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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
31 October 2013 (31.10.2013)

WIPO | PCT

(10) International Publication Number
WO 2013/159330 A1

(51) International Patent Classification:
H04N 7/32 (2006.01)

(21) International Application Number:
PCT/CN2012/074812

(22) International Filing Date:
27 April 2012 (27.04.2012)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant *(for all designated States except US)*: **NOKIA CORPORATION** [FI/FI]; Keilalahdentie 4, FI-02150 Espoo (FI).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **RUSANOVSKYY, Dmytro** [UA/FI]; Kuhapolku 2 F22, FI-37550 Lempäälä (FI). **HANNUKSELA, Miska** [FI/FI]; Pitkätie 14, FI-36110 Ruutana (FI). **SU, Wenyi** [CN/CN]; 443 Huangshan Road, P.O. Box 4, Shushan District, Hefei, Anhui 230027 (CN).

(74) Agent: **KING & WOOD MALLESONS**; 20th Floor, East Tower, World Financial Centre, No. 1 Dongsanhuan Zhonglu, Chaoyang District, Beijing 100020 (CN).

(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, 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, 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).

Declarations under Rule 4.17:

— *of inventorship (Rule 4.17(iv))*

Published:

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

(54) Title: AN APPARATUS, A METHOD AND A COMPUTER PROGRAM FOR VIDEO CODING AND DECODING

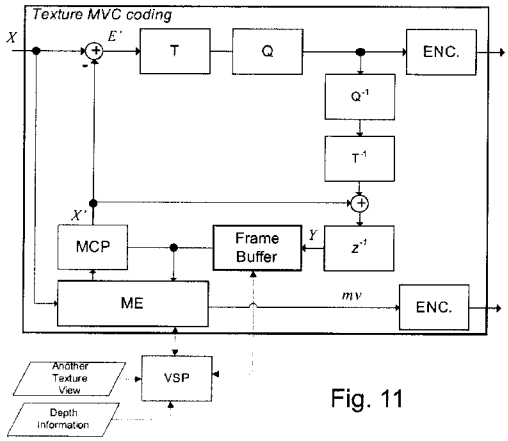

Fig. 11

(57) Abstract: A method and related apparatuses for encoding depth/disparity enhanced multiview video content comprising: obtaining a first uncompressed texture block (Cb1) of a first texture picture representing a first view; obtaining a first depth/disparity block (d(Cb1)) associated with the first texture block; encoding the first uncompressed texture block (Cb1) with a block-based view synthesis prediction (VSP) process utilizing the first depth/disparity block (d(Cb1)) and VSP source data {VSP_T2(Cb1), VSP_D2(Cb2)} available from a coded second view.

WO 2013/159330 A1

AN APPARATUS, A METHOD AND A COMPUTER PROGRAM FOR VIDEO CODING AND DECODING

Technical Field

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

Background Information

Various technologies for providing three-dimensional (3D) video content are currently investigated and developed. Especially, intense studies have been focused on various multiview applications wherein a viewer is able to see only one pair of stereo video from a specific viewpoint and another pair of stereo video from a different viewpoint. One of the most feasible approaches for such multiview applications has turned out to be such wherein only a limited number of input views, e.g. a mono or a stereo video plus some supplementary data, is provided to a decoder side and all required views are then rendered (i.e. synthesized) locally by the decoder to be displayed on a display.

Several technologies for view rendering are available, and for example, depth image-based rendering (DIBR) has shown to be a competitive alternative. A typical implementation of DIBR takes stereoscopic video and corresponding depth information with stereoscopic baseline as input and synthesizes a number of virtual views between the two input views. Thus, 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.

In the encoding of 3D video content, video compression systems, such as Advanced Video Coding standard H.26

4/AVC or the Multiview Video Coding MVC extension of H.264/AVC can be used. To develop an AVC/MVC compatible 3DV standard, MPEG 3DV has designed a 3DV-ATM reference test model. This test model utilizes in-loop View synthesis based Prediction (VSP). Despite of some coding gain achieved therein, the VSP is still remains a complex part of 3DV-ATM decoder.

A reason for that is that the current design of in-loop VSP of 3DV-ATM decoder utilizes a forward projection, which requires a frame-level implementation, i.e. synthesizing a complete frame to be used as reference picture. Since majority of AVC decoders implement motion compensation prediction (MCP) with reference frame at the block level, the current frame-based VSP implementation may be significantly more burdensome than MCP in terms of computational complexity, storage requirement, and memory access bandwidth.

Summary

Now there has been invented an improved method and technical equipment implementing the method. Various aspects of the invention include methods, apparatuses, computer programs, an encoder and decoder, which are characterized by what is stated in the independent claims. Various embodiments of the invention are disclosed in the dependent claims.

According to a first aspect, there is provided a method for encoding depth/disparity enhanced multiview video content comprising: obtaining a first uncompressed texture block (Cb1) of a first texture picture representing a first view; obtaining a first depth/disparity block (d(Cb1)) associated with the first texture block; encoding the first uncompressed texture block (Cb1) with a block-based view synthesis prediction (VSP) process utilizing the first depth/disparity block (d(Cb1)) and VSP source data {VSP_T2(Cb1), VSP_D2(Cb2)} available from a coded second view.

According to an embodiment, the method further comprises encoding each uncompressed texture block (Cb1i) of the first texture picture with the block-based view synthesis prediction (VSP) process utilizing the associated depth/disparity block (d(Cb1i)) and VSP source data {VSP_T2(Cb1), VSP_D2(Cb2)} available from the coded second view.

According to an embodiment, the view prediction synthesis process (VSP) on a block-level produces pixel values in a referenced area (R(Cb1i)) of a VSP frame, which is associated with the first texture picture.

According to an embodiment, the method further comprises obtaining a second uncompressed texture block (Cb2) of a first texture picture representing the first view and a second depth/disparity block (d(Cb2)) associated with the second texture block; encoding the second uncompressed texture block (Cb2) with the block-based view synthesis prediction (VSP) process utilizing the second depth/disparity block (d(Cb2)) and VSP source data {VSP_T2(Cb1), VSP_D2(Cb2)} available from the coded second view such that pixel values in a referenced area (R(Cb2)) of a second VSP frame overlap with the pixel values in the referenced area (R(Cb1)) of the first VSP frame.

According to an embodiment, the method further comprises obtaining a second uncompressed texture block (Cb2) of a first texture picture representing the first view and a second depth/disparity block (d(Cb2)) associated with the second texture block; encoding the second uncompressed texture block (Cb2) with the block-based view synthesis prediction (VSP) process utilizing the second depth/disparity block (d(Cb2)) and VSP source data {VSP_T2(Cb1), VSP_D2(Cb2)} available from the coded second view such that pixel values in a referenced area (R(Cb2)) of a second VSP frame are independent of the pixel values in the referenced area (R(Cb1)) of the first VSP frame.

According to an embodiment, the view prediction synthesis (VSP) process further comprises: converting ranging information d(Cb)1 associated with the first texture block to a disparity values D(Cb)1;

analyzing the disparity values D(Cb)1 to find minimal (min_D) and maximal (max_D) disparity values; specifying view synthesis prediction (VSP) source regions in a second view with size of NxM in texture image (VSP_T) and in associated depth image (VSP_D) on the basis of the disparity values min_D and max_D, where M is a size of the VSP source region in vertical direction and N=(max_D-min_D+1) is the size of VSP source region in horizontal direction; extending the VSP source regions in horizontal and/or vertical direction by K1 and K2 pixels respectively; and generating, on the basis of VSP source regions (VSP_T and VSP_D) of the second coded view, a referenced area R(Cb) in the VSP frame associated with the first coded view.

According to an embodiment, the method further comprises converting ranging information d(Cb) to a disparity values D(Cb) such that it reflects a difference in spatial coordinates between a particular sample of Cb taken from a first coded view and the corresponding sample in a second coded view.

According to an embodiment, the method further comprises estimating a value K1 such that arithmetic modulo operations with dividend (N+K1) and divisor n produces zero; and estimating a value K2 such that arithmetic modulo operations with dividend (M+K2) and divisor m produces zero.

According to an embodiment, the method further comprises splitting the VSP source region (VSP_T and VSP_D) in integer number of non-overlapping blocks (PBU) of a fixed size (n x m) and processing each PBU independently.

According to an embodiment, the method further comprises obtaining weighing parameters for pixel values in the referenced area (R(Cbi)) of the

VSP frame, which is synthesized or projected from a particular view; and weighing the pixel values with corresponding weighing parameters.

According to an embodiment, the method further comprises normalizing pixel values in the referenced area R(Cbi) of the VSP frame, which is synthesized or projected from a particular view with a corresponding weighing parameter.

According to an embodiment, the method further comprises calculating a weighted average of pixel values in the referenced area R(Cbi) of the VSP frame projected from a first view and a second view.

According to an embodiment, the method of weighting pixels of VS process are weighting parameters derivation are applied in coding systems with VSP design different from described in this intention and may not require ranging information associated with current view be available prior to coding of texture information of the current view.

According to an embodiment, the method further comprises encoding the uncompressed texture block Cb with the block-based view synthesis prediction (VSP) process that is performed from a particular reference view among several available views; deriving an identification of the view to be used for said VSP process prior to encoding of the uncompressed texture block Cb; and providing said identification to a decoder in a bitstream.

According to an embodiment, the method further comprises encoding the first uncompressed texture block Cb with the block-based view synthesis prediction (VSP) process that is performed from a first available reference view; calculating a first rate distortion metric cost for the encoded block Cb; encoding the first uncompressed texture block Cb with the block-based view synthesis prediction (VSP) process that is performed from a second available reference view; calculating a second rate distortion metric cost for the encoded block Cb; selecting an optimal reference view of view synthesis

process by finding minimal rate distortion cost metric from said first and second rate distortion metric cost; encoding the first uncompressed texture block Cb with block-based view synthesis prediction (VSP) process that performed from the optimal reference view; and signaling the optimal selected reference view for VSP to a decoder in a bitstream.

According to an embodiment, the method of uni-, bi- and multi- directional VS process with signaling/deriving index of view that provides source data for VSP are applied in coding systems with VSP design different from described in this intention and may not require ranging information associated with current view be available prior to coding of texture information of the current view.

This and the other aspects of the invention and the embodiments related thereto will become apparent in view of the detailed disclosure of the embodiments further below.

Description of the Drawings

For better understanding of the present invention, reference will now be made by way of example to the accompanying drawings in which:

Figure 1 shows a simplified 2D model of a stereoscopic camera setup.;

Figure 2 shows a simplified model of a multiview camera setup;

Figure 3 shows a simplified model of a multiview autostereoscopic display (ASD);

Figure 4 shows a simplified model of a DIBR-based 3DV system;

Figures 5a and 5b show the spatial and temporal neighborhood of the currently coded block serving as the candidates for MVP in H.264/AVC;

Figure 6 shows an example structure of video plus depth (MVD) data;

Figure 7 shows an example of a conventional VSP-enabled multiview video encoder;

Figure 8 shows a visualization of horizontal-vertical and disparity correspondence between texture and depth images in a first and a second coded view according to an embodiment of the invention;

Figure 9 shows a layout of a PBU within a VSP source blocks according to an embodiment of the invention;

Figure 10 shows view synthesis process (projection) which is performed over VSP source blocks {VSP_T and VSP_D} through a loop of fixed-sizes PBUs and results in producing pixel values of reference area R(Cb) according to an embodiment of the invention;

Figure 11 shows an encoder according to an embodiment as a simplified block diagram;

Figure 12 shows schematically an electronic device suitable for employing some embodiments of the invention;

Figure 13 shows schematically a user equipment suitable for employing some embodiments of the invention;

Figure 14 further shows schematically electronic devices employing embodiments of the invention connected using wireless and wired network connections.

Detailed Description of Some Example Embodiments of the Invention

In order to understand the various aspects of the invention and the embodiments related thereto, the following describes briefly some closely related aspects of video coding.

Some key definitions, bitstream and coding structures, and concepts of H.264/AVC 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. The aspects of the invention are not limited to H.264/AVC, but rather the description is given for one possible basis on top of which the invention may be partly or fully realized.

The H.264/AVC standard was developed by the Joint Video Team (JVT) of the Video Coding Experts Group (VCEG) of the Telecommunications Standardisation Sector of International Telecommunication Union (ITU-T) and the Moving Picture Experts Group (MPEG) of International Standardisation Organisation (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).

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. 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), which is specified in Annex C of H.264/AVC. The standard contains 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 encoder and the output of an H.264/AVC decoder is a picture. 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. A macroblock is a 16x16 block of luma samples and the corresponding blocks of chroma samples. 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

and consequently a macroblock contains one 8x8 block of chroma samples per each chroma component. A picture is partitioned to one or more slice groups, and a slice group contains one or more slices. A slice consists of an integer number of macroblocks ordered consecutively in the raster scan within a particular slice group.

The elementary unit for the output of an H.264/AVC encoder and the input of an H.264/AVC decoder is a Network Abstraction Layer (NAL) unit. Decoding of partially lost or corrupted NAL units is typically difficult. For transport over packet-oriented networks or storage into structured files, NAL units are typically encapsulated into packets or similar structures. A bytestream format has been specified in H.264/AVC 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 is performed always regardless of whether the bytestream format is in use or not.

H.264/AVC, as many other video coding standards, allows splitting of a coded picture into slices. In-picture prediction is 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 elementary units for transmission.

Some profiles of H.264/AVC enable the use of up to eight slice groups per coded picture. When more than one slice group is in use, the picture is partitioned into slice group map units, which are equal to two vertically consecutive macroblocks when the macroblock-adaptive frame-field (MBAFF)

coding is in use and equal to a macroblock otherwise. The picture parameter set contains data based on which each slice group map unit of a picture is associated with a particular slice group. A slice group can contain any slice group map units, including non-adjacent map units. When more than one slice group is specified for a picture, the flexible macroblock ordering (FMO) feature of the standard is used.

In H.264/AVC, a slice consists of one or more consecutive macroblocks (or macroblock pairs, when MBAFF is in use) within a particular slice group in raster scan order. If only one slice group is in use, H.264/AVC slices contain consecutive macroblocks in raster scan order and are therefore similar to the slices in many previous coding standards. In some profiles of H.264/AVC slices of a coded picture may appear in any order relative to each other in the bitstream, which is referred to as the arbitrary slice ordering (ASO) feature. Otherwise, slices must be in raster scan order in the bitstream.

NAL units consist of a header and payload. 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. The header for SVC and MVC NAL units additionally contains various indications related to the scalability and multiview hierarchy.

