METHOD AND APPARATUS FOR POINT CLOUD CODING

[0001] This present application claims the benefit of priority to U.S. Patent Application No. 17/064,029, “METHOD AND APPARATUS FOR POINT CLOUD CODING” filed on October 6, 2020, which claims the benefit of priority to U.S. Provisional Application No. 62/942,536, "FAST RECOLOR FOR POINT CLOUD CODING" filed on December 2, 2019.

TECHNICAL FIELD

[0002] The present disclosure describes embodiments generally related to point cloud coding.

BACKGROUND

[0003] The background description provided herein is for the purpose of generally presenting the context of the disclosure. Work of the presently named inventors, to the extent the work is described in this background section, as well as aspects of the description that may not otherwise qualify as prior art at the time of filing, are neither expressly nor impliedly admitted as prior art against the present disclosure.

[0004] Various technologies are developed to capture and represent the world, such as objects in the world, environments in the world, and the like in 3-dimensional (3D) space. 3D representations of the world can enable more immersive forms of interaction and communication. Point clouds can be used as a 3D representation of the world. A point cloud is a set of points in a 3D space, each with associated attributes, e.g. color, material properties, texture information, intensity attributes, reflectivity attributes, motion related attributes, modality attributes, and various other attributes. Such point clouds may include large amounts of data and may be costly and time-consuming to store and transmit.

SUMMARY

[0005] Aspects of the disclosure provide methods and apparatuses for point cloud compression and decompression. In some examples, an apparatus for point cloud compression/decompression includes processing circuitry. In some embodiments, the processing circuitry determines one or more original points in a point cloud that are associated with a reconstructed position. Positions of the one or more original points can be reconstructed, according to a geometry quantization, to the reconstructed position. The processing circuitry then determines an attribute value for the reconstructed position based on attribute information of

the one or more original points, and encodes texture of the point cloud with the reconstructed position having the determined attribute value.

[0006] In some embodiments, the apparatus includes a memory storing a data structure that associates the one or more original points with the reconstructed position. The data structure is accessed based on the reconstructed position to retrieve the one or more original points associated with the reconstructed position.

[0007] In some embodiments, the processing circuitry performs an octree partition that partitions a space of the point cloud into voxels during geometry quantization, and associates the one or more original points that are positioned in a voxel with the reconstructed position for representing the voxel. In an example, the association relationship is stored using a suitable data structure.

[0008] In some embodiments, the processing circuitry calculates an average of attribute values of multiple original points as the determined attribute value for the reconstructed position in response to the multiple original points being associated with the reconstructed position. In some examples, the processing circuitry calculates a weighted average of the attribute values of the multiple original points as the determined attribute value for the reconstructed position. In an example, the processing circuitry weights an attribute value of an original point in the multiple original points based on an inverse of a distance between the original point and the reconstructed position.

[0009] In some embodiments, the processing circuitry assigns a specific attribute value of a nearest point in multiple original points to be the determined attribute value for the reconstructed point in response to the multiple original points being associated with the reconstructed position. In an example, the processing circuitry selects a median attribute value among attribute values for nearest points in response to the nearest points in the multiple original points having a same shortest distance to the reconstructed position. In another example, the processing circuitry calculates a mean attribute value of attribute values for nearest points in response to the nearest points in the multiple original points having a same shortest distance to the reconstructed position.

[0010] Aspects of the disclosure also provide a non-transitory computer-readable medium storing instructions which when executed by a computer for point cloud encoding/decoding cause the computer to perform any one or a combination of the methods for point cloud encoding/decoding.

BRIEF DESCRIPTION OF THE DRAWINGS

[0011] Further features, the nature, and various advantages of the disclosed subject matter will be more apparent from the following detailed description and the accompanying drawings in which:

[0012] FIG. 1 is a schematic illustration of a simplified block diagram of a communication system in accordance with an embodiment;

[0013] FIG. 2 is a schematic illustration of a simplified block diagram of a streaming system in accordance with an embodiment;

[0014] FIG. 3 shows a block diagram of an encoder for encoding point cloud frames, according to some embodiments;

[0015] FIG. 4 shows a block diagram of a decoder for decoding a compressed bitstream corresponding to point cloud frames according to some embodiments;

[0016] FIG. 5 is a schematic illustration of a simplified block diagram of a video decoder in accordance with an embodiment;

[0017] FIG. 6 is a schematic illustration of a simplified block diagram of a video encoder in accordance with an embodiment;

[0018] FIG. 7 shows a block diagram of an encoder for encoding point cloud frames, according to some embodiments;

[0019] FIG. 8 shows a block diagram of a decoder for decoding a compressed bitstream corresponding to point cloud frames according to some embodiments;

[0020] FIG. 9 shows a diagram illustrating a partition of a cube based on the octree partition technique according to some embodiments of the disclosure.

[0021] FIG. 10 shows an example of an octree partition and an octree structure corresponding to the octree partition according to some embodiments of the disclosure.

[0022] FIG. 11 shows a diagram illustrating derivation techniques according to some embodiment of the disclosure.

[0023] FIG. 12 shows a flow chart outlining a process example in accordance with some embodiments.

[0024] FIG. 13 is a schematic illustration of a computer system in accordance with an embodiment.

DETAILED DESCRIPTION OF EMBODIMENTS

[0025] Aspects of the disclosure provide point cloud coding (PCC) techniques. PCC can be performed according to various schemes, such as a geometry-based scheme that is referred to as G-PCC, a video coding based scheme that is referred to as V-PCC, and the like. According to some aspects of the disclosure, the G-PCC encodes the 3D geometry directly and is a purely geometry-based approach without much to share with video coding, and the V-PCC is heavily based on video coding. For example, V-PCC can map a point of the 3D cloud to a pixel of a 2D grid (an image). The V-PCC scheme can utilize generic video codecs for point cloud compression. Moving picture experts group (MPEG) is working on G-PCC standard and V-PCC standard that respectively using the G-PCC scheme and the V-PCC scheme.

[0026] Aspects of the disclosure provide recoloring techniques for a PCC scheme, such as the G-PCC scheme and the V-PCC scheme. The recoloring techniques can improve coding speed for point cloud.

[0027] Point Clouds can be widely used in many applications. For example, point clouds can be used in autonomous driving vehicles for object detection and localization; point clouds can be used in geographic information systems (GIS) for mapping, and can be used in cultural heritage to visualize and archive cultural heritage objects and collections, etc.

[0028] Hereinafter, a point cloud generally may refer to a set of points in a 3D space, each point can be defined by position information with associated attributes, e.g. color, material properties, texture information, intensity attributes, reflectivity attributes, motion related attributes, modality attributes, and various other attributes. Point clouds can be used to reconstruct an object or a scene as a composition of such points. The points can be captured using multiple cameras, depth sensors or Lidar in various setups and may be made up of thousands up to billions of points in order to realistically represent reconstructed scenes. A patch generally may refer to a contiguous subset of the surface described by the point cloud. In an example, a patch includes points with surface normal vectors that deviate from one another less than a threshold amount.

[0029] Compression technologies can reduce the amount of data required to represent a point cloud for faster transmission or reduction of storage. As such, technologies are needed for lossy compression of point clouds for use in real-time communications and six Degrees of Freedom (6 DoF) virtual reality. In addition, technology is sought for lossless point cloud compression in the context of dynamic mapping for autonomous driving and cultural heritage applications, and the like.

[0030] According to an aspect of the disclosure, the main philosophy behind V-PCC is to leverage existing video codecs to compress the geometry, occupancy, and texture of a dynamic point cloud as three separate video sequences. The extra metadata needed to interpret the three video sequences are compressed separately. A small portion of the overall bitstream is the metadata, which could be encoded/decoded efficiently using software implementation. The bulk of the information is handled by the video codec.

[0031] FIG. 1 illustrates a simplified block diagram of a communication system (100) according to an embodiment of the present disclosure. The communication system (100) includes a plurality of terminal devices that can communicate with each other, via, for example, a network (150). For example, the communication system (100) includes a pair of terminal devices (110) and (120) interconnected via the network (150). In the FIG. 1 example, the first pair of terminal devices (110) and (120) may perform unidirectional transmission of point cloud data. For example, the terminal device (110) may compress a point cloud (e.g., points representing a structure) that is captured by a sensor (105) connected with the terminal device (110). The compressed point cloud can be transmitted, for example in the form of a bitstream, to the other terminal device (120) via the network (150). The terminal device (120) may receive the compressed point cloud from the network (150), decompress the bitstream to reconstruct the point cloud, and suitably display the reconstructed point cloud. Unidirectional data transmission may be common in media serving applications and the like.

