

# Distributed Parallel Image Signal Extrapolation Framework using Message Passing Interface

Jürgen Seiler and André Kaup

*Multimedia Communications and Signal Processing*

*Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Cauerstr. 7, 91058 Erlangen, Germany*

juergen.seiler@FAU.de, andre.kaup@FAU.de

**Abstract**—This paper introduces a framework for distributed parallel image signal extrapolation. Since high-quality image signal processing often comes along with a high computational complexity, a parallel execution is desirable. The proposed framework allows for the application of existing image signal extrapolation algorithms without the need to modify them for a parallel processing. The unaltered application of existing algorithms is achieved by dividing input images into overlapping tiles which are distributed to compute nodes via Message Passing Interface. In order to keep the computational overhead low, a novel image tiling algorithm is proposed. Using this algorithm, a nearly optimum tiling is possible at a very small processing time. For showing the efficacy of the framework, it is used for parallelizing a high-complexity extrapolation algorithm. Simulation results show that the proposed framework has no negative impact on extrapolation quality while at the same time offering good scaling behavior on compute clusters.

**Index Terms**—Parallelization, Tiling, Message Passing Interface, Image Signal Extrapolation

## I. INTRODUCTION

Looking back at the recent years, it can be discovered that the resolution of images and videos has steadily increased. This trend is very likely to persist for the coming years, introducing challenges to the algorithms and platforms that are utilized for processing this data. One of the problems that further arises is that with the high resolutions, a demand for a high quality processing comes along which typically further increases the complexity. This paper covers one class of image processing algorithms which is very important and at the same time computationally very demanding: image signal extrapolation. The task of image signal extrapolation is that an image which contains missing or undesired areas is reconstructed by extrapolating the available information into the unavailable regions. This extrapolation is required for very different tasks like error concealment, inpainting, defect pixel compensation, or resolution enhancement. In literature, there exist very different algorithms for solving this underdetermined problem as, e.g., patch-based inpainting [1], variational expectation-maximization based algorithms [2], statistically driven modeling [3], or the sparsity-based Frequency Selective Extrapolation (FSE) [4]. Comparing existing extrapolation algorithms, it can be discovered that as soon as they are able to reach a high reconstruction quality, they only achieve this at the cost of a high computational complexity as shown for example in [5].

This problem can be alleviated by exploiting the available compute power in the best possible way. As the single-thread performance of a central processing unit (CPU) could not be further increased that strongly, in the recent years, the trend came up to use multi-core designs for CPUs [6] which follow a shared-memory model and therefore allow for a processing of several threads on the same data. The use of special devices like GPUs or the Intel Xeon Phi could be regarded as a further extension of this concept. However, algorithms always would have to be adapted to the multi-threaded operation. As an example, in [7], it was shown how FSE could be effectively adapted to this, yielding a high parallelization gain.

However, the problem with shared-memory processing is that the number of CPU cores that is available in a computer cannot be arbitrarily enlarged and that the cost of computers which contain more than one CPU increases disproportionately. Thus, for further increasing the computing power, one has to switch to compute clusters. There, the CPUs do not have access to a shared memory and have to explicitly communicate among each other to solve the given task. One concept which allows for this is the Message Passing Interface (MPI) and especially its open source implementation OpenMPI [8]. This technique is quite often used in High Performance Computing and large effort is spent for adapting algorithms to exploit the capabilities of the underlying cluster in the best possible way. However, this often results in extensive redesigns and the direct use of already available algorithms often is not possible.

In this paper, a framework is introduced which allows for the use of compute clusters for image signal extrapolation purposes without the need to redesign the underlying algorithm. The parallelization builds up on OpenMPI, but at the same time the signal extrapolation algorithms do not have to be adapted and arbitrary algorithms can be used without modification. This is achieved by dividing the input image into overlapping tiles that are distributed to the compute nodes. For this purpose, a novel image tiling algorithm is introduced which operates very fast but at the same time yields an efficient tiling of the image. Within this paper, FSE will be considered as an example algorithm since it allows for a high quality reconstruction while at the same time being computationally demanding [4].

