(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
24 October 2013 (24.10.2013)

WIPO | PCT

(10) International Publication Number
WO 2013/156679 A1

(51) International Patent Classification:
H04N 7/26 (2006.01) *H04N 21/83* (2011.01)

(21) International Application Number:
PCT/FI2013/050419

(22) International Filing Date:
16 April 2013 (16.04.2013)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/624,932 16 April 2012 (16.04.2012) US

(71) Applicant: **NOKIA CORPORATION** [FI/FI]; Keilalahdentie 4, FI-02150 Espoo (FI).

(72) Inventors: **HANNUKSELA, Miska Matias**; Rusthollinrinne 2, FI-36110 Tampere (FI). **GOPALAKRISHNA, Srikanth Manchenahally**; c/o Nokia Corporation, Keilalahdentie 4, FI-02150 Espoo (FI).

(74) Agents: **NOKIA CORPORATION** et al.; IPR Department, Jussi Jaatinen, Keilalahdentie 4, FI-02150 Espoo (FI).

(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, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, 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, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, 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, 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, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report (Art. 21(3))*

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))*

WO 2013/156679 A1

(54) Title: METHOD AND APPARATUS FOR VIDEO CODING

(57) Abstract: There is disclosed a method, apparatus and computer program product in which a first parameter set is received and an identifier of the first parameter set is obtained. A second parameter set is also received. The validity of the first parameter set is determined on the basis of at least one of the following: receiving in the second parameter set a list of valid identifier values; and determining that the first parameter set is valid, if the identifier of the first parameter set is in the list of valid parameter values; receiving in the second parameter set an identifier of the second parameter set; and determining that the first parameter set is valid based on the identifier of the first parameter set and the identifier of the second parameter set. There is also disclosed a method, apparatus and computer program product in which a first parameter set is encoded and an identifier is attached to the first parameter set. A second parameter set is also encoded. The validity of the first parameter set is determined on the basis of at least one of the following: attaching the second parameter set a list of valid identifier values and determining that the first parameter set is valid, if the identifier of the first parameter set is in the list of valid parameter values; attaching in the second parameter set an identifier of the second parameter set and determining that the first parameter set is valid based on the identifier of the first parameter set and the identifier of the second parameter set.

METHOD AND APPARATUS FOR VIDEO CODING

TECHNICAL FIELD

The present application relates generally to an apparatus, a method and a computer program for video coding and decoding.

BACKGROUND

This section is intended to provide a background or context to the invention that is recited in the claims. The description herein may include concepts that could be pursued, but are not necessarily ones that have been previously conceived or pursued. Therefore, unless otherwise indicated herein, what is described in this section is not prior art to the description and claims in this application and is not admitted to be prior art by inclusion in this section.

In many video coding standards the syntax structures may be arranged in different layers, where a layer may be defined as one of a set of syntactical structures in a non-branching hierarchical relationship. Generally, higher layers may contain lower layers. The coding layers may consist for example of the coded video sequence, picture, slice, and treeblock layers. Some video coding standards introduce a concept of a parameter set. An instance of a parameter set may include all picture, group of pictures (GOP), and sequence level data such as picture size, display window, optional coding modes employed, macroblock allocation map, and others. Each parameter set instance may include a unique identifier. Each slice header may include a reference to a parameter set identifier, and the parameter values of the referred parameter set may be used when decoding the slice. Parameter sets may be used to decouple the transmission and decoding order of infrequently changing picture, GOP, and sequence level data from sequence, GOP, and picture boundaries. Parameter sets can be transmitted out-of-band using a reliable transmission protocol as long as they are decoded before they are referred. If parameter sets are transmitted in-band, they can be repeated multiple times to improve error resilience compared to conventional video coding schemes. The parameter sets may be transmitted at a session set-up time. However, in some systems, mainly broadcast ones, reliable out-of-band transmission of parameter sets may not be feasible, but rather parameter sets are conveyed in-band in Parameter Set NAL units.

SUMMARY

According to some example embodiments of the present invention there is provided methods, apparatuses and computer program products for transmitting and receiving parameter sets and providing identifiers for the parameter sets so that the identifiers enable determining the validity of the parameter sets. In some embodiments the parameter sets are adaptation parameter sets. In some embodiments identifier values of one or more parameter sets are used in determining whether the parameter set is valid.

Various aspects of examples of the invention are provided in the detailed description.

According to a first aspect of the present invention, there is provided a method comprising:

receiving a first parameter set;
obtaining an identifier of the first parameter set;
receiving a second parameter set;
determining the validity of the first parameter set on the basis of at least one of the following:

- receiving in the second parameter set a list of valid identifier values; and determining that the first parameter set is valid, if the identifier of the first parameter set is in the list of valid parameter values;
- receiving in the second parameter set an identifier of the second parameter set; and determining that the first parameter set is valid based on the identifier of the first parameter set and the identifier of the second parameter set.

According to a second aspect of the present invention there is provided a method comprising:
encoding a first parameter set;
attaching an identifier of the first parameter set to the first parameter set;
encoding a second parameter set;
determining the validity of the first parameter set on the basis of at least one of the following:

- attaching in the second parameter set a list of valid identifier values; and determining that the first parameter set is valid, if the identifier of the first parameter set is in the list of valid parameter values;
- attaching in the second parameter set an identifier of the second parameter set; and determining that the first parameter set is valid based on the identifier of the first parameter set and the identifier of the second parameter set.

According to a third aspect of the present invention there is provided an apparatus comprising at least one processor and at least one memory including computer program code, the at least one memory and the computer program code configured to, with the at least one processor, cause the apparatus to:
receive a first parameter set;
obtain an identifier of the first parameter set;
receive a second parameter set; and
determine the validity of the first parameter set on the basis of at least one of the following:

- by receiving in the second parameter set a list of valid identifier values; and determining that the first parameter set is valid, if the identifier of the first parameter set is in the list of valid parameter values;
- by receiving in the second parameter set an identifier of the second parameter set; and determining that the first parameter set is valid based on the identifier of the first parameter set and the identifier of the second parameter set.

According to a fourth aspect of the present invention there is provided an apparatus comprising at least one processor and at least one memory including computer program code, the at least one memory and the computer program code configured to, with the at least one processor, cause the apparatus to:

encode a first parameter set;

attach an identifier of the first parameter set to the first parameter set;

encode a second parameter set; and

determine the validity of the first parameter set on the basis of at least one of the following:

- by attaching in the second parameter set a list of valid identifier values; and determining that the first parameter set is valid, if the identifier of the first parameter set is in the list of valid parameter values;
- by attaching in the second parameter set an identifier of the second parameter set; and determining that the first parameter set is valid based on the identifier of the first parameter set and the identifier of the second parameter set.

According to a fifth aspect of the present invention there is provided a computer program product including one or more sequences of one or more instructions which, when executed by one or more processors, cause an apparatus to at least perform the following:

receive a first parameter set;

obtain an identifier of the first parameter set;

receive a second parameter set; determining the validity of the first parameter set on the basis of at least one of the following:

- receiving in the second parameter set a list of valid identifier values; and determining that the first parameter set is valid, if the identifier of the first parameter set is in the list of valid parameter values;
- receiving in the second parameter set an identifier of the second parameter set; and determining that the first parameter set is valid based on the identifier of the first parameter set and the identifier of the second parameter set.

According to a sixth aspect of the present invention there is provided a computer program product including one or more sequences of one or more instructions which, when executed by one or more processors, cause an apparatus to at least perform the following:

encode a first parameter set;

attach an identifier of the first parameter set;

encode a second parameter set; determine the validity of the first parameter set on the basis of at least one of the following:

- by attaching in the second parameter set a list of valid identifier values; and determining that the first parameter set is valid, if the identifier of the first parameter set is in the list of valid parameter values;
- by attaching in the second parameter set an identifier of the second parameter set; and determining that the first parameter set is valid based on the identifier of the first parameter set and the identifier of the second parameter set.

According to a seventh aspect of the present invention there is provided an apparatus comprising:
means for receiving a first parameter set;
means for obtaining an identifier of the first parameter set;
means for receiving a second parameter set; means for determining the validity of the first parameter set on the basis of at least one of the following:

- by receiving in the second parameter set a list of valid identifier values; and determining that the first parameter set is valid, if the identifier of the first parameter set is in the list of valid parameter values;
- by receiving in the second parameter set an identifier of the second parameter set; and determining that the first parameter set is valid based on the identifier of the first parameter set and the identifier of the second parameter set.

According to an eighth aspect of the present invention there is provided an apparatus comprising:
means for encoding a first parameter set;
means for attaching an identifier of the first parameter set;
means for encoding a second parameter set; and
means for determining the validity of the first parameter set on the basis of at least one of the following:

- by attaching in the second parameter set a list of valid identifier values; and determining that the first parameter set is valid, if the identifier of the first parameter set is in the list of valid parameter values;
- by attaching in the second parameter set an identifier of the second parameter set; and determining that the first parameter set is valid based on the identifier of the first parameter set and the identifier of the second parameter set.

According to a ninth aspect of the present invention there is provided a video decoder configured for:
receiving a first parameter set;
obtaining an identifier of the first parameter set;

receiving a second parameter set; determining the validity of the first parameter set on the basis of at least one of the following:

- receiving in the second parameter set a list of valid identifier values; and determining that the first parameter set is valid, if the identifier of the first parameter set is in the list of valid parameter values;
- receiving in the second parameter set an identifier of the second parameter set; and determining that the first parameter set is valid based on the identifier of the first parameter set and the identifier of the second parameter set.

According to a tenth aspect of the present invention there is provided a video encoder configured for:

encoding a first parameter set;

attaching an identifier of the first parameter set to the first parameter set;

encoding a second parameter set;

determining the validity of the first parameter set on the basis of at least one of the following:

- attaching in the second parameter set a list of valid identifier values; and determining that the first parameter set is valid, if the identifier of the first parameter set is in the list of valid parameter values;
- attaching in the second parameter set an identifier of the second parameter set; and determining that the first parameter set is valid based on the identifier of the first parameter set and the identifier of the second parameter set.

BRIEF DESCRIPTION OF THE DRAWINGS

For a more complete understanding of example embodiments of the present invention, reference is now made to the following descriptions taken in connection with the accompanying drawings in which:

Figure 1 shows schematically an electronic device employing some embodiments of the invention;

Figure 2 shows schematically a user equipment suitable for employing some embodiments of the invention;

Figure 3 further shows schematically electronic devices employing embodiments of the invention connected using wireless and wired network connections;

Figure 4a shows schematically an embodiment of the invention as incorporated within an encoder;

Figure 4b shows schematically an embodiment of an inter predictor according to some embodiments of the invention;

Figure 5 shows a simplified model of a DIBR-based 3DV system;

Figure 6 shows a simplified 2D model of a stereoscopic camera setup;

Figure 7 shows an example of definition and coding order of access units;

Figure 8 shows a high level flow chart of an embodiment of an encoder capable of encoding texture views and depth views; and

Figure 9 shows a high level flow chart of an embodiment of a decoder capable of decoding texture views and depth views.

DETAILED DESCRIPTON OF SOME EXAMPLE EMBODIMENTS

In the following, several embodiments of the invention will be described in the context of one video coding arrangement. It is to be noted, however, that the invention is not limited to this particular arrangement. In fact, the different embodiments have applications widely in any environment where improvement of reference picture handling is required. For example, the invention may be applicable to video coding systems like streaming systems, DVD players, digital television receivers, personal video recorders, systems and computer programs on personal computers, handheld computers and communication devices, as well as network elements such as transcoders and cloud computing arrangements where video data is handled.

The H.264/AVC standard was developed by the Joint Video Team (JVT) of the Video Coding Experts Group (VCEG) of the Telecommunications Standardization Sector of International Telecommunication Union (ITU-T) and the Moving Picture Experts Group (MPEG) of International Organisation for Standardization (ISO) / International Electrotechnical Commission (IEC). The H.264/AVC standard is published by both parent standardization organizations, and it is referred to as ITU-T Recommendation H.264 and ISO/IEC International Standard 14496-10, also known as MPEG-4 Part 10 Advanced Video Coding (AVC). There have been multiple versions of the H.264/AVC standard, each integrating new extensions or features to the specification. These extensions include Scalable Video Coding (SVC) and Multiview Video Coding (MVC).

There is a currently ongoing standardization project of High Efficiency Video Coding (HEVC) by the Joint Collaborative Team – Video Coding (JCT-VC) of VCEG and MPEG.

Some key definitions, bitstream and coding structures, and concepts of H.264/AVC and HEVC are described in this section as an example of a video encoder, decoder, encoding method, decoding method, and a bitstream structure, wherein the embodiments may be implemented. Some of the key definitions, bitstream and coding structures, and concepts of H.264/AVC are the same as in a draft HEVC standard – hence, they are described below jointly. The aspects of the invention are not limited to H.264/AVC or HEVC, but rather the description is given for one possible basis on top of which the invention may be partly or fully realized.

Similarly to many earlier video coding standards, the bitstream syntax and semantics as well as the decoding process for error-free bitstreams are specified in H.264/AVC and HEVC. The encoding process is not specified, but encoders must generate conforming bitstreams. Bitstream and decoder

conformance can be verified with the Hypothetical Reference Decoder (HRD). The standards contain coding tools that help in coping with transmission errors and losses, but the use of the tools in encoding is optional and no decoding process has been specified for erroneous bitstreams.

The elementary unit for the input to an H.264/AVC or HEVC encoder and the output of an H.264/AVC or HEVC decoder, respectively, is a picture. In H.264/AVC and HEVC, a picture may either be a frame or a field. A frame comprises a matrix of luma samples and corresponding chroma samples. A field is a set of alternate sample rows of a frame and may be used as encoder input, when the source signal is interlaced. Chroma pictures may be subsampled when compared to luma pictures. For example, in the 4:2:0 sampling pattern the spatial resolution of chroma pictures is half of that of the luma picture along both coordinate axes.

In H.264/AVC, a macroblock is a 16x16 block of luma samples and the corresponding blocks of chroma samples. For example, in the 4:2:0 sampling pattern, a macroblock contains one 8x8 block of chroma samples per each chroma component. In H.264/AVC, a picture is partitioned to one or more slice groups, and a slice group contains one or more slices. In H.264/AVC, a slice consists of an integer number of macroblocks ordered consecutively in the raster scan within a particular slice group.

In a draft HEVC standard, video pictures are divided into coding units (CU) covering the area of the picture. A CU consists of one or more prediction units (PU) defining the prediction process for the samples within the CU and one or more transform units (TU) defining the prediction error coding process for the samples in the CU. Typically, a CU consists of a square block of samples with a size selectable from a predefined set of possible CU sizes. A CU with the maximum allowed size is typically named as LCU (largest coding unit) and the video picture is divided into non-overlapping LCUs. An LCU can be further split into a combination of smaller CUs, e.g. by recursively splitting the LCU and resultant CUs. Each resulting CU typically has at least one PU and at least one TU associated with it. Each PU and TU can further be split into smaller PUs and TUs in order to increase granularity of the prediction and prediction error coding processes, respectively. The PU splitting can be realized by splitting the CU into four equal size square PUs or splitting the CU into two rectangle PUs vertically or horizontally in a symmetric or asymmetric way. The division of the image into CUs, and division of CUs into PUs and TUs is typically signalled in the bitstream allowing the decoder to reproduce the intended structure of these units.

In a draft HEVC standard, a picture can be partitioned in tiles, which are rectangular and contain an integer number of LCUs. In a draft HEVC standard, the partitioning to tiles forms a regular grid, where heights and widths of tiles differ from each other by one LCU at the maximum. In a draft HEVC, a slice consists of an integer number of CUs. The CUs are scanned in the raster scan order of LCUs within tiles or within a picture, if tiles are not in use. Within an LCU, the CUs have a specific scan order.

In a Working Draft (WD) 5 of HEVC, some key definitions and concepts for picture partitioning are defined as follows. A partitioning is defined as the division of a set into subsets such that each element of the set is in exactly one of the subsets.

A basic coding unit in a HEVC WD5 is a treeblock. A treeblock is an NxN block of luma samples and two corresponding blocks of chroma samples of a picture that has three sample arrays, or an NxN block of samples of a monochrome picture or a picture that is coded using three separate colour planes. A treeblock may be partitioned for different coding and decoding processes. A treeblock partition is a block of luma samples and two corresponding blocks of chroma samples resulting from a partitioning of a treeblock for a picture that has three sample arrays or a block of luma samples resulting from a partitioning of a treeblock for a monochrome picture or a picture that is coded using three separate colour planes. Each treeblock is assigned a partition signalling to identify the block sizes for intra or inter prediction and for transform coding. The partitioning is a recursive quadtree partitioning. The root of the quadtree is associated with the treeblock. The quadtree is split until a leaf is reached, which is referred to as the coding node. The coding node is the root node of two trees, the prediction tree and the transform tree. The prediction tree specifies the position and size of prediction blocks. The prediction tree and associated prediction data are referred to as a prediction unit. The transform tree specifies the position and size of transform blocks. The transform tree and associated transform data are referred to as a transform unit. The splitting information for luma and chroma is identical for the prediction tree and may or may not be identical for the transform tree. The coding node and the associated prediction and transform units form together a coding unit.

In a HEVC WD5, pictures are divided into slices and tiles. A slice may be a sequence of treeblocks but (when referring to a so-called fine granular slice) may also have its boundary within a treeblock at a location where a transform unit and prediction unit coincide. Treeblocks within a slice are coded and decoded in a raster scan order. For the primary coded picture, the division of each picture into slices is a partitioning.

In a HEVC WD5, a tile is defined as an integer number of treeblocks co-occurring in one column and one row, ordered consecutively in the raster scan within the tile. For the primary coded picture, the division of each picture into tiles is a partitioning. Tiles are ordered consecutively in the raster scan within the picture. Although a slice contains treeblocks that are consecutive in the raster scan within a tile, these treeblocks are not necessarily consecutive in the raster scan within the picture. Slices and tiles need not contain the same sequence of treeblocks. A tile may comprise treeblocks contained in more than one slice. Similarly, a slice may comprise treeblocks contained in several tiles.

In H.264/AVC and HEVC, in-picture prediction may be disabled across slice boundaries. Thus, slices can be regarded as a way to split a coded picture into independently decodable pieces, and slices are therefore often regarded as elementary units for transmission. In many cases, encoders may indicate in the bitstream which types of in-picture prediction are turned off across slice boundaries, and the decoder operation takes this information into account for example when concluding which prediction sources are available. For example, samples from a neighboring macroblock or CU may be regarded as unavailable for intra prediction, if the neighboring macroblock or CU resides in a different slice.

A syntax element may be defined as an element of data represented in the bitstream. A syntax structure may be defined as zero or more syntax elements present together in the bitstream in a specified order.

The elementary unit for the output of an H.264/AVC or HEVC encoder and the input of an H.264/AVC or HEVC decoder, respectively, is a Network Abstraction Layer (NAL) unit. For transport over packet-oriented networks or storage into structured files, NAL units may be encapsulated into packets or similar structures. A bytestream format has been specified in H.264/AVC and HEVC for transmission or storage environments that do not provide framing structures. The bytestream format separates NAL units from each other by attaching a start code in front of each NAL unit. To avoid false detection of NAL unit boundaries, encoders run a byte-oriented start code emulation prevention algorithm, which adds an emulation prevention byte to the NAL unit payload if a start code would have occurred otherwise. In order to enable straightforward gateway operation between packet- and stream-oriented systems, start code emulation prevention may always be performed regardless of whether the bytestream format is in use or not. A NAL unit may be defined as a syntax structure containing an indication of the type of data to follow and bytes containing that data in the form of an RBSP interspersed as necessary with emulation prevention bytes. A raw byte sequence payload (RBSP) may be defined as a syntax structure containing an integer number of bytes that is encapsulated in a NAL unit. An RBSP is either empty or has the form of a string of data bits containing syntax elements followed by an RBSP stop bit and followed by zero or more subsequent bits equal to 0.

NAL units consist of a header and payload. In H.264/AVC and HEVC, the NAL unit header indicates the type of the NAL unit and whether a coded slice contained in the NAL unit is a part of a reference picture or a non-reference picture. H.264/AVC includes a 2-bit nal_ref_idc syntax element, which when equal to 0 indicates that a coded slice contained in the NAL unit is a part of a non-reference picture and when greater than 0 indicates that a coded slice contained in the NAL unit is a part of a reference picture. A draft HEVC standard includes a 1-bit nal_ref_idc syntax element, also known as nal_ref_flag, which when equal to 0 indicates that a coded slice contained in the NAL unit is a part of a non-reference picture and when equal to 1 indicates that a coded slice contained in the NAL unit is a part of a reference picture. The header for SVC and MVC NAL units may additionally contain various indications related to the scalability and multiview hierarchy. In HEVC, the NAL unit header includes the temporal_id syntax element, which specifies a temporal identifier for the NAL unit.

NAL units can be categorized into Video Coding Layer (VCL) NAL units and non-VCL NAL units. VCL NAL units are typically coded slice NAL units. In H.264/AVC, coded slice NAL units contain syntax elements representing one or more coded macroblocks, each of which corresponds to a block of samples in the uncompressed picture. In HEVC, coded slice NAL units contain syntax elements representing one or more CU. In H.264/AVC and HEVC a coded slice NAL unit can be indicated to be a coded slice in an Instantaneous Decoding Refresh (IDR) picture or coded slice in a non-IDR picture. In

HEVC, a coded slice NAL unit can be indicated to be a coded slice in a Clean Decoding Refresh (CDR) picture (which may also be referred to as a Clean Random Access picture or a CRA picture).

