Multimedia Tools and Applications (2022) 81:2043–2063
https://doi.org/10.1007/s11042-021-11418-6

A string matching based ultra-low complexity lossless screen content coding technique

Yufen Yang[1] · **Tao Lin**[1] · **Liping Zhao**[2,3] · **Kailun Zhou**[1] · **Shuhui Wang**[1]

Received: 9 September 2020 / Revised: 3 August 2021 / Accepted: 10 August 2021 /
Published online: 19 October 2021

Abstract
Screen contents have become a popular image type driven by the growing market for transferring display screen between devices, especially mobile devices. Due to the ultra-high quality display featured in most of nowadays mobile devices, lossless screen content coding (SCC) is usually required or preferred. Mobile devices also require ultra-low power consumption in all tasks including SCC. To address these issues, this paper proposes an ultra-low coding complexity technique based on string matching for high efficiency lossless SCC. The technique covers three major coding phases of fast searching, prediction, and entropy coding. Condensed hash table (CHT) based fast searching is proposed to speed-up reference string searching process. Coplanar prediction (CP) and predictor-dependent residual (PDR) are presented to first efficiently predict an unmatchable pixel using multiple neighboring pixels and then further reduce the entropy of prediction residuals. To achieve a good trade-off between coding complexity and efficiency, 4-bit-aligned variable length code (4bVLC) and byte-aligned multi-variable-length-code (BMVLC) are proposed to code the prediction residuals and three string matching parameters, respectively. For 184 screen content images commonly used, compared with X265 and PNG in the default configuration and lossless mode, the proposed technique achieves 35.67% less total compressed bytes with only 0.96% encoding and 1.54% decoding runtime, and 10.04% less total compressed bytes with only 6.83% encoding and 24.32% decoding runtime, respectively. The proposed technique also outperforms X265 and PNG in all other configurations. For twelve HEVC-SCC CTC images, compared with PNG in fast, default and slow configurations and X265 in ultrafast and default configurations, the proposed technique shows significant advantage with both high coding efficiency and ultra-low coding complexity.

Keywords Screen content coding · String matching · Prediction coding · Variable length code · Hash table

✉ Liping Zhao
zhaoliping_jian@126.com

✉ Kailun Zhou
kailun_zh@tongji.edu.cn

Extended author information available on the last page of the article

1 Introduction

With the rapid advances of edge computing, cloud computing, machine learning, and associated display technologies and driven by the growing market for transferring display screen between devices, especially mobile devices, screen contents transmitted or stored in the form of image and video have become an important and popular type of image and video. The Internet has entered the 5G era. Images and videos emerging in various Internet applications are the main driver of bandwidth use, accounting for over 80 percent of consumer Internet traffic [3]. Real-time, low-delay transport of screen contents has become more and more popular in every aspect of daily life and work, such as wireless displays, second screen, screen sharing and collaboration, cloud computing, display stream compression, screenshot transferring, etc. [18]. In these inter-device-oriented applications, a coded representation of screen contents is needed. Next generation video coding standards that have technical requirements for screen contents coding include Versatile Video Coding (VVC) [22, 26] and the third generation of AVS (AVS3) [17].

Because the human eye is more sensitive to artifacts occurring in synthetic parts of images, it is necessary to explore lossless compression for image and video [18]. In particular, since most of nowadays mobile devices feature ultra-high quality display, lossless screen content coding (SCC) is usually required or preferred in these devices for users to obtain no-compromise visual experiences. Recently, enhanced soft context formation (SCF) technique [23] is proposed for lossless SCC and outperforms other state-of-the-art schemes by up to 33% bitrate savings.

Furthermore, in the 5G, AI, and IoT era, mobile devices and edge devices are growing rapidly. These devices usually require ultra-low power consumption and long battery runtime in all tasks including compute-intensive SCC. To address the issue, JPEG XS standard aims to design lossless low-latency and low-complexity lightweight image compression [20]. MPEG is also working on low complexity video coding enhancement [2]. It can be seen that low and ultra-low complexity has become a hot topic of image and video coding.

Compared with traditional camera-captured contents, screen contents have many unique characteristics such as sharp edges, arbitrary shapes at the transitions between foreground and background, computer-generated object, text, and line art of discontinuous-tone, many repeated identical patterns with a variety of shapes and sizes, large uniform areas, and limited number of colors in some regions. Hence, High Efficiency Video Coding (HEVC) standard has developed an SCC extension [19, 24, 33]. Intra Block Copy (IBC) [27, 34] is a dedicated HEVC-SCC tool to perform block matching within the same frame. Palette coding (PLT) [7] is another dedicated HEVC-SCC tool to code a CU with limited number of colors efficiently. Moreover, string matching (SM) [13, 14, 36–39, 41, 42] and other approaches [1, 30] are proposed to significantly improve coding efficiency for screen contents. The idea of the SM coding mainly comes from dictionary coding, e.g. LZ77 [46], ZLIB [6], LZMA [9] and LZ4 [16]. The latest screen content coding activities based on SM include an offset rotation mapping algorithm based on string matching [12] and new string matching approaches [15, 40, 43] for alpha image coding. It should be noted that nowadays PNG compressed image file format [21] is the most popular lossless coding format widely used for a variety of Internet applications, screenshots and other computer generated images in most devices. ZLIB is one of the basic coding tools used in PNG.

HEVC-SCC (HM-16.6 + SCM-5.2) achieves 70.2% bitrate reduction in AI configuration over H.264/AVC (JM-19.0) but has an encoding runtime ratio of 271% [11]. Compared with HEVC (HM-16.20), VVC (VTM-6.1) achieves 24.2% bitrate reduction in

AI configuration but the runtimes ratio is as high as 2716% for encoding and 187% for decoding [4].

Obviously, although VVC and HEVC standards have made excellent progress to improve coding efficiency, and existing SCC tools e.g. IBC, PLT, and SM have been proven to be very effective, they also increase coding complexity too much to be used in devices with requirement of low power consumption and ultra-low coding complexity. Thus, there is an urgent need to develop ultra-low complexity and high efficiency techniques for image and video coding, especially for SCC.

There are quite a few efforts to try reducing the coding complexity of HEVC-SCC. The encoding time for reduced-complexity Intra Block Copy (IntraBC) mode can be reduced by 17.07% on average compared with the conventional HEVC-SCC extension [27]. The encoding complexity of the hash-based block matching is 87% and 84% that of the HEVC range extension reference software HM12.1_Rext5.1 with intra and low-delay configurations, respectively [45]. Compared with the hash-based block matching scheme in the HEVC-SCC test model (SCM)-6.0, the fast hash-based inter-block matching scheme achieves 12% and 16% encoding time savings in random access (RA) and low-delay B coding structures [31]. Fast intra prediction method based on content property analysis for HEVC-based SCC can save 44.92% encoding time on average [10]. Compared with HM16.8 + SCM7.0, X265 can save 98.88% encoding time in fast configuration. Among them, X265 is the most popular and commercial-oriented relatively low complexity encoder implementation of HEVC-SCC. Compared with HEVC-SCC reference software, these efforts can achieve up to 98.88% coding complexity reduction, but is still far from enough for ultra-low complexity (additional 90% or more complexity reduction with almost same or even higher coding efficiency) screen content coding.

This paper proposes a string matching based ultra-low complexity and high efficiency lossless coding technique for screen contents. The technique covers three major coding phases, i.e. fast searching for optimal reference strings that match current strings being coded, prediction of unmatchable pixels to get prediction residuals, and entropy coding of string matching parameters and prediction residuals.

To find repeated identical patterns in an input image, condensed hash table (CHT) based fast reference string searching is proposed to speed-up the process of finding matching strings and unmatchable strings in the input image. To efficiently utilize the correlation between an unmatchable pixel and its neighboring pixels, coplanar prediction (CP) is proposed to obtain a predictor and predictor-dependent residual (PDR) is proposed to further reduce the entropy of prediction residuals. To achieve a good trade-off between complexity and efficiency, 4-bit-aligned variable length code (4bVLC) is proposed to code the prediction residuals of PDR and byte-aligned multi-variable-length-code (BMVLC) is proposed to pack at most three VLCs of string matching parameters into one byte. Both 4bVLC and BMVLC have the advantages of low complexity and high throughput by avoiding bit-by-bit operations and bitstream access.

