OPTIMIZED AND PARALLELIZED PROCESSING ORDER FOR IMPROVED FREQUENCY SELECTIVE SIGNAL EXTRAPOLATION

Jürgen Seiler and André Kaup

Chair of Multimedia Communications and Signal Processing,
University of Erlangen-Nuremberg, Cauerstr. 7, 91058 Erlangen, Germany
{seiler, kaup}@LNT.de

ABSTRACT

In the recent years, multi-core processor designs have found their way into many computing devices. To exploit the capabilities of such devices in the best possible way, signal processing algorithms have to be adapted to an operation in parallel tasks. In this contribution an optimized processing order is proposed for Frequency Selective Extrapolation, a powerful signal extrapolation algorithm. Using this optimized order, the extrapolation can be carried out in parallel. The algorithm scales very good, resulting in an acceleration of a factor of up to 7.7 for an eight core computer. Additionally, the optimized processing order aims at reducing the propagation of extrapolation errors over consecutive losses. Thus, in addition to the acceleration, a visually noticeable improvement in quality of up to 0.5 dB PSNR can be achieved.

1. INTRODUCTION

Signal extrapolation is a very important task in image and video signal processing. In this process, a signal is extended from regions where the signal is known into regions where no information about the signal is available. An important application for signal extrapolation algorithms is e. g. the concealment of image distortions in the case that errors occur during the transmission of compressed image and video signals. Concealment on the one hand side is important for providing a decent quality to the viewer, even if parts of an image are distorted. On the other hand side, error concealment is necessary for reducing error propagation if a predictive coding scheme is used. In [1, 2] a good overview of the importance of error concealment and different error concealment techniques for images and videos can be found.

An algorithm that is suited well for such tasks is the Frequency Selective Extrapolation (FSE) from [3]. This algorithm generates a model of the signal to be extrapolated as weighted superposition of basis functions. Unfortunately, the algorithm as it is proposed up to now is not suited for a parallelized execution. But, in the recent years a trend has evolved to use multi-core designs for CPUs, DSPs or GPUs [4, 5]. These devices have in common that they follow a shared memory model and are able to run several threads in parallel, operating on the same data set.

In the scope of this paper we propose an optimized processing order for FSE. This novel order has the advantage that a parallelized execution is possible, and, in addition to that, an improved extrapolation quality is achievable. Before the novel processing order is introduced in detail, FSE is briefly reviewed in the next section for providing an overview of the algorithm and for identifying the important properties that have to be taken into account for the improved and parallelizable processing order.
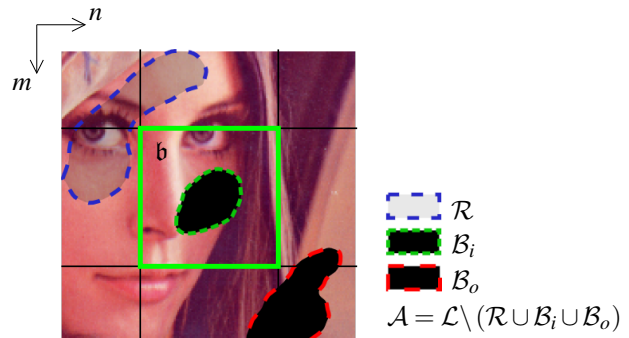

Figure 1: *Extrapolation area $\mathcal{L}$ for extrapolation in block ♭. $\mathcal{L}$ consists of reconstructed area $\mathcal{R}$ of previously extrapolated samples, loss area $\mathcal{B}_i$ of unknown samples within block ♭, loss area $\mathcal{B}_o$ of unknown samples outside block ♭, and support area $\mathcal{A}$ of all originally known samples.*

2. FREQUENCY SELECTIVE EXTRAPOLATION

For carrying out the Frequency Selective Extrapolation from [3], the image to be processed is divided into blocks. The block actually being processed is denoted by ♭. If this block contains areas that have to be extrapolated, the block itself and a frame of d samples width around the block are taken. This union is called extrapolation area $\mathcal{L}$ and consists of different subareas. All known samples are subsumed in support area $\mathcal{A}$, all unknown samples which are inside block ♭ are subsumed in loss area $\mathcal{B}_i$, and the unknown samples outside block ♭ in loss area $\mathcal{B}_o$. Samples that have already been extrapolated before can be used for extrapolating subsequent blocks and are subsumed in reconstructed area $\mathcal{R}$. Figure 1 shows the relation of the different areas with respect to the actually regarded block ♭. Altogether, extrapolation area $\mathcal{L}$ is depicted by spatial coordinates m and n and is of size $M \times N$.