A non-VCL NAL unit may be for example one of the following types: a sequence parameter set, a picture parameter set, a supplemental enhancement information (SEI) NAL unit, an access unit delimiter, an end of sequence NAL unit, an end of stream NAL unit, or a filler data NAL unit. Parameter sets may be needed for the reconstruction of decoded pictures, whereas many of the other non-VCL NAL units are not necessary for the reconstruction of decoded sample values.

Parameters that remain unchanged through a coded video sequence may be included in a sequence parameter set. In addition to the parameters that may be needed by the decoding process, the sequence parameter set may optionally contain video usability information (VUI), which includes parameters that may be important for buffering, picture output timing, rendering, and resource reservation. There are three NAL units specified in H.264/AVC to carry sequence parameter sets: the sequence parameter set NAL unit containing all the data for H.264/AVC VCL NAL units in the sequence, the sequence parameter set extension NAL unit containing the data for auxiliary coded pictures, and the subset sequence parameter set for MVC and SVC VCL NAL units. A picture parameter set contains such parameters that are likely to be unchanged in several coded pictures.

In a draft HEVC, there is also a third type of parameter sets, here referred to as an Adaptation Parameter Set (APS), which includes parameters that are likely to be unchanged in several coded slices but may change for example for each picture or each few pictures. In a draft HEVC, the APS syntax structure includes parameters or syntax elements related to quantization matrices (QM), adaptive sample offset (SAO), adaptive loop filtering (ALF), and deblocking filtering. In a draft HEVC, an APS is a NAL unit and coded without reference or prediction from any other NAL unit. An identifier, referred to as aps_id syntax element, is included in APS NAL unit, and included and used in the slice header to refer to a particular APS.

H.264/AVC and HEVC syntax allows many instances of parameter sets, and each instance is identified with a unique identifier. In order to limit the memory usage needed for parameter sets, the value range for parameter set identifiers has been limited. In H.264/AVC and a draft HEVC standard, each slice header includes the identifier of the picture parameter set that is active for the decoding of the picture that contains the slice, and each picture parameter set contains the identifier of the active sequence parameter set. In a HEVC standard, a slice header additionally contains an APS identifier. Consequently, the transmission of picture and sequence parameter sets does not have to be accurately synchronized with the transmission of slices. Instead, it is sufficient that the active sequence and picture parameter sets are received at any moment before they are referenced, which allows transmission of parameter sets "out-of-band" using a more reliable transmission mechanism compared to the protocols used for the slice data. For example, parameter sets can be included as a parameter in the session description for Real-time Transport Protocol (RTP) sessions. If parameter sets are transmitted in-band, they can be repeated to improve error robustness.

A SEI NAL unit may contain one or more SEI messages, which are not required for the decoding of output pictures but may assist in related processes, such as picture output timing, rendering, error detection, error concealment, and resource reservation. Several SEI messages are specified in H.264/AVC and HEVC, and the user data SEI messages enable organizations and companies to specify SEI messages for their own use. H.264/AVC and HEVC contain the syntax and semantics for the specified SEI messages but no process for handling the messages in the recipient is defined. Consequently, encoders are required to follow the H.264/AVC standard or the HEVC standard when they create SEI messages, and decoders conforming to the H.264/AVC standard or the HEVC standard, respectively, are not required to process SEI messages for output order conformance. One of the reasons to include the syntax and semantics of SEI messages in H.264/AVC and HEVC is to allow different system specifications to interpret the supplemental information identically and hence interoperate. It is intended that system specifications can require the use of particular SEI messages both in the encoding end and in the decoding end, and additionally the process for handling particular SEI messages in the recipient can be specified.

A coded picture is a coded representation of a picture. A coded picture in H.264/AVC comprises the VCL NAL units that are required for the decoding of the picture. In H.264/AVC, a coded picture can be a primary coded picture or a redundant coded picture. A primary coded picture is used in the decoding process of valid bitstreams, whereas a redundant coded picture is a redundant representation that should only be decoded when the primary coded picture cannot be successfully decoded. In a draft HEVC, no redundant coded picture has been specified.

In H.264/AVC and HEVC, an access unit comprises a primary coded picture and those NAL units that are associated with it. In H.264/AVC, the appearance order of NAL units within an access unit is constrained as follows. An optional access unit delimiter NAL unit may indicate the start of an access unit. It is followed by zero or more SEI NAL units. The coded slices of the primary coded picture appear next. In H.264/AVC, the coded slice of the primary coded picture may be followed by coded slices for zero or more redundant coded pictures. A redundant coded picture is a coded representation of a picture or a part of a picture. A redundant coded picture may be decoded if the primary coded picture is not received by the decoder for example due to a loss in transmission or a corruption in physical storage medium.

In H.264/AVC, an access unit may also include an auxiliary coded picture, which is a picture that supplements the primary coded picture and may be used for example in the display process. An auxiliary coded picture may for example be used as an alpha channel or alpha plane specifying the transparency level of the samples in the decoded pictures. An alpha channel or plane may be used in a layered composition or rendering system, where the output picture is formed by overlaying pictures being at least partly transparent on top of each other. An auxiliary coded picture has the same syntactic and semantic restrictions as a monochrome redundant coded picture. In H.264/AVC, an auxiliary coded picture contains the same number of macroblocks as the primary coded picture.

A coded video sequence is defined to be a sequence of consecutive access units in decoding order from an IDR access unit, inclusive, to the next IDR access unit, exclusive, or to the end of the bitstream, whichever appears earlier.

A group of pictures (GOP) and its characteristics may be defined as follows. A GOP can be decoded regardless of whether any previous pictures were decoded. An open GOP is such a group of pictures in which pictures preceding the initial intra picture in output order might not be correctly decodable when the decoding starts from the initial intra picture of the open GOP. In other words, pictures of an open GOP may refer (in inter prediction) to pictures belonging to a previous GOP. An H.264/AVC decoder can recognize an intra picture starting an open GOP from the recovery point SEI message in an H.264/AVC bitstream. An HEVC decoder can recognize an intra picture starting an open GOP, because a specific NAL unit type, CRA NAL unit type, is used for its coded slices. A closed GOP is such a group of pictures in which all pictures can be correctly decoded when the decoding starts from the initial intra picture of the closed GOP. In other words, no picture in a closed GOP refers to any pictures in previous GOPs. In H.264/AVC and HEVC, a closed GOP starts from an IDR access unit. As a result, closed GOP structure has more error resilience potential in comparison to the open GOP structure, however at the cost of possible reduction in the compression efficiency. Open GOP coding structure is potentially more efficient in the compression, due to a larger flexibility in selection of reference pictures.

The bitstream syntax of H.264/AVC and HEVC indicates whether a particular picture is a reference picture for inter prediction of any other picture. Pictures of any coding type (I, P, B) can be reference pictures or non-reference pictures in H.264/AVC and HEVC. The NAL unit header indicates the type of the NAL unit and whether a coded slice contained in the NAL unit is a part of a reference picture or a non-reference picture.

Many hybrid video codecs, including H.264/AVC and HEVC, encode video information in two phases. In the first phase, pixel or sample values in a certain picture area or "block" are predicted. These pixel or sample values can be predicted, for example, by motion compensation mechanisms, which involve finding and indicating an area in one of the previously encoded video frames that corresponds closely to the block being coded. Additionally, pixel or sample values can be predicted by spatial mechanisms which involve finding and indicating a spatial region relationship.

Prediction approaches using image information from a previously coded image can also be called as inter prediction methods which may also be referred to as temporal prediction and motion compensation. Prediction approaches using image information within the same image can also be called as intra prediction methods.

The second phase is one of coding the error between the predicted block of pixels or samples and the original block of pixels or samples. This may be accomplished by transforming the difference in pixel or sample values using a specified transform. This transform may be a Discrete Cosine Transform (DCT) or a variant thereof. After transforming the difference, the transformed difference is quantized and entropy encoded.

By varying the fidelity of the quantization process, the encoder can control the balance between the accuracy of the pixel or sample representation (i.e. the visual quality of the picture) and the size of the resulting encoded video representation (i.e. the file size or transmission bit rate).

The decoder reconstructs the output video by applying a prediction mechanism similar to that used by the encoder in order to form a predicted representation of the pixel or sample blocks (using the motion or spatial information created by the encoder and stored in the compressed representation of the image) and prediction error decoding (the inverse operation of the prediction error coding to recover the quantized prediction error signal in the spatial domain).

After applying pixel or sample prediction and error decoding processes the decoder combines the prediction and the prediction error signals (the pixel or sample values) to form the output video frame.

The decoder (and encoder) may also apply additional filtering processes in order to improve the quality of the output video before passing it for display and/or storing as a prediction reference for the forthcoming pictures in the video sequence.

In many video codecs, including H.264/AVC and HEVC, motion information is indicated by motion vectors associated with each motion compensated image block. Each of these motion vectors represents the displacement of the image block in the picture to be coded (in the encoder) or decoded (at the decoder) and the prediction source block in one of the previously coded or decoded images (or pictures). H.264/AVC and HEVC, as many other video compression standards, divide a picture into a mesh of rectangles, for each of which a similar block in one of the reference pictures is indicated for inter prediction. The location of the prediction block is coded as a motion vector that indicates the position of the prediction block relative to the block being coded.

Inter prediction process may be characterized using one or more of the following factors.

The accuracy of motion vector representation. For example, motion vectors may be of quarter-pixel accuracy, and sample values in fractional-pixel positions may be obtained using a finite impulse response (FIR) filter.

Block partitioning for inter prediction. Many coding standards, including H.264/AVC and HEVC, allow selection of the size and shape of the block for which a motion vector is applied for motion-compensated prediction in the encoder, and indicating the selected size and shape in the bitstream so that decoders can reproduce the motion-compensated prediction done in the encoder.

Number of reference pictures for inter prediction. The sources of inter prediction are previously decoded pictures. Many coding standards, including H.264/AVC and HEVC, enable storage of multiple reference pictures for inter prediction and selection of the used reference picture on a block basis. For example, reference pictures may be selected on macroblock or macroblock partition basis in H.264/AVC and on PU or CU basis in HEVC. Many coding standards, such as H.264/AVC and HEVC, include syntax structures in the bitstream that enable decoders to create one or more reference picture lists. A reference picture index to a reference picture list may be used to indicate which one of the multiple reference pictures is used for inter prediction for a particular block. A reference picture index may be

coded by an encoder into the bitstream is some inter coding modes or it may be derived (by an encoder and a decoder) for example using neighboring blocks in some other inter coding modes.

Motion vector prediction. In order to represent motion vectors efficiently in bitstreams, motion vectors may be coded differentially with respect to a block-specific predicted motion vector. In many video codecs, the predicted motion vectors are created in a predefined way, for example by calculating the median of the encoded or decoded motion vectors of the adjacent blocks. Another way to create motion vector predictions is to generate a list of candidate predictions from adjacent blocks and/or co-located blocks in temporal reference pictures and signalling the chosen candidate as the motion vector predictor. In addition to predicting the motion vector values, the reference index of previously coded/decoded picture can be predicted. The reference index is typically predicted from adjacent blocks and/or co-located blocks in temporal reference picture. Differential coding of motion vectors is typically disabled across slice boundaries.

Multi-hypothesis motion-compensated prediction. H.264/AVC and HEVC enable the use of a single prediction block in P slices (herein referred to as uni-predictive slices) or a linear combination of two motion-compensated prediction blocks for bi-predictive slices, which are also referred to as B slices. Individual blocks in B slices may be bi-predicted, uni-predicted, or intra-predicted, and individual blocks in P slices may be uni-predicted or intra-predicted. The reference pictures for a bi-predictive picture may not be limited to be the subsequent picture and the previous picture in output order, but rather any reference pictures may be used. In many coding standards, such as H.264/AVC and HEVC, one reference picture list, referred to as reference picture list 0, is constructed for P slices, and two reference picture lists, list 0 and list 1, are constructed for B slices. For B slices, when prediction in forward direction may refer to prediction from a reference picture in reference picture list 0, and prediction in backward direction may refer to prediction from a reference picture in reference picture list 1, even though the reference pictures for prediction may have any decoding or output order relation to each other or to the current picture.

Weighted prediction. Many coding standards use a prediction weight of 1 for prediction blocks of inter (P) pictures and 0.5 for each prediction block of a B picture (resulting into averaging). H.264/AVC allows weighted prediction for both P and B slices. In implicit weighted prediction, the weights are proportional to picture order counts, while in explicit weighted prediction, prediction weights are explicitly indicated.

In many video codecs, the prediction residual after motion compensation is first transformed with a transform kernel (like DCT) and then coded. The reason for this is that often there still exists some correlation among the residual and transform can in many cases help reduce this correlation and provide more efficient coding.

In a draft HEVC, each PU has prediction information associated with it defining what kind of a prediction is to be applied for the pixels within that PU (e.g. motion vector information for inter predicted PUs and intra prediction directionality information for intra predicted PUs). Similarly each TU is

associated with information describing the prediction error decoding process for the samples within the TU (including e.g. DCT coefficient information). It may be signalled at CU level whether prediction error coding is applied or not for each CU. In the case there is no prediction error residual associated with the CU, it can be considered there are no TUs for the CU.

In some coding formats and codecs, a distinction is made between so-called short-term and long-term reference pictures. This distinction may affect some decoding processes such as motion vector scaling in the temporal direct mode or implicit weighted prediction. If both of thereference pictures used for the temporal direct mode are short-term reference pictures, the motion vector used in the prediction may be scaled according to the picture order count (POC) difference between the current picture and each of the reference pictures. However, if at least one reference picture for the temporal direct mode is a long-term reference picture, default scaling of the motion vector may be used, for example scaling the motion to half may be used. Similarly, if a short-term reference picture is used for implicit weighted prediction, the prediction weight may be scaled according to the POC difference between the POC of the current picture and the POC of the reference picture. However, if a long-term reference picture is used for implicit weighted prediction, a default prediction weight may be used, such as 0.5 in implicit weighted prediction for bi-predicted blocks.

Some video coding formats, such as H.264/AVC, include the frame_num syntax element, which is used for various decoding processes related to multiple reference pictures. In H.264/AVC, the value of frame_num for IDR pictures is 0. The value of frame_num for non-IDR pictures is equal to the frame_num of the previous reference picture in decoding order incremented by 1 (in modulo arithmetic, i.e., the value of frame_num wrap over to 0 after a maximum value of frame_num).

H.264/AVC and HEVC include a concept of picture order count (POC). A value of POC is derived for each picture and is non-decreasing with increasing picture position in output order. POC therefore indicates the output order of pictures. POC may be used in the decoding process for example for implicit scaling of motion vectors in the temporal direct mode of bi-predictive slices, for implicitly derived weights in weighted prediction, and for reference picture list initialization. Furthermore, POC may be used in the verification of output order conformance. In H.264/AVC, POC is specified relative to the previous IDR picture or a picture containing a memory management control operation marking all pictures as “unused for reference”.

H.264/AVC specifies the process for decoded reference picture marking in order to control the memory consumption in the decoder. The maximum number of reference pictures used for inter prediction, referred to as M, is determined in the sequence parameter set. When a reference picture is decoded, it is marked as “used for reference”. If the decoding of the reference picture caused more than M pictures marked as “used for reference”, at least one picture is marked as “unused for reference”. There are two types of operation for decoded reference picture marking: adaptive memory control and sliding window. The operation mode for decoded reference picture marking is selected on picture basis. The adaptive memory control enables explicit signaling which pictures are marked as “unused for

reference" and may also assign long-term indices to short-term reference pictures. The adaptive memory control may require the presence of memory management control operation (MMCO) parameters in the bitstream. MMCO parameters may be included in a decoded reference picture marking syntax structure. If the sliding window operation mode is in use and there are M pictures marked as "used for reference", the short-term reference picture that was the first decoded picture among those short-term reference pictures that are marked as "used for reference" is marked as "unused for reference". In other words, the sliding window operation mode results into first-in-first-out buffering operation among short-term reference pictures.

One of the memory management control operations in H.264/AVC causes all reference pictures except for the current picture to be marked as "unused for reference". An instantaneous decoding refresh (IDR) picture contains only intra-coded slices and causes a similar "reset" of reference pictures.

In a draft HEVC standard, reference picture marking syntax structures and related decoding processes are not used, but instead a reference picture set (RPS) syntax structure and decoding process are used instead for a similar purpose. A reference picture set valid or active for a picture includes all the reference pictures used as reference for the picture and all the reference pictures that are kept marked as "used for reference" for any subsequent pictures in decoding order. There are six subsets of the reference picture set, which are referred to as namely RefPicSetStCurr0, RefPicSetStCurr1, RefPicSetStFoll0, RefPicSetStFoll1, RefPicSetLtCurr, and RefPicSetLtFoll. The notation of the six subsets is as follows. "Curr" refers to reference pictures that are included in the reference picture lists of the current picture and hence may be used as inter prediction reference for the current picture. "Foll" refers to reference pictures that are not included in the reference picture lists of the current picture but may be used in subsequent pictures in decoding order as reference pictures. "St" refers to short-term reference pictures, which may generally be identified through a certain number of least significant bits of their POC value. "Lt" refers to long-term reference pictures, which are specifically identified and generally have a greater difference of POC values relative to the current picture than what can be represented by the mentioned certain number of least significant bits. "0" refers to those reference pictures that have a smaller POC value than that of the current picture. "1" refers to those reference pictures that have a greater POC value than that of the current picture. RefPicSetStCurr0, RefPicSetStCurr1, RefPicSetStFoll0 and RefPicSetStFoll1 are collectively referred to as the short-term subset of the reference picture set. RefPicSetLtCurr and RefPicSetLtFoll are collectively referred to as the long-term subset of the reference picture set.

In a draft HEVC standard, a reference picture set may be specified in a sequence parameter set and taken into use in the slice header through an index to the reference picture set. A reference picture set may also be specified in a slice header. A long-term subset of a reference picture set is generally specified only in a slice header, while the short-term subsets of the same reference picture set may be specified in the picture parameter set or slice header. A reference picture set may be coded independently or may be predicted from another reference picture set (known as inter-RPS prediction). When a reference picture set is independently coded, the syntax structure includes up to three loops iterating over different types of

reference pictures; short-term reference pictures with lower POC value than the current picture, short-term reference pictures with higher POC value than the current picture and long-term reference pictures. Each loop entry specifies a picture to be marked as "used for reference". In general, the picture is specified with a differential POC value. The inter-RPS prediction exploits the fact that the reference picture set of the current picture can be predicted from the reference picture set of a previously decoded picture. This is because all the reference pictures of the current picture are either reference pictures of the previous picture or the previously decoded picture itself. It is only necessary to indicate which of these pictures should be reference pictures and be used for the prediction of the current picture. In both types of reference picture set coding, a flag (used_by_curr_pic_X_flag) is additionally sent for each reference picture indicating whether the reference picture is used for reference by the current picture (included in a *Curr list) or not (included in a *Foll list). Pictures that are included in the reference picture set used by the current slice are marked as "used for reference", and pictures that are not in the reference picture set used by the current slice are marked as "unused for reference". If the current picture is an IDR picture, RefPicSetStCurr0, RefPicSetStCurr1, RefPicSetStFoll0, RefPicSetStFoll1, RefPicSetLtCurr, and RefPicSetLtFoll are all set to empty.

A Decoded Picture Buffer (DPB) may be used in the encoder and/or in the decoder. There are two reasons to buffer decoded pictures, for references in inter prediction and for reordering decoded pictures into output order. As H.264/AVC and HEVC provide a great deal of flexibility for both reference picture marking and output reordering, separate buffers for reference picture buffering and output picture buffering may waste memory resources. Hence, the DPB may include a unified decoded picture buffering process for reference pictures and output reordering. A decoded picture may be removed from the DPB when it is no longer used as a reference and is not needed for output.

In many coding modes of H.264/AVC and HEVC, the reference picture for inter prediction is indicated with an index to a reference picture list. The index may be coded with variable length coding, which usually causes a smaller index to have a shorter value for the corresponding syntax element. In H.264/AVC and HEVC, two reference picture lists (reference picture list 0 and reference picture list 1) are generated for each bi-predictive (B) slice, and one reference picture list (reference picture list 0) is formed for each inter-coded (P) slice. In addition, for a B slice in HEVC, a combined list (List C) is constructed after the final reference picture lists (List 0 and List 1) have been constructed. The combined list may be used for uni-prediction (also known as uni-directional prediction) within B slices.

A reference picture list, such as reference picture list 0 and reference picture list 1, is typically constructed in two steps: First, an initial reference picture list is generated. The initial reference picture list may be generated for example on the basis of frame_num, POC, temporal_id, or information on the prediction hierarchy such as GOP structure, or any combination thereof. Second, the initial reference picture list may be reordered by reference picture list reordering (RPLR) commands, also known as reference picture list modification syntax structure, which may be contained in slice headers. The RPLR commands indicate the pictures that are ordered to the beginning of the respective reference picture list.

This second step may also be referred to as the reference picture list modification process, and the RPLR commands may be included in a reference picture list modification syntax structure. If reference picture sets are used, the reference picture list 0 may be initialized to contain RefPicSetStCurr0 first, followed by RefPicSetStCurr1, followed by RefPicSetLtCurr. Reference picture list 1 may be initialized to contain RefPicSetStCurr1 first, followed by RefPicSetStCurr0. The initial reference picture lists may be modified through the reference picture list modification syntax structure, where pictures in the initial reference picture lists may be identified through an entry index to the list.

