Compressing of Medium- to Low-Rate Transform Residuals with Semi-Extreme Sparse Coding as an Alternate Transform in Video

Michael G. Schimpf, Graduate Student Member, IEEE, Nam Ling, Fellow, IEEE, Ying Liu, Member, IEEE

***Abstract*—Modern hybrid video codecs split their video content into blocks. These blocks are predicted and the difference between original and predicted block is calculated. This residual block is transformed by a discrete cosine transform (DCT) from the pixel domain into the frequency domain and quantized by a dead-zone quantizer (DZQ), which removes high frequency signals, depending on the quantization parameter (QP). DZQ has an extended zone around zero, which acts as a noise-gate, removing noise as intended but also removing useful signals. DCT and DZQ are not the optimal solution, they create distortions and side effects at lower bitrates. In contrast, sparse coding does not have these problems at lower bitrates, but it compresses inefficient at higher bit rates. We propose a method to add sparse coding as an alternate transform, which is controlled by a rate-distortion optimization (RDO) decision, in a semi-extreme sparse coding (SESC) setup and uses a semi-extreme dictionary training (SEDT) process. It is integrated into the High Efficiency Video Coding (HEVC) test model HM-16.18 and screen content coding (HEVC-SCC) test model HM-16.18+SCM-8.7. Experimental results demonstrate that the proposed method achieves a Bjontegaard rate difference (BD-rate) of up to 5.5% compared to the standard.**

***Index Terms*— video coding, sparse coding, sparse representation, orthogonal matching pursuit, screen content coding, HEVC, residual coding, intra prediction, KSVD**

I. Introduction

Traditional video codecs follow a block-based hybrid approach. Codecs divide video content into frames and frames into blocks. A block is predicted as accurately as possible by multiple methods, based on the quality, complexity, and maturity of the video codec. In the transform coding process, the predicted signals are subtracted from the original, which results in residual signals that represent the difference respectively the errors of the process, which the discrete cosine transform (DCT) then transforms from the pixel domain to the frequency domain. After the transformation, it is easier to identify high frequencies which will be quantized and removed, depending on the QP setting. $QP_{\{22\}}$ corresponds to the highest video quality which removes not much high frequencies, and $QP_{\{37\}}$ corresponds to the lowest video quality which remove a lot medium to high frequencies. The reason is that the human visual system (HVS) is more sensitive to lower frequencies than higher frequencies. In general, high frequencies consume a lot of bandwidth. In many cases, a dead-zone quantizer (DZQ) is used, which means that the quantization around the 0 area is extended. This extended dead-zone acts as a noise gate—it removes noise, but at the same time it cuts off useful signals. When a DCT and a typical DZQ are used, they create artifacts and distortion. These are visible especially in the low-rate setup as blocking and rippling artifacts. Therefore, both tools, the DCT and DZQ together, compress the signal very well, but with higher distortion at lower bitrates. Sparse coding is a reconstruction of a given signal $y \in R^n$ (n is the signal dimension) by a linear combination of the coefficients of sparse vector $\alpha \in R^K$ (K = number of coefficients), from an overcomplete dictionary $D \in R^{n \times K}$. A dictionary is a matrix with dimension n times K. Sparse means that vector $\alpha \in R^K$, which is a collection of coefficients, should have as few coefficients as possible (constrained by $L0$ norm). A coefficient is a combination of an index, which points to the column/atom of a dictionary, and a value or multiplicity, which indicates the magnitude. This dictionary atom is multiplied and then added to the reconstructed signal $y \in R^n$. To find a sparse vector α, the following equation is used to approximate the reconstructed signal to the original signal y**:**

$$\|y - D\alpha\|_2 \, s.t. \, \|\boldsymbol{\alpha}\|_0 \leq \varepsilon, \tag{1}$$

where $\|\cdot\|_2$ represents the $L2$ norm and $\|\cdot\|_0$ the $L0$ pseudo norm. However, the process of finding the sparsest possible α vector to solve equation (1) is a NP-hard problem [1]. There are alternatives to a brute force approach; the most well-known are as follows: matching pursuit (MP) [2], basic pursuit (BP) [3], or orthogonal matching pursuit (OMP) [4]. OMP tries to solve this issue in linear time, but the complexity is high [5]:

$$O(ST_D + S(n + K) + S^3) \tag{2}$$

where S is the sparsity constraint, K is the number of atoms of the dictionary, n is the signal dimension, and T_D is the time for updating a residual in an iteration. Sparse coding in general performs better—in low-to-medium frequency/energy (LMFE) signals—in terms of the mean-squared error (MSE) between the original and reconstructed signals, in comparison to DCT + DZQ. Additionally, if the sparsity is low, it is more cost efficient in a rate-distortion optimization (RDO) sense. General side effects of sparse coding for lower-to-medium bitrates are blurriness and a loss in detail. Sparse coding in general has a problem with a higher complexity and, in contrast to DCT + DZQ, the sparse coefficient dictionary index has a uniform distribution. This means it cannot be compressed efficiently—especially with a typical context-adaptive binary arithmetic coding (CABAC) engine, which is considered close to optimal in the lossless compression regime. The coefficient index cannot be compressed lossy because it must be precise. Therefore, CABAC needs to be turned off for these sparse coefficient atom indices. This contrasts with DCT + DZQ, where almost all side information can be encoded with the CABAC engine turned on, and the difference is significant.

The proposed semi-extreme dictionary training (SEDT) process in this work uses K-means clustering singular value decomposition (KSVD) [6]. A dictionary has the dimension $D \in R^{n \times K}$ (n is the signal dimension and K is the number of atoms). The first step is to generate training data from a separate High Efficiency Video Coding (HEVC) training run (proposed method turned off) where the residual data, before quantization for each block size, is written to disk if the absolute sum of this block is bigger than 0. The second step is to initialize the empty dictionary $D \in R^{64 \times 2048}$ (8×8 block size) by a random selection of the gathered training data. The third step, is to make the current dictionary better by minimizing the error further, with singular-

value decomposition (SVD). SVD is $U\Sigma V^T$ where U is the orthogonal basis vectors, Σ is the singular values (eigenvalues squared), and V^T is the set of eigenvectors. U represents the orthogonal basis vectors, sorted in order of decreasing ability to represent the variance in the dataset. The finding of U is taken to update the dictionary. The fourth step is to check if the number of predetermined iterations has reached a specific threshold—or if the error went below a specific threshold—and if not, jump back to step three and repeat until the number does. If so, the dictionary D is finished.

This paper distinguishes between two major content areas: Camera Capture Content, which is handled by HEVC (HM-16.18) [7], and screen content, which is handled by HEVC-Screen Content Coding (HEVC-SCC) (HM-16.18+SCM-8.7) [8], which is an extension of HEVC. HEVC and HEVC-SCC have the same software architecture. HEVC-SCC is basically an extension of HEVC that adds important screen content coding tools, like the following (explained in [9]):

- Adaptive Color Transform (ACT)
- Adaptive Motion Vector Resolution (AMVR)
- Cross-Component Prediction (CCP)
- Intra Block Copy (IBC)
- Palette Mode (PM)
- Residual Rotation (RR)
- Residual Differential Pulse Code Modulation (RDPCM)
- Transform Skip (TS).

The idea is to take advantage of screen content like graphics, text, mixed content, and animation, because it has some important properties [9]:

- no sensor noises
- large uniform flat areas
- many repeated patterns
- highly saturated and limited colors
- high contrast
- discrete tones
- sharp edges,
- high frequencies in certain regions

In this paper we propose a semi-extreme sparse coding (SESC) method for video compression that uses sparse coding as an alternate transform to DCT. The decision whether to use the proposed method or DCT is based on a RDO decision, which is an important tool in video compression:

$$j(K) = d(K) + \lambda r(K) \quad (3)$$

$$\lambda = 0.57 \times 2^{\frac{QP-12}{3.0}} \quad (4)$$

where K is the number of nonzero coefficients and $QP_{\{22,27,32,37\}}$ is the quantization parameter. The idea of RDO [8, 10], as depicted in formulas (3) and (4), is to calculate an overall cost $j(K)$ for a specific combination of distortion $d(K)$, which is the MSE between the original and reconstructed image block; the number of bits needed $r(K)$ (after entropy coding); and λ, the Lagrange multiplier, which controls the balance between distortion and number of bits. SESC is implemented with an in-loop quantization (ILQ) approach that helps to reduce complexity. The whole proposed method is separated into three parts: training, analysis, and encoding. All parts are optimized for LMFE signals, emphasize sparsity over distortion (SoD), and are strictly limited to a maximum upper bound (MUB4) of only four sparse coefficients (regardless of block size and $QP_{\{22,27,32,37\}}$ setting). The SEDT process is additionally adjusted by the following modules: a category approach (CAT) where the test sequences are grouped into categories based on the common test conditions (CTC) [11, 12]; training data is down-sampled by a unique down-sampling approach (QPD), which optimizes the training data and converges/generalize the training process faster and creates a semi-extreme dictionary that emphasizes sparsity over distortion; and extremely overcomplete dictionaries, with a k-size of 2,048. The analysis process is additionally supported by the following modules: a residual classifier (RClass) which passes LMFE signals and a λ Lagrange multiplier adjustment (LMA) which emphasizes sparsity over distortion. Moreover, the encoding process uses a sparse coefficient quantization process (SQP), which utilizes only a subset of the standard rate-distortion optimized quantization (RDOQ) standard process. The coefficient encoding for the proposed method relies on an adapted sparse coefficient encoding process (ACP), which uses the standard transform coding process with optimizations and its own context models for CABAC encoding.

The remainder of this paper is organized as follows: section II investigates related works, section III explains the proposed method, section IV depicts implementation details, section V presents experimental results, and section VI provides the conclusion.

