Eurographics Symposium on Parallel Graphics and Visualization (2024)
D. Pugmire, G. Reina, S. Rizzi (Editors)

Efficient Adaptive Multiresolution Aggregations of Spatio-temporal Ensembles

Gabriel Borrelli, Marina Evers, and Lars Linsen

University of Münster, Germany

Abstract

Spatio-temporal ensemble data consist of several simulation runs with multiple spatial and a temporal dimension, where the runs are obtained using different parameter settings or initial conditions for the simulation. During analysis, one is interested in investigating the different facets of space, time, and parameter values. When globally analyzing some facet(s), others shall be aggregated to generate summary visualizations. Due to the large amount of data that an ensemble consists of, one may want to generate summary visualizations at multiple levels of detail. Wavelet transforms are a well-known concept for efficiently switching between multiple resolutions. We propose to extend this concept to ensemble data, where individual facets may be aggregated adaptively. We present how to apply the scheme for any data sizes to generate correct averages even when the number of samples is not a power of two in each dimension. We further develop an out-of-core strategy to handle large data sizes. Our scheme is coupled with common 1D, 2D, and 3D visualization methods for an interactive visual analysis of the ensemble data.

CCS Concepts

• ***Computing methodologies*** → *Parallel algorithms;* • ***Human-centered computing*** → ***Scientific visualization;***

1. Introduction

The increase in computational performance over the years opened the door to the simulation of ever more complex physical phenomena at higher precision. Simulations are run for many parameter combinations, such that the resulting set of simulation runs forms an ensemble. The complexity of spatio-temporal ensemble data is manifested by the different facets that come into play with the data being defined over an (often 3D) spatial domain, varying over time, and consisting of different ensemble members, which are often associated with the parameter settings that were used for running the simulation. We consider these facets (space, time, parameters) as dimensions that form a multidimensional hyper-cube, where the dimensionality depends on the number of spatial dimensions as well as on the dimensionality of the parameter space.

When visually analyzing the data, one often concentrates on one or a few facets. This can be achieved by picking specific values for the other facets, i.e., effectively slicing the multidimensional hyper-cube. For example, one may be interested in the temporal evolution and fixes the parameter settings to show an animation of a volume rendering over time. However, such a visualization only shows one ensemble member and not the whole ensemble. To obtain global summary views of the entire ensemble data, one needs to aggregate over some facets to show the change within other facets. For the stated example, one would aggregate over all ensemble members to visualize the temporal evolution of the ensemble mean. Similarly, one could aggregate over space and time to analyze the change when varying parameter settings.

The complexity of ensemble data is also reflected in the data size and poses a problem for the analysis. Nowadays, it is not unusual that the size of an ensemble dataset ranges from hundreds of gigabytes to many terabytes. In this paper, we propose a novel approach that supports the aggregation over any facets of spatio-temporal ensemble data to support summary visualizations of other facets through averaging. Moreover, the aggregation is provided at multiple levels of resolution to alleviate the data size issue. We present an efficient scheme that allows for quickly switching between different levels of resolution and adaptively refining/coarsening the level of resolution in chosen dimensions. In summary, the algorithm satisfies the following five requirements:
(R1) A progressive reconstruction of the data should allow for an analysis at different levels of detail.
(R2) The reconstruction should be flexible, allowing different levels of detail for each dimension.
(R3) The method should apply to large datasets.
(R4) The aggregations should represent an interpretable summary of the ensemble data, such as the arithmetic mean of the ensemble.
(R5) The algorithm should work on volume sizes that are not powers of two.

Satisfying all five imposed requirements is desirable for any method that is meant to be applied for the explorative analysis of

DOI: 10.2312/pgv.20241128

large multidimensional data. For instance, the combination of Requirements R1 and R3 ensures that the approach is scalable to large data with a high number of dimensions but the intermediate views would not be of much analytical use if the approach does not also satisfy Requirement R4. On the other hand, Requirement R2 caters to the needs of an explorative analysis, where the interesting facets of the data are not always known beforehand. In that case, one may wish to discard some dimensions interactively, allowing them to analyze the influence of the remaining ones. The last Requirement R5 ensures that an approach is directly applicable to a dataset of any size, avoiding the risk of introducing any resampling artifacts while preprocessing the data.

We achieve our goals by defining an aggregation scheme called *Multistage Multiresolution Aggregation (MMA)* that computes means based on the *Fast Wavelet Transform (FWT)*. We base MMA on the FWT, as it provides great flexibility in the manner of how to aggregate and reconstruct the data. We discuss how the FWT has prior been used to aggregate single volumes (see Section 3) and how it can be extended to the aggregation of spatio-temporal ensembles (see Section 4). Afterwards, we introduce MA to compute error-free means at all levels of detail (see Section 5.1). We refer to summarizations as error-free if the aggregated data on each level of detail agrees with the direct computation of the summarization. Second, we tackle the problem of large data sizes and provide MMA as an out-of-core extension of MA to enable the parallel processing of volumes, which may not fit into main memory (see Section 5.2). Third, we generate mean summary visualizations by integrating MMA into a visualization framework such that a suitable state-of-the-art 1D, 2D, or 3D visualization is chosen to visualize the aggregated data depending on its dimensionality (see Section 6). We evaluate our approach with multiple synthetic datasets (see Section 8) and show its usefulness by applying it to two real-world ensemble datasets (see Section 7).

2. Related Work

The visual analysis of *large volumetric data*, including ensemble data, has been an active field of research for many years [WHLS19, RH19]. For example, it has become common practice to reduce the amount of data actively residing in memory by employing various compression techniques.