[0032] In the FIG. 1 example, the terminal devices (110) and (120) may be illustrated as servers, and personal computers, but the principles of the present disclosure may be not so limited. Embodiments of the present disclosure find application with laptop computers, tablet computers, smart phones, gaming terminals, media players, and/or dedicated three-dimensional (3D) equipment. The network (150) represents any number of networks that transmit compressed point cloud between the terminal devices (110) and (120). The network (150) can include for example wireline (wired) and/or wireless communication networks. The network (150) may exchange data in circuit-switched and/or packet-switched channels. Representative networks include telecommunications networks, local area networks, wide area networks, and/or the Internet. For the purposes of the present discussion, the architecture and topology of the network (150) may be immaterial to the operation of the present disclosure unless explained herein below.

[0033] FIG. 2 illustrates a simplified block diagram of a streaming system (200) in accordance with an embodiment. The FIG. 2 example is an application for the disclosed subject matter for a point cloud. The disclosed subject matter can be equally applicable to other point cloud enabled applications, such as, 3D telepresence application, virtual reality application, and the like.

[0034] The streaming system (200) may include a capture subsystem (213). The capture subsystem (213) can include a point cloud source (201), for example light detection and ranging (LIDAR) systems, 3D cameras, 3D scanners, a graphics generation component that generates the uncompressed point cloud in software, and the like that generates for example point clouds (202) that are uncompressed. In an example, the point clouds (202) include points that are captured by the 3D cameras. The point clouds (202), depicted as a bold line to emphasize a high data volume when compared to compressed point clouds (204) (a bitstream of compressed point clouds). The compressed point clouds (204) can be generated by an electronic device (220) that includes an encoder (203) coupled to the point cloud source (201). The encoder (203) can include hardware, software, or a combination thereof to enable or implement aspects of the disclosed subject matter as described in more detail below. The compressed point clouds (204) (or bitstream of compressed point clouds (204)), depicted as a thin line to emphasize the lower data volume when compared to the stream of point clouds (202), can be stored on a streaming server (205) for future use. One or more streaming client subsystems, such as client subsystems (206) and (208) in FIG. 2 can access the streaming server (205) to retrieve copies (207) and (209) of the compressed point cloud (204). A client subsystem (206) can include a decoder (210), for example, in an electronic device (230). The decoder (210) decodes the incoming copy (207) of the compressed point clouds and creates an outgoing stream of reconstructed point clouds (211) that can be rendered on a rendering device (212).

[0035] It is noted that the electronic devices (220) and (230) can include other components (not shown). For example, the electronic device (220) can include a decoder (not shown) and the electronic device (230) can include an encoder (not shown) as well.

[0036] In some streaming systems, the compressed point clouds (204), (207), and (209) (e.g., bitstreams of compressed point clouds) can be compressed according to certain standards. In some examples, video coding standards are used in the compression of point clouds. Examples of those standards include, High Efficiency Video Coding (HEVC), Versatile Video Coding (VVC), and the like.

[0037] FIG. 3 shows a block diagram of a V-PCC encoder (300) for encoding point cloud frames, according to some embodiments. In some embodiments, the V-PCC encoder (300) can be used in the communication system (100) and streaming system (200). For example, the encoder (203) can be configured and operate in a similar manner as the V-PCC encoder (300).

[0038] The V-PCC encoder (300) receives point cloud frames as uncompressed inputs and generates bitstream corresponding to compressed point cloud frames. In some embodiments, the V-PCC encoder (300) may receive the point cloud frames from a point cloud source, such as the point cloud source (201) and the like.

[0039] In the Fig. 3 example, the V-PCC encoder (300) includes a patch generation module (306), a patch packing module (308), a geometry image generation module (310), a texture image generation module (312), a patch info module (304), an occupancy map module (314), a smoothing module (336), image padding modules (316) and (318), a group dilation module (320), video compression modules (322), (323) and (332), an auxiliary patch info compression module (338), an entropy compression module (334), and a multiplexer (324).

[0040] According to an aspect of the disclosure, the V-PCC encoder (300), converts 3D point cloud frames into an image-based representation along with some meta data (e.g., occupancy map and patch info) that is used to convert the compressed point cloud back into a decompressed point cloud. In some examples, the V-PCC encoder (300) can convert 3D point cloud frames into geometry images, texture images and occupancy maps, and then use video coding techniques to encode the geometry images, texture images and occupancy maps into a bitstream. Generally, a geometry image is a 2D image with pixels filled with geometry values associated with points projected to the pixels, and a pixel filled with a geometry value can be referred to as a geometry sample. A texture image is a 2D image with pixels filled with texture values associated with points projected to the pixels, and a pixel filled with a texture value can be referred to as a texture sample. An occupancy map is a 2D image with pixels filled with values that indicate occupied or unoccupied by patches.

[0041] The patch generation module (306) segments a point cloud into a set of patches (e.g., a patch is defined as a contiguous subset of the surface described by the point cloud), which may be overlapping or not, such that each patch may be described by a depth field with respect to a plane in 2D space. In some embodiments, the patch generation module (306) aims at decomposing the point cloud into a minimum number of patches with smooth boundaries, while also minimizing the reconstruction error.

[0042] The patch info module (304) can collect the patch information that indicates sizes and shapes of the patches. In some examples, the patch information can be packed into an image frame and then encoded by the auxiliary patch info compression module (338) to generate the compressed auxiliary patch information.

[0043] The patch packing module (308) is configured to map the extracted patches onto a 2 dimensional (2D) grid while minimize the unused space and guarantee that every $M \times M$ (e.g., 16x16) block of the grid is associated with a unique patch. Efficient patch packing can directly impact the compression efficiency either by minimizing the unused space or ensuring temporal consistency.

[0044] The geometry image generation module (310) can generate 2D geometry images associated with geometry of the point cloud at given patch locations. The texture image generation module (312) can generate 2D texture images associated with texture of the point cloud at given patch locations. The geometry image generation module (310) and the texture image generation module (312) exploit the 3D to 2D mapping computed during the packing process to store the geometry and texture of the point cloud as images. In order to better handle the case of multiple points being projected to the same sample, each patch is projected onto two images, referred to as layers. In an example, geometry image is represented by a monochromatic frame of WxH in YUV420-8bit format. To generate the texture image, the texture generation procedure exploits the reconstructed/smoothed geometry in order to compute the colors to be associated with the re-sampled points.

[0045] The occupancy map module (314) can generate an occupancy map that describes padding information at each unit. For example, the occupancy image includes a binary map that indicates for each cell of the grid whether the cell belongs to the empty space or to the point cloud. In an example, the occupancy map uses binary information describing for each pixel whether the pixel is padded or not. In another example, the occupancy map uses binary information describing for each block of pixels whether the block of pixels is padded or not.

[0046] The occupancy map generated by the occupancy map module (314) can be compressed using lossless coding or lossy coding. When lossless coding is used, the entropy compression module (334) is used to compress the occupancy map. When lossy coding is used, the video compression module (332) is used to compress the occupancy map.

[0047] It is noted that the patch packing module (308) may leave some empty spaces between 2D patches packed in an image frame. The image padding modules (316) and (318) can

fill the empty spaces (referred to as padding) in order to generate an image frame that may be suited for 2D video and image codecs. The image padding is also referred to as background filling which can fill the unused space with redundant information. In some examples, a good background filling minimally increases the bit rate while does not introduce significant coding distortion around the patch boundaries.

[0048] The video compression modules (322), (323), and (332) can encode the 2D images, such as the padded geometry images, padded texture images, and occupancy maps based on a suitable video coding standard, such as HEVC, VVC and the like. In an example, the video compression modules (322), (323), and (332) are individual components that operate separately. It is noted that the video compression modules (322), (323), and (332) can be implemented as a single component in another example.

[0049] In some examples, the smoothing module (336) is configured to generate a smoothed image of the reconstructed geometry image. The smoothed image can be provided to the texture image generation (312). Then, the texture image generation (312) may adjust the generation of the texture image based on the reconstructed geometry images. For example, when a patch shape (e.g. geometry) is slightly distorted during encoding and decoding, the distortion may be taken into account when generating the texture images to correct for the distortion in patch shape.

[0050] In some embodiments, the group dilation (320) is configured to pad pixels around the object boundaries with redundant low-frequency content in order to improve coding gain as well as visual quality of reconstructed point cloud.

[0051] The multiplexer (324) can multiplex the compressed geometry image, the compressed texture image, the compressed occupancy map, the compressed auxiliary patch information into a compressed bitstream.

