(12) BẢN MÔ TẢ SÁNG CHẾ THUỘC BẰNG ĐỘC QUYỀN SÁNG CHẾ

(19) **Cộng hòa xã hội chủ nghĩa Việt Nam (VN)**
CỤC SỞ HỮU TRÍ TUỆ

(11) **1-0044878**

(51)[2022.01] **H04N 19/597**; H04N 19/169; H04N 19/174

(13) **B**

(21) 1-2021-06547 (22) 23/06/2020
(86) PCT/US2020/070167 23/06/2020 (87) WO2020/264553 30/12/2020
(30) 62/867,063 26/06/2019 US; 62/904,384 23/09/2019 US; 62/910,387 03/10/2019 US; 16/909,642 23/06/2020 US
(45) 25/04/2025 445 (43) 25/04/2022 409A
(71) TENCENT AMERICA LLC (US)
2747 Park Boulevard, Palo Alto, California 94306, United States of America
(72) ZHANG, Xiang (CN); GAO, Wen (US); YEA, Sehoon (KR); LIU, Shan (US).
(74) Công ty TNHH một thành viên Sở hữu trí tuệ VCCI (VCCI-IP CO.,LTD)

(54) PHƯƠNG PHÁP VÀ THIẾT BỊ GIẢI MÃ HÌNH HỌC ĐÁM MÂY ĐIỂM VÀ PHƯƠNG TIỆN BẤT BIẾN ĐỌC ĐƯỢC BỞI MÁY TÍNH

(21) 1-2021-06547

(57) Sáng chế đề cập đến phương pháo và thiết bị giải mã hình học đám mây điểm, và phương tiện đọc được bởi máy tính. Phương pháp giải mã hình học đám mây điểm trong bộ giải mã đám mây điểm có thể bao gồm thu dòng bit bao gồm lát của khung đám mây điểm được mã hóa, và khôi phục cây bát phân mà biểu diễn hình học của các điểm trong hộp giới hạn của lát trong đó nút hiện tại của cây bát phân được phân chia với phân chia cây tứ phân (QT-quadtree) hoặc phân chia cây nhị phân (BT-binary tree).

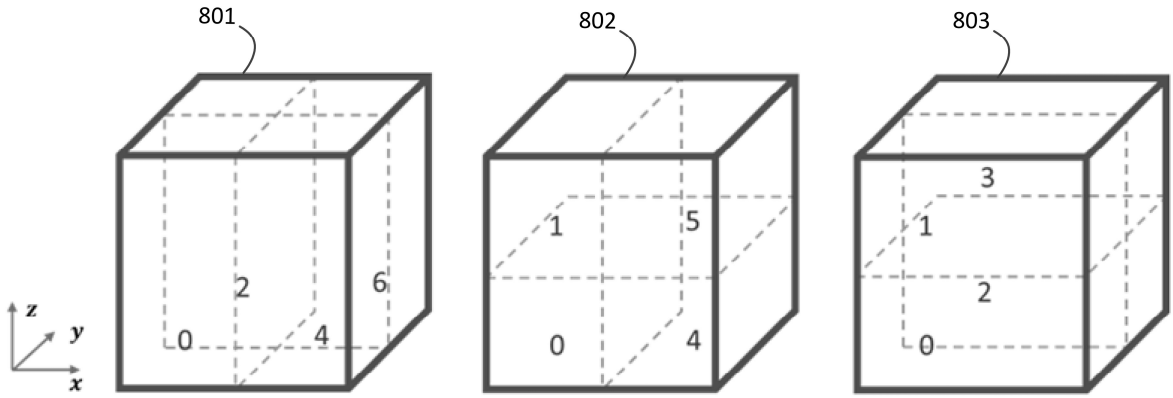


FIG. 8

Lĩnh vực kỹ thuật được đề cập

Sáng chế đề cập đến phương pháp và thiết bị giải mã hình học đám mây điểm, và phương tiện đọc được bởi máy tính.

Tình trạng kỹ thuật của sáng chế

Phần mô tả tình trạng kỹ thuật được đề xuất ở đây nhằm mục đích trình bày chung ngữ cảnh của sáng chế. Công việc của các tác giả sáng chế, trong phạm vi công việc được mô tả trong phần tình trạng kỹ thuật này, cũng như các khía cạnh của phần mô tả mà có thể không phải là kỹ thuật đã biết tại thời điểm nộp đơn, không được thừa nhận một cách rõ ràng hoặc ngụ ý như kỹ thuật đã biết so với sáng chế.

Đám mây điểm đã được sử dụng rộng rãi trong những năm gần đây. Ví dụ, đám mây điểm có thể được sử dụng trong các phương tiện giao thông lái tự động để dò tìm đối tượng và xác định vị trí. Đám mây điểm cũng được sử dụng trong các hệ thống thông tin địa lý (GIS-geographic information system) để tạo bản đồ, và trong di sản văn hóa để hiển thị hóa và lưu trữ các đối tượng và bộ sưu tập di sản văn hóa, v.v.

Khung đám mây điểm chứa tập hợp của các điểm đa chiều, của ba chiều (3D), mỗi chúng bao gồm thông tin vị trí 3D và các thuộc tính bổ sung như màu, độ phản xạ, v.v. Chúng có thể được thu nhờ sử dụng nhiều camera và các bộ cảm biến độ sâu, hoặc ra-đa laze trong các thiết lập khác nhau, và có thể được tạo thành từ hàng nghìn đến hàng tỷ điểm để biểu diễn thực các cảnh gốc.

Các kỹ thuật nén là cần thiết để làm giảm lượng dữ liệu được yêu cầu để biểu diễn đám mây điểm để truyền nhanh hơn hoặc làm giảm sự lưu trữ. ISO/IEC MPEG (JTC 1/SC 29/WG 11) đã tạo ra nhóm đặc biệt (MPEG-PCC) để chuẩn hóa các kỹ thuật nén đối với các đám mây tĩnh hoặc động.

Bản chất kỹ thuật của sáng chế

Các khía cạnh của sáng chế đề xuất phương pháp giải mã hình học đám mây điểm trong bộ giải mã đám mây điểm. Phương pháp này có thể bao gồm thu dòng bit bao gồm lát của khung đám mây điểm được mã hóa, và khôi phục cây bát phân mà biểu diễn hình học của các điểm trong hộp giới hạn của lát nhờ sử dụng phân chia hình học ẩn trong đó nút hiện tại của cây bát phân được phân chia với phân chia cây tứ phân (QT-quadtree) hoặc phân chia cây nhị phân (BT-binary tree).

Trong phương án của sáng chế, làm thế nào để phân chia nút hiện tại của cây bát phân nhờ sử dụng một trong số phân chia QT, phân chia BT, hoặc phân chia cây bát phân (OT-octree) được xác định dựa trên điều kiện định trước. Trong phương án của sáng chế, làm thế nào để phân chia nút hiện tại của cây bát phân nhờ sử dụng một trong số phân chia QT, phân chia BT, hoặc phân chia OT được xác định dựa trên một hoặc nhiều tham số. Một trong số một hoặc nhiều tham số có thể được báo hiệu trong dòng bit hoặc nhờ sử dụng giá trị được cấu hình trước cục bộ.

Trong phương án của sáng chế, thông tin chiếm giữ thuộc về mã chiếm giữ 8 ký tự nhị phân của nút hiện tại của cây bát phân được thu từ dòng bit. Mỗi bit chiếm giữ tương ứng với nút con được chiếm giữ của nút hiện tại của cây bát phân. 4 ký tự nhị phân thuộc về mã chiếm giữ 8 ký tự nhị phân không được báo hiệu trong dòng bit khi nút hiện tại của cây bát phân được phân chia với phân chia QT, và 6 ký tự nhị phân thuộc về mã chiếm giữ 8 ký tự nhị phân không được báo hiệu trong dòng bit khi nút hiện tại của cây bát phân được phân chia với phân chia BT. Theo sáng chế, cũng có thể gọi mã chiếm giữ 8 ký tự nhị phân như là mã chiếm giữ 8-bit hoặc mỗi ký tự nhị phân của mã chiếm giữ 8 ký tự nhị phân là bit.

Trong phương án của sáng chế, một hoặc nhiều phần tử cú pháp mà chỉ báo các kích cỡ ba-chiều (3D) của hộp giới hạn của lát của khung đám mây điểm được mã hóa được thu từ dòng bit. Theo phương án của sáng chế, giá trị của biến, được ký hiệu bởi partitionSkip, mà chỉ rõ loại phân chia và chiều phân chia của nút hiện tại của cây bát phân được xác định. Trong ví dụ của sáng chế,

biến partitionSkip được biểu diễn trong dạng nhị phân với ba bit tương ứng với các chiều x, y, và z, một cách lần lượt, và mỗi bit chỉ báo rằng việc có được thực hiện dọc theo chiều x, y, hoặc z tương ứng hay không. Theo ví dụ của sáng chế, độ sâu trong chiều x, y, hoặc z đối với nút con của nút hiện tại của cây bát phân được cập nhật dựa trên biến partitionSkip.

Theo phương án của sáng chế, bước khôi phục cây bát phân còn bao gồm thu phần tử cú pháp mà chỉ báo nút hiện tại của cây bát phân có một nút con được chiếm giữ, thu 1 ký tự nhị phân nếu biến partitionSkip chỉ báo phân chia BT, hoặc 2 ký tự nhị phân nếu biến partitionSkip chỉ báo phân chia QT, và xác định sơ đồ chiếm giữ mà nhận dạng các nút con được chiếm giữ của nút hiện tại của cây bát phân dựa trên 1 hoặc 2 ký tự nhị phân thu được.

Theo phương án của sáng chế, trong xử lý phân tích trên dòng bit để xác định phần tử cú pháp của sơ đồ chiếm giữ mà nhận dạng các nút con được chiếm giữ của nút hiện tại của cây bát phân, ký tự nhị phân hoặc nhiều ký tự nhị phân của phần tử cú pháp của sơ đồ chiếm giữ có thể được bỏ qua dựa trên biến partitionSkip.

Theo phương án của sáng chế, đối với nút con của nút hiện tại của cây bát phân được mã hóa trong chế độ trực tiếp, kích cỡ log2 đối với mỗi chiều x, y, và z, được ký hiệu dx, dy, và dz, một cách lần lượt, đối với nút con được xác định dựa trên biến partitionSkip. Các vị trí của các điểm trong nút con được mã hóa bởi mã hóa độ dài cố định với (dx, dy, dz) bit, một cách lần lượt.

Theo phương án của sáng chế, phần tử cú pháp được thu từ dòng bit mà chỉ báo một trong số các tham số sau đây: số lượng phân chia QT và BT ẩn lớn nhất được thực hiện trước các phân chia OT, kích cỡ nhỏ nhất của các phân chia QT và BT ẩn mà ngăn chặn các phân chia QT và BT ẩn của nút khi tất cả các chiều của nút nhỏ hơn hoặc bằng kích cỡ nhỏ nhất, hoặc mức ưu tiên mà chỉ báo phân chia QT hoặc BT ẩn nào được thực hiện đầu tiên khi cả hai phân chia QT và BT được cho phép.

Theo phương án của sáng chế, khi độ sâu cây bát phân của nút hiện tại nhỏ hơn tham số K, hoặc khi kích cỡ log2 nhỏ nhất trong số các kích cỡ log2 trong các chiều x, y, và z của nút hiện tại bằng tham số M, loại phân chia và chiều phân chia để phân chia nút hiện tại có thể được xác định theo các điều kiện trong Bảng sau đây:

Loại và chiều phân chia	QT dọc theo các trục x-y	QT dọc theo các trục x-z	QT dọc theo các trục y-z
Điều kiện	$d_z < d_x = d_y$	$d_y < d_x = d_z$	$d_x < d_y = d_z$
Loại và chiều phân chia	BT dọc theo trục x	BT dọc theo trục y	BT dọc theo trục z
Điều kiện	$d_y < d_x$ và $d_z < d_x$	$d_x < d_y$ và $d_z < d_y$	$d_x < d_z$ và $d_y < d_z$

trong đó, tham số K là số nguyên trong phạm vi của $0 \leq K \leq \max(d_x, d_y, d_z) - \min(d_x, d_y, d_z)$, và xác định số lần lớn nhất của các phân chia QT và BT ẩn mà được cho phép trước các phân chia OT, tham số M là số nguyên trong phạm vi của $0 \leq M \leq \min(d_x, d_y, d_z)$, xác định kích cỡ nhỏ nhất của các phân chia QT và BT ẩn, và ngăn chặn các phân chia QT và BT ẩn của nút khi tất cả các chiều của nút nhỏ hơn hoặc bằng M, và dx, dy, và dz là các kích cỡ log2 của nút hiện tại trong các chiều x, y, và z, một cách lần lượt. dx, dy, và dz tương ứng với d_x, d_y, và d_z trong Bảng nêu trên, một cách lần lượt. Hai ký hiệu được sử dụng hoán đổi trong sáng chế này.

Theo phương án của sáng chế, khi độ sâu cây bát phân của nút hiện tại nhỏ hơn tham số K, hoặc khi kích cỡ log2 nhỏ nhất trong số các kích cỡ log2 trong các chiều x, y, và z của nút hiện tại bằng tham số M, biến partitionSkip có thể được xác định như sau,

if (dx < MaxNodeDimLog2),

partitionSkip |= 4;

if (dy < MaxNodeDimLog2),

partitionSkip |= 2;

if (dz < MaxNodeDimLog2),

partitionSkip |= 1.

Biến partitionSkip được biểu diễn trong dạng nhị phân với ba bit, và chỉ rõ loại phân chia và chiều phân chia của nút hiện tại của cây bát phân. Tham số K là số nguyên trong phạm vi của $0 \leq K \leq \max(d_x, d_y, d_z) - \min(d_x, d_y, d_z)$, và xác định số lần phân chia QT và BT ẩn lớn nhất mà được cho phép trước các phân chia OT. Tham số M là số nguyên trong phạm vi của $0 \leq M \leq \min(d_x, d_y, d_z)$, xác định kích cỡ nhỏ nhất của các phân chia QT và BT ẩn, và ngăn chặn các phân chia QT và BT ẩn của nút khi tất cả các chiều của nút nhỏ hơn hoặc bằng M. dx, dy, và dz là các kích cỡ log2 của nút hiện tại trong các chiều x, y, và z, một cách lần lượt. MaxNodeDimLog2 biểu diễn kích cỡ log2 lớn nhất trong số dx, dy, và dz. Toán tử |= biểu diễn toán tử hoặc (OR) đảo bit phức hợp.

Theo phương án của sáng chế, cờ có thể thu được từ dòng bit mà chỉ báo rằng việc phân chia hình học ẩn được cho phép đối với chuỗi của các khung đám mây điểm hoặc lát của khung đám mây điểm được mã hóa. Theo phương án của sáng chế, chế độ phẳng được xác định là không khả dụng tại chiều x, y, hoặc z trong đó việc phân chia không được thực hiện đối với nút hiện tại. Theo phương án của sáng chế, xử lý phân tích chiếm giữ cây bát phân hình học được thực hiện trong đó đối ký tự nhị phân mà có chỉ số của binIdx trong mã chiếm giữ, biến binIsInferred được thiết lập theo như sau: (a) nếu một trong số các điều kiện sau đây là đúng, binIsInferred được thiết lập bằng 1: (1) biến NeighbourPattern bằng 0 và số lượng 1-valued ký tự nhị phân được giải mã trước đó nhỏ hơn hoặc bằng (binIdx + minOccupied – maxOccupied), hoặc (2) biến NeighbourPattern không bằng 0, binIdx bằng maxOccupied-1 và các giá trị của tất cả các ký tự nhị phân được giải mã trước đó bằng 0, trong đó minOccupied=2, và maxOccupied=8 nếu

phân chia OT được áp dụng, maxOccupied=4 nếu phân chia QT được áp dụng, maxOccupied=2 nếu phân chia BT được áp dụng, và (b) nếu không phải, nếu một trong số các điều kiện nêu trên không phải đúng, binIsInferred được thiết lập bằng 0.

Theo phương án của sáng chế, tham số K và tham số M được sử dụng. Tham số K chỉ báo số lượng phân chia QT và BT ẩn lớn nhất trước các phân chia OT, và tham số M chỉ báo kích cỡ nhỏ nhất của các phân chia QT và BT ẩn mà ngăn chặn các phân chia QT và BT ẩn của nút khi tất cả các chiều của nút nhỏ hơn hoặc bằng kích cỡ nhỏ nhất. Các tham số K và M có thể được cập nhật theo: (a) nếu K lớn hơn độ chênh lệch giữa chiều log2 nút gốc lớn nhất và chiều log2 nút gốc nhỏ nhất của lát, K được thay đổi thành độ chênh lệch giữa chiều log2 nút gốc lớn nhất và chiều log2 nút gốc nhỏ nhất của lát; (b) nếu M lớn hơn chiều log2 nút gốc nhỏ nhất của lát, M được thay đổi thành chiều log2 nút gốc nhỏ nhất của lát; (c) nếu chiều log2 nút gốc lớn nhất và chiều log2 nút gốc nhỏ nhất của lát bằng nhau, M được thay đổi thành 0; và (d) nếu chế độ hỗn hợp tam giác phi kết nối (trisoup) được cho phép, K được thay đổi thành độ chênh lệch giữa chiều log2 nút gốc lớn nhất và chiều log2 nút gốc nhỏ nhất của lát, và M được thay đổi thành chiều log2 nút gốc nhỏ nhất của lát. Lưu ý rằng chiều log2 nút hỗn hợp tam giác phi kết nối cần không lớn hơn chiều log2 nút gốc nhỏ nhất của lát.

Các khía cạnh của sáng chế đề xuất thiết bị giải mã hình học đám mây điểm. Thiết bị này có thể bao gồm hệ mạch có cấu trúc để thực hiện phương pháp theo phương pháp giải mã hình học đám mây điểm nêu.

Các khía cạnh của sáng chế đề xuất phương tiện bất biến đọc được bởi máy tính mà lưu trữ các lệnh mà, khi được thực thi bởi bộ xử lý, làm cho bộ xử lý thực hiện phương pháp giải mã hình học đám mây điểm.

Mô tả vắn tắt các hình vẽ

Các phương án của sáng chế mà được đề xuất như là các ví dụ sẽ được mô tả chi tiết có viện dẫn tới các hình vẽ kèm theo, trong đó các số chỉ dẫn giống nhau ký hiệu các bộ phận giống nhau, và trong đó:

FIG.1 thể hiện là xử lý phân chia đệ quy theo phương án của sáng chế.

FIG.2 thể hiện xử lý tạo mức chi tiết (LOD-level of detail) theo phương án của sáng chế.

FIG.3 thể hiện bộ mã hóa ví dụ theo phương án của sáng chế.

FIG.4 thể hiện bộ giải mã ví dụ theo phương án của sáng chế.

FIG.5 thể hiện phân chia cây bát phân trong không gian ba chiều (3D).

FIG.6 thể hiện ví dụ về phân chia cây bát phân (OT) hai mức và các mã chiếm giữ tương ứng.

FIG.7 thể hiện chuỗi đám mây điểm mà có các thành phần cơ bản trong các chiều x và y.

FIG.8 thể hiện các ví dụ về các phân chia cây tứ phân (QT) trong không gian 3D theo phương án của sáng chế.

FIG.9 thể hiện các ví dụ về các phân chia cây nhị phân (BT) trong không gian 3D theo phương án của sáng chế.

Các Fig.10-13 thể hiện các khối hai chiều (2D) để minh họa bốn xử lý phân chia khác nhau dựa trên các phân chia QT và BT ẩn.

FIG.14 thể hiện các khối 2D để minh họa hệ thống mô hình kiểm tra 13 (TMC13) trong đó các phân chia OT được thực hiện từ hộp giới hạn khối lập phương.

FIG.15 thể hiện lưu đồ của xử lý giải mã hình học dựa trên phân chia QT/BT để giải mã đám mây điểm theo phương án của sáng chế.

FIG.16 thể hiện hệ thống máy tính để thực hiện các kỹ thuật giải mã hình học trong các phương án khác nhau.

Mô tả chi tiết sáng chế

Sáng chế liên quan đến việc nén đám mây điểm dựa trên hình học (G-PCC). Các kỹ thuật phân chia cây tứ phân (QT) hoặc cây nhị phân (BT) ẩn được mô tả. Khi phân chia QT hoặc BT ẩn được áp dụng, dạng hình học của đám mây điểm có thể được phân chia ẩn theo tập con của tất cả các chiều khi các tiêu chuẩn định trước được thỏa mãn, thay vì phân chia theo tất cả các chiều.

I. Bộ mã hóa và bộ giải mã đám mây điểm

1. Dữ liệu đám mây điểm

Dữ liệu đám mây điểm (hoặc đám mây điểm) được sử dụng để biểu diễn cảnh ba chiều (3D) hoặc đối tượng trong một vài ứng dụng hiện nay như thực tế ảo (VR) nhập vai/thực tế tăng cường (AR)/thực tế kết hợp (MR), điều hướng tự động/rô-bôt, tạo ảnh y tế, và loại tương tự. Đám mây điểm có thể bao gồm tập hợp của các điểm 3D riêng biệt. Mỗi điểm được kết hợp với tập hợp của các tọa độ 3D mà chỉ báo vị trí 3D của điểm tương ứng và số lượng thuộc tính khác như màu, pháp tuyến bề mặt, độ mờ, độ phản xạ, v.v trong các phương án khác nhau, dữ liệu đám mây điểm đầu vào có thể được lượng tử hóa và sau đó được sắp xếp thành lưới 3D của các điểm ảnh ba chiều khối lập phương mà có thể được mô tả nhờ sử dụng cấu trúc dữ liệu cây bát phân. Cây bát phân được tạo điểm ảnh ba chiều thu được hỗ trợ duyệt cây, tìm kiếm, và truy nhập của dữ liệu đám mây điểm được lượng tử hóa.

Khung đám mây điểm có thể bao gồm tập hợp của các điểm 3D tại thời điểm cụ thể. Các khung đám mây điểm có thể được sử dụng để khôi phục đối tượng hoặc cảnh nhìn như là thành phần của các điểm này. Chúng có thể được chụp nhờ sử dụng các camera và các bộ cảm biến độ sâu trong các thiết lập khác nhau, và có thể được tạo thành từ hàng nghìn và thậm chí hàng tỷ điểm để biểu diễn thực tế các cảnh được khôi phục. Chuỗi của các khung đám mây điểm có thể được gọi là đám mây điểm. Tập hợp của các tọa độ Đề-các được kết hợp với

các điểm 3D của khung đám mây điểm có thể được gọi là dạng hình học của các điểm 3D này.

Các kỹ thuật nén là cần thiết để làm giảm lượng dữ liệu được yêu cầu để biểu diễn đám mây điểm (ví dụ, chuỗi của các khung đám mây điểm). Theo đó, các kỹ thuật là cần thiết để nén có tổn hao các đám mây điểm để sử dụng trong truyền thông thời gian thực và thực tế ảo sáu mức độ tự do (6 DoF). Ngoài ra, các kỹ thuật được yêu cầu cho nén đám mây điểm không tổn hao trong ngữ cảnh của ánh xạ động dùng cho lái xe tự động và các ứng dụng di sản văn hóa, v.v. Ngoài ra, các tiêu chuẩn là cần thiết để giải quyết việc nén của dạng hình học và các thuộc tính (ví dụ, các màu và độ phản xạ), mã hóa có khả năng biến đổi tỷ lệ/lũy tiến, mã hóa của các chuỗi đám mây điểm được thu theo thời gian, và truy nhập ngẫu nhiên tới các tập con của đám mây điểm.

2. Lượng tử hóa các tọa độ

Theo phương án của sáng chế, các tọa độ của các điểm trong dữ liệu đám mấy đầu vào có thể được lượng tử hóa đầu tiên. Ví dụ, các giá trị số thực của các tọa độ có thể được lượng tử hóa thành các giá trị nguyên. Sau khi lượng tử hóa, nhiều hơn một điểm có thể chia sẻ vị trí giống nhau trong một vài điểm ảnh ba chiều. Các điểm sao chép này, một cách tùy chọn, có thể được hợp nhất thành một điểm.

3. Mã hóa hình học dựa trên cây bát phân

FIG.1 thể hiện xử lý phân chia đệ quy 100 theo phương án của sáng chế. Xử lý 100 có thể được thực hiện để tạo ra cấu trúc cây bát phân để biểu diễn các vị trí của tập hợp các điểm trong đám mây điểm. Như được thể hiện, hộp giới hạn được căn thẳng theo trục khối lập phương 101 mà chứa tập hợp của các điểm được xác định đầu tiên. Sau đó, hộp giới hạn 101 được phân chia đệ quy để xây dựng cấu trúc cây bát phân. Như được thể hiện, tại mỗi tầng, khối lập phương hiện tại có thể được phân chia thành 8 khối lập phương con. Mã 8-bit, được gọi là mã chiếm giữ, có thể được tạo ra để chỉ báo rằng mỗi 8 khối lập phương con có chứa các điểm hay không. Ví dụ, mỗi khối lập phương con được kết hợp với giá

trị 1-bit. Nếu khối lập phương con được chiếm giữ, khối lập phương con tương ứng có giá trị bit bằng 1; nếu không phải, khối lập phương con tương ứng có giá trị bit bằng 0. Các khối lập phương con được chiếm giữ có thể được phân chia cho đến khi đạt tới kích cỡ của các khối lập phương con nhỏ nhất định trước. Khối lập phương con có kích cỡ nhỏ nhất là điểm ảnh ba chiều tương ứng với cấu trúc cây bát phân. Nhờ đó, chuỗi của các mã chiếm giữ có thể được tạo ra, và sau đó được nén và được truyền từ bộ mã hóa tới bộ giải mã. Bằng cách giải mã các mã chiếm giữ (ví dụ, thực hiện xử lý giải mã cây bát phân), bộ giải mã có thể thu nhận cấu trúc cây bát phân tương tự như bộ mã hóa, hoặc ước lượng của cấu trúc cây bát phân.

4. Truyền thuộc tính

Theo kết quả của xử lý tạo hoặc mã hóa cây bát phân, tại phía bộ mã hóa, khối lập phương con có kích cỡ nhỏ nhất có thể chứa nhiều hơn một điểm. Do đó, vị trí tương ứng với điểm ảnh ba chiều (ví dụ, trung tâm của khối lập phương con tương ứng) có thể tương ứng với nhiều tập hợp thuộc tính từ nhiều điểm. Trong trường hợp này, theo phương án của sáng chế, xử lý truyền thuộc tính có thể được thực hiện để xác định một tập hợp của các thuộc tính dựa trên nhiều tập hợp thuộc tính đối với điểm ảnh ba chiều tương ứng. Ví dụ, thuộc tính trung bình của tập con của các điểm lân cận gần nhất có thể được sử dụng như là thuộc tính của điểm ảnh ba chiều tương ứng. Các phương pháp khác nhau có thể được sử dụng trong các phương án khác nhau nhằm mục đích truyền thuộc tính.

5. Tạo mức chi tiết (LOD-Level of Detail)

FIG.2 thể hiện a xử lý tạo mức chi tiết (LOD-level of detail) 200 theo phương án của sáng chế. Xử lý 200 có thể được thực hiện trên các vị trí được lượng tử hóa (ví dụ, các vị trí điểm ảnh ba chiều) có thứ tự theo xử lý giải mã cây bát phân. Đối với kết quả của xử lý 200, các điểm có thể được tái tổ chức hoặc tái sắp xếp thành tập hợp của các mức tinh chỉnh. Xử lý 200 có thể được thực hiện đồng nhất tại bộ mã hóa và bộ giải mã. Xử lý mã hóa thuộc tính tiếp theo có

thể được thực hiện tại bộ mã hóa hoặc bộ giải mã theo thứ tự được xác định bởi xử lý 200 (được gọi là thứ tự dựa trên LOD).

Cụ thể, FIG.2 thể hiện ba LOD: LOD0, LOD1, và LOD2. Khoảng cách Oclit, d0, d1, hoặc d2, có thể được chỉ rõ đối với LOD0, LOD1, và LOD2, một cách lần lượt. Tập con của các điểm P0-P9 được chứa trong mỗi LOD. Các khoảng cách giữa mỗi cặp điểm trong LOD tương ứng lớn hơn hoặc bằng khoảng cách Oclit tương ứng. Các khoảng cách Oclit có thể được sắp xếp theo cách mà d0>d1>d2. Theo cách sắp xếp này, mức tinh chỉnh cao hơn bao gồm ít điểm hơn mà cách xa nhau hơn, và cung cấp biểu diễn thô hơn của đám mây điểm, trong khi mức tinh chỉnh thấp hơn bao gồm nhiều điểm gần nhau hơn, và cung cấp biểu diễn tinh hơn của đám mây điểm.

Theo kết quả của xử lý tạo LOD 200 nêu trên, các điểm trong thứ tự gốc (thứ tự giải mã cây bát phân) từ P0 đến P9 có thể được tổ chức lại thành thứ tự dựa trên LOD: P0, P5, P4, P2, P1, P6, P3, P9, P8, và P7.

6. Dự đoán các thuộc tính

Các thuộc tính được kết hợp với đám mây điểm có thể được mã hóa và được giải mã trong thứ tự được xác định bởi xử lý tạo LOD. Ví dụ, theo lần lượt từng điểm, trong thứ tự dựa trên LOD, dự đoán thuộc tính của mỗi điểm hiện tại (điểm đang được xử lý) có thể được xác định bằng cách thực hiện xử lý dự đoán thuộc tính tại bộ mã hóa và/hoặc bộ giải mã. Xử lý dự đoán thuộc tính tương tự có thể được thực hiện tại bộ mã hóa và bộ giải mã.

Với dự đoán thuộc tính thu được, tại bộ mã hóa, tín hiệu dư có thể được tạo ra bằng cách trừ giá trị dự đoán thuộc tính từ giá trị thuộc tính gốc tương ứng của điểm hiện tại. Sau đó, tín hiệu dư có thể, một cách riêng biệt hoặc kết hợp với các tín hiệu dư khác, được nén tiếp. Ví dụ, các hoạt động biến đổi và/hoặc lượng tử hóa có thể được thực hiện, và được theo sau bởi mã hóa entropy của các tín hiệu thu được. Tín hiệu dư được nén có thể được truyền tới bộ mã hóa trong dòng bit.

Tại bộ giải mã, tín hiệu dư có thể được khôi phục bằng cách thực hiện nghịch đảo của xử lý mã hóa tại bộ mã hóa để mã hóa tín hiệu dư. Với dự đoán thuộc tính thu được và tín hiệu dư được khôi phục tương ứng với điểm hiện tại, thuộc tính được khôi phục của điểm hiện tại có thể thu được. Theo cách tương tự, hoạt động khôi phục này có thể diễn ra tại bộ mã hóa để thu nhận thuộc tính được khôi phục.