The combined list in HEVC may be constructed as follows. If the modification flag for the combined list is zero, the combined list is constructed by an implicit mechanism; otherwise it is constructed by reference picture combination commands included in the bitstream. In the implicit mechanism, reference pictures in List C are mapped to reference pictures from List 0 and List 1 in an interleaved fashion starting from the first entry of List 0, followed by the first entry of List 1 and so forth. Any reference picture that has already been mapped in List C is not mapped again. In the explicit mechanism, the number of entries in List C is signaled, followed by the mapping from an entry in List 0 or List 1 to each entry of List C. In addition, when List 0 and List 1 are identical the encoder has the option of setting the ref_pic_list_combination_flag to 0 to indicate that no reference pictures from List 1 are mapped, and that List C is equivalent to List 0. Typical high efficiency video codecs such as a draft HEVC codec employ an additional motion information coding/decoding mechanism, often called merging/merge mode/process/mechanism, where all the motion information of a block/PU is predicted and used without any modification/correction. The aforementioned motion information for a PU comprises 1) The information whether 'the PU is uni-predicted using only reference picture list0' or 'the PU is uni-predicted using only reference picture list1' or 'the PU is bi-predicted using both reference picture list0 and list1' 2) Motion vector value corresponding to the reference picture list0 3) Reference picture index in the reference picture list0 4) Motion vector value corresponding to the reference picture list1 5) Reference picture index in the reference picture list1. Similarly, predicting the motion information is carried out using the motion information of adjacent blocks and/or co-located blocks in temporal reference pictures. Typically, a list, often called as a merge list, is constructed by including motion prediction candidates associated with available adjacent/co-located blocks and the index of selected motion prediction candidate in the list is signalled. Then the motion information of the selected candidate is copied to the motion information of the current PU. When the merge mechanism is employed for a whole CU and the prediction signal for the CU is used as the reconstruction signal, i.e. prediction residual is not processed, this type of coding/decoding the CU is typically named as skip mode or merge based skip mode. In addition to the skip mode, the merge mechanism is also employed for individual PUs (not necessarily the whole CU as in skip mode) and in this case, prediction residual may be utilized to improve prediction quality. This type of prediction mode is typically named as an inter-merge mode.

A syntax structure for decoded reference picture marking may exist in a video coding system. For example, when the decoding of the picture has been completed, the decoded reference picture marking

syntax structure, if present, may be used to adaptively mark pictures as "unused for reference" or "used for long-term reference". If the decoded reference picture marking syntax structure is not present and the number of pictures marked as "used for reference" can no longer increase, a sliding window reference picture marking may be used, which basically marks the earliest (in decoding order) decoded reference picture as unused for reference.

In scalable video coding, a video signal can be encoded into a base layer and one or more enhancement layers. An enhancement layer may enhance the temporal resolution (i.e., the frame rate), the spatial resolution, or simply the quality of the video content represented by another layer or part thereof. Each layer together with all its dependent layers is one representation of the video signal at a certain spatial resolution, temporal resolution and quality level. In this document, we refer to a scalable layer together with all of its dependent layers as a "scalable layer representation". The portion of a scalable bitstream corresponding to a scalable layer representation can be extracted and decoded to produce a representation of the original signal at certain fidelity.

In some cases, data in an enhancement layer can be truncated after a certain location, or even at arbitrary positions, where each truncation position may include additional data representing increasingly enhanced visual quality. Such scalability is referred to as fine-grained (granularity) scalability (FGS). FGS was included in some draft versions of the SVC standard, but it was eventually excluded from the final SVC standard. FGS is subsequently discussed in the context of some draft versions of the SVC standard. The scalability provided by those enhancement layers that cannot be truncated is referred to as coarse-grained (granularity) scalability (CGS). It collectively includes the traditional quality (SNR) scalability and spatial scalability. The SVC standard supports the so-called medium-grained scalability (MGS), where quality enhancement pictures are coded similarly to SNR scalable layer pictures but indicated by high-level syntax elements similarly to FGS layer pictures, by having the quality_id syntax element greater than 0.

SVC uses an inter-layer prediction mechanism, wherein certain information can be predicted from layers other than the currently reconstructed layer or the next lower layer. Information that could be inter-layer predicted includes intra texture, motion and residual data. Inter-layer motion prediction includes the prediction of block coding mode, header information, etc., wherein motion from the lower layer may be used for prediction of the higher layer. In case of intra coding, a prediction from surrounding macroblocks or from co-located macroblocks of lower layers is possible. These prediction techniques do not employ information from earlier coded access units and hence, are referred to as intra prediction techniques. Furthermore, residual data from lower layers can also be employed for prediction of the current layer.

SVC specifies a concept known as single-loop decoding. It is enabled by using a constrained intra texture prediction mode, whereby the inter-layer intra texture prediction can be applied to macroblocks (MBs) for which the corresponding block of the base layer is located inside intra-MBs. At the same time, those intra-MBs in the base layer use constrained intra-prediction (e.g., having the syntax

element "constrained_intra_pred_flag" equal to 1). In single-loop decoding, the decoder performs motion compensation and full picture reconstruction only for the scalable layer desired for playback (called the "desired layer" or the "target layer"), thereby greatly reducing decoding complexity. All of the layers other than the desired layer do not need to be fully decoded because all or part of the data of the MBs not used for inter-layer prediction (be it inter-layer intra texture prediction, inter-layer motion prediction or inter-layer residual prediction) is not needed for reconstruction of the desired layer.

A single decoding loop is needed for decoding of most pictures, while a second decoding loop is selectively applied to reconstruct the base representations, which are needed as prediction references but not for output or display, and are reconstructed only for the so called key pictures (for which "store_ref_base_pic_flag" is equal to 1).

The scalability structure in the SVC draft is characterized by three syntax elements: "temporal_id," "dependency_id" and "quality_id." The syntax element "temporal_id" is used to indicate the temporal scalability hierarchy or, indirectly, the frame rate. A scalable layer representation comprising pictures of a smaller maximum "temporal_id" value has a smaller frame rate than a scalable layer representation comprising pictures of a greater maximum "temporal_id". A given temporal layer typically depends on the lower temporal layers (i.e., the temporal layers with smaller "temporal_id" values) but does not depend on any higher temporal layer. The syntax element "dependency_id" is used to indicate the CGS inter-layer coding dependency hierarchy (which, as mentioned earlier, includes both SNR and spatial scalability). At any temporal level location, a picture of a smaller "dependency_id" value may be used for inter-layer prediction for coding of a picture with a greater "dependency_id" value. The syntax element "quality_id" is used to indicate the quality level hierarchy of a FGS or MGS layer. At any temporal location, and with an identical "dependency_id" value, a picture with "quality_id" equal to QL uses the picture with "quality_id" equal to QL-1 for inter-layer prediction. A coded slice with "quality_id" larger than 0 may be coded as either a truncatable FGS slice or a non-truncatable MGS slice.

For simplicity, all the data units (e.g., Network Abstraction Layer units or NAL units in the SVC context) in one access unit having identical value of "dependency_id" are referred to as a dependency unit or a dependency representation. Within one dependency unit, all the data units having identical value of "quality_id" are referred to as a quality unit or layer representation.

A base representation, also known as a decoded base picture, is a decoded picture resulting from decoding the Video Coding Layer (VCL) NAL units of a dependency unit having "quality_id" equal to 0 and for which the "store_ref_base_pic_flag" is set equal to 1. An enhancement representation, also referred to as a decoded picture, results from the regular decoding process in which all the layer representations that are present for the highest dependency representation are decoded.

As mentioned earlier, CGS includes both spatial scalability and SNR scalability. Spatial scalability is initially designed to support representations of video with different resolutions. For each time instance, VCL NAL units are coded in the same access unit and these VCL NAL units can correspond to different resolutions. During the decoding, a low resolution VCL NAL unit provides the

motion field and residual which can be optionally inherited by the final decoding and reconstruction of the high resolution picture. When compared to older video compression standards, SVC's spatial scalability has been generalized to enable the base layer to be a cropped and zoomed version of the enhancement layer.

MGS quality layers are indicated with "quality_id" similarly as FGS quality layers. For each dependency unit (with the same "dependency_id"), there is a layer with "quality_id" equal to 0 and there can be other layers with "quality_id" greater than 0. These layers with "quality_id" greater than 0 are either MGS layers or FGS layers, depending on whether the slices are coded as truncatable slices.

In the basic form of FGS enhancement layers, only inter-layer prediction is used. Therefore, FGS enhancement layers can be truncated freely without causing any error propagation in the decoded sequence. However, the basic form of FGS suffers from low compression efficiency. This issue arises because only low-quality pictures are used for inter prediction references. It has therefore been proposed that FGS-enhanced pictures be used as inter prediction references. However, this may cause encoding-decoding mismatch, also referred to as drift, when some FGS data are discarded.

One feature of a draft SVC standard is that the FGS NAL units can be freely dropped or truncated, and a feature of the SVCV standard is that MGS NAL units can be freely dropped (but cannot be truncated) without affecting the conformance of the bitstream. As discussed above, when those FGS or MGS data have been used for inter prediction reference during encoding, dropping or truncation of the data would result in a mismatch between the decoded pictures in the decoder side and in the encoder side. This mismatch is also referred to as drift.

To control drift due to the dropping or truncation of FGS or MGS data, SVC applied the following solution: In a certain dependency unit, a base representation (by decoding only the CGS picture with "quality_id" equal to 0 and all the dependent-on lower layer data) is stored in the decoded picture buffer. When encoding a subsequent dependency unit with the same value of "dependency_id," all of the NAL units, including FGS or MGS NAL units, use the base representation for inter prediction reference. Consequently, all drift due to dropping or truncation of FGS or MGS NAL units in an earlier access unit is stopped at this access unit. For other dependency units with the same value of "dependency_id," all of the NAL units use the decoded pictures for inter prediction reference, for high coding efficiency.

Each NAL unit includes in the NAL unit header a syntax element "use_ref_base_pic_flag." When the value of this element is equal to 1, decoding of the NAL unit uses the base representations of the reference pictures during the inter prediction process. The syntax element "store_ref_base_pic_flag" specifies whether (when equal to 1) or not (when equal to 0) to store the base representation of the current picture for future pictures to use for inter prediction.

NAL units with "quality_id" greater than 0 do not contain syntax elements related to reference picture lists construction and weighted prediction, i.e., the syntax elements "num_ref_active_lx_minus1" (x=0 or 1), the reference picture list reordering syntax table, and the weighted prediction syntax table are

not present. Consequently, the MGS or FGS layers have to inherit these syntax elements from the NAL units with “quality_id” equal to 0 of the same dependency unit when needed.

In SVC, a reference picture list consists of either only base representations (when “use_ref_base_pic_flag” is equal to 1) or only decoded pictures not marked as “base representation” (when “use_ref_base_pic_flag” is equal to 0), but never both at the same time.

As indicated earlier, MVC is an extension of H.264/AVC. Many of the definitions, concepts, syntax structures, semantics, and decoding processes of H.264/AVC apply also to MVC as such or with certain generalizations or constraints. Some definitions, concepts, syntax structures, semantics, and decoding processes of MVC are described in the following.

An access unit in MVC is defined to be a set of NAL units that are consecutive in decoding order and contain exactly one primary coded picture consisting of one or more view components. In addition to the primary coded picture, an access unit may also contain one or more redundant coded pictures, one auxiliary coded picture, or other NAL units not containing slices or slice data partitions of a coded picture. The decoding of an access unit results in one decoded picture consisting of one or more decoded view components, when decoding errors, bitstream errors or other errors which may affect the decoding do not occur. In other words, an access unit in MVC contains the view components of the views for one output time instance.

A view component in MVC is referred to as a coded representation of a view in a single access unit.

Inter-view prediction may be used in MVC and refers to prediction of a view component from decoded samples of different view components of the same access unit. In MVC, inter-view prediction is realized similarly to inter prediction. For example, inter-view reference pictures are placed in the same reference picture list(s) as reference pictures for inter prediction, and a reference index as well as a motion vector are coded or inferred similarly for inter-view and inter reference pictures.

An anchor picture is a coded picture in which all slices may reference only slices within the same access unit, i.e., inter-view prediction may be used, but no inter prediction is used, and all following coded pictures in output order do not use inter prediction from any picture prior to the coded picture in decoding order. Inter-view prediction may be used for IDR view components that are part of a non-base view. A base view in MVC is a view that has the minimum value of view order index in a coded video sequence. The base view can be decoded independently of other views and does not use inter-view prediction. The base view can be decoded by H.264/AVC decoders supporting only the single-view profiles, such as the Baseline Profile or the High Profile of H.264/AVC.

In the MVC standard, many of the sub-processes of the MVC decoding process use the respective sub-processes of the H.264/AVC standard by replacing term "picture", “frame”, and “field” in the sub-process specification of the H.264/AVC standard by "view component", "frame view component", and "field view component", respectively. Likewise, terms "picture", “frame”, and “field”

are often used in the following to mean "view component", "frame view component", and "field view component", respectively.

In scalable multiview coding, the same bitstream may contain coded view components of multiple views and at least some coded view components may be coded using quality and/or spatial scalability.

A texture view refers to a view that represents ordinary video content, for example has been captured using an ordinary camera, and is usually suitable for rendering on a display. A texture view typically comprises pictures having three components, one luma component and two chroma components. In the following, a texture picture typically comprises all its component pictures or color components unless otherwise indicated for example with terms luma texture picture and chroma texture picture.

Depth-enhanced video refers to texture video having one or more views associated with depth video having one or more depth views. A number of approaches may be used for representing of depth-enhanced video, including the use of video plus depth (V+D), multiview video plus depth (MVD), and layered depth video (LDV). In the video plus depth (V+D) representation, a single view of texture and the respective view of depth are represented as sequences of texture picture and depth pictures, respectively. The MVD representation contains a number of texture views and respective depth views. In the LDV representation, the texture and depth of the central view are represented conventionally, while the texture and depth of the other views are partially represented and cover only the dis-occluded areas required for correct view synthesis of intermediate views.

Depth-enhanced video may be coded in a manner where texture and depth are coded independently of each other. For example, texture views may be coded as one MVC bitstream and depth views may be coded as another MVC bitstream. Alternatively depth-enhanced video may be coded in a manner where texture and depth are jointly coded. When joint coding texture and depth views is applied for a depth-enhanced video representation, some decoded samples of a texture picture or data elements for decoding of a texture picture are predicted or derived from some decoded samples of a depth picture or data elements obtained in the decoding process of a depth picture. Alternatively or in addition, some decoded samples of a depth picture or data elements for decoding of a depth picture are predicted or derived from some decoded samples of a texture picture or data elements obtained in the decoding process of a texture picture.

It has been found that a solution for some multiview 3D video (3DV) applications is to have a limited number of input views, e.g. a mono or a stereo view plus some supplementary data, and to render (i.e. synthesize) all required views locally at the decoder side. From several available technologies for view rendering, depth image-based rendering (DIBR) has shown to be a competitive alternative.

A simplified model of a DIBR-based 3DV system is shown in Figure 5. The input of a 3D video codec comprises a stereoscopic video and corresponding depth information with stereoscopic baseline b0. Then the 3D video codec synthesizes a number of virtual views between two input views with baseline

(bi < b0). DIBR algorithms may also enable extrapolation of views that are outside the two input views and not in between them. Similarly, DIBR algorithms may enable view synthesis from a single view of texture and the respective depth view. However, in order to enable DIBR-based multiview rendering, texture data should be available at the decoder side along with the corresponding depth data.

In such 3DV system, depth information is produced at the encoder side in a form of depth pictures (also known as depth maps) for each video frame. A depth map is an image with per-pixel depth information. Each sample in a depth map represents the distance of the respective texture sample from the plane on which the camera lies. In other words, if the z axis is along the shooting axis of the cameras (and hence orthogonal to the plane on which the cameras lie), a sample in a depth map represents the value on the z axis.

Depth information can be obtained by various means. For example, depth of the 3D scene may be computed from the disparity registered by capturing cameras. A depth estimation algorithm takes a stereoscopic view as an input and computes local disparities between the two offset images of the view. Each image is processed pixel by pixel in overlapping blocks, and for each block of pixels a horizontally localized search for a matching block in the offset image is performed. Once a pixel-wise disparity is computed, the corresponding depth value z is calculated by equation (1):

$$z = \frac{f \cdot b}{d + \Delta d} \qquad (1),$$

where f is the focal length of the camera and b is the baseline distance between cameras, as shown in Figure 6. Further, d refers to the disparity observed between the two cameras, and the camera offset Δd reflects a possible horizontal misplacement of the optical centers of the two cameras. However, since the algorithm is based on block matching, the quality of a depth-through-disparity estimation is content dependent and very often not accurate. For example, no straightforward solution for depth estimation is possible for image fragments that are featuring very smooth areas with no textures or large level of noise.

Disparity or parallax maps, such as parallax maps specified in ISO/IEC International Standard 23002-3, may be processed similarly to depth maps. Depth and disparity have a straightforward correspondence and they can be computed from each other through mathematical equation.

The coding and decoding order of texture and depth view components within an access unit is typically such that the data of a coded view component is not interleaved by any other coded view component, and the data for an access unit is not interleaved by any other access unit in the bitstream/decoding order. For example, there may be two texture and depth views ($T0_t$, $T1_t$, $T0_{t+1}$, $T1_{t+1}$, $T0_{t+2}$, $T1_{t+2}$, $D0_t$, $D1_t$, $D0_{t+1}$, $D1_{t+1}$, $D0_{t+2}$, $D1_{t+2}$) in different access units (t, t+1, t+2), as illustrated in Figure 7, where the access unit t consisting of texture and depth view components ($T0_t$,$T1_t$, $D0_t$,$D1_t$) precedes in bitstream and decoding order the access unit t+1 consisting of texture and depth view components ($T0_{t+1}$,$T1_{t+1}$, $D0_{t+1}$,$D1_{t+1}$).

The coding and decoding order of view components within an access unit may be governed by the coding format or determined by the encoder. A texture view component may be coded before the respective depth view component of the same view, and hence such depth view components may be predicted from the texture view components of the same view. Such texture view components may be coded for example by MVC encoder and decoder by MVC decoder. An enhanced texture view component refers herein to a texture view component that is coded after the respective depth view component of the same view and may be predicted from the respective depth view component. The texture and depth view components of the same access units are typically coded in view dependency order. Texture and depth view components can be ordered in any order with respect to each other as long as the ordering obeys the mentioned constraints.

Texture views and depth views may be coded into a single bitstream where some of the texture views may be compatible with one or more video standards such as H.264/AVC and/or MVC. In other words, a decoder may be able to decode some of the texture views of such a bitstream and can omit the remaining texture views and depth views.

In this context an encoder that encodes one or more texture and depth views into a single H.264/AVC and/or MVC compatible bitstream is also called as a 3DV-ATM encoder. Bitstreams generated by such an encoder can be referred to as 3DV-ATM bitstreams. The 3DV-ATM bitstreams may include some of the texture views that H.264/AVC and/or MVC decoder cannot decode, and depth views. A decoder capable of decoding all views from 3DV-ATM bitstreams may also be called as a 3DV-ATM decoder.

3DV-ATM bitstreams can include a selected number of AVC/MVC compatible texture views. The depth views for the AVC/MVC compatible texture views may be predicted from the texture views. The remaining texture views may utilize enhanced texture coding and depth views may utilize depth coding.

A high level flow chart of an embodiment of an encoder 200 capable of encoding texture views and depth views is presented in Figure 8 and a decoder 210 capable of decoding texture views and depth views is presented in Figure 9. On these figures solid lines depict general data flow and dashed lines show control information signaling. The encoder 200 may receive texture components 201 to be encoded by a texture encoder 202 and depth map components 203 to be encoded by a depth encoder 204. When the encoder 200 is encoding texture components according to AVC/MVC a first switch 205 may be switched off. When the encoder 200 is encoding enhanced texture components the first switch 205 may be switched on so that information generated by the depth encoder 204 may be provided to the texture encoder 202. The encoder of this example also comprises a second switch 206 which may be operated as follows. The second switch 206 is switched on when the encoder is encoding depth information of AVC/MVC views, and the second switch 206 is switched off when the encoder is encoding depth information of enhanced texture views. The encoder 200 may output a bitstream 207 containing encoded video information.

The decoder 210 may operate in a similar manner but at least partly in a reversed order. The decoder 210 may receive the bitstream 207 containing encoded video information. The decoder 210 comprises a texture decoder 211 for decoding texture information and a depth decoder 212 for decoding depth information. A third switch 213 may be provided to control information delivery from the depth decoder 212 to the texture decoder 211, and a fourth switch 214 may be provided to control information delivery from the texture decoder 211 to the depth decoder 212. When the decoder 210 is to decode AVC/MVC texture views the third switch 213 may be switched off and when the decoder 210 is to decode enhanced texture views the third switch 213 may be switched on. When the decoder 210 is to decode depth of AVC/MVC texture views the fourth switch 214 may be switched on and when the decoder 210 is to decode depth of enhanced texture views the fourth switch 214 may be switched off. The Decoder 210 may output reconstructed texture components 215 and reconstructed depth map components 216.

Many video encoders utilize the Lagrangian cost function to find rate-distortion optimal coding modes, for example the desired macroblock mode and associated motion vectors. This type of cost function uses a weighting factor or λ to tie together the exact or estimated image distortion due to lossy coding methods and the exact or estimated amount of information required to represent the pixel/sample values in an image area. The Lagrangian cost function may be represented by the equation:

$C=D+\lambda R$

where C is the Lagrangian cost to be minimised, D is the image distortion (for example, the mean-squared error between the pixel/sample values in original image block and in coded image block) with the mode and motion vectors currently considered, λ is a Lagrangian coefficient and R is the number of bits needed to represent the required data to reconstruct the image block in the decoder (including the amount of data to represent the candidate motion vectors).

