Encoder-aware motion compensated temporal filtering for video compression

Rahul Vanam and Sriram Sethuraman
Amazon Prime Video, 2021 7th Ave, Seattle, WA 98121 USA
{vanarahu,sssethur}@amazon.com

ABSTRACT

In this paper, we present an encoder-aware motion compensated temporal pre-processing filter (EA-MCTF) that adapts the filter on a block-basis based upon the spatio-temporal content properties and block-level encoding parameters. Some sample parameters include block-level QP, variance and mean-squared error of motion compensated block difference, slice types of adjoining frames, and frequency of a block being used as a reference. Applying the EA-MCTF to a HEVC encoder yields -12.4% average VMAF BD-rate savings over unfiltered encodings, and furthermore, the EA-MCTF yields a superior BD-rate and computational complexity performance over the MCTF available in the HEVC reference software.

Keywords: Pre-processing filter, HEVC, MCTF.

1. INTRODUCTION

An encoder typically consumes large number of bits when compressing videos at high quality, thereby incurring high delivery cost. This drives the need to reduce the bits for lowering the delivery cost while still maintaining a high visual quality. Motion Compensated Temporal Filtering (MCTF) is a pre-filtering approach employed to improve the compression efficiency of a video encoder. MCTF is commonly applied on uncompressed frames prior to encoding as shown in Figure 1. MCTF performs motion estimation and temporal filtering to align blocks in a given frame with matching blocks in neighboring frames. This alignment reduces the inter-prediction residual energy that yields smaller residual coefficients in the transform domain, requiring fewer bits to encode after quantization, and thereby improving the compression performance. MCTF does not require signaling additional syntax elements in the encoded bitstream. Therefore, bitstreams generated by a video encoder using MCTF are still compliant to a given video coding standard, and can be decoded by any standard compliant video player.

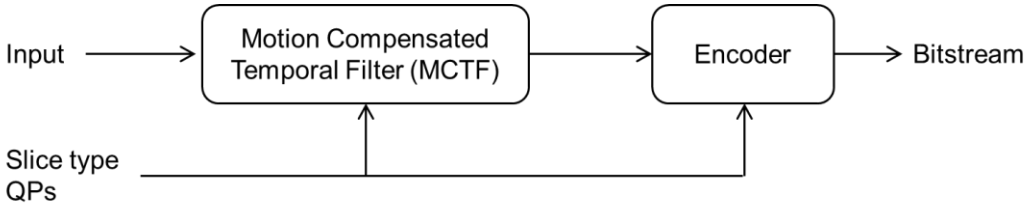

Figure 1. Typical motion compensated temporal filter (MCTF) used as a pre-processing step to an encoder.

Some of the earliest work on MCTF (e.g. [1]) demonstrated its ability to reduce noise in videos and improve compression when used as a pre-processing step to a DPCM-based encoder. In more recent work, MCTF has been used for improving the compression performance of HEVC, Versatile Video Coding (VVC), and AV1 encoders. The MCTF in [2] and [3] are a part of the HEVC [5] and VVC reference software [6], respectively. The MCTF design in [2] and [3] was further enhanced in [7] using a non-local mean temporal filter, and was applied to improve the compression efficiency of an AV1 encoder.

In prior work (e.g. [2], [3]), MCTF was designed for reference encoders that uses pre-defined frame-level QPs for different slice types and temporal layers, as shown in Figure 1. However, commercial encoders typically adapt QPs on a block-basis based upon frequency of the block being referenced, and the spatial complexity characteristics of the block. These block QPs may not match the pre-defined slice QPs. Therefore, MCTF from prior work when used with a commercial encoder may over-/under-filter certain blocks where the block QPs do not match the pre-defined QPs. Over-filtering can attenuate signal components such as edges, thereby severely degrading the quality of the compressed video. Under-filtering will

compromise the compression efficiency possible with optimal QP-aware filtering. Furthermore, the prior MCTF design is applicable to encoders operating with a fixed Group of Picture (GOP) structure, making such MCTF less practical for real-world encoders.

