Comparison of Code-Pass-Skipping Strategies for Accelerating a JPEG 2000 Decoder

Volker Bruns, Heiko Sparenberg
Moving Picture Technologies Department,
Fraunhofer Institute for Integrated Circuits
Germany, Erlangen 91058
{volker.bruns, heiko.sparenberg}@iis.fraunhofer.de

Abstract

Code-Pass-Skipping allows a JPEG 2000 decoder to be accelerated by sacrificing the output precision. This paper presents an evaluation on how the speed gain can be maximized and the quality loss minimized. In particular, the scenario of rendering a 24-bit preview of a Digital Cinema Package (DCP) with the maximum permitted bitrate is examined. A comparison shows that a new proposed strategy outperforms the reference implementation from *Kakadu Software* v6 by up to 1 dB. Furthermore, it is shown what speed gain can be achieved for a given acceptable quality loss.

1. Motivation

JPEG 2000 is a standard for still image compression, published jointly by ISO/IEC and ITU-T [1]. It is most prominently used as the compression format in Digital Cinema Packages (DCPs) but also finds applications in the fields of medical imaging and industrial sensor systems. It performs superior to JPEG in most - if not all – aspects, but this advantage comes at the cost of a significantly higher computational complexity. The maximum bitrate for the image essence of DCPs was specified to be 250 Mbit/s [2]. Cinema servers and also some postproduction workstations are equipped with dedicated hardware-boards, that are capable of processing video streams in real-time. For quality control and screenings, however, software-based solutions for PC or Mac are being used. The particular hardware-recommendations vary, but modern CPUs with four or more cores are required to guarantee decoding times that are sufficiently fast for real-time playback, even at the maximum bitrates of 250 Mbit/s. When the currently discussed amendment for higher frame rates with up to 60 fps per eye and a maximum bitrate of 500 Mbit/s is passed, the computational requirements will rise even higher.

For applications other than the theatrical playback itself, a decoding in full quality is not necessarily required. Smooth real-time playback, however, is a must. A simple solution would be to utilize the resolution scalability built into JPEG 2000 and discard the outermost resolution level. The decoding time would decrease noticeably but the spatial resolution would be halved both horizontally and vertically. Especially when viewing a DCP with a consumer-level or semi-professional LCD display or projector, this is not the smartest trade-off as the devices would in fact be capable of rendering the full resolution. A better approach would be to sacrifice part of the 36-bit precision, taking into account that the viewing devices can only render 24-bit colors anyway. The JPEG 2000 standard does accommodate for scalability by quality, but this feature is not permitted in the profiles specified by the Digital Cinema Initiative (DCI) for use in DCPs. The concept of „Code-Pass-Skipping“ does not pose any specific requirements on the encoded code-stream, however, and can thus be used to diminish precision in order to gain speed anyway. This way, a finely tunable compromise between quality and speed can be realized. It is desirable to implement the concept in a way that maximizes the speed gain and minimizes the quality loss. This paper examines several alternative strategies. Furthermore, it is evaluated what speed gain can be achieved when given a maximally tolerable quality loss measured in Peak-Signal-to-Noise-Ratio (PSNR).

2. Structure

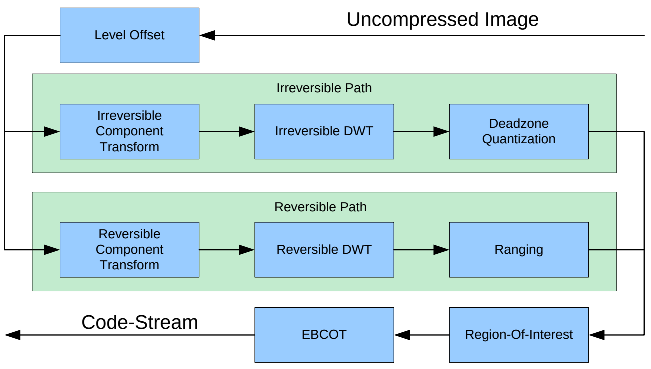

Figure 1 Block diagram of a JPEG 2000 encoder

For a better understanding, the relevant image compression algorithms used in JPEG 2000 will be presented briefly. Then, code-pass skipping will be explained along with an exemplary code-pass-histogram. Next, alternative strategies proposed by the authors are presented. Finally, the different strategies are compared and the results discussed. First, a JPEG 2000 encoder (Figure 1) transforms an uncompressed input image into a color representation that distinguishes between luminance and chrominance. Next, each color component individually undergoes a wavelet transform.

