DOI: 10.3969/j. issn. 1673-5188. 2015. 04. 008
http://www.cnki.net/kcms/detail/34.1294.TN.20150914.1512.002.html, published online September 14, 2015

Screen Content Coding with Primary and Secondary Reference Buffers for String Matching and Copying

Tao Lin, Kailun Zhou, and Liping Zhao
(Tongji University, Shanghai 200092, China)

Abstract

A screen content coding (SCC) algorithm that uses a primary reference buffer (PRB) and a secondary reference buffer (SRB) for string matching and string copying is proposed. PRB is typically the traditional reconstructed picture buffer which provides reference string pixels for the current pixels being coded. SRB stores a few of recently and frequently referenced pixels for repetitive reference by the current pixels being coded. In the encoder, searching of optimal reference string is performed in both PRB and SRB, and either a PRB or SRB string is selected as an optimal reference string on a string-by-string basis. Compared with HM-16.4+SCM-40 reference software, the proposed SCC algorithm can improve coding performance measured by bit-distortion rate reduction of average 4.19% in all-intra configuration for text and graphics with motion category of test sequences defined by JCT-VC common test condition.

Keywords

HEVC; Image Coding; Screen Content Coding; String Matching; Video Coding

1 Introduction

With continuous and rapid advancements in communications, networking, computers, semiconductors, and displays technologies, screen content coding (SCC) becomes a key technology for some emerging popular applications such as cloud computing, cloud-mobile computing, Wi-Fi display, smartphone and tablet second display, ultra-thin clients, remote desktops, and screen sharing [1]-[3]. The challenging requirement of SCC is to achieve both ultra-high visually lossless quality and ultra-high compression ratio up to 300:1-3000:1. In recent years, SCC has attracted increasing attention of researchers from both academia and industry [4]-[30]. ISO/IEC MPEG and ITU-T VCEG have set up the Joint Collaborative Team (JCT) for a joint project of developing a SCC version of High Efficiency Video Coding (HEVC, also known as H.265) standard and a joint call-for-proposal has been issued in January 2014 [14].

Characteristics of computer screen pictures are quite different from those of traditional pictures, for which traditional block-matching and transform-based hybrid coding technique can compress efficiently. In general, computer screen pictures are essentially a mix of two-type contents: discontinuous-tone content such as text, chart, graphics, and icon, and continuous-tone content such as photograph, movie/TV clips, natural picture video sequences, and sophisticated light-shaded and texture-mapped virtual-reality scenes. Continuous-tone content features relatively smooth edges, complicated textures, and thick lines with virtually unlimited colors. In contrast, discontinuous-tone content features very sharp edges, uncomplicated shapes, and thin lines with few colors, even one-pixel-wide single-color lines. General screen content pictures are often seen in web browsing, video conferencing with document sharing, office document editing, engineering drawing, hardware design engineering, software programming, gaming, map navigating, address direction searching, and other computer use cases.

Actually, typical computer screens for daily use are often rich in small and sharp bitmap structures such as text, menu, icon, button, slide-bar, and grid. There are usually many similar or identical patterns in a screen picture. These similar or identical patterns may have very different sizes from a few pixels to a few thousands of pixels and very different shapes. Existing techniques such as intra-prediction and intra block copy (IBC) [15]-[17] is not always efficient to code similar or identical pattern within a picture, because they use only 1-D pattern or 2-D pattern of a few fixed sizes and a few fixed rectangle or square shapes.

In order to address the issue and improve the coding efficiency of repeated patterns with different sizes and shapes, this paper proposes a coding technique based on string-matching (also called intra string copy or ISC). It provides a very generic and flexible solution to string matching of variable sizes and different shapes.

Two reference buffers are used in the proposed technique. One is primary reference buffer (PRB) that is typically the traditional reconstructed picture buffer to provide reference string pixels for the current pixels being coded. The other is secondary reference buffer (SRB), a lookup table (LUT) storing a few of recently and frequently referenced pixels for repetitive reference by the current pixels being coded. For any starting pixel in an encoding process, searching of optimal reference string is performed in both PRB and SRB. Either a PRB string or a SRB

This work was supported in part by National Natural Science Foundation of China under Grant No. 61201226 and 61271096, Natural Science Foundation of Shanghai under Grant No. 12ZR1433800, and Specialized Research Fund for the Doctoral Program under Grant No. 20130072110054.

string is selected as an optimal reference string on a string-by-string basis. A PRB string has two string-matching parameters. One is an offset that specifies the relative position (2D coordinates), i.e. 2D displacement from the reference string to the current string. The other is a length that specifies the number of pixels in the reference string. It is obvious that the reference string and the current string have the same length. A SRB string also has two string-matching parameters: an index that specifies the address of the reference pixel in the SRB and a length that specifies the duplication count of the reference pixel to reconstruct the current string. The string-matching parameters are then entropy-coded and put into the final bit-stream. At the decoder side, the string-matching parameters are actually string-copying parameters. In the decoding process, the string-matching (i.e. string-copying) parameters are parsed, decoded, and used to obtain the reference string pixels from either PRB or SRB. The values of the reference string pixels are then assigned to the current pixels being decoded to reconstruct the current string.

In Section 2, a general architecture of string-matching enhanced coding system using both PRB and SRB is proposed and the details of horizontally and vertically scanned 2D-shape-preserved matching modes are described. Section 3 presents a flexible hash-table framework specially designed for speeding up reference string searching in PRB. Section 4 describes a method to seamlessly mix PRB search with SRB search inside a coding unit (CU), and to select one-by-one either an optimal PRB string or an optimal SRB string based on average rate-distortion (RD) cost evaluation. In section 5, the proposed technique is compared with HM-16.4+SCM-40 reference software [31] that is developed based on HEVC Software Model HM-16.4 by JCT for SCC testing. The experimental results show that the proposed technique can achieve significant bit-distortion rate (BD-rate) [32]–[33] reduction in lossy coding and bit-rate saving in lossless coding using the test suite defined by JCT Common Test Condition[30]. We conclude the paper and introduce the future work on string-matching coding technique in section 6. In this paper, the proposed SCC algorithm is described for pictures of YUV4:4:4 and RGB formats, but it can also be applied to pictures of YUV4:2:2 and YUV4:2:0 formats with some modifications. This paper is partially based on JCT-video coding (VC) documents [18]–[27].

2 String-Matching Enhanced Coding System

A general architecture and major components of string-matching enhanced coding system are shown in **Fig. 1**.