[0052] FIG. 4 shows a block diagram of a V-PCC decoder (400) for decoding compressed bitstream corresponding to point cloud frames, according to some embodiments. In some embodiments, the V-PCC decoder (400) can be used in the communication system (100) and streaming system (200). For example, the decoder (210) can be configured to operate in a similar manner as the V-PCC decoder (400). The V-PCC decoder (400) receives the compressed bitstream, and generates reconstructed point cloud based on the compressed bitstream.

[0053] In the FIG. 4 example, the V-PCC decoder (400) includes a de-multiplexer (432), video decompression modules (434) and (436), an occupancy map decompression module (438),

an auxiliary patch-information decompression module (442), a geometry reconstruction module (444), a smoothing module (446), a texture reconstruction module (448), and a color smoothing module (452).

[0054] The de-multiplexer (432) can receive and separate the compressed bitstream into compressed texture image, compressed geometry image, compressed occupancy map, and compressed auxiliary patch information.

[0055] The video decompression modules (434) and (436) can decode the compressed images according to a suitable standard (e.g., HEVC, VVC, etc.) and output decompressed images. For example, the video decompression module (434) decodes the compressed texture images and outputs decompressed texture images; and the video decompression module (436) decodes the compressed geometry images and outputs the decompressed geometry images.

[0056] The occupancy map decompression module (438) can decode the compressed occupancy maps according to a suitable standard (e.g., HEVC, VVC, etc.) and output decompressed occupancy maps.

[0057] The auxiliary patch-information decompression module (442) can decode the compressed auxiliary patch information according to a suitable standard (e.g., HEVC, VVC, etc.) and output decompressed auxiliary patch information.

[0058] The geometry reconstruction module (444) can receive the decompressed geometry images, and generate reconstructed point cloud geometry based on the decompressed occupancy map and decompressed auxiliary patch information.

[0059] The smoothing module (446) can smooth incongruences at edges of patches. The smoothing procedure aims at alleviating potential discontinuities that may arise at the patch boundaries due to compression artifacts. In some embodiments, a smoothing filter may be applied to the pixels located on the patch boundaries to alleviate the distortions that may be caused by the compression/decompression.

[0060] The texture reconstruction module (448) can determine texture information for points in the point cloud based on the decompressed texture images and the smoothing geometry.

[0061] The color smoothing module (452) can smooth incongruences of coloring. Non-neighboring patches in 3D space are often packed next to each other in 2D videos. In some examples, pixel values from non-neighboring patches might be mixed up by the block-based video codec. The goal of color smoothing is to reduce the visible artifacts that appear at patch boundaries.

[0062] FIG. 5 shows a block diagram of a video decoder (510) according to an embodiment of the present disclosure. The video decoder (510) can be used in the V-PCC decoder (400). For example, the video decompression modules (434) and (436), the occupancy map decompression module (438) can be similarly configured as the video decoder (510).

[0063] The video decoder (510) may include a parser (520) to reconstruct symbols (521) from compressed images, such as the coded video sequence. Categories of those symbols include information used to manage operation of the video decoder (510). The parser (520) may parse / entropy-decode the coded video sequence that is received. The coding of the coded video sequence can be in accordance with a video coding technology or standard, and can follow various principles, including variable length coding, Huffman coding, arithmetic coding with or without context sensitivity, and so forth. The parser (520) may extract from the coded video sequence, a set of subgroup parameters for at least one of the subgroups of pixels in the video decoder, based upon at least one parameter corresponding to the group. Subgroups can include Groups of Pictures (GOPs), pictures, tiles, slices, macroblocks, Coding Units (CUs), blocks, Transform Units (TUs), Prediction Units (PUs) and so forth. The parser (520) may also extract from the coded video sequence information such as transform coefficients, quantizer parameter values, motion vectors, and so forth.

[0064] The parser (520) may perform an entropy decoding / parsing operation on the video sequence received from a buffer memory, so as to create symbols (521).

[0065] Reconstruction of the symbols (521) can involve multiple different units depending on the type of the coded video picture or parts thereof (such as: inter and intra picture, inter and intra block), and other factors. Which units are involved, and how, can be controlled by the subgroup control information that was parsed from the coded video sequence by the parser (520). The flow of such subgroup control information between the parser (520) and the multiple units below is not depicted for clarity.

[0066] Beyond the functional blocks already mentioned, the video decoder (510) can be conceptually subdivided into a number of functional units as described below. In a practical implementation operating under commercial constraints, many of these units interact closely with each other and can, at least partly, be integrated into each other. However, for the purpose of describing the disclosed subject matter, the conceptual subdivision into the functional units below is appropriate.

[0067] A first unit is the scaler / inverse transform unit (551). The scaler / inverse transform unit (551) receives a quantized transform coefficient as well as control information, including which transform to use, block size, quantization factor, quantization scaling matrices, etc. as symbol(s) (521) from the parser (520). The scaler / inverse transform unit (551) can output blocks comprising sample values that can be input into aggregator (555).

[0068] In some cases, the output samples of the scaler / inverse transform (551) can pertain to an intra coded block; that is: a block that is not using predictive information from previously reconstructed pictures, but can use predictive information from previously reconstructed parts of the current picture. Such predictive information can be provided by an intra picture prediction unit (552). In some cases, the intra picture prediction unit (552) generates a block of the same size and shape of the block under reconstruction, using surrounding already reconstructed information fetched from the current picture buffer (558). The current picture buffer (558) buffers, for example, partly reconstructed current picture and/or fully reconstructed current picture. The aggregator (555), in some cases, adds, on a per sample basis, the prediction information the intra prediction unit (552) has generated to the output sample information as provided by the scaler / inverse transform unit (551).

[0069] In other cases, the output samples of the scaler / inverse transform unit (551) can pertain to an inter coded, and potentially motion compensated block. In such a case, a motion compensation prediction unit (553) can access reference picture memory (557) to fetch samples used for prediction. After motion compensating the fetched samples in accordance with the symbols (521) pertaining to the block, these samples can be added by the aggregator (555) to the output of the scaler / inverse transform unit (551) (in this case called the residual samples or residual signal) so as to generate output sample information. The addresses within the reference picture memory (557) from where the motion compensation prediction unit (553) fetches prediction samples can be controlled by motion vectors, available to the motion compensation prediction unit (553) in the form of symbols (521) that can have, for example X, Y, and reference picture components. Motion compensation also can include interpolation of sample values as fetched from the reference picture memory (557) when sub-sample exact motion vectors are in use, motion vector prediction mechanisms, and so forth.

[0070] The output samples of the aggregator (555) can be subject to various loop filtering techniques in the loop filter unit (556). Video compression technologies can include in-loop filter technologies that are controlled by parameters included in the coded video sequence (also

referred to as coded video bitstream) and made available to the loop filter unit (556) as symbols (521) from the parser (520), but can also be responsive to meta-information obtained during the decoding of previous (in decoding order) parts of the coded picture or coded video sequence, as well as responsive to previously reconstructed and loop-filtered sample values.

[0071] The output of the loop filter unit (556) can be a sample stream that can be output to a render device as well as stored in the reference picture memory (557) for use in future inter-picture prediction.

[0072] Certain coded pictures, once fully reconstructed, can be used as reference pictures for future prediction. For example, once a coded picture corresponding to a current picture is fully reconstructed and the coded picture has been identified as a reference picture (by, for example, the parser (520)), the current picture buffer (558) can become a part of the reference picture memory (557), and a fresh current picture buffer can be reallocated before commencing the reconstruction of the following coded picture.

[0073] The video decoder (510) may perform decoding operations according to a predetermined video compression technology in a standard, such as ITU-T Rec. H.265. The coded video sequence may conform to a syntax specified by the video compression technology or standard being used, in the sense that the coded video sequence adheres to both the syntax of the video compression technology or standard and the profiles as documented in the video compression technology or standard. Specifically, a profile can select certain tools as the only tools available for use under that profile from all the tools available in the video compression technology or standard. Also necessary for compliance can be that the complexity of the coded video sequence is within bounds as defined by the level of the video compression technology or standard. In some cases, levels restrict the maximum picture size, maximum frame rate, maximum reconstruction sample rate (measured in, for example megasamples per second), maximum reference picture size, and so on. Limits set by levels can, in some cases, be further restricted through Hypothetical Reference Decoder (HRD) specifications and metadata for HRD buffer management signaled in the coded video sequence.

[0074] FIG. 6 shows a block diagram of a video encoder (603) according to an embodiment of the present disclosure. The video encoder (603) can be used in the V-PCC encoder (300) the compresses point clouds. In an example, the video compression module (322) and (323), and the video compression module (332) are configured similarly to the encoder (603).

