J. Vis. Commun. Image R. 89 (2022) 103672

Contents lists available at ScienceDirect

J. Vis. Commun. Image R.

journal homepage: www.elsevier.com/locate/jvci

Full Length Article

Accelerating BPC-PaCo through Visually Lossless Techniques☆

Francesc Aulí-Llinàs *, Carlos de Cea-Dominguez, Miguel Hernández-Cabronero

Department of Information and Communications Engineering, Universitat Autònoma de Barcelona, Bellaterra, 08193, Spain

ARTICLE INFO

Keywords:
High-throughput image coding
Visually lossless coding
JPEG2000

ABSTRACT

Fast image codecs are a current need in applications that deal with large amounts of images. Graphics Processing Units (GPUs) are suitable processors to speed up most kinds of algorithms, especially when they allow fine-grain parallelism. Bitplane Coding with Parallel Coefficient processing (BPC-PaCo) is a recently proposed algorithm for the core stage of wavelet-based image codecs tailored for the highly parallel architectures of GPUs. This algorithm provides complexity scalability to allow faster execution at the expense of coding efficiency. Its main drawback is that the speedup and loss in image quality is controlled only roughly, resulting in visible distortion at low and medium rates. This paper addresses this issue by integrating techniques of visually lossless coding into BPC-PaCo. The resulting method minimizes the visual distortion introduced in the compressed file, obtaining higher-quality images to a human observer. Experimental results also indicate 12% speedups with respect to BPC-PaCo.

1. Introduction

Imagery employed in myriad scenarios is nowadays experiencing a qualitative and quantitative step forward. Images with very high spatial and temporal resolution and with high dynamic range are currently commonplace. This quality increment is accompanied with a proliferation of image-based applications such as teleconferencing [1], geographic information systems [2], and online TV [3], among others [4–6]. The fast increase in both quality and quantity requires faster and more efficient codecs to accelerate the coding and transmission, and to reduce the storage costs of these images.

The development of faster codecs has followed different paths in the field of image coding. A common strategy is to reduce the computational complexity of the algorithms. Popular coding systems such as SPIHT [7] or EBCOT [8], which are employed in a wide variety of environments such as the medical, the remote sensing, or the video edition, have been re-formulated many times with the aim to reduce their complexity and/or memory requirements [9–14]. Another strategy is to implement the codec in hardware architectures such as field-programmable gate arrays [15–17] or application-specific integrated circuits [18–21], so that these codecs can be employed in mobile devices such as (video)cameras. More recently, the advent of multi-core Central Processing Units (CPUs) and highly parallel architectures such as Graphics Processing Units (GPUs) has stimulated the implementation of codecs with multi-thread capabilities for CPUs [22–25] and GPUs [26–29]. These works focus on the acceleration of the codecs without modifying their inner algorithms. This maintains compliance with the original systems, but limits the parallel strategies that can be employed. The main handicap of conventional coding engines is that the core of the coding system commonly uses a single-thread procedure to code a whole chunk of data, so only coarse-grain parallelism can be applied over a multi-tile division of the image. GPUs are mostly based on the single-instruction multiple data architecture. To fully use their resources, the core algorithms must be tailored for fine-grain parallelism [30], so that vector instructions can be efficiently employed.

The Joint Photographic Experts Group realized this facet and in 2019 a new part to the JPEG2000 [31] standard called High Throughput JPEG2000 (HTJ2K) was introduced [32]. The new part uses a core algorithm that supports vector instructions sets included in modern CPUs and GPUs [33–35], so it can be employed for fast TV production, digital cinema, and other scenarios that require real-time processing. The main drawback of HTJ2K is that it sacrifices quality scalability, which is a useful feature to transmit the image progressively by quality [36].

Well before the standardization of HTJ2K, we started a research line whose goal is to introduce fine-grain parallelism to all stages of a JPEG2000-like codec while keeping all features of the original system, including quality scalability. The first proposed techniques extricated the causality of single-threaded coding strategies [37], introduced a lightweight arithmetic coder [38], and a fast GPU implementation of

☆ This paper has been recommended for acceptance by Zicheng Liu.

* Corresponding author.
E-mail address: francesc.auli@uab.cat (F. Aulí-Llinàs).

https://doi.org/10.1016/j.jvcir.2022.103672
Received 20 June 2022; Received in revised form 22 September 2022; Accepted 15 October 2022
Available online 25 October 2022

the discrete wavelet transform [26]. The bitplane coder is the second and most complex stage of the system. It was re-formulated and tailored for fine grain-parallelism, resulting in a Bitplane Coding engine with Parallel Coefficient processing (BPC-PaCo) [27,39]. The end-to-end GPU implementation of our codec was presented in [29] achieving 12K resolution video compression in real time with only 2% penalization in coding efficiency. Our last step introduced a modification to BPC-PaCo to trade coding efficiency for computational complexity in an adjustable fashion, resulting in a Complexity Scalable BPC-PaCo engine (CS BPC-PaCo) [36,40].

The trading between throughput and image quality in CS BPC-PaCo is regulated via a user parameter. Rate–distortion optimization techniques minimize the impact in coding performance when more throughput is required, but the parameter neither regulates the amount of sacrificed quality nor the increase in throughput. This may not suit scenarios in which visual quality is essential. Visually lossless techniques are interesting in such scenarios since they preserve the level of quality at which the image distortion becomes indistinguishable for the human visual system. These techniques set a visually lossless threshold that estimates the maximum absolute error between the original and the compressed image, resulting in just imperceptible distortion for a human observer [41]. In the framework of JPEG2000, visually lossless coding has been proposed in [42–45] employing different distortion models.

