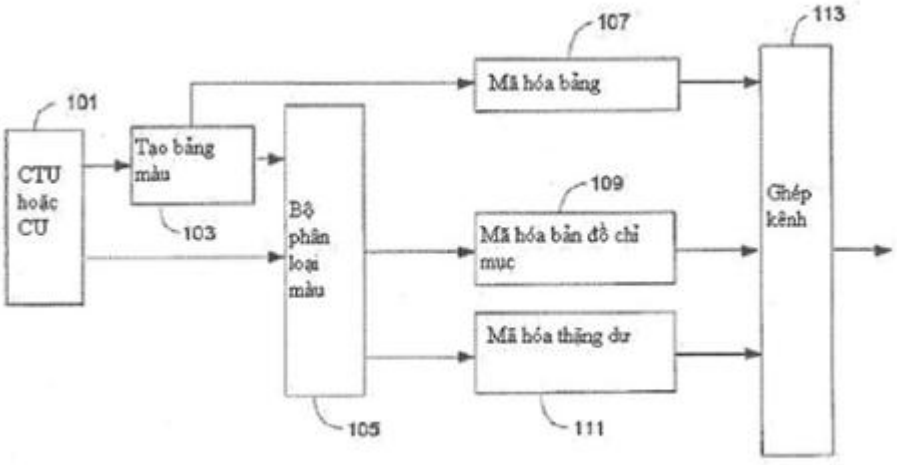
(12) BẢN MÔ TẢ SÁNG CHẾ THUỘC BẰNG ĐỘC QUYỀN SÁNG CHẾ
(19) **Cộng hòa xã hội chủ nghĩa Việt Nam (VN)**
CỤC SỞ HỮU TRÍ TUỆ
(11) **1-0029679**
(51)[7] **H04N 9/79**
(13) **B**

(21) 1-2017-00064
(22) 25/06/2015
(86) PCT/US2015/037779 25/06/2015
(87) WO2015/200690 30/12/2015
(30) 62/018,349 27/06/2014 US; 14/749,138 24/06/2015 US
(45) 25/10/2021 403
(43) 27/03/2017 348A
(73) HUAWEI TECHNOLOGIES CO., LTD. (CN)
Huawei Administration Building, Bantian, Longgang, Shenzhen, Guangdong 518129, China
(72) YU, Haoping (US); MA, Zhan (CN); WANG, Wei (CA); XU, Meng (CN).
(74) Công ty Luật TNHH Phạm và Liên danh (PHAM & ASSOCIATES)

(54) PHƯƠNG PHÁP VÀ THIẾT BỊ MÃ HÓA VÀ GIẢI MÃ NỘI DUNG MÀN HÌNH

(57) Sáng chế đề xuất thiết bị (100) được tạo cấu hình để thực hiện phương pháp (1700) để mã hóa nội dung màn hình. Phương pháp gồm bước suy ra (1701) bản đồ chỉ mục màu (311, 601, 1301, 1600) dựa trên CU (coding unit, khối mã hóa) hiện tại (101, 213, 401, 501). Phương pháp còn gồm mã hóa (1703) bản đồ chỉ mục màu, trong đó ít nhất một phần của bản đồ chỉ mục màu được mã hóa nhờ sử dụng kỹ thuật mã hóa thứ nhất, trong đó bộ chỉ báo thứ nhất biểu thị khoảng cách có nghĩa của kỹ thuật mã hóa thứ nhất. Phương pháp còn gồm tổ hợp (1705) bản đồ chỉ mục màu được mã hóa và bộ chỉ báo thứ nhất để truyền đến bộ nhận (200).



Lĩnh vực kỹ thuật được đề cập

Sáng chế chủ yếu liên đến mã hóa nội dung màn hình, và cụ thể là, đến việc mã hóa nội dung màn hình nâng cao với bảng màu cải tiến và mã hóa bản đồ chỉ mục.

Tình trạng kỹ thuật của sáng chế

Mã hóa nội dung màn hình đặt ra các thách thức mới để nén video do các đặc tính tín hiệu nổi bật so với các tín hiệu video truyền thống. Có nhiều kỹ thuật hiện tại để mã hóa nội dung màn hình nâng cao, chẳng hạn, giả so khớp chuỗi, mã hóa bảng màu, và bù trừ chuyển động bên trong hoặc sao chép khối bên trong. Trong số các kỹ thuật này, giả so khớp chuỗi thể hiện độ khuếch đại cao nhất để mã hóa không tổn hao, nhưng với chi phí bổ sung độ phức tạp đáng kể và nhiều khó khăn ở chế độ mã hóa tổn hao. Mã hóa bảng màu được phát triển cho nội dung màn hình với giả thiết rằng nội dung không được camera chụp (chẳng hạn, nội dung được máy tính tạo) thông thường gồm số lượng màu nổi bật bị giới hạn, thay vì các sắc màu liên tục hoặc gần liên tục được tìm thấy trong nhiều chuỗi video. Mặc dù giả so khớp chuỗi và các phương pháp mã hóa bảng màu cho thấy tiềm năng lớn, bù trừ chuyển động bên trong hoặc sao chép khối bên trong được đưa vào sử dụng phiên bản 4 WD (working draft, phác thảo) và phần mềm tham chiếu mở rộng khoảng mã hóa video hiệu dụng cao (High Efficiency Video Coding, HEVC) hiện tại để mã hóa nội dung màn hình. Tuy nhiên, hiệu năng mã hóa sao chép bên trong khối bị chặn do khai triển khối cố định. Việc thực hiện so khớp khối (tương tự việc ước tính chuyển động ở bên trong ảnh) còn tăng độ phức tạp bộ mã hóa đáng kể trên cả việc tính toán và truy nhập bộ nhớ.

Bản chất kỹ thuật của sáng chế

Theo một phương án thực hiện, phương pháp mã hóa nội dung màn hình được đề xuất. Phương pháp gồm bước suy ra bản đồ chỉ mục màu dựa trên CU (coding unit, khối mã hóa) hiện tại. Phương pháp còn gồm mã hóa bản đồ chỉ mục màu, trong đó ít nhất một phần của bản đồ chỉ mục màu được mã hóa nhờ sử dụng kỹ thuật mã hóa thứ nhất, trong đó bộ chỉ báo thứ nhất biểu thị khoảng cách có nghĩa của kỹ thuật mã hóa thứ nhất. Phương pháp còn gồm tổ hợp bản đồ chỉ mục màu được mã hóa và bộ chỉ báo thứ nhất để truyền đến bộ nhận.

Theo phương án thực hiện khác, phương pháp giải mã nội dung màn hình được đề xuất. Phương pháp gồm tiếp nhận dòng bit video bao gồm bản đồ chỉ mục màu. Phương pháp còn gồm tiếp nhận bộ chỉ báo thứ nhất. Phương pháp còn gồm giải mã ít nhất một phần của bản đồ chỉ mục màu nhờ sử dụng kỹ thuật giải mã thứ nhất, trong đó bộ chỉ báo thứ nhất biểu thị khoảng cách có nghĩa của kỹ thuật giải mã thứ nhất. Ngoài ra, phương pháp gồm tái tạo các điểm ảnh liên kết với CU hiện tại dựa trên bản đồ chỉ mục màu.

Các phương án thực hiện khác gồm các thiết bị được tạo cấu hình để thực hiện các phương pháp này.

Mô tả vắn tắt các hình vẽ

Để hiểu rõ hơn sáng chế, và các ưu điểm của nó, tham khảo phần mô tả sau cùng với các hình vẽ đi kèm, trong đó các số giống nhau chỉ định các đối tượng giống nhau, và trong đó:

Fig.1 minh họa sơ đồ khối chức năng của bộ truyền lấy làm ví dụ thực hiện quá trình mã hóa nội dung màn hình theo sáng chế;

Fig.2 minh họa sơ đồ khối chức năng của bộ nhận lấy làm ví dụ thực hiện quá trình giải mã nội dung màn hình theo sáng chế;

Fig.3 minh họa ví dụ của các môđun khác nhau và luồng xử lý bằng cách sử dụng bảng màu và bản đồ chỉ mục, theo sáng chế;

Fig.4 minh họa CU lấy làm ví dụ với các thành phần màu được thể hiện riêng rẽ và đóng gói;

Fig.5A minh họa bảng màu tham chiếu và bảng màu hiện tại sử dụng trong quá trình mã hóa nội dung màn hình;

Fig.5B minh họa ví dụ dự báo bảng màu nhờ sử dụng các khối tái tạo lân cận;

Fig.6 minh họa bản đồ chỉ mục màu lấy làm ví dụ cho CU 64x64 trong đó việc quét theo phương ngang hoặc thẳng đứng có thể được sử dụng;

Fig.7 minh họa một phần vectơ chỉ mục màu một chiều (1D) sau phép tìm kiếm 1D nhờ sử dụng quét theo phương ngang;

Fig.8 minh họa ví dụ của khối xử lý điểm ảnh cơ bản, gọi là môđun U_PIXEL;

Fig.9 minh họa ví dụ của môđun U_ROW;

Fig.10 minh họa ví dụ của môđun U_CMP;

Fig.11 minh họa ví dụ của môđun U_COL;

Fig.12 minh họa môđun U_2D_BLOCK lấy làm ví dụ;

Fig.13 minh họa các ví dụ quét theo phương thẳng đứng và phương nằm ngang để xử lý bản đồ chỉ mục;

Fig.14A và Fig.14B minh họa các vị dụ định dạng lấy mẫu sắc độ 4:2:0 và 4:4:4;

Fig.15 minh họa ví dụ của quá trình nội suy từ 4:4:4 đến 4:2:0 và ngược lại;

Fig.16 minh họa ví dụ xử lý bản đồ chỉ mục màu sử dụng bộ đệm dòng chỉ mục trên hoặc bộ đệm dòng chỉ mục trái;

Fig.17 minh họa phương pháp mã hóa nội dung màn hình theo sáng chế; và

Fig.18 minh họa phương pháp giải mã nội dung màn hình theo sáng chế.

Mô tả chi tiết sáng chế

Fig.1 đến Fig.18, nêu dưới đây, và các phương án thực hiện khác nhau được sử dụng để mô tả các nguyên lý của sáng chế trong bản mô tả chỉ nhằm minh họa và không được hiểu theo cách bất kỳ để giới hạn phạm vi của sáng chế. Người có hiểu biết trung bình trong lĩnh vực sẽ hiểu rằng các nguyên lý của

sáng chế có thể được triển khai ở loại thiết bị hoặc hệ thống được bố trí thích hợp.

Các tài liệu và các mô tả chuẩn sau được đưa vào sáng chế:

T. Lin, S. Wang, P. Zhang, K. Zhou, "AHG7: Full-chroma (YUV444) dictionary+hybrid dual-coder extension of HEVC", JCT-VC Document, JCTVC-K0133, Shanghai, China, October 2012 (ở đây gọi là "REF1");

W. Zhu, J. Xu, W. Ding, "RCE3 Test 2: Multi-stage Base Color and Index map", JCT-VC Document, JCTVC-N0287, Vienna, Austria, July 2013 (ở đây gọi là "REF2");

L. Guo, M. Karczewicz, J. Sole, "RCE3: Results of Test 3.1 on Palette Mode for screen content coding", JCT-VC Document, JCTVC-N0247, Vienna, Austria, July 2013 (ở đây gọi là "REF3");

L. Guo, M. Karczewicz, J. Sole, R. Joshi, "Non-RCE3: Modified Palette Mode for screen content coding", JCT-VC Document, JCTVC-N0249, Vienna, Austria, July 2013 (ở đây gọi là"REF4");

D.-K. Kwon, M. Budagavi, "RCE3: Results of test 3.3 on Intra motion compensation, JCT-VC Document, JCTVC-N0205, Vienna, Austria, July 2013 (ở đây gọi là "REF5");

C. Pang, J. Sole, L. Guo, M. Karczewicz, R. Joshi, "Non-RCE3: Intra Motion Compensation with 2-D MVs", JCT-VC Document, JCTVC-N0256, Vienna, Austria, July 2013 (ở đây gọi là "REF6");

C. Pang, J. Sole, L. Guo, M. Karczewicz, R. Joshi, "Non-RCE3: Pipeline Friendly Intra Motion Compensation", JCT-VC Document, JCTVC-N0254, Vienna, Austria, July 2013 (ở đây gọi là "REF7");

D. Flynn, J. Soel and T. Suzuki, "Range Extension Draft 4", JCTVC-L1005, August 2013 (ở đây gọi là "REF8"); và

H. Yu, K. McCann, R. Cohen, và P. Amon, "Draft call for proposals for coding of screen content and medical visual content", ISO/IEC JTC1/SC29/WG11 N13829, July 2013 (ở đây gọi là "REF9").

Các phương án thực hiện sáng chế đề xuất quá trình mã hóa nội dung màn hình nâng cao với bảng màu cải tiến và mã hóa bản đồ chỉ mục. Các phương án thực hiện tốt hơn đáng kể HEVC phiên bản 2. Các phương án thực hiện gồm nhiều thuật toán cụ thể để mã hóa nội dung màn hình. Các thuật toán này gồm biểu diễn điểm ảnh nhờ sử dụng bảng màu, nén bảng màu, nén bản đồ chỉ mục màu, so khớp chuỗi, và nén dư. Các phương án thực hiện nêu ở đây được phát triển, hài hòa, và tích hợp với Rext (HEVC Range Extension, khoảng HEVC mở rộng) như là các phần mở rộng HEVC tương lai để hỗ trợ mã hóa nội dung màn hình hiệu quả. Tuy nhiên, các phương án thực hiện có thể được triển khai bổ sung hoặc theo cách khác với các chuẩn video hiện tại hoặc các chuẩn video thích hợp khác bất kỳ. Để dễ giải thích, HEVC REext được sử dụng ở đây làm ví dụ để mô tả các phương án thực hiện khác. Một cách tương tự, phần mềm HEVC REext được sử dụng để triển khai các phương án thực hiện khác để thể hiện hiệu suất nén.

Fig.1 minh họa sơ đồ khối chức năng của bộ truyền lấy làm ví dụ thực hiện quá trình mã hóa nội dung màn hình theo sáng chế. Fig.2 minh họa sơ đồ khối chức năng của bộ nhận lấy làm ví dụ thực hiện quá trình giải mã nội dung màn hình theo sáng chế. Bộ truyền 100 và bộ nhận 200 theo các phương án thực hiện chỉ minh họa. Theo các phương án thực hiện khác, bộ truyền 100 và bộ nhận 200 có thể được sử dụng mà không xa rời phạm vi sáng chế.

Bộ truyền 100 được tạo cấu hình để thực hiện quá trình nén bảng màu (color palette compression, CPC) hiệu suất cao có thể được thực hiện trên mỗi CU hoặc khối cây mã hóa (coding tree unit, CTU) trong dòng bit. Như được thể hiện trên Fig.1, bộ truyền 100 bắt đầu với CU 101 trong dòng bit. CU là khối hoạt động cơ bản trong HEVC và HEVC REext, và là khối điểm ảnh bình phương gồm ba thành phần màu (chẳng hạn, RGB, YUV, XYZ, hoặc tương tự, như theo giải pháp đã biết). CU 101 lấy làm ví dụ được thể hiện trên Fig.3. CU 101 là CU 8 điểm ảnh x 8 điểm ảnh gồm giá trị màu tường minh (chẳng hạn, 47, 48, 49, v.v.) cho mỗi điểm ảnh. Theo các phương án thực hiện khác, kích thước của CU 101 có thể khác ngoài 8x8 điểm ảnh (chẳng hạn, 16x16 điểm ảnh,

32x32 điểm ảnh, v.v.). Theo một số phương án thực hiện, bộ truyền 100 có thể bắt đầu với CTU 101 thay vì CU 101. Để dễ giải thích, bộ truyền 100 sẽ được mô tả với CU 101. Chuyên gia trong lĩnh vực sẽ hiểu rằng bộ truyền 100 có thể thực hiện quá trình tương tự với CTU 101.

Khối tạo bảng màu 103 sử dụng CU 101 để suy ra hoặc tạo bảng màu (đôi lúc được gọi là bảng màu). Bảng màu 303 lấy làm ví dụ được thể hiện trên Fig.3. Để suy ra bảng màu 303, khối tạo bảng màu 103 sắp thứ tự các giá trị màu theo một hoặc nhiều quy tắc sắp xếp. Bảng màu 303 có thể được sắp thứ tự theo tần số xuất hiện của mỗi giá trị màu, cường độ màu thực của mỗi điểm ảnh của CU 101, hoặc mêtric sắp xếp thích hợp khác bất kỳ, để tăng hiệu suất các hoạt động mã hóa sau.

Dựa trên bảng màu được dẫn xuất 303, khối phân loại màu 105 sử dụng CU 101 để gán các màu hoặc các giá trị điểm ảnh của CU 101 vào bản đồ chỉ mục màu 311 và một hoặc nhiều sơ đồ dư dự báo 313. Khối mã hóa bảng 107 tiếp nhận bảng màu 303 và mã hóa các mục trong bảng màu 303. Khối mã hóa bản đồ chỉ mục 109 mã hóa bản đồ chỉ mục màu 311 được tạo bởi khối bộ phân loại màu 105. Các hoạt động này được mô tả chi tiết hơn dưới đây.

Khối mã hóa dư 111 mã hóa mỗi bản đồ thặng dư dự báo 313 được tạo bởi khối bộ phân loại màu 105. Theo một số phương án thực hiện, khối mã hóa dư 111 thực hiện nhị phân hóa thặng dư độ dài biến thiên hoặc độ dài cố định, như được biểu thị ở 321 trên Fig.3. Sau đó, khối ghép kênh (MUX) 113 tạo dòng bít nén nhờ sử dụng các phép so khớp chuỗi/khối 319 và các thặng dư dự báo mã hóa 321. Theo một số phương án thực hiện, phương pháp CABAC (context adaptive binary arithmetic coding, mã hóa số học nhị phân thích ứng ngữ cảnh) 323 có thể được sử dụng để kết hợp các phép so khớp chuỗi/khối 319 và các thặng dư dự báo mã hóa 321, như được thể hiện trên Fig.3.

Quay lại Fig.2, bộ nhận 200 được tạo cấu hình để thực hiện quá trình giải mã nội dung màn hình tương tự quá trình mã hóa nội dung màn hình được thực hiện bởi bộ truyền 100, như được mô tả trên đây. Bộ nhận 200 tiếp nhận dòng bit video nén, và sau đó, nhờ sử dụng bộ giải ghép kênh 201, phân tích cú pháp

dòng bit thành bảng màu được mã hóa, bản đồ chỉ mục màu, và các thặng dư dự báo mã hóa. Khối giải mã bảng 203 và khối tạo bảng màu 209 thực hiện các quá trình ngược với khối mã hóa bảng 107 và khối tạo bảng màu 103 để tái tạo, cho mỗi CU, bảng màu hoàn chỉnh. Tương tự, khối giải mã bản đồ chỉ mục 205 và khối giải mã thặng dư 207 thực hiện các quá trình ngược với khối mã hóa sơ đồ chỉ mục 109 và khối mã hóa dư 111 để tái tạo bản đồ chỉ mục màu. Khối giải phân loại màu 211 suy ra giá trị điểm ảnh ở mỗi vị trí nhờ kết hợp bản đồ chỉ mục màu và bảng màu, nhờ đó tái tạo CTU hoặc CU 213.

Mặc dù Fig.1 và Fig.2 minh họa các ví dụ của bộ truyền 100 và bộ nhận 200 để thực hiện giải mã và mã hóa nội dung màn hình, có thể thực hiện các thay đổi khác nhau cho Fig.1 và Fig.2. Chẳng hạn, các thành phần khác nhau trên Fig.1 và Fig.2 có thể được kết hợp, được phân chia nhỏ hơn, hoặc được bỏ qua và các thành phần bổ sung có thể được thêm vào theo các nhu cầu thực. Như là ví dụ cụ thể, các thành phần khác nhau có thể được bố trí cùng nhau trong một thân hoặc trên một bảng mạch, hoặc được thực hiện bởi một bộ xử lý hoặc khối xử lý.

Dựa trên bảng màu được dẫn xuất 303, mỗi điểm ảnh trong CU 101 gốc có thể được biến đổi thành chỉ mục màu trong bảng màu 303. Các phương án thực hiện sáng chế đề xuất các phương pháp để nén hiệu quả bảng màu 303 và bản đồ chỉ mục màu 311 (được mô tả dưới đây) cho mỗi CU 101 thành dòng. Ở phía bộ nhận, dòng bit được nén có thể được phân tích cú pháp để tái tạo, cho mỗi CU 101, bảng màu 303 hoàn chỉnh và bản đồ chỉ mục màu 311, và sau đó còn suy ra giá trị điểm ảnh ở mỗi vị trí bằng cách tổ hợp chỉ mục màu và bảng màu.