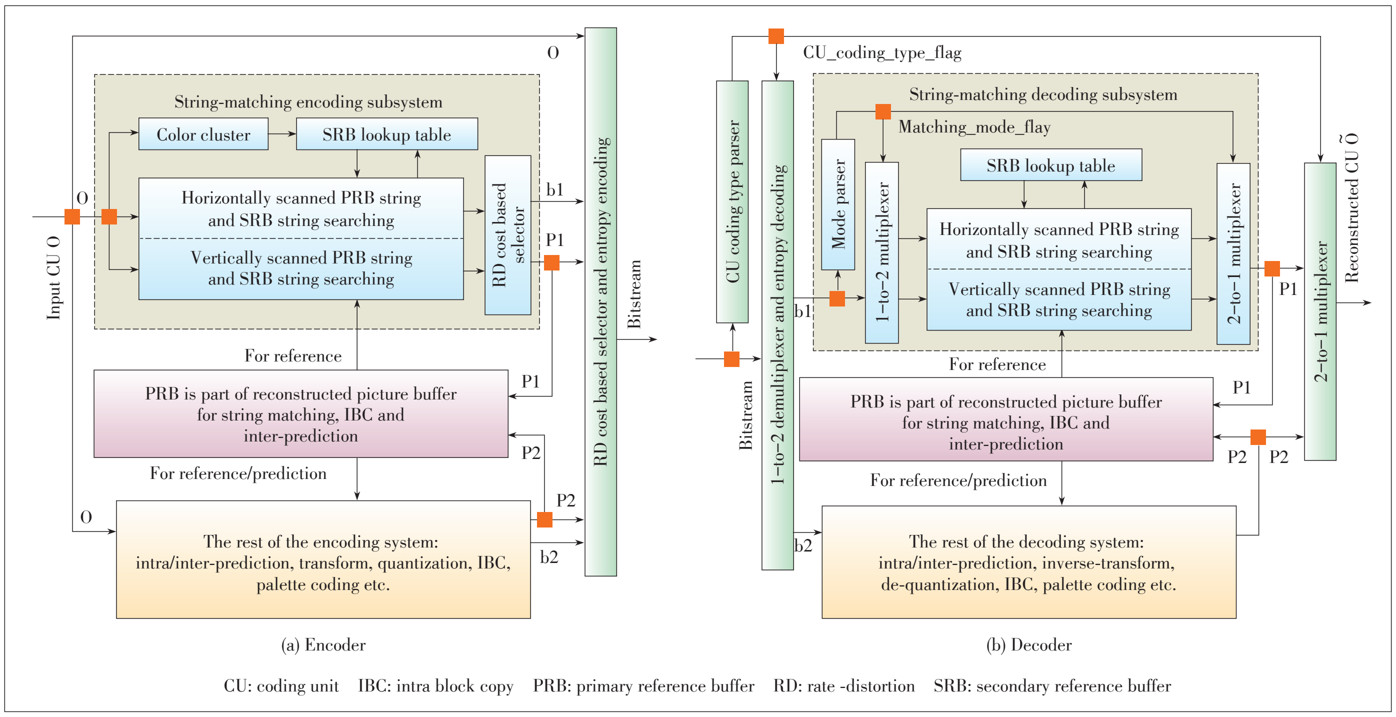
In the encoder of the string-matching enhanced coding system, the string-matching encoding subsystem performs color clustering, PRB string searching, and SRB string searching. The rest of the encoding system performs other traditional encoding operations such as intra-prediction, inter-prediction, transform, quantization, entropy encoding, IBC, and palette coding. The input CU O is fed to both string-matching encoding subsystem and the rest of the encoding system. The string-matching encoding subsystem codes O to generate coded bitstream b_1 and reconstructed CU P_1. The rest of the encoding system also codes O to generate coded bitstream b_2 and reconstructed CU P_2. O, P_1, b_1, P_2, and b_2 are sent to RD-cost-based selector that calculates the RD cost of two results and selects the one with the best RD performance as the final coding result for the CU. The corresponding coded bitstream b_1 or b_2 is selected and put into the output bitstream. Also, the corresponding reconstructed CU P_1 or P_2 is stored in reconstructed picture buffer, which is shared by both string-matching encoding subsystem as PRB for string-matching and the rest of the encoding system for inter prediction and IBC. The input CU O is also fed to a color cluster unit to obtain a few of most frequently occurred pixel colors. Some of the colors are put into the SRB LUT for SRB string searching.

In the decoder, the input bitstream is first parsed by CU coding type parser to get the CU_coding_type_flag that indicates whehter the current CU being decoded is coded by the proposed string-matching technique or by traditional coding techniques. If the CU is coded by the string-matching technique, the CU layer bitstream is sent to the string-matching decoding subsystem that decodes and reconstructs the CU P_1, which is stored in reconstructed picture buffer and is also the final reconstructed CU output $\tilde{O}$. If the CU is coded by traditional coding techniques, the CU layer bitstream is sent to the rest of the decoding system that performs traditional decoding tasks such as intra-prediction, inter-prediction, inverse-transform, de-quantization, IBC, and palette decoding. The CU P_2 is eventually reconstructed, which is stored in reconstructed picture buffer and is also the final reconstructed CU output $\tilde{O}$.

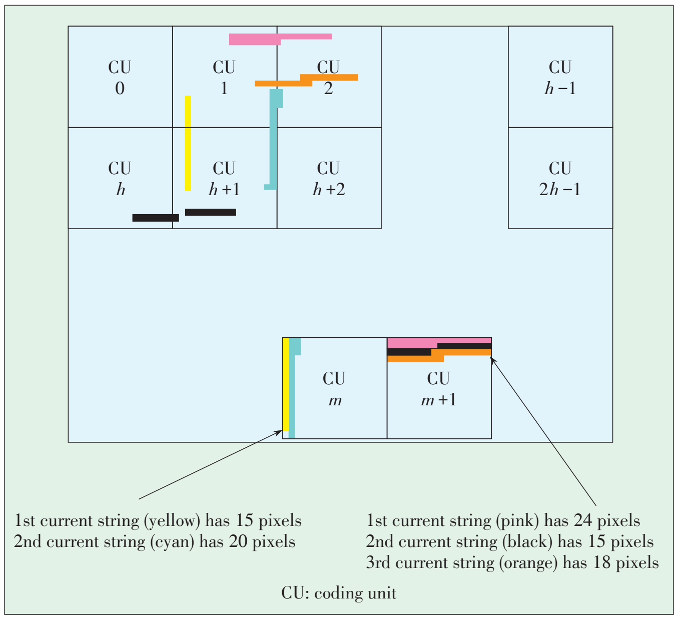
The string-matching coding subsystem has two CU level matching modes: horizontally scanned 2D-shape-preserved matching and vertically scanned 2D-shape-preserved matching (**Fig. 2**). A CU is pre-coded by both the modes. The mode minimizing RD cost is selected as the final string-matching mode for the CU. The CU size in Fig. 2 is 16x16 pixels in the packed pixel format (e.g. a Y sample is followed by a U sample and then a V sample, or a G sample is followed by a B sample and then an R sample).

