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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
4 July 2013 (04.07.2013)

WIPO | PCT

(10) International Publication Number
WO 2013/101095 Al

(51) International Patent Classification:
*H04N 7/24* (201 1.01) *G09G 5/00* (2006.01)

(21) International Application Number:
PCT/US201 1/067902

(22) International Filing Date:
29 December 201 1 (29. 12.201 1)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant *(for all designated States except US):* **INTEL CORPORATION** [US/US]; 2200 Mission College Boulevard, Santa Clara, California 75052 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only):* **AKEN-INE-MOLLER, Tomas G.** [SE/SE]; Rosenvagen 13, Lund, S-227 38 M. Sweden (SE). **NILSSON, Jim K.** [SE/SE]; Svenska vagen 4, Lund, S-226 39 M. Sweden (SE). **ANDERSSON, Magnus** [SE/SE]; Kaprifolgatan 37, Helsingborg, S-25375 M, Sweden (SE). **HASSELGREN, Jon N.** [SE/SE]; Sparvvagen 4, Bulkelflo strand, S-21832 M, Sweden (SE).

(74) Agent: **TROP, Timothy N.;** 161 6 S. Voss Rd., Ste. 750, Houston, Texas 77057-263 1 (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, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

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

**Declarations under Rule 4.17:**

— *as to the identity of the inventor (Rule 4.17(i))*

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(H))*

— *of inventorship (Rule 4.17(iv))*

**Published:**

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

**(54) Title:** VARIABLE DEPTH COMPRESSION

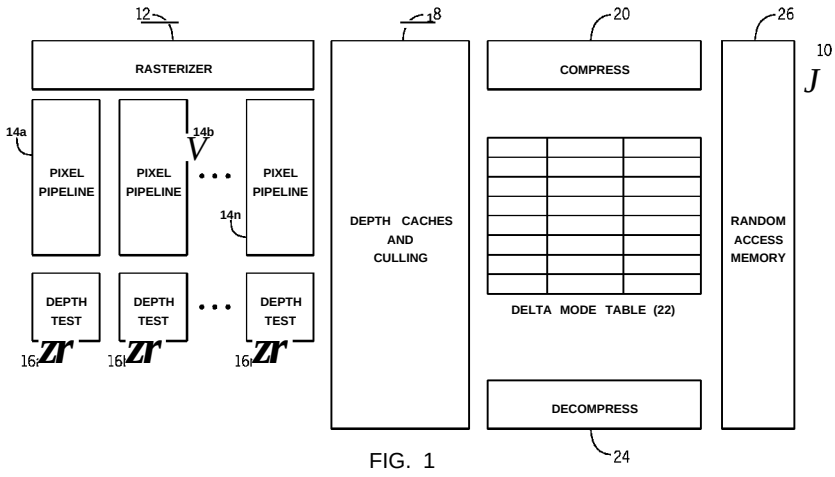


FIG. 1

**(57) Abstract:** In accordance with some embodiments, the number of bits allocated to depth compression may be changed variably based on a number of considerations. As a result, depth data may be compressed in a more efficient way.

WO 2013/101095 A1

# VARIABLE DEPTH COMPRESSION

## Background

[0001] This relates generally to graphics processing and, particularly, to compression of depth buffers for stochastic motion blur rasterization.

[0002] Motion blur rasterization attempts to more accurately represent moving objects and, particularly, to represent the blur that is observed when objects move fast enough. Ignoring motion blur, information about the depths of primitives within the scene may be readily compressed. Particularly, the depth of each primitive may be determined as a distance from the imaging device to the primitive. Algorithms exist for reducing this depth information when motion blur is not involved.

[0003] However, when motion blur is involved, it is a much more complex operation to attempt to compress depth information.

[0004] Graphics processors tend to be extremely sensitive to increased power consumption with increased memory bandwidth usage. Memory bandwidth is also in itself a scarce resource on modern graphics processors. Currently, there is more emphasis on obtaining stochastic rasterization of motion blur and depth of field and to obtain this information on an interactive or even real time rendering basis. This involves significant memory bandwidth usage, implicating increased power consumption.

## Brief Description Of The Drawings

[0005] Some embodiments are described with respect to the following figures:

Figure 1 is a schematic depiction of one embodiment of the present invention;

Figure 2 is a flow chart for mode selection for anchor points in accordance with one embodiment;

Figure 3 is a flow chart for dynamic residual value bit allocation in accordance with one embodiment; and

Figure 4 is a system depiction for one embodiment.

Detailed Description

[0006] In accordance with some embodiments, the number of bits allocated to depth compression may be changed variably based on a number of considerations. As a result, depth data may be compressed in a more efficient way.

[0007] Depth buffer compression is one technique to reduce memory bandwidth usage. Typically, the depth z of any primitive within a tile is linear over the primitive, which typically is a triangle. This may be exploited to construct inexpensive compression and decompression algorithms.