A coding standard may include a sub-bitstream extraction process, and such is specified for example in SVC, MVC, and HEVC. The sub-bitstream extraction process relates to converting a bitstream by removing NAL units to a sub-bitstream. The sub-bitstream still remains conforming to the standard. For example, in a draft HEVC standard, the bitstream created by excluding all VCL NAL units having a temporal_id greater than or equal to a selected value and including all other VCL NAL units remains conforming. Consequently, a picture having temporal_id equal to TID does not use any picture having a temporal_id greater than TID as inter prediction reference.

Fig. 1 shows a block diagram of a video coding system according to an example embodiment as a schematic block diagram of an exemplary apparatus or electronic device 50, which may incorporate a codec according to an embodiment of the invention. Fig. 2 shows a layout of an apparatus according to an example embodiment. The elements of Figs. 1 and 2 will be explained next.

The electronic device 50 may for example be a mobile terminal or user equipment of a wireless communication system. However, it would be appreciated that embodiments of the invention may be

implemented within any electronic device or apparatus which may require encoding and decoding or encoding or decoding video images.

The apparatus 50 may comprise a housing 30 for incorporating and protecting the device. The apparatus 50 further may comprise a display 32 in the form of a liquid crystal display. In other embodiments of the invention the display may be any suitable display technology suitable to display an image or video. The apparatus 50 may further comprise a keypad 34. In other embodiments of the invention any suitable data or user interface mechanism may be employed. For example the user interface may be implemented as a virtual keyboard or data entry system as part of a touch-sensitive display. The apparatus may comprise a microphone 36 or any suitable audio input which may be a digital or analogue signal input. The apparatus 50 may further comprise an audio output device which in embodiments of the invention may be any one of: an earpiece 38, speaker, or an analogue audio or digital audio output connection. The apparatus 50 may also comprise a battery 40 (or in other embodiments of the invention the device may be powered by any suitable mobile energy device such as solar cell, fuel cell or clockwork generator). The apparatus may further comprise an infrared port 42 for short range line of sight communication to other devices. In other embodiments the apparatus 50 may further comprise any suitable short range communication solution such as for example a Bluetooth wireless connection or a USB/firewire wired connection.

The apparatus 50 may comprise a controller 56 or processor for controlling the apparatus 50. The controller 56 may be connected to memory 58 which in embodiments of the invention may store both data in the form of image and audio data and/or may also store instructions for implementation on the controller 56. The controller 56 may further be connected to codec circuitry 54 suitable for carrying out coding and decoding of audio and/or video data or assisting in coding and decoding carried out by the controller 56.

The apparatus 50 may further comprise a card reader 48 and a smart card 46, for example a UICC and UICC reader for providing user information and being suitable for providing authentication information for authentication and authorization of the user at a network.

The apparatus 50 may comprise radio interface circuitry 52 connected to the controller and suitable for generating wireless communication signals for example for communication with a cellular communications network, a wireless communications system or a wireless local area network. The apparatus 50 may further comprise an antenna 44 connected to the radio interface circuitry 52 for transmitting radio frequency signals generated at the radio interface circuitry 52 to other apparatus(es) and for receiving radio frequency signals from other apparatus(es).

In some embodiments of the invention, the apparatus 50 comprises a camera capable of recording or detecting individual frames which are then passed to the codec 54 or controller for processing. In some embodiments of the invention, the apparatus may receive the video image data for processing from another device prior to transmission and/or storage. In some embodiments of the invention, the apparatus 50 may receive either wirelessly or by a wired connection the image for coding/decoding.

Fig. 3 shows an arrangement for video coding comprising a plurality of apparatuses, networks and network elements according to an example embodiment. With respect to Figure 3, an example of a system within which embodiments of the present invention can be utilized is shown. The system 10 comprises multiple communication devices which can communicate through one or more networks. The system 10 may comprise any combination of wired or wireless networks including, but not limited to a wireless cellular telephone network (such as a GSM, UMTS, CDMA network etc), a wireless local area network (WLAN) such as defined by any of the IEEE 802.x standards, a Bluetooth personal area network, an Ethernet local area network, a token ring local area network, a wide area network, and the Internet.

The system 10 may include both wired and wireless communication devices or apparatus 50 suitable for implementing embodiments of the invention. For example, the system shown in Figure 3 shows a mobile telephone network 11 and a representation of the internet 28. Connectivity to the internet 28 may include, but is not limited to, long range wireless connections, short range wireless connections, and various wired connections including, but not limited to, telephone lines, cable lines, power lines, and similar communication pathways.

The example communication devices shown in the system 10 may include, but are not limited to, an electronic device or apparatus 50, a combination of a personal digital assistant (PDA) and a mobile telephone 14, a PDA 16, an integrated messaging device (IMD) 18, a desktop computer 20, a notebook computer 22. The apparatus 50 may be stationary or mobile when carried by an individual who is moving. The apparatus 50 may also be located in a mode of transport including, but not limited to, a car, a truck, a taxi, a bus, a train, a boat, an airplane, a bicycle, a motorcycle or any similar suitable mode of transport.

Some or further apparatuses may send and receive calls and messages and communicate with service providers through a wireless connection 25 to a base station 24. The base station 24 may be connected to a network server 26 that allows communication between the mobile telephone network 11 and the internet 28. The system may include additional communication devices and communication devices of various types.

The communication devices may communicate using various transmission technologies including, but not limited to, code division multiple access (CDMA), global systems for mobile communications (GSM), universal mobile telecommunications system (UMTS), time divisional multiple access (TDMA), frequency division multiple access (FDMA), transmission control protocol-internet protocol (TCP-IP), short messaging service (SMS), multimedia messaging service (MMS), email, instant messaging service (IMS), Bluetooth, IEEE 802.11 and any similar wireless communication technology. A communications device involved in implementing various embodiments of the present invention may communicate using various media including, but not limited to, radio, infrared, laser, cable connections, and any suitable connection.

Figs. 4a and 4b show block diagrams for video encoding and decoding according to an example embodiment.

Figure 4a shows the encoder as comprising a pixel predictor 302, prediction error encoder 303 and prediction error decoder 304. Figure 4a also shows an embodiment of the pixel predictor 302 as comprising an inter-predictor 306, an intra-predictor 308, a mode selector 310, a filter 316, and a reference frame memory 318. In this embodiment the mode selector 310 comprises a block processor 381 and a cost evaluator 382. The encoder may further comprise an entropy encoder 330 for entropy encoding the bit stream.

Figure 4b depicts an embodiment of the inter predictor 306. The inter predictor 306 comprises a reference frame selector 360 for selecting reference frame or frames, a motion vector definer 361, a prediction list former 363 and a motion vector selector 364. These elements or some of them may be part of a prediction processor 362 or they may be implemented by using other means.

The pixel predictor 302 receives the image 300 to be encoded at both the inter-predictor 306 (which determines the difference between the image and a motion compensated reference frame 318) and the intra-predictor 308 (which determines a prediction for an image block based only on the already processed parts of a current frame or picture). The output of both the inter-predictor and the intra-predictor are passed to the mode selector 310. Both the inter-predictor 306 and the intra-predictor 308 may have more than one intra-prediction modes. Hence, the inter-prediction and the intra-prediction may be performed for each mode and t

he predicted signal may be provided to the mode selector 310. The mode selector 310 also receives a copy of the image 300.

The mode selector 310 determines which encoding mode to use to encode the current block. If the mode selector 310 decides to use an inter-prediction mode it will pass the output of the inter-predictor 306 to the output of the mode selector 310. If the mode selector 310 decides to use an intra-prediction mode it will pass the output of one of the intra-predictor modes to the output of the mode selector 310.

The mode selector 310 may use, in the cost evaluator block 382, for example Lagrangian cost functions to choose between coding modes and their parameter values, such as motion vectors, reference indexes, and intra prediction direction, typically on block basis. This kind of cost function uses a weighting factor *lambda* to tie together the (exact or estimated) image distortion due to lossy coding methods and the (exact or estimated) amount of information that is required to represent the pixel values in an image area: $C = D + lambda \times R$, where C is the Lagrangian cost to be minimized, D is the image distortion (e.g. Mean Squared Error) with the mode and their parameters, and R the number of bits needed to represent the required data to reconstruct the image block in the decoder (e.g. including the amount of data to represent the candidate motion vectors).

The output of the mode selector is passed to a first summing device 321. The first summing device may subtract the pixel predictor 302 output from the image 300 to produce a first prediction error signal 320 which is input to the prediction error encoder 303.

The pixel predictor 302 further receives from a preliminary reconstructor 339 the combination of the prediction representation of the image block 312 and the output 338 of the prediction error decoder 304. The preliminary reconstructed image 314 may be passed to the intra-predictor 308 and to a filter 316. The filter 316 receiving the preliminary representation may filter the preliminary representation and output a final reconstructed image 340 which may be saved in a reference frame memory 318. The reference frame memory 318 may be connected to the inter-predictor 306 to be used as the reference image against which the future image 300 is compared in inter-prediction operations. In many embodiments the reference frame memory 318 may be capable of storing more than one decoded picture, and one or more of them may be used by the inter-predictor 306 as reference pictures against which the future images 300 are compared in inter prediction operations. The reference frame memory 318 may in some cases be also referred to as the Decoded Picture Buffer.

The operation of the pixel predictor 302 may be configured to carry out any known pixel prediction algorithm known in the art.

The pixel predictor 302 may also comprise a filter 385 to filter the predicted values before outputting them from the pixel predictor 302.

The operation of the prediction error encoder 302 and prediction error decoder 304 will be described hereafter in further detail. In the following examples the encoder generates images in terms of 16x16 pixel macroblocks which go to form the full image or picture. However, it is noted that Fig. 4a is not limited to block size 16x16, but any block size and shape can be used generally, and likewise Fig. 4a is not limited to partitioning of a picture to macroblocks but any other picture partitioning to blocks, such as coding units, may be used. Thus, for the following examples the pixel predictor 302 outputs a series of predicted macroblocks of size 16x16 pixels and the first summing device 321 outputs a series of 16x16 pixel residual data macroblocks which may represent the difference between a first macroblock in the image 300 against a predicted macroblock (output of pixel predictor 302).

The prediction error encoder 303 comprises a transform block 342 and a quantizer 344. The transform block 342 transforms the first prediction error signal 320 to a transform domain. The transform is, for example, the DCT transform or its variant. The quantizer 344 quantizes the transform domain signal, e.g. the DCT coefficients, to form quantized coefficients.

The prediction error decoder 304 receives the output from the prediction error encoder 303 and produces a decoded prediction error signal 338 which when combined with the prediction representation of the image block 312 at the second summing device 339 produces the preliminary reconstructed image 314. The prediction error decoder may be considered to comprise a dequantizer 346, which dequantizes the quantized coefficient values, e.g. DCT coefficients, to reconstruct the transform signal approximately and an inverse transformation block 348, which performs the inverse transformation to the reconstructed transform signal wherein the output of the inverse transformation block 348 contains reconstructed block(s). The prediction error decoder may also comprise a macroblock filter (not shown) which may filter the reconstructed macroblock according to further decoded information and filter parameters.

In the following the operation of an example embodiment of the inter predictor 306 will be described in more detail. The inter predictor 306 receives the current block for inter prediction. It is assumed that for the current block there already exists one or more neighboring blocks which have been encoded and motion vectors have been defined for them. For example, the block on the left side and/or the block above the current block may be such blocks. Spatial motion vector predictions for the current block can be formed e.g. by using the motion vectors of the encoded neighboring blocks and/or of non-neighbor blocks in the same slice or frame, using linear or non-linear functions of spatial motion vector predictions, using a combination of various spatial motion vector predictors with linear or non-linear operations, or by any other appropriate means that do not make use of temporal reference information. It may also be possible to obtain motion vector predictors by combining both spatial and temporal prediction information of one or more encoded blocks. These kinds of motion vector predictors may also be called as spatio-temporal motion vector predictors.

Reference frames used in encoding may be stored to the reference frame memory. Each reference frame may be included in one or more of the reference picture lists, within a reference picture list, each entry has a reference index which identifies the reference frame. When a reference frame is no longer used as a reference frame it may be removed from the reference frame memory or marked as "unused for reference" or a non-reference frame wherein the storage location of that reference frame may be occupied for a new reference frame.

As described above, an access unit may contain slices of different component types (e.g. primary texture component, redundant texture component, auxiliary component, depth/disparity component), of different views, and of different scalable layers.

It has been proposed that at least a subset of syntax elements that have conventionally been included in a slice header are included in a GOS (Group of Slices) parameter set by an encoder. An encoder may code a GOS parameter set as a NAL unit. GOS parameter set NAL units may be included in the bitstream together with for example coded slice NAL units, but may also be carried out-of-band as described earlier in the context of other parameter sets.

The GOS parameter set syntax structure may include an identifier, which may be used when referring to a particular GOS parameter set instance for example from a slice header or another GOS parameter set. Alternatively, the GOS parameter set syntax structure does not include an identifier but an identifier may be inferred by both the encoder and decoder for example using the bitstream order of GOS parameter set syntax structures and a pre-defined numbering scheme.

The encoder and the decoder may infer the contents or the instance of GOS parameter set from other syntax structures already encoded or decoded or present in the bitstream. For example, the slice header of the texture view component of the base view may implicitly form a GOS parameter set. The encoder and decoder may infer an identifier value for such inferred GOS parameter sets. For example, the GOS parameter set formed from the slice header of the texture view component of the base view may be inferred to have identifier value equal to 0.

A GOS parameter set may be valid within a particular access unit associated with it. For example, if a GOS parameter set syntax structure is included in the NAL unit sequence for a particular access unit, where the sequence is in decoding or bitstream order, the GOS parameter set may be valid from its appearance location until the end of the access unit. Alternatively, a GOS parameter set may be valid for many access units.

The encoder may encode many GOS parameter sets for an access unit. The encoder may determine to encode a GOS parameter set if it is known, expected, or estimated that at least a subset of syntax element values in a slice header to be coded would be the same in a subsequent slice header.

A limited numbering space may be used for the GOS parameter set identifier. For example, a fixed-length code may be used and may be interpreted as an unsigned integer value of a certain range. The encoder may use a GOS parameter set identifier value for a first GOS parameter set and subsequently for a second GOS parameter set, if the first GOS parameter set is subsequently not referred to for example by any slice header or GOS parameter set. The encoder may repeat a GOS parameter set syntax structure within the bitstream for example to achieve a better robustness against transmission errors.

In many embodiments, syntax elements which may be included in a GOS parameter set are conceptually collected in sets of syntax elements. A set of syntax elements for a GOS parameter set may be formed for example on one or more of the following basis:

- Syntax elements indicating a scalable layer and/or other scalability features
- Syntax elements indicating a view and/or other multiview features
- Syntax elements related to a particular component type, such as depth/disparity
- Syntax elements related to access unit identification, decoding order and/or output order and/or other syntax elements which may stay unchanged for all slices of an access unit
- Syntax elements which may stay unchanged in all slices of a view component
- Syntax elements related to reference picture list modification
- Syntax elements related to the reference picture set used
- Syntax elements related to decoding reference picture marking
- Syntax elements related to prediction weight tables for weighted prediction
- Syntax elements for controlling deblocking filtering
- Syntax elements for controlling adaptive loop filtering
- Syntax elements for controlling sample adaptive offset
- Any combination of sets above

For each syntax element set, the encoder may have one or more of the following options when coding a GOS parameter set:

- The syntax element set may be coded into a GOS parameter set syntax structure, i.e. coded syntax element values of the syntax element set may be included in the GOS parameter set syntax structure.

- The syntax element set may be included by reference into a GOS parameter set. The reference may be given as an identifier to another GOS parameter set. The encoder may use a different reference GOS parameter set for different syntax element sets.
- The syntax element set may be indicated or inferred to be absent from the GOS parameter set.

The options from which the encoder is able to choose for a particular syntax element set when coding a GOS parameter set may depend on the type of the syntax element set. For example, a syntax element set related to scalable layers may always be present in a GOS parameter set, while the set of syntax elements which may stay unchanged in all slices of a view component may not be available for inclusion by reference but may be optionally present in the GOS parameter set and the syntax elements related to reference picture list modification may be included by reference in, included as such in, or be absent from a GOS parameter set syntax structure. The encoder may encode indications in the bitstream, for example in a GOS parameter set syntax structure, which option was used in encoding. The code table and/or entropy coding may depend on the type of the syntax element set. The decoder may use, based on the type of the syntax element set being decoded, the code table and/or entropy decoding that is matched with the code table and/or entropy encoding used by the encoder.

The encoder may have multiple means to indicate the association between a syntax element set and the GOS parameter set used as the source for the values of the syntax element set. For example, the encoder may encode a loop of syntax elements where each loop entry is encoded as syntax elements indicating a GOS parameter set identifier value used as a reference and identifying the syntax element sets copied from the reference GOP parameter set. In another example, the encoder may encode a number of syntax elements, each indicating a GOS parameter set. The last GOS parameter set in the loop containing a particular syntax element set is the reference for that syntax element set in the GOS parameter set the encoder is currently encoding into the bitstream. The decoder parses the encoded GOS parameter sets from the bitstream accordingly so as to reproduce the same GOS parameter sets as the encoder.

It has been proposed to have a partial updating mechanism for the Adaptation Parameter Set in order to reduce the size of APS NAL units and hence to spend a smaller bitrate for conveying APS NAL units. Although the APS provides an effective approach to share picture-adaptive information common at the slice level, coding of APS NAL units independently may be suboptimal when only a part of the APS parameters changes compared to one or more earlier Adaptation Parameter Sets.

In document JCTVC-H0069 (http://phenix.int-evry.fr/jct/doc_end_user/documents/8_San%20Jose/wg11/JCTVC-H0069-v4.zip), the APS syntax structure is subdivided into a number of groups of syntax elements, each associated with a certain coding technology (such as Adaptive In-Loop Filter (ALF), or Sample Adaptive Offset (SAO)). Each of these groups in the APS syntax structure is preceded by a flag indicating their respective presence. The APS syntax structure also includes a conditional reference to another APS. A ref_aps_flag signals the

presence of a reference ref_aps_id referred to by the current APS. With this link mechanism, a linked list of multiple APSs can be created. The decoding process during APS activation uses the reference in the slice header to address the first APS of the linked list. Those groups of syntax elements for which the associated flag (such as the aps_adaptive_loop_filter_data_present_flag) is set, are decoded from the subject APS. After this decoding, the linked list is followed to the next linked APS (if any—as indicated by ref_aps_flag equal to 1). Only those groups which were not signaled as present previously, but are signaled as present in the current APS, are decoded from the current APS. The mechanism continues along the list of linked APSs until one of three conditions are met: (1) all required groups of syntax elements (as indicated by SPS, PPS, or profile/level) have been decoded from the linked APS chain, (2) the end of the list is detected, and (3) a fixed, probably profile-dependent, number of links have been followed—the number could be as small as one. If there are any groups that are not signaled as present in any of the linked APSs, the related decoding tool is not used for this picture. Condition (2) prevents circular referencing loops. The complexity of the referencing mechanism is further limited by the finite size of the APS table. In JCTVC-H0069, the de-referencing, i.e. resolving the source for each group of syntax elements, is proposed to be performed each time an APS is activated, typically once at the beginning of decoding a slice.

It has also been proposed in document JCTVC-H0255 to include multiple APS identifiers in the slice header, each specifying the source APS for certain groups of syntax elements, e.g. one APS being the source for quantization matrices and another APS being the source for ALF parameters. In document JCTVC-H0381, a "copy" flag for each type of APS parameters was proposed, which allows copying that type of APS parameters from another APS. In document JCTVC-H0505, a Group Parameter Set (GPS) was introduced, which collects parameter set identifiers of different types of parameter sets (SPS, PPS, APS) and may contain multiple APS parameter set identifiers. Furthermore, it was proposed in JCTVC-H0505 that a slice header contains a GPS identifier to be used for decoding of the slice instead of individual PPS, and APS identifiers.

The above-mentioned options for coding of Adaptation Parameter Sets may have one or more of the following shortcomings:

Losses of APS NAL units cannot be detected and hence wrong APS parameter values may be used in decoding. It is allowed to encode and send an APS syntax structure that uses an APS identifier value which has earlier been used for another APS syntax structure. However, an APS syntax structure may be lost during transmission, particularly if APS NAL units are transmitted in-band and/or using unreliable transmission mechanism. There has not been presented means to detect the loss of an APS NAL unit. As the APS identifier value may be re-used, any reference (e.g. from slice header or another APS NAL unit for partial updating of APS parameters) for the APS identifier value used in a lost APS NAL unit may point to the previous APS NAL unit using the same APS identifier value. Consequently, wrong syntax element values would be used e.g. in slice decoding process or in partial updating of APS

parameters. Such use of wrong syntax element values may have severe impacts in the decoding, e.g. clearly visible errors may be present in decoded pictures or decoding may fail altogether.

Increased memory consumption. One option to avoid the loss resilience problem presented in the previous paragraph could be to avoid re-using of APS identifier values in APS NAL units. However, this could potentially lead to a need for having a great or unlimited value range for APS identifier values. In the above-mentioned options for coding Adaptation Parameter Sets, the decoder keeps all Adaptation Parameter Sets in the memory unless the same APS identifier value is used as earlier, in which case the earlier Adaptation Parameter Set is replaced with the new one. Thus, a great or unlimited value range of APS identifier values would lead to increased memory consumption. Furthermore, the worst-case memory consumption could be difficult to define.