[0075] The video encoder (603) may receive images, such as padded geometry images, padded texture images and the like, and generate compressed images.

[0076] According to an embodiment, the video encoder (603) may code and compress the pictures of the source video sequence (images) into a coded video sequence (compressed images) in real time or under any other time constraints as required by the application. Enforcing appropriate coding speed is one function of a controller (650). In some embodiments, the controller (650) controls other functional units as described below and is functionally coupled to the other functional units. The coupling is not depicted for clarity. Parameters set by the controller (650) can include rate control related parameters (picture skip, quantizer, lambda value of rate-distortion optimization techniques, …), picture size, group of pictures (GOP) layout, maximum motion vector search range, and so forth. The controller (650) can be configured to have other suitable functions that pertain to the video encoder (603) optimized for a certain system design.

[0077] In some embodiments, the video encoder (603) is configured to operate in a coding loop. As an oversimplified description, in an example, the coding loop can include a source coder (630) (e.g., responsible for creating symbols, such as a symbol stream, based on an input picture to be coded, and a reference picture(s)), and a (local) decoder (633) embedded in the video encoder (603). The decoder (633) reconstructs the symbols to create the sample data in a similar manner as a (remote) decoder also would create (as any compression between symbols and coded video bitstream is lossless in the video compression technologies considered in the disclosed subject matter). The reconstructed sample stream (sample data) is input to the reference picture memory (634). As the decoding of a symbol stream leads to bit-exact results independent of decoder location (local or remote), the content in the reference picture memory (634) is also bit exact between the local encoder and remote encoder. In other words, the prediction part of an encoder "sees" as reference picture samples exactly the same sample values as a decoder would "see" when using prediction during decoding. This fundamental principle of reference picture synchronicity (and resulting drift, if synchronicity cannot be maintained, for example because of channel errors) is used in some related arts as well.

[0078] The operation of the "local" decoder (633) can be the same as of a "remote" decoder, such as the video decoder (510), which has already been described in detail above in conjunction with FIG. 5. Briefly referring also to FIG. 5, however, as symbols are available and encoding/decoding of symbols to a coded video sequence by an entropy coder (645) and the

parser (520) can be lossless, the entropy decoding parts of the video decoder (510), including parser (520) may not be fully implemented in the local decoder (633).

[0079] An observation that can be made at this point is that any decoder technology except the parsing/entropy decoding that is present in a decoder also necessarily needs to be present, in substantially identical functional form, in a corresponding encoder. For this reason, the disclosed subject matter focuses on decoder operation. The description of encoder technologies can be abbreviated as they are the inverse of the comprehensively described decoder technologies. Only in certain areas a more detail description is required and provided below.

[0080] During operation, in some examples, the source coder (630) may perform motion compensated predictive coding, which codes an input picture predictively with reference to one or more previously-coded picture from the video sequence that were designated as "reference pictures". In this manner, the coding engine (632) codes differences between pixel blocks of an input picture and pixel blocks of reference picture(s) that may be selected as prediction reference(s) to the input picture.

[0081] The local video decoder (633) may decode coded video data of pictures that may be designated as reference pictures, based on symbols created by the source coder (630). Operations of the coding engine (632) may advantageously be lossy processes. When the coded video data may be decoded at a video decoder (not shown in FIG. 6), the reconstructed video sequence typically may be a replica of the source video sequence with some errors. The local video decoder (633) replicates decoding processes that may be performed by the video decoder on reference pictures and may cause reconstructed reference pictures to be stored in the reference picture cache (634). In this manner, the video encoder (603) may store copies of reconstructed reference pictures locally that have common content as the reconstructed reference pictures that will be obtained by a far-end video decoder (absent transmission errors).

[0082] The predictor (635) may perform prediction searches for the coding engine (632). That is, for a new picture to be coded, the predictor (635) may search the reference picture memory (634) for sample data (as candidate reference pixel blocks) or certain metadata such as reference picture motion vectors, block shapes, and so on, that may serve as an appropriate prediction reference for the new pictures. The predictor (635) may operate on a sample block-by-pixel block basis to find appropriate prediction references. In some cases, as determined by

search results obtained by the predictor (635), an input picture may have prediction references drawn from multiple reference pictures stored in the reference picture memory (634).

[0083] The controller (650) may manage coding operations of the source coder (630), including, for example, setting of parameters and subgroup parameters used for encoding the video data.

[0084] Output of all aforementioned functional units may be subjected to entropy coding in the entropy coder (645). The entropy coder (645) translates the symbols as generated by the various functional units into a coded video sequence, by lossless compressing the symbols according to technologies such as Huffman coding, variable length coding, arithmetic coding, and so forth.

[0085] The controller (650) may manage operation of the video encoder (603). During coding, the controller (650) may assign to each coded picture a certain coded picture type, which may affect the coding techniques that may be applied to the respective picture. For example, pictures often may be assigned as one of the following picture types:

[0086] An Intra Picture (I picture) may be one that may be coded and decoded without using any other picture in the sequence as a source of prediction. Some video codecs allow for different types of intra pictures, including, for example Independent Decoder Refresh ("IDR") Pictures. A person skilled in the art is aware of those variants of I pictures and their respective applications and features.

[0087] A predictive picture (P picture) may be one that may be coded and decoded using intra prediction or inter prediction using at most one motion vector and reference index to predict the sample values of each block.

[0088] A bi-directionally predictive picture (B Picture) may be one that may be coded and decoded using intra prediction or inter prediction using at most two motion vectors and reference indices to predict the sample values of each block. Similarly, multiple-predictive pictures can use more than two reference pictures and associated metadata for the reconstruction of a single block.

[0089] Source pictures commonly may be subdivided spatially into a plurality of sample blocks (for example, blocks of 4x4, 8x8, 4x8, or 16x16 samples each) and coded on a block-by-block basis. Blocks may be coded predictively with reference to other (already coded) blocks as determined by the coding assignment applied to the blocks' respective pictures. For example, blocks of I pictures may be coded non-predictively or they may be coded predictively with

reference to already coded blocks of the same picture (spatial prediction or intra prediction). Pixel blocks of P pictures may be coded predictively, via spatial prediction or via temporal prediction with reference to one previously coded reference picture. Blocks of B pictures may be coded predictively, via spatial prediction or via temporal prediction with reference to one or two previously coded reference pictures.

[0090] The video encoder (603) may perform coding operations according to a predetermined video coding technology or standard, such as ITU-T Rec. H.265. In its operation, the video encoder (603) may perform various compression operations, including predictive coding operations that exploit temporal and spatial redundancies in the input video sequence. The coded video data, therefore, may conform to a syntax specified by the video coding technology or standard being used.

[0091] A video may be in the form of a plurality of source pictures (images) in a temporal sequence. Intra-picture prediction (often abbreviated to intra prediction) makes use of spatial correlation in a given picture, and inter-picture prediction makes uses of the (temporal or other) correlation between the pictures. In an example, a specific picture under encoding/decoding, which is referred to as a current picture, is partitioned into blocks. When a block in the current picture is similar to a reference block in a previously coded and still buffered reference picture in the video, the block in the current picture can be coded by a vector that is referred to as a motion vector. The motion vector points to the reference block in the reference picture, and can have a third dimension identifying the reference picture, in case multiple reference pictures are in use.

[0092] In some embodiments, a bi-prediction technique can be used in the inter-picture prediction. According to the bi-prediction technique, two reference pictures, such as a first reference picture and a second reference picture that are both prior in decoding order to the current picture in the video (but may be in the past and future, respectively, in display order) are used. A block in the current picture can be coded by a first motion vector that points to a first reference block in the first reference picture, and a second motion vector that points to a second reference block in the second reference picture. The block can be predicted by a combination of the first reference block and the second reference block.

[0093] Further, a merge mode technique can be used in the inter-picture prediction to improve coding efficiency.

[0094] According to some embodiments of the disclosure, predictions, such as inter-picture predictions and intra-picture predictions are performed in the unit of blocks. For example, according to the HEVC standard, a picture in a sequence of video pictures is partitioned into coding tree units (CTU) for compression, the CTUs in a picture have the same size, such as 64x64 pixels, 32x32 pixels, or 16x16 pixels. In general, a CTU includes three coding tree blocks (CTBs), which are one luma CTB and two chroma CTBs. Each CTU can be recursively quadtree split into one or multiple coding units (CUs). For example, a CTU of 64x64 pixels can be split into one CU of 64x64 pixels, or 4 CUs of 32x32 pixels, or 16 CUs of 16x16 pixels. In an example, each CU is analyzed to determine a prediction type for the CU, such as an inter prediction type or an intra prediction type. The CU is split into one or more prediction units (PUs) depending on the temporal and/or spatial predictability. Generally, each PU includes a luma prediction block (PB), and two chroma PBs. In an embodiment, a prediction operation in coding (encoding/decoding) is performed in the unit of a prediction block. Using a luma prediction block as an example of a prediction block, the prediction block includes a matrix of values (e.g., luma values) for pixels, such as 8x8 pixels, 16x16 pixels, 8x16 pixels, 16x8 pixels, and the like.