II. Related Work

Sparse coding as a research area is remarkable, incorporating numerous subtopics for use in many applications. We concentrate on sparse coding in video compression. Sparse coding is used for the transform coding of the inter part (temporal, video), and intra part (spatial, video, and image). Video coding includes both parts: inter and intra, but in many cases sparse coding research in video compression concentrates on the inter part [5, 13-18]. The intra part is usually researched in depth by image coding. One conference paper from Kang et al., uses sparse coding for inter and intra [19] coding, and our research [20] compressed only the intra part. Veterli and Kalker were one of the first, in 1994 [18], to research the possibility of using matching pursuit in video coding, based on the findings of Mallat and Zhang [2] from 1993, before Bergeaud and Mallat [21] came up with a similar idea to use matching pursuit in image compression in 1995. They used a constructed dictionary with prototype waveforms, which learns from past frames, to compress motion prediction transform residuals. They also used a matching pursuit algorithm with a RDO cost calculation. Neff and Zakhor investigated, in their conference/journal papers [16, 17], matching pursuit for very low bitrates, with dictionaries of 2-D separable Gabor functions, to compress motion residual blocks. They replaced the DCT transform completely with the matching transform. Their proposed method is used to compress at very low bitrates, and they replace the DCT transform, instead of adding another alternate sparse coding/matching pursuit transform, which is nowadays a valid solution. Al-Shaykh et al. [22] expand on the idea from [16, 17] to scalability and arbitrary shape coding. Kang et al. was one of the first in video compression in 2011 [19] to train a dictionary by KSVD, and they used MP to code inter and intra prediction residuals. It replaces the core transforms through an adaptive mixed transform scheme. Their follow-up paper [5], expanded on this idea and put DCT and sparse coding in a sequence for a two-layered approach. Xiong et al. based their work [23] on a super-resolution paper [24] from Yang et al. and used a three-layer approach for low- to high-frequency dictionaries. In their paper [15], Xue and Wang investigated OMP in combination with a self-adaptive dictionary for inter-motion residuals. They included the DCT transform matrix in their dictionary, which combined DCT and their proposed method into one solution. Song et al. elaborated in [14] on anisotropic correlations (high frequencies/sharp edges), and they were the first to create a mixed DCT/Sparse coding approach, determined by a RDO calculation. Zhang and Yeh loosely investigate in [13] the work of Kang [5, 19], an adaptive dictionary learning approach for

inter-prediction residuals, which are learned from a previous intra-frame. The significant difference with the other research is that the created dictionaries have only 16 atoms, for all block sizes from 4×4 to 32×32, which represents an undercomplete dictionary, which is usually not considered sparse coding because a fundamental property of sparse coding is the use of an overcomplete dictionary. We elaborate in [20], on the use of sparse coding for screen content coding in video compression for intra-prediction residuals. In this paper, we extend our proposed method to inter-prediction residuals and to camera captured content and other improvements.

III. Proposed Method

Using sparse coding as an alternate transform for prediction residuals in video compression does not guarantee bitrate savings; there is a limited range which creates BD-rate savings. In many cases a sparse coding setup expands rather than compresses. A significant reason for this is that the standard transform coding process, which uses DCT + DZQ in combination with CABAC encoding, performs well and has improved over the last 50 years, which makes the standard process very efficient. An alternative sparse coding transform needs at least one bit that signals either that the proposed method is used (Flag = 1) or that the standard method is used (Flag = 0). This creates an additional overhead for all blocks and block sizes, even if the proposed method is used only for around 20% of the blocks, although 100% of all blocks need that additional flag (turned on/off). Another problem is that a sparse coefficient dictionary index has a uniform distribution, in contrast to DCT + DZQ coefficients, so it needs to be encoded with a fixed length (CABAC bypass). The fixed length encoding makes it unlikely for use in the high frequency or high quality setting.

A. *Semi-Extreme Sparse Coding (SESC)*

To get as close as possible to an extreme sparse representation, the proposed method follows the following three important rules. First, the proposed method targets only LMFE signals, because the number of sparse atoms are very limited based on the assumption that DCT + DZQ is not optimal for LMFE signals and that sparse coding is better in this signal range, if and only if the sparse approximation is as sparse as possible. The reason for this low sparsity constraint is due to the uniform distribution of the sparse coefficient indices. These indices needs to be encoded and compressed into the bitstream in fixed length (CABAC bypass), which is a problem because the proposed method RDO competes with DCT, where almost all information are encoded with the CABAC engine on, meaning that DCT coefficients can be encoded much more efficiently than sparse-coded coefficients. Some researchers try to alleviate this problem by using Huffman coding [5,16-17, 22], but in general the indices distribution is uniform, which makes its efficient compression difficult. Second, the SoD approach—as a result of the aforementioned sparsity constraint—necessitates that the proposed method needs to make the sparse coding part as sparse as possible even if distortion increases. To achieve this, we adjusted the Lagrange multiplier λ, which is shown in formula (3) and (4) and has the property to balance both the distortion and the number of bits in the RDO cost calculation. The proposed method adjusts λ towards SoD because of the uniform distribution of the sparse coefficient indices. λ is increased by +5% for all block sizes and $QP_{\{22,27,32,37\}}$ settings. The selection of the $QP_{\{22,27,32,37\}}$ parameter are based on the CTC [11, 12] requirements. This adjustment is only used for the proposed method-encoded blocks; the standard HEVC and HEVC-SCC use the original λ value. Third, because the MUB4 strictly limits the maximum number of sparse coefficients to four for all block sizes and $QP_{\{22,27,32,37\}}$ settings, this unusual low upper bound limit of sparse coefficients makes it unlikely that the proposed method is used for any high frequency block.

B. *Semi-Extreme Dictionary Training (SEDT) process*

The SEDT process uses KSVD, is optimized for SoD, and focuses on LMFE, using a novel QPD approach, which is an important optimization, to take training data from a higher quantization setup, to then be used for the test execution. In general, higher quantization encourages a sparser representation because—with a higher QP setting, which means lower quality—the intra/inter-prediction residuals are much more quantized, so that they get smaller, and coarser, which is better for the learning process. Furthermore, KSVD converges/generalize much faster to the given signal. This approach is inspired by a down-sampling method from Xiong et al. [23], but they down-sampled the resolution of the frames themselves and used this down-sampled signal for their dictionary-learning process. The proposed method instead lowers the quality and does not down-sample the resolution.

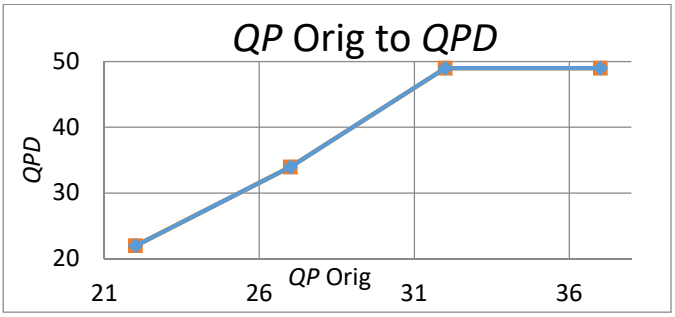

Fig. 1. quantization parameter (QP) original value to QPD value

In Fig. 1 the used QP original to QPD settings are shown. $QP_{\{22\}}$ test execution used $QP_{\{22\}}$ training data because it is not relevant at all, reason is that the partial BD-rate savings for this QP setting are always limited and approximately around 0.1% to 0.3%, which is insignificant. $QP_{\{27\}}$ test execution used the training data from $QP_{\{34\}}$ because it is a better quality setup and profits from training data which is closer to the original setting. $QP_{\{32,37\}}$ test execution is considered lower quality and can be trained by $QP_{\{49\}}$ which represents the lowest quality training data. Another major part of this process involves the creation of extremely overcomplete dictionaries with a bigger number of K atoms, which gives the KSVD/OMP algorithm [4, 6] more overall atoms to choose from. The reason for this is that, when a dictionary can offer a bigger variety of building blocks, the average sparsity is lower. The KSVD training process [6] offers only limited amount of hyper-parameters to adjust the learning process to a specific range of signals:

- K size is the number of atoms in the dictionary
- l is the number of sparse atoms used for the internal OMP algorithm to calculate the error
- i is the number of iterations to execute

The dictionary K size is 2,048; the number of atoms is set to two, with 800 iterations per dictionary. Parameter l is significant because it is the number of sparse coefficients that the internal KSVD algorithm uses for executing the OMP part, which is responsible for the learning update and for calculating the distortion. If l is set to a very low number like two, which we selected for our method, it encourages sparsity and requires less bits, although it tends to have higher distortion. In contrast, if l is set to a higher number like 16, a dictionary needs on average more sparse coefficients and more bits to create a signal with low distortion. Distortion can be minimized by the number of training epochs. All of these parts of our SEDT process create dictionaries that are optimized for low sparsity with as few bits as possible.