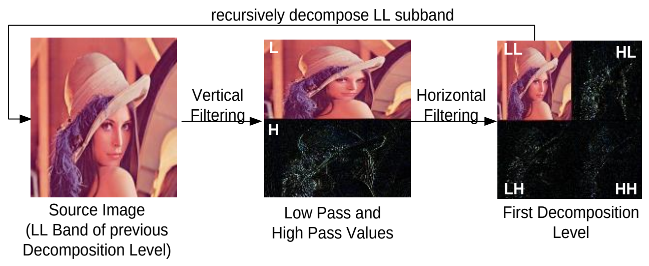
recursively decompose LL subband
Vertical Filtering
Horizontal Filtering
L
H
LL
HL
LH
HH
Source Image (LL Band of previous Decomposition Level)
Low Pass and High Pass Values
First Decomposition Level

Figure 2 2D Dyadic wavelet decomposition

In this step, the spectrum is separated into low and high frequencies, both in horizontal and vertical direction. Applying the transform once yields four subbands, denoted as LL, HL, LH and HH (Figure 2). Recursively, this procedure can be executed multiple times on the resulting LL subbands. For a DCI-compliant JPEG 2000 code-stream the transform is typically repeated five times, which yields six resolution levels with three subbands in each level, respectively four subbands in the lowest level.

Following, wavelet coefficients are quantized, with the quantization factor being dependent on the subband level. This way, higher frequencies can be quantized more heavily than lower frequencies. Subsequently, each subband is split into independent code-blocks, which are specified to be 32x32 pixels in the DCI profiles. Each code-block is then entropy coded.

JPEG 2000's entropy coder is named *Embedded Block Coder with Optimized Truncation* (EBCOT). The wavelet coefficients are first converted into a sign/magnitude representation and then processed bit-plane by bit-plane, starting with the most significant one (Figure 3). Each plane is traversed in a zigzag pattern in three passes. In exactly one of the three pass types a bit is passed to the context-adaptive arithmetic coder, which successively generates a bit-stream for the code-block. Additionally, a rate-distortion value pair is calculated after every completed code-pass. This value allows making a statement on how much the inclusion of this code-pass's information would raise the final code-stream's size (cost) and decrease the difference of the compressed image to the original one (benefit). Based on these values, the subsequent *Post-Compression Rate-Distortion Optimization* (PCRD-Optimization) algorithm decides which passes to remove in order to meet a user-defined target bit-rate, e.g. 250 Mbit/s. Consequently, it is not unordinary that more code-passes are spent for one code-block than for another, even if they stem from the same subband and are thus coded with the same nominal bit depth.

3. Code-Pass-Skipping

Since a JPEG 2000 code-stream is constructed in a way that it can be processed sequentially, it is assumed that the decoder will not load the entire code-stream into memory and therefore cannot access all packets at once. Each code-block is decoded independently. A code-stream packet's header contains the information how many code-passes have been included for the current code-block as well as the bit-depth that was used to store the wavelet coefficients. Without code-pass-skipping, a decoder would simply process all code-passes in order to reconstruct the coefficients as precisely as possible. Only after the wavelet synthesis and inverse color transform during the denormalization would the superfluous precision be truncated, when only a 24-bit presentation of the image is reconstructed.