Concepts for the parallelization of image processing tasks on clusters already exist in literature, however, they are not suited for image signal extrapolation. For example, in [9], a small scale division is considered with focus on very simple

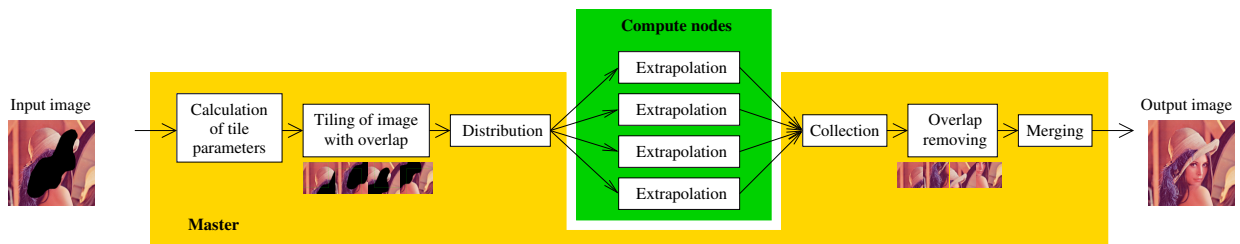


Fig. 1. Extrapolation framework for parallel extrapolation of unknown areas in the input image.

operations on large data sets. In [10], a middleware for vision processing is introduced at a very high abstraction level and the authors in [11] especially focus on the visualization of large amounts of data.

The paper is structured as follows. In the next section, the FSE algorithm is briefly revisited in order to provide a compact overview and to discuss the requirements of image signal extrapolation algorithms on the parallel extrapolation framework. Afterwards, the proposed distributed parallel extrapolation framework is introduced. Finally, simulation results are presented for assessing the abilities of the proposed concept, followed by the conclusion.

## II. BRIEF OVERVIEW OF FREQUENCY SELECTIVE EXTRAPOLATION

The Frequency Selective Extrapolation (FSE) [4] is a very generic signal extrapolation algorithm which also offers a high reconstruction quality. The basic idea of FSE is to divide an image into small blocks and reconstruct missing pixels within each block by generating a sparse model of the signal as weighted superposition of Fourier basis functions. To achieve the modeling, not only the known pixels of the currently considered block are used, but rather pixels from neighboring blocks as well, in order to additionally exploit this information.

On top of this, in [7] an algorithm is introduced for taking the shape and the distribution of the erroneous or unavailable pixels into account for deriving a useful processing order. In doing so, the algorithm is able to close large connected areas to be extrapolated from the outer margins to the center. Using this, the available information from all directions is used allowing for a stable estimation of unavailable pixels, even though the spatial distance to known pixels might be large.

In general, FSE performs best, if the region to be extrapolated is completely surrounded by known samples. This is a property that is also true for other image signal extrapolation algorithms and all algorithms perform inferior if there is no spatial information all around the boundaries of the area to be extrapolated.

## III. OPENMPI-PARALLELIZATION FRAMEWORK

In the following, the novel parallelization framework is introduced that allows for the extrapolation of image signals on a compute cluster of independent nodes without the necessity to redesign the underlying extrapolation algorithm. The basic idea of the framework is to divide the image to be extrapolated

in a set of rectangular tiles where the number of tiles is equal to the number of compute nodes. As the extrapolation quality typically is higher if the area to be extrapolated is completely enclosed by known samples, the framework uses an overlap between the tiles. In doing so, distortions by boundary effects can be alleviated and an independent processing of the tiles at the compute nodes is possible. The parallel extrapolation framework can be described as shown in Fig. 1. Starting with the input image with regions to be extrapolated, the master first calculates the parameters of the tiling. Afterwards, the image is divided into the tiles while a stripe of  $d$  samples width is added to each tile to allow an overlap of the different tiles. In doing so, it is possible to consider pixels in one tile for the extrapolation in the neighboring tile, as well. In the next step, the tiles are distributed to the compute nodes via MPI. In order to avoid delays, a non-blocking communication should be used for this. All these tiles can be processed independently at the compute nodes and the unknown areas can be extrapolated there. The extrapolated tiles are collected by the master again, the overlapping regions are removed, and the tiles are merged together in order to obtain the output image.

