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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
17 July 2014 (17.07.2014)

WIPO | PCT

(10) International Publication Number
WO 2014/107944 A1

(51) International Patent Classification:
G06T 9/00 (2006.01)

(21) International Application Number:
PCT/CN2013/077404

(22) International Filing Date:
18 June 2013 (18.06.2013)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
PCT/CN2013/070310
10 January 2013 (10.01.2013) CN

(71) Applicant *(for all designated States except US)*: **THOMSON LICENSING** [FR/FR]; 1-5 rue Jeanne d'Arc, F-92130 Issy-les-Moulineaux (FR).

(72) Inventors; and
(71) Applicants *(for US only)*: **JIANG, Wenfei** [CN/CN]; Long Xiang Central Area 14-3-301, Xisanqi JianCaicheng West Road, Haidian District, Beijing 100192 (CN). **CAI, Kangying** [CN/CN]; 6-2-901, Luozhuangxili Residency, Zhichun Road, Haidian District, Beijing 100190 (CN). **TIAN, Jiang** [CN/CN]; 22-1002, Qingnian Road 29, Chaoyang District, Beijing 100025 (CN).

(74) Agent: **LIU, SHEN & ASSOCIATES**; A0601, Huibin Building, No. 8 Beichen Dong Street, Chaoyang District, Beijing 100101 (CN).

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

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

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

(54) Title: METHOD AND APPARATUS FOR VERTEX ERROR CORRECTION

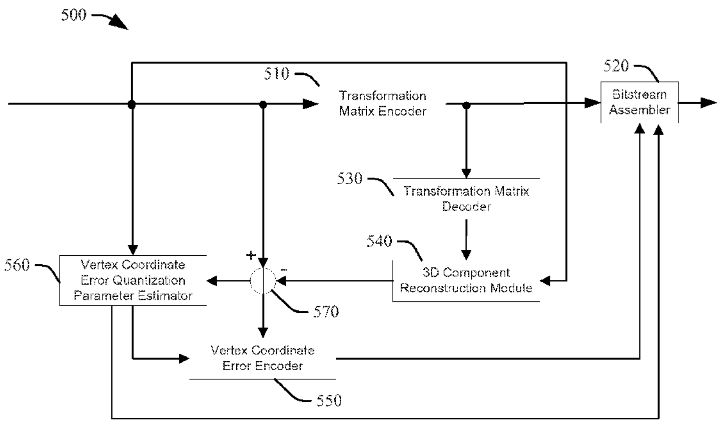

FIG. 5

(57) **Abstract**: ABSTRACT A 3D model can be modeled using "pattern-instance" representation, wherein an instance component may be represented as transformation (for example, rotation, translation, and scaling) of a pattern. Quantization errors may be introduced when encoding rotation information, causing different vertex coordinate errors at different vertices of an instance. To efficiently compensate the vertex coordinate errors, the encoder decides a quantization parameter for compensating a vertex coordinate error. The quantization parameter is signaled in the bitstream as a quantization index. The quantization index, a quantization table the indicates a mapping between quantization indices and quantization parameters, and vertex coordinate errors are encoded into a bitstream. The quantization table may be built based on statistical data. At the decoder, the vertex coordinate error is decoded based on a quantization parameter, which is determined from a received quantization index.

WO 2014/107944 A1

METHOD AND APPARATUS FOR VERTEX ERROR CORRECTION

TECHNICAL FIELD

This invention relates to a method and apparatus for generating a bitstream representative of a 3D model, and a method and apparatus for decoding the same.

BACKGROUND

In practical applications, many 3D models consist of a large number of connected components. These multi-component 3D models usually contain many repetitive structures in various transformations, as shown in FIG. 1.

Compression algorithms for multi-component 3D models that take advantage of repetitive structures in the input models are known. Repetitive structures of a 3D model are discovered in various positions, orientations, and scaling factors. The 3D model is then organized into “pattern-instance” representation. A pattern is used to denote a representative geometry of the corresponding repetitive structure. Components belonging to a repetitive structure are denoted as instances of the corresponding pattern and may be represented by a pattern ID and transformation information, for example, reflection, translation, rotation and possible scaling with respect to the pattern. The instance transformation information may be organized into, for example, reflection part, translation part, rotation part, and possible scaling part. There might be some components of the 3D models that are not repetitive, which are referred to as unique components.

A commonly owned PCT application, entitled “Vertex Correction for Rotated 3D Components” by W. Jiang, K. Cai, and J. Tian (PCT/CN2012/074286, Attorney Docket No. PA120012, hereinafter “Jiang”), the teachings of which are specifically incorporated herein by reference, discloses a method and apparatus for vertex error compensation when encoding and decoding a 3D model.

SUMMARY

The present principles provide a method for generating a bitstream representing a 3D model, comprising the steps of: accessing a reconstructed instance corresponding to an instance; determining a quantization parameter based on a vertex coordinate error between a vertex of the instance and a corresponding vertex of the reconstructed instance; determining a quantization index in response to the determined quantization parameter; and encoding the quantization index and the vertex coordinate error into the bitstream as described below. The present principles also provide an apparatus for performing these steps.

The present principles provide a method for decoding a bitstream representing a 3D model, comprising the steps of: accessing a reconstructed instance corresponding to an instance; determining a quantization index from the bitstream; determining a quantization parameter in response to the quantization index; decoding a vertex coordinate error representative of an error between a vertex of the instance and a corresponding vertex of the reconstructed instance; and refining the reconstructed instance in response to the decoded vertex coordinate error as described below. The present principles also provide an apparatus for performing these steps.

The present principles also provide a computer readable storage medium having stored thereon instructions for generating or decoding a bitstream representing a 3D model according to the methods described above.

The present principles also provide a computer readable storage medium having stored thereon a bitstream representing a 3D model generated according to the methods described above.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 shows exemplary 3D models with a large number of connected components and repetitive structures;

FIG. 2A shows pictorial examples depicting patterns and FIG. 2B shows pictorial examples depicting corresponding instances and reconstructed instances;

FIG.3 is a flow diagram depicting an example for encoding an instance of a 3D model, in accordance with an embodiment of the present principles;

FIG.4 is a flow diagram depicting an example for decoding an instance of a 3D model, in accordance with an embodiment of the present principles;

FIG. 5 shows an exemplary instance encoder according to the present principles; and

FIG. 6 shows an exemplary instance decoder according to the present principles.