The work proposed herein joins the complexity scalability techniques employed in CS BPC-PaCo with a visually lossless model tailored for our engine. Briefly described, the coding process uses the original algorithm of BPC-PaCo until the visually lossless threshold is attained. This maximizes the quality when the image is progressively transmitted. Then, the codec is maximally accelerated at the expense of coding efficiency via a fast mode. This penalizes the efficiency in compression only in data segments that are not visually relevant. Since the employed model determines the visually lossless threshold autonomously, the codec does not require user supervision and maximizes the throughput and quality of the image that is visually relevant for a human observer.

The rest of the paper is organized as follows. Section 2 reviews the original CS BPC-PaCo. Section 3 describes the distortion model and details its integration in the coding engine. The proposed method is implemented in a Java framework. Coding performance and computational complexity are compared with BPC-PaCo and CS BPC-PaCo in Section 4. Experimental results show that the proposed method significantly reduces the impact on image quality when the image is transmitted progressively while increasing the coding speed by about 8%, on average. The last section provides conclusions.

2. Review of complexity scalable BPC-PaCo

The main stages of the JPEG2000 coding pipeline are [46]: wavelet transform, bitplane coding, and codestream re-organization. The discrete wavelet transform decorrelates the image information producing a multi-resolution decomposition of the image organized in wavelet subbands that contain the vertical, horizontal, and diagonal details of the image. Each subband is partitioned in small sets of typically 64×64 wavelet coefficients called codeblocks. Each codeblock is coded independently by the bitplane coding stage, which produces a quality embedded bitstream that can be truncated at different rates. The last stage (possibly truncates and) re-organizes the bitstreams of all codeblocks in a single or multiple layers of quality satisfying a target rate, commonly employing rate–distortion optimization techniques.

Bitplane coding is the most sophisticated stage of the image codec, entailing approximately 80% of the total coding time [27]. Let ω denote a wavelet coefficient and υ an integer with the magnitude of the quantization index obtained for such coefficient. The binary representation of υ is expressed as $[b_{M-1}, b_{M-2}, \dots, b_1, b_0]$, $b_i \in \{0,1\}$, with an M sufficiently large to hold all coefficients within the codeblock. The same bit position b_j of all coefficients is defined as bitplane j. Bitplane coding strategies code all bits from $M-1$, the highest bitplane of the codeblock, to bitplane 0. The collection of bits in each bitplane is coded in three coding passes. The first is called Significance Propagation Pass (SPP) and codes bits from coefficients that were not significant in previous bitplanes but that have significant adjacent neighbors. A coefficient becomes significant at bitplane s when $b_s = 1$ with $b_{s'} = 0, s' > s$. The second coding pass is referred to as Magnitude Refinement Pass (MRP) and codes the bits of coefficients that where significant in previous bitplanes, more precisely, those bits b_r, $r < s$. The last is the Cleanup Pass (CP) and codes the bits of the remaining coefficients. If a coefficient becomes significant in the SPP or CP, its sign $d \in \{+,-\}$ is coded immediately after so that it can be partially reconstructed at the decoder. The bitstream can be truncated at the end of each coding pass, if necessary, by the rate–distortion optimization method to minimize the distortion of the image when it is progressively transmitted.

As seen in Fig. 1, the main difference between JPEG2000 and CS BPC-PaCo is that JPEG2000 uses a single thread (per codeblock) that scans the coefficients one by one, sending b_i to an arithmetic coder that produces an embedded bitstream. This bitstream can only be decoded bit by bit through the inverse procedure, without opportunities for parallelization. Contrarily, CS BPC-PaCo employs a parallel scanning order in which half the coefficients in a row (typically, 32 for codeblocks of 64×64) are coded in parallel. The bits of these coefficients are coded with multiple arithmetic coders that are synchronized at some points to produce the final bitstream. This is the main insight to obtain fine-grain parallelism.

Regardless of using one or multiple threads of execution, the coefficients are scanned many times. This slows the coding process because, in most architectures, not all coefficients can be simultaneously stored in the register space of the processor and they are transferred back and forth from the memory, causing the so-called register spilling [36]. To accelerate the coding engine, register spilling must be minimized. The complexity scalable technique employed in CS BPC-PaCo selects a bitplane N from which the bits of all coefficients are coded in a single pass. So, bitplanes $[M-1, N]$ are coded with three coding passes, providing truncation points that can be used by the rate–distortion optimization method to optimize the image quality, and bitplanes $[N-1, 0]$ are coded in a single pass to avoid register spilling, speeding up the coding process. The selection of N is key to balance throughput and compression efficiency. CS BPC-PaCo determines it as

$$N = \min\left(M, \left\lfloor M \cdot \frac{K}{\mathcal{L}_u} \right\rfloor\right), \tag{1}$$

with $\mathcal{L}_u$ being the L_2norm of the synthesis basis vectors of the filter-bank of subband u (equal energy gain factor is assumed in all subbands). K is the user-controlled parameter. Experimental evidence suggests that this parameter should be in the range $K \in [0.5, 1.5]$ to obtain a compromise between throughput and quality.