The major contributions of this paper are as follows.

1) An ultra-low complexity and high efficiency coding framework for lossless screen content coding.
2) CHT based fast string matching search.
3) CP and PDR for prediction and residual coding.
4) 4bVLC and BMVLC for entropy coding.

For a set of 184 screen content test images commonly used to evaluate the performance of SCC tools, compared with PNG and X265 [8] (HEVC), experimental results show that the proposed technique has the advantages of both ultra-low coding complexity and high coding efficiency.

The rest of the paper is structured as follows. Section 2 provides Related work. Section 3 presents the details of the technique proposed. Experimental results are given in Section 4 while conclusions and future work are drawn in Section 5.

2 Related work

PNG is a lossless data compression algorithm derived from LZ77 [46]. PNG includes two stages of filtering in differential coding and compression. The filtering process is to transform the original data into a group of data with the smallest sum of absolute difference by difference method. The compression process means to compress the filtered pixels using the DEFLATE algorithm [5] (that is, Huffman coding is used for the compressed result of the LZ77 algorithm).

In X265, Lossless operation is theoretically simple. Rate control, by definition, is disabled and the encoder disables all quality metrics since they would only waste CPU cycles. Instead, X265 reports only a compression factor at the end of the encode.

A low-complexity lossless screen content coding scheme using the dictionary coding was proposed in [32]. The technique uses 1D-string matching and selects the horizontal scan mode and vertical scan mode adaptively based on the conventional rate-distortion cost measurement. The technique leverages 2D image features such as directional texture pattern and high correlation of neighboring blocks, and constrains the dictionary size within largest coding unit (LCU) and line from its left or upper neighbors.

United coding (UC) method for lossless screen content coding was proposed in [28]. UC unites intraframe hybrid coder and several lossless coding tools. In UC, several typical lossless coding tools such as dictionary-entropy coder, run-length encoding (RLE), portable network graphics (PNG) filters, and hextile coding are included.

A Pseudo-2D-matching (P2M) coder based enhancement to HEVC for screen contents was proposed in [29]. The P2M uses three matching modes including (1) vertically scanned string matching; (2) horizontally scanned 2D-shape-preserved matching; and (3) vertically scanned 2D-shape-preserved matching to select the optimal mode by minimizing rate-distortion.

A lossless compression algorithm based on string matching with high performance and low complexity was proposed in [14]. The main new ideas are using pixel instead of byte as the basic unit for 1D string searching and matching, adopting joint optimal coding of three parameters of literal length, match length and offset mapping for three parameters. According to the statistical characteristics of the encoding parameter offset, an offset rotation mapping algorithm was proposed in [12].

A flexible and uniform 2D string matching technique named universal string matching (USM) for general screen content coding (SCC) was proposed in [44]. USM includes three modes: general string (GS) mode, constrained string 1 (CS1) mode, and constrained string 2 (CS2) mode. When using USM to code a CU, one of the three string constraint modes is selected to code the CU based on its characteristics and each of the three USM coding modes plays an indispensable role.

The methods of [29, 32, 44] are all based on HEVC, and the method of [28] is based on H.264. Since both H.264 and HEVC were developed to improve coding efficiency at the cost of tremendous increase of coding complexity, none of these methods has low coding complexity.

The low coding complexity 1D string matching coding schemes of [14] and [12] use picture level raster scanning to convert a 2D picture into a 1D string. Picture level raster scanning has the problem that two strings with short 2D distance may be converted into two strings with long 1D distance. For example, 2D distance of 1 with horizontal distance of 0 and vertical distance of 1 is converted into 1D (linear) distance of picture width. Usually, the longer the distance between a current string and its reference string is, the more bits are spent to code the distance. Therefore, picture level raster scanning usually has low coding efficiency. To overcome the problem, this paper uses LCU level raster scanning to effectively reduce 1D distance. For example, 1D distance of picture width (e.g. 4096 for a 4 K picture) in the previous example is reduced to 1D distance of LCU width (e.g. 64).

3 An ultra-low complexity and high efficiency technique for lossless screen content coding

3.1 Framework of proposed technique

Figure 1 illustrates the framework of the proposed technique. Firstly, the input image of $W \times H$ pixels is partitioned into $M \times N$ LCUs, where the size of each LCU is 64×64 pixels, and $M = W/64$, $N = H/64$. Secondly, the image is scanned LCU by LCU in horizontal raster-scan order of LCU, and inside each LCU, the pixels are horizontally raster-scanned. Each pixel has three components (Y, U, V) or (R, G, B). Each component is also called a sample. A *condensed hash table* (CHT) *based fast string searching* is applied to the 1D input string to obtain both matching strings and unmatchable strings. Each matching string preceded by a (null or real) unmatchable string is represented by a 3-tuple (uml, ml, offset) and a string of unmatchable samples $\{UM_i\}$, $1 \le i \le \text{uml} \times 3$, if $\text{uml} > 0$. If $\text{uml} = 0$, then the unmatchable string is null. The 3-tuples are coded by *byte-aligned multi-variable-length-code* (BMVLC). *Coplanar prediction* (CP) is used to get predictors of unmatchable samples. Then *predictor-dependent residual* (PDR) is calculated as prediction residuals of the unmatchable samples. PDR is coded by *4-bit-aligned variable length code* (4bVLC) or fixed-length code. For an image, if 4bVLC spends more bits than fixed-length code, then PDR of the entire image is coded by fixed-length code.

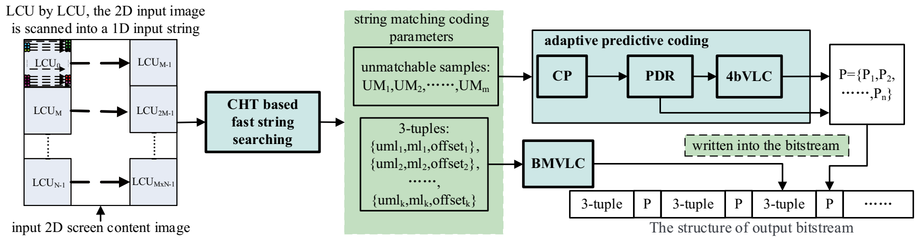

LCU by LCU, the 2D input image is scanned into a 1D input string
LCU$_0$
LCU$_{M-1}$
LCU$_M$
LCU$_{2M-1}$
LCU$_{N-1}$
LCU$_{MxN-1}$
input 2D screen content image
CHT based fast string searching
string matching coding parameters
unmatchable samples: UM$_1$,UM$_2$,······,UM$_m$
3-tuples: {uml$_1$,ml$_1$,offset$_1$}, {uml$_2$,ml$_2$,offset$_2$}, ······, {uml$_k$,ml$_k$,offset$_k$}
adaptive predictive coding
CP
PDR
4bVLC
P={P$_1$,P$_2$, ······,P$_n$}
BMVLC
written into the bitstream
3-tuple
P
3-tuple
P
3-tuple
P
······
The structure of output bitstream

Fig. 1 An ultra-low complexity lossless coding framework for screen content coding

3.2 Condensed hash table based fast string searching

The role of hash table is to quickly find the location (address) of the starting pixel of a potential reference string in a predetermined one-dimensional (1D) searching window by table-look-up. The 1D searching window is actually a 1D array storing pixels that have been coded before the pixel currently being coded and can be used as reference pixels. Each pixel in the 1D array has an address. The hash table is organized into multiple hash chains. Each chain links all addresses of pixels with the same hash values together to provide a set of pixel addresses for starting pixels of potential reference string search. When coding a current pixel, a 15-bit *hash_value* of the pixel is calculated by $hash\_value = \{[(Y << 16) + (U << 8) + V] \times 2654435761U)\} >> 17$, where Y, U, and V are three 8-bit components of a pixel, and the hash chain with the *hash_value* is selected for hash searching, which goes through the selected hash chain one node (address) by one node to start potential reference string searching one by one. Obviously, the more nodes a hash chain has, the more compute-intensive a hash searching is. A conventional hash chain includes all nodes of the same hash value and may have thousands of nodes and even more, resulting in high coding complexity. If some "insignificant" nodes can be "skipped" when building up a hash chain, then the hash searching complexity can be reduced without sacrificing the coding efficiency. A node is added to a hash chain whenever a corresponding current pixel has been coded as either a pixel of a matching string or as an unmatchable pixel. It turns out that nodes corresponding to pixels in the middle of a matching string are "insignificant". Based on the finding, a condensed hash chain is built up by adding only the nodes corresponding to unmatched pixels and the first two and last two pixels of any matching string to the hash table.