Of course, the overlap of the tiles is necessary for having enough information available for the extrapolation, even in the case that the area to be extrapolated is located at the outer margin of the tile. However, as the overlapping regions also have to be processed, a computational overhead arises which harms the overall efficiency. Thus, it is of great importance to use an intelligent tiling in order to keep the computational overhead as small as possible. Since the number of pixels which have to be processed as overhead directly depends on the length of the boundary of the tiles, the tiling should be carried out in such a way that the tiles are preferably compact, that is to say, have a good ratio between the area and the length of the boundary. In the ideal case, this would be squares.

However, this ideal case usually cannot be assured, since the geometry of the input image and the number of tiles have to be considered. Thus, a sensible tiling algorithm is required which satisfies the following properties:

- 1) The tiling should produce tiles of approximately equal size in order to put equal load to each compute node.
- 2) The boundary of the tiles should be as small as possible in order to keep overhead caused by the overlapping regions small.
- 3) The tiling algorithm itself has to exhibit a low complexity as it is running solely at the master.

In [12], an algorithm is proposed which perfectly fulfills the first two properties. This algorithm achieves an optimum tiling, however this is bought at the expense of a high computational complexity. The algorithm can be efficiently implemented recursively, but as it makes use of a brute-force search, its complexity increases exponentially with the number of tiles. Thus, the third property is not met. Alternatively, one might think of tiling an image into horizontal or vertical stripes. This could be achieved extremely efficient and a division into equally sized tiles is no problem. However, the boundary length is longer than necessary, yielding a high computational overhead.

As neither of these two algorithms can fulfill all requirements, a novel tiling algorithm is proposed in the following. This algorithm can be used within the parallel extrapolation framework presented above and provides a tiling which is nearly as good as the one presented in [12] while the computational complexity is of the same magnitude as for the case of a fixed vertical or horizontal division.

The task for the proposed tiling algorithm is to divide an image of width  $W$  and height  $H$  into  $N$  tiles. For the presentation, landscape orientation is assumed, i.e.,  $W > H$ . If this not fulfilled, the coordinate axes have to be swapped accordingly. The first step is to calculate the aspect ratio

$$r = W/H \quad (1)$$

of the image. This is necessary in order to determine how many rows of tiles should be present in the image. Since the primary objective of the tiling algorithm is to obtain compact tiles that have a good ratio between area and boundary, the aspect ratio is considered for calculating the number of rows

$$n_R = \max\left(1, \text{round}\left(\sqrt{N/r}\right)\right). \quad (2)$$

In this context, the rounding operation is necessary since  $n_R$  obviously has to be integer.

Afterwards, the number of tiles  $n_C$  in every row is determined. As  $n_C$  also has to be integer, it is not possible to just divide the desired number of tiles by the number of rows, but rather they have to be distributed unequally. In order to allow for a fair distribution of tiles per row,  $n_C$  is calculated as follows

$$n_C = \begin{cases} \lfloor N/n_R \rfloor & \text{for } i > N \bmod n_R \\ \lceil N/n_R \rceil & \text{for } i \leq N \bmod n_R \end{cases} \quad (3)$$

where  $i$  depicts the row index. After the number of tiles in the currently considered tile row has been determined, the height  $h_T$  of all tiles in this row have to be determined. Of course, this depends on the number of tiles in the row and is determined by

$$h_T = \lfloor H \cdot n_C / N \rfloor + \lceil ((H \cdot n_C) \bmod N) / n_R \rceil. \quad (4)$$

The height of the last row is selected so that the whole image height is covered. Next, width  $w_T$  of the different tiles in the currently considered row has to be calculated. Again, it has to

---

**Algorithm 1** Proposed algorithm for dividing an image into compact tiles.  $x_T$  and  $y_T$  depict offset of the tiles,  $w_T$  and  $h_T$  their width and height. The input image is assumed to be in landscape format.