3. Proposed method

3.1. Distortion model

For the user, parameter K is only indicative. A given value of K may sacrifice quality and throughput differently for different images. A more suitable strategy is to reduce the compression efficiency only when the visual quality of the image is indistinguishable from the original, i.e., when the visually lossless threshold is attained. The most accurate distortion model for the deadzone quantizer employed by JPEG2000 is described in [43]. Such a quantizer reconstructs the wavelet coefficient as

$$\hat{\upsilon} = \begin{cases} 0 & \text{if } j > s \\ d([b_{M-1}, b_{M-2}, \dots, b_j] + \delta) \cdot \Delta_u 2^j & \text{otherwise} \end{cases}, \tag{2}$$

where $\delta \in [0,1)$ is commonly set to 0.5 to reconstruct the coefficient in the middle of the quantization interval. Δ_u is the step size

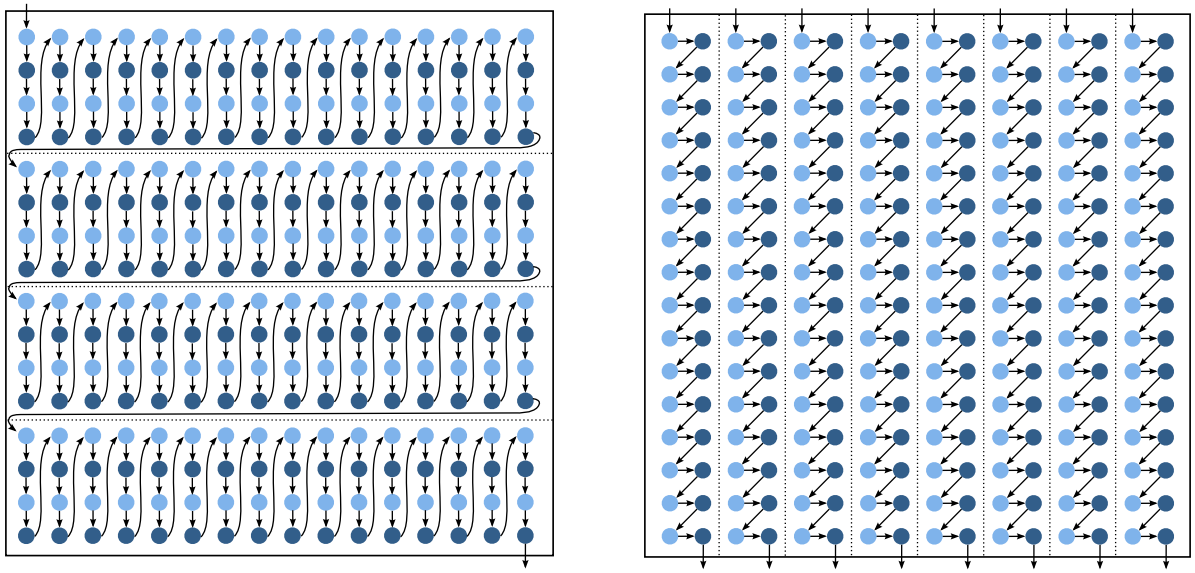

Fig. 1. Scanning order employed in: **(a)** JPEG2000 standard, and **(b)** CS BPC-PaCo.

employed for subband u. The distortion model of [43] considers that coefficients within the deadzone (i.e., those reconstructed as 0 above), have the same distribution as the original coefficients, and the remaining coefficients have a uniform quantization distortion. This model is employed to estimate visually lossless thresholds for an assumed coefficient variance σ^2 and quantization step size Δ_u, denoted as $\mathrm{VT}_u(\sigma^2)$. These thresholds are based on the well-known work of Watson and Daly [47,48].

In the encoder of [43], the above perceptual model is introduced as follows. At the end of each coding pass the maximum absolute error between the original coefficients and those reconstructed by the decoder is computed as $D = \max(|w - \hat{w}|)$. When $D \leq \mathrm{VT}_u(\sigma^2)$, the remaining passes are discarded since they do not contain visually relevant information. $\mathrm{VT}_u(\sigma^2)$ is determined at the beginning of the coding process computing the coefficients' variance. Other techniques (such as visual masking) are also employed in [43] to better refine this threshold, though they are not employed in our codec due to their computational costs.

3.2. Visually lossless BPC-PaCo

The goal of [43] is to reduce the codestream size by discarding the lesser visually relevant information. Contrarily, the method presented herein employs the distortion model to accelerate the codec only when coding such information. To this end, three coding passes per bitplane are generated before reaching the threshold, and then one more for the visually non-relevant information. The aim of this strategy is to provide many coding passes at visually relevant rates, optimizing the quality scalability of the codestream when the visual information is most relevant. To this end, instead of computing the bitplane at which the codec is accelerated via Eq. (1) and parameter K, it is determined autonomously during the coding process when $\mathrm{VT}_u(\sigma^2)$ is attained.

Unfortunately, the method proposed in [43] is not suitable for our engine. The main difficulty is that the encoding process requires a version of the reconstructed coefficients (i.e., $\hat{w}$) to compute D. This may significantly increase the memory needed per coefficient, resulting in more register spilling. An alternative is to use the model employed in [44], which is devised for the decoder. It modifies [43] in two aspects. The first is to estimate the variance σ^2 for the codeblock depending on the wavelet subband u and the number of magnitude bitplanes M. The second is to employ the distortion upper bound D' instead of D. D' is determined in bitplane j as

$$D' = \begin{cases} \left.\begin{array}{ll} \Delta_i 2^j & \text{if pass} = \text{CP} \\ \Delta_i 2^{j+1} & \text{otherwise} \end{array}\right\} & \text{if } \exists\ \hat{w} = 0 \\ \left.\begin{array}{ll} \Delta_i 2^j & \text{if pass} = \text{SPP} \\ \Delta_i 2^{j-1} & \text{otherwise} \end{array}\right\} & \text{otherwise} \end{cases}. \tag{3}$$

In our engine, σ^2 does need to be estimated since it can be computed when the data are produced just after the wavelet transform, negligibly increasing computational costs. Bounding the distortion via D' is helpful and thus applied to our codec because avoids the need for computing the reconstructed coefficients, maintaining the same memory requirements of the original CS BPC-PaCo.