DETAILED DESCRIPTION

As shown in FIG. 1, there may be many repetitive structures in 3D models. To efficiently encode the 3D models, the repetitive structures may be organized into patterns and instances, wherein an instance can be represented as a transformation of a corresponding pattern, for example, using a pattern ID and a transformation matrix which contains information such as translation, rotation, and scaling.

When an instance is represented by a pattern ID and a transformation matrix, the pattern ID and the transformation matrix are to be compressed when compressing the instance. Consequently, an instance may be reconstructed through the pattern ID and the decoded transformation matrix, that is, an instance may be reconstructed as transformation (from the decoded transformation matrix) of a decoded pattern indexed by the pattern ID. In one embodiment, when encoding the transformation matrix, the rotation part of the transformation matrix may be quantized, for example, using a constant number of bits. Because of loss introduced at quantization, the decoded rotation part may be different from the original rotation part.

FIGs. 2A and 2B illustrate exemplary components in a 2D representation, wherein components 210 and 220 are patterns, components 250 and 270 (in solid lines) are original instances to be compressed, and components 260 and 280 (in dashed lines) are reconstructed instances. In particular, instances 250 and 270 can be represented as transformed (i.e., rotated and translated) versions of patterns 210 and 220, respectively.

In the examples of FIG. 2B, quantization of rotation introduces an error of about 5°, thus causing differences between original instances and reconstructed instances. As can be seen in FIG. 2B, while the rotation errors (in angular measure) are similar for instances 250 and 270, the vertex coordinate errors (i.e., vertex shifts, for example, from A to A', B to B' in FIG. 2B, between original instances and reconstructed instances) caused by rotation quantization vary significantly between both instances, with instance 270 having much larger vertex coordinate errors. Consequently, the quality of reconstructed components may be inconsistent, for example, a larger instance may have lower reconstruction quality than a smaller instance.

In Jiang, to efficiently compensate the vertex coordinate errors, an upper bound can be estimated for the vertex coordinate error of a vertex. Based on the upper bound, the codec decides whether the vertex coordinate error of the vertex needs to be compensated, and decides a quantization parameter for compensating the vertex coordinate error if compensation is needed. The upper bound can be estimated at both the encoder and decoder, and thus, no explicit signaling is needed to indicate whether vertex coordinate error compensation is used or to indicate the quantization parameter for the vertex coordinate error.

The present principles also provide a method and apparatus for efficiently compensating the vertex coordinate errors caused by rotation quantization. To reduce the computation load at a decoder, the quantization parameter is signaled in a bitstream. In one embodiment, an index corresponding to the number of quantization bits used for quantizing the vertex coordinate error is transmitted in the bitstream through a quantization table.

FIG. 3 illustrates an exemplary method 300 for encoding an instance of a 3D model. Method 300 starts at step 305. At step 310, 3D model data is input and initialization is performed. Additional data, such as a quality parameter, a maximum tolerable vertex coordinate error, quantization parameters for the translation part and rotation part of a transformation matrix, may also be input or inferred from the input. In one exemplary embodiment, the initialization step may organize the repetitive structures into patterns and instances, generate transformation matrices for instances, and encode patterns to form reconstructed patterns. For a particular to-be-encoded instance (denoted as C), the corresponding original pattern, reconstructed pattern, and transformation matrix are denoted as P, P', and T, respectively. It is possible that an instance can be precisely represented as a transformation of a pattern, that is, C=TP. Alternatively, the transformation of the pattern may be an approximate of the instance under some circumstances, that is, $C \approx TP$.

The transformation matrix (T) is encoded at step 320. At step 330, the encoded transformation matrix is then decoded as T', and the instance is reconstructed, for example, using a corresponding reconstructed pattern and decoded transformation matrix (C'=T'P').

At step 340, the vertex coordinate error (E_i) between the vertex (V_i') in the reconstructed instance and the corresponding vertex (V_i) in the original instance is calculated, for example, as $E_i=V_i-V_i'$. In order to encode the vertex coordinate error, a quantization parameter is estimated at step 350. The vertex coordinate error is quantized and coded at step 360. In addition, the quantization parameter is signaled in the bitstream. At step 370, it checks whether more vertices need to be processed. If more vertices are to be processed, the control is returned to step 340. Otherwise, the control is passed to the end step 399.

To efficiently signal the quantization parameter, an index of the quantization parameter, rather than an actual quantization parameter, can be encoded into the bitstream. Using the number of quantization bits as an exemplary quantization parameter, the quantization process is discussed in further detail. The present principles can also be applied when other quantization parameters, for example, but not limited to, quantization step size, are used.

TABLE 1

Quantization index	Number of quantization bits
0	2
1	4
2	5
3	6

TABLE 1 illustrates an exemplary quantization table, wherein the number of quantization bits is mapped to a quantization index. Specifically,

QBtable[0] = 2;
QBtable[1] = 4;
QBtable[2] = 5;
QBtable[3] = 6.

We denote MaxErrorAllow as quality requirement (i.e., the maximum tolerable vertex coordinate error) provided by a user and Error as the difference between vertex coordinates of the original instance and the reconstructed instance for a particular vertex. For the particular vertex, we estimate the initial number of bits needed to quantize the vertex coordinate error as:

$$QB = \text{ceil}[\log2(\text{Error} / \text{MaxErrorAllow})]. \quad (1)$$

Then we search the quantization table for the number of quantization bits that is closest to QB. For example, when QB = 7, a quantization index 3 (QBtable[3] = 6) is selected as a corresponding quantization index and QB is set to 6. Subsequently, Error / MaxErrorAllow is quantized into a binary code of QB bits. The quantization index and the binary code are then encoded into the bitstream. As can be seen from TABLE 1, the value of a quantization index is usually smaller than the corresponding number of quantization bits, and may require fewer bits to transmit. Thus, sending a quantization index rather than sending the number of quantization bits directly may reduce the bit rate.

At the decoder, the maximum tolerable error (MaxErrorAllow) and quantization table can be derived from the bitstream. For a vertex, the quantization index is received from the bitstream, and the number of quantization bits QB can be determined from the quantization index and the quantization table. The quantized vertex coordinate error Q_value can then be read as QB bits from the bitstream. The vertex coordinate error can be calculated as:

$$\text{Error'} = \text{MaxErrorAllow} * \text{Q\_value}. \quad (2)$$

For example, we assume the quantization index and the binary code representative of the quantized vertex coordinate error are encoded in the bitstream contiguously and they are ‘1010100111...’.If the quantization index is encoded with 2 bits, we get a quantization index ‘10’ = 2. If the quantization table as shown in TABLE 1 is used, the number of quantization bits can be derived as QBtable[2] = 5. Subsequently, we read 5 bits ‘10100’ from the bitstream and determine the quantized vertex coordinate error as Q_value = ‘10100’=20.Thus, Error’=MaxErrorAllow * 20.