In this paper, to overcome the drawbacks of traditional MCTF, we propose an encoder-aware MCTF (EA-MCTF) that uses parameters derived within an encoder to adapt the temporal filtering strength on a block-basis. This allow us to design the temporal filter such that the filter induced error is within the quantization noise level of a given block. This paper focuses on the EA-MCTF design for the x265 encoder [8], a popular open-source HEVC encoder. The design can however be easily ported to other video encoders.

This paper is organized as follows. The encoder-aware MCTF is described in Section 2. Experimental results and performance comparison are made in Section 3. Conclusions are provided in Section 4.

2. ENCODER-AWARE MCTF

The EA-MCTF has been integrated into the x265 encoder as shown in Figure 2, to allow access to different parameters within the encoder, including lookahead and rate control parameters. The EA-MCTF is applied to an uncompressed frame just before it can be encoded by the core encoder module.

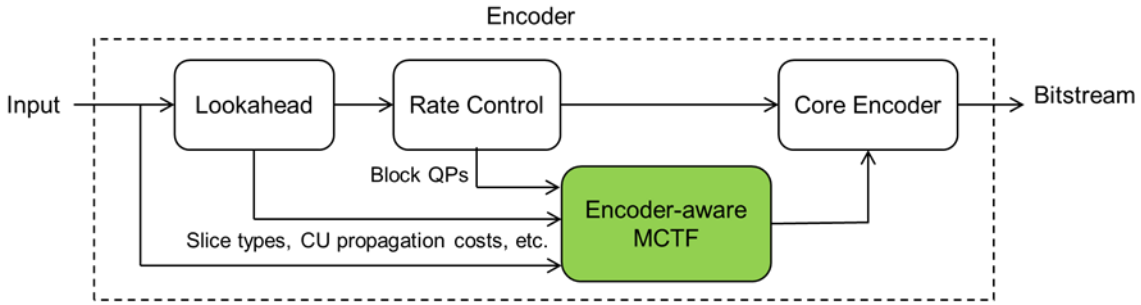

Figure 2. Proposed encoder-aware MCTF (EA-MCTF) operating within the encoder.

Similar to a classical MCTF, the EA-MCTF has broadly three main stages: motion estimation, temporal filter parameter derivation, and filtering. The temporal filter parameters consist of encoder derived parameters, and pre-defined parameters. The three stages of EA-MCTF are described next.

2.1. Motion Estimation

Motion estimation (ME) on the luma component is performed between a given frame *t* and each of its neighboring frames within a temporal window. The ME performed for EA-MCTF is different from the ME performed within the encoder. In EA-MCTF, there is a need to find true motion vectors (MVs), while ME in the encoder considers rate-distortion cost that may not yield true MVs. Furthermore, as ME in encoder is less robust to noise, it may yield inaccurate MVs in the presence of noise. Therefore, the hierarchical motion estimation (HME) is employed by EA-MCTF since it is more robust than ME used in encoder, and can cover large search ranges. To improve the MV accuracy of HME, a three-layer Gaussian pyramid is used instead of the mean pyramid in [2, 4]. Remaining HME components are similar to [2, 4].

2.2. Encoder derived temporal filter parameters

The encoder-derived temporal filter parameters are derived on a block-basis for both luma and chroma components, and are described next.

2.2.1 Block QPs, motion compensated MSE, and spatial variance parameter

The x265 encoder includes adaptive quantization-mode (aq-mode) and CU-tree that can help improve the compression efficiency. Both tools adapt QPs on a block-basis considering the spatial complexity of the block and the frequency with which the block is referred to for prediction. We use the block-level QPs derived from the rate control as a parameter for the temporal filter. For the chroma temporal filter, slice- and PPS-level chroma QP offsets are also included in the block-level QPs.

