



(51) International Patent Classification:

Not classified

(21) International Application Number:

PCT/US2024/037747

(22) International Filing Date:

12 July 2024 (12.07.2024)

(25) Filing Language:

English

(26) Publication Language:

English

(30) Priority Data:

63/526,440	12 July 2023 (12.07.2023)	US
63/526,441	12 July 2023 (12.07.2023)	US
63/598,097	11 November 2023 (11.11.2023)	US
18/769,745	11 July 2024 (11.07.2024)	US

(71) Applicant: TENCENT AMERICA LLC [US/US]; 2747

Park Boulevard, Palo Alto, California 94306 (US).

(72) Inventors: NGUYEN CANH, Thuong; c/o TENCENT

AMERICA LLC, 2747 Park Boulevard, Palo Alto, California 94306 (US). HUANG, Chao; c/o TENCENT AMERICA LLC, 2747 Park Boulevard, Palo Alto, California 94306 (US). XU, Xiaozhong; c/o TENCENT AMERICA LLC, 2747 Park Boulevard, Palo Alto, California 94306 (US). LIU, Shan; c/o TENCENT AMERICA LLC, 2747 Park Boulevard, Palo Alto, California 94306 (US).

(74) Agent: RABENA, John F. et al.; SUGHRUE MION,

PLLC, 2000 Pennsylvania Ave., N.W., Suite 9000, Washington, District of Columbia 20006 (US).

(81) Designated States (unless otherwise indicated, for every

kind of national protection available): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CV, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IQ, IR, IS, IT, JM, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, MG, MK, MN, MU, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

(84) Designated States (unless otherwise indicated, for every

kind of regional protection available): ARIPO (BW, CV,

GH, GM, KE, LR, LS, MW, MZ, NA, RW, SC, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, ME, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

— without international search report and to be republished upon receipt of that report (Rule 48.2(g))

(54) Title: OPTIMIZED POSITION AND CONNECTIVITY CODING FOR DUAL DEGREE MESH COMPRESSION

(57) Abstract: A method and apparatus comprising computer code configured to cause a processor or processors to obtain, from a bitstream, a mesh representing an encoded volumetric data of at least one three-dimensional (3D) visual content; partition a plurality of vertices of the mesh into a plurality of groups; and decode the encoded volumetric data by predicting the vertices in each group of the plurality of groups based on a plurality of traversal orders depending on an adaptive reference vertex of the vertices, and the plurality of traversal orders includes a first order from the adaptive reference vertex to a first alternative reference vertex of the vertices, and wherein the plurality of traversal orders includes a second order from the adaptive reference vertices to a second alternative reference vertex of the vertices.



OPTIMIZED POSITION AND CONNECTIVITY CODING FOR DUAL DEGREE MESH COMPRESSION

CROSS REFERENCE TO RELATED APPLICATION

[0001] This application claims priority to US provisional application 63/526,441, filed on July 12, 2023, US provisional application 63/526,440, filed on July 12, 2023, US provisional application 63/598,097, filed on November 11, 2023, and US application 18/769,745, filed on July 11, 2024, the disclosures of which are incorporated herein by reference in their entireties.

BACKGROUND

1. Field

[0002] This disclosure relates novel methods for efficiently compressing the connectivity and attribute information of triangular and polygonal meshes by employing an optimized dual degree connectivity coding scheme, for efficiently compressing the position attribute with refining the input mesh, for interleave coding of connectivity and position attribute in dual degree mesh coding, and for dual degree mesh coding based on adaptive multiple parallelogram prediction.

2. Description of Related Art

[0003] The advances in 3D capture, modeling, and rendering have promoted the ubiquitous presence of 3D contents across several platforms and devices. Nowadays, it is possible to capture a baby's first step in one continent and allow the grandparents to see (and maybe interact) and enjoy a full immersive experience with the child in another continent. Nevertheless, in order to achieve such realism, models are becoming ever more sophisticated,

and a significant amount of data is linked to the creation and consumption of those models. 3D meshes are widely used to represent such immersive contents.

[0004] A mesh is composed of several polygons that describe the surface of a volumetric object. Each polygon is defined by its vertices in 3D space and the information of how the vertices are connected, referred to as connectivity information. Optionally, vertex attributes, such as colors, normals, etc., could be associated with the mesh vertices. Attributes could also be associated with the surface of the mesh by exploiting mapping information that parameterizes the mesh with 2D attribute maps. Such mapping is usually described by a set of parametric coordinates, referred to as UV coordinates or texture coordinates, associated with the mesh vertices. 2D attribute maps are used to store high resolution attribute information such as texture, normals, displacements etc. Such information could be used for various purposes such as texture mapping and shading.

[0005] A dynamic mesh sequence may require a large amount of data since it may consist of a significant amount of information changing over time. Therefore, efficient compression technologies are required to store and transmit such contents. Mesh compression standards IC, MESHGRID, FAMC were previously developed by MPEG to address dynamic meshes with constant connectivity and time varying geometry and vertex attributes. However, these standards do not take into account time varying attribute maps and connectivity information. DCC (Digital Content Creation) tools usually generate such dynamic meshes. In counterpart, it is challenging for volumetric acquisition techniques to generate a constant connectivity dynamic mesh, especially under real time constraints. This type of contents is not supported by the existing standards. MPEG is planning to develop a new mesh compression standard to directly handle dynamic meshes with time varying connectivity information and

optionally time varying attribute maps. This standard targets lossy, and lossless compression for various applications, such as real-time communications, storage, free viewpoint video, AR and VR. Functionalities such as random access and scalable/progressive coding are also considered.

[0006] And for any of those reasons there is therefore a desire for technical solutions to such problems that arose in video coding technology.

SUMMARY

[0007] There is included a method and apparatus comprising memory configured to store computer program code and a processor or processors configured to access the computer program code and operate as instructed by the computer program code. The computer program is configured to cause the processor implement code configured to cause the at least one processor to obtain, from a bitstream, a mesh representing an encoded volumetric data of at least one three-dimensional (3D) visual content; partitioning code configured to cause the at least one processor to partition a plurality of vertices of the mesh into a plurality of groups; and decoding code configured to cause the at least one processor to decode the encoded volumetric data by predicting the vertices in each group of the plurality of groups based on a plurality of traversal orders depending on an adaptive reference vertex of the vertices, the plurality of traversal orders includes a first order from the adaptive reference vertex to a first alternative reference vertex of the vertices, and the plurality of traversal orders includes a second order from the adaptive reference vertices to a second alternative reference vertex of the vertices.

[0008] A binary flag may be included in the bitstream and indicates either the first order or the second order.

[0009] Decoding the encoded volumetric data may be based on determining at least one of the plurality of traversal orders based on a number of faces of the mesh and without any binary flag included in the bitstream and indicating any of the first order and the second order.

[0010] Content of the bitstream may be based on a rate estimated from a sum of absolute differences (SAD) of a residual vector.

[0011] Content of the bitstream may represent a replacement vertex of the adaptive vertex and is determined based on determining a plurality of fractions of a line segment of a face of the mesh.

[0012] Decoding the encoded volumetric data may be based on interleave coding indicating at least one of face traversal and pivot traversal of the mesh.

[0013] The mesh may include a parallelogram.

BRIEF DESCRIPTION OF THE DRAWINGS

[0014] Further features, nature, and various advantages of the disclosed subject matter will be more apparent from the following detailed description and the accompanying drawings in which:

[0015] Fig. 1 is a schematic illustration of a computer environment in accordance with embodiments;

[0016] Fig. 2 is a simplified block diagram of media processing in accordance with embodiments;

[0017] Fig. 3 is a simplified illustration of decoding in accordance with embodiments;

[0018] Fig. 4 is a simplified illustration of encoding in accordance with embodiments;

[0019] Fig. 5 is a simplified illustration of media processing in accordance with embodiments;

[0020] Fig. 6 is a simplified illustration of media processing in accordance with embodiments;

[0021] Fig. 7 is a simplified illustration of media processing in accordance with embodiments;

[0022] Fig. 8 is a simplified illustration of mesh features in accordance with embodiments;

[0023] Fig. 9 is a simplified illustration of mesh features in accordance with embodiments;

[0024] Fig. 10 is a simplified flow diagram for media processing in accordance with embodiments;

[0025] Fig. 11 is a simplified flow diagram for media processing in accordance with embodiments;

[0026] Fig. 12 is a simplified flow diagram for media processing in accordance with embodiments;

[0027] Fig. 13 is a simplified illustration for mesh features in accordance with embodiments;

[0028] Fig. 14 is a simplified illustration for mesh features in accordance with embodiments;

[0029] Fig. 15 is a simplified illustration for mesh features in accordance with embodiments;

[0030] Fig. 16 is a simplified illustration for mesh features in accordance with embodiments;

[0031] Fig. 17 is a simplified illustration for mesh features in accordance with embodiments;

[0032] Fig. 18 is a simplified illustration for mesh features in accordance with embodiments;

[0033] Fig. 19 is a simplified illustration for mesh features in accordance with embodiments;

[0034] Fig. 20 is a simplified illustration for mesh features in accordance with embodiments; and

[0035] Fig. 21 is a simplified d of computer features in accordance with embodiments.

DETAILED DESCRIPTION

[0036] The proposed features discussed below may be used separately or combined in any order. Further, the embodiments 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.

[0037] Fig. 1 illustrates a simplified block diagram of a communication system 100 according to an embodiment of the present disclosure. The communication system 100 may include at least two terminals 102 and 103 interconnected via a network 105. For unidirectional transmission of data, a first terminal 103 may code video data at a local location for transmission to the other terminal 102 via the network 105. The second terminal 102 may receive the coded video data of the other terminal from the network 105, decode the coded data and display the recovered video data. Unidirectional data transmission may be common in media serving applications and the like.

[0038] Fig. 1 illustrates a second pair of terminals 101 and 104 provided to support bidirectional transmission of coded video that may occur, for example, during videoconferencing. For bidirectional transmission of data, each terminal 101 and 104 may code video data captured at a local location for transmission to the other terminal via the network 105. Each terminal 101 and 104 also may receive the coded video data transmitted by the other terminal, may decode the coded data and may display the recovered video data at a local display device.

[0039] In Fig. 1, the terminals 101, 102, 103 and 104 may be illustrated as servers, personal computers and smart phones but the principles of the present disclosure are not so limited. Embodiments of the present disclosure find application with laptop computers, tablet computers, media players and/or dedicated video conferencing equipment. The network 105 represents any number of networks that convey coded video data among the terminals 101, 102, 103 and 104, including for example wireline and/or wireless communication networks. The communication network 105 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 105 may be immaterial to the operation of the present disclosure unless explained herein below.

[0040] Fig. 2 illustrates, as an example for an application for the disclosed subject matter, the placement of a video encoder and decoder in a streaming environment. The disclosed subject matter can be equally applicable to other video enabled applications, including, for example, video conferencing, digital TV, storing of compressed video on digital media including CD, DVD, memory stick and the like, and so on.

[0041] A streaming system may include a capture subsystem 203, that can include a video source 201, for example a digital camera, creating, for example, an uncompressed video sample stream 213. That sample stream 213 may be emphasized as a high data volume when compared to encoded video bitstreams and can be processed by an encoder 202 coupled to the video source 201, which may be for example a camera as discussed above. The encoder 202 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 encoded video bitstream 204, which may be emphasized as a lower data volume when compared to the sample stream, can be stored on a streaming server 205 for future use. One or more streaming clients 212 and 207 can access the streaming server 205 to retrieve copies 208 and 206 of the encoded video bitstream 204. A client 212 can include a video decoder 211 which decodes the incoming copy of the encoded video bitstream 208 and creates an outgoing video sample stream 210 that can be rendered on a display 209 or other rendering device (not depicted). In some streaming systems, the video bitstreams 204, 206 and 208 can be encoded according to certain video coding/compression standards. Examples of those standards are noted above and described further herein.

[0042] Fig. 3 may be a functional block diagram of a video decoder 300 according to an embodiment of the present invention.

[0043] A receiver 302 may receive one or more codec video sequences to be decoded by the decoder 300; in the same or another embodiment, one coded video sequence at a time, where the decoding of each coded video sequence is independent from other coded video sequences. The coded video sequence may be received from a channel 301, which may be a hardware/software link to a storage device which stores the encoded video data. The receiver

302 may receive the encoded video data with other data, for example, coded audio data and/or ancillary data streams, that may be forwarded to their respective using entities (not depicted).

The receiver 302 may separate the coded video sequence from the other data. To combat network jitter, a buffer memory 303 may be coupled in between receiver 302 and entropy decoder / parser 304 (“parser” henceforth). When receiver 302 is receiving data from a store/forward device of sufficient bandwidth and controllability, or from an isosynchronous network, the buffer 303 may not be needed, or can be small. For use on best effort packet networks such as the Internet, the buffer 303 may be required, can be comparatively large and can advantageously of adaptive size.

[0044] The video decoder 300 may include a parser 304 to reconstruct symbols 313 from the entropy coded video sequence. Categories of those symbols include information used to manage operation of the decoder 300, and potentially information to control a rendering device such as a display 312 that is not an integral part of the decoder but can be coupled to it. The control information for the rendering device(s) may be in the form of Supplementary Enhancement Information (SEI messages) or Video Usability Information (VUI) parameter set fragments (not depicted). The parser 304 may parse / entropy-decode the coded video sequence received. The coding of the coded video sequence can be in accordance with a video coding technology or standard, and can follow principles well known to a person skilled in the art, including variable length coding, Huffman coding, arithmetic coding with or without context sensitivity, and so forth. The parser 304 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 parameters 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 entropy decoder / parser may also extract from the coded video sequence information such as transform coefficients, quantizer parameter values, motion vectors, and so forth.

[0045] The parser 304 may perform entropy decoding / parsing operation on the video sequence received from the buffer 303, so to create symbols 313. The parser 304 may receive encoded data, and selectively decode particular symbols 313. Further, the parser 304 may determine whether the particular symbols 313 are to be provided to a Motion Compensation Prediction unit 306, a scaler / inverse transform unit 305, an Intra Prediction Unit 307, or a loop filter 311.

[0046] Reconstruction of the symbols 313 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 304. The flow of such subgroup control information between the parser 304 and the multiple units below is not depicted for clarity.

[0047] Beyond the functional blocks already mentioned, decoder 300 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.

[0048] A first unit is the scaler / inverse transform unit 305. The scaler / inverse transform unit 305 receives quantized transform coefficient as well as control information,

including which transform to use, block size, quantization factor, quantization scaling matrices, etc. as symbol(s) 313 from the parser 304. It can output blocks comprising sample values, that can be input into aggregator 310.

[0049] In some cases, the output samples of the scaler / inverse transform 305 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 307. In some cases, the intra picture prediction unit 307 generates a block of the same size and shape of the block under reconstruction, using surrounding already reconstructed information fetched from the current (partly reconstructed) picture 309. The aggregator 310, in some cases, adds, on a per sample basis, the prediction information the intra prediction unit 307 has generated to the output sample information as provided by the scaler / inverse transform unit 305.

[0050] In other cases, the output samples of the scaler / inverse transform unit 305 can pertain to an inter coded, and potentially motion compensated block. In such a case, a Motion Compensation Prediction unit 306 can access reference picture memory 308 to fetch samples used for prediction. After motion compensating the fetched samples in accordance with the symbols 313 pertaining to the block, these samples can be added by the aggregator 310 to the output of the scaler / inverse transform unit (in this case called the residual samples or residual signal) so to generate output sample information. The addresses within the reference picture memory from where the motion compensation unit fetches prediction samples can be controlled by motion vectors, available to the motion compensation unit in the form of symbols 313 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 when sub-sample exact motion vectors are in use, motion vector prediction mechanisms, and so forth.

[0051] The output samples of the aggregator 310 can be subject to various loop filtering techniques in the loop filter unit 311. Video compression technologies can include in-loop filter technologies that are controlled by parameters included in the coded video bitstream and made available to the loop filter unit 311 as symbols 313 from the parser 304, 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.

[0052] The output of the loop filter unit 311 can be a sample stream that can be output to the render device 312 as well as stored in the reference picture memory 557 for use in future inter-picture prediction.

[0053] Certain coded pictures, once fully reconstructed, can be used as reference pictures for future prediction. Once a coded picture is fully reconstructed and the coded picture has been identified as a reference picture (by, for example, parser 304), the current reference picture 309 can become part of the reference picture buffer 308, and a fresh current picture memory can be reallocated before commencing the reconstruction of the following coded picture.

[0054] The video decoder 300 may perform decoding operations according to a predetermined video compression technology that may be documented 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 it adheres to the syntax of the video compression technology or standard, as specified in the video compression technology document or standard and specifically in the profiles document therein. 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.

[0055] In an embodiment, the receiver 302 may receive additional (redundant) data with the encoded video. The additional data may be included as part of the coded video sequence(s). The additional data may be used by the video decoder 300 to properly decode the data and/or to more accurately reconstruct the original video data. Additional data can be in the form of, for example, temporal, spatial, or signal-to-noise ratio (SNR) enhancement layers, redundant slices, redundant pictures, forward error correction codes, and so on.

[0056] Fig. 4 may be a functional block diagram of a video encoder 400 according to an embodiment of the present disclosure.

[0057] The encoder 400 may receive video samples from a video source 401 (that is not part of the encoder) that may capture video image(s) to be coded by the encoder 400.

[0058] The video source 401 may provide the source video sequence to be coded by the encoder (303) in the form of a digital video sample stream that can be of any suitable bit depth (for example: 8 bit, 10 bit, 12 bit, ...), any colorspace (for example, BT.601 Y CrCb, RGB, ...) and any suitable sampling structure (for example Y CrCb 4:2:0, Y CrCb 4:4:4). In a media serving system, the video source 401 may be a storage device storing previously prepared video. In a videoconferencing system, the video source 401 may be a camera that captures local image information as a video sequence. Video data may be provided as a plurality of individual

pictures that impart motion when viewed in sequence. The pictures themselves may be organized as a spatial array of pixels, wherein each pixel can comprise one or more samples depending on the sampling structure, color space, etc. in use. A person skilled in the art can readily understand the relationship between pixels and samples. The description below focuses on samples.

[0059] According to an embodiment, the encoder 400 may code and compress the pictures of the source video sequence into a coded video sequence 410 in real time or under any other time constraints as required by the application. Enforcing appropriate coding speed is one function of Controller 402. Controller controls other functional units as described below and is functionally coupled to these units. The coupling is not depicted for clarity. Parameters set by controller 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. A person skilled in the art can readily identify other functions of controller 402 as they may pertain to video encoder 400 optimized for a certain system design.

[0060] Some video encoders operate in what a person skilled in the art readily recognizes as a “coding loop.” As an oversimplified description, a coding loop can consist of the encoding part of an encoder 400 (“source coder” henceforth) (responsible for creating symbols based on an input picture to be coded, and a reference picture(s)), and a (local) decoder 406 embedded in the encoder 400 that reconstructs the symbols to create the sample data that 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). That reconstructed sample stream is input to the reference picture memory 405. As the decoding of a symbol stream

leads to bit-exact results independent of decoder location (local or remote), the reference picture buffer content is also bit exact between 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 well known to a person skilled in the art.