Fig.4 minh họa ví dụ khác của CU 401 với các thành phần mục được thể hiện riêng rẽ và gói gọn. CU 401 có thể đại diện CU 101. Như được thể hiện trên Fig.4, CU 401 là CU 8 điểm ảnh x 8 điểm ảnh. Dĩ nhiên, CU 401 có thể là điểm ảnh NxN, trong đó N=8, 16, 32, 64 để tương thích với HEVC. Mỗi điểm ảnh của CU 401 gồm ba thành phần màu, ở các tỷ lệ lấy mẫu khác nhau (chẳng hạn, 4:4:4, 4:2:2, 4:2:0). Tức là, CU 401 gồm các thành phần màu đỏ (R) 402,

thành phần màu xanh lá (G) 403, và thành phần màu xanh đậm (B) 404 riêng rẽ. Theo các phương án thực hiện khác, các thành phần màu có thể là Y, Cb, Cr, hoặc X, Y Z hoặc tổ hợp các thành phần thích hợp khác.

Để đơn giản, các chuỗi 4:4:4 được sử dụng trong bản mô tả. Đối với các video 4:2:2 và 4:2:0, việc lấy mẫu lên sắc độ có thể được áp dụng để thu được các chuỗi 4:4:4, hoặc mỗi thành phần sắc độ 402-404 có thể được xử lý độc lập. Trong trường hợp của các video đơn sắc 4:0:0, có thể được xử lý làm mặt phẳng riêng rẽ 4:4:4 không có hai mặt phẳng còn lại. Tất cả các phương pháp cho 4:4:4 có thể được áp dụng trực tiếp.

Các thành phần màu 402-404 có thể được đan xen cùng nhau trong quá trình đóng gói, thu được CU 401 đóng gói. Theo phương án thực hiện, cờ enable_packed_component_flag được định nghĩa cho mỗi CU 101 để chỉ báo liệu CU 101 có được xử lý nhờ sử dụng chế độ đóng gói (do vậy thu được CU 401) hoặc chế độ phẳng đã biết (tức là, các thành phần G, B, R hoặc Y, U, V 402-404 được xử lý độc lập).

Cả chế độ đóng gói lẫn chế độ phẳng có thể có các ưu điểm và các nhược điểm. Chẳng hạn, chế độ phẳng hỗ trợ xử lý thành phần màu song song cho G/B/R hoặc Y/U/V. Tuy nhiên, chế độ phẳng có thể dẫn đến hiệu suất mã hóa thấp. Chế độ đóng gói có thể chia sẻ thông tin tiêu đề (như bảng màu 303 và bản đồ chỉ mục màu 311) cho CU 101 trong số các thành phần màu khác nhau. Tuy nhiên, chế độ đóng gói có thể ngăn không cho nhiều thành phần màu được xử lý đồng thời hoặc theo kiểu song song. Một phương pháp đơn giản để quyết định xem liệu CU hiện tại 101 có nên được mã hóa trong chế độ đóng gói để đo chi phí R-D (rate distortion, méo tốc độ).

enable_packed_component_flag được sử dụng để báo hiệu tường minh chế độ mã hóa cho bộ giải mã. Bên cạnh việc định nghĩa enable_packed_component_flag ở mức CU để xử lý mức thấp, cờ có thể được sao chép ở tiêu đề phiến hoặc thậm chí mức chuỗi (chẳng hạn, tập hợp tham số chuỗi hoặc tập hơn tham số ảnh) để cho phép xử lý mức phiến hoặc mức chuỗi, tùy thuộc vào yêu cầu ứng dụng cụ thể.

Bảng màu và dẫn xuất bản đồ chỉ mục

Phần sau mô tả các hoạt động ở khối tạo bảng màu 103 và khối mã hóa bảng 107 trên Fig.1. Đối với mỗi CU 101, các vị trí điểm ảnh được cắt ngang và bảng màu 303 và bản đồ chỉ mục màu 311 để xử lý tuần tự được suy ra. Mỗi màu nổi bật được sắp thứ tự trong bảng màu 303, tùy thuộc vào biểu đồ của nó (tức là, tần số xuất hiện), hoặc cường độ, hoặc phương pháp tùy ý bất kỳ để tăng hiệu suất của quá trình mã hóa tiếp theo. Chẳng hạn, nếu quá trình mã hóa sử dụng phương pháp điều biến mã xung vi phân (differential pulse code modulation, DPCM) để mã hóa hiệu số giữa các điểm ảnh liền kề, kết quả mã hóa tối ưu có thể thu được nếu các điểm ảnh liền kề được gán với chỉ mục màu liền kề trong bảng màu 303.

Dẫn xuất bảng màu dựa trên băm mới sẽ được mô tả, có thể được sử dụng để xác định hiệu quả các màu chính và giảm lỗi. Đối với mỗi CU 101, khối tạo bảng màu 103 kiểm tra giá trị màu của mỗi điểm ảnh trong CU 101 và tạo biểu đồ màu nhờ sử dụng ba thành phần màu cùng nhau, tức là, G, B, R đóng gói hoặc Y, Cb, Cr đóng gói theo tần số xuất hiện của mỗi màu theo thứ tự giảm dần. Để biểu thị mỗi màu 24 bit, các thành phần màu G và B (hoặc các thành phần màu Y và Cb) có thể được dịch chuyển bit tương ứng. Tức là, mỗi màu được đóng gói có thể được đại diện theo giá trị (G<<16)+(B<<8)+(R) hoặc (Y<<16)+(Cb<<8)+(Cr), trong đó <<x là toán tử dịch bit trái. Biểu đồ được sắp xếp theo tần số xuất hiện màu theo thứ tự giảm.

Để mã hóa tổn hao, sau đó khối tạo bảng màu 103 áp dụng quá trình nhóm màu lân cận dựa vào băm trên dữ liệu màu được sắp thứ tự bằng biểu đồ để có biểu diễn bảng màu nhỏ gọn hơn. Đối với mỗi thành phần màu, X bit trọng số nhỏ nhất (tùy thuộc vào QP (quantization parameter, tham số lượng tử hóa) được xóa và biểu diễn băm tương ứng được tạo nhờ sử dụng hàm băm (G>>X<<(16+X))|(B>>X<<(8+X))|(R>>X<<X) hoặc (Y>>X<<(16+X))|(Cb>>X<<(8+X))|(Cr>>X<<X), trong đó >>x là toán tử dịch bit sang phải, và X được xác định dựa trên QP. Bảng băm hoặc theo cách khác cấu trúc dữ liệu BST (binary search tree, cây tìm kiếm nhị phân) được

khai thức để tìm kiếm nhanh các màu có chung giá trị băm. Đối với hai giá trị băm bất kỳ, khoảng cách của nó được định nghĩa là hiệu số tuyệt đối lớn nhất của các thành phần màu tương ứng.

Trong quá trình nhóm màu lân cận, khối tạo bảng màu 103 xử lý các màu được đóng gói theo thứ tự giảm của tần số xuất hiện, cho đến khi N màu được xử lý. Nếu số lượng màu ở CU hiện tại nhỏ hơn N, thì tất cả các màu trong CU hiện tại được xử lý. N được giới hạn bởi số màu lớn nhất định trước (max_num_of_colors). Theo một số phương án thực hiện, max_num_of_colors = 128, tức là, N <= 128. Sau khi nhóm màu dựa vào băm, N màu được chọn (hoặc tất cả các màu trong trường hợp mà số lượng màu ở CU hiện tại nhỏ hơn N), sau đó được sắp thứ tự lại bằng cách sắp xếp màu theo thứ tự tăng dựa trên giá trị của mỗi màu được đóng gói. Kết quả là bảng màu như bảng màu 303 được thể hiện trên Fig.3. Bảng màu 303 có kích thước bốn màu (tức là, N = 4). Theo nhiều phương án thực hiện, N > 4. Tuy nhiên, để dễ giải thích, N được lựa chọn bằng 4 trên Fig.3.

Khi số lượng màu được đại diện ở CU 101 lớn hơn số lượng màu N trong bảng màu 303, các màu xuất hiện với tần số nhỏ hơn được bố trí như là các thặng dư ngoài bảng màu 303. Chẳng hạn, các giá trị màu 49, 53, 50, và 51 là một phần bảng màu 303, trong khi các giá trị màu 48, 52, 47, 54, 55, và 56 là các màu thặng dư 305 ngoài bảng màu 303.

Dẫn xuất bảng màu 303, như được thực hiện bởi khối tạo bảng màu 103, có thể được mô tả bằng giả mã dưới đây.

(Giả mã):

```
H = DeriveHistogram();
H' = CreateEmptyHistorgram();
processed_color_count = 0;
while( processed_color_count < N and H is not empty )
{
 C = GetMostFrequentColor( H );
 if( lossy coding )
```

```
    {
      hash = ComputeHash( C, QP );
      find all colors Cx satisfying: dist( hash, ComputeHash( Cx, QP ) ) <= 1;
      merge all colors in Cx to C;
      remove Cx from H;
    }
     save C to H' and remove C from H;
  }
  H = H';
  Reorder( H );
```

Trong đoạn giả mã trên đây, ComputeHash(C, QP) áp dụng hàm băm (G>>X<<(16+X))|(B>>X<<(8+X))|(R>>X<<X) hoặc (Y>>X<<(16+X))|(Cb>>X<<(8+X))|(Cr>>X<<X) để tạo giá trị băm, trong đó X phụ thuộc vào QP. Dist(hash1, hash2) thu được hiệu số tuyệt đối lớn nhất của các thành phần màu tương ứng trong hash1 và hash2. Ở đây, các cấu trúc dữ liệu bảng băm và BTS được tận dụng để tìm nhanh các màu thỏa mãn điều kiện cụ thể dựa trên giá trị băm của nó.

Như nêu trên, dựa trên bảng màu được suy ra 303, khối bộ phân loại màu 105 sử dụng CU 101 để gán các màu hoặc các giá trị điểm ảnh của CU 101 vào bản đồ chỉ mục màu 311 và một hoặc nhiều bản đồ thặng dư dự báo 313. Tức là, khối bộ phân loại màu 105 gán mỗi màu trong bảng màu 303 cho chỉ mục màu trong bảng màu 303. Chẳng hạn, như được biểu thị ở 307 trên Fig.3, màu 49 được gán chỉ mục màu 0 (ColorIdx = 0), màu 53 được gán chỉ mục màu 1, màu 50 được gán chỉ mục màu 2, và màu 51 được gán chỉ mục màu 3 (ColorIdx = 3). Khi các màu trong bảng màu 303 được gán chỉ mục, bản đồ chỉ mục màu 311 có thể được tạo từ CU 101 nhờ sử dụng các chỉ mục của mỗi màu. Việc xử lý bản đồ chỉ mục màu 311 được mô tả chi tiết hơn dưới đây. Tương tự, mỗi màu thặng dư 305 ngoài bảng màu 303 được gán giá trị thặng

dư dự báo, như được biểu thị ở 309. Khi các màu thặng dư 305 được gán giá trị thặng dư dự báo, bản đồ thặng dư dự báo 313 có thể được tạo từ CU 101.

Đối với CU phẳng, mỗi thành phần màu có thể có bảng màu riêng, như colorTable_Y, colorTable_U, colorTable_V hoặc colorTable_R, colorTable_G, colorTable_B. Theo một số phương án thực hiện, bảng màu đối với thành phần chính có thể được dẫn xuất, như Y trong YUV hoặc G trong GBR, và bảng này có thể được chia sẻ cho tất cả các thành phần. Thông thường, bằng cách sử dụng bảng màu Y hoặc G được chia sẻ, các thành phần màu khác ngoài Y hoặc G sẽ có so khớp sau tương đối với các màu điểm ảnh gốc từ các màu trong bảng màu được chia sẻ. Sau đó cơ cấu thặng dư (như các phương pháp mã hóa các hệ số HEVC) có thể được áp dụng để mã hóa các thặng dư so khớp lệch đó. Theo các phương án thực hiện khác, đối với CU được đóng gói, một bảng màu có thể được chia sẻ trong số tất cả các thành phần.

Giả mã sau lấy bảng màu và dẫn xuất bản đồ chỉ mục làm ví dụ minh họa.

(Giả mã):

```
deriveColorTableIndexMap()
{
  deriveColorTable();
  deriveIndexMap();
}

  deriveColorTable(src, cuWidth, cuHeight, maxColorNum)
  {
    // src – input video source in planar or packed mode
    // cuWidth, cuHeight – width and height of current CU
    /* maxColorNum – max num of colors allowed in palette table*/
    /*transverse */
    //
    //   memset(colorHist, 0, (1<<bitDepth)*sizeof(UINT))
    pos=0;
```

```
cuSize=cuWidth*cuHeight;
while (pos<cuSize) {
   colorHist[src[pos++]]++;
}

/*just pick non-zero entry in colorHist[] for color intensity ordered table*/
j=0;
for(i=0;i<(1<<bitDepth);i++)
{
  if(colorHist[i]!=0)
    colorTableIntensity[j++] = colorHist[i];
 }
 colorNum=j;
      /*quicksort for histgram*/
     colorTableHist = quickSort(colorTableIntensity, colorNum);
 /*nếu maxColorNum >= colorNum, all colors will be picked*/
 /*if maxColorNum < colorNum, only maxColorNum colors will be picked for colorTableHist. Trong trường hợp này, tất cả các điểm ảnh sẽ tìm thấy màu được so khớp tốt nhất của nó và chỉ mục tương ứng có khác biệt (điểm ảnh thực và màu tương ứng) được mã hóa bằng cơ cấu thặng dư.*/
 /*Số màu tốt nhất trong bảng màu có thể được xác định bằng cách dẫn xuất chi phí R-D lặp lại!*/
}
deriveIndexMap()
{
  pos=0;
  cuSize=cuWidth*cuHeight;
  while ( pos < cuSize)
  {
```

```
            minErr=MAX_UINT;
            for (i=0;i<colorNum;i++)
            {
              err = abs(src[pos] – colorTable[i]);
              if (err<minErr)
              {
                minErr = err;
                idx = i;
              }
            }
            idxMap[pos] = idx;
        }
    }
```

Xử lý bảng màu

Đối với mỗi CU 101, bộ truyền 100 có thể suy ra bảng màu 303 từ CU hiện tại 101 (được gọi là bàn trượt bảng màu tường minh) hoặc bộ truyền 100 có thể suy ra bảng màu 303 từ lân cận trên hoặc bên trái của CU hiện tại 101 (được gọi là bàn trượt bảng màu ngầm). Khối mã hóa bảng 107 tiếp nhận bảng màu 303 và mã hóa các mục trong bảng màu 303.

Việc xử lý bảng màu liên quan việc mã hóa kích thước của bảng màu 303 (tức là, tổng số màu nổi bật) và chính mỗi màu. Đa số các bit được tiêu thụ bằng cách mã hóa mỗi màu trong bảng màu 303. Ở đây, tiêu điểm sẽ được đặt vào việc mã hóa màu (tức là, mã hóa mỗi mục trong bảng màu 303).

Phương pháp dễ nhất để mã hóa các màu trong bảng màu là sử dụng thuật toán kiểu PCM (pulse code modulation, điều biến mã xung), trong đó mỗi màu được mã hóa độc lập. Theo cách khác, dự báo gần nhất cho màu kế tiếp có thể được áp dụng, và sau đó delta dự báo có thể được mã hóa thay vì cường độ màu mặc định, vốn là kiểu PCM vi phân (differential PCM, DPCM). Sau đó, cả hai phương pháp có thể được mã hóa entropy bằng cách sử dụng mô hình xác

suất cân bằng hoặc mô hình ngữ cảnh thích ứng, tùy thuộc vào cân bằng giữa chi phí độ phức tạp và hiệu suất mã hóa.

Các phương án thực hiện sáng chế đề xuất phương tiện cải tiến khác, gọi là hợp nhất bảng màu lân cận, trong đó color_table_merge_flag được định nghĩa để chỉ báo liệu CU hiện tại (chẳng hạn, CU 101) sử dụng bảng màu liên kết với lân cận CU trái hoặc lân cận CU trên. Nếu không, CU hiện tại mang báo hiệu bảng màu tường minh. Quá trình này có thể còn được gọi là chia sẻ bảng màu lân cận. Với quá trình hợp nhất, cờ color_table_merge_direction biểu thị hướng hợp nhất, vốn là từ CU trên hoặc từ CU trái. Dĩ nhiên, các ứng viên hướng hợp nhất có thể theo các hướng khác ngoài CU trên hoặc CU trái (chẳng hạn, trái trên, phải trên, và tương tự). Tuy nhiên, CU trên và CU trái được sử dụng trong sáng chế để minh họa khái niệm. Mỗi điểm ảnh trong CU hiện tại được so sánh với các mục trong bảng màu hiện tại cùng với CU trái hoặc CU trên và được gán chỉ mục tạo ra hiệu số dự báo nhỏ nhất (tức là, điểm ảnh trừ màu gần nhất trong bảng màu) thông qua giả mã deriveIdxMap() được thể hiện trên đây. Đối với trường hợp trong đó hiệu số dự báo khác 0, tất cả các thặng dư được mã hóa sử dụng cơ cấu thặng dư RExt. Quyết định liệu có sử dụng quá trình hợp nhật bảng có thể được xác định bởi chi phi R-D.

Khi bảng màu được mang tường minh trong dòng bit, có thể được mã hóa tuần tự cho mỗi thành phần màu. Việc nhồi bảng màu liên bảng hoặc DPCM màu bên trong bảng được áp dụng như được mô tả dưới đây để mã hóa mỗi mục tuần tự cho tất cả ba thành phần màu.

Nhồi bảng màu liên bảng

Thậm chí khi phương pháp chia sẻ bảng màu không được sử dụng, vẫn có thể có màu chung giữa bảng màu 303 và bộ dự báo bảng màu. Do vậy, việc áp dụng kỹ thuật nhồi bảng màu liên bảng theo từng mục có thể còn cải thiện hiệu quả mã hóa. Ở đây, bộ dự báo bảng màu được suy ra từ khối lân cận, như CU lân cận trái hoặc CU lân cận bên trên. Fig.5A minh họa bộ dự báo bảng màu 551 và bảng màu hiện tại 553 có thể được sử dụng với kỹ thuật nhồi bảng màu

liên bảng theo sáng chế. Bảng màu 553 hiện tại có thể đại diện bảng màu 303 trên Fig.3. Bộ dự báo bảng màu 551 có thể được xây dựng từ CU lân cận trái của CU hiện tại. Ở phía bộ giải mã, bảng màu được cập nhật thích hợp theo bộ dự báo bảng màu 551 từ các lân cận tham chiếu. Theo một số phương án thực hiện, bộ dự báo bảng màu có thể được suy ra từ CU lân cận được tái tạo hoặc CTU hoặc từ bảng toàn cục ở phiến hoặc mức chuỗi. Theo giải pháp kỹ thuật đã biết, phiến gồm nhiều CU trong ảnh. Ảnh có thể gồm một hoặc nhiều phiến. Chuỗi gồm nhiều phiến.

Cho c(i) và r(j) lần lượt đại diện mục thứ i trong bảng màu hiện tại 553 và mục thứ j trong bộ dự báo bảng màu 551. Lại lưu ý rằng mỗi mục chứa ba thành phần màu (GBR, YCbCr, hoặc tương tự). Đối với mỗi mục màu c(i), i<=N, trong bảng hiện tại 553, khối mã hóa bảng 107 tìm thấy so khớp giống hệt r(j) từ bộ dự báo bảng màu 551. Thay vì báo hiệu c(i), j được mã hóa dự báo. Bộ dự báo được xác định làm chỉ số nhỏ nhất k lớn hơn j được tái tạo trước đó và thỏa mãn r(k)[0] > = c(i – 1)[0]. Hiệu số dự báo (j – k) được báo hiệu trong dòng bit. Do hiệu số (j – k) là dương, không cần bit dấu.

Lưu ý rằng mô hình thích ứng ngữ cảnh hoặc mô hình đi vòng có thể được sử dụng để mã hóa (j-k), theo giải pháp kỹ thuật đã biết. Thông thường, mô hình thích ứng ngữ cảnh được sử dụng cho các mục đích hiệu suất cao trong khi mô hình đi vòng được sử dụng cho yêu cầu thông cao và độ phức tạp thấp. Theo một số phương án thực hiện sáng chế, hai mô hình thích ứng ngữ cảnh có thể được sử dụng để mã hóa hiệu số dự báo chỉ mục (j–k), nhờ sử dụng phương tiện nhị phân hóa một ngôi nón cụt động .