C. Proposed Method Block Diagram

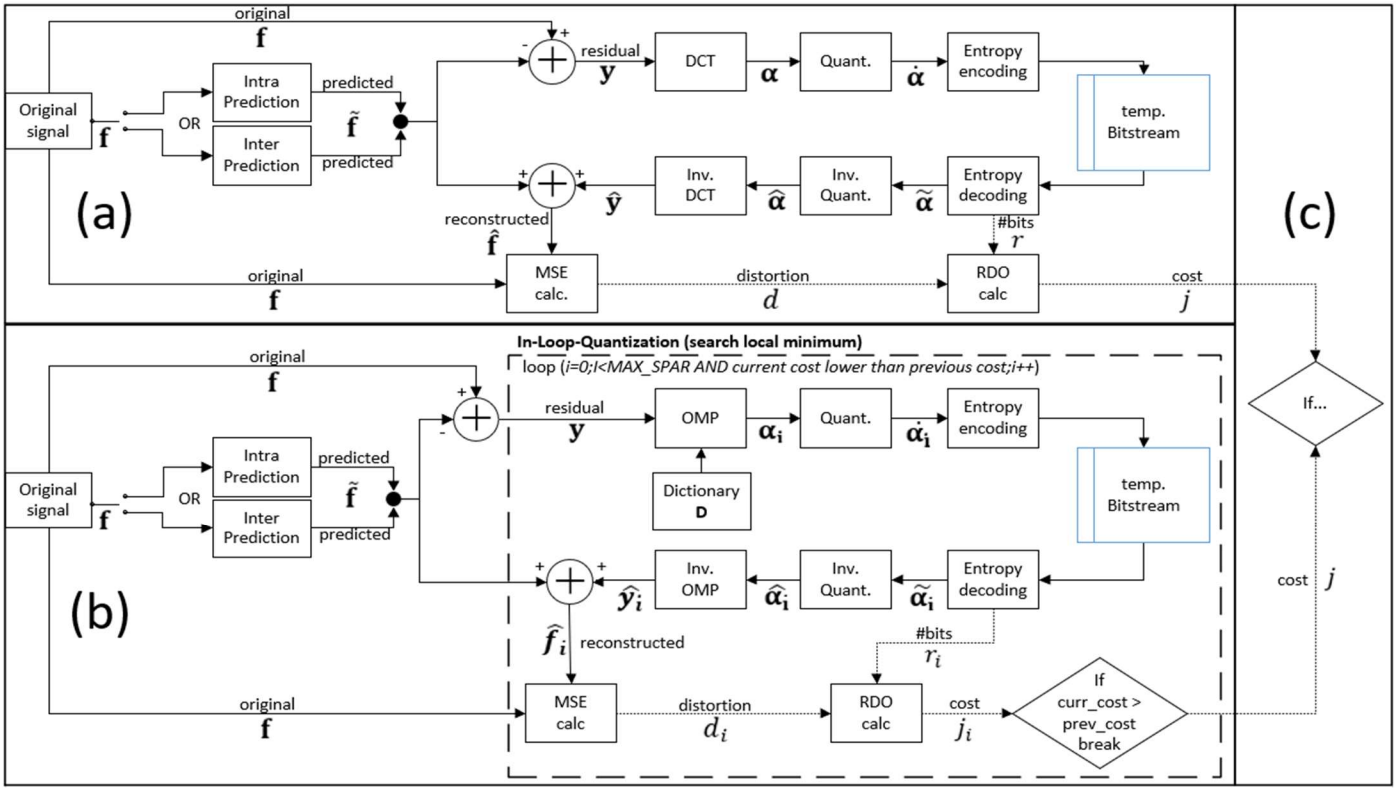

Fig. 2. Proposed Method Block Diagram

Fig. 2 shows the block diagram of the proposed method. It is separated in three parts: (a) shows the standard DCT transform and coding process; (b) shows the proposed method transform and coding process, which is probed if and only if DCT is selected as the candidate for encoding in previous step (a); and (c) shows an additional step which compares both costs and selects the solution with the lower cost, which is finally encoded in the bitstream. In general, the standard DCT process is executed in the following way: the original signal, which is represented by f and can be any block size $\{4\times4, 8\times8, 16\times16, 32\times32\}$, is intra/inter-predicted $\tilde{f}$. y is the residual signal which is calculated by original – predicted. The standard process transforms the residual with DCT into the frequency domain which is represented as a DCT coefficient vector α. The DCT coefficients are then quantized into vector $\dot{\alpha}$. The video codec then usually entropy encodes it into a temporary bitstream, subsequently decoding it back to vector $\tilde{\alpha}$ and inverse quantizing it into vector $\hat{\alpha}$. Inverse DCT transform is executed, which is represented by the reconstructed residual vector $\hat{y}$. To create the reconstructed signal $\hat{f}$, $\hat{y}$ and $\tilde{f}$ are added together. The distortion d is calculated between f and $\hat{f}$, and the number of bits r is taken from the previous executed entropy-coding process. Finally, the distortion j is calculated by formula (3). This is done to simulate a whole transform coding process in order to get as close as possible to realistic numbers. The proposed method (b) makes no difference between intra/inter-prediction, or which modes or other parameters are used in the prediction process itself. A significant difference of the proposed method (b) to the standard DCT process (a) is that (b) uses an ILQ scheme. The standard process transforms the residual signal y in one step into the frequency domain α and follows the process depicted in (a) to finally calculates the cost j. The proposed method does this for every finding of a sparse coefficient in a loop, and the loop is shown in Fig. 2 part (b) as a dotted rectangle. All loop dependent vectors are marked with the subscript i. This means that quantization is part of the OMP algorithm itself, which is calculated after every finding of a new sparse coefficient, it includes not only the quantization part, but also the entropy encoding/decoding process, inverse quantization, inverse transform, reconstruct residual, and cost calculation for every found sparse coefficient. The process depicted in Fig. 2 (b) takes the residual signal y and executes one OMP iteration at a time to find the next best sparse coefficient that minimizes the error. With every iteration it adds one more sparse coefficient to vector α_i. It is then quantized into $\dot{\alpha}_i$, entropy encoded into the bitstream, decoded back to $\tilde{\alpha}_i$, inverse quantized into $\hat{\alpha}_i$, and then inverse transformed into $\hat{y}_i$ —which represents the reconstructed residual signal and is then added to the predicted signal $\tilde{f}$ to represent the reconstructed signal $\hat{f}_i$. The distortion d_i is calculated, the rate r_i is taken from the entropy coding process, and finally the cost j_i for the current iteration is computed. The ILQ process searches for a local minimum and calculates the cost of every iteration separately. With every ILQ iteration, the cost is compared to the previous cost, and as long as the cost is going down, the next iteration is executed unless the cost goes up again, which would indicate that the minimal cost was found in the previous iteration. In this case the last sparse coefficient is deleted from α_i, which then represents the local minimum solution. With every iteration of the ILQ, the cost j_i is calculated by an adapted RDO method, where λ is increased by 5% to emphasize distortion over bitrate. The maximum sparsity is a global constant MUB4 and set to four, independent of block size or $QP_{\{22,27,32,37\}}$ setting. When the local minimum solution was found, the cost of the DCT and the proposed method approach is compared in (c), and the solution with the lower cost will be finally encoded into the bitstream. It is possible that this local minimum approach is not always sufficient because quantized spare coefficients are sometimes chaotic in nature, but MUB4 constraints represent an upper bound, that limits the probability of significant error margins.

D. Category Approach (CAT)

For the proposed method we used a CAT approach which is loosely inspired by [25], where they divided training samples into different subsets. Our dictionaries are trained by categories based on the CTC [11, 12] classes. This simultaneously simulates an application-specific approach and an online-learning approach. The training data, which includes basically all intra-predicted residual blocks with an absolute sum > 0, is created by a separated HM-16.18 [7] and HM-16.18+SCM-8.7 [8] run, for the HEVC and HEVC-SCC configuration. All CTC [11, 12] test sequences are strictly split, with 20% of the frames for training purposes, which remain unseen for the test execution, and 80% of the frames for the test execution. These trained files are grouped and randomly shuffled into the mentioned categories (based on CTC [11, 12]), which goes into an offline KSVD training process [6]). This split is unusual, but mixing a setup of similar test sequences in one category simulates noise and distortion, which would not be the case in a pure online learning solution, which concentrates only on the current test sequence. Selecting 20% as unseen training data and 80% as test data is an arbitrary cut, and can be minimized further, but it demonstrates that not much training data is needed to create a useful semi-extreme dictionary. We used for convenience reasons only intra-prediction residuals as training data.

Table I
HEVC Class A1 to D Resolutions

HEVC Category	Resolutions
Class A1 (4K)	4096×2160 &
Class A2 (4K)	3840×2160
Class B (HD)	1920×1080
Class E	1280×720
Class C	832×480
Class D	416×240
Class F	1280×720 & 1024×768 & 832×480

Table I shows for HEVC and camera-captured content; the common test conditions categories [11] are mainly based on their resolution.

Table II
HEVC-SCC Categories and Resolutions

HEVC-SCC Category	Resolution
Animation (A)	1024×768 & 1280×720
Mixed Content (MC)	1920×1080 & 2560×1440
Text, Graphics, Motion (TGM720)	1280×720
Text, Graphics, Motion (TGM1080)	1920×1080

Table II shows, for HEVC-SCC, the common test condition categories which are mainly based on their content and not on the resolution itself [12].

E. Summary

The originality of this research concentrates on the semi-extreme approach which is in contrast to the extreme sparse representation, as mentioned in [6], which is considered if and only if one sparse coefficient is used with a coefficient value of 1. This constraint seems to be too restricted. Semi-extreme is easing this condition because the sparse coefficient dictionary indices have a uniform distribution, in contrast to DCT + DZQ coefficients, so it needs to be encoded with a fixed length (CABAC bypass). The fixed length encoding makes it unlikely for use in the high frequency or high quality setting. As a consequence, the proposed method uses a novel SESC approach, which means in average less than two sparse coefficients are allowed with a non-restricted coefficient value, additional sparsity is more important than distortion. Other adjustments are integrated, like an extreme overcomplete dictionary, with 2,048 atoms. The training process is adapted; the proposed method targets basically only low frequency signals which can be in average encoded with a minimum of sparse coefficients. This leaves high frequency signals out of the process.

IV. Implementation Details

A. Sparse Quantization Process (SQP)

The standard RDOQ [10] process from HEVC is not used because it is not well designed to handle sparse coefficients—rather it is optimized to quantize DCT coefficients. Part of the difference is that the direct current (DC) component does not exist for sparse coefficients. Therefore, the RDOQ process analyzes the first sparse coefficient value and determines how to quantize the following sparse coefficient values. In general, sparse coefficient values are on average higher, and they strictly decrease from the most to the least important sparse coefficient value. This means that, if the standard RDOQ process is used for coding the proposed method's sparse coefficient values, the first coefficient value is quantized as expected; however, all the following sparse coefficient values would be quantized to zero, with almost no exception. The proposed method uses only a subset of the RDOQ process, which is the Qstep calculation:

$$Qstep(QP) = \left(2^{\frac{1}{6}}\right)^{QP-4} \tag{5}$$

where $QP_{\{22,27,32,37\}}$ is the quantization parameter. But in general, it is used to calculate a staircase quantization scheme, and our SQP method uses it in a scalar way, which means that, per $QP_{\{22,27,32,37\}}$ setting, there is simply one and only one divider/multiplier for every sparse coefficient, independent of the importance of the coefficient. This is in contrast to the standard process, where the DCT quantization has its own quantization matrix with a different quantization for different DCT coefficients, depending on the position of the coefficients in the matrix. The reason for this is that sparse coefficients are organized in a linear vector and not in a coefficient matrix.

B. Adapted Sparse Coefficient Encoding process (ACP)

In general, all sparse coefficient indices are coded in fixed-length code because of the uniform distribution. The proposed coefficient coding process follows the "transform coefficient coding" [26] with its own context models and adjustments. One adjustment, for example, is the significance map from the transform coefficient coding process, which is deactivated for the proposed method because all sparse coefficient values are significant with no exception. The reason here is that if a coefficient value is quantized to zero, the whole sparse coefficient will be removed.

C. In-Loop-Quantization (ILQ)

An important difference from the standard process is the ILQ approach. This means that after every sparse iteration, a complete transform coding process is done, including quantization, CABAC encoding, and RDO calculation, based on formulas (3), (4) and (5), as depicted and explained in Fig. 2. An in-loop quantization has a deadlocking problem [27], which is prevented by comparing every newly found sparse coefficient with the previously found coefficient. If the coefficients are the same, this stops the loop, and returns the previous cost calculation finding.

D. Residual Classifier (RClass)

To save computation time, we create RClass which analyses the DCT residuals for LMFE signals. The reason is that the complexity, especially in the analysis phase, is very high due to a multitude of permutations being tested. RClass saves complexity by passing LMFE signals. The task is to check every candidate residual block to see if it is suitable for the proposed method. The classifier calculates the absolute sum, after quantization, of the DCT residuals; if the sum is in a specific range—above the lower and below the upper bound—then the block is allowed to be probed by the proposed method. The lower and upper bound are uniquely adapted for every $QP_{\{22,27,32,37\}}$ settings and every block size, based on a statistical analysis.

V. Experimental Results

A. HEVC/HEVC-SCC versus VVC

The proposed method is implemented in HEVC HM-16.18 [7] and HEVC HM-16.18+SCM-8.7 [8]. This research chooses HEVC instead of the VVC standard due to its wide deployment currently. Also, HEVC is less code and module complex, and the run time complexity of VVC is 7.4x to 34.0x as complex as HEVC [28, 29], which would be a significant time issue while testing all tests of the common test conditions with the available test hardware.

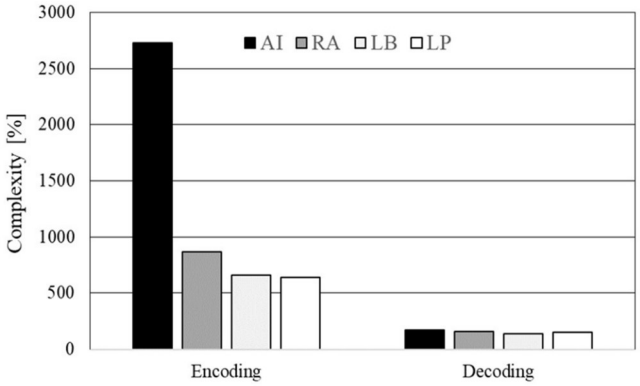

Fig. 3. Complexity of VVC compared to HEVC [30]

Additionally, as can be seen in Fig. 3, the execution time of the All Intra test scenario is even much higher, which would additionally significant increase the overall test time because the proposed method is more focused on the intra part.