---

**Input:** Image width  $W$ , image height  $H$ , number of tiles  $N$

**Output:**  $N$  compact tiles of approximately equal area

Aspect ratio  $r = W/H$

Number of rows  $n_R = \max\left(1, \text{round}\left(\sqrt{N/r}\right)\right)$

$x_T = 0, y_T = 0$

**for all**  $i = 1, \dots, n_R$  **do**

5: Number of tiles in current row

$$n_C = \begin{cases} \lfloor N/n_R \rfloor & \text{for } i > N \bmod n_R \\ \lceil N/n_R \rceil & \text{for } i \leq N \bmod n_R \end{cases}$$

$$h_T = \lfloor H \cdot n_C / N \rfloor + \lceil ((H \cdot n_C) \bmod N) / n_R \rceil$$

**if**  $i = n_R$  **then**  $h_T = H - y_T$

**end if**

10: **for all**  $j = 1, \dots, n_C$  **do**

$$w_T = \begin{cases} \lfloor W/n_C \rfloor & \text{for } j > W \bmod n_C \\ \lceil W/n_C \rceil & \text{for } j \leq W \bmod n_C \end{cases}$$

Store  $x_T, y_T, w_T, h_T$  for current tile

$$x_T = x_T + w_T$$

**end for**

15:  $y_T = y_T + h_T, x_T = 0$

**end for**

---

be considered that  $w_T$  is integer. With  $j$  depicting the index of the tile in the current row,

$$w_T = \begin{cases} \lfloor W/n_C \rfloor & \text{for } j > W \bmod n_C \\ \lceil W/n_C \rceil & \text{for } j \leq W \bmod n_C \end{cases} \quad (5)$$

follows.

By applying the above presented operations, the tiling parameters can be determined quite efficiently. In order to provide a compact overview of the proposed tiling algorithm, its pseudo code is given in Algorithm 1. The outputs of the algorithm are on the one hand the horizontal and vertical offsets  $x_T$  and  $y_T$  of the individual tiles, and on the other hand the corresponding widths  $w_T$  and heights  $h_T$ . In addition to the pseudo code, Fig. 2 shows an example for tiling an image of width  $W = 1024$  and height  $H = 480$  into  $N = 22$  tiles, either in vertical tiles, or using the tiling from Bezrukov [12], or the proposed one. It can be discovered, that the objective of the algorithm is fulfilled and that the output tiles are compact as well as of equal size, in the same way as would be achieved by the elaborate algorithm from [12].

#### IV. SIMULATIONS AND RESULTS

For evaluating the performance of the proposed parallel image signal extrapolation framework, tests have been carried out on a small test cluster. The master of the cluster is an Intel Core2 Quad Q9550 and for the computation four AMD Opteron 6274 are available. Each of these CPUs has 8 independent cores. By using every core as independent node, the cluster is able to provide up to 32 compute nodes of equal speed and a tiling of up to the same number can be

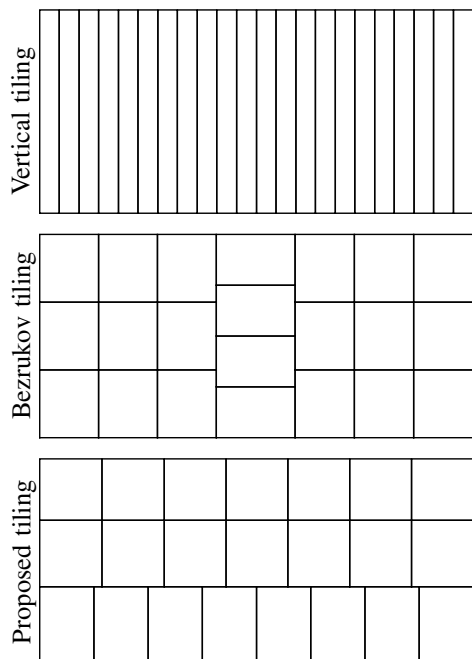


Fig. 2. Example for tiling an image of width  $W = 1024$  and height  $H = 480$  in  $N = 22$  tiles.