[0008] Plane encoding attempts to take advantage of this linear relationship. The rasterizer feeds exact plane equations to a compressor and, therefore, no residual terms are needed. However, when motion blur is involved, each vertex may move according to a linear function. Then the depth function at a sample is a rational cubic function. This function makes it substantially harder to predict depth over an entire tile using a simple predictor function. As a consequence, the standard depth buffer compression techniques, especially those exploiting exact plane equations, may fail to compress such noisy buffers.

[0009] A block of pixels, called a tile, may be processed independently. Current depth compression schemes do not handle motion blur and depth of field explicitly and, therefore, they do not have the time component or lens parameters.

[001 0] Generally, depth compression algorithms use three common steps which may be called clustering, predictor function generation, and residual encoding. Clustering is used when sets of samples in a tile belong to different layers, for example a background layer and a foreground layer. In this case, it is hard to compress all depths in the tile using the same predictor function.

[001 1] The clustering step attempts to separate the samples of the tile into two or more layers, where the samples in each layer typically share some characteristic, such as lying in a common plane. A goal of splitting the samples into two or more layers is that each layer may ideally become simpler to compress compared to compressing all the samples in a single layer.

[001 2] In predictor function generation, each layer generates its own predictor function. A goal is to use the depth samples and possibly their fixed coordinates to create a predictor function to predict the depth of each sample using an inexpensive function. For example, assume that a rectangle with small per pixel displacements has been rendered to a tile. As a predictor function, one may use the plane of the rectangle, since it is probably a good guess on where the displaced depths will lie. Even if the guess is not perfect, imperfections can be handled in the next step.

[001 3] Residual encoding enables more exact depths to be reconstructed during decompression of the tile, since a common requirement of graphics processors is that the depth buffer be non-lossy. The residual is the difference between the predictor function and the actual sample depths. Given a good predictor function, the residuals between the depth of the samples and the predictor function may be small. As a consequence, those residuals can be encoded using relatively few bits. Then good compression ratios can be achieved (if there are a small number of layers) and storage needed for the predictor function is small.

[0014] In some embodiments, the depth buffer compression may use anchor encoding, but other techniques may be used as well, including plane encoding. The construction of plane equations and plane encoding, based on plane data derivatives or delta vectors, fails to handle extreme planes in the sense that the true derivatives are very large compared to the allocated bit budget for storing these derivatives. Consequently, the actual plane representation accuracy may be reduced. This accuracy reduction increases with distance within the tile from the point where the delta vectors are evaluated. On the other hand, naively allocating too many bits for delta vectors reduces the number of bits available for storing depth value residuals.

[001 5] Thus, in some embodiments, the impact of statically assigning available bits to delta vectors may be reduced by allowing dynamic allocation of bits depending on the nature of the depth data.

[001 6] Depending on the rasterized geometry, a tile can be the target of a multitude of primitives, resulting in a complex distribution of depth values over the time. Using one or more plane equations as a base prediction of these depth values, the residual values are encoded in the compressed depth data format. As a consequence of the nature of the predictor planes, predicted depth values may be more or less correct, resulting in varying need for residual correction bits.

[001 7] Current depth compression mechanisms statically allocate an equal number of residual bits for all tile positions, which may potentially be a poor match for actual depth values. In some embodiments, the impact of statically assigning available bits to residual correction bits is reduced by, instead, allowing for dynamically allocating the number of residual bits for individual tile positions.

[001 8] In an anchor encoding embodiment, a depth data compression mechanism detects plane representations of rasterized primitives by picking one or more points in the tile as candidate anchor points. The evaluation of an anchor point includes the calculation of local and planar X and Y derivatives based on depth data values. A predictor plane representation ($zp(x,y) = a + b^*x + c^*y$) is encoded in the compressed depth data as three values: a, b, and c, where a is the depth value at the anchor point, b is dZ/dX, and c is dZ/dY. The values b and c are the delta values or "vectors".

[001 9] For each depth, z(x,y), in the tile, residual depth values are stored as the difference between the predicted and the true depth values. The value d is a residual value.

[0020] By selecting the number of bits allocated for delta and residual values, the total bit budget for the tile depth data can be reduced without loss of the depth value precision. A net compression of depth data may be achieved in some embodiments.

[0021] We introduce the notion of delta modes, which designate a particular combination of X, Y, and R, where X is the number of bits for the delta-X vectors, Y is the number of bits for the delta-Y vectors, an R is the number of residual bits per depth. At compression, these modes are available for selection for each anchor point in the tile.

[0022] In one variation, each anchor point may potentially have its own set of available modes, where the modes are dynamically created during and guided by the results of compression.

[0023] In another variation, to simplify encoding, the same number of bits can be used for both directions (X and Y). Below, we consider a mode table to be statically created and stored before compression begins. See Table 1 for an example set of delta modes where B = 512, N = 4, M = 3, A = 32, T = 32.

| Mode | X | Y | R |
|---|---|---|---|
| 1 | 3 | 3 | 14 |
| 2 | 6 | 6 | 13 |
| 3 | 9 | 9 | 11 |
| 4 | 8 | 10 | 11 |
| 5 | 10 | 8 | 11 |
| 6 | 13 | 13 | 10 |
| 7 | 11 | 17 | 9 |
| 8 | 17 | 11 | 9 |

Table 1. Example delta mode table.