Different surveys [BHP15, BRGIG*14, LMG*18] give extensive overviews of the common techniques. Among those are the use of Octrees for subsampling [Ose09, Ose11], statistically based representations of the data [TLB*11, WLW*17], tensor approximations [Ose09, Ose11, BRP16, BRP19], wavelet transforms [GG16, GS01, GS04, Mur92, IP99, SG10], hierarchical residual encodings [BMSK23], multi-component expansion [ML23] and approaches based on MGARD [ATWK18, ATWK19, GWZ*22, LGC*21]. These techniques generally comprise a compression preprocessing step, followed by an offline reconstruction of the volume at a lower level of fidelity or an online decompression, which is coupled with a visualization, often a direct volume rendering [Max95]. In either case, the compression scheme can be lossy or loss-free, balancing decompression speed, memory requirement, and visualization fidelity. Given that a spatio-temporal ensemble dataset can be interpreted as a single multidimensional volume, with non-overlapping sub-volumes for each ensemble member, those techniques are conceptually also applicable for use on ensemble data. However, none of these techniques fulfills all the requirements stated in the introduction.

Table 1: *Comparison of different methods in relation to the stated requirements. FWT only fulfills R4 with errors.*

Method	R1	R2	R3	R4	R5
Subsampling [Ose09]	✓		✓	✓	
HIRE [BMSK23]	✓		✓	✓	✓
Multi-Component Expansion [ML23]	✓		✓		✓
MGARD [ATWK18]	✓		✓		✓
Curve Boxplot [MWK14]			✓	✓	✓
Contour Boxplots [WMK13]			✓	✓	✓
Streamline Variability Plots [FBW16]			✓	✓	✓
FWT [KMMG87]	✓	✓		(✓)	
MA (ours)	✓	✓		✓	✓
MMA (ours)	✓	✓	✓	✓	✓

The work of Barbarioli et al. [BMSK23] proposes a hierarchical multiresolution compression. While originally proposed as a compression scheme, their work could also be repurposed for data aggregation by choosing an appropriate approximation function. Then, the approximations of X could be reinterpreted as summarizations of X. While possible, a shortcoming of their approach is that, so far, the applicability to the multidimensional case remains unclear. Related to that, the work of Magri and Lindstrom [ML23] proposes Multi-Component expansion for the compression of scalar fields. However, as the multi-component representation uses rounding to low-precision floating-point numbers, it neither provides meaningful aggregations nor allows for flexible reconstructions along single dimensions. Similar shortcomings are also applicable to the other compression methods like subsampling [Ose09] or MGARD [ATWK18], which primarily aim to alleviate memory constraints by representing the data at reduced fidelity, as to fit in the available storage, being either persistent storage or RAM. As such, they are generally not useful for the task of data summarization.

None of the discussed approaches could satisfy all requirements identified in the introduction, as shown in Tab. 1. In particular, compression methods always satisfy the requirement R3, since they reduce the memory requirements for a given dataset but never satisfy R2. On the other hand, summarization approaches satisfy R4, but lack in either R2 or R3. Only MMA fulfills all the requirements.

While related, the use of compression techniques is orthogonal to the concept of *data aggregation*, which this paper focuses on. The main goal of data compression is to enable the visualization of large datasets, where large refers to the amount of memory required to store or work with the data. In contrast, an aggregation seeks to reduce the complexity of the data by creating intermediate, lower-dimensional views summarizing the given ensemble. Approaches relying on a compression/decompression workflow mainly focus on achieving high compression ratios and fast full-fidelity reconstructions with low amounts of error, while, within an aggregation workflow, the focus lies on the ability

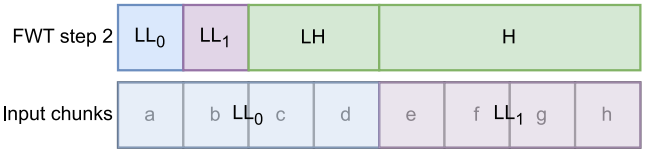

Figure 1: *Schematic comparison of averaging using the FWT (top) and chunks (bottom) on an input with eight elements. LL_0 and LL_1 are the approximations after two transformation steps, while LH and H are the detail coefficients. Computing the second FWT step is equivalent to computing the average inside the chunk of size* 4.

to compute many different low-dimensional aggregations in a fast manner, while rarely requiring the full non-aggregated view on the data. The aggregation facilitates the visual analysis of ensembles by allowing a human analyst to examine the impact of individual facets contained in the ensemble. Typical summarization methods employ statistical summaries like computing means and variances [MWK14, WMK13] or applying some sort of clustering [FBW16] to remove the members dimension of the ensemble. Potter et al. [PWB*09] proposed making use of linked views to display summary statistics of ensemble members in selected regions of space. Luo et al. [LKP03] proposed methods to extend standard visualization techniques to make them viable for ensemble data. The presented methods, among other publications, only focus on summarizing single facets of the ensemble, often the member dimension. Therefore, they are not suited for the task at hand.

3. Background

Of particular interest for the scope of this paper is the *wavelet transform* (WT), mainly the discrete wavelet transform (DWT) and the fast wavelet transform (FWT). It is defined as the convolution $W_\psi(a,b)$ of an input signal $x(t)$ and a wavelet function $\psi(t)$ [KMMG87]. The WT is often unpractical, as it requires computing and storing many coefficients. Alternatively, one can discretize a and b, deriving the DWT. Common choices are to use $a_j = 2^j$ and $b_k = k * 2^j$, where $j,k \in \mathbb{Z}$. Multidimensional signals can be transformed by using an appropriate multidimensional wavelet function, or by applying the 1d wavelet transform in all dimensions of the input signal separately [SDS96].