For each 4×4 block *b* and color component *p*, the following filter weight is computed:

$$s_{blk} = e^{-\frac{MSE_{b,p}(i)}{\psi(QP(b,p)-\beta)^2}}, \quad (1)$$

$$\text{where } \psi = \max(Sigmoid(\gamma, \gamma_0, L, k), 1),$$

$$\gamma = \frac{MSE_{b,p}(i)}{\sigma_{b,p}^2},$$

$QP(b,p)$ is the associated block QP, $MSE_{b,p}(i)$ is the mean squared error between a 4×4 block in frame *t* and its associated motion compensated block in reference frame *i*, and $\sigma_{b,p}^2$ is the block variance in frame *t*. The sigmoid function is defined as

$$Sigmoid(\gamma, \gamma_0, L, k) = \frac{L}{1 + \exp\left(-k(\gamma - \gamma_0)\right)}, \quad (2)$$

where k, L, β and γ_0 are the filter parameters. Example filter parameters have been listed in Section 3. For well motion compensated blocks where $MSE_{b,p}(i)$ is fairly small, ψ is derived using Equation (1), otherwise ψ is set to one. The ψ parameter regulates the filtering strength, by yielding higher s_{blk} weights for blocks having higher spatial variance and lower motion compensated MSE. While blocks having lower spatial variance and higher MSE yield lower weights. Figure 3 illustrates variation of s_{blk} with $MSE_{b,p}(i)$ and QP, for a fixed γ. Since the quantization noise level becomes smaller at lower QPs, s_{blk} is reduced with lower QPs to keep the filter introduced distortion within the quantization noise level. The weight is also reduced with higher MSE, as shown in Figure 3, to reduce the influence of dissimilar neighboring blocks on the current block during temporal filtering. Unlike our filter weight in Equation (1), the MCTF in the HEVC reference software (referred hereafter as HM-MCTF) does not depend upon γ.

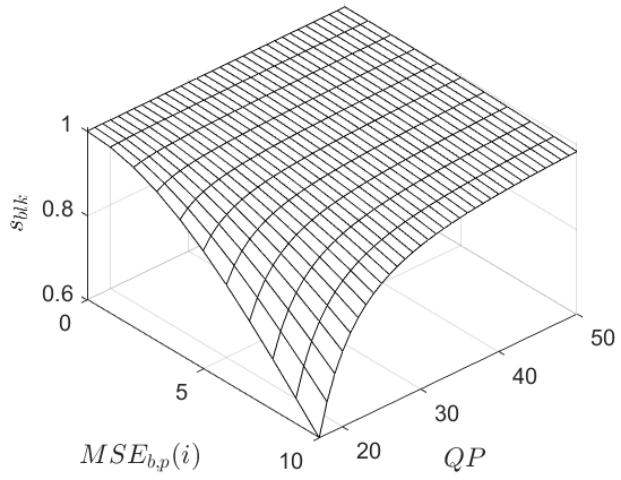

Figure 3. Variation of s_{blk} with $MSE_{b,p}(i)$ and QP, when $\gamma = 4$.

2.2.2 Coding Unit-tree propagation cost parameter

The x265 encoder assigns a CU-tree propagation cost to each block in a given frame indicating how frequently the block is used as a reference for prediction by other frames during encoding. Blocks that are frequently being referenced have a large CU-tree propagation cost. We use this cost to derive a filter weight. Frequently referenced blocks are benefited more by EA-MCTF, as they allow such blocks to be better aligned with their matching blocks, thereby improving prediction. Therefore, if a block in a frame *t* has a large propagation cost, a weight closer to one is assigned, which results in temporal filtering of the block. However, if the current block in a frame *t* is not referenced (i.e., has a propagation cost close to zero), a weight with value closer to zero is assigned, thereby reducing the influence of neighboring reference frames during temporal filtering. The following function is used for deriving weights using the CU-tree propagation cost *c*, and reference frame index *i*

$$s_w(i,c) = \begin{cases} f_1(c) = Sigmoid(c, c_0, M, k_1), i \in \{(t-1), (t+1)\} \\ \min\left(f_1(c), Sigmoid(c, c_0, M, k_2)\right), i \in \{(t-2), (t+2)\}, \end{cases} \quad (3)$$

where $Sigmoid(.)$ is defined in Eq (2). Example filter parameters c_0, M, k_1, and k_2 are listed in Section 3. Figure 4 illustrates Equation (3). For adjacent reference frames, the weight is designed to quickly rise to one, as these frames are likely to be more correlated to the current frame.