[0061] The operation of the “local” decoder 406 can be the same as of a “remote” decoder 300, which has already been described in detail above in conjunction with Fig. 3. Briefly referring also to Fig. 4, however, as symbols are available and en/decoding of symbols to a coded video sequence by entropy coder 408 and parser 304 can be lossless, the entropy decoding parts of decoder 300, including channel 301, receiver 302, buffer 303, and parser 304 may not be fully implemented in local decoder 406.

[0062] 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. 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.

[0063] As part of its operation, the source coder 403 may perform motion compensated predictive coding, which codes an input frame predictively with reference to one or more previously-coded frames from the video sequence that were designated as “reference frames.” In this manner, the coding engine 407 codes differences between pixel blocks of an input frame and

pixel blocks of reference frame(s) that may be selected as prediction reference(s) to the input frame.

[0064] The local video decoder 406 may decode coded video data of frames that may be designated as reference frames, based on symbols created by the source coder 403. Operations of the coding engine 407 may advantageously be lossy processes. When the coded video data may be decoded at a video decoder (not shown in Fig. 4), the reconstructed video sequence typically may be a replica of the source video sequence with some errors. The local video decoder 406 replicates decoding processes that may be performed by the video decoder on reference frames and may cause reconstructed reference frames to be stored in the reference picture memory 405, which may be for example a cache. In this manner, the encoder 400 may store copies of reconstructed reference frames locally that have common content as the reconstructed reference frames that will be obtained by a far-end video decoder (absent transmission errors).

[0065] The predictor 404 may perform prediction searches for the coding engine 407. That is, for a new frame to be coded, the predictor 404 may search the reference picture memory 405 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 404 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 404, an input picture may have prediction references drawn from multiple reference pictures stored in the reference picture memory 405.

[0066] The controller 402 may manage coding operations of the source coder 403, which may be for example a video coder, including, for example, setting of parameters and subgroup parameters used for encoding the video data.

[0067] Output of all aforementioned functional units may be subjected to entropy coding in the entropy coder 408. The entropy coder translates the symbols as generated by the various functional units into a coded video sequence, by loss-less compressing the symbols according to technologies known to a person skilled in the art as, for example Huffman coding, variable length coding, arithmetic coding, and so forth.

[0068] The transmitter 409 may buffer the coded video sequence(s) as created by the entropy coder 408 to prepare it for transmission via a communication channel 411, which may be a hardware/software link to a storage device which would store the encoded video data. The transmitter 409 may merge coded video data from the source coder 403 with other data to be transmitted, for example, coded audio data and/or ancillary data streams (sources not shown).

[0069] The controller 402 may manage operation of the encoder 400. During coding, the controller 402 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 frame types:

[0070] An Intra Picture (I picture) may be one that may be coded and decoded without using any other frame in the sequence as a source of prediction. Some video codecs allow for different types of Intra pictures, including, for example Independent Decoder Refresh Pictures. A person skilled in the art is aware of those variants of I pictures and their respective applications and features.

[0071] 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.

[0072] 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.

[0073] Source pictures commonly may be subdivided spatially into a plurality of sample blocks (for example, blocks of 4 x 4, 8 x 8, 4 x 8, or 16 x 16 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 non-predictively, via spatial prediction or via temporal prediction with reference to one previously coded reference pictures. Blocks of B pictures may be coded non-predictively, via spatial prediction or via temporal prediction with reference to one or two previously coded reference pictures.

[0074] The encoder 400, which may be for example a video coder, may perform coding operations according to a predetermined video coding technology or standard, such as ITU-T Rec. H.265. In its operation, the encoder 400 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.

[0075] In an embodiment, the transmitter 409 may transmit additional data with the encoded video. The source coder 403 may include such data as part of the coded video sequence. Additional data may comprise temporal/spatial/SNR enhancement layers, other forms of redundant data such as redundant pictures and slices, Supplementary Enhancement Information (SEI) messages, Visual Usability Information (VUI) parameter set fragments, and so on.

[0076] Fig. 5 illustrates a simplified block-style workflow diagram 500 of exemplary view-port dependent processing in an Omnidirectional Media Application Format (OMAF) that may allow for 360-degree virtual reality (VR360) streaming described in OMAF.

[0077] At acquisition block 501, video data A is acquired, such as data of multiple images and audio of same time instances in a case that the image data may represent scenes in VR360. At processing block 503, the images B_i of the same time instance are processed by one or more of being stitched, mapped onto a projected picture with respect to one or more virtual reality (VR) angles or other angles/viewpoint(s) and region-wise packed. Additionally, metadata may be created indicating any of such processed information and other information so as to assist in delivering and rendering processes.

[0078] With respect to data D, at image encoding block 505, the projected pictures are encoded to data E_i and composed into a media file, and in viewport-independent streaming, and at video encoding block 504, the video pictures are encoded as data E_v as a single-layer bitstream, for example, and with respect to data B_a the audio data may also be encoded into data E_a at audio encoding block 502.

[0079] The data E_a , E_v , and E_i , the entire coded bitstream F_i and/or F may be stored at a (content delivery network (CDN)/cloud) server, and typically may be fully transmitted, such as at delivery block 507 or otherwise, to an OMAF player 520 and may be fully decoded by a decoder such that at least an area of a decoded picture corresponding to a current viewport is rendered to the user at display block 516 with respect to the various metadata, file playback, and orientation/viewport metadata, such as an angle at which a user may be looking through a VR image device with respect to viewport specifications of that device, from the head/eye tracking block 508. A distinct feature of VR360 is that only a viewport may be displayed at any particular time, and such feature may be utilized to improve the performance of omnidirectional video systems, through selective delivery depending on the user's viewport (or any other criteria, such as recommended viewport timed metadata). For example, viewport-dependent delivery may be enabled by tile-based video coding according to exemplary embodiments.

[0080] As with the encoding blocks described above, the OMAF player 520 according to exemplary embodiments may similarly reverse one or more facets of such encoding with respect to the file/segment decapsulation of one or more of the data F' and/or F'_i and metadata, decode the audio data E'_i at audio decoding block 510, the video data E'_v at video decoding block 513, and the image data E'_i at image decoding block 514 to proceed with audio rendering of the data B'_a at audio rendering block 511 and image rendering of the data D' at image rendering block 515 so as to output, in a VR360 format according to various metadata such as the orientation/viewport metadata, display data A'_i at display block 516 and audio data A'_s at the loudspeakers/headphones block 512. The various metadata may influence ones of the data decoding and rendering processes depending on various tracks, languages, qualities, views, that may be selected by or for a user of the OMAF player 520, and it is to be understood that the

order of processing described herein is presented for exemplary embodiments and may be implemented in other orders according to other exemplary embodiments.

[0081] Fig. 6 illustrates a simplified block-style content flow process diagram 600 for (coded) point cloud data with view-position and angle dependent processing of point cloud data (herein “V-PCC”) with respect to capturing/generating/(de)coding/rendering/displaying 6 degree-of-freedom media. It is to be understood that the described features may be used separately or combined in any order and elements such as for encoding and decoding, among others illustrated, may be implemented by processing circuitry (*e.g.*, one or more processors or one or more integrated circuits), and the one or more processors may execute a program that is stored in a non-transitory computer-readable medium according to exemplary embodiments.

[0082] The diagram 600 illustrates exemplary embodiments for streaming of coded point cloud data according to V-PCC.

[0083] At the volumetric data acquisition block 601, a real-world visual scene or a computer-generated visual scene (or combination of them) may be captured by a set of camera devices or synthesized by a computer as a volumetric data, and the volumetric data, which may have an arbitrary format, may be converted to a (quantized) point cloud data format, through image processing at the converting to point cloud block 602. For example, data from the volumetric data may be area data by area data converted into ones of points of the point cloud by pulling one or more of the values described below from the volumetric data and any associated data into a desired point cloud format according to exemplary embodiments. According to exemplary embodiments, the volumetric data may be a 3D data set of 2D images, such as slices from which a 2D projection of the 3D data set may be projected for example. According to exemplary embodiments, point cloud data formats include representations of data points in one

or more various spaces and may be used to represent the volumetric data and may offer improvements with respect to sampling and data compression, such as with respect to temporal redundancies, and, for example, a point cloud data in an x, y, z, format representing, at each point of multiple points of the cloud data, color values (e.g., RGB, etc.), luminance, intensity, etc. and could be used with progressive decoding, polygon meshing, direct rendering, octree 3D representations of 2D quadtree data.

[0084] At projection to images block 603, the acquired point cloud data may be projected onto 2D images and encoded as image/video pictures with video-based point cloud coding (V-PCC). The projected point cloud data may be composed of attributes, geometry, occupancy map, and other metadata used for point cloud data reconstruction such as with painter's algorithms, ray casting algorithms, (3D) binary space partition algorithms, among others for example.

[0085] At the scene generator block 609, on the other hand, a scene generator may generate some metadata to be used for rendering and displaying 6 degrees-of-freedom (DoF) media, by a director's intention or a user's preference for example. Such 6 DoF media may include the 360VR like 3D viewing of a scene from rotational changes on 3D axis X, Y, Z in addition to additional dimension allowing for movement front/back, up/down, and left/right with respect to a virtual experience within or at least according to point cloud coded data. The scene description metadata defines one or more scene composed of the coded point cloud data and other media data, including VR360, light field, audio, etc. and may be provided to one or more cloud servers and or file/segment encapsulation/decapsulation processing as indicated in Fig. 6 and related descriptions.

[0086] After video encoding block 604 and image encoding block 605 similar to the video and image encoding described above (and as will be understood, audio encoding also may

be provided as described above), file/segment encapsulation block 606 processes such that the coded point cloud data are composed into a media file for file playback or a sequence of an initialization segment and media segments for streaming according to a particular media container file format such as one or more video container formats and such as may be used with respect to DASH, among others as such descriptions represent exemplary embodiments. The file container also may include the scene description metadata, such as from the scene generator block 1109, into the file or the segments.

[0087] According to exemplary embodiments, the file is encapsulated depending on the scene description metadata to include at least one view position and at least one or more angle views at that/those view position(s) each at one or more times among the 6DoF media such that such file may be transmitted on request depending on user or creator input. Further, according to exemplary embodiments, a segment of such file may include one or more portions of such file such as a portion of that 6DoF media indicating a single viewpoint and angle thereat at one or more times; however, these are merely exemplary embodiments and may be changed depending on various conditions such as network, user, creator capabilities and inputs.

[0088] According to exemplary embodiments, the point cloud data is partitioned into multiple 2D/3D regions, which are independently coded such as at one or more of video encoding block 604 and image encoding block 605. Then, each independently coded partition of point cloud data may encapsulated at file/segment encapsulation block 606 as a track in a file and/or segment. According to exemplary embodiments, each point cloud track and/or a metadata track may include some useful metadata for view-position/angle dependent processing.

[0089] According to exemplary embodiments, the metadata, such as included in a file and/or segment encapsulated with respect to the file/segment encapsulation block, useful for the

view-position/angle dependent processing includes one or more of the following: layout information of 2D/3D partitions with indices, (dynamic) mapping information associating a 3D volume partition with one or more 2D partitions (e.g. any of a tile/tile group/slice/sub-picture), 3D positions of each 3D partition on a 6DoF coordinate system, representative view position/angle lists, selected view position/angle lists corresponding to a 3D volume partition, indices of 2D/3D partitions corresponding to a selected view position/angle, quality (rank) information of each 2D/3D partition, and rendering information of each 2D/3D partition for example depending on each view position/angle. Calling on such metadata when requested, such as by a user of the V-PCC player or as directed by a content creator for the user of the V-PCC player, may allow for more efficient processing with respect to specific portions of the 6DoF media desired with respect to such metadata such that the V-PCC player may deliver higher quality images of focused on portions of the 6DoF media than other portions rather than delivering unused portions of that media.

[0090] From the file/segment encapsulation block 606, the file or one or more segments of the file may be delivered using a delivery mechanism (*e.g.*, by Dynamic Adaptive Streaming over HTTP (DASH)) directly to any of the V-PCC player 625 and a cloud server, such as at the cloud server block 607 at which the cloud server can extract one or more tracks and/or one or more specific 2D/3D partitions from a file and may merge multiple coded point cloud data into one data.

[0091] According to data such as with the position/viewing angle tracking block 608, if the current viewing position and angle(s) is/are defined on a 6DoF coordinate system, at a client system, then the view-position/angle metadata may be delivered, from the file/segment encapsulation block 606 or otherwise processed from the file or segments already at the cloud

server, at cloud server block 607 such that the cloud sever may extract appropriate partition(s) from the store file(s) and merge them (if necessary) depending on the metadata from the client system having the V-PCC player 625 for example, and the extracted data can be delivered to the client, as a file or segments.

[0092] With respect to such data, at the file/segment decapsulation block 615, a file decapsulator processes the file or the received segments and extracts the coded bitstreams and parses the metadata, and at video decoding and image decoding blocks 610 and 611, the coded point cloud data are then decoded into decoded and reconstructed, at point cloud reconstruction block 612, to point cloud data, and the reconstructed point cloud data can be displayed at display block 614 and/or may first be composed depending on one or more various scene descriptions at scene composition block 613 with respect to scene description data according to the scene generator block 609.

[0093] In view of the above, such exemplary V-PCC flow represents advantages with respect to a V-PCC standard including one or more of the described partitioning capabilities for multiple 2D/3D areas, a capability of a compressed domain assembly of coded 2D/3D partitions into a single conformant coded video bitstream, and a bitstream extraction capability of coded 2D/3D of a coded picture into conformant coded bitstreams, where such V-PCC system support is further improved by including container formation for a VVC bitstream to support a mechanism to contain metadata carrying one or more of the above-described metadata.

[0094] In that light and according to exemplary embodiments further described below, the term “mesh” indicates a composition of one or more polygons that describe the surface of a volumetric object. Each polygon is defined by its vertices in 3D space and the information of how the vertices are connected, referred to as connectivity information. Optionally, vertex

attributes, such as colors, normals, etc., could be associated with the mesh vertices. Attributes could also be associated with the surface of the mesh by exploiting mapping information that parameterizes the mesh with 2D attribute maps. Such mapping may be described by a set of parametric coordinates, referred to as UV coordinates or texture coordinates, associated with the mesh vertices. 2D attribute maps are used to store high resolution attribute information such as texture, normals, displacements etc. Such information could be used for various purposes such as texture mapping and shading according to exemplary embodiments.

[0095] Nonetheless, a dynamic mesh sequence may require a large amount of data since it may consist of a significant amount of information changing over time. For example, in contrast to a “static mesh”, or “static mesh sequence,” in which information of that mesh may not change from one frame to another, a “dynamic mesh”, or a “dynamic mesh sequence”, indicates motion in which ones of vertices represented by that mesh change from one frame to another. Therefore, efficient compression technologies are required to store and transmit such contents. Mesh compression standards IC, MESHGRID, FAMC were previously developed by MPEG to address dynamic meshes with constant connectivity and time varying geometry and vertex attributes. However, these standards do not take into account time varying attribute maps and connectivity information. DCC (Digital Content Creation) tools usually generate such dynamic meshes. In counterpart, it is challenging for volumetric acquisition techniques to generate a constant connectivity dynamic mesh, especially under real time constraints. This type of contents is not supported by the existing standards. According to exemplary embodiments herein, there is described aspects of a new mesh compression standards to directly handle dynamic meshes with time varying connectivity information and optionally time varying attribute maps, this standard targets lossy, and lossless compression for various applications, such as real-time

communications, storage, free viewpoint video, AR and VR. Functionalities such as random access and scalable/progressive coding are also considered.

[0096] Fig. 7 represents an example framework 700 of one dynamic mesh compression such as for a 2D atlas sampling based method. Each frame of the input meshes 701 can be preprocessed by a series of operations, e.g., tracking, remeshing, parameterization, voxelization. Note that, these operations can be encoder-only, meaning they might not be part of the decoding process and such possibility may be signaled in metadata by a flag such as indicating 0 for encoder only and 1 for other. After that, one can get the meshes with 2D UV atlases 702, where each vertex of the mesh has one or more associated UV coordinates on the 2D atlas. Then, the meshes can be converted to multiple maps, including the geometry maps and attribute maps, by sampling on the 2D atlas. Then these 2D maps can be coded by video/image codecs, such as HEVC, VVC, AV1, AVS3, etc. On the decoder 703 side, the meshes can be reconstructed from the decoded 2D maps. Any post-processing and filtering can also be applied on the reconstructed meshes 704. Note that other metadata might be signaled to the decoder side for the purpose of 3D mesh reconstruction. Note that the chart boundary information, including the uv and xyz coordinates, of the boundary vertices can be predicted, quantized and entropy coded in the bitstream. The quantization step size can be configured in the encoder side to tradeoff between the quality and the bitrates.

[0097] In some implementations, a 3D mesh can be partitioned into several segments (or patches/charts), one or more 3D mesh segments may be considered to be a “3D mesh” according to exemplary embodiments. Each segment is composed of a set of connected vertices associated with their geometry, attribute, and connectivity information. As illustrated in the example 800 of volumetric data in Fig. 8, the UV parameterization process 802 of mapping from 3D mesh

segments onto 2D charts, such as to the above noted 2D UV atlases 702 block, maps one or more mesh segments 801 onto a 2D chart 803 in the 2D UV atlas 804. Each vertex (v_n) in the mesh segment will be assigned with a 2D UV coordinates in the 2D UV atlas. Note that the vertices (v_n) in a 2D chart form a connected component as their 3D counterpart. The geometry, attribute, and connectivity information of each vertex can be inherited from their 3D counterpart as well. For example, information may be indicated that vertex v_4 connects directly to vertices v_0 , v_5 , v_1 , and v_3 , and similarly information of each of the other vertices may also be likewise indicated. Further, such 2D texture mesh would, according to exemplary embodiments, further indicate information, such as color information, in a patch-by-patch basis such as by patches of each triangle, *e.g.*, v_2 , v_5 , v_3 as one “patch”.

[0098] For example, further to the features of the example 800 of Fig. 8, see the example 900 of Fig. 9 where the 3D mesh segment 801 can be also mapped to multiple separate 2D charts 901 and 902. In this case, a vertex in 3D could corresponds to multiple vertices in 2D UV atlas. As shown in Fig. 9, the same 3D mesh segment is mapped to multiple 2D charts, instead of a single chart as in Fig. 8, in the 2D UV atlas. For example, 3D vertices v_1 and v_4 each have two 2D correspondences $v_1, v_{1'}$, and $v_4, v_{4'}$, respectively. As such, a general 2D UV atlas of a 3D mesh may consist of multiple charts as shown in Figure 14, where each chart may contain multiple (usually more than or equal to 3) vertices associated with their 3D geometry, attribute, and connectivity information.