The vertically scanned 2D-shape-preserved matching mode is used to code CU *m* in Fig. 2. In this mode, PRB is treated as a 2D plane while CU *m* is considered a vertically scanned 1D pixel string. Starting from the 1st pixel of CU *m*, the encoder searches the optimal (e.g. longest in lossless case) string in the PRB searching range that matches the pixel string in CU and also keeps exactly the same 2D shape as the pixel string in CU *m*. The string found in the searching range is called a reference string and the pixel string in CU *m* is called current string. Fig. 2 shows the first two current strings in CU *m* and their corresponding reference strings:

1) The 1st reference (and current) string in yellow has 15 pix-


Input CU O
O
String-matching encoding subsystem
Color cluster
SRB lookup table
Horizontally scanned PRB string and SRB string searching
Vertically scanned PRB string and SRB string searching
RD cost based selector
b1
P1
For reference
PRB is part of reconstructed picture buffer for string matching, IBC and inter-prediction
P1
P2
For reference/prediction
P2
The rest of the encoding system: intra/inter-prediction, transform, quantization, IBC, palette coding etc.
b2
RD cost based selector and entropy encoding
Bitstream
(a) Encoder
CU_coding_type_flag
CU coding type parser
1-to-2 demultiplexer and entropy decoding
Bitstream
String-matching decoding subsystem
Matching_mode_flay
Mode parser
1-to-2 multiplexer
SRB lookup table
Horizontally scanned PRB string and SRB string searching
Vertically scanned PRB string and SRB string searching
2-to-1 multiplexer
b1
P1
For reference
PRB is part of reconstructed picture buffer for string matching, IBC and inter-prediction
P1
P2
P2
For reference/prediction
b2
The rest of the decoding system: intra/inter-prediction, inverse-transform, de-quantization, IBC, palette coding etc.
2-to-1 multiplexer
Reconstructed CU Õ
(b) Decoder
CU: coding unit IBC: intra block copy PRB: primary reference buffer RD: rate -distortion SRB: secondary reference buffer

▲**Figure 1. String-matching enhanced coding system architecture.**


CU 0
CU 1
CU 2
CU h-1
CU h
CU h+1
CU h+2
CU 2h-1
CU m
CU m+1
1st current string (yellow) has 15 pixels
2nd current string (cyan) has 20 pixels
1st current string (pink) has 24 pixels
2nd current string (black) has 15 pixels
3rd current string (orange) has 18 pixels
CU: coding unit

▲**Figure 2. Two matching modes of string-matching coding.**

els and the 2D-shape-preserved reference string is located across CU 1 and CU h+1;

2) The 2nd reference (and current) string in cyan has 20 pixels and the 2D-shape-preserved reference string is located across CU 1, CU 2, and CU h+1.

The horizontally scanned 2D-shape-preserved matching mode is used to code CU m+1 in Fig. 2. In this mode, PRB is treated as a 2D plane while CU m+1 is considered to be a horizontally scanned 1D pixel string. Starting from the 1st pixel of CU m+1, the encoder searches the optimal string in the PRB searching range that matches the pixel string in CU m+1 and also keeps exactly the same 2D shape as the pixel string in CU m+1. The string found in the searching range is called a reference string and the pixel string in CU m+1 is called the current string. Fig. 2 shows the first three current strings in CU m+1 and their corresponding reference strings:

1) The 1st reference (and current) string in pink has 24 pixels and the 2D-shape-preserved reference string is located across CU 1 and CU 2;
2) The 2nd reference (and current) string in black has 15 pixels and the 2D-shape-preserved reference string is located across CU h and CU h+1;
3) The 3rd reference (and current) string in orange has 18 pixels and the 2D-shape-preserved reference string is located across CU 1 and CU 2.

There are usually two types of pixel scanning methods (and paths) available for a CU or a largest coding unit (LCU). One is raster-scan: a CU or LCU is scanned line by line, either horizontally or vertically, and all lines have the same scan direction (Fig. 2). The other is traverse-scan: a CU or LCU is also scanned line by line, either horizontally or vertically, but odd lines and even lines have opposite scan direction. Once a scanning method (and path) is determined, all pixels inside a CU or LCU are lined up following the scanning path. Thus, starting from a current pixel being coded inside a CU $P_{m,n}$, a current string cur$S_{m,n}$ can be defined following the scanning path. The pixels of cur$S_{m,n}$ are designated as $S_{m,n}(0)$, $S_{m,n}(1)$, ... , $S_{m,n}(L-1)$, following the order of the pixels on the scanning path, where L is the length of the string in unit of pixel. Using the designation, cur$S_{m,n}$ and its pixels can be expressed as cur$S_{m,n}$ = $\{S_{m,n}(k)$: $0 \leq k < L\}$. Given a current string cur$S_{m,n}$ and a reference pixel $P_{i,j}$, a reference string ref$S_{i,j}$ = $\{S_{i,j}(k)$: $0 \leq k < L\}$ is the string that starts from $P_{i,j}$, i.e. $S_{i,j}(0) = P_{i,j}$, and has exactly the same 2D shape and scanning order as cur$S_{m,n}$.

Fig. 3 illustrates an example of the current string cur$S_{m,n}$ = $\{S_{m,n}(k)$: $0 \leq k < L\}$ starting from the current pixel $P_{m,n}$ inside a 8×8 CU of horizontal traverse-scan and its reference string ref$S_{i,j}$ = $\{S_{i,j}(k)$: $0 \leq k < L\}$ starting from the reference pixel $P_{i,j}$. Both cur$S_{m,n}$ and ref$S_{i,j}$ have identical 2D shape and length of $L = 9$.

Obviously, given a current pixel $P_{m,n}$, the current string cur$S_{m,n}$ of length L can be uniquely specified within a CU with known size and scanning method, and a reference string ref$S_{i,j}$ is uniquely specified by a displacement vector ($\boldsymbol{DV}_h$, $\boldsymbol{DV}_v$) = (i-m, j-n). In Fig. 3, $\boldsymbol{DV}_h = i - m$ is the horizontal displacement between cur$S_{m,n}$ and ref$S_{i,j}$, and $\boldsymbol{DV}_v = j - n$ is the vertical displacement between cur$S_{m,n}$ and ref$S_{i,j}$.

A matching reference string ref$S_{i,j}$ = $\{S_{i,j}(k)$: $0 \leq k < L\}$ for a given current string cur$S_{m,n}$ = $\{S_{m,n}(k)$: $0 \leq k < L\}$ satisfies the constraint that the difference between $S_{i,j}(k)$ and $S_{m,n}(k)$ is within a predetermined threshold for all k. One common option used for difference measurement is the absolute difference $|S_{i,j}(k)_Y - S_{m,n}(k)_Y|$, $|S_{i,j}(k)_U - S_{m,n}(k)_U|$, and $|S_{i,j}(k)_V - S_{m,n}(k)_V|$, where the subscripts Y, U, V designate the Y color component, U color component, and V color component for the corresponding pixels $S_{i,j}(k)$ or $S_{m,n}(k)$.