[0095] FIG. 7 shows a block diagram of a G-PPC encoder (700) in accordance with an embodiment. The encoder (700) can be configured to receive point cloud data and compress the point cloud data to generate a bit stream carrying compressed point cloud data. In an embodiment, the encoder (700) can include a position quantization module (710), a duplicated points removal module (712), an octree encoding module (730), an attribute transfer module (720), a level of detail (LOD) generation module (740), an attribute prediction module (750), a residual quantization module (760), an arithmetic coding module (770), an inverse residual quantization module (780), an addition module (781), and a memory (790) to store reconstructed attribute values.

[0096] As shown, an input point cloud (701) can be received at the encoder (700). Positions (e.g., 3D coordinates) of the point cloud (701) are provided to the quantization module (710). The quantization module (710) is configured to quantize the coordinates to generate quantized positions. The duplicated points removal module (712) is configured to receive the quantized positions and perform a filter process to identify and remove duplicated points. The octree encoding module (730) is configured to receive filtered positions from the duplicated points removal module (712), and perform an octree-based encoding process to generate a

sequence of occupancy codes that describe a 3D grid of voxels. The occupancy codes are provided to the arithmetic coding module (770).

[0097] The attribute transfer module (720) is configured to receive attributes of the input point cloud, and perform an attribute transfer process to determine an attribute value for each voxel when multiple attribute values are associated with the respective voxel. The attribute transfer process can be performed on the re-ordered points output from the octree encoding module (730). The attributes after the transfer operations are provided to the attribute prediction module (750). The LOD generation module (740) is configured to operate on the re-ordered points output from the octree encoding module (730), and re-organize the points into different LODs. LOD information is supplied to the attribute prediction module (750).

[0098] The attribute prediction module (750) processes the points according to an LOD-based order indicated by the LOD information from the LOD generation module (740). The attribute prediction module (750) generates an attribute prediction for a current point based on reconstructed attributes of a set of neighboring points of the current point stored in the memory (790). Prediction residuals can subsequently be obtained based on original attribute values received from the attribute transfer module (720) and locally generated attribute predictions. When candidate indices are used in the respective attribute prediction process, an index corresponding to a selected prediction candidate may be provided to the arithmetic coding module (770).

[0099] The residual quantization module (760) is configured to receive the prediction residuals from the attribute prediction module (750), and perform quantization to generate quantized residuals. The quantized residuals are provided to the arithmetic coding module (770).

[0100] The inverse residual quantization module (780) is configured to receive the quantized residuals from the residual quantization module (760), and generate reconstructed prediction residuals by performing an inverse of the quantization operations performed at the residual quantization module (760). The addition module (781) is configured to receive the reconstructed prediction residuals from the inverse residual quantization module (780), and the respective attribute predictions from the attribute prediction module (750). By combining the reconstructed prediction residuals and the attribute predictions, the reconstructed attribute values are generated and stored to the memory (790).

[0101] The arithmetic coding module (770) is configured to receive the occupancy codes, the candidate indices (if used), the quantized residuals (if generated), and other information, and

perform entropy encoding to further compress the received values or information. As a result, a compressed bitstream (702) carrying the compressed information can be generated. The bitstream (702) may be transmitted, or otherwise provided, to a decoder that decodes the compressed bitstream, or may be stored in a storage device.

[0102] FIG. 8 shows a block diagram of a G-PCC decoder (800) in accordance with an embodiment. The decoder (800) can be configured to receive a compressed bitstream and perform point cloud data decompression to decompress the bitstream to generate decoded point cloud data. In an embodiment, the decoder (800) can include an arithmetic decoding module (810), an inverse residual quantization module (820), an octree decoding module (830), an LOD generation module (840), an attribute prediction module (850), and a memory (860) to store reconstructed attribute values.

[0103] As shown, a compressed bitstream (801) can be received at the arithmetic decoding module (810). The arithmetic decoding module (810) is configured to decode the compressed bitstream (801) to obtain quantized residuals (if generated) and occupancy codes of a point cloud. The octree decoding module (830) is configured to determine reconstructed positions of points in the point cloud according to the occupancy codes. The LOD generation module (840) is configured to re-organize the points into different LODs based on the reconstructed positions, and determine an LOD-based order. The inverse residual quantization module (820) is configured to generate reconstructed residuals based on the quantized residuals received from the arithmetic decoding module (810).

[0104] The attribute prediction module (850) is configured to perform an attribute prediction process to determine attribute predictions for the points according to the LOD-based order. For example, an attribute prediction of a current point can be determined based on reconstructed attribute values of neighboring points of the current point stored in the memory (860). The attribute prediction module (850) can combine the attribute prediction with a respective reconstructed residual to generate a reconstructed attribute for the current point.

[0105] A sequence of reconstructed attributes generated from the attribute prediction module (850) together with the reconstructed positions generated from the octree decoding module (830) corresponds to a decoded point cloud (802) that is output from the decoder (800) in one example. In addition, the reconstructed attributes are also stored into the memory (860) and can be subsequently used for deriving attribute predictions for subsequent points.

[0106] In various embodiments, the encoder (300), the decoder (400), the encoder (700), and/or the decoder (800) can be implemented with hardware, software, or combination thereof. For example, the encoder (300), the decoder (400), the encoder (700), and/or the decoder (800) can be implemented with processing circuitry such as one or more integrated circuits (ICs) that operate with or without software, such as an application specific integrated circuit (ASIC), field programmable gate array (FPGA), and the like. In another example, the encoder (300), the decoder (400), the encoder (700), and/or the decoder (800) can be implemented as software or firmware including instructions stored in a non-volatile (or non-transitory) computer-readable storage medium. The instructions, when executed by processing circuitry, such as one or more processors, cause the processing circuitry to perform functions of the encoder (300), the decoder (400), the encoder (700), and/or the decoder (800).

[0107] It is noted that the attribute prediction modules (750) and (850) configured to implement the attribute prediction techniques disclosed herein can be included in other decoders or encoders that may have similar or different structures from what is shown in FIG. 7 and FIG. 8. In addition, the encoder (700) and decoder (800) can be included in a same device, or separate devices in various examples.

[0108] According to some aspects of the disclosure, in some related examples (e.g., current version of the TMC13 model), a recoloring process is applied at the encoder-side when the geometry is quantized and the duplicated positions are merged. The recolor process is relatively complex. For example, the recolor process relies on a k-d tree (e.g., 3 dimensional tree) data structure to search for nearest neighbor(s). The searches based on the k-d tree can be complex and time consuming.

[0109] In a related example (e.g., a version of TMC13 model), the inputs to the recoloring algorithm can include the original positions of the original points in the point cloud, attributes associated with the original positions and reconstructed positions (e.g., results of the geometry quantization and merging of duplicated positions). The recoloring algorithm can transfer the attributes associated with the original positions to attributes associated with the reconstructed positions, for example with minimized attribute distortions. The attributes associated with the reconstructed positions can form reconstructed points in a reconstructed point cloud. It is noted that the recoloring algorithm can transfer any suitable attribute, such as color, material properties, texture information, intensity attributes, reflectivity attributes, motion related

attributes, modality attributes, and various other attributes. In an example, the recoloring algorithm includes five steps that are described in detail in the following description.

[0110] In a first step, the original positions of the points in the point cloud, attributes associated with the original positions and the reconstructed positions are received. In an example, $(X_i)_{i=0\ldots N-1}$ denote the original positions of the original points in the original point cloud, N denotes the number of points in the original point cloud, $(\tilde{X}_i)_{i=0\ldots N_{rec}-1}$ denote the reconstructed positions, and N_{rec} denotes the number of reconstructed positions. If duplicated points exist and are merged, then $N_{rec} < N$, otherwise $N_{rec} = N$.

[0111] In a second step, for each position $\tilde{X}_i$ in the reconstructed point cloud, a search process is performed, for example, based on a k-d tree search, to determine X_i^* that is a nearest neighbour in the original point cloud and a_i^* denotes the attribute value associated with the nearest neighbour in the original point cloud.