[0099] Fig. 9 shows an example 903 illustrating a derived triangulation in a chart with boundary vertices $B_0, B_1, B_2, B_3, B_4, B_5, B_6, B_7$. When presented with such information, any triangulation method can be applied to create connectivity among the vertices (including boundary vertices and sampled vertices). For example, for each vertex, find the closest two

vertices. Or for all vertices, continuously generate triangles until a minimum number of triangles is achieved after a set number of tries. As shown in the example 903, there are various regularly shaped, repeating triangles and various oddly shaped triangles, generally closest to the boundary vertices, having their own unique dimensions that may or may not be shared with any other of the triangles. The connectivity information can be also reconstructed by explicit signaling. If a polygon cannot be recovered by implicit rules, the encoder can signal the connectivity information in the bitstream according to exemplary embodiments.

[0100] Boundary vertices $B_0, B_1, B_2, B_3, B_4, B_5, B_6, B_7$ are defined in the 2D UV space.

A boundary edge can be determined by checking if the edge is only appeared in one triangle.

The following information of boundary vertices is significant and should be signaled in the bitstream according to exemplary embodiments: geometry information, *e.g.*, the 3D XYZ coordinates even though currently in the 2D UV parametric form, and the 2D UV coordinates.

[0101] For a case in which a boundary vertex in 3D corresponds to multiple vertices in 2D UV atlas, such as shown in Fig. 9, the mapping from 3D XYZ to 2D UV can be one-to-multiple. Therefore, a UV-to-XYZ (or referred to as UV2XYZ) index can be signaled to indicate the mapping function. UV2XYZ may be a 1D-array of indices that correspond each 2D UV vertex to a 3D XYZ vertex.

[0102] According to exemplary embodiments, to represent a mesh signal efficiently, a subset of the mesh vertices may be coded first, together with the connectivity information among them. In the original mesh, the connection among these vertices may not exist as they are subsampled from the original mesh. There are different ways to signal the connectivity information among the vertices, and such subset is therefore referred to as the base mesh or as base vertices.

[0103] According to exemplary embodiments, a number of methods are implemented for dynamic mesh compression and are part of the above-mentioned edge-based vertex prediction framework, where a base mesh is coded first and then more additional vertices are predicted based on the connectivity information from the edges of the base mesh. Note that they can be applied individually or by any form of combinations.

[0104] For example, consider the vertex grouping for prediction mode example flowchart 1001 of Fig. 10. At S101, vertices inside a mesh may be obtained and can be divided at S102 into different groups for prediction purposes, for example see Fig. 9. In one example, the division is done using the patch/chart partitioning at S104. In another example, the division is done under each patch/chart S105. The decision S103 whether to proceed to S104 or S105 may be signaled by a flag or the like. In the case of S105, several vertices of the same patch/chart form a prediction group and will share the same prediction mode, while several other vertices of the same patch/chart can use another prediction mode. Herein, a “prediction mode” may be considered to be a specific mode that a decoder uses to make a prediction for a video content including the patch, the prediction mode can categorically be divided into intra prediction modes and inter prediction modes, and within each category, there can be different specific modes that the decoder chooses from. According to exemplary embodiments, each group, a “prediction group” may share a same specific mode (e.g., an angular mode at a specific angle) or a same categorical prediction mode (e.g., all intra prediction mode but can be predicted at different angles) according to exemplary embodiments. Such grouping at S106 can be assigned at different levels by determining respective number of vertices involved per group. For example, every 64, 32 or 16 vertices following a scan order inside a patch/chart will be assigned the same prediction mode according to exemplary embodiments and other vertices may be differently

assigned. For each group, a prediction mode can be intra prediction mode or inter prediction mode. This can be signaled or assigned. According to the example flowchart 1000, if a mesh frame or mesh slice is determined to be in intra type at S107, such as by checking whether a flag of that mesh frame or mesh slice indicates an intra type, then all groups of vertices inside that mesh frame or mesh slice shall use intra prediction mode; otherwise, at S108 either intra prediction or inter prediction mode may be chosen per group for all vertices therein.

[0105] Further, for a group of mesh vertices using intra prediction mode, its vertices can only be predicted by using previously coded vertices inside the same sub-partition of the current mesh. Sometimes the sub-partition can be the current mesh itself according to exemplary embodiments, and for a group of mesh vertices using inter prediction mode, its vertices can only be predicted by using previously coded vertices from another mesh frame according to exemplary embodiments. Each of the above-noted information may be determined and signaled by a flag or the like. Said prediction features may occur at S110 and results of said prediction and signaling may occur at S111

[0106] According to exemplary embodiments, for each vertex in a group of vertices in the example flowchart 1000 and in the flowchart 1100 described below, after prediction, the residue will be a 3D displacement vector, indicating the shift from the current vertex to its predictor. The residues of a group of vertices need to be further compressed. In one example, transformation at S111, along with the signaling thereof, can be applied to the residues of a vertex group, before entropy coding. The following methods may be implemented to handle the coding of a group of displacement vectors. For example, in one method, to properly signal the case where a group of displacement vectors, some displacement vectors, or its components have only zero values. In another embodiment, a flag is signaled for each displacement vectors

whether this vector has any non-zero component, and if no, the coding of all components for this displacement vector can be skipped. Further, in another embodiment, a flag is signaled for each group of displacement vectors whether this group has any non-zero vectors, and if no, the coding of all displacement vectors of this group can be skipped. Further, in another embodiment, a flag is signaled for each component of a group of displacement vectors whether this component of the group has any non-zero vectors, and if no, the coding of this component of all displacement vectors of this group can be skipped. Further, in another embodiment, there may be a signaling of the case where a group of displacement vectors, or a component of the group of displacement vectors, needs a transformation, and if not, the transformation can be skipped, and quantization/entropy coding can be directly applied to the group or the group components. Further, in another embodiment, a flag may be signaled for each group of displacement vectors whether this group needs to go through transformation, and if no, the transform coding of all displacement vectors of this group can be skipped. Further, in another embodiment, a flag is signaled for each component of a group of displacement vectors whether this component of the group needs to go through transformation, and if no, the transform coding of this component of all displacement vectors of this group can be skipped. The above-described embodiments in this paragraph, which regard handling of vertex prediction residues, may also be combined and implemented in parallel on different patches respectively.

[0107] Fig. 11 shows the example flowchart 1100 where, at S121 a mesh frame can be obtained coded as an entire data unit, meaning all vertices or attributes of the mesh frame may have correlation among them. Alternatively, depending on a determination at S122, a mesh frame can be divided at S123 into smaller independent sub-partitions, similar in concept to slices or tiles in 2D videos or images. A coded mesh frame or a coded mesh sub-partition can be

assigned with a prediction type at S124. Possible prediction types include intra coded type and inter coded type. For intra coded type, only predictions from the reconstructed parts of the same frame or slice are allowed at S125. On the other hand, an inter prediction type will allow at S125 predictions from a previously coded mesh frame, in addition to intra mesh frame predictions. Further, inter prediction type may be classified with more sub-types such as P type or B type. In P type, only one predictor can be used for prediction purposes, while in B type, two predictors, from two previously coded mesh frames, may be used to generate the predictor. Weighted average of the two predictors can be one example. When the mesh frame is coded as a whole, the frame can be regarded as an intra or inter coded mesh frame. In case of inter mesh frame, P or B type may be further identified via signaling. Or, if a mesh frame is coded with further splitting inside a frame, assign prediction type for each of the sub-partitions occurs at S124. Each of the above-noted information may be determined and signaled by a flag or the like, and like with S110 and S111 of Fig. 10, said prediction features may occur at S126 and results of said prediction and signaling may occur at S127.

[0108] As such, although dynamic mesh sequence may require a large amount of data since it may consist of a significant amount of information changing over time, efficient compression technologies are required to store and transmit such contents, and the herein described features represent such improved efficiencies by allowing at least for improved mesh vertex 3D location prediction by either using previously decoded vertices in the same mesh frame (intra prediction) or from a previous coded mesh frame (inter prediction).

[0109] Further, exemplary embodiments may generate the displacement vectors of a third layer 1303 of a mesh, based on one or more the reconstructed vertices of its previous layer(s) such as a second layer 1302 and a first layer 1301. Assuming the index of the second layer 1302

is T , the predictors for vertices in third layer 1303 $T+1$ are generated based on the reconstructed vertices of at least the current layer or second layer 1302. An example of such layer based prediction structure is shown example 1300 in Fig. 13 which illustrates reconstruction based vertex prediction: progressive vertex prediction using edge-based interpolation, where predictors are generated based on previously decoded vertices, not predictor vertices. The first layer 1301 may be a mesh bounded by a first polygon 1340 having, as vertices thereof, decoded vertices, at boundaries thereof, and interpolated vertices, along ones of lines between ones of those decoded vertices. As the progressive coding proceeds from the first layer 1301 to the second layer 1302, an additional polygon 1341 may be formed by displacement vectors from ones of the interpolated vertices of the first layer to additional vertices of the second layer 1302, and as such, a total number of vertices of the second layer 1302 may be greater than that of the first layer 1301. Likewise, proceeding to the third layer 1303, the additional vertices of the second layer 1302, along with the decoded vertices from the first layer 1301, may then serve in the coding in a similar manner as did the decoded vertices served in proceeding from the first layer 1301 to the second layer 1303; that is, multiple additional polygons may be formed. As note, see the example 1400 in Fig. 14 illustrating such progressive coding where, unlike in Fig. 13, the example 1400 illustrates that, in proceeding from the first layer 1401 to the second layer 1403 and then to the third layer 1403, each of the additionally formed polygons may be entirely within a polygon formed by bounds of the first layer 1401.

[0110] For such example 1300 and/or 1400, see, according to exemplary embodiments the example flowchart 1200 of Fig. 12 where since the interpolated vertices on the current layer are predicted values, such values need to be reconstructed, before being used to generate predictors of vertices on the next layer. This is done by coding a base mesh at S131,

implementing vertices prediction as such at S132, then at S133 adding the decoded displacement vectors of the current layer to the vertex's predictors, such as of layer 1302. Then the reconstructed vertices of this layer together with all decoded vertices of previous layer(s), such as checking for addition vertices values of such layers at S134, can be used to generate and signal the predictor vertices of next layer 1303 at S135. This process can also be summarized as follows: Let $P[t](V_i)$ represent the predictor of vertex V_i on a layer t ; let $R[t](V_i)$ represent the reconstructed vertex V_i on layer t ; let $D[t](V_i)$ represent the displacement vector of vertex V_i on layer t ; let $f(*)$ represent the predictor generator, which, in particular, can be the average of the two existing vertices. Then for each layer t , there is the following according to exemplary embodiments:

$$P[t](V_i) = f(R[s|s < t](V_j), R[m|m < t](V_k)), \text{ where}$$

V_j and V_k are reconstructed vertices of previous layers

$$R[t](V_i) = P[t](V_i) + D[t](V_i) \quad - \text{Eq. (1)}$$

[0111] Then, for all vertices in one mesh frame, divide them into layer 0 (the base mesh), layer 1, layer 2, Etc. Then the reconstruction of vertices on one layer relies on the reconstruction of those on previous layer(s). In the above, each of P , R and D represents a 3D vector under the context of 3D mesh representation. D is the decoded displacement vector, and quantization may or may not apply to this vector.

[0112] According to exemplary embodiments, the vertex prediction using reconstructed vertices may only apply to certain layers. For example, layer 0 and layer 1. For other layers, the vertex prediction can still use neighboring predictor vertices without adding displacement vectors to them for reconstruction. So that these other layers can be processed at the same time without waiting one previous layer to reconstruct. According to exemplary embodiments, for

each layer, whether to choose reconstruction based vertex prediction or predictor based vertex prediction, can be signaled, or the layer (and its subsequent layers) that does not use reconstruction based vertex prediction, can be signaled.

[0113] For the displacement vectors whose vertex predictors are generated by reconstructed vertices, quantization can be applied to them, without further performing transformation, such as wavelet transform, etc. For the displacement vectors whose vertex predictors are generated by other predictor vertices, transformation may be needed and quantization can be applied to the transform coefficients of those displacement vectors.

[0114] As such, since a dynamic mesh sequence may require a large amount of data since it may consist of a significant amount of information changing over time. Therefore, efficient compression technologies are required to store and transmit such contents. In the framework of interpolation-based vertex prediction method described above, one important procedure is to compress the displacement vectors, and this takes up a major part in the coded bitstream, and the focus of this disclosure, and the features this disclosure alleviate such problem by providing for such compression.

[0115] Further, similar to the other examples described above, even with those embodiments, a dynamic mesh sequence may nonetheless require a large amount of data since it may consist of a significant amount of information changing over time, and as such, efficient compression technologies are required to store and transmit such contents. In the framework of 2D atlas sampling based methods indicated above, an important advantage may be achieved by inferring the connectivity information from the sampled vertices plus boundary vertices on decoder side. This is a major part in decoding process, and a focus of further examples described below.

[0116] According to exemplary embodiments, the connectivity information of the base mesh can be inferred (derived) from the decoded boundary vertices and the sampled vertices for each chart on both encoder and decoder sides.

[0117] As similarly described above, any triangulation method can be applied to create connectivity among vertices (including boundary vertices and sampled vertices). According to exemplary embodiments, connectivity types can be signaled in high-level syntax, such as sequence header, slice header.

[0118] As mentioned above, connectivity information can be also reconstructed by explicitly signaling, such as for the irregularly shaped triangle meshes. That is, if it is determined that a polygon cannot be recovered by implicit rules, the encoder can signal the connectivity information in the bitstream. And according to exemplary embodiments, the overhead of such explicit signaling may be reduced depending on the boundaries of polygons.

[0119] According to embodiments, only the connectivity information between boundary vertices and sampled positions is determined to be signaled, while the connectivity information among the sampled positions themselves is inferred.

[0120] Also, in any of the embodiments, the connectivity information may be signaled by prediction, such that only the difference from the inferred connectivity (as prediction) from one mesh to another may be signaled in bitstream.

[0121] As a note, the orientation of inferred triangles (such as to be inferred in a clockwise manner or in a counterclockwise manner per triangle) can be either signaled for all charts in high-level syntax, such as sequence header, slice header, etc., or fixed (assumed) by encoder and decoder according to exemplary embodiments. The orientation of inferred triangles can be also signaled differently for each chart.

[0122] As a further note, any reconstructed mesh may have different connectivity from the original mesh. For example, the original mesh may be a triangle mesh, while the reconstructed mesh may be a polygonal mesh (*e.g.*, quad mesh).

[0123] According to exemplary embodiments, the connectivity information of any base vertices may not be signaled and instead the edges among base vertices may be derived using the same algorithm at both encoder and decoder side. And according to exemplary embodiments, interpolation of predicted vertices for the additional mesh vertices may be based on the derived edges of the base mesh.

[0124] According to exemplary embodiments, a flag may be used to signal whether the connectivity information of the base vertices is to be signaled or derived, and such flag can be signaled at different level of the bitstream, such as at sequences level, frame level, etc.

[0125] According to exemplary embodiments, the edges among the base vertices are first derived using the same algorithm at both encoder and decoder side. Then compared with the original connectivity of the base mesh vertices, the difference between the derived edges and the actual edges will be signaled. Therefore, after decoding the difference, the original connectivity of the base vertices can be restored.

[0126] In one example, for a derived edge, if determined to be wrong when compared to the original edge, such information may signaled in the bitstream (by indicating the pair of vertices that form this edge); and for an original edge, if not derived, may be signaled in the bitstream (by indicating the pair of vertices that form this edge). Further, connectivity on boundary edges and vertex interpolation involving boundary edges may be done separately from the internal vertices and edges.

[0127] Accordingly, by exemplary embodiments described herein, the technical problems noted above may be advantageously improved upon by one or more of these technical solutions. For example, since a dynamic mesh sequence may require a large amount of data since it may consist of a significant amount of information changing over time, and therefore, the exemplary embodiments described herein represent at least efficient compression technologies to store and transmit such contents.

[0128] The above-described embodiments may be further applied to instance-based mesh coding, where an instance may be a mesh of an object or a part of an object. For example, the illustration example 1500 of Fig. 15 illustrates a mesh example 1501 in which various instances 1502 (representing a mesh of a cup), 1503 (representing a mesh of a spoon), and 1504 (representing a mesh of a plate) are present and may be separated and coded respectively. And each of the instances 1501, 1502, 1503, and 1504 are illustrated in respective ones of bounding boxes, but, as a note, it may be considered that the instance 1501 may be illustrated as a bounded by a “mesh-based bounding box” whereas each of instances 1502, 1503, and 1504 may be considered illustrated as bounding by respective ones of an “instance-based bounding box.”

[0129] And further, viewing the example 1600 of Fig. 16 and other figures, consider that, according to embodiments, a “dual degree” mesh coding is a specialized technique aimed at efficiently encoding the connectivity of polygon meshes. Through the application of the mathematical principle of duality, this method encodes connectivity data by constructing two separate sequences: one characterizing the degrees of vertices and the other depicting the degrees of faces (refer to example 1601 for an illustration, which is an example illustration of a dual degree traversal). That is, the encoding process involves simultaneous traversal around both the faces and vertices, and specifically, the traversal commences with an arbitrary seed face, from

which the degree of the face is recorded (e.g., F3 which represents the valance of the vertex and the degree of the face). Subsequently, the degrees of the adjacent vertices are also noted (e.g., V5, V5, V5) as in example 1601. The algorithm then selects what is termed a 'pivot vertex', identified by possessing the minimum degree of freedom - signifying the count of its adjacent faces that have not been traversed. The traversal continues around this pivot vertex, appending new faces and vertices and recording their respective degrees. To address unique scenarios involving the split or merge of vertices, supplementary symbols are employed according to embodiments. And while such design highlights the impressive efficiency of Dual Degree mesh coding for polygonal meshes, including those with high irregularities or in worst-case scenarios, the performance of the Dual Degree coding is largely contingent on the regularity of face degrees and vertex valences.

[0130] In terms of position attributes coding, in addition to connectivity, a mesh usually comprises other attributes such as vertex positions, texture coordinates, normal vectors, and associated texture maps. And in that light, consider that connectivity often constitutes only a small portion of the mesh data and may contribute less than 10% or even 1% of the total bitstream while position attributes could contribute to more than half of the bitstream.

[0131] And in terms of position prediction with a parallelogram, see the example 1602, where among the attributes, the 3D positions of vertices typically account for the majority of the bits required for geometric attributes. To efficiently encode vertex positions, predictive coding schemes are employed according to embodiments, and one notable example being parallelogram prediction. In the context of polygonal meshes, parallelogram prediction has been found to perform optimally with quadrilateral meshes.