DPCM màu bên trong bảng

Nếu không tìm thấy so khớp trong bộ dự báo bảng màu 551 cho mục thứ i trong bảng màu hiện tại 553, giá trị của mục thứ i được trừ khỏi mục trước mục thứ (i-1) và hiệu số tuyệt đối (|d(i)|) được mã hóa sử dụng DPCM màu cho mỗi thành phần. Nói chung, ít bit hơn cho hiệu số dự báo tuyệt đối và bit dấu sẽ được tạo và mã hóa nhờ sử dụng DPCM màu bên trong bảng. Mô hình thích

ứng ngữ cảnh hoặc mô hình đi vòng có thể được sử dụng để mã hóa hiệu số dự báo tuyệt đối và bit dấu liên kết, theo giải pháp kỹ thuật đã biết. Ngoài ra, bit dấu có thể bị giấu hoặc không được mã hóa cho một số trường hợp. Chẳng hạn, cho trước bảng màu hiện tại 553 đã được sắp thứ tự theo thứ tự tăng, hiệu số thành phần Y (hoặc G) không yêu cầu bit dấu. Tương tự, hiệu số thành phần Cb (hoặc B) không cần bit dấu nếu hiệu số Y (hoặc G) tương ứng bằng 0. Ngoài ra, hiệu số thành phần Cr (hoặc R) không cần bit dấu nếu cả hiệu số Y (hoặc G) lẫn Cb (or B) bằng 0. Như là ví dụ khác, bit dấu có thể được che giấu nếu hiệu số tuyệt đối bằng 0. Như là ví dụ khác nữa, bit dấu có thể được che giấu nếu điều kiện biên sau được thỏa mãn: $c[i-1] - |d(i)| < 0$ hoặc $c[i-1] + |d(i)| > 255$.

Đối với mục thứ nhất c(0) của bảng hiện tại 553, nếu kỹ thuật nhồi bảng màu liên bảng không được sử dụng, mỗi thành phần c(0) có thể được mã hóa nhờ sử dụng mô hình ngữ cảnh đi vòng 8 bit cố định. Ngoài ra, có thể được mã hóa nhờ sử dụng mô hình ngữ cảnh thích ứng để cải thiện thêm hiệu năng.

Để minh họa rõ hơn các kỹ thuật nhồi bảng màu liên bảng và DPCM màu bên trong bảng, ví dụ sử dụng dữ liệu ở bảng màu hiện tại 553 sẽ được mô tả.

Bắt đầu từ mục thứ nhất c(0) của bảng màu hiện tại 553, tức là, (G, B, R) = (0, 0, 192), có thể thấy rằng c(0) không có so khớp trong bộ dự báo bảng màu 551, do vậy c(0) được mã hóa độc lập. Mục thứ hai c(1) của bảng màu hiện tại 553 ((G, B, R) = (0, 0, 240) còn không có so khớp trong bộ dự báo bảng màu 551. Giả sử mục thứ nhất c(0) đã được mã hóa, chỉ hiệu số dự báo giữa c(1) và c(0) nên được mang trong dòng bit, tức là, (0, 0, 240) – (0, 0, 192) = (0, 0, 48). Đối với mục thứ ba c(2) của bảng hiện tại 553, nhận diện so khớp chính xác trong bộ dự báo bảng màu 551 trong đó j = 1. Chỉ mục dự báo bằng cách sử dụng mục màu được mã hóa trước đó bằng 0, do vậy, chỉ (1- 0) = 1 cần được mã hóa. Các kỹ thuật mã hóa này được áp dụng cho đến mục cuối cùng của bảng hiện tại 553 (tức là, idx = 12 trên Fig.5A) được mã hóa. Bảng 1 minh họa từng bước cách áp dụng chia sẻ liên bảng và DPCM bên trong bảng trên bảng hiện tại 553 sử dụng bộ dự báo bảng màu khả dụng 551.

Bảng 1: Phương pháp mã hóa cho bảng lấy làm ví dụ trên Fig.5A

i (chỉ mục bảng hiện tại)	Phương pháp mã hóa	j (chỉ mục so khớp trong bảng tham chiếu (bộ dự báo bảng màu))	k (chỉ mục so khớp dự báo)	Phần tử mã hóa
0	Bên trong bảng			(0, 0, 192)
1	Bên trong bảng			(0, 0, 48)
2	Liên bảng	1	0	1
3	Liên bảng	2	2	0
4	Liên bảng	3	3	0
5	Bên trong bảng			(0, 0, 2)
6	Bên trong bảng			(60, 10, -12)
7	Liên bảng	8	7	1
8	Bên trong bảng			(0, 30, -30)
9	Bên trong bảng			(20, -50, 0)
10	Liên bảng	9	9	0
11	Bên trong bảng			(30, 0, 0)
12	Liên bảng	15	11	4

Mã hóa tường minh bảng màu sắc được tóm tắt trong đoạn giả mã sau, trong đó N và M lần lượt là số mục trong bảng màu hiện tại và chuẩn.

(Giả mã):

```
encode N;
prev_j = 0;
for ( i = 0; i < N; i++ )
{
    if exist j such that r(j) = = c(i) // Liên bảng palette stuffing
    {
            inter_table_sharing_flag = 1;
            encode inter_table_sharing_flag;
            if ( j = = 0 )
                    k = 0;
            else
                    k = minimum x satisfying x > prev_j and r(k)[0] >= c(i –
1)[0];
            prev_j = k;
            delta = j – k;
```

```
            encode delta;
    }
    else // bên trong bảng color DPCM
    {
            if ( prev_j < M )
            {
                    inter_table_sharing_flag = 0;
                    encode inter_table_sharing_flag;
            }
            if ( i = = 0 )
                    encode c(i) ;
            else
            {
                    delta = c(i) – c(i – 1);
                    encode delta;
            }
    }
}
```

Việc giải mã tường minh bảng màu sắc được tóm tắt trong đoạn giả mã sau.

(Giả mã):

```
decode N;
prev_j = 0;
inter_table_sharing_flag = 0;
for ( i = 0; i < N; i++ )
{
    if ( prev_j < M )
            decode inter_table_sharing_flag;
    if ( decode inter_table_sharing_flag = = 1 )
    {
            decode delta;
```

```
            if ( j = = 0 )
                    k = 0;
            else
                    k = minimum x satisfying x > prev_j and r(k)[0] >= c(i –
1)[0];
            prev_j = k;
            j = k + delta;
            c(i) = r(j);
        }
        else // bên trong bảng color DPCM
        {
            if ( i = = 0 )
                    decode c(i);
            else
            {
                    decode delta;
                    c(i) = c(i – 1) + delta;
            }
        }
    }
```

Có một vài phương pháp tạo các bảng màu lân cận sử dụng trong quá trình hợp nhất khi mã hóa CU hiện tại. Tùy thuộc vào việc triển khai, một trong các phương pháp (được gọi là Phương pháp A để dễ giải thích) cần cập nhật ở cả bộ mã hóa lẫn bộ giải mã. Phương pháp khác (được gọi là Phương B) là xử lý chỉ phía bộ mã hóa. Cả hai phương pháp sẽ được mô tả.

Phương pháp A: Ở phương pháp này, các bảng màu của các CU lân cận được tạo trên điểm ảnh được tái tạo khả dụng, bất kể độ sâu của CU, kích thước, v.v.. Cho mỗi CU, các tái tạo được truy tìm cho CU lân cận của nó ở cùng kích thước và cùng độ sâu (giả sử độ tương đồng màu sẽ cao hơn trong trường hợp này).

Fig.5B minh họa ví dụ tái tạo bảng màu sử dụng Phương pháp A, theo sáng chế. Như được thể hiện trên Fig.5B, CU hiện tại 501 là khối 16x16 có chiều sâu = 2. Các CU lân cận của CU hiện tại 501 gồm CU trên 502 và CU trái 503. CU trên 502 là khối 32x32 có chiều sâu = 1. CU trên 502 gồm khối trên 16x16 504. CU trái 503 là khối 8x8 có chiều sâu = 3, và là một phần của khối 16x16 505. Sử dụng phương pháp A, bất kể phân vùng các CU lân cận (chẳng hạn, CU trái 8x8 503 hoặc CU trên 32x32 502), độ lệch điểm ảnh (=16) sẽ được đặt từ gốc của CU hiện tại 501 sang hướng trái để xử lý khối trái 16x16 505 và sang hướng trên để xử lý khối trên 16x16 504. Cả bộ mã hóa lẫn giải mã duy trì độ lệch này.

Phương pháp B: Ở phương pháp này, quá trình hợp nhất xuất hiện khi CU hiện tại chia sẻ chung kích thước và độ sâu làm CU của nó trên lân cận và/hoặc lân cận CU trái của nó. Các bảng màu của các lân cận khả dụng được sử dụng để suy ra bản đồ chỉ mục màu của CU hiện tại cho các hoạt động tuần tự. Chẳng hạn, đối với CU hiện tại 16x16, nếu CU lân cận của nó (tức là, lân cận trên của nó hoặc lân cận trái) được mã hóa sử dụng bảng màu và phương pháp chỉ mục, bảng màu của CU lân cận được sử dụng cho CU hiện tại để suy ra chi phí R-D. Chi phí hợp nhất được so sánh với trường hợp trong đó CU hiện tại suy ra bảng màu của nó một cách tường minh (cũng như các chế độ đã biết khác có thể tồn tại trong HEVC hoặc HEVC REXt). Trường hợp bất kỳ tạo chi phí R-D thấp nhất được lựa chọn làm chế độ được viết vào dòng bit đầu ra. Ở phương pháp B, chỉ bộ mã hóa được yêu cầu mô phỏng các chế độ tiềm năng khác. Ở bộ giải mã, color_table_merge_flag và color_table_merge_direction flag chỉ báo quyết định hợp nhất và hướng hợp nhất mà không cần xử lý thêm bởi bộ giải mã.

Bảng màu bộ dự báo

Để giảm độ phức tạp, bảng màu bộ dự báo được sử dụng để trữ các màu đến từ bảng màu được mã hóa trước đó hoặc bảng màu bộ dự báo khác, dần dần đến từ bảng màu được mã hóa trước đó. Theo một phương án thực hiện, các

mục trong bảng màu bộ dự báo đến từ bảng màu bộ dự báo hoặc bảng màu được mã hóa của CU trái hoặc CU trên của CU hiện tại. Sau khi CU được mã hóa bằng bảng màu, bảng màu bộ dự báo được cập nhật nếu kích thước CU này lớn hơn hoặc bằng kích thước CU cùng với bảng màu bộ dự báo và bảng màu hiện tại khác với bảng màu bộ dự báo. Nếu CU hiện tại không được mã hóa bằng cách sử dụng chế độ bảng màu, không thay đổi với bảng màu bộ dự báo. Đây còn được gọi là lan truyền bảng màu bộ dự báo. Bảng màu bộ dự báo có thể được đặt lại từ đầu mỗi ảnh hoặc phiến hoặc mỗi hàng CU.

Các phương pháp khả dụng để xây dựng bảng màu bộ dự báo. Ở phương pháp thứ nhất, cho mỗi CU mã hóa, bảng màu bộ dự báo được xây dựng từ bảng màu bộ dự báo của CU trái hoặc CU trên. Ở phương pháp này, một bảng màu bộ dự báo được lưu lại cho mỗi CU.

Phương pháp thứ hai khác phương pháp thứ nhất trong bảng màu đó, thay vì bảng màu bộ dự báo, cùng với CU trên được sử dụng trong quá trình dự báo.

Xử lý/mã hóa bản đồ chỉ mục màu

Khối mã hóa sơ đồ chỉ mục 109 mã hóa bản đồ chỉ mục màu 311 được tạo bởi khối bộ phân loại màu 105. Để mã hóa bản đồ chỉ mục màu 311, khối mã hóa sơ đồ chỉ mục 109 thực hiện ít nhất một hoạt động quét (phương nằm ngang 315 hoặc phương thẳng đứng 317) để biến đổi bản đồ chỉ mục màu 2D 311 thành chuỗi 1D. Sau đó khối mã hóa sơ đồ chỉ mục 109 thực hiện thuật toán tìm kiếm chuỗi (nêu dưới đây) để tạo các phép so khớp. Theo một số phương án thực hiện, khối mã hóa sơ đồ chỉ mục 109 thực hiện các hoạt động quét theo phương nằm ngang hoặc thẳng đứng và thực hiện thuật toán tìm kiếm chuỗi để xác định phương nào cho kết quả tốt hơn. Fig.6 minh họa ví dụ của các hoạt động quét theo phương thẳng đứng và phương nằm ngang. Trên Fig.6, bản đồ chỉ mục màu 2D 601 lấy làm ví dụ được thể hiện. Bản đồ chỉ mục màu 601 có thể đại diện bản đồ chỉ mục màu 311 trên Fig.3. Bản đồ chỉ mục màu 601 là bản đồ 64x64, nhưng các kích thước khác của bản đồ chỉ mục màu là khả thi. Như được thể hiện trên Fig.6, quét (tìm kiếm) theo phương ngang 602

hoặc quét (hoặc tìm kiếm) theo phương thẳng đứng 603 có thể được thực hiện trên bản đồ chỉ mục màu 601.

Các phương án thực hiện sáng chế đề xuất kỹ thuật so khớp chuỗi 1D và biến thể 2D đối với mã hóa bản đồ chỉ mục màu 311. Ở mỗi vị trí, kỹ thuật mã hóa tìm thấy điểm so khớp và ghi nhận khoảng cách được so khớp và độ dài để so khớp chuỗi 1D, hoặc ghi nhận độ rộng và chiều cao của phép so khớp đối với so khớp chuỗi 2D. Đối với vị trí không so khớp, cường độ chỉ mục, hoặc theo cách khác, giá trị delta giữa cường độ chỉ mục và cường độ chỉ mục dự báo, có thể được mã hóa trực tiếp.

Phương pháp tìm kiếm 1D dễ dàng có thể được thực hiện trên bản đồ chỉ mục màu 601. Chẳng hạn, Fig.7 minh họa một phần của vectơ chỉ mục màu 1D 700 sau phép tìm kiếm 1D bằng cách sử dụng quét theo phương ngang từ vị trí chỉ mục thứ nhất của bản đồ chỉ mục màu 601. Sau đó, tìm kiếm chuỗi được áp dụng cho vectơ chỉ mục màu 1D 700. Nhìn ở vị trí thứ nhất 701 của vectơ chỉ mục màu 700 (vốn là '14' như được thể hiện trên Fig.7), do không có tham chiếu đệm, vị trí thứ nhất 701 được xử lý như là "cặp không so khớp". Cặp không so khớp được gán các giá trị từ -1 và 1 đến khoảng cách và độ dài tương ứng, được ký hiệu là (dist, len) = (-1, 1). Vị trí thứ hai 702 là '14' khác. Vị trí thứ hai 702 là chỉ mục thứ nhất được mã hóa làm tham chiếu. Do vậy khoảng cách của cặp so khớp, dist=1. Do có '14' khác ở vị trí thứ ba 703, độ dài của cặp so khớp là 2, tức là, len=2. Di chuyển dọc theo tới vị trí thứ tư 704, gặp giá trị '17', không được nhìn thấy trước đó. Ở đó, vị trí thứ tư 704 được mã hóa dưới dạng cặp không so khớp khác, tức là, (dist, len) = (-1, 1). Đối với mỗi cặp không so khớp, cờ so khớp/không so khớp được mã hóa để báo hiệu không có chỉ mục so khớp được tìm thấy cho chỉ mục hiện tại, và cờ này tiếp theo bởi giá trị thực của chỉ mục (chẳng hạn, xuất hiện thứ nhất '14', '17', '6', v.v.). Đối với mỗi cặp so khớp, cờ so khớp/không so khớp được mã hóa để báo hiệu rằng đã tìm thấy chuỗi chỉ mục so khớp, và cờ này theo sau bởi độ dài của chuỗi so khớp.

Phần sau là nhóm kết quả cho kỹ thuật mã hóa sử dụng một phần của vectơ chỉ mục màu 1D 700 được thể hiện trên Fig.7.

dist = -1, len = 1, idx=14 (không so khớp)

dist = 1, len = 2 (có so khớp)

dist = -1, len = 1, idx=17 (không so khớp)

dist = 1, len = 3 (có so khớp)

dist = -1, len = 1, idx= 6 (không so khớp)

dist = 1, len = 25 (có so khớp)

dist = 30, len = 4 (có so khớp) /*cho "17" xuất hiện trước đó*/

....

Đoạn giả mã sau được dành cho dẫn xuất cặp so khớp này.

(Giả mã):

```
Void deriveMatchedPairs ( TComDataCU* pcCU, Pel* pIdx, Pel* pDist, Pel* pLen, UInt uiWidth, UInt uiHeight)
{
// pIdx is a idx CU bounded within uiWidth*uiHeight

UInt uiTotal = uiWidth*uiHeight;
UInt uiIdx   = 0;
Int j   = 0;
Int len = 0;

// first pixel coded as itself if there isn't left/upper buffer
pDist[uiIdx] = -1;
pLen[uiIdx]  =  0;
uiIdx++;

while (uiIdx < uiTotal )
{
```

```
        len   = 0;
        dist  = -1;
        for ( j=uiIdx-1; j >= 0; j-- )
        {
                // if finding matched pair, currently exhaustive search is
applied
                // fast string search could be applied
                if ( pIdx[j] == pIdx[uiIdx] )
                {
                        for (len = 0; len < (uiTotal-uiIdx); len++ )
                        {
                        if ( pIdx[j+len] != pIdx[len+uiIdx] )
                                break;
                        }
                }
                if ( len > maxLen ) /*better to change with R-D decision*/
                {
                        maxLen = len;
                        dist   = (uiIdx - j );
                }
        }

        pDist[uiIdx] = dist;
        pLen[uiIdx]  = maxLen;
        uiIdx = uiIdx + maxLen;
        }
    }
```

Mã hóa bản đồ chỉ mục màu đơn giản

Theo một số phương án thực hiện, các hoạt động sau có thể được thực hiện dưới dạng phương pháp được đơn giản hóa để xử lý bản đồ chỉ mục màu theo

cách thức 1D. Như được mô tả trên đây, bản đồ chỉ mục màu 601 có thể được biểu diễn bằng các cặp so khớp hoặc không so khớp. Đối với các cặp so khớp, cặp khoảng cách và độ dài được so khớp của các chỉ mục nhóm được báo hiệu cho bộ nhận.

Có các ngữ cảnh có thể nhận thấy trong đó CU gồm chỉ một số màu. Điều này có thể dẫn đến một hoặc nhiều đoạn liên tục hoặc liền kết có giá trị chỉ mục tương tự. Ở những trường hợp này, báo hiệu cặp (khoảng cách, độ dài) có thể dẫn đến nhiều chi phí bổ sung hơn so với cần thiết. Để giải quyết vấn đề này, quá trình xử lý bản đồ chỉ mục màu đơn giản hóa được mô tả dưới đây còn giảm số lượng bit được dùng khi mã hóa bản đồ chỉ mục màu.

Như ở giải pháp mã hóa bản đồ chỉ mục 1D, khái niệm "khoảng cách" có thể được phân chia thành hai phân loại chính: khoảng cách có nghĩa và khoảng cách thường. Khoảng cách thường được mã hóa sử dụng các ngữ cảnh. Sau đó, các độ dài liên kết được mã hóa tuần tự.

Theo các phương án thực hiện, phương pháp này sử dụng khoảng cách có nghĩa. Có hai loại khoảng cách có nghĩa đối với phương pháp này. Một là distance = blockWidth. Còn lại là distance = 1. Đây là hai loại khoảng cách có nghĩa phản ánh quan sát distance = 1 và distance = blockWidth cùng với phần trăm có nghĩa nhất của toàn bộ phân bố khoảng cách. Hai loại khoảng cách có nghĩa sẽ được mô tả nhằm minh họa.

Phương pháp mã hóa sử dụng distance = blockWidth còn được gọi là mã hóa CopyAbove. Để minh họa phương pháp mã hóa CopyAbove, bản đồ chỉ mục màu 64x64 601 trên Fig.6 lại được xem xét. Bản đồ chỉ mục màu 601 có blockWidth = 64. Trong bản đồ chỉ mục màu 64x64 601 là hai chuỗi 611-612 chỉ mục được chỉ báo bằng đường gạch nét. Các giá trị chỉ mục trong chuỗi 612 giống hệt các giá trị chỉ mục tương ứng trong chuỗi 611 ngay phía trên. Do các giá trị chỉ mục trong chuỗi 612 giống hệt các giá trị chỉ mục trong chuỗi 611, các giá trị chỉ mục trong chuỗi 612 có thể được mã hóa bằng cách tham chiếu các giá trị chỉ mục trong chuỗi 611. Khi bản đồ chỉ mục màu 601 được biến đổi thành vectơ chỉ mục màu 1D sử dụng quét theo phương ngang (như được thể

hiện trên vectơ chỉ mục màu 1D 700 trên Fig.7), "khoảng cách" dọc theo vectơ chỉ mục màu 1D giữa các giá trị chỉ mục tương ứng trong các chuỗi 611-612 bằng 64, vốn là độ rộng khối của bản đồ chỉ mục màu 601. Chẳng hạn, khi bản đồ chỉ mục màu 601 được biến đổi thành vectơ chỉ mục màu 1D có 64 x 64 = 4096 phần tử, khoảng cách dọc theo vector giữa giá trị chỉ mục '6' vốn là giá trị thứ nhất trong chuỗi 611, và giá trị chỉ mục '6' vốn là giá trị thứ nhất trong chuỗi 612, bằng 64. Độ dài của các chuỗi so khớp 611-612 bằng 27, do mỗi chuỗi 611-612 gồm 27 giá trị chỉ mục. Do vậy, chuỗi 612 có thể được mã hóa chỉ bằng cách chỉ báo phương pháp mã hóa CopyAbove và độ dài của 27 các giá trị chỉ mục.