[0024] Given the tile dimensions and bit budget for compression, there is a tradeoff between the number of bits available for delta values and residual values. Not all combinations of X, Y, and R, are relevant. It is only meaningful to reduce X and/or Y if R can be increased by at least one bit. This restricts the number of available modes.

[0025] The following condition has to be met to enable compression of a tile:

$$T1og2(N)+ N^{*} (M + A + X + Y) + (T - N^{*} 3)^{*} R < B$$

where

B = total bit budget to enable compression

N = number of anchor points

M = log2(number of modes) bits to represent the used mode for an anchor point

A = number of bits used for anchor point depth value

X = number of bits used for X delta

Y = number of bits used for Y delta

T = total number of depths in tile

R = number of bits used for residual depth values.

[0026] The $T^{*}log2(N)$-term reserves log2(N) bits per depth in a tile in order to indicate which plane equation a depth is using. The $(T - N^{*} 3)$ term implies that the anchor point depth, as well as the first-order derivatives (dZ/dX, and dZ/dY) are exact, i.e., they do not require storage of residual bits. In another embodiment, neither the anchor point, nor the first order derivatives, are exact, which would require residual bits for those depths as well, making the last term of the above inequality be T instead of $(T - N^{*} 3)$. If the anchor point is exact but not the derivatives, the term instead becomes (T - N). In the example above, the anchor value as well as derivatives land exactly on the correct depth value, which often is a reasonable assumption.

[0027] The algorithm for selecting the mode for each anchor point is as follows, according to one embodiment:

[0028] A table with 2M rows is constructed that stores a number of allocated delta bits for each direction (X and Y). See Table 1 for example. Then, N anchor point positions are given, where N = 1, 2, 3, ..., T (usually between 1 and 4). Next, for each anchor point, the minimum required number of bits to represent the delta vectors is calculated. This may be done by simply computing the delta vectors from the anchor point's depth to the right (or left) neighbor for dZ/dX, and to the upper (or

lower) neighbor for dZ/dY, and examining how many bits are needed (by finding the first and most significant set bit). The corresponding mode with the maximum number of residual bits is selected as the mode for this anchor point.

[0029] An advantage of this simple scheme is that it allows more tiles to be compressed, which, in the end, reduces the memory bandwidth usage for depth (which is a significant consumer of bandwidth in a graphics processor). For example, a tile with large delta vectors which at the same time only needs a few residual bits per depth can be compressed. Similarly, a tile with small delta vectors which require many residual bits can be compressed as well.

[0030] Due to insufficient accuracy and/or precision, in turn due to extreme plane representations or simply complex tile data, plane predictors need residual bits per tile position to adjust to correct depth values. Depending on the amount of inaccuracy in the prediction, a varying number of residual bits are actually needed to encode the difference between the prediction and the correct value.

[0031] We propose the use of a residual mode mask with one entry per tile location to store an indicator of how many residual bits are needed for that location. The aim is to use as few bits as possible for each location, resulting in a total minimum of required bits.

[0032] The possible number of residual bits used for each location can be statically assigned with appropriate values, or dynamically calculated based on tile data.

[0033] For dynamically assigned residual modes, the algorithm works as follows, in one embodiment:

[0034] During compression of a tile, the predictor depth is compared with the actual depth and calculate residual value. Then the shortest mode (i.e., the mode that captures required difference, with the least number of bits) is selected to encode the residuals. Next, the corresponding mode value is filled in the residual mode mask. Finally, the residual mode mask is stored in the encoded tile data.

[0035] The technique may be illustrated with an example.

| Mode | Residual bits |
|---|---|
| 00=0 | 2 |
| 01=1 | 8 |
| 10=2 | 12 |
| 11=3 | 16 |

Table 2.

Example residual mode table with four modes.

| | X0 | X1 | X2 | X3 |
|---|---|---|---|---|
| Y0 | 00 | 00 | 01 | 10 |
| Y1 | 00 | 01 | 11 | 00 |
| Y2 | 00 | 10 | 01 | 00 |
| Y3 | 01 | 00 | 00 | 00 |

Table 3.

Example tile of 4x4 pixels with corresponding encoding.

[0036] In the example, the total number of bits for residuals is 90 (9x2+4x8+2x1 2+1 6) because there are nine instances of mode 00 which uses 2 bits, etc. Storing full value residuals results in 256 bits (16x16).

[0037] In this example, we have a 2-bit value for each depth value. However, one may also choose to use a 2-bit value for each 2x2 depths (in order to reduce the number of 2-bit values used). In general each WxH depth can use Q bits to indicate the mode.