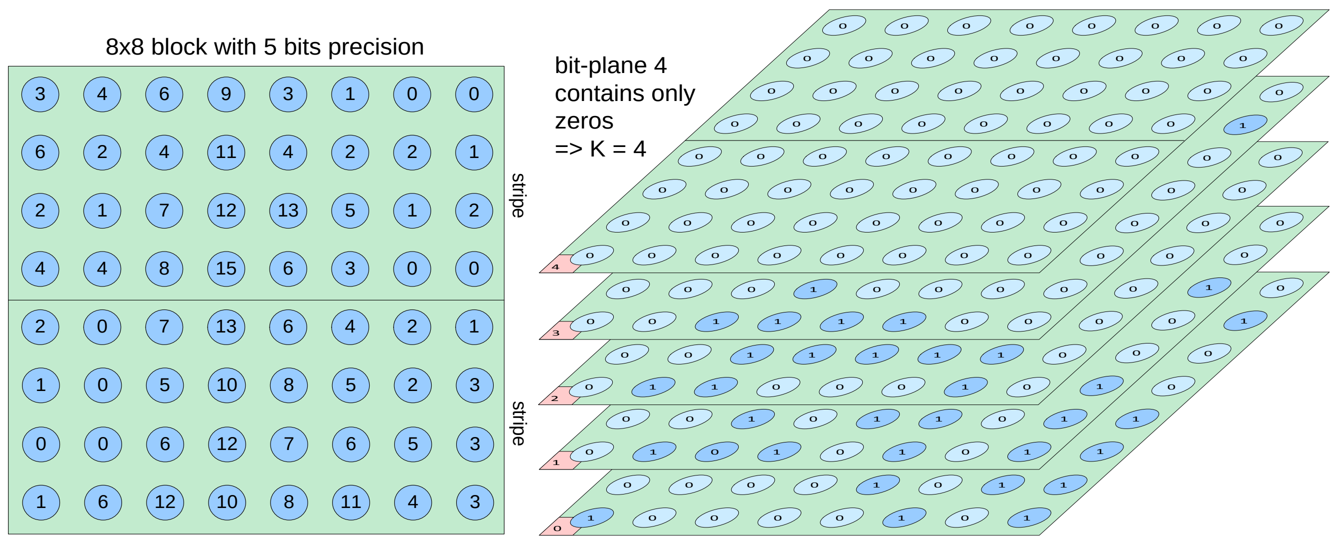

Figure 3 Bit-plane representation of an 8 x 8 block of quantization index magnitudes. An additional plane not included in the diagram contains the samples' signs.

Table 1 Code-Pass Histogram, 00077.tif.250mbits.j2c

Bit-plane	Code-Pass	Total	rlvl 0	rlvl 1	rlvl 2	rlvl 3	rlvl 4	rlvl 5
Σ	Σ	48254	180	441	1525	5232	14648	26204
16	46	0	0	0	0	0	0	0
	45	0	0	0	0	0	0	0
15	44	0	0	0	0	0	0	0
	43	42	0	0	0	0	42	0
	42	42	0	0	0	0	42	0
14	41	42	0	0	0	0	42	0
	40	273	2	3	11	22	184	51
	39	273	2	3	11	22	184	51
13	38	273	2	3	11	22	184	51
	37	706	4	5	19	73	319	286
	36	706	4	5	19	73	319	286
12	35	706	4	5	19	73	319	286
	34	1601	6	7	25	99	598	866
	33	1601	6	7	25	99	598	866
11	32	1601	6	7	25	99	598	866
	31	3069	6	10	38	153	892	1970
	30	3053	6	10	38	153	892	1954
10	29	3052	6	10	38	153	892	1953
	28	4183	6	16	55	232	1003	2871
	27	4130	6	16	55	232	1003	2818
9	26	4053	6	16	55	232	1003	2741
	25	4004	6	16	69	280	1008	2625
	24	3135	6	16	69	280	1008	1756
8	23	2893	6	16	69	280	897	1625
	22	2661	6	18	72	288	897	1380
	21	2176	6	18	72	288	896	896
7	20	947	6	18	72	288	560	3
	19	655	6	18	72	288	268	3
	18	384	6	18	72	288	0	0
6	17	384	6	18	72	288	0	0
	16	384	6	18	72	288	0	0
	15	384	6	18	72	288	0	0
5	14	369	6	18	72	273	0	0
	13	174	6	18	72	78	0	0
	12	96	6	18	72	0	0	0
4	11	96	6	18	72	0	0	0
	10	34	6	18	10	0	0	0
	9	24	6	18	0	0	0	0
3	8	24	6	18	0	0	0	0
	7	24	6	18	0	0	0	0
	6	0	0	0	0	0	0	0
2	5	0	0	0	0	0	0	0
	4	0	0	0	0	0	0	0
	3	0	0	0	0	0	0	0
1	2	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0