NAL units can be categorized into Video Coding Layer (VCL) NAL units and non-VCL NAL units. VCL NAL units are either coded slice NAL units, coded slice data partition NAL units, or VCL prefix NAL units. 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. There are four types of coded slice NAL units: coded slice in an Instantaneous Decoding Refresh (IDR) picture, coded slice in a non-IDR picture, coded slice of an auxiliary coded picture (such as an alpha plane) and coded slice extension (for SVC slices not in the base layer or MVC slices not in the base view). A set of three coded slice data partition NAL units

contains the same syntax elements as a coded slice. Coded slice data partition A comprises macroblock headers and motion vectors of a slice, while coded slice data partition B and C include the coded residual data for intra macroblocks and inter macroblocks, respectively. It is noted that the support for slice data partitions is only included in some profiles of H.264/AVC. A VCL prefix NAL unit precedes a coded slice of the base layer in SVC and MVC bitstreams and contains indications of the scalability hierarchy of the associated coded slice.

A non-VCL NAL unit may be of 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 are essential for the reconstruction of decoded pictures, whereas the other non-VCL NAL units are not necessary for the reconstruction of decoded sample values and serve other purposes presented below.

Parameters that remain unchanged through a coded video sequence are included in a sequence parameter set. In addition to the parameters that are essential to the decoding process, the sequence parameter set may optionally contain video usability information (VUI), which includes parameters that are important for buffering, picture output timing, rendering, and resource reservation. A picture parameter set contains such parameters that are likely to be unchanged in several coded pictures. No picture header is present in H.264/AVC bitstreams but the frequently changing picture-level data is repeated in each slice header and picture parameter sets carry the remaining picture-level parameters. H.264/AVC syntax allows many instances of sequence and picture parameter sets, and each instance is identified with a unique identifier. 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. 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 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 H.264/AVC Real-time Transport Protocol (RTP) sessions. If parameter sets are transmitted in-band, they can be repeated to improve error robustness.

An SEI NAL unit contains one or more SEI messages, which are not required for the decoding of output pictures but 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 the user data SEI messages enable organizations and companies to specify SEI messages for their own use. H.264/AVC contains 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 when they create SEI messages, and decoders conforming to the H.264/AVC standard 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 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 in H.264/AVC consists of the VCL NAL units that are required for the decoding of the picture. 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 H.264/AVC, an access unit consists of a primary coded picture and those NAL units that are associated with it. 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 or slice data partitions of the primary coded picture appear next, followed by coded slices for zero or more redundant coded pictures.

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 always results in one decoded picture consisting of one or more decoded view components. 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. 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.

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) is 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. 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, 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 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. 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, 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 be also referred to as temporal prediction, motion-compensated prediction (MCP) 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, 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, as many other video compression standards, divides 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 motion vector that indicates the position of the prediction block compared 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. In H.264/AVC, motion vectors are of quarter-pixel accuracy, and sample values in fractional-pixel positions are obtained using a finite impulse response (FIR) filter.

Block partitioning for inter prediction. A basic unit for inter prediction in many coding standards is a macroblock, corresponding to a 16x16 block of luma samples and corresponding chroma samples. In H.264/AVC, a macroblock can be further divided to 16x8, 8x16, or 8x8 macroblock partitions, and the 8x8 partition can be further divided to 4x4, 4x8, or 8x4 sub-macroblock partitions, and a motion vector is coded for each partition. In the following, a block is used to refer to a unit for inter prediction, which may be of a different level in the partitioning structure. For example in the case of H.264/AVC, a block in the following may refer to a macroblock, a macroblock partition or a sub-macroblock partition, whichever is used as a unit for inter prediction.

Number of reference pictures for inter prediction. The sources of inter prediction are previously decoded pictures. H.264/AVC enables storage of multiple reference pictures for inter prediction and selection of the used reference picture on macroblock or macroblock partition basis.

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. Differential coding of motion vectors is typically disabled across slice boundaries.

Multi-hypothesis motion-compensated prediction. H.264/AVC enables the use of a single prediction block in P and SP 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 or SP slices may be uni-predicted or intra-predicted. In H.264/AVC the reference pictures for a bi-predictive picture are not limited to be the subsequent picture and the previous picture in output order, but rather any reference pictures can be used. Uni-prediction may also be referred to as uni-directional prediction, bi-prediction as bi-directional prediction, and multi-hypothesis prediction as multi-directional prediction.

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.

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 requires the presence of memory management control operation (MMCO) parameters in the bitstream. 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.

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 provides 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 reference and needed for output.

In H.264/AVC, the reference picture for inter prediction is indicated with an index to a reference picture list. The index is coded with variable length coding, i.e., the smaller the index is, the shorter the corresponding syntax

element becomes. Two reference picture lists (reference picture list 0 and reference picture list 1) are generated for each bi-predictive (B) slice of H.264/AVC, and one reference picture list (reference picture list 0) is formed for each inter-coded (P or SP) slice of H.264/AVC. A reference picture list is constructed in two steps: first, an initial reference picture list is generated, and then the initial reference picture list may be reordered by reference picture list reordering (RPLR) commands contained in slice headers. The RPLR commands indicate the pictures that are ordered to the beginning of the respective reference picture list.

The frame_num syntax element 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).

A value of picture order count (POC) is derived for each picture and is non-decreasing with increasing picture position in output order relative to the previous IDR picture or a picture containing a memory management control operation marking all pictures as “unused for reference”. POC therefore indicates the output order of pictures. It is also used in the decoding process 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 of B slices. Furthermore, POC is used in the verification of output order conformance.

In MVC, view dependencies are specified in the sequence parameter set (SPS) MVC extension. The dependencies for anchor pictures and non-anchor pictures are independently specified. Therefore anchor pictures and non-anchor pictures can have different view dependencies. However, for the set of pictures that refer to the same SPS, all the anchor pictures have the

same view dependency, and all the non-anchor pictures have the same view dependency. Furthermore, in the SPS MVC extension, dependent views are signaled separately for the views used as reference pictures in reference picture list 0 and for the views used as reference pictures in reference picture list 1.

In MVC, there is an "inter_view_flag" in the network abstraction layer (NAL) unit header which indicates whether the current picture is not used or is allowed to be used for inter-view prediction for the pictures in other views. Non-reference pictures (with "nal_ref_idc" equal to 0) that are used as for inter-view prediction reference (i.e. having "inter_view_flag" equal to 1) are called inter-view only reference pictures. Pictures with "nal_ref_idc" greater than 0 and that are used for inter-view prediction reference (i.e. having "inter_view_flag equal to 1") are called inter-view reference pictures.

In MVC, inter-view prediction is supported by texture prediction (i.e., the reconstructed sample values may be used for inter-view prediction), and only the decoded view components of the same output time instance (i.e., the same access unit) as the current view component are used for inter-view prediction. The fact that reconstructed sample values are used in inter-view prediction also implies that MVC utilizes multi-loop decoding. In other words, motion compensation and decoded view component reconstruction are performed for each view.

The process of constructing reference picture lists in MVC is summarized as follows. First, an initial reference picture list is generated in two steps: i) An initial reference picture list is constructed including all the short-term and long-term reference pictures that are marked as "used for reference" and belong to the same view as the current slice as done in H.264/AVC. Those short-term and long-term reference pictures are named intra-view references for simplicity. ii) Then, inter-view reference pictures and inter-view only reference pictures are appended after the intra-view references, according to

the view dependency order indicated in the active SPS and the "inter_view_flag" to form an initial reference picture list.

After the generation of an initial reference picture list in MVC, the initial reference picture list may be reordered by reference picture list reordering (RPLR) commands which may be included in a slice header. The RPLR process may reorder the intra-view reference pictures, inter-view reference pictures and inter-view only reference pictures into a different order than the order in the initial list. Both the initial list and final list after reordering must contain only a certain number of entries indicated by a syntax element in the slice header or the picture parameter set referred by the slice.

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.

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.

In the case of joint coding of texture and depth for depth-enhanced video, view synthesis can be utilized in the loop of the codec, thus providing view synthesis prediction (VSP). In VSP, a prediction signal, such as a VSP reference picture, is formed using a DIBR or view synthesis algorithm, utilizing texture and depth information. For example, a synthesized picture (i.e., VSP reference picture) may be introduced in the reference picture list in a similar way as it is done with interview reference pictures and inter-view only reference pictures. Alternatively or in addition, a specific VSP prediction mode for certain prediction blocks may be determined by the encoder, indicated in the bitstream by the encoder, and used as concluded from the bitstream by the decoder.

In MVC, both inter prediction and inter-view prediction use essentially the same motion-compensated prediction process. Inter-view reference pictures and inter-view only reference pictures are essentially treated as long-term reference pictures in the different prediction processes. Similarly, view synthesis prediction may be realized such a manner that it uses the essentially the same motion-compensated prediction process as inter prediction and inter-view prediction. To differentiate from motion-compensated prediction taking place only within a single view without any VSP, motion-compensated prediction that includes and is capable of flexibly

selecting mixing inter prediction, inter-prediction, and/or view synthesis prediction is herein referred to as mixed-direction motion-compensated prediction.

As reference picture lists in MVC and MVD may contain more than one type of reference pictures, i.e. inter reference pictures (also known as intra-view reference pictures), inter-view reference pictures, inter-view only reference pictures,and VSP reference pictures, a term prediction direction is defined to indicate the use of intra-view reference pictures (temporal prediction), inter-view prediction, or VSP. For example, an encoder may choose for a specific block a reference index that points to an inter-view reference picture, thus the prediction direction of the block is inter-view.

Motion vector (MV) prediction specified in H.264/AVC/MVC utilizes correlation which is present in neighboring blocks of the same image (spatial correlation) or in the previously coded image (temporal correlation). The motion vectors (MVs) of a currently coded block (cb) are estimated through a motion estimation and motion compensation process and are coded in differential pulse code modulation (DPCM) manner and transmitted in the form of a residual or a motion vector difference (MVd) between the motion vector prediction/predictor (MVp) and the actual MV as MVd(x, y) = MV(x, y) − MVp(x, y). The horizontal residual component MVd(x) = MV(x) – MVp(x) may be coded and transmitted in a separate codeword from the vertical residual component MVd(y) = MV(y) – MVp(y).

In H.264/AVC motion vector components MVd(x) and MVd(y) for P macroblocks or macroblock partitions are differentially coded using either median or directional prediction from spatially neighboring blocks, wherein a neighboring block may be a macroblock, macroblock partition or a sub-macroblock partition. Directional motion vector prediction is used for certain shapes of macroblock partitions, namely 8x16 and 16x8, when the reference index of cb is the same as in certain spatially neighboring block determined

by the shape of the current macroblock partition and its location within the current macroblock. In directional motion vector prediction, the motion vector of certain block is taken as the motion vector predictor for the current macroblock partition.

In median motion vector prediction of H.264/AVC, inherited in MVC as well, the reference indices of the neighboring blocks immediately above (block B), diagonally above and to the right (block C), and immediately left (block A) of the current block cb are first compared to the reference index of cb. If one and only one of the reference indices of blocks A, B, and C is equal to the reference index of cb, then the motion vector predictor for cb is equal to the motion vector of the block A, B, or C for which the reference index is equal to the reference index of cb. Otherwise, the motion vector predictor for cb is derived as a median value of the motion vectors of blocks A, B, and C regardless of their reference index value. Figure 5a shows how blocks A, B, and C are spatially related to the currently coded block (cb).

In other words, the median motion vector prediction process for deriving MVp is specified in H.264/AVC as follows:

1. When only one of the spatial neighboring blocks (A,B,C) has identical reference index as the current block (cb):

   MVp = mvN, where N is one of (A, B, C)

2. When more than one or no neighboring blocks (A,B,C) have identical reference index as the cb:

   MVp =median{mvA,mvB,mvC},

where mvA, mvB, mvC are motion vectors (without reference index) of the spatially neighboring blocks.

In H.264/AVC, the motion vectors and reference indexes for the neighboring blocks A, B, C of the current block cb are determined based on their availability as follows. If the top-right (C) block is not available, for example when it is in a different slice than cb or outside picture boundaries, the

location on the top-right (C) is replaced with the top-left block (D). If a block A, B, or C is not available, for example it is in a different slice than cb or outside picture boundaries or coded in intra mode, the motion vector mvA, mvB, or mvC, respectively, is equal to 0 and the reference index for block A, B, or C, respectively is -1 for motion vector prediction and mixed-direction motion-compensated prediction.

In addition to the motion-compensated macroblock modes for which a differential motion vector is coded, a P macroblock may also be coded in the so-called P_Skip type in H.264/AVC. For this coding type, no differential motion vector, reference index, or quantized prediction error signal is coded into the bitstream. The reference picture of a macroblock coded with the P_Skip type has index 0 in reference picture list 0. The motion vector used for reconstructing the P_Skip macroblock is obtained using median motion vector prediction for the macroblock without any differential motion vector being added. P_Skip may be beneficial for compression efficiency particularly in areas where the motion field is smooth.

In B slices of H.264/AVC, four different types of inter prediction are supported: uni-predictive from reference picture list 0, uni-directional from reference picture list 1, bi-predictive, direct prediction, and B_skip. The type of inter prediction can be selected separately for each macroblock partition. B slices utilize a similar macroblock partitioning as P slices. For a bi-predictive macroblock partition, the prediction signal is formed by a weighted average of motion-compensated list 0 and list 1 prediction signals. Reference indices, motion vector differences, as well as quantized prediction error signal may be coded for uni-predictive and bi-predictive B macroblock partitions.

Two direct modes are included in H.264/AVC, temporal direct and spatial direct, and one of them can be selected into use for a slice in a slice header. In the temporal direct mode, the reference index for reference picture list 1 is set to 0 and the reference index for reference picture list 0 is set to point to

the reference picture that is used in the co-located block (compared to cb) of the reference picture having index 0 in the reference picture list 1 if that reference picture is available, or set to 0 if that reference picture is not available. The motion vector predictor for cb is essentially derived by considering the motion information within a co-located block of the reference picture having index 0 in reference picture list 1. Motion vector predictors for a temporal direct block are derived by scaling a motion vector from the co-located block where the scaling weight is proportional to picture order count differences between the current picture and the reference pictures associated with the inferred reference indexes in list 0 and list 1, and by selecting the sign for the motion vector predictor depending on which reference picture list it is using. Figure 5b shows an example illustration of co-located blocks of the currently coded block (cb) for MVP of the temporal direct mode of H.264/AVC. The use of temporal direct mode is not supported in any present coding profile for MVC, even though the syntax of the standard supports also the temporal direct mode.

In spatial direct mode of H.264/AVC, motion information of spatially adjacent blocks is exploited in a similar manner to P blocks. Motion vector prediction in spatial direct mode can be divided into three steps: reference index determination, determination of uni- or bi-prediction, and motion vector prediction. In the first step, the reference picture with the minimum non-negative reference index (i.e., non-intra block) is selected from each of reference picture list 0 and reference picture list 1 of the neighboring blocks A, B, and C. If no non-negative reference index exists in reference picture list 0 of the neighboring blocks A, B, and C, and likewise no non-negative reference index exists in reference picture list 1 of the neighboring blocks A, B, and C, reference index 0 is selected for both reference picture lists.