Phương pháp mã hóa sử dụng distance = 1 còn được gọi là mã hóa IndexMode hoặc mã hóa CopyLeft. Để minh họa mã hóa IndexMode, xem xét chuỗi 613 của các chỉ mục trong bản đồ chỉ mục màu 601. Chuỗi 613 gồm giá trị chỉ mục thứ nhất '14' tiếp theo bởi 51 giá trị chỉ mục tuần tự '14'. Do mỗi giá trị trong các giá trị chỉ mục trong chuỗi 613 là tương tự, 51 giá trị chỉ mục của chuỗi 613 tiếp theo '14' thứ nhất có thể được mã hóa đồng thời sử dụng distance = 1 (vốn biểu thị rằng giá trị chỉ mục vốn là khoảng cách bằng 1 sang trái giá trị chỉ mục hiện tại có giá trị tương tự). Độ dài của chuỗi so khớp 613 bằng 51. Do vậy, chuỗi 613 có thể được mã hóa đơn giản bằng cách chỉ báo phương pháp mã hóa IndexMode và độ dài của 51 giá trị chỉ mục.

Như được mô tả trên đây, đối với phương pháp mã hóa bản đồ chỉ mục màu đơn giản, khoảng cách được sử dụng để mã hóa có thể được giới hạn ở chỉ các vị trí đáng kể; tức là, khoảng cách đối với các phương án thực hiện này có thể bị giới hạn chỉ ở 1 hoặc blockWidth. Để giảm thêm chi phí bổ sung, độ dài của chỉ mục so khớp còn có thể được giới hạn ở độ rộng CU. Sử dụng định nghĩa này, cặp khoảng cách và độ dài có thể được báo hiệu nhờ sử dụng chỉ hai cờ nhị phân (tức là, 2 nhị phân) mà không gửi chi phí bổ sung của độ dài và khoảng cách (được suy ra dưới dạng độ rộng khối). Chẳng hạn, cờ thứ nhất có thể biểu thị nếu việc mã hóa sử dụng khoảng cách có nghĩa hoặc không sử dụng khoảng cách có nghĩa. Nếu cờ thứ nhất biểu thị rằng việc mã hóa sử dụng

khoảng cách có nghĩa, thì cờ thứ hai có thể biểu thị nếu khoảng cách có nghĩa bằng 1 (tức là, IndexMode) hoặc blockWidth (tức là, CopyAbove). Do chuỗi so khớp xuất hiện theo dòng (hoặc theo hàng) trong CU, các chỉ mục bất kỳ trong dòng không được so khớp bởi distance = 1 hoặc distance = blockWidth được xử lý dưới dạng các chỉ mục không được so khớp. Các chỉ mục không được so khớp này được mã hóa lần lượt riêng rẽ. Đối với các chỉ mục không được so khớp này, các phương pháp dự báo nêu trên có thể được sử dụng để cải thiện hiệu quả.

Bộ giải mã có thể thực hiện hoạt động giải mã tương tự các kỹ thuật mã hóa CopyAbove và mã hóa IndexMode nêu trên. Chẳng hạn, bộ giải mã có thể nhận cờ thứ hai, và dựa trên giá trị của cờ thứ hai, bộ giải mã biết giải mã theo kỹ thuật giải mã CopyAbove hoặc IndexMode.

Biến thể 2D của kỹ thuật so khớp chuỗi 1D nêu trên có thể còn được sử dụng. Kỹ thuật so khớp 2D gồm các bước sau:

Bước 1: Vị trí của điểm ảnh hiện tại và điểm ảnh tham chiếu được nhận diện như là điểm bắt đầu.

Bước 2: tìm kiếm chuỗi 1D nằm ngang được áp dụng cho hướng phải của điểm ảnh hiện tại và điểm ảnh tham chiếu. Độ dài tìm kiếm lớn nhất bị giới hạn bởi cuối hàng ngang hiện tại. Độ dài tìm kiếm lớn nhất có thể được ghi nhận như là right_width.

Bước 3: tìm kiếm chuỗi 1D nằm ngang được áp dụng cho hướng trái của điểm ảnh hiện tại và điểm ảnh tham chiếu. Độ dài tìm kiếm lớn nhất bị giới hạn bởi bắt đầu hàng ngang hiện tại, và có thể còn bị giới hạn bởi right_width của phép so khớp 2D trước. Độ dài tìm kiếm lớn nhất có thể được ghi nhận như là left_width.

Bước 4: Tìm kiếm chuỗi 1D tương tự được thực hiện ở hàng tiếp theo, sử dụng điểm ảnh dưới điểm ảnh hiện tại và điểm ảnh tham chiếu làm điểm ảnh hiện tại mới và điểm ảnh tham chiếu.

Bước 5: Dừng khi right_width == left_width == 0.

Bước 6: Đối với mỗi height[n] = {1, 2, 3…}, có mảng tương ứng width[n] (chẳng hạn, {left_width[1], right_width[1]}, {left_width[2], right_width[2]}, {left_width[3], right_width[3]}…}.

Bước 7: Mảng min_width mới được định nghĩa là {{lwidth[1], rwidth[1]}, {lwidth[2], rwidth[2]}, {lwidth[3], rwidth[3]}…} đối với mỗi height[n], trong đó lwidth[n] = min(left_width[1:n-1]), rwidth[n] = min(right_width[1:n-1]).

Bước 8: Mảng kích thước {size[1], size[2], size[3]…} còn được định nghĩa, trong đó size[n] = height[n] x (lwidth[n]+hwidth[n]).

Bước 9: Giả sử rằng size[n] giữ giá trị lớn nhất trong mảng kích thước, độ rộng và chiều cao của phép so khớp chuỗi 2D được lựa chọn bằng cách sử dụng {lwidth[n], rwidth[n], height[n]} tương ứng.

Một kỹ thuật để tối ưu hóa tốc độ của phép tìm kiếm 1D hoặc 2D là sử dụng băm chạy. Theo một số phương án thực hiện, cấu trúc băm chạy 4 điểm ảnh có thể được sử dụng. Băm chạy được tính toán cho mỗi điểm ảnh theo hướng ngang để tạo mảng băm ngang running_hash_h[]. Băm chạy khác được tính toán trên cùng running_hash_h[] để tạo mảng băm 2D running_hash_hv[]. Mỗi phép so khớp giá trị trong mảng băm 2D running_hash_hv[] biểu thị phép so khớp khối 4x4. Để thực hiện so khớp 2D, các phép so khớp khối 4x4 được tìm thấy trước khi thực hiện so sanh theo điểm ảnh với các lân cận. Do phép so sánh theo điểm ảnh bị giới ảnh ở 1 đến 3 điểm ảnh, tốc độ tìm kiếm có thể được tăng đáng kể.

Từ phần mô tả trên đây, các độ rộng so khớp của mỗi hàng khác nhau, do vậy mỗi hàng phải được xử lý riêng rẽ. Để đạt hiệu suất và độ phức tạp thấp, các phương án thực hiện sáng chế đề xuất thuật toán dựa trên khối có thể được sử dụng khi triển khai cả phần cứng lẫn phần mềm. Tương tự theo một số khía cạnh về ước tính chuyển động chuẩn, thuật toán này xử lý một khối hình chữ nhật tại một thời điểm.

Fig.8 minh họa ví dụ của khối xử lý điểm ảnh cơ bản ở thuật toán này, được gọi là môđun U_PIXEL 800. Môđun U_PIXEL 800 tiếp nhận tín hiệu được mã hóa 801 và tín hiệu đầu vào 802, và gồm các cổng lôgic từ 803 đến 806. Tín

hiệu được mã hóa 801 là cờ biểu thị nếu điểm ảnh tham chiếu đã được mã hóa từ hoạt động so khớp chuỗi trước đó. Một cách tùy chọn, tín hiệu đầu vào 802 (Cmp[n-1]) có thể bị đẩy về "0", cho phép loại bỏ cổng "OR" 806 cuối cùng khỏi môđun U_PIXEL 800.

Lấy khối 4x4 làm ví dụ. Bước thứ nhất là để xử lý song song mỗi hàng. Mỗi điểm ảnh ở một hàng chữ nhật được gán cho một môđun U_PIXEL 800. Khối xử lý để xử lý mỗi hàng được gọi là môđun U_ROW. Fig.9 minh họa ví dụ của môđun U_ROW 900. Môđun U_ROW 900 gồm các môđun U_PIXEL 800. Đối với trường hợp của khối 4x4, môđun U_ROW 900 gồm bốn môđun U_PIXEL 800. Như được thể hiện trên Fig.9, môđun U_ROW 900 đang xử lý hàng thứ nhất, hàng 0, như được biểu thị ở 901.

Bốn môđun U_ROW 900 được sử dụng để xử lý bốn hàng của khối 4x4. Bốn môđun U_ROW 900 có thể được bố trí song song ở môđun U_CMP. Fig.10 minh họa ví dụ của môđun U_CMP 1000 gồm bốn môđun U_ROW 900. Đầu ra của môđun U_CMP 1000 là mảng cmp[4][4].

Bước tiếp theo của thuật toán là để xử lý song song mỗi cột của mảng cmp. Mỗi cmp trong cột của mảng cmp được xử lý bởi môđun U_COL. Fig.11 minh họa ví dụ của môđun U_COL 1100 tiếp nhận bốn cột 1101-1104 của mảng cmp. Bốn môđun U_COL 1100 có thể được sử dụng để xử lý bốn cột của khối 4x4. Bốn môđun U_COL 1100 có thể được bố trí song song trong môđun U_2D_BLOCK. Fig.12 minh họa môđun U_2D_BLOCK 1200 lấy làm ví dụ gồm bốn môđun U_COL 1100. Đầu ra của môđun U_2D_BLOCK 1200 là mảng rw[4][4].

Sau đó số lượng 0 trong mỗi hàng của mảng rw[n][0-3] được đếm và bốn kết quả được ghi vào mảng r_width[n]. Mảng r_width[n] tương tự mảng rwidth[n] ở bước 7 của kỹ thuật so khớp 2D nêu trên. Mảng l_width[n] được tạo theo cách thức tương tự. Mảng min_width ở bước 7 có thể thu được dưới dạng {{l_width[1], r_width[1]}, { l_width[2], r_width[2]}, { l_width[3], r_width[3]}…}.

Thuật toán này có thể được triển khai ở phần cứng hoặc tổ hợp cả phần cứng lẫn phần mềm trong khung xử lý song song của CPU (central processing unit, khối xử lý trung tâm) hiện tại, DSP (digital signal processor, bộ xử lý tín hiệu số), hoặc GPU (graphics processing unit, khối xử lý đồ họa). Giả mã đơn giản hóa để triển khai phần mềm nhanh được liệt kê dưới đây.

Giả mã :

```
// 1. Tạo mảng C[][]
For(y = 0; y < height; ++y)
{
    For(x = 0; x < width; ++x)
    {
        tmp1 = cur_pixel ^ ref_pixel;
        tmp2 = tmp1[0] | tmp1[1] | tmp1[2] | tmp1[3] | tmp1[4] | tmp1[5] |
tmp1[6] | tmp1[7];
        C[y][x] = tmp2 & (!coded[y][x]);
    }
}
// 2. Tạo mảng CMP[][]
For(y = 0; y < height; ++y)
{
    CMP[y][0] = C[y][0];
}
For(x = 1; x < width; ++x)
{
    For(y = 0; y < height; ++y)
    {
        CMP[y][x] = C[y][x] | CMP[y][x-1]
    }
}
// 3. Tạo mảng RW[][] hoặc LW[][]
```

```
For(x = 0; x < width; ++x)
{
    RW[0][x] = CMP[0][x];
}
For(y = 1; y < height; ++y)
{
    For(x = 0; x < width; ++x)
    {
        RW[y][x] = CMP[y][x] | RW[y-1][x];
    }
}
// 4. Biến đổi RW[][] thành R_WIDTH[]
For(y = 0; y < height; ++y)
{
    // đếm 0, hoặc dò 0
    R_WIDTH[y] = LZD(RW[y][0], RW[y][1], RW[y][2], RW[y][3]);
}
```

Như được thể hiện trên đoạn giả mã trên, không có phụ thuộc dữ liệu trong mỗi vòng lặp FOR, do vậy các phương pháp xử lý song song phần mềm đặc trưng, như thả vòng hoặc MMX/SSE, có thể được áp dụng để tăng tốc độ thực thi.

Thuật toán này có thể còn áp dụng cho tìm kiếm 1D nếu số hàng bị giới hạn bằng 1. Đoạn giả mã đơn giản hóa để triển khai phần mềm nhanh của phép tìm kiếm 1D dựa vào độ dài cố định được liệt kê dưới đây.

(Giả mã):

```
// 1. Tạo mảng C[]
For(x = 0; x < width; ++x)
{
    tmp1 = cur_pixel ^ ref_pixel;
```

```
        tmp2 = tmp1[0] | tmp1[1] | tmp1[2] | tmp1[3] | tmp1[4] | tmp1[5] |
tmp1[6] | tmp1[7];
        C[x] = tmp2 & (!coded[x]);
    }

    // 2. Tạo mảng RW[] or LW[]
        If (last "OR" operation in U_PIXEL module is removed)
        Assign RW[] = C[]
        Else {
        RW [0] = C[0];
        For(x = 1; x < width; ++x)
        {
                RW [x] = C[x] | RW [x-1]
        }
    ]
    // 3. Biến đổi RW[][] thành R_WIDTH[]
    // đếm 0, hoặc dò 0
        If(last "OR" operation in U_PIXEL module is removed)
        R_WIDTH = LZD(RW[0], RW[1], RW[2], RW[3]);
        Else
        R_WIDTH[y] = COUNT_ZERO(RW[0], RW[1], RW[2], RW[3]);
```

Sau khi hoàn thành cả phép tìm kiếm 1D lẫn tìm kiếm 2D, giá trị lớn nhất của (1D length, 2D size (width x height)) được lựa chọn làm "giá trị chiến thắng". Nếu lwidth (left width) của phép so khớp 2D khác 0, độ dài của phép so khớp 1D trước (length = length – lwidth) có thể được điều chỉnh để tránh trùng lặp giữa phép so khớp 1D trước và phép so khớp 2D hiện tại. Nếu độ dài của phép so khớp 1D trước bằng 0 sau khi điều chỉnh, nó nên được loại khỏi danh sách so khớp.

Tiếp theo, vị trí bắt đầu được tính toán sử dụng current_location + length nếu so khớp trước đó là so khớp 1D, hoặc current_location + (lwidth+rwidth)

nếu so khớp trước đó là so khớp 2D. Khi tìm kiếm 1D được thực hiện, nếu điểm ảnh sẽ được so khớp bất kỳ nằm trong vùng so khớp 2D trước đó bất kỳ trong đó vị trí của nó đã được mã hóa bởi phép so khớp 2D, điểm ảnh tiếp theo hoặc điểm ảnh được quét qua cho đến khi điểm ảnh được tìm thấy không được mã hóa bởi phép so khớp trước.

Sau khi thu thập các cặp so khớp, cấu trúc entropy có thể được áp dụng để biến đổi các phần tử mã hóa này thành dòng nhị phân. Theo một số phương án thực hiện, cấu trúc entropy có thể sử dụng mô hình xác suất cân bằng. Mô hình ngữ cảnh thích ứng nâng cao có thể được áp dụng cho hiệu quả nén tốt hơn. Đoạn giả mã sau là ví dụ của thủ tục mã hóa cho mỗi cặp so khớp.

(Giả mã):

```
// lặp cho mỗi CU, uiTotal=uiWidth*uiHeight, uiIdx=0;
while ( uiIdx < uiTotal) {
    // *pDist: lưu trữ giá trị khoảng cách cho mỗi cặp so khớp
    // *pIdx: lưu trữ giá trị chỉ mục cho mỗi cặp so khớp
    // *pLen: lưu trữ giá trị chiều dài cho mỗi cặp so khớp
    // encodeEP() và encodeEPs() sử dụng lại HEVC hoặc mã hóa entropy đường vòng.
    if (pDist[uiIdx] == -1 )
    {
    //mã hóa mộtbin với mô hình xác suất cân bằng để biểu thị
    //liệu cặp hiện tại có so khớp hoặc không.
            unmatchedPairFlag = TRUE;
            encodeEP(unmatchedPairFlag);
            //uiIndexBits được điều khiển bởi kích thước bảng màu
            // tức là, đối với 24 màu khác, cần 5 bit, cho 8 màu, 3 bit
            encodeEPs(pIdx[uiIdx], uiIndexBits);
            uiIdx++;
    }
    else
```

```
    {
        unmatchedPairFlag= FALSE;
        encodeEP(unmatchedPairFlag);

        /*liên kết nhị phân hóa với giá trị khả thi max*/
        UInt uiDistBits =0;
//offset được sử dụng để thêm các tham chiếu bổ sung từ các khối lân cận
    //ở đây, trước hết cho offset=0;
        while( (1<<uiDistBits)<= (uiIdx+offset))
        {
            uiDistBits++;
        }
        encodeEPs(pDist[uiIdx], uiDistBits);
        /*liên kết nhị phân hóa với giá trị khả thi max*/
        UInt uiLenBits =0;
        while( (1<<uiLenBits)<= (uiTotal-uiIdx))
        {
            uiLenBits++;
        }
        encodeEPs(pLen[uiIdx], uiLenBits);
        uiIdx += pLen[uiIdx];
    }
}
```

Tương ứng, quá trình giải mã cho cặp so khớp được đề xuất trong giả mã sau.

```
(Giả mã):
// lặp cho mỗi CU, uiTotal=uiWidth*uiHeight, uiIdx=0;
while ( uiIdx < uiTotal) {
    // *pDist: lưu trữ giá trị khoảng cách cho mỗi cặp so khớp
    // *pIdx: lưu trữ giá trị chỉ mục cho mỗi cặp so khớp
```

```
// *pLen: lưu trữ giá trị chiều dài cho mỗi cặp so khớp
// parseEP() và parseEPs() sử dụng lại HEVC hoặc mã hóa entropy vòng tương tự .
// phân tích cấu trúc cờ cặp không so khớp
parseEP(&uiUnmatchedPairFlag);
        if (uiUnmatchedPairFlag )
{
        parseEPs( uiSymbol, uiIndexBits );
        pIdx[uiIdx] = uiSymbol;
        uiIdx++;
}
else
{
        /*liên kết nhị phân hóa với giá trị khả thi max*/
        UInt uiDistBits =0;
        // offset được sử dụng để thêm các tham chiếu bổ sung từ các khối lân cận
        // ở đây, trước hết cho offset=0;
        while( (1<<uiDistBits)<= (uiIdx+offset))
                uiDistBits++;
        UInt uiLenBits =0;
        while( (1<<uiLenBits)<= (uiTotal-uiIdx))
                uiLenBits++;
        parseEPs( uiSymbol, uiDistBits);
        pDist[uiIdx] = uiSymbol;
        parseEPs( uiSymbol, uiLenBits);
        pLen[uiIdx] = uiSymbol;
        for(UInt i=0; i< pLen[uiIdx]; i++)
                pIdx[i+uiIdx] = pIdx[i+uiIdx- pDist[uiIdx]];
        uiIdx += pLen[uiIdx];
```

}

}

Lưu ý rằng chỉ điểm ảnh ở các vị trí không so khớp sẽ được mã hóa thành dòng bit. Để có mô hình thống kê chính xác hơn, một số phương án thực hiện có thể sử dụng chỉ điểm ảnh và các lân cận để suy ra bảng màu, thay vì sử dụng tất cả điểm ảnh trong CU.

Đối với các chế độ mã hóa xác định chỉ mục hoặc đầu ra delta, các kết quả mã hóa chủ yếu chứa số lượng hữu hạn các giá trị duy nhất. Các phương án thực hiện sáng chế bảng màu delta thứ hai để tận dụng quan sát này. Bảng màu delta này có thể được tạo sau khi tất cả dữ liệu chữ thu được ở CU hiện tại. Bảng màu delta có thể được báo hiệu tường minh trong dòng bit. Theo cách khác, có thể được tạo một cách thích ứng trong quá trình mã hóa, sao cho bảng không được gồm trong dòng bit. delta_color_table_adaptive_flag được đề xuất để chọn.

Theo một số phương án thực hiện, phương tiện nâng cao khác, được gọi là hợp nhất bảng màu delta lân cận, được đề xuất. Để tạo bảng màu delta thích ứng, bộ mã hóa có thể sử dụng bảng màu delta từ trên hoặc CU trái làm điểm bắt đầu ban đầu. Để tạo bảng màu không thích ứng, bộ mã hóa có thể còn sử dụng bảng màu delta từ trên hoặc CU trái, và sau đó so sánh chi phí R-D trong số các CU trên cùng, bên trái và hiện tại.

delta_color_table_merge_flag được định nghĩa để chỉ báo liệu CU hiện tại có sử dụng bảng màu delta từ CU trái hoặc CU trên. CU hiện tại mang báo hiệu bảng màu delta tường minh chỉ khi delta_color_table_adaptive_flag==0 và delta_color_table_merge_flag==0 cùng lúc. Đối với quá trình hợp nhất, nếu delta_color_table_merge_flag được xác nhận, cờ khác, delta_color_table_merge_direction, được định nghĩa để chỉ báo liệu ứng viên hợp nhất là từ CU trên hoặc CU trái.