[0132] That is, the example 1602 represents an illustration of an “across” and a “within” parallelogram prediction for vertex coding in polygonal mesh with A, B, and C being three reference positions for either of those parallelogram predictions. According to embodiments, for a given polygonal mesh, the position of a vertex to be predicted (V) utilizes three previously encoded vertices (A, B, C) as references to estimate its position (V), as in example 1602, as follows:

$$V = w_1A + w_2B + w_3C, \quad - \text{Eq. (2)}$$

where weighted factor are often chosen as $w_1 = w_3 = 1, w_2 = -1$. These weights can be further modified to adapt to different polygonal structures. Within the context of polygonal meshes, parallelogram prediction can be categorized into two types: “within prediction” and “across prediction”, as depicted in example 1602. In “within prediction”, all three reference vertices are situated within the same face, whereas, in “across prediction”, vertex C is obtained from the opposite face.

[0133] And according to embodiments, see also the example 1603 regarding position prediction with multi-parallelogram. That is, as an extension of the parallelogram prediction, known as multi-parallelogram prediction, has been introduced to enhance position prediction for triangular meshes. As depicted in example 1603, multi-parallelogram prediction “a” employs the average position derived from two or more parallelogram predictions when feasible. And prediction “b” illustrates an instance of two-parallelogram prediction. The number of available prediction candidates dictates how many predictors are averaged.

[0134] The proposed methods may be used separately or combined in any order. Further, the proposed methods may be implemented by processing circuitry (e.g., one or more processors or one or more integrated circuits). While some methods explicitly applied for triangle mesh, it

also be used for arbitrary polygonal mesh. In this disclosure, the proposed a method to adaptively merge triangle face to quad face following the traversal order to create within prediction for parallelogram prediction.

[0135] For example, example 1604 illustrates an adaptive parallelogram prediction, where, in the context of a triangular mesh, as in example 1604, and following the traversal order with pivot vertex A, the next face for traversal is F_0 , and vertex to be encoded is D.

[0136] The parallelogram prediction, according to such embodiments in view of example 1604, uses reference vertices A, B, and C_0 to predict D. Although vertices A and B are in the same face (F_0) as D, vertex C_0 is chosen from the opposite face (F_1) over edge A-B, as shown by C_1 in example 1604. However, in many cases, this prediction may not be accurate, especially if the quadrilateral face formed by vertices A, C, B, and D is concave or if the position of vertex D is irregular.

[0137] And to address this issue, embodiments herein provide an alternative reference vertex, C_1 , where C_1 is the vertex connected to A in the face opposite to F_1 (namely face F_2 in example 1604) over edge A- C_1 . In this scenario, the encoder can select either C_0 or C_1 as the reference to predict D according to at least any of the following embodiments.

[0138] In one embodiment, the selection criteria with respect to example 1604 for reference C is based on the Sum of Absolute Differences (SAD) of the residual vector between the predicted position $P_i = A + B - C_i$, and the position to be encoded, calculated as $R_i = C_i - P_i$, where $i = 0, 1$.

[0139] In another embodiment, the selection with respect to example 1604 between C_0 or C_1 is based on the estimated bits required to encode the residual, termed $\text{bitCost}(R_i)$. This cost calculation also includes bits for signaling the adaptive flags.

[0140] In a further embodiment, with respect to example 1604, an additional reference vertex, C_2 , is chosen from the face opposite to F_1 over edge $B-C_0$.

[0141] In yet another embodiment, with respect to example 1604, the number of reference vertices for predicting D is expanded to include all neighboring vertices to A and B , which increases the signaling bits.

[0142] In yet another embodiment, with respect to example 1604, the number of reference vertices for predicting D is expanded to include first N available neighboring vertices to A and B , where N can be either assigned at a header syntax (such as frame level or sequence level headers) or it can be assumed by both encoder and decoder. Assuming the all available neighboring vertices is larger than the number N , the first N in a given order will be used as candidates, to reduce signaling cost.

[0143] This disclosure also provides further for signaling of adaptive reference selection, including signaling adaptive reference using flags and hiding adaptive reference flag by modifying face degrees.

[0144] For signaling adaptive reference using flags according to embodiments, to indicate which reference vertex C (either C_0 or C_1) is being used, a straightforward approach is to signal a binary flag for each face. Specifically, each face will have a bit that indicates whether vertex C_0 (flag value 0) or vertex C_1 (flag value 1) is used. Binary arithmetic coding is employed to encode these flags according to embodiments. And, for at least further optimization of the entropy coding of the adaptive reference flags, the context of flags for faces adjacent to F_1 is utilized to estimate the probability of the current adaptive reference flag.

[0145] And for hiding adaptive reference flag by modifying face degrees, given that C_0 is often preferred in vertex prediction, signaling the adaptive reference flag can create significant

bit overhead. To address this, embodiments herein employ concealing the adaptive reference flag using a simple face-merging algorithm. That is, for example for triangle mesh, the face degree is always 3. At the encoder side, the traversal follows the dual-degree connectivity coding, and modified the face degree to $3+i$ with i denotes the corresponding prediction mode if and only if $i > 0$. At decoder side, the vertex predictor will pick up corresponding reference based on the current face degree.

[0146] In addition, for hiding adaptive reference flag by modifying face degrees, embodiments herein may constrain the only one reference flag for each face to ensure the decodability. And by doing so, such embodiments avoid signaling the most common case (i.e., $i=0$) and conceal other adaptive reference flag within the face symbol of dual-degree connectivity coding. And as such, it results in an increase in bits for connectivity but saves bits for position coding. And, according to embodiments, where more than two reference modes need to be handled, the face degree can be increased correspondingly. Less preferable reference selections are added at a higher cost. For instance, if the reference candidate list includes C_0 , C_1 , and C_2 , then the degrees will be incremented by +0, +1, and +2 respectively.

[0147] Embodiments herein also provide for adaptive reference of reference vertices A and B for triangle mesh. For example, the illustrations 1700 include an example 1701, where, according to such embodiments, reference vertices A and B are adaptively selected, while vertex C is consistently chosen from the vertex in the opposite face. When encoding vertex D with the current pivot A, the first edge A-B is defined. Subsequently, neighboring edges in the active edge list, A-A1 and B-B1, are also used. Similar to example 1604, the adaptive reference flag can either be signaled through the bitstream or concealed by incrementing the face degree of the current face.

[0148] Embodiments herein also provide for adaptive selection of reference vertices A, B, and C for triangle mesh with reduced candidate list where aspects of examples 1604 and 1701 are integrated to signal a set of reference vertices A, B, and C to pinpoint the optimal prediction. Consequently, this significantly increases the number of predictions. To enhance efficiency, such embodiments employ a reduced reference list. For example, for edge A-B, only two references, C_0 and C_1 from example 1604, are utilized. Two additional references, A-A₁ and B-B₁, are included if they form an angle greater than 180 degrees. For instance, if $\text{Angle}(A_1-A, A-B) > 180$ degrees, then A₁-A is considered as a candidate. The same condition is applied to B-B₁.

[0149] And embodiments herein also provide for adaptive vertex prediction for polygonal mesh, as in example 1702 which shows an adaptive vertex prediction for polygonal mesh with $N = 5$, where in the case of higher-order polygonal meshes, parallelogram prediction differs slightly from triangular meshes due to the availability of two possible references in the opposite faces: vertex C_1 adjacent to vertex A, and vertex C_3 adjacent to vertex B, as illustrated in example 1702. Generally, if the predicted vertex D is adjacent to vertex B, then vertex C adjacent to A (in this case, C_1) should be selected as the reference to maximize diagonal prediction for regular meshes. However, this approach is not effective for irregular polygonal meshes.

[0150] In that light, for cross-vertex prediction, there are more reference vertices to choose from in the opposite face. Since C_1 is likely to be the best reference, it is the default prediction for an polygonal face. And for N-polygonal face, additional reference vertices $C_2, \dots, C_{(n-2)}$ are signaled by modifying the face degree to $N+1, \dots, 2N-3$ respectively. And for an arbitrary polygonal mesh, the maximum face degree of d_{max} is signaled through the bitstream.

The reference mode is hidden in face degree by modify face degree of the current face to $d = 2d_{max} - d + i$ if the best reference mode i is greater than 0.

[0151] And as such, by embodiments, problems with dual degree mesh coding, that is heavily reliant on the regularity of edge valence and face degree and does not account for attribute compression, are addressed. Further, embodiments herein represent improvements in light of considering that parallelogram prediction exhibits subpar performance when applied to irregular meshes, that connectivity account very little bitstream size, that in multi-parallelogram prediction, averaging the predictors may not yield the optimal predictor, and that there is a scarcity of efficient position coding methods tailored for polygonal meshes.

[0152] That is, embodiments herein disclose a novel method for efficiently compressing the connectivity and attribute information of triangular and polygonal meshes by employing an optimized dual degree connectivity coding scheme. The method significantly reduces the bit rate required for encoding both the connectivity structure and the geometric attributes of the meshes, thereby achieving a more compact representation.

[0153] And according to further embodiments, there is also position attribute refinement features, such as where, in lossy mesh compression, position attribute refinement is often used to adjust the vertex after decimation. That is, the vertex is moving around an initial position of that vertex to better approximate the mesh surface. However, the usage in V-DMC only consider to geometric distortion and does not pay attention to the rate of encoding vertex position. And as such, even with the above-described embodiments, there may still be problems since dual degree mesh coding is heavily reliant on the regularity of edge valence and face degree, and does not account for attribute compression, and there is a scarcity of efficient lossy position coding

methods tailored for polygonal meshes, and there may need to be consideration of vertex position rate while refining vertex position.

[0154] The proposed methods may be used separately or combined in any order for arbitrary polygonal mesh. Further, the proposed methods may be implemented by processing circuitry (e.g., one or more processors or one or more integrated circuits).

[0155] The example 1800 of Fig. 18 relates to embodiments involving refine position for vertex coding. That is, as with example 1800, connectivity and position coding are compressed in an interleaved way in dual degree manner. That is, embodiments follow the visited vertex order to refine the input vertex position minimize the residual vector $r = p - v$, while maintaining high approximation fidelity to the original surface, where p is the predictor, v is the vertex position. The predictor can utilize various methods, including but not limited to, single or multiple parallelogram prediction techniques.

[0156] In general, the search space according to embodiments is represented as a sphere centered at the predicted vertex. Within this search space, the best replacement vertex is selected with a minimum rate distortion cost as

$$J = D + \lambda R, \quad - \text{Eq. (3)}$$

where R signifies the estimated bit rate necessary for encoding the residual vector, and D represents the geometric distortion introduced due to the adjustment of the vertex V . The parameter λ serves as a weighting factor to regulate the trade-off between geometric distortion (D) and bit-rate efficiency (R).

[0157] And as depicted in Fig. 18, the optimization process aims to identify a replacement vertex closer proximity to the predicted vertex, while constraint the distortion. Therefore, it can be represented by two spheres, as illustrated, one centered at the predicted

vertex (denoted as $\mathcal{S}_R$) to constraint the bit rate, and another sphere centered around the current vertex (denoted as $\mathcal{S}_D$) to constraint the geometric distortion. The intersection of these two spheres forms the search space. And to provide flexibility in controlling the search space and prioritizing either geometric fidelity or bit rate efficiency, the radius of spheres are defined as r_R, r_D accordingly. And as such, Fig. 18 represents an example illustration of refining position for current vertex during traversal according to embodiments.

[0158] That is, there is also provided herein embodiments that reduce search space and provide for simplified rate estimation. For example, according to embodiments herein, the complexity of cost calculation and rate estimation is reduced by selectively evaluating positions lying on the line segment connecting the predicted and current vertices. This approach is based on the observation that positions along this line are likely to have a reduced Sum of Absolute Differences (SAD) in residual prediction, and hence significantly narrows the search space.

[0159] More specifically, in an embodiment, the rate is estimated as a scaled SAD of residual vector. That is $R = \alpha \times |p - v|_1$, where $\alpha > 0$ is a scaling factor to adjust the rate approximation.

[0160] And to further streamline the search for a suitable replacement vertex, especially in cases involving large residual vectors, specific positions along the aforementioned line segment are chosen as candidates. These positions, denoted as $p_0, p_{1/4}, p_{1/3}, \dots, p_1$, represent fractions (0, 1/4, 1/3, 1/2, 2/3, 3/4, 1) of the line segment and are calculated using the formula

$$p_w = (1 - w) * v + p. \quad - \text{Eq. (4)}$$

Where p_w represents the position at fraction w of the line segment, v is the current vertex and p is the predicted vertex. Thus, p_0 corresponds to the current vertex v , and p_1 corresponds to the predicted vertex p .

[0161] And to determine the optimal replacement vertex, the costs associated with each candidate position are compared using the change in the rate-distortion function, denoted as ΔJ . The expression for ΔJ is given by:

$$\Delta J = \Delta D + \lambda \Delta R \quad - \text{Eq. (5)}$$

where ΔD represents the change in geometric distortion and ΔR denotes the change in bit rate. The embodiment may employ further simplifications for the calculation of ΔD and ΔR , facilitating a more efficient evaluation process. This streamlined approach enables the selection of a replacement vertex that strikes a balance between geometric fidelity and bit rate efficiency, while also reducing computational overhead.

[0162] And for rate estimation according to embodiments, given a residual vector r , the bit rate to encode it is estimated by four components: a sign bit ($Sign$), a bit indicates the magnitude is greater 1 ($gt1$), and a bit for greater than 2 ($gt2$), and the remainder bits necessary for encoding the magnitude $- 2$ as

$$R = Sign(r) + gt1(r) + gt2 + rem. \quad - \text{Eq. (6)}$$

The rate different are represented by

$$\Delta R = \Delta Sign + \Delta gt1 + \Delta gt2 + \Delta rem \quad - \text{Eq. (7)}$$

[0163] However, since the vertices are aligned on the same line, the residual vectors consistently have the same sign, rendering $\Delta Sign$ equal to 0. Additionally, in many instances where the distance is substantial, the bits $gt1$ and $gt2$ will also remain unchanged, i.e., $\Delta gt1 = 0$ and $\Delta gt2 = 0$. Consequently, only the remainder needs to be estimated. For estimating the remainder, a simplified k -truncated Golomb coding is employed, with the same truncation value. Therefore, the change in remainder is approximately:

$$R = \Delta rem \approx modulo((rem_i - rem_j), K) \quad - \text{Eq. (8)}$$

[0164] And for distortion, according to embodiments, the distortion arising from the modified vertex 'V' can be approximated using a quadratic error term, similar to the methods found in mesh simplification algorithms. Alternatively, conventional metrics such as D1-PSNR (Peak Signal-to-Noise Ratio with D1 distance) can also be employed as a measure of distortion according to other embodiments.

[0165] And also, there is provided early termination with binary search features according to embodiments, where to further streamline the complexity associated with rate estimation, the methods according to embodiments herein incorporate an iterative binary search algorithm with an early termination criterion.

[0166] For example, see Fig. 19 showing flowchart 1900 where, as a step 1, see S1901 where initially, the costs of positions p_0 and $p_{1/2}$ are estimated. If p_0 yields a lower cost at S1902, then, at S1903, the algorithm proceeds to evaluate position $p_{1/4}$; otherwise, at S1904, position $p_{3/4}$ is examined. And as a step 2, at S1905, the difference in delta rate between consecutive iterations is assessed. If this difference is below a certain threshold, then at S1906, the search is terminated early. This two-step process efficiently narrows down the search space while reducing the computational burden by avoiding unnecessary evaluations.

[0167] According to embodiments, there is a two-step search where, an initial binary search is conducted for 'M' iterations to locate an approximate solution. Subsequently, an exhaustive search is executed in the vicinity of the position obtained from the binary search. The step size for this exhaustive search is set to $1/s$ times the step size of the final iteration in the binary search. And this two-step search strategy ensures a more accurate and refined selection of the replacement vertex by combining the efficiency of binary search to quickly converge to a neighborhood, followed by an exhaustive search within that neighborhood for fine-tuning.

Through this combination, the method effectively maintains a balance between computational efficiency and the accuracy of the selected vertex.

[0168] As such, by embodiments herein, there is provided features for efficiently compressing the position attribute with refining the input mesh.

[0169] Further, embodiments herein also additionally provide for multi-parallelogram prediction which uses the average position given by two or more parallelogram predictions whenever it is possible. For example, for the current position to be encoded, a neighbor's faces and vertices are checked for availability (i.e. already encoded). And if so, then such embodiments use up to 4 parallelogram predictions and then averaged out. See example 2001, in the illustrations 2000 of Fig. 20, which shows an example of three parallelogram prediction. The final prediction is:

$$P = \frac{1}{n} (P_1 + \dots + P_n). \quad - \text{Eq. (9)}$$

[0170] But as with other features described herein, there are technical problem in even multi-parallelogram coding such as that interleave coding of connectivity and position could bring potential coding gain. It was only briefly mentioned in previous work, that position information is only needed at the time of selecting pivot, that multiple parallelograms are only designed for triangle mesh with extension for higher polygonal mesh being desired, that prediction are not equally important, therefore, equal weight is not optimal, and that multi-parallelogram needs to be extended adjust to fit the dual degree coding due to the restriction of traversal path.

[0171] Embodiments herein are applied to any position attribute compression algorithm and improve the parallelogram features by checking the reliability of the prediction based on

extending the method to multi-parallelogram prediction for dual degree then optimized prediction order for the mesh with multiple connected components.

[0172] For example, there is interleave coding of connectivity and position features according to exemplary embodiments. In general, connectivity and attribute could be encoded separately in which connectivity is often been encoded first then position and other attributes. However, there are benefits for interleave encoding of connectivity and other attributes like position in improving compressing efficiency. Firstly, as the position information are available at the traversal time, geometry structure of input mesh can be used to guide the traversal path to reduce the merge/split symbol in connectivity coding and/or select the traversal path favorable for position coding.

[0173] And as such, see example 2002 of interleave traversal and position coding where there is illustrated a method for interleave coding for polygonal mesh according to exemplary embodiments. In interleave coding of connectivity and position, a face is marked as traversed. In this approach, position is encoded following the exact traversal order. As example in example 2002, for “face traversal”, when f_0 is traversed, v_0, v_1, v_3 are added sequentially. For pivot traversal at v_0 , subsequence faces are added f_1, f_2, f_3 and corresponding vertex in order of v_3, v_4, v_5, v_6 . The position is encoded immediately after each vertex is traversed following the exactly traversal order.

[0174] And, according to embodiments, there is also to delay the position coding until traversal is completed for the current pivot. By doing so, such features give a completed connectivity information of the current pivot for encoding position without compromise the ability to select pivots based on position information. And in this case, the order of encoding

position could be adaptive for efficient encoding. And according to embodiments, there is an adaptive prediction order.