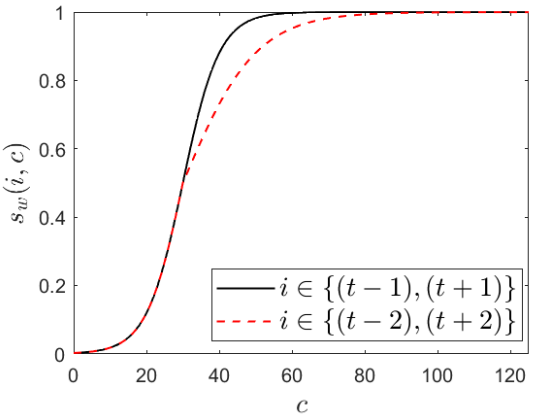

Figure 4. $s_w(i,c)$ as a function of CU-tree propagation cost c and reference frame index i.

2.2.3 Temporal window adaptation

Operating an encoder with adaptive B-frame placement can improve its compression efficiency, as it gives the encoder the flexibility to use B-frames anywhere without adhering to a fixed GOP (Group of Picture) structure. Sometimes, this may yield consecutive P-frames, especially in segments having fast motion. Fast motion may impact the accuracy of motion estimation in EA-MCTF, resulting in temporal filtering across dissimilar samples, impacting quality. To avoid this, we propose to disable reference frames that are chosen as P-frames by the lookahead stage of the encoder. While the filter term in Equation (1) already reduces filtering strength for dissimilar neighboring blocks, this feature provides an additional guardrail by skipping the use of reference frames containing fast motion during temporal filtering.

If one of the adjoining reference frames ($t-1$ or $t+1$) is a P-frame, EA-MCTF is disabled for the current frame, as illustrated in Figure 5(a). Similarly, if a n-th reference frame is a P-frame, we reduce the temporal window to 2n-1, and do not use reference frames outside the temporal window for filtering. Figure 5(b) illustrates for P-frame at n=2, where the temporal window is reduced to $(t-1, t, t+1)$.

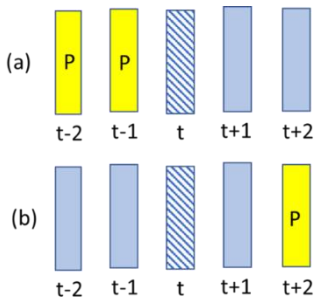

Figure 5. Examples for temporal filter window adaptation. (a) Temporal window reduced to t. (b) Temporal filter window reduced to $(t-1, t, t+1)$.

2.2.4 Local luma parameter

Temporal filtering on dark regions of a video may filter fine spatial gradients or details making it prone to banding artifacts during encoding. To avoid this, a filter term $s_Y(b)$ is computed based on the average luma of a 4×4 block that assigns smaller weights to lower luma values, by using a sigmoid function similar to Equation (3).

The EA-MCTF uses pre-defined parameters from the HM-MCTF [2]. This includes the color component parameter s_p, reference frame parameter $s_r(i,a)$ associated with a reference index i and window size a, and temporal layer parameter s_0.

2.3. Temporal filtering

The overall filter weight for a block b in a reference frame i, having a CU-tree propagation cost c, belonging to a temporal window of size a, and color component p, is obtained by taking the product of individual filter weights as follows:

$$w_r(i,a,b,c,p) = s_0 s_p s_Y(b) s_{var}(b) s_r(i,a) s_w(i,c) e^{-\frac{MSE_{b,p}(i)}{\psi(QP(b,p)-\beta)^2}}. \quad (4)$$

