints, and that these specific goals will vary from one implementation to another and from one developer to another. Moreover, it will be appreciated that a development effort might be complex and time-consuming, but would nevertheless be a routine undertaking of engineering for those of ordinary skill in the field of sound processing having the benefit of the present disclosure.

[00128] In accordance with the present disclosure, the processors/modules, processing operations, and/or data structures described herein may be implemented using various types of operating systems, computing platforms, network devices, computer programs, and/or general purpose machines. In addition, those of ordinary skill in the art will recognize that devices of a less general purpose nature, such as hardwired devices, field programmable gate arrays (FPGAs), application specific integrated circuits (ASICs), or the like, may also be used. Where a method comprising a series of operations and sub-operations is implemented by a processor, computer or a machine and those operations and sub-operations may be stored as a series of non-transitory code instructions readable by the processor, computer or machine, they may be stored on a tangible and/or non-transient medium.

[00129] The coding and decoding systems and methods as described herein may use software, firmware, hardware, or any combination(s) of software, firmware, or hardware suitable for the purposes described herein.

[00130] In the coding and decoding systems and methods as described herein, the various operations and sub-operations may be performed in various orders and some of the operations and sub-operations may be optional.

[00131] Although the present disclosure has been described hereinabove by way of non-restrictive, illustrative embodiments thereof, these embodiments may be modified at will within the scope of the appended claims without departing from the spirit and nature of the present disclosure.

8.0 References

[00132] The following references are referred to in the present disclosure and the full contents thereof are incorporated herein by reference

- [1] 3GPP Spec. TS 26.445: "Codec for Enhanced Voice Services (EVS). Detailed Algorithmic Description," v.12.0.0, Sep. 2014.
- [2] V. Eksler, "Method and Device for Allocating a Bit-budget Between Sub-frames in a CELP Codec," PCT patent application PCT/CA2018/51175

9.0 Further embodiments

[00133] The following embodiments (Embodiments 1 to 83) are part of the present disclosure related to the invention.

[00134] Embodiment 1. A system for coding an object-based audio signal comprising audio objects in response to audio streams with associated metadata, comprising:

an audio stream processor for analyzing the audio streams; and

a metadata processor responsive to information on the audio streams from the analysis by the audio stream processor for encoding the metadata of the input audio streams.

[00135] Embodiment 2. The system of embodiment 1, wherein the metadata processor outputs information about metadata bit-budgets of the audio objects, and wherein the system further comprises a bit-budget allocator responsive to information

about metadata bit-budgets of the audio objects from the metadata processor to allocate bitrates to the audio streams.

[00136] Embodiment 3. The system of embodiment 1 or 2, comprising an encoder of the audio streams including the coded metadata.

[00137] Embodiment 4. The system of any one of embodiments 1 to 3, wherein the encoder comprises a number of Core-Coders using the bitrates allocated to the audio streams by the bit-budget allocator.

[00138] Embodiment 5. The system of any one of embodiments 1 to 4, wherein the object-based audio signal comprises at least one of speech, music and general audio sound.

[00139] Embodiment 6. The system of any one of embodiments 1 to 5, wherein the object-based audio signal represents or encodes a complex audio auditory scene as a collection of individual elements, said audio objects.

[00140] Embodiment 7. The system of any one of embodiments 1 to 6, wherein each audio object comprises an audio stream with associated metadata.

[00141] Embodiment 8. The system of any one of embodiments 1 to 7, wherein the audio stream is an independent stream with metadata.

[00142] Embodiment 9. The system of any one of embodiments 1 to 8, wherein the audio stream represents an audio waveform and usually comprises one or two channels.

[00143] Embodiment 10. The system of any one of embodiments 1 to 9, wherein the metadata is a set of information that describes the audio stream and an artistic intention used to translate the original or coded audio objects to a final reproduction system.

[00144] Embodiment 11. The system of any one of embodiments 1 to 10,

wherein the metadata usually describes spatial properties of each audio object.

[00145] Embodiment 12. The system of any one of embodiments 1 to 11, wherein the spatial properties include one or more of a position, orientation, volume, width of the audio object.

[00146] Embodiment 13. The system of any one of embodiments 1 to 12, wherein each audio object comprises a set of metadata referred to as input metadata defined as an unquantized metadata representation used as an input to a codec.

[00147] Embodiment 14. The system of any one of embodiments 1 to 13, wherein each audio object comprises a set of metadata referred to as coded metadata defined as quantized and coded metadata which are part of a bit-stream sent from an encoder to a decoder.

[00148] Embodiment 15. The system of any one of embodiments 1 to 14, wherein a reproduction system is structured to render the audio objects in a 3D audio space around a listener using the transmitted metadata and artistic intention at a reproduction side.

[00149] Embodiment 16. The system of any one of embodiments 1 to 15, wherein the reproduction system comprises a head-tracking device for dynamically modify the metadata during rendering the audio objects.

[00150] Embodiment 17. The system of any one of embodiments 1 to 16, comprising a framework for a simultaneous coding of several audio objects.

[00151] Embodiment 18. The system of any one of embodiments 1 to 17, wherein the simultaneous coding of several audio objects uses a fixed constant overall bitrate for encoding the audio objects.

[00152] Embodiment 19. The system of any one of embodiments 1 to 18, comprising a transmitter for transmitting a part or all of the audio objects.

[00153] Embodiment 20. The system of any one of embodiments 1 to 19, wherein, in the case of coding a combination of audio formats in the framework, a constant overall bitrate represents a sum of the bitrates of the formats.

[00154] Embodiment 21. The system of any one of embodiments 1 to 20, wherein the metadata comprises two parameters comprising azimuth and elevation.

[00155] Embodiment 22. The system of any one of embodiments 1 to 21, wherein the azimuth and elevation parameters are stored per each audio frame for each audio object.

[00156] Embodiment 23. The system of any one of embodiments 1 to 22, comprising an input buffer for buffering at least one input audio stream and input metadata associated to the audio stream.

[00157] Embodiment 24. The system of any one of embodiments 1 to 23, wherein the input buffer buffers each audio stream for one frame.

[00158] Embodiment 25. The system of any one of embodiments 1 to 24, wherein the audio stream processor analyzes and processes the audio streams.

[00159] Embodiment 26. The system of any one of embodiments 1 to 25, wherein the audio stream processor comprises at least one of the following elements: a time-domain transient detector, a spectral analyser, a long-term prediction analyser, a pitch tracker and voicing analyser, a voice/sound activity detector, a band-width detector, a noise estimator and a signal classifier.

[00160] Embodiment 27. The system of any one of embodiments 1 to 26, wherein the signal classifier performs at least one of coder type selection, signal classification, and speech/music classification.

[00161] Embodiment 28. The system of any one of embodiments 1 to 27, wherein the metadata processor analyzes, quantizes and encodes the metadata of

the audio streams.

[00162] Embodiment 29. The system of any one of embodiments 1 to 28, wherein, in inactive frames, no metadata is encoded by the metadata processor and sent by the system in a bit-stream for the corresponding audio object.

[00163] Embodiment 30. The system of any one of embodiments 1 to 29, wherein, in active frames, the metadata are encoded by the metadata processor for the corresponding object using a variable bitrate.

[00164] Embodiment 31. The system of any one of embodiments 1 to 30, wherein the bit-budget allocator sums the bit-budgets of the metadata of the audio objects, and adds the sum of bit-budgets to a signaling bit-budget in order to allocate the bitrates to the audio streams.

[00165] Embodiment 32. The system of any one of embodiments 1 to 31, comprising a pre-processor to further process the audio streams when configuration and bit-rate distribution between audio streams has been done.

[00166] Embodiment 33. The system of any one of embodiments 1 to 32, wherein the pre-processor performs at least one of further classification of the audio streams, core encoder selection, and resampling.

[00167] Embodiment 34. The system of any one of embodiments 1 to 33, wherein the encoder sequentially encodes the audio streams.

[00168] Embodiment 35. The system of any one of embodiments 1 to 34, wherein the encoder sequentially encodes the audio streams using a number fluctuating bitrate Core-Coders.

[00169] Embodiment 36. The device of any one of embodiments 1 to 35, wherein the metadata processor encodes the metadata sequentially in a loop with dependency between quantization of the audio objects and metadata parameters of

the audio objects.

[00170] Embodiment 37. The system of any one of embodiments 1 to 36, wherein the metadata processor, to encode a metadata parameter, quantizes a metadata parameter index using a quantization step.