tested. The cluster is running the Rocks Cluster Distribution system [13] with OpenMPI version 1.6.2. The FSE used for the extrapolation is written in C and is running single-threaded on each node. The parameters for FSE have been selected according to [7] and a block size of 4 is used. The used test images are shown in Fig. 3 and are called Lion, Elephant and Giraffe and are of size of  $6024 \times 4024$  pixels. Furthermore, two masks that define the areas to be extrapolated are considered. Using Mask1, 14% of the pixels have to be extrapolated, for Mask2 62% of the pixels have to be extrapolated.

Since the extrapolation algorithm is left unchanged, there are two factors in the proposed parallel extrapolation framework that have an impact on the performance. This is the algorithm used for the tiling on the one hand and the size  $d$  of the overlap between the tiles on the other hand. In order to analyze the first one, Fig. 4 shows the time that is necessary just for calculating the tiling parameters at the master. For this, the cases are considered that a simple vertical tiling, the algorithm from Bezrukov [12], or the proposed algorithm is used. Thereby, it has to be noted that the measurement accuracy of the time is of the magnitude of  $10^{-7}$  seconds. Examining the plot, it can be seen that the time necessary for the vertical and the proposed tiling is extremely small and in the order of the measurement accuracy while the calculation time of the Bezrukov tiling exponentially increases with the number of tiles to be determined. For 32 tiles, this already ends up in 79 seconds just for calculating the tiling parameters.

The second important parameter is the width  $d$  of the overlapping areas. As discussed above, a proper overlap is required as otherwise boundary effects may occur which harm the overall extrapolation quality. However, the overlapping

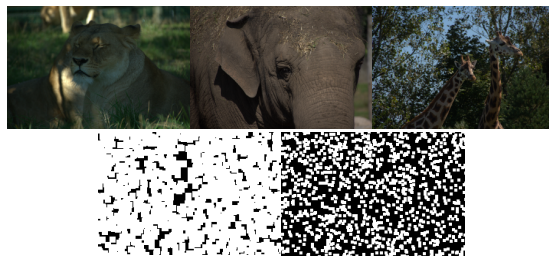


Fig. 3. Test images Lion, Elephant, Giraffe and details of size  $240 \times 160$  pixels of masks Mask1 and Mask2 used for evaluation.

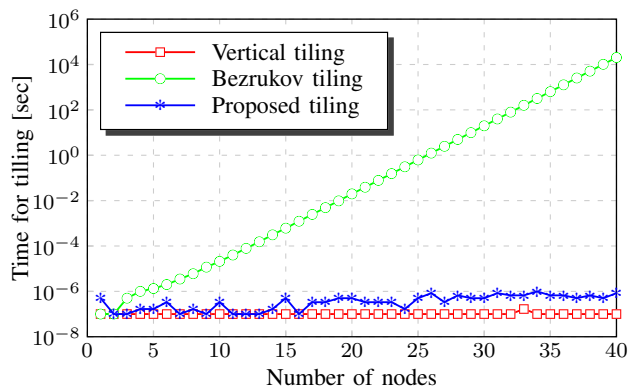


Fig. 4. Processing time for calculation of tile parameters. Measurement accuracy is of the magnitude of  $10^{-7}$  seconds.

areas increase the absolute area to be processed, having an impact on the overhead for the parallelization. In order to assess the impact on the extrapolation quality, Fig. 5 shows the reconstruction quality in terms of PSNR with respect to the width  $d$  of the overlapping area for the test image Elephant and Mask2. It can be seen quite well that small values of  $d$  yield a reduced extrapolation quality. However, it can also be seen that an overlap of  $d = 32$  pixels is sufficient and larger overlapping areas do not yield any improvement. A similar behavior can be observed for the other test images and mask, as well.