Temporal filtering on original pixel $I_0(p)$ using motion compensated pixels $I_r(i,p)$ and weights $w_r(i,a,b,c,p)$ is as follows

$$I_n = \frac{I_0(p) + \sum_{i=-2}^{2} w_r(i,a,b,c,p) I_r(i,p)}{1 + \sum_{i=-2}^{2} w_r(i,a,b,c,p)}. \quad (5)$$

2.4. Speed optimizations

The EA-MCTF has number of features that lowers its computational complexity over the HM-MCTF. The EA-MCTF is a block-adaptive filter operating on 4×4 blocks, unlike the pixel-adaptive HM-MCTF. For integer motion estimation in HME, the EA-MCTF uses a smaller search window of 3×3 for both L1 and L2 layers, while the HM-MCTF uses 5×5 and 17×17 search windows for L1 and L2 layers, respectively. Furthermore, the EA-MCTF only evaluates 3×3 motion vector predictor candidates from a previous HME layer, while HM-MCTF evaluates 5×5 candidates. The EA-MCTF uses the x265's sub-pel ME implementation and derives quarter-pel motion vectors (MVs) using the same interpolation filter as in HM-MCTF. The HM-MCTF derives 1/16-pel MVs for luma.

3. EXPERIMENTS AND RESULTS

The performance of EA-MCTF in the x265 encoder is compared against no MCTF encodings and encodings using the HM-MCTF. The x265 encoder is configured to use the slow preset parameter settings [9] that yields higher compression efficiency at the cost of slower encoding speed. The slow preset includes parameters such as adaptive B-frame placement based on trellis, use of 25 lookahead frames for slice-type decision, four reference frames, use of star motion search method, and the use of subme = 3 that performs higher number of half-pel and quarter-pel iterations and steps during the sub-pel motion estimation. A GOP length of 16 is chosen, which typically results in a hierarchical coding structure with three temporal layers, with layer-0 containing I- and P-frames, layer-1 containing reference B-frames, and layer-3 containing the non-reference B-frames. The use of adaptive B-frame placements also allows the encoder to the change the above GOP structure based upon motion characteristics. The encoder is operated with the CRF (constant rate factor) rate control [14] that yields quality-controlled variable bitrate [9].

The HM-MCTF results were obtained by replacing the EA-MCTF filter with HM-MCTF filter in our implementation, so that HM-MCTF also uses encoder derived block-level QPs. Both the EA-MCTF and HM-MCTF are applied to layer-0 and layer-1 frames only, and use a maximum temporal window size of five frames (with two reference frames in each direction). The EA-MCTF is configured to use filter parameters $k = -1, \gamma_0 = 8, L = 16, \beta = 10, c_0 = 30, M = 1, k_1 = 0.2$, and $k_2 = 0.1$. Ten 1920×1080 8-bit test sequences from the derf's collection [10] and the JVET CTC test dataset [11] are used in our evaluation, listed in Table 1. Sequences that are 10-bit YUV420p10LE format are converted to 8-bit YUV420p format, and used in our tests.

Video quality is measured using the VMAF (Video Multimethod Assessment Fusion) metric [12]. Among the different video quality metrics, the VMAF metric was chosen since it was shown to have higher correlation to subjective scores across different resolutions and bitrates for SDR content [13]. The harmonic mean is used for generating the final VMAF score per encode. The EA-MCTF and HM-MCTF encodes are operated at four CRF values {19, 19.5, 20, 20.5} that yield high quality bitstreams. The encoder without the MCTF is operated at CRF = {20, 20.5, 21, 21.5} to yield bitrates similar to the MCTF encodes, for facilitating the VMAF BD-rate calculations.

VMAF BD-rates are computed for HM-MCTF and EA-MCTF using the no MCTF encodes as the anchor, and the results are presented in Table 1. The HM-MCTF yields a BD-rate loss on many of the clips, with a maximum loss of 22.3% on

the pedestrian_area sequence, and yields -1.9% average gain across all clips. The EA-MCTF yields a superior BD-rate performance across all test clips, with a maximum and average BD-rate gains of -50.6% and -12.4%, respectively.