Mallat [Mal89] proposed a method, called the FWT, to compute the DWT in linear time, assuming that the input contains $2^N, N \in \mathbb{N}$ elements. He describes replacing the wavelet function with a high-pass and a low-pass filter pair, which are used to compute the WT in $\log_2(2^N) = N$ steps. The FWT transforms an input signal by recursively splitting it into an array of approximations and an array of detail coefficients, using both a low-pass and high-pass filter, until the remaining data has reached a minimal size. If we choose a function that computes the average of its input as the low-pass filter, the array of approximations at a step k is equivalent to computing the mean of our input in chunks, where the chunk size is 2^k (see Fig. 1). For instance, with a chunk size of 4, containing the input elements a,b,c, and d, the resulting approximation LL_0 is equal to both the average of our four input elements and the average of the two approximations from the previous step. This equivalency is also valid for the multidimensional case, as multidimensional FWT can be computed by applying the one-dimensional FWT on each dimension of our input separately. In that case, we have a chunk of the size 2^{k_i} along each dimension, where k_i is the number of FWT steps performed for the dimension i. Formally, our low-pass filter f is a bivariate function, defined as $f(a,b) = (a+b)/2$, where a and b are two neighboring elements of the previous step. To later be able to reconstruct the volume, we also have to define a high-pass filter. We are free to choose any function that, together with the results of the low-pass filter, is reversible, as we do not care for the computed detail coefficients in any shape or form, except for reconstruction. Therefore, we choose our high-pass filter g as the function $g(a,b) = a - f(a,b)$. The transformations are reversible, as the equations can be easily inverted. As a result, we can reconstruct the data progressively and can switch between any levels of resolution. Moreover, since the filter pair takes two inputs and produces two outputs, we do not need to increase the memory requirements to store the transformed volume, as the number of stored coefficients remains the same when performing a transformation step (see Fig. 1).

If we partially reconstruct the volume along single dimensions in an adaptive setting, we may encounter the case that the number of elements at a reconstruction step does not match the number of detail coefficients. We can interpret this situation as our detail coefficients containing information to reconstruct data that we already discarded. In that case, we can remove this unnecessary information by reapplying the FWT to the detail coefficients until they contain only the details we require, i.e., until the size of the number of detail coefficients matches the number of elements at a given step.

By storing the detail coefficients at each step, along with the final approximation coefficient, one can thus, essentially, invert the default level of detail of the volume from the highest one to the lowest one, while still being able to reconstruct each level of detail. Therefore, this inverts the cost associated with accessing the different data views, with the highly aggregated views being very cheap to access, but making it expensive to access the fully reconstructed input volume. This is advantageous, as we expect that, during an interactive analysis, one will use a low-dimensional summary of the data most of the time. Therefore, we prioritize the time required to access different data summaries, over the time required to access the full volume.

We showed that the FWT can effectively be used for data aggregation purposes. However, there are some limitations, which make the presented aggregation scheme unfeasible for ensemble datasets. First, the FWT is only defined for inputs where the number of elements is a power of two. Therefore, we would be required to expand the input data until it reaches the required size, resulting in more expensive transformations and reconstructions, and, more importantly, wrong aggregation, as we would compute the means of the expanded input instead of the means of the original input. Second, the FWT requires loading the entire input into memory, making it not trivial to implement efficiently in cases where the data does not fit, which may be the case with a multidimensional ensemble dataset. We propose to tackle the first shortcoming by extending the described aggregation scheme with our MA approach. The second

shortcoming is tackled by our out-of-core extension of the algorithm, which we refer to as MMA.

4. Aggregation of Spatio-temporal Ensembles

Our approach requires an ND volume as its input, which can then be aggregated along the different dimensions. However, due to their size, ensemble datasets are usually stored as a collection of volumes where each volume covers only a subset of the ensemble, for example, as 3D volumes containing a single time step of an ensemble member each. To apply FWT to ensemble data, we first need to map the lower-dimensional volumes to a multidimensional (logical) volume that contains the entire dataset with all its facets. A spatio-temporal ensemble dataset can be described as a $(4+m)$D volume, where three dimensions correspond to the spatial dimensions as in the original data, one dimension represents the temporal dimension, and m dimensions represent dimensions for the ensemble members. Depending on the analysis goals, the set of ensemble members can be represented as a single dimension ($m = 1$). Alternatively, different dimensions can be used for different input parameters (m equals the number of input parameters), allowing for aggregations among individual input parameters. This mapping can be achieved by indexing the data according to the different dimensions and redirecting these indices to the volumes stored on the disk. Thus, no data transformation is required.

In case all ensemble members are considered to form a set, they can be mapped to a single dimension, but the ordering is not predefined. While the order does not influence the result when completely aggregating along the member dimension, it strongly influences the outcome for intermediate aggregation levels. One potential avenue for deriving an ensemble ordering could be to order the ensembles based on the 1D embedding of the member field similarities. If applicable to the dataset, one could use parameter dependencies as a starting point for the ordering. In other cases, depending on the dataset, the member ordering is irrelevant, in which case one could use a random ordering. Defining a general ordering relation for ensemble members is beyond the scope of this paper, and reserve it for future work. For our examples, we define an ordering on a case-by-case basis.

Another challenge when representing the ensemble as a multidimensional volume is caused by multi-field data. While aggregations of different data types, such as scalar fields, vector fields, and tensor fields, are not well-defined, even the aggregation among scalar fields is often not meaningful, such as aggregating a temperature field with a humidity field. If an aggregation is meaningful, multiple fields can be included as an additional dimension. Otherwise, like for the use case presented in Section 7, we create one logical volume per field such that the different fields can be investigated separately or shown in juxtaposed views.