3 String-Matching Oriented Hash-Table Framework

String-matching coding performance heavily depends on the PRB searching range. The larger the searching range is, the better the coding performance can achieve. However, the longer the searching time is, the more the computing power requires. Hash-table can be used to speed up reference string searching. Therefore, the key to design a practical string-matching encoding subsystem is to have a single and efficient string searching oriented hash-table that should work and speed up the searching in all two matching modes.

In the string-matching oriented hash-table framework, the basic role of a hash-table is to quickly find the first matching reference pixel in the PRB searching range by table-look-up. First of all, we need to define a pixel-order for all pixels in the PRB searching range. Naturally, we use the order defined in the horizontally scanned string matching mode, that is, all pixels are ordered one LCU followed by another LCU in LCU coding order. Within an LCU, horizontal scanning is employed to order pixels. All pixels with the same hash-value are chained together, following the pixel-order. The hash-value of any current pixel being coded is calculated and the encoder only needs to search through the hash chain of the same hash-value, instead of all pixels of the entire PRB searching range, to find a potential matching reference pixel. This potential pixel is the first pixel of a potential reference string. In this way, the searching time can be significantly reduced.

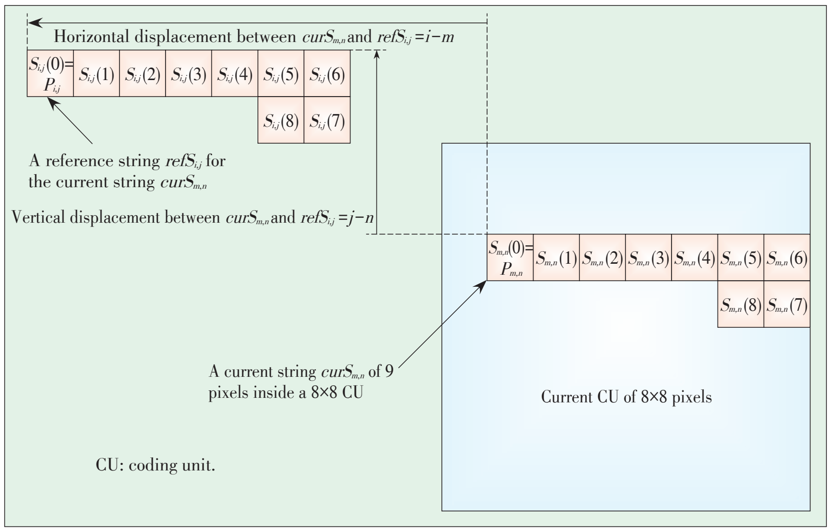
In **Fig. 4**, the pixels (in the searching range) lined up in pixel-order are $P_{0,0}$, $P_{1,0}$, $P_{2,0}$, $P_{3,0}$, ... , $P_{i,j}$, $P_{i+1,j}$, $P_{i+2,j}$, $P_{i+3,j}$, ... , $P_{i,j+1}$, ... , followed by the current pixel being coded, $P_{m,n}$. The hash values ha_val$_{0,0}$, ha_val$_{1,0}$, ha_val$_{2,0}$, ha_val$_{3,0}$, ... , ha_val$_{i,j}$, ha_val$_{i+1,j}$, ha_val$_{i+2,j}$, ha_val$_{i+3,j}$, ... , ha_val$_{i,j+1}$ are computed from the corresponding pixels. There are three hash chain examples in this figure.

1) Example 1

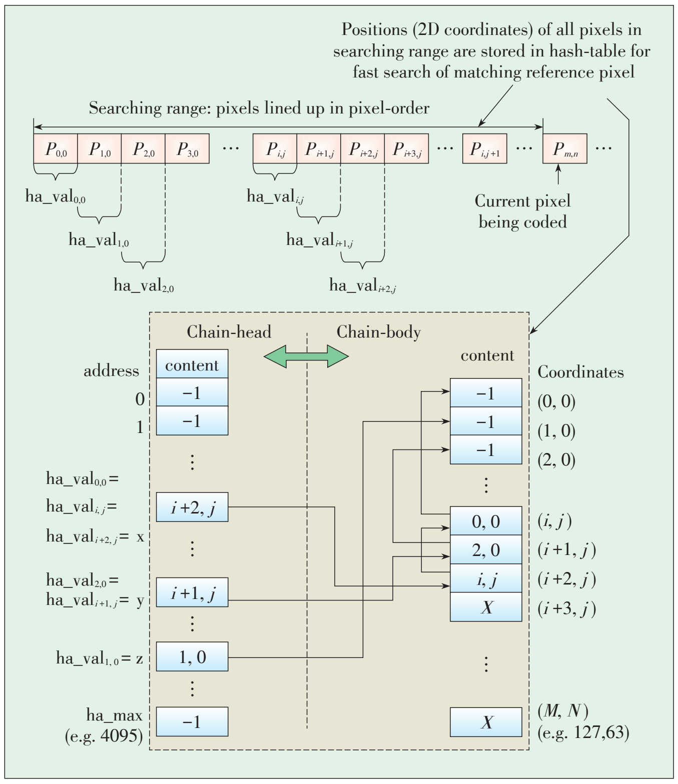
Three pixels $P_{0,0}$, $P_{i,j}$, and $P_{i+2,j}$ have the same hash value, i.e. ha_val$_{0,0}$ = ha_val$_{i,j}$ = ha_val$_{i+2,j}$ = x. Coordinates (0, 0), (i, j), and (i+2, j) of the three pixels are stored in the hash-table and form a hash chain of hash-value x. The hash-table actually has two parts: chain-head and chain-body. Among the three coordinates, the coordinates (i+2, j) having the largest pixel-order is stored in the chain-head slot of address x. The coordinates (i, j) having the second largest pixel-order is stored in chain-body location of coordinates (i+2, j). The coordinates (0, 0) having the third largest pixel-order is stored in the chain-body location of coordinates (i, j). Finally, -1 is stored in the chain-body location of coordinates (0, 0) to indicate the end of the chain.

2) Example 2

Two pixels $P_{2,0}$ and $P_{i+1,j}$ have the same hash value, i.e. ha_val$_{2,0}$ = ha_val$_{i+1,j}$ = y. Coordinates (2, 0) and (i+1, j) form a hash chain of hash-value y in the hash-table. In the hash chain, the coordinates (i+1, j) having the largest pixel-order is stored in the chain-head slot of address y. The coordinates (2, 0) having the second largest pixel-order is stored in the chain-body location of coordinates (i+1, j). Finally, -1 is


Horizontal displacement between curS_{m,n} and refS_{i,j} = i−m
S_{i,j}(0)= P_{i,j}
S_{i,j}(1)
S_{i,j}(2)
S_{i,j}(3)
S_{i,j}(4)
S_{i,j}(5)
S_{i,j}(6)
S_{i,j}(8)
S_{i,j}(7)
A reference string refS_{i,j} for the current string curS_{m,n}
Vertical displacement between curS_{m,n} and refS_{i,j} = j−n
S_{m,n}(0)= P_{m,n}
S_{m,n}(1)
S_{m,n}(2)
S_{m,n}(3)
S_{m,n}(4)
S_{m,n}(5)
S_{m,n}(6)
S_{m,n}(8)
S_{m,n}(7)
A current string curS_{m,n} of 9 pixels inside a 8×8 CU
Current CU of 8×8 pixels
CU: coding unit.