[0175] According to embodiments there is also multi-parallelogram prediction in dual degree mesh compression for triangle meshes. For example, see example 2003 of such multi-parallelogram prediction in dual degree with interleave connectivity and position coding where, in case of separate connectivity and position coding, multi-parallelogram for triangle mesh is performed similar to other embodiments described above for example. That is, visit all the neighbor face of the current vertex and select the one with valid parallelogram prediction. Up to 4 parallelogram predictors can be selected according to embodiments. And the final predictor is the average between them. However, due to the simultaneous encoding scheme of connectivity and position, connectivity information may not be available at decoder time, especially for multi-parallelogram prediction.

[0176] And as such, according to embodiments there is provided a method to derived multi-parallelogram for interleave dual degree coding is presented for triangle mesh. For example, because the nature of dual degree traversal, the lowest free degree vertex will be selected as the next pivot. As a result, pivot vertex often have a free degree of 0, 1, or 2. Therefore, when connectivity and position are encoded simultaneously, there are only maximum 2 parallelogram available for prediction. This happens for the last position to be encoded in the current pivot traversal, e.g., see example 2003 . The method to derived multiple parallelograms are given as follow.

[0177] That is, in embodiments, the mechanism to select reference parallelogram for dual degree coding with interleave position and connectivity such that a mean for current vertex V , faces f_0 - f_3 are already visited but not f_4 . And for the current pivot traversal at B , with degree d_B ,

the last position to be visit and encode is V. Faces f_0 -3 are already traversed and f_4 will be traversed after completed encoding V.

[0178] And for a first parallelogram, for the current position V (position to be encoded) as the last vertex to be encoded in the current pivot traversal. The first parallelogram is the common edge of the current face, which is A-B and position C from the corresponding opposite face according to embodiments.

[0179] And for a second parallelogram, note that, in this case B is also the pivot vertex. The valence of B is the number of neighbor face denoted as d_B , which is 5 in example 2003. Then, we traversal in clockwise order with $d_B - 2$ face. In example 2003, it is traversal back from face f_0 to f_3 , and extract the corresponding second parallelogram prediction, that is common edge.

[0180] And for further **parallelograms**, since the connectivity of target vertex V is unknown, it is not straight forward to find other neighbor face. However, it is possible to derived third and fourth parallelogram predictor by checking derived connectivity according to embodiments as follows.

[0181] For example, firstly, V is known to connect to {A, B, D} vertices. Based on the degree of freedom on connected vertex to V to derive new connection.

[0182] For *0 degree of freedom*, like B, the encoding is completed, and there is no derived connectivity.

[0183] For *1 degree of freedom*, like D, one face is to be encoded, as a triangle, it is expected to be V-D-H. In this case, V is derived to connect to H, and additional parallelogram prediction is found as H-D edge.

[0184] And for 2 or higher degree of freedom, there is high uncertainty on the connectivity, and thus no periodogram or parallelogram is found.

[0185] And for multi-parallelogram prediction in dual degree mesh compression for polygonal meshes, there is provided, as in example 2004, which is an example of multi-parallelogram prediction for polygonal mesh based on a same edge, according to exemplary embodiments, multi-parallelogram prediction for polygonal mesh is presented based on the same edges polygonal structural. For example, the nearest common edge for the current face is B-A. Then all the triangulated sets are used for parallelogram prediction. For example, A-B-C1, A-B-C2, A-B-C3 are being used as multiple parallelogram predictor for vertex V. And as A-B-C1 is the default parallelogram predictor in polygonal mesh. In one or more embodiments, a triangle with its area overlap to A-B-C1 triangle greater than a given threshold is used as parallelogram predictor. For example, A-B-C2 is used but not A-B-C2.

[0186] As such, embodiments herein provide a method to interleave coding of connectivity and position attribute in dual degree mesh coding and further improve position coding for dual degree mesh coding based on adaptive multiple parallelogram prediction.

[0187] The techniques described above, can be implemented as computer software using computer-readable instructions and physically stored in one or more computer-readable media or by a specifically configured one or more hardware processors. For example, FIG. 21 shows a computer system 2100 suitable for implementing certain embodiments of the disclosed subject matter.

[0188] 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 computer central processing units (CPUs), Graphics Processing Units (GPUs), and the like.

[0189] 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.

[0190] The components shown in FIG. 21 for computer system 2100 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 2100.

[0191] Computer system 2100 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).

[0192] Input human interface devices may include one or more of (only one of each depicted): keyboard 2101, mouse 2102, trackpad 2103, touch screen 2110, joystick 2105, microphone 2106, scanner 2108, camera 2107.

[0193] Computer system 2100 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 2110, or joystick 2105, but there can also be tactile feedback devices that do not serve as input devices), audio output devices (such as: speakers 2109, headphones (not depicted)), visual output devices (such as screens 2110 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).

[0194] Computer system 2100 can also include human accessible storage devices and their associated media such as optical media including CD/DVD ROM/RW 2120 with CD/DVD 2111 or the like media, thumb-drive 2122, removable hard drive or solid state drive 2123, 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.

[0195] 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.

[0196] Computer system 2100 can also include interface 2199 to one or more communication networks 2198. Networks 2198 can for example be wireless, wireline, optical. Networks 2198 can further be local, wide-area, metropolitan, vehicular and industrial, real-time, delay-tolerant, and so on. Examples of networks 2198 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 2198 commonly require external network interface adapters that attached to certain general-purpose data ports or peripheral buses (2150 and 2151) (such as, for example USB ports of the computer system 2100; others are commonly integrated into the core of the computer system 2100 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 2198, computer system 2100 can communicate with other entities. Such communication can be uni-directional, receive only (for example, broadcast TV), uni-directional send-only (for example CANbusto 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.

[0197] Aforementioned human interface devices, human-accessible storage devices, and network interfaces can be attached to a core 2140 of the computer system 2100.

[0198] The core 2140 can include one or more Central Processing Units (CPU) 2141, Graphics Processing Units (GPU) 2142, a graphics adapter 2117, specialized programmable processing units in the form of Field Programmable Gate Areas (FPGA) 2143, hardware accelerators for certain tasks 2144, and so forth. These devices, along with Read-only memory (ROM) 2145, Random-access memory 2146, internal mass storage such as internal non-user accessible hard drives, SSDs, and the like 2147, may be connected through a system bus 2148. In some computer systems, the system bus 2148 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 2148, or through a peripheral bus 2149. Architectures for a peripheral bus include PCI, USB, and the like.

[0199] CPUs 2141, GPUs 2142, FPGAs 2143, and accelerators 2144 can execute certain instructions that, in combination, can make up the aforementioned computer code. That computer code can be stored in ROM 2145 or RAM 2146. Transitional data can be also be stored in RAM 2146, whereas permanent data can be stored for example, in the internal mass storage 2147. Fast storage and retrieval 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 2141, GPU 2142, mass storage 2147, ROM 2145, RAM 2146, and the like.

[0200] 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.

[0201] As an example and not by way of limitation, the computer system having architecture 2100, and specifically the core 2140 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 2140 that are of non-transitory nature, such as core-internal mass storage 2147 or ROM 2145. The software implementing various embodiments of the present disclosure can be stored in such devices and executed by core 2140. A computer-readable medium can include one or more memory devices or chips, according to particular needs. The software can cause the core 2140 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 2146 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 2144), 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.

[0202] While this disclosure has described several exemplary embodiments, there are alterations, permutations, and various substitute equivalents, which fall within the scope of the disclosure. It will thus be appreciated that those skilled in the art will be able to devise numerous systems and methods which, although not explicitly shown or described herein, embody the principles of the disclosure and are thus within the spirit and scope thereof.

WHAT IS CLAIMED IS:

1. A method for video decoding, the method performed by at least one processor and comprising:

obtaining, from a bitstream, a mesh representing an encoded volumetric data of at least one three-dimensional (3D) visual content;

partitioning a plurality of vertices of the mesh into a plurality of groups; and

decoding the encoded volumetric data by predicting the vertices in each group of the plurality of groups based on a plurality of traversal orders depending on an adaptive reference vertex of the vertices,

wherein the plurality of traversal orders comprises a first order from the adaptive reference vertex to a first alternative reference vertex of the vertices, and

wherein the plurality of traversal orders comprises a second order from the adaptive reference vertices to a second alternative reference vertex of the vertices.

2. The method according to claim 1,

wherein a binary flag is included in the bitstream and indicates either the first order or the second order.

3. The method according to claim 1,

wherein decoding the encoded volumetric data is based on determining at least one of the plurality of traversal orders based on a number of faces of the mesh and without any binary flag included in the bitstream and indicating any of the first order and the second order.

4. The method according to claim 1,
wherein content of the bitstream is based on a rate estimated from a sum of absolute differences (SAD) of a residual vector.
5. The method according to claim 4,
wherein the content of the bitstream represents a replacement vertex of the adaptive vertex and is determined based on determining a plurality of fractions of a line segment of a face of the mesh.
6. The method according to claim 1,
wherein decoding the encoded volumetric data is based on interleave coding indicating at least one of face traversal and pivot traversal of the mesh.
7. The method according to claim 1,
wherein the mesh comprises a parallelogram.
8. An apparatus for video decoding, the apparatus comprising:
at least one memory configured to store computer program code;
at least one processor configured to access the computer program code and operate as instructed by the computer program code, the computer program code including:
obtaining code configured to cause the at least one processor to obtain, from a bitstream, a mesh representing an encoded volumetric data of at least one three-dimensional (3D) visual content;

partitioning code configured to cause the at least one processor to partition a plurality of vertices of the mesh into a plurality of groups; and

decoding code configured to cause the at least one processor to decode the encoded volumetric data by predicting the vertices in each group of the plurality of groups based on a plurality of traversal orders depending on an adaptive reference vertex of the vertices,

wherein the plurality of traversal orders comprises a first order from the adaptive reference vertex to a first alternative reference vertex of the vertices, and

wherein the plurality of traversal orders comprises a second order from the adaptive reference vertices to a second alternative reference vertex of the vertices.

9. The apparatus according to claim 8,

wherein a binary flag is included in the bitstream and indicates either the first order or the second order.

10. The apparatus according to claim 8,

wherein decoding the encoded volumetric data is based on determining at least one of the plurality of traversal orders based on a number of faces of the mesh and without any binary flag included in the bitstream and indicating any of the first order and the second order.

11. The apparatus according to claim 8,

wherein content of the bitstream is based on a rate estimated from a sum of absolute differences (SAD) of a residual vector.

12. The apparatus according to claim 11,
wherein the content of the bitstream represents a replacement vertex of the adaptive vertex and is determined based on determining a plurality of fractions of a line segment of a face of the mesh.

13. The apparatus according to claim 8,
wherein decoding the encoded volumetric data is based on interleave coding indicating at least one of face traversal and pivot traversal of the mesh.

14. The apparatus according to claim 8,
wherein the mesh comprises a parallelogram.

15. A non-transitory computer readable medium storing a program causing a computer to:
obtaining, from a bitstream, a mesh representing an encoded volumetric data of at least one three-dimensional (3D) visual content;
partitioning a plurality of vertices of the mesh into a plurality of groups; and
decoding the encoded volumetric data by predicting the vertices in each group of the plurality of groups based on a plurality of traversal orders depending on an adaptive reference vertex of the vertices,
wherein the plurality of traversal orders comprises a first order from the adaptive reference vertex to a first alternative reference vertex of the vertices, and

wherein the plurality of traversal orders comprises a second order from the adaptive reference vertices to a second alternative reference vertex of the vertices.

16. The non-transitory computer readable medium according to claim 15, wherein a binary flag is included in the bitstream and indicates either the first order or the second order.

17. The non-transitory computer readable medium according to claim 15, wherein decoding the encoded volumetric data is based on determining at least one of the plurality of traversal orders based on a number of faces of the mesh and without any binary flag included in the bitstream and indicating any of the first order and the second order.

18. The non-transitory computer readable medium according to claim 15, wherein content of the bitstream is based on a rate estimated from a sum of absolute differences (SAD) of a residual vector.

19. The non-transitory computer readable medium according to claim 18, wherein the content of the bitstream represents a replacement vertex of the adaptive vertex and is determined based on determining a plurality of fractions of a line segment of a face of the mesh.

20. The non-transitory computer readable medium according to claim 15,

wherein decoding the encoded volumetric data is based on interleave coding indicating at least one of face traversal and pivot traversal of the mesh.