5. Efficient Aggregation

An effective and efficient scheme for data aggregation with spatio-temporal ensembles can be found by using the FWT. However, we also pointed out that there are still two major shortcomings, which make the direct application of the FWT unfeasible for our stated use case. In the following, we will derive a scheme that allows working

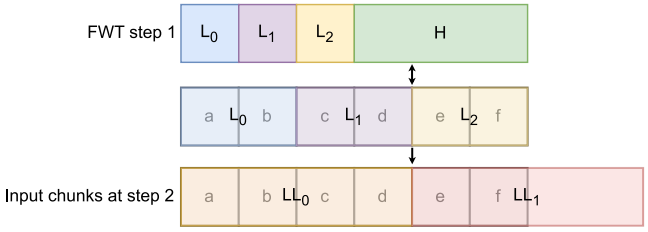

Figure 2: *Schematic comparison of averaging using the FWT (top) and chunks (middle) on an input with six elements. L_0, L_1 and L_2 are the approximations after one transformation step, while H are the detail coefficients. Computing the second FWT step is equivalent to computing the average inside the chunks of size 4 (bottom), which is undefined as the second chunk (LL_1) goes out of bounds.*

with volumes that are not powers of two, as well as an extension to tackle the large ensemble sizes. Our approach is related to the lifting scheme for wavelet transformations [DS98]. However, the modified predict and update steps do not only depend on the results of the splitting step but require additional information for error-free aggregation.

5.1. Multiresolution Aggregation (MA)

To better understand the reason the FWT is restricted to powers of two, it is useful to look at the case where this precondition does not apply. Again, for the sake of simplicity, we will restrict our descriptions to one-dimensional volumes, but all steps apply to the multidimensional case. Let n be the size of our input. We have shown that applying the ith FWT step is equivalent to computing the average inside chunks of size 2^i (see Fig. 1). If n is a power of two, then $2^i | n$ (meaning that 2^i is a divisor of n) for all $i = 0, 1, \ldots, \log_2 n$, i.e., for each FWT step the chunk is fully contained inside the bounds of our input and covers each element, without any overlap. If n is not a power of two, then there exists an $i \in \{0, 1, \ldots, \lceil \log_2 n \rceil\}$ where $2^i \nmid n$. In other words, the chunk goes out of bounds. Transposing it again to the FWT aggregation, this would be equivalent to the number of elements in a step not being divisible by two (see Fig. 2). This can be resolved by expanding the input signal, such that each possible chunk is fully in bounds. Evidently, this would lead to an error in the computed aggregate. Instead, we propose to define the out-of-bounds values dynamically, for each chunk, such that the aggregation inside the chunk is not affected by them. As we are computing the average, we can simplify this to shortening the chunk size until it is fully in bounds. In terms of a FWT step, it is equivalent to leaving the last element unchanged when the number of elements is uneven. We are below presenting a scheme to efficiently compute that. Given that we are now diverging from the FWT, we are from now on going to refer to our aggregation scheme as Multiresolution Aggregation (MA).

An MA step is now a mapping of a sequence of n elements to a sequence of $\lceil \frac{n}{2} \rceil$ elements, and therefore we lose the information whether n was even or uneven after applying the step. This information is critical when we invert the step, i.e., reconstruct the data, so we store it during the aggregation. Further, with the FWT we implicitly assumed that all elements of a step contributed equally

to the aggregate, this is not the case anymore with MA, as each element contributes proportionally to the size of the chunk it was derived from. A straightforward way to solve this problem is to introduce a weighting factor to each element. Initially, at step zero, the elements are weighted uniformly, but we change the weights with each aggregation step. The weight of an element at a specific step can be computed on-demand, as it depends solely on the number of elements in the input sequence, or they can be derived by aggregating the weights of the previous step. In any case, they do not need to be stored along with the aggregated data and detail coefficients, as they can be reconstructed from the input size and the number of aggregation steps applied to the input.

We define our weights as tuples of the form $w = (L,R) \in \mathbb{N} \times \mathbb{N}$ with $l((L,R)) = L$ and $r((L,R)) = R$. The tuples signify the number of elements from the input covered by an element in the approximation, with the constituent parts indicating the number of elements covered by the element at the prior step. The weights are initialized to $(1,0)$ for each element at step zero. To simplify the handling of weights, we further define for each $w_1, w_2 \in \mathbb{N} \times \mathbb{N}$ the relations $|w| = l(w) + r(w)$ and $w_1 + w_2 = (l(w_1) + l(w_2), r(w_1) + r(w_2))$. With these definitions, we can redefine our low-pass filter f to be the weighted arithmetic mean of our input

$$f(a,b,w_a,w_b) = \frac{|w_a| \cdot a + |w_b| \cdot b}{|w_a + w_b|} \tag{1}$$

Similar to the FWT aggregation scheme, we can rewrite an aggregation step in an element-wise notation, with the addition of w_k^i being the weight of the kth element at the step i, and derive the reverse step thereafter, by

$$\begin{aligned} a_k^{i+1} &= \frac{|w_{2k}^i| \cdot a_{2k}^i + |w_{2k+1}^i| \cdot a_{2k+1}^i}{|w_{2k}^i + w_{2k+1}^i|} \\ d_k^{i+1} &= a_{2k}^i - a_k^{i+1} \\ w_k^{i+1} &= (|w_{2k}^i|, |w_{2k+1}^i|) \end{aligned} \tag{2}$$

These equations can, again, be rearranged to a_{2k}^i and a_{2k+1}^i, yielding the reverse step

$$\begin{aligned} a_{2k}^i &= a_k^{i+1} + d_k^{i+1} \\ a_k^{i+1} &= \frac{l(w_k^{i+1}) \cdot a_{2k}^i + r(w_k^{i+1}) \cdot a_{2k+1}^i}{|w_k^{i+1}|} \\ \implies a_{2k+1}^i &= a_k^{i+1} - \frac{l(w_k^{i+1})}{r(w_k^{i+1})} \cdot d_k^{i+1} \end{aligned} \tag{3}$$

If the input size is actually a power of two, then for each $i,k \in \mathbb{N}$ we obtain that $l(w_k^{i+1}) = r(w_k^{i+1})$, i.e., the two Equations 2 and 3 are equal to computations of the FWT. Therefore, the aggregation scheme using the FWT can be considered a special case of MA when the input size is a power of two. Thus, we optimize the MA by reproducing the prior aggregation scheme when applicable, saving us the additional complexity and memory requirements introduced by the weights.