Các kỹ thuật dự đoán thuộc tính khác nhau có thể được sử dụng trong các phương án khác nhau để xác định dự đoán thuộc tính. Thông thường, dự đoán thuộc tính của điểm hiện tại được thực hiện nhờ sử dụng các thuộc tính được khôi phục trước đó của các điểm lân cận điểm hiện tại. Khi xử lý dự đoán thuộc tính được bắt đầu, dựa trên thứ tự dựa trên LOD, các giá trị thuộc tính được khôi phục của các điểm trước điểm hiện tại đã là khả dụng. Ngoài ra, từ xử lý mã hóa hoặc giải mã cây bát phân, các vị trí (các tọa độ 3D) của các điểm trong đám mây điểm cũng là khả dụng. Do đó, xử lý dự đoán thuộc tính có thể được thực hiện với sự nhận biết của các thuộc tính được khôi phục và các tọa độ 3D của các điểm lân cận của điểm hiện tại.

Ví dụ, tập hợp của các điểm lân cận của điểm hiện tại có thể đầu tiên được xác định nhờ sử dụng các thuật toán khác nhau. Trong một ví dụ, xử lý tìm kiếm dựa trên cấu trúc cây k-d có thể được thực hiện để xác định tập hợp của các điểm gần nhất với điểm hiện tại.

Ví dụ, phương pháp dựa trên khoảng cách hình học và/hoặc khoảng cách thuộc tính được sử dụng để xác định dự đoán thuộc tính. Thuộc tính dự đoán có thể được xác định dựa trên tổng có trọng số (hoặc trung bình có trọng số) của các thuộc tính được khôi phục của tập hợp các điểm lân cận được xác định tại bộ mã hóa hoặc bộ giải mã. Ví dụ, với tập hợp của các điểm lân cận được xác định, tổng có trọng số (hoặc trung bình có trọng số) của các thuộc tính được khôi phục của các điểm lân cận được xác định có thể được xác định là thuộc tính dự đoán tại bộ mã hóa hoặc bộ giải mã. Ví dụ, trọng số được sử dụng trong kỹ thuật dựa trên tổng có trọng số (cũng được gọi là kỹ thuật dựa trên nội suy) có thể là nghịch đảo

của (hoặc tỷ lệ nghịch) khoảng cách hình học, hoặc khoảng cách thuộc tính. Ngoài ra, trọng số có thể là trọng số hai chiều thu được từ kết hợp của trọng số dựa trên khoảng cách hình học (trọng số hình học) và trọng số dựa trên khoảng cách thuộc tính (trọng số thuộc tính).

Ví dụ, phương pháp dựa trên méo tỷ lệ (RD-rate-distortion) được sử dụng để xác định dự đoán thuộc tính. Ví dụ, chỉ số ứng viên có thể được kết hợp với mỗi giá trị thuộc tính được khôi phục của tập hợp các điểm lân cận tại bộ mã hóa hoặc bộ giải mã. Tại bộ mã hóa, xử lý dựa trên tối ưu hóa RD có thể được thực hiện để đánh giá giá trị nào trong số các giá trị thuộc tính được khôi phục ứng viên của các điểm lân cận là lựa chọn tốt nhất cần được sử dụng như là dự đoán thuộc tính. Ví dụ, độ méo có thể được đo lường bởi độ chênh lệch giữa giá trị thuộc tính gốc (hoặc đúng) của điểm hiện tại và dự đoán ứng viên (giá trị thuộc tính được khôi phục ứng viên). Tỷ lệ có thể là giá trị mã hóa chỉ số của dự đoán ứng viên được lựa chọn. Hàm giá trị RD La-răng có thể được định nghĩa để xác định ứng viên tín hiệu dự đoán tốt nhất. Chỉ số ứng viên của dự đoán ứng viên được lựa chọn có thể nhờ đó được báo hiệu tới bộ giải mã.

Do đó, bộ giải mã có thể đầu tiên xác định tập hợp giống nhau của các điểm lân cận của diểm hiện tại tương ứng, và kết hợp chỉ số tới các giá trị thuộc tính được khôi phục của cùng tập hợp của điểm lân cận theo cách thức tương tự như phía bộ mã hóa. Sau đó, bộ giải mã có thể xác định dự đoán thuộc tính từ các giá trị thuộc tính được khôi phục của các điểm lân cận nhờ sử dụng chỉ số ứng viên được báo hiệu.

7. Các ví dụ hệ thống mã hóa để nén đám mây điểm

FIG.3 thể hiện bộ mã hóa 300 ví dụ theo phương án của sáng chế. Bộ mã hóa có thể có cấu trúc để thu dữ liệu đám mây điểm và nén dữ liệu đám mây điểm để tạo ra dòng bit mà mang dữ liệu đám mây điểm được nén. Theo phương án của sáng chế, bộ mã hóa 300 có thể bao gồm a môđun lượng tử hóa vị trí 310, môđun loại bỏ các điểm trùng lặp 312, môđun mã hóa cây bát phân 330, môđun truyền thuộc tính 320, môđun tạo LOD 340, môđun dự đoán thuộc tính

350, môđun lượng tử hóa phần dư 360, môđun mã hóa số học 370, môđun lượng tử hóa phần dư ngược 380, môđun cộng 381, và bộ nhớ 390 để lưu trữ các giá trị thuộc tính được khôi phục.

Như được thể hiện, đám mây điểm đầu vào 301 có thể được thu nhận tại bộ mã hóa 300. Các vị trí (các tọa độ 3D) của đám mây điểm 301 được cấp tới môđun lượng tử hóa 310. Môđun lượng tử hóa 310 có cấu trúc để lượng tử hóa các tọa độ để tạo các vị trí được lượng tử hóa. Môđun loại bỏ các điểm trùng lặp 312 tùy chọn có cấu trúc để thu các vị trí được lượng tử hóa và thực hiện xử lý lọc để nhận dạng và loại bỏ các điểm trùng lặp. Môđun mã hóa cây bát phân 330 có cấu trúc để thu các vị trí được lọc từ môđun loại bỏ các điểm trùng lặp, và thực hiện xử lý mã hóa dựa trên cây bát phân để tạo ra chuỗi của các mã chiếm giữ mà mô tả lưới 3D của các điểm ảnh ba chiều. Các mã chiếm giữ được cấp tới môđun mã hóa số học 370.

Môđun truyền thuộc tính 320 có cấu trúc để thu các thuộc tính của đám mây điểm đầu vào, và thực hiện xử lý truyền thuộc tính để xác định giá trị thuộc tính đối với mỗi điểm ảnh ba chiều khi các giá trị thuộc tính được kết hợp với điểm ảnh ba chiều tương ứng. Xử lý truyền thuộc tính có thể được thực hiện trên các điểm được sắp xếp lại được xuất ra từ môđun mã hóa cây bát phân 330. Các thuộc tính sau các hoạt động truyền được cấp tới môđun dự đoán thuộc tính 350. Môđun tạo LOD 340 có cấu trúc để hoạt động trên các điểm được sắp xếp lại được xuất ra từ môđun mã hóa cây bát phân 330, và tổ chức lại các điểm thành các LOD khác nhau. Thông tin LOD được cấp tới môđun dự đoán thuộc tính 350.

Môđun dự đoán thuộc tính 350 xử lý các điểm theo thứ tự dựa trên LOD được chỉ báo bởi thông tin LOD từ môđun tạo LOD 340. Môđun dự đoán thuộc tính 350 tạo ra dự đoán thuộc tính đối với điểm hiện tại dựa trên các thuộc tính được khôi phục của tập hợp các điểm lân cận của điểm hiện tại được lưu trữ trong bộ nhớ 390. Sau đó, các phần dư dự đoán có thể được thu nhận dựa trên các giá trị thuộc tính gốc thu được từ môđun truyền thuộc tính 320 và các dự đoán thuộc tính được tạo ra cục bộ. Khi các chỉ số ứng viên được sử dụng trong xử lý

dự đoán thuộc tính tương ứng, chỉ số tương ứng với ứng viên dự đoán được lựa chọn có thể được cấp tới môđun mã hóa số học 370.

Môđun lượng tử hóa phần dư 360 có cấu trúc để thu các phần dư dự đoán từ môđun dự đoán thuộc tính 350, và thực hiện việc lượng tử hóa để tạo ra các phần dư được lượng tử hóa. Các phần dư được lượng tử hóa được cấp tới môđun mã hóa số học 370.

Môđun lượng tử hóa phần dư ngược 380 có cấu trúc để thu các phần dư được lượng tử hóa từ môđun lượng tử hóa phần dư 360, và tạo ra các phần dư dự đoán được khôi phục bằng cách thực hiện nghịch đảo của các hoạt động lượng tử hóa được thực hiện tại môđun lượng tử hóa phần dư 360. Môđun cộng 381 có cấu trúc để thu các phần dư dự đoán được khôi phục từ môđun lượng tử hóa phần dư ngược 380, và các dự đoán thuộc tính tương ứng từ môđun dự đoán thuộc tính 350. Bằng cách kết hợp các phần dư dự đoán được khôi phục và các dự đoán thuộc tính, các giá trị thuộc tính được khôi phục được tạo ra và được lưu trữ tới bộ nhớ 390.

Môđun mã hóa số học 370 có cấu trúc để thu các mã chiếm giữ, các chỉ số ứng viên (nếu được sử dụng), các phần dư được lượng tử hóa (nếu được tạo ra), và thông tin khác, và thực hiện mã hóa entropy để nén tiếp cac giá trị hoặc thông tin thu được. Kết quả là, dòng bit 302 được nén mà mang thông tin được nén có thể được tạo ra. Dòng bit 302 có thể được truyền, hoặc nếu không được cấp, tới bộ giải mã mà giải mã dòng bit được nén, hoặc có thể được lưu trữ trong thiết bị lưu trữ. Dòng bit có thể liên quan đến chuỗi của các bit mà tạo ra biểu diễn của các khung đám mây điểm được mã hóa. Khung đám mây điểm được mã hóa có thể liên quan đến biểu diễn được mã hóa của khung đám mây điểm.

FIG.4 thể hiện bộ giải mã 400 ví dụ theo phương án của sáng chế. Bộ giải mã 400 có thể có cấu trúc để thu dòng bit được nén và thực hiện giải nén dữ liệu đám mây điểm để giải nén dòng bit để tạo ra dữ liệu đám mây điểm được giải mã. Theo phương án của sáng chế, bộ giải mã 400 có thể bao gồm môđun giải mã số học 410, môđun lượng tử hóa phần dư ngược 420, môđun giải mã cây

bát phân 430, môđun tạo LOD 440, môđun dự đoán thuộc tính 450, và bộ nhớ 460 để lưu trữ các giá trị thuộc tính được khôi phục.

Như được thể hiện, dòng bit được nén 401 có thể được thu nhận tại môđun giải mã số học 410. Môđun giải mã số học 410 có cấu trúc để giải mã dòng bit được nén 401 để thu nhận các phần dư được lượng tử hóa (nếu được tạo ra) và các mã chiếm giữ của đám mây điểm. Môđun giải mã cây bát phân 430 có cấu trúc để xác định các vị trí được khôi phục của các điểm trong đám mây điểm theo các mã chiếm giữ. Môđun tạo LOD 440 có cấu trúc để tổ chức lại các điểm thành các LOD khác nhau dựa trên các vị trí được khôi phục, và xác định thứ tự dựa trên LOD. Môđun lượng tử hóa phần dư ngược 420 có cấu trúc để tạo ra các phần dư được khôi phục dựa trên các phần dư được lượng tử hóa thu được từ môđun giải mã số học 410.

Môđun dự đoán thuộc tính 450 có cấu trúc để thực hiện xử lý dự đoán thuộc tính để xác định các dự đoán thuộc tính đối với các điểm theo thứ tự dựa trên LOD. Ví dụ, dự đoán thuộc tính của điểm hiện tại có thể được xác định dựa trên các giá trị thuộc tính được khôi phục của các điểm lân cận của điểm hiện tại được lưu trữ trong bộ nhớ 460. Môđun dự đoán thuộc tính 450 có thể kết hợp dự đoán thuộc tính với phần dư được khôi phục tương ứng để tạo ra thuộc tính được khôi phục đối với điểm hiện tại.

Chuỗi của các thuộc tính được khôi phục được tạo ra từ môđun dự đoán thuộc tính 450 cùng với các vị trí được khôi phục được tạo ra từ môđun giải mã cây bát phân 430 tương ứng với đám mây điểm được giải mã 402 mà được xuất ra từ bộ giải mã 400 trong một ví dụ. Ngoài ra, các thuộc tính được khôi phục cũng được lưu trữ vào bộ nhớ 460 và sau đó, có thể được sử dụng để thu nhận các dự đoán thuộc tính đối với các điểm tiếp theo.

Trong các phương án khác nhau, bộ mã hóa 300 và bộ giải mã 400 có thể được thực hiện với phần cứng, phần mềm, hoặc kết hợp của chúng. Ví dụ, bộ mã hóa 300 và bộ giải mã 400 có thể được thực hiện với hệ mạch xử lý như một hoặc nhiều mạch tích hợp (IC) mà hoạt động có hoặc không có phần mềm,

như mạch tích hợp ứng dụng riêng (ASIC-application specific integrated circuit), mảng cổng khả trình dạng trường (FPGA-field programmable gate array), và loại tương tự. Theo ví dụ khác, bộ mã hóa 300 và bộ giải mã 400 có thể được thực hiện như là phần mềm hoặc vi chương trình bao gồm các lệnh được lưu trữ trong phương tiện lưu trữ bất biến (hoặc không chuyển tiếp) có thể đọc được bởi máy tính. Các lệnh, khi được thực thi bởi hệ mạch xử lý, như một hoặc nhiều bộ xử lý, làm cho hệ mạch xử lý thực hiện các chức năng của bộ mã hóa 300 và bộ giải mã 400.

Lưu ý rằng các bộ phận trong bộ mã hóa 300 và bộ giải mã 400 có cấu trúc để thực hiện các kỹ thuật khác nhau được bộc lộ ở đây có thể được chứa trong các bộ giải mã hoặc các bộ mã hóa khác mà có thể có các cấu trúc giống hoặc khác với những gì được thể hiện trên FIG.3 và FIG.4. Ngoài ra, bộ mã hóa 300 và bộ giải mã 400 có thể được chứa trong cùng thiết bị, hoặc các thiết bị riêng biệt trong các ví dụ khác nhau.

II. G-PCC trong mô hình kiểm tra 13 (TMC13)

G-PCC giải quyết việc nén của các đám mây điểm trong cả Danh mục 1 (các đám mây điểm tĩnh) và Danh mục 3 (các điểm mây điểm thu được động). Mô hình kiểm tra TMC13 đã được phát triển bởi Nhóm chuyên gia ảnh động (MPEG) như là cơ sở để nghiên cứu các kỹ thuật mã hóa đám mây điểm có khả năng. Mô hình TMC13 nén riêng biệt dạng hình học và các thuộc tính được kết hợp như màu hoặc độ phản xạ. Dạng hình học, mà là các tọa độ 3D của các đám mây điểm, được mã hóa bởi phân chia cây bát phân. Sau đó, các thuộc tính được nén dựa trên dạng hình học được khôi phục nhờ sử dụng các kỹ thuật dự đoán và phép nâng.

1. Phân chia cây bát phân và mã hóa thông tin chiếm giữ trong phương án TMC13 hiện tại

Việc phân chia đều của khối lập phương 3D sẽ tạo ra tám khối lập phương con, mà được biết đến là phân chia cây bát phân (OT) trong nén đám mây điểm (PCC-point cloud compression). Việc phân chia OT tương đồng với a phân

chia cây nhị phân (BT) trong một-chiều và phân chia cây tứ phân (QT) trong không gian hai chiều. Việc phân chia OT được minh họa trong FIG 5, trong đó khối lập phương 3D 501 trong đường liền nét được phân chia thành tám khối lập phương có kích cỡ bằng nhau nhỏ hơn trong đường đứt nét. 4 khối lập phương phía bên trái được kết hợp với các chỉ số 0-3, trong khi 4 khối lập phương phía bên phải được kết hợp với các chỉ số 4-7.

Trong TMC13, nếu việc mã hóa giải mã hình học cây bát phân được sử dụng, việc mã hóa hình học diễn ra như sau. Đầu tiên, hộp giới hạn được căn thẳng theo trục khối lập phương B được xác định bởi hai điểm (0,0,0) và $(2^d, 2^d, 2^d)$, trong đó 2^d xác định kích cỡ của B và d được mã hóa trong dòng bit. Giả định tất cả các điểm cần được nén nằm bên trong hộp giới hạn B được xác định.

Sau đó, cấu trúc cây bát phân được xây dựng bằng cách phân chia đệ quy B. Tại mỗi tầng, khối lập phương được phân chia thành 8 khối lập phương con. Kích cỡ của khối lập phương con sau khi phân chia lặp lại k ($k \leq d$) lần sẽ là $(2^{d-k}, 2^{d-k}, 2^{d-k})$. Mã 8-bit, tức là mã chiếm giữ, sau đó được bằng cách kết hợp giá trị 1-bit với mỗi khối lập phương con để chỉ báo rằng nó có chứa các điểm (tức là, đầy đủ và có giá trị 1) hay không (tức là, trống và có giá trị 0). Chỉ các khối lập phương con đầy đủ với kích cỡ lớn hơn 1 (tức là, không phải các điểm ảnh ba chiều) được phân chia tiếp. Sau đó, mã chiếm giữ đối với mỗi khối lập phương có thể được nén bởi bộ mã hóa số học.

Xử lý giải mã bắt đầu bằng cách đọc từ dòng bit các chiều của hộp giới hạn B. Sau đó, cấu trúc cây bát phân giống nhau được xây dựng bằng cách phân chia B theo các mã chiếm giữ được giải mã. Ví dụ về phân chia OT mức và các mã chiếm giữ tương ứng được thể hiện trên FIG.6, trong đó các khối lập phương và các nút trong màu tối chỉ báo chúng được chiếm giữ bởi các điểm.

Như được thể hiện, khối lập phương 601 được phân chia thành 8 khối lập phương con. Với cùng phương pháp đánh chỉ số được sử dụng trong FIG.5, các khối lập phương con thứ 0 và thứ 7- được phân tiếp thành 8 khối lập phương

con. Cây bát phân 610 tương ứng với các phân chia tới khối lập phương 601 bao gồm nút gốc 611 tại mức thứ nhất. Nút gốc 611 được phân chia thành 8 nút con mà có thể được đánh chỉ số từ 0 đến 7. Các nút thứ 0 và thứ 7 (612-613) tại mức thứ hai còn được phân chia thành 16 nút con. Mức của nút trong cây bát phân 610 có thể tương ứng với số lượng chặng từ gốc tới nút tương ứng, và có thể được gọi là các độ sâu của cây bát phân 610. Các độ sâu 0-2 tương ứng với các mức thứ nhất đến thứ ba của cây bát phân 610 được chỉ báo trong FIG.6.

2. Mã hóa của các mã chiếm giữ

Mã chiếm giữ của mỗi nút có thể được nén bởi bộ mã hóa số học. Mã chiếm giữ có thể được ký hiệu là S mà là số nguyên 8-ký tự nhị phân, và mỗi ký tự nhị phân trong S chỉ báo trạng thái chiếm giữ của mỗi nút con. Hai phương pháp mã hóa đối với mã chiếm giữ tồn tại trong TMC13, tức là, các phương pháp mã hóa đảo bit và mã hóa đảo byte, và việc mã hóa đảo bit được kích hoạt mặc định. Cách thức thực hiện việc mã hóa số học với mô hình ngữ cảnh để mã hóa mã chiếm giữ, trong đó trạng thái ngữ cảnh được khởi tạo tại bắt đầu của toàn bộ xử lý mã hóa và được cập nhật trong xử lý mã hóa.

Đối với mã hóa đảo bit, tám ký tự nhị phân trong S được mã hóa trong thứ tự định trước trong đó mỗi ký tự nhị phân được mã hóa bằng cách viện dẫn tới trạng thái chiếm giữ của các nút lân cận và các nút con của các nút lân cận, trong đó các nút lân cận nằm tại cùng mức của nút hiện tại.

Đối với mã hóa đảo byte, S được mã hóa bằng cách viện dẫn tới Bảng tra cứu thích nghi (A-LUT), mà theo dõi N (ví dụ, 32) mã chiếm giữ thường xuyên nhất, và bộ lưu trữ đệm mà theo dõi M (ví dụ, 16) mã chiếm giữ được quan sát khác cuối cùng.

Cờ nhị phân mà chỉ báo rằng S có nằm trong A-LUT hay không được mã hóa. Nếu S nằm trong A-LUT, chỉ số trong A-LUT được mã hóa bằng cách sử dụng bộ mã hóa số học nhị phân. Nếu S không nằm trong A-LUT, thì cờ nhị phân mà chỉ báo rằng S có nằm trong bộ lưu trữ đệm hay không được mã hóa. Nếu S nằm trong bộ lưu trữ đệm, thì biểu diễn nhị phân của chỉ số của nó trong

bộ lưu trữ đệm được mã hóa bằng cách sử dụng bộ mã hóa số học nhị phân. Nếu không phải, nếu S không nằm trong bộ lưu trữ đệm, thì biểu diễn nhị phân của S được mã hóa bằng cách sử dụng bộ mã hóa số học nhị phân.

Xử lý giải mã bắt đầu bằng cách phân tích các chiều của hộp giới hạn B từ dòng bit. Sau đó, cấu trúc cây bát phân giống nhau được xây dựng bằng cách phân chia B theo các mã chiếm giữ được giải mã.

III. Phân chia hình học ẩn để mã hóa đám mây điểm

1. Vấn đề

Trong thiết kế TMC13, hộp giới hạn được giới hạn là khối lập phương B mà có cùng kích cỡ đối với tất cả các chiều, và phân chia OT được thực hiện đối với các khối lập phương con đầy đủ tại mỗi nút nhờ đó các khối lập phương con được giảm một nửa về kích cỡ của tất cả các chiều. Phân chia OT được thực hiện đệ quy cho đến khi kích cỡ của các khối lập phương con đạt tới một hoặc không có điểm được chứa trong các khối lập phương con. Tuy nhiên, cách thức này không hiệu quả đối với tất cả các trường hợp, đặc biệt khi các điểm không được phân bố đều trong cảnh 3D. Một trường hợp giới hạn là mặt phẳng 2D trong không gian 3D, trong đó tất cả các điểm nằm trên mặt phẳng x-y và thay đổi trong trục z là 0. Trong trường hợp này, phân chia OT được thực hiện trên khối lập phương B như là điểm bắt đầu sẽ lãng phí số lượng lớn các bit để biểu diễn thông tin chiếm giữ trong chiều z, mà là sự dư thừa và vô ích. Trong các ứng dụng thực tế, trường hợp xấu nhất có thể không xảy ra thường xuyên, tuy nhiên, thường có đám mây điểm mà có hộp giới hạn bất đối xứng, mà có độ dài ngắn hơn trong một vài chiều.

Như được thể hiện trên FIG.7, chuỗi đám mây điểm có tên "ford_01_vox1mm" được sử dụng để kiểm tra trong MPEG-PCC có các thành phần cơ bản trong các chiều x và y. Thực tế, nhiều dữ liệu đám mây điểm được tạo ra từ bộ cảm biến Lidar có cùng các đặc tính.

2. Các phân chia QT và BT ẩn

Các khía cạnh của sáng chế đề xuất các phương án giải quyết việc phân chia hộp giới hạn khối phỏng lập phương chữ nhật, trong đó khối lập phương hoặc nút trong khi phân chia có thể được xác định ẩn là được phân chia QT hoặc BT, thay vì luôn là OT. Các bit chiếm giữ mà chỉ báo thông tin chiếm giữ có thể được tiết kiệm từ các phân chia QT và BT ẩn.

Đối với hộp giới hạn mà có thể không phải là khối lập phương lý tưởng, trong một vài trường hợp, các nút trong các mức khác nhau có thể không (hoặc không thể) được phân chia theo tất cả các chiều. Nếu việc phân chia được thực hiện trên tất cả ba chiều, việc phân chia thường là phân chia OT. Nếu việc phân chia được thực hiện trên hai trong số ba chiều, việc phân chia là phân chia QT trong không gian 3D. Nếu việc phân chia được thực hiện trên chỉ một chiều, thì việc phân chia là phân chia BT trong không gian 3D. Các ví dụ của các phân chia QT và BT trong không gian 3D được thể hiện trên FIG.8 và FIG.9, một cách lần lượt. Để minh họa, các hình vẽ thể hiện các phân chia QT và BT từ khối lập phương lý tưởng, nhưng lưu ý rằng có thể được phân chia từ bất kỳ khối lập phương dạng chữ nhật chung mà tạo thành hộp giới hạn.

FIG.8 thể hiện ba khối lập phương (801-803) được phân chia dọc theo các trục (hoặc các chiều) x-y, x-z, và y-z, một cách lần lượt. Các nút con trong mỗi khối lập phương (801-803) được gán các chỉ số mà là tập con của 8 chỉ số để đánh chỉ số 8 nút con trong phân chia OT trong ví dụ trên FIG.5. Với các chỉ số được gán, các phân chia QT với ba chiều khác nhau có thể được biểu diễn nhờ sử dụng các mã chiếm giữ trong cấu trúc cây bát phân. Ví dụ, mã chiếm giữ mà biểu diễn phân chia QT theo các trục x-y tới khối lập phương 801 có thể có dạng x0x0x0x0, trong đó bit tại vị trí x có thể được sử dụng để chỉ báo trạng thái chiếm giữ (ví dụ, có thể có giá trị 1 hoặc 0). Tương tự, mã chiếm giữ mà biểu diễn phân chia QT theo các trục x-z tới khối lập phương 802 có thể có dạng xx00xx00, trong khi mã chiếm giữ mà biểu diễn phân chia QT theo cách trục y-z tới khối lập phương 803 có thể có dạng xxxx0000. Như được thể hiện, các chỉ số

được gán tới các khối lập phương con này cũng chỉ báo các vị trí của các bit tương ứng với các nút con thu được trong 8-bit mã chiếm giữ.

FIG.9 thể hiện ba khối lập phương (901-903) được phân chia theo trục x, y, và z, một cách lần lượt. Theo cách thức tương tự với FIG.8, các nút con trong mỗi khối lập phương (901-903) được gán các chỉ số tương ứng với các vị trí trong mã chiếm giữ.

Khi các điều kiện định trước được thỏa mãn, các phân chia QT và BT có thể được thực hiện ẩn. "Ẩn" có nghĩa rằng không có bit báo hiệu bổ sung được cần thiết để chỉ báo phân chia QT hoặc BT được sử dụng thay vì phân chia OT. Bộ giải mã có thể xác định loại (ví dụ, phân chia QT hoặc BT) và chiều của việc phân chia theo cùng cách thức như bộ mã hóa dựa trên các điều kiện định trước. Ngoài ra, các bit có thể được tiết kiệm từ phân chia QT hoặc BT ẩn so với phân chia OT khi báo hiệu thông tin chiếm giữ của mỗi nút con. QT yêu cầu bốn bit, mà làm giảm từ tám, để biểu diễn trạng thái chiếm giữ của bốn nút con, trong khi BT chỉ yêu cầu hai bit. Ví dụ, như trong các ví dụ trên Fig.8-9, các bit trong mã chiếm giữ tương ứng với các chỉ số được gán tới các nút con có thể được báo hiệu, trong khi các bit khác trong mã chiếm giữ không liên quan đến các nút con có thể được bỏ qua (không được báo hiệu). Do đó, mã chiếm giữ có thể bao gồm các bit được bỏ qua và các bit được báo hiệu khi các phân chia QT và BT được đưa vào.

Lưu ý rằng các phân chia QT và BT có thể được thực hiện trong cùng cấu trúc của phân chia OT. Lựa chọn ngữ cảnh dựa trên các khối lập phương được mã hóa lân cận và bộ mã hóa entropy có thể được áp dụng trong các cách thức tương tự. Mô hình ngữ cảnh của mã chiếm giữ từ QT hoặc BT có thể được thay đổi theo dạng bất đối xứng của các nút con.

Việc mã hóa của các mã chiếm giữ của các phân chia QT và BT ẩn có thể được mô tả như trong các ví dụ sau đây. Đầu tiên, một ví dụ có thể giả định rằng mã chiếm giữ của OT được mã hóa trong thứ tự của các chỉ số như được thể hiện trên FIG.5. Sau đó, mã chiếm giữ của phân chia QT theo các trục x-y,

như được thể hiện trong biểu đồ ngoài cùng bên trái của FIG.8 (khối lập phương 801), có thể được mã hóa bằng cách bỏ qua các bit trong các vị trí 1, 3, 5, 7, khi chúng có thể được xem là 0 tại bộ giải mã, và chỉ các bit trong các vị trí 0, 2, 4, 6 được báo hiệu. Tương tự, đối với BT theo trục x, như được thể hiện trong biểu đồ ngoài cùng bên trái của FIG.9 (khối lập phương 901), thông tin chiếm giữ trong các vị trí 0 và 4 có thể được báo hiệu, và sáu bit khác có thể được xem là 0.

Ngoài ra, TMC13 hiện tại có chế độ đặc biệt để mã hóa hình học, mà là chế độ trực tiếp (DM-direct mode), và cho phép mã hóa (x, y, z) các vị trí trong nút con một cách trực tiếp mà không cần các phân chia thêm. Ví dụ, các vị trí là các vị trí tương đối so với gốc của nút con với mã hóa độ dài cố định, trong đó độ dài bit được xác định bởi chiều của nút con hiện tại. Do việc phân chia ẩn có thể dẫn đến các nút con có các kích cỡ không bằng nhau trong (x, y, z) các chiều, chế độ DM có thể được thay đổi tương ứng. Ví dụ, nếu nút con với kích cỡ bằng $(2^{d_x}, 2^{d_y}, 2^{d_z})$ cần được mã hóa trong chế độ DM, các vị trí tương đối của mỗi điểm trong các nút con được mã hóa bởi mã hóa độ dài cố định với (d_x, d_y, d_z) bit, một cách lần lượt.

3. Báo hiệu hộp giới hạn khối phỏng lập phương chữ nhật

Đầu tiên, hộp giới hạn B không bị giới hạn với kích cỡ giống nhau trong tất cả các chiều, thay vì đó, hộp giới hạn có thể có dạng khối phỏng lập phương chữ nhật có kích cỡ tùy ý để tương thích hơn với dạng của cảnh 3D hoặc các đối tượng. Trong các phương án khác nhau, kích cỡ của hộp giới hạn B có thể được biểu diễn là lũy thừa của hai, tức là, $(2^{d_x}, 2^{d_y}, 2^{d_z})$. $d_x, d_y,$ and d_z are được gọi là các kích cỡ log2 của hộp giới hạn. Lưu ý rằng d_x, d_y, d_z không được giả định bằng nhau, và có thể được báo hiệu riêng biệt trong thông tin tiêu đề chuỗi (ví dụ, tập hợp tham số chuỗi (SPS)) hoặc thông tin tiêu đề lát của dòng bit.