As discussed above, a quantization table is used to indicate the quantization parameter in the bitstream. In one embodiment, the quantization table can be specified by metadata or user input. In another embodiment, the quantization table can be built based on statistical data.

For example, we can calculate values for QB, based on Eq. (1), using different vertices from a large amount of components from different 3D models. After we get a large set of QB values, we can choose n most frequently occurred QB values as elements of the QB table. If we denote the n most frequently occurred QB values as QB_0, QB_1, …, QB_{n-1}, the quantization table can be illustrated as shown in TABLE 2. When fixed-length coding is used to encode the quantization index, we may build the table such that $QB_0 < QB_1 < \ldots < QB_{n-1}$. When variable-length coding is used to encode the quantization index, to reduce the amount of data used to send quantization indices, we may set up the quantization table such that $Prob(QB_0) > Prob(QB_1) > \ldots > Prob(QB_{n-1})$. That is, a more probable number of quantization bits corresponds to a smaller index, which usually requires fewer bits to encode.

TABLE 2

Quantization index	Number of quantization bits
0	QB_0
1	QB_1
…	…
n-1	QB_{n-1}

FIG. 4 illustrates an exemplary method 400 for decoding an instance of a 3D model. The input of method 400 may include a bitstream, for example, the bitstream generated using method 300. Additional data, for example, a reconstructed pattern (P') corresponding to the instance to be decoded may also be included as an input. Method 400 starts at step 405. At step 410, initialization is performed, for example, the quantization parameters for the transformation matrix and quality parameter are derived from the input bitstream and the maximum tolerable vertex coordinate error is calculated from the quality parameter.

The transformation matrix is decoded as T', and the instance is reconstructed as C' at step 420, for example, using a corresponding reconstructed pattern and decoded transformation matrix (C'=T'P'). At step 430, a quantization parameter, for example, the number of quantization bits (QB), is determined from the bitstream. The encoded vertex coordinate error is decoded at step 440, for example, QB bits are read from the bitstream and the vertex coordinate error is calculated using Eq. (2). At step 450, the decoded vertex coordinate error (E_i') is used to compensate the corresponding vertex (V_i') of the instance initially reconstructed at step 420, for example, as V_i''=V_i'+E_i'. That is, the vertex of the reconstructed instance is refined. Method 400 ends at step 499.

A vertex coordinate error compensation flag may be used to indicate whether the vertex coordinate error is compensated. The flag should be known at both the encoder and decoder. When the flag is set to 1, the error compensation is used. Otherwise, the vertex coordinate error compensation is not used. Specifically, when methods 300 and 400 are used, steps 340-370 in method 300 and steps 430-460 are not needed if the flag is 0.

In one exemplary embodiment, the decoding process of a bitstream representing a 3D modelcan be described using the following pseudo-code.

```
void PB3DMC_Decoder()
{
Read PB3DMC_stream_header;
if (uni_part_bit == 0 && repeat_struc_bit == 0)
{
Decode the 3D model using the decoder indicated by 3d_model_compr_mode;
}
else
{
```

```
if (uni_part_bit == 1)
{
Decode the unique part using the decoder indicated by 3d_model_compr_mode;
Separate different unique components by traversal based on connectivity;
Decode the translation vectors of unique components;
Reconstruct the unique part by translating all reconstructed unique components to their positions;
}
if (repeat_struc_bit == 1)
  {
  Repeat_Struc_Decoder();
  }
}
}

void Repeat_Struc_Decoder ()
{
Decode all patterns;
if (sym_instance_num > 0)
{
Decode all symmetric instances;
Decode all stitching information;
}
Reconstruct all unconnected-repetitive-structure patterns and unique components which include symmetric structures, using the recovered patterns, symmetric instances and stitching information;
Decode the translation vectors of all unconnected-repetitive-structure patterns and unique components which include symmetric structures;
Reconstruct those components corresponding to all unconnected-repetitive-structure patterns and unique components which include symmetric structures using the decoded translation vectors;
if (insta_trans_elem_bit == 1)
{
Instance_Elementary_Mode_Decoder();
}
else
{
```

```
Instance_Grouped_Mode_Decoder();
}
}

void Instance_Elementary_Mode_Decoder()
{
for (i = 0; i < numInstance; i ++)
{
Read elem_insta_QP_translation_flag;
Read elem_insta_QP_rotation_flag;
if(elem_insta_QP_translation_flag == 1)
{
Decode elem_QP_translation;
}
    else
{
elem_QP_translation  = QP_Translation;
}
if(elem_insta_QP_ rotation_flag == 1)
{
Decode elem_QP_ rotation;
}
    else
{
elem_QP_ rotation = QP_ rotation;
}
Decode idPattern;
Read elem_insta_flip_flag;
Read elem_insta_reflection_flag;
Read elem_insta_attribute_header;

Decode instance translation vector by fixed length decoder whose parameter is QB_translation;

Decode Euler angles by fixed length decoder whose parameter is QB_rotation;
Recover rotation matrix using the decoded Euler angles;
```

```
if (using_scaling_bit == 1)
        Decode scaling factor;

if (error_compen_enable_bit == 1)
{
Read elem_insta_error_compen_flag;
if (elem_insta_error_compen_flag == 1)
Decode error compensation data;
}

Recover the geometry of current instance by the pattern indicated by idPattern, recovered
translation vector, recovered rotation matrix, reflection transformation if there is any,
scaling factor if there is any, error compensation data if there is any;
if(elem_insta_flip_flag == 1)
Flip all triangles of current instance;

Decode the attribute data of current instance;
}
}

void Instance_Grouped_Mode_Decoder()
{
Read the elem_insta_QB_translation_flag of all instances;
Read the elem_insta_QB_rotation_flag of all instances;
Decode the elem_insta_QB_translation of those instances whose
elem_insta_QB_translation_flag is 1;
Decode the elem_insta_QB_rotation of those instances whose elem_insta_QB_rotation
_flag is 1;

Read compr_insta_patternID_header;
Decode pattern IDs of all instances;

Read the elem_insta_flip_flag of all instances;
Read the elem_insta_reflection_flag of all instances;

Read compr_insta_transl_header;
  Decode translation vectors of all instances by octree decomposition based decoder;
```

```
Read compr_insta_rotat_header;
Decode the Euler angles of all instances;
Recover the rotation matrices of all instances;

if (use_scaling_bit == 1)
{
Read compr_insta_scaling_header;
Decode the scaling factors of all instances;
}

if (error_compen_enable_bit == 1)
{
Read elem_insta_error_compen_flag of all instances;
for (i = 0; i < numInstance; i ++)
{
if (the corresponding elem_insta_error_compen_flag is 1)
Decode the error compensation data for the current instances;
}
}
Recover the geometry of all instances by the recovered patterns, recovered translation vectors, recovered rotation matrices, reflection transformations if there are any, scaling factors if there are any, error compensation data if there is any;
Decode the attribute data of all instances if there is any;
}
```

In TABLE 3, exemplary syntax and semantics are illustrated for the quantization table, which may be included in a bitstream header.