In adaptive bitrate streaming, the encoder is typically operated with a peak bitrate constraint. We encoded six test clips using a peak bitrate of 9000 kbps, with the previous encoder parameter settings, and results are shown in Table 2. For all the clips, the encoder yields bitrate close to the peak bitrate, when operating either with no MCTF, HM-MCTF, or EA-MCTF. For both HM-MCTF and EA-MCTF, we measure

$$\Delta VMAF = VMAF_{MCTF} - VMAF_{no\ MCTF}. \quad (6)$$

For the EA-MCTF, we observe that there is improvement in VMAF score across all test clips over the unfiltered encodings, with a maximum improvement of 0.52 VMAF score. For challenging content, the encoder may reach the peak bitrate, restricting the encoder from using additional bits, thereby yielding a lower quality/VMAF score. Applying the EA-MCTF on such content can help the encoder to lower the residuals and bit consumption, and allow the encoder to use more bits for challenging regions while meeting the peak bitrate constraint, and thereby help improve the quality/VMAF score. The use of HM-MCTF lowers the VMAF scores on some test clips, with maximum VMAF drop of -0.27, as shown in Table 2.

Table 1. VMAF BD-rate results using the no MCTF encodes as the anchor.

Sequence	HM-MCTF BD-rate	EA-MCTF BD-rate
pedestrian_area	22.30%	-8.30%
rush_hour	10.70%	-8.90%
station2	-49.70%	-50.60%
sunflower	-3.30%	-11.30%
ducks_take_off	9.70%	-1.80%
crowd_run	-2.70%	-7.80%
BasketballDrive	7.80%	-5.70%
Cactus	-14.90%	-18.10%
ArenaOfValor	3.80%	-2.10%
MarketPlace	-2.60%	-9.40%
Average	-1.90%	-12.40%

Table 2. Results when operating the encoder with a peak bitrate of 9000 kbps.

	No MCTF		HM-MCTF			EA-MCTF		
Sequence	Bitrate (kbps)	VMAF	Bitrate (kbps)	VMAF	ΔVMAF	Bitrate (kbps)	VMAF	ΔVMAF
crowd_run	9502.1	80.59	9500.9	80.68	0.10	9502.6	80.81	0.22
BasketballDrive	9462.1	98.85	9486.2	98.58	-0.27	9464.7	98.93	0.08
Cactus	9483.7	95.69	9487.4	96.21	0.52	9488.0	96.21	0.52
ducks_take_off	9503.2	66.19	9504.9	65.97	-0.22	9503.2	66.41	0.22
ArenaOfValor	9496.5	95.65	9498.2	95.41	-0.24	9493.9	95.76	0.11
MarketPlace	9126.9	92.99	9139.5	93.16	0.17	9122.7	93.41	0.42

3.1. Computational complexity vs. compression efficiency trade-off

The computational complexity of the HM-MCTF and EA-MCTF are compared by running the x265 encoder using a single thread on a Linux machine with a 2.9 GHz Intel CPU. Tests include the six 1080p test clips listed in Table 3, and the timing measurements were made operating the x265 encoder with CRF=20. Table 3 lists the relative MCTF time measured as $(T_{enc+MCTF} - T_{enc}) \times 100/T_{enc}$, where $T_{enc+MCTF}$ and T_{enc} are the encode times with and without MCTF, respectively. Table 3 shows that HM-MCTF has enormous computational complexity with 784% average computational complexity relative to the encoder, while yielding an average BD-rate loss of 0.2%. The EA-MCTF yields a superior compression efficiency and computational complexity performance over the HM-MCTF, yielding 17% average complexity and -7.5% average BD-rate gain. Applying the EA-MCTF only to the luma component, yields 15% average complexity and -6.7% average BD-rate gain. Disabling the sub-pixel ME in the EA-MCTF yields further speedup with 13% average complexity and -6.4% average BD-rate gain. All the three variants of the EA-MCTF yield different speed vs. compression efficiency trade-offs, and they all perform superior to the HM-MCTF.