Ngoài ra, lưu ý rằng kích cỡ của hộp giới hạn B có thể là bất kỳ các số dương mà không giới hạn ở lũy thừa của hai. FIG.7 thể hiện ví dụ về hộp giới hạn phỏng lập phương chữ nhật để bao trùm cảnh, trong đó chiều z có kích cỡ nhỏ nhất.

Trong Phần III của phần mô tả chi tiết của các phương án, một vài phương án được thể hiện là các thay đổi so với các tiêu chuẩn kỹ thuật của phác thảo về nén đám mây điểm dựa trên hình học, ISO/IEC 23090-9:2019(E), giai đoạn WD, ISO/IEC JTC 1/SC 29/WG 11 W18179, tháng 3 năm 2019.

Phương án A

Trong một phương án, các kích cỡ hộp giới hạn của ba chiều có thể được báo hiệu trong thông tin tiêu đề lát hình học trong dạng của log2 như được thể hiện trong Bảng 1. Thông tin tiêu đề lát hình học có thể bao gồm các phần tử cú pháp được áp dụng tới lát. Nói chung, lát có thể liên quan đến một loạt các phần tử cú pháp mà biểu diễn một phần hoặc toàn bộ khung đám mây điểm được mã hóa. Các điểm của lát có thể được chứa trong hộp giới hạn tương ứng với lát.

Bảng 1

#	geometry_slice_header( ) {	Mô tả
1	gsh_geometry_parameter_set_id	ue(v)
2	gsh_tile_id	ue(v)
3	gsh_slice_id	ue(v)
4	if(gps_box_present_flag) {	
5	gsh_box_log2_scale	ue(v)
6	gsh_box_origin_x	ue(v)
7	gsh_box_origin_y	ue(v)
8	gsh_box_origin_z	ue(v)
9	}	

10*	gsh_log2_max_nodesize_x	ue(v)
11*	gsh_log2_max_nodesize_y	ue(v)
12*	gsh_log2_max_nodesize_z	ue(v)
13	gsh_points_number	ue(v)
14	byte_alignment( )	
15	}	

Cú pháp thông tin tiêu đề lát hình học trong Bảng 1 được cải biến bằng cách cộng các phần tử cú pháp sau đây tại các hàng 10-12:

gsh_log2_max_nodesize_x chỉ rõ kích cỡ hộp giới hạn trong chiều x, tức là, MaxNodesizeX mà được sử dụng trong xử lý giải mã như sau:

$$\text{MaxNodeSizeX} = 2^{\text{gsh\_log2\_max\_nodesize\_x}}.$$

$$\text{MaxNodeSizeLog2X} = \text{gsh\_log2\_max\_nodesize\_x}.$$

gsh_log2_max_nodesize_y chỉ rõ kích cỡ hộp giới hạn trong chiều y, tức là, MaxNodesizeY mà được sử dụng trong xử lý giải mã như sau:

$$\text{MaxNodeSizeY} = 2^{\text{gsh\_log2\_max\_nodesize\_y}}.$$

$$\text{MaxNodeSizeLog2Y} = \text{gsh\_log2\_max\_nodesize\_y}.$$

gsh_log2_max_nodesize_z chỉ rõ kích cỡ hộp giới hạn trong chiều z, tức là, MaxNodesizeZ mà được sử dụng trong xử lý giải mã như sau:

$$\text{MaxNodeSizeZ} = 2^{\text{gsh\_log2\_max\_nodesize\_z}}.$$

$$\text{MaxNodeSizeLog2Z} = \text{gsh\_log2\_max\_nodesize\_z}.$$

Phương án B

Trong phương án khác, các kích cỡ của ba chiều có thể được báo hiệu trong thông tin tiêu đề lát hình học trong dạng của log2. Thay vì báo hiệu ba giá trị một cách độc lập, có thể báo hiệu bởi các độ chênh lệch của chúng như sau.

Bảng 2

#	geometry_slice_header( ) {	Mô tả
1	gsh_geometry_parameter_set_id	ue(v)
2	gsh_tile_id	ue(v)
3	gsh_slice_id	ue(v)
4	if(gps_box_present_flag) {	
5	gsh_box_log2_scale	ue(v)
6	gsh_box_origin_x	ue(v)
7	gsh_box_origin_y	ue(v)
8	gsh_box_origin_z	ue(v)
9	}	
10*	gsh_log2_max_nodesize_x	ue(v)
11*	gsh_log2_max_nodesize_y_minus_x	se(v)
12*	gsh_log2_max_nodesize_z_minus_y	se(v)
13	gsh_points_number	ue(v)
14	byte_alignment( )	
15	}	

Cú pháp thông tin tiêu đề lát hình học trong Bảng 2 được cải biến bằng cách cộng các phần tử cú pháp sau đây tại các hàng 10-12:

gsh_log2_max_nodesize_x chỉ rõ kích cỡ hộp giới hạn trong chiều x, tức là, MaxNodesizeX mà được sử dụng trong xử lý giải mã như sau:

MaxNodeSizeX=2gsh_log2_max_nodesize_x.

MaxNodeSizeLog2X = gsh_log2_max_nodesize_x.

gsh_log2_max_nodesize_y_minus_x chỉ rõ kích cỡ hộp giới hạn trong chiều y, tức là, MaxNodesizeY mà được sử dụng trong xử lý giải mã như sau:

$$MaxNodeSizeY = 2^{gsh\_log2\_max\_nodesize\_y\_minus\_x+gsh\_log2\_max\_nodesize\_x}.$$

MaxNodeSizeLog2Y = gsh_log2_max_nodesize_y_minus_x + gsh_log2_max_nodesize_x.

gsh_log2_max_nodesize_z_minus_y chỉ rõ kích cỡ hộp giới hạn trong chiều z, tức là, MaxNodesizeZ mà được sử dụng trong xử lý giải mã như sau:

$$MaxNodeSizeZ = 2^{gsh\_log2\_max\_nodesize\_z\_minus\_y+MaxNodeSizeLog2Y}.$$

MaxNodeSizeLog2Z = gsh_log2_max_nodesize_z_minus_y + MaxNodeSizeLog2Y.

Phương án C

Trong phương án khác, các kích cỡ của ba chiều có thể được báo hiệu trong thông tin tiêu đề lát hình học bởi các vị trí Đề-các của chúng như sau.

Bảng 3

#	geometry_slice_header( ) {	Mô tả
1	gsh_geometry_parameter_set_id	ue(v)
2	gsh_tile_id	ue(v)
3	gsh_slice_id	ue(v)

4	if(gps_box_present_flag) {	
5	gsh_box_log2_scale	ue(v)
6	gsh_box_origin_x	ue(v)
7	gsh_box_origin_y	ue(v)
8	gsh_box_origin_z	ue(v)
9	}	
10*	gsh_max_nodesize_x	ue(v)
11*	gsh_max_nodesize_y	ue(v)
12*	gsh_max_nodesize_z	ue(v)
13	gsh_points_number	ue(v)
14	byte_alignment( )	
15	}	

Cú pháp thông tin tiêu đề lát hình học trong Bảng 3 được cải biến bằng cách cộng các phần tử cú pháp sau đây tại các hàng 10-12:

gsh_max_nodesize_x chỉ rõ kích cỡ hộp giới hạn trong chiều x, tức là, MaxNodeSizeX mà được sử dụng trong xử lý giải mã như sau:

$$\text{MaxNodeSizeX} = \text{gsh\_max\_nodesize\_x}.$$

$$\text{MaxNodeSizeLog2X} = \text{ilog2}(\text{MaxNodeSizeX}).$$

trong đó ilog2(v) tính toán số nguyên nhỏ nhất mà lớn hơn hoặc bằng log2 của v.

gsh_max_nodesize_y chỉ rõ kích cỡ hộp giới hạn trong chiều y, tức là, MaxNodeSizeY mà được sử dụng trong xử lý giải mã như sau:

$$\text{MaxNodeSizeY} = \text{gsh\_max\_nodesize\_y}.$$

MaxNodeSizeLog2Y = ilog2(MaxNodeSizeY).

gsh_max_nodesize_z chỉ rõ kích cỡ hộp giới hạn trong chiều z, tức là, MaxNodeSizeZ mà được sử dụng trong xử lý giải mã như sau:

MaxNodeSizeZ = gsh_max_nodesize_z.

MaxNodeSizeLog2Z = ilog2(MaxNodeSizeZ).

Phương án D

Trong phương án khác, các kích cỡ của ba chiều có thể được báo hiệu trong thông tin tiêu đề lát hình học bởi các vị trí Đề-các của chúng trừ đi 1 như sau.

Bảng 4

#	geometry_slice_header( ) {	Mô tả
1	gsh_geometry_parameter_set_id	ue(v)
2	gsh_tile_id	ue(v)
3	gsh_slice_id	ue(v)
4	if(gps_box_present_flag) {	
5	gsh_box_log2_scale	ue(v)
6	gsh_box_origin_x	ue(v)
7	gsh_box_origin_y	ue(v)
8	gsh_box_origin_z	ue(v)
9	}	
10*	gsh_max_nodesize_x_minus_one	ue(v)
11*	gsh_max_nodesize_y_minus_one	ue(v)

12*	gsh_max_nodesize_z_minus_one	ue(v)
13	gsh_points_number	ue(v)
14	byte_alignment( )	
15	}	

Cú pháp thông tin tiêu đề lát hình học trong Bảng 4 được cải biến bằng cách cộng các phần tử cú pháp sau đây tại các hàng 10-12:

gsh_max_nodesize_x_minus_one chỉ rõ kích cỡ hộp giới hạn trong chiều x, tức là, MaxNodeSizeX mà được sử dụng trong xử lý giải mã như sau:

$$\text{MaxNodeSizeX} = \text{gsh\_max\_nodesize\_x\_minus\_one} + 1.$$

$$\text{MaxNodeSizeLog2X} = \text{ilog2}(\text{MaxNodeSizeX}).$$

gsh_max_nodesize_y_minus_one chỉ rõ kích cỡ hộp giới hạn trong chiều y, tức là, MaxNodeSizeY mà được sử dụng trong xử lý giải mã như sau:

$$\text{MaxNodeSizeY} = \text{gsh\_max\_nodesize\_y\_minus\_one} + 1.$$

$$\text{MaxNodeSizeLog2Y} = \text{ilog2}(\text{MaxNodeSizeY}).$$

gsh_max_nodesize_z_minus_one chỉ rõ kích cỡ hộp giới hạn trong chiều z, tức là, MaxNodeSizeZ mà được sử dụng trong xử lý giải mã như sau:

$$\text{MaxNodeSizeZ} = \text{gsh\_max\_nodesize\_z\_minus\_one} + 1.$$

$$\text{MaxNodeSizeLog2Z} = \text{ilog2}(\text{MaxNodeSizeZ}).$$

Phương án E

Trong phương án khác, chỉ một trong số ba chiều cho phép có kích cỡ khác nhau. Trong trường hợp này, các kích cỡ của các chiều x và y là

giống nhau, và chiều z có thể khác nhau, do đó hai giá trị được báo hiệu trong thông tin tiêu đề lát hình học trong dạng của log2 như sau.

Bảng 5

#	geometry_slice_header( ) {	Mô tả
1	gsh_geometry_parameter_set_id	ue(v)
2	gsh_tile_id	ue(v)
3	gsh_slice_id	ue(v)
4	if(gps_box_present_flag) {	
5	gsh_box_log2_scale	ue(v)
6	gsh_box_origin_x	ue(v)
7	gsh_box_origin_y	ue(v)
8	gsh_box_origin_z	ue(v)
9	}	
10*	gsh_log2_max_nodesize_x_y	ue(v)
11*	gsh_log2_max_nodesize_z	ue(v)
12	gsh_points_number	ue(v)
13	byte_alignment( )	
14	}	

Cú pháp thông tin tiêu đề lát hình học trong Bảng 5 được cải biến bằng cách cộng các phần tử cú pháp sau đây tại các hàng 10-11:

gsh_log2_max_nodesize_x_y chỉ rõ kích cỡ hộp giới hạn trong các chiều x và y, tức là, MaxNodesizeX và MaxNodesizeY mà được sử dụng trong xử lý giải mã như sau:

$$\text{MaxNodeSizeX} = \text{MaxNodeSizeY} = 2^{\text{gsh\_log2\_max\_nodesize\_x\_y}}.$$

$$\text{MaxNodeSizeLog2X} = \text{MaxNodeSizeLog2Y} = \text{gsh\_log2\_max\_nodesize\_x\_y}.$$

gsh_log2_max_nodesize_z chỉ rõ kích cỡ hộp giới hạn trong chiều z, tức là, MaxNodesizeZ mà được sử dụng trong xử lý giải mã như sau:

$$\text{MaxNodeSizeZ} = 2^{\text{gsh\_log2\_max\_nodesize\_z}}.$$

$$\text{MaxNodeSizeLog2Z} = \text{gsh\_log2\_max\_nodesize\_z}.$$

Phương án F

Trong phương án khác, chỉ một trong số ba chiều cho phép có kích cỡ khác nhau. Trong trường hợp này, các kích cỡ của các chiều x và y là giống nhau, và chiều z có thể khác nhau, do đó hai giá trị được báo hiệu trong thông tin tiêu đề lát hình học trong dạng của log2. Thay vì báo hiệu hai giá trị một cách độc lập, có thể báo hiệu bởi các độ chênh lệch của chúng như sau.

Bảng 6

#	geometry_slice_header( ) {	Mô tả
1	gsh_geometry_parameter_set_id	ue(v)
2	gsh_tile_id	ue(v)
3	gsh_slice_id	ue(v)
4	if(gps_box_present_flag) {	
5	gsh_box_log2_scale	ue(v)
6	gsh_box_origin_x	ue(v)

7	gsh_box_origin_y	ue(v)
8	gsh_box_origin_z	ue(v)
9	}	
10*	gsh_log2_max_nodesize_x_y	ue(v)
11*	gsh_log2_max_nodesize_z_minus_xy	ue(v)
12	gsh_points_number	ue(v)
13	byte_alignment( )	
14	}	

Cú pháp thông tin tiêu đề lát hình học trong Bảng 6 được cải biến bằng cách cộng các phần tử cú pháp sau đây tại các hàng 10-11:

gsh_log2_max_nodesize_x_y chỉ rõ kích cỡ hộp giới hạn trong các chiều x và y, tức là, MaxNodesizeX và MaxNodesizeY mà được sử dụng trong xử lý giải mã như sau:

$$\text{MaxNodeSizeX} = \text{MaxNodeSizeY} = 2^{\text{gsh\_log2\_max\_nodesize\_x\_y}}.$$

$$\text{MaxNodeSizeLog2X} = \text{MaxNodeSizeLog2Y} = \text{gsh\_log2\_max\_nodesize\_x\_y}.$$

gsh_log2_max_nodesize_z_minus_xy chỉ rõ kích cỡ hộp giới hạn trong chiều z, tức là, MaxNodesizeZ mà được sử dụng trong xử lý giải mã như sau:

$$\text{MaxNodeSizeZ} = 2^{\text{gsh\_log2\_max\_nodesize\_z\_minus\_xy}+\text{gsh\_log2\_max\_nodesize\_x\_y}}.$$

$$\text{MaxNodeSizeLog2Z} = \text{gsh\_log2\_max\_nodesize\_z\_minus\_xy} + \text{gsh\_log2\_max\_nodesize\_x\_y}.$$

Phương án G

Trong phương án khác, chỉ một trong số ba chiều cho phép có kích cỡ khác nhau. Trong trường hợp này, các kích cỡ của các chiều x và y là giống nhau, và chiều z có thể khác nhau, do đó hai giá trị được báo hiệu trong thông tin tiêu đề lát hình học bởi các vị trí Đề-các của chúng như sau.

Bảng 7

#	geometry_slice_header( ) {	Mô tả
1	gsh_geometry_parameter_set_id	ue(v)
2	gsh_tile_id	ue(v)
3	gsh_slice_id	ue(v)
4	if(gps_box_present_flag) {	
5	gsh_box_log2_scale	ue(v)
6	gsh_box_origin_x	ue(v)
7	gsh_box_origin_y	ue(v)
8	gsh_box_origin_z	ue(v)
9	}	
10*	gsh_max_nodesize_x_y	ue(v)
11*	gsh_max_nodesize_z	ue(v)
12	gsh_points_number	ue(v)
13	byte_alignment( )	
14	}	

Cú pháp thông tin tiêu đề lát hình học trong Bảng 7 được cải biến bằng cách cộng các phần tử cú pháp sau đây tại các hàng 10-11:

gsh_max_nodesize_x_y chỉ rõ kích cỡ hộp giới hạn trong các chiều x và y, tức là, MaxNodesizeX và MaxNodesizeY mà được sử dụng trong xử lý giải mã như sau:

$$\text{MaxNodeSizeX} = \text{MaxNodeSizeY} = \text{gsh\_max\_nodesize\_x\_y}.$$

$$\text{MaxNodeSizeLog2X} = \text{MaxNodeSizeLog2Y} = \text{ilog2}(\text{gsh\_max\_nodesize\_x\_y}).$$

gsh_max_nodesize_z chỉ rõ kích cỡ hộp giới hạn trong chiều z, tức là, MaxNodeSizeZ mà được sử dụng trong xử lý giải mã như sau:

$$\text{MaxNodeSizeZ} = \text{gsh\_max\_nodesize\_z}.$$

$$\text{MaxNodeSizeLog2Z} = \text{ilog2}(\text{gsh\_max\_nodesize\_z}).$$

Phương án H

Trong phương án khác, chỉ một trong số ba chiều cho phép có kích cỡ khác nhau. Trong trường hợp này, các kích cỡ của các chiều x và y là giống nhau, và chiều z có thể khác nhau, do đó hai giá trị được báo hiệu trong thông tin tiêu đề lát hình học bởi các vị trí Đề-các của chúng trừ đi 1 như sau.

Bảng 8

#	geometry_slice_header( ) {	Mô tả
1	gsh_geometry_parameter_set_id	ue(v)
2	gsh_tile_id	ue(v)
3	gsh_slice_id	ue(v)
4	if(gps_box_present_flag) {	
5	gsh_box_log2_scale	ue(v)

6	gsh_box_origin_x	ue(v)
7	gsh_box_origin_y	ue(v)
8	gsh_box_origin_z	ue(v)
9	}	
10*	gsh_max_nodesize_x_y_minus_one	ue(v)
11*	gsh_max_nodesize_z_minus_one	ue(v)
12	gsh_points_number	ue(v)
13	byte_alignment( )	
14	}	

Cú pháp thông tin tiêu đề lát hình học trong Bảng 8 được cải biến bằng cách cộng các phần tử cú pháp sau đây tại các hàng 10-11:

gsh_max_nodesize_x_y_minus_one chỉ rõ kích cỡ hộp giới hạn trong các chiều x và y, tức là, MaxNodesizeX và MaxNodesizeY mà được sử dụng trong xử lý giải mã như sau:

$$\text{MaxNodeSizeX} = \text{MaxNodeSizeY} = \text{gsh\_max\_nodesize\_x\_y\_minus\_one} + 1.$$

$$\text{MaxNodeSizeLog2X} = \text{MaxNodeSizeLog2Y} = \text{ilog2}(\text{gsh\_max\_nodesize\_x\_y\_minus\_one} + 1).$$

gsh_max_nodesize_z_minus_one chỉ rõ kích cỡ hộp giới hạn trong chiều z, tức là, MaxNodeSizeZ mà được sử dụng trong xử lý giải mã như sau:

$$\text{MaxNodeSizeZ} = \text{gsh\_max\_nodesize\_z\_minus\_one} + 1.$$

MaxNodeSizeLog2Z = ilog2(gsh_max_nodesize_z_minus_one + 1).

4. Báo hiệu của phân chia QT và BT ẩn

Phương án A

Trong một phương án, cú pháp mã hóa hình học là như sau.

Bảng 9

#	geometry_slice_data( ) {	Mô tả
1	for(depth = 0; depth < MaxGeometryOctreeDepth; depth++) {	
2	for(nodeIdx = 0; nodeIdx < NumNodesAtDepth[depth]; nodeIdx++) {	
3 *	/* NB: implicit partition is described in semantics */	
4 *	partitionSkip = $b_x b_y b_z$ // depend on implicit partition conditions	
5 *	If (!(partitionSkip & 4))	
6 *	depthX = depthX + 1;	
7 *	If (!(partitionSkip & 2))	
8 *	depthY = depthY + 1;	
9 *	If (!(partitionSkip & 1))	
10*	depthZ = depthZ + 1;	

11*	xN = NodeX[depthX][nodeIdx]	
12*	yN = NodeY[depthY][nodeIdx]	
13*	zN = NodeZ[depthZ][nodeIdx]	
14*	geometry_node(depthX, depthY, depthZ, partitionSkip, nodeIdx, xN, yN, zN)	
15	}	
16	}	
17	if (log2_trisoup_node_size > 0)	
18	geometry_trisoup_data( )	
19	}	

Cú pháp dữ liệu lát hình học trong Bảng 9 được cải biến bằng cách cộng hoặc thay đổi các phần tử cú pháp tại các hàng 3-14. Tại hàng 14, các biến mới depthX, depthY, depthZ, và partitionSkip được thêm vào như là các đầu vào đối với hàm geometry_node.

Các biến NodeX[depthX][nodeIdx], NodeY[depthY][nodeIdx], và NodeZ[depthZ][nodeIdx] biểu diễn các tọa độ x, y, và z của nút thứ nodeIdx trong thứ tự giải mã tại độ sâu định trước. Biến NumNodesAtDepth[depth] biểu diễn số lượng nút cần được giải mã tại độ sâu định trước. Các biến depthX, depthY, và depthZ chỉ rõ độ sâu trong các chiều x, y và z, một cách lần lượt.

Biến partitionSkip chỉ rõ loại và chiều phân chia bởi Bảng đưới đây (Bảng 10).

Bảng 10

Loại và chiều phân chia	QT dọc theo các trục x-y	QT dọc theo các trục x-z	QT dọc theo các trục y-z	OT
partitionSkip	b001	b010	b100	b000
Loại và chiều phân chia	BT dọc theo trục x	BT dọc theo trục y	BT dọc theo trục z	
partitionSkip	b011	b101	b110	

Biến partitionSkip được biểu diễn trong dạng nhị phân với ba bit $b_x b_y b_z$ mà chỉ rõ rằng có bỏ qua phân chia theo chiều x, y và z, một cách lần lượt hay không. Ví dụ, $b_x = 1$ chỉ báo không phân chia theo chiều x (việc phân chia được bỏ qua). Loại và chiều phân chia có thể được xác định bởi các điều kiện định trước.

Bảng 11

0 *	geometry_node(depthX, depthY, depthZ, partitionSkip, nodeIdx, xN, yN, zN) {	Mô tả
1	if(NeighbourPattern == 0) {	
2	single_occupancy_flag	ae(v)
3	if(single_occupancy_flag)	
4 *	occupancy_idx	ae(v)
5	}	
6	if(!single_occupancy_flag)	

7	if(bitwise_occupancy_flag)	
8 *	occupancy_map	ae(v)
9	nếu không phải	
10	occupancy_byte	ae(v)
11*	/* NB: splitting of the current node is described in semantics */	
12*	if(depthX >= MaxNodeSizeLog2X − 1 && depthY >= MaxNodeSizeLog2Y – 1 && depthZ >= MaxNodeSizeLog2Z − 1) // [NB ie NodeSize = 2]	
13	if(!unique_geometry_points_flag)	
14	for(child = 0; child < GeometryNodeChildrenCnt; child++) {	
15	num_points_eq1_flag [Ed(df): _gt1_flag?]	ae(v)
16	if(!num_points_eq1_flag)	
17	num_points_minus2	ae(v)
18	}	
19	} else {	
20	if(DirectModeFlagPresent)	
21	direct_mode_flag	ae(v)
22	if(direct_mode_flag) {	

23	num_direct_points_minus1	ae(v)
24*	for(i = 0; i <= num_direct_points_minus1; i++)	
25*	for(j = 0; j < ChildNodeSizeLog2X; j++) {	
26*	point_rem_x[i][j]	ae(v)
27*	}	
28*	for(j = 0; j < ChildNodeSizeLog2Y; j++) {	
29*	point_rem_y[i][j]	ae(v)
30*	}	
31*	for(j = 0; j < ChildNodeSizeLog2Z; j++) {	
32*	point_rem_z[i][j]	ae(v)
33*	}	
34	}	
35	}	
36	}	

Cú pháp nút hình học trong Bảng 11 được cải biến bằng cách cộng hoặc thay đổi các hàng 0, 4, 8, 11-12, và 24-33. Tại hàng 0, các biến đầu vào mới depthX, depthY, depthZ, partitionSkip được đưa vào để hỗ trợ các phân chia QT và BT. Tại hàng 12, điều kiện để kiểm tra rằng kích cỡ nút nhỏ nhất đã được đạt tới hay chưa được cải biến để dựa trên đánh giá của ba chiều do các kích cỡ nút tại các chiều khác nhau có thể khác nhau khi các phân chia QT và BT được áp dụng.

Cấu trúc cú pháp tại các hàng 24-33 mô tả các phần tử cú pháp của các tọa độ điểm 3D trong nút con được xử lý với DM. Các biến ChildNodeSizeLog2X, ChildNodeSizeLog2Y và ChildNodeSizeLog2Z chỉ rõ kích cỡ nút con đối với mỗi chiều, và có thể được xác định bởi các phân chia QT và BT ẩn như sau:

NodeSizeLog2X = MaxNodeSizeXLog2 – depthX;

NodeSizeLog2Y = MaxNodeSizeYLog2 – depthY;

NodeSizeLog2Z = MaxNodeSizeZLog2 – depthZ;

if(partitionSkip==b001 || partitionSkip==b010 || partitionSkip==b011 || partitionSkip==b000)

ChildNodeSizeLog2X = NodeSizeLog2X – 1;

else

ChildNodeSizeLog2X = NodeSizeLog2X;

if(partitionSkip==b100 || partitionSkip==b001 || partitionSkip==b101 || partitionSkip==b000)

ChildNodeSizeLog2Y = NodeSizeLog2Y – 1;

else

ChildNodeSizeLog2Y = NodeSizeLog2Y;

if(partitionSkip==b010 || partitionSkip==b100 || partitionSkip==b110 || partitionSkip==b000)

ChildNodeSizeLog2Z = NodeSizeLog2Z – 1;

else

ChildNodeSizeLog2Z = NodeSizeLog2Z.

Trong xử lý nêu trên, nếu phân chia diễn ra tại một chiều, kích cỡ log2 nút con tại chiều này sẽ là kích cỡ log2 nút tại chiều này trừ 1.

Tại hàng 4, phần tử cú pháp occupancy_idx nhận dạng chỉ số của một nút con được chiếm giữ của nút hiện tại trong thứ tự duyệt nút con cây bát phân hình học. Khi biểu diễn, biến OccupancyMap có thể thu được như sau:

OccupancyMap = 1 << occupancy_idx.

Xử lý phân tích tương ứng của occupancy_idx có thể được mô tả như sau:

Đầu vào đối với xử lý này là biến partitionSkip của nút hiện tại.

Đầu ra từ xử lý này là giá trị phần tử cú pháp của occupancy_idx, được xây dựng như sau:

```
occupancy_idx = 0;
if( !(partitionSkip & 1) )
  occupancy_idx = readBinEq();
if( !(partitionSkip & 2) )
  occupancy_idx |= readBinEq() << 1;
if( !(partitionSkip & 4) )
  occupancy_idx |= readBinEq() << 2;
```

Trong xử lý nêu trên, đối với phân chia QT hoặc BT, chỉ các bit liên quan đến các chiều trong đó việc phân chia được diễn ra được báo hiệu. Do đó, ít bit hơn được báo hiệu so với phân chia OT để chỉ báo một vị trí nút con trong mã chiếm giữ. Toán tử hoặc đảo bit phức hợp |= "set" (set bằng 1) bit cụ thể trong biến occupancy_idx.

Tại hàng 8, phần tử cú pháp occupancy_map là sơ đồ bit mà nhận dạng các nút con được chiếm giữ của nút hiện tại. Khi biểu diễn, biến OccupancyMap được thiết lập bằng đầu ra của xử lý hoán vị sơ đồ chiếm giữ hình học khi được gọi ra với các biến NeighbourPattern và occupancy_map như là các đầu vào. NeighbourPattern (mẫu lân cận) là mẫu chiếm giữ lân cận đối với mô hình ngữ cảnh. Như được mô tả có viện dẫn tới các Fig.8-9, các ký hiệu nhị phân

trong mã chiếm giữ có thể được bỏ qua báo hiệu đối với phân chia QT hoặc BT. Bốn ký tự nhị phân có thể được bỏ qua đối với phân chia QT, trong khi 6 ký tự nhị phân có thể được bỏ qua đối với phân chia BT. Do đó, các bit tại các vị trí bit được bỏ qua này trong mã chiếm giữ có thể được xem là 0 tại a bộ giải mã.

Xử lý phân tích chiếm giữ cây bát phân hình học tương ứng có thể được mô tả như sau trong đó một vài bit được xem là 0 dựa trên biến partitionSkip mà chỉ báo loại phân chia và chiều phân chia nhờ sử dụng biến binIsSkiped[binIdx].

Xử lý này khôi phục phần tử cú pháp occupancy_map.

Đầu vào đối với xử lý này là NeighbourPattern, binIsSkiped và binIsInferred của nút hiện tại.