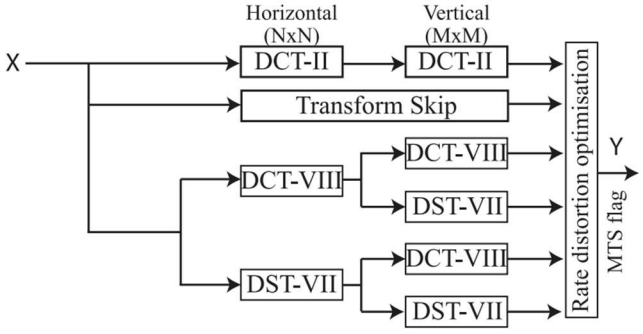

Fig. 4. The concept of 2D separable transforms selection in VVC. [31]

Fig. 4 shows the VVC transforms which has more options to choose from in contrast to HEVC which has only a small subset of these options. With more choice the VVC standard has an advantage over the HEVC standard, which would have an impact on the outcome of the proposed method results.

B. Max and Min Test Setups

We constructed a Maximum-BD rate setup (Max-BD) which consider maximum BD-rate savings, and a Minimum-Execution time setup (Min-Exe) for which the execution time was considered. Both setups were integrated on HEVC HM-16.18 [7] and HEVC HM-16.18+SCM-8.7 [8], which strictly follows the CTC [11, 12]. The Max-BD setup was used for intra and inter signals, for all $QP_{\{22,27,32,37\}}$ settings, all block sizes $\{4\times4, 8\times8, 16\times16, 32\times32\}$, and all test setups: AllIntra (AI), RandomAccess (RA), LowDelayB (LDB), and LowDelayP (LDP) which is only used in HEVC. The major difference between the two implementations, HEVC and HEVC-SCC, is that HEVC-SCC uses a variety of tools that are specialized for compressing screen content. The proposed method for HEVC-SCC was tested only with the 4:2:0 color scheme, which means that four of seven screen content tools were activated: TS, RDPCM, IBC, and PM. Some elements for the Min-Exe test setup, are excluded. Block size 4×4 was not activated, the reason being that block size 4×4 has the lowest execution time overhead at around 4%, and that it does not compress enough—usually around 0.1% BD-rate savings for HEVC-SCC, or 0.2% BD-rate savings for HEVC. The BD-rate savings for $QP_{\{22\}}$, was always very low because it is the high quality setup and as a consequence the number of sparse coefficients raised significantly, which was a problem because the sparse coefficient indices were coded in fixed length (CABAC bypass). As an outcome, the BD-rate savings for $QP_{\{22\}}$ is approximately close to around 0.1% to 0.3%. In general, while the number of sparse coefficients get higher the encoded efficiency for the proposed method goes down significantly. The number of sparse coefficients reflects linearly the quality of the outcome. As a rule, the higher the frequency (sharp edges) of a signal, the more sparse coefficients needed to reconstruct it. A signal can be perfectly reconstructed, even in a high frequency case, when the dimension of a signal is the same as the number of sparse coefficients. But in this case, the compression rate is worse than the standard method. The key is to be as sparse as possible. Therefore, $QP_{\{22\}}$ was deactivated for the Min-Exe setup. Nonetheless, block size 32×32 performed well, but the execution time expanded exponentially. So, for the Min-Exe, 32×32 was also deactivated.

Table III
Major differences between the two test sets

	Max-BD	**Min-Exe**
	High BD-rate savings	**Min execution time**
Block sizes	$\{4\times4, \ldots, 32\times32\}$	$\{8\times8, 16\times16\}$
***QP* settings**	$\{22, 27, 32, 37\}$	$\{27, 32, 37\}$
Residual Classifier	No	Yes
Chroma	No	No

Table III shows the major differences between the two test setups Max-BD and Min-Exe.

In general, we tried to follow the CTC [11, 12] strictly, and tried to test as much as possible. The only difference is that this research used the first 20% of a test sequence for training purpose and selected the remaining unseen 80% of the test sequence frames for testing. This tends to be closer to a realistic less biased test towards an online learning approach, which uses the first frames for training and improves with every frame the dictionary. This was done on the encoder as wells as on the decoder side in the same way.

This article has been accepted for publication in IEEE Transactions on Consumer Electronics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TCE.2023.3268061

Table IV
All HEVC and HEVC-SCC test results

Test Name	Luma BD-rate	Block Size	Intra		OMP Used %	Inter		OMP Used %	Avg Sparsity	Execution Time	
			OMPFlag1	OMPFlag0		OMPFlag1	OMPFlag0			Encoder	Decoder
SCC_AI4x32_Max-BD	-4.766 %	4x4	3,526,548	53,236,429	6.6%				1.19	1,021%	4,993%
		8x8	4,787,222	13,862,871	34.5%				1.52		
		16x16	2,457,848	5,171,209	47.5%				2.17		
		32x32	991,806	2,074,050	47.8%				2.58		
SCC_RA4x32_Max-BD	-3.234 %	4x4	116,601	3,515,124	3.3%	8,824	10,492,210	0.1%	1.22	1,390%	7,421%
		8x8	176,313	813,168	21.7%	110,389	1,037,164	10.6%	1.52		
		16x16	84,259	327,977	25.7%	37,333	477,844	7.8%	2.10		
		32x32	33,767	205,019	16.5%	13,536	310,693	4.4%	2.48		
SCC_LDB4x32_Max-BD	-2.578 %	4x4	30,544	1,265,479	2.4%	19,970	12,685,192	0.2%	1.11	1,381%	10,236%
		8x8	64,967	328,873	19.8%	164,964	1,703,677	9.7%	1.35		
		16x16	29,871	176,076	17.0%	60,302	922,787	6.5%	1.66		
		32x32	14,896	166,717	8.9%	26,759	584,217	4.6%	1.99		
SCC_AI8x16_Min-Exe	-2.501 %	8x8	3,869,633	9,068,174	42.7%				1.54	220%	244%
		16x16	2,337,254	4,177,609	55.9%				2.23		
SCC_RA8x16_Min-Exe	-1.534 %	8x8	145,617	506,463	28.8%	118,796	461,347	25.7%	1.47	270%	352%
		16x16	90,090	320,086	28.1%	52,022	469,793	11.1%	1.98		
SCC_LDB8x16_Min-Exe	-1.114 %	8x8	53,372	169,590	31.5%	170,513	788,590	21.6%	1.29	247%	371%
		16x16	25,982	110,144	23.6%	64,171	521,664	12.3%	1.61		
HEVC_AI4x32_Max-BD	-5.525 %	4x4	14,437,493	277,364,683	5.2%				1.10	3,276%	10,513%
		8x8	27,562,097	136,790,051	20.1%				1.34		
		16x16	15,970,839	56,120,009	28.5%				1.57		
		32x32	5,386,339	25,186,027	21.4%				1.98		
HEVC_RA4x32_Max-BD	-2.476 %	4x4	1,047,441	31,257,980	3.4%	68,529	14,978,562	0.5%	1.12	1,836%	10,592%
		8x8	2,985,636	16,274,568	18.3%	809,187	12,028,761	6.7%	1.30		
		16x16	2,223,144	8,826,824	25.2%	715,090	7,931,077	9.0%	1.49		
		32x32	652,800	4,639,018	14.1%	255,620	4,602,167	5.6%	1.78		
HEVC_LDB4x32_Max-BD	-4.620 %	4x4	796653	19632338	4.1%	309868	39458604	0.8%	1.12	1,805%	18,170%
		8x8	2657428	11678910	22.8%	2411411	24886063	9.7%	1.30		
		16x16	1945721	7184906	27.1%	1720925	15619769	11.0%	1.49		
		32x32	538643	3587575	15.0%	7494	662500	1.1%	1.78		
HEVC_LDP4x32_Max-BD	-4.641 %	4x4	873,646	22,563,381	3.9%	285,409	44,675,472	0.6%	1.12	2,696%	26,693%
		8x8	3,050,409	13,951,111	21.9%	2,142,065	26,163,711	8.2%	1.30		
		16x16	2,359,404	8,586,423	27.5%	1,504,331	14,947,719	10.1%	1.49		
		32x32	636,430	4,187,715	15.2%	532,050	7,900,674	6.7%	1.78		
HEVC_AI8x16_Min-Exe	-2.750 %	8x8	21,519,766	68,766,993	31.3%				1.26	486%	307%
		16x16	15,627,731	39,638,053	39.4%				1.51		
HEVC_RA8x16_Min-Exe	-0.961 %	8x8	1,939,316	6,813,799	28.5%	423,412	2,983,070	14.2%	0.78	253%	432%
		16x16	1,793,628	5,099,246	35.2%	490,959	3,120,249	15.7%	1.40		
HEVC_LDB8x16_Min-Exe	-1.808 %	8x8	1,883,832	4,790,684	39.3%	1,127,614	6,307,114	17.9%	1.17	296%	677%
		16x16	1,593,832	4,062,323	39.2%	1,232,370	6,841,544	18.0%	1.33		
HEVC_LDP8x16_Min-Exe	-1.789 %	8x8	2,046,142	5,397,039	37.9%	998,235	6,607,593	15.1%	1.16	406%	927%
		16x16	1,829,219	4,720,432	38.8%	1,053,868	6,622,729	15.9%	1.32		
Averages	-2.878 %		3,576,052	21,014,646	24.4%	564,534	9,226,419	9.4%	1.52	1,113%	6,566%

Table IV shows the major test results, for HEVC and HEVC-SCC, against the standard test runs using CTC [11, 12]. In general, the results are organized in the following way. The "Test Name" specifies the test setup and various details about it. First, the prefix HEVC stands for camera-captured content and "SCC" for screen content. Second, the kind of test—AI, RA, LDB, and LDP which is used only for HEVC. Third, the block sizes the proposed method is using—either ($\{4\times4, \ldots, 32\times32\}$) or ($\{8\times8, \ldots, 16\times16\}$). Fourth, test setup, which is either Max-DB or Min-Exe. The next column is the BD-rate savings. The next columns give more information about the details for each block size: $\{4\times4, 8\times8, 16\times16, 32\times32\}$, which reveals the numbers of intra and inter blocks compressed by the proposed method

for every block size and test execution. It also shows the percentage of the blocks used by the proposed method, and the average sparsity per block size and test setup.