[0112] In a third step, for each position $\tilde{X}_i$ in the reconstructed point cloud, search processes are performed, for example, based on a k-d tree search, to determine a set of original positions in the original point cloud that is denoted by $\mathbb{Q}^+(i) = (X_i^+(h))_{h\in\{1,\ldots,H(i)\}}$, the set of original positions share $\tilde{X}_i$ as their nearest neighbour in the reconstructed point cloud. $H(i)$ denotes the number of elements in $\mathbb{Q}^+(i)$, and $X_i^+(h)$ denotes one of the elements of $\mathbb{Q}^+(i)$. It is noted that $\mathbb{Q}^+(i)$ could be empty or could have one or multiple elements.

[0113] In a fourth step, $\mathbb{Q}^+(i)$ is checked. When $\mathbb{Q}^+(i)$ is empty, the attribute value $\tilde{a}_i$ associated with the position $\tilde{X}_i$ is set to a_i^*.

[0114] In a fifth step, if $\mathbb{Q}^+(i)$ is not empty, the attribute value $\tilde{a}_i$ associated with the position $\tilde{X}_i$ can be calculated based on, for example (Eq. 1):

$$\tilde{a}_i = \frac{1}{H(i)}\sum_{h=1}^{H(i)} a_i^+(h) \qquad \text{(Eq. 1)}$$

[0115] In the related example, searches, such as k-d tree (e.g., 3 dimensional tree) searches are used to search for nearest neighbor(s). The searches based on the k-d tree can be complex and time consuming. The present disclosure provides recoloring techniques to reduce complexity and improve processing speed. For example, the searches of the nearest neighbors can be determined based on information stored during geometry quantization and merging of duplicated points.

[0116] The proposed methods may be used separately or combined in any order. Further, each of the methods (or embodiments), encoder, and decoder may be implemented by processing

circuitry (e.g., one or more processors or one or more integrated circuits). In one example, the one or more processors execute a program that is stored in a non-transitory computer-readable medium.

[0117] According to some aspects of the disclosure, geometry information and the associated attributes of a point cloud, such as color, reflectance and the like can be separately compressed (e.g., in the Test Model 13 (TMC13) model). In some embodiments, the geometry information of the point cloud, which includes the 3D coordinates of the points in the point cloud, can be processed according to quantization and merge of duplication process. In some examples, the quantization and merge of duplication process can based on octree partition, and the geometry information can be coded by an octree partition with occupancy information of the partitions. The attributes can be compressed based on a reconstructed geometry using, for example, prediction, lifting and region adaptive hierarchical transform techniques. For example, the coded geometry information can be dequantized to reconstruct the geometry information for the point cloud, such as reconstructed positions. The attributes of the original points are transferred to the reconstructed positions and then encoded.

[0118] According to some aspects of the disclosure, a three dimensional space can be partitioned using octree partition. Octrees are the three dimensional analog of quadtrees in the two dimensional space. Octree partition technique refers to the partition technique that recursively subdivides three dimensional space into eight octants, and an octree structure refers to the tree structure that represents the partitions. In an example, each node in the octree structure corresponds to a three dimensional space, and the node can be an end node (no more partition, also referred to as leaf node in some examples) or a node with a further partition. A partition at a node can partition the three dimensional space represented by the node into eight octants. In some examples, nodes corresponding to partitions of a specific node can be referred to as child nodes of the specific node.

[0119] FIG. 9 shows a diagram illustrating a partition of a 3D cube (900) (corresponding to a node) based on the octree partition technique according to some embodiments of the disclosure. The partition can divide the 3D cube (900) into eight smaller equal-sized cubes 0-7 as shown in FIG. 9.

[0120] The octree partition technique (e.g., in TMC13) can recursively divide an original 3D space into the smaller units, and the occupancy information of every sub-space can be encoded to represent geometry positions.

[0121] In some embodiments (e.g., In TMC13), an octree geometry codec is used. The octree geometry codec can perform geometry encoding. In some examples, geometry encoding is performed on a cubical box. For example, the cubical box can be an axis-aligned bounding box B that is defined by two points (0,0,0) and $(2^{M-1}, 2^{M-1}, 2^{M-1})$, where 2^{M-1} defines the size of the bounding box B and M can be specified in the bitstream.

[0122] Then, an octree structure is built by recursively subdividing the cubical box. For example, the cubical box defined by the two points (0,0,0) and $(2^{M-1}, 2^{M-1}, 2^{M-1})$ is divided into 8 sub cubical boxes, then an 8-bit code, that is referred to as an occupancy code, is generated. Each bit of the occupancy code is associated with a sub cubical box, and the value of the bit is used to indicate whether the associated sub cubical box contains any points of the point cloud. For example, value 1 of a bit indicates that the sub cubical box associated with the bit contains one or more points of the point cloud; and value 0 of a bit indicates that the sub cubical box associated with the bit contains no point of the point cloud.

[0123] Further, for empty sub cubical box (e.g., the value of the bit associated with the sub cubical box is 0), no more division is applied on the sub cubical box. When a sub cubical box has one or more points of the point cloud (e.g., the value of the bit associated with the sub cubical box is 1), the sub cubical box is further divided into 8 smaller sub cubical boxes, and an occupancy code can be generated for the sub cubical box to indicate the occupancy of the smaller sub cubical boxes. In some examples, the subdivision operations can be repetitively performed on non-empty sub cubical boxes until the size of the sub cubical boxes is equal to a predetermined threshold, such as size being 1. In some examples, the sub cubical boxes with a size of 1 (unit size) are referred to as voxels, and the sub cubical boxes that have larger sizes than voxels can be referred to as non-voxels.

[0124] FIG. 10 shows an example of an octree partition (1010) and an octree structure (1020) corresponding to the octree partition (1010) according to some embodiments of the disclosure. FIG. 10 shows two levels of partitions in the octree partition (1010). The octree structure (1020) includes a node (N0) corresponding to the cubical box for octree partition (1010). At a first level, the cubical box is partitioned into 8 sub cubical boxes that are numbered 0-7 according to the numbering technique shown in FIG. 9. The occupancy code for the partition of the node N0 is “10000001” in binary, which indicates the first sub cubical box represented by node N0-0 and the eighth sub cubical box represented by node N0-7 includes points in the point cloud and other sub cubical boxes are empty.

[0125] Then, at the second level of partition, the first sub cubical box (represented by node N0-0) and the eighth sub cubical box (represented by node N0-7) are further respectively sub-divided into eight octants. For example, the first sub cubical box (represented by node N0-0) is partitioned into 8 smaller sub cubical boxes that are numbered 0-7 according to the numbering technique shown in FIG. 9. The occupancy code for the partition of the node N0-0 is "00011000" in binary, which indicates the fourth smaller sub cubical box (represented by node N0-0-3) and the fifth smaller sub cubical box (represented by node N0-0-4) includes points in the point cloud and other smaller sub cubical boxes are empty. At the second level, the seventh sub cubical box (represented by node N0-7) is similarly partitioned into 8 smaller sub cubical boxes as shown in FIG. 10.

[0126] In the FIG. 10 example, the nodes corresponding to non-empty cubical space (e.g., cubical box, sub cubical boxes, smaller sub cubical boxes and the like) are colored in grey, and referred to as shaded nodes.

[0127] In some embodiments, for a voxel of a unit size cubical box, point(s) in the unit size cubical box can be quantized to, for example, the center position of the unit size cubical box. When there are multiple points in the unit size cubical box, the points are merged to the center position of the unit size cubical box. The center positions of the voxels can form reconstructed positions for reconstructed points in a reconstructed point cloud.

[0128] According to some aspects of the disclosure, a suitable data structure (such as a lookup table, a list, and the like) can be used to associate original points in a voxel with a reconstructed position (e.g., center position) of the voxel. For example, an entry in a lookup table can associate an index of a voxel with indexes of original points in the voxel. The data structure can be stored. Thus, for each position in the reconstructed point cloud that is a center point of a voxel, the original points in the voxel that are merged to the center position can be determined based on the data structure.

[0129] According to an aspect of the disclosure, the geometry of point cloud is quantized by scalar quantization, and the scalar quantization is the only source of geometry distortion in some examples. In some embodiments, when the octree partition is used for geometry quantization, the voxels are the unit size cubical box that each includes one or more original points of the point cloud, and the one or more original points are quantized to the center point of the voxel. Thus, the one or more original points in the voxel share the same center point (in the reconstructed point cloud) of the voxel as the nearest neighbor, the set $\mathbb{Q}^+(i)$ of each position in

reconstructed point cloud is not empty. Therefore, the fourth step in the recoloring algorithm is not needed.