Nếu delta_color_table_adaptive_flag==1, phần sau là ví dụ của quá trình mã hóa để tạo bảng màu delta thích ứng. Ở phía bộ giải mã, bất cứ khi nào bộ

giải mã tiếp nhận dữ liệu dạng chữ, sau đó bộ giải mã có thể tạo lại bảng màu delta nhờ sử dụng các bước ngược lại.

Bước 1: Các mảng palette_table[] và palette_count[] được định nghĩa.

Bước 2: Mảng palette_table[] được khởi tạo dưới dạng palette_table(n) = n (n = 0…255). Theo cách khác, palette_table[] từ CU trên cùng hoặc CU trái có thể được sử dụng làm giá trị ban đầu.

Bước 3: Mảng palette_count[] được khởi tạo dưới dạng palette_count(n) = 0 (n = 0…255). Theo cách khác, palette_count[] từ CU trên cùng hoặc CU trái có thể được sử dụng làm giá trị ban đầu.

Bước 4: Đối với giá trị delta c' bất kỳ, các hoạt động sau được thực hiện:

a) Định vị n sao cho palette_table(n) == delta c';

b) Sử dụng n làm chỉ mục mới của delta c';

c) ++palette_count(n);

d) Sắp xếp palette_count[] sao cho nó theo thứ tự giảm; và

e) Sắp xếp palette_table[] tương ứng.

Bước 5: Quá trình trở về bước 1 và quá trình được lặp lại cho đến khi tất cả delta c' trong CU hiện tại được xử lý.

Đối với khối bất kỳ gồm cả văn bản lẫn đồ họa, cờ mặt nạ có thể được sử dụng để tách riêng phần văn bản và phần đồ họa. Phần văn bản có thể được nén nhờ sử dụng phương pháp nén nêu trên; phần đồ họa có thể được nén bởi phương pháp nén khác. Do giá trị của điểm ảnh bất kỳ được bao phủ bởi cờ mặt nạ được mã hóa không tổn hao bởi lớp văn bản, mỗi điểm ảnh trong phần đồ họa có thể được xem là "điểm ảnh không bận tân". Khi phần đồ họa được nén, giá trị tùy ý bất kỳ có thể được gán cho điểm ảnh không bận tâm để có hiệu năng nén tối ưu.

Bản đồ chỉ mục và các thặng dư được tạo trong quá trình dẫn xuất bảng màu. Việc nén không tổn hạo bản đồ chỉ mục cho phép xử lý hiệu quả nhờ sử dụng phép tìm kiếm chuỗi 1D hoặc 2D. Theo một số phương án thực hiện, phép tìm kiếm chuỗi 1D hoặc 2D bị giới hạn trong CU hiện tại; Tuy nhiên, cửa sổ tìm kiếm có thể được mở rộng vượt qua CU hiện tại. Khoảng cách so khớp

có thể được mã hóa nhờ sử dụng cặp vectơ chuyển động theo các phương ngang và thẳng đứng, chẳng hạn, (MVy=matched_distance/cuWidth, MVy=matched_distance-cuWidth*MVy).

Do ảnh có thể có các định hướng kết cấu không gian khác nhau ở các vùng cục bộ, phép tìm kiếm 1D có thể được thực hiện theo các hướng ngang hoặc thẳng đứng dựa trên giá trị của bộ chỉ báo color_idx_map_pred_direction. Hướng quét chỉ mục tối ưu có thể được xác định dựa trên chi phí R-D. Fig.13 minh họa ví dụ của các hoạt động quét theo phương ngang hoặc thẳng đứng. Trên Fig.13, bản đồ chỉ mục màu 2D 1301 lấy làm ví dụ được thể hiện. Bản đồ chỉ mục màu 1301 có thể đại diện bản đồ chỉ mục màu 311 trên Fig.3. Bản đồ chỉ mục màu 1301 là bản đồ 8x8, nhưng các kích thước khác của bản đồ chỉ mục màu là có thể. Như được thể hiện trên Fig.13, quét theo phương ngang 1302 hoặc quét theo phương thẳng đứng 1303 có thể được thực hiện trên bản đồ chỉ mục màu 1301. Theo một số phương án thực hiện, deriveMatchPairs() và các bước mã hóa entropy liên kết được thực hiện hai lần để quét theo phương ngang lẫn quét theo phương thẳng đứng. Sau đó, hướng quét cuối cùng được chọn làm hướng với chi phí R-D nhỏ nhất.

Nhị phân hóa cải tiến

Như được thể hiện trên đây, bảng màu và cặp thông tin so khớp cho bản đồ chỉ mục màu có thể được mã hóa nhờ sử dụng nhị phân hóa độ dài cố định. Theo cách khác, nhị phân hóa có chiều dài biến thiên có thể được sử dụng. Chẳng hạn, đối với bảng màu mã hóa, bảng màu có thể có tám giá trị màu khác nhau. Do vậy, bản đồ chỉ mục màu tương ứng có thể chứa chỉ tám chỉ mục khác nhau. Thay vì sử dụng ba bin cố định để mã hóa từng giá trị chỉ mục ngang bằng, chỉ một bin có thể được sử dụng để biểu thị điểm ảnh nền. Chẳng hạn, điểm ảnh nền có thể được biểu thị bằng 0. Sau đó, bảy giá trị điểm ảnh còn lại có thể được biểu thị bằng cách sử dụng các từ mã có độ dài cố định như 1000, 1001, 1010, 1011, 1100, 1101, và 1110 để mã hóa chỉ mục màu. Điều này dựa trên thực tế là màu nền có thể chiếm phần trăm ảnh lớn nhất, và do vậy

từ mã nổi bật chỉ một bit cho màu nền có thể tiết kiệm tổng không gian. Ngữ cảnh này xuất hiện thông thường cho nội dung màn hình. Như là ví dụ, xem xét CU 16x16. Sử dụng nhị phân hóa ba bin cố định, bản đồ chỉ mục màu yêu cầu 3x16x16=768 bin. Theo cách khác, cho màu nền, chiếm 40% ảnh, được đánh chỉ mục bằng 0, trong khi các màu còn lại được phân bố đều. Trong trường hợp này, bản đồ chỉ mục màu chỉ yêu cầu 2,8x16x16<768 bin.

Để mã hóa cặp so khớp, giá trị khả thi lớn nhất của khoảng cách và độ dài so khớp có thể được sử dụng để giới hạn nhị phân hóa, các giới hạn hiện tại cụ thể của công nghệ trong khu vực của CU hiện tại. Về mặt toán học, khoảng cách và chiều dài so khớp dài bằng 64x64=4K trong mỗi trường hợp. Tuy nhiên, điều này thông thường không xuất hiện đồng thời. Đối với từng vị trí so khớp, khoảng cách so khớp được giới hạn bởi khoảng cách giữa vị trí hiện tại và vị trí thứ nhất trong bộ đệm tham chiếu (chẳng hạn, vị trí thứ nhất ở CU hiện tại), có thể được biểu thị như là L. Do vậy, các bin lớn nhất để nhị phân hóa khoảng cách bằng log2(L)+1 (thay vì fixed length), và các bin lớn nhất để nhị phân hóa độ dài là log2(cuSize-L)+1 với cuSize=cuWidth*cuHeight.

Bên cạnh bảng màu và bản đồ chỉ mục, mã hóa hệ số dư có thể được cải thiện đáng kể bởi các phương pháp nhị phân hóa khác nhau. Như đối với các phiên bản HEVC RExt và HEVC, hệ số biến đổi được nhị phân hóa nhờ sử dụng chiều dài biến thiên dựa trên quan sát rằng hệ số được tạo sau khi dự báo, biến đổi và lượng tử hóa nhờ sử dụng các phương pháp đã biết thông thường có độ lớn gần bằng 0, và các giá trị khác 0 thường được đặt ở góc trên bên trái của khối biến đổi. Tuy nhiên, sau khi giới thiệu dụng cụ mã hóa bỏ qua biến đổi trong HEVC RExt mà cho phép đi vòng toàn bộ quá trình biến đổi, việc phân tán độ lớn thặng dư đã thay đổi. Đặc biệt khi cho phép bỏ qua biến đổi trên nội dung màn hình với các màu nổi bật, thông thường có các hệ số với các giá trị lớn(tức là, các giá trị không gần bằng 0, như ‘1’, ‘2’, hoặc ‘0’) và các giá trị khác 0 có thể xuất hiện ở các vị trí ngẫu nhiên trong khối biến đổi. Nếu nhị phân hóa hệ số HEVC hiện tại được sử dụng, có thể thu được từ mã rất dài. Theo cách khác, nhị phân hóa độ dài cố định có thể được sử dụng, có thể tiết

kiệm độ dài mã cho các hệ số dư được tạo bằng bảng màu và chế độ mã hóa chỉ mục.

Phương pháp tạo điểm ảnh dự báo mới

Như được mô tả trên đây, tìm kiếm chuỗi 1D/2D được thực hiện trong mã hóa bản đồ chỉ mục màu. Ở vị trí bất kỳ trong bản đồ chỉ mục màu trong đó chỉ mục so khớp được tìm thấy, bộ giải mã lấy điểm ảnh ở vị trí so khớp và điểm ảnh gốc trừ nó để tạo điểm ảnh thặng dư. Thủ tục này có thể được thực hiện hoặc bằng cách sử dụng màu tương ứng trong bảng màu được đại diện bằng chỉ mục màu ở vị trí so khớp, hoặc bằng cách sử dụng điểm ảnh được tái tạo ở vị trí so khớp.

Có hai phương pháp tạo giá trị dự báo dựa trên hai phương pháp nêu trên. Ở phương pháp thứ nhất, đối với vị trí điểm ảnh đích bất kỳ, giá trị RGB được suy ra từ bảng màu bởi chỉ mục màu chính ở vị trí so khớp, và giá trị RGB này được sử dụng làm giá trị dự báo của điểm ảnh đích. Tuy nhiên, phương pháp này buộc bộ giải mã thực hiện thủ tục dẫn xuất chỉ mục màu đến điểm ảnh ngoài CU hiện tại, làm tăng thời gian giải mã.

Để tránh thủ tục dẫn xuất chỉ mục màu ở phương pháp thứ nhất, phương pháp thứ hai được áp dụng trong đó, đối với vị trí điểm ảnh đích bất kỳ, giá trị điểm ảnh được tái tạo ở vị trí so khớp được sử dụng làm giá trị dự báo. Ở phương pháp này, giá trị được tái tạo không hợp lệ khi điểm ảnh dự báo trong CU hiện tại. Ở trường hợp này, tuy nhiên, chỉ mục màu là khả dụng và màu tương ứng trong bảng màu có thể được sử dụng làm điểm ảnh dự báo.

Giá trị thặng dư của điểm ảnh bất kỳ trong CU hiện tại có thể được suy ra bằng cách lấy giá trị gốc trừ giá trị dự báo. Sau đó, được lượng tử hóa và được mã hóa thành dòng bit. Giá trị tái tạo của điểm ảnh bất kỳ ở CU hiện tại có thể được suy ra bằng cách thêm giá trị dự báo và giá trị thặng dư lượng tử hóa.

Chế độ màu đơn

CU màu đơn có thể là CU có duy nhất một màu ở mọi vị trí điểm ảnh hoặc CU có màu đơn trong bảng màu với bản đồ chỉ mục đơn giá trị đồng đều. Có nhiều phương pháp để nén CU đơn màu ở chế độ bảng màu. Theo một phương pháp, tức là, chế độ đơn màu, chỉ thông tin bảng màu màu đơn này được mã hóa và được bao gồm trong dòng bit. Toàn bộ đoạn bản đồ chỉ mục màu bị bỏ qua. Điều này ngược với mã hóa và truyền bản đồ chỉ mục toàn 0 đồng đều. Ở phía bộ giải mã, nếu chỉ có màu đơn trong bảng màu mà không có bản đồ chỉ mục, mỗi vị trí điểm ảnh trong CU hiện tại sẽ được lấp đầy màu trong bảng màu.

Sao chép chuỗi miền điểm ảnh

Như được mô tả trên đây, sao chép chuỗi 1D/2D được áp dụng trong miền bản đồ chỉ mục màu. Sao chép chuỗi 1D/2D có thể còn được áp dung trong miền điểm ảnh. So với miền bản đồ chỉ mục sao chép chuỗi 1D/2D, sao chép chuỗi 1D/2D trong miền điểm ảnh gồm nhiều thay đổi. Các thay đổi như sau:

1. Bảng màu và quá trình tạo bản đồ chỉ mục không cần thiết và có thể được bỏ qua. Tùy chọn khác, thực hiện tất cả việc tạo bảng màu, tạo bản đồ chỉ mục, và tìm kiếm chuỗi 1D/2D trên miền chỉ mục vẫn được thực hiện, nhưng bảng màu không được viết vào dòng bit. Bản đồ được mã hóa đượ tạo dựa trên độ dài của phép so khớp chuỗi 1D hoặc chiều rộng và chiều cao của phép so khớp chuỗi 2D. Bản đồ được mã hóa biểu thị liệu vị trí điểm ảnh có được bao phủ bởi phép so khớp trước hay không. Vị trí bắt đầu tiếp theo là vị trí thứ nhất không được bao phủ bởi phép so khớp trước.

2. Khi mã hóa dữ liệu không được so khớp, giá trị RGB của nó (thay vì giá trị chỉ mục màu) được viết vào dòng bit. Khi mã hóa dữ liệu không được so khớp, phương pháp mã hóa chỉ mục điểm ảnh có thể còn được áp dụng trong đó cờ một bit được thêm vào trước giá trị RBG này trong bảng cấu trúc. Nếu giá trị RBG này xuất hiện lần thứ nhất, cờ được đặt bằng 1 và chính giá trị RBG này được mã hóa vào dòng bit. Giá trị RBG này được thêm vào bảng tra cứu sau đó.

Nếu giá trị RBG này lại xuất hiện, cờ được đặt bằng 0 và giá trị chỉ mục bảng tra cứu thay vì giá trị RBG này được mã hóa.

3. Phương pháp tạo điểm ảnh dự báo sử dụng tùy chọn 2 của chế độ đơn màu (giá trị điểm ảnh được tái tạo từ vị trí điểm ảnh dự báo được sử dụng làm giá trị dự báo).

4. Đối với CU đơn màu, tùy chọn 1 hoặc tùy chọn 2 của chế độ đơn màu có thể được lựa chọn. Khi tùy chọn 1 được lựa chọn, giá trị RGB của màu chính được viết vào đoạn bảng màu của dòng bit. Khi tùy chọn 2 được lựa chọn, nếu dòng trên không được sử dụng trong phép tìm kiếm 1D và tùy chọn 2D không được phép cho CU hiện tại, giá trị RGB của màu chính được viết vào đoạn bảng màu của dòng bit.

Nói chung, sao chép chuỗi 2D là thuật toán linh hoạt; có thể thực hiện các hoạt động trên các khối có các độ rộng và chiều cao khác nhau để tìm khối so khớp. Khi sao chép chuỗi 2D bị giới hạn ở độ rộng và chiều cao của CU, sao chép chuỗi 2D trở thành sao chép khối độ rộng/chiều cao cố định. IBC (Intra block copy, sao chép bên trong khối) gần như giống hệ trường hợp cụ thể sao chép chuỗi 2D này mà hoạt động trên khối độ rộng/chiều cao cố định. Khi sao chép chuỗi 2D độ rộng/chiều cao cố định, giá trị thặng dư cũng được mã hóa. Điều này cũng gần như tương tự với phương pháp mã hóa thặng dư được sử dụng bởi IBC.

Lấy mẫu sắc độ thích ứng cho nội dung hỗn hợp

Các phương án thực hiện nêu trên đề xuất các kỹ thuật khác nhau để mã hóa nội dung màn hình hiệu suất cao dưới khuôn khổ của HEVC/HEVC-REXt. Thực tế, bên cạnh nội dung màn hình thuần túy (như văn bản, đồ họa) hoặc video tự nhiên thuần túy, còn có nội dung chứa chất liệu màn hình được máy tính tạo ra và video tự nhiên do camera bắt giữ. Đây được gọi là nội dung hỗn hợp. Hiện tại, nội dung hỗn hợp được xử lý bằng việc lấy mẫu sắc độ 4:4:4. Tuy nhiên, đối với phần video tự nhiên được camera bắt giữ được nhúng trong nội dung hỗn hợp này, việc lấy mẫu sắc độ 4:2:0 có thể đủ để cho chất lượng

không tổn hao về mặt cảm tính. Đây là do thực tế rằng thị giác con người ít nhạy cảm với các thay đổi không gian ở các thành phần sắc độ so với các thay đổi từ các thành phần so với thay đổi từ các thành phần độ chói. Ở đây, việc lấy mẫu phụ thường được thực hiện trên các thành phần sắc độ (chẳng hạn, định dạng 4:2:0 phổ thông) để đạt được việc giảm tốc độ bit nhận thấy được trong khi duy trì chất lượng hình ảnh được tái tạo giống hệt.

Các phương án thực hiện sáng chế đề xuất cờ, enable_chroma_subsampling, được định nghĩa và báo hiệu ở mức CU một cách đệ quy. Đối với mỗi CU, bộ mã hóa xác định liệu có được mã hóa nhờ sử dụng 4:2:0 hoặc 4:4:4 theo chi phí R-D. Fig.14A và Fig.14B minh họa các ví dụ về các định dạng lấy mẫu sắc độ 4:2:0 và 4:4:4. Fig.14A thể hiện ví dụ lấy mẫu 4:2:0 và Fig.14B thể hiện ví dụ lấy mẫu 4:4:4.

Ở phía bộ mã hóa, cho mỗi CU, giả sử đầu vào là nguồn 4:4:4 được thể hiện trên Fig.14B, chi phí R-D được suy ra trực tiếp nhờ sử dụng thủ tục mã hóa 4:4:4 với enable_chroma_subsampling = 0 hoặc FALSE. Sau đó, quá trình lấy mẫu phụ các mẫu 4:4:4 đến 4:2:0 để suy ra việc tiêu thụ bit. Định dạng 4:2:0 được tái tạo được nội suy ngược về định dạng 4:4:4 để đo độ biến dạng (chẳng hạn, sử dụng tổng lỗi bình phương (SSE), hoặc tổng hiệu số tuyệt đối (SAD)). Cùng với tiêu thụ bit, chi phí R-D được suy ra khi mã hóa CU ở không gian 4:2:0 và so sánh nó với chi phí khi mã hóa CU ở 4:4:4. Sau đó, phương pháp mã hóa bất kể nào cho chi phí R-D thấp hơn được chọn để mã hóa cuối cùng.

Fig.15 minh họa ví dụ của quá trình nội suy từ 4:4:4 đến 4:2:0 và ngược lại. Thông thường, quá trình biến đổi định dạng lấy mẫu màu video có thể yêu cầu số lượng bộ lọc nội suy lớn. Để giảm độ phức tạp triển khai, bộ lọc nội suy HEVC (tức là, DCT-IF) có thể được sử dụng. Như được thể hiện trên Fig.15, các hộp vuông biểu thị các mẫu 4:4:4 ban đầu. Từ 4:4:4 đến 4:2:0, điểm ảnh nửa điểm (được biểu thị bằng các đường tròn) được nội suy sử dụng DCT-IF thẳng đứng cho các thành phần sắc độ. Còn được thể hiện trên Fig.15 là các vị trí một phần điểm ảnh, được biểu thị bằng các hình kim cương. Các đường tròn sắc độ xám được lựa chọn để tạo các mẫu 4:2:0. Để nội suy từ 4:2:0 đến 4:4:4,

quá trình bắt đầu với các đường tròn xám ở các thành phần sắc độ, các vị trí nửa điểm ảnh được nội suy ngang để thu được tất cả các đường tròn, và sau đó các ô vuông được nội suy nhờ sử dụng DCT-IF theo phương thẳng đứng. Tất cả các ô vuông được nội suy được lựa chọn để tạo tín hiệu 4:4:4 được tái tạo.

Điều khiển bộ mã hóa

Như nêu trên, nhiều cờ được đề xuất để điều khiển xử lý mức độ thấp ở bộ mã hóa. Chẳng hạn, enable_packed_component_flag được sử dụng để chỉ báo liệu CU hiện tại có sử dụng định dạng được đóng gói hoặc định dạng phẳng đã biết để xử lý mã hóa. Quyết định liệu có cho phép định dạng đóng gói hay không có thể tùy thuộc vào chi phí R-D được tính toán ở bộ mã hóa. Khi triển khai bộ mã hóa, có thể đạt được giải pháp độ phức tạp thấp nhờ phân tích biểu đồ của CU và tìm ngưỡng tốt nhất cho quyết định.