An example of a conventional hash chain vs. a condensed hash chain is illustrated in Fig. 2. In the example, the 15-bit hash value of the hash chain is 22330. The first node (address) of the hash chain is 140, which is stored in the hash-head of the hash table. Therefore, the hash chain starts from node 140. Figure 2a shows the conventional hash chain with 50 nodes linked by blue arrows. The first node 140 is followed by nodes 139 to 110 (30 nodes), 95, 70, 69, 24 to 9 (16 nodes). Figure 2b shows the condensed hash chain with 11 nodes linked by red arrows. The first node 140 is followed by nodes 139, 111, 110, 95, 70, 69, 24, 23, 10, 9. The number of nodes is reduced from 50 to 11.

It should be noted that N nodes on a hash chain mean that there are up to N potential reference strings to be searched. The searching process starts from the first node on the

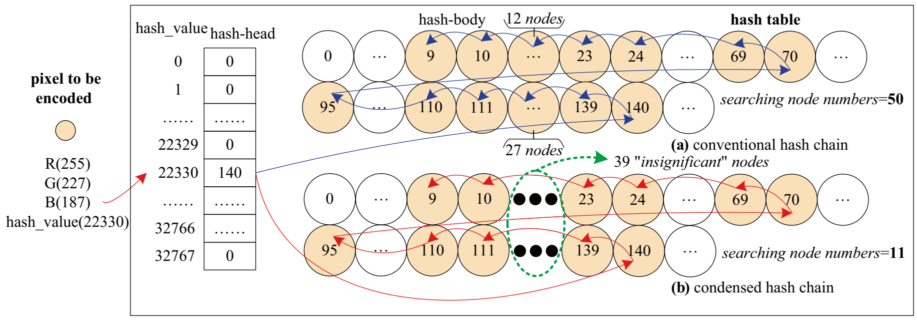


Fig. 2 Condensed hash table based string searching

hash chain with the same hash value as the current pixel, and finally the longest reference string is selected as the optimal reference string. As shown in Fig. 3, the proposed condensed hash table based fast string searching has the following steps.

(1) Search for special “above” position reference string with offset equal to 64. The offset equal to 64 has the highest percentage among all offset values. If the “above” matching reference string exists, then terminate the current string searching process and go to step (3), otherwise, go to step (2).
(2) Perform hash searching. The hash value of the current pixel being coded is calculated and used to go through the hash chain nodes in the CHT one by one until the final node is reached. No matter if a matching reference string is found or not, go to step (3).
(3) Add nodes to CHT. If a matching reference string is found in steps (1) or (2), then the nodes corresponding to the first two and last two (F2L2 in Fig. 3) pixels of the current string are added to CHT, otherwise, the node corresponding to the unmatchable pixel is added to CHT. If the last pixel of the image has been reached, then the string searching process is terminated, otherwise, go to step (1) to code the next pixel.

Experiments show that compared with conventional hash table, CHT can reduce total encoding runtime by about 20% without loss of coding efficiency.

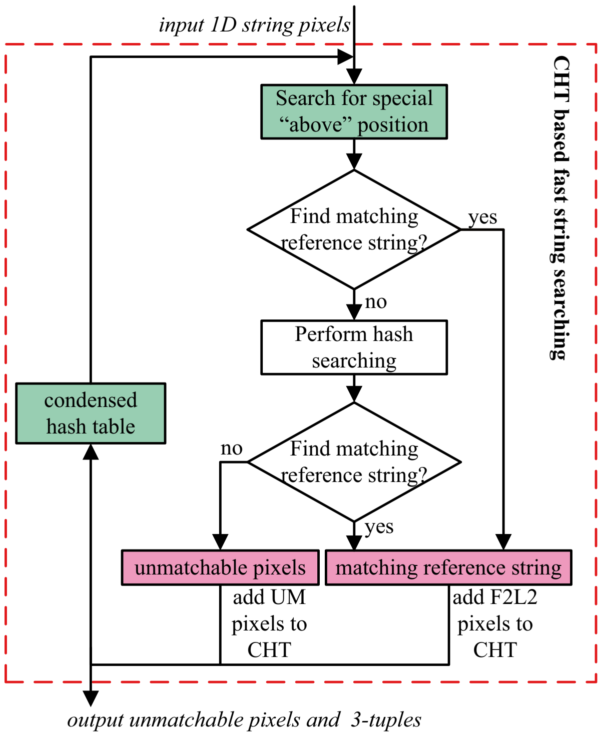

input 1D string pixels
CHT based fast string searching
Search for special “above” position
Find matching reference string?
yes
no
Perform hash searching
condensed hash table
no
Find matching reference string?
yes
unmatchable pixels
matching reference string
add UM pixels to CHT
add F2L2 pixels to CHT
output unmatchable pixels and 3-tuples

Fig. 3 CHT based fast string searching

3.3 Coplanar prediction and predictor-dependent residual

To the best of our knowledge, existing dictionary coding, SM coding, and palette coding techniques for lossless SCC perform entropy coding on raw unmatchable pixels directly without prediction. However, we observed that quite a few unmatchable pixels are anti-aliased boundaries or edges of solid color regions rendered by 2D computer graphics operation or shaded color areas rendered by 3D computer graphics operation. These rendering operations usually use linear interpolation model to generate pixels from neighboring pixels. Therefore, such an unmatchable pixel usually can be well predicted from its neighboring pixels using a coplanar model, i.e. assuming the unmatchable pixel and three pixels properly selected from its neighboring pixels satisfy the condition of four coplanar points.

To evaluate the effectiveness of prediction on unmatchable pixels, we calculated the entropies H(*UM*), H(*RO*), and H(*RP*) of the 8-bit raw unmatchable pixel samples *UM*, the ordinary subtraction based prediction residuals *RO* resulting from coplanar prediction (CP, described in Section 3.3.1), and the predictor-dependent residuals (PDR, described in Section 3.3.2) *RP*, respectively.

The entropy of a data set *D* is calculated as follows.

1) Let the value range S of $d \in D$ be $S = \{S_1, S_2, _{...}, S_n\}$.
2) Compute the frequency $F(S_k)$ of d being equal to S_k for $k = 1$ to n.
3) Compute the percentage $P(S_k)$ of $F\ (S_k)$ over N_D:

$$P(S_k) = \frac{F(S_k)}{N_D}, \tag{1}$$

where N_D is the number of total elements in *D*.

4) The entropy H(*D*) of the data set *D* is calculated by

$$H(D) = -\sum_{k=1}^{n} P(S_k) \log_2 P(S_k). \tag{2}$$

For each of the 184 screen content test images (see Section 4 for details) coded by the proposed technique, the entropies H(*UM*), H(*RO*), and H(*RP*) are plotted in Fig. 4a, where the black, blue and red curves are H(*UM*), H(*RO*), and H(*RP*), respectively. For about

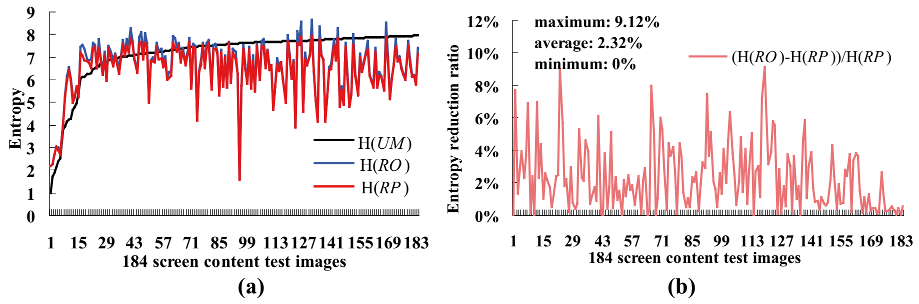


Fig. 4 Entropy curves related to unmatchable pixels of 184 test images: **a** H(*UM*), H(*RO*), H(*RP*); **b** [H(*RO*) – H(*RP*)]/H(*RP*)

80% of the screen content test images, H(*RP*) is smaller than H(*UM*) by 0 ~ 6 as shown in Fig. 4a, i.e. PDR reduces the bits spent to code an unmatchable sample (a sample is a pixel component such as R or G or B or Y or U or V or alpha) by 0 ~ 6 bits from the entropy point of view. Moreover, to evaluate the advantage of H(*RP*) over H(*RO*), for the 184 test images, Fig. 4b plots the percentages of [H(*RO*) − H(*RP*)]/H(*RP*), which show that H(*RP*) is always smaller than H(*RO*) with maximum entropy reduction ratio of 9.12% and average entropy reduction ratio of 2.32%.