[0130] FIG. 11 shows a diagram in one-dimension (1D) to illustrate derivation techniques that can be used in three-dimension (3D) to determine the set of points sharing the reconstructed position ($\mathbb{Q}^+(i)$) during geometry quantization. The diamond-shaped points 1-8 are the original points and the round black dots R1-R3 are reconstructed positions for reconstructed points (also referred to as quantized points in some examples).

[0131] In an example, the quantization step is q, so all the points in the range $[\left(n-\frac{1}{2}\right)q, \left(n+\frac{1}{2}\right)q)$ can be be quantized to the position nq, for $n = 0, \pm 1, \pm 2, ...$ For example, the original points 1-3 are in a range of [-3q/2, -q/2) and are quantized to the reconstructed position R1, the original points 4-5 are in a range of [-q/2, q/2) and are quantized to the reconstructed position R2, the original points 6-8 are in a range of [q/2, 3q/2) and are quantized to the reconstructed position R3. Therefore, the nearest neighbor of an original point in the reconstructed point cloud is the reconstructed position. In other words, the set $\mathbb{Q}^+(i)$ of each reconstructed position is the original points that are quantized to the reconstructed position. Thus, in some embodiments, a suitable data structure is used to store the set $\mathbb{Q}^+(i)$ associated with the reconstructed position. Then, based on the data structure, for the reconstructed position, the set $\mathbb{Q}^+(i)$ can be quickly determined without additional k-d tree search.

[0132] Based on the quantization results stored in the suitable data structure, the recoloring algorithm can be simplified. For example, the attribute value of each quantized position is assigned based on the attribute values at original positions that are quantized to the quantized position. Specifically, in some examples, the recoloring algorithm can include 3 steps that are described in detail in the following description.

[0133] In a first step, the original positions of the points in the point cloud, attributes associated with the original positions and the reconstructed positions are received. In an example, $(X_i)_{i=0\ldots N-1}$ denote the original positions of the points in the original point cloud, N denotes the number of points in the original point cloud, $(\tilde{X}_i)_{i=0\ldots N_{rec}-1}$ denote the reconstructed positions of the reconstructed point cloud, and N_{rec} denotes the number of reconstructed positions in the reconstructed point cloud. If duplicated points exist and are merged, then $N_{rec} < N$, otherwise $N_{rec} = N$.

[0134] In a second step, for each reconstructed position $\tilde{X}_i$ in the reconstructed point cloud, the data structure that stores a set of original positions in association with each reconstructed position can be accessed based on the reconstructed position to determine the set of original positions $\mathbb{Q}(i) = (X_i(h))_{h\in\{1,\ldots,H(i)\}}$, original positions in the set are quantized and dequantized to $\tilde{X}_i$. $H(i)$ denotes the number of elements in $\mathbb{Q}(i)$, and $X_i(h)$ denotes one of the elements of $\mathbb{Q}(i)$. $\mathbb{Q}(i)$ can be obtained during the geometry quantization process.

[0135] In a third step, the attribute value $\tilde{a}_i$ associated with the reconstructed position $\tilde{X}_i$ can be obtained from attributes values associated with the original positions in the set $\mathbb{Q}(i)$.

[0136] Various techniques can be used to calculate attribute value $\tilde{a}_i$.

[0137] In an embodiment, an average of the attributes values is calculated as the attribute value $\tilde{a}_i$ for the reconstructed position. In an example, $a_i(h)$ denotes an attribute value associated with $X_i(h)$, the attribute value $\tilde{a}_i$ associated with the reconstructed point $\tilde{X}_i$ can be calculated using (Eq. 2):

$$\tilde{a}_i = \frac{1}{H(i)}\sum_{h=1}^{H(i)} a_i(h) \qquad \text{(Eq. 2)}$$

[0138] In another embodiment, $\tilde{a}_i$ is calculated by the weighted average of the attributes, such as using (Eq. 3):

$$\tilde{a}_i = \frac{1}{H(i)}\sum_{h=1}^{H(i)} w_i(h)\cdot a_i(h) \qquad \text{(Eq. 3)}$$

where $\sum_{h=1}^{H(i)} w_i(h) = 1$, $w_i(h) \geq 0$ and denotes the weight for attribute at a corresponding position, which can be inversely proportional to the distance between the original position $X_i(h)$ and reconstructed position $\tilde{X}_i$. The distance between the original position $X_i(h)$ and reconstructed position $\tilde{X}_i$ can be evaluated by any suitable distance measure. In an example, the distance is a spatial distance, such as Euclidean distance.

[0139] In another embodiment, $\tilde{a}_i$ is assigned by the attribute value of a nearest original position in $\mathbb{Q}(i)$ to the reconstructed position $\tilde{X}_i$. When multiple original positions have the same nearest distance to the reconstructed position $\tilde{X}_i$, a median attribute value or a mean attribute value of the attribute values of the multiple original positions can be assigned to $\tilde{a}_i$. The distance between the original position $X_i(h)$ and reconstructed position $\tilde{X}_i$ can be evaluated by any other distance measure. In an example, the distance is a spatial distance, such as Euclidean distance.

[0140] FIG. 12 shows a flow chart outlining a process (1200) according to an embodiment of the disclosure. The process (1200) can be used during a coding process for point clouds. In various embodiments, the process (1200) is executed by processing circuitry, such as the processing circuitry in the terminal devices (110), the processing circuitry that performs functions of the encoder (203) and/or the decoder (201), the processing circuitry that performs functions of the encoder (300), the decoder (400), the encoder (700), and/or the decoder (800), and the like. In some embodiments, the process (1200) is implemented in software instructions, thus when the processing circuitry executes the software instructions, the processing circuitry performs the process (1200). The process starts at (S1201) and proceeds to (S1210).

[0141] At (S1210), one or more original points in a point cloud that are associated with a reconstructed position are determined. Positions of the one or more original points are reconstructed, according to a geometry quantization, to the reconstructed position.

[0142] In some examples, octree partition can be performed for coding geometry information. The octree partition can partition a space of the point cloud into voxels. The one or more original points are in a voxel, and are associated with the reconstructed position for representing the voxel. The geometry information of the one or more original points can be reconstructed (e.g., quantized and dequantized) to the reconstructed position. In an embodiment, a data structure that associates the one or more original points with the reconstructed position is stored. The data structure can be accessed based on the reconstructed position to retrieve the one or more original points associated with the reconstructed position.

[0143] At (S1220), an attribute value for the reconstructed position is determined based on attribute information of the one or more original points.

[0144] In an embodiment, an average of attribute values of multiple original points is calculated as the determined attribute value for the reconstructed position when multiple original points are associated with the reconstructed position.

[0145] In some examples, an attribute value of an original point in the multiple original points can be weighed based on an inverse of a distance between the original point and the reconstructed position.

[0146] In another embodiment, a weighted average of the attribute values of the multiple original points is calculated as the determined attribute value for the reconstructed position. In an example, an attribute value of an original point in the multiple original points is weighted based on an inverse of a distance between the original point and the reconstructed position.

[0147] In another embodiment, a specific attribute value of a nearest point in multiple original points is assigned to be the determined attribute value for the reconstructed position in response to the multiple original points being associated with the reconstructed position. In an example, a median attribute value among attribute values for nearest points is selected in response to an existing of multiple nearest points having a same shortest distance to the reconstructed position. In another example, a mean attribute value of attribute values for nearest points is calculated in response to an existing of multiple nearest points having a same shortest distance to the reconstructed position.

[0148] At (S1230), texture of the point cloud is encoded with the reconstructed position having the determined attribute value. In an example, attribute values for the reconstructed positons are compressed and included in a coded bitstream for the point cloud as the texture information of the point cloud. Then, the process proceeds to (S1299) and terminates.

[0149] The techniques described above, can be implemented as computer software using computer-readable instructions and physically stored in one or more computer-readable media. For example, FIG. 13 shows a computer system (1300) suitable for implementing certain embodiments of the disclosed subject matter.

[0150] The computer software can be coded using any suitable machine code or computer language, that may be subject to assembly, compilation, linking, or like mechanisms to create code comprising instructions that can be executed directly, or through interpretation, micro-code execution, and the like, by one or more computer central processing units (CPUs), Graphics Processing Units (GPUs), and the like.

[0151] The instructions can be executed on various types of computers or components thereof, including, for example, personal computers, tablet computers, servers, smartphones, gaming devices, internet of things devices, and the like.

[0152] The components shown in FIG. 13 for computer system (1300) are exemplary in nature and are not intended to suggest any limitation as to the scope of use or functionality of the computer software implementing embodiments of the present disclosure. Neither should the configuration of components be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in the exemplary embodiment of a computer system (1300).