Kích thước của bảng màu có tác động trực tiếp lên độ phức tạp. Tham số maxColorNum được đưa vào để điều khiển cân bằng giữa độ phức tạp và hiệu quả mã hóa. Cách dễ nhất là chọn lựa dẫn đến chi phí R-D thấp nhất. Hướng bản đồ chỉ mục mã hóa có thể được xác định nhờ tối ưu hóa R-D, hoặc bằng cách sử dụng định hướng không gian cục bộ (chẳng hạn, ước lượng hướng biên sử dụng bộ vận hành Sobel).

Một số phương án thực hiện nêu trên có thể giới hạn xử lý trong mỗi CTU hoặc CU. Thực tế, có thể nới lỏng ràng buộc này. Chẳng hạn, để xử lý bản đồ chỉ mục màu, bộ đệm dòng từ CU trên hoặc CU trái có thể được sử dụng, như được thể hiện trên Fig.16. Fig.16 minh họa ví dụ xử lý bản đồ chỉ mục màu sử dụng bộ đệm dòng chỉ mục trên hoặc bộ đệm dòng chỉ mục trái. Với bộ đệm trên và bộ đệm trái, tìm kiếm có thể được mở rộng để cải thiện thêm hiệu quả mã hóa. Giả sử các bộ đệm trên và trái được tạo nhờ sử dụng điểm ảnh được tái tạo từ các CU lân cận, các điểm ảnh này (cũng như các chỉ mục tương ứng của nó) khả dụng để tham chiếu trước khi xử lý CU hiện tại bản đồ chỉ mục. Chẳng hạn, như được thể hiện trên Fig.16, sau khi sắp lại trật tự, CU hiện tại bản đồ chỉ mục 1600 có thể là 14, 14, 14, ... 1, 2, 1 (được biểu thị dưới dạng chuỗi

1D). Không có tham chiếu bộ đệm dòng, "14" thứ nhất có thể được mã hóa như là cặp không so khớp. Tuy nhiên, với bộ đệm dòng lân cận, "14" thứ nhất so khớp "14" trong bộ đệm dòng chỉ mục trên hoặc bộ đệm dòng chỉ mục trái. Do vậy, sao chép chuỗi có thể bắt đầu ở điểm ảnh thứ nhất.

Cú pháp bộ giải mã

Thông tin dưới đây có thể được sử dụng để mô tả các hoạt động giải mã của bộ nhận 200 được thể hiện trên Fig.2. Cú pháp được thể hiện dưới đây được căn thẳng với bản dự thảo của HEVC REext.

7.3.5.8 Cú pháp CU:

coding_unit(x0, y0, log2CbSize) {	Descriptor
if(transquant_bypass_enabled_flag)	
cu_transquant_bypass_flag	ae(v)
if(slice_type != I)	
cu_skip_flag[x0][y0]	ae(v)
nCbS = (1 << log2CbSize)	
if(cu_skip_flag[x0][y0])	
prediction_unit(x0, y0, nCbS, nCbS)	
else {	
if(intra_block_copy_enabled_flag)	
intra_bc_flag[x0][y0]	ae(v)
if(color_table_enabled_flag)	
color_table_flag[x0][y0]	ae(v)
if(delta_color_table_enabled_flag)	
delta_**color_table_flag**[x0][y0]	ae(v)
if(!intra_bc_flag[x0][y0]) {	
if(slice_type != I)	
pred_mode_flag	ae(v)
if(CuPredMode[x0][y0] != MODE_INTRA || log2CbSize == MinCbLog2SizeY)	
part_mode	ae(v)
}	
if(CuPredMode[x0][y0] == MODE_INTRA) {	
if(PartMode == PART_2Nx2N && pcm_enabled_flag && !intra_bc_flag log2CbSize >= Log2MinIpcmCbSizeY && log2CbSize <= Log2MaxIpcmCbSizeY)	
pcm_flag[x0][y0]	ae(v)
if(pcm_flag[x0][y0]) {	
while(!byte_aligned( ))	

pcm_alignment_zero_bit	f(1)
pcm_sample(x0, y0, log2CbSize)	
} else if(intra_bc_flag[x0][y0]) {	
mvd_coding(x0, y0, 2)	
} else if(color_table_flag[x0][y0] || delta_color_table_flag[x0][y0]) {	
enable_packed_component_flag	ae(v)
if(color_table_flag[x0][y0]) {	
color_table_merge_flag	ae(v)
if (color_table_merge_flag){	
color_table_merge_idx	ae(v)
}else{	
color_table_size	ae(v)
for(i=0;i< color_table_size;i++)	
color_table_entry[i]	ae(v)
}	
color_idx_map_pred_direction	ae(v)
}	
if(delta_color_table_flag[x0][y0]) {	
delta_color_table_adaptive_flag	ae(v)
delta_color_table_merge_flag	ae(v)
if (delta_color_table_merge_flag){	
delta_color_table_merge_idx	ae(v)
}else if (!delta_color_table_adaptive_flag){	
delta_color_table_size	ae(v)
for(i=0;i< delta_color_table_size;i++)	
delta_color_table_entry[i]	ae(v)
}	
}	
Pos=0; cuWidth=1<<log2CbSize; cuHeight=1<<log2CbSize; while (Pos<cuWidth*cuHeight){	
matched_flag	ae(v)
if(matched_flag) {	
matched_distance /*MVx, MVy*/	ae(v)
matched_length	ae(v)
}else{	
index_delta	ae(v)
}	
}	
} else {	
.....pbOffset = (PartMode == PART_NxN) ? (nCbS / 2) : nCbS	
....	

Fig.17 minh họa phương pháp mã hóa nội dung màn hình theo sáng chế. Phương pháp 1700 được thể hiện trên Fig.17 dựa trên các khái niệm chủ yếu nêu trên. Phương pháp 1700 có thể được thực hiện bởi bộ truyền 100 trên Fig.1. Tuy nhiên, phương pháp 1700 có thể còn được sử dụng với thiết bị thích hợp bất kỳ khác hoặc hệ thống.

Ở hoạt động 1701, thiết bị suy ra bản đồ chỉ mục màu dựa trên CU hiện tại. Ở hoạt động 1703, thiết bị mã hóa bản đồ chỉ mục màu. Thiết bị mã hóa ít nhất một phần của bản đồ chỉ mục màu nhờ sử dụng kỹ thuật mã hóa thứ nhất. Bộ chỉ báo thứ nhất biểu thị khoảng cách có nghĩa của kỹ thuật mã hóa thứ nhất. Chẳng hạn, theo một số phương án thực hiện, giá trị thứ nhất của bộ chỉ báo thứ nhất biểu thị kỹ thuật mã hóa IndexMode mà sử dụng khoảng cách có nghĩa bằng 1, và giá trị thứ hai của bộ chỉ báo thứ nhất biểu thị kỹ thuật mã hóa CopyAbove mà sử dụng khoảng cách có nghĩa bằng chiều rộng khối của CU hiện tại.

Một phần của bản đồ chỉ mục màu mà thiết bị mã hóa bằng cách sử dụng kỹ thuật mã hóa thứ nhất là chuỗi chỉ mục thứ nhất có chuỗi chỉ mục thứ hai so khớp ngay phía trên chuỗi chỉ mục thứ nhất ở CU hiện tại, hoặc chuỗi chỉ mục thứ ba đều có giá trị tương tự như giá trị chỉ mục tham chiếu ngay sang trái của chỉ mục thứ nhất trong số chuỗi chỉ mục thứ ba ở CU hiện tại.

Ở hoạt động 1705, thiết bị kết hợp bản đồ chỉ mục màu được mã hóa và bộ chỉ báo thứ nhất để truyền đến bộ nhận.

Mặc dù Fig.17 minh họa một ví dụ về phương pháp 1700 để mã hóa nội dung màn hình, có thể thực hiện các thay đổi khác nhau với Fig.17. Chẳng hạn, trong khi được thể hiện dưới dạng một chuỗi các bước, các bước khác nhau được thể hiện trên Fig.17 có thể trùng lặp, xuất hiện song song, xuất hiện theo thứ tự khác nhau, hoặc xuất hiện nhiều lần. Ngoài ra, một số bước có thể được kết hợp hoặc loại bỏ và các bước bổ sung có thể được thêm theo các nhu cầu thực.

Fig.18 minh họa phương pháp giải mã nội dung màn hình theo sáng chế. Phương pháp 1800 được thể hiện trên Fig.18 dựa trên các khái niệm chính nêu

trên. Phương pháp 1800 có thể được thực hiện bởi bộ nhận 200 trên Fig.2. Tuy nhiên, phương pháp 1800 còn có thể được sử dụng với thiết bị hoặc hệ thống thích hợp khác bất kỳ.

Ở hoạt động 1801, thiết bị tiếp nhận dòng bit video nén từ bộ truyền. Dòng bit video gồm bản đồ chỉ mục màu được mã hóa. Thiết bị còn tiếp nhận bộ chỉ báo thứ nhất. Bộ chỉ báo thứ nhất biểu thị khoảng cách có nghĩa của kỹ thuật giải mã thứ nhất. Chẳng hạn, theo một số phương án thực hiện, giá trị thứ nhất của bộ chỉ báo thứ nhất biểu thị kỹ thuật giải mã IndexMode mà sử dụng khoảng cách có nghĩa bằng 1, và giá trị thứ hai của bộ chỉ báo thứ nhất biểu thị kỹ thuật giải mã CopyAbove mà sử dụng khoảng cách có nghĩa bằng chiều rộng khối của CU hiện tại.

Ở hoạt động 1803, thiết bị giải mã ít nhất một phần của bản đồ chỉ mục màu sử dụng kỹ thuật giải mã thứ nhất, trong đó bộ chỉ báo thứ nhất biểu thị khoảng cách có nghĩa của kỹ thuật giải mã thứ nhất. Sau đó, ở hoạt động 1805, thiết bị tái tạo điểm ảnh cùng với CU hiện tại dựa trên bản đồ chỉ mục màu.

Mặc dù Fig.18 minh họa một ví dụ về phương pháp 1800 để giải mã nội dung màn hình, có thể thực hiện các thay đổi khác nhau đối với Fig.18. Chẳng hạn, trong khi được thể hiện dưới dạng chuỗi các bước, các bước khác nhau được thể hiện trên Fig.18 có thể trùng lặp, xuất hiện song song, xuất hiện theo thứ tự khác, hoặc xuất hiện nhiều lần. Ngoài ra, một số bước có thể được kết hợp hoặc loại bỏ và các bước bổ sung có thể được thêm theo các nhu cầu thực.

Theo một số phương án thực hiện, một số hoặc tất cả các chức năng hoặc các quá trình của một hoặc nhiều thiết bị được triển khai được triển khai hoặc được hỗ trợ bởi chương trình máy tính được tạo trên mã chương trình máy tính đọc được và được triển khai trong vật lưu trữ máy tính đọc được. Thuật ngữ "mã chương trình máy tính đọc được" gồm loại mã máy tính bất kỳ, gồm mã nguồn, mã đối tượng, và mã thực thi được. Thuật ngữ "vật lưu trữ máy tính đọc được" gồm loại môi trường bất kỳ có thể truy nhập bởi máy tính, như bộ nhớ chỉ đọc (read only memory, ROM), bộ nhớ truy xuất ngẫu nhiên (random

access memory, RAM), ổ đĩa cứng, đĩa CD, DVD, hoặc loại bộ nhớ khác bất kỳ.

Có thể có lợi khi nêu các định nghĩa về các từ cụ thể và các thuật ngữ được sử dụng trong suốt bản mô tả. Các thuật ngữ "gồm" cũng như các dẫn xuất của nó, nghĩa bao gồm không giới hạn. Thuật ngữ "hoặc" bao gồm, nghĩa và/hoặc. Các thuật ngữ "cùng với" và "liên kết với," cùng như các dẫn xuất của nó, nghĩa gồm, được bao gồm trong, liên kết với, chứa, được chứa trong, nối với, ghép nối với, truyền thông được với, phối hợp với, đan xen, kề nhau, gần với, bị giới hạn ở, có, có thuộc tính, hoặc tương tự.

Trong khi sáng chế đã mô tả các phương án thực hiện cụ thể và các phương pháp liên kiết chung, các thay thế và hoán vị của các phương án thực hiện và các phương pháp sẽ là rõ ràng với người có hiểu biết trung bình trong lĩnh vực. Do đó, phần mô tả trên đây của các phương án thực hiện làm ví dụ không định nghĩa hoặc giới hạn sáng chế. Các thay đổi, thay thế, và còn có thể mà không xa rời bản chất và phạm vi sáng chế, như được định nghĩa bởi các điểm yêu cầu bảo hộ sau.

YÊU CẦU BẢO HỘ

1. Phương pháp mã hóa nội dung màn hình, trong đó phương pháp bao gồm các bước:

dẫn xuất bản đồ chỉ số màu dựa trên khối mã hóa (coding unit, CU) hiện tại;

mã hóa bản đồ chỉ số màu, trong đó ít nhất một phần bản đồ chỉ số màu được mã hóa nhờ sử dụng kỹ thuật mã hóa thứ nhất, trong đó bộ chỉ báo thứ nhất chỉ báo khoảng cách quan trọng của kỹ thuật mã hóa thứ nhất; và

kết hợp bản đồ chỉ số màu được mã hóa và bộ chỉ báo thứ nhất để truyền đến bộ nhận;

trong đó giá trị thứ nhất của bộ chỉ báo thứ nhất chỉ báo kỹ thuật mã hóa IndexMode mà sử dụng khoảng cách quan trọng bằng 1, và giá trị thứ hai của bộ chỉ báo thứ nhất chỉ báo kỹ thuật mã hóa CopyAbove mà sử dụng khoảng cách quan trọng bằng chiều rộng khối của CU hiện tại.

2. Phương pháp theo điểm 1, trong đó ít nhất một phần bản đồ chỉ số màu được mã hóa nhờ sử dụng kỹ thuật mã hóa thứ nhất là một trong:
chuỗi chỉ số thứ nhất mà có chuỗi chỉ số thứ hai so khớp ngay phía trên chuỗi chỉ số thứ nhất trong CU hiện tại; hoặc
chuỗi chỉ số thứ ba đều có giá trị giống như giá trị chỉ số tham chiếu ngay bên trái của chỉ số thứ nhất trong số chuỗi chỉ số thứ ba trong CU hiện tại.

3. Phương pháp theo điểm 2, trong đó chuỗi chỉ số thứ nhất được mã hóa nhờ sử dụng kỹ thuật mã hóa CopyAbove, và đầu ra của kỹ thuật mã hóa CopyAbove bao gồm chiều dài của chuỗi chỉ số thứ nhất.

4. Phương pháp theo điểm 2, trong đó chuỗi chỉ số thứ ba được mã hóa nhờ sử dụng kỹ thuật mã hóa IndexMode, và đầu ra của kỹ thuật mã hóa IndexMode bao gồm chiều dài của chuỗi chỉ số thứ ba.

5. Phương pháp theo điểm 1, trong đó bộ chỉ báo thứ hai chỉ báo rằng ít nhất một phần bản đồ chỉ số màu được mã hóa nhờ sử dụng kỹ thuật mã hóa thứ nhất thay vì kỹ thuật mã hóa thứ hai.

6. Phương pháp theo điểm 5, trong đó:
các bộ chỉ báo thứ nhất và thứ hai lần lượt bao gồm các cờ nhị phân thứ nhất và thứ hai;
cờ nhị phân thứ hai chỉ báo rằng kỹ thuật mã hóa thứ nhất được sử dụng;
cờ nhị phân thứ nhất chỉ báo rằng khoảng cách quan trọng bằng chiều rộng khối của CU hiện tại; và
đường mã hóa của CU hiện tại giống hệt đường nêu trên được báo hiệu nhờ sử dụng chỉ các cờ nhị phân thứ nhất và thứ hai.

7. Thiết bị được tạo cấu hình để mã hóa nội dung màn hình, trong đó thiết bị bao gồm:

ít nhất một bộ nhớ; và

ít nhất một bộ xử lý được ghép nối với ít nhất một bộ nhớ, ít nhất một bộ xử lý được tạo cấu hình để:

dẫn xuất bản đồ chỉ số màu dựa trên CU hiện tại;

mã hóa bản đồ chỉ số màu, trong đó ít nhất một phần bản đồ chỉ số màu được mã hóa nhờ sử dụng kỹ thuật mã hóa thứ nhất, trong đó bộ chỉ báo thứ nhất chỉ báo khoảng cách quan trọng của kỹ thuật mã hóa thứ nhất; và

kết hợp bản đồ chỉ số màu được mã hóa và bộ chỉ báo thứ nhất để truyền đến bộ nhận;

trong đó giá trị thứ nhất của bộ chỉ báo thứ nhất chỉ báo kỹ thuật mã hóa IndexMode mà sử dụng khoảng cách quan trọng bằng 1, và giá trị thứ hai của bộ chỉ báo thứ nhất chỉ báo kỹ thuật mã hóa CopyAbove mà sử dụng khoảng cách quan trọng bằng chiều rộng khối của CU hiện tại.

8. Thiết bị theo điểm 7, trong đó ít nhất một phần bản đồ chỉ số màu được mã hóa nhờ sử dụng kỹ thuật mã hóa thứ nhất là một trong:

chuỗi chỉ số thứ nhất mà có chuỗi chỉ số thứ hai so khớp ngay phía trên chuỗi chỉ số thứ nhất trong CU hiện tại; hoặc

chuỗi chỉ số thứ ba đều có giá trị giống như giá trị chỉ số tham chiếu ngay bên trái của chỉ số thứ nhất trong số chuỗi chỉ số thứ ba trong CU hiện tại.

9. Thiết bị theo điểm 8, trong đó chuỗi chỉ số thứ nhất được mã hóa nhờ sử dụng kỹ thuật mã hóa CopyAbove, và đầu ra của kỹ thuật mã hóa CopyAbove bao gồm chiều dài của chuỗi chỉ số thứ nhất.

10. Thiết bị theo điểm 8, trong đó chuỗi chỉ số thứ ba được mã hóa nhờ sử dụng kỹ thuật mã hóa IndexMode, và đầu ra của kỹ thuật mã hóa IndexMode bao gồm chiều dài của chuỗi chỉ số thứ ba.

11. Thiết bị theo điểm 7, trong đó bộ chỉ báo thứ hai chỉ báo rằng ít nhất một phần bản đồ chỉ số màu được mã hóa sử dụng kỹ thuật mã hóa thứ nhất thay vì kỹ thuật mã hóa thứ hai.

12. Thiết bị theo điểm 11, trong đó:

các bộ chỉ báo thứ nhất và thứ hai lần lượt bao gồm các cờ nhị phân thứ nhất và thứ hai;

cờ nhị phân thứ hai chỉ báo rằng kỹ thuật mã hóa thứ nhất được sử dụng;

cờ nhị phân thứ nhất chỉ báo rằng khoảng cách quan trọng bằng chiều rộng khối của CU hiện tại; và

đường mã hóa của CU hiện tại mà có giá trị giống hệt được báo hiệu nhờ sử dụng chỉ các cờ nhị phân thứ nhất và thứ hai.

13. Phương pháp giải mã nội dung màn hình, trong đó phương pháp bao gồm các bước:

nhận dòng bit viđeo bao gồm bản đồ chỉ số màu;

nhận bộ chỉ báo thứ nhất;

giải mã ít nhất một phần bản đồ chỉ số màu nhờ sử dụng kỹ thuật giải mã thứ nhất, trong đó bộ chỉ báo thứ nhất chỉ báo khoảng cách quan trọng của kỹ thuật giải mã thứ nhất; và

tái tạo các điểm ảnh được liên kết với CU hiện tại dựa trên bản đồ chỉ số màu;

trong đó giá trị thứ nhất của bộ chỉ báo thứ nhất chỉ báo kỹ thuật giải mã IndexMode mà sử dụng khoảng cách quan trọng bằng 1, và giá trị thứ hai của bộ chỉ báo thứ nhất chỉ báo kỹ thuật giải mã CopyAbove mà sử dụng khoảng cách quan trọng bằng chiều rộng khối của CU hiện tại.

14. Phương pháp theo điểm 13, trong đó ít nhất một phần bản đồ chỉ số màu được giải mã nhờ sử dụng kỹ thuật giải mã thứ nhất là một trong:

chuỗi chỉ số thứ nhất mà có chuỗi chỉ số thứ hai so khớp ngay phía trên chuỗi chỉ số thứ nhất trong CU hiện tại; hoặc

chuỗi chỉ số thứ ba đều có giá trị giống như giá trị chỉ số tham chiếu ngay bên trái của chỉ số thứ nhất trong số chuỗi chỉ số thứ ba trong CU hiện tại.