▲Figure 3. An example of current string and its reference string.


Positions (2D coordinates) of all pixels in searching range are stored in hash-table for fast search of matching reference pixel
Searching range: pixels lined up in pixel-order
Current pixel being coded
Chain-head
Chain-body
address
content
Coordinates
ha_max (e.g. 4095)
(M, N) (e.g. 127,63)

▲Figure 4. Hash -table structure and contents.

stored in the chain-body location of coordinates (2, 0) to indicate the end of the chain.

3) Example 3

The third hash chain has only one pixel $P_{1,0}$ whose hash value is ha_val$_{1,0}$ = z. Therefore, the coordinates (1, 0) is stored in the chain-head slot of address z and -1 is stored in the chain-body location of coordinates (1, 0) to indicate the end of the chain.

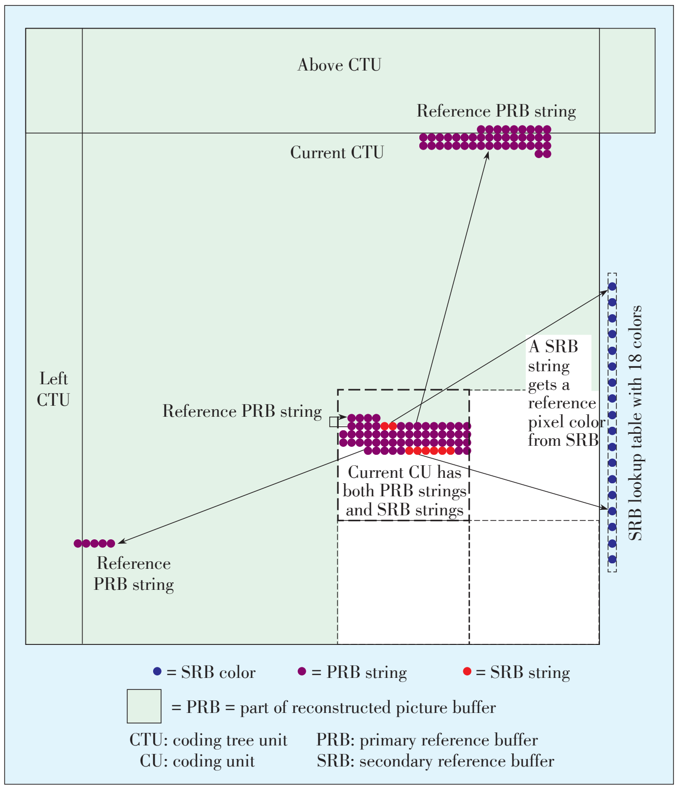
When the current pixel $P_{m,n}$ is encoded and the current string starts from $P_{m,n}$, the hash value ha_val$_{m,n}$ of $P_{m,n}$ is first computed and then used as the chain-head address to find the 1st coordinates in the hash chain of the same hash value. In the hash chain examples in 0, if ha_val$_{m,n}$ = x, the 1st coordinates is (i+2, j); If ha_val$_{m,n}$ = y, the 1st coordinates is (i+1, j); If ha_val$_{m,n}$ = z, the 1st coordinates is (1, 0). The content of the 1st coordinates in the chain-body is the 2nd coordinates in the hash chain of the same hash value, and the content of the 2nd coordinates in the chain-body is the 3rd coordinates in the hash chain of the same hash value, and so on. Therefore, the string-matching encoder can use the hash value of the current pixel being coded to find all pixels and their locations that have the same hash value in the searching range. Moreover, a hash chain starts from a hash-head slot, the 1st coordinates have the largest pixel-order, the 2nd coordinates have the second largest pixel-order, and so on.

For a pixel P = (Y, U, V) or (G, B, R), where Y, U, V (or G, B, R) are three 8-bit color components of P, a 12-bit hash value ha_val is computed. For lossy coding, ha_val is computed by concatenating 4-bit most significant bit (MSB) of Y, U, V (or G, B, R) to form a 12-bit value. For lossless coding, ha_val is computed by cyclic redundancy check (CRC)-12 algorithm [34] to get a 12-bit value. Obviously, all the pixels are always located in the same hash chain, no matter they have the identical 4-bit MSB component value in a lossy coding case or the identical 8-bit component value in a lossless coding case. Therefore, a hash chain with the same hash value provides a small and efficient candidate set of reference pixels for the starting pixel of a potential reference string.

4 PRB and SRB Based String-Matching

In the proposed string-matching technique, a current string being coded gets reference pixels from either PRB or SRB. When a current string gets the reference pixels from PRB, the reference pixels form a reference string that has exactly the same 2D shape and length (number of pixels) as the current string. The reference string can be in any valid location inside PRB. When a current string gets the reference pixels from SRB, it actually gets only one single reference pixel color from SRB LUT and all pixels of the entire string has the same color.

A CU coded by the string-matching technique can have both PRB and SRB strings (**Fig. 5**). In the figure, the 1st string in purple dots is a 4-pixel PRB string. Its reference string is just above it and has the same horizontal line shape and length of 4


Above CTU
Reference PRB string
Current CTU
Left CTU
Reference PRB string
A SRB string gets a reference pixel color from SRB
Current CU has both PRB strings and SRB strings
Reference PRB string
SRB lookup table with 18 colors
● = SRB color
● = PRB string
● = SRB string
= PRB = part of reconstructed picture buffer
CTU: coding tree unit
CU: coding unit
PRB: primary reference buffer
SRB: secondary reference buffer

▲Figure 5. A CU coded by string -matching technique.

pixels. The 2nd string in red dots is a 2-pixel SRB string. Its reference pixels are the 1st SRB LUT pixel color duplicated twice. The 3rd string in purple dots is a 43-pixel PRB string. Its reference string is located across the boundary between current coding tree unit (CTU) and above CTU. The PRB string and its reference string have exactly the same horizontal traverse-scan shape and length of 43 pixels. The 4th string in red dots is a 6-pixel SRB string. Its reference pixels are the 15th SRB LUT pixel color duplicated six times. The 5th string in purple dots is a 5-pixel PRB string. Its reference string is located across the boundary between current CTU and left CTU. The PRB string and its reference string have exactly the same horizontal line shape and length of 5 pixels.