TABLE 3

if (error_compen_enable_bit==’1’){
error_compen_QB_table[0..3]
}

error_compen_enable_bit: This 1-bit unsigned integer indicates whether or not there are data fields of compressed coding error compensation data for some instances in the bitstream. 0 means there is no data field of compressed coding error compensation data of instances in the bitstream and 1 means there are data fields of compressed coding error compensation data of some instances in the bitstream. This bit corresponds to the vertex coordinate error compensation flag discussed before for methods 300 and 400.

error_compen_QB_table: If the error compensation mode is activated, the number of quantization bits of the compensated value for each vertex may be adaptively determined at the encoder. The encoder transmits an index of the number of quantization bits instead of the number of quantization bits itself into the bitstream. The decoder looks up the quantization table to determine the number of the quantization bits. There are 4 pre-defined quantization bits in the table, each represented by one 5-bit unsigned integer.

In TABLE 4, exemplary syntax and semantics for vertex error compensation data are illustrated. In this example, class compr_elem_insta_error_compen_data contains the compressed vertex error compensation data of the i^{th} instance.

TABLE 4

classcompr_elem_insta_error_compen_data{
for (j = 0; j < numofvertex; j ++) {
elem_compen_err_QB_id
compr_ver_compen_err_data
}

elem_compen_err_QB_id: This 2-bit unsigned integer indicates the index of the number of quantization bits for the j^{th} vertex of the instance in error_compen_QB_table.

compr_ver_compen_err_data: This data field contains the compressed compensated value of the j^{th} vertex of the instance.

FIG. 5 depicts a block diagram of an exemplary instance encoder 500. The input of apparatus 500 may include an instance (C) to be encoded, a corresponding pattern (P) and reconstructed pattern (P'), transformation matrix T, a quality parameter, and quantization parameters for transformation matrix.

Transformation matrix encoder 510 encodes the transformation matrix T, for example, based on the quantization parameters for different parts of the transformation matrix. Transformation matrix decoder 530 decodes the output of encoder 510 to get a reconstructed transformation matrix T'. Using a corresponding reconstructed pattern P' and T', the instance may be reconstructed as C'=T'P' at 3D component reconstruction module 540. Adder 570 takes differences between the original instance and the reconstructed instance, for example, as E=C-C'.

Based on the vertex coordinate error E, vertex coordinate error quantization parameter estimator 560 estimates a quantization parameter for quantizing the vertex coordinate error at vertex coordinate error encoder 550, for example, using Eq. (1). Quantization parameter estimator 560 may further obtain a corresponding index for the estimated quantization parameter from a quantization table, and the quantization parameter may be adjusted based on the quantization index. The outputs of transformation matrix encoder 510 and vertex coordinate error encoder 550, and the quantization index are assembled by bitstream assembler 520 into a bitstream, which can be combined with other bitstreams representing the pattern or other components to form an overall bitstream for a 3D model.

FIG. 6 depicts a block diagram of an exemplary instance decoder 600. The input of apparatus 600 may include a bitstream corresponding to an instance (C), for example, a bitstream generated according to method 300 or by encoder 500, and a corresponding reconstructed pattern (P'). Entropy decoder 610 decodes the bitstream, for example, to get quantized vertex coordinate errors, quantization parameters used for transformation matrix, and the quantization table and quantization indices used for vertex coordinate error compensation.

Transformation matrix decoder 620 reconstructs transformation matrix T', for example, based on the quantization parameters for different parts of the transformation matrix. Using a corresponding reconstructed pattern P' and T', the instance may be reconstructed as C'=T'P' at 3D component reconstruction module 630.

Vertex coordinate error decoder 640 derives the quantization parameter, for example, based on the quantization index and quantization table. It then may decode the vertex coordinate error. The decoded vertex coordinate errors E' are used to refine the instance initially reconstructed at 3D component reconstruction module 630. In particular, adder 650 sums up the decoded coordinate errors (E') and the initial reconstructed instance (C'), for example, as C''=C'+E'. C'' usually provides a more accurate representation of the original instance than the initial reconstructed instance C'.

Several of the implementations and features described in this application may be used in the context of the MPEG 3DGC Standard and its extensions.

The implementations described herein may be implemented in, for example, a method or a process, an apparatus, a software program, a data stream, or a signal. Even if only discussed in the context of a single form of implementation (for example, discussed only as a method), the implementation of features discussed may also be implemented in other forms (for example, an apparatus or program). An apparatus may be implemented in, for example, appropriate hardware, software, and firmware. The methods may be implemented in, for example, an apparatus such as, for example, a processor, which refers to processing devices in general, including, for example, a computer, a microprocessor, an integrated circuit, or a programmable logic device. Processors also include communication devices, such as, for example, computers, cell phones, portable/personal digital assistants ("PDAs"), and other devices that facilitate communication of information between end-users.

Reference to “one embodiment” or “an embodiment” or “one implementation” or “an implementation” of the present principles, as well as other variations thereof, mean that a particular feature, structure, characteristic, and so forth described in connection with the embodiment is included in at least one embodiment of the present principles. Thus, the appearances of the phrase “in one embodiment” or “in an embodiment” or “in one implementation” or “in an implementation”, as well any other variations, appearing in various places throughout the specification are not necessarily all referring to the same embodiment.

Additionally, this application or its claims may refer to “determining” various pieces of information. Determining the information may include one or more of, for example, estimating the information, calculating the information, predicting the information, or retrieving the information from memory.

Further, this application or its claims may refer to “accessing” various pieces of information. Accessing the information may include one or more of, for example, receiving the information, retrieving the information (for example, from memory), storing the information, processing the information, transmitting the information, moving the information, copying the information, erasing the information, calculating the information, determining the information, predicting the information, or estimating the information.