Transmission of APS NAL units is required to be synchronous with the video coding NAL units; otherwise, wrong APS parameter values may be used in decoding. As explained earlier, parameter sets have been designed for both out-of-band and in-band transmission, where the benefit of out-of-band transmission may be better error resilience thanks to the use of reliable transmission mechanisms. When transmitting parameter sets out-of-band, they have to be available prior to their activation – which is a well-known feature already from the SPS and PPS design of H.264/AVC – hence, a rough level of synchronization between parameter sets sent out-of-band and the video coding layer NAL units is needed. However, in document JCTVC-H0069 the de-referencing of a partially updated APS, i.e. resolving the source for each group of syntax elements, was proposed to be performed each time the APS is activated, typically once at the beginning of decoding a slice. Even if the APS NAL unit referred to by a slice header did not change compared to an earlier slice header, one of the APS NAL units referred to by the linked list created through the partial updating mechanism might have been re-sent and consequently some of the APS parameter values of the APS NAL unit referred to by the current slice header might have changed too. Consequently, transmission of APS NAL units has to be synchronized with VCL NAL units, because otherwise the de-referenced APS might differ in the encoder and in the decoder. Alternatively, the decoder has to synchronize the received APS NAL units with the VCL NAL units in the same order as the encoder created or used them.

In example embodiments, common notation for arithmetic operators, logical operators, relational operators, bit-wise operators, assignment operators, and range notation e.g. as specified in H.264/AVC or a draft HEVC may be used. Furthermore, common mathematical functions e.g. as specified in H.264/AVC or a draft HEVC may be used and a common order of precedence and execution order (from left to right or from right to left) of operators e.g. as specified in H.264/AVC or a draft HEVC may be used.

In example embodiments, the following descriptors may be used to specify the parsing process of each syntax element.

- b(8): byte having any pattern of bit string (8 bits).
- se(v): signed integer Exp-Golomb-coded syntax element with the left bit first.

– u(n): unsigned integer using n bits. When n is "v" in the syntax table, the number of bits varies in a manner dependent on the value of other syntax elements. The parsing process for this descriptor is specified by n next bits from the bitstream interpreted as a binary representation of an unsigned integer with the most significant bit written first.

– ue(v): unsigned integer Exp-Golomb-coded syntax element with the left bit first.

An Exp-Golomb bit string may be converted to a code number (codeNum) for example using the following table:

Bit string	codeNum
1	0
0 1 0	1
0 1 1	2
0 0 1 0 0	3
0 0 1 0 1	4
0 0 1 1 0	5
0 0 1 1 1	6
0 0 0 1 0 0 0	7
0 0 0 1 0 0 1	8
0 0 0 1 0 1 0	9
…	…

A code number corresponding to an Exp-Golomb bit string may be converted to se(v) for example using the following table:

codeNum	syntax element value
0	0
1	1
2	−1
3	2
4	−2
5	3
6	−3
…	…

In various embodiments, the encoder may encode or create APS NAL units, and the order of created APS NAL units is referred to as the APS decoding order. The APS identifier value in APS NAL units may be assigned according to a pre-defined numbering scheme in the APS decoding order. For example, the APS identifier value may be incremented by one for each APS in the APS decoding order. In some embodiments, the numbering scheme may be determined by the encoder and indicated for

example in the sequence parameter set. In some embodiments, the initial value of the numbering scheme may be pre-determined for example so that value 0 is used for the first APS NAL unit transmitted for a coded video sequence, while in other embodiments the initial value of the numbering scheme may be determined by the encoder. In some embodiments, the numbering scheme may depend on other syntax element values of the APS NAL unit, such as the values of temporal_id and nal_ref_flag. For example, the APS identifier value may be incremented by one relative to the previous APS NAL unit having the same temporal_id value as the current APS NAL unit being encoded. If an APS NAL unit is used only in one non-reference picture, the encoder may set the nal_ref_flag of the APS NAL unit to 0 and the APS identifier values may be incremented only relative to APS identifier values in APS NAL units having nal_ref_flag equal to 1. The APS identifier value may be coded with different coding schemes, which may be pre-determined in the coding standard, for example, or determined by the encoder and indicated for example in the sequence parameter set. For example, a variable length code, such as an unsigned integer Exp-Golomb code, ue(v), may be used for coding the APS identifier value in the APS syntax structure and whenever the APS identifier value is used to refer to an APS NAL unit. In another example, a fixed-length code, such as u(n), may be used where n may be pre-defined or determined by the encoder and indicated for example in the sequence parameter set. In some embodiments, the value range for the coded APS identifier value may be limited. The limits of the value range may be inferred from the coding of the APS identifier value. For example, if the APS identifier value is u(n)-coded, the value range may be inferred both in the encoder and in the decoder to be from 0 to n-1, inclusive. In some embodiments, the value range may be pre-defined for example in a coding standard or may be determined by the encoder and indicated for example in a sequence parameter set. For example, the APS identifier value may be ue(v)-coded and the value range may be defined to be from 0 to value N, where N is indicated through a syntax element in the sequence parameter set syntax structure. The APS identifier numbering scheme may use modulo arithmetic such that when the identifier exceeds the maximum value in the value range, it wraps over the minimum value in the value range. For example, if the APS identifiers are incremented by 1 in APS decoding order and the value range is from 0 to N, the value of the identifier may be determined to be (prevValue + 1) % (N+1), where prevValue is the previous APS identifier value and % indicates the modulo operation.

Thanks to the pre-defined or signaled numbering scheme for APS identifier values in the APS decoding order, losses and/or out-of-order delivery of APS NAL units can be detected in the receiving end for example by the decoder. In other words, the decoder may use the same APS identifier numbering scheme as the encoder used and hence conclude which APS identifier value should be present in the next received APS NAL unit. If an APS NAL unit with a different APS identifier value is received, a loss or out-of-order delivery may be concluded. In some embodiments, it may be allowed to repeat an APS NAL unit for error robustness – hence, no loss or out-of-order delivery should be concluded if an APS NAL unit is received with the same APS identifier value as that in the previous APS NAL unit in reception order. As explained above, the numbering scheme may depend on other parameter values in the APS

NAL unit, such as temporal_id and nal_ref_flag, in which case the APS identifier value of a received APS NAL unit may be compared to the expected value compared to the previous APS NAL unit meeting the qualifications defined in the numbering scheme. For example, in some embodiments a temporal_id based numbering scheme may be used and the decoder expects the APS identifier value to be incremented by 1 relative to the previous APS NAL unit having the same temporal_id value as that of the current APS NAL unit; if the decoder receives an APS NAL unit with another APS identifier value, it may conclude a loss and/or out-of-order delivery. In some embodiments, the receiver or the decoder or alike may include a buffer and/or a process for re-ordering APS NAL units from their reception order to their decoding order based on the numbering scheme used for the APS identifier values.

In some embodiments, however, a gap in APS identifier value may indicate an intentional removal or accidental loss of an APS NAL unit. An APS NAL unit may be intentionally removed for example through a sub-bitstream extraction process, which removes a scalable layer or view or alike from the bitstream. Thus, in some embodiments, a gap in expected APS identifier value assignments in APS NAL units may be handled by the decoder as follows. First, the missing APS identifier values between the previous APS identifier value and the current APS identifier value in APS NAL units in APS decoding order are concluded. For example, if the previous APS identifier value is 3 and the current APS identifier value is 6 and APS identifier values are incremented by one per each APS NAL unit according to the numbering scheme in use, APS NAL units with identifier values 4 and 5 may be concluded to be missing. The Adaptation Parameter Sets for the missing APS identifier values may be specifically marked for example as “non-existing”. If a “non-existing” APS is referred to in the decoding process, for example using the APS reference identifier in the slice header or through an APS partial updating mechanism, the decoder may conclude an accidental loss of an APS.

In the following, different options for determining which adaptation parameter sets are kept in the memory or buffer for encoding and decoding are described. It is noted that even though expressions such as “removed from the buffer” are used in the description, the adaptation parameter set may not be removed from the memory or buffer but just marked as invalid, unused, non-existing, inactivated, or anything alike so that it will no longer be used for encoding and/or decoding. Similarly, while expressions such as “kept in the buffer” may be used in the description, the adaptation parameter set may be maintained in any type of a memory arrangement or other storage and just associated with or marked as valid, used, existing, active, or anything alike so that it can be used in encoding and/or decoding. When validity of adaptation sets is examined or determined, those adaptation parameter sets that are “kept in the buffer” or marked as valid, used, existing, active, or anything alike may be determined as valid, and those adaptation parameter sets that have been “removed from the buffer” or marked as invalid, unused, non-existing, inactive, or anything alike may be determined as invalid.

In some embodiments, the maximum number of Adaptation Parameter Sets, referred to as max_aps, kept in the memory by the encoder and the decoder may be pre-determined for example by a coding standard or determined by the encoder and indicated in the coded bitstream for example in the

sequence parameter set. In some embodiments, both the encoder and the decoder may perform first-in-first-out buffering (also known as sliding window buffering) for adaptation parameter sets in a buffer memory that has max_aps slots, where one slot can hold one adaptation parameter set. The "non-existing" APS may take part in the sliding window buffering. When all the slots of the APS sliding-window buffer are occupied and a new APS is decoded, the eldest APS in APS decoding order is removed from the sliding-window buffer. In some embodiments the numbering scheme may depend on other parameters in the APS NAL unit and there may be more than one sliding-window buffer and decoder operation. For example, if the number scheme is specific to a temporal_id value, there may be a separate sliding-window buffer for each temporal_id value and max_aps may be indicated separately for each temporal_id value. In some embodiments, the encoder may code specific APS buffer management operations into the bitstream, such as removal of an APS with an indicated APS identifier value from the sliding-window buffer. The decoder decodes such APS buffer management operations and therefore maintains the APS sliding-window buffer state identically compared to that of the encoder. In some embodiments, certain adaptation parameter sets may be assigned by the encoder to be long-term adaptation parameter sets. Such long-term assignment may be done for example by using an APS identifier value that is outside the value range reserved for APS identifier values of regular adaptation parameter sets or through a specific APS buffer management operation. Long-term adaptation parameter sets are not subject to the sliding-window operation, i.e. a long-term adaptation parameter set is not removed from the sliding-window buffer even if it were the eldest in APS decoding order. The number or the maximum number of long-term APSes may be indicated for example in the sequence parameter set, or a decoder may infer the number based on the assignments of adaptation parameter sets as long-term. In some embodiments, the sliding-window buffer may be adjusted to have a number of slots equal to max_aps minus the number or the maximum number of long-term adaptation parameter sets. It may be required for example by a coding standard that a bitstream is encoded in a way that APS identifier values for long-term adaptation parameter sets are never reused within the same coded video sequence by another long-term adaptation parameter set. Alternatively, it may be required or encouraged that whenever an APS NAL unit is sent that overrides an earlier long-term adaptation parameter set, the transmission for that APS NAL unit is reliable.

In some embodiments, a value specifying the maximum APS identifier value difference that is kept in the memory by the encoder and the decoder may be pre-defined for example in a coding standard or may be determined by the encoder and indicated in the bitstream for example in a sequence parameter set. This value may be referred to as max_aps_id_diff. The encoder and the decoder may keep in the memory and/or mark as "used" only those adaptation parameter sets whose APS identifier value is within the limit determined by max_aps_id_diff relative to the APS identifier value of a particular adaptation parameter set, such as the latest APS NAL unit in APS decoding order or the latest APS NAL unit having temporal_id equal to 0 in APS decoding order. In the following example, it is assumed that APS identifiers have a definite value range from 0 to max_aps_id, inclusive, where the value of max_aps_id

may be pre-defined for example in a coding standard or may be determined by the encoder and indicated in the bitstream for example in a sequence parameter set. When an APS NAL unit with APS identifier value equal to curr_aps_id is encoded or decoded, the following may be performed by assigning rp_aps_id equal to curr_aps_id. If rp_aps_id >= max_aps_id_diff, all adaptation parameter sets with APS identifier value less than rp_aps_id – max_aps_id_diff and greater than rp_aps_id are removed from the buffer. If rp_aps_id < max_aps_id_diff, all adaptation parameter sets with APS identifier value greater than rp_aps_id and less than or equal to max_aps_id – (max_aps_id_diff – (rp_aps_id + 1)) are removed. The other adaptation parameter sets are kept in the memory/buffer. If such an adaptation parameter set that is removed from the memory/buffer is referred to in the decoding process, for example through APS identifier reference in the slice header or through a partial APS update mechanism, the decoder may conclude an accidental loss of the referred APS.

In some embodiments, the encoder and the decoder may maintain a reference point APS identifier value, rp_aps_id as follows. When the first APS NAL unit for a coded video sequence is encoded or decoded rp_aps_id is set to the APS identifier value of the first APS NAL unit. Each time when a subsequent APS NAL unit with APS identifier value equal to curr_aps_id is encoded or decoded in APS decoding order, rp_aps_id may be updated to curr_aps_id if curr_aps_id is incremented from rp_aps_id. As modulo arithmetic may be used for the APS identifier values, the comparison whether curr_aps_id has incremented relative to rp_aps_id may require taking into account the wraparound after max_aps_id. In order to distinguish between a curr_aps_id increment relative to rp_aps_id (in modulo arithmetic) from a curr_aps_id decrement relative to rp_aps_id, it may be considered that the maximum allowed decrement has a threshold, which may be equal to or relative to max_aps_id_diff or may be pre-defined for example in a coding standard or may be determined by the encoder and indicated in the bitstream for example in a sequence parameter set. For example, the following may be performed. If curr_aps_id > rp_aps_id and curr_aps_id < rp_aps_id + max_aps_id – threshold, rp_aps_id may be set to curr_aps_id. If curr_aps_id < rp_aps_id – threshold, rps_aps_id may be set to curr_aps_id. Otherwise, rp_aps_id is kept unchanged. Determining which adaptation parameter sets are removed from the memory and which ones are kept in the memory may be done as explained in the previous paragraph, with the difference that rp_aps_id is not assigned equal to curr_aps_id for each APS NAL unit but according to the scheme presented in this paragraph. The scheme presented in this paragraph may allow for example resending of APS NAL units for error resilience purposes.

In some embodiments, the encoder may determine the value of max_aps_id_diff or alike for each or some of the coded adaptation parameter sets and include max_aps_id_diff in the adaptation parameter set NAL unit. The decoder may then use the max_aps_id_diff in the adaptation parameter set NAL unit rather than equivalent syntax element elsewhere in the bitstream, such as in the sequence parameter set.

In some embodiments, an APS syntax structure may contain a reference set for adaptation parameter sets (APSRS), where each item in the set may be identified through an APS identifier value. The APSRS may determine the adaptation parameter sets that are kept in the buffer by the encoder and in

the decoder, while the other adaptation parameter sets having identifier values that are not in the APSRS are removed from the memory/buffer. If such an adaptation parameter set that is removed from the memory/buffer is referred to in the decoding process, for example through APS identifier reference in the slice header or through a partial APS update mechanism, the decoder may conclude an accidental loss of the referred APS. In some embodiments, particularly when sub-bitstream extraction has not been applied, if an APSRS contains an identifier value for an APS that is not in the buffer, the decoder may conclude an accidental loss of that APS.

In some embodiments, a picture of one or more specific types may cause removal of APS NAL units from the memory. For example, an IDR picture may cause all APS NAL units to be removed from the memory. In some example, a CRA picture may cause all APS NAL units to be removed from the memory.

In some embodiments, a partial APS updating mechanism may be enabled in the APS syntax structure for example as follows. For each group of syntax elements (e.g. QM, ALF, SAO, and deblocking filter parameters), the encoder may have one or more of the following options when coding an APS syntax structure:

- The group of syntax elements may be coded into an APS syntax structure, i.e. coded syntax element values of the syntax element set may be included in the APS parameter set syntax structure.
- The group of syntax elements may be included by reference into the APS. The reference may be given as an identifier to another APS. The encoder may use a different reference APS identifier for different groups syntax elements.
- The group of syntax elements set may be indicated or inferred to be absent from the APS.

The options from which the encoder is able to choose for a particular group of syntax elements when coding an APS may depend on the type of the syntax element group. For example, it may be required that syntax elements of a certain type syntax are always present in the APS syntax structure, while other groups of syntax elements may be included by reference or be present in the APS syntax structure. The encoder may encode indications in the bitstream, for example in an APS syntax structure, which option was used in encoding. The code table and/or entropy coding may depend on the type of the group of syntax elements. The decoder may use, based on the type of the group of syntax elements being decoded, the code table and/or entropy decoding that is matched with the code table and/or entropy encoding used by the encoder.

The encoder may have multiple means to indicate the association between a group of syntax elements and the APS used as the source for the values of the syntax element set. For example, the encoder may encode a loop of syntax elements where each loop entry is encoded as syntax elements indicating an APS identifier value used as a reference and identifying the syntax element sets copied from the reference APS. In another example, the encoder may encode a number of syntax elements, each

indicating an APS. The last APS in the loop containing a particular group of syntax elements is the reference for that group of syntax elements in APS the encoder is currently encoding into the bitstream. The decoder parses the encoded adaptation parameter sets from the bitstream accordingly so as to reproduce the same adaptation parameter sets as the encoder.

In some embodiments, the requirements for synchronizing or ordering of APS NAL units with VCL NAL units are as follows. If APS NAL units are transmitted out-of-band, it is sufficient that the decoding order APS NAL units is maintained during transmission or APS decoding order is reconstructed in the receiving end with buffering for example as explained above. Additionally, the out-of-band transmission mechanism and/or the synchronization mechanism should be such that an APS NAL unit is provided to decoding before the APS NAL unit is referred from a VCL NAL unit, such as from a coded slice NAL unit. If APS identifier values are re-used, the transmission and/or synchronization mechanism should take care that an APS NAL unit is not decoded before NAL unit containing the last reference to the previous APS NAL unit having the same identifier value is decoded. However, there is no need for accurate synchronization, such as being able to resolve the respective encoding order of APS and VCL NAL units as required in the partial updating scheme of JCTVC-H0069. The synchronization or ordering of APS NAL units with VCL NAL units meeting the above-mentioned requirements may be performed by various means. For example, all the adaptation parameter sets needed for decoding of all pictures in the first coded video sequence or GOP may be transmitted in the session establishment phase and are hence available for decoding when the session has been established and first VCL data arrives for decoding. Adaptation parameter sets for the subsequent coded video sequence or GOP may be done immediately after that using different identifier values than those used for the first coded video sequence or GOP. Hence, the adaptation parameter sets for the second coded video sequence or GOP are transmitted, while the VCL data of the first coded video sequence or GOP is transmitted. The transmission of adaptation parameter sets for subsequent coded video sequences or GOPs may be handled similarly.

In some embodiments, the dereferencing or decoding of the APS NAL units may be done at any time prior to the APS is referred to from a VCL NAL unit as long as APS NAL units are decoded in the APS decoding order. The decoding of an APS NAL unit may be done by resolving the references and copying the referenced groups of syntax elements into the APS being decoded. In some embodiments, the dereferencing or decoding of an APS NAL unit may be done when a VCL NAL unit refers to it the first time. In some embodiments, the dereferencing or decoding of an APS NAL unit may be done each time when a VCL NAL unit refers to it.

In example embodiments, syntax structures, semantics of syntax elements, and decoding process may be specified as follows. Syntax elements in the bitstream are represented in **bold** type. Each syntax element is described by its name (all lower case letters with underscore characters), optionally its one or two syntax categories, and one or two descriptors for its method of coded representation. The decoding process behaves according to the value of the syntax element and to the values of previously decoded

syntax elements. When a value of a syntax element is used in the syntax tables or the text, it appears in regular (i.e., not bold) type. In some cases the syntax tables may use the values of other variables derived from syntax elements values. Such variables appear in the syntax tables, or text, named by a mixture of lower case and upper case letter and without any underscore characters. Variables starting with an upper case letter are derived for the decoding of the current syntax structure and all depending syntax structures. Variables starting with an upper case letter may be used in the decoding process for later syntax structures without mentioning the originating syntax structure of the variable. Variables starting with a lower case letter are only used within the context in which they are derived. In some cases, "mnemonic" names for syntax element values or variable values are used interchangeably with their numerical values. Sometimes "mnemonic" names are used without any associated numerical values. The association of values and names is specified in the text. The names are constructed from one or more groups of letters separated by an underscore character. Each group starts with an upper case letter and may contain more upper case letters.

In example embodiments, a syntax structure may be specified using the following. A group of statements enclosed in curly brackets is a compound statement and is treated functionally as a single statement. A "while" structure specifies a test of whether a condition is true, and if true, specifies evaluation of a statement (or compound statement) repeatedly until the condition is no longer true. A "do … while" structure specifies evaluation of a statement once, followed by a test of whether a condition is true, and if true, specifies repeated evaluation of the statement until the condition is no longer true. An "if … else" structure specifies a test of whether a condition is true, and if the condition is true, specifies evaluation of a primary statement, otherwise, specifies evaluation of an alternative statement. The "else" part of the structure and the associated alternative statement is omitted if no alternative statement evaluation is needed. A "for" structure specifies evaluation of an initial statement, followed by a test of a condition, and if the condition is true, specifies repeated evaluation of a primary statement followed by a subsequent statement until the condition is no longer true.

In some embodiments the syntax of the sequence parameter set syntax structure may be appended to include max_aps_id and max_aps_id_diff syntax elements as follows.

seq_parameter_set_rbsp( ) {	**Descriptor**
...	
max_aps_id	ue(v)
if(max_aps_id > 0)	
max_aps_id_diff	ue(v)
...	

The semantics of max_aps_id and max_aps_id_diff syntax elements may be specified as follows. **max_aps_id** specifies the maximum allowed aps_id value. **max_aps_id_diff** specifies the value range of aps_id values of adaptation parameter sets marked as “used”.