In a string-matching encoder, the best matching reference string for a current starting pixel $P_{m,n}$ being coded is found via the following steps:

1) Searching the best reference PRB string for a current string cur$S_{m,n}$ via three sub-steps:

a) Calculating the hash value ha_val$_{m,n}$ of $P_{m,n}$.

b) For the pixel coordinates (i, j) obtained from the hash chain with the same hash value ha_val$_{m,n}$, determining the longest matching reference string ref$S_{i,j}$ = $\{S_{i,j}(k)$: $0\leqslant k<L_{i,j}\}$, i.e. determining the largest $L_{i,j}$ that satisfies the constraint $|S_{i,j}(k)_Y - S_{m,n}(k)_Y| \leqslant E$, $|S_{i,j}(k)_U - S_{m,n}(k)_U| \leqslant E$, $|S_{i,j}(k)_V - S_{m,n}(k)_V| \leqslant E$ for all $k < L_{i,j}$, where E is a predetermined threshold. After going through all the pixel coordinates (i, j) on the hash chain of hash value ha_val$_{m,n}$, multiple matching reference strings are found as best reference PRB string candidates.

c) The best reference PRB string is selected from the best reference PRB string candidates, based on average RD cost evaluation. For a given current string cur$S_{m,n}$ and its reference string ref$S_{i,j}$ of length L, the average RD cost is calculated by (1)

$$\text{avgRDcost}(\text{cur}S_{m,n}, \text{ref}S_{i,j}) = [\lambda \text{bits}(\text{ref}S_{i,j}) + \text{distortion}(\text{cur}S_{m,n}, \text{ref}S_{i,j})]/L \quad (1)$$

where bits(ref$S_{i,j}$) is the number of bits for coding the reference string ref$S_{i,j}$, λ is a weighting factor, and distortion(cur$S_{m,n}$, ref$S_{i,j}$) is the distortion between the current string cur$S_{m,n}$ and the reference string ref$S_{i,j}$.

2) Searching the best reference SRB string, which is straightforward because a reference SRB string is a simple duplication of an SRB pixel color.

3) Either the best reference PRB string or the best reference SRB string is selected as the best matching reference string based on average RD cost evaluation.

If no PRB string or SRB string can be found for the current pixel $P_{m,n}$ being coded, the pixel itself is coded directly as an unmatched pixel.

5 Experimental Results

As an implementation example, the proposed string-matching technique is integrated into HM-16.4+SCM-4.0 reference software [31]. All experimental results were generated under the common test conditions and configurations defined in [30] for HEVC SCC project.

Thirteen test sequences (**Table 1**) are used in the experiment. The test sequences are classified into four categories: text and graphics with motion (TGM), mixed content (MC), camera captured (CC), and animation (ANI). Each test sequence has a RGB color format version and a YCbCr (YUV) color format version. Therefore, there are 26 sequences in total used in the experiment.

To evaluate the overall coding performance, the HEVC BD-rate metric [32], [33] is used for lossy coding and bit-rate saving metric is used for lossless coding. In lossy coding, an average BD-rate reduction is calculated by three color components G/Y, B/U, and R/V for each color format and category. In lossless coding, four bit-rate saving values (total, average, minimum, and maximum) are calculated for each color format and category. Both lossy coding and lossless coding experiments used three configurations: all intra (AI), random access (RA), and low delay B (LB). Encoding and decoding software runtime were also compared for evaluating the complexity of the encoder and decoder.

Two coding methods were compared:

1) HM-16.4+SCM-4.0 (SCM) with default setting. In particular, the IBC search range is full frame.

2) HM-16.4+SCM-4.0 integrated with string-matching technique (SCM-SM). The string-matching search range is the current LCU and left LCU.

Table 2 shows coding performance comparison (BD-rate reduction percentage in negative numbers) between SCM and SCM-SM in the lossy coding case. **Table 3** shows coding performance comparison (bit-rate saving percentage in negative numbers) between SCM and SCM-SM in the lossless coding case. Each row of data in the two tables corresponds to a com-

Table 1. Four-category test sequences

Resolution	Sequence name	Category	fps	Frames to be encoded
1920×1080	sc_flyingGraphics_1920x1080_60_8bit	TGM	60	0-299*
	sc_desktop_1920x1080_60_8bit	TGM	60	0-599
	sc_console_1920x1080_60_8bit	TGM	60	0-599
	MissionControlClip3_1920x1080_60p_8b444	MC	60	0-599
	EBURainFruits_1920x1080_50_10bit	CC	50	0-249**
	Kimono1_1920x1080_24_10bit	CC	24	0-119***
1280×720	sc_web_browsing_1280x720_30_8bit	TGM	30	0-299
	sc_map_1280x720_60_8bit	TGM	60	0-599
	sc_programming_1280x720_60_8bit	TGM	60	0-599
	sc_SlideShow_1280x720_20_8bit	TGM	20	0-499
	sc_robot_1280x720_30_8bit	ANI	30	0-299
2560×1440	Basketball_Screen_2560x1440_60p_8b444	MC	60	322-621
	MissionControlClip2_2560x1440_60p_8444	MC	60	120-419

*Only the first 300 frames of this sequence are used.
**Only the first 250 frames of this 10 -bit sequence are used, and InternalBitDepth is set to 8
***Only the first 120 frames of this 10 -bit sequence are used, and InternalBitDepth is set to 8

ANI: animation
CC: camera -captured content
MC: mixed content
TGM: Text and graphics with motion

Table 2. Performance comparison between SCM and SCM-SM in the lossy coding case

Anchor: SCM	All intra			Random access			Low delay B		
Tested: SCM-SM	G/Y	B/U	R/V	G/Y	B/U	R/V	G/Y	B/U	R/V
RGB, text & graphics with motion, 1080p & 720p	−3.77%	−4.23%	−4.19%	−2.48%	−2.65%	−2.68%	−1.79%	−1.92%	−1.95%
RGB, mixed content, 1440p & 1080p	−1.15%	−1.68%	−1.70%	−0.62%	−1.01%	−1.03%	−0.15%	−0.95%	−0.77%
RGB, Animation, 720p	0.02%	−0.03%	−0.02%	0.07%	0.02%	0.08%	0.00%	0.05%	0.10%
RGB, camera captured, 1080p	0.03%	0.02%	0.03%	0.11%	0.07%	0.14%	0.06%	0.01%	0.04%
YUV, text & graphics with motion, 1080p & 720p	−4.19%	−4.31%	−4.33%	−2.38%	−2.38%	−2.64%	−1.72%	−1.95%	−1.85%
YUV, mixed content, 1440p & 1080p	−1.51%	−2.52%	−2.79%	−0.85%	−1.95%	−2.36%	−0.31%	−1.74%	−2.04%
YUV, Animation, 720p	0.01%	−0.02%	−0.03%	0.02%	−0.18%	0.17%	0.20%	0.18%	0.31%
YUV, camera captured, 1080p	0.04%	0.03%	0.04%	0.10%	−0.07%	0.04%	0.02%	0.03%	0.09%
Encoding time (%)	121.69%			111.08%			106.73%		
Decoding time (%)	100.04%			106.50%			107.87%		