Additionally, this application or its claims may refer to “receiving” various pieces of information. Receiving is, as with “accessing”, intended to be a broad term. Receiving the information may include one or more of, for example, accessing the information, or retrieving the information (for example, from memory). Further, “receiving” is typically involved, in one way or another, during operations such as, for example, storing the information, processing the information, transmitting the information, moving the information, copying the information, erasing the information, calculating the information, determining the information, predicting the information, or estimating the information.

As will be evident to one of skill in the art, implementations may produce a variety of signals formatted to carry information that may be, for example, stored or transmitted. The information may include, for example, instructions for performing a method, or data produced by one of the described implementations. For example, a signal may be formatted to carry the bitstream of a described embodiment. Such a signal may be formatted, for example, as an electromagnetic wave (for example, using a radio frequency portion of spectrum) or as a baseband signal. The formatting may include, for example, encoding a data stream and modulating a carrier with the encoded data stream. The information that the signal carries may be, for example, analog or digital information. The signal may be transmitted over a variety of different wired or wireless links, as is known. The signal may be stored on a processor-readable medium.

CLAIMS

1. A method for generating a bitstream representing a 3D model, comprising the steps of:

accessing (330) a reconstructed instance corresponding to an instance;

determining (350) a quantization parameter based on a vertex coordinate error between a vertex of the instance and a corresponding vertex of the reconstructed instance;

determining a quantization index in response to the determined quantization parameter; and

encoding (360) the quantization index and the vertex coordinate error into the bitstream.

2. The method of claim 1, wherein the determining a quantization index step is based on a syntax that indicates a mapping between a plurality of quantization indices and a plurality of respective quantization parameters.

3. The method of claim 2, further comprising the step of:

determining the mapping between the plurality of quantization indices and the plurality of respective quantization parameters based on statistical data.

4. The method of claim 3, wherein a smaller quantization index corresponds to a more frequent quantization parameter in the statistical data.

5. The method of claim 1, wherein the quantization parameter is determined further in response to a maximum tolerable error.

6. The method of claim 1, wherein the quantization parameter corresponds to at least one of the number of quantization bits and a quantization step size.

7. A method for decoding a bitstream representing a 3D model, comprising the steps of:

accessing (420) a reconstructed instance corresponding to an instance;

determining (430) a quantization index from the bitstream;

determining (430) a quantization parameter in response to the quantization index;

decoding (440) a vertex coordinate error representative of an error between a vertex of the instance and a corresponding vertex of the reconstructed instance; and

refining (450) the reconstructed instance in response to the decoded vertex coordinate error.

8. The method of claim 7, wherein the determining a quantization parameter step is based on a syntax that indicates a mapping between a plurality of quantization indices and a plurality of respective quantization parameters.

9. The method of claim 7, wherein the vertex coordinate error is decoded further responsive to a maximum tolerable error.

10. The method of claim 7, wherein the quantization parameter corresponds to at least one of the number of quantization bits and a quantization step size.

11. An apparatus (500) for generating a bitstream representing a 3D model, comprising:

a 3D component reconstruction module (540) reconstructing a reconstructed instance corresponding to an instance;

a vertex coordinate error quantization parameter estimator (560) determining a quantization parameter based on a vertex coordinate error between a vertex of the instance and a corresponding vertex of the reconstructed instance and determining a quantization index in response to the determined quantization parameter; and

a vertex coordinate error encoder (580) encoding the quantization index and the vertex coordinate error into the bitstream.

12. The apparatus of claim 11, wherein the vertex coordinate error quantization parameter estimator (560) determines a quantization index based on a syntax that indicates a mapping between a plurality of quantization indices and a plurality of respective quantization parameters.

13. The apparatus of claim 12, wherein the vertex coordinate error quantization parameter estimator (560) determines the mapping between the plurality of quantization indices and the plurality of respective quantization parameters based on statistical data.

14. The apparatus of claim 13, wherein a smaller quantization index corresponds to a more frequent quantization parameter in the statistical data.

15. The apparatus of claim 11, wherein the quantization parameter is determined further in response to a maximum tolerable error.

16. The apparatus of claim 11, wherein the quantization parameter corresponds to at least one of the number of quantization bits and a quantization step size.

17. An apparatus (600) for decoding a bitstream representing a 3D model, comprising:

a 3D component reconstruction module (630) reconstructing a reconstructed instance corresponding to an instance;

an entropy decoder (610) determining a quantization index from the bitstream;

a vertex coordinate error decoder (640) determining a quantization parameter corresponding to the quantization index and decoding a vertex coordinate error representative of an error between a vertex of the instance and a corresponding vertex of the reconstructed instance; and

an adder (650) refining the reconstructed instance in response to the decoded vertex coordinate error.

18. The apparatus of claim 17, wherein the vertex coordinate error decoder (640) determines a quantization parameter based on a syntax that indicates a mapping between a plurality of quantization indices and a plurality of respective quantization parameters.

19. The apparatus of claim 17, wherein vertex coordinate error decoder (640) decodes the vertex coordinate error further responsive to a maximum tolerable error.

20. The apparatus of claim 17, wherein the quantization parameter corresponds to at least one of the number of quantization bits and a quantization step size.

21. A computer readable storage medium having stored thereon instructions for generating or decoding a bitstream representing a 3D model according to claims 1-20.

22. A computer readable storage medium having stored thereon a bitstream representing a 3D model generated according to claims 1-10.