[00171] Embodiment 38. The system of any one of embodiments 1 to 37, wherein the metadata processor, to encode the azimuth parameter, quantizes an azimuth index using a quantization step and, to encode the elevation parameter, quantizes an elevation index using a quantization step.

[00172] Embodiment 39. The device of any one of embodiments 1 to 38, wherein a total metadata bit-budget and a number of quantization bits are dependent on a codec total bitrate, a metadata total bitrate, or a sum of metadata bit budget and Core-Coder bit budget related to one audio object.

[00173] Embodiment 40. The system of any one of embodiments 1 to 39, wherein the azimuth and elevation parameters are represented as one parameter.

[00174] Embodiment 41. The system of any one of embodiments 1 to 40, wherein the metadata processor encodes the metadata parameter indexes either absolutely or differentially.

[00175] Embodiment 42. The system of any one of embodiments 1 to 41, wherein the metadata processor encodes the metadata parameter indices using absolute coding when there is a difference between current and previous parameter indices that results in a higher or equal number of bits needed for the differential coding than the absolute coding.

[00176] Embodiment 43. The system of any one of embodiments 1 to 42, wherein the metadata processor encodes the metadata parameter indices using absolute coding when there were no metadata present in a previous frame.

[00177] Embodiment 44. The system of any one of embodiments 1 to 43, wherein the metadata processor encodes the metadata parameter indices using absolute coding when a number of consecutive frames using differential coding is higher than a number of maximum consecutive frames coded using differential coding.

[00178] Embodiment 45. The system of any one of embodiments 1 to 44, wherein the metadata processor, when encoding the metadata parameter indices using absolute coding, writes an absolute coding flag distinguishing between absolute and differential coding following a metadata parameter absolute coded index.

[00179] Embodiment 46. The system of any one of embodiments 1 to 45, wherein the metadata processor, when encoding the metadata parameter indices using differential coding, sets the absolute coding flag to 0 and writes a zero coding flag, following the absolute coding flag, signaling if the difference between a current and a previous frame index is 0.

[00180] Embodiment 47. The system of any one of embodiments 1 to 46, wherein, if the difference between a current and a previous frame index is not equal to 0, the metadata processor continues coding by writing a sign flag followed by an adaptive-bits difference index.

[00181] Embodiment 48. The system of any one of embodiments 1 to 47, wherein the metadata processor uses an intra-object metadata coding logic to limit a range of metadata bit-budget fluctuation between frames and to avoid too low a bit-budget left for the core coding.

[00182] Embodiment 49. The system of any one of embodiments 1 to 48, wherein the metadata processor, in accordance with the intra-object metadata coding logic, limits the use of absolute coding in a given frame to one metadata parameter only or to a number as low as possible of metadata parameters.

[00183] Embodiment 50. The system of any one of embodiments 1 to 49, wherein the metadata processor, in accordance with the intra-object metadata coding

logic, avoids absolute coding of an index of one metadata parameter if the index of another metadata coding logic was already coded using absolute coding in a same frame.

[00184] Embodiment 51. The system of any one of embodiments 1 to 50, wherein the intra-object metadata coding logic is bitrate dependent.

[00185] Embodiment 52. The system of any one of embodiments 1 to 51, wherein the metadata processor uses an inter-object metadata coding logic used between metadata coding of different objects to minimize a number of absolutely coded metadata parameters of different audio objects in a current frame.

[00186] Embodiment 53. The system of any one of embodiments 1 to 52, wherein the metadata processor, using the inter-object metadata coding logic, controls frame counters of absolutely coded metadata parameters.

[00187] Embodiment 54. The system of any one of embodiments 1 to 53, wherein the metadata processor, using the inter-object metadata coding logic, when the metadata parameters of the audio objects evolve slowly and smoothly, codes (a) a first metadata parameter index of a first audio object using absolute coding in a frame M , (b) a second metadata parameter index of the first audio object using absolute coding in a frame $M+1$, (c) the first metadata parameter index of a second audio object using absolute coding in a frame $M+2$, and (d) the second metadata parameter index of the second audio object using absolute coding in a frame $M+3$.

[00188] Embodiment 55. The system of any one of embodiments 1 to 54, wherein the inter-object metadata coding logic is bitrate dependent.

[00189] Embodiment 56. The system of any one of embodiments 1 to 55, wherein the bit-budget allocator uses a bitrate adaptation algorithm to distribute the bit-budget for encoding the audio streams.

[00190] Embodiment 57. The system of any one of embodiments 1 to 56,

wherein the bit-budget allocator, using the bitrate adaptation algorithm, obtains a metadata total bit-budget from a metadata total bitrate or codec total bitrate.

[00191] Embodiment 58. The system of any one of embodiments 1 to 57, wherein the bit-budget allocator, using the bitrate adaptation algorithm, computes an element bit-budget by dividing the metadata total bit-budget by the number of audio streams.

[00192] Embodiment 59. The system of any one of embodiments 1 to 58, wherein the bit-budget allocator, using the bitrate adaptation algorithm, adjusts the element bit-budget of a last audio stream to spend all available metadata bit-budget.

[00193] Embodiment 60. The system of any one of embodiments 1 to 59, wherein the bit-budget allocator, using the bitrate adaptation algorithm, sums a metadata bit-budget of all the audio objects and adds said sum to a metadata common signaling bit-budget resulting in a Core-Coder side bit-budget.

[00194] Embodiment 61. The system of any one of embodiments 1 to 60, wherein the bit-budget allocator, using the bitrate adaptation algorithm, (a) splits the Core-Coder side bit-budget equally between the audio objects and (b) uses the split Core-Coder side bit-budget and the element bit-budget to compute a Core-Coder bit-budget for each audio stream.

[00195] Embodiment 62. The system of any one of embodiments 1 to 61, wherein the bit-budget allocator, using the bitrate adaptation algorithm, adjusts the Core-Coder bit-budget of a last audio stream to spend all available Core-Coder bit-budget.

[00196] Embodiment 63. The system of any one of embodiments 1 to 62, wherein the bit-budget allocator, using the bitrate adaptation algorithm, computes a bitrate for encoding one audio stream in a Core-Coder using the Core-Coder bit-budget.

[00197] Embodiment 64. The system of any one of embodiments 1 to 63, wherein the bit-budget allocator, using the bitrate adaptation algorithm in inactive frames or in frames with low energy, lowers and sets to a constant value the bitrate for encoding one audio stream in a Core-Coder, and redistribute a saved bit-budget between the audio streams in active frames.

[00198] Embodiment 65. The system of any one of embodiments 1 to 64, wherein the bit-budget allocator, using the bitrate adaptation algorithm in active frames, adjusts the bitrate for encoding one audio stream in a Core-Coder based on a metadata importance classification.

[00199] Embodiment 66. The system of any one of embodiments 1 to 65, wherein the bit-budget allocator, in inactive frames ($VAD = 0$), lowers the bitrate for encoding one audio stream in a Core-Coder and redistribute a bit-budget saved by said bitrate lowering between audio streams in frames classified as active.

[00200] Embodiment 67. The system of any one of embodiments 1 to 66, wherein the bit-budget allocator, in a frame, (a) sets to every audio stream with inactive content a lower, constant Core-Coder bit-budget, (b) computes a saved bit-budget as a difference between the lower constant Core-Coder bit-budget and the Core-Coder bit-budget, and (c) redistributes the saved bit-budget between the Core-Coder bit-budget of the audio streams in active frames.

[00201] Embodiment 68. The system of any one of embodiments 1 to 67, wherein the lower, constant bit-budget is dependent upon the metadata total bit-rate.

[00202] Embodiment 69. The system of any one of embodiments 1 to 68, wherein the bit-budget allocator computes the bitrate to encode one audio stream in a Core-Coder using the lower constant Core-Coder bit-budget.

[00203] Embodiment 70. The system of any one of embodiments 1 to 69, wherein the bit-budget allocator uses an inter-object Core-Coder bitrate adaptation based on a classification of metadata importance.

[00204] Embodiment 71. The system of any one of embodiments 1 to 70, wherein the metadata importance is based on a metric indicating how critical coding of a particular audio object at a current frame to obtain a decent quality of the decoded synthesis is.