Đầu ra từ xử lý này là giá trị phần tử cú pháp, được xây dựng như sau:

```
value = 0;
for (binIdx = 0; binIdx < 8; binIdx++) {
    if( binIsSkiped[binIdx] )
                bin = 0;
    else if( binIsInferred )
                bin = 1;
    else
                bin = readBin(binIdx)
    value = value | (bin << bitCodingOrder[ binIdx ]);
}
```

trong đó biến binIsSkiped[binIdx] được thiết lập theo như sau,

```
if( (partitionSkip&1) &&
 ( inverseMap[NeighbourPattern][bitCodingOrder[binIdx]]==1
 || inverseMap[NeighbourPattern][bitCodingOrder[binIdx]]==3
```

```
|| inverseMap[NeighbourPattern][bitCodingOrder[binIdx]]==5
|| inverseMap[NeighbourPattern][bitCodingOrder[binIdx]]==7 )
  binIsSkiped[binIdx]=1;
else if( (partitionSkip&2) &&
( inverseMap[NeighbourPattern][bitCodingOrder[binIdx]]==2
|| inverseMap[NeighbourPattern][bitCodingOrder[binIdx]]==3
|| inverseMap[NeighbourPattern][bitCodingOrder[binIdx]]==6
|| inverseMap[NeighbourPattern][bitCodingOrder[binIdx]]==7 )
  binIsSkiped[binIdx]=1;
else if( (partitionSkip&4) &&
( inverseMap[NeighbourPattern][bitCodingOrder[binIdx]]==4
|| inverseMap[NeighbourPattern][bitCodingOrder[binIdx]]==5
|| inverseMap[NeighbourPattern][bitCodingOrder[binIdx]]==6
|| inverseMap[NeighbourPattern][bitCodingOrder[binIdx]]==7 )
  binIsSkiped[binIdx]=1;
else
  binIsSkiped[binIdx]=0.
```

bitCodingOrder[i] được xác định bởi Bảng dưới đây:

Bảng 12

i	0	1	2	3	4	5	6	7
bitCodingOrder[i]	1	7	5	3	2	4	6	0

inverseMap[i][j] được xác định bởi các bảng dưới đây (các bảng 13-14):

Bảng 13

inverseMap[i][j]	j							
i	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	0	1	2	3	4	5	6	7
2	6	7	4	5	2	3	0	1
3	0	1	2	3	4	5	6	7
4	2	3	6	7	0	1	4	5
5	2	3	6	7	0	1	4	5
6	6	7	4	5	2	3	0	1
7	4	5	0	1	6	7	2	3
8	4	5	0	1	6	7	2	3
9	0	1	2	3	4	5	6	7
10	4	5	0	1	6	7	2	3
11	2	3	6	7	0	1	4	5
12	4	5	0	1	6	7	2	3
13	6	7	4	5	2	3	0	1
14	0	1	2	3	4	5	6	7
15	0	1	2	3	4	5	6	7

16	5	1	7	3	4	0	6	2
17	3	1	2	0	7	5	6	4
18	5	7	4	6	1	3	0	2
19	2	0	6	4	3	1	7	5
20	7	3	6	2	5	1	4	0
21	3	2	7	6	1	0	5	4
22	7	6	5	4	3	2	1	0
23	5	4	1	0	7	6	3	2
24	1	5	0	4	3	7	2	6
25	1	0	3	2	5	4	7	6
26	5	4	1	0	7	6	3	2
27	3	2	7	6	1	0	5	4
28	0	4	2	6	1	5	3	7
29	7	6	5	4	3	2	1	0
30	1	0	3	2	5	4	7	6
31	1	0	3	2	5	4	7	6
32	4	0	6	2	5	1	7	3

Bảng 14 (tiếp tục từ Bảng 13)

inverseMap[i][j]	j							
i	0	1	2	3	4	5	6	7
33	2	0	3	1	6	4	7	5
34	4	6	5	7	0	2	1	3
35	3	1	7	5	2	0	6	4
36	6	2	7	3	4	0	5	1
37	2	3	6	7	0	1	4	5
38	6	7	4	5	2	3	0	1
39	4	5	0	1	6	7	2	3
40	0	4	1	5	2	6	3	7
41	0	1	2	3	4	5	6	7
42	4	5	0	1	6	7	2	3
43	2	3	6	7	0	1	4	5
44	1	5	3	7	0	4	2	6
45	6	7	4	5	2	3	0	1
46	0	1	2	3	4	5	6	7
47	0	1	2	3	4	5	6	7
48	4	0	6	2	5	1	7	3

49	7	5	6	4	3	1	2	0
50	1	3	0	2	5	7	4	6
51	2	0	3	1	6	4	7	5
52	5	1	4	0	7	3	6	2
53	0	4	1	5	2	6	3	7
54	2	0	3	1	6	4	7	5
55	6	4	2	0	7	5	3	1
56	3	7	2	6	1	5	0	4
57	4	6	5	7	0	2	1	3
58	6	2	7	3	4	0	5	1
59	0	2	4	6	1	3	5	7
60	0	4	1	5	2	6	3	7
61	2	6	0	4	3	7	1	5
62	4	0	6	2	5	1	7	3
63	0	1	2	3	4	5	6	7

Phương án B

Trong phương án khác, chỉ một trong số ba chiều cho phép có kích cỡ khác nhau. Trong trường hợp này, nếu chiều này có kích cỡ lớn hơn hai chiều còn lại, chỉ các phân chia BT ẩn theo chiều này được thực hiện. Nếu chiều này có kích cỡ nhỏ hơn hai chiều còn lại, chỉ các phân chia QT ẩn theo hai chiều còn lại được thực hiện. Báo hiệu của BT ẩn hoặc QT ẩn trong phương án này là tương tự như được mô tả trong phương án trước đó.

Phương án C

Các tham số có thể được xác định để chỉ rõ các điều kiện định trước của các phân chia QT và BT ẩn. Các tham số này có thể là cố định đối với bộ mã hóa và bộ giải mã (lấy giá trị được cấu hình trước cục bộ, như giá trị mặc định), hoặc chúng có thể được báo hiệu tại thông tin tiêu đề của dòng bit để cho phép các tối ưu hóa mức chuỗi hoặc mức lát. Các phương án C-G được mô tả dưới đây thể hiện làm thế nào các tham số hữu ích để xác định các điều kiện của các phân chia QT và BT ẩn được báo hiệu hoặc được cấu hình.

Trong phương án C, các tham số được báo hiệu, mà có thể nằm trong thông tiêu đề chuỗi hoặc lát, như sau.

Bảng 15

#	geometry_parameter_set( ) {	Mô tả
1	gps_geom_parameter_set_id	ue(v)
2	gps_seq_parameter_set_id	ue(v)
3	gps_box_present_flag	u(1)
4	unique_geometry_points_flag	u(1)
5	neighbour_context_restriction_flag	u(1)
6	inferred_direct_coding_mode_enabled_flag	u(1)
7	bitwise_occupancy_coding_flag	u(1)
8	child_neighbours_enabled_flag	u(1)
9	geom_occupancy_ctx_reduction_factor	ue(v)
10	log2_neighbour_avail_boundary	ue(v)

11	log2_intra_pred_max_node_size	ue(v)
12	log2_trisoup_node_size	ue(v)
13*	gps_max_num_implicit_qtbt_before_ot	ue(v)
14*	gps_min_size_implicit_qtbt	ue(v)
15*	gps_implicit_bt_before_implicit_qt_flag	u(1)
16	gps_extension_present_flag	u(1)
17	if(gps_extension_present_flag)	
18	while(more_data_in_byte_stream( ))	
19	gps_extension_data_flag	u(1)
20	byte_alignment( )	
21	}	

Cú pháp tập hợp tham số hình học trong Bảng 15 được cải biến bằng cách bổ sung các phần tử cú pháp để báo hiệu các tham số để điều khiển các phân chia QT và BT tại các hàng 13-15.

gps_max_num_implicit_qtbt_before_ot chỉ rõ số lượng phân chia QT và BT ẩn lớn nhất trước các phân chia OT, mà được ký hiệu bởi K.

gps_min_size_implicit_qtbt chỉ rõ kích cỡ nhỏ nhất của các phân chia QT và BT ẩn, mà được ký hiệu bởi M. Tham số này ngăn chặn các phân chia QT và BT ẩn khi tất cả các chiều của nút hiện tại nhỏ hơn hoặc bằng M.

gps_implicit_bt_before_implicit_qt_flag chỉ rõ mức ưu tiên của các phân chia QT và BT ẩn, mà được ký hiệu bởi $BTFirst$. Nếu $BTFirst = 1$, các phân chia BT ẩn được thực hiện trước các phân chia QT ẩn. Nếu $BTFirst = 0$, các phân chia QT ẩn được thực hiện trước các phân chia BT ẩn.

Phương án D

Trong phương án khác, một phần tham số được báo hiệu trong thông tin tiêu đề chuỗi hoặc lát, trong khi phần còn lại được cố định. Ví dụ sau đây cố định M và BTFirst, và các báo hiệu K bởi gps_max_num_implicit_qtbt_before_ot.

Bảng 16

#	geometry_parameter_set( ) {	Mô tả
1	gps_geom_parameter_set_id	ue(v)
2	gps_seq_parameter_set_id	ue(v)
3	gps_box_present_flag	u(1)
4	unique_geometry_points_flag	u(1)
5	neighbour_context_restriction_flag	u(1)
6	inferred_direct_coding_mode_enabled_flag	u(1)
7	bitwise_occupancy_coding_flag	u(1)
8	child_neighbours_enabled_flag	u(1)
9	geom_occupancy_ctx_reduction_factor	ue(v)
10	log2_neighbour_avail_boundary	ue(v)
11	log2_intra_pred_max_node_size	ue(v)
12	log2_trisoup_node_size	ue(v)
13*	gps_max_num_implicit_qtbt_before_ot	ue(v)
14	gps_extension_present_flag	u(1)

15	if(gps_extension_present_flag)	
16	while(more_data_in_byte_stream( ))	
17	gps_extension_data_flag	u(1)
18	byte_alignment( )	
19	}	

Cú pháp tập hợp tham số hình học trong Bảng 16 được cải biến bằng cách bổ sung phần tử cú pháp để báo hiệu tham số để điều khiển các phân chia QT và BT tại các hàng 13.

gps_max_num_implicit_qtbt_before_ot chỉ rõ số lượng lớn nhất của các phân chia QT và BT ẩn trước các phân chia OT. Trong trường hợp này, các tham số khác được cố định và do đó không được báo hiệu trong dòng bit, ví dụ, M luôn bằng 0, và $BTFirst$ luôn bằng 1.

Phương án E

Trong phương án khác, K và BTFirst được cố định, và M được báo hiệu bởi gps_min_size_implicit_qtbt, như sau.

Bảng 17

#	geometry_parameter_set( ) {	Mô tả
1	gps_geom_parameter_set_id	ue(v)
2	gps_seq_parameter_set_id	ue(v)
3	gps_box_present_flag	u(1)
4	unique_geometry_points_flag	u(1)
5	neighbour_context_restriction_flag	u(1)

6	inferred_direct_coding_mode_enabled_flag	u(1)
7	bitwise_occupancy_coding_flag	u(1)
8	child_neighbours_enabled_flag	u(1)
9	geom_occupancy_ctx_reduction_factor	ue(v)
10	log2_neighbour_avail_boundary	ue(v)
11	log2_intra_pred_max_node_size	ue(v)
12	log2_trisoup_node_size	ue(v)
13*	gps_min_size_implicit_qtbt	ue(v)
14	gps_extension_present_flag	u(1)
15	if(gps_extension_present_flag)	
16	while(more_data_in_byte_stream( ))	
17	gps_extension_data_flag	u(1)
18	byte_alignment( )	
19	}	

Cú pháp tập hợp tham số hình học trong Bảng 17 được cải biến bằng cách bổ sung phần tử cú pháp để báo hiệu tham số để điều khiển các phân chia QT và BT tại các hàng 13.

gps_min_size_implicit_qtbt chỉ rõ kích cỡ nhỏ nhất của các phân chia QT và BT ẩn, mà là M. Tham số này ngăn chặn các phân chia QT và BT ẩn khi tất cả các chiều nhỏ hơn hoặc bằng M. Trong trường hợp này, các tham số khác được cố định và do đó không được báo hiệu trong dòng bit, ví dụ, K luôn bằng 0, và $BTFirst$ luôn bằng 1.

Phương án F

Trong phương án khác, báo hiệu của các tham số để phân chia ẩn phụ thuộc vào các cú pháp khác. Trong ví dụ dưới đây, báo hiệu của các tham số đối với phân chia ẩn phụ thuộc vào log2_trisoup_node_size như sau.

Bảng 18

#	geometry_parameter_set( ) {	Mô tả
1	gps_geom_parameter_set_id	ue(v)
2	gps_seq_parameter_set_id	ue(v)
3	gps_box_present_flag	u(1)
4	unique_geometry_points_flag	u(1)
5	neighbour_context_restriction_flag	u(1)
6	inferred_direct_coding_mode_enabled_flag	u(1)
7	bitwise_occupancy_coding_flag	u(1)
8	child_neighbours_enabled_flag	u(1)
9	geom_occupancy_ctx_reduction_factor	ue(v)
10	log2_neighbour_avail_boundary	ue(v)
11	log2_intra_pred_max_node_size	ue(v)
12	log2_trisoup_node_size	ue(v)
13*	if(log2_trisoup_node_size == 0) {	
14*	gps_max_num_implicit_qtbt_before_ot	ue(v)
15*	}	

16	gps_extension_present_flag	u(1)
17	if(gps_extension_present_flag)	
18	while(more_data_in_byte_stream( ))	
19	gps_extension_data_flag	u(1)
20	byte_alignment( )	
21	}	

Cú pháp tập hợp tham số hình học trong Bảng 18 được cải biến bằng cách bổ sung cấu trúc cú pháp để báo hiệu tham số để điều khiển các phân chia QT và BT tại các hàng 13-15.

gps_max_num_implicit_qtbt_before_ot chỉ rõ số lượng phân chia QT và BT ẩn lớn nhất trước các phân chia OT, mà là K. Trong trường hợp này, gps_max_num_implicit_qtbt_before_ot được báo hiệu chỉ nếu log2_trisoup_node_size là 0. Nếu log2_trisoup_node_size không phải 0, K sẽ được thiết lập là giá trị lớn nhất một cách mặc định. Các tham số khác được cố định và do đó không được báo hiệu trong dòng bit, ví dụ, M luôn bằng 0, và $BTFirst$ luôn bằng 1.

Phương án G

Trong phương án khác, các tham số đối với phân chia ẩn này không được báo hiệu, và các tham số này đều có thể là cố định. Ví dụ, chúng được cố định là K=3, M=0, và BTFirst=1.

5. Các điều kiện đối với các phân chia QT và BT ẩn

Các phương án của các điều kiện khác nhau đối với các phân chia QT và BT ẩn được mô tả trong các phần phụ như sau. Bằng cách sử dụng các tham số điều khiển khác nhau và thiết lập các điều kiện khác nhau, các phương pháp hoặc các xử lý phân chia hình học khác nhau có thể được thực hiện.

5.1 Thực hiện các phân chia QT và BT ẩn sau OT

Trong phương pháp thứ nhất, các phân chia OT được thực hiện hoàn chỉnh cho đến khi một vài chiều không thể được phân chia tiếp. Do đó, điều kiện để thực hiện các phân chia QT và BT ẩn trong phương pháp này là khi một hoặc hai chiều đạt tới đơn vị phân chia nhỏ nhất (tức là, một điểm ảnh ba chiều).

Cụ thể, loại và chiều phân chia có thể được xác định theo Bảng 19 hoặc Bảng 20. Tham số ưu tiên BTFirst là cố định hoặc được chỉ rõ bởi gps_implicit_bt_before_implicit_qt_flag. Nếu BTFirst = 1, các phân chia BT ẩn có mức ưu tiên cao hơn và sẽ được thực hiện trước các phân chia QT ẩn, và Bảng 19 áp dụng trong trường hợp này. Nếu BTFirst = 0, các phân chia QT ẩn có mức ưu tiên cao hơn và sẽ được thực hiện trước các phân chia BT ẩn, và Bảng 20 áp dụng trong trường hợp này. Nếu các điều kiện được liệt kê trong các bảng không được thỏa mãn, phân chia OT được thực hiện.

Bảng 19 các điều kiện để thực hiện phân chia ẩn khi BTFirst=1

Loại và chiều phân chia	QT dọc theo các trục x-y	QT dọc theo các trục x-z	QT dọc theo các trục y-z
Điều kiện	$d_z = 0 < d_x = d_y$	$d_y = 0 < d_x = d_z$	$d_x = 0 < d_y = d_z$
Loại và chiều phân chia	BT dọc theo trục x	BT along y axis	BT dọc theo trục z
Điều kiện	$d_y = 0 \leq d_z < d_x$	$d_x = 0 \leq d_z < d_y$	$d_x = 0 \leq d_y < d_z$
	$d_z = 0 \leq d_y < d_x$	$d_z = 0 \leq d_x < d_y$	$d_y = 0 \leq d_x < d_z$

Bảng 20 các điều kiện để thực hiện phân chia ẩn khi BTFirst=0

Loại và chiều phân chia	QT dọc theo các trục x-y	QT dọc theo các trục x-z	QT dọc theo các trục y-z
Điều kiện	$d_z = 0 < d_x \leq d_y$	$d_y = 0 < d_x \leq d_z$	$d_x = 0 < d_y \leq d_z$
Loại và chiều phân chia	BT dọc theo trục x	BT along y axis	BT dọc theo trục z
Điều kiện	$d_y = d_z = 0 < d_x$	$d_x = d_z = 0 < d_y$	$d_x = d_y = 0 < d_z$

Đặt hộp giới hạn B có kích cỡ bằng $(2^{d_x}, 2^{d_y}, 2^{d_z})$. Không cần tổn hao chung, có thể giả định rằng $0 < d_x \leq d_y \leq d_z$. Hai phương án được mô tả dưới đây.

Trong một phương án, BTFirst = 1 được cố định hoặc được chỉ rõ bởi gps_implicit_bt_before_implicit_qt_flag, mà chỉ báo rằng các phân chia BT ẩn có mức ưu tiên cao hơn và sẽ được thực hiện trước các phân chia QT ẩn, và Bảng 19 được áp dụng. Trong phương án này, các phân chia OT có thể được thực hiện tại d_x độ sâu phân chia đầu tiên. Sau các OT, các nút con sẽ có dạng $2^{(0,d_y-d_x,d_z-d_x)}$. Sau đó, các phân chia BT ẩn sẽ được thực hiện theo trục z tại $d_z - d_y$ độ sâu tiếp theo. Sau các phân chia BT ẩn, dạng của các nút con sẽ là $2^{(0,d_y-d_x,d_y-d_x)}$, và sau đó các phân chia QT ẩn được thực hiện theo các trục y-z tại $d_y - d_x$ độ sâu cuối cùng cho đến khi chạm tới các nút nhánh.

Trong phương án khác, BTFirst = 0 được cố định hoặc được chỉ rõ bởi gps_implicit_bt_before_implicit_qt_flag, mà chỉ báo rằng các phân chia QT ẩn có mức ưu tiên cao hơn và sẽ được thực hiện trước các phân chia BT ẩn,

và Bảng 20 được áp dụng. Trong phương án này, các phân chia OT sẽ được thực hiện tại d_x độ sâu phân chia đầu tiên. Sau các OT, các nút con sẽ có dạng $2^{(0,d_y-d_x,d_z-d_x)}$. Sau đó, các phân chia QT ẩn có thể được thực hiện theo các trục y-z tại $d_y - d_x$ độ sâu tiếp theo. Sau các phân chia QT ẩn, dạng của các nút con sẽ là $2^{(0,0,d_z-d_y)}$, và sau đó các phân chia BT ẩn được thực hiện theo các trục z tại $d_z - d_y$ độ sâu cuối cùng cho đến khi chạm tới các nút nhánh.

5.2 Thực hiện các phân chia QT và BT ẩn trước OT

Phương pháp thứ hai là để thực hiện các phân chia QT và BT ẩn trước bất kỳ phân chia OT để tạo ra các nút con có dạng khối lập phương. Do đó, trong trường hợp này, điều kiện là một hoặc hai chiều có các kích cỡ nhỏ hơn chiều lớn nhất.

Cụ thể, loại và chiều phân chia có thể được xác định theo Bảng 21 hoặc Bảng 22. Tham số ưu tiên BTFirst là cố định hoặc được chỉ rõ bởi gps_implicit_bt_before_implicit_qt_flag. Nếu BTFirst $= 1$, các phân chia BT ẩn có mức ưu tiên cao hơn và sẽ được thực hiện trước các phân chia QT ẩn, và Bảng 21 được áp dụng. Nếu BTFirst $= 0$, các phân chia QT ẩn có mức ưu tiên cao hơn và sẽ được thực hiện trước các phân chia BT ẩn, và Bảng 22 được áp dụng các. Nếu các điều kiện được liệt kê trong các bảng không được thỏa mãn, phân chia OT được thực hiện.

Bảng 21 các điều kiện để thực hiện phân chia ẩn khi BTFirst=1

Loại và chiều phân chia	QT dọc theo các trục x-y	QT dọc theo các trục x-z	QT dọc theo các trục y-z
Điều kiện	$d_z < d_x = d_y$	$d_y < d_x = d_z$	$d_x < d_y = d_z$
Loại và chiều phân chia	BT dọc theo trục x	BT along y axis	BT dọc theo trục z

Điều kiện	$d_y < d_x \; and \; d_z < d_x$	$d_x < d_y \; and \; d_z < d_y$	$d_x < d_z \; and \; d_y < d_z$

Bảng 22 các điều kiện để thực hiện phân chia ẩn khi BTFirst=0

Loại và chiều phân chia	QT dọc theo các trục x-y	QT dọc theo các trục x-z	QT dọc theo các trục y-z
Điều kiện	$d_z < d_x \; and \; d_z < d_y$	$d_y < d_x \; and \; d_y < d_z$	$d_x < d_y \; and \; d_x < d_z$
Loại và chiều phân chia	BT dọc theo trục x	BT along y axis	BT dọc theo trục z
Điều kiện	$d_y = d_z < d_x$	$d_x = d_z < d_y$	$d_x = d_y < d_z$

Đặt hộp giới hạn B có kích cỡ bằng $(2^{d_x}, 2^{d_y}, 2^{d_z})$. Không cần tổn hao chung, có thể giả định rằng $0 < d_x \leq d_y \leq d_z$. Hai phương án được mô tả dưới đây.

Trong một phương án, BTFirst = 1 được cố định hoặc được chỉ rõ bởi gps_implicit_bt_before_implicit_qt_flag, mà chỉ báo rằng các phân chia BT ẩn có mức ưu tiên cao hơn và sẽ được thực hiện trước các phân chia QT ẩn, và Bảng 21 được áp dụng. Trong phương án này, các phân chia BT ẩn sẽ được thực hiện theo trục z tại $d_z - d_y$ độ sâu đầu tiên, và sau đó các phân chia QT ẩn được thực hiện theo các trục y-z tại $d_y - d_x$ độ sâu tiếp theo. Sau các phân chia QT và BT ẩn, kích cỡ của tất cả nút con sẽ là $2^{(d_x,d_x,d_x)}$, các phân chia OT được thực hiện d_x lần để chạm tới các nút nhánh.

Trong phương án khác, BTFirst = 0 được cố định hoặc được chỉ rõ bởi gps_implicit_bt_before_implicit_qt_flag, mà chỉ báo rằng các phân chia

QT ẩn có mức ưu tiên cao hơn và sẽ được thực hiện trước các phân chia BT ẩn, và Bảng 22 được áp dụng. Trong phương án này, các phân chia QT ẩn sẽ được thực hiện theo các trục y-z tại $d_y - d_x$ độ sâu đầu tiên, và sau đó các phân chia BT ẩn được thực hiện theo trục z tại $d_z - d_y$ độ sâu tiếp theo. Sau các phân chia QT và BT ẩn, kích cỡ của tất cả nút con sẽ là $2^{(d_x,d_x,d_x)}$, các phân chia OT được thực hiện d_x lần để chạm tới các nút nhánh.

5.3 Phương pháp lai của các phân chia QT và BT ẩn

Phương pháp thứ ba là kết hợp của các phương pháp trong các phần III.5.1 và III.5.3. Trong trường hợp này, ngưỡng $K (0 \leq K \leq \max(d_x, d_y, d_z) - \min(d_x, d_y, d_z))$ xác định số lần phân chia QT và BT ẩn lớn nhất mà có thể được thực hiện trước các phân chia OT. Phương pháp này là tổng quát hóa của hai phương pháp đầu tiên, phương pháp này hạ xuống phương pháp của phần III.5.1 khi $K = 0$, và phương pháp này hạ xuống phương pháp của phần III.5.2 khi $K = \max(d_x, d_y, d_z) - \min(d_x, d_y, d_z)$.

Cụ thể, tại K độ sâu phân chia đầu tiên, các quyết định đối với loại và chiều phân chia tuân theo các điều kiện được xác định trong Bảng 21 hoặc 22, sau đó tuân theo Bảng 19 hoặc 20. K được cố định hoặc được chỉ rõ bởi gps_max_num_implicit_qtbt_before_ot. Tham số ưu tiên BTFirst là cố định hoặc được chỉ rõ bởi gps_implicit_bt_before_implicit_qt_flag. Nếu $BTFirst = 1$, các phân chia BT ẩn có mức ưu tiên cao hơn và sẽ được thực hiện trước các phân chia QT ẩn, và Bảng 19 và Bảng 21 được áp dụng. Nếu $BTFirst = 0$, các phân chia QT ẩn có mức ưu tiên cao hơn và sẽ được thực hiện trước các phân chia BT ẩn, và Bảng 20 và Bảng 22 được áp dụng. Nếu các điều kiện được liệt kê trong các bảng không được thỏa mãn, phân chia OT được thực hiện.

Đặt hộp giới hạn B có kích cỡ bằng $(2^{d_x}, 2^{d_y}, 2^{d_z})$. Không cần tổn hao chung, có thể giả định rằng $0 < d_x \leq d_y \leq d_z$.

Trong một phương án, K được cố định hoặc được chỉ rõ bởi gps_max_num_implicit_qtbt_before_ot, mà chỉ báo rằng K lần phân chia BT và

QT ẩn sẽ được thực hiện đầu tiên. BTFirst = 1 được cố định hoặc được chỉ rõ bởi gps_implicit_bt_before_implicit_qt_flag, mà chỉ báo rằng các phân chia BT ẩn có mức ưu tiên cao hơn và sẽ được thực hiện trước các phân chia QT ẩn, và Bảng 19 và Bảng 21 được áp dụng. Trong phương án này, tại K ($K \leq d_z - d_x$) độ sâu đầu tiên, các phân chia BT ẩn được thực hiện theo trục z và các phân chia QT ẩn sau đó được thực hiện theo các trục y-z theo Bảng 21. Sau đó, kích cỡ của các nút con là $2^{(d_x, d_x+\delta_y, d_x+\delta_z)}$, trong đó các giá trị của δ_y và δ_z ($\delta_z \geq \delta_y \geq 0$) phụ thuộc vào giá trị của K. Sau đó, các phân chia OT được thực hiện d_x lần mà làm cho các nút con còn lại có kích cỡ bằng $2^{(0,\delta_y,\delta_z)}$. Cuối cùng, theo Bảng 19, các phân chia BT ẩn được thực hiện theo trục z $\delta_z - \delta_y$ lần, và các phân chia QT ẩn sau đó được thực hiện theo các trục y-z δ_y lần.

Trong phương án khác, K được cố định hoặc được chỉ rõ bởi gps_max_num_implicit_qtbt_before_ot, mà chỉ báo rằng K lần phân chia BT và QT ẩn sẽ được thực hiện đầu tiên. BTFirst = 0 được cố định hoặc được chỉ rõ bởi gps_implicit_bt_before_implicit_qt_flag, mà chỉ báo rằng các phân chia QT ẩn có mức ưu tiên cao hơn và sẽ được thực hiện trước các phân chia BT ẩn, và Bảng 20 và Bảng 22 được áp dụng. Trong phương án này, tại K ($K \leq d_z - d_x$) độ sâu đầu tiên, các phân chia QT ẩn được thực hiện theo các trục y-z và các phân chia BT ẩn sau đó được thực hiện theo trục z theo Bảng 22. Sau đó, kích cỡ của các nút con là $2^{(d_x, d_x+\delta_y, d_x+\delta_z)}$, trong đó các giá trị của δ_y và δ_z ($\delta_z \geq \delta_y \geq 0$) phụ thuộc vào giá trị của K. Sau đó, các phân chia OT được thực hiện d_x lần mà làm cho các nút con còn lại có kích cỡ bằng $2^{(0,\delta_y,\delta_z)}$. Cuối cùng, theo Bảng 20, các phân chia QT ẩn được thực hiện theo các trục y-z δ_y lần, và các phân chia BT ẩn sau đó được thực hiện theo trục z $\delta_z - \delta_y$ lần.

5.4 Phương pháp lai với kích cỡ nhỏ nhất của các phân chia QT và BT ẩn

Phương pháp thứ tư đặt thêm ràng buộc đối với các phương pháp trước đó. Trong trường hợp này, ngưỡng K ($0 \leq K \leq \max(d_x, d_y, d_z) -$

$\min(d_x, d_y, d_z)$) xác định số lần phân chia QT và BT ẩn lớn nhất mà có thể được thực hiện trước các phân chia OT. Tham số khác M ($0 \leq M \leq \min(d_x, d_y, d_z)$) xác định kích cỡ nhỏ nhất của các phân chia QT và BT ẩn, và ngăn chặn các phân chia QT và BT ẩn khi tất cả các chiều nhỏ hơn hoặc bằng M. Phương pháp thứ tư là tổng quát hóa của ba phương pháp đầu tiên. Phương pháp thứ tư hạ xuống phương pháp của phần III.5.1 khi $K = M = 0$, và hạ xuống phương pháp của phần III.5.2 khi $K = \max(d_x, d_y, d_z) - \min(d_x, d_y, d_z)$, $M = 0$, và hạ xuống phương pháp của phần III.5.3 khi $0 < K < \max(d_x, d_y, d_z) - \min(d_x, d_y, d_z)$, $M = 0$.

Cụ thể, tại K mức phân chia đầu tiên, các quyết định đối với loại và chiều phân chia tuân theo các điều kiện được xác định trong Bảng 21 hoặc 22, sau đó tuân theo Bảng 23 hoặc 24. Bảng 23 và 24 tương tự Bảng 19 và 20 bằng cách thay thế 0 với M. K được cố định hoặc được chỉ rõ bởi gps_max_num_implicit_qtbt_before_ot. M được cố định hoặc được chỉ rõ bởi gps_min_size_implicit_qtbt. Tham số ưu tiên BTFirst là cố định hoặc được chỉ rõ bởi gps_implicit_bt_before_implicit_qt_flag. Nếu $BTFirst = 1$, các phân chia BT ẩn có mức ưu tiên cao hơn và sẽ được thực hiện trước các phân chia QT ẩn, và Bảng 21 và Bảng 23 được áp dụng. Nếu $BTFirst = 0$, các phân chia QT ẩn có mức ưu tiên cao hơn và sẽ được thực hiện trước các phân chia BT ẩn, và Bảng 22 và Bảng 24 được áp dụng. Nếu các điều kiện được liệt kê trong các bảng không được thỏa mãn, phân chia OT được thực hiện.