Remarkable for Table IV is that, in general, all AI test executions have the most blocks overall. The reason is that the inter part has in many cases only a limited number of blocks because they are encoded in a different way. The low number/percentage of proposed coded blocks for the 4 × 4 block size reflects that 4 × 4 only minimally contributes to the outcome. This is because the training data for this block size is much bigger, and the KSVD algorithm cannot converge/generalize to create an efficient semi-extreme sparse dictionary. AI always performs best because there are a lot of blocks to probe and encode with the proposed method. RA performs well because it has more intra frames regularly inserted into the bitstream, and LDB performs well too. Significant is that the average proposed method block usage is 24.4% for intra and 9.4% for inter. MUB4 for all block sizes and $QP_{\{22,27,32,37\}}$ settings are four and, on average, the proposed method uses only 1.5 sparse atoms per block, which is very low and reflects that this is close to the extreme sparse representation explained in [6], which represents our semi-extreme sparse coding proposed method. This also shows that no high-frequency signals are encoded by the proposed method. In general, bigger block sizes lead to bigger BD-rate savings. The difference between the percentage of blocks encoded by the proposed method for intra which is 24.4% and for inter which is 9.4%, is remarkable. The reason for the difference is on one hand, that our convenient approach to simply use the intra trained dictionaries also for encoding inter blocks is not sufficient enough. On the other hand, intra residual data has more energy than inter residual training data. As a consequence, the inter training process needs to be optimized further to increase the proposed method selection to a higher percentage. The execution time for the Max-BD setup is high and extends the original execution time by a factor of 11 to 33. $QP_{\{22\}}$ does not perform well, because it is the highest quality, which means that the lowest distortion is needed. However, if more sparse coefficients are needed, it creates a problem because every sparse coefficient index is encoded in fixed length (CABAC bypassed), which creates too much overhead. And additionally there is a max sparsity limitation of four (MUB4). This is the reason the proposed method is not used much, and [5, 19] make similar observations. In general, $QP_{\{32\}}$ performs the best. The reason that $QP_{\{37\}}$ does not perform that well, in comparison to $QP_{\{32\}}$, is that for $QP_{\{37\}}$, the limiting factor is not the distortion, but the bits for encoding. Not many bits are allowed.

Table V
Detailed BD-rate savings of "SCC_AI4x32_Max-BD" test. Broken down by separate test sequences compared to HM-16.18+SCM-8.7
Compression Ratio (CR)

Video Sequence	Category	Resolution	CR 1:	BD-rate savings[1] luma %
BasketballScreen	MixedContent	2560 × 1440	1.09	-15.5%
MissionControl2	MixedContent	2560 × 1440	1.02	-4.3%
MissionControl3	MixedContent	1920 × 1080	1.04	-8.0%
FlyingGraphics	TGM1080	1920 × 1080	1.00	-0.1%
Desktop	TGM1080	1920 × 1080	1.02	-2.6%
Console	TGM1080	1920 × 1080	1.00	0.0%
ChineseEditing	TGM1080	1920 × 1080	1.03	-3.7%
WebBrowsing	TGM720	1280 × 720	1.05	-6.0%
Map	TGM720	1280 × 720	1.06	-8.3%
Programming	TGM720	1280 × 720	1.02	-3.4%
SlideShow	TGM720	1280 × 720	1.00	-0.6%
Robot	Animation	1280 × 720	1.00	-0.3%
ChinaSpeed	Animation	1024 × 768	1.05	-9.2%

Table V shows the detailed results for the test execution of Table IV test "SCC_AI4x32_Max-BD". The percentage relation is similar through all other test executions. The outliers: FlyingGraphics, Console, SlideShow, and Robot show 0% and up to 0.6% BD-rate savings. One positive outlier is BasketballScreen with 15.5% BD-rate saving, which is remarkable. In the case of SlideShow, it is simply that the test sequence is dominated by big, white one-color areas where the proposed method cannot compete against DCT + DZQ + CABAC. The console test sequence is a different example which contains a lot of text and simple graphic elements and is thus not frequently selected by the proposed method a lot. A general reason for these outliers is mostly that the category boundaries based on the CTC [12] categories are not performing well for all test sequences.

Table VI
Detailed BD-rate savings of "HEVC_AI4x32_Max-BD" test. Broken down by separate test sequences compared to HM-16.18+SCM-8.7
Compression Ratio (CR)

Video Sequence	Category Class	Resolution	CR 1:	BD-rate savings[1] luma %
Tango	A1	4096 × 2160	0.99	-0.6%
Drums100	A1	3840 × 2160	1.00	-0.3%
CampfireParty	A1	3840 × 2160	1.00	-0.7%
ToddlerFountain	A1	4096 × 2160	1.00	-0.3%
CatRobot	A2	3840 × 2160	1.00	0.0%
TrafficFlow	A2	3840 × 2160	1.05	-0.0%
DaylightRoad	A2	3840 × 2160	1.00	-0.3%
Rollercoaster	A2	4096 × 2160	1.00	-1.0%
Kimono	B	1920 × 1080	1.00	-0.3%
ParkScene	B	1920 × 1080	1.00	-0.2%
Cactus	B	1920 × 1080	1.08	-14.3%
BasketballDrive	B	1920 × 1080	1.00	-0.6%
BQTerrace	B	1920 × 1080	1.00	-1.0%
FourPeople	E	1280 × 720	1.15	-23.4%
Johnny	E	1280 × 720	1.09	-15.1%
KirstenAndSara	E	1280 × 720	1.10	-15.0%
BasketballDrill	C	832 × 480	1.14	-30.4%
BQMall	C	832 × 480	1.00	-0.3%
PartyScene	C	832 × 480	1.00	-0.1%
RaceHorses	C	832 × 480	1.00	-0.2%
BasketballPass	D	416 × 240	1.00	-0.3%
BQSquare	D	416 × 240	1.00	-0.3%
BlowingBubbles	D	416 × 240	1.00	-0.1%
RaceHorses	D	416 × 240	1.00	-0.1%
BasketballDrillText	F	832 × 480	1.13	-27.3%
ChinaSpeed	F	1024 × 768	1.07	-10.6%
SlideEditing	F	1280 × 720	1.10	-11.4%
SlideShow	F	1280 × 720	1.00	-0.8%

Table VI shows the detailed results for each test sequence and category of Table IV test "HEVC_AI4x32_Max-BD" execution. For camera-captured content, 8 of 24 test sequences are outliers in regard to the

upper bound, and the rest of the test sequences are between 0.0% and 1.0% BD-rate savings. This is in part a problem of the category composition for HEVC in CTC [11], which is based on the resolution and not on the similarity of the content as it is in HEVC-SCC. ClassF is a good example that performs very well with almost every test sequence except SlideShow. The reason SlideShow does not perform well is already explained, because most of the content is simply white background. It can be assumed, if the category selection is based on the content and not on the resolution, that the HEVC test sequences would perform in a more balanced manner with similar overall BD-rate savings as for HEVC-SCC.

For the RDO curves we used the Bjøntegaard model [32] where we transformed the PSNR into a logarithmic scale (dB).

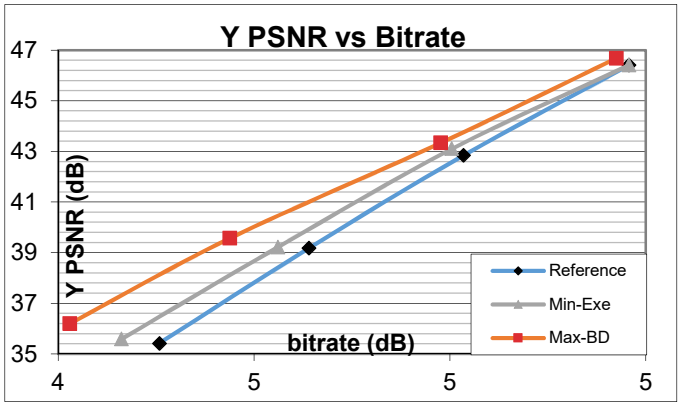

Fig. 5. HEVC-SCC: AI - MixedContent – BasketballScreen test sequence RDO curve for Min-Exe and Max-BD to reference run

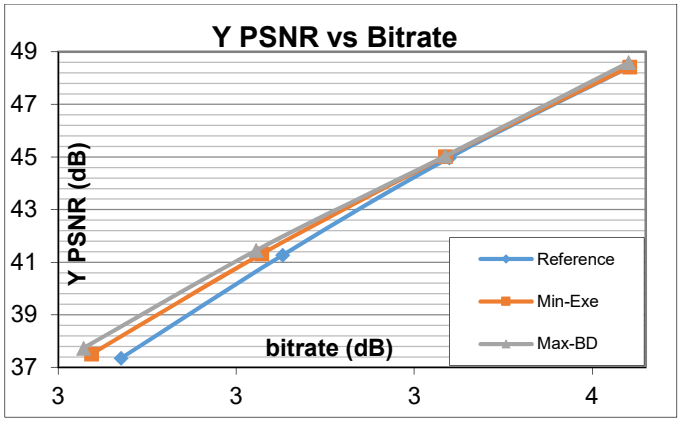

Fig. 6. HEVC-SCC: RA - MixedContent - MissionControlClip3 test sequence RDO curve for Min-Exe and Max-BD to reference run.

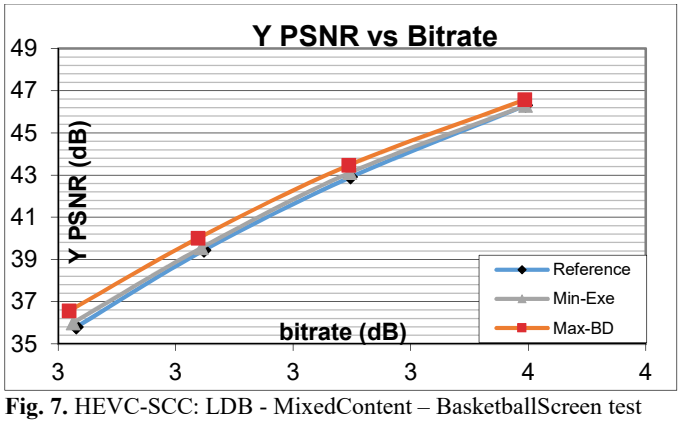

Fig. 7. HEVC-SCC: LDB - MixedContent – BasketballScreen test sequence RDO curve for Min-Exe and Max-BD to reference run

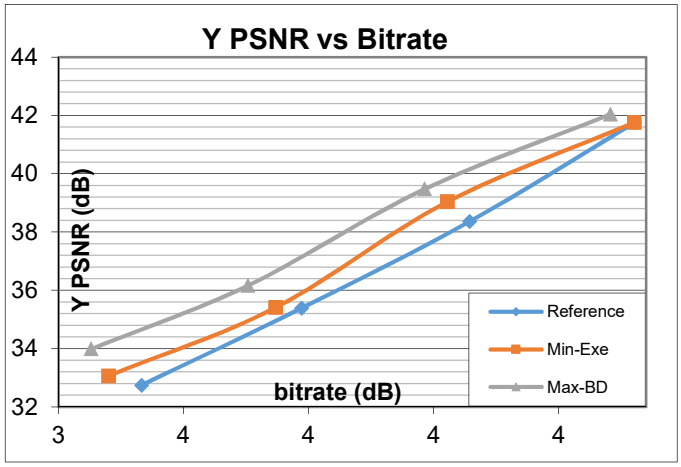

Fig. 8. HEVC: AI - ClassC - BasketballDrill test sequence RDO curve for Min-Exe and Max-BD to reference run.