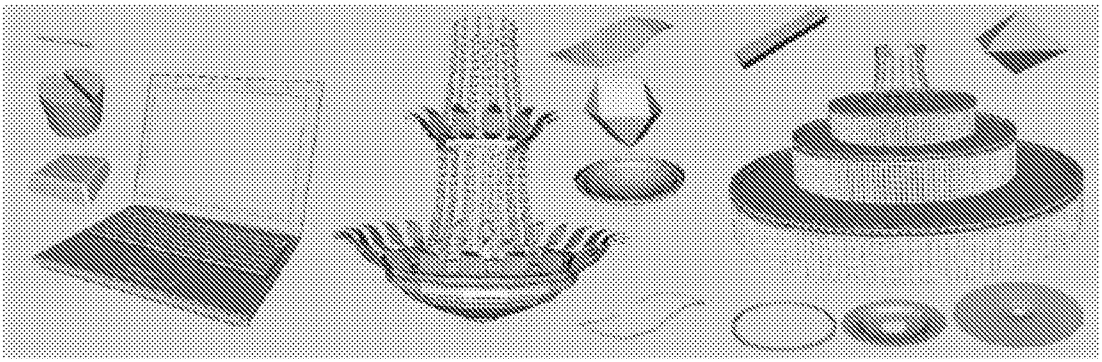

FIG. 1

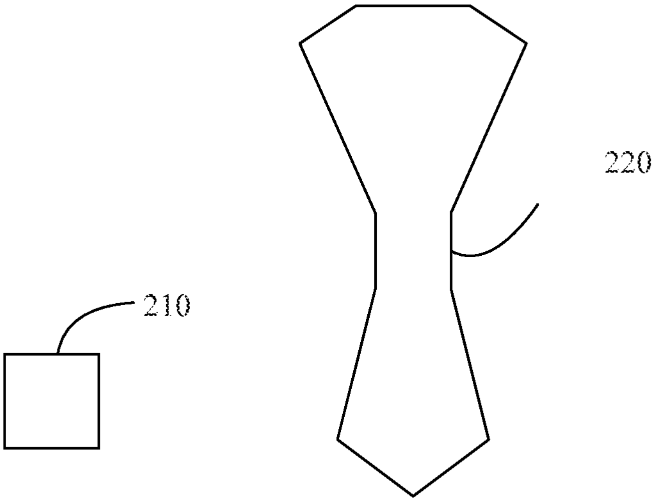

FIG. 2A

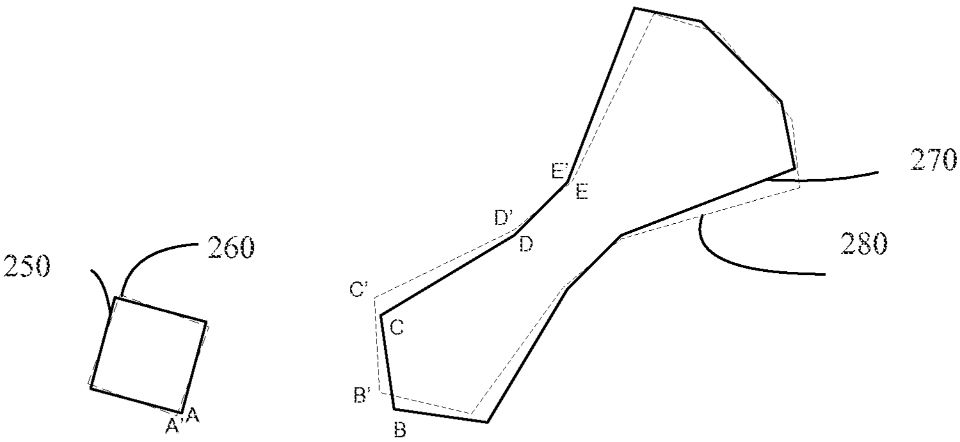

FIG. 2B

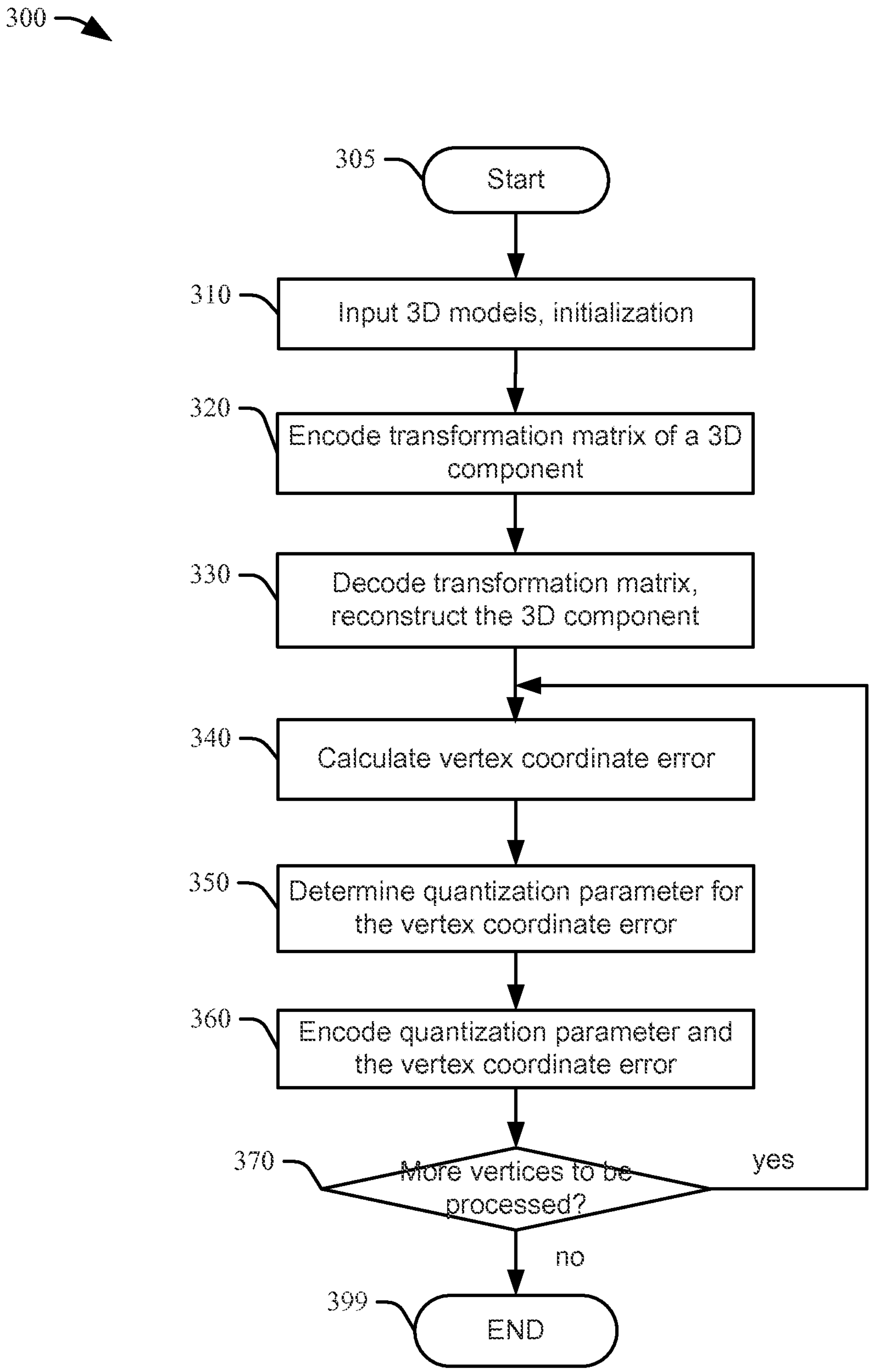
300
305
Start
310
Input 3D models, initialization
320
Encode transformation matrix of a 3D component
330
Decode transformation matrix, reconstruct the 3D component
340
Calculate vertex coordinate error
350
Determine quantization parameter for the vertex coordinate error
360
Encode quantization parameter and the vertex coordinate error
370
More vertices to be processed?
yes
no
399
END