Bảng 23 Các điều kiện để thực hiện phân chia QT hoặc BT ẩn khi BTFirst=1

	QT dọc theo các trục x-y	QT dọc theo các trục x-z	QT dọc theo các trục y-z
Điều kiện	$d_z = M < d_x = d_y$	$d_y = M < d_x = d_z$	$d_x = M < d_y = d_z$
	BT dọc theo trục x	BT along y axis	BT dọc theo trục z

Điều kiện	$d_y = M \leq d_z < d_x$	$d_x = M \leq d_z < d_y$	$d_x = M \leq d_y < d_z$
	$d_z = M \leq d_y < d_x$	$d_z = M \leq d_x < d_y$	$d_y = M \leq d_x < d_z$

Bảng 24 Các điều kiện để thực hiện phân chia QT hoặc BT ẩn khi BTFirst=0

	QT dọc theo các trục x-y	QT dọc theo các trục x-z	QT dọc theo các trục y-z
Điều kiện	$d_z = M < d_x \leq d_y$	$d_y = M < d_x \leq d_z$	$d_x = M < d_y \leq d_z$
	BT dọc theo trục x	BT along y axis	BT dọc theo trục z
Điều kiện	$d_y = d_z = M < d_x$	$d_x = d_z = M < d_y$	$d_x = d_y = M < d_z$

Đặt hộp giới hạn B có kích cỡ bằng $(2^{d_x}, 2^{d_y}, 2^{d_z})$. Không cần tổn hao chung, có thể giả định rằng $0 < d_x \leq d_y \leq d_z$.

Trong một phương án, K được cố định hoặc được chỉ rõ bởi gps_max_num_implicit_qtbt_before_ot, mà chỉ báo rằng K lần phân chia BT và QT ẩn sẽ được thực hiện đầu tiên. M được cố định hoặc được chỉ rõ bởi gps_min_size_implicit_qtbt, mà chỉ báo rằng kích cỡ nhỏ nhất của các phân chia QT và BT ẩn. BTFirst = 1 được cố định hoặc được chỉ rõ bởi gps_implicit_bt_before_implicit_qt_flag, mà chỉ báo rằng các phân chia BT ẩn có mức ưu tiên cao hơn và sẽ được thực hiện trước các phân chia QT ẩn, và Bảng 21 và Bảng 22 được áp dụng. Trong phương án này, tại K ($K \leq d_z - d_x$) độ sâu đầu tiên, các phân chia BT ẩn được thực hiện theo trục z và các phân chia QT ẩn sau

đó được thực hiện theo các trục y-z dựa trên Bảng 21. Sau đó, kích cỡ của các nút con trở thành $2^{(d_x,d_x+\delta_y,d_x+\delta_z)}$, trong đó giá trị của δ_y và δ_z ($\delta_z \geq \delta_y \geq 0$) phụ thuộc vào giá trị của K. Sau đó, các phân chia OT được thực hiện $d_x - M$ lần mà làm cho các nút con còn lại có kích cỡ bằng $2^{(M,M+\delta_y,M+\delta_z)}$. Tiếp theo, theo các bảng 23, các phân chia BT ẩn được thực hiện theo trục z $\delta_z - \delta_y$ lần, và các phân chia QT ẩn sau đó được thực hiện theo các trục y-z δ_y lần. Các nút còn lại với kích cỡ bằng $2^{(M,M,M)}$, do đó các phân chia OT được thực hiện M lần để chạm tới các đơn vị nhỏ nhất.

Trong phương án khác, K được cố định hoặc được chỉ rõ bởi gps_max_num_implicit_qtbt_before_ot, mà chỉ báo rằng K lần phân chia BT và QT ẩn sẽ được thực hiện đầu tiên. M được cố định hoặc được chỉ rõ bởi gps_min_size_implicit_qtbt, mà chỉ báo kích cỡ nhỏ nhất của các phân chia QT và BT ẩn. $BTFirst = 0$ được cố định hoặc được chỉ rõ bởi gps_implicit_bt_before_implicit_qt_flag, mà chỉ báo các phân chia QT ẩn có mức ưu tiên cao hơn và sẽ được thực hiện trước các phân chia BT ẩn, và Bảng 22 và Bảng 24 được áp dụng. Trong phương án này, tại K ($K \leq d_z - d_x$) độ sâu đầu tiên, các phân chia BT ẩn được thực hiện theo trục z và các phân chia QT ẩn sau đó được thực hiện theo các trục y-z dựa trên Bảng 22. Sau đó, kích cỡ của các nút con trở thành $2^{(d_x,d_x+\delta_y,d_x+\delta_z)}$, trong đó giá trị của δ_y và δ_z ($\delta_z \geq \delta_y \geq 0$) phụ thuộc vào giá trị của K. Sau đó, các phân chia OT được thực hiện $d_x - M$ lần mà làm cho các nút con còn lại có kích cỡ bằng $2^{(M,M+\delta_y,M+\delta_z)}$. Tiếp theo, theo Bảng 24, các phân chia QT ẩn được thực hiện theo các trục y-z δ_y lần, và các phân chia BT ẩn sau đó được thực hiện theo trục z $\delta_z - \delta_y$ lần. Các nút còn lại với kích cỡ bằng $2^{(M,M,M)}$, do đó các phân chia OT được thực hiện M lần để chạm tới các đơn vị nhỏ nhất.

5.5 Minh họa khối 2D đối với các phương pháp phân chia khác nhau của phân chia QT và BT

Bốn phương pháp phân chia nêu trên có thể được minh họa trong khối 2D như được thể hiện trong các Fig.10-13, một cách lần lượt, trong đó hộp giới hạn B chữ nhật 16×4 được phân chia bởi các phân chia lặp lại 4-mức tới đơn vị nhỏ nhất. Ngoài ra, phương pháp TMC13, trong đó các phân chia OT được thực hiện từ hộp giới hạn khối lập phương mở rộng, được minh họa trong Fig.10 đối với các trường hợp 2D. Trong minh họa 2D, BT tương tự với BT hoặc QT trong 3D, và QT tương tự với OT trong 3D.

Ví dụ về phương pháp của phần III.5.1

Trong FIG.10, BT được thực hiện sau QT, mà tương đương với việc thực hiện QT và BT sau OT trong 3D. Như được thể hiện, B đầu tiên được phân chia thành bốn khối con 4×1 bởi phân chia QT hai mức và sau đó, khối con còn lại được phân chia bởi phân chia BT hai mức theo trục x. Thứ tự phân chia là QT ẩn, QT, BT, BT.

Ví dụ về phương pháp của phần III.5.2

Trong FIG.11, BT được thực hiện trước QT, mà tương đương với việc thực hiện QT và BT trước OT trong 3D. Như được thể hiện, B đầu tiên được phân chia bởi các phân chia BT hai mức theo trục x để tạo thành bốn khối con 4×4, và sau đó, mỗi khối con được phân chia bởi các phân chia QT. Thứ tự phân chia là BT ẩn, BT, QT, QT.

Ví dụ về phương pháp của phần III.5.3

Trong FIG.12, việc phân chia được thực hiện bởi thứ tự BT, QT, BT, mà tương đương với phương pháp lai trong 3D. Phương pháp lai với K=1 được thể hiện trên FIG.12, trong đó B được phân chia bởi phân chia BT một lần để thu được hai khối con 8×4 và sua đó, mỗi khối con được phân chia bởi các phân chia QT hai mức để thu nhận các khối con nhỏ hơn 2×1 mà được chia cuối cùng thành đơn vị nhỏ nhất bởi phân chia BT khác theo trục x. Thứ tự phân chia là BT ẩn, QT, QT, BT.

Ví dụ về phương pháp của phần III.5.4

Trong FIG.13, việc phân chia được thực hiện bởi thứ tự BT, QT, BT, QT, mà tương đương với phương pháp lai với điều kiện kích cỡ nhỏ nhất trong 3D. Phương pháp lai với điều kiện kích cỡ BT nhỏ nhất được thể hiện trên FIG.13, trong đó K=M=1. Đầu tiên, phân chia BT được thực hiện trước phân chia QT do K=1, sau đó phân chia QT thực hiện cho đến khi chiều ngắn hơn đạt tới 2^M=2, tức là, các khối con 4×2. Tiếp theo, phân chia BT được yêu cầu do điều kiện kích cỡ BT nhỏ nhất, kết quả là, các khối con còn lại có kích cỡ 2×2, và cuối cùng phân chia QT được thực hiện. Thứ tự phân chia là BT ẩn, QT, BT, QT.

Ví dụ về thiết kế TMC13

FIG.14 thể hiện phương pháp QT từ hộp giới hạn được mở rộng trong 2D, mà tương đương với phương pháp phân chia OT của thiết kế TMC13 trong 3D. Như được thể hiện, các phân chia QT đệ quy bắt đầu từ hộp giới hạn chữ nhật 16×16, mà được thể hiện bởi các đường đứt nét trong hình vẽ.

Giả thiết rằng các điểm được bố trí trên các vị trí như được đánh dấu bởi các dấu gạch chéo trong FIG.10-14, có thể tính toán các bit được yêu cầu bởi mỗi phương pháp. Như được tóm tắt trong cột thứ nhất của Bảng 25, mà từ đó có thể thấy rằng bốn phương pháp có giá trị ít bit hơn so với phương pháp TMC13. Các phương pháp của phần III.5.3 và III.5.4 thực hiện tốt nhất. Cột thứ hai trong Bảng 25 tính toán các bit được yêu cầu tại trường hợp xấu nhất trong đó tất cả vị trí được chiếm giữ. Từ mô phỏng này, có thể thấy ít bit nhất từ phương pháp của phần III.5.2, và các phương pháp của phần III.5.2 và III.5.4 thực hiện tốt hơn so với phương pháp TMC13. Do đó, bằng cách lựa chọn thích hợp các tham số (K và M) và các điều kiện, các kết quả mã hóa hình học tốt có thể đạt được.

Bảng 25 So sánh các bit được yêu cầu bởi các phương pháp phân chia khác nhau

	Các bit được yêu cầu nếu chiếm giữ như được thể hiện trong các Fig.10-14	Các bit được yêu cầu tại trường hợp xấu nhất của chiếm giữ hoàn toàn

Phương pháp trong phần III.5.1	$1 \times 4 + 4 \times 4 + 8 \times 2 + 9 \times 2 = 54$	$1 \times 4 + 4 \times 4 + 16 \times 2 + 32 \times 2 = 116$
Phương pháp trong phần III.5.2	$1 \times 2 + 2 \times 2 + 4 \times 4 + 7 \times 4 = 50$	$1 \times 2 + 2 \times 2 + 4 \times 4 + 16 \times 4 = 86$
Phương pháp trong phần III.5.3	$1 \times 2 + 2 \times 4 + 5 \times 4 + 9 \times 2 = 48$	$1 \times 2 + 2 \times 4 + 8 \times 4 + 32 \times 2 = 106$
Phương pháp trong phần III.5.4	$1 \times 2 + 2 \times 4 + 5 \times 2 + 7 \times 4 = 48$	$1 \times 2 + 2 \times 4 + 8 \times 2 + 16 \times 4 = 90$
Phương pháp trong TMC13	$1 \times 4 + 2 \times 4 + 4 \times 4 + 7 \times 4 = 56$	$1 \times 4 + 2 \times 4 + 4 \times 4 + 16 \times 4 = 92$

IV. Tương tác với chế độ phẳng

1. Chế độ phẳng

Trong một vài phương án, các chế độ phẳng trong ba chiều x, y, và z được đưa vào để mã hóa hình học đám mây điểm. Các chế độ phẳng có thể được kích hoạt tại mức nút khả dụng thông qua cờ kích hoạt chế độ phẳng được mã hóa trong dòng bit. Ngoài ra, cú pháp bổ sung được thêm vào dòng bit để chỉ báo vị trí của mặt phẳng được kết hợp với các chế độ phẳng được kích hoạt.

Dự đoán của cờ và vị trí mặt phẳng cũng được đưa vào để đảm bảo việc nén tốt của phần tử cú pháp mới. Cuối cùng, tiêu chuẩn khả dụng cục bộ có thể được sử dụng để tránh sử dụng các chế độ phẳng trong các vùng đối diện

của đám mây điểm và nhờ đó tránh được hiệu năng nén kém, cụ thể trên các đám mây điểm dày.

2. Các vấn đề

Phân chia QT/BT ẩn và các chế độ phẳng được dựa trên các xem xét khác nhau và có cùng tăng ích mã hóa trên các đám mây điểm thưa. Nói chung, phân chia hình học ẩn với độ phức tạp thấp và chế độ phẳng là phức tạp hơn nhưng thích nghi hơn. Liên quan đến hiệu năng, hai kỹ thuật mã hóa đạt được cùng hiệu năng trên dữ liệu thưa, và chế độ phẳng có thể xử các đám mây điểm với nhiều mẫu mặt phẳng tốt hơn.

Đầu tiên, hai kỹ thuật mã hóa chia sẻ một vài khái niệm và phương pháp mã hóa tương tự trong mã hóa chiếm giữ. Hai kỹ thuật mã hóa đều thay đổi phương pháp mã hóa chiếm giữ, trong đó tại các điều kiện định trước chỉ một phần của mã chiếm giữ được mã hóa trong khi phần còn lại của mã chiếm giữ có thể được bỏ qua và được xem xét tại phía bộ giải mã. Đối với phân chia hình học ẩn, điều kiện được xác định ẩn bởi một vài tham số định trước. Trong khi đối với chế độ phẳng, điều kiện được xác định bởi bộ mã hóa và cờ và chỉ số được báo hiệu rõ ràng để chỉ báo phần nào của mã chiếm giữ được bỏ qua.

Ngoài ra, việc phân chia hình học ẩn có thể bắt đầu phân chia cây bát phân từ hộp giới hạn không phải khối lập phương và dạng của nút cây bát phân có thể cũng là bất đối xứng, trong khi chế độ phẳng luôn giả định các dạng đối xứng của các nút cây bát phân.

Hai kỹ thuật này không xung đột một cách trực tiếp về khái niệm. Chúng có các điểm tương tự/chồng lấn và cũng như các khác biệt. Trong các phương án khác nhau, hai kỹ thuật có thể được hợp nhất cùng nhau.

Trong các phần dưới đây, các phương án kết hợp việc phân chia hình học ẩn và chế độ phẳng được mô tả. Trong phần mô tả sau đây, một vài phương án được thể hiện là các thay đổi so với các tiêu chuẩn kỹ thuật của phác thảo đối với nén đám mây điểm dựa trên hình học, ISO/IEC 23090-9:2019(E), giai đoạn CD, ISO/IEC JTC 1/SC 29/WG 11 W18478, tháng 6 năm 2019.

3. Chỉ phân chia hình học ẩn hoặc chỉ chế độ phẳng

Trong một vài phương án, một trong số hai kỹ thuật được kích hoạt khi mã hóa dạng hình học của đám mây điểm, tức là, kích hoạt phân chia hình học ẩn (trong khi ngắt chế độ phẳng) hoặc kích hoạt chế độ phẳng (trong khi ngắt phân chia hình học ẩn).

3.1 Báo hiệu cờ điều khiển trong cú pháp mức cao đối với phân chia hình học ẩn

Để kích hoạt và ngắt phân chia hình học ẩn, một cờ có thể được báo hiệu trong cú pháp mức cao. Cờ có thể được chỉ rõ trong tập hợp tham số chuỗi hoặc thông tin tiêu đề lát hoặc tập hợp tham số hình học của dòng bit.

Phương án A

Trong một phương án, cờ của phân chia hình học ẩn được chỉ rõ trong tập hợp tham số hình học như sau.

Bảng 26

#	geometry_parameter_set( ) {	Mô tả
1	gps_geom_parameter_set_id	ue(v)
2	gps_seq_parameter_set_id	ue(v)
3	gps_box_present_flag	u(1)
4	if(gps_box_present_flag){	
5	gps_gsh_box_log2_scale_present_flag	u(1)
6	if(gps_gsh_box_log2_scale_present_flag == 0)	
7	gps_gsh_box_log2_scale	ue(v)

8	}	
9	unique_geometry_points_flag	u(1)
10	neighbour_context_restriction_flag	u(1)
11	inferred_direct_coding_mode_enabled_flag	u(1)
12	bitwise_occupancy_coding_flag	u(1)
13	adjacent_child_contextualization_enabled_flag	u(1)
14	log2_neighbour_avail_boundary	ue(v)
15	log2_intra_pred_max_node_size	ue(v)
16	log2_trisoup_node_size	ue(v)
17*	gps_implicit_geom_partition_flag	u(1)
18	gps_extension_present_flag	u(1)
19	if(gps_extension_present_flag)	
20	while(more_data_in_byte_stream( ))	
21	gps_extension_data_flag	u(1)
22	byte_alignment( )	
23	}	

Cú pháp tập hợp tham số hình học trong Bảng 26 được cải biến bằng cách cộng phần tử cú pháp tại hàng 17.

gps_implicit_geom_partition_flag bằng 1 chỉ rõ rằng việc phân chia hình học ẩn được kích hoạt đối với chuỗi hoặc lát.

gps_implicit_geom_partition_flag bằng 0 chỉ rõ rằng việc phân chia hình học ẩn được ngắt đối với chuỗi hoặc lát.

Phương án B

Trong phương án khác, cờ được chỉ rõ khi phương án hỗn hợp tam giác phi kết nối (Trisoup) được ngắt, trong đó trường hợp thay đổi đối với tập hợp tham số hình học được thể hiện tại các hàng 6-8 trong Bảng 27.

Bảng 27

#	geometry_parameter_set( ) {	Mô tả
1	gps_geom_parameter_set_id	ue(v)
2	gps_seq_parameter_set_id	ue(v)
3	…	…
4	log2_intra_pred_max_node_size	ue(v)
5	log2_trisoup_node_size	ue(v)
6 *	if (log2_trisoup_node_size== 0) {	
7 *	gps_implicit_geom_partition_flag	u(1)
8 *	}	
9	…	…
10	byte_alignment( )	
11	}	

khi log2_trisoup_node_size lớn hơn 0, gps_implicit_geom_partition_flag có thể được xem là 0 mà không báo hiệu rõ ràng.

Phương án C

Trong phương án khác, ngoài cờ, các tham số khác mà liên quan đến phân chia hình học ẩn có thể được chỉ rõ trong cú pháp mức cao khi cờ gps_implicit_geom_partition_flag bằng 1. Cú pháp tập hợp tham số hình học trong Bảng 28 được cải biến tại các hàng 6-10.

Bảng 28

#	geometry_parameter_set( ) {	Mô tả
1	gps_geom_parameter_set_id	ue(v)
2	gps_seq_parameter_set_id	ue(v)
3	…	…
4	log2_intra_pred_max_node_size	ue(v)
5	log2_trisoup_node_size	ue(v)
6 *	gps_implicit_geom_partition_flag	u(1)
7 *	if (gps_implicit_geom_partition_flag) {	
8 *	gps_max_num_implicit_qtbt_before_ot	ue(v)
9 *	gps_min_size_implicit_qtbt	ue(v)
10*	}	
11	…	…
12	byte_alignment( )	
13	}	

gps_max_num_implicit_qtbt_before_ot chỉ rõ số lượng lớn nhất của các phân chia QT và BT ẩn trước các phân chia OT.

gps_min_size_implicit_qtbt chỉ rõ kích cỡ nhỏ nhất của các phân chia QT và BT ẩn.

Phương án D

Trong phương án khác, báo hiệu của hộp giới hạn của dạng hình học có thể phụ thuộc vào giá trị của gps_implicit_geom_partition_flag như sau. Cú pháp thông tin tiêu đề lát hình học trong Bảng 29 được thay đổi bằng cách bổ sung cấu trúc cú pháp tại các hàng 10-16.

Bảng 29

#	geometry_slice_header( ) {	Mô tả
1	gsh_geometry_parameter_set_id	ue(v)
2	gsh_tile_id	ue(v)
3	gsh_slice_id	ue(v)
4	if(gps_box_present_flag) {	
5	gsh_box_log2_scale	ue(v)
6	gsh_box_origin_x	ue(v)
7	gsh_box_origin_y	ue(v)
8	gsh_box_origin_z	ue(v)
9	}	
10*	if (gps_implicit_geom_partition_flag) {	
11*	gsh_log2_max_nodesize_x	ue(v)
12*	gsh_log2_max_nodesize_y	ue(v)

13*	gsh_log2_max_nodesize_z	ue(v)
14*	} else {	
15*	gsh_log2_max_nodesize	ue(v)
16*	}	
17	gsh_num_points	ue(v)
18	byte_alignment( )	
19	}	

gsh_log2_max_nodesize_x, gsh_log2_max_nodesize_y, và gsh_log2_max_nodesize_z chỉ rõ kích cỡ hộp giới hạn của các chiều x, y, z trong tỷ lệ log2, một cách lần lượt. Chúng được chỉ rõ chỉ khi gps_implicit_geom_partition_flag bằng 1. Khi gps_implicit_geom_partition_flag bằng 0, chỉ một kích cỡ được chỉ rõ bởi gsh_log2_max_nodesize, và trong trường hợp này, ba chiều được giả định có cùng kích cỡ.

Phương án E

Trong phương án khác, các kích cỡ hộp giới hạn được chỉ rõ bởi các độ chênh lệch của chúng. Cú pháp thông tin tiêu đề lát hình học trong Bảng 30 được cải biến tại các hàng 10-16.

Bảng 30

#	geometry_slice_header( ) {	Mô tả
1	gsh_geometry_parameter_set_id	ue(v)
2	gsh_tile_id	ue(v)
3	gsh_slice_id	ue(v)

4	if(gps_box_present_flag) {	
5	gsh_box_log2_scale	ue(v)
6	gsh_box_origin_x	ue(v)
7	gsh_box_origin_y	ue(v)
8	gsh_box_origin_z	ue(v)
9	}	
10*	if (gps_implicit_geom_partition_flag) {	
11*	gsh_log2_max_nodesize_x	ue(v)
12*	gsh_log2_max_nodesize_y_minus_x	se(v)
13*	gsh_log2_max_nodesize_z_minus_y	se(v)
14*	} else {	
15*	gsh_log2_max_nodesize	ue(v)
16*	}	
17	gsh_num_points	ue(v)
18	byte_alignment( )	
19	}	

3.2 Báo hiệu các cờ điều khiển trong cú pháp mức cao xem xét phương pháp kết hợp

Trong phần phụ này, một vài các phương án được mô tả như là các ví dụ để thể hiện làm thế nào để báo hiệu các cờ điều khiển trong cú pháp mức cao khi xem xét kết hợp của việc phân chia hình học ẩn và chế độ phẳng.

Phương án A

Trong một phương án, hai cờ điều khiển được chỉ rõ một cách độc lập trong tập hợp tham số hình học như sau. Cú pháp tập hợp tham số hình học trong Bảng 31 được cải biến tại các hàng 6-7.

Bảng 31

#	geometry_parameter_set( ) {	Mô tả
1	gps_geom_parameter_set_id	ue(v)
2	gps_seq_parameter_set_id	ue(v)
3	…	…
4	log2_intra_pred_max_node_size	ue(v)
5	log2_trisoup_node_size	ue(v)
6 *	gps_implicit_geom_partition_flag	u(1)
7 *	gps_planar_mode_flag	u(1)
8	…	…
9	byte_alignment( )	
10	}	

gps_planar_mode_flag bằng 1 chỉ rõ rằng chế độ phẳng được kích hoạt đối với chuỗi hoặc lát. gps_planar_mode_flag bằng 0 chỉ rõ rằng chế độ phẳng được ngắt đối với chuỗi hoặc lát.

Tuy nhiên, trong phương pháp kết hợp này, giả thiết rằng chỉ một trong số hai phương pháp này được kích hoạt. Ví dụ, gps_implicit_geom_partition_flag và gps_planar_mode_flag không thể cùng bằng 1.

Phương án B

Trong phương án khác, cờ điều khiển gps_planar_mode_flag được chỉ rõ phụ thuộc vào giá trị của gps_implicit_geom_partition_flag trong tập hợp tham số hình học như sau. Cú pháp tập hợp tham số hình học trong Bảng 32 được cải biến tại các hàng 6-8.

Bảng 32

#	geometry_parameter_set( ) {	Mô tả
1	gps_geom_parameter_set_id	ue(v)
2	gps_seq_parameter_set_id	ue(v)
3	…	…
4	log2_intra_pred_max_node_size	ue(v)
5	log2_trisoup_node_size	ue(v)
6 *	gps_implicit_geom_partition_flag	u(1)
7 *	if (gps_implicit_geom_partition_flag == 0)	
8 *	gps_planar_mode_flag	u(1)
9	…	…
10	byte_alignment( )	
11	}	

Trong trường hợp này, gps_planar_mode_flag được chỉ rõ chỉ khi gps_implicit_geom_partition_flag bằng 0. Khi gps_implicit_geom_partition_flag bằng 1, gps_planar_mode_flag có thể được xem là 0.

Phương án C

Trong phương án khác, cờ điều khiển gps_implicit_geom_partition_flag được chỉ rõ phụ thuộc vào giá trị của gps_planar_mode_flag trong tập hợp tham số hình học như sau. Cú pháp tập hợp tham số hình học trong Bảng 33 được cải biến tại các hàng 6-8.

Bảng 33

#	geometry_parameter_set( ) {	Mô tả
1	gps_geom_parameter_set_id	ue(v)
2	gps_seq_parameter_set_id	ue(v)
3	…	…
4	log2_intra_pred_max_node_size	ue(v)
5	log2_trisoup_node_size	ue(v)
6 *	gps_planar_mode_flag	u(1)
7 *	if (gps_planar_mode_flag == 0)	
8 *	gps_implicit_geom_partition_flag	u(1)
9	…	…
10	byte_alignment( )	
11	}	

Trong trường hợp này, gps_implicit_geom_partition_flag được chỉ rõ chỉ khi gps_planar_mode_flag bằng 0. Khi gps_planar_mode_flag bằng 1, gps_implicit_geom_partition_flag có thể được xem là 0.

Phương án D

Trong phương án khác, cờ điều khiển gps_implicit_geom_partition_flag được chỉ rõ phụ thuộc vào giá trị của

gps_planar_mode_flag và giá trị của log2_trisoup_node_size trong tập hợp tham số hình học như sau. Cú pháp tập hợp tham số hình học trong Bảng 34 được cải biến tại các hàng 6-8.

Bảng 34

#	geometry_parameter_set( ) {	Mô tả
1	gps_geom_parameter_set_id	ue(v)
2	gps_seq_parameter_set_id	ue(v)
3	…	…
4	log2_intra_pred_max_node_size	ue(v)
5	log2_trisoup_node_size	ue(v)
6 *	gps_planar_mode_flag	u(1)
7 *	if (gps_planar_mode_flag == 0 && log2_trisoup_node_size == 0)	
8 *	gps_implicit_geom_partition_flag	u(1)
9	…	…
10	byte_alignment( )	
11	}	

Trong trường hợp này, gps_implicit_geom_partition_flag được chỉ rõ chỉ khi cả gps_planar_mode_flag và log2_trisoup_node_size bằng 0. Nếu không phải, gps_implicit_geom_partition_flag có thể được xem là 0. Nói cách khác, phân chia hình học ẩn có thể được kích hoạt chỉ khi cả chế độ hỗn hợp tam giác phi kết nối (Trisoup) và chế độ phẳng được ngắt.

4. Chế độ phẳng là khả dụng chỉ khi nút cây bát phân hiện tại là khối lập phương

Phương pháp kết hợp thứ hai có thể cho phép phân chia hình học ẩn và chế độ phẳng tại cùng thời điểm. Nhưng một vài ràng buộc được bao gồm đối với chế độ phẳng. Nếu phân chia hình học ẩn được kích hoạt, hộp giới hạn và các nút cây bát phân có thể không phải khối lập phương. Trong phương pháp này, điều kiện để áp dụng chế độ phẳng đó là chế độ phẳng là khả dụng chỉ khi nút cây bát phân hiện tại là khối lập phương, mà chỉ báo $d_x = d_y = d_z$.

Trong một phương án, cú pháp mã hóa hình học được thể hiện trong Bảng 35, mà có thể tương tự như trong Bảng 9.

Bảng 35

#	geometry_slice_data( ) {	Mô tả
12	for(depth = 0; depth < MaxGeometryOctreeDepth; depth++) {	
13	for(nodeIdx = 0; nodeIdx < NumNodesAtDepth[depth]; nodeIdx++) {	
14*	/* NB: implicit partition is described in semantics */	
15*	partitionSkip = $b_x b_y b_z$ // depend on implicit partition conditions	
16*	If (!(partitionSkip & 4))	
17*	depthX = depthX + 1;	
18*	If (!(partitionSkip & 2))	

19*	depthY = depthY + 1;	
20*	If (!(partitionSkip & 1))	
21*	depthZ = depthZ + 1;	
22*	xN = NodeX[depthX][nodeIdx]	
23*	yN = NodeY[depthY][nodeIdx]	
24*	zN = NodeZ[depthZ][nodeIdx]	
25*	geometry_node(depthX, depthY, depthZ, partitionSkip, nodeIdx, xN, yN, zN)	
26	}	
27	}	
28	if (log2_trisoup_node_size > 0)	
29	geometry_trisoup_data( )	
30	}	

Các biến ChildNodeSizeXLog2, ChildNodeSizeYLog2 và ChildNodeSizeZLog2 chỉ rõ kích cỡ nút con đối với mỗi chiều, và có thể được xác định bởi các phân chia QT và BT ẩn như sau,

NodeSizeXLog2 = MaxNodeSizeXLog2 – depthX;

NodeSizeYLog2 = MaxNodeSizeYLog2 – depthY;

NodeSizeZLog2 = MaxNodeSizeZLog2 – depthZ;

if(!(partitionSkip & 4))

ChildNodeSizeXLog2 = NodeSizeXLog2 – 1;

else

```
                ChildNodeSizeXLog2 = NodeSizeXLog2;
        if( !(partitionSkip & 2 )
                ChildNodeSizeYLog2 = NodeSizeYLog2 – 1;
        else
                ChildNodeSizeYLog2 = NodeSizeYLog2;
        if( !(partitionSkip & 1 )
                ChildNodeSizeZLog2 = NodeSizeZLog2 – 1;
        else
                ChildNodeSizeZLog2 = NodeSizeZLog2.
```

Trong phương án này, chế độ phẳng là không khả dụng khi NodeSizeLog2X, NodeSizeLog2Y và NodeSizeLog2Z không bằng nhau như sau,

```
if ((NodeSizeXLog2 != NodeSizeYLog2) || (NodeSizeXLog2 !=
NodeSizeZLog2) || (NodeSizeYLog2!= NodeSizeZLog2)) {
    planarModeEligibilityX = 0;
        planarModeEligibilityY = 0;
    planarModeEligibilityZ = 0;
}
```

trong đó planarModeEligibilityX, planarModeEligibilityY và planarModeEligibilityZ chỉ rõ rằng chế độ phẳng có khả dụng đối với các chiều X, Y, Z hay không, một cách lần lượt, trong nút được mã hóa hiện tại.