In order to extrapolate the unknown samples, FSE iteratively generates the model

$$g[m,n] = \sum_{k \in \mathcal{K}} \hat{c}_k \varphi_k[m,n] \qquad (1)$$

of the signal in whole area $\mathcal{L}$ as weighted superposition of the basis functions $\varphi_k[m,n]$. Set $\mathcal{K}$ contains the indices of all basis functions used for model generation and the expansion coefficients $\hat{c}_k$ control the weight of each basis function. In every iteration, FSE selects one basis function to add to the model generated so far and estimates the corresponding expansion coefficient. Finally, the samples in area $\mathcal{B}_i$ are taken from the model and are used for extrapolation. For a detailed

discussion of the original FSE and the improved expansion coefficient estimation by Orthogonality Deficiency Compensation, please refer to [3, 6].

In order to control the influence each sample has on the model generation, FSE utilizes the weighting function

$$
w[m,n] = \begin{cases} \hat{\rho}^{\sqrt{\left(m-\frac{M-1}{2}\right)^2 + \left(n-\frac{N-1}{2}\right)^2}} & \text{for } (m,n) \in \mathcal{A} \\ \delta\hat{\rho}^{\sqrt{\left(m-\frac{M-1}{2}\right)^2 + \left(n-\frac{N-1}{2}\right)^2}} & \text{for } (m,n) \in \mathcal{R} \\ 0 & \text{for } (m,n) \in \mathcal{B}_i \cup \mathcal{B}_o \end{cases}
$$
(2)

within the iteration loop to assign a certain weight to each sample, depending on its position. Using $w[m,n]$, samples far away from the actual block $\mathfrak{b}$ get an exponentially decreasing weight with increasing distance, controlled by decay factor $\hat{\rho}$. As the samples within $\mathcal{B}_i$ and $\mathcal{B}_o$ are unknown, they cannot contribute to the model generation and have to be weighted with 0. Already reconstructed samples from neighboring blocks can also contribute to the model generation. Since these samples are not as reliable as the original ones, the exponentially decreasing weight is further attenuated by a factor δ between 0 and 1 in area $\mathcal{R}$. This is required for reducing the influence of already reconstructed samples on the model generation and allaying error propagation.

FSE, as it is proposed up to now, extrapolates the individual blocks of an image in line scan order. This processing order has two disadvantages if consecutive losses occur. First, by just extrapolating a block without examining its neighborhood, it may be possible that blocks get extrapolated that have only few known or already reconstructed samples adjacent to them. These blocks can only be extrapolated poorly and in addition to that, the low extrapolation quality propagates to adjacent not yet extrapolated blocks. Second, using line scan order, consecutive loss areas are processed from left to right and top to bottom. But, as already reconstructed samples are less reliable than the original ones, it is beneficial to close large consecutive loss areas from all directions in order to reduce error propagation. In order to resolve these drawbacks we subsequently propose a novel processing order for FSE. This optimized processing order is able to exploit the known samples more effectively for model generation, yielding an improved extrapolation quality. Additionally, it allows for a parallel extrapolation of different blocks, leading to an improved extrapolation speed on multi-core devices.

3. OPTIMIZING THE PROCESSING ORDER

For optimizing the order in which the individual image blocks are processed, the properties of FSE have to be taken into account. First of all, extrapolation quality generally increases with an increasing number of known samples in the neighborhood. Due to the reuse of already extrapolated samples, this also means that a block should not be processed until as many as possible of its neighboring blocks are available. In addition to that, it is not advisable to process neighboring blocks at the same time, as the extrapolation results from one block then cannot be used for improving the model generation of the other one.