[00205] Embodiment 72. The system of any one of embodiments 1 to 71, wherein the bit-budget allocator bases the classification of metadata importance on at least one of the following parameters: coder type (*coder_type*), FEC signal classification (*class*), speech/music classification decision, and SNR estimate from the open-loop ACELP/TCX core decision module (*snr_celp*, *snr_tcx*).

[00206] Embodiment 73. The system of any one of embodiments 1 to 72, wherein the bit-budget allocator bases the classification of metadata importance on the coder type (*coder_type*).

[00207] Embodiment 74. The system of any one of embodiments 1 to 73, wherein the bit-budget allocator defines the four following distinct metadata importance classes (*class_{ISM}*):

- No metadata class, *ISM_NO_META*: frames without metadata coding, for example in inactive frames with VAD = 0
- Low importance class, *ISM_LOW_IMP*: frames where *coder_type* = *UNVOICED* or *INACTIVE*
- Medium importance class, *ISM_MEDIUM_IMP*: frames where *coder_type* = *VOICED*
- High importance class *ISM_HIGH_IMP*: frames where *coder_type* = *GENERIC*).

[00208] Embodiment 75. The system of any one of embodiments 1 to 74, wherein the bit-budget allocator uses the metadata importance class in the bitrate

adaptation algorithm to assign a higher bit-budget to audio streams with a higher importance and a lower bit-budget to audio streams with a lower importance.

[00209] Embodiment 76. The system of any one of embodiments 1 to 75, wherein the bit-budget allocator uses, in a frame, the following logic:

1. $class_{ISM} = ISM_NO_META$ frames: the lower constant Core-Coder bitrate is assigned;
2. $class_{ISM} = ISM_LOW_IMP$ frames: the bitrate to encode one audio stream in a Core-Coder ($total_brate$) is lowered as

$$total_brate_{new}[n] = \max(\alpha_{low} * total_brate[n], B_{low})$$

where the constant α_{low} is set to a value lower than 1.0, and the constant B_{low} is a minimum bitrate threshold supported by the Core-Coder;

3. $class_{ISM} = ISM_MEDIUM_IMP$ frames: the bitrate to encode one audio stream in a Core-Coder ($total_brate$) is lowered as

$$total_brate_{new}[n] = \max(\alpha_{med} * total_brate[n], B_{low})$$

where the constant α_{med} is set to a value lower than 1.0 but higher than a value α_{low} ;

4. $class_{ISM} = ISM_HIGH_IMP$ frames: no bitrate adaptation is used.

[00210] Embodiment 77. The system of any one of embodiments 1 to 76, wherein the bit-budget allocator redistributes a saved bit-budget expressed as a sum of differences between the previous and new bitrates $total_brate$ between the audio streams in frames classified as active.

[00211] Embodiment 78. A system for decoding audio objects in response to audio streams with associated metadata, comprising:

a metadata processor for decoding metadata of the audio streams with active contents;

a bit-budget allocator responsive to the decoded metadata and respective bit-budgets of the audio objects to determine Core-Coder bitrates of the audio streams; and

a decoder of the audio streams using the Core-Coder bitrates determined in the bit-budget allocator.

[00212] Embodiment 79. The system of embodiment 78, wherein the metadata processor is responsive to metadata common signaling read from an end of a received bitstream.

[00213] Embodiment 80. The system of embodiment 78 or 79, wherein the decoder comprises Core-Decoders to decode the audio streams.

[00214] Embodiment 81. The system of any one of embodiments 78 to 80, wherein the Core-Decoders comprise fluctuating bitrate Core-Decoders to sequentially decode the audio streams at their respective Core-Coder bitrates.

[00215] Embodiment 82. The system of any one of embodiments 78 to 81, wherein a number of decoded audio objects is lower than a number of Core-Decoders.

[00216] Embodiment 83. The system of any one of embodiments 78 to 83, comprising a renderer of audio objects in response to the decoded audio streams and decoded metadata.

[00217] Any of embodiments 2 to 77 further describing the elements of embodiments 78 to 83 can be implemented in any of these embodiments 78 to 83. As

an example, the Core-Coder bitrates per audio stream in the decoding system are determined using the same procedure as in the coding system.

[00218] The present invention is also concerned with a method of coding and a method of decoding. In this respect, system embodiments 1 to 83 can be drafted as method embodiments in which the elements of the system embodiments are replaced by an operation performed by such elements.

WHAT IS CLAIMED IS:

1. A system for coding an object-based audio signal comprising audio objects in response to audio streams with associated metadata, comprising:
 - an audio stream processor for analyzing the audio streams;
 - a metadata processor responsive to information on the audio streams from the analysis by the audio stream processor for coding the metadata, wherein the metadata processor uses a logic for controlling a metadata coding bit-budget; and
 - an encoder for coding the audio streams.
2. The system according to claim 1, wherein the metadata processor uses an intra-object metadata coding logic to limit a range of metadata coding bit-budget fluctuation between frames of the object-based audio signal and to avoid too low a bit-budget left for coding the audio streams.
3. The system according to claim 2, wherein the metadata processor, using the intra-object metadata coding logic, limits absolute coding in a given frame to one metadata parameter or to a number as low as possible of metadata parameters.
4. The system according to claim 2 or 3, wherein the metadata processor, using the intra-object metadata coding logic, avoids in a same frame absolute coding of a first metadata parameter if a second metadata parameter was already coded using absolute coding.
5. The system according to any one of claims 2 to 4, wherein the intra-object metadata coding logic is bitrate dependent to enable absolute coding of a plurality of metadata parameters in the same frame if the bitrate is sufficiently large.
6. The system according to claim 1, wherein the metadata processor applies an inter-object metadata coding logic to metadata coding of different audio objects to

minimize, in a current frame, a number of metadata parameters of different audio objects coded using absolute coding.

7. The system according to claim 6, wherein the metadata processor, using the inter-object metadata coding logic, controls frame counters of metadata parameters coded using absolute coding.

8. The system according to claim 6 or 7, wherein the metadata processor, using the inter-object metadata coding logic, codes one audio object metadata parameter by frame.

9. The system according to any one of claims 6 to 8, wherein the metadata processor, using the inter-object metadata coding logic when the metadata parameters of the audio objects evolve slowly and smoothly, codes (a) a first metadata parameter of a first audio object using absolute coding in a frame M , (b) a second metadata parameter of the first audio object using absolute coding in a frame $M+1$, (c) the first metadata parameter of a second audio object using absolute coding in a frame $M+2$, and (d) second metadata parameter of the second audio object using absolute coding in a frame $M+3$.

10. The system according to any one of claims 6 to 9, wherein the inter-object metadata coding logic is bitrate dependent to enable absolute coding of a plurality of metadata parameters of the audio objects in the same frame if the bitrate is sufficiently large.

11. The system according to any one of claims 1 to 10, comprising an input buffer for buffering a number of audio objects each including one of the audio streams with the associated metadata.

12. The system according to any one of claims 1 to 11, wherein:

- the audio stream processor analyzes the audio streams to detect voice activity;
- the metadata processor comprises an analyzer of the metadata of each audio object using the voice activity detection from the audio stream processor to determine if a current frame is inactive or active with respect to the audio object;
- in inactive frames, the metadata processor codes no metadata relative to the audio object; and
- in active frames, the metadata processor codes the metadata for the audio object.

13. The device according to any one of claims 1 to 12, wherein the metadata processor codes the metadata sequentially in a loop with dependency between quantization of the audio objects and metadata parameters of the audio objects.

14. The system according to any one of claims 1 to 13, wherein the metadata processor comprises, to quantize a metadata parameter of an audio object, a quantizer of a metadata parameter index using a quantization step.

15. The system according to any one of claims 1 to 14, wherein:

the metadata of each audio object comprise an azimuth parameter and an elevation parameter; and

the metadata processor comprises, to quantize the azimuth and elevation parameters, a quantizer of an azimuth index using a quantization step and of an elevation parameter index using a quantization step.

16. The system according to claim 14 or 15, wherein a total metadata bit-budget for coding the metadata and a total number of quantization bits for quantizing the metadata parameter indexes are dependent on a codec total bitrate, a metadata total bitrate, or a sum of a metadata bit-budget and a core-encoder bit-budget related to one audio object.