For *Kakadu Software*, it has been shown that EBCOT Tier-1 is the most computationally expensive step of a JPEG 2000 decoder and takes up to two thirds of the entire decoding time [6]. This is exactly where Code-Pass-Skipping takes effect by skipping additional code-passes in the decoder. Mathematically, this can be regarded as another quantization. The more code-passes are dropped, the less time is required for EBCOT Tier-1, the more the wavelet coefficients are quantized and the larger the difference between the reconstructed image and the original becomes. So more precisely the research question is what strategy to employ in a decoder to decide which code-passes to drop. This task is very similar to that of the PCRD-Optimization algorithm in the encoder, but with the significant difference, that the decoder does not have rate-distortion values at its hand to base its decision on.

Table 2 Code-Pass Histogram, 00077.tif.75mbits.j2c

Bit-plane	Code-Pass	Total	rlvl 0	rlvl 1	rlvl 2	rlvl 3	rlvl 4	rlvl 5
Σ	Σ	20792	164	379	1268	3971	8495	6515
[…]	[…]	[…]	[…]	[…]	[…]	[…]	[…]	[…]
	27	1001	6	16	55	232	541	151
9	26	715	6	16	55	232	401	5
	25	708	6	16	69	280	337	0
	24	707	6	16	69	280	336	0
8	23	516	6	16	69	280	145	0
	22	521	6	18	72	288	137	0
	21	384	6	18	72	288	0	0
7	20	373	6	18	72	277	0	0
	19	366	6	18	72	270	0	0
	18	201	6	18	72	105	0	0
6	17	192	6	18	72	96	0	0
	16	163	6	18	72	67	0	0
	15	99	6	18	72	3	0	0
5	14	50	6	18	26	0	0	0
	13	36	6	18	12	0	0	0
	12	27	6	18	3	0	0	0
4	11	12	6	6	0	0	0	0
	10	3	1	2	0	0	0	0
	9	3	1	2	0	0	0	0
3	8	0	0	0	0	0	0	0
	7	0	0	0	0	0	0	0
	6	0	0	0	0	0	0	0
2	5	0	0	0	0	0	0	0
	4	0	0	0	0	0	0	0
	3	0	0	0	0	0	0	0
1	2	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0

4. Code-Pass-Histogram

In order to gain an overview of the code-passes present in a test image and infer strategies from it, it is helpful to create a code-pass-histogram [Table 1]. Every possible code-pass-index is regarded as a bin and it is counted for how many code-blocks this particular pass (type and bit-plane) has to be executed. The code-block count is additionally subdivided by resolution-levels. In this particular example, the decoder reconstructs wavelet coefficients with 16 bits precision. Thus, the highest pass index is 3x16-2 = 46. Initially, a coefficient is zero and then it's precision is successively improved by processing more and more code-passes, beginning with the most significant non-zero bit-plane and moving towards the lower bit-planes. Every code-pass may contain the information to toggle the appropriate bit from 0 to 1. When employing code-pass-skipping, some code-passes are truncated, which effectively leads to the less significant bits staying zero. Consequently, the difference between the reconstructed coefficient and the coded coefficient may grow with every additional ignored code-pass.

It is visible (and highlighted by the horizontal bold border) that subbands in the various resolution levels from lowest (rlvl 0) to highest (rlvl 5) frequencies have been coded with decreasing bit-depths of 14,14,13,12,10, respectively 10 bits. Any bit planes lower than these bit-depths will stay zero. For all code-blocks except a few from rlvl 4 the top-most two bit-planes consist entirely of zero bits, so that

they have not been processed in code-passes but instead been flagged as zero-bit-planes in the packet header.
In the example shown in Table 1, the lowest resolution level (rlvl 0) consists of six code-blocks all of which utilize the maximum available precision of 14 bits (bit-planes 3-16). Apparently the encoder did not strip any code-passes for these blocks because the target bitrate was met to begin with.

Table 2 shows the code-passes of the same image compressed at only 75 Mbit/s (at 24 fps). Now, the encoder had to drop numerous code-passes in order to meet the bitrate limit. This can be inferred from the lower code-pass-counts in the less significant bit-planes.

5. Strategies

Next, the strategy implemented by *Kakadu Software* v6 will be described and alternative algorithms proposed by the authors will be presented. Last, all strategies will be evaluated and compared against each other.