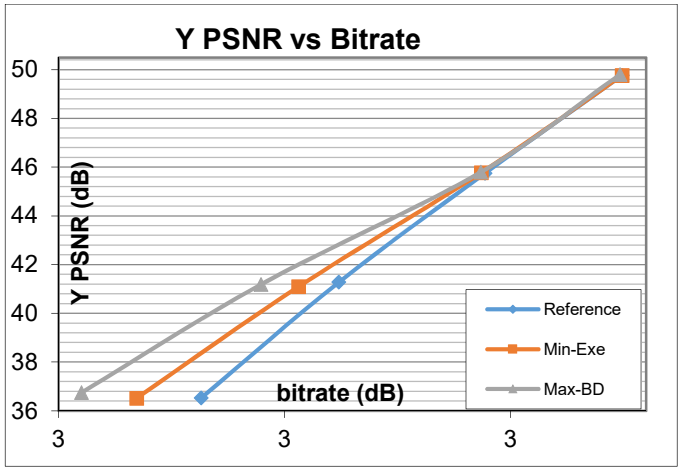

Fig. 9. HEVC: RA – ClassF - SlideEditing test sequence RDO curve for Min-Exe and Max-BD to reference run.

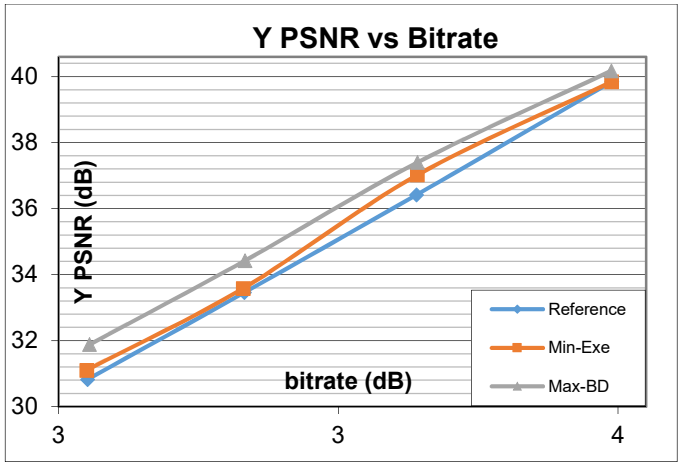

Fig. 10. HEVC: LDP – ClassF – BasketballDrillText test sequence RDO curve for Min-Exe and Max-BD to reference run.

By comparing the curves in Fig. 5 to Fig. 9, a general tendency is that as mentioned before the compression at higher PSNR values gets closer to zero, which is an outcome of the demand for more sparse atoms, so that the proposed method is not used overall.

C. *Contribution of the proposed method parts*

Here we provide a quick overview of which parts are contributing how much to BD-rate saving terms. These tests were conducted with the HEVC-SCC AI test setup.

Table VII
Different parts of the proposed method and their impact on BD-rate savings tested with HEVC-SCC AI Max-BD setup

	Test	BD-rate savings	Execution Time
1	**Dic K-Size 512**	-1.893%	842%
2	**Dic K-Size 1,024**	-3.151%	887%
3	**Dic K-Size 2,048**	**-4.765%**	**1,021%**
4	**Block Size 4x4**	-0.050%	103%
5	**Block Size 8x8**	-0.802%	141%
6	**Block Size 16x16**	-2.070%	257%
7	**Block Size 32x32**	-1.567%	494%
9	**λ -5% (LMA)**	-4.658%	1,000%
10	**λ 0% (LMA)**	-4.747%	1,037%
11	**λ +5% (LMA)**	**-4.765%**	**1,021%**
12	**λ +10% (LMA)**	-4.757%	1,086%
13	**QPD off**	-2.641%	1,036%
14	**Chroma On**	Around -1% for chroma parts	1,103%
15	**PON (palette mode on)**	-3.641%	1,149%
16	**TON (transform skip TS on)**	-5.048%	1,738%
17	**TRON (TS & residual rotation on)**	-4.540%	1,707%
18	**ION (IBC on)**	-6.009%	1,130%
19	**All SCC tools off**	-11.764%	1,853%

Table VII shows BD-rate and complexity numbers for various test setups. Tested with the Max-BD test setup on HEVC-SCC AI, Tests 1-3 show the impact of the K-size of a dictionary. The complexity increases for higher K-sizes, but at the same time the BD-rate savings goes up significantly. Tests 4-7 show the BD-rate and complexity for the block sizes separately. A 4×4 block size contributes only 0.05% BD-rate savings and exceeds the execution time by 3%, which was the reason it was deactivated in our Min-Exe setup. On the other end, a 32×32 block size performs well in terms of BD-rate savings, but the execution time is too high, which is the reason it was deactivated for the Min-Exe test setup. Tests 9-12 reveal the impact of various λ settings on the BD-rate savings and encoder execution time. If LMA goes negative, the BD-rate savings are worse because distortion is emphasized over sparsity, but the proposed method follows the opposite configuration with SoD. If LMA goes positive, the results improve, and a +5% λ adjustment created the best results. Test 13 shows the BD-rate savings results of the QPD part, which performs very well and improves the BD-rate by around 2.1%. Test 14 shows the impact of the chroma part. Tests 15-19 show the impact of the HEVC-SCC tools on the BD-rate savings and execution time. Interestingly, in the "All tools off" setup (screen content tools off), the BD-rate savings is more than double compared to the "All tools on" setup test 3. This shows that the internal screen content tools of HEVC-SCC perform well in removing a lot of content that would otherwise be a candidate for our proposed method.

D. *Crop Compare*

The visual quality is not an issue and tends to be in general, consistent with the PSNR values.

Fig. 11. ChinaSpeed original yuv test sequence 1024x768 $QP_{\{32\}}$, *AllIntra, screenshot*

Fig. 12. ChinaSpeed original yuv test sequence 1024x768 $QP_{\{32\}}$, AllIntra, snapshot enlargement

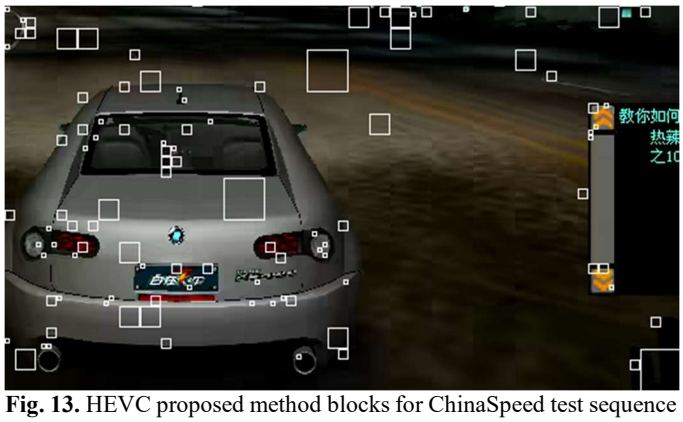

Fig. 13. HEVC proposed method blocks for ChinaSpeed test sequence 1024×768, $QP_{\{32\}}$, AllIntra, snapshot enlargement

By comparing Fig. 12 which is an enlarged snapshot of Fig. 11 With Fig. 13 it is not easy to detect any PSNR degradation at all.

Table VIII
HEVC BD-rate and execution time Comparison

Paper/year	Block size	Test Setup	Test Classes	BD-rate	Execution Time
[13] **2016 "Orthogonal-Matching-Pursuit Based Residual Coding with Content Adaptive Dictionary in HEVC"**	8×8 to 32×32	LDP	C BasketballDrill 480p RaceHorses 480p D BlowingBubbles 240p BQSquare 240p RaceHorses 240p	-2.3%	143%
This paper (Approximation)	4×4 to 32×32			-4.2%	3,613%
[14] **2016 "OMP-based transform for inter coding in HEVC"**	4×4 to 32×32	RA	A, B, C, D, F	-1.0%	Not Available
		LDB	B, C, D, E, F	-0.8%	
		LDP		-3.3%	
This paper (Table IV)		RA	A, B, C, D, F	-2.5%	1,836%
		LDB	B, C, D, E, F	-4.6%	1,805%
		LDP	B, C, D, E, F	-4.6%	2,696%

Table VIII lists the BD-rate and execution time for related sparse video compression research for comparison with Table IV. For execution time, 100% means that the execution time is the same as the reference run without the proposed method. Overall, there are around 11, more or less, sparse coding papers, but only three papers [5, 13, 14] compare with HEVC and only ours [20] compares with HEVC-SCC. The rest of the past research used H.264 to H.263 and other codecs. Paper [5] uses sparse coding and DCT in a sequence. They tested B, C, and D, but the method is too different to be listed for an unbiased comparison in Table VIII. In general, paper [13] tested only a small subset of the CTC [11] (classes C and D), but this research tested all the CTC [11] test sequences, so it was possible to adapt the test results from Table IV to the test setup of paper [13] for an unbiased comparison. Also different is the block size of 8×8 to 32×32 and our block sizes includes everything from 4×4 to 32×32, but as mentioned before, the contribution of the 4×4 block size for the proposed method is very low, so that the difference to [13] can be ignored. Paper [14] on the other hand, follows like our research, strictly with the CTC [11]. The only difference between the results between [13], [14] and this research are the number of frames tested. [13] tested all frames, [14] tested only first 11 frames, and our research takes the first 20% of the frames for training and uses the remaining unseen 80% frames for the test. Execution time comparison is a general problem, some research did not mention it, like [14, 17, 19], and [22], **or** mention it in an unusual way like [15] as "Complexity (pel/s)" and "Runtime (s/frame)" or in [16] with millions of operations per frame, which goes from a ratio of 3.94 to 7.40. We consider [13] as an efficient implementation of sparse coding, it has a 43% increase of execution time. With more research into optimization techniques like vectorization, plus using the latest and most efficient sparse coding solution, and not using a generic OMP MATLAB translation into C++, plus a more optimized residual classifier (RClass), and additionally, if the 16x16 and 32x32 blocks are all split up in 8x8 blocks to reduce the exponential increase of execution time for 16x16 and 32x32 blocks, the complexity increase should be around the same level or even lower. The BD-rate test results are promising, compared to [13] we improved the BD-rate by 1.9%, but our execution time was almost 35 times slower. For [14] we improved it for RA by 1.5%, for LDB by 3.8%, and for LDP by 1.3%. The execution time for this case, could not be compared because [14] did not mention it. A surprising issue with the comparison of this work with others is also that sparse coding in video compression is in many cases were done only for the inter part. To compare our results with intra compression it is needed to compare it with sparse coding for image compression which is in principle intra. But, on the other hand, it would be a biased comparison because of the missing CABAC compression option, so we skip it here.