Like with the FWT, we have to consider the special case, where we partially reconstruct an input, which may lead to the number of detail coefficients and the number of elements not matching. We then have to remove the additional details of the detail coefficients by aggregating them with the MA. If we were to compute the aggregation with Equation 2, we would not be able to use the aggregated detail coefficients to reverse the aggregation step on our input data, as the computed weights irreversibly lose the information required for correctly applying Equation 3. In other words, the computed weights would be able to reconstruct the newly aggregated detail coefficients, but not the original input. We therefore modify the handling of the weights of Equation 2 to $w_k^{i+1} = w_{2k}^i + w_{2k+1}^i$. Again, when the input size is a power of two, we obtain $w_{2k}^i + w_{2k+1}^i = (|w_{2k}^i|, |w_{2k+1}^i|)$, thus preserving the full compatibility of the MA and FWT approaches.

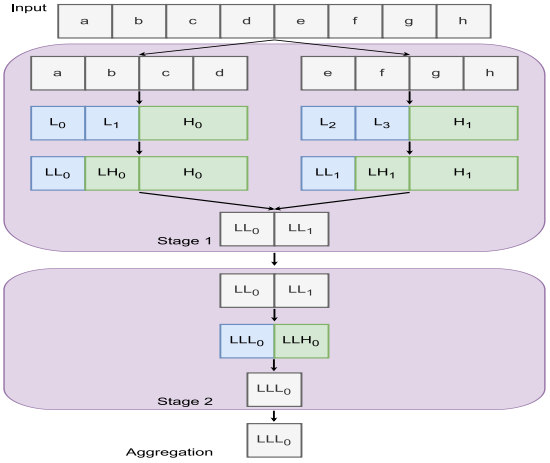

Figure 3: *Schematic visualization of aggregating an input of size eight with MMA and a block size of four. The weights are not displayed, as, in this case, they are not required. Elements ending with an L represent the approximations, likewise elements ending with an H represent the detail coefficients. Each stage splits the result of the previous stage into blocks of size four and applies the MA on each block. The last approximation coefficient of each block is then gathered into an array of partial aggregates. After the first stage, we obtain a partial aggregation of the input with two elements remaining. The complete aggregate is obtained by applying a second stage on the gathered result of the first stage. We can reconstruct the input by reverting the procedure of each stage.*

5.2. Out-of-core Aggregation

The other limiting factor of the FWT aggregation scheme, now replaced by the MA scheme, is the limited scalability due to the large memory requirements with an increasing number of elements. Due to the curse of dimensionality, it is often unfeasible to aggregate large multidimensional volumes using the MA scheme, and therefore we extend the MA to an out-of-core aggregation scheme called Multistage Multiresolution Aggregation (MMA). Instead of applying the MA on the entirety of the input, we subdivide it into multiple non-overlapping blocks and apply the MA on each block individually, leading to a partial aggregation. The result of each block, i.e., the fully aggregated voxel, is then gathered, and the process is

repeated (see Fig. 3). We define a stage as being the operation of splitting the input, applying the MA, and gathering the results. This approach can be implemented with an arbitrary number of stages by caching the result of one stage to persistent memory. Each additional stage reduces the amount of data that has to be processed by the final stage, allowing for the transformation of bigger inputs.

For simplicity, we focus on utilizing only two stages. In that case, the second stage consists of a simple application of MA on the result from the first stage. A stage can be parallelized by processing multiple blocks simultaneously, though with an increase in the runtime memory requirements. As a further optimization, we restrict the block size to be a power of two along each dimension, the size can differ in each dimension. If our input is a n-dimensional volume, each stage may produce 2^n different block shapes, as the block size along each dimension does not necessarily evenly divide the volume along the same dimension. Thus, we have one block shape equaling the chosen block size, i.e., filled blocks, and up to $2^n - 1$ different shapes of partially filled blocks. The partially filled blocks can only occur as the last block along each dimension and are therefore relatively infrequent when compared to the number of filled blocks. Given that we can simplify MA to FWT when applied on a volume with a size equaling a power of two, we can therefore speed up the computation and reduce the working memory required to aggregate filled blocks, while reserving the more complex aggregation for the infrequent occurrence of partial blocks. This optimization presumes uniform weights inside the block, but the same reasoning still applies, as we expect the occurrence of filled blocks with nonuniform weights to be more infrequent than the occurrence of partial blocks, given that the former can only occur as a result of having partial blocks in a prior stage.

The reconstruction of the input with the MMA also occurs in multiple stages, but in reverse order. We start with the last stage and divide the input into single elements, each element belonging to a block of the stage. Then, each block is reconstructed to the requested level of detail by reverting the MA. Finally, the resulting blocks are combined, and the process is repeated with the next stage. If only a specific range of interest is required during the reconstruction, this process can be further optimized, by considering that we can map each block inside a stage to the range of the input volume covered by said block. With this information, we can check whether a block covered some part of the range of interest, and skip it otherwise. As a result, we can significantly reduce the time and memory required to reconstruct the required range, if the range of interest and the block size are small enough.