In order to obey these requirements we propose the following algorithm for determining the processing order. The main criterion for selecting a block to extrapolate is the number $\mathcal{N}(\mathfrak{b})$ of not yet extrapolated blocks adjacent to block $\mathfrak{b}$. A block is regarded as not extrapolated if it contains a loss

---

Algorithm 1 *Frequency Selective Extrapolation with optimized processing order. Function* `neighbors(b)` *returns indices of blocks spatially adjacent to block* $\mathfrak{b}$*,* $\mathcal{N}$ *holds the number of not extrapolated neighboring blocks for block* $\mathfrak{b}$*.*

input: Input signal, divided in blocks
/* Initialization */
for all blocks $\mathfrak{b}$ **do**
   if block $\mathfrak{b}$ has samples to extrapolate **then**
      $\mathcal{N}(\text{neighbors}(\mathfrak{b})) = \mathcal{N}(\text{neighbors}(\mathfrak{b})) + 1$
   end if
   if block $\mathfrak{b}$ is margin block **then**
      $\mathcal{N}(\mathfrak{b}) = \mathcal{N}(\mathfrak{b}) + 3$
   end if
   if block $\mathfrak{b}$ is corner block **then**
      $\mathcal{N}(\mathfrak{b}) = \mathcal{N}(\mathfrak{b}) + 5$
   end if
end for
for all blocks $\mathfrak{b}$ **do**
   if block $\mathfrak{b}$ has no samples to extrapolate **then**
      $\mathcal{N}(\mathfrak{b}) = -1$
   end if
end for
/* Extrapolation */
while not all blocks finished **do**
   $\mathcal{N}_{\min} = \min(\max(0, \mathcal{N}(\mathfrak{b})))$
   $\mathcal{S} = \{\}$
   for all blocks $\mathfrak{b}$ **do**
      if $\mathcal{N}(\mathfrak{b}) == \mathcal{N}_{\min}$ `&&` neighbors$(\mathfrak{b}) \notin \mathcal{S}$ **then**
         $\mathcal{S} = \mathcal{S} \cup \mathfrak{b}$
      end if
   end for
   /* From here on: **parallel** execution possible */
   for all blocks $\mathfrak{b} \in \mathcal{S}$ **do**
      Extrapolation for block $\mathfrak{b}$
      Insert extrapolated samples in block $\mathfrak{b}$
      $\mathcal{N}(\mathfrak{b}) = -1$
      $\mathcal{N}(\text{neighbors}(\mathfrak{b})) = \mathcal{N}(\text{neighbors}(\mathfrak{b})) - 1$
   end for
end while
output: Extrapolated signal

---

region that has not been extrapolated before, i. e. if $\mathcal{B}_i \neq \{\}$ for the regarded block. Contrary, a block is regarded as extrapolated if all samples are originally known or if the unknown samples have been extrapolated before. So, to start the extrapolation for an image, the number of not yet extrapolated neighboring blocks is determined for every block. Initially, $\mathcal{N}(\mathfrak{b})$ is set to 0 for all blocks. Next, the image is scanned block by block and if a block $\mathfrak{b}$ contains a loss area, $\mathcal{N}(\text{neighbors}(\mathfrak{b}))$ of all blocks adjacent to $\mathfrak{b}$ is increased by 1. In doing so, function `neighbors(b)` returns the indices of the blocks spatially adjacent to block $\mathfrak{b}$. In addition to that, if a block $\mathfrak{b}$ is located at the corner of the image, $\mathcal{N}(\mathfrak{b})$ is further increased by 5, if a block lies at the margin of the image, $\mathcal{N}(\mathfrak{b})$ is increased by 3. This is due to the fact that the number of neighboring blocks is smaller for blocks that lie at the outer rim of the image. In a second blockwise scan pass, $\mathcal{N}(\mathfrak{b})$ is set to -1 for all image blocks $\mathfrak{b}$ that do not contain any unknown samples. With this, all blocks for which no operation is required can be marked.