Table IX
All Intra, RandomAccess, LowDelayB and LowDelayP

Test Scenario	Details
AllIntra (AI)	- Each picture is an I picture - Good for low delay - Higher bit rate applications - QP is constant
RandomAccess (RA) -Streaming -Broadcast -Blue-ray -DVD	-Hierarchical B structure is used -Coding efficiency is highest -Larger delay (reordering) -For errors control, - Every second an I pic
LowDelayB (LDB) -Medium comp. -Video Conference	-First pic is an I pic -All others are encoded as P pics -Reordering of pics not allowed -only past pics prediction -Low coding delay
LowDelayP (LDP) -HEVC only -Medium comp. -Video Conferencing	-First pic is an I pic -All others are encoded as B pics. -Reordering of pics not allowed -Low coding delay -Higher coding efficiency as LDP

E. HEVC & HEVC-SCC: ChinaSpeed test sequence and proposed method comparison

HEVC is tested by CTC [11], and HEVC-SCC is tested by the CTC [12]. Both CTC [11, 12] test-sequence selection are different except for two, which are for HEVC-SCC and also for HEVC (ClassF optional) ChinaSpeed and SlideShow.

Table X
Comparison of common test sequences ChinaSpeed and SlideShow between HEVC and HEVC-SCC

Test sequence	BD-Rate HEVC	BD-Rate HEVC-SCC
ChinaSpeed	-10.5%	-9.4%
SlideShow	-0.7%	-0.9%

Table X shows the differences in BD-rate savings between the two test sequences that are common to both CTC [11, 12]. An interesting example is the ChinaSpeed test sequence for comparing the decoded YUV output between HEVC and HEVC-SCC. This comparison

reveals in general where the proposed method compresses well, which part is not compressed, and whether or not there is a significant difference between the two codecs. ChinaSpeed is overall very well compressed by the proposed method, for both codec HEVC by -10.5% and HEVC-SCC by -9.4%.

Fig. 14. HEVC proposed method blocks for ChinaSpeed test sequence 1024x768, $QP_{\{32\}}$, AI.

Fig. 14 shows the ChinaSpeed test sequence, handled by the HEVC codec, without screen content coding tools. As it can be seen, almost all Chinese characters are not encoded by the proposed method.

Fig. 15. HEVC-SCC proposed method blocks for ChinaSpeed test sequence 1024x768, $QP_{\{32\}}$, AI.

Fig. 15 shows the same ChinaSpeed test sequence like Fig. 14 but with the HEVC-SCC codec compression. As expected, despite the fact that two different sets of dictionaries were used, the outcome is similar. The BD-Rate savings for both are close, with a difference of 1.1%. The compression of HEVC-SCC is less because the HEVC-SCC tools already compress a significant amount of screen content blocks on their own, and these are subsequently not compressed by the proposed method. In general, for the ChinaSpeed test sequence for both codecs, the proposed method is used more in some areas of the picture. Overall, the proposed method is used for simple planar blocks, and blocks with straight lines in any angles, but not for simple, single-color blocks with no change at all.

F. HEVC & HEVC-SCC: SlideShow test sequence and proposed method comparison

SlideShow is the other test sequence that both test conditions share and exists on the other end of the spectrum that is poorly compressed by the proposed method.

Fig. 16. HEVC proposed method blocks for SlideShow test sequence 1280x720, $QP_{\{32\}}$, AI.

Fig. 16 shows the SlideShow test sequence, where the proposed encoded blocks are highlighted with black boxes. In comparison to ChinaSpeed, it is obvious that SlideShow does not perform well with the proposed method. A major reason is that SlideShow contains numerous white color background blocks, which are efficiently compressed by DCT + DZQ + CABAC. In these cases, the proposed method cannot compete against the standard process.

Fig. 17. HEVC-SCC proposed method blocks for SlideShow test sequence 1280x720, $QP_{\{32\}}$, AI.

Fig. 17 shows the same picture/frame as shown in Fig. 16, but encoded with HEVC-SCC instead of HEVC. The two BD-rate savings are close: with HEVC at -0.7% and HEVC-SCC at -0.9%. The test setup between the two codecs was similar, and both used their own trained dictionary sets. The differences are that the proposed method for HEVC compresses more of the middle graphics, an area that is compressed less by HEVC-SCC. Additionally, the grass is more compressed in HEVC than in HEVC-SCC.

G. *Dictionary visualization HEVC*

We visualized the dictionaries to show the common structures, to reveal any surprising patterns, to get an idea of the direction in which the whole training process goes, and to demonstrate what the general characteristic of a trained dictionary is—as well as to determine what the KSVD process considered important, and what it considered unimportant.

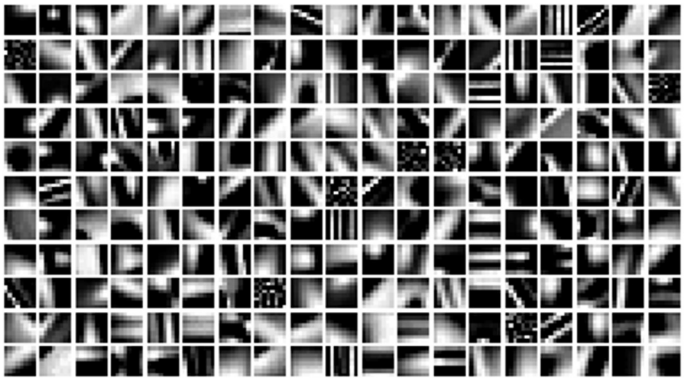

Fig. 18. ClassA2 (4k) 8×8 $\mathrm{QP}_{\{34\}}$ subset.

Fig. 18 shows the ClassA2 (4k) dictionary subset. It features fluffy diagonal blocks, including horizontal blocks. This reflects that the training data process have a significant impact on the dictionary training process.

Fig. 19. ClassD 32×32 $QP_{\{34\}}$ subset.

Fig. 19 shows a dictionary subset, which contrasts with many other dictionaries because it looks almost like noise, waves, or shadows.

H. *Dictionary visualization HEVC-SCC*

Fig. 20. Visual dictionary TGM1080 8×8 $QP_{\{34\}}$ subset.

Fig. 20 shows a typical subset of a TGM1080 dictionary, where the blocks themselves look more detailed. It also shows more lined dictionary atoms, in this case a majority are straight horizontal and vertical, and some are diagonal.

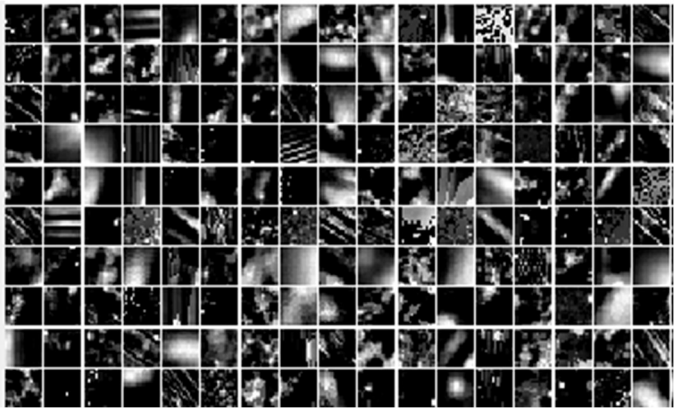

Fig. 21. Visual dictionary MixedContent 16×16 $QP_{\{22\}}$ subset.

Fig. 21 shows a dictionary, which is special because this subset features content parts from MissionControl2 test sequence, where the background is star-like, as can be seen in this dictionary subset. Overall, as expected, every category is developed in the training process, with characteristics adjusted to the content used for the process.

VI. CONCLUSION

We adapted, with encouraging results, a semi-extreme sparse coding solution to HEVC and HEVC-SCC for LMFE prediction residuals signals. The BD-rate savings for the HEVC intra setup can reach up to 5.5%, with enough BD-rate savings depending on [33] to be considered for the standardization process, but the execution time is a problem. This BD-rate savings shows that the standard HEVC inter/intra prediction and transform coding process is not perfect and has flaws and limitations. A significant problem with using sparse coding is the increase in complexity. Our execution time needs to be taken with caution. As mentioned before based on paper [13] the execution time for the encoder can be around the same increase of 43%, by following basic optimization techniques. The execution time for the decoder should be slightly better than for DCT because sparse coding uses only a linear combination to reconstruct the signal. Additionally, the overall average sparsity for HEVC and HEVC-SCC is around 1.5 atoms. On the other hand, DCT + DZQ are in principle matrix based, and have a higher complexity than a linear combination. The mentioned low overall average number of sparse coefficients means that most blocks are compressed by less than two sparse coefficients, which is close to the extreme sparse representation mentioned in [6], which would be the case, if only one sparse coefficient is used with a coefficient value of 1. We consider an average sparsity of less than two sparse coefficients as semi-extreme. The proposed category approach is only a temporary simulation/solution and a placeholder for an online learning approach, but it still demonstrates that there will be not a single universal dictionary that can handle all contents and applications. The reason is: to have a dictionary that is suitable for every content, it needs to be learned by every content, and if it is learned by every content, then it must be more generic and thus needs a significant amount of basic building blocks to include the large number of different details. Therefore, the sparse coding finding process requires on average more coefficients to construct a specific signal.

REFERENCES

[1] G. Davis, S. Mallat, and M. Avellaneda, "Adaptive greedy approximations," J. Construct. Approx., vol. 13, pp. 57–98, 1997.

[2] S. G. Mallat and Zhifeng Zhang, "Matching pursuits with time-frequency dictionaries," in IEEE Transactions on Signal Processing, vol. 41, no. 12, pp. 3397-3415, Dec. 1993, doi: 10.1109/78.258082.

[3] S. S. Chen, D. L. Donoho and M. A. Saunders, "Atomic Decomposition by Basis Pursuit", Stanford, 1995, Technical Report-Statistics.

[4] Y. C. Pati, R. Rezaiifar and P. S. Krishnaprasad, "Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition," Proceedings of 27th Asilomar Conference on Signals, Systems and Computers, 1993, pp. 40-44 vol.1, doi: 10.1109/ACSSC.1993.342465.

[5] J. Kang, M. Gabbouj and C. -. J. Kuo, "Sparse/DCT (S/DCT) Two-Layered Representation of Prediction Residuals for Video Coding," in IEEE Transactions on Image Processing, vol. 22, no. 7, pp. 2711-2722, July 2013, doi: 10.1109/TIP.2013.2256917.

[6] M. Aharon, M. Elad and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," in IEEE Transactions on Signal Processing, vol. 54, no. 11, pp. 4311-4322, Nov. 2006, doi: 10.1109/TSP.2006.881199.

[7] HEVC Reference Software HM-16.18 (2016 February) [Online]. Available: https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-16.8/

[8] HEVC Reference Software HM-16.18+SCM-8.7 (2018 January) [Online]. Available: https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-16.18+SCM-8.7

[9] J. Xu, R. Joshi and R. A. Cohen, "Overview of the Emerging HEVC Screen Content Coding Extension," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 26, no. 1, pp. 50-62, Jan. 2016, doi: 10.1109/TCSVT.2015.2478706.