[0038] For static assignment, we can supply several modes per tile to choose from. For example, one mode can be that the residual bits are spread out over the depths in a totally uniform way. Another mode may use fewer bits closer to the anchor points and more bits the farther away from the anchor point the depth is located in the tile.

[0039] While an embodiment using an anchor encoding-based compression technique has been described, dynamic residual bit allocation can also be used for any compressor that encodes residuals. The dynamic delta value bit allocation is also possible to use with other predictors. For example, if we use a bilinear patch (which is created from four depth values plus their x,y-positions) as a predictor, we can encode that patch as one anchor point, and two delta-vectors, and then a residual value for the fourth point, which is the difference between the plane equation from the first three points and the fourth point. The number of bits spent on these two delta vectors, and the residual bits for the fourth point can be dynamically assigned in a similar way as described above for anchor encoding. The same can easily apply to other predictor functions as well.

[0040] Referring to Figure 1, an apparatus 10 may include a rasterizer and a series of pixel pipelines 14a-14n. A depth test unit 16a-16n may be provided for each pixel pipeline. The depth caches and culling unit 18 may be used for depth value culling. Compression may be done at a compressor 20 and decompression may be done at a decompressor 24. A delta mode table 22 (like Table 1) may be provided as well. Finally, random access memory 26 may be provided for the compression and decompression engines.

[0041] Referring to Figure 2, a mode selection sequence 30 for an anchor point may be implemented in hardware, software, and/or firmware. In software and firmware embodiments, it may be implemented by computer executed instructions stored in a non-transitory computer readable medium, such as a magnetic, optical, or semiconductor memory.

[0042] The mode selection for the anchor points begins by constructing a table for the delta bits, as indicated in block 32. Then the anchor point positions are given in block 34. Finally, the bits to represent each delta vector for each anchor point are calculated in block 36.

[0043] A sequence 38 for dynamic residual value bit allocation may be implemented in software, firmware, and/or hardware. In software and firmware embodiments, it may be implemented by computer executed instructions stored in a

non-transitory computer readable medium, such as an optical, magnetic, or semiconductor memory.

[0044] The sequence beings by comparing a predicted depth to the actual depth, as indicated in block 40. Then the residual value is calculated, as shown in block 42.

[0045] The shortest mode to encode the residual value bits is then selected (block 44). The mode value is filled in in the residual mode mask (block 46). Finally, the residual mode mask is stored in the encoded tile data, as indicated in block 48.

[0046] The computer system 130, shown in Figure 4, may include a hard drive 134 and a removable medium 136, coupled by a bus 104 to a chipset core logic 110. The computer system may be any computer system, including a smart mobile device, such as a smart phone, tablet, or a mobile Internet device. A keyboard and mouse 120, or other conventional components, may be coupled to the chipset core logic via bus 108. The core logic may couple to the graphics processor 112, via a bus 105, and the central processor 100 in one embodiment. The graphics processor 112 may also be coupled by a bus 106 to a frame buffer 114. The frame buffer 114 may be coupled by a bus 107 to a display screen 118. In one embodiment, a graphics processor 112 may be a multi-threaded, multi-core parallel processor using single instruction multiple data (SIMD) architecture.

[0047] In the case of a software implementation, the pertinent code may be stored in any suitable semiconductor, magnetic, or optical memory, including the main memory 132 (as indicated at 139) or any available memory within the graphics processor. Thus, in one embodiment, the code to perform the sequences of Figures 2 and 3 may be stored in a non-transitory machine or computer readable medium 130, such as the memory 132, and/or the graphics processor 112, and/or the central processor 100 and may be executed by the processor 100 and/or the graphics processor 112 in one embodiment.

[0048] Figures 2 and 3 are flow charts. In some embodiments, the sequences depicted in these flow charts may be implemented in hardware, software, or firmware. In a software embodiment, a non-transitory computer readable medium,

such as a semiconductor memory, a magnetic memory, or an optical memory may be used to store instructions and may be executed by a processor to implement the sequences shown in Figures 2 and 3.

[0049] The graphics processing techniques described herein may be implemented in various hardware architectures. For example, graphics functionality may be integrated within a chipset. Alternatively, a discrete graphics processor may be used. As still another embodiment, the graphics functions may be implemented by a general purpose processor, including a multicore processor.

[0050] References throughout this specification to "one embodiment" or "an embodiment" mean that a particular feature, structure, or characteristic described in connection with the embodiment is included in at least one implementation encompassed within the present invention. Thus, appearances of the phrase "one embodiment" or "in an embodiment" are not necessarily referring to the same embodiment. Furthermore, the particular features, structures, or characteristics may be instituted in other suitable forms other than the particular embodiment illustrated and all such forms may be encompassed within the claims of the present application.

[0051] While the present invention has been described with respect to a limited number of embodiments, those skilled in the art will appreciate numerous modifications and variations therefrom. It is intended that the appended claims cover all such modifications and variations as fall within the true spirit and scope of this present invention.

What is claimed is:

1. A method comprising:

using a graphics processor to compress depth data by varying a number of bits used to encode residual values depending on the depth data.