Algorithm 1 details the main operations of the proposed Visually Lossless BPC-PaCo (VL BPC-PaCo). The algorithm is expressed as the

Algorithm 1 VL BPC-PaCo

Parameters: M total magnitude bitplanes, σ^2 codeblock's variance, t thread

```
1: j ← M − 1
2: repeat
3:     SignificancePropagationPass(j)
4:     MagnitudeRefinementPass(j)
5:     CleanupPass(j)
6:     D′ = Δ_i 2^j
7:     j ← j − 1
8: until D′ ≤ VT_u(σ^2)
9: for y ∈ [0, numRows − 1] do
10:    for x ∈ [t · 2, t · 2 + 1] do
11:        for j′ ∈ [j, 0] do
12:            if j′ ≥ s then
13:                ACEncodeSignificance(b_j′, t)
14:                if j′ = s then
15:                    ACEncodeSign(d, t)
16:                end if
17:            else
18:                ACEncodeRefinement(b_j′, t)
19:            end if
20:        end for
21:    end for
22: end for
```

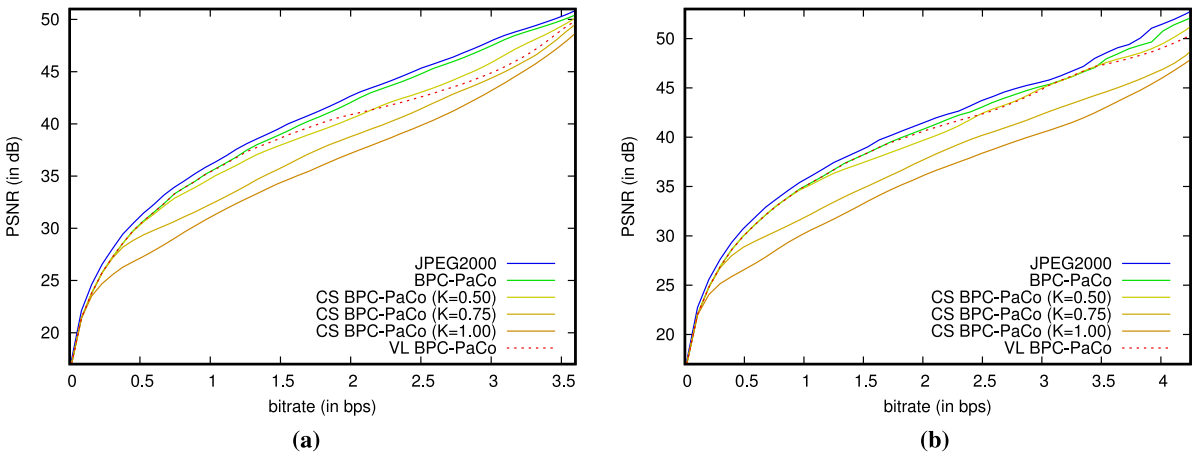

Fig. 2. Evaluation of coding performance for the "Cafeteria" image when the codestream is transmitted progressively with a **(a)** lossy and **(b)** lossless regime.

execution of one of the threads employed to code the data of a codeblock. t denotes the thread and serves to compute the columns that are coded. The first loop (lines 2 to 8) codes the data in the conventional mode of BPC-PaCo, i.e., using three coding passes per bitplane.[1] At the end of each bitplane (i.e., just after the CP) D' is computed to check whether the visually lossless threshold has been attained or not. If so, all remaining bitplanes of each coefficient are coded at once (lines 9 to 22). Functions ACEncode {Significance| Sign| Refinement} code the corresponding bit using the arithmetic coder for that thread. More details on the context formation and processing of the bitstream can be found in [39,40].

It is worth noting that Eq. (3) is not strictly applied in Algorithm 1. To do so, D' should be computed after the SPP only when there is none reconstructed coefficient (i.e., $\hat{w}$) equal to 0 at bitplane j. Experimental tests indicate that this is an extremely rare case, so it is deliberately omitted in our algorithm to simplify and accelerate the implementation.

4. Experimental results

The 8 color images of the ISO 12640-1 corpus are employed in the following experiments. They are 2560 × 2048 and have a bit-depth of 8 bits per sample (bps). Images of different sizes and from different corpora produce similar results as those reported in the following tests. The results report the performance achieved by JPEG2000, the original BPC-PaCo, CS BPC-PaCo (with $K \in \{0.5, 0.75, 1\}$), and the proposed VL BPC-PaCo. The values of K employed for CS BPC-PaCo are chosen to achieve high throughput (when $K = 1$), minimize the impact on coding performance (when $K = 0.5$), and a compromise in between. All methods are implemented in the same Java framework [49] so differences in throughput and coding efficiency are only related to the methods evaluated. To assess the throughput, the codec is run with a single thread, with the execution time providing an approximation of computational complexity. The reversible CDF 5/3 and irreversible CDF 9/7 filter banks are respectively used for lossless and lossy compression applying 5 levels of decomposition.