17. The system according to any one of claims 1 to 16, wherein:
the metadata of each audio object comprise a plurality of metadata parameters;
the metadata processor represents the plurality of metadata parameters as one parameter; and
the metadata processor comprises a quantizer of an index of the said one parameter.
18. The system according to any one of claims 14 to 16, wherein the metadata processor comprises a metadata encoder for coding the metadata parameter indexes using either absolute or differential coding.
19. The system according to claim 18, wherein the metadata encoder codes the metadata parameter indexes using absolute coding if a difference between current and previous values of the parameter index results in a higher or equal number of bits for using differential coding compared to using absolute coding.
20. The system according to claim 18 or 19, wherein the metadata encoder processor codes the metadata parameter indexes using absolute coding if no metadata were present in a previous frame.
21. The system according to any one of claims 18 to 20, wherein the metadata encoder codes the metadata parameter indexes using absolute coding when a number of consecutive frames using differential coding is higher than a number of maximum consecutive frames coded using differential coding.
22. The system according to any one of claims 18 to 21, wherein the metadata encoder, when coding a metadata parameter index using absolute coding, produces

an absolute coding flag distinguishing between absolute and differential coding and followed by the metadata parameter index coded using absolute coding.

23. The system according to claim 22, wherein the metadata encoder, when encoding a metadata parameter index using differential coding, sets the absolute coding flag to 0 and produces a zero coding flag following the absolute coding flag, signaling a difference between the metadata parameter index in a current frame and the metadata parameter index in a previous frame equal to 0.

24. The system according to claim 23, wherein, if the difference between the metadata parameter index in the current frame and the metadata parameter index in the previous frame is not equal to 0, the metadata encoder produces a sign flag indicative of a plus or minus sign of the difference followed by a difference index indicative of the value of the difference.

25. The system according to any one of claims 1 to 24, wherein the metadata processor outputs information about bit-budgets for the coding of the metadata of the audio objects, and wherein the system further comprises a bit-budget allocator responsive to information about the bit-budgets for the coding of the metadata of the audio objects from the metadata processor to allocate bitrates for the coding of the audio streams.

26. The system according to claim 25, wherein the bit-budget allocator sums the bit-budgets for the coding of the metadata of the audio objects, and adds the sum of the bit-budgets to a signaling bit-budget to perform bitrate distribution between the audio streams.

27. The system according to claims 25 or 26, comprising a pre-processor to further process the audio streams once bitrate distribution by the bit-budget allocator between the audio streams is completed.

28. The system according to claim 27, wherein the pre-processor performs at least one of further classification of the audio streams, core-encoder selection, and resampling.

29. The system according to any one of claims 1 to 28, wherein the encoder of the audio streams comprises a number of core-encoders for coding the audio streams.

30. The system according to claim 29, wherein the core-encoders are fluctuating bitrate core-encoders sequentially coding the audio streams.

31. An encoder device for coding a complex audio auditory scene comprising scene-based audio, multi-channels, and object-based audio signals, comprising a system according to any one of claims 1 to 30 for coding the object-based audio signals.

32. A method for coding an object-based audio signal comprising audio objects in response to audio streams with associated metadata, comprising:

analyzing the audio streams;

coding the metadata using (a) information on the audio streams from the analysis of the audio streams, and (b) a logic for controlling a metadata coding bit-budget; and

encoding the audio streams.

33. The method according to claim 32, wherein using a logic for controlling the metadata coding bit-budget comprises using an intra-object metadata coding logic to limit a range of metadata coding bit-budget fluctuation between frames of the object-based audio signal and to avoid too low a bit-budget left for coding the audio streams.

34. The method according to claim 33, wherein using the intra-object metadata coding logic comprises limiting absolute coding in a given frame to one metadata parameter or to a number as low as possible of metadata parameters.

35. The method according to claim 33 or 34, wherein using the intra-object metadata coding logic comprises avoiding in a same frame absolute coding of a first metadata parameter if a second metadata parameter was already coded using absolute coding.

36. The method according to any one of claims 33 to 35, wherein the intra-object metadata coding logic is bitrate dependent to enable absolute coding of a plurality of metadata parameters in the same frame if the bitrate is sufficiently large.

37. The method according to claim 32, wherein using a logic for controlling a metadata coding bit-budget comprises using an inter-object metadata coding logic for metadata coding of different audio objects to minimize, in a current frame, a number of metadata parameters of different audio objects coded using absolute coding.

38. The method according to claim 37, wherein using the inter-object metadata coding logic comprises controlling frame counters of metadata parameters coded using absolute coding.

39. The method according to claim 37 or 38, wherein using the inter-object metadata coding logic comprises coding one audio object metadata parameter by frame.

40. The method according to any one of claims 37 to 39, wherein using the inter-object metadata coding logic comprises, when the metadata parameters of the audio objects evolve slowly and smoothly, coding (a) a first metadata parameter of a first audio object using absolute coding in a frame M , (b) a second metadata parameter of

the first audio object using absolute coding in a frame $M+1$, (c) the first metadata parameter of a second audio object using absolute coding in a frame $M+2$, and (d) second metadata parameter of the second audio object using absolute coding in a frame $M+3$.

41. The method according to any one of claims 37 to 40, wherein the inter-object metadata coding logic is bitrate dependent to enable absolute coding of a plurality of metadata parameters of the audio objects in the same frame if the bitrate is sufficiently large.

42. The method according to any one of claims 32 to 41, comprising input buffering a number of audio objects each including one of the audio streams with the associated metadata.

43. The method according to any one of claims 32 to 42, comprising:

- detecting voice activity upon analyzing the audio streams;
- analyzing the metadata of each audio object using the voice activity detection to determine if a current frame is inactive or active with respect to the audio object;
- in inactive frames, encoding no metadata relative to the audio object; and
- in active frames, encoding the metadata for the audio object.

44. The method according to any one of claims 32 to 43, wherein the metadata are coded sequentially in a loop with dependency between quantization of the audio objects and metadata parameters of the audio objects.

45. The method according to any one of claims 32 to 44 comprising, to quantize a metadata parameter of an audio object, quantizing a metadata parameter index using a quantization step.

46. The method according to any one of claims 32 to 45, wherein:
the metadata of each audio object comprise an azimuth parameter and an elevation parameter; and
quantizing the azimuth and elevation parameters comprises quantizing an azimuth index using a quantization step and quantizing an elevation parameter index using a quantization step.
47. The method according to claim 45 or 46, wherein a total metadata bit-budget for coding the metadata and a total number of quantization bits for quantizing the metadata parameter indexes are dependent on a codec total bitrate, a metadata total bitrate, or a sum of a metadata bit-budget and a core-encoder bit-budget related to one audio object.
48. The method according to any one of claims 32 to 47, wherein the metadata of each audio object comprise a plurality of metadata parameters, and wherein the method comprises:
representing the plurality of metadata parameters as one parameter; and
quantizing an index of the said one parameter.
49. The method according to any one of claims 45 to 47, comprising coding the metadata parameter indexes using either absolute or differential coding.
50. The method according to claim 49, wherein coding the metadata parameter indexes comprises using absolute coding if a difference between current and previous values of the parameter index results in a higher or equal number of bits for using differential coding compared to using absolute coding.
51. The method according to claim 49 or 50, wherein coding the metadata parameter indexes comprises using absolute coding if no metadata were present in a previous frame.

52. The method according to any one of claims 49 to 51, wherein coding the metadata parameter indexes comprises using absolute coding when a number of consecutive frames using differential coding is higher than a number of maximum consecutive frames coded using differential coding.

53. The method according to any one of claims 49 to 52, wherein coding a metadata parameter index using absolute coding comprises producing an absolute coding flag distinguishing between absolute and differential coding and followed by the metadata parameter index coded using absolute coding.

54. The method according to claim 53, wherein coding a metadata parameter index using differential coding comprises setting the absolute coding flag to 0 and producing a zero coding flag following the absolute coding flag, signaling a difference between the metadata parameter index in a current frame and the metadata parameter index in a previous frame equal to 0.

55. The method according to claim 54, wherein coding a metadata parameter index using differential coding comprises, if the difference between the metadata parameter index in the current frame and the metadata parameter index in the previous frame is not equal to 0, producing a sign flag indicative of a plus or minus sign of the difference followed by a difference index indicative of the value of the difference.

56. The method according to any one of claims 32 to 55, wherein coding the metadata comprises outputting information about bit-budgets for the coding of the metadata of the audio objects, and wherein the method comprises a bit-budget allocation responsive to information about the bit-budgets for the coding of the metadata of the audio objects to allocate bitrates for the coding of the audio streams.