The syntax of an Adaptation Parameter Set RBSP, aps_rbsb( ), may be specified in some example embodiments as follows:

aps_rbsp( ) {	**Descriptor**
aps_id	ue(v)
partial_update_flag	u(1)
if(partial_update_flag) {	
common_reference_aps_flag	u(1)
if(common_reference_aps_flag)	
common_reference_aps_id	ue(v)
}	
aps_scaling_list_data_present_flag	u(1)
if(aps_scaling_list_data_present_flag) {	
if(partial_update_flag)	
aps_scaling_list_data_referenced_flag	u(1)
if(!partial_update_flag || !aps_scaling_list_data_referenced_flag)	
scaling_list_param( )	
else if(!common_reference_aps_flag)	
aps_scaling_list_data_reference_aps_id	ue(v)
}	
aps_deblocking_filter_flag	u(1)
if(aps_deblocking_filter_flag) {	
if(partial_update_flag)	
aps_deblocking_filter_referenced_flag	u(1)
if(!partial_update_flag || !aps_deblocking_filter_referenced_flag) {	
disable_deblocking_filter_flag	u(1)
if(!disable_deblocking_filter_flag) {	
beta_offset_div2	se(v)
tc_offset_div2	se(v)
}	
} else if(!common_reference_aps_flag)	
aps_deblocking_filter_reference_aps_id	ue(v)
}	
aps_sao_interleaving_flag	u(1)
if(!aps_sao_interleaving_flag) {	
aps_sample_adaptive_offset_flag	u(1)
if(aps_sample_adaptive_offset_flag) {	
if(partial_update_flag)	

aps_sao_referenced_flag	u(1)
if(!partial_update_flag || !aps_sao_referenced_flag)	
aps_sao_param( )	
else if(!common_reference_aps_flag)	
aps_sao_reference_aps_id	ue(v)
}	
aps_adaptive_loop_filter_flag	u(1)
if(aps_adaptive_loop_filter_flag) {	
if(partial_update_flag)	
aps_alf_referenced_flag	u(1)
if(!partial_update_flag || !aps_alf_referenced_flag)	
alf_param( )	
else if(!common_reference_aps_flag)	
aps_alf_reference_aps_id	ue(v)
}	
aps_extension_flag	u(1)
if(aps_extension_flag)	
while(more_rbsp_data( ))	
aps_extension_data_flag	u(1)
rbsp_trailing_bits( )	
}	

The semantics of aps_rbsp( ) may be specified as follows.

aps_id specifies an identifier value that identifies the adaptation parameter set.

partial_update_flag equal to 0 specifies that no syntax element is included in this APS by reference. partial_update_flag equal to 1 specifies that syntax elements may be included in this APS by reference.

common_reference_aps_flag equal to 0 specifies that each group of syntax elements included by reference in this APS may have a different source APS identified by a different APS identifier value. common_reference_aps_flag equal to 1 specifies that each group of syntax elements included by reference in this APS are from the same source APS.

common_reference_aps_id specifies the APS identifier value for the source APS for all groups of syntax elements included in this APS by reference.

aps_scaling_list_data_present_flag equal to 1 specifies that the scaling list parameters exist in this APS, equal to 0 specifies that scaling list parameters do not exist in this APS.

aps_scaling_list_data_referenced_flag equal to 0 specifies that the scaling list parameters are present in this aps_rbsp( ). aps_scaling_list_data_referenced_flag equal to 1 specifies that the scaling list parameters are included in this APS by reference.

aps_scaling_list_data_reference_aps_id specifies the APS identifier value for the APS from which the scaling list parameters are included in this APS by reference.

aps_deblocking_filter_flag equal to 1 specifies that deblocking parameters are present in the APS. aps_deblocking_filter_flag equal to 0 specifies that deblocking parameters do not exist in this APS.

aps_deblocking_filter_referenced_flag equal to 0 specifies that the deblocking parameters are present in this aps_rbsp( ). aps_deblocking_filter_referenced_flag equal to 1 specifies that the deblocking parameters are included in this APS by reference.

aps_deblocking_filter_reference_aps_id specifies the APS identifier value for the APS from which the deblocking parameters are included in this APS by reference.

aps_sao_interleaving_flag equal to 1 specifies that the SAO parameters are interleaved in slice data for slices referring to the current APS; equal to 0 specifies that the SAO parameters are in APS for slices referring to the current APS. When there is no active APS, aps_sao_interleaving_flag is inferred to be 0.

aps_sample_adaptive_offset_flag equal to 1 specifies that the SAO is on for slices referring to the current APS; equal to 0 specifies that the SAO is off for slices referring to the current APS. When there is no active APS, the aps_sample_adaptive_offset_flag value is inferred to be 0.

aps_sao_referenced_flag equal to 0 specifies that the SAO parameters are present in this aps_rbsp( ). aps_sao_referenced_flag equal to 1 specifies that the SAO parameters are included in this APS by reference.

aps_sao_reference_aps_id specifies the APS identifier value for the APS from which the SAO parameters are included in this APS by reference.

aps_adaptive_loop_filter_flag equal to 1 specifies that the ALF is on for slices referring to the current APS; equal to 0 specifies that the ALF is off for slices referring to the current APS. When there is no active APS, the aps_adaptive_loop_filter_flag value is inferred to be 0.

aps_alf_referenced_flag equal to 0 specifies that the ALF parameters are present in this aps_rbsp( ). aps_alf_referenced_flag equal to 1 specifies that the ALF parameters are included in this APS by reference.

aps_alf_reference_aps_id specifies the APS identifier value for the APS from which the ALF parameters are included in this APS by reference.

aps_extension_flag equal to 0 specifies that no aps_extension_data_flag syntax elements are present in the picture parameter set RBSP syntax structure. aps_extension_flag shall be equal to 0 in bitstreams conforming to this Recommendation | International Standard. The value of 1 for aps_extension_flag is reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore all data that follow the value 1 for aps_extension_flag in a picture parameter set NAL unit.

aps_extension_data_flag may have any value. Its value does not affect decoder conformance to profiles specified in this Recommendation | International Standard.

In some embodiments, all or some adaptation parameter set identifiers and related syntax elements, such as aps_id, common_reference_aps_id, aps_XXX_referenced_aps_id (with XXX being equal to scaling_list_data, deblocking_filter, alf, or sao), and max_aps_id_diff, may be coded as u(v). The length of the mentioned u(v)-coded syntax elements may be determined by the value of max_aps_id. For example, Ceil(Log2(max_aps_id + 1) bits may be used for these syntax elements, where Ceil(x) is the smallest integer greater than or equal to x and Log2(x) returns the base-2 logarithm of x. As max_aps_id is included in the sequence parameter set in many example embodiments, the adaptation parameter set syntax structure may be appended to contain an identifier for the active sequence parameter set.

In some embodiments, the aps_rbsp( ) syntax structure or alike may be extended for example through aps_extension_flag equal to 1. The extension may be used for example to carry groups of syntax elements related to scalable, multiview, or 3D extensions. An APS syntax structure with aps_extension_flag equal to 0 may include by reference those types of groups of syntax elements that are included in aps_rbsp( ) syntax structure with aps_extension_flag equal to 0 even if aps_extension_flag were equal to 1 in the referred APS.

In some embodiments an adaptation parameter set NAL unit may be decoded using the following ordered steps:

- Let currApsId be equal to the aps_id value of the adaptation parameter set NAL unit being decoded.
- When currApsId is greater than or equal to max_aps_id_diff, all adaptation parameter sets with aps_id value less than currApsId – max_aps_id_diff and greater than currApsId are marked as "unused".
- When currApsId is smaller than max_aps_id_diff, all adaptation parameter sets with aps_id value greater than currApsId and less than or equal to max_aps_id – (max_aps_id_diff – (currApsId + 1)) are marked as "unused".
- When partial_update_flag is equal to 1 and aps_scaling_list_data_referenced_flag is equal to 1, the values of syntax elements in the scaling_list_param( ) syntax structure are inferred to have the same values as in the scaling_list_param( ) syntax structure for the APS NAL unit with aps_id equal to common_reference_aps_id, if present, or aps_scaling_list_data_reference_aps_id, otherwise.
- When partial_update_flag is equal to 1 and aps_deblocking_filter_flag is equal to 1, the values of disable_deblocking_filter_flag, beta_offset_div2, and tc_offset_div2 are inferred to have the same values, respectively, as disable_deblocking_filter_flag, beta_offset_div2, if present, and tc_offset_div2, if present, in the APS NAL unit with aps_id equal to common_reference_aps_id, if present, or aps_deblocking_filter_reference_aps_id, otherwise.

- When partial_update_flag is equal to 1, aps_sao_interleaving_flag is 0, and aps_sample_adaptive_offset_flag is equal to 1, the values of syntax elements in the aps_sao_param( ) syntax structure are inferred to have the same values as in the aps_sao_param( ) syntax structure for the APS NAL unit with aps_id equal to common_reference_aps_id, if present, or aps_sao_reference_aps_id, otherwise.
- When partial_update_flag is equal to 1 and aps_adaptive_loop_filter_flag is equal to 1, the values of syntax elements in the alf_param( ) syntax structure are inferred to have the same values as in the alf_param( ) syntax structure for the APS NAL unit with aps_id equal to common_reference_aps_id, if present, or aps_alf_reference_aps_id, otherwise.
- The adaptation parameter set NAL unit being decoded is marked as "used".

In the above, the example embodiments have been described with the help of syntax of the bitstream. It needs to be understood, however, that the corresponding structure and/or computer program may reside at the encoder for generating the bitstream and/or at the decoder for decoding the bitstream. Likewise, where the example embodiments have been described with reference to an encoder, it needs to be understood that the resulting bitstream and the decoder have corresponding elements in them. Likewise, where the example embodiments have been described with reference to a decoder, it needs to be understood that the encoder has structure and/or computer program for generating the bitstream to be decoded by the decoder.

In the above, embodiments have been described in relation to adaptation parameter set. It needs to be understood, however, that embodiments could be realized with any type of parameter set, such as GOS parameter set, picture parameter, and sequence parameter set.

Although the above examples describe embodiments of the invention operating within a codec within an electronic device, it would be appreciated that the invention as described below may be implemented as part of any video codec. Thus, for example, embodiments of the invention may be implemented in a video codec which may implement video coding over fixed or wired communication paths.

Thus, user equipment may comprise a video codec such as those described in embodiments of the invention above. It shall be appreciated that the term user equipment is intended to cover any suitable type of wireless user equipment, such as mobile telephones, portable data processing devices or portable web browsers.

Furthermore elements of a public land mobile network (PLMN) may also comprise video codecs as described above.

In general, the various embodiments of the invention may be implemented in hardware or special purpose circuits, software, logic or any combination thereof. For example, some aspects may be implemented in hardware, while other aspects may be implemented in firmware or software which may be executed by a controller, microprocessor or other computing device, although the invention is not limited thereto. While various aspects of the invention may be illustrated and described as block

diagrams, flow charts, or using some other pictorial representation, it is well understood that these blocks, apparatuses, systems, techniques or methods described herein may be implemented in, as non-limiting examples, hardware, software, firmware, special purpose circuits or logic, general purpose hardware or controller or other computing devices, or some combination thereof.

The embodiments of this invention may be implemented by computer software executable by a data processor of the mobile device, such as in the processor entity, or by hardware, or by a combination of software and hardware. Further in this regard it should be noted that any blocks of the logic flow as in the Figures may represent program steps, or interconnected logic circuits, blocks and functions, or a combination of program steps and logic circuits, blocks and functions. The software may be stored on such physical media as memory chips, or memory blocks implemented within the processor, magnetic media such as hard disk or floppy disks, and optical media such as for example DVD and the data variants thereof, CD.

The various embodiments of the invention can be implemented with the help of computer program code that resides in a memory and causes the relevant apparatuses to carry out the invention. For example, a terminal device may comprise circuitry and electronics for handling, receiving and transmitting data, computer program code in a memory, and a processor that, when running the computer program code, causes the terminal device to carry out the features of an embodiment. Yet further, a network device may comprise circuitry and electronics for handling, receiving and transmitting data, computer program code in a memory, and a processor that, when running the computer program code, causes the network device to carry out the features of an embodiment.

The memory may be of any type suitable to the local technical environment and may be implemented using any suitable data storage technology, such as semiconductor-based memory devices, magnetic memory devices and systems, optical memory devices and systems, fixed memory and removable memory. The data processors may be of any type suitable to the local technical environment, and may include one or more of general purpose computers, special purpose computers, microprocessors, digital signal processors (DSPs) and processors based on multi-core processor architecture, as non-limiting examples.

Embodiments of the inventions may be practiced in various components such as integrated circuit modules. The design of integrated circuits is by and large a highly automated process. Complex and powerful software tools are available for converting a logic level design into a semiconductor circuit design ready to be etched and formed on a semiconductor substrate.

Programs, such as those provided by Synopsys Inc., of Mountain View, California and Cadence Design, of San Jose, California automatically route conductors and locate components on a semiconductor chip using well established rules of design as well as libraries of pre-stored design modules. Once the design for a semiconductor circuit has been completed, the resultant design, in a standardized electronic format (e.g., Opus, GDSII, or the like) may be transmitted to a semiconductor fabrication facility or "fab" for fabrication.

The foregoing description has provided by way of exemplary and non-limiting examples a full and informative description of the exemplary embodiment of this invention. However, various modifications and adaptations may become apparent to those skilled in the relevant arts in view of the foregoing description, when read in conjunction with the accompanying drawings and the appended claims. However, all such and similar modifications of the teachings of this invention will still fall within the scope of this invention.

In the following some examples will be provided.

According to a first example there is provided a method comprising:
receiving a first parameter set;
obtaining an identifier of the first parameter set;
receiving a second parameter set;
determining the validity of the first parameter set on the basis of at least one of the following:

- receiving in the second parameter set a list of valid identifier values; and determining that the first parameter set is valid, if the identifier of the first parameter set is in the list of valid parameter values;
- receiving in the second parameter set an identifier of the second parameter set; and determining that the first parameter set is valid based on the identifier of the first parameter set and the identifier of the second parameter set.

In some embodiments the method comprises defining a valid range of identifier values.

In some embodiments the method comprises:
defining a maximum difference of identifier values; and
defining a maximum identifier value;
wherein the method comprises determining that the first parameter set is valid, if one of the following conditions is true:

- the identifier of the second parameter set is greater than the identifier of the first parameter set and the difference between the identifier of the second parameter set and the identifier of the first parameter set is smaller than or equal to the maximum difference of identifier values;
- the identifier of the first parameter set is greater than the identifier of the second parameter set and the identifier of the second parameter set is smaller than or equal to the maximum difference of identifiers and the difference between the identifier of the first parameter set and the identifier of the second parameter set is greater than the difference between the maximum identifier value and the maximum difference of identifier values.

In some embodiments the method comprises using the difference between the identifier of the second parameter set and the identifier of the first parameter set to determine whether a third parameter set encoded between the first parameter set and the second parameter set has not been received.

In some embodiments the method comprises:
decoding the second parameter set;
examining whether the second parameter set comprises a reference to the first parameter set which has not been determined valid.

In some embodiments the method comprises:
buffering the first parameter set and the second parameter set into a buffer; and
marking the first parameter set unused if it is determined not valid.

According to a second example there is provided a method comprising:
encoding a first parameter set;
attaching an identifier of the first parameter set to the first parameter set;
encoding a second parameter set;
determining the validity of the first parameter set on the basis of at least one of the following:

- attaching in the second parameter set a list of valid identifier values; and determining that the first parameter set is valid, if the identifier of the first parameter set is in the list of valid parameter values;
- attaching in the second parameter set an identifier of the second parameter set; and determining that the first parameter set is valid based on the identifier of the first parameter set and the identifier of the second parameter set.

In some embodiments the method comprises defining a valid range of identifier values.

In some embodiments the method comprises selecting the identifier from the valid range of identifier values.

In some embodiments the method comprises:
defining a maximum difference of identifier values; and
defining a maximum identifier value.

In some embodiments the method comprises setting the identifier of the second parameter set different from the identifier from the first parameter set, if the first parameter set has been determined valid.

In some embodiments the method comprises:

allowing the second parameter set refer to the first parameter set, if the first parameter set has been determined valid.

According to a third example there is provided an apparatus comprising at least one processor and at least one memory including computer program code, the at least one memory and the computer program code configured to, with the at least one processor, cause the apparatus to:

receive a first parameter set;

obtain an identifier of the first parameter set;

receive a second parameter set; and

determine the validity of the first parameter set on the basis of at least one of the following:

- by receiving in the second parameter set a list of valid identifier values; and determining that the first parameter set is valid, if the identifier of the first parameter set is in the list of valid parameter values;
- by receiving in the second parameter set an identifier of the second parameter set; and determining that the first parameter set is valid based on the identifier of the first parameter set and the identifier of the second parameter set.

In some embodiments of the apparatus said at least one memory stored with code thereon, which when executed by said at least one processor, further causes the apparatus to define a valid range of identifier values.

In some embodiments of the apparatus said at least one memory stored with code thereon, which when executed by said at least one processor, further causes the apparatus to:

define a maximum difference of identifier values;

define a maximum identifier value; and

determine that the first parameter set is valid, if one of the following conditions is true:

- the identifier of the second parameter set is greater than the identifier of the first parameter set and the difference between the identifier of the second parameter set and the identifier of the first parameter set is smaller than or equal to the maximum difference of identifier values;
- the identifier of the first parameter set is greater than the identifier of the second parameter set and the identifier of the second parameter set is smaller than or equal to the maximum difference of identifier values and the difference between the identifier of the first parameter set and the identifier of the second parameter set is greater than the difference between the maximum identifier value and the maximum difference of identifier values.

In some embodiments of the apparatus said at least one memory stored with code thereon, which when executed by said at least one processor, further causes the apparatus to use the difference between the identifier of the second parameter set and the identifier of the first parameter set to determine whether a third parameter set encoded between the first parameter set and the second parameter set has not been received.

In some embodiments of the apparatus said at least one memory stored with code thereon, which when executed by said at least one processor, further causes the apparatus to:

decode the second parameter set; and

examine whether the second parameter set comprises a reference to the first parameter set which has not been determined valid.

In some embodiments of the apparatus said at least one memory stored with code thereon, which when executed by said at least one processor, further causes the apparatus to:

buffer the first parameter set and the second parameter set into a buffer; and

mark the first parameter set unused if it is determined not valid.

According to a fourth example there is provided an apparatus comprising at least one processor and at least one memory including computer program code, the at least one memory and the computer program code configured to, with the at least one processor, cause the apparatus to:

encode a first parameter set;

attach an identifier of the first parameter set to the first parameter set;

encode a second parameter set; and

determine the validity of the first parameter set on the basis of at least one of the following:

- by attaching in the second parameter set a list of valid identifier values; and determining that the first parameter set is valid, if the identifier of the first parameter set is in the list of valid parameter values;
- by attaching in the second parameter set an identifier of the second parameter set; and determining that the first parameter set is valid based on the identifier of the first parameter set and the identifier of the second parameter set.

In some embodiments of the apparatus said at least one memory stored with code thereon, which when executed by said at least one processor, further causes the apparatus to define a valid range of identifier values.

In some embodiments of the apparatus said at least one memory stored with code thereon, which when executed by said at least one processor, further causes the apparatus to select the identifier from the valid range of identifier values.

In some embodiments of the apparatus said at least one memory stored with code thereon, which when executed by said at least one processor, further causes the apparatus to:

define a maximum difference of identifier values; and

define a maximum identifier value.

In some embodiments of the apparatus said at least one memory stored with code thereon, which when executed by said at least one processor, further causes the apparatus to set the identifier of the second parameter set different from the identifier from the first parameter set, if the first parameter set has been determined valid.

In some embodiments of the apparatus said at least one memory stored with code thereon, which when executed by said at least one processor, further causes the apparatus to allow the second parameter set refer to the first parameter set, if the first parameter set has been determined valid.

According to a fifth example there is provided a computer program product including one or more sequences of one or more instructions which, when executed by one or more processors, cause an apparatus to at least perform the following:

receive a first parameter set;

obtain an identifier of the first parameter set;

receive a second parameter set; determine the validity of the first parameter set on the basis of at least one of the following:

- receiving in the second parameter set a list of valid identifier values; and determining that the first parameter set is valid, if the identifier of the first parameter set is in the list of valid parameter values;
- receiving in the second parameter set an identifier of the second parameter set; and determining that the first parameter set is valid based on the identifier of the first parameter set and the identifier of the second parameter set.

In some embodiments the computer program product includes one or more sequences of one or more instructions which, when executed by one or more processors, cause the apparatus to at least define a valid range of identifier values.

In some embodiments the computer program product includes one or more sequences of one or more instructions which, when executed by one or more processors, cause the apparatus to at least:

define a maximum difference of identifier values;

define a maximum identifier value; and

determine that the first parameter set is valid, if one of the following conditions is true:

- the identifier of the second parameter set is greater than the identifier of the first parameter set and the difference between the identifier of the second parameter set and the identifier of the first parameter set is smaller than or equal to the maximum difference of identifier values;
- the identifier of the first parameter set is greater than the identifier of the second parameter set and the identifier of the second parameter set is smaller than or equal to the maximum difference of identifier values and the difference between the identifier of the first parameter set and the identifier of the second parameter set is greater than the difference between the maximum identifier value and the maximum difference of identifier values.

In some embodiments the method comprises using the difference between the identifier of the second parameter set and the identifier of the first parameter set to determine whether a third parameter set encoded between the first parameter set and the second parameter set has not been received.