It can be seen that CP and PDR can significantly improve the coding efficiency for unmatchable pixels.

3.3.1 Coplanar prediction

In CP, each sample with value Z and coordinates (x, y) is considered as a 3D point (x, y, Z). The 3D plane defined by three non-collinear points $Q_0 = (0, 0, 0)$, $Q_1 = (x_1, y_1, Z_1)$, $Q_2 = (x_2, y_2, Z_2)$ in the 3D space is given by the plane equation $(y_1Z_2 - y_2Z_1)x + (Z_1x_2 - Z_2x_1)y + (x_1y_2 - x_2y_1)Z = 0$. Therefore, for a coplanar point (x, y, Z) of the plane, Z is given by

$$Z = -\left[\left(y_1Z_2 - y_2Z_1\right)x + \left(Z_1x_2 - Z_2x_1\right)y\right] / \left(x_1y_2 - x_2y_1\right). \tag{3}$$

In coplanar prediction, the predictor P of an unmatchable sample X at coordinates (x, y) is given by $P = Z + Z_0$, where Z is calculated from three neighboring points (samples) Q_0, Q_1, Q_2 by (3) and Z_0 is to compensate the preadjustment of Q_0 from $(0, 0, Q_0)$ to (0, 0, 0) for simplifying the plane equation and (3).

Given a current sample X to be predicted, its direct neighboring samples are its left sample, top-left sample and above sample denoted by L, B, A, respectively. CP uses the most relevant trio (L, B, A) of neighboring samples properly selected from all neighboring samples to calculate the predictor P of X.

Let B have coordinates (0, 0), then A, L, X have coordinates (1, 0), (0, 1), (1, 1), respectively. Let Q_0, Q_1, Q_2 be (0, 0, 0), (1, 0, A–B), (0, 1, L–B), respectively, then $Z_0 = B$ and the coplanar predictor P_{cp} of X is given by $Z + B = -[-(A - B) - (L - B)] + B = A + L - B$, where Z is calculated using (3). This paper uses P_{cp} with clipping as the predictor P of X by

$$P = \text{clip}(0, 2^{bitdepth} - 1, P_{cp}) \tag{4}$$

where *bitdepth* is the bit depth of samples and usually is 8–12.

3.3.2 Predictor-dependent residual (PDR)

Consolidating the value range of a data set may increase the percentage of certain values to be taken, and thus reduces the entropy of the data set. The proposed PDR is a way to consolidate the value range of prediction residuals.

For any sample X and its predictor P with value range (expressed as an interval in math) $[-I, J)$, i.e. $-I \le X, P < J$, the ordinary subtraction based prediction residual $R = X - P$ has the value range of $(-K, K)$ with a size (i.e. length of the interval) of 2K, where $K = I + J$. However, because the value of P is known, the true value range of R is actually $(-I–P, J–P)$, which is P dependent and has the size of $I + J = K$. Taking the advantage of the feature and noting $-I–P \le 0 < J–P$, PDR divides the value range $(-I–P, J–P)$ into two subranges $(-I–P, 0)$ and $(0, J - P)$, and shifts the first subrange $(-I–P, 0)$ to the right by adding K to obtain $(-I–P + K, K) = (J–P, K)$. The second subrange $(0, J–P)$ and the shifted first subrange (J–P, K) are then recombined to obtain the final PDR value range (0, K). In this way, PDR consolidates the original value range of (−K, K) into new value range of (0, K) with only half size.

In this paper, I = 0 and J = $2^{bitdepth}$, so K = $2^{bitdepth}$. Therefore, for an unmatchable sample *UM* and its predictor *P* (P_{cp} described in Section 3.3.1), the predictor-dependent residual *RP* is calculated by

$$RP = \begin{cases} UM - P + 2^{bitdepth} & UM < P \\ UM - P & UM \geq P \end{cases}. \tag{5}$$

PDR has a wrap-around effect, e.g. $UM - P = -1, -2, -3, -4, -5, \ldots$ becomes 255, 254, 253, 252, 251,…, respectively for *bitdepth* = 8, which makes the subsequent entropy coding of *RP* complicated. Thus, the following wrap-around correction mapping is proposed to correct the problem and *mR* is the final mapped predictor-dependent residual:

$$mR = \begin{cases} RP \times 2 & 0 \leq RP \leq 2^{bitdepth-1} - 1 \\ \left(2^{bitdepth} - RP\right) \times 2 - 1 & 2^{bitdepth-1} \leq RP \leq 2^{bitdepth} - 1 \end{cases}. \tag{6}$$

After the mapping, the highly occurring values $UM - P = 0, -1, 1, -2, 2, -3, 3, -4, 4, -5, 5, \ldots$ become sequential $mR = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, \ldots$, respectively, which are suitable for entropy coding.

3.4 4bVLC and BMVLC

Existing entropy coding schemes such as CABAC used in HEVC require bit-by-bit operations and bitstream access which are compute-intensive and have low throughput while fixed-length-code has very low coding efficiency. Hence, to provides a good trade-off between coding efficiency and coding complexity, 4bVLC consisting of only 4-bit, 8-bit, 12-bit, and 16-bit codes is used to code the predictor-dependent residuals and BMVLC that packs at most three VLCs into one byte is used to code three string matching parameters including uml, ml, and offset, which denote the length of an unmatchable string, the length of a matching string, and the offset between a current string and its reference string, respectively.

To design a high efficiency BMVLC, we studied the value distribution of uml, ml, and offset and found:

(1) The percentage of uml = 0 and ml = 1 is 41.4%,
(2) The percentage of uml = 0 and ml > 1 is 44.0%,
(3) The percentage of uml > 0 and ml = 1 is 9.0%,
(4) The percentage of uml > 0 and ml > 1 is 5.6%.

Based on these findings, a high efficiency BMVLC is designed to code 3-tuple (uml, ml, offset) as to be described in Section 3.4.2.

3.4.1 4bVLC for PDR

As listed in Table 1, 4bVLC uses 4-bit, 8-bit, 12-bit, and 16-bit code to code *mR* (described in Section 3.3.2) in the four integer value ranges [0, 11], [12, 67], [68, 179], and [180, 255], respectively. Specifically, 4-bit-code equal to 0uuu (uuu = 000 ~ 111) specifies *mR* = 0 ~ 7.

Table 1 4bVLC for mR coding

1st 4-bit[1]	2nd 4-bit[1]	3rd 4-bit[1]	4th 4-bit[1]	*mR*
0uuu	–	–	–	0 ~ 7
10uu	–	–	–	8 ~ 11
11oo[2]	uuuu	–	–	12 ~ 59
1111	0uuu	–	–	60 ~ 67
1111	1uuu[2]	uuuu	–	68 ~ 179
1111	1111	uuuu	uuuu	180 ~ 255

Note 1-A u or o denotes a bit to code *mR*

Note 2-Excluding 1111

4-bit-code equal to 10uu (uu = 00 ~ 11) specifies *mR* = 8 ~ 11. 8-bit-code equal to 11oo uuuu (oo uuuu = 00 0000 ~ 10 1111) specifies *mR* = 12 ~ 59. 8-bit-code equal to 1111 0uuu (uuu = 000 ~ 111) specifies *mR* = 60 ~ 67. 12-bit-code equal to 1111 1uuu uuuu (uuu uuuu = 000 0000 ~ 110 1111) specifies *mR* = 68 ~ 179. 16-bit-code equal to 1111 1111 uuuu uuuu (uuuu uuuu = 0000 0000 ~ 0100 1011) specifies *mR* = 180 ~ 255.