[0153] Computer system (1300) may include certain human interface input devices. Such a human interface input device may be responsive to input by one or more human users

through, for example, tactile input (such as: keystrokes, swipes, data glove movements), audio input (such as: voice, clapping), visual input (such as: gestures), olfactory input (not depicted). The human interface devices can also be used to capture certain media not necessarily directly related to conscious input by a human, such as audio (such as: speech, music, ambient sound), images (such as: scanned images, photographic images obtain from a still image camera), video (such as two-dimensional video, three-dimensional video including stereoscopic video).

[0154] Input human interface devices may include one or more of (only one of each depicted): keyboard (1301), mouse (1302), trackpad (1303), touch screen, data-glove (not shown), joystick (1305), microphone (1306), scanner (1307), camera (1308).

[0155] Computer system (1300) may also include certain human interface output devices. Such human interface output devices may be stimulating the senses of one or more human users through, for example, tactile output, sound, light, and smell/taste. Such human interface output devices may include tactile output devices (for example tactile feedback by the touch-screen, data-glove (not shown), or joystick (1305), but there can also be tactile feedback devices that do not serve as input devices), audio output devices (such as: speakers (1309), headphones (not depicted)), visual output devices (such as screens (1310) to include CRT screens, LCD screens, plasma screens, OLED screens, each with or without touch-screen input capability, each with or without tactile feedback capability—some of which may be capable to output two dimensional visual output or more than three dimensional output through means such as stereographic output; virtual-reality glasses (not depicted), holographic displays and smoke tanks (not depicted)), and printers (not depicted).

[0156] Computer system (1300) can also include human accessible storage devices and their associated media such as optical media including CD/DVD ROM/RW (1320) with CD/DVD or the like media (1321), thumb-drive (1322), removable hard drive or solid state drive (1323), legacy magnetic media such as tape and floppy disc (not depicted), specialized ROM/ASIC/PLD based devices such as security dongles (not depicted), and the like.

[0157] Those skilled in the art should also understand that term “computer readable media” as used in connection with the presently disclosed subject matter does not encompass transmission media, carrier waves, or other transitory signals.

[0158] Computer system (1300) can also include an interface to one or more communication networks. Networks can for example be wireless, wireline, optical. Networks can further be local, wide-area, metropolitan, vehicular and industrial, real-time, delay-tolerant,

and so on. Examples of networks include local area networks such as Ethernet, wireless LANs, cellular networks to include GSM, 3G, 4G, 5G, LTE and the like, TV wireline or wireless wide area digital networks to include cable TV, satellite TV, and terrestrial broadcast TV, vehicular and industrial to include CANBus, and so forth. Certain networks commonly require external network interface adapters that attached to certain general purpose data ports or peripheral buses (1349) (such as, for example USB ports of the computer system (1300)); others are commonly integrated into the core of the computer system (1300) by attachment to a system bus as described below (for example Ethernet interface into a PC computer system or cellular network interface into a smartphone computer system). Using any of these networks, computer system (1300) can communicate with other entities. Such communication can be uni-directional receive only (for example, broadcast TV), uni-directional send-only (for example CANbus to certain CANbus devices), or bi-directional, for example to other computer systems using local or wide area digital networks. Certain protocols and protocol stacks can be used on each of those networks and network interfaces as described above.

[0159] Aforementioned human interface devices, human-accessible storage devices, and network interfaces can be attached to a core (1340) of the computer system (1300).

[0160] The core (1340) can include one or more Central Processing Units (CPU) (1341), Graphics Processing Units (GPU) (1342), specialized programmable processing units in the form of Field Programmable Gate Areas (FPGA) (1343), hardware accelerators for certain tasks (1344), and so forth. These devices, along with Read-only memory (ROM) (1345), Random-access memory (1346), internal mass storage such as internal non-user accessible hard drives, SSDs, and the like (1347), may be connected through a system bus (1348). In some computer systems, the system bus (1348) can be accessible in the form of one or more physical plugs to enable extensions by additional CPUs, GPU, and the like. The peripheral devices can be attached either directly to the core's system bus (1348), or through a peripheral bus (1349). Architectures for a peripheral bus include PCI, USB, and the like.

[0161] CPUs (1341), GPUs (1342), FPGAs (1343), and accelerators (1344) can execute certain instructions that, in combination, can make up the aforementioned computer code. That computer code can be stored in ROM (1345) or RAM (1346). Transitional data can be also be stored in RAM (1346), whereas permanent data can be stored for example, in the internal mass storage (1347). Fast storage and retrieve to any of the memory devices can be enabled through

the use of cache memory, that can be closely associated with one or more CPU (1341), GPU (1342), mass storage (1347), ROM (1345), RAM (1346), and the like.

[0162] The computer readable media can have computer code thereon for performing various computer-implemented operations. The media and computer code can be those specially designed and constructed for the purposes of the present disclosure, or they can be of the kind well known and available to those having skill in the computer software arts.

[0163] As an example and not by way of limitation, the computer system having architecture (1300), and specifically the core (1340) can provide functionality as a result of processor(s) (including CPUs, GPUs, FPGA, accelerators, and the like) executing software embodied in one or more tangible, computer-readable media. Such computer-readable media can be media associated with user-accessible mass storage as introduced above, as well as certain storage of the core (1340) that are of non-transitory nature, such as core-internal mass storage (1347) or ROM (1345). The software implementing various embodiments of the present disclosure can be stored in such devices and executed by core (1340). A computer-readable medium can include one or more memory devices or chips, according to particular needs. The software can cause the core (1340) and specifically the processors therein (including CPU, GPU, FPGA, and the like) to execute particular processes or particular parts of particular processes described herein, including defining data structures stored in RAM (1346) and modifying such data structures according to the processes defined by the software. In addition or as an alternative, the computer system can provide functionality as a result of logic hardwired or otherwise embodied in a circuit (for example: accelerator (1344)), which can operate in place of or together with software to execute particular processes or particular parts of particular processes described herein. Reference to software can encompass logic, and vice versa, where appropriate. Reference to a computer-readable media can encompass a circuit (such as an integrated circuit (IC)) storing software for execution, a circuit embodying logic for execution, or both, where appropriate. The present disclosure encompasses any suitable combination of hardware and software.

[0164] While this disclosure has described several exemplary embodiments, there are alterations, permutations, and various substitute equivalents, which fall within the scope of the disclosure..

CLAIMS:

1. A method for point cloud coding, comprising:

determining, by a processor, one or more original points in a point cloud that are associated with a reconstructed position, positions of the one or more original points being reconstructed, according to a geometry quantization, to the reconstructed position;

determining, by the processor, an attribute value for the reconstructed position based on attribute information of the one or more original points; and

encoding, by the processor, texture of the point cloud with the reconstructed position having the determined attribute value.

2. The method of claim 1, further comprising:

storing, a data structure that associates the one or more original points with the reconstructed position according to the geometry quantization.

3. The method of claim 1, further comprising:

performing, by the processor, an octree partition that partitions a space of the point cloud into voxels; and

associating, by the processor, the one or more original points that are positioned in a voxel, of the voxels, with the reconstructed position for representing the voxel.

4. The method of claim 2, further comprising:

accessing the data structure based on the reconstructed position to retrieve the one or more original points associated with the reconstructed position.

5. The method of claim 1, further comprising:

calculating, by the processor, an average of attribute values of multiple original points as the determined attribute value for the reconstructed position in response to the multiple original points being associated with the reconstructed position.

6. The method of claim 1, further comprising:

calculating, by the processor, a weighted average of the attribute values of the multiple original points as the determined attribute value for the reconstructed position.

7. The method of claim 6, further comprising:

weighting, by the processor, an attribute value of an original point in the multiple original points based on an inverse of a distance between the original point and the reconstructed position.

8. The method of claim 1, further comprising:

assigning, by the processor, a specific attribute value of a nearest point in multiple original points to be the determined attribute value for the reconstructed position in response to the multiple original points being associated with the reconstructed position.

9. The method of claim 8, further comprising:

selecting, by the processor, a median attribute value among attribute values for nearest points in response to the nearest points in the multiple original points having a same shortest distance to the reconstructed position.

10. The method of claim 8, further comprising:

calculating, by the processor, a mean attribute value of attribute values for nearest points in response to the nearest points in the multiple original points having a same shortest distance to the reconstructed position.

11. An apparatus for point cloud coding, comprising:

processing circuitry configured to perform the method according to any one of claims 1 to 10.

12. A non-transitory computer-readable medium, storing a program, wherein:

the program when executed by one or more processors is configured to perform the method according to any one of claims 1 to 10.