The use of uni- or bi-prediction for H.264/AVC spatial direct mode is determined as follows: If a minimum non-negative reference index for both reference picture lists was found in the reference index determination step,

bi-prediction is used. If a minimum non-negative reference index for either but not both of reference picture list 0 or reference picture list 1 was found in the reference index determination step, uni-prediction from either reference picture list 0 or reference picture list 1, respectively, is used.

In the motion vector prediction for H.264/AVC spatial direct mode, certain conditions, such as whether a negative reference index was concluded in the first step, are checked and, if fulfilled, a zero motion vector is determined. Otherwise, the motion vector predictor is derived similarly to the motion vector predictor of P blocks using the motion vectors of spatially adjacent blocks A, B, and C.

No motion vector differences or reference indices are present in the bitstream for a direct mode block, while quantized prediction error signal may be coded and present therefore present in the bitstream.

A B_skip macroblock mode is similar to the direct mode but no prediction error signal is coded and included in the bitstream.

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).

The Multiview Video Coding (MVC) extension of H.264 referred above enables to implement a multiview functionality at the decoder, thereby allowing the development of three-dimensional (3D) multiview applications. Next, for better understanding the embodiments of the invention, some aspects of 3D multiview applications and the concepts of depth and disparity information closely related thereto are described briefly.

Stereoscopic video content consists of pairs of offset images that are shown separately to the left and right eye of the viewer. These offset images are captured with a specific stereoscopic camera setup and it assumes a particular stereo baseline distance between cameras.

Figure 1 shows a simplified 2D model of such stereoscopic camera setup. In Fig. 1, C1 and C2 refer to cameras of the stereoscopic camera setup, more particularly to the center locations of the cameras, b is the distance between the centers of the two cameras (i.e. the stereo baseline), f is the focal length of cameras and X is an object in the real 3D scene that is being captured. The real world object X is projected to different locations in images captured by the cameras C1 and C2, these locations being x1 and x2 respectively. The horizontal distance between x1 and x2 in absolute coordinates of the image is called disparity. The images that are captured by the camera setup are called stereoscopic images, and the disparity presented in these images creates or enhances the illusion of depth. For enabling the images to be shown separately to the left and right eye of the viewer, typically specific 3D glasses are required to be used by the viewer. Adaptation of the disparity is a key feature for adjusting the stereoscopic video content to be comfortably viewable on various displays.

However, disparity adaptation is not a straightforward process. It requires either having additional camera views with different baseline distance (i.e., b is variable) or rendering of virtual camera views which were not available in real world. Figure 2 shows a simplified model of such multiview camera setup that suits to this solution. This setup is able to provide stereoscopic video content captured with several discrete values for stereoscopic baseline and thus allow stereoscopic display to select a pair of cameras that suits to the viewing conditions.

A more advanced approach for 3D vision is having a multiview autostereoscopic display (ASD) that does not require glasses. The ASD emits more than one view at a time but the emitting is localized in the space in such a way that a viewer sees only a stereo pair from a specific viewpoint, as illustrated in Figure 3, wherein the boat is seen in the middle of the view when looked at the right-most viewpoint. Moreover, the viewer is able see another stereo pair from a different viewpoint, e.g. in Fig. 3 the boat is seen at the right border of the view when looked at the left-most viewpoint. Thus, motion parallax viewing is supported if consecutive views are stereo pairs and they are arranged properly. The ASD technologies may be capable of showing for example 52 or more different images at the same time, of which only a stereo pair is visible from a specific viewpoint. This supports multiuser 3D vision without glasses, for example in a living room environment.

The above-described stereoscopic and ASD applications require multiview video to be available at the display. The MVC extension of H.264/AVC video coding standard allows the multiview functionality at the decoder side. The base view of MVC bitstreams can be decoded by any H.264/AVC decoder, which facilitates introduction of stereoscopic and multiview content into existing services. MVC allows inter-view prediction, which can result into significant bitrate saving compared to independent coding of all views, depending on how correlated the adjacent views are. However, the rate of MVC coded video is proportional to the number of views. Considering that

ASD may require 52 views, for example, as input, the total bitrate for such number of views will challenge the constraints of the available bandwidth.

Consequently, it has been found that a more feasible solution for such multiview application 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.

Various DIBR methods have been proposed, some of which are briefly described in the following.

The DIBR techniques wherein a novel view is rendered from densely sampled views may be based on the so-called plenoptic function or its subsets. The plenoptic function was originally proposed as a 7D function to define the intensity of light rays passing through the camera center at every 3D location (3 parameters), at every possible viewing angle (2 parameters), for every wavelength, and at every time. If the time and the wavelength are known, the function simplifies into 5D function. The so-called light field system and Lumigraph system, in turn, represent objects in an outside-looking-in manner using a 4D subset of the plenoptic function.

However, DIBR techniques based on the plenoptic function may encounter a problem related to sampling theory. Based on sampling theory, minimum camera spacing density enabling to sample the plenoptic function densely enough to reconstruct the continuous function can be defined. Nevertheless, even the functions have been simplified in accordance with the minimum camera spacing density, enormous amount of data may still be required for storage. Theoretical approach may result in a camera array system with more than 100 cameras, which is impossible for most practical applications, due to limitation of camera size and storage capacity.

In contrast to the DIBR techniques more or less based on the plenoptic function, DIBR techniques related to image warping may be used to render a novel view from sparsely sampled views.

The 3D image warping techniques may conceptually construct a 3D point (named unprojection) and reproject it to a 2D image belonging to the new image plane. The whole system may have a number of views, while when interpolating a novel view, only one or more closest views may be used.

In 3D image warping, a perspective warp may be first used for small rotations in a scene. The perspective warp may compensate rotation but may not be sufficient to compensate translation, therefore depth of objects in the scene is considered. One approach is to divide a scene into multiple layers and each layer is independently rendered and warped. Then those warped layers may be composed to form the image for the novel view. Independent image layers may be composed in front-to-back manner. This approach may have a disadvantage that occlusion cycles may exist among three or more layers. An alternative solution is to resolve the composition with Z-buffering. It does not require the different layers to be non-overlapping.

Other warping techniques may employ explicit or implicit depth information (per-pixel), which thus can overcome certain restrictions, e.g., relating to translation. One can use e.g. pre-computed 3D warps, referred to as morph maps, which hold the per-pixel, image-space disparity motion vectors for a 3D warp of the associated reference image to the fiducial viewpoint. When generating a novel view, offset vectors in the direction specified by its morph-map entry is utilized to interpolate the depth value of the new pixel. If the reference image viewpoint and the fiducial viewpoint are close to each other, the linear approximation is good.

When pixels are mapped to the same pixel in the destination/target image, Z-buffering concept may be adopted to resolve the visibility. The offset vectors, which can also be referred to as disparity, are closely related to epipolar geometry, wherein the epipole of a second image is the projection of the center (viewpoint) of planar the second image to a first image.

A simplified model of a DIBR-based 3DV system is shown in Figure 4. 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 1. 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.

It is assumed that the depth or disparity information (Di) for a current block (cb) of texture data is available through decoding of coded depth or disparity information or can be estimated at the decoder side prior to decoding of the current texture block, and this information can be utilized in MVP.

In many embodiments the coding order of texture and depth view components with an access unit is such that the texture and depth view components of the base view are coded first in any order. Then, the depth view component of a non-base view is coded before the texture view component of the same non-base view. The depth view components are coded in their inter-view dependency order, and the texture view components are likewise coded in their inter-view dependency order. For example, there may be three texture and depth views (T0, T1, T2, D0, D1, D2) where the

inter-view dependency order is 0, 1, 2 (0 is the base view, 1 is a non-base which may be inter-view predicted from view 0, and 2 is a non-base view which may be inter-view predicted from views 0 and 1). These three texture and depth views may be coded for example in order T0 D0 D1 T1 D2 T2 or D0 D1 D2 T0 T1 T2 or D0 T0 D1 T1 D2 T2. The bitstream order of the coded view components may be the same as their coding order, and likewise the decoding order of the coded view components may be the same as the bitstream order. The inter-view dependency order of depth is typically, but needs not be, the same as the inter-view dependency order for texture. The number of depth views may differ from that of texture views. For example, no depth view may be coded for a texture view from which no other texture view is predicted. Slices of depth view components may be differentiated from slices of texture view components for example using a different NAL unit type value.

In some embodiments the coding/decoding order of texture and depth may be interleaved using smaller units than view components, such as on block or slice basis. The respective coding/decoding order of coded texture and depth units, such as blocks, may follow the ordering rules described in the previous paragraph. For example, there may be two spatially adjacent texture blocks, ta and tb, where tb follows ta in coding/decoding order, and two depth/disparity blocks, da and db, spatially co-located with ta and tb, respectively. When prediction parameters for ta and tb are derived with the assistance of da and db, respectively, the coding/decoding order of the blocks may be (da, ta, db, tb) or (da, db, ta, tb). The bitstream order of the blocks may be the same as their coding order.

In some embodiments the coding/decoding order of texture and depth may be interleaved using greater units than view components, such as group of pictures or coded video sequences.

It is assumed herein that the based depth or disparity information associated with currently coded block of texture data is utilized in MVP decisions, and therefore it is assumed that the depth or disparity information is available at the decoder side in advance. In some MVD systems, the texture data (2D video) is coded and transmitted along with pixel-wise depth map or disparity information. Thus, a coded block of texture data (Cb) can be pixel-wise associated with a block of depth/disparity data d(Cb). Figure 6 shows an example structure of video plus depth (MVD) data and how currently coded block of texture Cb and the texture ranging information, e.g. depth map d(Cb) are associated with each other.

In the case of MVD data coding, texture data can be coded with utilization of View Synthesis Prediction (VSP). A conventional implementation of the VSP assumes a frame level implementation, with frame-level post-processing for view synthesis artifacts suppression and for occlusion handling.

Figure 7 shows an example of a conventional VSP-enabled multiview video encoder. The encoder 800 receives 802 a block of a current frame of a texture view for encoding. The block can also be called as a current block. The current block is provided to a first combiner 804, such as a subtracting element, and to the motion estimator 806. The motion estimator 806 has access to a frame buffer 812 storing previously encoded frame(s) or the motion estimator 806 may be provided by other means one or more blocks of one or more previously encoded frames. The motion estimator 806 examines which of the one or more previously coded blocks might provide a good basis for using the block as a prediction reference for the current block. If an appropriate prediction reference has been found, the motion estimator 806 calculates a motion vector which indicates where the selected block is located in the reference frame with respect to the location of the current block in the current frame. The motion vector information may be encoded by a first entropy encoder 814. Information of the prediction reference is also provided to the motion predictor 810 which calculates the predicted block.

The first combiner 804 determines the difference between the current block and the predicted block 808. The difference may be determined e.g. by calculating difference between pixel values of the current block and corresponding pixel values of the predicted block. This difference can be called as a prediction error. The prediction error is transformed by a transform element 816 to a transform domain. The transform may be e.g. a discrete cosine transform (DCT). The transformed values are quantized by a quantizer 818. The quantized values can be encoded by the second entropy encoder 820. The quantized values can also be provided to an inverse quantizer 822 which reconstructs the transformed values. The reconstructed transformed values are then inverse transformed by an inverse transform element 824 to obtain reconstructed prediction error values. The reconstructed prediction error values are combined by a second combiner 826 with the predicted block to obtain reconstructed block values of the current block. The reconstructed block values are ordered in a correct order by an ordering element 828 and stored into the frame buffer 812.

The encoder 800 also comprises a view synthesis predictor 830 which may use texture view frames of one or more other views 832 and depth information 834 (e.g. the depth map) to synthesize other views 836 on the basis of e.g. two different views captured by two different cameras. The other texture view frames 832 and/or the synthesized views 836 may also be stored to the frame buffer 812 so that the motion estimator 806 may use the other views and/or synthesized views in selecting a prediction reference for the current block. In some cases (e.g. for particular types of reference pictures), the encoder may define the set of valid motion vector component values to be empty, in which case the encoding of syntax element(s) related to the motion vector component may be omitted and the value of the motion vector component may be derived in the decoding. In some cases, the encoder may define the set of valid motion vector component values to be smaller than a normal set of motion vector values for inter prediction, in which

case the entropy coding for the motion vector differences may be changed e.g. by using a different CABAC context for these motion vector differences than the context for conventional inter prediction motion vectors. The different CABAC contexts may be used by the first entropy encoder 814 when encoding motion vector information. In some cases, the likelihood of motion vector values may depend more heavily on the reference picture than on the neighboring motion vectors (e.g. spatially or temporally) and hence entropy coding may differ from that used for other motion vectors, e.g. a different initial CABAC context may be used, and/or a different CABAC binarization scheme may be used, and/or a different context may be maintained and adapted or a non-context-adaptive coding of a syntax element may be used e.g. based on the type of the reference picture (e.g. view synthesis picture, inter-view reference picture, inter-layer reference picture, inter/temporal reference picture), and/or the scalability layer of the current picture and reference picture and/or the view of the current picture and the reference picture and/or any other property of the used reference picture and potentially the current picture.

In the case of MVD coding, view synthesis can be utilized in the loop of the codec, thus providing view synthesis prediction (VSP). A view synthesis prediction (VSP) picture (reference component) is synthesized from coded texture views and depth views and contains samples that may be used for VSP, as disclosed in Figure 7.

A non-limiting example of view synthesis (VS) process is described below. Inputs of this process are decoded a luma component of the texture view component srcPicY, two chroma components srcPicCb and srcPicCr up-sampled to the resolution of srcPicY, and a depth picure DisPic.

The output of this process is a sample array of a synthetic reference component vspPic which is produced through disparity-based warping:

```
for( j = 0; j < PicHeigh ; j++ ) {
```

```
        for( i = 0; i < PicWidth; i++ ) {
        dX =  Disparity(DisPic(j,i));
        outputPicY[ i+dX, j ] = srcTexturePicY[ i, j ];
        if( chroma_format_idc  = =  1 ) {
                outputPicCb[ i+dX, j ] = normTexturePicCb[ i, j ]
                outputPicCr[ i+dX, j ] = normTexturePicCr[ i, j ]
                }
        }
}
```

where the function "Disparity()" converts a depth map value at spatial location i, j to a disparity value dX which reflects spatial displacement in horizontal direction of pixel of the current view to a new spatial location in a target view to which VS is performed.