2. The method of claim 1 including varying a number of bits used to enable delta values.

3. The method of claim 1 including using anchor point encoding.

4. The method of claim 2 including selecting a number of bits for X vectors and Y vectors and assigning a number of residual bits per depth.

5. The method of claim 4 including enabling the selection of each of said bit numbers for X vectors, Y vectors, and residual bits per depth.

6. The method of claim 2 including providing a plurality of selectable modes, each mode specifying a particular number of bits for X vectors, Y vectors, and residual bits per depth.

7. The method of claim 6 including calculating, for an anchor point, a minimum number of bits needed to represent the delta vectors.

8. The method of claim 7 including selecting a mode with a maximum number of residual bits that provides the needed number of bits for the delta vectors.

9. The method of claim 8 including storing an indicator of how many residual bits are needed for a given anchor point.

10. The method of claim 2 including reducing the number of bits for X or Y values only if the number of bits for the residual value can be increased correspondingly.

11. A non-transitory computer readable medium storing instructions executed by a processor to:

compress depth data by varying a number of bits used to encode residual values depending on the depth data.

12. The medium of claim 11 further storing instructions to vary a number of bits used to enable delta values.

13. The medium of claim 11 further storing instructions to use anchor point encoding.

14. The medium of claim 12 further storing instructions to select a number of bits for X vectors and Y vectors and assigning a number of residual bits per depth.

15. The medium of claim 14 further storing instructions to enable the selection of each of said bit numbers for X vectors, Y vectors, and residual bits per depth.

16. The medium of claim 12 further storing instructions to provide a plurality of selectable modes, each mode specifying a particular number of bits for X vectors, Y vectors, and residual bits per depth.

17. The medium of claim 16 further storing instructions to calculate, for an anchor point, a minimum number of bits needed to represent the delta vectors.

18. The medium of claim 17 further storing instructions to select a mode with a maximum number of residual bits that provides the needed number of bits for the delta vectors.

19. The medium of claim 18 further storing instructions to store an indicator of how many residual bits are needed for a given anchor point.

20. The medium of claim 12 further storing instructions to reduce the number of bits for X or Y values only if the number of bits for the residual value can be increased correspondingly.

21. An apparatus comprising:

a processor to compress depth data by varying a number of bits used to encode residual values depending on the depth data; and

a storage coupled to said processor.

22. The apparatus of claim 21, said processor to vary a number of bits used to enable delta values.

23. The apparatus of claim 21, said processor to use anchor point encoding.

24. The apparatus of claim 22, said processor to select a number of bits for X vectors and Y vectors and assigning a number of residual bits per depth.

25. The apparatus of claim 24, said processor to enable the selection of each of said bit numbers for X vectors, Y vectors, and residual bits per depth.

26. The apparatus of claim 22, said processor to provide a plurality of selectable modes, each mode specifying a particular number of bits for X vectors, Y vectors, and residual bits per depth.

27. The apparatus of claim 26, said processor to calculate, for an anchor point, a minimum number of bits needed to represent the delta vectors.

28. The apparatus of claim 27, said processor to select a mode with a maximum number of residual bits that provides the needed number of bits for the delta vectors.

29. The apparatus of claim 28, said processor to store an indicator of how many residual bits are needed for a given anchor point.

30. The apparatus of claim 22, said processor to reduce the number of bits for X or Y values only if the number of bits for the residual value can be increased correspondingly.