15. Phương pháp theo điểm 14, trong đó chuỗi chỉ số thứ nhất được giải mã nhờ sử dụng kỹ thuật giải mã CopyAbove, và đầu vào của kỹ thuật giải mã CopyAbove bao gồm chiều dài của chuỗi chỉ số thứ nhất.

16. Phương pháp theo điểm 14, trong đó chuỗi chỉ số thứ ba được giải mã nhờ sử dụng kỹ thuật giải mã IndexMode, và đầu vào của kỹ thuật mã hóa IndexMode bao gồm chiều dài của chuỗi chỉ số thứ ba.

17. Phương pháp theo điểm 13, trong đó bộ chỉ báo thứ hai được nhận chỉ báo rằng ít nhất một phần bản đồ chỉ số màu được giải mã nhờ sử dụng kỹ thuật giải mã thứ nhất thay vì kỹ thuật giải mã thứ hai.

18. Phương pháp theo điểm 17, trong đó:

các bộ chỉ báo thứ nhất và thứ hai lần lượt bao gồm các cờ nhị phân thứ nhất và thứ hai;

cờ nhị phân thứ hai chỉ báo rằng kỹ thuật giải mã thứ nhất được sử dụng;

cờ nhị phân thứ nhất chỉ báo rằng khoảng cách quan trọng bằng chiều rộng khối của CU hiện tại; và

đường mã hóa của CU hiện tại giống hệt đường nêu trên được báo hiệu nhờ sử dụng chỉ các cờ nhị phân thứ nhất và thứ hai.

19. Thiết bị được tạo cấu hình để giải mã nội dung màn hình, thiết bị bao gồm:

ít nhất một bộ nhớ; và

ít nhất một bộ xử lý được ghép nối với ít nhất một bộ nhớ, ít nhất một bộ xử lý được tạo cấu hình để:

nhận dòng bit viđeo bao gồm bản đồ chỉ số màu;

nhận bộ chỉ báo thứ nhất;

giải mã ít nhất một phần bản đồ chỉ số màu nhờ sử dụng kỹ thuật giải mã thứ nhất, trong đó bộ chỉ báo thứ nhất chỉ báo khoảng cách quan trọng của kỹ thuật giải mã thứ nhất; và

tái tạo các điểm ảnh được liên kết với CU hiện tại dựa trên bản đồ chỉ số màu;

trong đó giá trị thứ nhất của bộ chỉ báo thứ nhất chỉ báo kỹ thuật giải mã IndexMode mà sử dụng khoảng cách quan trọng bằng 1, và giá trị thứ hai của bộ chỉ báo thứ nhất chỉ báo kỹ thuật giải mã CopyAbove mà sử dụng khoảng cách quan trọng bằng chiều rộng khối của CU hiện tại.

20. Thiết bị theo điểm 19, trong đó ít nhất một phần bản đồ chỉ số màu được giải mã nhờ sử dụng kỹ thuật giải mã thứ nhất là một trong:

chuỗi chỉ số thứ nhất mà có chuỗi chỉ số thứ hai so khớp ngay phía trên chuỗi chỉ số thứ nhất trong CU hiện tại; hoặc

chuỗi chỉ số thứ ba đều có giá trị giống như giá trị chỉ số tham chiếu ngay bên trái của chỉ số thứ nhất trong số chuỗi chỉ số thứ ba trong CU hiện tại.

21. Thiết bị theo điểm 20, trong đó chuỗi chỉ số thứ nhất được giải mã nhờ sử dụng kỹ thuật giải mã CopyAbove, và đầu vào của kỹ thuật giải mã CopyAbove bao gồm chiều dài của chuỗi chỉ số thứ nhất.

22. Thiết bị theo điểm 21, trong đó chuỗi chỉ số thứ ba được giải mã nhờ sử dụng kỹ thuật giải mã IndexMode, và đầu vào của kỹ thuật mã hóa IndexMode bao gồm chiều dài của chuỗi chỉ số thứ ba.

23. Thiết bị theo điểm 19, trong đó bộ chỉ báo thứ hai chỉ báo rằng ít nhất một phần bản đồ chỉ số màu được giải mã nhờ sử dụng kỹ thuật giải mã thứ nhất thay cho kỹ thuật giải mã thứ hai.

24. Thiết bị theo điểm 23, trong đó:

các bộ chỉ báo thứ nhất và thứ hai lần lượt bao gồm các cờ nhị phân thứ nhất và thứ hai;

cờ nhị phân thứ hai chỉ báo rằng kỹ thuật giải mã thứ nhất được sử dụng;

cờ nhị phân thứ nhất chỉ báo rằng khoảng cách quan trọng bằng chiều rộng khối của CU hiện tại; và

đường mã hóa của CU hiện tại mà có giá trị giống hệt được báo hiệu nhờ sử dụng chỉ các cờ nhị phân thứ nhất và thứ hai.

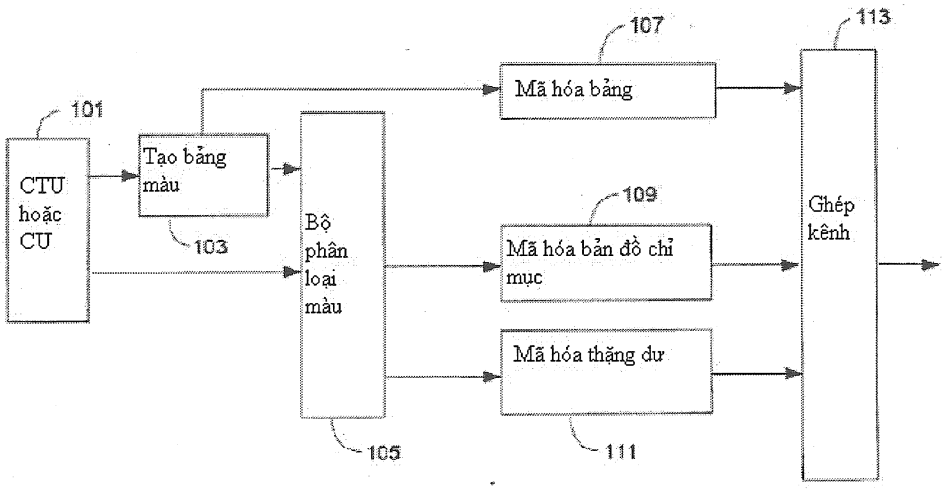

101
CTU hoặc CU
Tạo bảng màu
103
Bộ phân loại màu
105
107
Mã hóa bảng
109
Mã hóa bản đồ chỉ mục
Mã hóa thặng dư
111
113
Ghép kênh

Fig.1

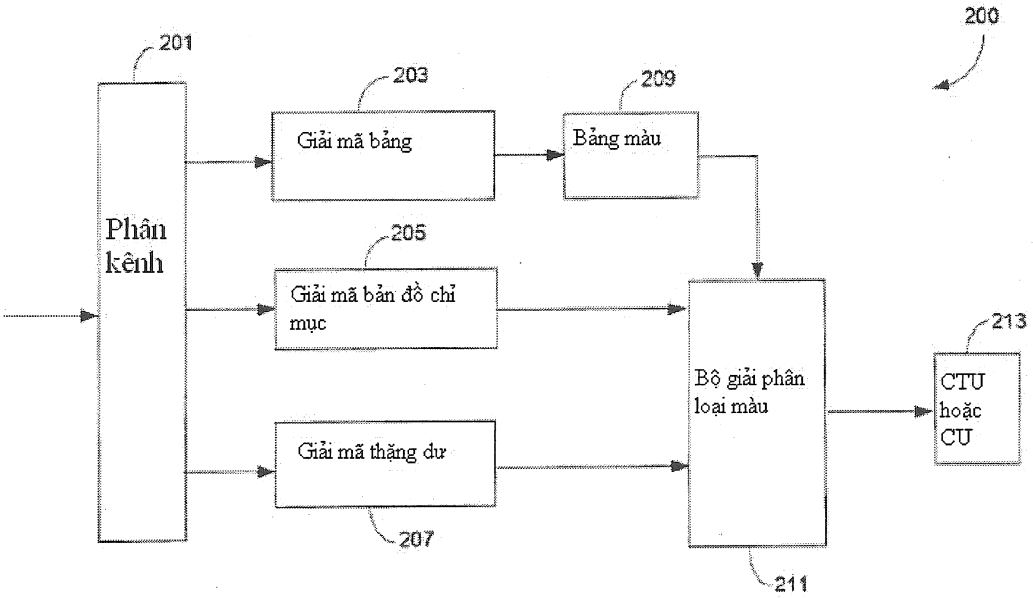

200
201
Phân kênh
203
Giải mã bảng
209
Bảng màu
205
Giải mã bản đồ chỉ mục
Giải mã thặng dư
207
Bộ giải phân loại màu
211
213
CTU hoặc CU

Fig.2

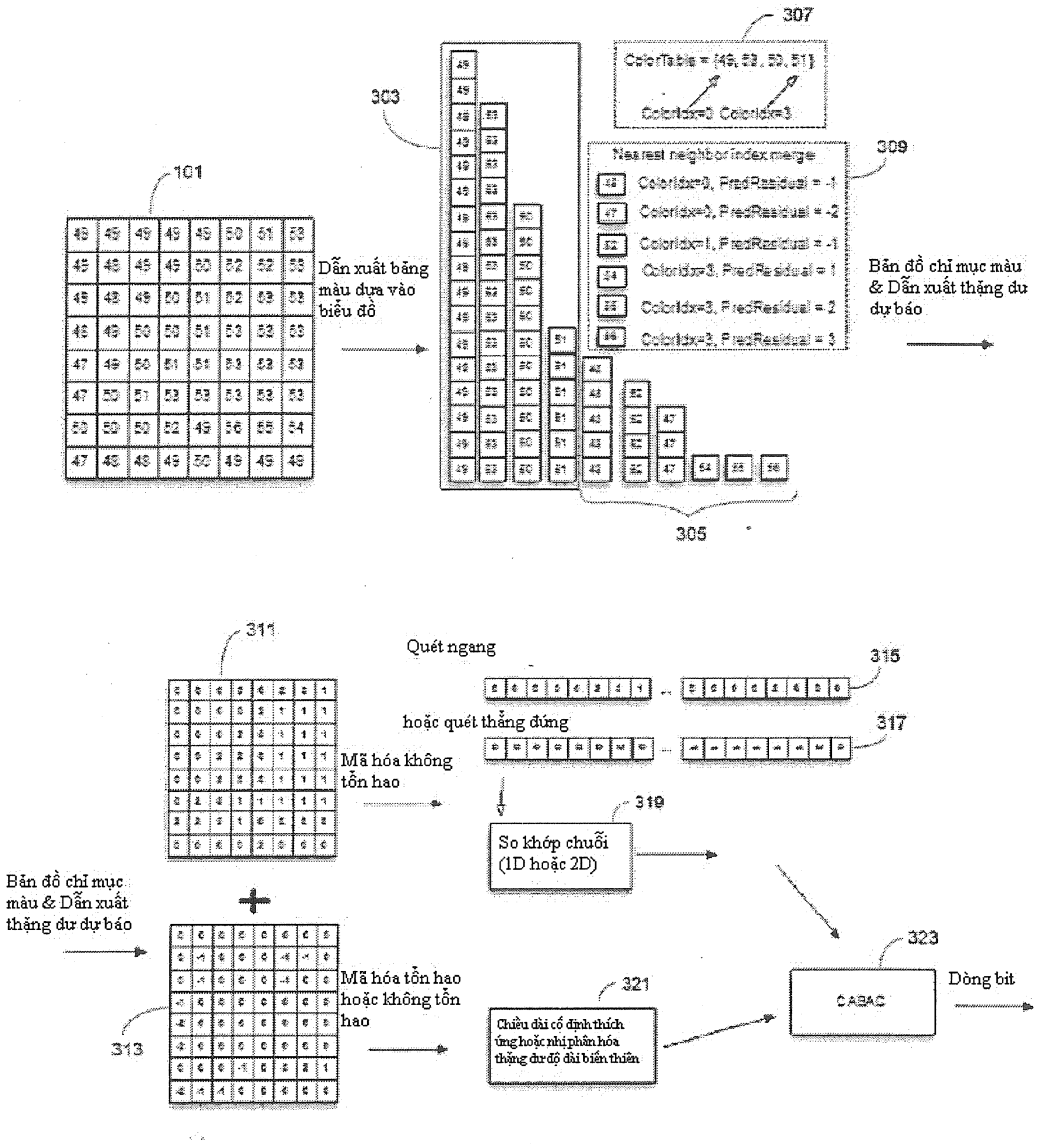

307
303
101
309
Dẫn xuất bảng màu dựa vào biểu đồ
Bản đồ chỉ mục màu & Dẫn xuất thặng dư dự báo
305
311
Quét ngang
315
hoặc quét thẳng đứng
317
Mã hóa không tổn hao
319
So khớp chuỗi (1D hoặc 2D)
Bản đồ chỉ mục màu & Dẫn xuất thặng dư dự báo
323
CABAC
Dòng bit
313
Mã hóa tổn hao hoặc không tổn hao
321
Chiều dài cố định thích ứng hoặc nhị phân hóa thặng dư độ dài biến thiên

Fig.3

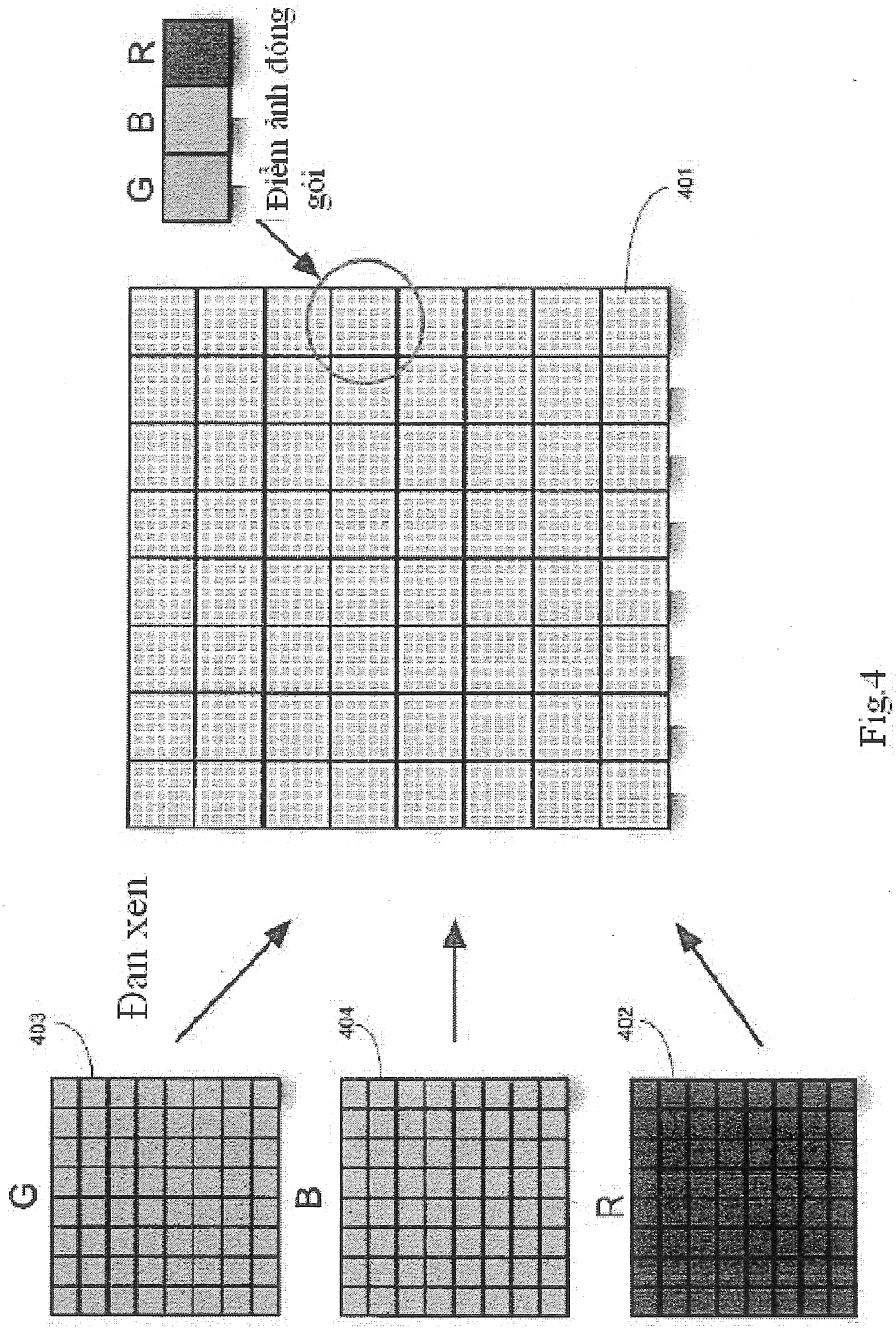


Fig.4

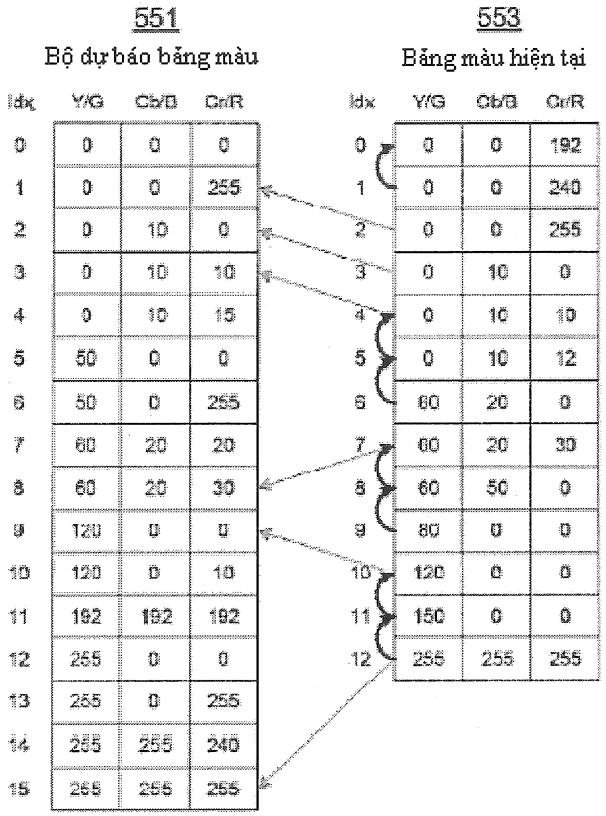

551
Bộ dự báo bảng màu
553
Bảng màu hiện tại
Idx Y/G Cb/B Cr/R
0 0 0 0
1 0 0 255
2 0 10 0
3 0 10 10
4 0 10 15
5 50 0 0
6 50 0 255
7 60 20 20
8 60 20 30
9 120 0 0
10 120 0 10
11 192 192 192
12 255 0 0
13 255 0 255
14 255 255 240
15 255 255 255
Idx Y/G Cb/B Cr/R
0 0 0 192
1 0 0 240
2 0 0 255
3 0 10 0
4 0 10 10
5 0 10 12
6 60 20 0
7 60 20 30
8 60 50 0
9 80 0 0
10 120 0 0
11 150 0 0
12 255 255 255