57. The method according to claim 56, wherein the bit-budget allocation comprises summing the bit-budgets for the coding of the metadata of the audio objects, and adding the sum of the bit-budgets to a signaling bit-budget to perform bitrate distribution between the audio streams.

58. The method according to claims 56 or 57, comprising pre-processing the audio streams once bitrate distribution by the bit-budget allocation between the audio streams is completed.

59. The method according to claim 58, wherein pre-processing the audio streams comprises performing at least one of further classification of the audio streams, core-encoder selection, and resampling.

60. The method according to any one of claims 32 to 59, wherein encoding the audio streams comprises respective core-encoding of the audio streams.

61. The method according to claim 60, wherein encoding the audio streams comprises using fluctuating bitrates for sequentially coding the audio streams.

62. An encoding method for coding a complex audio auditory scene comprising scene-based audio, multi-channels, and object-based audio signals, comprising a method according to any one of claims 32 to 61 for coding the object-based audio signals.

1/8

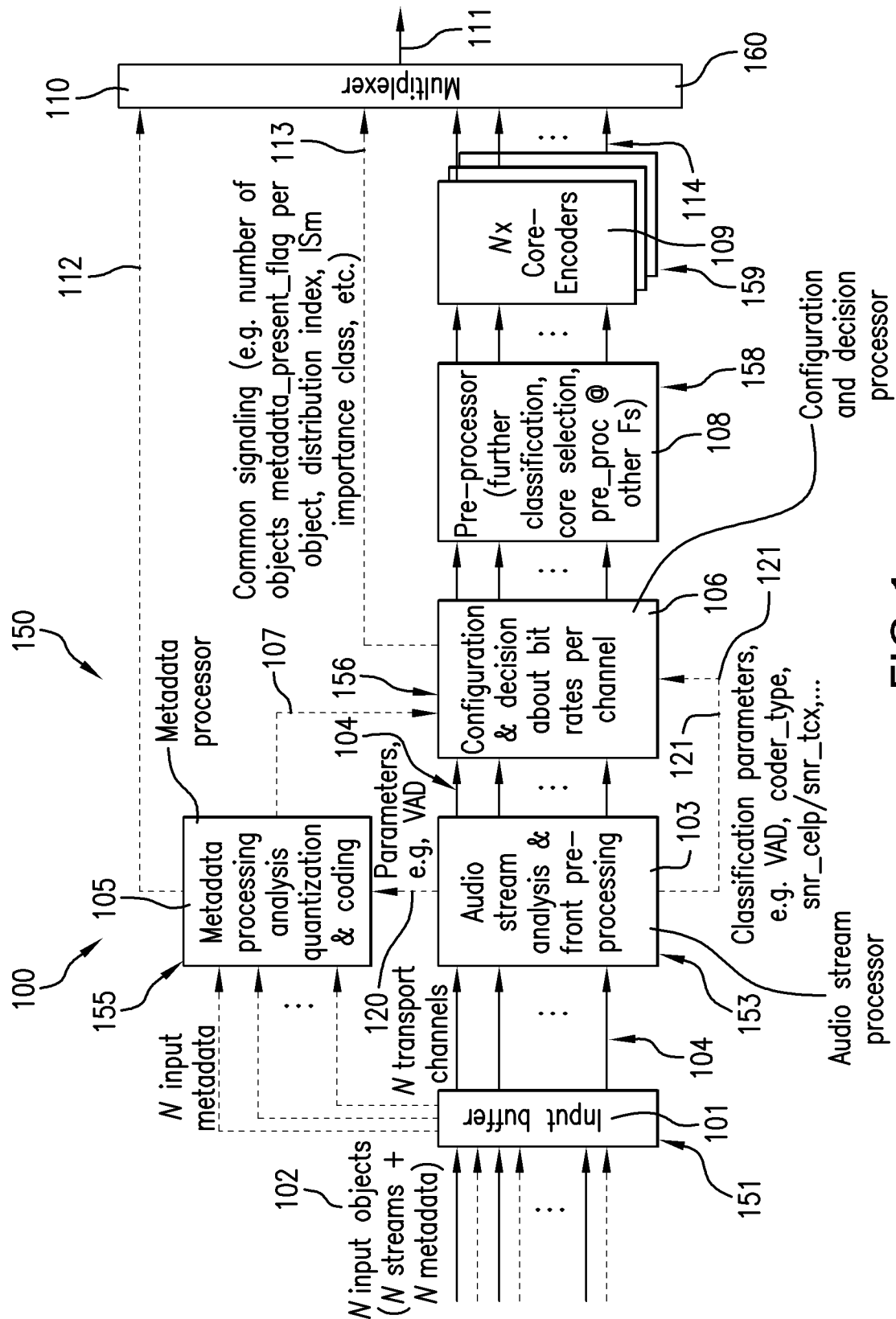


FIG. 1

absolute	<table><tr><td>$flag_{abs}$ 1</td><td>$index: B_{02}$ bits</td></tr></table>	$flag_{abs}$ 1	$index: B_{02}$ bits				
$flag_{abs}$ 1	$index: B_{02}$ bits						
differential $\Delta = 0$	<table><tr><td>$flag_{abs}$ 0</td><td><table><tr><td>$flag_{zero}$ 1</td></tr></table></td></tr></table>	$flag_{abs}$ 0	<table><tr><td>$flag_{zero}$ 1</td></tr></table>	$flag_{zero}$ 1			
$flag_{abs}$ 0	<table><tr><td>$flag_{zero}$ 1</td></tr></table>	$flag_{zero}$ 1					
$flag_{zero}$ 1							
differential positive Δ	<table><tr><td>$flag_{abs}$ 0</td><td><table><tr><td>$flag_{zero}$ 0</td><td><table><tr><td>$flag_{sign}$ 0</td><td>$index: 1$ to $(B_{02}-3)$ bits</td></tr></table></td></tr></table></td></tr></table>	$flag_{abs}$ 0	<table><tr><td>$flag_{zero}$ 0</td><td><table><tr><td>$flag_{sign}$ 0</td><td>$index: 1$ to $(B_{02}-3)$ bits</td></tr></table></td></tr></table>	$flag_{zero}$ 0	<table><tr><td>$flag_{sign}$ 0</td><td>$index: 1$ to $(B_{02}-3)$ bits</td></tr></table>	$flag_{sign}$ 0	$index: 1$ to $(B_{02}-3)$ bits
$flag_{abs}$ 0	<table><tr><td>$flag_{zero}$ 0</td><td><table><tr><td>$flag_{sign}$ 0</td><td>$index: 1$ to $(B_{02}-3)$ bits</td></tr></table></td></tr></table>	$flag_{zero}$ 0	<table><tr><td>$flag_{sign}$ 0</td><td>$index: 1$ to $(B_{02}-3)$ bits</td></tr></table>	$flag_{sign}$ 0	$index: 1$ to $(B_{02}-3)$ bits		
$flag_{zero}$ 0	<table><tr><td>$flag_{sign}$ 0</td><td>$index: 1$ to $(B_{02}-3)$ bits</td></tr></table>	$flag_{sign}$ 0	$index: 1$ to $(B_{02}-3)$ bits				
$flag_{sign}$ 0	$index: 1$ to $(B_{02}-3)$ bits						
differential negative Δ	<table><tr><td>$flag_{abs}$ 0</td><td><table><tr><td>$flag_{zero}$ 0</td><td><table><tr><td>$flag_{sign}$ 1</td><td>$index: 1$ to $(B_{02}-3)$ bits</td></tr></table></td></tr></table></td></tr></table>	$flag_{abs}$ 0	<table><tr><td>$flag_{zero}$ 0</td><td><table><tr><td>$flag_{sign}$ 1</td><td>$index: 1$ to $(B_{02}-3)$ bits</td></tr></table></td></tr></table>	$flag_{zero}$ 0	<table><tr><td>$flag_{sign}$ 1</td><td>$index: 1$ to $(B_{02}-3)$ bits</td></tr></table>	$flag_{sign}$ 1	$index: 1$ to $(B_{02}-3)$ bits
$flag_{abs}$ 0	<table><tr><td>$flag_{zero}$ 0</td><td><table><tr><td>$flag_{sign}$ 1</td><td>$index: 1$ to $(B_{02}-3)$ bits</td></tr></table></td></tr></table>	$flag_{zero}$ 0	<table><tr><td>$flag_{sign}$ 1</td><td>$index: 1$ to $(B_{02}-3)$ bits</td></tr></table>	$flag_{sign}$ 1	$index: 1$ to $(B_{02}-3)$ bits		
$flag_{zero}$ 0	<table><tr><td>$flag_{sign}$ 1</td><td>$index: 1$ to $(B_{02}-3)$ bits</td></tr></table>	$flag_{sign}$ 1	$index: 1$ to $(B_{02}-3)$ bits				
$flag_{sign}$ 1	$index: 1$ to $(B_{02}-3)$ bits						