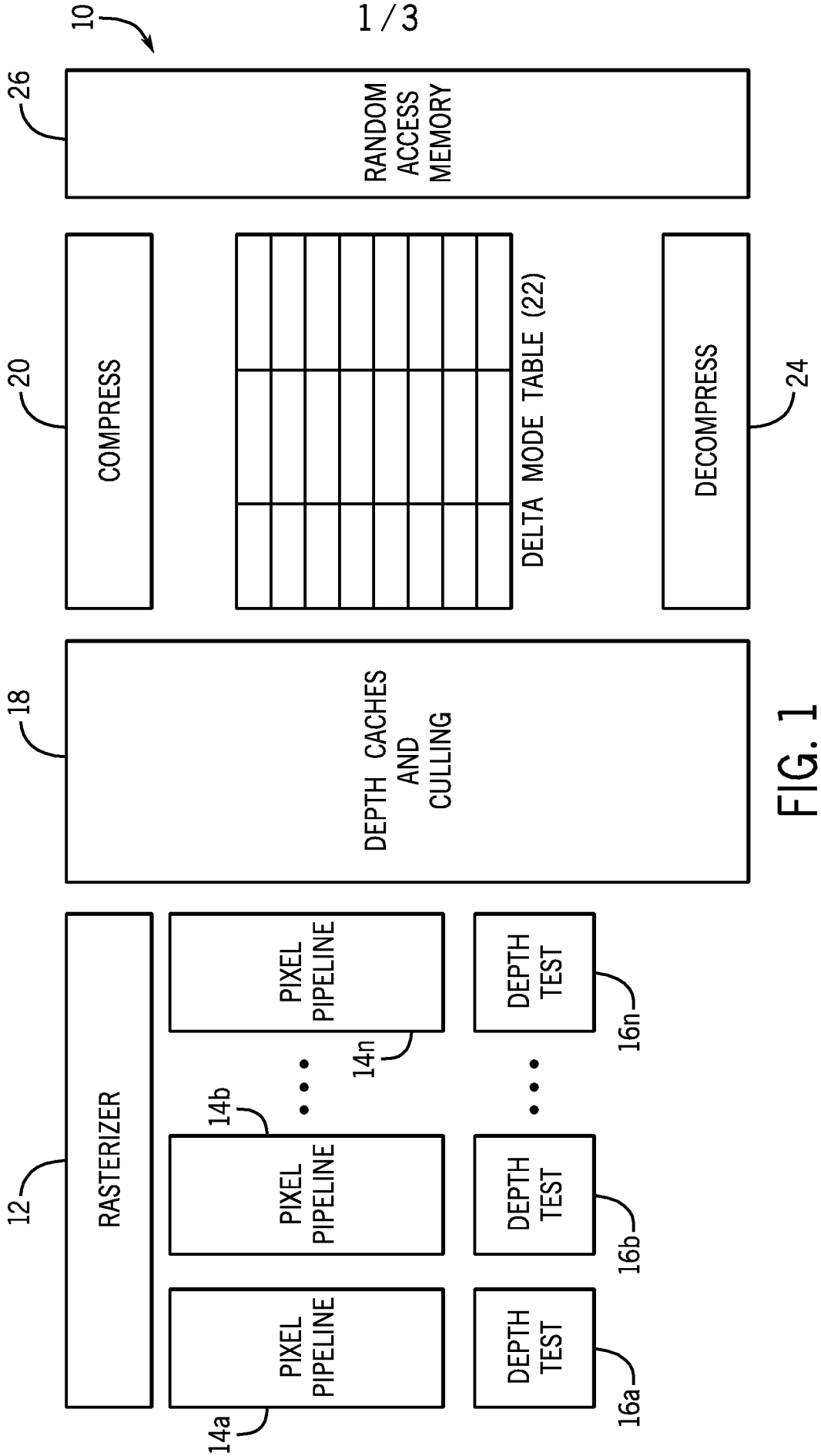
10
26
RANDOM ACCESS MEMORY
20
COMPRESS
DELTA MODE TABLE (22)
DECOMPRESS
24
18
DEPTH CACHES AND CULLING
12
RASTERIZER
PIXEL PIPELINE
14a
14b
14n
DEPTH TEST
16a
16b
16n

FIG. 1

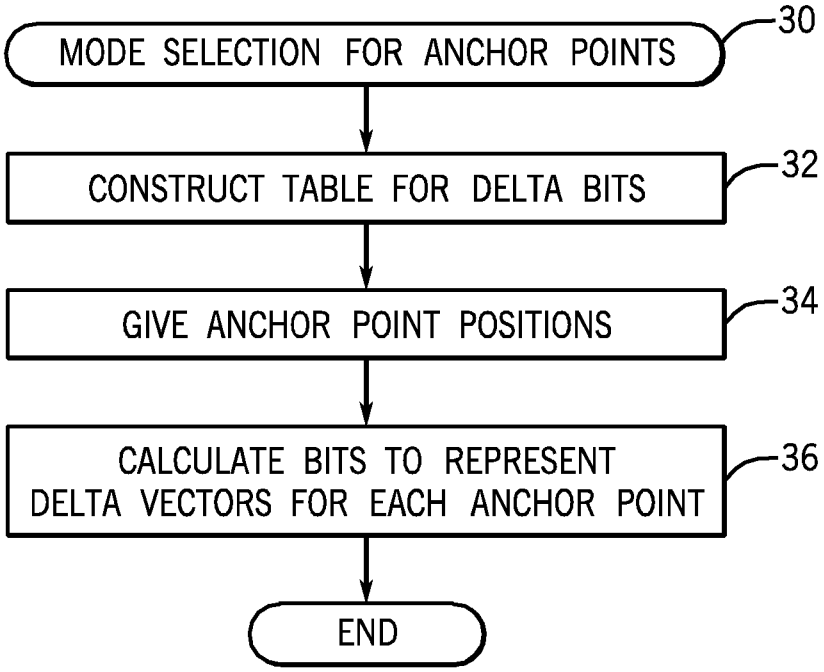
MODE SELECTION FOR ANCHOR POINTS
30
CONSTRUCT TABLE FOR DELTA BITS
32
GIVE ANCHOR POINT POSITIONS
34
CALCULATE BITS TO REPRESENT DELTA VECTORS FOR EACH ANCHOR POINT
36
END