FIG. 3

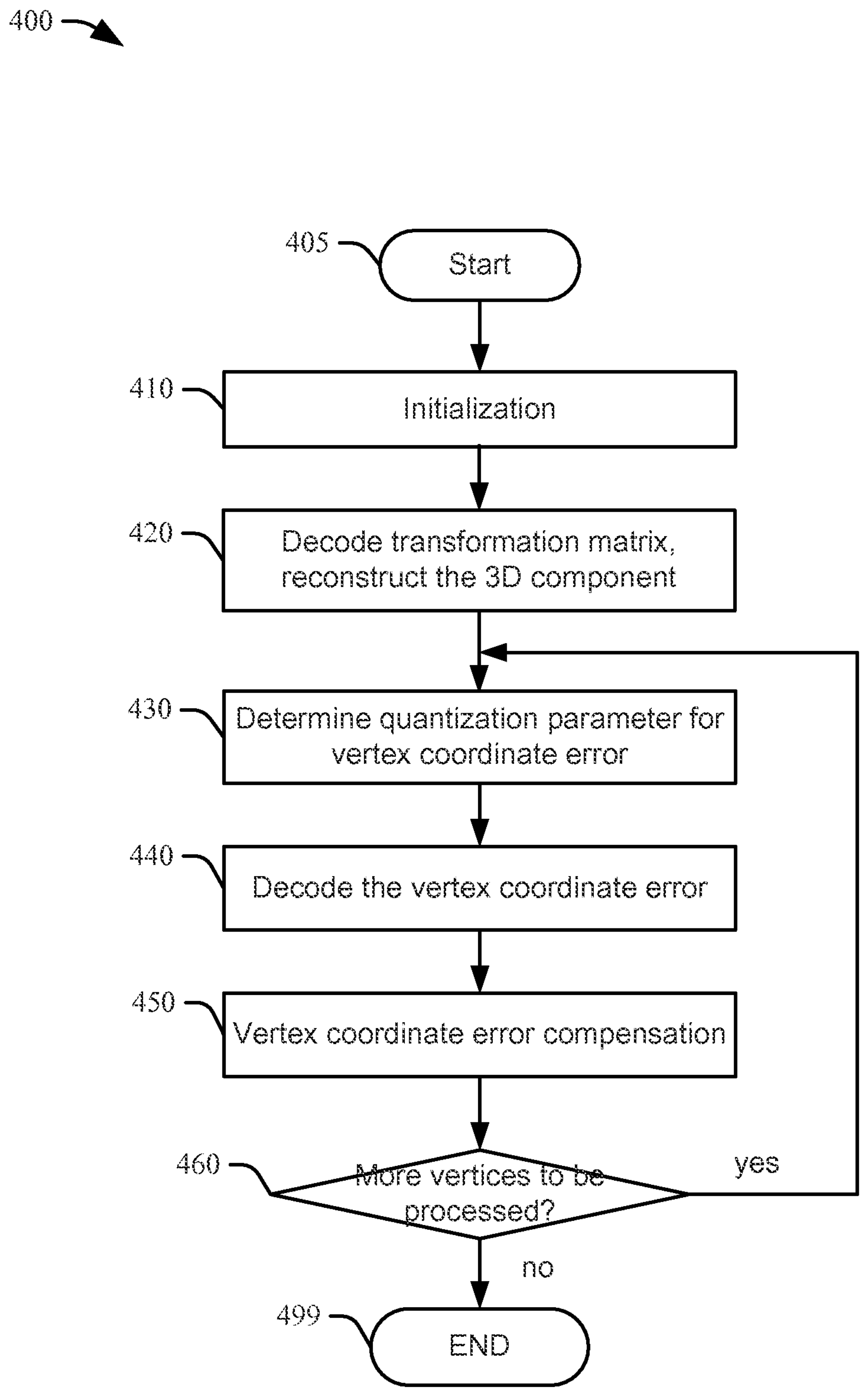
400
405
Start
410
Initialization
420
Decode transformation matrix, reconstruct the 3D component
430
Determine quantization parameter for vertex coordinate error
440
Decode the vertex coordinate error
450
Vertex coordinate error compensation
460
More vertices to be processed?
yes
no
499
END