FIG. 1

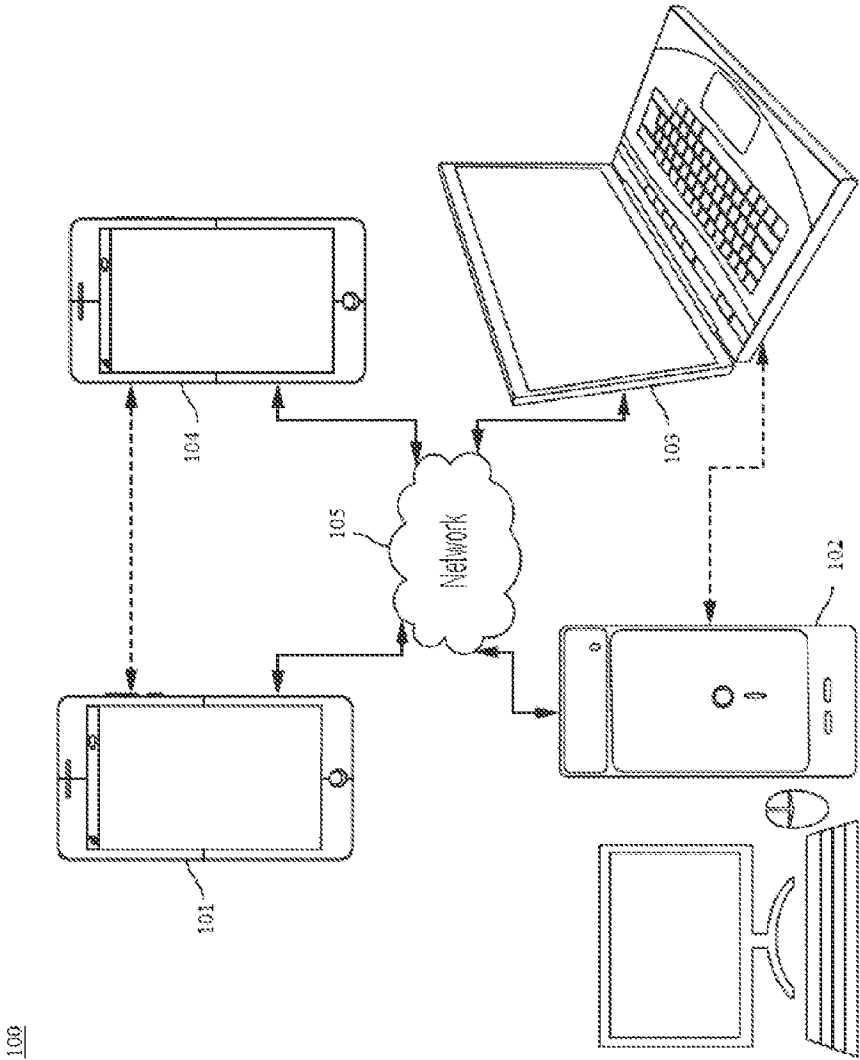


FIG. 2

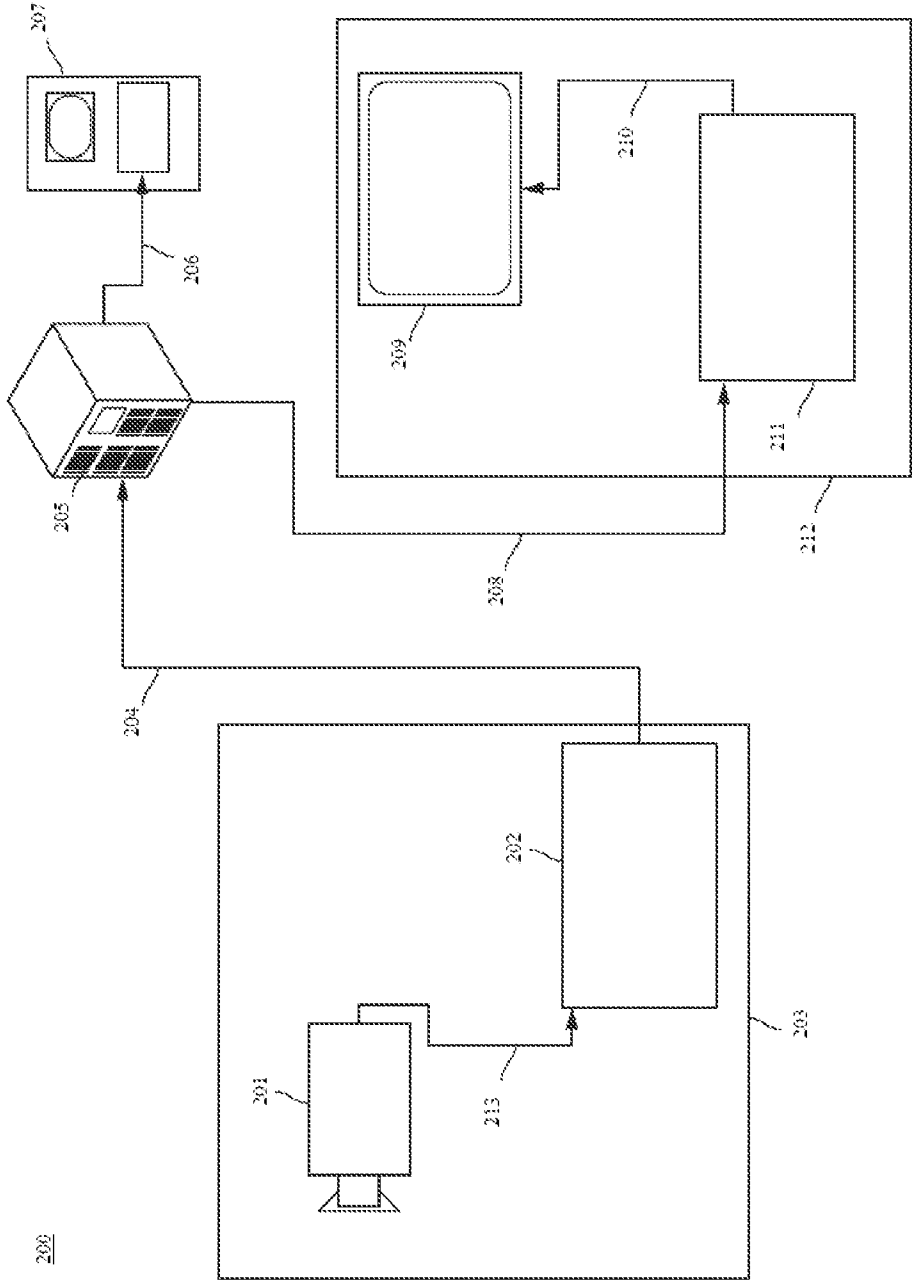


FIG. 3

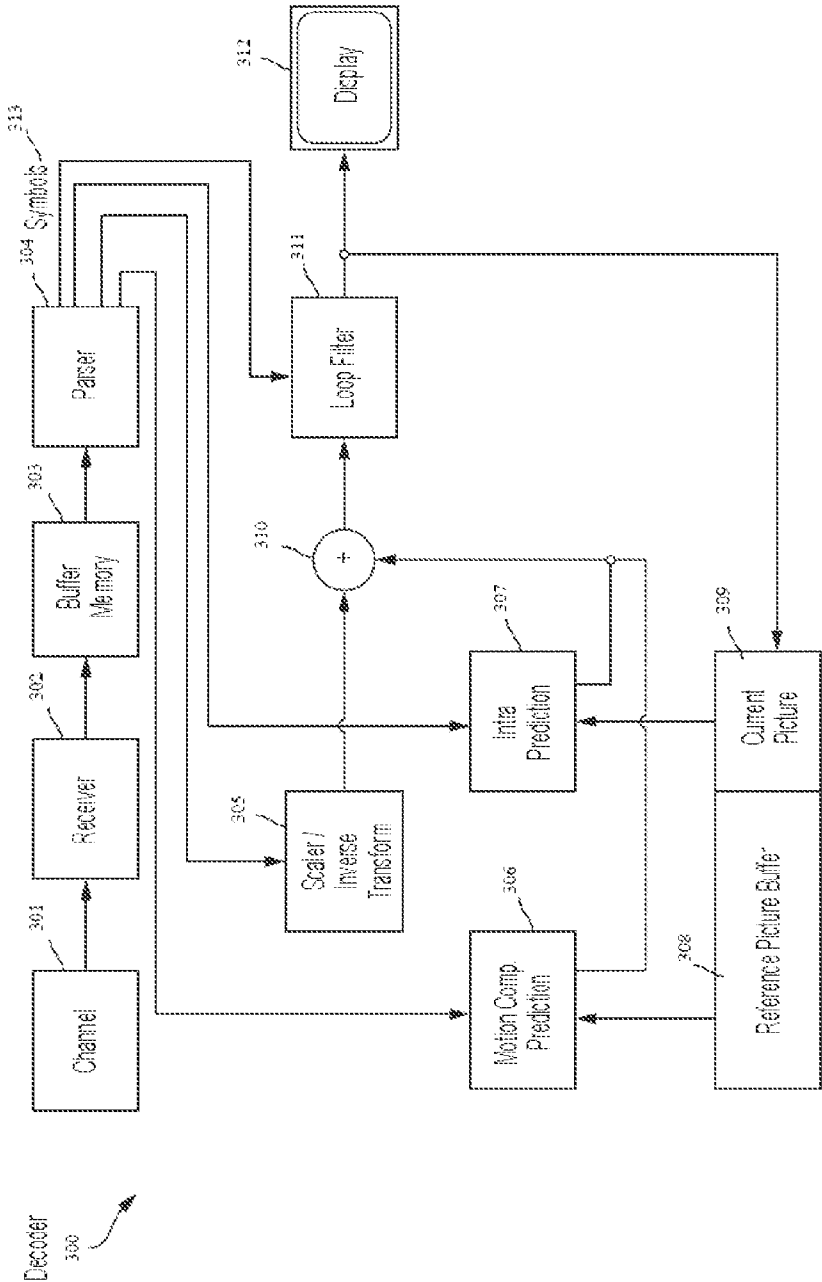


FIG. 4

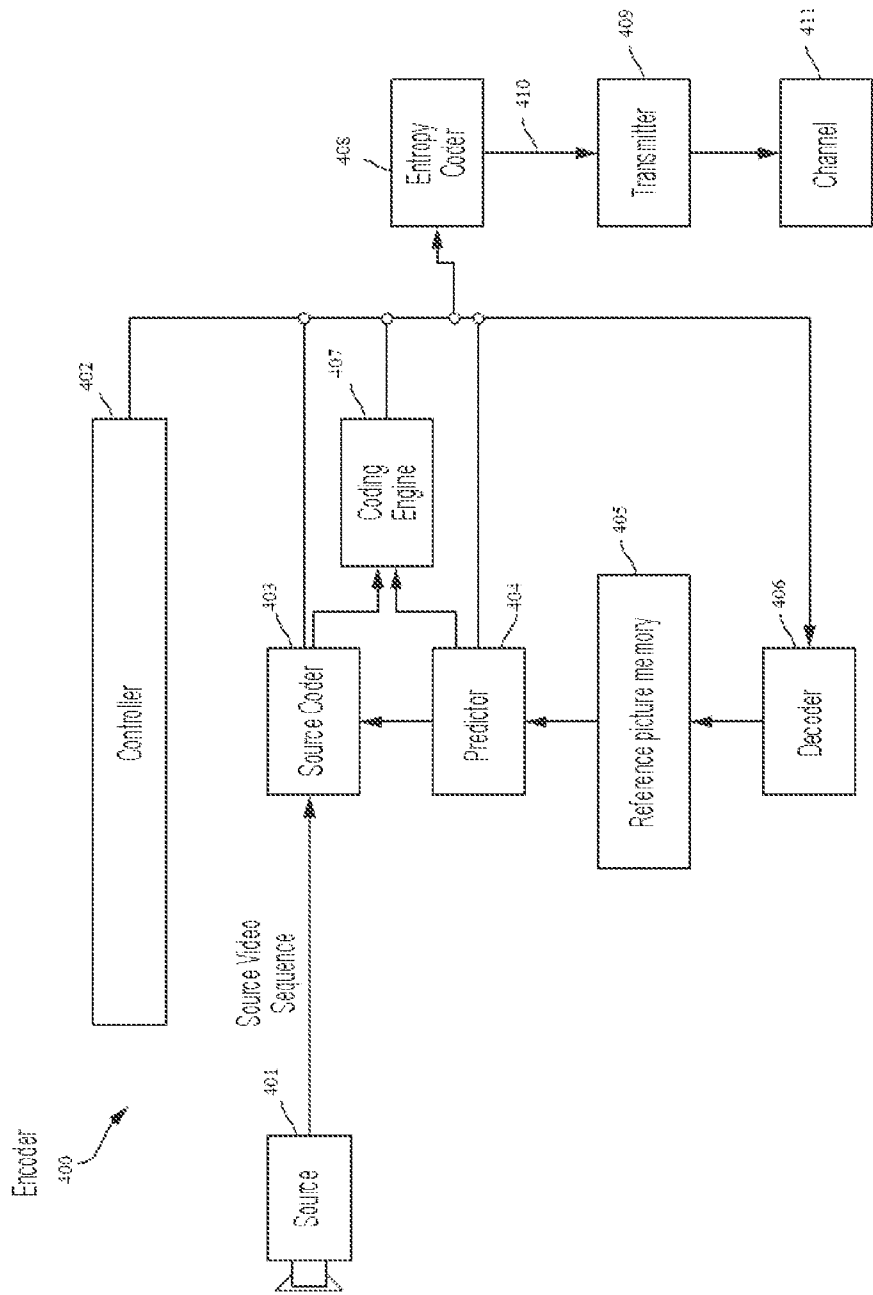


FIG. 5

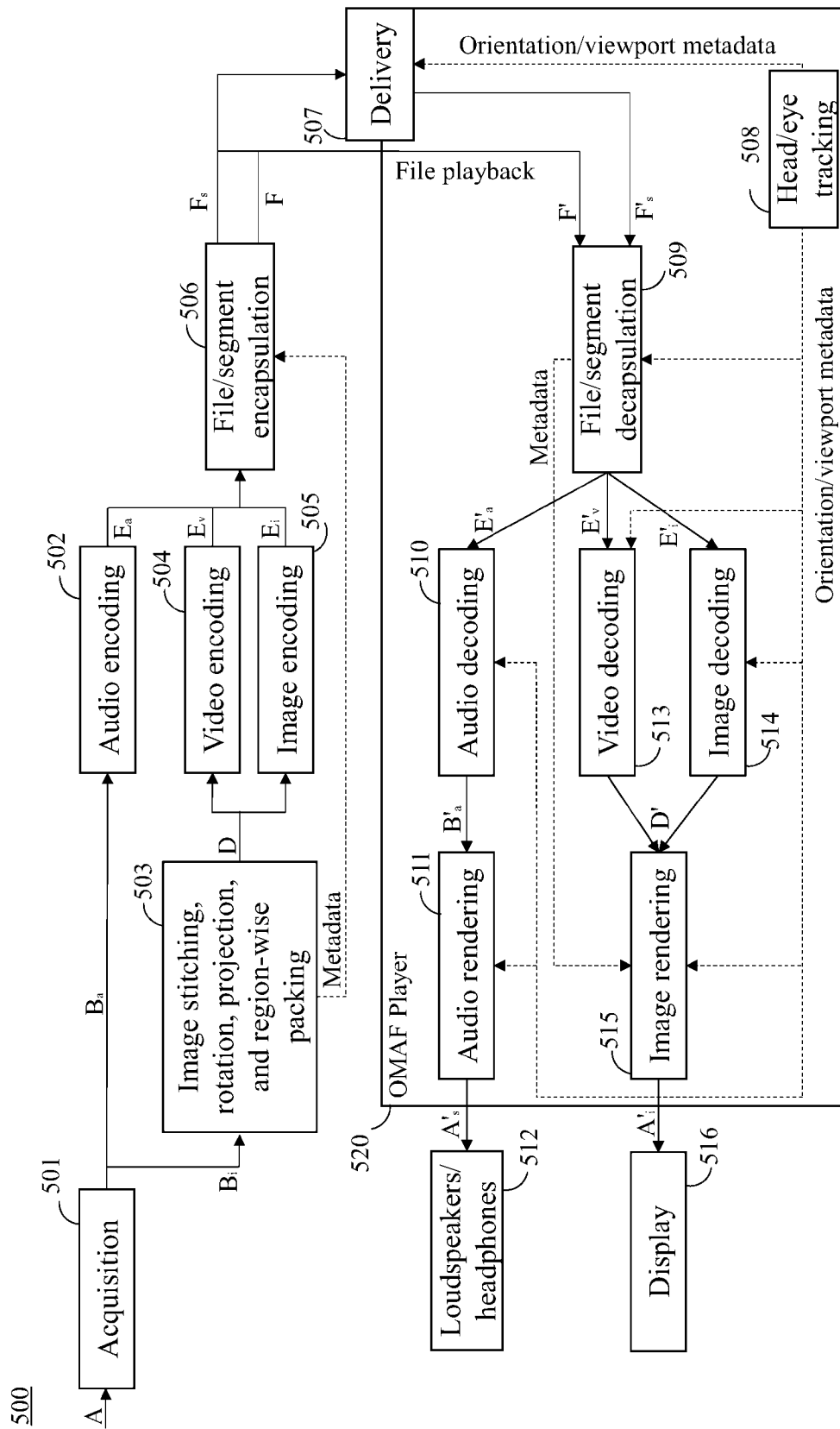


FIG. 6

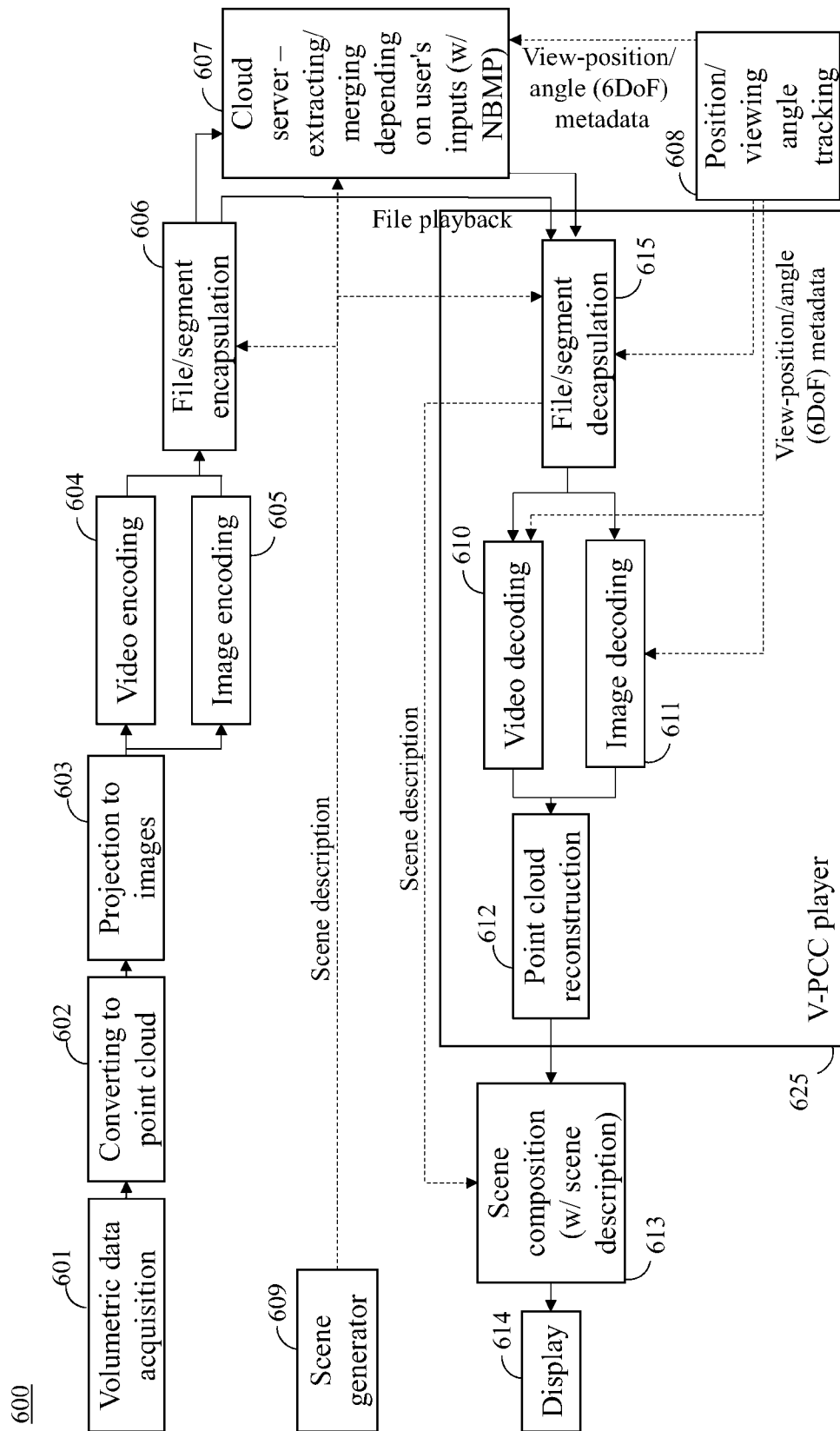


FIG. 7

700

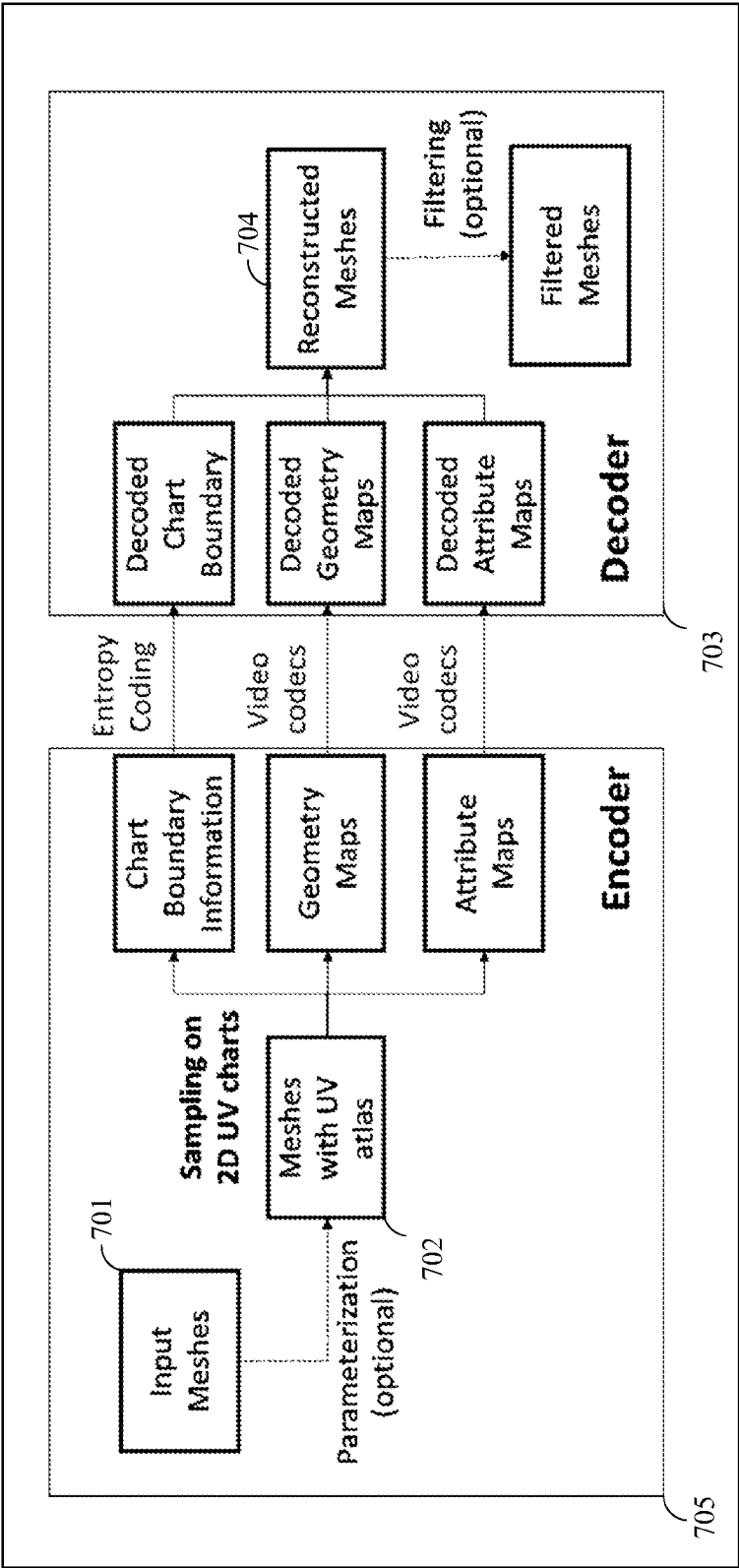


FIG. 8