In some embodiments the computer program product includes one or more sequences of one or more instructions which, when executed by one or more processors, cause the apparatus to at least:

decode the second parameter set;

examine whether the second parameter set comprises a reference to the first parameter set which has not been determined valid.

In some embodiments the computer program product includes one or more sequences of one or more instructions which, when executed by one or more processors, cause the apparatus to at least:

buffer the first parameter set and the second parameter set into a buffer; and

mark the first parameter set unused if it is determined not valid.

According to a sixth example there is provided a computer program product including one or more sequences of one or more instructions which, when executed by one or more processors, cause an apparatus to at least perform the following:

encode a first parameter set;

attach an identifier of the first parameter set;

encode a second parameter set; determine the validity of the first parameter set on the basis of at least one of the following:

- by attaching in the second parameter set a list of valid identifier values; and determining that the first parameter set is valid, if the identifier of the first parameter set is in the list of valid parameter values;
- by attaching in the second parameter set an identifier of the second parameter set; and determining that the first parameter set is valid based on the identifier of the first parameter set and the identifier of the second parameter set.

In some embodiments the computer program product includes one or more sequences of one or more instructions which, when executed by one or more processors, cause the apparatus to at least define a valid range of identifier values.

In some embodiments the computer program product includes one or more sequences of one or more instructions which, when executed by one or more processors, cause the apparatus to at least select the identifier from the valid range of identifier values.

In some embodiments the computer program product includes one or more sequences of one or more instructions which, when executed by one or more processors, cause the apparatus to at least:

define a maximum difference of identifier values; and

define a maximum identifier value.

In some embodiments the computer program product includes one or more sequences of one or more instructions which, when executed by one or more processors, cause the apparatus to at least set the identifier of the second parameter set different from the identifier from the first parameter set, if the first parameter set has been determined valid.

In some embodiments the computer program product includes one or more sequences of one or more instructions which, when executed by one or more processors, cause the apparatus to at least allow the second parameter set refer to the first parameter set, if the first parameter set has been determined valid.

According to a seventh example there is provided an apparatus comprising:

means for receiving a first parameter set;

means for obtaining an identifier of the first parameter set;

means for receiving a second parameter set; means for determining the validity of the first parameter set on the basis of at least one of the following:

- by receiving in the second parameter set a list of valid identifier values; and determining that the first parameter set is valid, if the identifier of the first parameter set is in the list of valid parameter values;
- by receiving in the second parameter set an identifier of the second parameter set; and determining that the first parameter set is valid based on the identifier of the first parameter set and the identifier of the second parameter set.

According to an eighth example there is provided an apparatus comprising:
means for encoding a first parameter set;
means for attaching an identifier of the first parameter set;
means for encoding a second parameter set; and
means for determining the validity of the first parameter set on the basis of at least one of the following:

- by attaching in the second parameter set a list of valid identifier values; and determining that the first parameter set is valid, if the identifier of the first parameter set is in the list of valid parameter values;
- by attaching in the second parameter set an identifier of the second parameter set; and determining that the first parameter set is valid based on the identifier of the first parameter set and the identifier of the second parameter set.

According to a ninth example there is provided a video decoder configured for:
receiving a first parameter set;
obtaining an identifier of the first parameter set;
receiving a second parameter set; determining the validity of the first parameter set on the basis of at least one of the following:

- receiving in the second parameter set a list of valid identifier values; and determining that the first parameter set is valid, if the identifier of the first parameter set is in the list of valid parameter values;
- receiving in the second parameter set an identifier of the second parameter set; and determining that the first parameter set is valid based on the identifier of the first parameter set and the identifier of the second parameter set.

According to a tenth example there is provided a video encoder configured for:
encoding a first parameter set;
attaching an identifier of the first parameter set to the first parameter set;
encoding a second parameter set;
determining the validity of the first parameter set on the basis of at least one of the following:

- attaching in the second parameter set a list of valid identifier values; and determining that the first parameter set is valid, if the identifier of the first parameter set is in the list of valid parameter values;
- attaching in the second parameter set an identifier of the second parameter set; and determining that the first parameter set is valid based on the identifier of the first parameter set and the identifier of the second parameter set.

We claim:

1. A method comprising:
receiving a first parameter set;
obtaining an identifier of the first parameter set;
receiving a second parameter set;
determining the validity of the first parameter set on the basis of at least one of the following:
- receiving in the second parameter set a list of valid identifier values; and determining that the first parameter set is valid, if the identifier of the first parameter set is in the list of valid parameter values;
- receiving in the second parameter set an identifier of the second parameter set; and determining that the first parameter set is valid based on the identifier of the first parameter set and the identifier of the second parameter set.

2. The method according to claim 1 further comprising defining a valid range of identifier values.

3. The method according to claim 2, wherein the defining of the valid range of identifier values further comprises:
defining a reference point identifier; and
defining the valid range of identifier values on the basis of the reference point identifier.

4. The method according to claim 3 further comprising:
receiving a third parameter set;
obtaining an identifier of the third parameter set to the third parameter set, the identifier incremented relative to the reference point identifier; and
setting the reference point identifier to the identifier of the third parameter set.

5. The method according to claim 2 further comprising:
defining a maximum difference of identifier values; and
defining a maximum identifier value;
wherein the method comprises determining that the first parameter set is valid, if one of the following conditions is true:
- the identifier of the second parameter set is greater than the identifier of the first parameter set and the difference between the identifier of the second parameter set and the identifier of the first parameter set is smaller than or equal to the maximum difference of identifier values;
- the identifier of the first parameter set is greater than the identifier of the second parameter set and the identifier of the second parameter set is smaller than or equal to the maximum difference

of identifiers and the difference between the identifier of the first parameter set and the identifier of the second parameter set is greater than the difference between the maximum identifier value and the maximum difference of identifier values.

6. The method according to any of the claims 1 to 5 further comprising using the difference between the identifier of the second parameter set and the identifier of the first parameter set to determine whether a third parameter set encoded between the first parameter set and the second parameter set has not been received.

7. The method according to any of the claims 1 to 6 further comprising:
decoding an identifier reference of a parameter set to be used in decoding;
examining whether the identifier reference is within the valid range of identifier values.

8. The method according to claim 7 further comprising:
decoding the identifier reference from the second parameter set, wherein the identifier reference is to be used in decoding of the second parameter set.

9. The method according to claim 7 or 8 further comprising
concluding a loss of a parameter set on the basis that the identifier reference is outside the valid range of identifier values.

10. The method according to any of the claims 1 to 9 further comprising:
buffering the first parameter set and the second parameter set into a buffer; and
marking the first parameter set unused if it is determined not valid.

11. A method comprising:
encoding a first parameter set;
attaching an identifier of the first parameter set to the first parameter set;
encoding a second parameter set;
determining the validity of the first parameter set on the basis of at least one of the following:

- attaching in the second parameter set a list of valid identifier values; and determining that the first parameter set is valid, if the identifier of the first parameter set is in the list of valid parameter values;
- attaching in the second parameter set an identifier of the second parameter set; and determining that the first parameter set is valid based on the identifier of the first parameter set and the identifier of the second parameter set.

12. The method according to claim 11 further comprising defining a valid range of identifier values.

13. The method according to claim 12 wherein the defining of the valid range of identifier values further comprises
defining a reference point identifier; and
defining the valid range of identifier values on the basis of the reference point identifier.

14. The method according to claim 13 further comprising:
encoding a third parameter set;
attaching an identifier of the third parameter set to the third parameter set, the identifier incremented relative to the reference point identifier; and
setting the reference point identifier to the identifier of the third parameter set.

15. The method according to claim 12, 13 or 14 further comprising encoding an identifier reference of a parameter set to be used in decoding, selecting the identifier reference from the valid range of identifier values.

16. The method according to any of the claims 11 to 15 further comprising:
defining a maximum difference of identifier values; and
defining a maximum identifier value.

17. The method according to any of the claims 7 to 16 further comprising setting the identifier of the second parameter set different from the identifier from the first parameter set, if the first parameter set has been determined valid.

18. The method according to any of the claims 7 to 17 further comprising:
allowing the second parameter set refer to the first parameter set, if the first parameter set has been determined valid.

19. An apparatus comprising at least one processor and at least one memory including computer program code, the at least one memory and the computer program code configured to, with the at least one processor, cause the apparatus to:
receive a first parameter set;
obtain an identifier of the first parameter set;
receive a second parameter set; and
determine the validity of the first parameter set on the basis of at least one of the following:

- by receiving in the second parameter set a list of valid identifier values; and determining that the first parameter set is valid, if the identifier of the first parameter set is in the list of valid parameter values;
- by receiving in the second parameter set an identifier of the second parameter set; and determining that the first parameter set is valid based on the identifier of the first parameter set and the identifier of the second parameter set.

20. The apparatus according to claim 19, said at least one memory stored with code thereon, which when executed by said at least one processor, further causes the apparatus to define a valid range of identifier values.

21. The apparatus according to claim 20, said at least one memory stored with code thereon, which when executed by said at least one processor, further causes the apparatus to define a valid range of identifier values by:

defining a reference point identifier; and

defining the valid range of identifier values on the basis of the reference point identifier.

22. The apparatus according to claim 21, said at least one memory stored with code thereon, which when executed by said at least one processor, further causes the apparatus to:

decode a third parameter set;

obtain an identifier of the third parameter set to the third parameter set, the identifier increment relative to the reference point identifier; and

set the reference point identifier to the identifier of the third parameter set.

23. The apparatus according to claim 20, 21 or 22, said at least one memory stored with code thereon, which when executed by said at least one processor, further causes the apparatus to:

define a maximum difference of identifier values;

define a maximum identifier value; and

determine that the first parameter set is valid, if the identifier of the first parameter set is within the valid range of parameter values.

24. The apparatus according to any of the claims 19 to 23, said at least one memory stored with code thereon, which when executed by said at least one processor, further causes the apparatus to use the difference between the identifier of the second parameter set and the identifier of the first parameter set to determine whether a third parameter set encoded between the first parameter set and the second parameter set has not been received.

25. The apparatus according to any of the claims 13 to 24, said at least one memory stored with code thereon, which when executed by said at least one processor, further causes the apparatus to:

decode an identifier reference of a parameter set to be used in decoding;

examine whether the identifier reference is within the valid range of identifier values.

26. The apparatus according to claim 25, said at least one memory stored with code thereon, which when executed by said at least one processor, further causes the apparatus to:

decode the identifier reference from the second parameter set, wherein the identifier reference is to be used in decoding of the second parameter set.

27. The apparatus according to claim 25 or 26, said at least one memory stored with code thereon, which when executed by said at least one processor, further causes the apparatus to:

conclude a loss of a parameter set on the basis that the identifier reference is outside the valid range of identifier values.

28. The apparatus according to any of the claims 19 to 27, said at least one memory stored with code thereon, which when executed by said at least one processor, further causes the apparatus to:

buffer the first parameter set and the second parameter set into a buffer; and

mark the first parameter set unused if it is determined not valid.

29. An apparatus comprising at least one processor and at least one memory including computer program code, the at least one memory and the computer program code configured to, with the at least one processor, cause the apparatus to:

encode a first parameter set;

attach an identifier of the first parameter set to the first parameter set;

encode a second parameter set; and

determine the validity of the first parameter set on the basis of at least one of the following:

- by attaching in the second parameter set a list of valid identifier values; and determining that the first parameter set is valid, if the identifier of the first parameter set is in the list of valid parameter values;
- by attaching in the second parameter set an identifier of the second parameter set; and determining that the first parameter set is valid based on the identifier of the first parameter set and the identifier of the second parameter set.

30. The apparatus according to claim 29, said at least one memory stored with code thereon, which when executed by said at least one processor, further causes the apparatus to define a valid range of identifier values.

31. The apparatus according to claim 30, said at least one memory stored with code thereon, which when executed by said at least one processor, further causes the apparatus to define a valid range of identifier values by:

defining a reference point identifier; and

defining the valid range of identifier values on the basis of the reference point identifier.

32. The apparatus according to claim 31, said at least one memory stored with code thereon, which when executed by said at least one processor, further causes the apparatus to:

encode a third parameter set;

attach an identifier of the third parameter set to the third parameter set, the identifier increment relative to the reference point identifier; and

set the reference point identifier to the identifier of the third parameter set.

33. The apparatus according to claim 30, 31 or 32, said at least one memory stored with code thereon, which when executed by said at least one processor, further causes the apparatus to encode an identifier reference of a parameter set to be used in decoding, and select the identifier reference from the valid range of identifier values.

34. The apparatus according to any of the claims 29 to 33. said at least one memory stored with code thereon, which when executed by said at least one processor, further causes the apparatus to:

define a maximum difference of identifier values; and

define a maximum identifier value.

35. The apparatus according to any of the claims 29 to 34, said at least one memory stored with code thereon, which when executed by said at least one processor, further causes the apparatus to set the identifier of the second parameter set different from the identifier from the first parameter set, if the first parameter set has been determined valid.

36. The apparatus according to any of the claims 29 to 35, said at least one memory stored with code thereon, which when executed by said at least one processor, further causes the apparatus to allow the second parameter set refer to the first parameter set, if the first parameter set has been determined valid.

37. A computer program product including one or more sequences of one or more instructions which, when executed by one or more processors, cause an apparatus to at least perform the following:

receive a first parameter set;

obtain an identifier of the first parameter set;

receive a second parameter set; determine the validity of the first parameter set on the basis of at least one of the following:

- receiving in the second parameter set a list of valid identifier values; and determining that the first parameter set is valid, if the identifier of the first parameter set is in the list of valid parameter values;
- receiving in the second parameter set an identifier of the second parameter set; and determining that the first parameter set is valid based on the identifier of the first parameter set and the identifier of the second parameter set.

38. The computer program product according to claim 37 including one or more sequences of one or more instructions which, when executed by one or more processors, cause the apparatus to at least define a valid range of identifier values.

39. The computer program product according to claim 37 or 38 including one or more sequences of one or more instructions which, when executed by one or more processors, cause the apparatus to at least:

define a maximum difference of identifier values;

define a maximum identifier value; and

determine that the first parameter set is valid, if one of the following conditions is true:

- the identifier of the second parameter set is greater than the identifier of the first parameter set and the difference between the identifier of the second parameter set and the identifier of the first parameter set is smaller than or equal to the maximum difference of identifier values;
- the identifier of the first parameter set is greater than the identifier of the second parameter set and the identifier of the second parameter set is smaller than or equal to the maximum difference of identifier values and the difference between the identifier of the first parameter set and the identifier of the second parameter set is greater than the difference between the maximum identifier value and the maximum difference of identifier values.

40. The computer program product according to claim 37, 38 or 39 including one or more sequences of one or more instructions which, when executed by one or more processors, cause the apparatus to at least use the difference between the identifier of the second parameter set and the identifier of the first parameter set to determine whether a third parameter set encoded between the first parameter set and the second parameter set has not been received.

41. The computer program product according to any of the claims 37 to 40 including one or more sequences of one or more instructions which, when executed by one or more processors, cause the apparatus to at least:

decode the second parameter set;

examine whether the second parameter set comprises a reference to the first parameter set which has not been determined valid.

42. The computer program product according to any of the claims 37 to 41 including one or more sequences of one or more instructions which, when executed by one or more processors, cause the apparatus to at least:

buffer the first parameter set and the second parameter set into a buffer; and

mark the first parameter set unused if it is determined not valid.

43. A computer program product including one or more sequences of one or more instructions which, when executed by one or more processors, cause an apparatus to at least perform the following:

encode a first parameter set;

attach an identifier of the first parameter set;

encode a second parameter set; determine the validity of the first parameter set on the basis of at least one of the following:

- by attaching in the second parameter set a list of valid identifier values; and determining that the first parameter set is valid, if the identifier of the first parameter set is in the list of valid parameter values;
- by attaching in the second parameter set an identifier of the second parameter set; and determining that the first parameter set is valid based on the identifier of the first parameter set and the identifier of the second parameter set.

44. The computer program product according to claim 43 including one or more sequences of one or more instructions which, when executed by one or more processors, cause the apparatus to at least define a valid range of identifier values.

45. The computer program product according to claim 43 or 44 including one or more sequences of one or more instructions which, when executed by one or more processors, cause the apparatus to at least select the identifier from the valid range of identifier values.

46. The computer program product according to claim 43, 44 or 45 including one or more sequences of one or more instructions which, when executed by one or more processors, cause the apparatus to at least:

define a maximum difference of identifier values; and
define a maximum identifier value.

47. The computer program product according to any of the claims 43 to 46 including one or more sequences of one or more instructions which, when executed by one or more processors, cause the apparatus to at least set the identifier of the second parameter set different from the identifier from the first parameter set, if the first parameter set has been determined valid.

48. The computer program product according to any of the claims 43 to 47 including one or more sequences of one or more instructions which, when executed by one or more processors, cause the apparatus to at least allow the second parameter set refer to the first parameter set, if the first parameter set has been determined valid.

49. An apparatus comprising:
means for receiving a first parameter set;
means for obtaining an identifier of the first parameter set;
means for receiving a second parameter set; means for determining the validity of the first parameter set on the basis of at least one of the following:

- by receiving in the second parameter set a list of valid identifier values; and determining that the first parameter set is valid, if the identifier of the first parameter set is in the list of valid parameter values;
- by receiving in the second parameter set an identifier of the second parameter set; and determining that the first parameter set is valid based on the identifier of the first parameter set and the identifier of the second parameter set.

50. An apparatus comprising:
means for encoding a first parameter set;
means for attaching an identifier of the first parameter set;
means for encoding a second parameter set; and
means for determining the validity of the first parameter set on the basis of at least one of the following:

- by attaching in the second parameter set a list of valid identifier values; and determining that the first parameter set is valid, if the identifier of the first parameter set is in the list of valid parameter values;
- by attaching in the second parameter set an identifier of the second parameter set; and determining that the first parameter set is valid based on the identifier of the first parameter set and the identifier of the second parameter set.

51. A video decoder configured for:
receiving a first parameter set;
obtaining an identifier of the first parameter set;
receiving a second parameter set; determining the validity of the first parameter set on the basis of at least one of the following:

- receiving in the second parameter set a list of valid identifier values; and determining that the first parameter set is valid, if the identifier of the first parameter set is in the list of valid parameter values;
- receiving in the second parameter set an identifier of the second parameter set; and determining that the first parameter set is valid based on the identifier of the first parameter set and the identifier of the second parameter set.

52. A video encoder configured for:
encoding a first parameter set;
attaching an identifier of the first parameter set to the first parameter set;
encoding a second parameter set;
determining the validity of the first parameter set on the basis of at least one of the following:

- attaching in the second parameter set a list of valid identifier values; and determining that the first parameter set is valid, if the identifier of the first parameter set is in the list of valid parameter values;
- attaching in the second parameter set an identifier of the second parameter set; and determining that the first parameter set is valid based on the identifier of the first parameter set and the identifier of the second parameter set.