Disparity is computed taking into consideration camera settings, such as translation between two views b, camera's focal length f and parameters of depth map representation (Z_{near}, Z_{far}) as shown below:

$$dX(i,j) = \frac{f \cdot b}{z(i,j)}; \quad z(i,j) = \frac{1}{\frac{DisPic(i,j)}{255} \cdot \left(\frac{1}{Z_{near}} - \frac{1}{Z_{far}} \right) + \frac{1}{Z_{far}}} \qquad (2)$$

The vspPic picture resulting from described above process may typically include various warping artifacts, such as holes and/or occlusions and to suppress those artifacts, various post-processing operations may be applied. However, these operations may be avoided to reduce computational complexity, since a view synthesis picture vspPic is utilized for a reference pictures for prediction and not outputted to a display. A synthesized picture {outputPicY, outputPicCb , outputPicCr} is introduced in the reference picture list in a similar way as it is done with interview reference pictures. Signaling and operations with reference picture list in the case of VSP may remain identical to those specified in H.264/AVC, clauses H.7.3.3.1 and H.7.4.3.1.

Similarly processes of motion information derivation and their applications in VSP may remain identical to processes specified for inter and interview prediction of H.264/AVC, clauses H.8.3. Therefore, introducing VSP in 3DV may do not affect such low level operations as motion information signaling and decoding, thus preserving low-level compatibility with existing H.264/AVC coding standard.

To develop an AVC/MVC compatible 3DV standard, MPEG 3DV has designed a 3DV-ATM reference test model. This test model utilizes in-loop View synthesis based Prediction (VSP).

Conventional design of in-loop VSP of 3DV-ATM decoder utilizes a forward projection, which requires a frame level implementation, i.e. synthesizing a complete frame to be used as reference picture. Since majority of AVC decoders implement motion compensation prediction (MCP) with reference frame at the block level, the current frame-based VSP implementation may be significantly more burdensome than MCP in terms of computational complexity, storage requirement, and memory access bandwidth.

According to an aspect of the invention, coding/decoding of block Cb of texture/video view # 1 is performed with depth/depth map/disparity or any other ranging information d(Cb) which is associated with this texture information and available prior to coding/decoding of texture block. At the decoder side, depth/depth map/disparity or any other ranging information may be made available either by decoding this information from coded bitstream, by decoder side estimation or any other means.

According to an embodiment, VSP for each texture block Cb is performed in a block based approach and the results of such block-based VSP may be different from a frame-level VSP implementation. Thus, the block-based VSP

prediction is performed with identical algorithm and in the identical processing order at the encoding and decoding sides.

According to an embodiment, VS process for each texture block Cb is performed in a block based approach and it results in producing (synthesizing, computing, interpolating) of pixel values in a referenced area of a VSP-frame R(Cb). Note that referenced area R(Cb) may be larger in size than Cb block and can be considered as "Motion Estimation/Motion Compensation (MCP) search area" of regular inter-prediction.

According to an embodiment, VS process for two different texture blocks Cb1 and Cb2 is performed in a block based approach and they may result in produced pixel values in associated referenced areas (2D block of pixel locations) of a VSP-frame R1(Cb) and R2(Cb) and these regions may overlap. Thus actual pixel values that belong to the overlapped regions of R1 and R2 will depend on the order VSP is performed, in other words on the order of coding/decoding Cb1 and Cb2. Therefore the VS process for Cb1 and Cb2 coding/decoding order is kept identical at encoder and decoder sides.

According to an embodiment, VS process for two different texture blocks Cb1 and Cb2 is performed independently and each coded block Cbi operates with own copy of referenced VSP regions R1(Cb) and R2(Cb). In such embodiment, the results of VSP for Cb1 and Cb2 remain independent and do not affect result of each other, thus allowing flexible and/or parallel implementation of VSP and coding/decoding process for blocks Cb1 and Cb2.

According to an embodiment, the process of VSP for Cb is performed with a 2-way projection approach and may utilize at least some of the following steps:

a. Ranging information d(Cb) in the view #1 is converted to a disparity values D(Cb).

b. Disparity samples of D(Cb) are analyzed to find minimal and maximal disparity values within the D(Cb) block, min_D and max_D correspondingly.

c. Disparity values min_D and max_D specify VSP source regions size of MxN in Texture (VSP_T) and depth images (VSP_D, such that VSP_D=d(VSP_T)) of view #0, where M is a size of the VSP source region in vertical direction (number of lines) and N=(max_D-min_D+1) is the size of source regions in horizontal direction.

d. Value K1 is estimated in such a way that arithmetic modulo operations with dividend (N+K1) and divisor n produces zero. Value K2 is estimated in such a way such that arithmetic modulo operations with dividend (M+K2) and divisor m produces zero

e. VSP source blocks {VSP_T and VSP_D} are extended in horizontal direction by K1 and in vertical direction by K2 pixels in both or in either of sides, in order to form a VSP source region size of (M+K2) x (N+K1).

f. VSP source region (VSP_T and VSP_D) in view #0 is split in integer number of not-overlapping blocks (Projection Block Units (PBU) of a fixed size (m x n) and VS process of each PBU is performed separately.

g. VS process utilizes VSP source information (VSP_T and VSP_D) of view #0 which is processed in a fixed-size PBUs and produces a referenced area R(Cb) in VSP-frame associated with a coded frame of view #1.

According to an embodiment, an alternative conception of utilization of weighted prediction parameters is specified for coding of multiview video data with use of VSP.

Next, a more detailed description of some embodiments is given below. According to an embodiment, an independent VS process is used for the currently coded block Cb. Let us assume that coded MVD data consists of texture and depth map components which represents multiple videos captured with a parallel camera set up and these captured views being rectified. Terms Ti and Di would represents texture and depth map components of view #i respectively. Texture and depth components of MVD data may be coded in different coding order, e.g. (T0, D0, T1, D1) or (D0, D1, T0, T1).

In an embodiment, it is assumed that depth map component Di is available (decoded or estimated at decoder side) prior to texture component Ti with the same value of i, and Di is utilized in the coding/decoding of Ti.

For example, let us assume that the following coding order is utilized: (T0, D0, D1, T1), and the texture component T1 is coded with the VSP, and the currently coded block Cb1 has a partition of 16x16. The depth map data associated with Cb1 is d(Cb1) and it consists of block of the depth map samples of the same size 16x16.

An embodiment of targeting coding MVD (multiview video plus depth) may comprise at least some of the following steps.

1. Depth-to-disparity conversion

Samples of depth map data d(Cb1) is converted to a block D(Cb1) of disparity samples. The process of conversion may be performed for example with following equations or with its integer arithmetic implementations:

$$Z(Cb1(j,i)) = \frac{1}{\frac{d(Cb1(j,i))}{255}\cdot\left(\frac{1}{Znear}-\frac{1}{Zfar}\right)+\frac{1}{Zfar}};\ D(Cb1(j,i)) = \frac{f\cdot b}{Z(Cb1(j,i))}; \quad (3)$$

where j and i are local or global spatial coordinates, d($Cb1(j,i)$) is a depth map value in depth map image of a view #1, Z is its actual depth value, and D is a disparity to a particular view #0. The parameters f, b, Z_{near} and Z_{far} are parameters specifying the camera setup; i.e. the used focal length (f), camera separation (b) between view #1 and view #0i and depth range (Z_{near},Z_{far}) representing parameters of depth map conversion.

2. An analysis of disparity to find disparity range

Following this, the block of disparity values D(Cb1) is analyzed for defining its disparity range (min, max disparity).

Min_D = min(D(Cb1(I,j)), Max_D = max(D(Cb(I,j)) (4)

3. Defining VSP source region in another view

Being applied to spatial coordinate (j and i) from the currently coded view #1 to a view #0, this disparity range defines a source data for VSP process {VSP_T and VSP_D, such that VSP_D = d(VSP_T)}, in corresponding view #0. E.g, assuming that Cb1=T(j:j+15;i:i+15), where i, j are spatial coordinates in T1, and associated spatial coordinates for VSP source in view #0 can be defined as:

$j_0 = j;$

$min\_i_0 = i\text{-}Max\_D;$

$max\_i_0 = i+15\text{-}min\_D$

where i_0 and j_0 are spatial coordinates in view #0, which define VSP source blocks of texture and depth in view #0 as:

VSP_T = T(j:j+15,min_i_0:max_i_0)

VSP_D = D(j:j+15,min_i_0:max_i_0)

4. Extending the VSP source blocks

The VSP source blocks VSP_T and VSP_D may be extended in horizontal directions with Ext_left, Ext_right values to satisfy requirements of a VSP process, i.e. multi-length filtering, coordinates should be within the actual image borders block size and resulting block size in horizontal direction shall be dividable without reminder by 4.

```
min_i=CLIP(min_i-Ext_Left,0,width_minus1);
max_i =CLIP(max_i+Ext_Right,0,width_minus1);

min_i = ( min_i >>2)<<2;
max_i =( max_i >>2)<<2;
```

The resulting VSP source blocks VSP_T and VSP_D have the size of [16 x vsp_w] and they are redefined as:

```
vsp_w = max_i - min_i +1;
modulo(vsp_w, 4) == 0 //remain from division by 4 is equal to 0
VSP_T = T(j:j+15,min_i:max_i)
VSP_D = D(j:j+15, min_i:max_i)
```

Figure 8 shows a visualization of the steps 2,3 and 4 above, wherein the straight dotted lines provide guidelines on horizontal-vertical correspondence, and the curved dotted lines Min_D and Max_D show disparity correspondence between images in View 0 and View 1.

5. Processing the VSP source blocks in several PBU blocks of a fixed size of 16x4

The VSP source blocks VSP_T and VSP_D span over vsp_w in horizontal direction and have fixed size in vertical direction, which is equal to vertical size of “Cb”. Value of vsp_w may be unknown in advance and to enable

block-based VSP, source blocks {VSP_T and VSP_D} are processed in tiles, so called Projection Block Unit (PBU) with size (e.g 16x4) known in advance. Note that PBU is a tile of VSP_T and VSP_D, and consists of both texture and depth components (PBU_T and PBU_D). Figure 9 shows a layout of a PBU within a VSP source blocks {VSP_T and VSP_D}.

6. Loop over the VSP source blocks with the fixed-sizes PBUs

A concept visualization of the Step 6 is shown in Figure 10. PBU_T is projected to a target view #0 with utilization of PBU_D with a projection algorithm, e.g. with BBView_1LWH_splatting specified in 3DV-ATM. Result of this projection not a complete VSP frame, but a reference R(Cb) area which is “populated” with corresponding texture pixels projected PBU_T utilized for prediction Cb in MCP.

```
For (c=0; c<vsp_w%d; c++)
        PBU_T = VSP_T(c)
        PBU_D = VSP_D(c)
        R0(Cb) = Project(PBU_T, PBU_D, view_id);
End;

// Project_PBU:
for( j = 0; j < 16; j++ ) {
        for( i = 0; i < 4; i++ ) {
        dX =  Disparity(PBU_D(j,i));
        R0[ j, i+dX ] = VSP_T[j,i ];
        }
}
```

The complete reference area R(Cb) may be created when all PBUs consisting of VSP source blocks {VSP_T, VSP_D} have been processed. Due to possible geometrical distortions present in multiview representation of

the real scene or due to possible inaccuracy of depth information, the projection of PBU may result to overwriting pixel values located in the same spatial locations of referenced area R(Cb).

7. Predicting Cb from R(Cb)

Cb is predicted from R(Cb) for example in a conventional for MCP way, reference index referencing the VSP frame, and MV components mv_x and mv_y are referencing particular spatial location in this VSP frame. Since in the current embodiment, it is assumed that the VSP frame is not being completely produced during the VSP process, some restrictions are imposed on the Motion Estimation process.

According to an embodiment, these restrictions may include one of the following:

a. Motion vector components are restricted to Zero, thus mv_x = mv_y = 0.

This would simplify decoding operations for motion vector decoding, VSP production and reference frame interpolation.

b. Vertical motion vector component is restricted to Zero, and horizontal motion vector component is restricted to some range [-a, a], thus mv_y = 0 and –a <mv_x < a.

This would simplify decoding operations for mv_y, restrict the size of R(Cb) area, relax computation demand for VSP, and allow avoid interpolation in vertical direction.

8. Memory management with R(Cb)

The VSP for a Cb is performed independently and results in independent R(Cb) area. The VSP process for different Cb may result in overlapping R(Cb) areas and may result in different sample values for a particular spatial

location. However, since each of these processes are independent, different Cb may be predicted from different pixel values although utilizing the identical motion information (reference index and MV components).

Figure 11 shows an encoder according to an embodiment as a simplified block diagram for implementing a ME/MCP chain of texture coding with use of the VSP as described in the above embodiments. As a difference to a prior art encoder of Figure 7, both the another texture view and the depth information are subjected directly to the VSP, which carries out the process on block-based level. It is noted that the VSP does not produce a complete VSP frame, but produces only reference area R(Cb) on request from ME/MCP chain.

In some embodiments, it is assumed that the VSP for every Cb is performed independently and each Cb is predicted from its individual copy of R(Cb), even if motion vectors refer to the same spatial coordinates of the virtual VSP frame

In yet another embodiment, samples of the reference area R(Cbi) are projected with VS process to an actual VSP frame and may be utilized for the prediction of another Cbj.

In an embodiment, the reference VSP-frame is produced with a fixed-size PBU units, and the encoder and the decoder maintain and update information, e.g. vsp_pbu_map, indicating which PBUs have already been synthesized and which PBUs are not yet synthesized. Once a PBU is processed, the information may be updated that the PBU has been processed, e.g. a corresponding vsp_pbu_map map may be updated to value of 1. Then, the VSP for this particular PBU may skipped during further processing. In an example embodiment, the processing steps with enumeration aligned with the one above may be performed as follows.

6. Loop over VSP source blocks with fixed-sizes PBUs

```
For (c=0; c<vsp_w%d; c++){
        pbu_idx = getIndexPBU( c);
        if (vsp_pbu_map(pbu_idx) == 0){
          PBU_T = VSP_T(c)
          PBU_D = VSP_D(c)
          R0(Cb) = Project(PBU_T, PBU_D, view_id);
        vsp_pbu_map(pbu_idx) = 1;
        }
}
```

8. Memory management with R(Cb)

Since R(Cb) for different Cb would be performed to the same VSP frame, and the produced reference areas R(Cb) may overlap, the VSP frame is updated during the coding/decoding of Cbs. Further, different Cb may be predicted from different pixel values even if utilizing the identical motion information (reference index and MV components). Therefore, to produce identical results the encoder and the decoder maintain the same coding/decoding Cb order and thus the same order of VSP-frame updating.

According to another embodiment, the above principles may be implemented in multi-directional VSP with within a Rate-Distortion-Optimization (RDO) loop. In the case of multiview 3DV coding, a VSP-frame may result from view synthesis from multiple reference views. E.g. assuming 3-view coding, MVD components may be coded with T0-D0-D1-D2-T1-T2 order. With this order, the texture view T1 can utilize VSP and the corresponding VSP-frame is projected from view #0. The texture view T2, in contrast, may utilize a VSP frame which is produced from view #0 and view #1, therefore it may utilize either multiple VSP-frames for coding/decoding, or competing VSP-frames may be fused to improve the quality of VSP in some specified process.