800

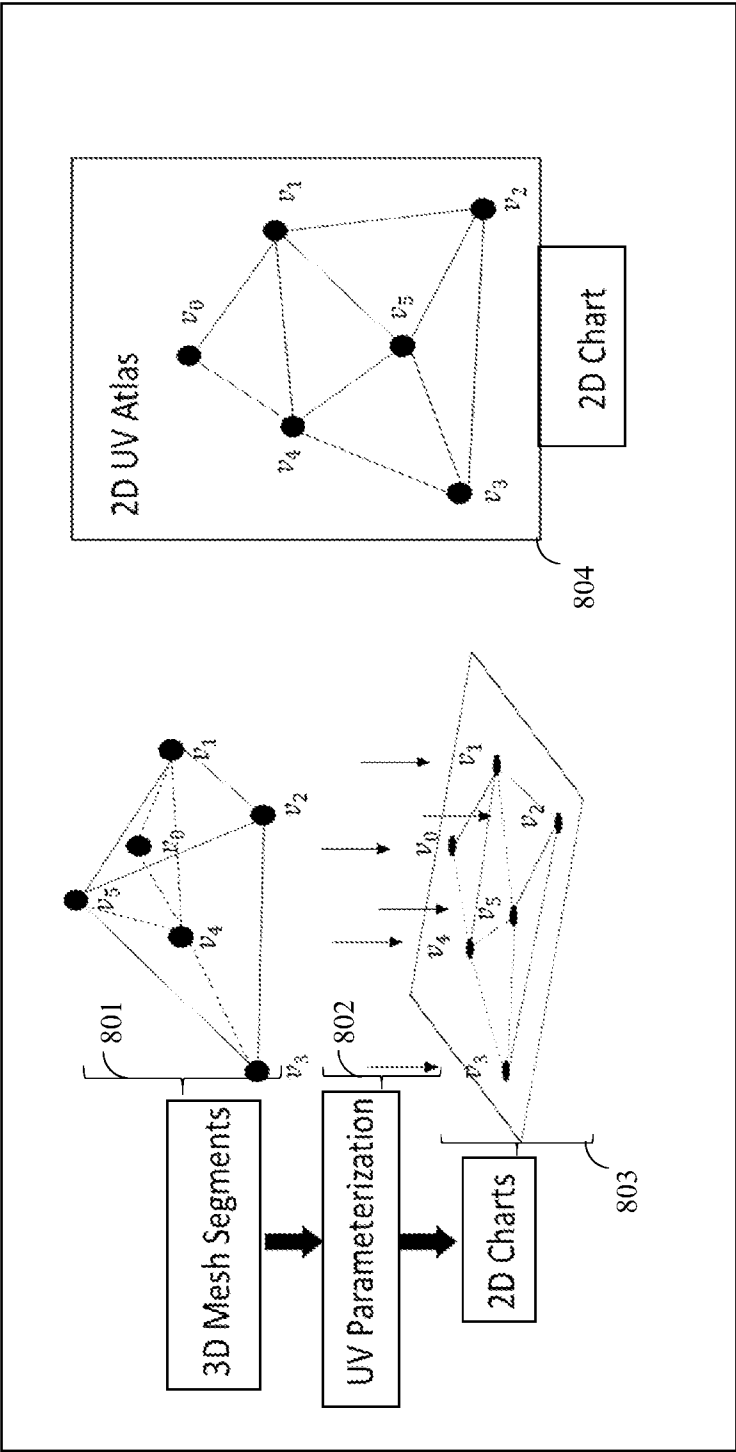


FIG. 9

900

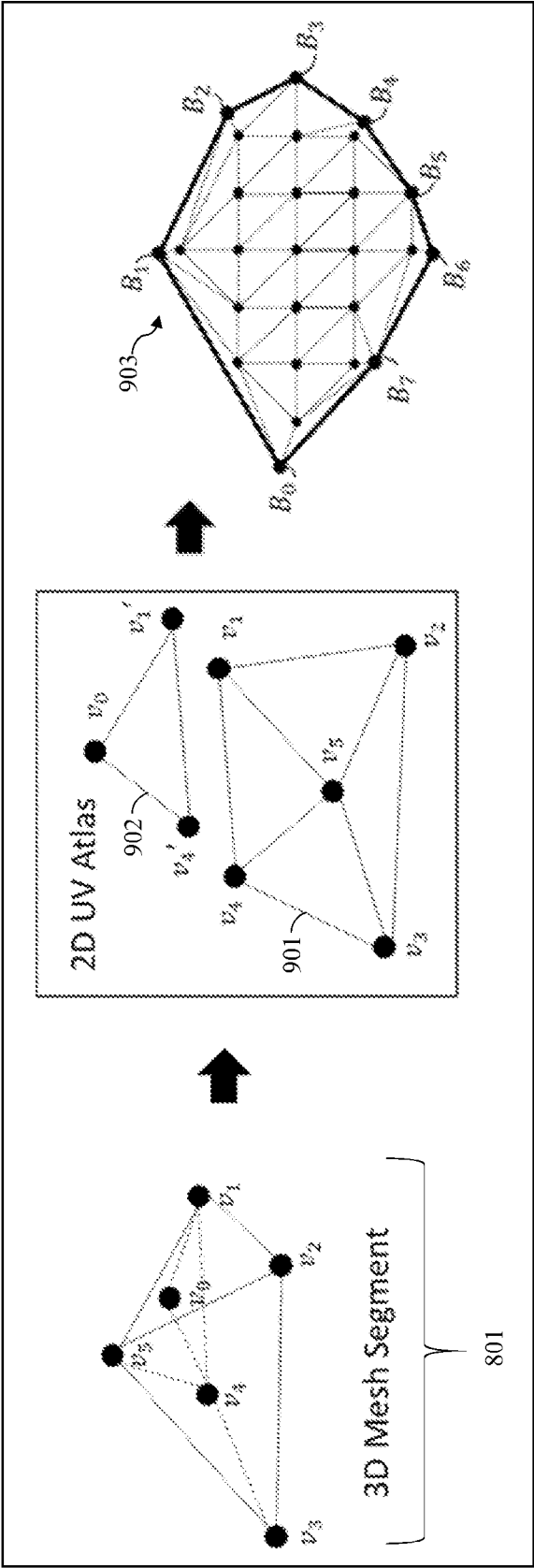


FIG. 10

1000

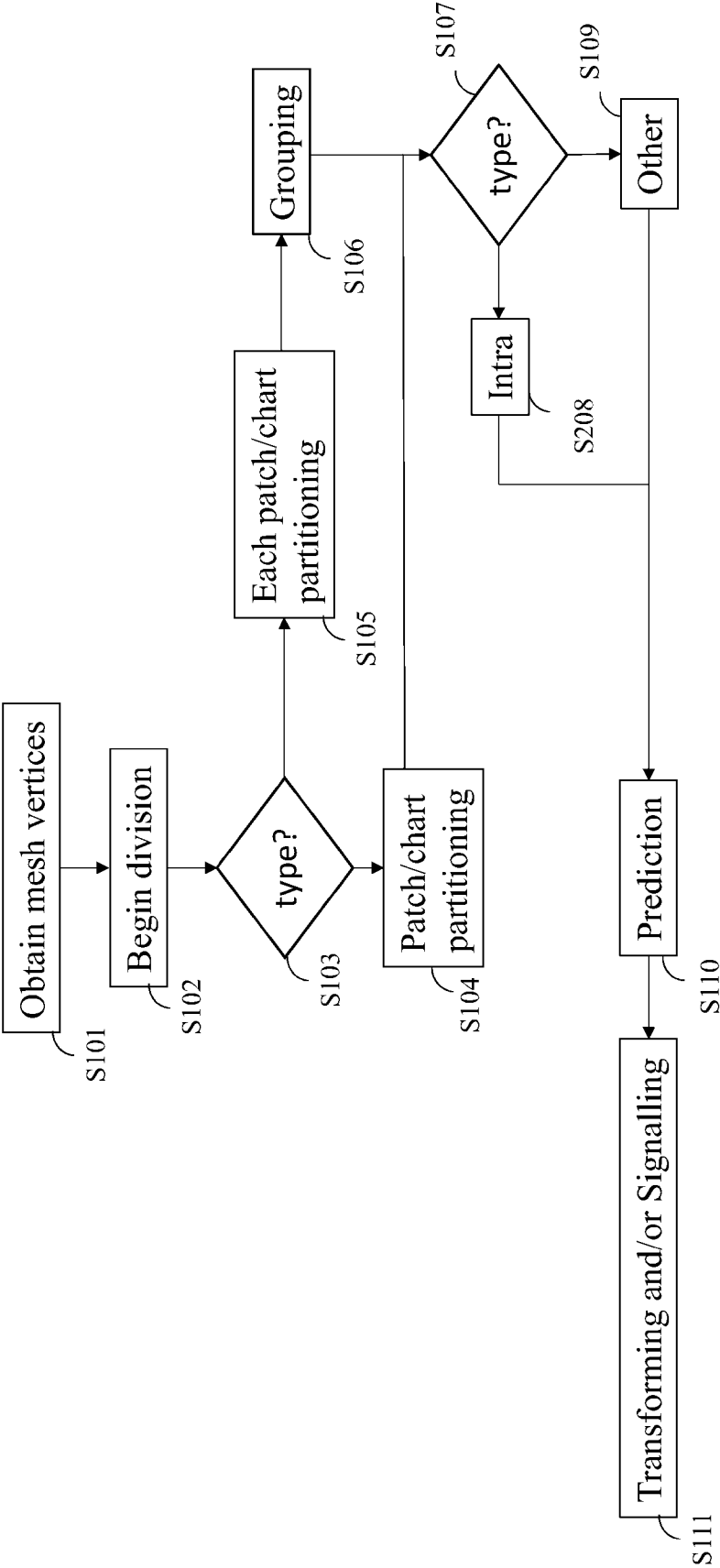


FIG. 11

1100

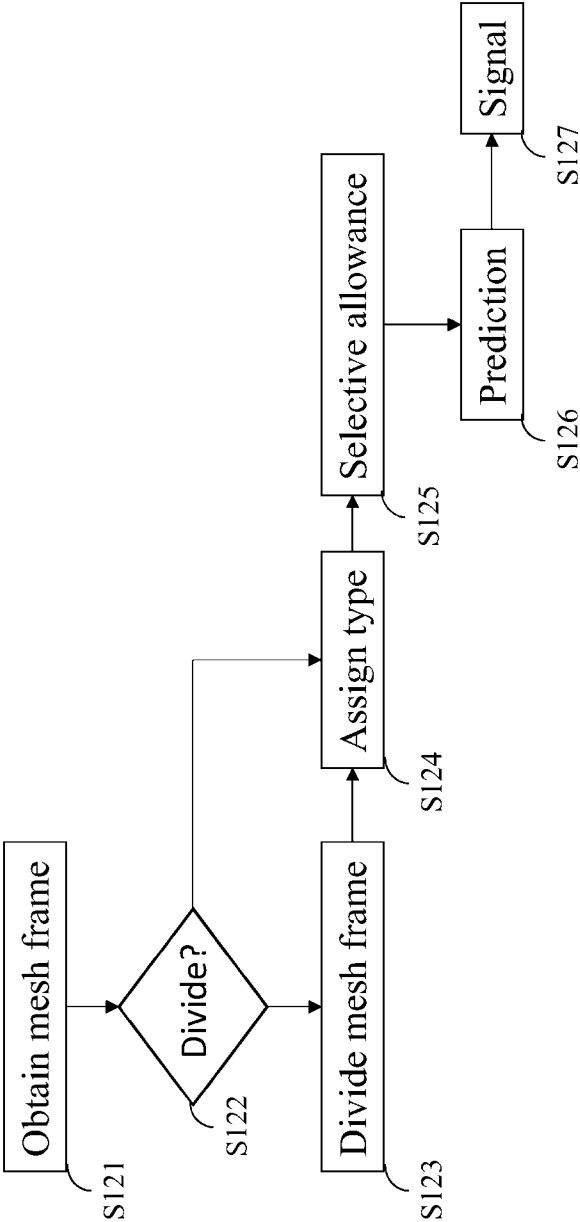


FIG. 12

1200

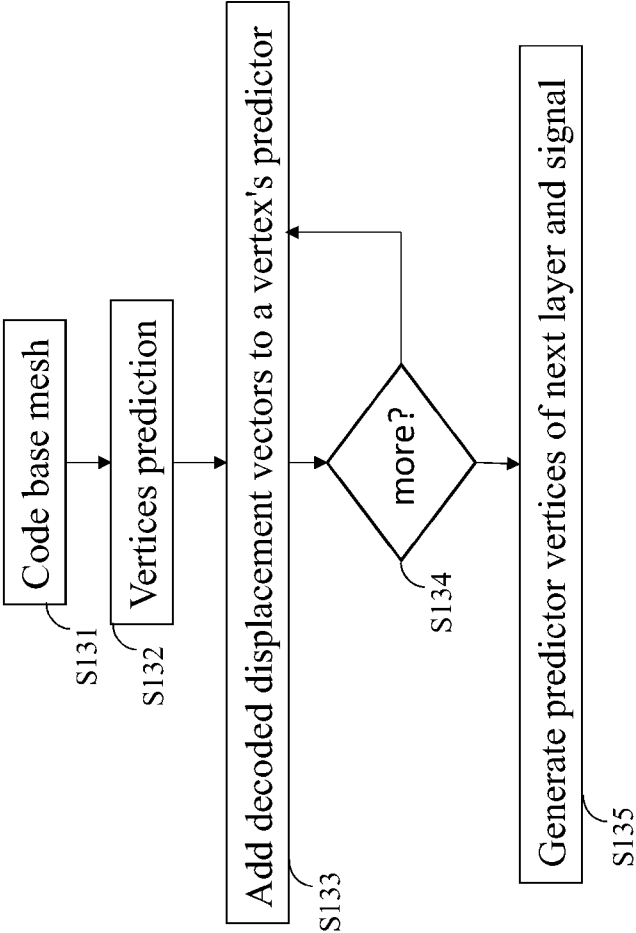


FIG. 13

1300

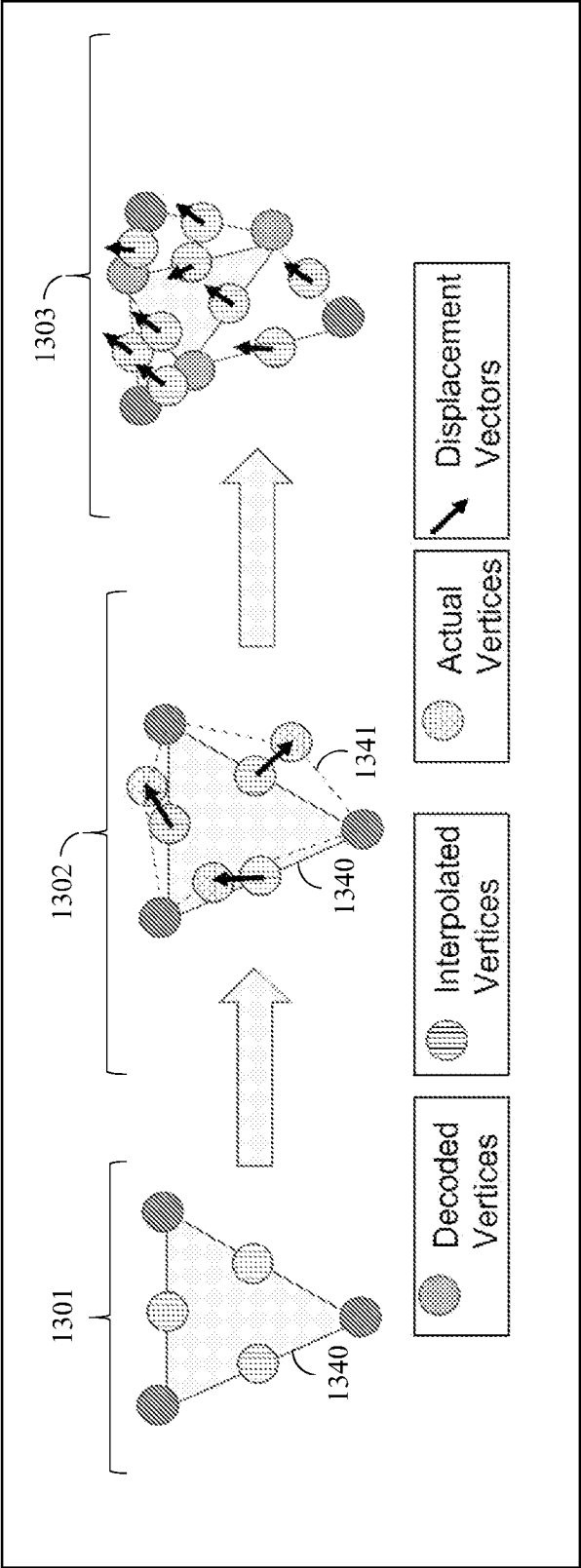


FIG. 14

1400