SCM: HM-16.4+SCM-4.0 SCM-SM: HM-16.4+SCM-4.0 integrated with string-matching technique

Table 3. Performance comparison between SCM and SCM-SM in lossless coding case

Anchor:SCM	All intra				Random access				Low delay B			
Tested: SCM-SM	Bit-rate change (Total)	Bit-rate change (Avg)	Bit-rate change (Max)	Bit-rate change (Min)	Bit-rate change (Total)	Bit-rate change (Avg)	Bit-rate change (Max)	Bit-rate change (Min)	Bit-rate change (Total)	Bit-rate change (Avg)	Bit-rate change (Max)	Bit-rate change (Min)
RGB, TGM	−3.87%	−4.64%	−14.0%	−0.47%	−2.19%	−2.99%	−9.56%	−0.26%	−2.06%	−2.28%	−6.12%	−0.20%
RGB, MC	−0.63%	−0.69%	−1.19%	−0.16%	−0.11%	−0.10%	−0.11%	−0.09%	−0.06%	−0.06%	−0.08%	−0.04%
RGB, ANI	0.00%	0.00%	0.00%	0.00%	0.01%	0.01%	0.01%	0.01%	0.00%	0.00%	0.00%	0.00%
RGB, CC	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
YUV, TGM	−4.39%	−5.17%	−14.5%	−0.31%	−2.39%	−3.47%	−9.90%	−0.16%	−2.23%	−2.76%	−6.46%	−0.13%
YUV, MC	−0.73%	−0.78%	−1.31%	−0.15%	−0.12%	−0.11%	−0.14%	−0.08%	−0.05%	−0.06%	−0.07%	−0.04%
YUV, ANI	0.00%	0.00%	0.00%	0.00%	0.01%	0.01%	0.01%	0.01%	0.01%	0.01%	0.01%	0.01%
YUV, CC	0.00%	0.00%	0.00%	0.01%	0.00%	0.00%	0.00%	0.01%	0.00%	0.00%	0.00%	0.01%
Encoding time (%)	152.10%				111.39%				107.15%			
Decoding time (%)	100.67%				106.44%				105.89%			

SCM: HM -16.4+SCM -4.0 SCM -SM: HM -16.4+SCM -4.0 integrated with string -matching technique

bination of one color format and one category. There are totally eight combinations. Each combination contains 1—7 sequences. The encoding and decoding runtime ratios are also shown in the tables.

The experimental results include:

1) In the lossy coding case, SCM-SM can achieve up to 4.33% for AI, 2.68% for RA, 2.04% for LB average BD-rate reduction over SCM.
2) In the lossless coding case, SCM - SM can achieve up to 14.5% for AI, 9.9% for RA, 6.46% for LB maximum bit-rate saving and 5.17% for AI, 3.47% for RA, 2.76% for LB average bit-rate saving over SCM.
3) The improvement of SCM-SM over SCM depends on the contents. In YUV TGM case, the average bit - rate saving is 5.17% in lossless coding case and the average BD-rate reduction of components Y, U and V are 4.19%, 4.31% and 4.33%, respectively in lossy coding case for AI configuration. SCM-SM also has good BD-rate reduction over SCM for mixed content in all configurations, but no coding performance improvement for camera captured and animation contents.
4) In the lossy coding case, encoding runtime increase is about 22% for AI, 11% for RA, 7% for LB. In lossless coding case, encoding runtime increase is about 52% for AI, 11% for RA, 7% for LB

6 Conclusions

This paper proposes a string-matching coding technique for SCC. Both PRB and SRB are used for string-matching. The resulting bitstream is a combination of PRB string coding parameters, SRB string coding parameters, and unmatched pixel mixed on a string-by-string basis. The experiments using common test condition [30] show that the string-matching coding technique can achieve a lossy coding BD-rate reduction of up to 4.33%, 2.68%, 2.04% for AI, RA, and LB configurations respectively, and a lossless coding average bit-rate saving of up to 5.17%, 3.47%, 2.76% for AI, RA, and LB configurations respectively.

Our future work includes: 1) implementing rate-control in string-matching enhanced coding system; 2) optimizing string-matching coding to improve coding performance and reduce coding complexity further; 3) achieving idempotent (re-loss-free) [35] coding in repetitive (multi-generation) compression using the string-matching enhanced coding system.

References

[1] T. Lin, K. Zhou, and S. Wang. "Cloudlet-screen computing: a client-server architecture with top graphics performance," *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 13, no. 2, pp. 96–108, Jun. 2013. doi: 10.1504/IJAHUC.2013.054174.

[2] Y. Lu, S. Li, and H. Shen, "Virtualized screen: a third element for cloud_mobile convergence," *IEEE Multimedia*, vol. 18, no. 2, pp. 4–11, Apr. 2011. doi: 10.1109/MMUL.2011.33.

[3] T. Lin and S. Wang. "Cloudlet-screen computing: a multi-core-based, cloud-computing-oriented, traditional-computing-compatible parallel computing paradigm for the masses," in *IEEE International Conference on Multimedia and Expo*, New York, USA, Jul. 2009, pp. 1805–1808. doi: 10.1109/ICME.2009.5202873.

[4] T. Lin, K. Zhou, X. Chen, and S. Wang, "Arbitrary shape matching for screen content coding," *Picture Coding Symposium*, San Jose, USA, Dec. 2013, pp. 369–372. doi: 10.1109/PCS.2013.6737760.

[5] W. Zhu, J. Xu, W. Ding, *et al.*, "Adaptive LZMA-based coding for screen content," *Picture Coding Symposium*, San Jose, USA, Dec. 2013, pp. 373–376. doi: 10.1109/PCS.2013.6737761.

[6] T. Lin, X. Chen, and S. Wang, "Pseudo-2-D-matching based dual-coder architecture for screen contents coding," *IEEE International Conference on Multimedia and Expo Workshops*, San Jose, USA, Jul. 2013, pp. 1–4. doi: 10.1109/ICMEW.2013.6618315.

[7] S. Wang and T. Lin, "Compound image compression based on unified LZ and hybrid coding," *IET Image Processing*, vol. 7, no. 5, pp. 484–499, May 2013. doi: 10.1049/iet-ipr.2012.0439.

[8] T. Lin, P. Zhang, S. Wang, *et al.*, "Mixed chroma sampling-rate high efficiency video coding for full-chroma screen content," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 1, pp. 173–185, Jan. 2013. doi: 10.1109/TCSVT.2012.2223871.

[9] W. Zhu, W. Ding, R. Xiong, *et al.*, "Compound image compression by multi-stage prediction," *IEEE Visual Communications and Image Processing Conference*, San Diego, USA, Nov. 2012, pp. 1–6. doi: 10.1109/VCIP.2012.6410758.

[10] S. Wang and T. Lin, "United coding method for compound image compression," *Multimedia Tools and Applications*, vol. 71, no. 3, pp. 1263–1282, Aug. 2014. doi: 10.1007/s11042-012-1274-y.