Table 3. Comparison of VMAF BD-rate and relative MCTF time for the HM-MCTF, EA-MCTF, EA-MCTF applied to luma only, and EA-MCTF using integer ME only.

	HM-MCTF		EA-MCTF		EA-MCTF luma only		EA-MCTF integer ME	
Sequence	BD-rate	MCTF time	BD-rate	MCTF time	BD-rate	MCTF time	BD-rate	MCTF time
crowd_run	-2.7%	651%	-7.8%	14%	-7.2%	11%	-6.2%	10%
BasketballDrive	7.8%	890%	-5.7%	19%	-4.9%	16%	-4.9%	14%
Cactus	-14.9%	831%	-18.1%	19%	-16.2%	17%	-17.1%	14%
ducks_take_off	9.7%	560%	-1.8%	9%	-1.2%	8%	-1.5%	7%
ArenaOfValor	3.7%	941%	-2.1%	29%	-1.8%	26%	-0.6%	23%
MarketPlace	-2.6%	830%	-9.4%	14%	-8.7%	12%	-8.1%	10%
Average	0.2%	784%	-7.5%	17%	-6.7%	15%	-6.4%	13%

4. CONCLUSIONS

In this paper, we presented an encoder-aware MCTF (EA-MCTF) that is integrated into the x265 encoder, and our EA-MCTF adapts on a block-basis based upon encoder derived parameters and spatio-temporal characteristics of the content. The EA-MCTF yields BD-rate gain across all test clips, with an average and maximum VMAF BD-rate gains of -12.4% and -50.6% over unfiltered encodings, respectively. While the HM-MCTF yields BD-rate loss on multiple test clips, with an average VMAF BD-rate gain of -1.9%. When operating the encoder with a peak bitrate constraint, the EA-MCTF consistently yields higher VMAF compared to the unfiltered encodings. Finally, computational complexity analysis shows that the EA-MCTF and its variants are significantly faster and have higher compression efficiency compared to the HM-MCTF.

REFERENCES

[1] E. Dubois and S. Sabri, "Noise Reduction in Image Sequences Using Motion-Compensated Temporal Filtering," *IEEE Transactions on Communications*, vol. 32, no. 7, pp. 826-831, July 1984.

[2] P. Wennersten, J. Östrand and R. Sjöberg, "Encoder only GOP based temporal filter," JCTVC AI0023, Geneva, March 2019.

[3] P. Wennersten, R. Sjöberg and J. Enhorn, "AHG10: Encoder only GOP based temporal filter," JVET O0549, Gothenburg, July 2019.

[4] J. Enhorn, R. Sjöberg and P. Wennersten, "A temporal pre-filter for video coding based on bilateral filtering," *IEEE International Conference on Image Processing (ICIP)*, pp. 1161-1165, 2020.

[5] HEVC reference encoder software, URL: https://vcgit.hhi.fraunhofer.de/jvet/HM.

[6] VVC reference encoder software, URL: https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM.

[7] C. Chen, J. Han and Y. Xu, "A Non-local Mean Temporal Filter for Video Compression," *IEEE International Conference on Image Processing (ICIP)*, pp. 1142-1146, 2020.

[8] x265 software encoder, URL: https://github.com/videolan/x265.

[9] x265 command line options, URL: https://x265.readthedocs.io/en/master/cli.html.

[10] derf's collection, URL: https://media.xiph.org/video/derf/.

[11] JVET CTC test sequences, URL: ftp://ftp.ient.rwth-aachen.de.

[12] VMAF, URL: https://github.com/Netflix/vmaf.

[13] Z. Li, A. Aaron, I. Katsavounidis, A. Moorthy and M. Manohara "Toward A Practical Perceptual Video Quality Metric," Netflix Technology Blog, June 2016, URL: https://netflixtechblog.com/toward-a-practical-perceptual-video-quality-metric-653f208b9652.

[14] L. Merritt and R. Vanam, "Improved rate control and motion estimation for H.264 encoder," *IEEE International Conference on Image Processing (ICIP)*, vol. 5, pp. V-309, 2007.