4bVLC uses only 4-bit operations without any bit-by-bit operations and has ultra-low coding complexity with very high throughput. On the other hand, typically, more than 80% of *mR* is coded by 4-bit code. Therefore, 4bVLC provides a good trade-off between ultra-low coding complexity and high coding efficiency.

3.4.2 BMVLC for 3-tuples

BMVLC packs three subcodes, i.e. uml-code, ml-code, and offset-code into one or more whole bytes to code uml, ml, offset together, where uml ≥ 0, offset ≥ 1, ml ≥ 1 always hold.

BMVLC for a 3-tuple (uml, ml, offset) always starts with an uml-ml-code. As shown in Table 2, in the first byte, uml-ml-code uses 2 ~ 8 variable bits, ml-code and offset-code use the rest 0 ~ 6 variable bits. uml-ml-code equal to 00 or 01110 or 010 or 01111111 specifies uml = 0 and ml = 1. uml-ml-code equal to 10000 or 0110 or 10001 or 10001111 specifies uml = 0 and ml = 2. uml-ml-code equal to 100100 or 10010110 or 01111 or 1001010 or 10010111 specifies uml = 0 and ml = 3. uml-ml-code equal to 100110 or 10011110 or 1001110 or 10011111 specifies uml = 0 and ml = 4. uml-ml-code equal to 10100 specifies uml = 0 and ml = 5. uml-ml-code equal to 10101 specifies uml = 0 and ml = 6. uml-ml-code equal to 11000 specifies uml = 0 and ml = 7. uml-ml-code equal to 11001 specifies uml = 0 and ml = 8. uml-ml-code equal to 10110 or 10111 specifies uml = 0 and ml ≥ 9, and additional bits are needed to specify the value of ml. uml-ml-code equal to 1101 and 1110 specifies uml = 1 and 2, respectively and ml ≥ 1, and additional bits are needed to specify the value of ml. uml-ml-code equal to 1111 specifies uml ≥ 3 and ml ≥ 1, and additional bits are needed to specify the value of uml and ml. More details are given in Table 2.

BMVLC uses at least 2-bit operations and mostly byte operations without bit-by-bit operations and has ultra-low coding complexity with very high throughput. On the other hand, as mentioned in Section 3.4, more than 85% of 3-tuples have uml = 0 and most of them are actually coded by one byte code. Therefore, BMVLC also provides a good trade-off between ultra-low coding complexity and high coding efficiency.

Table 2 BMVLC for 3-tuple coding

1st byte[1]	+ byte (uml)	+ byte (ml)	+ byte (offset)	uml	ml	offset
00oooooo	0	0	0	0	1	64, 1 ~ 63
01110ooo	0	0	0	0	1	128, 65 ~ 71
010ooooo	0	0	1	0	1	$72 \sim 72 + 2^{5+8} - 1$
01111111	0	0	2	0	1	$72 + 2^{5+8} \sim 65{,}535$
10000ooo	0	0	0	0	2	64, 1 ~ 7
0110oooo	0	0	0	0	2	8 ~ 23
10001ooo[2]	0	0	1	0	2	$24 \sim 24 + 7 \times 2^{8} - 1$
10001111	0	0	2	0	2	$24 + 7 \times 2^{8} \sim 65{,}535$
100100oo	0	0	0	0	3	64, 1 ~ 3
10010110	0	0	0	0	3	4
01111ooo[3]	0	0	0	0	3	5 ~ 11
1001010o	0	0	1	0	3	$12 \sim 12 + 2^{1+8} - 1$
10010111	0	0	2	0	3	$12 + 2^{1+8} \sim 65{,}535$
100110oo	0	0	0	0	4	64, 1 ~ 3
10011110	0	0	0	0	4	4
1001110o	0	0	1	0	4	$5 \sim 5 + 2^{1+8} - 1$
10011111	0	0	2	0	4	$5 + 2^{1+8} \sim 65{,}535$
10100ooo	0	0	0/1/2	0	5	the same as ml = 4
10101ooo	0	0	0/1/2	0	6	the same as ml = 4
11000ooo	0	0	0/1/2	0	7	the same as ml = 4
11001ooo	0	0	0/1/2	0	8	the same as ml = 4
10110ooo	0	1	0/1/2	0	9 ~ 6 + 255	the same as ml = 4
10111ooo	0	$j \geq 2$	0/1/2	0	ml[5]	the same as ml = 4
1101mmmo	0	$k \geq 0$	1/2	1	ml[6]	1 ~ 256/257 ~ 65,535
1110mmmo	0	$k \geq 0$	1/2	2	ml[6]	1 ~ 256/257 ~ 65,535
1111mmmo	$i \geq 1$	$k \geq 0$	1/2	uml[4]	ml[6]	1 ~ 256/257 ~ 65,535

Note 1-An m or o denotes a bit to code ml or offset, respectively

Note 2-Excluding 10001111

Note 3-Excluding 01111111

Note 4-$\text{uml} = 3 + 255 \times (i - 1) \sim 3 + 255 \times i - 1$

Note 5-$\text{ml} = 7 + 255 \sim 65{,}541$ for $j = 2$ or $65{,}542 + 255 \times (j - 3) \sim 65{,}542 + 255 \times (j - 2) - 1$ for $j \geq 3$

Note 6-$\text{ml} = 1 \sim 6$ for $k = 0$ or $7 \sim 6 + 255$ for $k = 1$ or $7 + 255 \sim 7 + 65{,}534$ for $k = 2$ or $65{,}535 + 7 + 255 \times (k - 3) \sim 65{,}535 + 7 + 255 \times (k - 2) - 1$ for $k \geq 3$

3.4.3 Bitstream structure

Figure 5 shows the bitstream structure of the proposed technique. In Fig. 5, LSB and MSB are the abbreviation for least significant bit and most significant bit, respectively. The bitstream consists of two parts: header and body. The header uses four bytes to code the picture width, the picture height, and the flag specifying the coding method of PDR. If the flag is 1, 4bVLC is used, otherwise, fixed-length code is used. The body uses

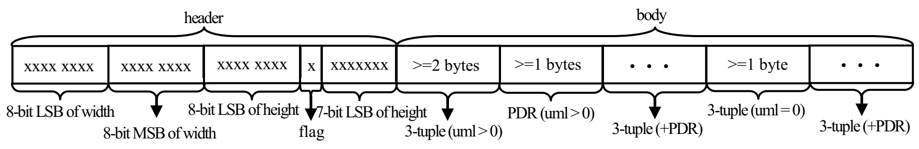


Fig. 5 Bitstream structure

BMVLC to code 3-tuples one by one. Each 3-tuple with uml greater than 0 is followed by 4bVLC or fixed-length code for PDR.

4 Experiments

4.1 Description of screen content test images

To evaluate the effectiveness of screen content coding methods, this paper uses twelve HEVC-SCC 8-bit common test condition (CTC) images used by JCT-VC [35] in the standard development and shown in Table 3 and 184 typical and commonly used [23] screen content images covering various application scenarios. PPM format of the 184 images can be downloaded via [25]. PPM format is converted to 8-bit RGB format, which is then converted to 8-bit YUV format using ITU-R Rec. BT. 709.

All formats of the 184 test images are available on https://github.com/SccVlsi/Hyper-screen-Content/tree/master/screen%20content%20test%20images.

Nine examples of the 196 images including twelve HEVC-SCC CTC images and 184 test images commonly used are shown in Fig. 6.