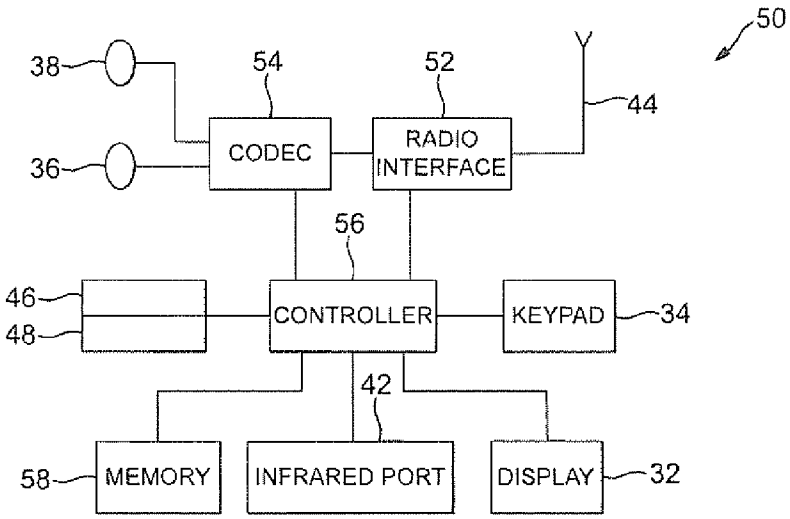
50
38
54
52
44
36
CODEC
RADIO INTERFACE
56
46
48
CONTROLLER
KEYPAD
34
42
58
MEMORY
INFRARED PORT
DISPLAY
32

FIG.1

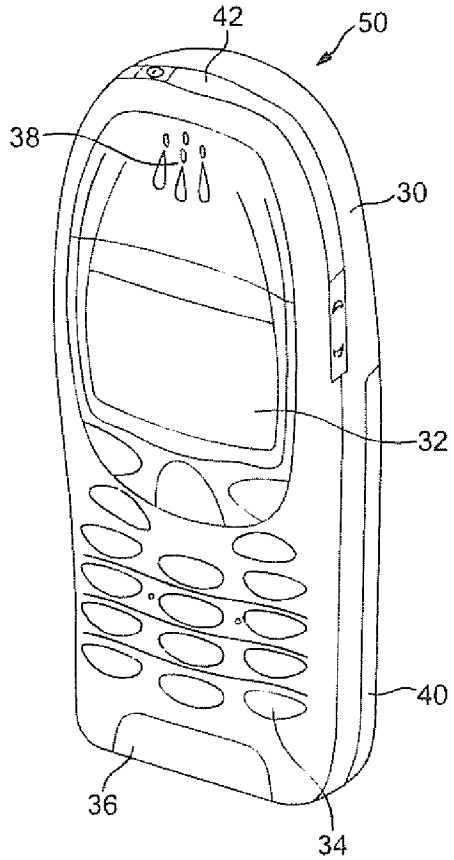
42
50
38
30
32
40
36
34

FIG. 2

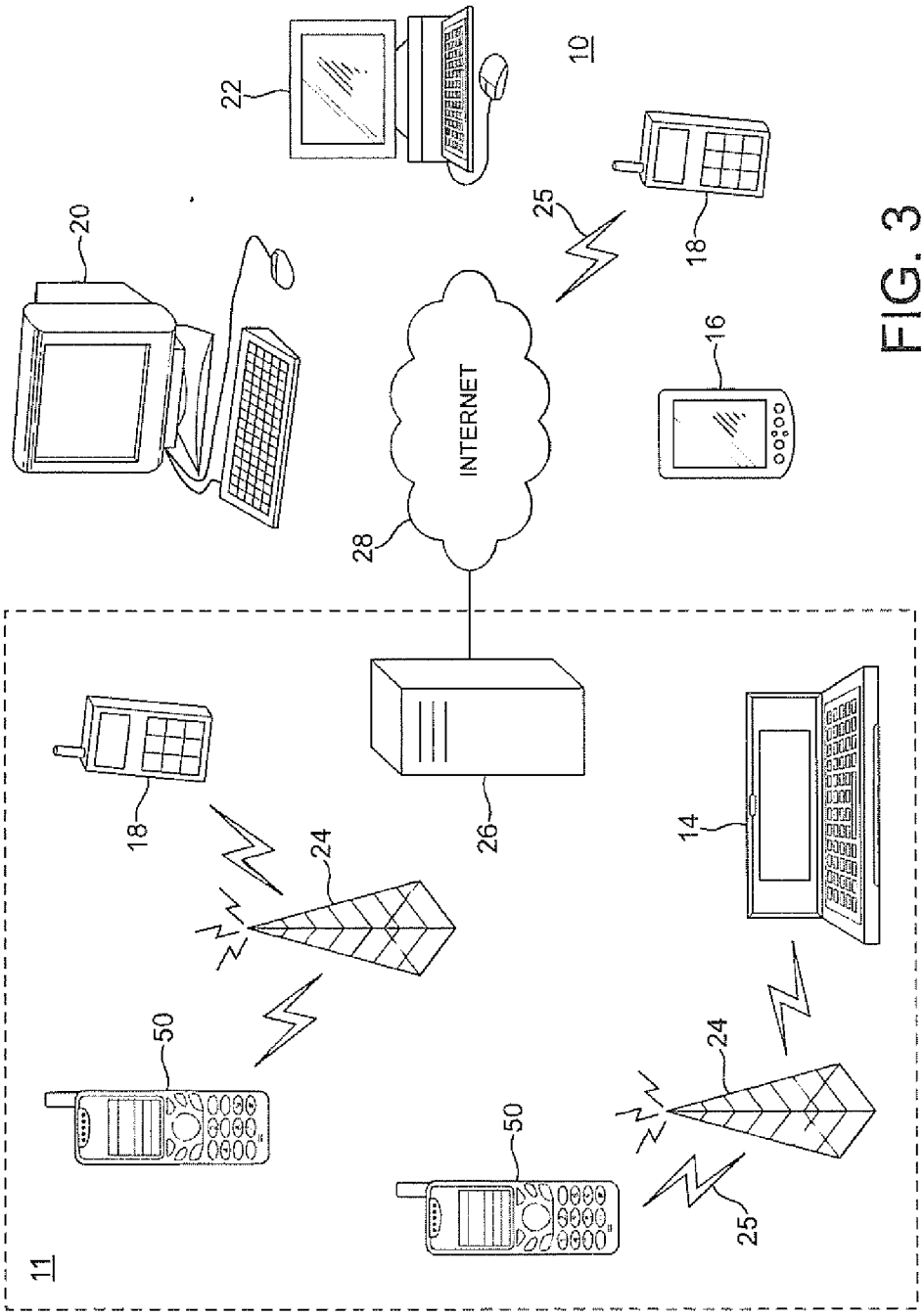

FIG. 3

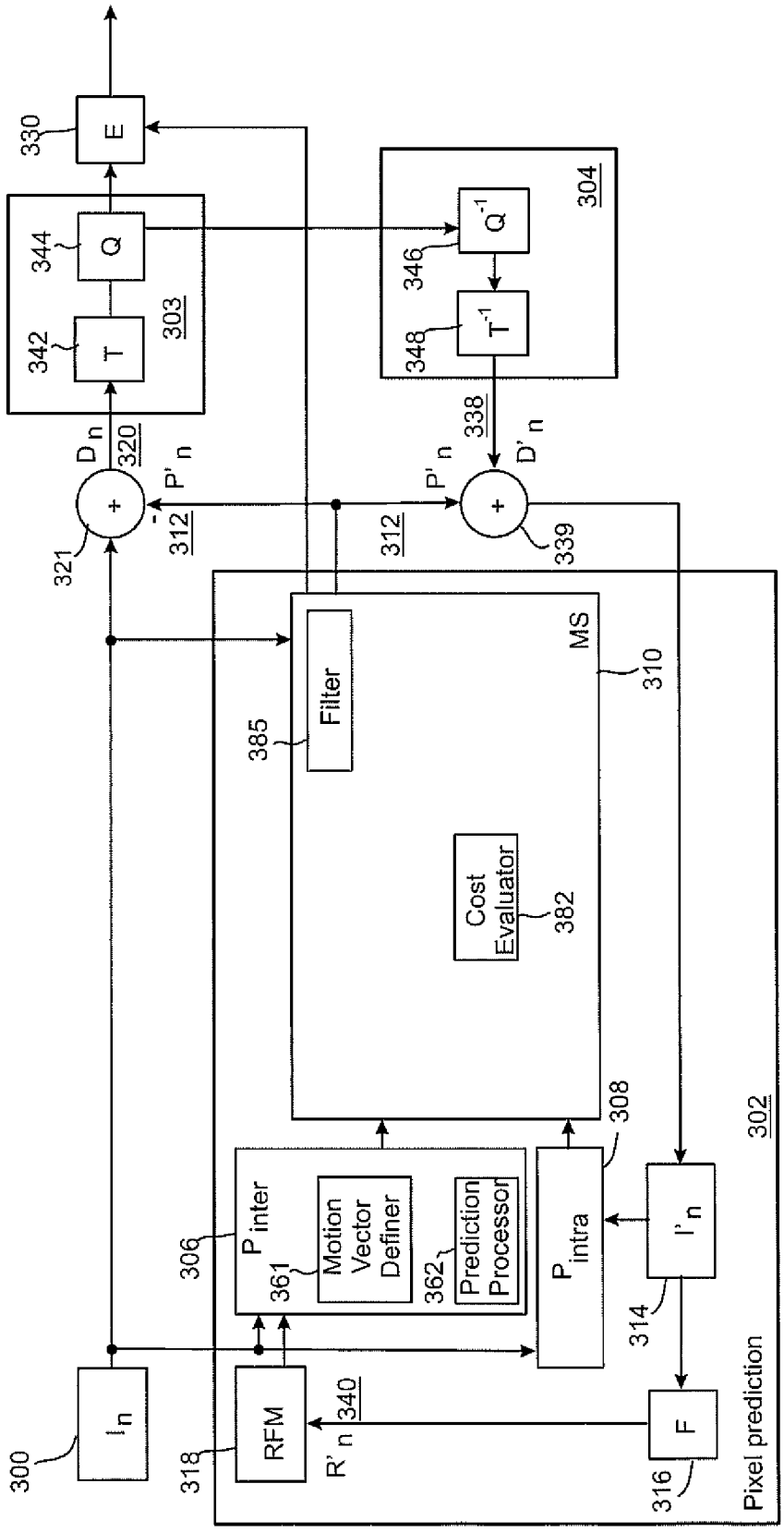
330
E
344
Q
342
T
303
304
346
Q-1
348
T-1
338
Dn
320
P'n
D'n
321
312
339
MS
310
385
Filter
Cost Evaluator
382
308
306
P inter
361
Motion Vector Definer
362
Prediction Processor
P intra
I'n
314
302
300
In
318
RFM
R'n 340
316
F
Pixel prediction

Fig. 4a

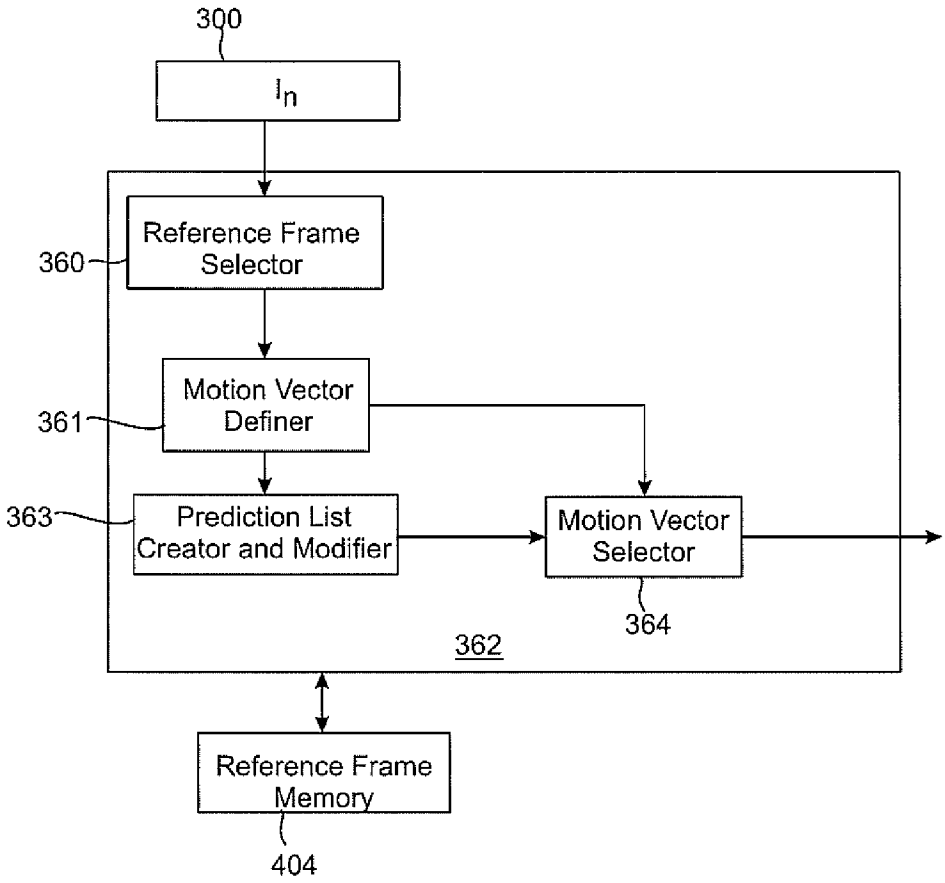
300
I_n
Reference Frame Selector
360
Motion Vector Definer
361
Prediction List Creator and Modifier
363
Motion Vector Selector
364
362
Reference Frame Memory
404

Fig. 4b

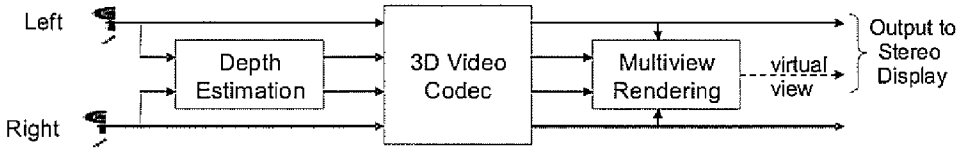
Left
Right
Depth Estimation
3D Video Codec
Multiview Rendering
virtual view
Output to Stereo Display

Fig. 5

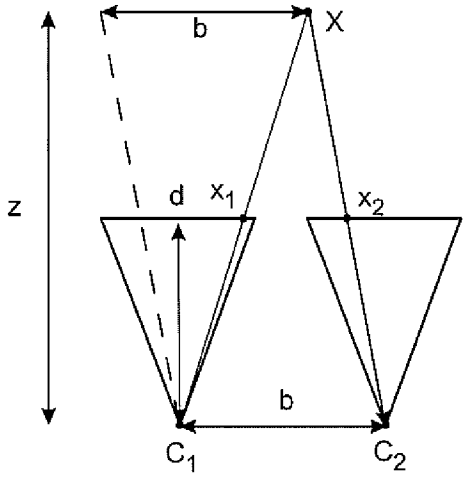
b
X
z
d
x1
x2
b
C1
C2

Fig. 6

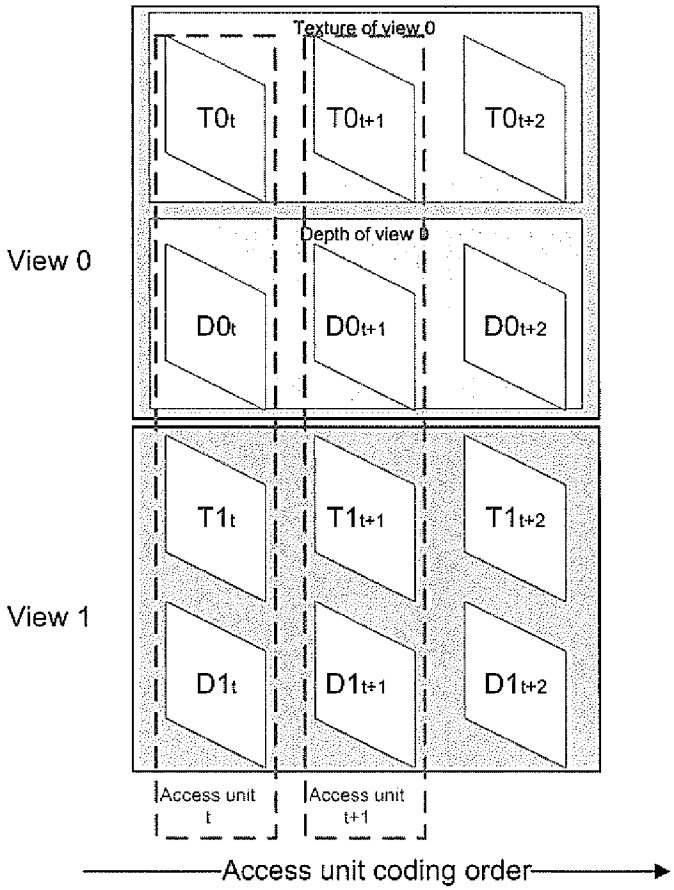
Texture of view 0
T0t
T0t+1
T0t+2
Depth of view 0
D0t
D0t+1
D0t+2
View 0
T1t
T1t+1
T1t+2
View 1
D1t
D1t+1
D1t+2
Access unit t
Access unit t+1
Access unit coding order

Fig. 7

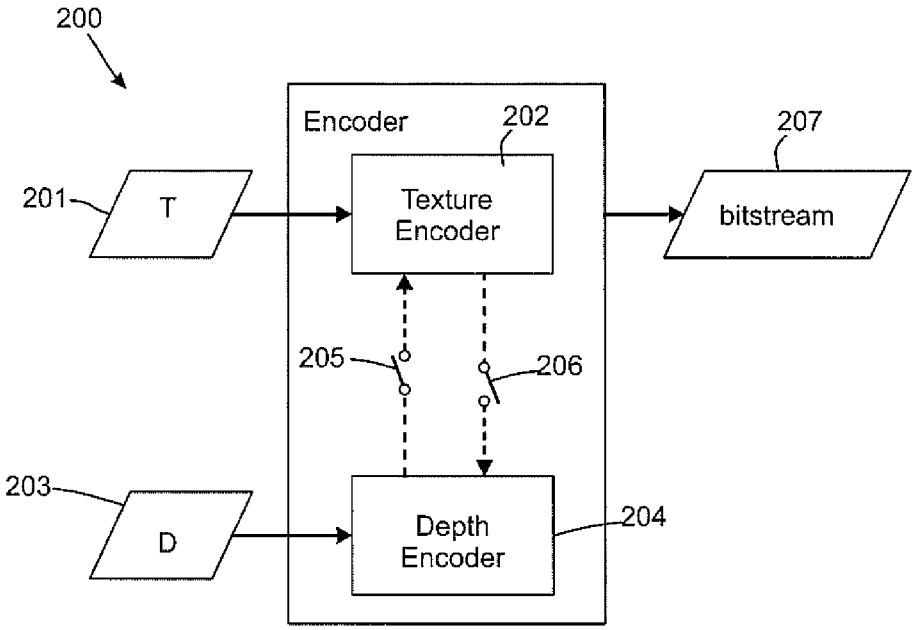
200
Encoder
202
207
201
T
Texture Encoder
bitstream
205
206
203
D
Depth Encoder
204

Fig. 8

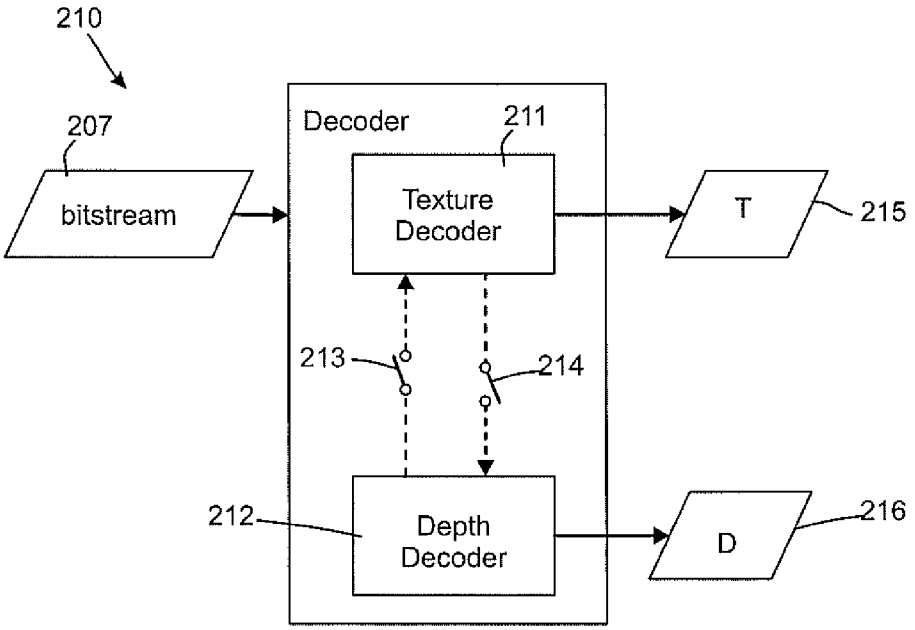
210
Decoder
211
207
bitstream
Texture Decoder
T
215
213
214
212
Depth Decoder
D
216

Fig. 9

INTERNATIONAL SEARCH REPORT

International application No.
PCT/FI2013/050419

A. CLASSIFICATION OF SUBJECT MATTER

See extra sheet

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: H04N

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

FI, SE, NO, DK

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

EPO-Internal, WPI, IEEE Xplore, Google Scholar

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2008310500 A1 (WINGER, L. L.) 18 December 2008 (18.12.2008) paragraph [0007]; claims 1, 4, 11	1-52
A	US 2005254526 A1 (WANG, Y.-K. et al.) 17 November 2005 (17.11.2005) the whole document	1-52
A	US 2007219808 A1 (HERRE, J. et al.) 20 September 2007 (20.09.2007) the whole document	1-52
A	US 2007201564 A1 (JOCH, A. et al.) 30 August 2007 (30.08.2007) the whole document	1-52

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

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"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
"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
"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
"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
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
04 September 2013 (04.09.2013)	06 September 2013 (06.09.2013)

Name and mailing address of the ISA/FI
National Board of Patents and Registration of Finland
P.O. Box 1160, FI-00101 HELSINKI, Finland
Facsimile No. +358 9 6939 5328

Authorized officer
Ari Viholainen
Telephone No. +358 9 6939 500

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/FI2013/050419

Patent document cited in search report	Publication date	Patent family members(s)	Publication date
US 2008310500 A1	18/12/2008	US 2005123055 A1	09/06/2005
		US 7415069 B2	19/08/2008
		US 8233547 B2	31/07/2012
US 2005254526 A1	17/11/2005	None	
US 2007219808 A1	20/09/2007	AU 2005281937 A1	16/03/2006
		AU 2005281937 B2	09/10/2008
		BR PI0515623 A	29/07/2008
		CA 2578190 A1	16/03/2006
		CA 2578190 C	11/09/2012
		CN 101044550 A	26/09/2007
		CN 101044550 B	11/05/2011
		DE 102004042819 A1	23/03/2006
		EP 1763870 A1	21/03/2007
		HK 1107174 A1	05/08/2011
		IL 181469 D0	04/07/2007
		IL 181469 A	27/09/2011
		JP 2008511848 A	17/04/2008
		JP 4856641 B2	18/01/2012
		KR 20070051875 A	18/05/2007
		MX 2007002569 A	05/07/2007
		NO 20070903 A	03/04/2007
		RU 2007112113 A	20/10/2008
		RU 2379768 C2	20/01/2010
		US 8145498 B2	27/03/2012
		WO 2006027138 A1	16/03/2006
US 2007201564 A1	30/08/2007	US 2004101059 A1	27/05/2004
		US 7050504 B2	23/05/2006
		US 2006104349 A1	18/05/2006
		US 7227901 B2	05/06/2007
		US 2006250653 A1	09/11/2006
		US 7869523 B2	11/01/2011
		US 2006227869 A1	12/10/2006
		US 7894534 B2	22/02/2011
		US 8005151 B2	23/08/2011

INTERNATIONAL SEARCH REPORT

International application No.
PCT/FI2013/050419

CLASSIFICATION OF SUBJECT MATTER

Int.Cl.
H04N 7/26 (2006.01)
H04N 21/83 (2011.01)