FIG. 4

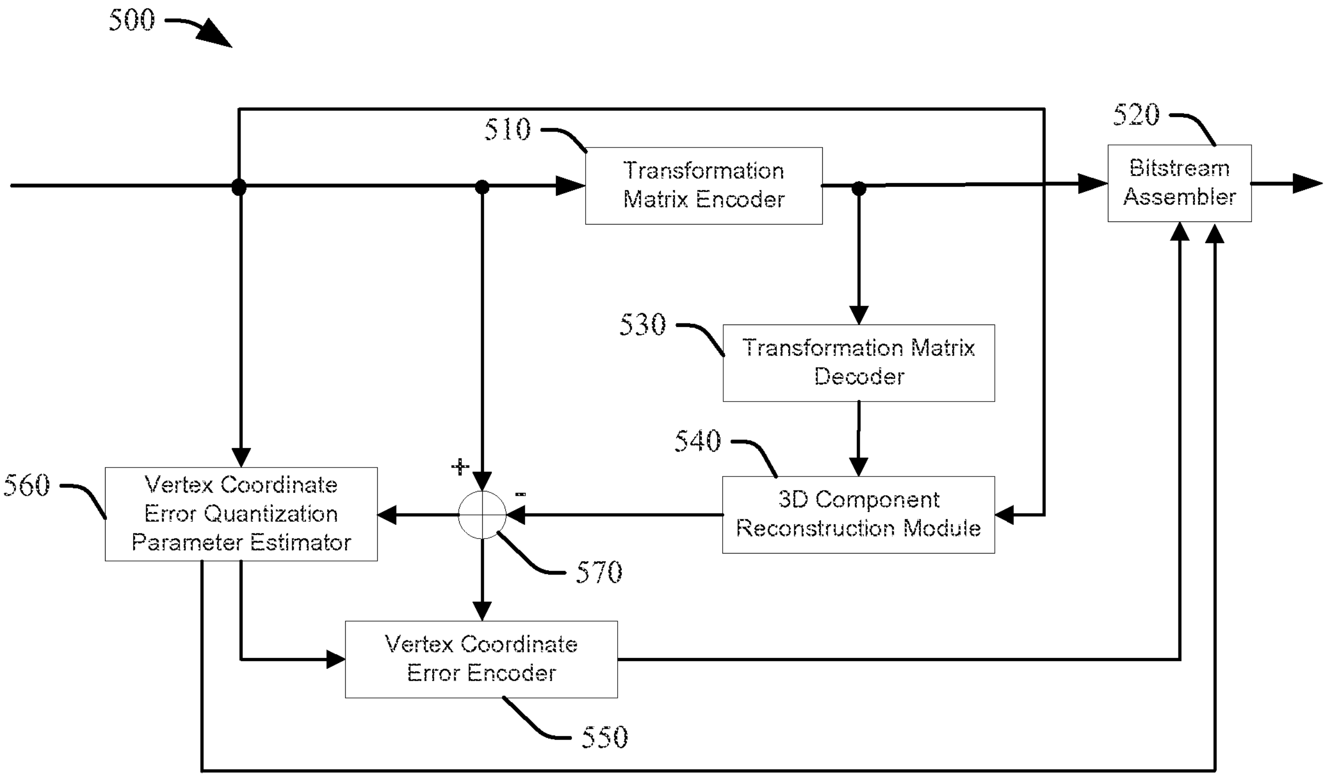
500
510
Transformation Matrix Encoder
520
Bitstream Assembler
530
Transformation Matrix Decoder
540
3D Component Reconstruction Module
560
Vertex Coordinate Error Quantization Parameter Estimator
+
-
570
Vertex Coordinate Error Encoder
550

FIG. 5

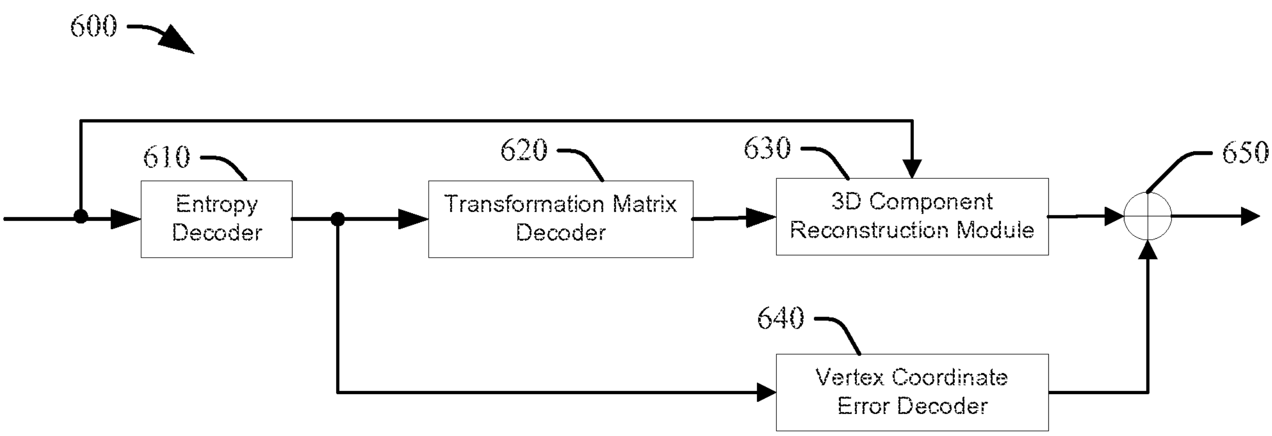
600
610
Entropy Decoder
620
Transformation Matrix Decoder
630
3D Component Reconstruction Module
650
640
Vertex Coordinate Error Decoder

FIG. 6

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2013/077404

A. CLASSIFICATION OF SUBJECT MATTER

G06T 9/00 (2006.01) i

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC: G06T, G06F

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

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

WPI; EPODOC; CPRS; CNKI:

Bitstream, bit stream, 3D, 3-D, three dimensional, 3 dimensional, encod+, cod+, decod+, instance, vertex, quantiz+, correct+, parameter, index, coordinate error

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US6947045B1 (AT&T Corporation) 20 Sep.2005 (20.09.2005) See the whole document	1-22
A	US5905502A (Sun Microsystems, Inc.) 18 May 1999 (18.05.1999) See the whole document	1-22
A	US20120075302A1 (THOMSON LICENSING LLC) 29 Mar.2012 (29.03.2012) See the whole document	1-22
A	US20090184956A1 (SAMSUNG ELECTRONICS CO., LTD., et al) 23 Jul.2009 (23.07.2009) See the whole document	1-22

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

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim (S) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&"document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
09 Oct.2013 (09.10.2013)	**24 Oct. 2013 (24.10.2013)**

Name and mailing address of the ISA/CN
The State Intellectual Property Office, the P.R.China
6 Xitucheng Rd., Jimen Bridge, Haidian District, Beijing, China
100088
Facsimile No. 86-10-62019451

Authorized officer
WEN,Rui
Telephone No. (86-10)6241 2083

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CN2013/077404

Patent Documents referred in the Report	Publication Date	Patent Family	Publication Date
US6947045B1	20.09.2005	None	
US5905502A	18.05.1999	JPH09171568A	30.06.1997
		US6239805B1	29.05.2001
		US5870094A	09.03.1999
		US6532012B2	11.03.2003
		US5867167A	02.02.1999
		US6603470B1	05.08.2003
		JP3212885B2	25.09.2001
		US5793371A	11.08.1998
		US2002050992A1	02.05.2002
		EP0757332B1	06.11.2002
		DE69624636D1	12.12.2002
		EP0757332A2	05.02.1997
		EP0757332A3	14.05.1997
US20120075302A1	29.03.2012	EP2261859A1	15.12.2010
		WO2010142743A3	03.02.2011
		CN102460513A	16.05.2012
		EP2441049A2	18.04.2012
		KR20120036834A	18.04.2012
		WO2010142743A2	16.12.2010
		JP2012529825A	22.11.2012
US20090184956A1	23.07.2009	WO2009093836A3	29.10.2009
		EP2243118A2	27.10.2010
		KR20090080421A	24.07.2009
		WO2009093836A2	30.07.2009
		US8310480B2	13.11.2012