4.2 Experiment settings

The experiments evaluate and compare the lossless coding efficiency, the encoding runtime, the decoding runtime of the following three CODECs:

Table 3 Twelve HEVC-SCC CTC images used in the experiment

Category	Resolution	Sequence name
Text & Graphics with Motion	1920×1080	sc_flyingGraphics
Text & Graphics with Motion		sc_desktop
Text & Graphics with Motion		sc_console
Text & Graphics with Motion		ChineseEditing
Text & Graphics with Motion	1280×720	sc_web_browsing
Text & Graphics with Motion		sc_map
Text & Graphics with Motion		sc_programming
Text & Graphics with Motion		sc_SlideShow
Animation	1280×720	sc_robot
Mixed content	2560×1440	Basketball_Screen
Mixed content	2560×1440	MissionControlClip2
Mixed content	1920×1080	MissionControlClip3

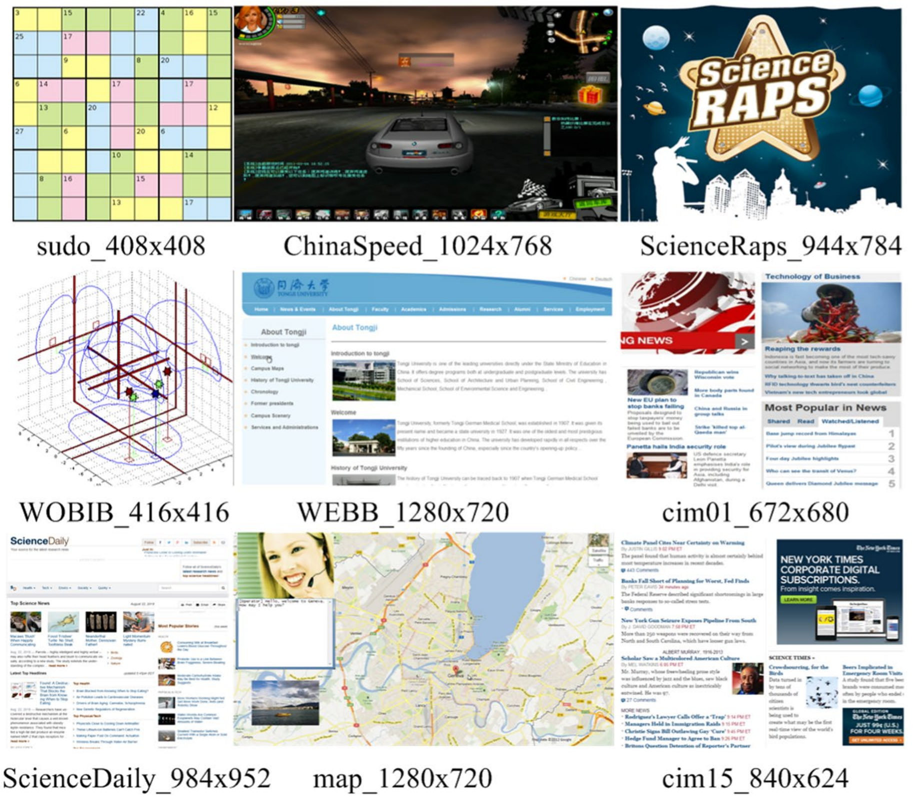


Fig. 6 Nine examples of the 196 images in downsized version

(1) PNG, one of the most widely used file formats nowadays for compressed image in RGB format, especially in a variety of mobile Internet applications (available on http://www.libpng.org/pub/png/libpng.html). The coding configuration options of PNG include level and strategy. The values of level range from -1 to 9 for ultrafast to slow. The values of strategy range from 0 to 4 for Z_DEFAULT_STRATEGY (normal data), Z_FILTERED (data generated by filter), Z_HUFFMAN_ONLY (mandatory Huffman coding), Z_RLE (mandatory run-length coding), Z_FIXED (forbidden to use Dynamic Huffman coding), respectively. The value of strategy used in this paper is 0.

(2) X265, one of the highest coding efficiency CODECs usually in YUV format, a commercial-oriented relatively low complexity encoder implementation of the latest video coding standard HEVC (available on https://bitbucket.org/multicoreware/X265/downloads/), where the decoding time is measured by HM reference software decoder (available on https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/) because X265 has only encoder. The coding configuration options of X265 include preset and lossless. The values of preset range from 0 to 9 for ultrafast to placebo. X265 can achieve lossless coding by using the –lossless option. It should be noted that although X265 is not based on the state-of-art techniques such as VVC and AVS3, it is still the most appropriate CODEC for comparisons of both ultra-low complexity and high efficiency since the state-of-art techniques are not developed to reduce coding complexity at all

and usually increase coding complexity by much more than 10 times with less than 30% coding efficiency improvement.

(3) The proposed fast 1D string matching (F1DSM) coding technique with ultra-low complexity and high efficiency.

All experimental results are evaluated in a Windows 7×64 system with Intel (R) Core (TM) i5-2430 M CPU (2.79 GHz).

The lossless coding efficiency comparison between a tested CODEC and an anchor CODEC for a single image is measured by the compressed bit ratio (CBR) of the two CODECs defined as follows:

$$\text{CBR} = \frac{comp\_bits\_tested}{comp\_bits\_anchor}, \tag{7}$$

where *comp_bits_tested* and *comp_bits_anchor* denote the compressed bits of the image coded by the tested CODEC and the anchor CODEC, respectively.

The overall lossless coding efficiency comparison between a tested CODEC and an anchor CODEC for 12 HEVC-SCC CTC images [35] or 184 test images commonly used [23] is measured by the total compressed bit ratio (TCBR) of the two CODECs defined as follows:

$$\text{TCBR} = \frac{\sum_{i=1}^{\text{N}} comp\_bits\_tested_i}{\sum_{i=1}^{\text{N}} comp\_bits\_anchor_i}, \tag{8}$$

where $comp\_bits\_tested_i$ and $comp\_bits\_anchor_i$ denote the compressed bits of the image i coded by the tested CODEC and the anchor CODEC, respectively and N = 12 or 184. To evaluate the encoder and decoder complexity, encoding and decoding software runtimes are also measured and the ratio of the tested CODEC runtime over the anchor CODEC runtime is calculated to compare the complexity of the two CODECs. The encoding runtime ratio, the decoding runtime ratio, and the encoding plus decoding runtime ratio are denoted by EnTime, DeTime, and EnTime + DeTime, respectively. A microsecond is used as the timing unit. Although the recorded times are only approximate, as they were measured using batch file, the data allows a sufficient comparison of the complexity of the different CODECs.

The compression ratio of an image i is defined as

$$\text{CR}_i = \frac{\text{W} \times \text{H} \times 3}{com\_bytes}, \tag{9}$$

where *com_bytes* denote the compressed bytes of the image i coded by one of three CODECs.

The average compression ratio of 12 HEVC-SCC CTC images [35] or 184 test images commonly used [23] is defined as

$$\text{avgCR} = \frac{\sum_{i=1}^{\text{N}} \text{CR}_i}{\text{N}}, \text{ where N} = 12 \text{ or } 184. \tag{10}$$

Table 4 Configurations for three CODECs

CODEC name	configuration parameter
F1DSM	Level 4
PNG	Fast (level 1), default (level 6), slow (level 9)
X265	Ultrafast (preset 0), default (preset 5)

The level or preset configuration parameters for the three CODECs are listed in Table 4. The parameters are used to select different tradeoff between coding efficiency and encoding complexity. Larger level or preset number means higher encoding complexity and higher coding efficiency.

4.3 Experimental results

For the 12 HEVC-SCC CTC images [35] and 184 screen content test images commonly used [23], the overall comparison between F1DSM (level 4, tested) and other CODECs (PNG or X265 as anchor) is shown in Tables 5, 6, respectively. And the average compression ratio of CODECs is shown in Fig. 7.

For nine typical screen content test images shown in Fig. 6, Table 7 lists the coding efficiency and the coding complexity comparison between F1DSM (level4) and PNG (default).

The experimental results can be summarized as follows.

(1) For the 12 HEVC-SCC CTC images and 184 screen content test images, the experimental results show that F1DSM has the advantage of both high coding efficiency and ultra-low coding complexity.

a) For the 184 screen content test images, as shown in Table 5, compared with PNG, the encoding time of F1DSM is 2.57% ~ 21.04% and the decoding time of F1DSM is 24.32% ~ 27.25%, while the ratio of the total compressed bits is 78.97% ~ 91.88%. In particular, compared with PNG in default and slow configurations, not only the encoding time of F1DFM is just 6.83% and 2.57%, i.e. ***93.17% and 97.43% reduction***, respectively, but also F1DSM achieves ***coding efficiency gain of 10.04% and 8.12%***, respectively.