Finally, Fig. 6 shows the speedup compared to a single-threaded operation on one node. Here, an overlap of  $d = 32$  is used and again the three different tiling modes are considered. It can be seen quite well that the vertical tiling scales less well. This is due to the fact that the vertical tiling produces an unnecessary large overhead. However, it can also be discovered that the tiling of the proposed algorithm is as good as the one from the optimum algorithm [12] and they scale equally for small number of nodes. If the number of nodes increases, the required time for calculating the tiles becomes significant for the Bezrukov tiling, harming the overall performance. For 32 nodes, the performance even drops below the vertical tiling. All together, the proposed tiling allows a speedup of a factor of 22 if all nodes are used. Of course, the speedup is smaller than the number of nodes, however, this is due to different reasons. First, the actual area to be processed is larger due to the overlapping tiles. Second, additional time is required for tiling the image, distributing the tiles to the nodes and collecting and

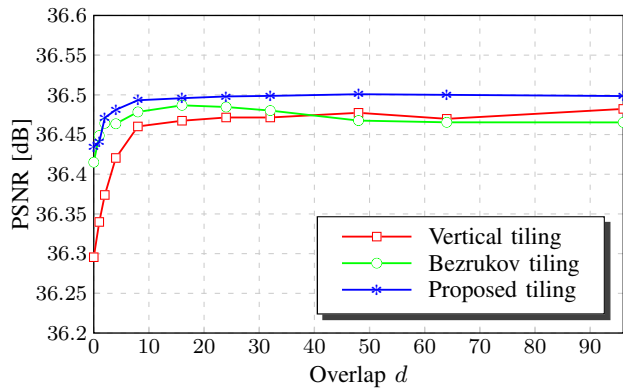


Fig. 5. Influence of overlap size  $d$  on reconstruction quality for test image Elephant and Mask2.

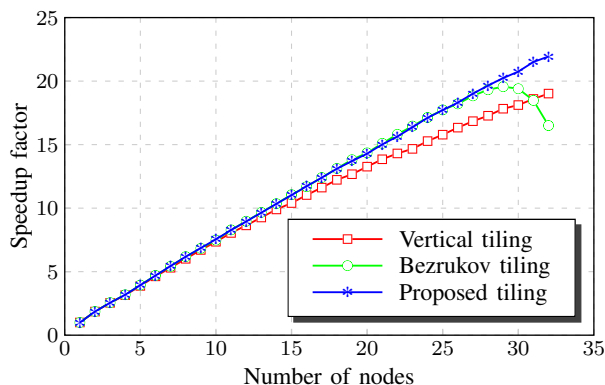


Fig. 6. Speedup of the reconstruction process with respect to the number of nodes for test images with Mask2.

combining the processed tiles again. And finally, since the number of samples to be extrapolated may vary from tile to tile, the processing time per node varies accordingly. However, since the overall time is determined by the longest running node, this further reduces the parallelization gain.

Nevertheless, it has to be noted that the actual extrapolation algorithm can be left unchanged allowing for the use of existing algorithms without the need to adapt them to a parallel processing. For this, a speedup of a factor of 22 still is very beneficial. With smaller overlap  $d$ , a higher parallelization gain would be possible, but at the cost of small quality impairments. In order to show that the proposed parallel extrapolation framework has no negative impact on the extrapolation quality, Tab. I shows the difference in quality, measured in PSNR and SSIM for the case that the proposed parallel framework is used, compared to an extrapolation of the whole images without tiling. Apparently, the difference is negligible and the framework exhibits no negative impact on the extrapolation quality, instead sometimes it may even lead to small gains.