Similar to MA, we have to handle the case where we partially reconstruct the input in an adaptive setting. The procedure remains largely unchanged, except that now we also have to handle the case that a prior stage was reconstructed partially, in addition to the partial reconstruction of each block. In that case, we also have to merge multiple blocks, e.g., for two-dimensional volume spanning, the X and Y axes, where we only reconstruct the X axis, we merge all blocks along the Y axis. The naive approach of loading all blocks into memory and aggregating the resulting super block may lead to the loading of large amounts of data into memory, in the worst case, we would be required to merge all blocks into one. As a workaround, we use the property of MA being able to produce partial aggregates of two blocks when the weights of the blocks are known, which they are, given that we can always rebuild the weights on demand. Therefore, we merge the blocks sequentially, requiring us to only keep two blocks worth of data in memory at any given time (see Appendix Fig. 1).

6. Explorative Analysis of Ensembles

The aggregated information computed by the MMA can be used to generate mean summary visualizations. The MMA is a data preprocessing and loading scheme and is therefore not tied to one visualization method. As a result, we can easily integrate our approach into preexisting analytical workflows and have access to a broad range of visualization methods. The aggregates of an n-dimensional input volume may possess any number of dimensions, ranging from one to n. In the one- and two-dimensional case, one could use any visualization such as line plots for the 1D case or heatmaps for 2D to directly visualize the aggregated data. For three-dimensional aggregations one could, e.g., use a direct volume rendering [Lev89, Max95], isosurfaces [LC87], etc. We integrated our MMA approach into the visualization framework Voreen [DLJL22], as it already provides many visualization methods for the different dimensionalities out of the box.

As we described above, the MMA trades off fast access to the high-resolution ensemble in exchange for the ability to construct low-resolution aggregates cheaply. The low cost of constructing the aggregates is mainly of benefit during an interactive explorative analysis of the ensemble. During such an analysis, one is often primarily concerned with gaining insight into the patterns contained in the data. Therefore, it is beneficial to quickly change between the different views, which requires adaptive changes of aggregation dimensions and levels. For example, if the users are interested in finding similarities between multiple ensemble members, they may start with a visualization that aggregates every facet except for the ensemble members (see Section 4).Afterward, they may continue by changing to an aggregation that does not aggregate over the temporal dimension,enabling an exploration of the similarity over time. Consequently, they may incorporate other facets in their analysis, like other spatial dimensions. This kind of analysis would be unfeasible if changing the aggregation would incur long computation times.

With our MMA approach, we can avoid long reconstruction times by being able to directly transition from the current level of detail to another in an adaptive manner (in any dimension) without the need to go through the entire multiresolution hierarchy. MMA provides a progressive reconstruction scheme with multiple levels of detail. Thus, instead of reconstructing to the desired level of detail directly, we can split the reconstruction into multiple steps, where each step adds details, i.e., increases the resolution, to the results of the prior step. If, for example, we have many levels of detail $l_0, \ldots, l_n$ for one facet, where l_0 is the lowest and l_n is the highest, we could either reconstruct l_n directly or through a chain of reconstructions $l_0 \rightarrow l_1 \rightarrow \cdots \rightarrow l_n$. While this approach slightly increases the total time required to reach the desired level of detail, it benefits from a reduced interaction latency. It has to be noted, that each step up in the level of detail roughly doubles the resolution and is about twice as costly as the step to the prior level of detail, with

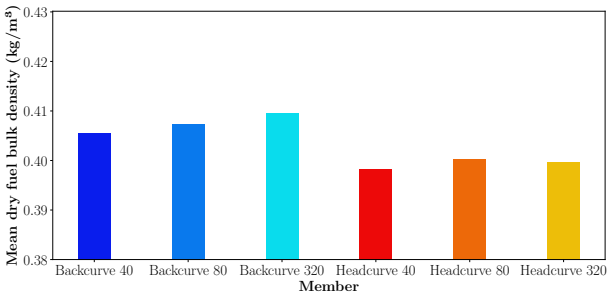

Figure 4: *One-dimensional plot showing the density of dry fuel field (rhof_1) for the time* 0 s-690 s *as bar chart. The reconstruction is configured such that only the values residing directly on the terrain are considered. The x, y, and temporal dimensions of the field are aggregated, and therefore each cell represents the arithmetic mean of the dry fuel density on the terrain. Shades of blue represent the aggregations of the Backcurve runs, and shades of orange represent the Headcurve runs.*

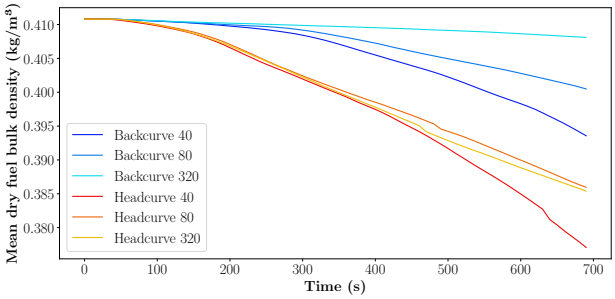

Figure 5: *Two-dimensional plot of the rhof_1 field for the time* 0 s-690 s. *The reconstruction is configured such that only the values residing directly on the terrain are considered. The x- and y-dimensions of the field remain aggregated, and therefore each cell represents the arithmetic mean of the dry fuel density on the terrain. The x-axis represents the time, while the aggregated field values are mapped to the y-axis.*

l_0 being very cheap to reconstruct. Therefore, the progressive reconstruction reduces the response time drastically, allowing us to quickly determine whether the inclusion/exclusion of some facet is of benefit for the analysis.