Subsequent to this, a loop is started that lasts until all image blocks that contain loss areas are extrapolated, or respectively until $\mathcal{N}(\mathfrak{b}) < 0, \forall \mathfrak{b}$. Every cycle starts with the determination of

$$
\mathcal{N}_{\min} = \min_{\mathfrak{b}}\left(\max_{\mathfrak{b}}(0, \mathcal{N}(\mathfrak{b}))\right),
$$
(3)

denoting the minimal, non-negative number of not extrapolated neighbors that a block can have. Next, a list $\mathcal{S}$ is
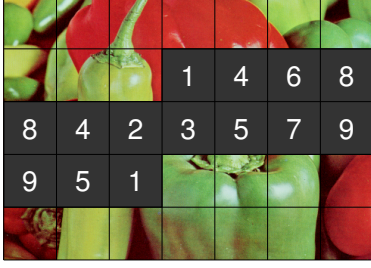
Figure 2: *Example for optimized processing order. Blocks with same indices can be extrapolated in parallel.*

Figure 3: *Examined test patterns. From left to right: Isolated losses, consecutive losses and mixed losses.*

generated that holds all the blocks to be extrapolated in this run. In order to be added to this list, a block on the one hand side has to posses the just determined minimum number $\mathcal{N}_{\text{min}}$ of not extrapolated neighbors. On the other hand side, a block is only added if none of its neighboring blocks is included in $\mathcal{S}$ yet. Following these two requirements, all the blocks of an image can be identified that posses the minimum number of not extrapolated neighbors and that can be processed independently at the same time. Thus, all blocks from list $\mathcal{S}$ can be processed in parallel and as many threads as available on the the processor can be opened for working off list $\mathcal{S}$. In this process, for every regarded block, FSE is conducted and after the model generation has been finished, the corresponding samples of the model are inserted into the loss area of the regarded block. As a processed block influences the number of not extrapolated neighbors of its adjacent blocks, $\mathcal{N}(\texttt{neighbors}(\mathfrak{b}))$ is decreased by 1 in order to update the corresponding reliability information. The steps of selecting the blocks with the lowest number of not extrapolated neighbors and extrapolating them are repeated until all loss regions are filled. In order to give an overview of the proposed optimized processing order, Algorithm 1 shows the pseudo code of the above outlined approach.

Furthermore, Figure 2 shows an example for extrapolating a large region of consecutive blocks. The numbers indicate the order in which the individual blocks are extrapolated. Blocks that posses the same number can be processed in parallel. Obviously, at the beginning, the blocks with the largest support areas $\mathcal{A}$ are selected and based on them the large consecutive loss is continuously closed.

In general, one might also envisage different criteria for determining the processing order. These may for example take into account the number of unknown samples in a neighboring block and not only if the block contains unknown samples. But in this context, it has to be considered that even if a more precise criterion might improve the extrapolation quality a little bit more, it also harms the parallelization. This is due to the fact that less blocks exist that fulfill the more precise criterion at the same time and thus only fewer blocks can be processed in parallel.