5. Chế độ phẳng không khả dụng tại các chiều định trước mà không được phân chia

Phương pháp kết hợp thứ ba còn làm giảm ràng buộc của chế độ phẳng, trong đó chế độ phẳng là khả dụng ngay cả khi nút cây bát phân hiện tại không phải khối lập phương. Nếu một hoặc hai chiều không được phân chia tại một độ sâu, chế độ phẳng là không khả dụng tại các chiều này trong khi là khả

dụng tại các chiều con lại. Ví dụ, nếu phân chia hình học ẩn quyết định thực hiện việc phân chia theo các chiều x và y mà không phải trục z tại độ sâu phân chia định trước, thì chế độ phẳng chỉ khả dụng đối với các chiều x và y.

Trong phương án này, chế độ phẳng không khả dụng tại các chiều nhất định trong đó việc phân chia được bỏ qua bởi phương pháp phân chia hình học ẩn như sau,

```
if( partitionSkip & 4 )
    planarModeEligibilityX = 0;
if( partitionSkip & 2 )
    planarModeEligibilityY = 0;
if( partitionSkip & 1 )
    planarModeEligibilityZ = 0.
```

Lưu ý rằng trong các ví dụ khác, chế độ phẳng có thể được thay đổi trong các khía cạnh khác và trong cách thức phức tạp hơn để căn thẳng với thực tế rằng nút cây bát phân có thể là khối phỏng lập phương chữ nhật.

V. Các phương án bổ sung của phân chia hình học ẩn

Phương án A

Phương án A đề xuất cải tiến đối với phân chia hình học ẩn, và tiết kiệm nhiều bit hơn khi mã hóa mã chiếm giữ với phân chia hình học ẩn. Cụ thể, xử lý phân tích chiếm giữ cây bát phân hình học được mô tả trong phần III có thể được cải biến bằng cách đưa ra xử lý để xác định biến binIsInferred tại phần cuối của xử lý phân tích chiếm giữ cây bát phân hình học:

Xử lý này khôi phục phần tử cú pháp occupancy_map.

Đầu vào đối với xử lý này là NeighbourPattern, binIsSkiped và binIsInferred của nút hiện tại.

Đầu ra từ xử lý này là giá trị phần tử cú pháp, được xây dựng như sau:

```
value = 0;
for (binIdx = 0; binIdx < 8; binIdx++) {
  if( binIsSkiped[binIdx] )
    bin = 0;
  else if( binIsInferred )
    bin = 1;
  else
    bin = readBin(binIdx)
  value = value | (bin << bitCodingOrder[ binIdx ]);
}
```

trong đó biến binIsSkiped[binIdx] được thiết lập theo như sau

```
if( (partitionSkip & 1) && (
    inverseMap[NeighbourPattern][bitCodingOrder[binIdx]] & 1 ) )
  binIsSkiped[binIdx]=1;
else if( (partitionSkip & 2) && (
    inverseMap[NeighbourPattern][bitCodingOrder[binIdx]] & 2) )
  binIsSkiped[binIdx]=1;
else if( (partitionSkip & 4) && (
    inverseMap[NeighbourPattern][bitCodingOrder[binIdx]] & 4 ) )
  binIsSkiped[binIdx]=1;
```

nếu không phải

```
  binIsSkiped[binIdx]=0;
```

và trong đó, đối với mỗi ký tự nhị phân, biến binIsInferred được thiết lập theo như sau:

– nếu một trong các điều kiện sau đây là đúng, binIsInferred được thiết lập bằng 1:

- NeighbourPattern bằng 0 và số lượng của 1-valued ký tự nhị phân được giải mã trước đó nhỏ hơn hoặc bằng (binIdx + minOccupied – maxOccupied).
- NeighbourPattern không bằng bằng 0, binIdx bằng maxOccupied-1 và các giá trị của tất cả ký tự nhị phần được giải mã trước đó bằng 0.

trong đó minOccupied=2, và maxOccupied=8 (if OT được áp dụng), maxOccupied=4 (if QT được áp dụng), maxOccupied=2 (nếu BT được áp dụng).

– Nếu không phải, nếu các điều kiện nêu không là đúng, binIsInferred được thiết lập bằng 0.

Phương án B

Trong một phương án, cờ điều khiển gps_implicit_geom_partition_flag và hai tham số K và M được chỉ rõ tại tập hợp tham số hình học như sau,

Bảng 36

geometry_parameter_set( ) {	Mô tả
gps_geom_parameter_set_id	ue(v)
gps_seq_parameter_set_id	ue(v)
gps_box_present_flag	u(1)
if(gps_box_present_flag){	
gps_gsh_box_log2_scale_present_flag	u(1)

if(gps_gsh_box_log2_scale_present_flag == 0)	
gps_gsh_box_log2_scale	ue(v)
}	
unique_geometry_points_flag	u(1)
neighbour_context_restriction_flag	u(1)
inferred_direct_coding_mode_enabled_flag	u(1)
bitwise_occupancy_coding_flag	u(1)
adjacent_child_contextualization_enabled_flag	u(1)
log2_neighbour_avail_boundary	ue(v)
log2_intra_pred_max_node_size	ue(v)
log2_trisoup_node_size	ue(v)
gps_implicit_geom_partition_flag	u(1)
if (gps_implicit_geom_partition_flag) {	
gps_max_num_implicit_qtbt_before_ot	ue(v)
gps_min_size_implicit_qtbt	ue(v)
}	
gps_extension_present_flag	u(1)
if(gps_extension_present_flag)	
while(more_data_in_byte_stream( ))	

gps_extension_data_flag	u(1)
byte_alignment( )	
}	

gps_implicit_geom_partition_flag bằng 1 chỉ rõ rằng việc phân chia hình học ẩn được kích hoạt đối với chuỗi hoặc lát. gps_implicit_geom_partition_flag bằng 0 chỉ rõ rằng việc phân chia hình học ẩn được ngắt đối với chuỗi hoặc lát. Nếu gps_implicit_geom_partition_flag bằng 1, hai tham số sau đây được báo hiệu:

(1) gps_max_num_implicit_qtbt_before_ot chỉ rõ số lượng lớn nhất của các phân chia QT và BT ẩn trước các phân chia OT, tức là, K = gps_max_num_implicit_qtbt_before_ot.

(2) gps_min_size_implicit_qtbt chỉ rõ kích cỡ nhỏ nhất của các phân chia QT và BT ẩn, tức là, M = gps_min_size_implicit_qtbt. Tham số M này ngăn chặn các phân chia QT và BT ẩn khi tất cả các chiều nhỏ hơn hoặc bằng M.

Phương án C

Trong một phương án, nếu QTBT ẩn được kích hoạt, kích cỡ của hộp giới hạn được chỉ rõ bởi ba giá trị trong thông tin tiêu đề lát hình học như sau,

geometry_slice_header( ) {	Mô tả
gsh_geometry_parameter_set_id	ue(v)
gsh_tile_id	ue(v)
gsh_slice_id	ue(v)
if(gps_box_present_flag) {	
gsh_box_log2_scale	ue(v)

gsh_box_origin_x	ue(v)
gsh_box_origin_y	ue(v)
gsh_box_origin_z	ue(v)
}	
if (gps_implicit_geom_partition_flag) {	
gsh_log2_max_nodesize_x	ue(v)
gsh_log2_max_nodesize_y_minus_x	se(v)
gsh_log2_max_nodesize_z_minus_y	se(v)
} else {	
gsh_log2_max_nodesize	ue(v)
}	
gsh_num_points	ue(v)
byte_alignment( )	
}	

gsh_log2_max_nodesize_x chỉ rõ kích cỡ hộp giới hạn trong chiều x, tức là, MaxNodesizeXLog2 mà được sử dụng trong xử lý giải mã như sau:

MaxNodeSizeXLog2 = gsh_log2_max_nodesize_x.

MaxNodeSizeX = 1 << MaxNodeSizeXLog2.

gsh_log2_max_nodesize_y_minus_x chỉ rõ kích cỡ hộp giới hạn trong chiều y, tức là, MaxNodesizeYLog2 mà được sử dụng trong xử lý giải mã như sau:

MaxNodeSizeYLog2 = gsh_log2_max_nodesize_y_minus_x + MaxNodeSizeXLog2.

MaxNodeSizeY = 1 << MaxNodeSizeYLog2.

gsh_log2_max_nodesize_z_minus_y chỉ rõ kích cỡ hộp giới hạn trong chiều z, tức là, MaxNodesizeZLog2 mà được sử dụng trong xử lý giải mã như sau:

MaxNodeSizeZLog2 = gsh_log2_max_nodesize_z_minus_y + MaxNodeSizeYLog2.

MaxNodeSizeZ = 1 << MaxNodeSizeZLog2.

Sau đó, các tham số K và M được cập nhật như sau,

gsh_log2_max_nodesize = max{ MaxNodeSizeXLog2, MaxNodeSizeYLog2, MaxNodeSizeZLog2}

gsh_log2_min_nodesize = min{ MaxNodeSizeXLog2, MaxNodeSizeYLog2, MaxNodeSizeZLog2}

```
if (K > (gsh_log2_max_nodesize - gsh_log2_min_nodesize))
    K = gsh_log2_max_nodesize - gsh_log2_min_nodesize;
if (M > gsh_log2_min_nodesize)
    M = gsh_log2_min_nodesize;
if (gsh_log2_max_nodesize == gsh_log2_min_nodesize)
    M = 0;
if (log2_trisoup_node_size != 0) {
    K = gsh_log2_max_nodesize - gsh_log2_min_nodesize;
    M = 0;
}
```

Theo phương án của sáng chế, khi chế độ hỗn hợp tam giác phi kết nối (trisoup) được kích hoạt (log2_trisoup_node_size != 0 là đúng), M được thay đổi thành

chiều log2 nút gốc nhỏ nhất của lát. Lưu ý rằng chiều log2 hỗn hợp tam giác phi kết nối cần không lớn hơn chiều log2 nút gốc nhỏ nhất của lát.

Phương án D

Theo phương án của sáng chế, cú pháp dữ liệu lát hình học trong Bảng 37 được sử dụng trong vị trí của cấu trúc cú pháp trong Bảng 35 hoặc Bảng 9. Hàm quyết định QT/BT ẩn được đưa vào mới tại hàng 3 của Bảng 37. Cú pháp của hàm quyết định QT/BT ẩn được thể hiện trong Bảng 38 trong đó biến partitionSkip được xác định tại các hàng 9-16, và các độ sâu trong các chiều x, y, và z được cập nhật tại các hàng 17-23.

Bảng 37

#	geometry_slice_data( ) {	Mô tả
1 *	depthX = 0; depthY = 0; depthZ = 0;	
2	for(depth = 0; depth < MaxGeometryOctreeDepth; depth++) {	
3 *	partitionSkip = implicit_qtbt_decision(K, M, depth, depthX, depthY, depthZ)	
4	for(nodeIdx = 0; nodeIdx < NumNodesAtDepth[depth]; nodeIdx++) {	
5	xN = NodeX[depthX][nodeIdx]	
6	yN = NodeY[depthY][nodeIdx]	
7	zN = NodeZ[depthZ][nodeIdx]	
8	geometry_node(depthX, depthY, depthZ, partitionSkip, nodeIdx, xN, yN, zN)	

9	}	
10	}	
11	if (log2_trisoup_node_size > 0)	
12	geometry_trisoup_data( )	
13	}	

Bảng 38

#	implicit_qtbt_decision (K, M, depth, depthX, depthY, depthZ) {	Mô tả
1	partitionSkip = 0;	
2	NodeSizeXLog2 = MaxNodeSizeXLog2 - depthX;	
3	NodeSizeYLog2 = MaxNodeSizeYLog2 – depthY;	
4	NodeSizeZLog2 = MaxNodeSizeZLog2 - depthZ;	
5	MinNodeSizeLog2 = min{ NodeSizeXLog2, NodeSizeYLog2, NodeSizeZLog2};	
6	MaxNodeSizeLog2 = max{ NodeSizeXLog2, NodeSizeYLog2, NodeSizeZLog2};	
7	If (MinNodeSizeLog2 == MaxNodeSizeLog2)	
8	M = 0;	
9 *	if (K > depth || M == MinNodeSizeLog2) {	
10*	if (NodeSizeXLog2 < MaxNodeSizeLog2)	

11*	partitionSkip |= 4;	
12*	if (NodeSizeYLog2 < MaxNodeSizeLog2)	
13*	partitionSkip |= 2;	
14*	if (NodeSizeZLog2 < MaxNodeSizeLog2)	
15*	partitionSkip |= 1;	
16*	}	
17*	if (!(partitionSkip & 4))	
18*	depthX = depthX + 1;	
19*	if (!(partitionSkip & 2))	
20*	depthY = depthY + 1;	
21*	if (!(partitionSkip & 1))	
22*	depthZ = depthZ + 1;	
23*	}	

VI. Các ví dụ xử lý giải mã hình học

Fig.15 thể hiện lưu đồ mô tả xử lý 1500 theo phương án của sáng chế. Ví dụ, xử lý 1500 có thể được sử dụng để giải mã đám mây điểm. Xử lý 1500 có thể được sử dụng to tạo ra cấu trúc cây bát phân để biểu diễn hình học của các điểm trong hộp giới hạn của lát tại bộ giải mã đám mây điểm. Trong các phương án khác nhau, xử lý 1500 có thể được thực hiện bởi hệ mạch xử lý, như hệ mạch xử lý mà thực hiện các chức năng của môđun giải mã số học 410 và môđun giải mã cây bát phân 430 tại bộ giải mã 400. Trong một vài phương án, xử lý 1500 được thực hiện trong các lệnh phần mềm, nhờ đó khi hệ mạch xử lý

thực thi các lệnh phần mềm, hệ mạch xử lý thực hiện xử lý 1500. Xử lý bắt đầu tại bước 1501 và chuyển sang bước 1510.

Tại bước 1510, dòng bit bao gồm lát của khung đám mây điểm được mã hóa có thể được thu nhận. Ví dụ, lát có thể bao gồm một loạt các phần tử cú pháp và các ký tự nhị phân mà biểu diễn một phần hoặc toàn bộ khung đám mây điểm được mã hóa.

Tại 1520, cây bát phân biểu diễn hình học của các điểm trong hộp giới hạn của lát có thể được khôi phục. Trong xử lý xây dựng cây bát phân, nút hiện tại của cây bát phân có thể được phân chia với phân chia cây tứ phân (QT), hoặc phân chia cây nhị phân (BT). Ví dụ, làm thế nào để phân chia nút hiện tại của cây bát phân nhờ sử dụng một trong số phân chia QT, phân chia BT hoặc phân chia cây bát phân (OT) có thể được xác định dựa trên các điều kiện định trước. Việc làm thế nào để phân chia nút hiện tại của cây bát phân nhờ sử dụng một trong số phân chia QT, phân chia BT hoặc phân chia OT có thể cũng được xác định dựa trên một hoặc nhiều tham số mà mỗi tham số có thể được báo hiệu trong dòng bit, hoặc sử dụng giá trị được cấu hình trước cục bộ.

Trong khi xây dựng cây bát phân, giá trị của biến, được ký hiệu bởi partitionSkip, mà chỉ rõ loại phân chia và chiều phân chia của nút hiện tại của cây bát phân có thể được xác định. Trong ví dụ của sáng chế, biến partitionSkip có thể được biểu diễn trong dạng nhị phân với ba bit tương ứng với các chiều x, y, và z, một cách lần lượt, và mỗi bit chỉ báo rằng việc có được thực hiện dọc theo chiều x, y, hoặc z tương ứng hay không. Dựa trên biến partitionSkip, độ sâu trong chiều x, y, hoặc z đối với nút con của nút hiện tại của cây bát phân có thể được cập nhật.

Trong khi xây dựng của cây bát phân, các bit chiếm giữ thuộc về mã chiếm giữ 8 ký tự nhị phân của nút hiện tại của cây bát phân có thể được thu nhận (hoặc được phân tích) từ dòng bit. Mỗi bit chiếm giữ tương ứng với nút con được chiếm giữ của nút hiện tại của cây bát phân. 4 ký tự nhị phân thuộc về mã chiếm giữ 8 ký tự nhị phân không được báo hiệu trong dòng bit khi nút hiện tại

của cây bát phân được phân chia với phân chia QT, và 6 ký tự nhị phân thuộc về mã chiếm giữ 8 ký tự nhị phân không được báo hiệu trong dòng bit khi nút hiện tại của cây bát phân được phân chia với phân chia BT.

Ngoài ra, một hoặc nhiều phần tử cú pháp mà chỉ báo các kích cỡ ba-chiều (3D) của hộp giới hạn của lát của khung đám mây điểm được mã hóa có thể được thu từ dòng bit. Ví dụ, hộp giới hạn của lát có thể có dạng của khối phỏng lập phương chữ nhật.

Trong khi xây dựng cây bát phân, sau khi thu phần tử cú pháp mà chỉ báo nút hiện tại của cây bát phân có một nút con được chiếm giữ, 1 ký tự nhị phân có thể được thu nhận nếu biến partitionSkip chỉ báo phân chia BT, hoặc 2 ký tự nhị phân có thể được thu nhận nếu biến partitionSkip chỉ báo phân chia QT. Sơ đồ chiếm giữ mà nhận dạng các nút con được chiếm giữ của nút hiện tại của cây bát phân có thể được xác định dựa trên 1 hoặc 2 ký tự nhị phân thu được.

Trong khi xây dựng cây bát phân, xử lý phân tích có thể được thực hiện trên dòng bit để xác định phần tử cú pháp của sơ đồ chiếm giữ mà nhận dạng các nút con được chiếm giữ của nút hiện tại của cây bát phân. Trong xử lý phân tích, ký tự nhị phân của phần tử cú pháp của sơ đồ chiếm giữ có thể được xác định là được bỏ qua dựa trên biến partitionSkip.

Trong khi xây dựng cây bát phân, đối với nút con của nút hiện tại của cây bát phân được mã hóa trong chế độ trực tiếp, kích cỡ log2 đối với mỗi chiều x, y, và z, được ký hiệu dx, dy, và dz, một cách lần lượt, có thể được xác định đối với nút con dựa trên biến partitionSkip. Có thể được xác định rằng các vị trí của các điểm trong nút con được mã hóa bởi mã hóa độ dài cố định với (dx, dy, dz) bit, một cách lần lượt. Do đó, các ký tự nhị phân tương ứng với các tọa độ các điểm trong nút con có thể được thu nhận dựa trên độ dài mã hóa đã biết.

Theo ví dụ của sáng chế, phần tử cú pháp mà chỉ báo một trong số cá tham số sau đây có thể được thu nhận trong dòng bit để khôi phục cây bát phân: số lượng phân chia QT và BT ẩn lớn nhất được thực hiện trước các phân chia OT, kích cỡ nhỏ nhất của các phân chia QT và BT ẩn mà ngăn chặn các phân

chia QT và BT ẩn của nút khi tất cả các chiều của nút nhỏ hơn hoặc bằng kích cỡ nhỏ nhất, hoặc mức ưu tiên mà chỉ báo phân chia QT hoặc BT ẩn nào được thực hiện đầu tiên khi cả hai phân chia QT và BT được cho phép.

Theo ví dụ của sáng chế, trong khi xây dựng cây bát phân, khi độ sâu cây bát phân của nút hiện tại nhỏ hơn tham số K, hoặc khi kích cỡ log2 nhỏ nhất trong số các kích cỡ log2 trong các chiều x, y, và z của nút hiện tại bằng tham số M, loại phân chia và chiều phân chia để phân chia nút hiện tại có thể được xác định theo các điều kiện trong Bảng sau đây:

Bảng 39

Loại và chiều phân chia	QT dọc theo các trục x-y	QT dọc theo các trục x-z	QT dọc theo các trục y-z
Điều kiện	$d_z < d_x = d_y$	$d_y < d_x = d_z$	$d_x < d_y = d_z$
Loại và chiều phân chia	BT dọc theo trục x	BT along y axis	BT dọc theo trục z
Điều kiện	$d_y < d_x \; and \; d_z < d_x$	$d_x < d_y \; and \; d_z < d_y$	$d_x < d_z \; and \; d_y < d_z$

Tham số K có thể là số nguyên trong phạm vi của $0 \leq K \leq \max(d_x, d_y, d_z) - \min(d_x, d_y, d_z)$, và xác định số lần phân chia QT và BT ẩn lớn nhất mà được cho phép trước các phân chia OT. Tham số M có thể là số nguyên trong phạm vi của $0 \leq M \leq \min(d_x, d_y, d_z)$, và xác định kích cỡ nhỏ nhất của các phân chia QT và BT ẩn mà ngăn chặn các phân chia QT và BT ẩn của nút khi tất cả các chiều của nút nhỏ hơn hoặc bằng M. Các kích cỡ dx, dy, và dz có thể là các kích cỡ log2 của nút hiện tại trong các chiều x, y, và z, một cách lần lượt. Theo ví dụ của sáng chế, loại phân chia và chiều phân chia nêu trên để phân chia nút hiện tại có thể được biểu diễn bởi biến partitionSkip.

Theo ví dụ của sáng chế, mỗi tham số K và M có thể được báo hiệu tới bộ giải mã, hoặc lấy các giá trị được cấu hình trước, và được cập nhật sau, ví dụ, dựa trên các kích cỡ 3D của nút gốc của lát, hoặc chế độ hỗn hợp tam giác phi kết nối (trisoup) có được kích hoạt hay không.

Lưu ý rằng các tham số K hoặc M, hoặc các tham số khác được sử dụng để điều khiển hoặc thực hiện phân chia QT/BT ẩn chỉ được mô tả như là các ví dụ để minh họa một vài phương án của sáng chế. Có thể có các dạng khác nhau của các tham số được báo hiệu trong dòng bit và được sử dụng để thực hiện các phân chia QT/BT tại bộ giải mã. Các tham số này có thể có hoặc có thể không tương tự với K hoặc M, nhưng có thể được sử dụng tương tự để điều khiển làm thế nào các phân chia QT/BT được thực hiện sao cho hiệu năng nén đám mây điểm tốt hơn có thể đạt được.

Sau bước 1520, xử lý 1500 có thể kết thúc tại bước 1599.

VII. Hệ thống máy tính

Các kỹ thuật được mô tả nêu trên, có thể được thực hiện như là phần mềm máy tính nhờ sử dụng các lệnh có thể đọc được bởi máy tính và được lưu trữ vật lý trong một hoặc nhiều phương tiện đọc được bởi máy tính. Ví dụ, Fig.16 thể hiện hệ thống máy tính 1600 thích hợp để thực hiện các phương án của đối tượng được bộc lộ.

Phần mềm máy tính có thể được mã hóa nhờ sử dụng bất kỳ mã máy hoặc ngôn ngữ máy tính thích hợp, mà có thể được tập hợp, biên dịch, liên kết, hoặc cơ chế tương tự để tạo ra mã bao gồm các lệnh mà có thể được thực thi trực tiếp, hoặc thông qua diễn dịch, thực thi vi mã, và loại tương tự, bởi một hoặc nhiều bộ xử lý trung tâm (CPU) máy tính, các bộ xử lý đồ họa (GPU), và loại tương tự.

Các lệnh có thể được thực thi trên các loại máy tính khác nhau hoặc các thành phần của chúng, bao gồm, ví dụ, các máy tính cá nhân, máy tính bảng, máy chủ, điện thoại thông minh, thiết bị chơi trò chơi, thiết bị Internet vạn vật, và loại tương tự.

Các bộ phận được thể hiện trên Fig.16 đối với hệ thống máy tính 1600 là ví dụ về mặt bản chất và không dự định đề xuất bất kỳ giới hạn nào đối với phạm vi sử dụng hoặc chức năng của phần mềm máy tính mà thực hiện các phương án của sáng chế. Cấu trúc của các bộ phận không được hiểu là có bất kỳ sự phụ thuộc hoặc yêu cầu liên quan đến bất kỳ một hoặc kết hợp của các bộ phận được minh họa trong phương án ví dụ của hệ thống máy tính 1600.

Hệ thống máy tính 1600 có thể bao gồm các thiết bị đầu vào giao diện người dùng định trước. Thiết bị đầu vào giao diện người dùng này có thể phản hồi lại việc nhập vào bởi một hoặc nhiều người dùng thông qua, ví dụ, việc nhập xúc giác (như: nhấn phím, trượt, di chuyển bao dữ liệu), nhập audio (như: thoại, vỗ tay), nhập trực quan (như: các cử chỉ), nhập khứu giác (không được thể hiện). Các thiết bị giao diện người dùng có thể cũng được sử dụng để quay nội dung đa phương tiện không cần thiết liên quan trực tiếp đến việc nhập nhận thức bởi người dùng, như audio (như: âm nhạc xung quanh, âm nhạc thoại), các ảnh (như: các ảnh được quét, các ảnh chụp thu nhận từ camera ảnh tĩnh), video (như video hai chiều, video ba chiều bao gồm video lập thể).

Các thiết bị giao diện người dùng đầu vào có thể bao gồm một hoặc nhiều trong số (chỉ một trong số các thiết bị được thể hiện): bản phím 1601, chuột 1602, bàn di chuột 1603, màn chạm 1610, bao dữ liệu (không được thể hiện), cần điều khiển 1605, microphôn 1606, máy quét 1607, camera 1608.

Hệ thống máy tính 1600 có thể cũng bao gồm các thiết bị đầu ra giao diện người dùng định trước. Các thiết bị đầu ra giao diện người dùng này có thể là sự mô phỏng các giác quan của một hoặc nhiều người dùng thông qua, ví dụ, xúc giác, âm thanh, ánh sáng, và mùi/vị giác. Các thiết bị đầu ra giao diện người dùng này có thể bao gồm các thiết bị đầu ra xúc giác (ví dụ xúc giác phản hồi bởi màn chạm 1610, bao dữ liệu (không được thể hiện), hoặc cần điều khiển 1605, nhưng có thể cũng có các thiết bị phản hồi xúc giác mà không đóng vai trò là các thiết bị đầu vào), các thiết bị xuất audio (như: các loa 1609, các tai nghe (không được thể hiện)), các thiết bị đầu ra trực quan (như các màn hình 1610 bao

gồm các màn hình CRT, màn hình LCD, màn hình plasma, màn hình OLED, mỗi trong số chúng có hoặc không có khả năng nhập màn chạm, có hoặc không có khả năng phản hồi xúc giác—một vài trong số chúng có thể có khả năng xuất ra đầu ra trực quan hai chiều hoặc đầu ra nhiều hơn ba chiều thông qua phương tiện như đầu ra lập thể; các kính thực tế ảo (không được thể hiện), các màn hình toàn ký và bể khói (không được thể hiện)), và các máy in (không được thể hiện).

Hệ thống máy tính 1600 có thể cũng bao gồm các thiết bị lưu trữ có thể truy nhập được bởi người dùng và phương tiện được kết hợp của chúng như phương tiện quang bao gồm CD/DVD ROM/RW 1620 với CD/DVD hoặc phương tiện tương tự 1621, ổ chớp 1622, ổ cứng tháo rời hoặc ổ bán dẫn 1623, phương tiện từ truyền thống như băng và đĩa mềm (không được thể hiện), các thiết bị dựa trên ROM/ASIC/PLD chuyên dụng như các khóa điện tử bảo mật (không được thể hiện), và loại tương tự.

Người có hiểu biết trung bình trong lĩnh vực kỹ thuật cũng sẽ hiểu rằng thuật ngữ "phương tiện đọc được bởi máy tính" như được sử dụng kết hợp với đối tượng được bộc lộ không bao hàm môi trường truyền, các sóng mang, hoặc các tín hiệu chuyển tiếp khác.

Hệ thống máy tính 1600 có thể cũng bao gồm giao diện 1654 tới một hoặc nhiều mạng truyền thông 1655. Các mạng có thể ví dụ là mạng không dây, nối đây, quang học. Các mạng có thể là mạng cục bộ, mạng diện rộng, mạng đô thị, mạng phương tiện giao thông và công nghiệp, thời gian thực, dung sai trễ, và v.v. Các ví dụ của các mạng bao gồm các mạng cục bộ như Ethernet, LAN không dây, mạng tế bào bao gồm GSM, 3G, 4G, 5G, LTE và loại tương tự, TV các mạng số diện rộng có dây hoặc không dây bao gồm TV cáp, TV vệ tinh, và TV quảng bá mặt đất, phương tiện giao thông và công nghiệp bao gồm CANBus, và v.v. Các mạng nhất định thường yêu cầu các bộ thích ứng giao diện mạng phía ngoài mà được gắn vào cổng dữ liệu mục đích chung định trước hoặc các kênh truyền ngoại vi 1649 (như, ví dụ các cổng USB của hệ thống máy tính 1600); các phần khác thường được tích hợp vào lõi của hệ thống máy tính 1600 bằng cách

gắn tới kênh truyền hệ thống như được mô tả dưới đây (ví dụ giao diện Ethernet vào hệ thống máy tính PC hoặc giao diện mạng tế bào vào hệ thống máy tính điện thoại thông minh). Nhờ sử dụng bất kỳ các mạng này, hệ thống máy tính 1600 có thể truyền thông với các thực thể khác. Việc truyền thông này có thể là chỉ thu đơn hướng (ví dụ, TV quảng bá), chỉ gửi đơn hướng (ví dụ các thiết bị CANbus tới CANbus định trước), hoặc hai chiều, ví dụ tới các hệ thống máy tính khác nhờ sử dụng các mạng số diện rộng hoặc cục bộ. Các giao thức định trước và các ngăn giao thức có thể được sử dụng trên mỗi mạng và giao diện mạng này như được mô tả nêu trên.

Các thiết bị giao diện người dùng, các thiết bị lưu trữ có thể truy nhập được bởi người dùng, và các giao diện mạng nêu trên có thể được gắn vào lõi 1640 của hệ thống máy tính 1600.

Lõi 1640 có thể bao gồm một hoặc nhiều bộ xử lý trung tâm (CPU) 1641, các bộ xử lý đồ họa (GPU) 1642, các bộ xử lý khả trình chuyên dụng dưới dạng của mảng cổng khả trình dạng trường (FPGA) 1643, các bộ tăng tốc phần cứng đối với các tác vụ định trước 1644, các bộ điều chỉnh thích ứng đồ họa 1650, và v.v. Các thiết bị này, cùng với bộ nhớ chỉ đọc (ROM) 1645, bộ nhớ truy nhập ngẫu nhiên 1646, bộ lưu trữ khối bên trong như các ổ cứng không thể truy nhập bởi người dùng bên trong, SSD, và loại tương tự 1647, có thể được kết nối thông qua kênh truyền hệ thống 1648. Trong một vài hệ thống máy tính, kênh truyền hệ thống 1648 có thể truy nhập được trong dạng của một hoặc nhiều đầu cắm vật lý để cho phép các phần mở rộng bởi các CPU, GPU bổ sung, và loại tương tự. Các thiết bị ngoại vi có thể được gắn trực tiếp tới kênh truyền hệ thống của lõi 1648, hoặc thông qua kênh truyền ngoại vi 1649. Trong ví dụ của sáng chế, màn hình 1610 có thể được kết nối tới bộ điều chỉnh thích ứng đồ họa 1650. Các cấu trúc của các kênh truyền ngoại vi bao gồm PCI, USB, và loại tương tự.