FIG. 2

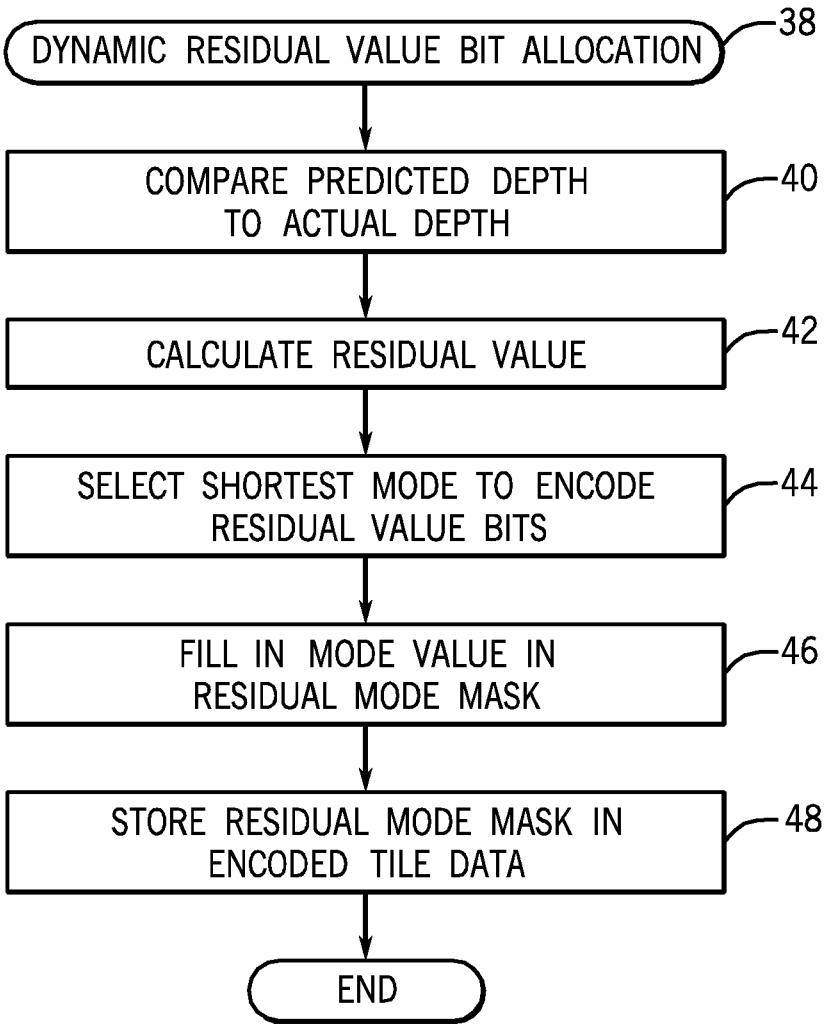
DYNAMIC RESIDUAL VALUE BIT ALLOCATION
38
COMPARE PREDICTED DEPTH TO ACTUAL DEPTH
40
CALCULATE RESIDUAL VALUE
42
SELECT SHORTEST MODE TO ENCODE RESIDUAL VALUE BITS
44
FILL IN MODE VALUE IN RESIDUAL MODE MASK
46
STORE RESIDUAL MODE MASK IN ENCODED TILE DATA
48
END

FIG. 3

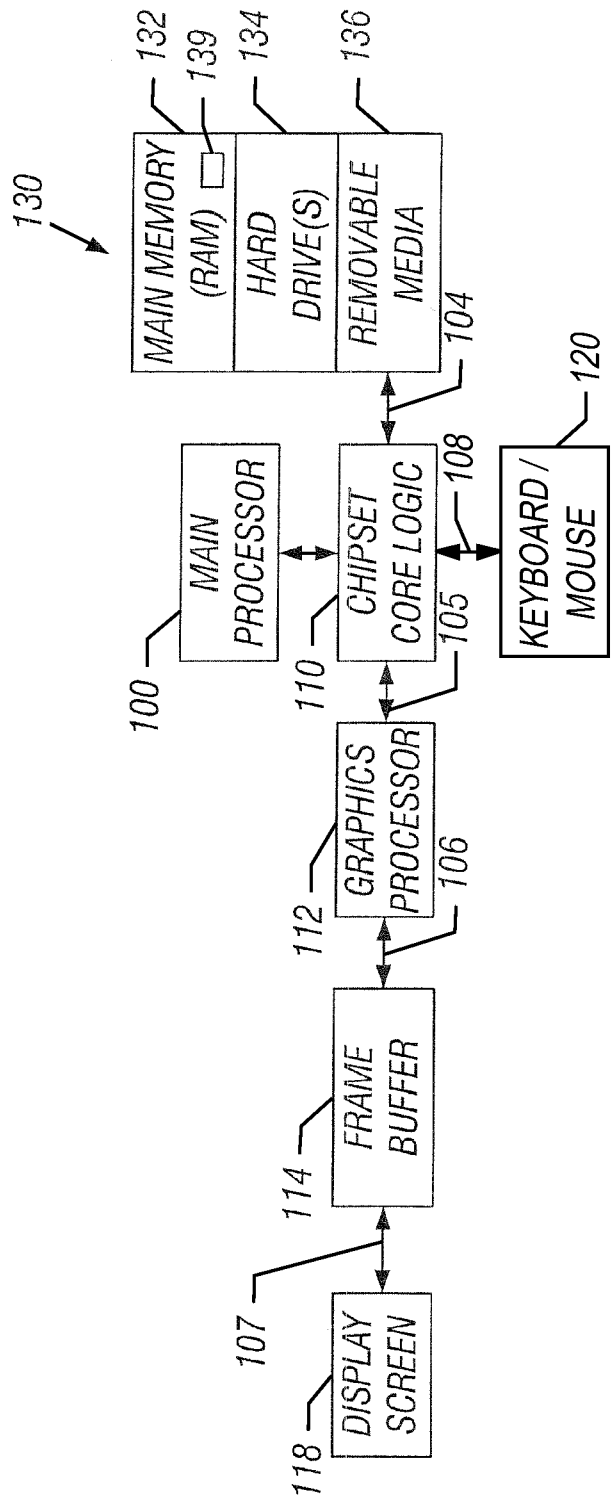
130
MAIN MEMORY (RAM)
132
139
HARD DRIVE(S)
134
REMOVABLE MEDIA
136
104
MAIN PROCESSOR
100
CHIPSET CORE LOGIC
110
105
108
KEYBOARD / MOUSE
120
GRAPHICS PROCESSOR
112
106
FRAME BUFFER
114
107
DISPLAY SCREEN
118