5.1. Strategy kdu standard

Kakadu Software offers the possibility to enable code-pass-skipping and configure it by specifying a truncation factor. For a factor of 0, no passes are skipped. For every multiple of 256 one additional code-pass in each code-block is skipped. For intermediate values, only some of the code-blocks, starting with the high frequencies, loose an additional code-pass. The flaw of this strategy might be that the number of existing code-passes is not taken into consideration. Regardless of this number, all code-blocks are truncated equally strong for every multiple of 256.

5.2. Strategy Block-Pass-Count-Relative

The number of skipped code-passes depends on the total number of passes present for a code-block. By specifying a quality index, the quality-to-speed trade-off can be configured. A range of 0-100 quality points is mapped linearly to 0% – 70% of the total number of passes present for a code-block. In effect, a code-block with 30 passes loses more passes than a code-block for which only 10 passes exist. It is not expected that this strategy will deliver the best results, because in this example the encoder must have had a good reason to spend 30 passes on one block and only 10 on the other. Apparently, the former block contains very important information. However, according to this strategy, the more important code-block will be truncated more heavily than the less important one.

5.3. Strategy Subband-Relative

The number of skipped code-passes depends solely on the bit-depth of the subband the code-block belongs to. If a subband has K_{max} bits, a code-block within this band can have $3 \times K_{max} - 2$ passes at most. This maximum number is decreased and only those blocks affected by the lower limit are truncated. A range of 0-100 quality points is mapped to the passes between a deliberate minimum index ($3 \times K_{max} - 2 - 17$) and the maximum index possible for the subband ($3 \times K_{max} - 2$). Decreasing the quality index equally affects all code-blocks. In this strategy, the relation between the numbers of code-passes for blocks from the same subband will be maintained. Hence, the encoder's PCRD-Optimization algorithm's decisions are not overruled, but rather maintained. Furthermore, code-passes are stripped in equal amounts from all resolution levels.

Table 3 PSNR-values without code-pass-skipping (Data-rates with respect to 24 fps)

	250 Mbit/s	150 Mbit/s	75 Mbit/s
Image 3837	41,8 dB	37,7 dB	35,4 dB
Image 1278	39,0 dB	35,7 dB	33,9 dB
Image 0077	40,2 dB	37,0 dB	34,5 dB

Table 4 PSNR-values of the intersection points in Figure 5 (Data-rates with respect to 24 fps)

	250 Mbit/s	150 Mbit/s	75 Mbit/s
Image 3837	36,3 dB	35,2 dB	34,3 dB
Image 1278	34,8 dB	33,9 dB	32,8 dB
Image 0077	35,5 dB	33,4 dB	32,7 dB

5.4. Strategy Image-Relative

The number of skipped code-passes is calculated with respect to maximum bit-depth of all subbands. Usually the maximum bit-depth should be that of the lowest resolution level, which contains the image's important low frequencies. A precision of 14 bits yields a maximum number of code-passes of $3 \times 14 - 2$. This maximum is now decreased so that only those code-blocks affected by the lower maximum will be affected. Again, a deliberate minimum number of $3 \times K_{max}^{max} - 2 - 17$ is introduced to prevent the quality from deteriorating too much. This strategy leads to low-frequency code-blocks being truncated more significantly than high-frequency code-blocks and therefore it is not expected that it will deliver very good results.

5.5. Modified Strategies

Any of the strategies can easily be modified by preventing that code-blocks from certain subbands will be truncated. This way, high or low frequencies can be spared.

6. Test Setup

All strategies were implemented in *Kakadu Software* v6 and can be selected to replace the original strategy. They were applied to a set of three test images (Figure 4), each encoded with 75 Mbit/s, 150 Mbit/s and 250 Mbit/s (with respect to 24 fps).

The quality index is varied between 100 (highest quality) and 0 (lowest quality) in incremental steps of 10 points.
The decoding time (mean of three repetitions) is measured

Figure 4 Images 0077, 1278, 3837 (top to bottom)

for each of the nine source code-streams. *Kakadu Software*'s multi-threading engine was disabled. Additionally, the PSNR-value between the 24-bit reconstructed image and the 48-bit original is calculated. For this, the Mean-Square-Error (MSE) is calculated based on the normalized floating-point representations (value range 0.0 – 1.0) of the images. Subsequently, a maximum value of 28-1 = 255 is used for calculating the PSNR-value.