4. SIMULATION SETUP AND RESULTS

For evaluating the gain that can be achieved by the optimized processing order, an implementation of FSE in programming language C is regarded. The test device is a Dual quad-core AMD Opteron 2354, running at 2.2 GHz and equipped with 32 GB RAM. Due to the two quad-cores, the computer can effectively run up to 8 threads in parallel. For parallelization, the shared memory base of such a computer is exploited and parallelization is conducted by using OpenMP [7].
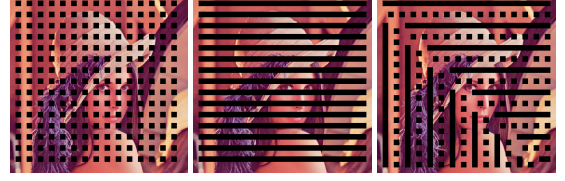
Figure 3 shows the test patterns that are used for the evaluation. The areas to extrapolate are isolated losses, consecutive losses, and a mixture of both. The patterns consist of square areas of size 16×16 samples or respectively of consecutive losses of 16 samples width. According to [3], the used basis functions are the ones from the Discrete Fourier Transform, as these basis functions are suited well for extrapolating smooth as well as noise like areas and edges. In addition to that, by using this basis function set an efficient implementation in the transform domain is possible as shown in [8]. The frame of supporting samples around block $\mathfrak{b}$ is $d = 16$ samples wide, the weighting function decays with $\hat{\rho} = 0.8$, and already reconstructed areas are further weighted by $\delta = 0.2$. The factor γ, that is necessary for Orthogonality Deficiency Compensation [6] during the model generation is set to 0.5 and altogether 200 iterations are carried out for generating the model.

For providing some visual results, the mid and right columns of Figure 5 show the extrapolation results for test images "Lena", "Peppers", and "Baboon" with the test pattern of mixed losses. The blocks to be extrapolated are set to a size of 4×4 samples. Hence, the loss areas with their extent of 16×16 samples are divided into 16 small blocks for which the model is generated successively. As the consecutive loss areas are closed from all directions instead of only from left to right, the propagation of extrapolation errors is reduced by the optimized processing order. This effect becomes especially apparent at the right side of the hat in test image "Lena". Comparing the Peak Signal to Noise Ratio (PSNR) of the luminance component in the extrapolated areas, an improvement between 0.2 dB for "Baboon" and 0.5 dB for "Peppers" can be discovered, if the optimized processing order is used instead of line scan order. Additionally, Table 1

Table 1: *Extrapolation quality in* PSNR *for test image "Lena" if processing is carried out in line scan order or the proposed optimized order.*

Loss pattern	Block size	Line Scan	Optimized
Isolated	16×16	25.35 dB	25.35 dB
	8×8	25.99 dB	25.99 dB
	4×4	25.85 dB	26.19 dB
	2×2	25.71 dB	26.23 dB
Consecutive	16×16	23.70 dB	23.74 dB
	8×8	23.83 dB	23.93 dB
	4×4	23.74 dB	24.20 dB
	2×2	23.47 dB	24.37 dB
Mixed	16×16	24.24 dB	24.56 dB
	8×8	24.30 dB	24.58 dB
	4×4	24.13 dB	24.36 dB
	2×2	24.03 dB	24.35 dB

Table 2: *Mean* PSNR *gain achievable by optimized processing order over line scan order.*

Loss pattern	Block size	Gain
Isolated	16×16	0.00 dB
	8×8	0.00 dB
	4×4	0.19 dB
	2×2	0.33 dB
Consecutive	16×16	0.00 dB
	8×8	-0.03 dB
	4×4	0.23 dB
	2×2	0.48 dB
Mixed	16×16	0.14 dB
	8×8	0.19 dB
	4×4	0.31 dB
	2×2	0.44 dB

Figure 4: *Parallelization gain achievable by optimized processing order for the test pattern of mixed losses.*

shows the extrapolation quality for the considered test patterns in test image "Lena" with various block sizes. While the optimized processing order does not have an effect if isolated losses are concealed and the considered block size is equal to the loss size, a gain of up to 0.5 dB can be achieved if the area to be extrapolated is divided into smaller areas and is processed according to the proposed processing order. Furthermore, Table 2 shows the average gain in PSNR for test images "Lena", "Baboon" and "Peppers" for the considered test patterns and different block sizes. With the only small exception of block size 8×8 for the consecutive loss pattern, the optimized processing order leads to a significantly improved extrapolation quality yielding up to 0.5 dB.