Figs. 2 and 3 report the results achieved for the "Cafeteria" and "Musicians" images when they are transmitted progressively from 0.01 bps to approximately 4 bps, at 50 equally spaced rates. The vertical axis reports the Peak Signal to Noise Ratio (PSNR), whereas the horizontal axis reports the bit rate of the compressed image. As seen in the figures, JPEG2000 and the original BPC-PaCo achieve similar performance.

[1] Our previous work also describes a variation of this coding scheme that employs two coding passes per bitplane (significance and refinement). The proposed method can be directly applied when using two coding passes too, so it is omitted herein for simplicity.

Table 1
Evaluation of lossless coding performance. Results are reported in bps. The lowest results among CS BPC-PaCo and VL BPC-PaCo are reported in bold.

	JP2	BPC-PaCo				
		Original	VL	CS $K = 0.5$	CS $K = 0.75$	CS $K = 1$
Portrait	3.80	4.00	4.02	**4.00**	4.04	4.08
Cafeteria	4.68	4.80	**4.81**	4.82	4.92	4.99
Fruit	3.96	4.15	4.19	**4.16**	4.20	4.22
Wine	3.94	4.12	**4.13**	**4.13**	4.16	4.19
Bicycle	3.90	4.09	**4.10**	**4.10**	4.16	4.20
Orchid	3.44	3.68	3.73	**3.69**	3.72	3.75
Musicians	5.34	5.52	**5.54**	5.58	5.77	5.86
Candle	4.74	4.87	**4.89**	4.91	5.03	5.10
average	*4.22*	*4.42*	***4.44***	***4.44***	*4.50*	*4.55*

CS BPC-PaCo penalizes more the coding performance for larger values of K. This penalization is approximately uniform from low to high rates. Contrarily, at low rates, VL BPC-PaCo achieves almost the same performance as that of the original BPC-PaCo. At medium and high rates in lossy regimes, the quality of the transmitted image starts decreasing since most codeblocks attain the visually lossless threshold. In lossless regimes, the proposed method penalizes almost nothing the image quality as compared to the original BPC-PaCo. Regardless of the compression regime, the penalization in quality produced by VL BPC-PaCo occurs at a higher rate than when using CS BPC-PaCo. This indicates that VL BPC-PaCo preserves better the quality at low rates, when it is more visually relevant. Similar results hold for the remaining images of the corpus.

Table 1 reports the coding performance achieved with a lossless regime for all images. The proposed method penalizes very little the coding performance, with an increase of only 0.02 bps on average with respect to the original BPC-PaCo. This increase is the same as that achieved with CS BPC-PaCo when $K = 0.5$. Nonetheless, the performance achieved by VL BPC-PaCo when the image is progressively transmitted is significantly higher than that achieved by CS BPC-PaCo with the same K, as seen in Figs. 2 and 3.

The next test evaluates the throughput achieved by the proposed method. Results are reported as the percentage of throughput increase in the bitplane coding engine achieved by the indicated method with respect to the original BPC-PaCo. Fig. 4 depicts average results for all images when coding in lossy and lossless regimes. As seen in the figure, VL BPC-PaCo obtains a throughput increase between that achieved for $K = 0.5$ and $K = 0.75$. For lossy compression, the proposed codec accelerates both the encoding and decoding process by approximately 8.5%, whereas for lossless compression the speedup is approximately 6%. Table 2 details the results for all images of the corpus. For some

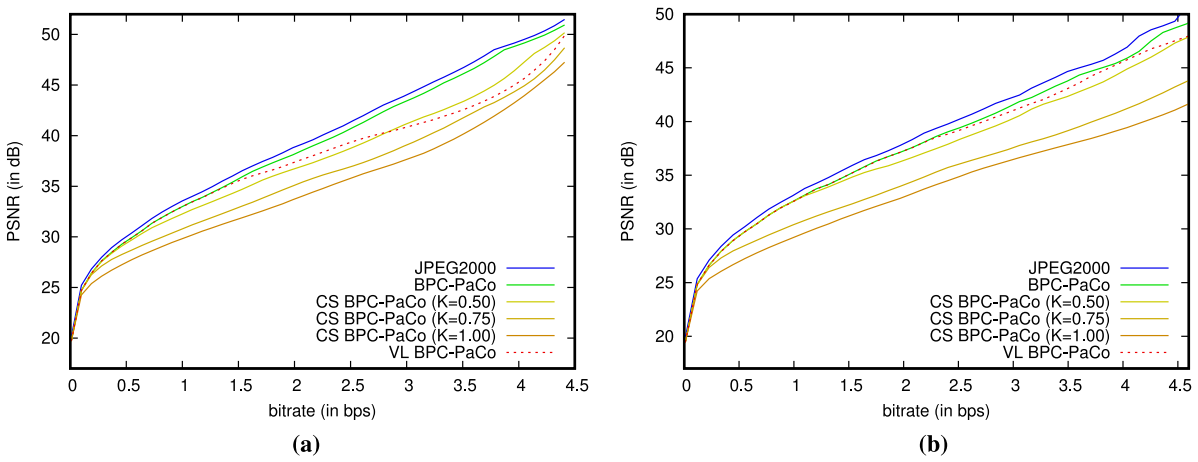

Fig. 3. Evaluation of coding performance for the "Musicians" image when the codestream is transmitted progressively with a **(a)** lossy and **(b)** lossless regime.