Decoding operations for this embodiment may be different from the above embodiments step 1. The decoder reads an information indicator from the bitstream or extracts it from a memory location available at the decoder, which information indicator specifies the view (view_id) from which the VSP should be performed. Different view_id (VSP direction) would have different input to the step 1, such as a translation parameter b or focal length and results in different disparity values. Following this, the decoder performs the above-specified steps 2-6 as it shown in the above embodiments with no changes.

On the encoder-side, no changes may be required and the encoder may perform the above steps 1 - 8 as such for all available views, which results in multiple copies of coded Cb. The view_id which provides minimal cost in some rate-distortion optimization may be selected for coding and may be signaled to the decoder side. Alternatively, the encoder may extract information on optimal VSP-direction from available information and perform coding of Cb without signaling. In such embodiments, decoder would perform extraction of view-id at the decoder side with a procedure providing an identical VSP-direction or source view(s) for VSP to that/those obtained by the encoder.

According to yet another embodiment, the above principles may be implemented in multi-directional VSP with Depth-Aware Selection. Alternatively or in addition to the above embodiment of multi-directional VSP with RDO, the VSP-direction may be selected at the encoder and decoder side based on the depth information available at the encoder/and decoder sides prior to coding/decoding Cb.

Since depth information d(Cb) within view 2, corresponding VSP_D0 from view 0, and VSP_D1 from view 1 are all available at encoder and decoder sides prior to coding of Cb, it can be utilized for decision making on preferable VSP direction for Cb. For example, the direction or view which

provides a minimal Euclidian distance between the average depth/disparity of d(Cb) and the average depth/disparity of VSP_D may be selected for prediction.

Cost1 = min(average(d(Cb)) – average(VSP_D1)

Cost2 = min(average(d(Cb)) – average(VSP_D2)

If (Cost1 < Cost2) Vsp_id = 1 Else Vsp_id = 2,

It is noted that a different distortion or similarity metric may be utilized in this embodiment.

According to an embodiment, the method of uni-, bi- and multi- directional VS process with signaling/deriving index of view that provides source data for VSP are applied in coding systems with VSP design different from described in this intention and may not require ranging information associated with current view be available prior to coding of texture information of the current view.

According to yet another embodiment, the above principles may be implemented in bi-directional VSP. Alternatively or in addition to the above embodiments of multi-directional VSP with RDO and multi-directional VSP with Depth-Aware Selection, Cb in view #2 can be predicted with a bi-directional VSP prediction. In such an embodiment, R0(Cb) and R1(Cb) may be created for example from reference views #0 and #1 and utilized for prediction of current Cb for example in view #2 in a form of weighted prediction.

According to yet another embodiment, weighted prediction may be applied to the uni-, multi- or bi-directional VSP of the above embodiments. VSP-frame is associated with a particular weight to be used in weighted prediction, which is either computed at the encoder side and signaled to the decoder, or computed at both the encoder and the decoder side for example from the extrinsic camera parameters, such as an absolute difference of translational camera position, view order, or view_id.

In yet another embodiment, the VSP assumes a projection of actual pixel values from a particular view (VSP-direction), weighting parameters for those pixels may be inherited from corresponding views. For example, wp1 parameters, which are utilized for inter-view prediction view #2 from view #0, and wp2 parameters, which are utilized for inter-view prediction view #2 from view #1, may be used for VSP references of view #2 created from view #0 and view #1, respectively.

In yet another embodiment, pixel data R(Cb) projected from a particular view may be re-scaled (normalized) with a corresponding weighting parameter.

In some embodiments with bi-directional VSP, pixel data R(Cb) may be computed as a weighted average of pixel data projected from view #0 and view #1.

In yet another embodiment, weighted prediction parameters can be estimated based on the depth information available at encoder or decoder side.

Wp1= function(d(Cb), VSP_D1)
Wp2= function(d(Cb), VSP_D2)

According to yet another embodiment, the method of weighting pixels of VS process or weighting parameters derivation are applied in coding systems with VSP design which is different from described in this intention and may not require ranging information associated with current view be available prior to coding of texture information of the current view.

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 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, a synthetic reference component (a.k.a. a VSP referene picture) may be associated with VSP reference index, which indicates which earlier view component(s) in the access unit are used as source in the view synthesis. A reference picture list modfication syntax may include an entry type indicating reordering of synthetic reference component of an indicated VSP reference index. It may be specified that for the current texture view component with view order index equal to vOIdx, a synthetic reference component with a VSP reference index equal to i refers to the output of the decoding process for view synthesis reference component generation or any similar view synthesis process when the inputs are the texture and depth view components with view order index equal to (vOIdx – 1 – i).

In some embodiments, the reference picture list modification syntax may be for example the following or similar.

ref_pic_list_3dvc_modification( ) {	C	Descriptor
if(slice_type % 5 != 2 && slice_type % 5 != 4) {		
ref_pic_list_modification_flag_l0	2	u(1)
if(ref_pic_list_modification_flag_l0)		
do {		
modification_of_pic_nums_idc	2	ue(v)
if(modification_of_pic_nums_idc == 0 || modification_of_pic_nums_idc == 1)		
abs_diff_pic_num_minus1	2	ue(v)
else if(modification_of_pic_nums_idc == 2)		
long_term_pic_num	2	ue(v)
else if (modification_of_pic_nums_idc == 4 || modification_of_pic_nums_idc == 5)		
abs_diff_view_idx_minus1	2	ue(v)
else if(modification_of_pic_nums_idc == 6)		
vsp_ref_idx	2	ue(v)
} while(modification_of_pic_nums_idc != 3)		
}		
if(slice_type % 5 == 1) {		
ref_pic_list_modification_flag_l1	2	u(1)
if(ref_pic_list_modification_flag_l1)		
do {		
modification_of_pic_nums_idc	2	ue(v)
if(modification_of_pic_nums_idc == 0 || modification_of_pic_nums_idc == 1)		
abs_diff_pic_num_minus1	2	ue(v)
else if(modification_of_pic_nums_idc == 2)		
long_term_pic_num	2	ue(v)
else if (modification_of_pic_nums_idc == 4 || modification_of_pic_nums_idc == 5)		
abs_diff_view_idx_minus1	2	ue(v)
else if(modification_of_pic_nums_idc == 6)		
vsp_ref_idx	2	ue(v)
} while(modification_of_pic_nums_idc != 3)		
}		
}		

An entry in the either do loop in the syntax may correspond to ordering of one temporal reference picture (modification_of_pic_nums_idc equal to 0 or 1), view component for inter-inter prediction (modification_of_pic_nums_idc equal to 4 or 5), or synthetic reference component (modification_of_pic_nums_idc equal to 6). vsp_ref_idx may include the VSP

reference index of the synthetic reference component included in a reference picture list.

In some embodiments, the reference picture list modification process for synthetic reference component (e.g. process for decoding modification_of_pic_nums_idc equal to 6 in the syntax above) may be done as follows. Inputs to this process may be an index refIdxLX and a reference picture list RefPicListX (with X being 0 or 1). Outputs of this process may be an incremented index refIdxLX and a modified reference picture list RefPicListX. The following procedure may be conducted to place a synthetic reference component with VSP reference index equal to vsp_ref_idx into the index position refIdxLX, shift the position of any other remaining pictures to later in the list, and increment the value of refIdxLX:

```
for( cIdx = num_ref_idx_lX_active_minus1 + 1; cIdx > refIdxLX; cIdx−− )
    RefPicListX[ cIdx ] = RefPicListX[ cIdx − 1]
RefPicListX[ refIdxLX++ ] = synthetic reference component with VSP reference index equal
to vsp_ref_idx
```

In some embodiments, multiple VSP-frames may be fused for example by averaging or the DIBR or view synthesis process may take as input view components from more than one view. In some embodiments, the derivation of VSP reference index associated with such multi-source VSP frame may use a mathematical formula that takes as input two values, such as two view order index values, each associated with an input view for the generation of the multi-source VSP frame, and produces a single value. The same derivation algorithm of VSP reference index may be used in the encoder and the decoder. For example, if the maximum number of views available as input views for VSP is N and an index i0 and i1, both in the range of 0 to N-1, inclusive, indicate the input views for VSP, the VSP reference index may be derived as (i0 * N + i1). In some embodiments, reference picture list modification syntax may include more than one VSP reference index per a synthetic reference component to be reordered or placed in a reference picture list. If there are more than one VSP reference index per a synthetic reference component, then the VSP process to generate the synthetic

reference component may use as input the pictures associated with the listed VSP reference indexes.

In the above, example embodiments have been described in relation to particular codecs, such as H.264/AVC and 3DV-ATM. It needs to be understood, however, that embodiments can be realized similarly with other codecs.

In the above, 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 some embodiments, the VSP processes may be run in parallel both at encoder and decoder side. The embodiments also enable a hardware-friendly block-based architecture of the VSP.

The following describes in further detail suitable apparatus and possible mechanisms for implementing the embodiments of the invention. In this regard reference is first made to Figure 12 which shows a schematic block diagram of an exemplary apparatus or electronic device 50, which may incorporate a codec according to an embodiment of the invention.

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.

As disclosed in Figure 13, 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 other embodiments of the invention, the apparatus may receive the video image data for processing from another device prior to transmission and/or storage. In other embodiments of the invention, the apparatus 50 may receive either wirelessly or by a wired connection the image for coding/decoding.

With respect to Figure 14, 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 14 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 apparatus 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.

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,

apparatus, 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 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.

What is claimed is:

1. A method for encoding depth/disparity enhanced multiview video content comprising:

obtaining a first uncompressed texture block (Cb1) of a first texture picture representing a first view;

obtaining a first depth/disparity block (d(Cb1)) associated with the first texture block;

encoding the first uncompressed texture block (Cb1) with a block-based view synthesis prediction (VSP) process utilizing the first depth/disparity block (d(Cb1)) and VSP source data {VSP_T2(Cb1), VSP_D2(Cb2)} available from a coded second view.

2. A method according to claim 1, further comprising

encoding each uncompressed texture block (Cb1) of the first texture picture with the block-based view synthesis prediction (VSP) process utilizing the associated depth/disparity block (d(Cb1)) and VSP source data {VSP_T2(Cb1), VSP_D2(Cb2)} available from the coded second view.

3. A method according to claim 1 or 2, wherein

the view prediction synthesis process (VSP) on a block-level produces pixel values in a referenced area (R(Cb1)) of a VSP frame, which is associated with the first texture picture.

4. A method according to claim 3, further comprising

obtaining a second uncompressed texture block (Cb2) of a first texture picture representing the first view and a second depth/disparity block (d(Cb2)) associated with the second texture block;

encoding the second uncompressed texture block (Cb2) with the block-based view synthesis prediction (VSP) process utilizing the second depth/disparity block (d(Cb2)) and VSP source data {VSP_T2(Cb1), VSP_D2(Cb2)} available from the coded second view such that pixel values

in a referenced area (R(Cb2)) of a VSP frame overlap with the pixel values in the referenced area (R(Cb1)) of the VSP frame.

5. A method according to claim 3, further comprising

obtaining a second uncompressed texture block (Cb2) of a first texture picture representing the first view and a second depth/disparity block (d(Cb2)) associated with the second texture block;

encoding the second uncompressed texture block (Cb2) with the block-based view synthesis prediction (VSP) process utilizing the second depth/disparity block (d(Cb2)) and VSP source data {VSP_T2(Cb1), VSP_D2(Cb2)} available from the coded second view such that pixel values in a referenced area (R(Cb2)) of a VSP frame are independent of the pixel values in the referenced area (R(Cb1)) of the VSP frame.

6. A method according to any preceding claim, wherein the view prediction synthesis (VSP) process further comprises:

converting ranging information d(Cb)1 associated with the first texture block to a disparity values D(Cb)1;

analyzing the disparity values D(Cb)1 to find minimal (min_D) and maximal (max_D) disparity values;

specifying view synthesis prediction (VSP) source regions in a second view with size of NxM in texture image (VSP_T) and in associated depth image (VSP_D) on the basis of the disparity values min_D and max_D, where M is a size of the VSP source region in vertical direction and N=(max_D-min_D+1) is the size of VSP source region in horizontal direction;

extending the VSP source regions in horizontal and/or vertical direction by K1 and K2 pixels respectively; and

generating, on the basis of VSP source regions (VSP_T and VSP_D) of the second coded view, a referenced area R(Cb) in the VSP frame associated with the first coded view.

7. A method according to claim 6, further comprising

converting ranging information d(Cb) to a disparity values D(Cb) such that it reflects a difference in spatial coordinates between a particular sample of Cb taken from a first coded view and the corresponding sample in a second coded view.

8. A method according to claim 6, further comprising

estimating a value K1 such that arithmetic modulo operations with dividend (N+K1) and divisor *n* produces zero; and

estimating a value K2 such that arithmetic modulo operations with dividend (M+K2) and divisor *m* produces zero.

9. A method according to claim 6, further comprising

splitting the VSP source region (VSP_T and VSP_D) in integer number of non-overlapping blocks (PBU) of a fixed size (n x m) and processing each PBU independently.

10. A method according to any preceding claim, further comprising

obtaining weighing parameters for pixel values in the referenced area (R(Cbi)) of the VSP frame, which is synthesized or projected from a particular view; and

weighing the pixel values with corresponding weighing parameters.

11. A method according to claim 6, further comprising

normalizing pixel values in the referenced area R(Cbi) of the VSP frame, which is synthesized or projected from a particular view with a corresponding weighing parameter.

12. A method according to claim 6, further comprising

calculating a weighted average of pixel values in the referenced area R(Cbi) of the VSP frame projected from a first view and a second view.

13. A method according to claim 1, further comprising

encoding the uncompressed texture block Cb with the block-based view synthesis prediction (VSP) process that is performed from a particular reference view among several available views;

deriving an identification of the view to be used for said VSP process prior to encoding of the uncompressed texture block Cb; and

providing said identification to a decoder in a bitstream.

14. A method according to claim 13, further comprising

encoding the first uncompressed texture block Cb with the block-based view synthesis prediction (VSP) process that is performed from a first available reference view;

calculating a first rate distortion metric cost for the encoded block Cb;

encoding the first uncompressed texture block Cb with the block-based view synthesis prediction (VSP) process that is performed from a second available reference view;

calculating a second rate distortion metric cost for the encoded block Cb;

selecting an optimal reference view of view synthesis process by finding minimal rate distortion cost metric from said first and second rate distortion metric cost;

encoding the first uncompressed texture block Cb with block-based view synthesis prediction (VSP) process that performed from the optimal reference view; and

signaling the selected reference view for VSP to a decoder in a bitstream.