Table 5 Comparison between F1DSM and PNG for 12 HEVC-SCC CTC images and 184 test images (RGB)

Test images	12 HEVC-SCC CTC images			184 Test images		
F1DSM vs. PNG	PNG (fast)	PNG (default)	PNG (slow)	PNG (fast)	PNG (default)	PNG (slow)
TCBR (%)	85.46	95.17	96.70	78.97	89.96	91.88
EnTime (%)	27.10	13.49	1.90	21.04	6.83	2.57
DeTime (%)	33.53	32.41	33.10	27.25	24.32	25.18
EnTime + DeTime(%)	28.57	16.00	2.54	22.46	8.53	3.42

Table 6 Comparison between F1DSM and X265 for 12 HEVC-SCC CTC images and 184 test Images (YUV)

Test images	12 HEVC-SCC CTC images		184 test images	
F1DSM vs. X265	X265 (ultrafast)	X265 (default)	X265 (ultrafast)	X265 (default)
TCBR (%)	60.87	78.95	48.54	64.33
EnTime (%)	3.50	1.41	2.39	0.96
DeTime (%)	2.99	2.14	1.93	1.54
EnTime + DeTime(%)	3.34	1.55	2.24	1.08

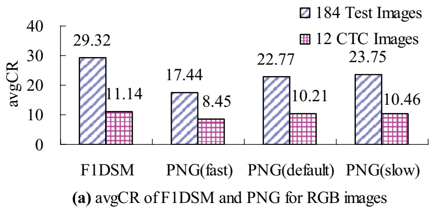
b) For the 184 screen content test images, as shown in Table 6, compared with X265 in default configuration, not only the encoding time and the decoding time of F1DSM is just 0.96% and 1.54% of X265, i.e. ***99.04% and 98.46% reduction***, but also F1DSM achieves ***coding efficiency gain of 35.67%*** over X265. Compared with X265 in fastest configuration, not only the encoding time and the decoding time of F1DSM is just 2.39% and 1.93%, i.e. ***97.61% and 98.07% reduction***, but also F1DSM achieves ***coding efficiency gain of 51.46%*** over X265.

c) For the 12 HEVC-SCC CTC images, as shown in Table 5, compared with PNG, the encoding time of F1DSM is 1.90% ~ 27.10% and the decoding time of F1DSM is 24.41% ~ 33.53%, while the ratio of the total compressed bits is 85.46% ~ 96.70%. In particular, compared with PNG in slow configurations, not only the encoding time of F1DFM is just 1.90%, i.e. ***98.10% reduction***, but also F1DSM achieves ***coding efficiency gain of 3.30%***.

d) For the 12 HEVC-SCC CTC images, as shown in Table 6, compared with X265 in default configuration, not only the encoding time and the decoding time of F1DSM is just 1.41% and 2.14% of X265, i.e. ***98.59% and 97.86% reduction***, but also F1DSM achieves ***coding efficiency gain of 21.05%*** over X265. Compared with X265 in fastest configuration, not only the encoding time and the decoding time of F1DSM is just 3.50% and 2.99%, i.e. ***96.50% and 97.01% reduction***, but also F1DSM achieves ***coding efficiency gain of 39.13%*** over X265.

e) As shown in Fig. 7, compared with PNG in fast, default and slow configurations and X265 in ultrafast and default configurations, the average compression ratio of F1DSM is the highest.

(2) In F1DSM, all of CP, PDR, 4bVLC, and BMVLC contribute to improving the coding efficiency. As a result, F1DSM shows significant advantage over PNG and X265 with both high coding efficiency and ultra-low coding complexity. Actually, as evaluated by the experimental



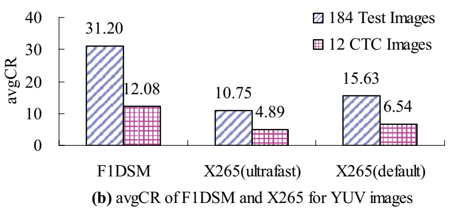


Fig. 7 The average compression ratio of CODECs

Table 7 Comparison of F1DSM vs. PNG for nine screen content images

Nine typical tested screen content images		F1DSM (leve4) vs. PNG(default)		
Image name in Fig. 5	resolution	CBR (%)	EnTime (%)	DeTime (%)
sudoku_408×408	408×408	**73.52**	7.76	18.36
ChinaSpeed_1024×768	1024×768	102.72	10.73	49.93
ScienceRaps_944×784	944×784	103.23	9.69	48.14
WOBIB_416×416	416×416	**95.75**	10.26	25.60
WEBB_1280×720	1280×720	**95.56**	12.31	26.14
cim01_672×680	672×680	103.34	14.52	36.77
ScienceDaily_984×952	984×952	**93.23**	12.91	28.33
map_1280×720	1280×720	**66.49**	14.77	34.27
cim15_840×624	840×624	**98.63**	16.67	34.04
TCBR(%)		**89.52**	**12.39**	**36.10**

results, in comparison with PNG and X265, the proposed technique can achieve 90% or much more reduction of coding complexity with even higher coding efficiency.

(3) For nine typical screen content images shown in Table 7, the overall coding efficiency of F1DSM (level4) is higher than PNG (default), while the encoding and decoding complexity is much lower than PNG. TCBR of F1DSM (level4) over PNG (default) is 89.52%, resulting in a big compressed bit saving of 10.48%. Moreover, the encoding time ratio of F1DSM (level4) over PNG (default) is only 12.39%, resulting in a very large encoding time reduction of 87.61%. Meanwhile, the decoding time of F1DSM is also significantly less than PNG by 63.90%.

(4) F1DSM can improve the coding efficiency of many different types of screen content, as illustrated in Fig. 6, quite significantly. F1DSM is very efficient for coding at least three types of common screen content. The first one is the content with some uniform areas. The second one is the content with sharp edges and arbitrary shapes at the transitions between foreground and background. The third one is various discrete-tone contents. F1DSM is also quite suitable for software implementations of screen content coding.

5 Conclusion and future work

This paper proposes an ultra-low complexity and high efficiency technique based on string matching for lossless screen content coding. The technique includes five main parts to cover three major coding phases. Condensed hash table (CHT) is proposed for fast reference string searching. Both coplanar prediction (CP) and predictor-dependent residual (PDR) are proposed for prediction of unmatchable pixels. 4-bit-aligned VLC (4bVLC) and byte-aligned multi-VLC (BMVLC) are proposed for entropy coding.

Experimental results show that the proposed technique significantly outperforms existing lossless coding techniques such as currently most popular PNG and the latest coding standard HEVC based X265. Not only the coding complexity of the proposed technique is considerably lower than the existing techniques, but also the proposed technique has higher coding efficiency than the existing techniques for most typical screen content images.

Future work includes: (1) Improving CP by using more trios (modes), context-based mode selection, and combining other prediction methods with CP; (2) Improving entropy coding efficiency further by context-adaptive or dynamic BMVLC, 4bVLC, 2bVLC, etc.; (3) Improving string matching efficiency by low complexity 2D-shape string matching; (4) Exploring other low coding complexity tools such as transform and quantization and harmonizing them with string matching for low complexity lossless and lossy screen content coding; (5) Optimizing low complexity string matching coding techniques for other types of hyper-screen content such as image segmentation maps widely used in many AI applications, e.g. auto-driving and object tracking.

Acknowledgements This work is supported by the Natural Science Foundation of Zhejiang Province (Grant No. LY19F020015), the National Science Foundation of China (Grant No. 61871289), the Public Service Technology Application Research Project of Shaoxing city (Grant No. 2018C10015), the Natural Science Foundation of Shanghai (Grant Nos. 18ZR1440600, 19ZR1461100).