7. Usage Scenarios

In the following, we apply MMA to ensemble analysis and discuss the insights gained by investigating the data on different aggregation levels.

Wildfire Ensemble. We applied our MMA approach to the SciVis 2022 Contest wildfire ensemble dataset [Lab22], which models wildfire propagation depending on the terrain shape. For the evaluation, we use six simulation runs named Headcurve 40/80/320 and Backcurve 40/80/320. The simulation runs differ in either the form of the terrain or the starting position of the fire. For the Headcurve runs, the fire is positioned on the wind-facing side of the mountain, while it is located on the mountain lee side for the Backcurve runs. The number behind the Headcurve and Backcurve runs represents the mountain curvature. Each run simulated six scalar fields and one vector field over about 70 time steps in 10-second increments. The spatial resolution of the fields is $600 \times 500 \times 63$. Interesting in our case are the rhof_1 field (bulk density of the dry fuel), and the theta field (potential temperature). We use these fields to visualize similarities between the speed and location of the propagating fire of each run. As a first step, we aggregated the ensemble by applying MMA with a block size of $64 \times 64 \times 64 \times 8 \times 8$, i.e., a block size of 64 for the spatial dimensions and a block size of 8 for the temporal dimension and the ensemble members. The ensemble members were ordered lexicographically which in this case agrees with first ordering the Backcurve runs based on the curvature followed by the Headcurve runs.

The shape of the mountain and the location of the fires' origin may have an impact on the propagation of the fire. The speed of the propagating fire is correlated with a decrease in the rhof_1 field, as the fire would consume the available dry fuel. First, we examine the arithmetic mean of the dry fuel mass density on the mountain surface for each simulation run (see Fig. 4). Here, we observe small to no differences between the members of the same group, indicating similarities in the evolution of the propagating fire within the same group. To gain more insight into the propagation of the fire over time, we also reconstruct the temporal dimension (see Fig. 5). We observe a pronounced effect of the shape of the mountain and its starting point on the propagation of the fire. In all Headcurve runs, the mean density decreases faster than in the Backcurve runs, indicating that the faster propagation is aided by the wind traveling up the mountain. We further observe that the curvature of the mountain also affects the propagation, with the lower curvatures causing a stronger decrease in the fuel density. We also observe that the curves of the two groups diverge immediately after 50 s, which corresponds to the time at which the fire is lit.

In a second step, we analyze the propagating fire of the two groups individually by visualizing the temperature field over time. The aggregations of the Backcurve and Headcurve runs are included in the supplemental materials (see WILDFIRE ENSEMBLE) and Appendix Fig. 2, including a video animation. There, we show a volume rendering where the ensemble is reconstructed on the xy-plane, on the mountain surface and the z-axis of the aggregated 3D space is mapped to time. The volume rendering of the aggregation allows for directly studying the evolution of the fire spread over time. In accordance with our previous observations, the visualization depicts stronger fire propagations for the Headcurve runs. Those observations are aligned with our previous analysis of the same dataset [BHS*23]. Compared to the previous analysis of the dataset, our approach enables flexible aggregations along different dimensions without large preprocessing times.

Grand Ensemble As a second usage scenario, we apply MMA to the historical data of the RCP8.5 scenario of the Max Planck Institute Grand Ensemble [MMSG*19] dataset (MPI-GE), which consists of 100 members with a spatial resolution of 192×92 and

1872 timesteps. We consider the anomaly of the monthly surface temperature.

First, we investigate the mean temperature variations over time, where we average over both spatial dimensions and the member dimension. The temperature anomaly variation over time is shown in Fig. 6a. Besides the general increase over time, which can be attributed to global warming, the mean temperature slightly decreased between 1880 and 1900. To investigate this phenomenon in more detail, we progressively refine the representation up to the full resolution shown in Fig. 6b. While significant fluctuations dominate the temporal variation, we observe a steep drop in 1883. This corresponds to the volcanic winter caused by the eruption of Krakatau, which significantly influenced the global climate [Sel92].

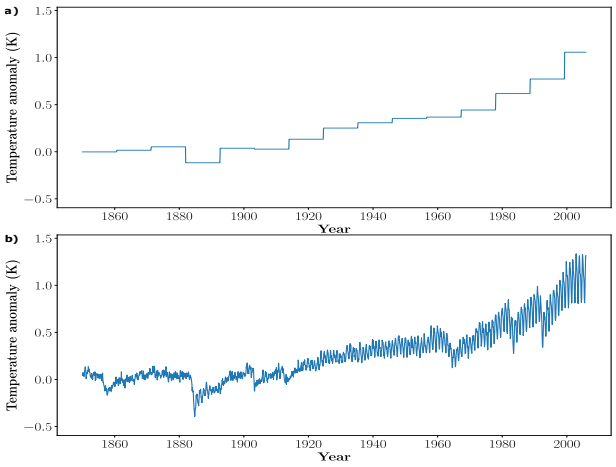

Figure 6: *Mean temperature variation over time for the Grand Ensemble dataset at the 4th level of detail (a) and at the full resolution (b).*

In the next step, we also want to consider the spatial variations of the temperature. Therefore, we create a 3D volume spanned by the two spatial dimensions and the temporal dimensions. For the volume rendering, we limit the color map to temperature anomalies of more than 1.9 K. In the temporally aggregated visualization (see Fig. 7a), we can see that the high-temperature anomalies occur for later timesteps which agrees with the findings of the 1D investigations. However, we observe a strong spatial variation, with the highest temperature anomalies occurring in the arctic region. The full resolution (see Fig. 7b) reveals that in the other regions, the temperature anomaly temporally decreases below 1.9 K, which is not the case for the arctic region, indicating the strong global warming. Thus, our approach allows for a flexible analysis on different levels of detail, facilitating an explorative analysis. While these visualizations can also be obtained without our approach by averaging the complete dataset, these computations would take significantly longer.