## V. CONCLUSIONS

In this paper, a novel framework for distributed parallel image processing with application to signal extrapolation is introduced. Using this framework, the underlying extrapolation algorithms can be left unchanged and the parallelization is

TABLE I  
LOSS IN PSNR / SSIM CAUSED BY USAGE OF PARALLEL FRAMEWORK WITH 32 NODES COMPARED TO DIRECT EXTRAPOLATION.

	Mask1	Mask2
Elephant	0.0041 dB / $-7.6 \cdot 10^{-7}$	-0.0201 dB / $-3.5 \cdot 10^{-5}$
Lion	-0.0009 dB / $1.5 \cdot 10^{-6}$	0.0046 dB / $-3.1 \cdot 10^{-7}$
Giraffe	-0.0023 dB / $1.6 \cdot 10^{-7}$	-0.0090 dB / $-5.4 \cdot 10^{-6}$

achieved by dividing the image into overlapping tiles. For this purpose, a novel tiling algorithm is introduced which is able to effectively divide the image into compact tiles and therewith keeping the overhead small. Tests have shown that a speedup of a factor of 22 is possible for the case that 32 nodes are used. At the same time, there is no negative impact on the extrapolation quality compared to the case that the underlying algorithm was directly applied to the whole image. Further research aims at including a load balancing to the tiling algorithm for making full use of heterogeneous clusters. Additionally, the distributed parallel framework will be extended to other signal processing tasks like denoising, image rendering, or super-resolution.

## REFERENCES

- [1] A. Criminisi, P. Pérez, and K. Toyama, "Region filling and object removal by exemplar-based image inpainting," *IEEE Transactions on Image Processing*, vol. 13, no. 9, pp. 1200–1212, Sept. 2004.
- [2] X. Li, "Variational bayesian image processing on stochastic factor graphs," in *Proceedings International Conference on Image Processing*, San Diego, USA, Oct. 2008, pp. 1748–1751.
- [3] J. Koloda, A. M. Peinado, and V. Sánchez, "Kernel-based MMSE multimedia signal reconstruction and its application to spatial error concealment," *IEEE Transactions on Multimedia*, vol. 16, no. 6, pp. 1729–1738, Oct. 2014.
- [4] 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.
- [5] J. Seiler, M. Jonscher, M. Schöberl, and A. Kaup, "Resampling images to a regular grid from a non-regular subset of pixel positions using frequency selective reconstruction," *IEEE Transactions on Image Processing*, vol. 24, no. 11, pp. 4540–4555, Nov. 2015.
- [6] 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.
- [7] J. Seiler and A. Kaup, "Optimized and parallelized processing order for improved frequency selective signal extrapolation," in *Proceedings European Signal Processing Conference*, Barcelona, Spain, Aug. 2011, pp. 269–273.
- [8] E. Gabriel, G. E. Fagg, G. Bosilca, T. Angskun, J. J. Dongarra, J. M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, R. H. Castain, D. J. Daniel, R. L. Graham, and T. S. Woodall, "Open MPI: Goals, concept, and design of a next generation mpi implementation," in *Proceedings 11th European PVM/MPI Users Group Meeting*, Budapest, Hungary, Sept. 2004, pp. 97–104.
- [9] J. M. Squyres, A. Lumsdaine, and R. L. Stevenson, "A cluster-based parallel image processing toolkit," in *Proceedings IS&T Conference on Image and Video Processing*, San Jose, USA, Feb. 1995.
- [10] A. Afrah, G. Miller, D. Parks, M. Finke, and S. Fels, "Hive: A distributed system for vision processing," in *Proceedings ACM/IEEE International Conference on Distributed Smart Cameras*, Stanford, USA, Sept. 2008, pp. 1–9.
- [11] K. A. Perrine and D. R. Jones, "Interactive imaging science on parallel computers: getting immediate results," in *Proceedings International Parallel and Distributed Processing Symposium*, Nice, France, April 2003.
- [12] S. Bezrukov and B. Rován, "On partitioning grids into equal parts," *Computers and Artificial Intelligence*, vol. 16, pp. 153–165, 1997.
- [13] [Online]. Available: <http://www.rocksclusters.org/>