Các CPU 1641, GPU 1642, FPGA 1643, và các bộ tăng tốc 1644 có thể thực thi các lệnh định trước mà, một cách kết hợp, có thể thực thi mã máy tính nêu trên. Mã máy tính có thể được lưu trữ trong ROM 1645 hoặc RAM 1646.

Dữ liệu chuyển tiếp có thể cũng được lưu trữ trong RAM 1646, trong khi dữ liệu cố định có thể được lưu trữ ví dụ, trong bộ nhớ khối bên trong 1647. Việc lưu trữ nhanh và truy xuất tới bất kỳ thiết bị bộ nhớ có thể được cho phép thông qua việc sử dụng của bộ nhớ lưu trữ đệm, mà có thể được kết hợp mật thiết với một hoặc nhiều CPU 1641, GPU 1642, bộ lưu trữ cỡ lớn 1647, ROM 1645, RAM 1646, và loại tương tự.

Phương tiện đọc được bởi máy tính có thể có mã máy tính trên đó để thực hiện các hoạt động được thực hiện bởi máy tính khác nhau. Phương tiện và mã máy tính có thể được thiết kế và xây dựng đặc biệt cho mục đích của sáng chế, hoặc chúng có thể là loại được biết đến rộng rãi và khả dụng đối với người có hiểu biết trung bình trong lĩnh vực phần mềm máy tính.

Theo ví dụ của sáng chế và không bị giới hạn, hệ thống máy tính có cấu trúc 1600, và cụ thể, lõi 1640 có thể cung cấp chức năng như là kết quả của các bộ xử lý (bao gồm các CPU, GPU, FPGA, các bộ tăng tốc, và loại tương tự) mà thực thi phần mềm trong một hoặc nhiều phương tiện hữu hình đọc được bởi máy tính. Các phương tiện đọc được bởi máy tính này có thể là phương tiện được kết hợp với bộ lưu trữ khối có thể truy nhập bởi người dùng như được giới thiếu nêu trên, cũng như bộ lưu trữ định trước của lõi 1640 mà có bản chất bất biến, như bộ lưu trữ khối bên trong của lõi 1647 hoặc ROM 1645. Phần mềm mà thực hiện các phương án của sáng chế có thể được lưu trữ trong các thiết bị này và được thực thi bởi lõi 1640. Phương tiện đọc được bởi máy tính có thể bao gồm một hoặc nhiều thiết bị bộ nhớ hoặc chip, theo nhu cầu cụ thể. Phần mềm có thể làm cho lõi 1640 và một cách cụ thể, các bộ xử lý trong đó (bao gồm CPU, GPU, FPGA, và loại tương tự) thực hiện các xử lý cụ thể hoặc các phần cụ thể của các xử lý cụ thể được mô tả ở đây, bao gồm xác định các cấu trúc dữ liệu được lưu trữ trong RAM 1646 và cải biến các cấu trúc dữ liệu này theo các xử lý được xác định bởi phần mềm. Ngoài ra hoặc như là tùy chọn khác, hệ thống máy tính có thể cung cấp chức năng như là kết quả của mạng cứng logic hoặc được lắp trong mạch (ví dụ: bộ tăng tốc 1644), mà có thể hoạt động trong vị trí của hoặc cùng

với phần mềm để thực hiện các xử lý cụ thể hoặc các phần cụ thể của các xử lý cụ thể được mô tả ở đây. Việc tham chiếu tới phần mềm có thể bao hàm logic, và ngược lại, nếu cần. Việc tham chiếu tới phương tiện đọc được bởi máy tính có thể bao hàm mạch (như mạch tích hợp (IC)) mà lưu trữ phần mềm để thực thi, mạch thực hiện logic để thực thi, hoặc cả hai, nếu cần. Sáng chế bao hàm kết hợp thích hợp của phần cứng và phần mềm.

Trong khi các khía cạnh của sáng chế đã được mô tả kết hợp với các phương án cụ thể mà được đề xuất như là ví dụ, các thay thế, cải biến, và biến thể đối với các ví dụ này có thể được thực hiện. Do đó, các phương án được mô tả ở đây nhằm mục đích minh họa và không giới hạn. Có các thay đổi mà có thể được thực hiện mà không đi chệnh khỏi phạm vi của yêu cầu bảo hộ dưới đây.

YÊU CẦU BẢO HỘ

1. Phương pháp giải mã hình học đám mây điểm trong bộ giải mã đám mây điểm, bao gồm:

thu dòng bit bao gồm lát của khung đám mây điểm được mã hóa; và

khôi phục cây bát phân mà biểu diễn hình học của các điểm trong hộp giới hạn của lát mà trong đó loại phân chia và chiều phân chia được xác định cho nút hiện tại của cây bát phân mà được phân chia với phân chia cây tứ phân (QT-quatree) hoặc phân chia cây nhị phân (BT- binary tree),

trong đó việc khôi phục của cây bát phân bao gồm:

xác định giá trị của biến, được ký hiệu bởi biến partitionSkip, mà chỉ rõ loại phân chia và hướng phân chia của nút hiện tại của cây bát phân.

2. Phương pháp theo điểm 1, trong đó bước xây dựng cây bát phân bao gồm:

xác định làm thế nào để phân chia nút hiện tại của cây bát phân nhờ sử dụng một trong số phân chia QT, phân chia BT, hoặc phân chia cây bát phân (OT-octree) dựa trên điều kiện định trước.

3. Phương pháp theo điểm 1, trong đó bước xây dựng cây bát phân bao gồm:

xác định làm thế nào để phân chia nút hiện tại của cây bát phân nhờ sử dụng một trong số phân chia QT, phân chia BT, hoặc phân chia OT dựa trên một hoặc nhiều tham số, một trong số một hoặc nhiều tham số được báo hiệu trong dòng bit hoặc sử dụng giá trị được cấu hình trước cục bộ.

4. Phương pháp theo điểm 1, trong đó bước xây dựng cây bát phân bao gồm:

thu các ký tự nhị phân chiếm giữ thuộc về mã chiếm giữ 8 ký tự nhị phân của nút hiện tại của cây bát phân từ dòng bit,

trong đó mỗi ký tự nhị phân chiếm giữ tương ứng với nút con được chiếm giữ của nút hiện tại của cây bát phân,

4 ký tự nhị phân thuộc về mã chiếm giữ 8 ký tự nhị phân không được báo hiệu trong dòng bit khi nút hiện tại của cây bát phân được phân chia với phân chia QT, và

6 ký tự nhị phân thuộc về mã chiếm giữ 8 ký tự nhị phân không được báo hiệu trong dòng bit khi nút hiện tại của cây bát phân được phân chia với phân chia BT.

5. Phương pháp theo điểm 1, trong đó bước xây dựng cây bát phân bao gồm:

thu một hoặc nhiều phần tử cú pháp mà chỉ báo các kích cỡ ba chiều (3D) của hộp giới hạn của lát của khung đám mây điểm được mã hóa từ dòng bit.

6. Phương pháp theo điểm 1, trong đó biến partitionSkip được biểu diễn trong dạng nhị phân với ba bit tương ứng với các chiều x, y, và z, một cách lần lượt, và mỗi bit chỉ báo rằng việc có được thực hiện dọc theo chiều x, y, hoặc z tương ứng hay không.

7. Phương pháp theo điểm 1, trong đó bước xây dựng cây bát phân còn bao gồm:

cập nhật độ sâu trong chiều x, y, hoặc z đối với nút con của nút hiện tại của cây bát phân dựa trên biến partitionSkip.

8. Phương pháp theo điểm 1, trong đó bước xây dựng cây bát phân còn bao gồm:

thu phần tử cú pháp mà chỉ báo nút hiện tại của cây bát phân có một nút con được chiếm giữ;

thu 1 ký tự nhị phân nếu biến partitionSkip chỉ báo phân chia BT, hoặc 2 ký tự nhị phân nếu biến partitionSkip chỉ báo phân chia QT; và

xác định sơ đồ chiếm giữ mà nhận dạng các nút con được chiếm giữ của nút hiện tại của cây bát phân dựa trên 1 hoặc 2 ký tự nhị phân thu được.

9. Phương pháp theo điểm 1, trong đó bước xây dựng cây bát phân còn bao gồm:

trong xử lý phân tích trên dòng bit để xác định phần tử cú pháp của sơ đồ chiếm giữ mà nhận dạng các nút con được chiếm giữ của nút hiện tại của cây bát

phân, xác định ký tự nhị phân của phần tử cú pháp của sơ đồ chiếm giữ được bỏ qua dựa trên biến partitionSkip.

10. Phương pháp theo điểm 1, trong đó bước xây dựng cây bát phân còn bao gồm:

đối với nút con của nút hiện tại của cây bát phân được mã hóa trong chế độ trực tiếp, xác định kích cỡ log2 đối với mỗi chiều x, y, và z, được ký hiệu dx, dy, và dz, một cách lần lượt, đối với nút con dựa trên biến partitionSkip, trong đó các vị trí của các điểm trong nút con được mã hóa bởi mã hóa độ dài cố định với (dx, dy, dz) bit, một cách lần lượt.

11. Phương pháp theo điểm 1, trong đó bước xây dựng cây bát phân còn bao gồm:

thu phần tử cú pháp từ dòng bit mà chỉ báo một trong số các tham số sau đây:

số lượng phân chia QT và BT ẩn lớn nhất được thực hiện trước các phân chia OT,

kích cỡ nhỏ nhất của các phân chia QT và BT ẩn mà ngăn chặn các phân chia QT và BT ẩn của nút khi tất cả các chiều của nút nhỏ hơn hoặc bằng kích cỡ nhỏ nhất, hoặc

mức ưu tiên mà chỉ báo phân chia nào trong số phân chia QT hoặc BT ẩn được thực hiện đầu tiên khi cả hai phân chia QT và BT được cho phép.

12. Phương pháp theo điểm 1, trong đó bước xây dựng cây bát phân còn bao gồm:

để phản hồi lại việc độ sâu cây bát phân của nút hiện tại nhỏ hơn tham số K, hoặc kích cỡ log2 nhỏ nhất trong số các kích cỡ log2 trong các chiều x, y, và z của nút hiện tại bằng tham số M, xác định loại phân chia và chiều phân chia để phân chia nút hiện tại theo các điều kiện trong Bảng sau đây:

Loại và chiều phân chia	QT dọc theo các trục x-y	QT dọc theo các trục x-z	QT dọc theo các trục y-z

Điều kiện	$d_z < d_x = d_y$	$d_y < d_x = d_z$	$d_x < d_y = d_z$
Loại và chiều phân chia	BT dọc theo trục x	BT along y axis	BT dọc theo trục z
Điều kiện	$d_y < d_x$ và $d_z < d_x$	$d_x < d_y$ và $d_z < d_y$	$d_x < d_z$ và $d_y < d_z$

trong đó:

tham số K là số nguyên trong phạm vi của $0 \leq K \leq \max(d_x, d_y, d_z) - \min(d_x, d_y, d_z)$, và xác định số lần phân chia QT và BT ẩn lớn nhất mà được cho phép trước các phân chia OT,

tham số M là số nguyên trong phạm vi của $0 \leq M \leq \min(d_x, d_y, d_z)$, xác định kích cỡ nhỏ nhất của các phân chia QT và BT ẩn, và ngăn chặn các phân chia QT và BT ẩn của nút khi tất cả các chiều của nút nhỏ hơn hoặc bằng M, và

d_x, d_y, và d_z là các kích cỡ log2 của nút hiện tại trong các chiều x, y, và z, một cách lần lượt.

13. Phương pháp theo điểm 1, trong đó bước xây dựng cây bát phân còn bao gồm:

để phản hồi lại việc độ sâu cây bát phân của nút hiện tại nhỏ hơn tham số K, hoặc kích cỡ log2 nhỏ nhất trong số các kích cỡ log2 trong các chiều x, y, và z của nút hiện tại bằng tham số M, xác định biến partitionSkip như sau,

if (dx < MaxNodeDimLog2),

partitionSkip |= 4;

if (dy < MaxNodeDimLog2),

partitionSkip |= 2;

if (dz < MaxNodeDimLog2),

partitionSkip |= 1

trong đó:

biến partitionSkip được biểu diễn trong dạng nhị phân với ba bit, và chỉ rõ loại phân chia và chiều phân chia của nút hiện tại của cây bát phân,

tham số K là số nguyên trong phạm vi của $0 \leq K \leq \max(d_x, d_y, d_z) - \min(d_x, d_y, d_z)$, và xác định số lần phân chia QT và BT ẩn lớn nhất mà được cho phép trước các phân chia OT,

tham số M là số nguyên trong phạm vi của $0 \leq M \leq \min(d_x, d_y, d_z)$, xác định kích cỡ nhỏ nhất của các phân chia QT và BT ẩn, và ngăn chặn các phân chia QT và BT ẩn của nút khi tất cả các chiều của nút nhỏ hơn hoặc bằng M,

dx, dy, và dz là các kích cỡ log2 của nút hiện tại trong các chiều x, y, và z, một cách lần lượt,

MaxNodeDimLog2 biểu diễn kích cỡ log2 lớn nhất trong số dx, dy, và dz, và

toán tử |= biểu diễn toán tử hoặc (OR) đảo bit phức hợp.

14. Phương pháp theo điểm 1, trong đó bước xây dựng cây bát phân còn bao gồm:

thu từ dòng bit cờ mà chỉ báo rằng phân chia hình học ẩn được kích hoạt đối với chuỗi của các khung đám mây điểm hay lát của khung đám mây điểm được mã hóa.

15. Phương pháp theo điểm 1, trong đó bước xây dựng cây bát phân bao gồm:

xác định chế độ phẳng là không khả dụng tại chiều x, y, hoặc z trong đó việc phân chia không được thực hiện đối với nút hiện tại.

16. Phương pháp theo điểm 1, trong đó bước xây dựng cây bát phân bao gồm:

thực hiện xử lý phân tích chiếm giữ cây bát phân hình học, trong đó đối với ký tự nhị phân mà có chỉ số bằng binIdx trong mã chiếm giữ, biến binIsInferred được thiết lập theo như sau:

nếu một trong các điều kiện sau đây là đúng, binIsInferred được thiết lập bằng 1:

(1) biến NeighbourPattern bằng 0 và số lượng của 1-valued ký tự nhị phân được giải mã trước đó nhỏ hơn hoặc bằng (binIdx + minOccupied – maxOccupied), hoặc

(2) biến NeighbourPattern không bằng bằng 0, binIdx bằng maxOccupied-1 và các giá trị của tất cả ký tự nhị phần được giải mã trước đó bằng 0,

trong đó minOccupied=2, và maxOccupied=8 nếu phân chia OT được áp dụng, maxOccupied=4 nếu phân chia QT được áp dụng, maxOccupied=2 nếu phân chia BT được áp dụng, và

nếu không phải, nếu các điều kiện nêu không là đúng, binIsInferred được thiết lập bằng 0.

17. Phương pháp theo điểm 1, trong đó bước xây dựng cây bát phân còn bao gồm:

cập nhật tham số K mà chỉ báo số lượng phân chia QT và BT ẩn lớn nhất trước các phân chia OT, và tham số M mà chỉ báo kích cỡ nhỏ nhất của các phân chia QT và BT ẩn mà ngăn chặn các phân chia QT và BT ẩn của nút khi tất cả các chiều của nút nhỏ hơn hoặc bằng kích cỡ nhỏ nhất theo:

nếu K lớn hơn độ chênh lệch giữa chiều log2 nút gốc lớn nhất và chiều log2 nút gốc nhỏ nhất của lát, K được thay đổi thành độ chênh lệch giữa chiều log2 nút gốc lớn nhất và chiều log2 nút gốc nhỏ nhất của lát;

nếu M lớn hơn chiều log2 nút gốc nhỏ nhất của lát, M được thay đổi thành chiều log2 nút gốc nhỏ nhất của lát;

nếu chiều log2 nút gốc lớn nhất và chiều log2 nút gốc nhỏ nhất của lát bằng nhau, M được thay đổi thành 0; và

nếu chế độ hỗn hợp tam giác phi kết nối (trisoup) được kích hoạt, K được thay đổi thành độ chênh lệch giữa chiều log2 nút gốc lớn nhất và chiều log2 nút gốc nhỏ nhất của lát, và M được thay đổi thành chiều log2 nút gốc nhỏ nhất của lát.

18. Thiết bị giải mã hình học đám mây điểm, bao gồm hệ mạch có cấu trúc để thực hiện phương pháp theo điểm bất kỳ trong số các điểm 1 đến 17.

19. Phương tiện bất biến đọc được bởi máy tính mà lưu trữ các lệnh mà, khi được thực thi bởi bộ xử lý, làm cho bộ xử lý thực hiện phương pháp theo điểm bất kỳ trong số các điểm 1 đến 17.

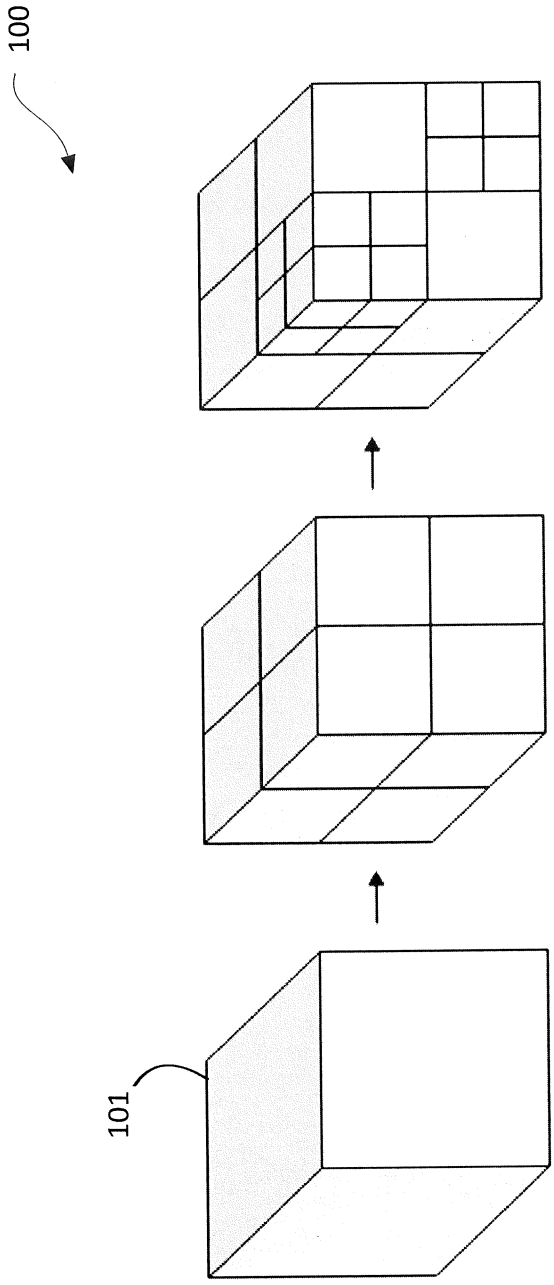


FIG. 1

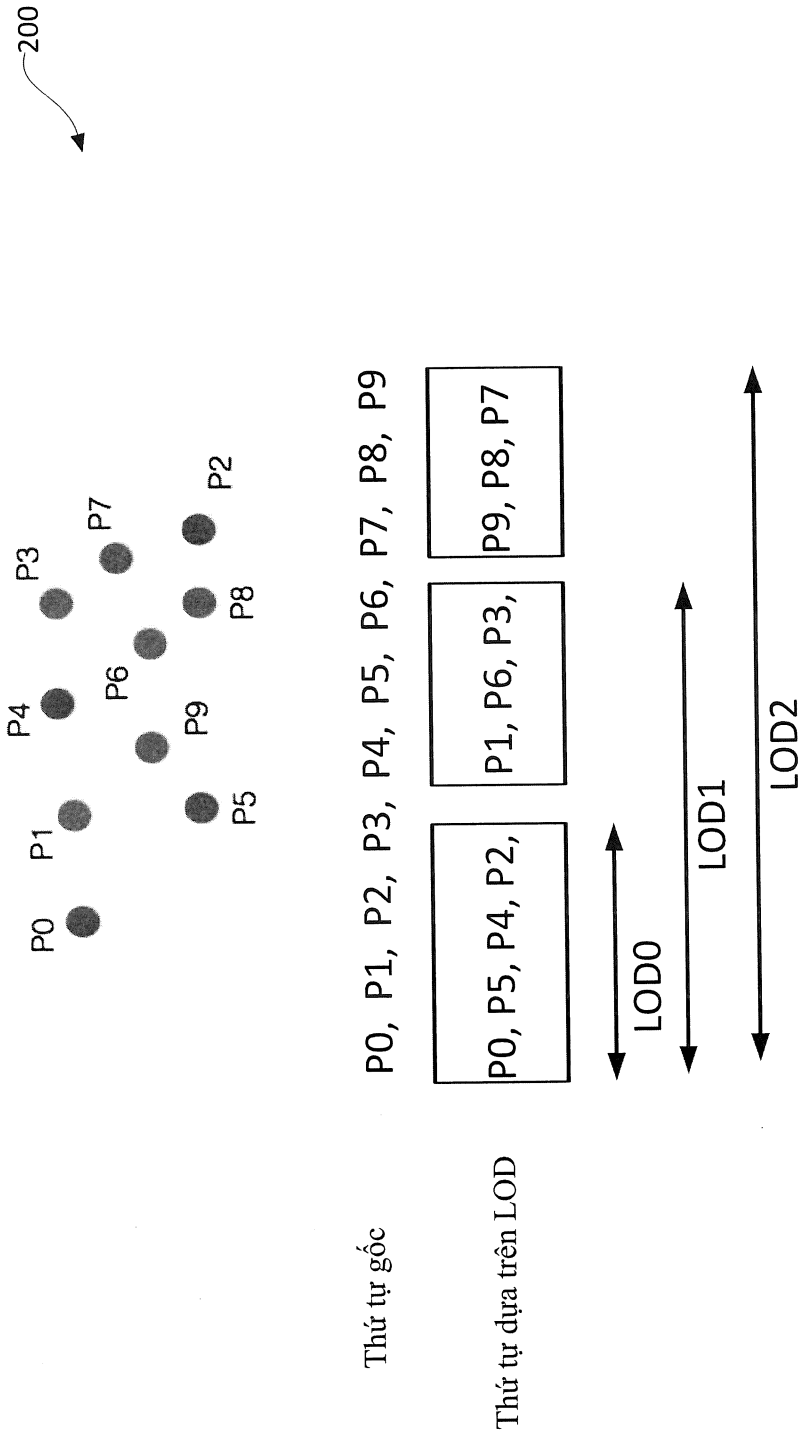

200
P0
P1
P4
P3
P7
P2
P6
P8
P9
P5
Thứ tự gốc
P0, P1, P2, P3, P4, P5, P6, P7, P8, P9
Thứ tự dựa trên LOD
P0, P5, P4, P2,
P1, P6, P3,
P9, P8, P7
LOD0
LOD1
LOD2

FIG. 2

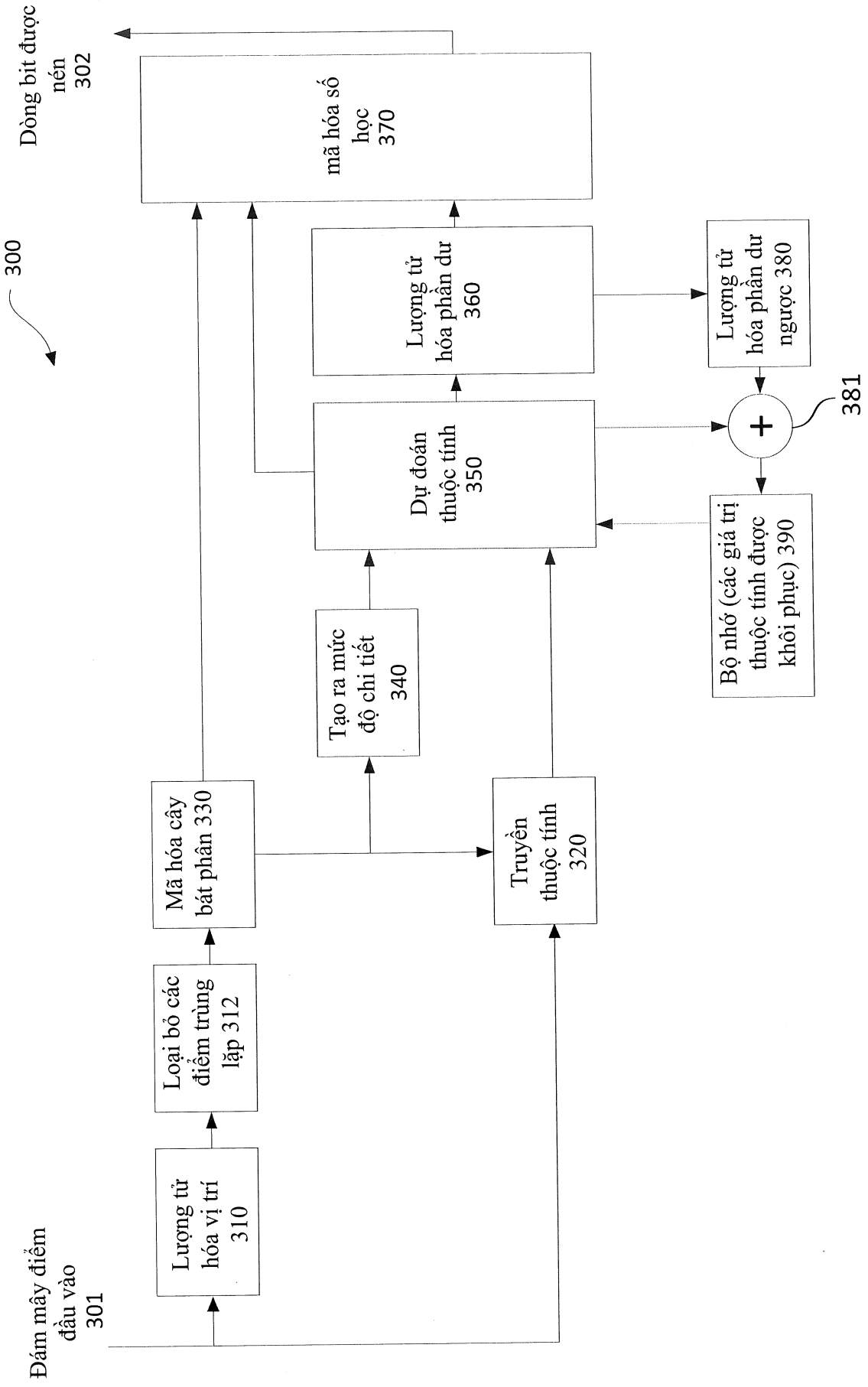

Đám mây điểm đầu vào 301
Lượng tử hóa vị trí 310
Loại bỏ các điểm trùng lặp 312
Mã hóa cây bát phân 330
Truyền thuộc tính 320
Tạo ra mức độ chi tiết 340
Dự đoán thuộc tính 350
Lượng tử hóa phần dư 360
mã hóa số học 370
Dòng bit được nén 302
300
Bộ nhớ (các giá trị thuộc tính được khôi phục) 390
Lượng tử hóa phần dư ngược 380
381
+

FIG. 3

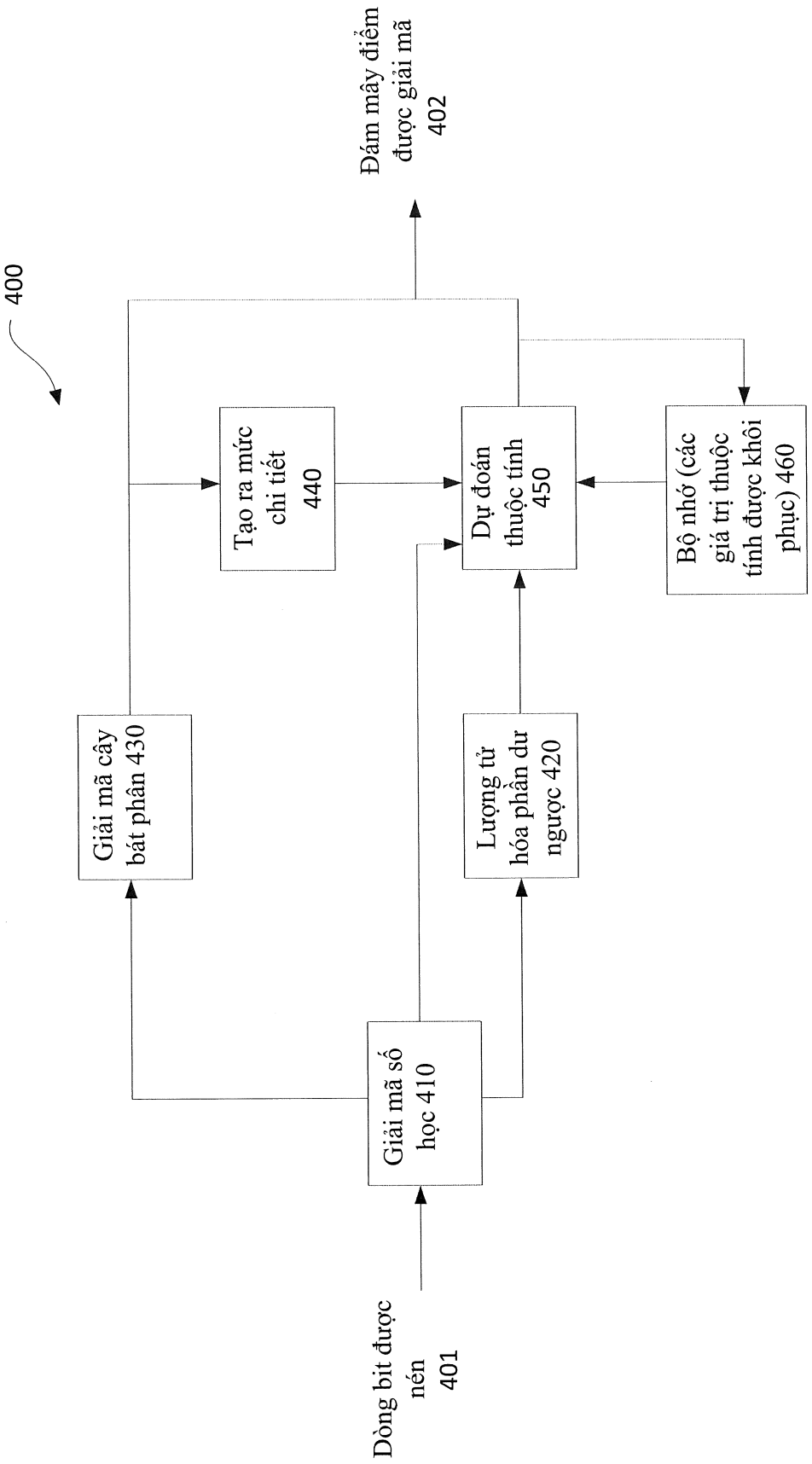

400
Đám mây điểm được giải mã 402
Tạo ra mức chi tiết 440
Dự đoán thuộc tính 450
Bộ nhớ (các giá trị thuộc tính được khôi phục) 460
Giải mã cây bát phân 430
Lượng tử hóa phần dư ngược 420
Giải mã số học 410
Dòng bit được nén 401