7. Evaluation

Figure 5 shows the results for a 1.24 MB large test image. The PSNR-value for the completely decoded 24-bit image compared to the 48-bit original is 41.8 dB. On an Intel i7 3960x CPU, the decoding time (with multi-threading disabled) is 234 msecs. Up to a decoding time of only 100 msecs, the PSNR-value decreases fairly linearly to 36.5 dB. In other words, a speed-up of 25 msecs costs about 1 dB quality.

Two of the strategies proposed by the authors slightly outperform the reference from *Kakadu Software* v6, but only up to a deterioration of 5 dB. They achieve a higher quality while providing the same speed-up.

Up to a deterioration of 4 dB the graphs have a fairly constant offset and after that they approximate each other until they cross. This offset is 0.4 – 0.5 dB for the "Block-Pass-Count-Relative"-strategy. For the "Subband-Relative"-strategy the PSNR-value is even 0.9 – 1.1 dB higher than that of the reference implementation. By modifying the latter strategy to spare code-blocks from the lowest three resolution levels, the deterioration of the PSNR-value for high speedups can be further decreased. However, at a speed-up of 100 msecs and a PSNR-value of 36.2 dB the curve falls below that of the reference implementation. The "Image-Relative"-strategy performs constantly poorer than all other strategies, except for the one, which spares not the lower, but the higher frequencies. As expected, this strategy is a negative example and proves that the lower frequency wavelet coefficients have a very strong influence on the PSNR-value and should be advantaged rather than dis-advantaged.

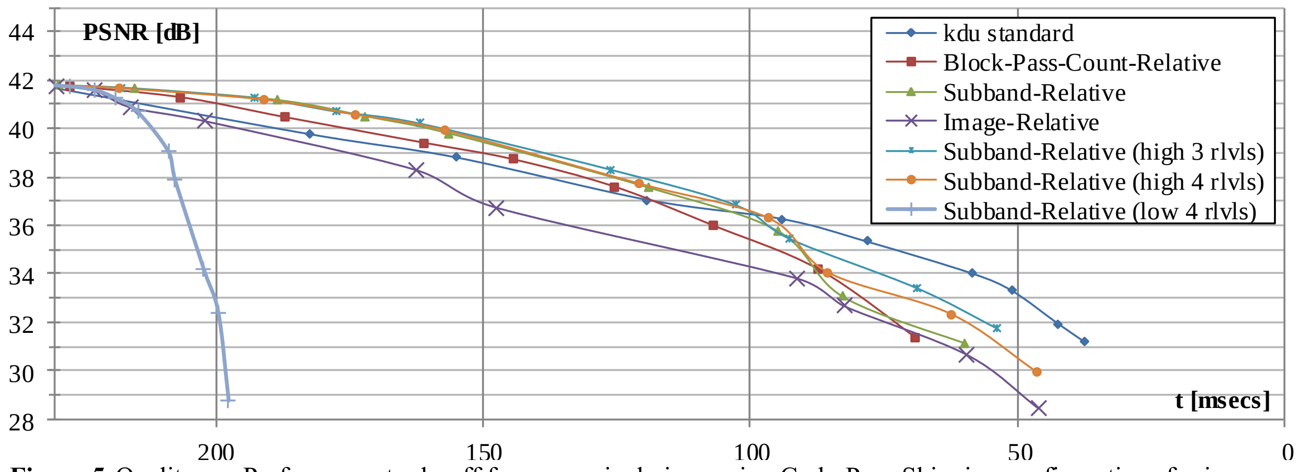

Figure 5 Quality-vs.-Performance trade-off for successively increasing Code-Pass-Skipping configurations for image 3837 (DCI-compliant, 250 Mbit/s with respect to 24 fps, 24-bit decode)

The graph depicted in figure 5 looks very similar for the other two test images, encoded at the same bitrate. They are merely shifted on the y-axis as not all compressed images achieve the same maximum PSNR-value. At full quality, the PSNR-values of the 24-bit reconstructed images compared to the original ones are 41.8 dB, 39.0 dB, and 40.2 dB (refer table 3).