FIG. 4

**INTERNATIONAL SEARCH REPORT**

International application No.
**PCT/US2011/067902**

**A. CLASSIFICATION OF SUBJECT MATTER**

***H04N 7/24(2011.01)i, G09G 5/00(2006.01)1***

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)

H04N 7/24; G06K 9/36; H04N 13/00; H04N 7/26; H04N 7/32; H04N 7/50

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

Korean utility models and applications for utility models

Japanese utility models and applications for utility models

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

eKOMPASS(KIPO internal) & Keywords: encoding, residual value, depth data

**C. DOCUMENTS CONSIDERED TO BE RELEVANT**

| Category * | Citation of document, with indication, where appropriate, of the relevant passages | Relevant to claim No. |
|---|---|---|
| X<br>A | KR 10-2010-0083980 A (SAMSUNG ELECTRONICS CO LTD. et al.) 23 July 2010<br>See abstract ; figures 4, 5; paragraphs 0021-0043 ; claims 1 and 2 | 1,11,21<br>2-10 ,12-20 ,22-30 |
| A | US 2011-0206288 Al (LEE JAEJOON et al.) 25 August 2011<br>See abstract ; figures 7, 8; paragraphs 0086-0102 ; Claims 27 and 28 | 1-30 |
| A | KR 10-2011-0124447 A (SAMSUNG ELECTRONICS CO LTD.) 17 November 2011<br>See abstract ; figure 9; paragraphs 0100-0104 ; Claims 1 and 4 | 1-30 |
| A | US 2010-0202535 Al (FANG PING et al.) 12 August 2010<br>See abstract ; figure 4; paragraphs 0091-0098 ; Claim 1 | 1-30 |

☐ 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 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 |
|---|---|
| 20 SEPTEMBER 2012 (20.09.2012) | **21 SEPTEMBER 2012 (21.09.2012)** |

| Name and mailing address of the ISA/KR | Authorized officer |
|---|---|
| Korean Intellectual Property Office<br>189 Cheongsa-ro, Seo-gu, Daejeon Metropolitan City, 302-701, Republic of Korea | CHO, Woo Yeon |
| Facsimile No. 82-42-472-7140 | Telephone No. 82-42-481-8524 |

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

**INTERNATIONAL SEARCH REPORT**

Information on patent family members

International application No.

**PCT/US201 1/067902**

| Patent document cited in search report | Publication date | Patent family member(s) | Publication date |
|---|---|---|---|
| KR 10-20 10-0083980 A | 23 .07 .20 10 | None | |
| US 201 1-0206288 A1 | 25 .08 .2011 | EP 2360927 A2 | 24. 08 .201 1 |
| | | EP 2360927 A3 | 28 .09 .201 1 |
| | | KR 10-20 11-0093532 A | 18 .08 .201 1 |
| KR 10-20 11-0 124447 A | 17 .11.20 11 | EP 2387244 A2 | 16 .11.201 1 |
| | | US 2011-028049 1 A1 | 17 .11.201 1 |
| US 2010-0202535 A1 | 12 .08 .20 10 | CN 10 14151 14 A | 22 .04 .2009 |
| | | CN 10 14151 14 B | 25 .08 .2010 |
| | | EP 2209320 A1 | 21.07 .2010 |
| | | EP 2209320 A4 | 01.12 .2010 |
| | | W0 2009-052730 A1 | 30 .04 .2009 |