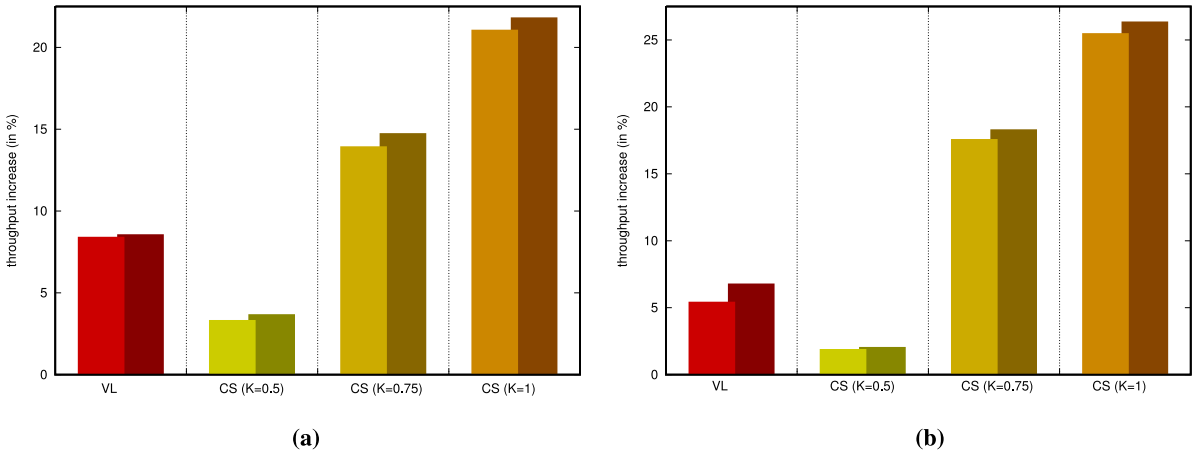

Fig. 4. Evaluation of the throughput increase achieved by VL BPC-PaCo (VL) and CS BPC-PaCo (CS) with respect to the original BPC-PaCo, on average for all images of the corpus when using (**a**) lossy and (**b**) lossless compression. The left column in each method reports the results for the encoder, whereas the right column for the decoder.

Table 2
Evaluation of the throughput increase (in percentage) achieved by VL BPC-PaCo and CS BPC-PaCo with respect to the original BPC-PaCo. Results are reported for the encoder (left) and decoder (right) in each cell.

	lossy								lossless							
	VL		CS BPC-PaCo						VL		CS BPC-PaCo					
	BPC-PaCo		$K = 0.5$		$K = 0.75$		$K = 1$		BPC-PaCo		$K = 0.5$		$K = 0.75$		$K = 1$	
Portrait	10.8	11.2	0.6	3.4	10.6	11.1	15.5	18.0	10.1	9.5	0.7	0.2	16.2	15.3	23.3	23.2
Cafeteria	9.3	8.2	6.7	5.1	17.2	17.7	26.0	26.1	3.8	5.9	4.2	4.2	21.1	20.5	29.4	28.5
Fruit	12.9	10.8	2.4	2.5	13.6	12.4	18.6	17.8	5.9	4.2	0.1	2.1	12.5	13.0	21.7	21.3
Wine	8.0	9.5	2.5	2.1	11.2	13.6	20.2	19.2	7.8	7.1	0.2	0.4	16.1	17.6	23.7	23.4
Bicycle	8.5	10.0	2.1	3.8	13.0	15.1	21.4	20.4	6.1	10.4	1.6	3.8	19.6	18.9	26.6	27.4
Orchid	12.4	13.3	1.3	3.9	9.8	11.1	15.4	18.9	8.2	11.7	−1.7	2.4	12.6	16.5	18.1	21.1
Musicians	4.5	3.7	5.2	4.2	17.7	17.0	23.7	25.3	−1.0	0.3	2.0	1.4	18.8	19.8	27.3	30.3
Candle	4.2	5.3	3.7	3.8	14.9	16.9	23.4	24.9	5.2	7.1	6.1	5.5	21.1	22.6	30.3	32.1
Average	*8.4*	*8.6*	*3.3*	*3.7*	*14.0*	*14.8*	*21.1*	*21.8*	*5.4*	*6.8*	*1.9*	*2.1*	*17.6*	*18.3*	*25.5*	*26.4*

images, the proposed method accelerates the codec by more than 12%. In general, images with fewer details reach the visually lossless threshold after coding fewer bitplanes, so the achieved speedup is higher.

5. Conclusions

BPC-PaCo is a bitplane coding engine for wavelet-based coding systems tailored for fine-grain parallelism in highly parallel architectures. Its main insight is to code multiple coefficients within a codeblock in parallel through vector instructions such as those found in modern CPUs and GPUs. Despite achieving high throughput, the bottleneck of BPC-PaCo originates in the use of three coding passes per bitplane, which causes register spilling since the coefficients are transferred back and forth from the memory to the register space of the processor. This shortcoming is alleviated with the introduction of a complexity scalable technique that safeguards quality scalability while accelerating the coding process. The drawback of CS BPC-PaCo is that the quality and speedup can only be roughly controlled via a user parameter. This paper introduces a visually lossless model for BPC-PaCo that, without needing the user supervision, maximizes both the throughput and the quality of the image that is visually relevant for a human observer.

VL BPC-PaCo is implemented for this paper in a Java framework to compare its coding performance and computational complexity with BPC-PaCo and CS BPC-PaCo. Experimental tests suggest that the proposed method significantly smooths the impact on quality when the image is progressively transmitted while accelerating the codec by about 8% on average. Future work will evaluate this algorithm on modern GPUs.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data are from public corpus ISO 12640-1.