FIG. 4

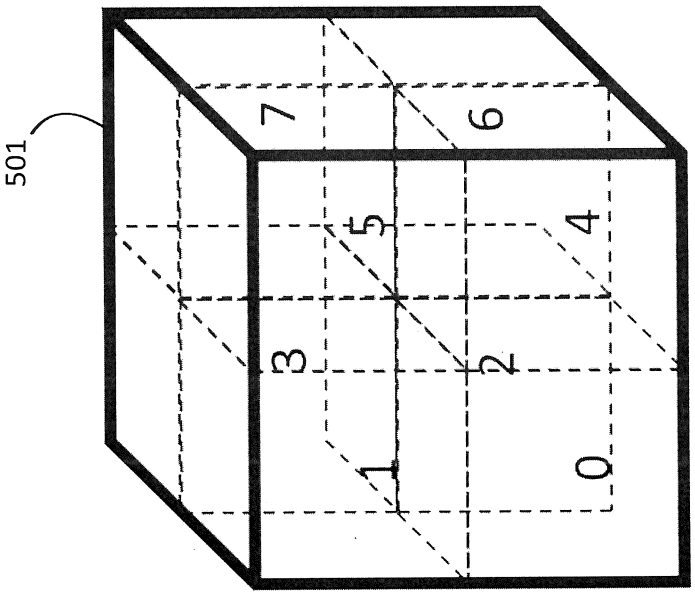


FIG. 5

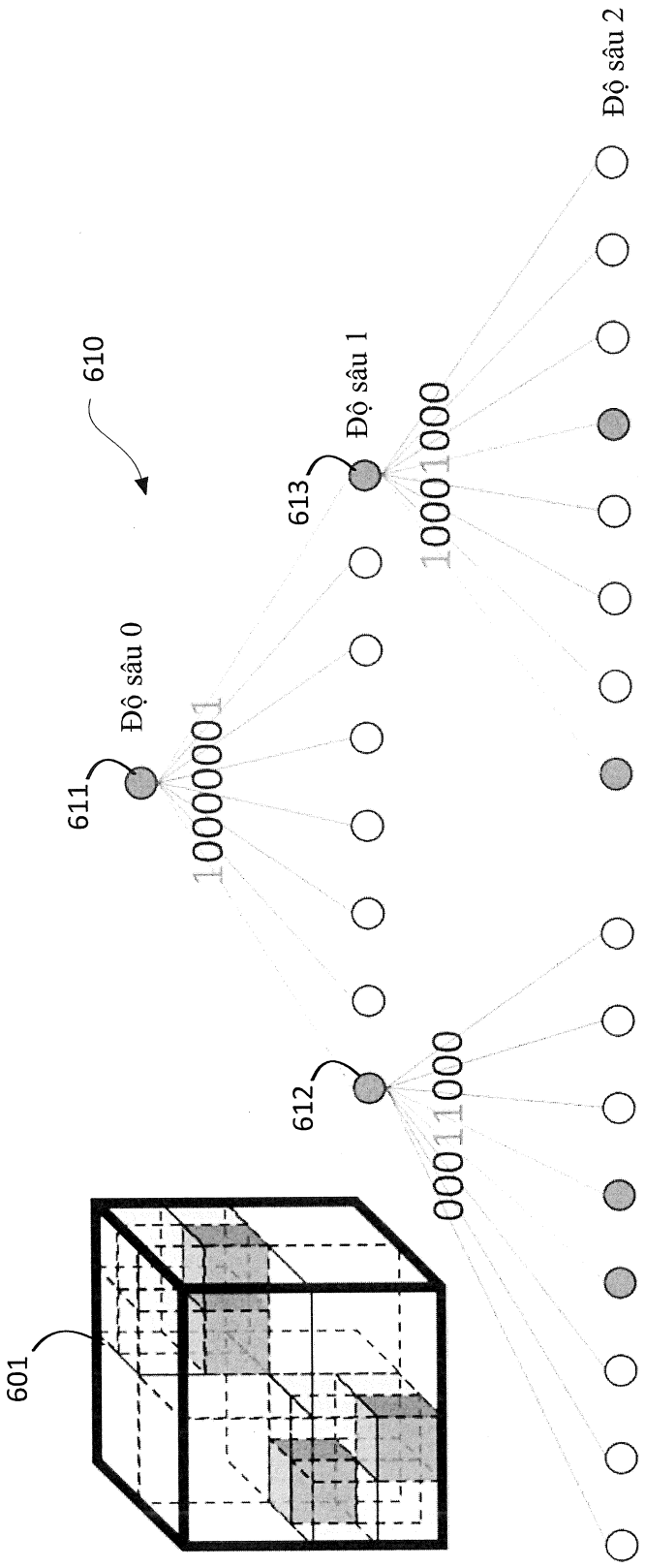

610
Độ sâu 0
Độ sâu 1
Độ sâu 2
611
612
613
10000001
10001000
00011000
601

FIG. 6

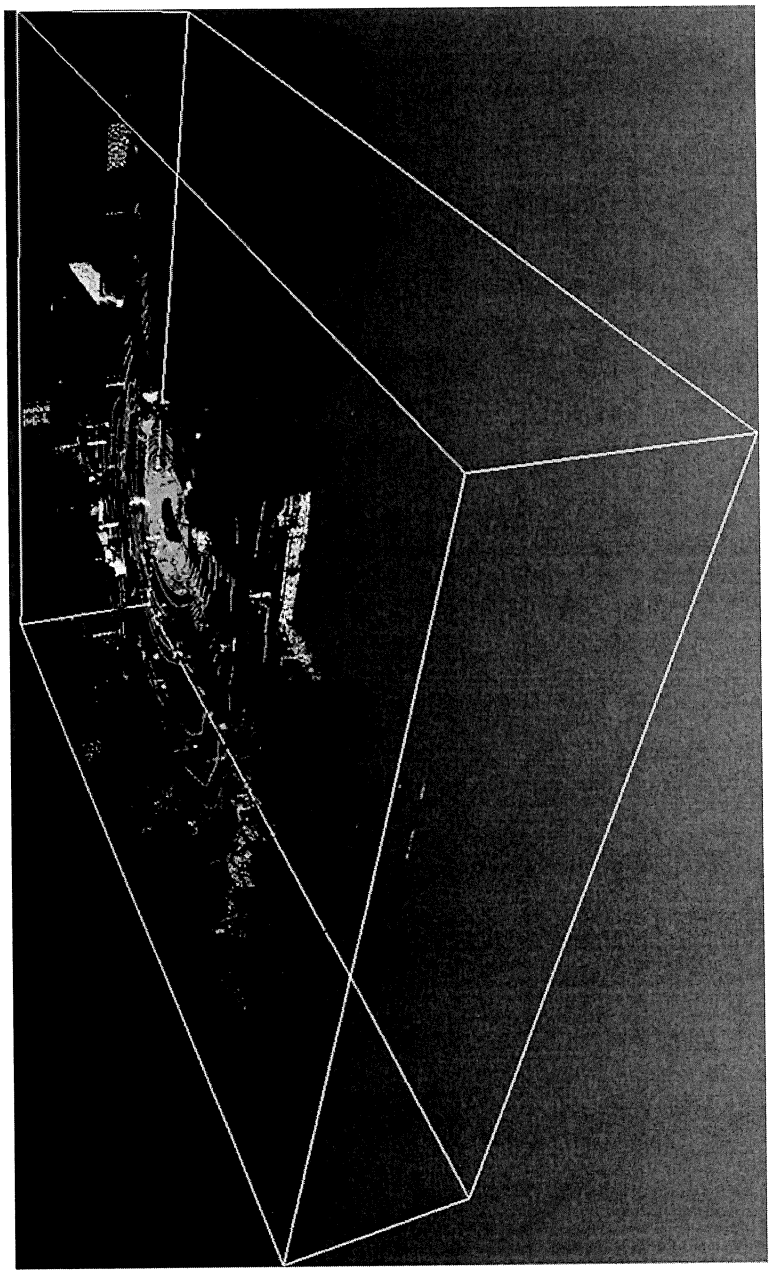


FIG. 7

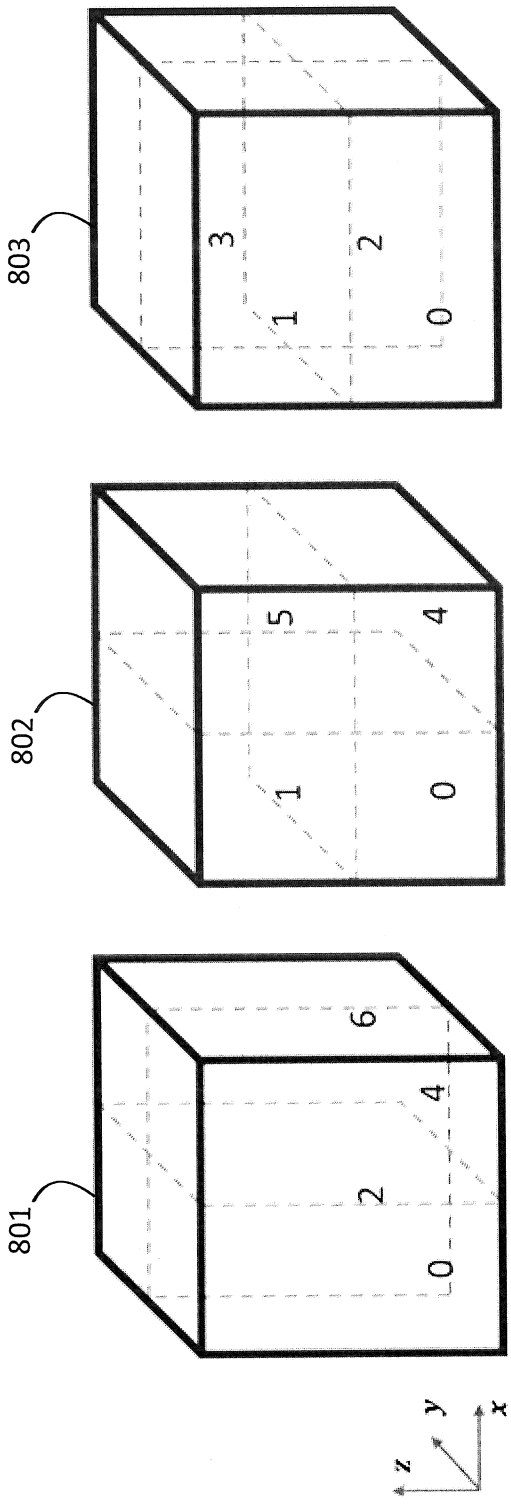


FIG. 8

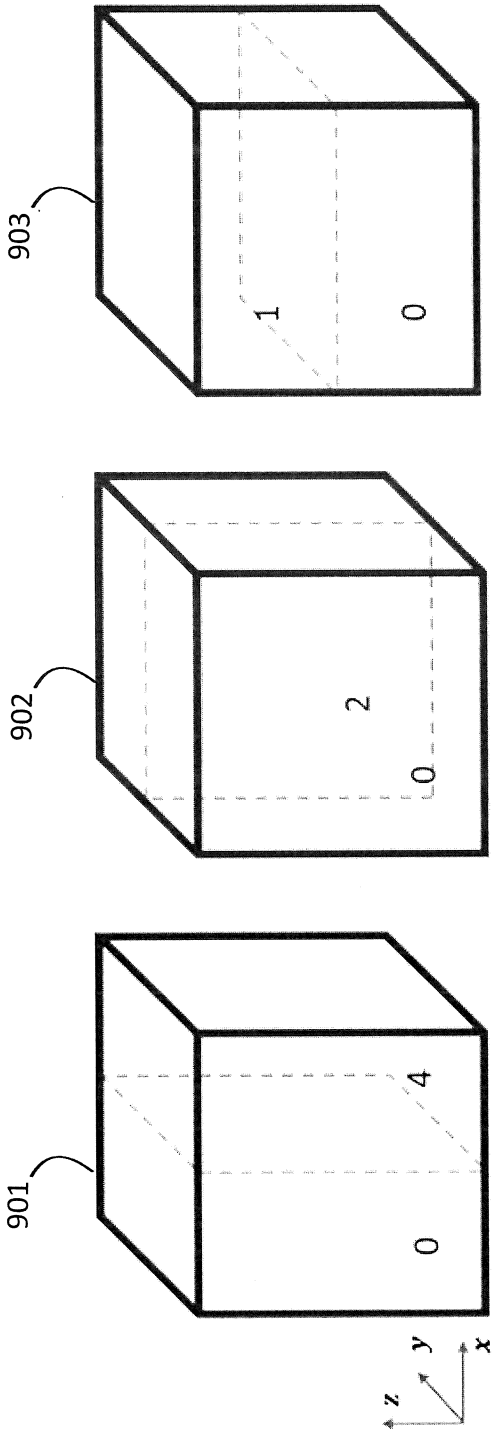


FIG. 9

FIG. 10

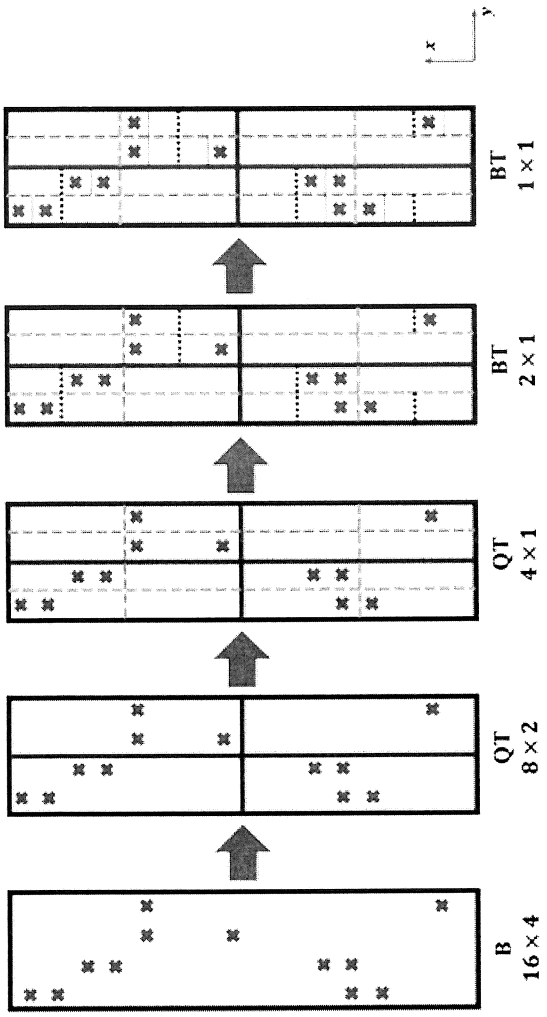


FIG. 11

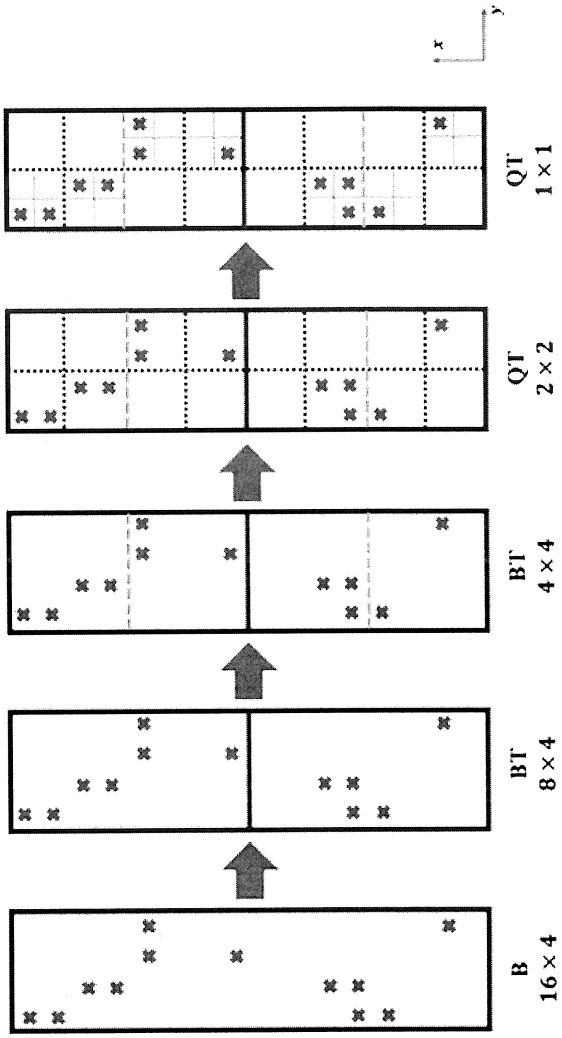


FIG. 12

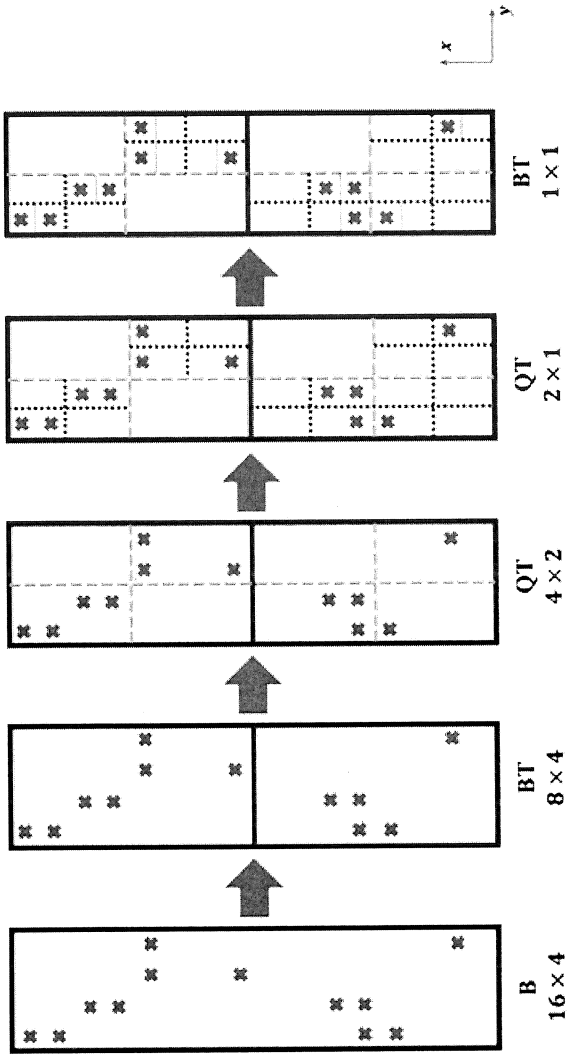
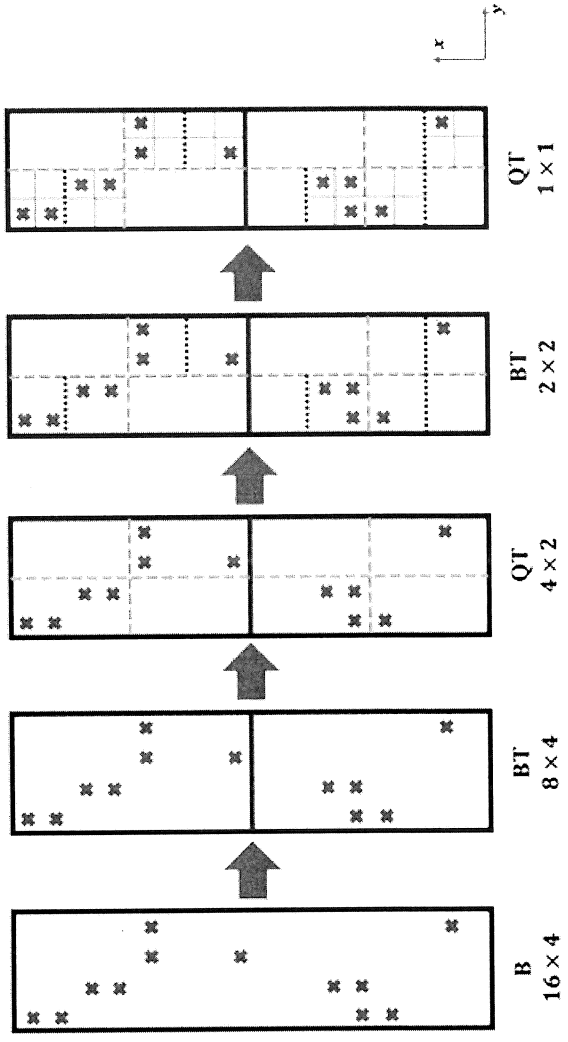


FIG. 13



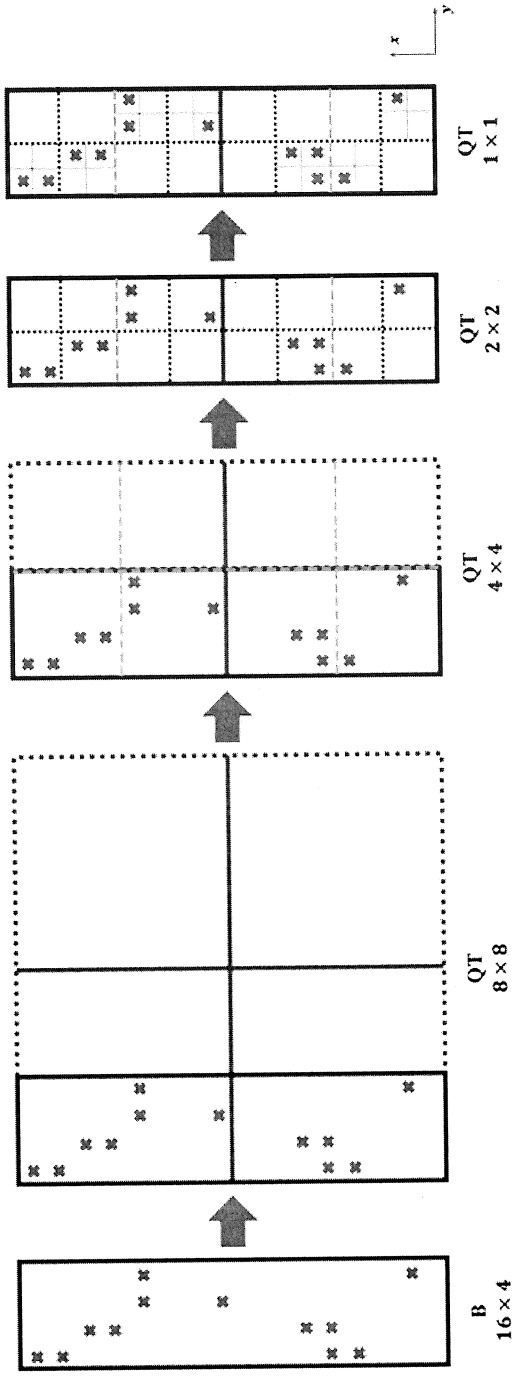


FIG. 14

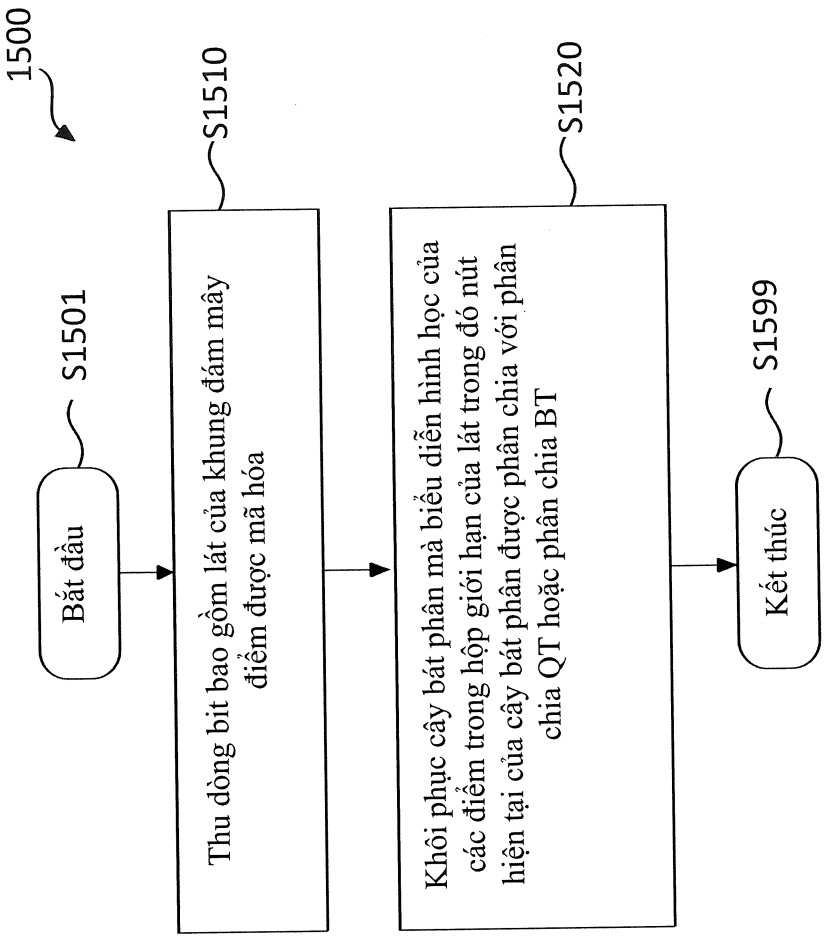

1500
S1501
Bắt đầu
S1510
Thu dòng bit bao gồm lát cửa khung đám mây điểm được mã hóa
S1520
Khôi phục cây bát phân mà biểu diễn hình học của các điểm trong hộp giới hạn của lát trong đó nút hiện tại của cây bát phân được phân chia với phân chia QT hoặc phân chia BT
S1599
Kết thúc

FIG. 15

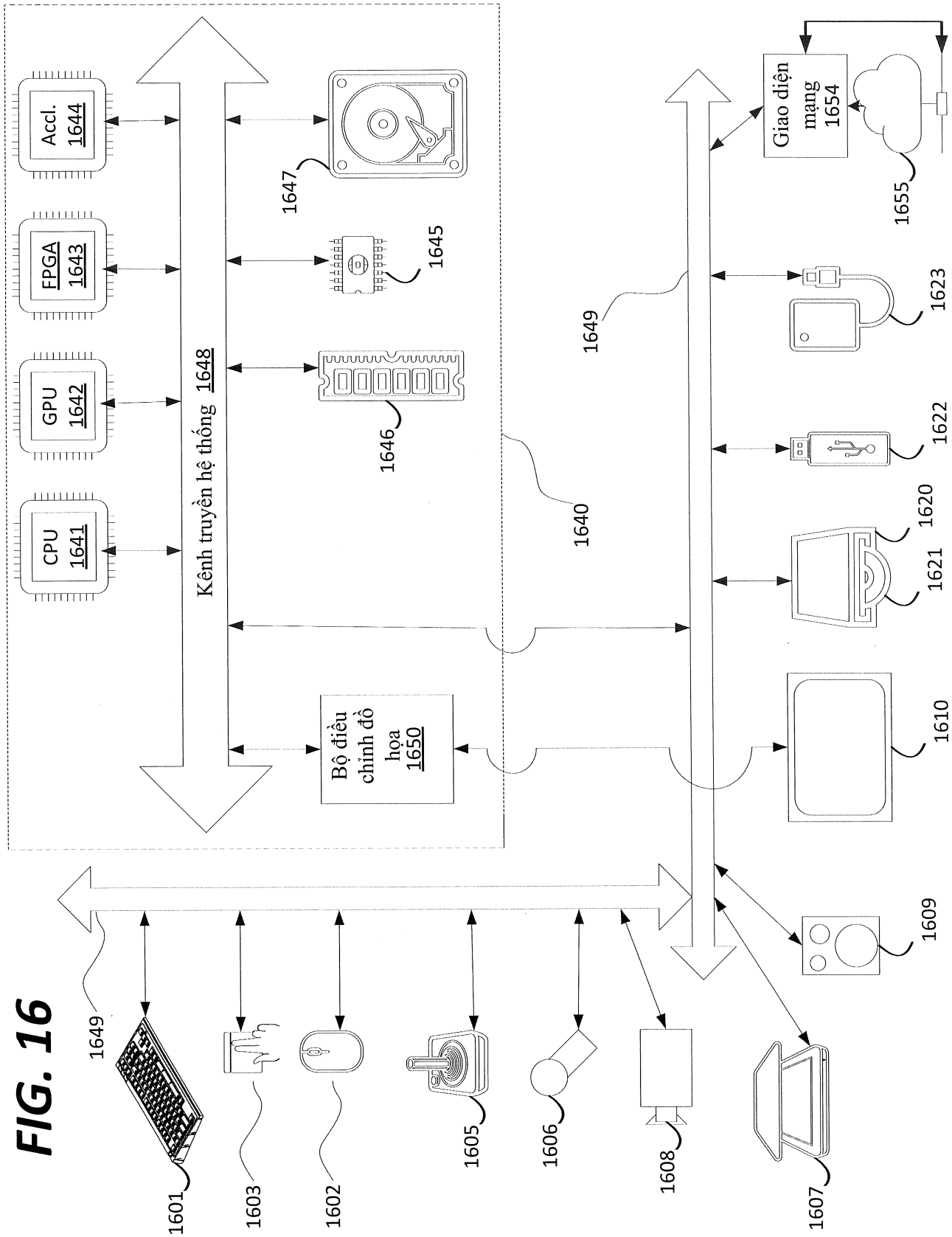

Accl. 1644
FPGA 1643
GPU 1642
CPU 1641
Kênh truyền hệ thống 1648
1647
1645
1646
Bộ điều chỉnh đồ họa 1650
1640
1649
Giao diện mạng 1654
1655
1623
1622
1620
1621
1610
1609
1601
1603
1602
1605
1606
1608
1607

FIG. 16