FIG.2

3/8

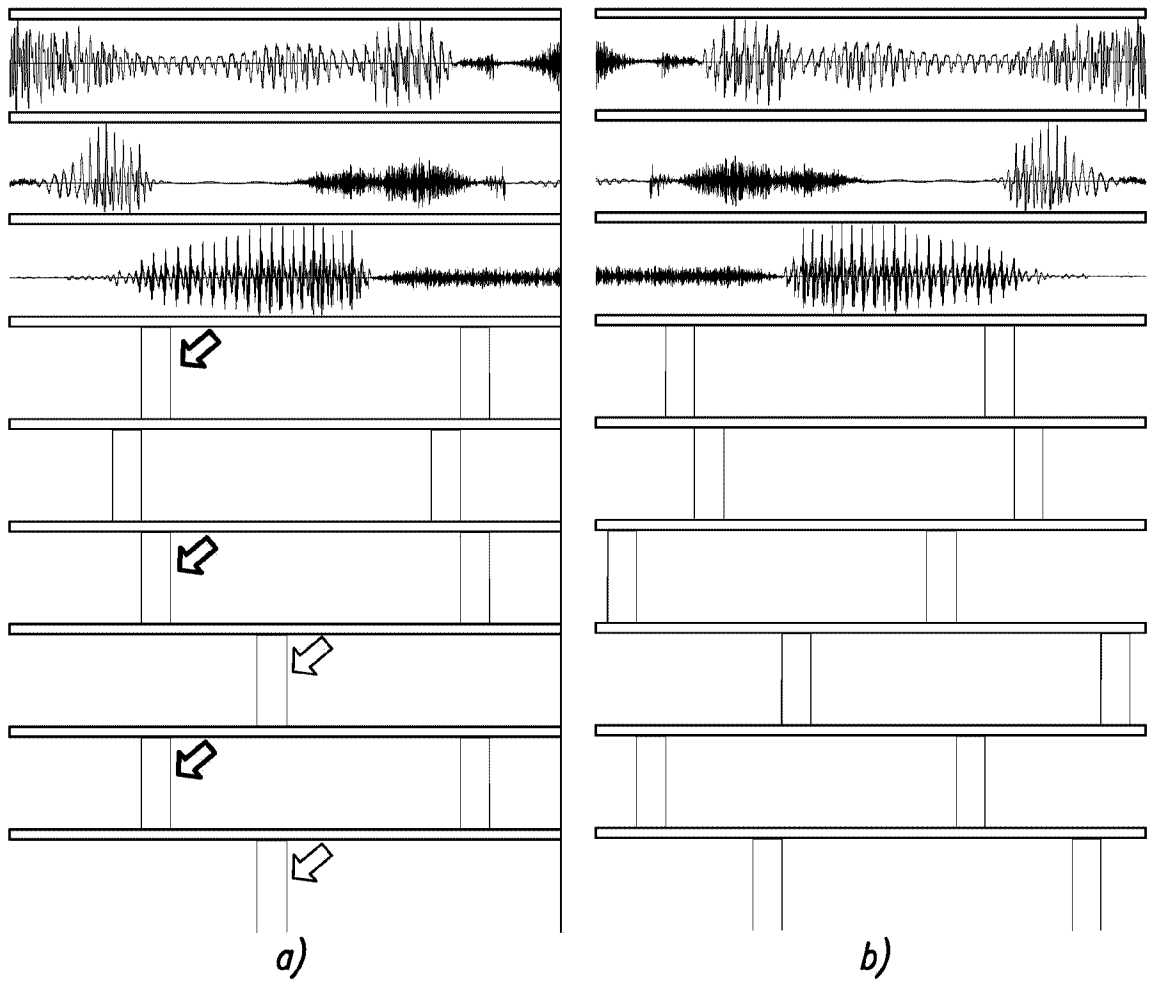


FIG. 3

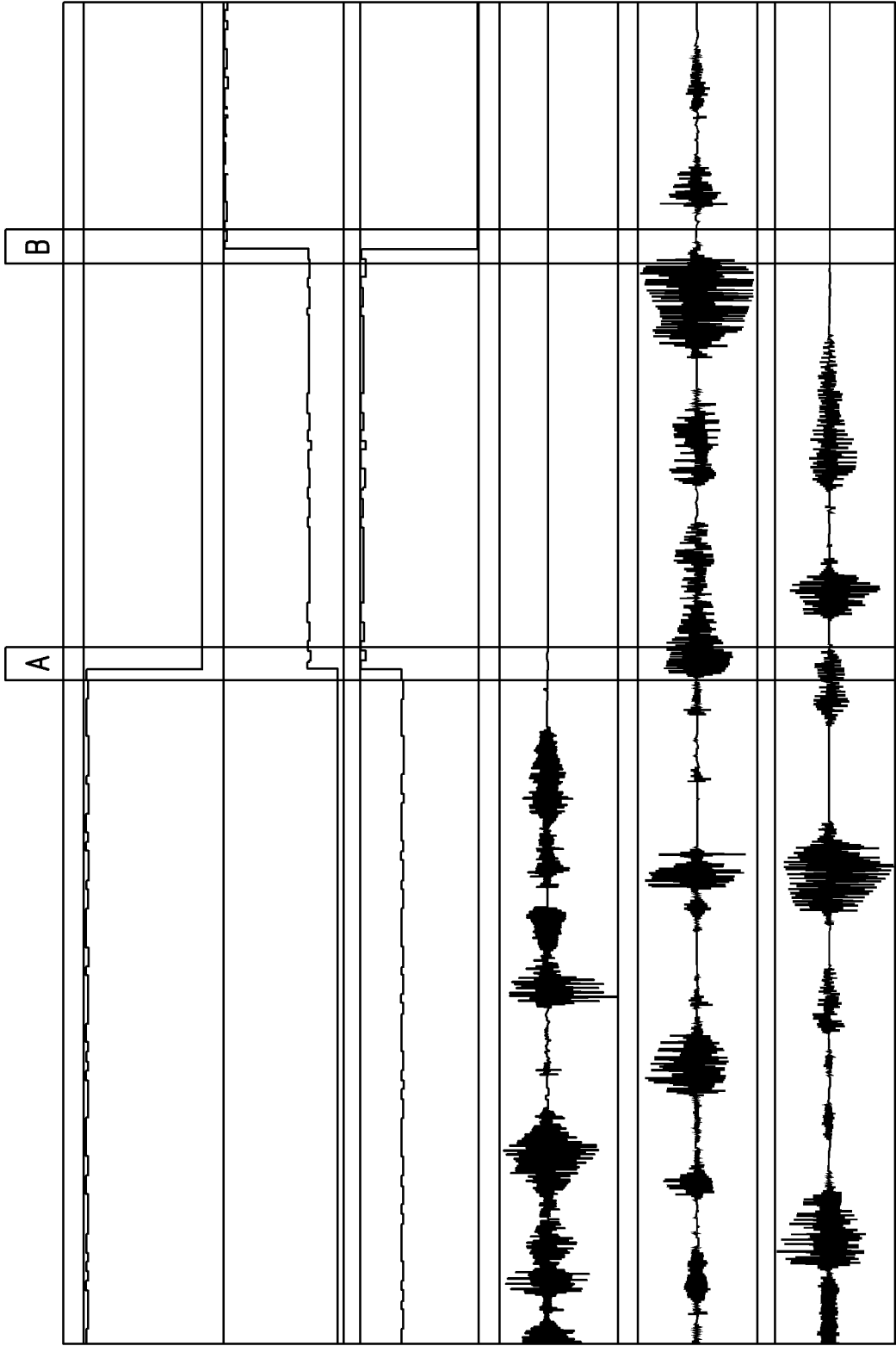