Besides the improved extrapolation quality, the gain in extrapolation speed that can be achieved by parallelization is important. To illustrate this, Figure 4 shows the parallelization gain with respect to the number of used threads for different block sizes. In this context, the test pattern of mixed losses is examined. As the test platform has 8 individual processing cores, it can execute up to 8 threads in parallel. Looking closer at the plot, one can recognize that the algorithm scales nearly perfectly for up to 4 threads, independently of the block size. For a larger number of threads and large block sizes, the performance drops a little bit, but the optimized processing order still is at least 6.9 times faster than the original single thread line scan order. For small block sizes, the algorithm scales even better, achieving a parallelization gain of 7.7. The small drop for large block sizes can be explained by the circumstance that the list $\mathcal{S}$ of blocks to be extrapolated in parallel is more often not filled completely in this case. Thus, the number of blocks that can be processed in parallel is smaller than the number of available threads. Due to this, not all threads can be fed with blocks to be extrapolated as good as for small block sizes. For larger images or larger loss areas this effect dissolves and the algorithm scales well, even for large block sizes.

5. CONCLUSION

In this contribution an optimized processing order for Frequency Selective Extrapolation is introduced. This novel processing order yields an improved extrapolation quality since the area to be extrapolated is closed from all directions. In addition to that, the proposed algorithm is optimized for a parallel execution of the extrapolation in many threads and a parallelization gain of up to 7.7 for 8 threads is
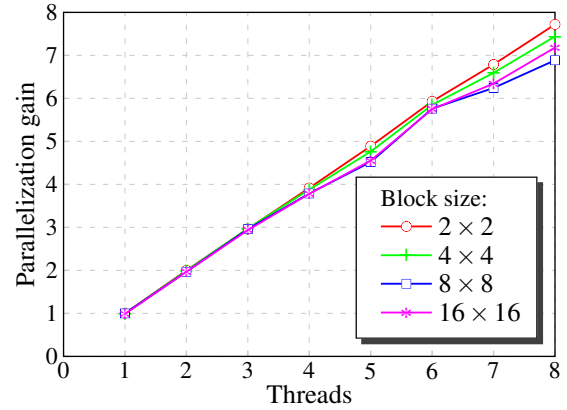
achievable. Although the algorithm has been introduced for two-dimensional signals only, it can be extended to higher dimensional signals easily by making use of [9]. Due to the high computational complexity of the three-dimensional extrapolation, a parallelized execution that makes full use of the processing capabilities becomes even more important, there.

REFERENCES

[1] S. Tsekeridou and I. Pitas, "MPEG-2 error concealment based on block-matching principles," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, no. 4, pp. 646–658, June 2000.

[2] W.-Y. Kung, C.-S. Kim, and C.-C. J. Kuo, "Spatial and temporal error concealment techniques for video transmission over noisy channels," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 7, pp. 789– 803, July 2006.

[3] A. Kaup, K. Meisinger, and T. Aach, "Frequency selective signal extrapolation with applications to error concealment in image communication," *Int. J. Electron. Commun. (AEÜ)*, vol. 59, pp. 147–156, June 2005.

[4] G. Blake, R. G. Dreslinski, and T. Mudge, "A survey of multicore processors," *IEEE Signal Processing Magazine*, vol. 26, no. 6, pp. 26–37, Nov. 2009.

[5] L. J. Karam, I. AlKamal, A. Gatherer, G. A. Frantz, D. V. Anderson, and B. L. Evans, "Trends in multicore DSP platforms," *IEEE Signal Processing Magazine*, vol. 26, no. 6, pp. 38–49, Nov. 2009.

[6] J. Seiler and A. Kaup, "Fast orthogonality deficiency compensation for improved frequency selective image extrapolation," in *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, Las Vegas, USA, March 2008, pp. 781–784.

[7] OpenMP. [Online]. Available: http://www.openmp.org

[8] J. Seiler and A. Kaup, "Complex-valued frequency selective extrapolation for fast image and video signal extrapolation," *IEEE Signal Processing Letters*, vol. 17, no. 11, pp. 949–952, Nov. 2010.