Acknowledgments

This work has been partially supported by the Spanish Ministry of Science, Innovation and Universities (MICIU) and by the European Regional Development Fund (FEDER) under Grants RTI2018-095287-B-I00 and PID2021-125258OB-I00, by the Catalan Government under Grants 2018-BP-00008 and 2017SGR-463, and by the Horizon 2020 under the Marie Skłodowska-Curie grant agreement #801370.

References

[1] H. Chang, M. Varvello, F. Hao, S. Mukherjee, A tale of three videoconferencing applications: Zoom, Webex, and Meet, IEEE/ACM Trans. Netw. (2022) in press.
[2] Y. Al-Mulla, A. Al-Ruheili, A. Al-Lawati, K. Parimi, A. Ali, N. Al-Sadi, F. Al-Harrasi, Assessment of urban expansion's impact on changes in vegetation patterns in Dhofar, Oman, using remote sensing and GIS techniques, IEEE Access 10 (2022) 86782–86792.
[3] S.-H. Lee, S.-H. Yoon, H.-W. Kim, Prediction of online video advertising inventory based on TV programs: A deep learning approach, IEEE Access 9 (2021) 22516–22527.
[4] C. Yan, B. Gong, Y. Wei, Y. Gao, Deep multi-view enhancement hashing for image retrieval, IEEE Trans. Pattern Anal. Mach. Intell. 43 (2021) 1445–1451.
[5] C. Yan, Y. Hao, L. Li, J. Yin, A. Liu, Z. Mao, Z. Chen, X. Gao, Task-adaptive attention for image captioning, IEEE Trans. Circuits Syst. Video Technol. 32 (2022a) 43–51.
[6] C. Yan, L. Meng, L. Li, J. Zhang, Z. Wang, J. Yin, J. Zhang, Y. Sun, B. Zheng, Age-invariant face recognition by multi-feature fusion and decomposition with self-attention, ACM Trans. Multimedia Comput. Commun. Appl. 18 (2022b) 1–18.
[7] A. Said, W.A. Pearlman, A new, fast, and efficient image codec based on set partitioning in hierarchical trees, IEEE Trans. Circuits Syst. Video Technol. 6 (1996) 243–250.
[8] D. Taubman, High performance scalable image compression with EBCOT, IEEE Trans. Image Process. 9 (2000) 1158–1170.
[9] W.A. Pearlman, A. Islam, N. Nagaraj, A. Said, Efficient, low-complexity image coding with a set-partitioning embedded block coder, IEEE Trans. Circuits Syst. Video Technol. 14 (2004) 1219–1235.
[10] G. Xie, H. Shen, Highly scalable, low-complexity image coding using zeroblocks of wavelet coefficients, IEEE Trans. Image Process. 15 (2005) 762–770.
[11] M. Dyer, D. Taubman, S. Nooshabadi, A.K. Gupta, Concurrency techniques for arithmetic coding in JPEG2000, IEEE Trans. Circuits Syst. I 53 (2006) 1203–1212.
[12] M. Rhu, I.-C. Park, Optimization of arithmetic coding for JPEG2000, IEEE Trans. Circuits Syst. Video Technol. 20 (2010) 446–451.
[13] F. Auli-Llinas, M.W. Marcellin, Scanning order strategies for bitplane image coding, IEEE Trans. Image Process. 21 (2012) 1920–1933.
[14] X. Song, Q. Huang, S. Chang, J. He, H. Wang, Three-dimensional separate descendant-based SPIHT algorithm for fast compression of high-resolution medical image sequences, IET Image Process. 11 (2017) 80–87.
[15] A. Descampe, F.-O. Devaux, G. Rouvroy, J.-D. Legat, J.-J. Quisquater, B. Macq, A flexible hardware JPEG 2000 decoder for digital cinema, IEEE Trans. Circuits Syst. Video Technol. 16 (2006) 1397–1410.
[16] Y. Li, L. Claesen, K. Huang, M. Zhao, A real-time high-quality complete system for depth image-based rendering on FPGA, IEEE Trans. Circuits Syst. Video Technol. 29 (2019) 1179–1193.
[17] J.W. Park, H. Lee, B. Kim, D.-G. Kang, S.O. Jin, H. Kim, H.-J. Lee, A low-cost and high-throughput FPGA implementation of the retinex algorithm for real-time video enhancement, IEEE Trans. VLSI Syst. 28 (2020) 101–114.
[18] A.K. Gupta, S. Nooshabadi, D. Taubman, M. Dyer, Realizing low-cost high-throughput general-purpose block encoder for JPEG2000, IEEE Trans. Circuits Syst. Video Technol. 16 (2006) 843–858.
[19] K. Mei, N. Zheng, C. Huang, Y. Liu, Q. Zeng, VLSI design of a high-speed and area-efficient JPEG2000 encoder, IEEE Trans. Circuits Syst. Video Technol. 17 (2007) 1065–1078.
[20] M. Dyer, S. Nooshabadi, D. Taubman, Design and analysis of system on a chip encoder for JPEG2000, IEEE Trans. Circuits Syst. Video Technol. 19 (2009) 215–225.
[21] S. Kim, D. Lee, J.-S. Kim, H.-J. Lee, A high-throughput hardware design of a one-dimensional SPIHT algorithm, IEEE Trans. Multimedia 18 (2016) 392–404.
[22] H.-C. Fang, Y.-W. Chang, T.-C. Wang, C.-J. Lian, L.-G. Chen, Parallel embedded block coding architecture for JPEG 2000, IEEE Trans. Circuits Syst. Video Technol. 15 (2005) 1086–1097.
[23] Y. Li, M. Bayoumi, A three-level parallel high-speed low-power architecture for EBCOT of JPEG 2000, IEEE Trans. Circuits Syst. Video Technol. 16 (2006) 1153–1163.
[24] K. Sarawadekar, S. Banerjee, An efficient pass-parallel architecture for embedded block coder in JPEG 2000, IEEE Trans. Circuits Syst. Video Technol. 21 (2011) 825–836.
[25] Y. Jin, H.-J. Lee, A block-based pass-parallel SPIHT algorithm, IEEE Trans. Circuits Syst. Video Technol. 22 (2012) 1064–1075.
[26] P. Enfedaque, F. Auli-Llinas, J.C. Moure, Implementation of the DWT in a GPU through a register-based strategy, IEEE Trans. Parallel Distrib. Syst. 26 (2015) 3394–3406.
[27] P. Enfedaque, F. Auli-Llinas, J.C. Moure, GPU implementation of bitplane coding with parallel coefficient processing for high performance image compression, IEEE Trans. Parallel Distrib. Syst. 28 (2017) 2272–2284.
[28] M. Diaz, R. Guerra, P. Horstrand, E. Martel, S. Lopez, J. Lopez, R. Sarmiento, Real-time hyperspectral image compression onto embedded GPUs, IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens. 12 (2019) 2792–2809.
[29] C. de Cea-Dominguez, J.C. Moure, J. Bartrina-Rapesta, F. Auli-Llinas, GPU-oriented architecture for an end-to-end image/video codec based on JPEG2000, IEEE Access 8 (2020) 68474–68487.
[30] M.S. Nobile, P. Cazzaniga, A. Tangherloni, D. Besozzi, Graphics processing units in bioinformatics, computational biology and systems biology, Brief. Bioinform. 18 (2017) 870–885.
[31] Information technology - JPEG 2000 image coding system - part 1: Core coding system, 2000.
[32] Information technology - JPEG 2000 image coding system - part 15: High throughput JPEG 2000, 2019.
[33] A. Naman, D. Taubman, Decoding high-throughput JPEG2000 (HTJ2K) on a GPU, in: Proc. IEEE International Conference on Image Processing, 2019, pp. 1084–1088.
[34] D. Taubman, A. Naman, R. Mathew, High throughput block coding in the HTJ2K compression standard, in: Proc. IEEE International Conference on Image Processing, 2019, pp. 1079–1083.
[35] A. Naman, D. Taubman, Encoding high-throughput JPEG2000 (HTJ2K) images on a GPU, in: Proc. IEEE International Conference on Image Processing, 2020, pp. 1171–1175.
[36] C. de Cea-Dominguez, J.C. Moure, J. Bartrina-Rapesta, F. Auli-Llinas, Real-time 16K video coding on a GPU with complexity scalable BPC-PaCo, ELSEVIER Signal Process.: Image Commun. 99 (2021) 1–10.
[37] F. Auli-Llinas, Stationary probability model for bitplane image coding through local average of wavelet coefficients, IEEE Trans. Image Process. 20 (2011) 2153–2165.
[38] F. Auli-Llinas, Entropy-based evaluation of context models for wavelet-transformed images, IEEE Trans. Image Process. 24 (2015) 57–67.
[39] F. Auli-Llinas, P. Enfedaque, J.C. Moure, V. Sanchez, Bitplane image coding with parallel coefficient processing, IEEE Trans. Image Process. 25 (2016) 209–219.
[40] C. de Cea-Dominguez, J.C. Moure, J. Bartrina-Rapesta, F. Auli-Llinas, Complexity scalable bitplane image coding with parallel coefficient processing, IEEE Signal Process. Lett. 27 (2020) 840–844.
[41] A. Watson, G. Yang, J. Solomon, J. Villasenor, Visibility of wavelet quantization noise, IEEE Trans. Image Process. 6 (1997) 1164–1175.