FIG.4

5/8

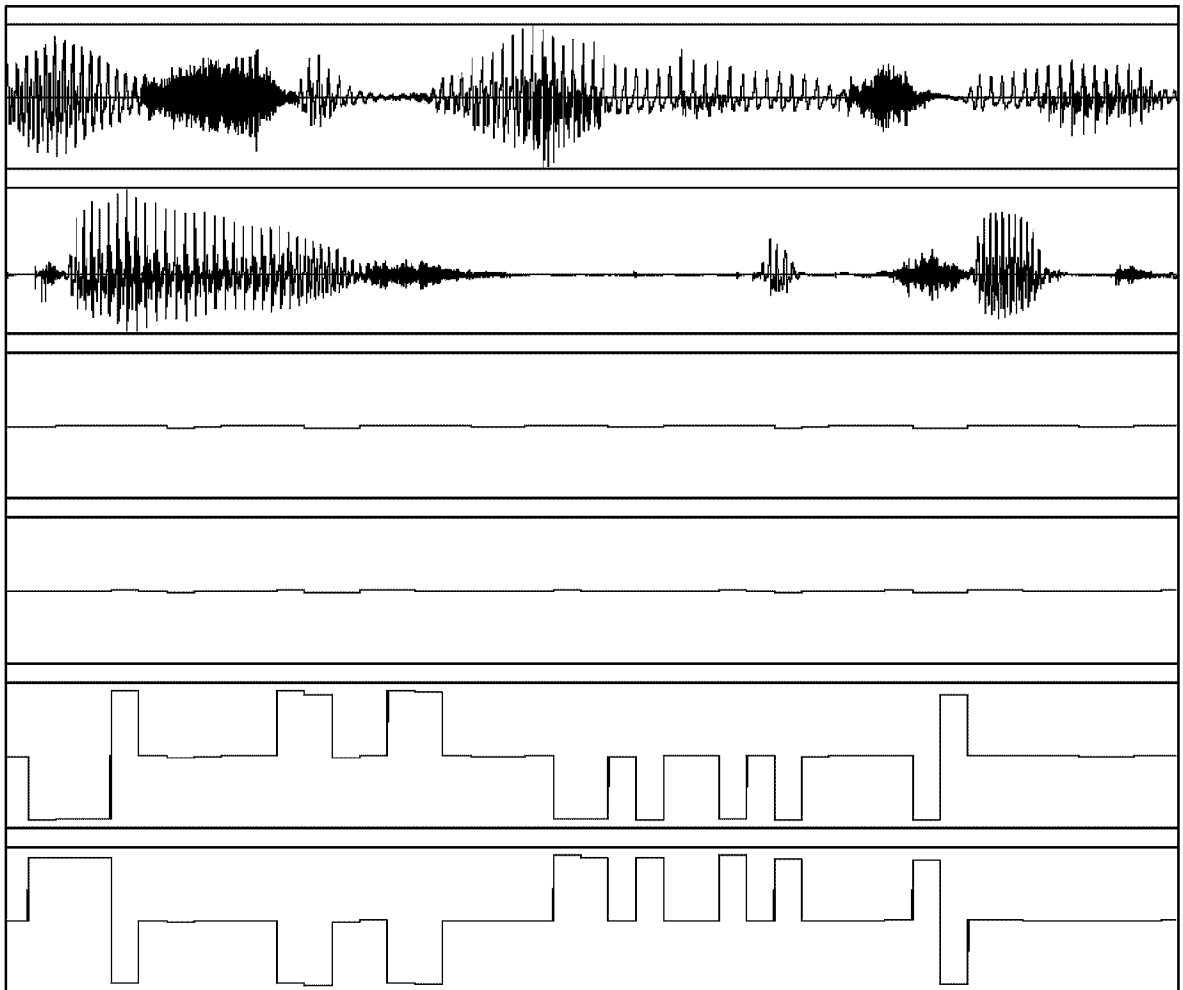
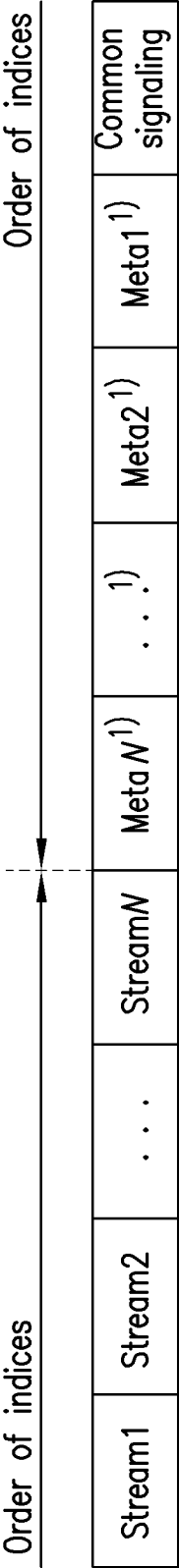