[10] L. Limin and T. Alexis, "Rate distortion optimized quantization in the JM reference software." JVT-AA027, 2008.

[11] Karsten Suehring, "JVET common test conditions and software reference configurations", JVET-B1010, 2016.

[12] Haoping Yu, "Common test conditions for screen content coding", JCTVC-Z1015, 2017.

[13] Z. T. Zhang and C. Yeh, "Orthogonal-Matching-Pursuit Based Residual Coding with Content Adaptive Dictionary in HEVC," 2016 International Computer Symposium (ICS), 2016, pp. 355-358, doi: 10.1109/ICS.2016.0078.

[14] R. Song, C. Lan, H. Li, J. Xu and F. Wu, "OMP-based transform for inter coding in HEVC," 2016 IEEE International Symposium on Circuits and Systems (ISCAS), 2016, pp. 798-801, doi: 10.1109/ISCAS.2016.7527361.

[15] Y. Xue and Y. Wang, "Video coding using a self-adaptive redundant dictionary consisting of spatial and temporal prediction candidates," 2014 IEEE International Conference on Multimedia and Expo (ICME), 2014, pp. 1-6, doi: 10.1109/ICME.2014.6890314.

[16] R. Neff and A. Zakhor, "Very low bit-rate video coding based on matching pursuits," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 7, no. 1, pp. 158-171, Feb. 1997, doi: 10.1109/76.554427.

[17] R. Neff and A. Zakhor, "Matching pursuit video coding at very low bit rates," Proceedings DCC '95 Data Compression Conference, 1995, pp. 411-420, doi: 10.1109/DCC.1995.515531.

[18] M. Vetterli and T. Kalker, "Matching pursuit for compression and application to motion compensated video coding," Proceedings of 1st International Conference on Image Processing, 1994, pp. 725-729 vol.1, doi: 10.1109/ICIP.1994.413410.

[19] J. Kang, C. -. J. Kuo, R. Cohen and A. Vetro, "Efficient dictionary based video coding with reduced side information," 2011 IEEE International Symposium of Circuits and Systems (ISCAS), 2011, pp. 109-112, doi: 10.1109/ISCAS.2011.5937513.

[20] M. G. Schimpf, N. Ling, Y. Shi and Y. Liu, "Sparse Coding of Intra Prediction Residuals for Screen Content Coding," 2021 IEEE International Conference on Consumer Electronics (ICCE), 2021, pp. 1-6, doi: 10.1109/ICCE50685.2021.9427722.

[21] F. Bergeaud and S. Mallat, "Matching pursuit of images," Proceedings., International Conference on Image Processing, 1995, pp. 53-56 vol.1, doi: 10.1109/ICIP.1995.529037.

[22] O. K. Al-Shaykh, E. Miloslavsky, T. Nomura, R. Neff and A. Zakhor, "Video compression using matching pursuits," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 9, no. 1, pp. 123-143, Feb. 1999, doi: 10.1109/76.744280.

[23] H. Xiong, Z. Pan, X. Ye and C. W. Chen, "Sparse Spatio-Temporal Representation With Adaptive Regularized Dictionary Learning for Low Bit-Rate Video Coding," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 23, no. 4, pp. 710-728, April 2013, doi: 10.1109/TCSVT.2012.2221271.

[24] J. Yang, J. Wright, T. S. Huang and Y. Ma, "Image Super-Resolution Via Sparse Representation," in IEEE Transactions on Image Processing, vol. 19, no. 11, pp. 2861-2873, Nov. 2010, doi: 10.1109/TIP.2010.2050625.

[25] F. Zhou, W. Yang and Q. Liao, "Single image super-resolution using incoherent sub-dictionaries learning," in IEEE Transactions on Consumer Electronics, vol. 58, no. 3, pp. 891-897, August 2012, doi: 10.1109/TCE.2012.6311333.

[26] J. Sole, R. Joshi, W. -. Chien and M. Karczewicz, "Transform coefficient coding in HEVC," 2012 Picture Coding Symposium, 2012, pp. 461-464, doi: 10.1109/PCS.2012.6213254.

[27] M. Kalluri, M. Jiang, N. Ling, J. Zheng and P. Zhang, "Adaptive RD Optimal Sparse Coding With Quantization for Image Compression," in IEEE Transactions on Multimedia, vol. 21, no. 1, pp. 39-50, Jan. 2019, doi: 10.1109/TMM.2018.2847228.

[28] M. Viitanen, J. Sainio, A. Mercat, A. Lemmetti and J. Vanne, "From HEVC to VVC: The First Development Steps of a Practical Intra Video Encoder," in IEEE Transactions on Consumer Electronics, vol. 68, no. 2, pp. 139-148, May 2022, doi: 10.1109/TCE.2022.3146016.

[29] W. Hamidouche, F. Pescador, T. Biatek and E. François, "Editorial Real-Time Implementation of VVC Standard for Consumer Electronic Devices," in IEEE Transactions on Consumer Electronics, vol. 68, no. 2, pp. 93-95, May 2022, doi: 10.1109/TCE.2022.3176714.

[30] K. Choi, The Van Le, Y. Choi and J. Y. Lee, "Low-Complexity Intra Coding in Versatile Video Coding," in IEEE Transactions on Consumer Electronics, vol. 68, no. 2, pp. 119-126, May 2022, doi: 10.1109/TCE.2022.3145397.

[31] I. Farhat, W. Hamidouche, A. Grill, D. Ménard and O. Déforges, "Lightweight Hardware Transform Design for the Versatile Video Coding 4K ASIC Decoders," in IEEE Transactions on Consumer Electronics, vol. 67, no. 4, pp. 329-340, Nov. 2021, doi: 10.1109/TCE.2021.3126549.

[32] G. Bjøntegaard, Calculation of Average PSNR Differences Between RD-Curves, document VCEG-M33, ITU-T SG 16/Q6, 13th VCEG Meeting, Austin, TX, USA, Apr. 2001

[33] Nam Ling, C.-C. Jay Kuo, Gary J. Sullivan, Dong Xu, Shan Liu, Hsueh-Ming Hang, Wen-Hsiao Peng, and Jiaying Liu, "The Future of Video Coding," APSIPA Transactions on Signal and Information Processing, Vol. 11, Issue 1, pp. 1-29, Jun 2022.

Michael G. Schimpf (S'14) received the B.Sc. degree in information technology with his bachelor thesis: "Development of a server component for multimedia and safe Internet payment at Questico AG" from Beuth University of Applied Sciences Berlin, Berlin, Germany, in 2002, and the M.A. degree in data- and information-management with his master thesis: "Developing a Java EE5 [Platform] – Transformation Components for the Purpose of Exchanging/Interchanging Business Process Models based on Inter-Media Format EPMAL on the Example of Microsoft Visio" from the University of Hamburg, Hamburg, Germany, in 2006. From 2002 to 2006 he worked as Software Developer for the Mecom GmbH, Hamburg, Germany. From 2006 to 2012 he worked as Senior IT consultant for multiple customers as Software Developer, Business Analyst, Application Manager and Technical Project Manager for the Prodyna AG, Frankfurt, Germany. He moved from Germany to Santa Clara to begin his PhD degree at Santa Clara University with the Department of Computer Science and Engineering, Santa Clara University. Michael is working as a peer advisor and teaching assistant for various courses for Santa Clara University, Santa Clara, CA, USA. He is IEEE student member since 2014. Author of the conference paper "Sparse Coding of Intra Prediction Residuals for Screen Content Coding," 2021 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 2021. And coauthor of the patent: "System and Method for Coding Intra Prediction Mode using a Second Set of Most Probable Modes". His research interest is video compression, screen content, machine learning, spare coding.

Ying Liu (S'11-M'13) received the B.S. degree in communications engineering from Beijing University of Posts and Telecommunications, Beijing, China, in 2006, the M.S. and Ph.D. degrees in Electrical Engineering from The State University of New York at Buffalo, NY, USA, in 2008 and 2012, respectively. She is currently an Assistant Professor in the Department of Computer Science and Engineering at Santa Clara University, Santa Clara, CA, USA. She serves as an Associate Editor for the IEEE Transactions on Circuits and Systems for Video Technology. Her main research interests are in image and video processing, deep learning, and computer vision.

Nam Ling (S'88-M'90-SM'99-F'08-LF'22) received the B.Eng. degree from the National University of Singapore, Singapore, in 1981, and the M.S. and Ph.D. degrees from the University of Louisiana at Lafayette, Lafayette, LA, USA, in 1985 and 1989, respectively. From 2002 to 2010, he was an Associate Dean with the School of Engineering, Santa Clara University, Santa Clara, CA, USA. He was the Sanfilippo Family Chair Professor, and is currently the Wilmot J. Nicholson Family Chair Professor and the Chair with the Department of Computer Science and Engineering, Santa Clara University. He is/was also a Consulting Professor with the National University of Singapore, a Guest Professor with Tianjin University, Tianjin, China, a Guest Professor with Shanghai Jiao Tong University, Shanghai, China, a Cuiying Chair Professor with Lanzhou University, Lanzhou, China, a Chair Professor and Minjiang Scholar with Fuzhou University, Fuzhou, China, and a Distinguished Professor with the Xi'an University of Posts and Telecommunications, Xi'an, China. He has authored or coauthored over 250 publications and seven adopted standard contributions. He has been granted 20 U.S. patents so far. He is an IEEE Fellow due to his contributions to video coding algorithms and architectures. He is also an IET Fellow. He was named as an IEEE Distinguished Lecturer twice and also an APSIPA Distinguished Lecturer. He was a recipient of the IEEE ICCE Best Paper Award (First Place) and the Umedia Best/Excellent Paper Award three times. He received six awards from Santa Clara University, four at the University level (Outstanding Achievement, Recent Achievement in Scholarship, President's Recognition, and Sustained Excellence in Scholarship), and two at the School/College level (Researcher of the Year and Teaching Excellence). He was a Keynote Speaker for IEEE APCCAS, VCVP (twice), JCPC, IEEE ICAST, IEEE ICIEA, IET FC Umedia, IEEE Umedia, IEEE ICCIT, ICNLP/SSPS/CVPS, and Workshop at XUPT (twice). He has served as a General Chair/CoChair for IEEE Hot Chips, VCVP (twice), IEEE ICME, Umedia (seven times), IEEE SiPS, and IEEE VCIP. He was an Honorary Co-Chair for IEEE Umedia 2017. He has also served as a Technical Program Co-Chair for IEEE ISCAS (twice), APSIPA ASC, IEEE APCCAS, IEEE SiPS (twice), DCV, and IEEE VCIP. He was a Technical Committee Chair for IEEE CASCOM TC and IEEE TCMM, and is currently the Chair of the APSIPA U.S. Chapter. He has served as a Guest Editor or an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS — I: REGULAR PAPERS, the IEEE JOURNAL OF SELECTED TOPICS IN SIGNAL PROCESSING, Springer JSPS, and Springer MSSP. He has delivered more than 120 invited colloquia worldwide and has served as Visiting Professor/Consultant/Scientist for many institutions/ companies.