15. An apparatus for encoding depth/disparity enhanced multiview video content, said apparatus comprising a video encoder configured for:

obtaining a first uncompressed texture block (Cb1) of a first texture picture representing a first view;

obtaining a first depth/disparity block (d(Cb1)) associated with the first texture block;

encoding the first uncompressed texture block (Cb1) with a block-based view synthesis prediction (VSP) process utilizing the first depth/disparity block (d(Cb1)) and VSP source data {VSP_T2(Cb1) and VSP_D2(Cb2)} available from a coded second view.

16. An apparatus according to claim 15, the video encoder further being configured for

encoding each uncompressed texture block (Cb1) of the first texture picture with the block-based view synthesis prediction (VSP) process utilizing the associated depth/disparity block (d(Cb1)) and VSP source data {VSP_T2(Cb1) and VSP_D2(Cb2)} available from the coded second view.

17. An apparatus according to claim 15 or 16, wherein

the view prediction synthesis process (VSP) on a block-level produces pixel values in a referenced area (R(Cb1)) of a VSP frame, which is associated with the first texture picture.

18. An apparatus according to claim 17, the video encoder further being configured for

obtaining a second uncompressed texture block (Cb2) of the first texture picture representing the first view and a second depth/disparity block (d(Cb2)) associated with the second texture block;

encoding the second uncompressed texture block (Cb2) with the block-based view synthesis prediction (VSP) process utilizing the second depth/disparity block (d(Cb2)) and VSP source data {VSP_T2(Cb1) and VSP_D2(Cb2)} available from the coded second view such that pixel values in a referenced area (R(Cb2)) of a VSP frame overlap with the pixel values in the referenced area (R(Cb1)) of the VSP frame.

19. An apparatus according to claim 17, the video encoder further being configured for

obtaining a second uncompressed texture block (Cb2) of the first texture picture representing the first view and a second depth/disparity block (d(Cb2)) associated with the second texture block;

encoding the second uncompressed texture block (Cb2) with the block-based view synthesis prediction (VSP) process utilizing the second depth/disparity block (d(Cb2)) and VSP source data {VSP_T2(Cb1) and VSP_D2(Cb2)} available from the coded second view such that pixel values in a referenced area (R(Cb2)) of a VSP frame are independent of the pixel values in the referenced area (R(Cb1)) of the VSP frame.

20. An apparatus according to any of claims 15 - 19, the video encoder further being configured for

converting ranging information d(Cb)1 associated with the first texture block to a disparity values D(Cb)1;

analyzing the disparity values D(Cb)1 to find minimal (min_D) and maximal (max_D) disparity values;

specifying view synthesis prediction (VSP) source regions in a second view with size of NxM in texture image (VSP_T) and in associated depth image (VSP_D) on the basis of the disparity values min_D and max_D, where M is a size of the VSP source region in vertical direction and N=(max_D-min_D+1) is the size of VSP source region in horizontal direction;

extending the VSP source regions in horizontal and/or vertical direction by K1 and K2 pixels respectively; and

generating, on the basis of VSP source regions (VSP_T and VSP_D) of the second coded view, a referenced area R(Cb) in the VSP frame associated with the first coded view.

21. An apparatus according to claim 20, the video encoder further being configured for

converting ranging information d(Cb) to a disparity values D(Cb) such that it reflects a difference in spatial coordinates between a particular sample of Cb taken from a first coded view and the corresponding sample in a second coded view.

22. An apparatus according to claim 20, the video encoder further being configured for

estimating a value K1 such that arithmetic modulo operations with dividend (N+K1) and divisor *n* produces zero; and

estimating a value K2 such that arithmetic modulo operations with dividend (M+K2) and divisor *m* produces zero.

23. An apparatus according to claim 20, the video encoder further being configured for

splitting the VSP source region (VSP_T and VSP_D) in integer number of non-overlapping blocks (PBU) of a fixed size (n x m) and processing each PBU independently.

24. An apparatus according to any of claims 15 - 19, the video encoder further being configured for

obtaining weighing parameters for pixel values in the referenced area (R(Cbi)) of the VSP frame, which is synthesized or projected from a particular view; and

weighing the pixel values with corresponding weighing parameters.

25. An apparatus according to claim 24, the video encoder further being configured for

normalizing pixel values in the referenced area R(Cbi) of the VSP frame, which is synthesized or projected from a particular view with a corresponding weighing parameter.

26. An apparatus according to claim 24, the video encoder further being configured for

calculating a weighted average of pixel values in the referenced area R(Cbi) of the VSP frame projected from a first view and a second view.

27. An apparatus according to claim 15, the video encoder further being configured for

encoding the uncompressed texture block Cb with the block-based view synthesis prediction (VSP) process that is performed from a particular reference view among several available views;

deriving an identification of the view to be used for said VSP process prior to encoding of the uncompressed texture block Cb; and

providing said identification to a decoder in a bitstream.

28. An apparatus according to claim 27, the video encoder further being configured for

encoding the first uncompressed texture block Cb with the block-based view synthesis prediction (VSP) process that is performed from a first available reference view;

calculating a first rate distortion metric cost for the encoded block Cb;

encoding the first uncompressed texture block Cb with the block-based view synthesis prediction (VSP) process that is performed from a second available reference view;

calculating a second rate distortion metric cost for the encoded block Cb;

selecting an optimal reference view of view synthesis process by finding minimal rate distortion cost metric from said first and second rate distortion metric cost;

encoding the first uncompressed texture block Cb with block-based view synthesis prediction (VSP) process that performed from the optimal reference view; and

signaling the optimal reference view for VSP to a decoder in a bitstream.

29. A computer readable storage medium stored with code thereon for use by an apparatus, which when executed by a processor, causes the apparatus to perform:

obtaining a first uncompressed texture block (Cb1) of a first texture picture representing a first view;

obtaining a first depth/disparity block (d(Cb1)) associated with the first texture block;

encoding the first uncompressed texture block (Cb1) with a block-based view synthesis prediction (VSP) process utilizing the first depth/disparity block (d(Cb1)) and VSP source data {VSP_T2(Cb1) and VSP_D2(Cb2)} available from a coded second view.

30. At least one processor and at least one memory, said at least one memory stored with code thereon, which when executed by said at least one processor, causes an apparatus to perform:

obtaining a first uncompressed texture block (Cb1) of a first texture picture representing a first view;

obtaining a first depth/disparity block (d(Cb1)) associated with the first texture block;

encoding the first uncompressed texture block (Cb1) with a block-based view synthesis prediction (VSP) process utilizing the first depth/disparity block (d(Cb1)) and VSP source data {VSP_T2(Cb1) and VSP_D2(Cb2)} available from a coded second view.

31. A video encoder configured for

obtaining a first uncompressed texture block (Cb1) of a first texture picture representing a first view;

obtaining a first depth/disparity block (d(Cb1)) associated with the first texture block;

encoding the first uncompressed texture block (Cb1) with a block-based view synthesis prediction (VSP) process utilizing the first depth/disparity block (d(Cb1)) and VSP source data {VSP_T2(Cb1) and VSP_D2(Cb2)} available from a coded second view.

32. A method for decoding depth/disparity enhanced multiview video content comprising:

obtaining a first encoded texture block (Cb1) of a first texture picture representing a first view;

obtaining a first depth/disparity block (d(Cb1)) associated with the first encoded texture block;

decoding the first encoded texture block (Cb1) with a block-based view synthesis prediction (VSP) process utilizing the first depth/disparity block (d(Cb1)) and VSP source data {VSP_T2(Cb1) and VSP_D2(Cb2)} available from a coded second view .

33. A method according to claim 32, further comprising

decoding each encoded texture block (Cbi) of the first texture picture with the block-based view synthesis prediction (VSP) process utilizing the associated depth/disparity block (d(Cbi)) and VSP source data {VSP_T2(Cb1) and VSP_D2(Cb2)} available from the coded second view.

34. A method according to claim 33 or 34, wherein

the view prediction synthesis process (VSP) on a block-level produces pixel values in a referenced area (R(Cbi)) of a VSP frame.

35. A method according to claim 34, further comprising

obtaining a second encoded texture block (Cb2) of the first texture picture representing a first view and a second depth/disparity block (d(Cb2)) associated with the second texture block;

encoding the second encoded texture block (Cb2) with the block-based view synthesis prediction (VSP) process utilizing the second

depth/disparity block (d(Cb2)) and VSP source data {VSP_T2(Cb1) and VSP_D2(Cb2)} available from the coded second view such that pixel values in a referenced area (R(Cb2)) of a VSP frame overlap with the pixel values in the referenced area (R(Cb1)) of the VSP frame.

36. A method according to claim 34, further comprising

obtaining a second encoded texture block (Cb2) of the first texture picture representing a first view and a second depth/disparity block (d(Cb2)) associated with the second texture block;

encoding the second encoded texture block (Cb2) with the block-based view synthesis prediction (VSP) process utilizing the second depth/disparity block (d(Cb2)) and VSP source data {VSP_T2(Cb1) and VSP_D2(Cb2)} available from the coded second view such that pixel values in a referenced area (R(Cb2)) of a VSP frame are independent of the pixel values in the referenced area (R(Cb1)) of the VSP frame.

37. A method according to any of claims 32 – 36, wherein the view prediction synthesis (VSP) process further comprises:

converting ranging information d(Cb)1 associated with the first texture block to a disparity values D(Cb)1;

analyzing the disparity values D(Cb)1 to find minimal (min_D) and maximal (max_D) disparity values;

specifying view synthesis prediction (VSP) source regions in a second view with size of NxM in texture image (VSP_T) and in associated depth image (VSP_D) on the basis of the disparity values min_D and max_D, where M is a size of the VSP source region in vertical direction and N=(max_D-min_D+1) is the size of VSP source region in horizontal direction;

extending the VSP source regions in horizontal and/or vertical direction by K1 and K2 pixels respectively; and

generating, on the basis of VSP source regions (VSP_T and VSP_D) of the second coded view, a referenced area R(Cb) in the VSP frame associated with the first coded view.

38. A method according to claim 37, further comprising

converting ranging information d(Cb) to a disparity values D(Cb) such that it reflects a difference in spatial coordinates between a particular sample of Cb taken from a first coded view and the corresponding sample in a second coded view.

39. A method according to claim 37, further comprising

estimating a value K1 such that arithmetic modulo operations with dividend (N+K1) and divisor *n* produces zero; and

estimating a value K2 such that arithmetic modulo operations with dividend (M+K2) and divisor *m* produces zero.

40. A method according to claim 37, further comprising

splitting the VSP source region (VSP_T and VSP_D) in integer number of non-overlapping blocks (PBU) of a fixed size (n x m) and processing each PBU independently.

41. A method according to any of claims 32 - 36, further comprising

obtaining weighing parameters for pixel values in the referenced area (R(Cbi)) of the VSP frame, which is synthesized or projected from a particular view; and

weighing the pixel values with corresponding weighing parameters.

42. A method according to claim 41, further comprising

normalizing pixel values in the referenced area R(Cbi) of the VSP frame, which is synthesized or projected from a particular view with a corresponding weighing parameter.

43. A method according to claim 41, further comprising

calculating a weighted average of pixel values in the referenced area R(Cbi) of the VSP frame projected from a first view and a second view.

44. A method according to claim 32, further comprising

obtaining an optimal reference view for VSP from a bitstream, said optimal reference view for VSP having been defined as minimal rate distortion cost metric calculated from a first and a second reference view; and

decoding the first uncompressed texture block Cb with block-based view synthesis prediction (VSP) process that performed from the optimal reference view.

45. An apparatus for decoding depth/disparity enhanced multiview video content, said apparatus comprising a video decoder configured for:

obtaining a first encoded texture block (Cb1) of a first texture picture representing a first view;

obtaining a first depth/disparity block (d(Cb1)) associated with the first texture block;

decoding the first encoded texture block (Cb1) with a block-based view synthesis prediction (VSP) process utilizing the first depth/disparity block (d(Cb1)) and VSP source data {VSP_T2(Cb1), VSP_D2(Cb2)} available from a coded second view.

46. An apparatus according to claim 45, the video decoder further being configured for

decoding each encoded texture block (Cb1) of the first texture picture representing the first view with the block-based view synthesis prediction (VSP) process utilizing the associated depth/disparity block (d(Cb1)) and VSP source data {VSP_T2(Cb1), VSP_D2(Cb2)} available from a coded second view.

47. An apparatus according to claim 45 or 46, wherein

the view prediction synthesis process (VSP) on a block-level produces pixel values in a referenced area (R(Cbi)) of a VSP frame associated with the first texture picture.

48. An apparatus according to claim 47, the video decoder further being configured for

obtaining a second encoded texture block (Cb2) of the first texture representing the first view picture and a second depth/disparity block (d(Cb2)) associated with the second texture block;

decoding the second encoded texture block (Cb2) with the block-based view synthesis prediction (VSP) process utilizing the second depth/disparity block (d(Cb2)) and VSP source data {VSP_T2(Cb1), VSP_D2(Cb2)} available from a coded second view such that pixel values in a referenced area (R(Cb2)) of a VSP frame overlap with the pixel values in the referenced area (R(Cb1)) of the VSP frame.

49. An apparatus according to claim 47, the video decoder further being configured for

obtaining a second encoded texture block (Cb2) of the first texture picture representing the first view and a second depth/disparity block (d(Cb2)) associated with the second texture block;

decoding the second encoded texture block (Cb2) with the block-based view synthesis prediction (VSP) process utilizing the second depth/disparity block (d(Cb2)) and VSP source data {VSP_T2(Cb1), VSP_D2(Cb2)} available from a coded second view such that pixel values in a referenced area (R(Cb2)) of a VSP frame are independent of the pixel values in the referenced area (R(Cb1)) of the VSP frame.

50. An apparatus according to any of claims 45 - 49, the video decoder further being configured for

converting ranging information d(Cb)1 associated with the first texture block to a disparity values D(Cb)1;

analyzing the disparity values D(Cb)1 to find minimal (min_D) and maximal (max_D) disparity values;

specifying view synthesis prediction (VSP) source regions in a second view with size of NxM in texture image (VSP_T) and in associated depth image (VSP_D) on the basis of the disparity values min_D and max_D, where M is a size of the VSP source region in vertical direction and N=(max_D-min_D+1) is the size of VSP source region in horizontal direction;

extending the VSP source regions in horizontal and/or vertical direction by K1 and K2 pixels respectively; and

generating, on the basis of VSP source regions (VSP_T and VSP_D) of the second coded view, a referenced area R(Cb) in the VSP frame associated with the first coded view.

51. An apparatus according to any of claims 45 - 50, the video decoder further being configured for

obtaining weighing parameters for pixel values in the referenced area (R(Cbi)) of the VSP frame which is synthesized or projected from a particular view; and

weighing the pixel values with corresponding weighing parameters.