FIG.5



¹⁾Optional bits

FIG.6

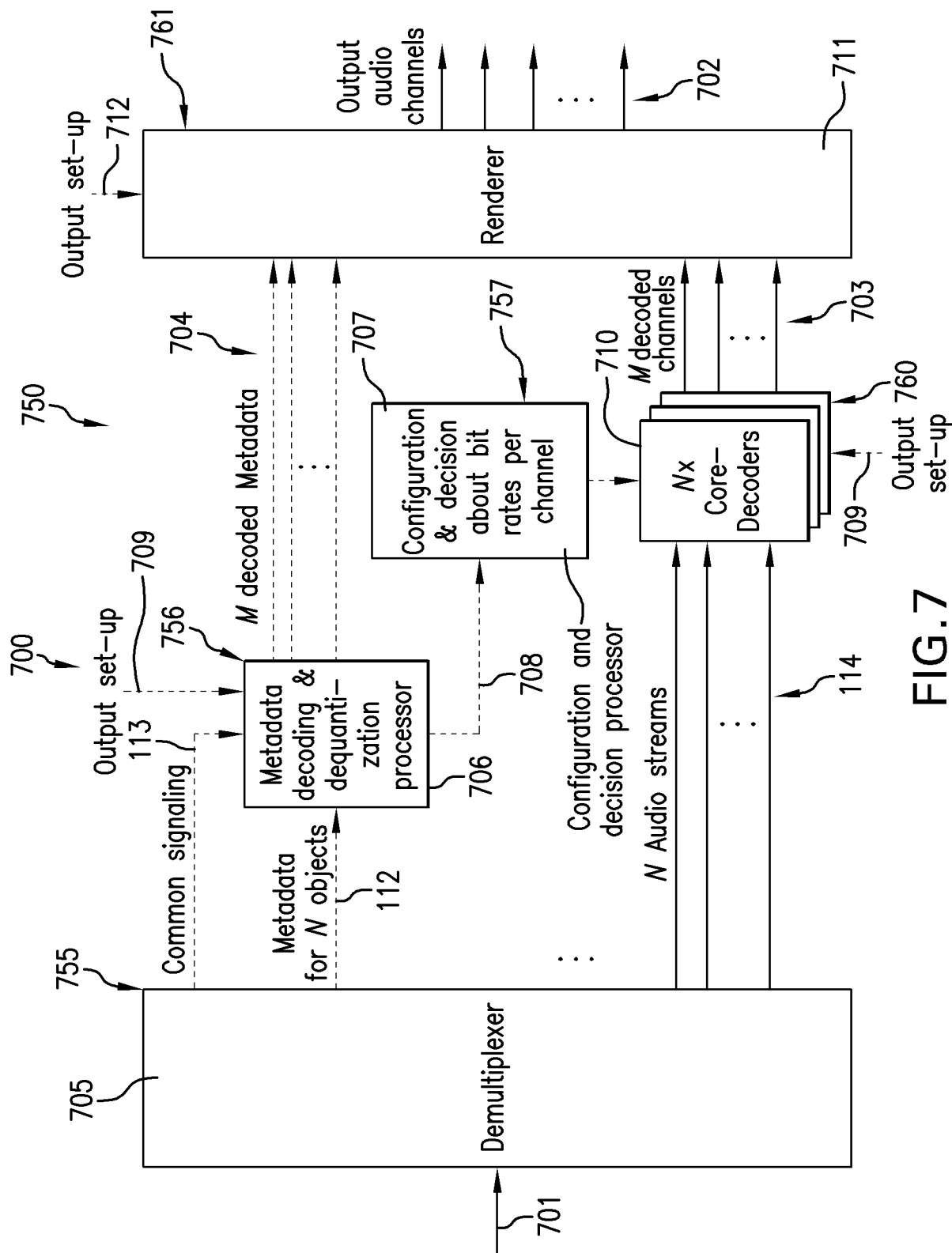


FIG. 7

8/8

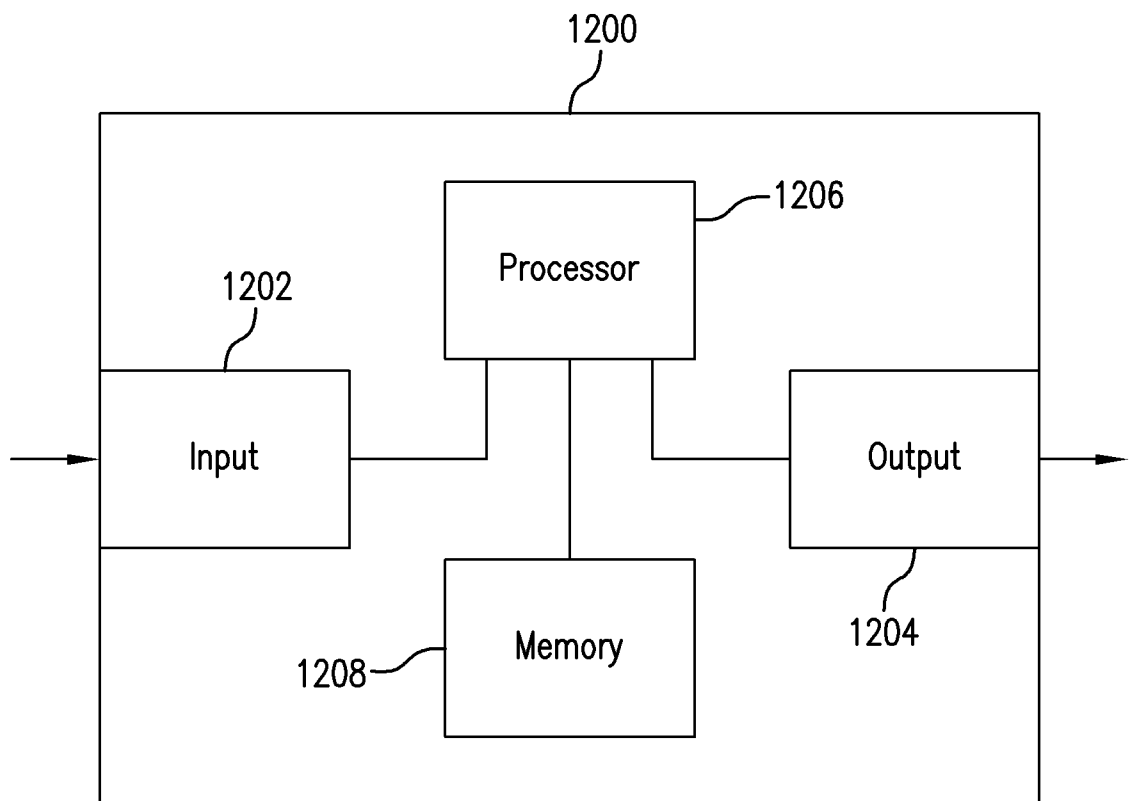


FIG.8

INTERNATIONAL SEARCH REPORT

International application No.

PCT/CA2020/050943A. CLASSIFICATION OF SUBJECT MATTER
IPC: **G10L 19/00** (2013.01)

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC: **G10L 19/00** (2013.01)

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic database(s) consulted during the international search (name of database(s) and, where practicable, search terms used)
Questel Orbit, Esp@cenet, Canadian Patent Database, USPTO Database, IEEE Xplore.

keywords: encode, decode, audio, stream, bit, budget, metadata.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	CA3074750 (Eksler, Vaclav) 28 March 2019 (28-03-2019) -see abstract; -see paragraphs 0007, 0011, 0029, 0032, 0034, 0040, 0045, 0048, 0049 and 0051; -see Figures 1-3.	1 and 32
A	US7657427 (Jelinek, Milan) 2 February 2010 (02-02-2010) -see whole document.	1-62

☐ Further documents are listed in the continuation of Box C.☒ See patent family annex.

* Special categories of cited documents:	"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A" document defining the general state of the art which is not considered to be of particular relevance	"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"D" document cited by the applicant in the international application	"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"E" earlier application or patent but published on or after the international filing date	"&" document member of the same patent family
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	
"O" document referring to an oral disclosure, use, exhibition or other means	
"P" document published prior to the international filing date but later than the priority date claimed	

Date of the actual completion of the international search
10 September 2020 (10-09-2020)Date of mailing of the international search report
09 October 2020 (09-10-2020)Name and mailing address of the ISA/CA
Canadian Intellectual Property Office
Place du Portage I, C114 - 1st Floor, Box PCT
50 Victoria Street
Gatineau, Quebec K1A 0C9
Facsimile No.: 819-953-2476

Authorized officer

Hassan Bayaa (819) 639-2301

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.

PCT/CA2020/050943

Patent Document Cited in Search Report	Publication Date	Patent Family Member(s)	Publication Date
CA3074750A1	28 March 2019 (28-03-2019)	AU2018337086A1	19 March 2020 (19-03-2020)
		AU2018338424A1	19 March 2020 (19-03-2020)
		CA3074749A1	28 March 2019 (28-03-2019)
		CN111133510A	08 May 2020 (08-05-2020)
		CN111149160A	12 May 2020 (12-05-2020)
		EP3685375A1	29 July 2020 (29-07-2020)
		EP3685376A1	29 July 2020 (29-07-2020)
		KR20200054221A	19 May 2020 (19-05-2020)
		KR20200055726A	21 May 2020 (21-05-2020)
		US2020243100A1	30 July 2020 (30-07-2020)
		WO2019056107A1	28 March 2019 (28-03-2019)
		WO2019056108A1	28 March 2019 (28-03-2019)
US7657427B2	02 February 2010 (02-02-2010)	US2005177364A1	11 August 2005 (11-08-2005)
		AT505786T	15 April 2011 (15-04-2011)
		AU2003278013A1	04 May 2004 (04-05-2004)
		AU2003278013A8	04 May 2004 (04-05-2004)
		AU2003278014A1	04 May 2004 (04-05-2004)
		AU2003278014A8	04 May 2004 (04-05-2004)
		BR0315179A	23 August 2005 (23-08-2005)
		BR0315216A	16 August 2005 (16-08-2005)
		CA2501368A1	22 April 2004 (22-04-2004)
		CA2501368C	25 June 2013 (25-06-2013)
		CA2501369A1	22 April 2004 (22-04-2004)
		CN1703736A	30 November 2005 (30-11-2005)
		CN1703737A	30 November 2005 (30-11-2005)
		CN1703737B	15 May 2013 (15-05-2013)
		DE60336744D1	26 May 2011 (26-05-2011)
		EG23923A	30 December 2007 (30-12-2007)
		EP1550108A2	06 July 2005 (06-07-2005)
		EP1554718A2	20 July 2005 (20-07-2005)
		EP1554718B1	13 April 2011 (13-04-2011)
		EP1808852A1	18 July 2007 (18-07-2007)
		ES2361154T3	14 June 2011 (14-06-2011)
		JP2006502426A	19 January 2006 (19-01-2006)
		JP2006502427A	19 January 2006 (19-01-2006)
		KR20050049537A	25 May 2005 (25-05-2005)
		KR100711280B1	25 April 2007 (25-04-2007)
		KR20050049538A	25 May 2005 (25-05-2005)
		MY134085A	30 November 2007 (30-11-2007)
		MY138212A	29 May 2009 (29-05-2009)
		RU2005113877A	10 October 2005 (10-10-2005)
		RU2331933C2	20 August 2008 (20-08-2008)
		RU2005113876A	10 October 2005 (10-10-2005)
		RU2351907C2	10 April 2009 (10-04-2009)
		US2005267746A1	01 December 2005 (01-12-2005)
		US7203638B2	10 April 2007 (10-04-2007)
		WO2004034376A2	22 April 2004 (22-04-2004)
		WO2004034376A3	10 June 2004 (10-06-2004)
		WO2004034379A2	22 April 2004 (22-04-2004)
		WO2004034379A3	23 December 2004 (23-12-2004)