[42] Z. Liu, L.J. Karam, A.B. Watson, JPEG2000 encoding with perceptual distortion control, IEEE Trans. Image Process. 15 (2006) 1763–1778.
[43] H. Oh, A. Bilgin, M. Marcellin, Visually lossless encoding for JPEG 2000, IEEE Trans. Image Process. 22 (2013) 189–201.
[44] L. Jimenez-Rodriguez, F. Auli-Llinas, M. Marcellin, Visually lossless strategies to decode and transmit JPEG2000 imagery, IEEE Signal Process. Lett. 21 (2014) 35–38.
[45] H. Oh, A. Bilgin, M. Marcellin, Visually lossless JPEG 2000 for remote image browsing, Information 7 (2016) 1–20.
[46] D.S. Taubman, M.W. Marcellin, JPEG2000 Image Compression Fundamentals, Standards and Practice, Kluwer Academic Publishers, Norwell, Massachusetts 02061 USA, 2002.
[47] A. Watson, Efficiency of a model human image code, J. Opt. Soc. Amer. 4 (1987) 2401–2417.
[48] S. Daly, The visible differences predictor: An algorithm for the assessment of image fidelity, in: Proc. SPIE Human Vision, Visual Processing and DigitalDisplay, 1992, pp. 2–15.
[49] F. Auli-Llinas, BOI codec, 2022, URL: https://deic.uab.cat/~francesc/software/boi.