52. An apparatus according to claim 51, the video decoder further being configured for

normalizing pixel values in the referenced area R(Cbi) of the VSP frame which is synthesized or projected from a particular view with a corresponding weighing parameter.

53. An apparatus according to claim 51, the video decoder further being configured for

calculating a weighted average of pixel values in the referenced area R(Cbi) of the VSP frame which is synthesized or projected from a first view and a second view.

54. An apparatus according to claim 45, the video decoder further being configured for

obtaining an optimal reference view for VSP from a bitstream, said optimal reference view for VSP having been defined as minimal rate distortion cost metric calculated from a first and a second reference view; and

decoding the first uncompressed texture block Cb with block-based view synthesis prediction (VSP) process that performed from the optimal reference view.

55. A computer readable storage medium stored with code thereon for use by an apparatus, which when executed by a processor, causes the apparatus to perform:

obtaining a first encoded texture block (Cb1) of a first texture picture representing a first view;

obtaining a first depth/disparity block (d(Cb1)) associated with the first texture block;

decoding the first encoded texture block (Cb1) with a block-based view synthesis prediction (VSP) process utilizing the first depth/disparity block (d(Cb1)) and VSP source data {VSP_T2(Cb1), VSP_D2(Cb2)} available from a coded second view.

56. At least one processor and at least one memory, said at least one memory stored with code thereon, which when executed by said at least one processor, causes an apparatus to perform:

obtaining a first encoded texture block (Cb1) of a first texture picture representing a first view;

obtaining a first depth/disparity block (d(Cb1)) associated with the first texture block;

decoding the first encoded texture block (Cb1) with a block-based view synthesis prediction (VSP) process utilizing the first depth/disparity block (d(Cb1)) and VSP source data {VSP_T2(Cb1), VSP_D2(Cb2)} available from a coded second view.

57. A video decoder configured for

obtaining a first encoded texture block (Cb1) of a first texture picture representing a first view;

obtaining a first depth/disparity block (d(Cb1)) associated with the first texture block;

decoding the first encoded texture block (Cb1) with a block-based view synthesis prediction (VSP) process utilizing the first depth/disparity block (d(Cb1)) and VSP source data {VSP_T2(Cb1), VSP_D2(Cb2)} available from a coded second view.

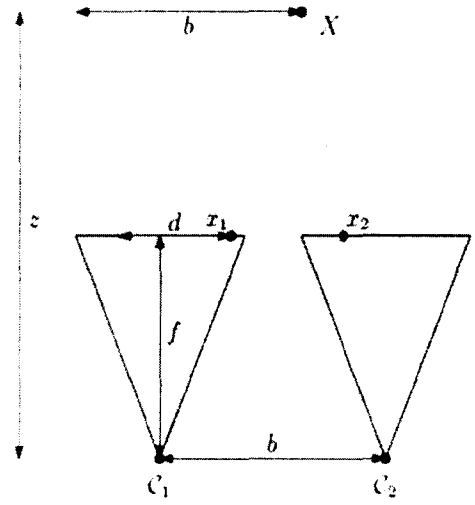

Fig. 1

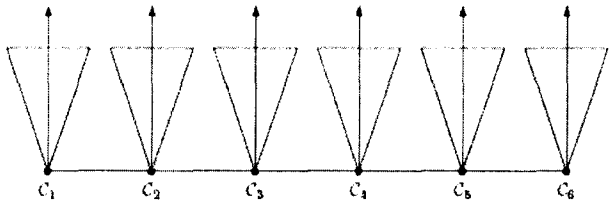

Fig. 2

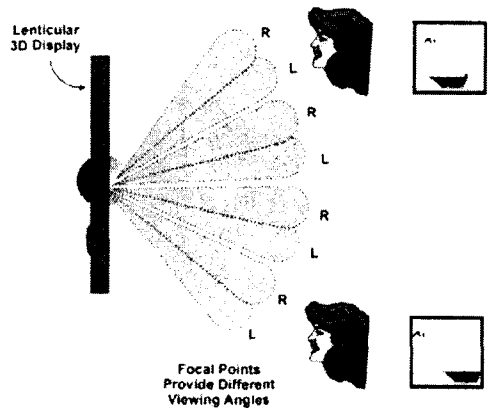

Fig. 3

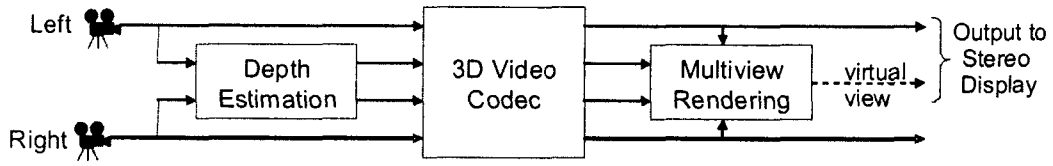

Fig. 4

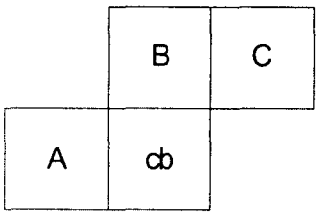

Fig. 5a

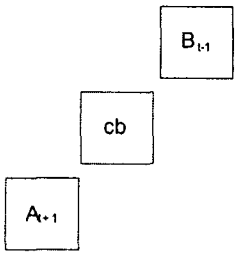

Fig.5b

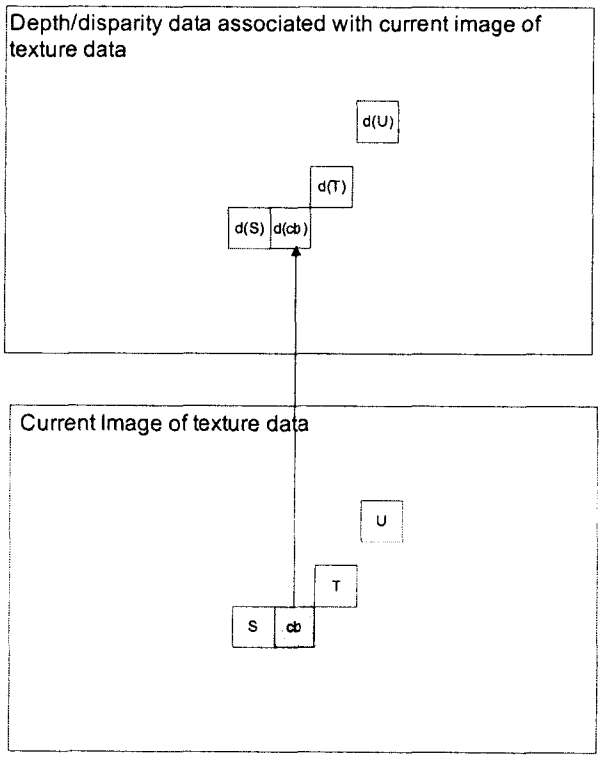

Fig. 6

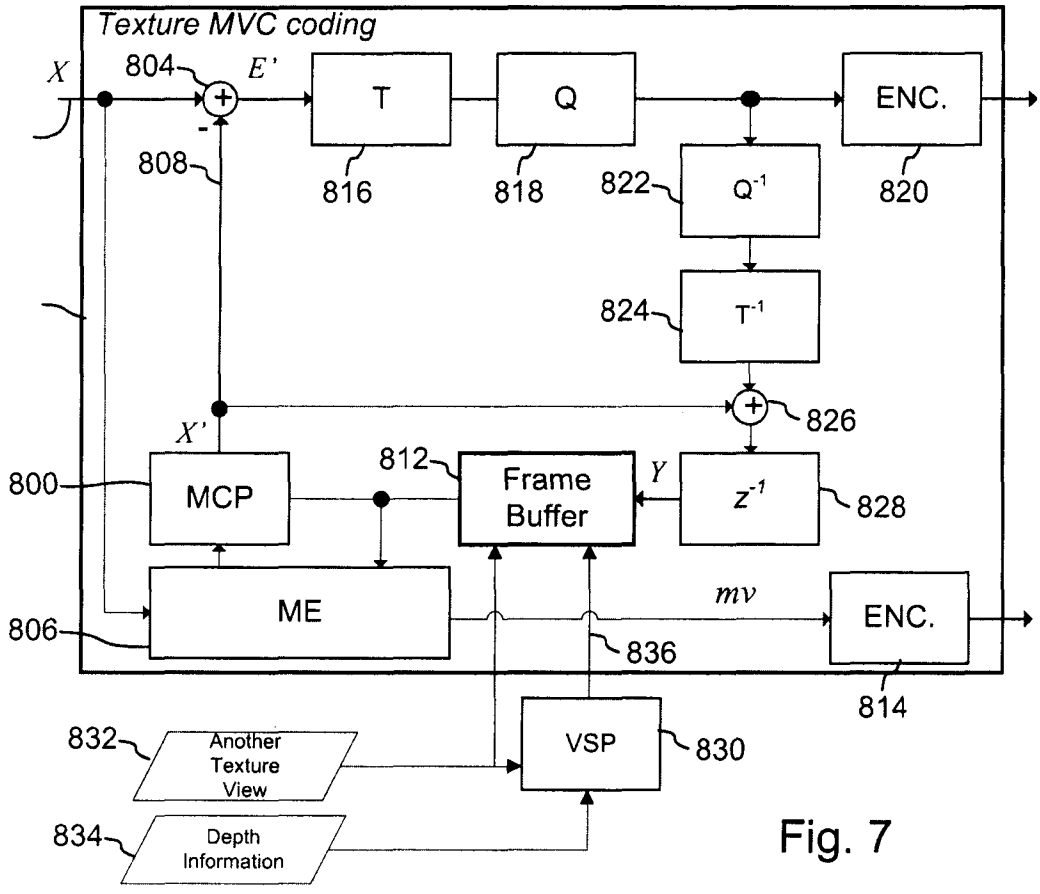
Texture MVC coding
X
804
E'
T
Q
ENC.
808
816
818
822
Q-1
820
824
T-1
826
X'
812
Frame Buffer
Y
800
MCP
z-1
828
ME
mv
ENC.
806
836
814
832
Another Texture View
VSP
830
834
Depth Information

Fig. 7

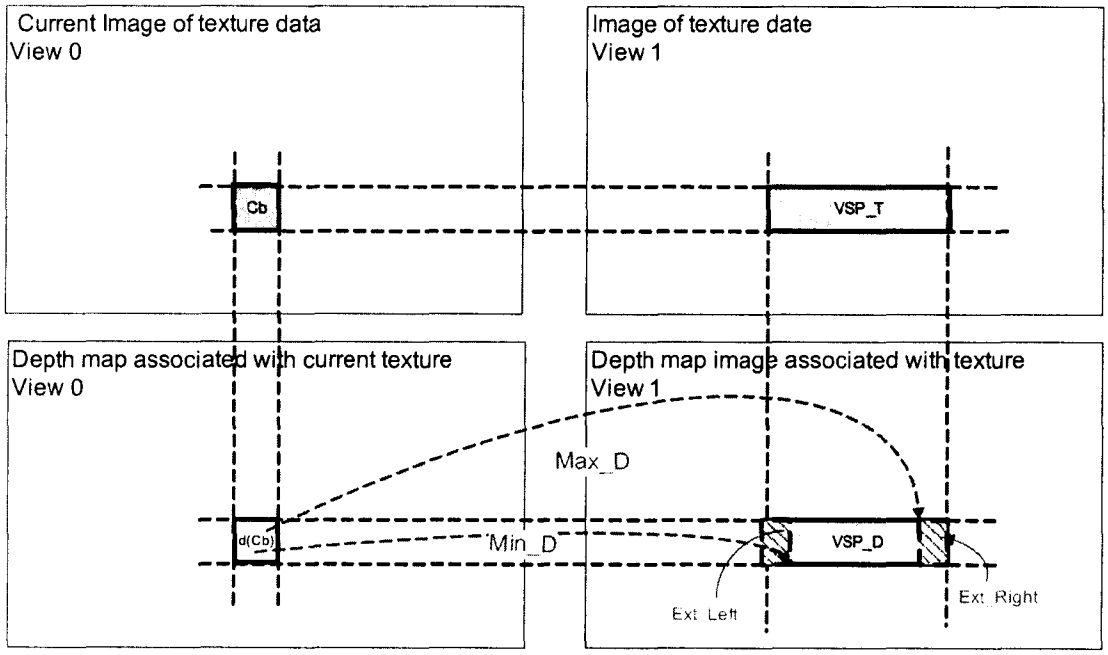
Current Image of texture data
View 0
Image of texture date
View 1
Cb
VSP_T
Depth map associated with current texture
View 0
Depth map image associated with texture
View 1
Max_D
d(Cb)
Min_D
VSP_D
Ext Left
Ext Right

Fig. 8

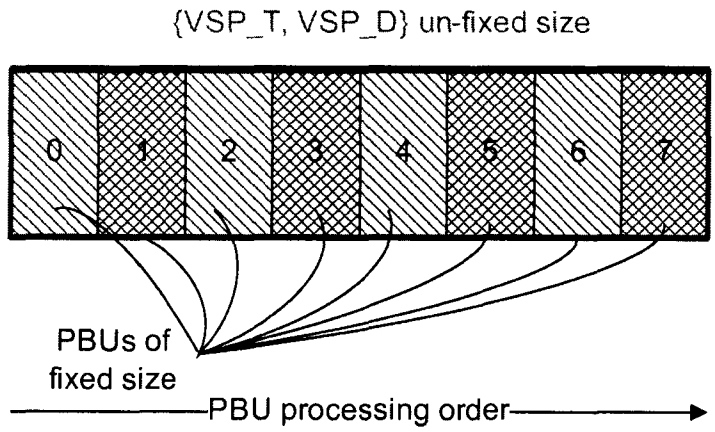
{VSP_T, VSP_D} un-fixed size
0
1
2
3
4
5
6
7
PBUs of fixed size
PBU processing order