Fig.5A

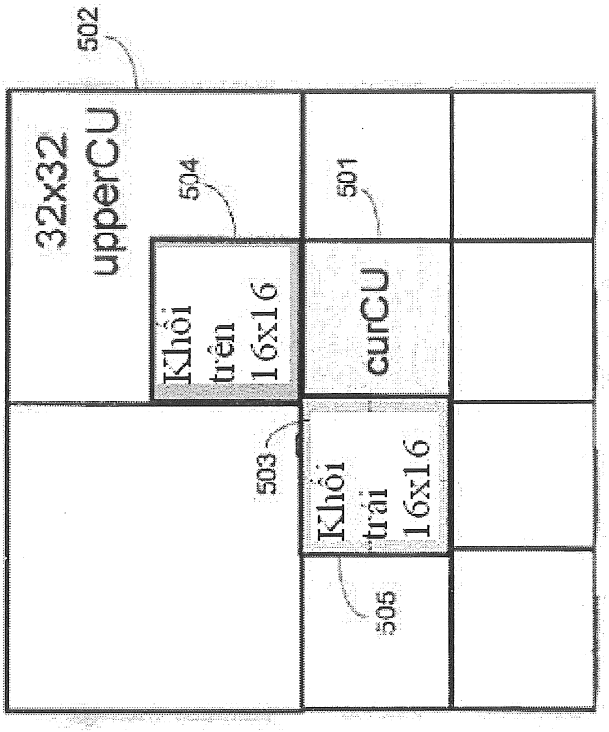

502
32x32
upperCU
504
Khối
trên
16x16
501
curCU
503
Khối
trái
16x16
505

Fig.5B

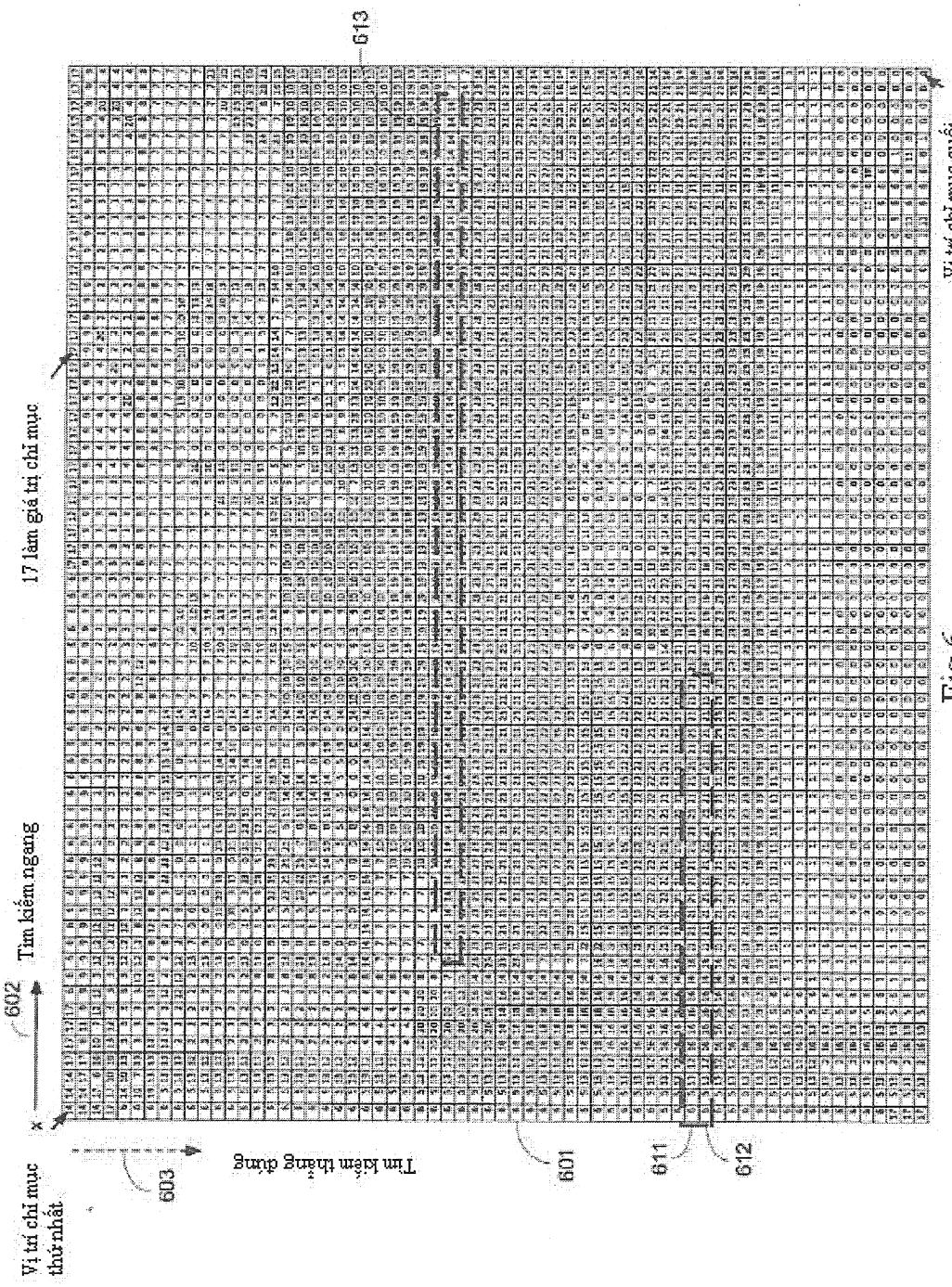

613
17 làm giá trị chỉ mục
Vị trí chỉ mục cuối
602
Tìm kiếm ngang
Vị trí chỉ mục thứ nhất
603
Tìm kiếm thẳng đứng
601
611
612

Fig.6

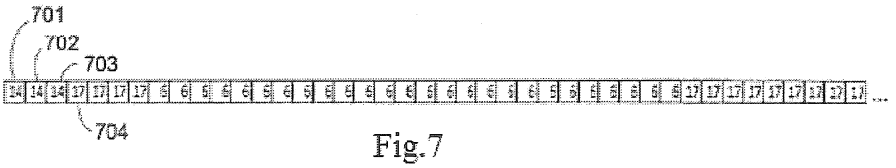


Fig.7

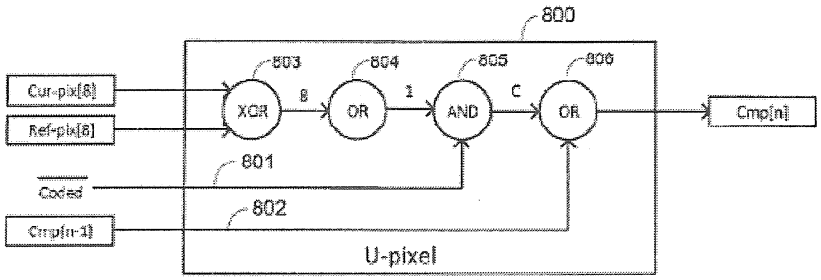


Fig.8

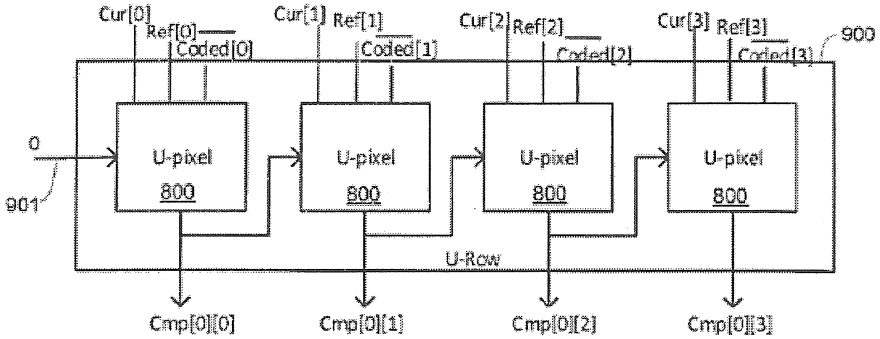


Fig.9

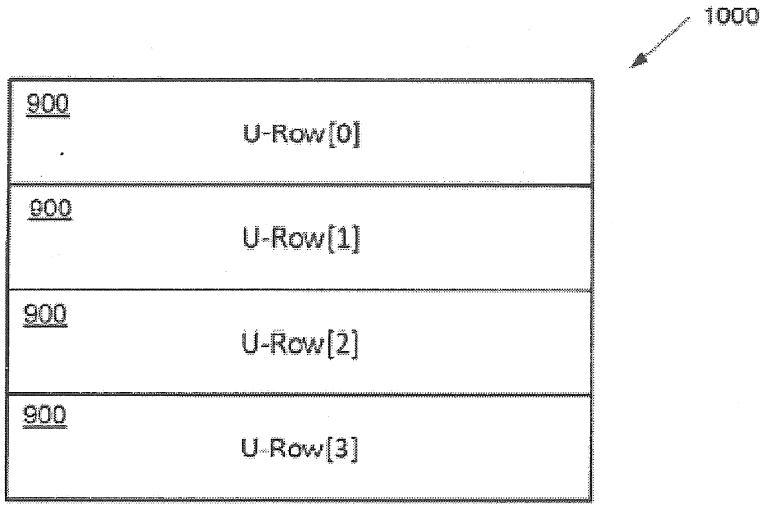

1000
900
U-Row[0]
900
U-Row[1]
900
U-Row[2]
900
U-Row[3]

Fig.10

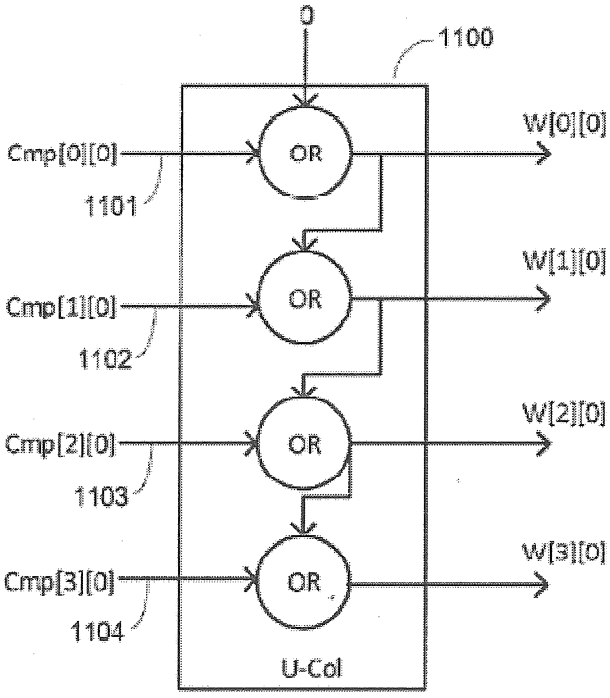

0
1100
Cmp[0][0]
1101
OR
W[0][0]
Cmp[1][0]
1102
OR
W[1][0]
Cmp[2][0]
1103
OR
W[2][0]
Cmp[3][0]
1104
OR
W[3][0]
U-Col

Fig.11

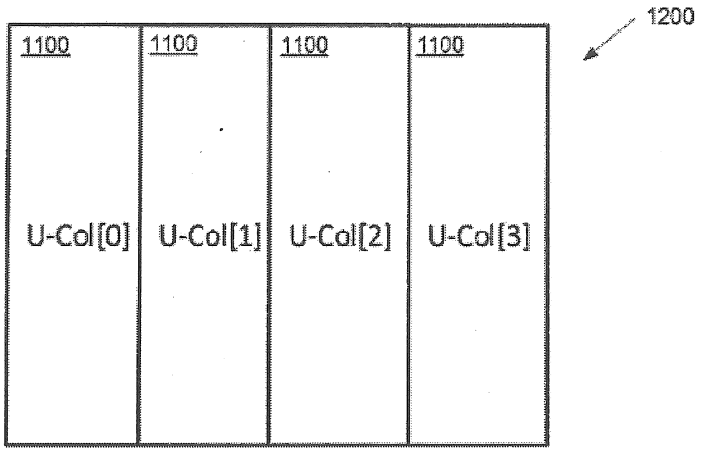

1200
1100
1100
1100
1100
U-Col[0]
U-Col[1]
U-Col[2]
U-Col[3]

Fig.12

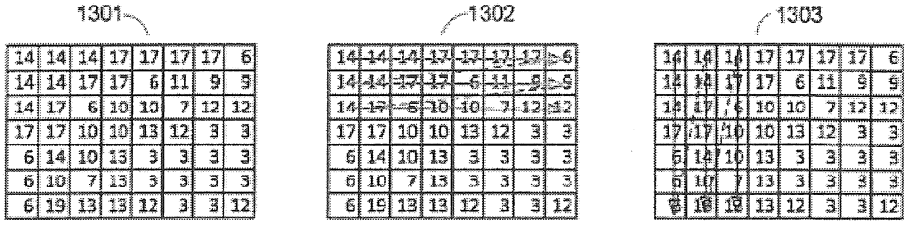

1301
1302
1303

Fig.13

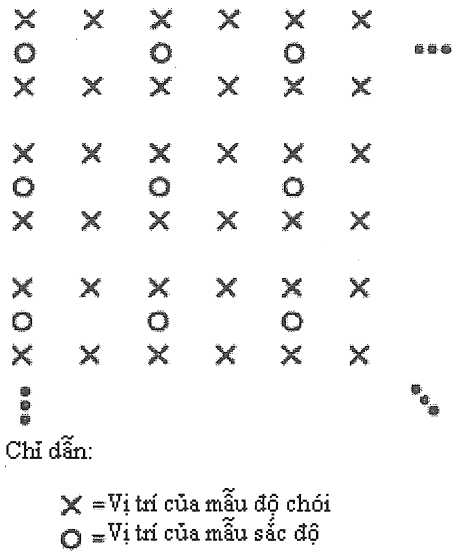


Fig.14A

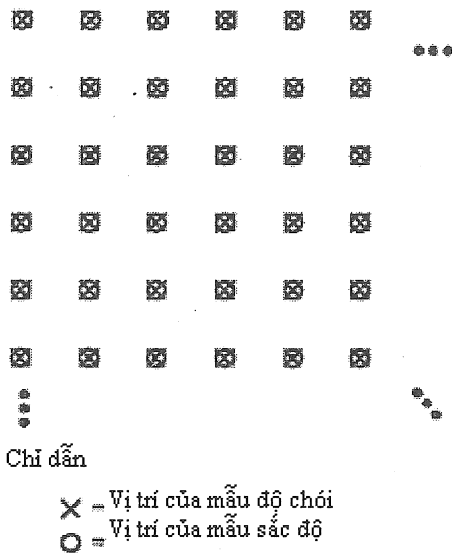


Fig.14B

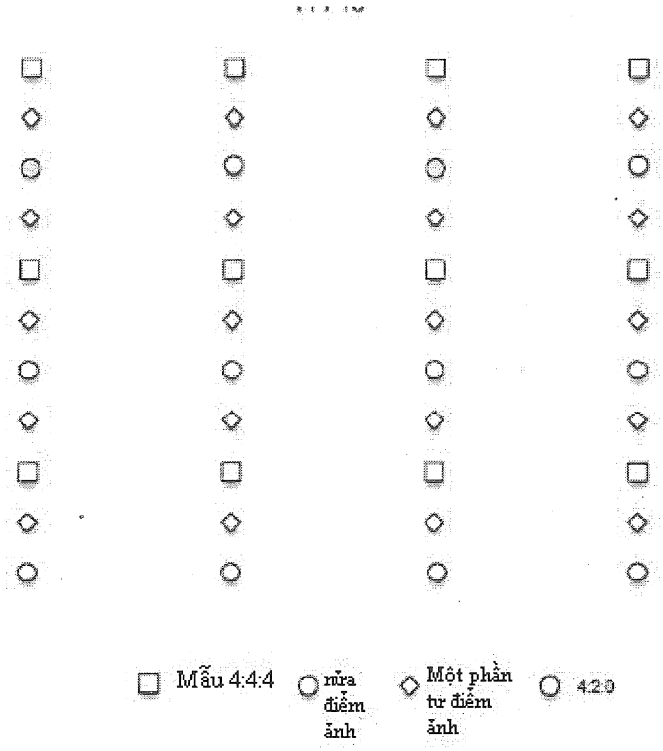

Mẫu 4:4:4
nửa điểm ảnh
Một phần tư điểm ảnh
4:2:0

Fig.15

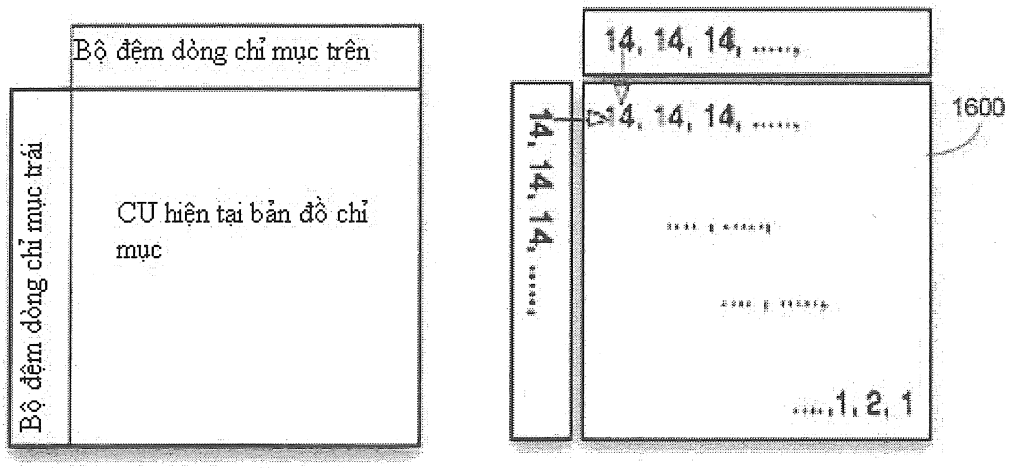

Bộ đệm dòng chỉ mục trên
Bộ đệm dòng chỉ mục trái
CU hiện tại bản đồ chỉ mục
14, 14, 14, ...
14, 14, 14, ...
14, 14, 14, ...
1600
....1, 2, 1

Fig.16

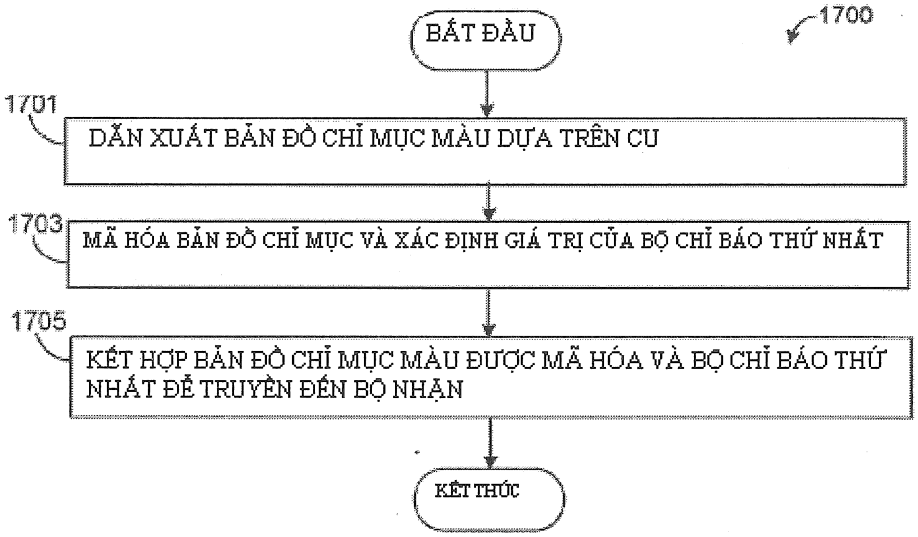

1700
BẮT ĐẦU
1701
DẪN XUẤT BẢN ĐỒ CHỈ MỤC MÀU DỰA TRÊN CU
1703
MÃ HÓA BẢN ĐỒ CHỈ MỤC VÀ XÁC ĐỊNH GIÁ TRỊ CỦA BỘ CHỈ BÁO THỨ NHẤT
1705
KẾT HỢP BẢN ĐỒ CHỈ MỤC MÀU ĐƯỢC MÃ HÓA VÀ BỘ CHỈ BÁO THỨ NHẤT ĐỂ TRUYỀN ĐẾN BỘ NHẬN
KẾT THÚC

Fig.17

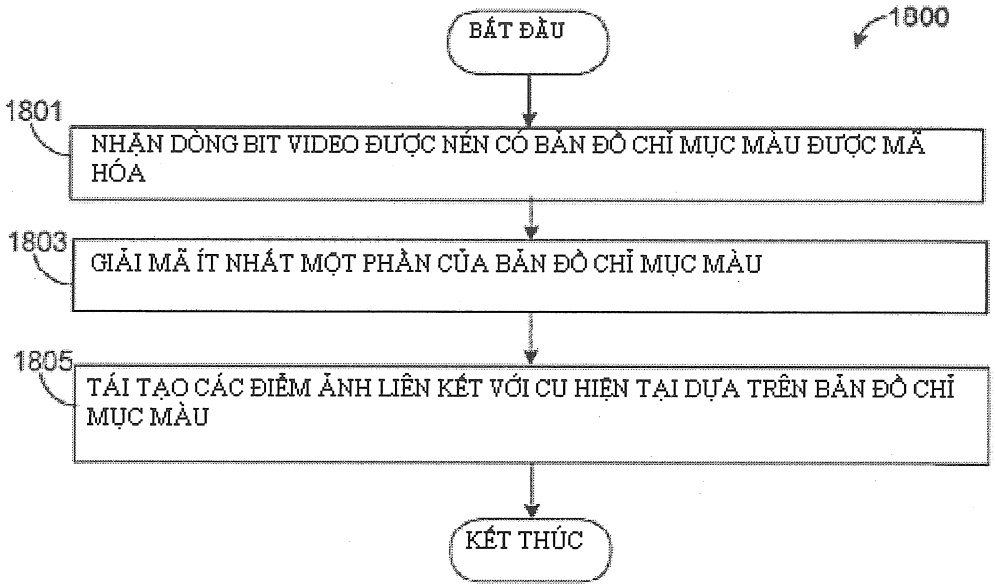

1800
BẮT ĐẦU
1801
NHẬN DÒNG BIT VIDEO ĐƯỢC NÉN CÓ BẢN ĐỒ CHỈ MỤC MÀU ĐƯỢC MÃ HÓA
1803
GIẢI MÃ ÍT NHẤT MỘT PHẦN CỦA BẢN ĐỒ CHỈ MỤC MÀU
1805
TÁI TẠO CÁC ĐIỂM ẢNH LIÊN KẾT VỚI CU HIỆN TẠI DỰA TRÊN BẢN ĐỒ CHỈ MỤC MÀU
KẾT THÚC

Fig.18