The strategies' ranking list looks identical for the more heavily compressed test images. Due to the lower data-rate, the overall achieved PSNR-values and execution times are lower, which was expected. The points at which the proposed strategies "Subband-relative" and "Image-relative" intersect with the reference strategy from *Kakadu Software* are located at 4.2 – 5.5 dB (250 Mbit/s), 1.8 – 3.6 dB (150 Mbit/s), respectively 1.0 – 1.8 dB (75 Mbit/s) below the maximum PSNR-values. This equals absolute values of 32.7 – 36.3 dB for the different bitrates (refer table 4). For test images encoded at 250 Mbit/s, this translates to skipping more than half of all the code-passes and for 75 Mbit/s still one third of all code-passes are skipped. Therefore, applications in which qualities as low as that are still acceptable should employ a hybrid strategy.

Both the "Subband-Relative" and "Image-Relative" strategies calculate the number of skipped passes independently of the total number of existing passes and thus take less effect at high quality indices for the test images compressed at 150 Mbit/s and 75 Mbit/s. This behavior is desirable since it allows parameterizing code-pass-skipping in a way that only high-data code-streams are truncated. Low-data scenes, on the other hand, take less decoding-time anyway and are not further diminished unnecessarily.
A DCP-Player can utilize the a-priori knowledge that all sources have the same bit-depth, i.e. 36-bit. Even if this assumption could not be made, the strategies could easily be modified to work with a global K_{max} value, specified by the application instead of the source. This way, a source with a lower bit-depth would only be truncated, if a very low quality index were configured. The quality parameter would be independent of the source's bit-depth, which again is a desirable behavior as the user should not have to adjust the quality-vs.-speed trade-off for every source. Rather, the trade-off should be dependent on the available computational power, which stays constant.

8. Conclusion and Outlook

It has been shown, that code-pass-skipping provides a significant speed-up at quality losses that are tolerable in many applications. While providing the same speed-up, the "Subband-Relative"-strategy diminishes the quality-loss significantly, compared to the reference implementation from *Kakadu Software* v6. For DCI-compliant JPEG 2000 code-streams that fully utilize the maximum permitted bitrate of 250 Mbit/s (with respect to 24 fps) the reference is outperformed by up to 1.1 dB. When skipping more than half of the existing code-passes, however, the reference implementation delivers best results, which is why a hybrid strategy should be pursued, when skipping that many code-passes can still be tolerated.

Furthermore, the test results show what speed-up can be achieved for a given tolerable quality loss (measured in PSNR). The decision on what quality loss is still tolerable can be forwarded to the user, e.g. in the form of a quality-vs.-performance slider. For the proposed strategies no knowledge of the source's bit-depth is required, so that the user only needs to adjust the code-pass-skipping configuration once to the available computation power. The fact that the best-performing strategy's curve initially stays level shows that when reconstructing only a 24-bit preview from a 36-bit source, e.g. a DCP, code-pass-skipping can be utilized to achieve a significant speed-up at almost no cost. In combination with a high-performance decoder like *Kakadu Software* the hardware requirements for real-time playback of DCPs can be decreased. Alternatively, a powerful hardware enables real-time playback of bitrates beyond 250 Mbit/s or frame-rates higher than 24 fps.

9. References

[1] ISO/IEC 15444-1, Information technology - JPEG 2000 image coding system - Part 1: Core coding system, 2000

[2] Digital Cinema Initiatives (DCI), Digital Cinema System Specification V1.2, 7th March 2008

[3] Kakadu Software, http://www.kakadusoftware.com/

[4] Morgan Multimedia M-JPE2000 Codec, http://www.morgan-multimedia.com/morgan/php/products.php?sProductId=5&sProductSub=screenshots

[5] MainConcept JPEG2000 Codec, http://www.mainconcept.com/products/sdks/video/jpeg-2000.html

[6] Dyer, M., Gupta, A., Galin, N., Nooshabadi, S., Case Study: Hardware Acceleration of the JPEG2000 Kakadu Library, 49th IEEE International Midwest Symposium on Circuits and Systems, 2006

[7] Taubman, D. S., Marcellin, M. W., JPEG2000: Image compression fundamentals, standard and practice, Kluwer, Boston, 2002