[11] S. Wang and T. Lin, "United coding for compound image compression," *3rd International Congress on Image and Signal Processing*, Yantai, China, pp. 566–570, Oct. 2010. doi: 10.1109/CISP.2010.5647270.

[12] S. Wang and T. Lin, "A unified LZ and hybrid coding for compound image partial-lossless compression," *2nd International Conference on Image and Signal Processing*, Tianjin, China, Oct. 2009, pp. 1–5. doi: 10.1109/CISP.2009.5301019.

[13] W. Ding, Y. Lu, and F. Wu, "Enable efficient compound image compression in h.264/AVC intra coding," *IEEE International Conference on Image Processing*, San Antonio, USA, Oct. 2007, vol. 2, pp. 337–340. doi: 10.1109/ICIP.2007.4379161.

[14] ISO/IEC JTC1/SC29/WG11 and ITU-T Q6/16, "Joint call for proposals for coding of screen content," ISO/IEC JTC 1/SC 29/WG 11 (MPEG), San Jose, USA, Doc. N14175, Jan. 2014.

[15] D.-K. Kwon and M. Budagavi, "Intra motion compensation with variable length intra MV coding," JCT-VC, Vienna, Austria, JCTVC-N0206, Jul. 2013.

[16] C. Pang, J. Sole, L. Guo, *et al.*, "Intra motion compensation with 2-D MVs," JCT-VC, Vienna, Austria, JCTVC-N0256, Jul. 2013.

[17] C. Pang, J. Sole, L. Guo, *et al.*, "Displacement vector signaling for intra block copying," JCT-VC, Geneva, Switzerland, JCTVC-O0154, Oct. 2013.

[18] T. Lin, S. Wang, P. Zhang, and K. Zhou, "AHG7: full-chroma (YUV444) dictionary+hybrid dual-coder extension of HEVC," JCT-VC, Shanghai, China, JCTVC-K0133, Oct. 2012.

[19] M. Budagavi, "Cross-check of JCTVC-K0133 (dictionary+hybrid dual-coder extension of HEVC)," JCT-VC, Shanghai, China, JCTVC-K0329, Oct. 2012.

[20] T. Lin, S. Wang, P. Zhang, and K. Zhou, "AHG8: P2M based dual-coder extension of HEVC," JCT-VC, Geneva, Switzerland, JCTVC-L0303, Jan. 2013.

[21] J. Ye, S. Liu, S. Lei, *et al.*, "Improvements on 1D dictionary coding," JCT-VC, Valencia, Spain, JCTVC-Q0124, 2014.

[22] R. Cohen, "Crosscheck for JCTVC-Q0124 improvements on 1D dictionary coding mode," JCT-VC, Valencia, Spain, JCTVC-Q0125, 2014.

[23] L. Zhao, X. Chen, T. Lin, *et al.*, "SCCE4: Results of subtest 3.3," JCT-VC, Sapporo, Japan, JCTVC-R0060, 2014.

[24] B. Li and J. Xu, "Cross-check of test 3.3 (JCTVC-R0060)," JCT-VC, Sapporo, Japan, JCTVC-R0109, 2014.

[25] K. Zhou, L. Zhao, X. Chen, and T. Lin, "Non-CE10: improvement on coding of ISC parameters and comparison to Palette coding," JCT-VC, Strasbourg, France, JCTVC-S0250, Oct. 2014.

[26] L. Zhao, K. Zhou, and T. Lin, "CE3: results of test B.4.2 (minimum string length of 20 pixels) on intra string copy," JCT-VC, Geneva, Switzerland, JCTVC-T0136, Feb. 2015.

[27] C.-H. Hung, Y.-J. Chang, J.-S. Tu, *et al.*, "CE3: crosscheck of CE3 test B.4.2 (JCTVC-T0136)," JCT-VC, Geneva, Switzerland, JCTVC-T0179, Feb. 2015.

[28] L. Zhao, K. Zhou, S. Wang, and T. Lin, "Non-CE3: improvement on intra string copy," JCT-VC, Geneva, Switzerland, JCTVC-T0139, Feb. 2015.

[29] R.-L. Liao, C.-C. Chen, W.-H. Peng, and H.-M. Hang, "Crosscheck of non-CE3: improvement on intra string copy (JCTVC-T0139)," JCT-VC, Geneva, Switzerland, JCTVC-T0200, Feb. 2015.

[30] H. Yu, R. Cohen, K. Rapaka, and J. Xu, "Common conditions for screen content coding tests," JCT-VC, Strasbourg, France, JCTVC-S1015, Oct. 2014.

[31] Heinrich Hertz Institute. (2015). *Rec. ITU-T H.265 | ISO/IEC 23008-2 High Efficiency Video Coding* [Online]. Available: https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-16.4+SCM-4.0

[32] G. Bjøntegaard, "Calculation of average PSNR differences between RD-Curves," ITU-T SG16 Q.6 Document, VCEG-M33, Austin, USA, Apr. 2001.

[33] G. Bjøntegaard, "Improvements of the BD-PSNR model," ITU-T SG16 Q.6 Document, VCEG-AI11, Berlin, Germany, Jul. 2008.

[34] T. V. Ramabadran and S. S. Gaitonde, "A tutorial on CRC computations," *IEEE Micro*, vol. 8, no. 4, pp. 62–75, Aug. 1988. doi: 10.1109/40.7773.

[35] T. Lin, "Achieving re-loss-free video coding," *IEEE Signal Processing Letters*, vol. 16, no. 4, pp. 323–326, Apr. 2009. doi: 10.1109/LSP.2009.2014285.

Manuscript received: 2015-05-13

Biographies

Tao Lin (lintao@tongji.edu.cn) received his MS and PhD degrees from Tohoku University, Japan, in 1985 and 1989. He has been with VLSI Lab, Tongji University, China since 2003. In 2005, he was awarded "Chang Jiang Scholars", the highest honor given by China Ministry of Education. From 1988 to 2002, he was a postdoctoral researcher with University of California, Berkeley, and developed multimedia ICs and products at several companies in Silicon Valley, including Integrated Device Technology, Inc., PMC-Sierra Inc., Cypress Semiconductor Corp., and NeoMagic Corp. He has been granted 24 US patents and 14 China patents. His current research interests include cloud-mobile computing, digital signal processing, audiovisual coding, and multimedia SoC design.

Kailun Zhou (vlsi@tongji.edu.cn) received his MS degree from Shanghai Jiaotong University, China in 2003. He is currently pursuing the PhD degree with Tongji University, China. His current research interests include embedded system design, video coding, and ASIC architecture, design and verification.

Liping Zhao (vlsi@tongji.edu.cn) received her MS degree in computer science and technology from Hunan University, China in 2009. She is currently a PhD candidate in control science and engineering at VLSI lab of Tongji University, China. Her current research interests include screen content coding and video coding.