References

1. Abdoli M, Henry F, Brault P et al (2018) Short-distance intra prediction of screen content in versatile video coding (VVC). IEEE Signal Process Lett 25(11):1690–1694
2. Baroncini V, Ferrara S, Ye Y (2018) Call for Proposals for Low Complexity Video Coding Enhancements. ISO/IEC JTC1/SC29/WG11, N17944
3. Beyond HEVC: Versatile Video Coding project starts strongly in Joint Video Experts Team [Online]. http://news.itu.int/versatile-video-coding-project-starts-strongly/. Accessed 3 Sep 2020
4. Bossen F, Li X, Suehring K (2019) AHG report: Test model software development (AHG3). JVET Doc JVET-P0003
5. Deutsch P (1996) DEFLATE Compressed Data Format Specification version 1.3 [Online], http://www.ietf.org/rfc/rfc1951.txt. Accessed 30 Jul 2021
6. Deutsch PL, Gailly JL (1996) ZLIB Compressed Data Format Specification version 3.3. RFC 1950
7. Guo LW, Pu W, Zou F et al (2014) Color palette for screen content coding. IEEE International Conference on Image Processing (ICIP), pp. 5556–5560
8. Guo L, Cock JD, Aaron A (2018) Compression Performance Comparison of x264, X265, libvpx and aomenc for On-Demand Adaptive Streaming Applications. IEEE Picture Coding Symposium, pp. 26–30
9. Lan CL, Xu JZ, Zeng WJ et al (2015) Compound image compression using lossless and lossy LZMA in HEVC. IEEE International Conference on Multimedia and Expo (ICME), pp. 1–6
10. Lei JJ, Li DY, Pan ZM et al (2017) Fast intra prediction based on content property analysis for low complexity HEVC-based screen content coding. IEEE Trans Broadcast 63(1):48–58
11. Li B, Xu J, Sullivan GJ (2015) Comparison of Compression Performance of HEVC Screen Content Coding Extensions Test Model 5 with AVC High 4:4:4 Predictive profile. JCTVC Doc JCTVC-V0033
12. Lin T, Yang YF (2019) Offset rotation mapping algorithm based on string matching for screen content coding. J Jishou Univ 40(3):28–32
13. Lin T, Zhang PJ, Wang SH et al (2013) Mixed chroma sampling-rate high efficiency video coding for full-chroma screen content. IEEE Trans Circuits Syst Video Technol 23(1):173–185
14. Lin T, Cai WT, Chen XY et al (2017) Lossless compression algorithm based on string matching with high performance and low complexity for screen content coding. J Electron Inf Technol 39(2):351–359
15. Lin T, Zhang DY, Zhao LP (2019) An improved entropy coding algorithm in string matching based on alpha image coding. J Jishou Univ 40(1):57–60
16. Liu WQ, Mei FQ, Wang CH et al (2018) Data compression device based on modified LZ4 algorithm. IEEE Trans Consum Electron 64(1):110–117
17. Luo F, Ma S (2017) The demand V1.0 for the new generation of AVS video coding technology. AVS N2495
18. Peng WH, Walls FG, Cohen RA, Xu JZ, Ostermann J, MacInnis A, Lin T (2016) Overview of screen content video coding technologies, standards, and beyond. IEEE J Emerg Sel Topics Circuits Syst 6(4):393–408
19. Requirements for Future Extensions of HEVC in Coding Screen Content, ISO/IEC JTC1/SC29/WG11, N14174, 2014

20. Richter T, Keinert J, Descampe A et al (2018) Entropy Coding and Entropy Coding Improvements of JPEG XS. 2018 Data Compression Conference, pp. 87–96
21. Schalnat GE, Dilger A, Truta C (2020) Libpng [Online]. http://www.libpng.org/pub/png/libpng.html. Accessed 4 Sep 2020
22. Segall A, Baroncini V, Boyce J et al (2017) Joint Call for Proposals on Video Compression with Capability beyond HEVC. JVET Doc JVET-H1002
23. Strutz T, Möller P (2020) Screen content compression based on enhanced soft context formation. IEEE Trans Multimed 22(5):1126–1138
24. Sullivan GJ, Boyce JM, Chen Y et al (2013) Standardized extensions of high efficiency video coding(HEVC). IEEE J Sel Topics Signal Process 7(6):1001–1016
25. Strutz T (2020) Enhanced soft context formation [Online]. http://www1.hft-leipzig.de/strutz/Papers/SCFenhanced-resources/. Accessed 2 Sep 2020
26. Sullivan G, Boyce J, Wiegand T (2017) Requirement for Future Video Coding(FVC). VCEG Doc VCEG-BD03
27. Tsang SH, Chan YL, Kuang W et al (2019) Reduced-complexity intra block copy (IntraBC) mode with early CU splitting and pruning for HEVC screen content coding. IEEE Trans Multimed 21(2):269–283
28. Wang SH, Lin T (2014) United coding method for compound image compression. Multimed Tools Appl 71(3):1263–1282
29. Wang SH, Lin T, Zhou KL et al (2015) Pseudo-2D-matching based enhancement to high efficiency video coding for screen contents. Multimed Tools Appl 74(18):7753–7771
30. Wang SQ, Zhang XF, Liu XM et al (2017) Utility-driven adaptive preprocessing for screen content video compression. IEEE Trans Multimed 19(3):660–667
31. Xiao W, Shi GM, Li B et al (2018) Fast hash-based inter-block matching for screen content coding. IEEE Trans Circuits Syst Video Technol 28(5):1169–1182
32. Xu M, Ma Z, Wang W, Wang X and Yu H (2014) Low-complexity dictionary based lossless screen content coding. 2014 IEEE International Conference on Image Processing (ICIP), pp. 3200–3203
33. Xu JZ, Joshi R, Cohen RA (2015) Overview of the emerging HEVC screen content coding extension. IEEE Trans Circuits Syst Video Technol 26(1):50–62
34. Xu XZ, Liu S, Chuang TD et al (2016) Intra block copy in HEVC screen content coding extensions. IEEE J Emerg Sel Topics Circuits Syst 6(4):409–419
35. Yu HP, Cohen R, Rapaka K et al (2015) Common Test Conditions for Screen Content Coding. JCT-VC doc JCTVC-U1015
36. Zhao LP, Lin T, Zhou KL et al (2016) Pseudo 2D string matching technique for high efficiency screen content coding. IEEE Trans Multimed 18(3):339–350
37. Zhao LP, Lin T, Zhou KL et al (2017) An efficient ISC offset parameter coding algorithm in screen content coding. Chin J Comput 40(5):1218–1228
38. Zhao LP, Zhou KL, Guo J et al (2018) A universal string matching approach to screen content coding. IEEE Trans Multimed 20(4):796–809
39. Zhao LP, Zhou KL, Guo J et al (2018) Pixel string matching for full-chroma screen and mixed content coding in AVS2. Chin J Comput 41(11):2482–2495
40. Zhao LP, Lin T, Zhou KL (2018) A byte-size multi-variable-length-code based string matching algorithm for alpha image coding. Telecommun Sci 34(11):96–104
41. Zhao LP, Zhou KL, Lin T et al (2019) A Universal string prediction approach and its application in AVS2 mixed content coding. Chin J Comput 42(9):2100–2113
42. Zhao LP, Lin T, Guo J et al (2019) Universal string prediction-based inter coding algorithm optimization in AVS2 mixed content coding. Chin J Comput 42(10):2190–2202
43. Zhao LP, Lin T, Zhang DY et al (2020) An ultra-low complexity and high efficiency approach for lossless alpha channel coding. Trans Multimedia 22(3):786–794
44. Zhou KL, Zhao LP, Lin T (2018) A flexible and uniform string matching technique for general screen content coding. Multimed Tools Appl 77:23751–23775
45. Zhu WJ, Ding WP, Xu JZ et al (2015) Hash-based block matching for screen content coding. IEEE Trans Multimed 17(7):935–944
46. Ziv J, Lempel A (1977) A universal algorithm for sequential data compression. IEEE Trans Inf Theory 23(3):337–343

Authors and Affiliations

Yufen Yang[1] · **Tao Lin**[1] · **Liping Zhao**[2,3] · **Kailun Zhou**[1] · **Shuhui Wang**[1]

[1] VLSI Lab, College of Electronics and Information Engineering, Tongji University, Shanghai 200092, China

[2] Department of Computer Science and Engineering, Shaoxing University, Shaoxing 312000, China

[3] Information Technology R&D Innovation Center of Peking University, Shaoxing 312000, China