[9] K. Meisinger and A. Kaup, "Spatiotemporal selective extrapolation for 3-D signals and its applications in video communications," *IEEE Trans. Image Process.*, vol. 16, no. 9, pp. 2348–2360, Sept. 2007.
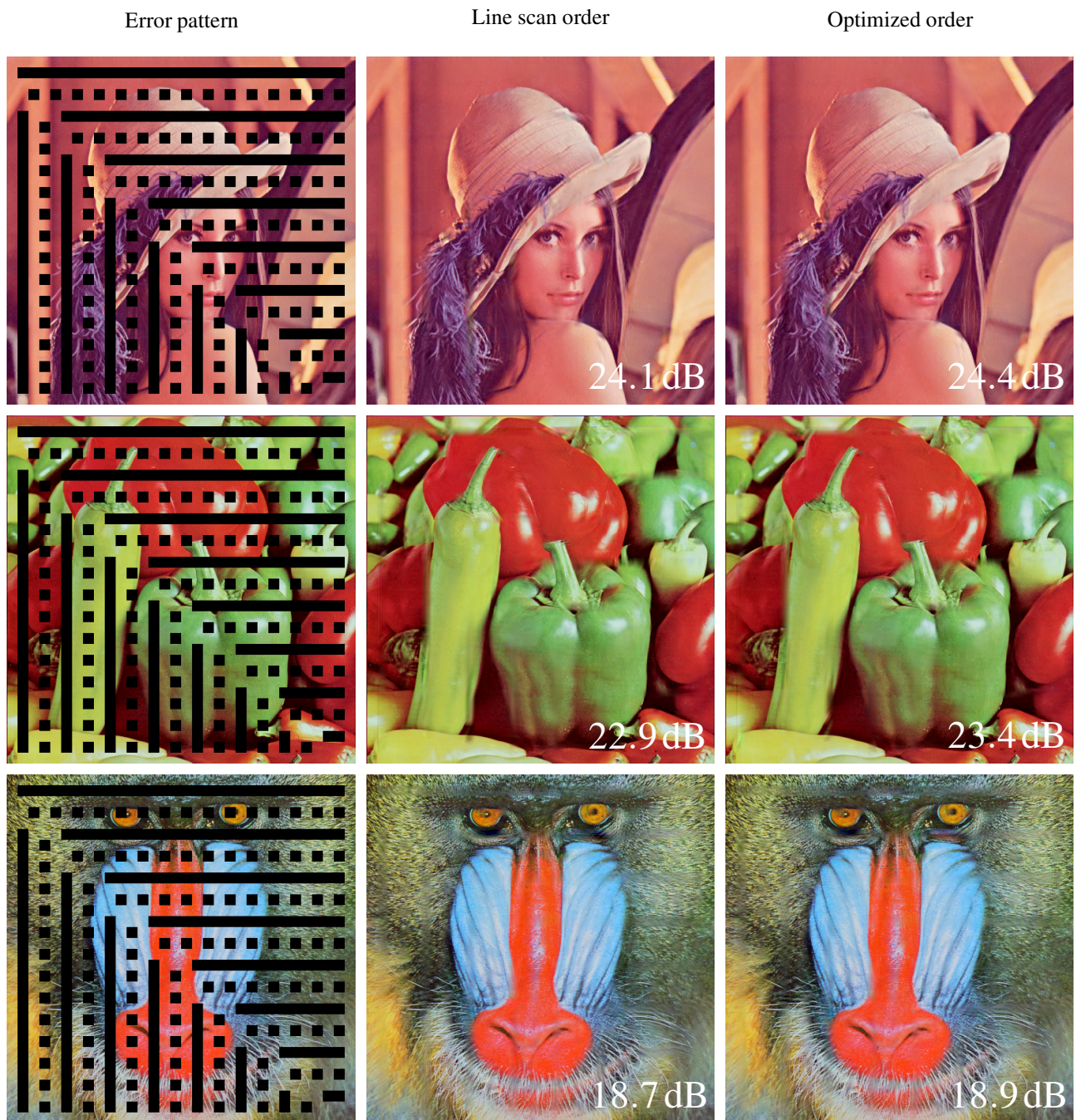
Figure 5: Visual results for the extrapolation of isolated and consecutive losses of 16 samples width.