Fig. 9

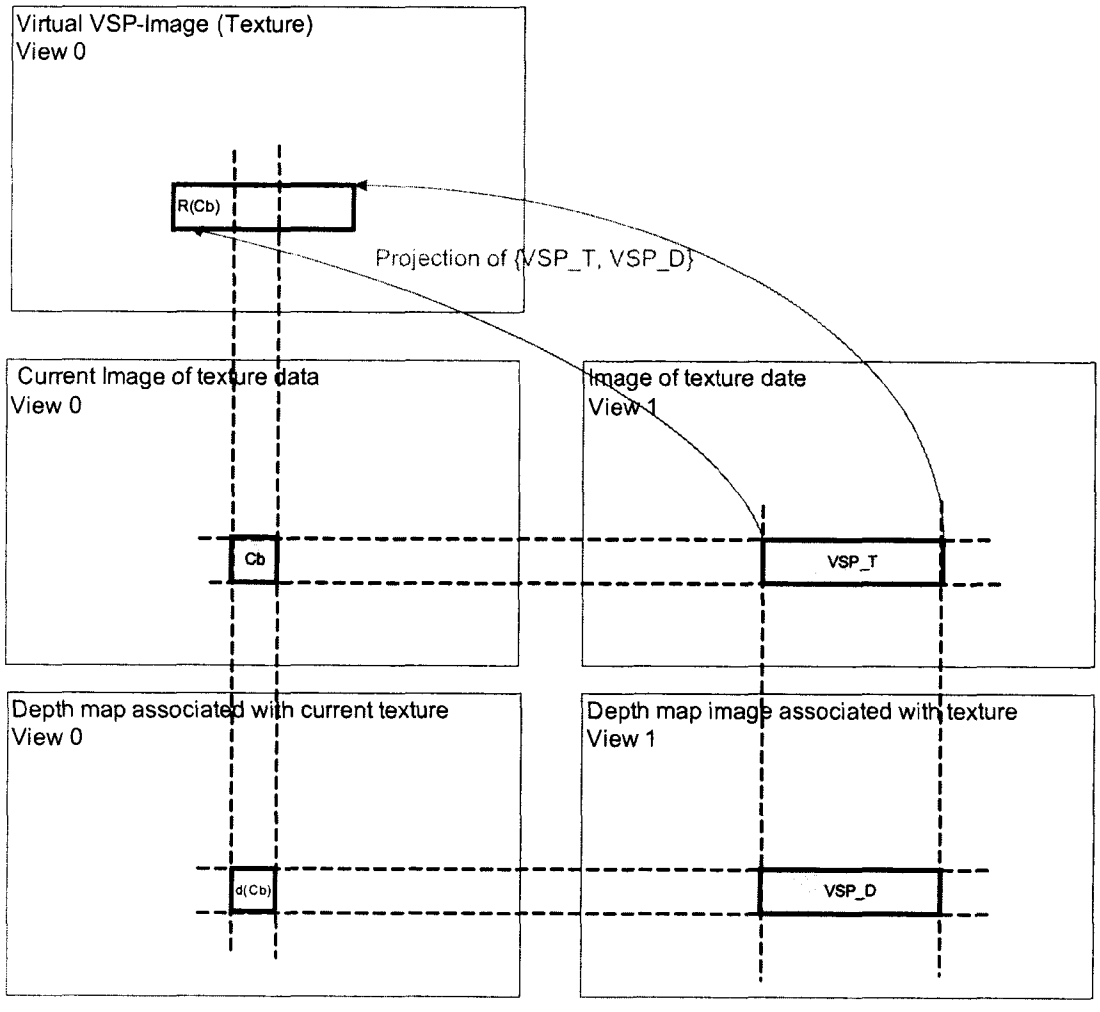
Virtual VSP-Image (Texture)
View 0
R(Cb)
Projection of {VSP_T, VSP_D}
Current Image of texture data
View 0
Image of texture date
View 1
Cb
VSP_T
Depth map associated with current texture
View 0
Depth map image associated with texture
View 1
d(Cb)
VSP_D

Fig. 10

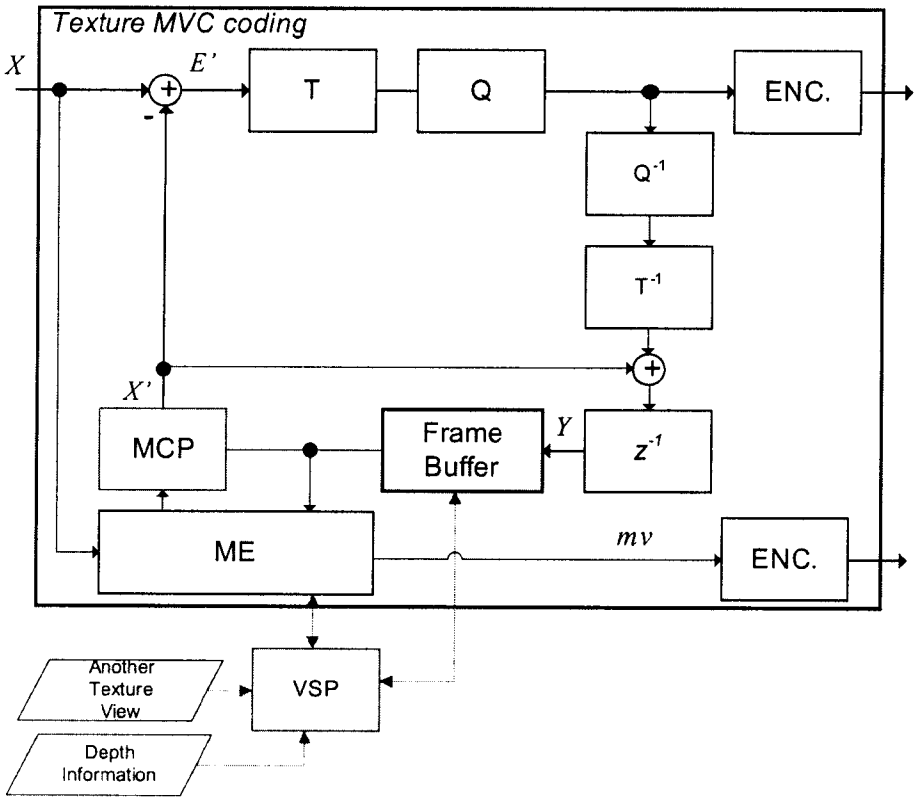
Texture MVC coding
X
E'
T
Q
ENC.
Q-1
T-1
X'
MCP
Frame Buffer
Y
z-1
ME
mv
ENC.
Another Texture View
VSP
Depth Information

Fig. 11

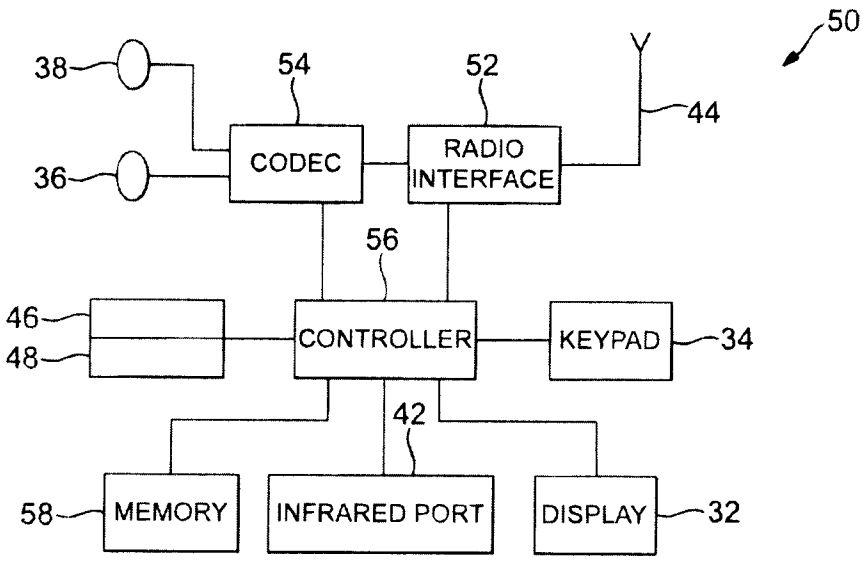
50
38
54
52
44
CODEC
RADIO INTERFACE
36
56
46
48
CONTROLLER
KEYPAD
34
42
58
MEMORY
INFRARED PORT
DISPLAY
32

Fig. 12

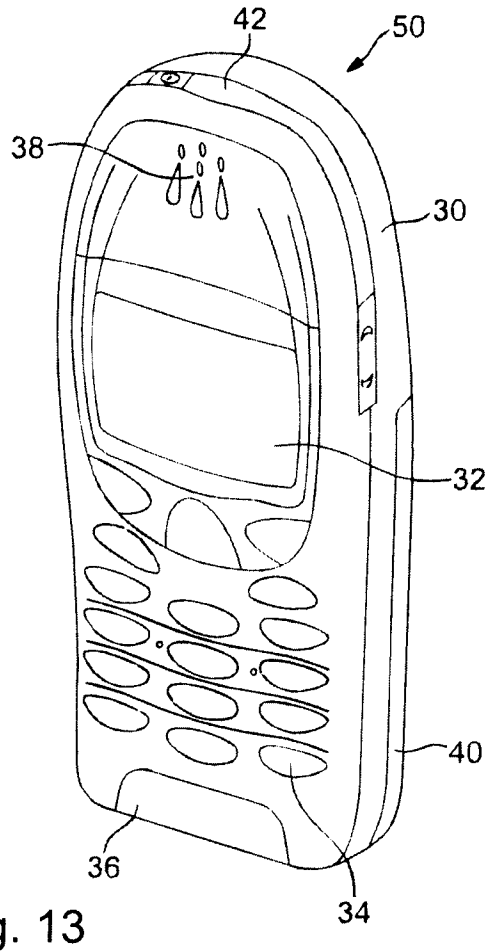
42
50
38
30
32
40
36
34

Fig. 13

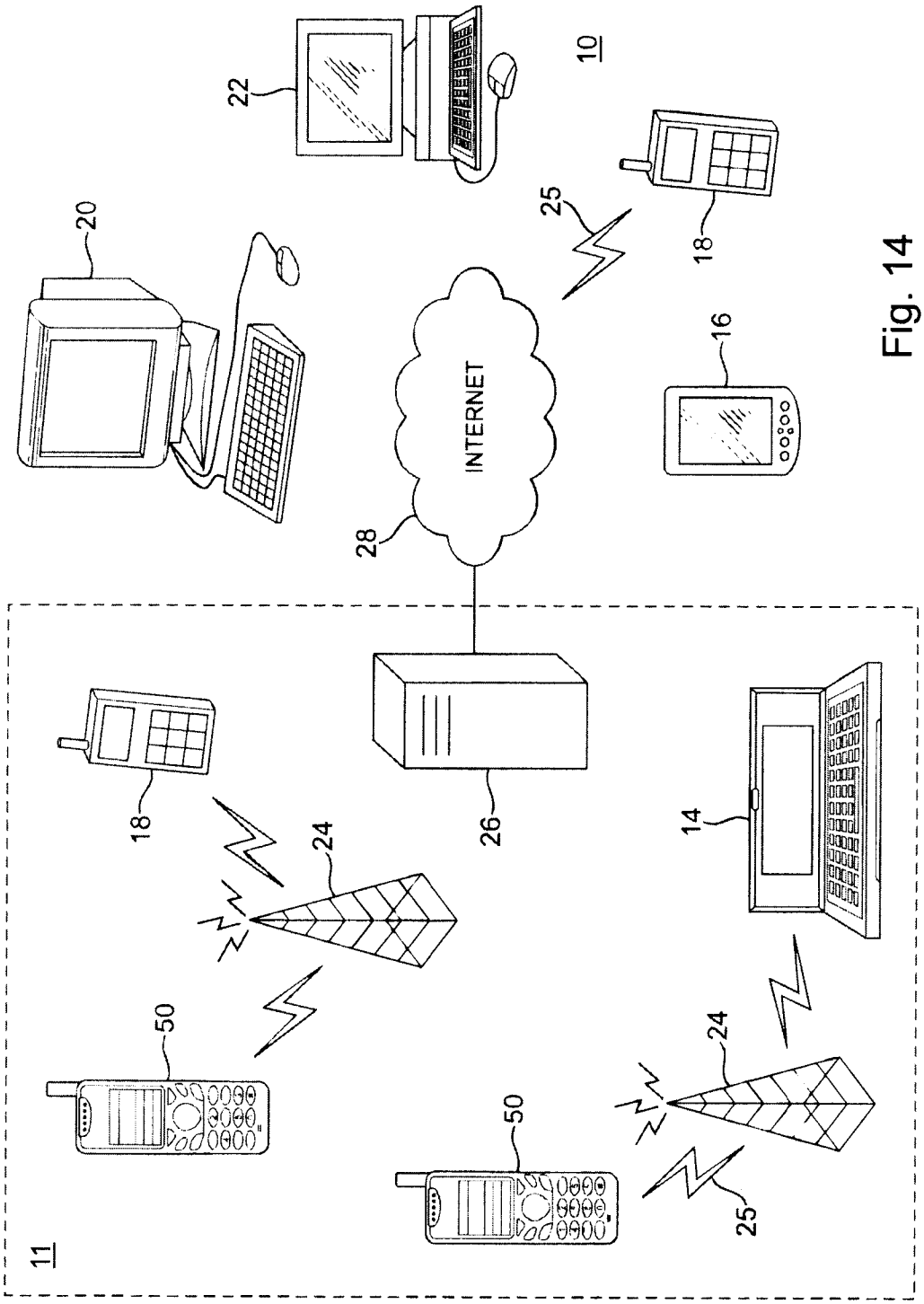

Fig. 14

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2012/074812

A. CLASSIFICATION OF SUBJECT MATTER

H04N7/32(2006.01)i

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; G06K

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

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

CNTXT, VEN, WOTXT, USTXT, EPTXT: view synthesis prediction, view interpolation prediction, texture, 3d, multi-view

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US2012027291A1 (Shimizu et al.) 02 Feb. 2012 (02.02.2012) the whole document	1-57
A	US2011286678A1 (Shimizu et al) 24 Nov. 2011 (24.11.2011) the whole document	1-57
A	US2008198924A1 (Ho et al.) 21 Aug. 2008 (21.08.2008) the whole document	1-57

☐ 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
14 Jan. 2013 (14.01.2013)	**14 Feb. 2013 (14.02.2013)**
Name and mailing address of the ISA/CN The State Intellectual Property Office, the P.R.China 6 Xitucheng Rd., Jimen Bridge, Haidian District, Beijing, China 100088 Facsimile No. 86-10-62019451	Authorized officer GE, Xiaolan Telephone No. (86-10)62411470

Form PCT/ISA /210 (second sheet) (July 2009)

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CN2012/074812

Patent Documents referred in the Report	Publication Date	Patent Family	Publication Date
US2012027291A1	02.02.2012	CN102326391A	18.01.2012
		IN201105890P4	21.12.2012
		WO2010095471A1	26.08.2010
		EP2400759A1	28.12.2011
		TW201103339A	16.01.2011
		JPWO2010095471SX	23.08.2012
		CA2752567A1	26.08.2010
		KR20110119709A	02.11.2011
US2011286678A1	24.11.2011	KR20110114614A	19.10.2011
		WO2010092772A1	19.08.2010
		EP2398241A1	21.12.2011
		CA2751297A1	19.08.2010
		TW201034469A	16.09.2010
		CN102308584A	04.01.2012
		JPWO2010092772SX	16.08.2012
US2008198924A1	21.08.2008	US8165201B2	24.04.2012
		KR100801968B1	12.02.2008