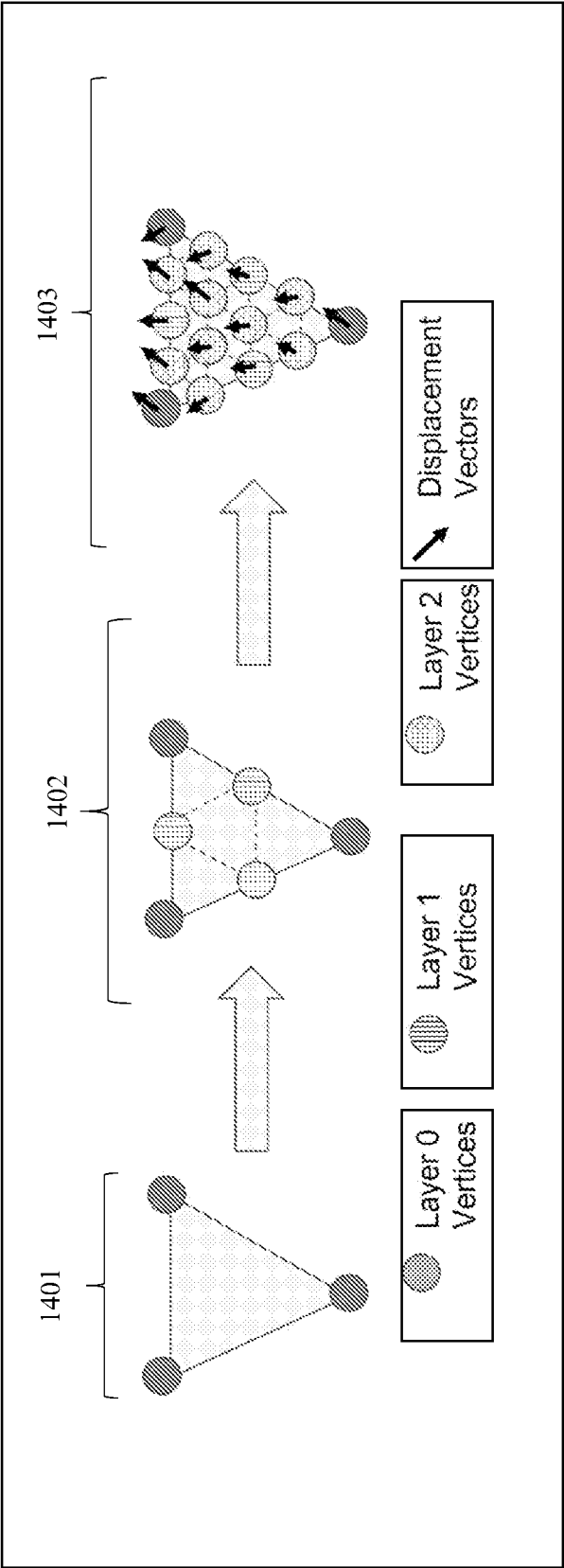


FIG. 15

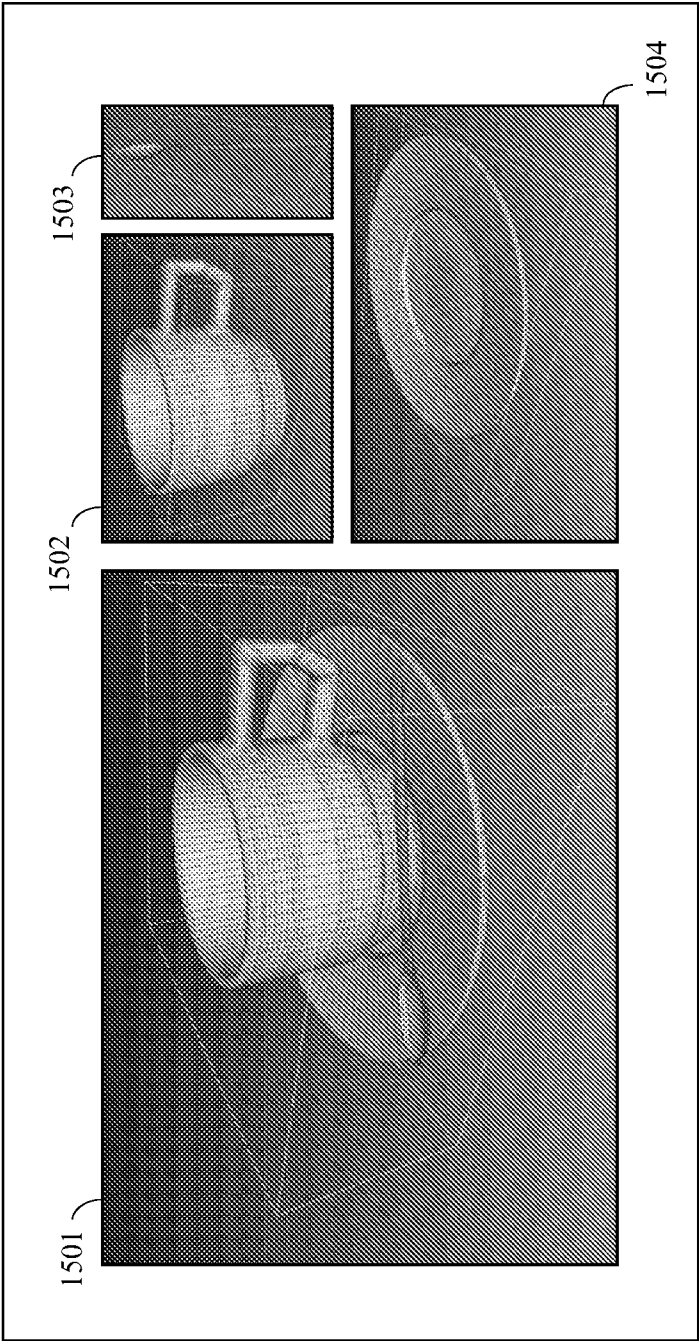


FIG. 16

1600

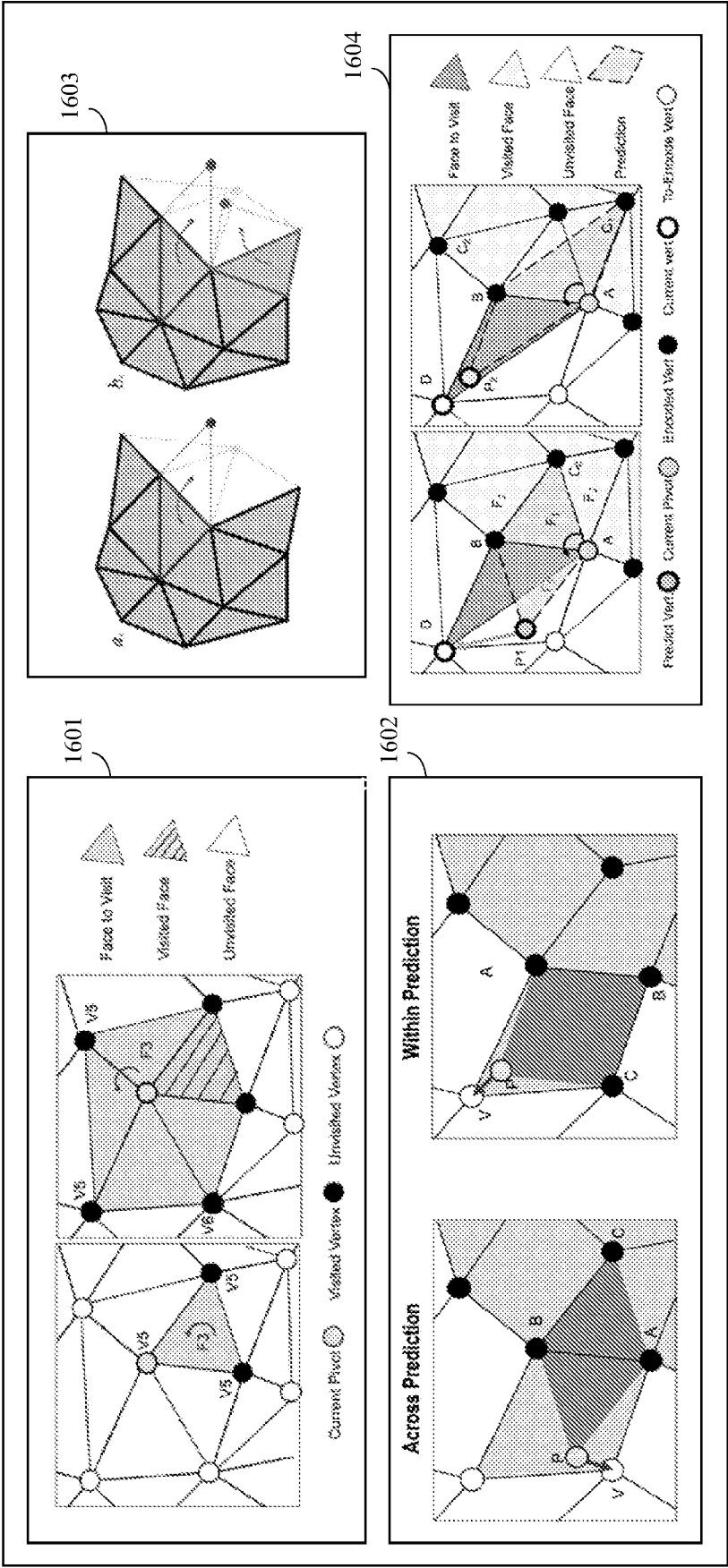


FIG. 17

1700

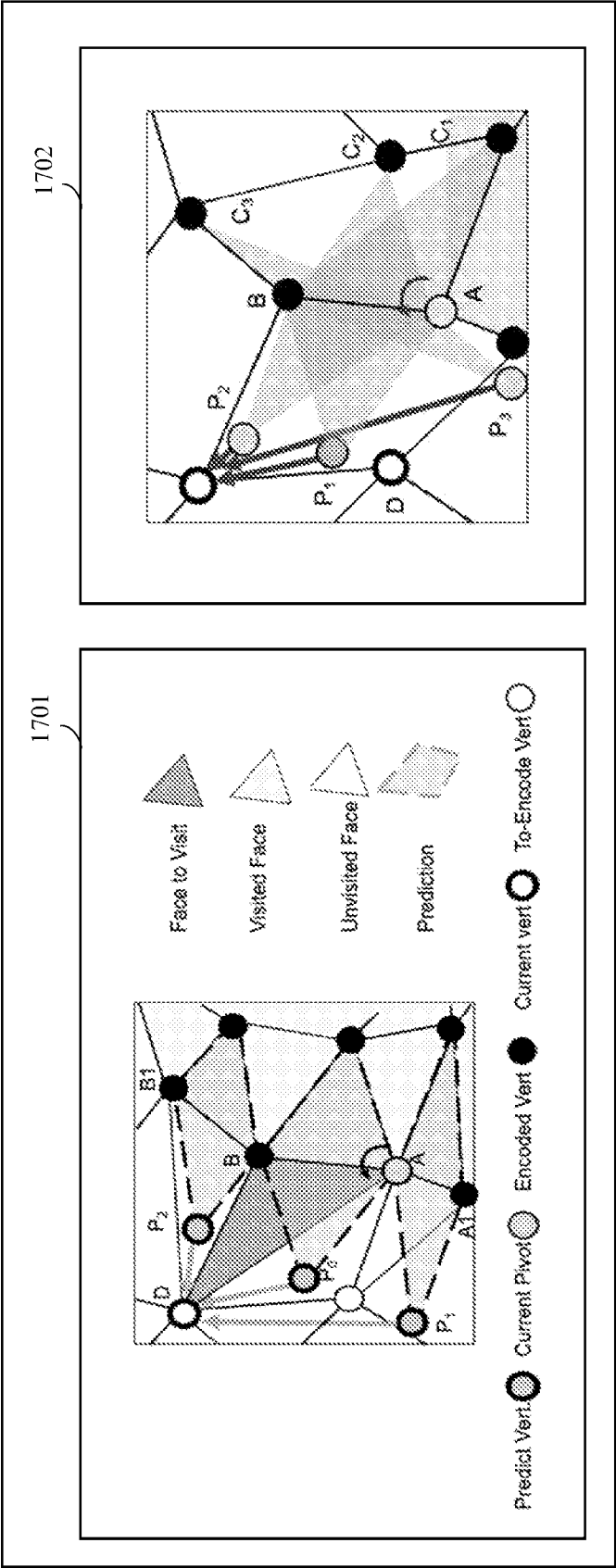


FIG. 18

1800

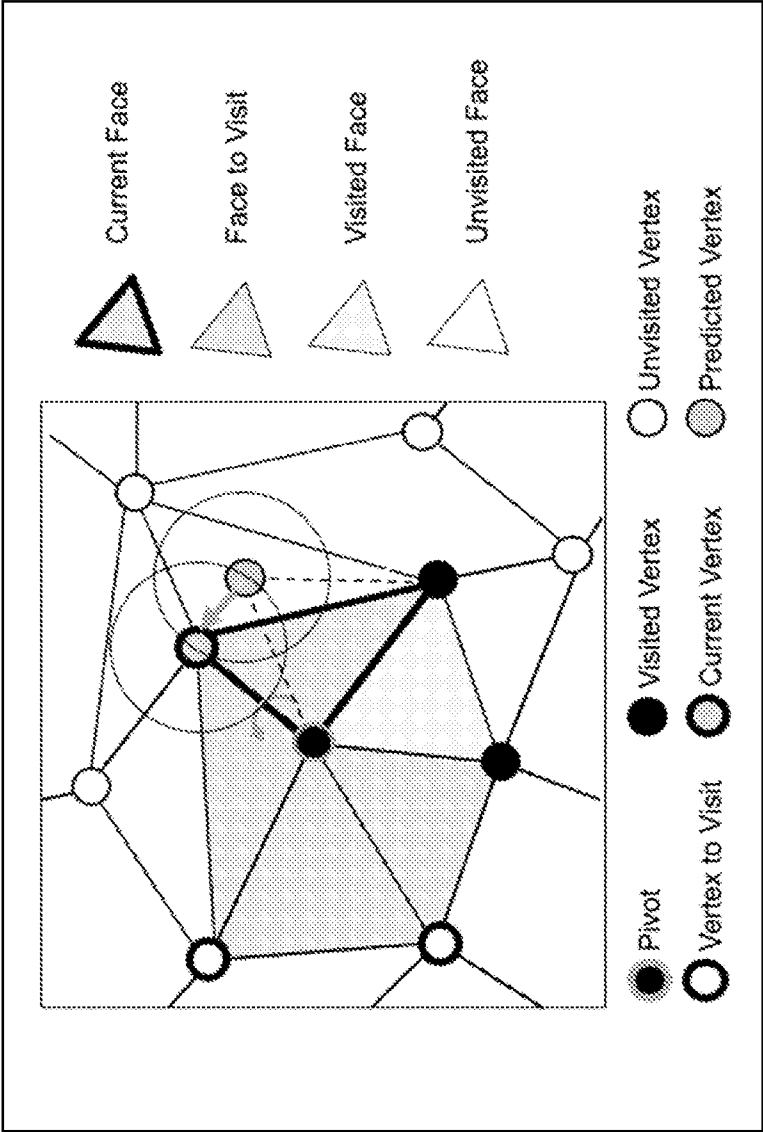


FIG. 19

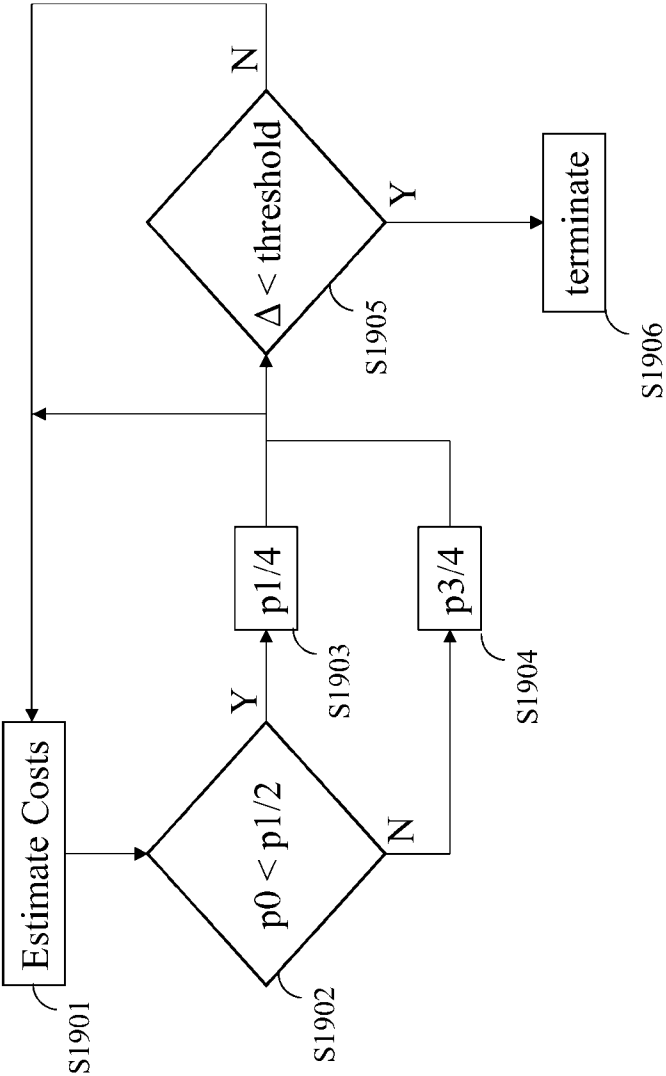


FIG. 20

2000

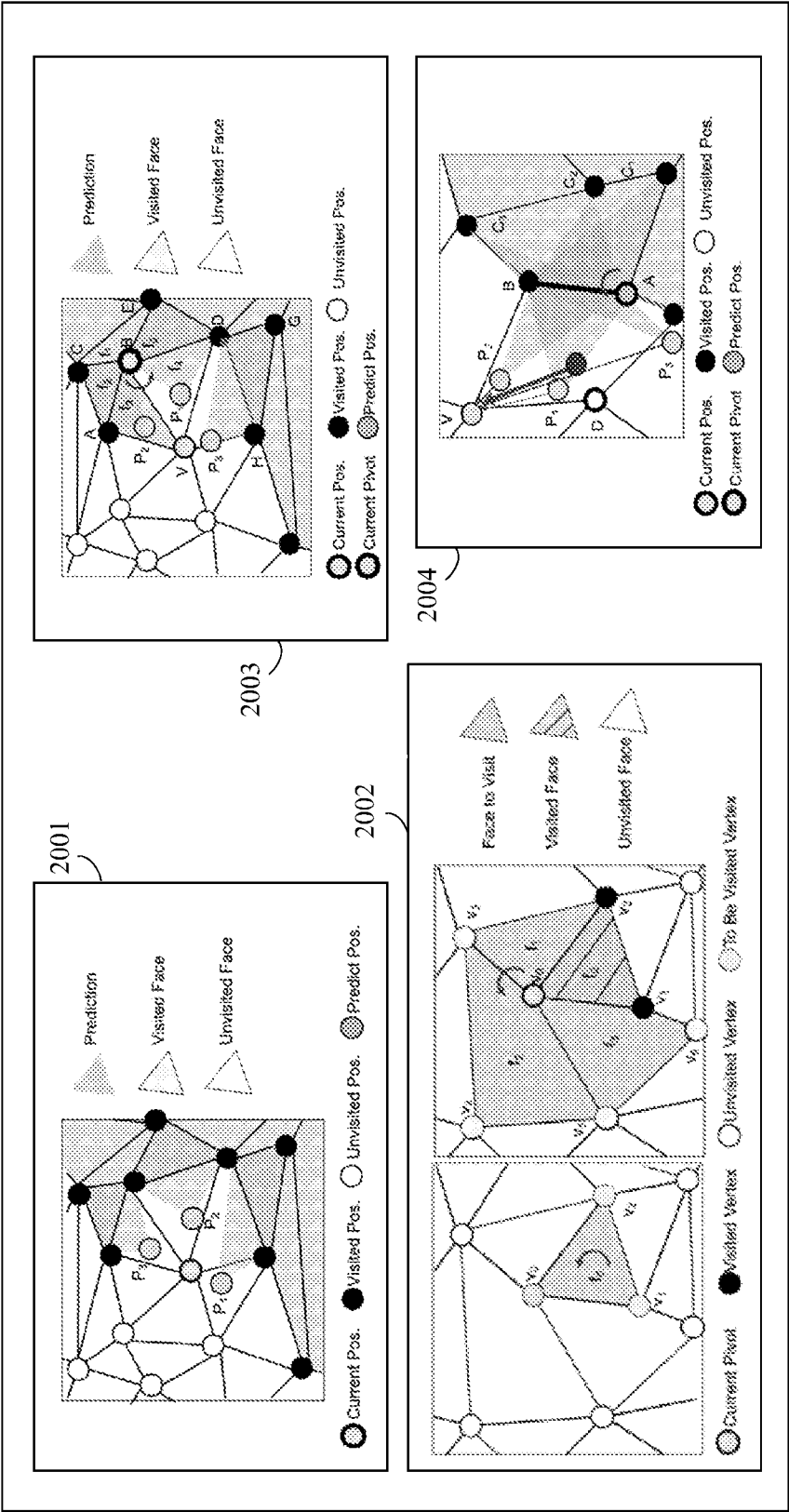


FIG. 21