8. Evaluation

First, we evaluate MMA according to (1) the error contained in the reconstructed data, (2) the memory required to store our results, and (3) the computation time of reconstructing the data. We

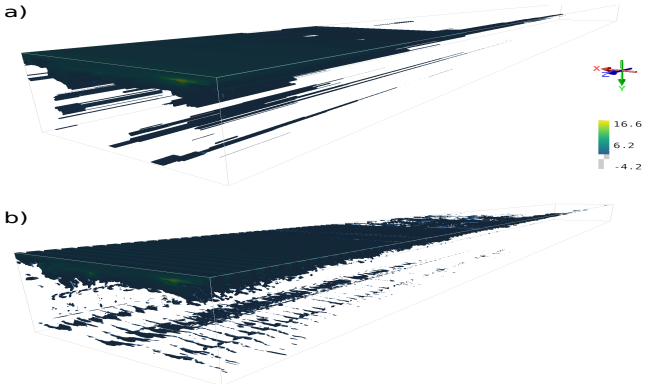

Figure 7: *Spatial variation of high-temperature anomalies over time (z-axis) on the 5th level of detail in time (a) and for the full resolution (b). The spatial dimensions (x- and y-axis) are fully resolved, while the member dimension is fully aggregated.*

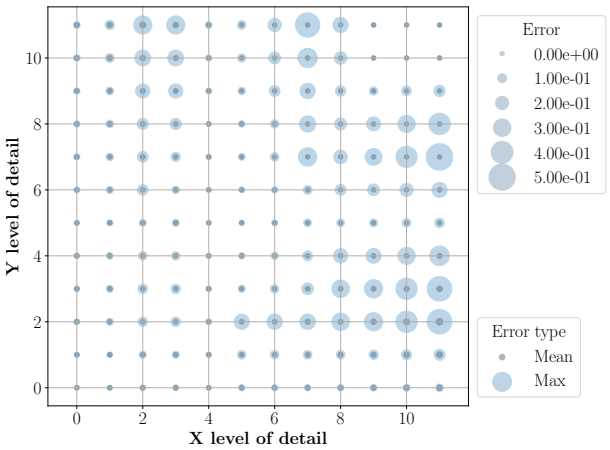

Figure 8: *Mean and maximum errors for the reconstruction of the d_1 dataset with the FWT approach, for each possible level of detail.*

compare our MMA approach to a variant in which we aggregate the blocks using the FWT approach detailed in Section 2 instead of using the MA. The tests that utilize our MMA approach are labeled as "MMA", while the tests using the FWT variant are labeled as "FWT". In the FWT tests, blocks that are not powers of two are filled with zeros at the end up to the next power of two. We test the two methods using one two-dimensional and two five-dimensional synthetic volumes containing randomized values. The volumes are called d_1 (resolution 16534×15873), d_2 (resolution $32 \times 32 \times 16 \times 8 \times 4$), and d_3 (resolution $51 \times 48 \times 37 \times 21 \times 5$). The d_1 volume uses the single precision float data type, while d_2 and d_3 are stored using the double-precision floating-point format. All tests were run on a Linux system with an AMD Ryzen 9 5950X CPU, 32 GB of RAM, 32 CPU threads and a Samsung 960 EVO SSD formatted as Btrfs. Both approaches utilize simple thread pools, where the processing of each block corresponds to one task of the thread pool, no further parallelism has been em-

ployed beyond processing multiple blocks simultaneuosly, but is conceivable with the help of some parallelism library like OpenMP. For the MMA we utilized the filter triple detailed in Section 5.1, corresponding to a weighted arithmetic mean. Accordingly, for the FWT we utilize the equivalent unweighted filter pair described in Section 3, corresponding to an application of the Haar wavelet.

8.1. Reconstruction Error

We compare the two approaches according to the error contained in the (partially) reconstructed data (see Fig. 8). For that purpose, we ran the reconstruction on the d_1 dataset, resampled to the size 1653×1587 using a numeric datatype with arbitrary precision. We define the error as the absolute difference between the reconstruction and a direct aggregation to the requested level of detail, therefore smaller errors indicate higher aggreements between the reconstruction and the direct summarization of the volume. The measurements show that the FWT approach is unable to reconstruct our dataset without introducing a summarization error, while the reconstructions with our MMA approach are free of summarization errors. This is expected, as the FWT approach requires the data to be padded to the next power of two, which alters the aggregation. In practice, the reconstruction will introduce small additional errors, due to the numerical imprecision of the data types. For instance, our measurements show a maximum numerical error of $2.4 \cdot 10^{-5}$ when reconstructing the d_1 dataset with the float datatype. However, this is several orders of magnitude smaller than the maximum error of $5.0 \cdot 10^{-1}$ when reconstructed with the FWT approach. We also observe that the error in the FWT approach tends to increase with higher levels of detail, especially when reconstructing along a single dimension. This is expected, as lower levels of detail distribute the error along the entire aggregation, while with higher levels of detail the error location gets more localized to the blocks where the additional data was inserted. We get similar results in our tests of the d_2 and d_3 datasets (see ADDITIONAL RUNTIMES AND ERRORS in the supplemental materials).

8.2. Memory Consumption

In theory, as our approach is derived from the FWT, we should be able to create the aggregation without increasing the memory footprint of the dataset if the dataset is appropriately sized. In practice, we include additional metadata, like the block size and which MMA steps have been applied, to each block. This metadata aided in the implementation of the approach, but adds a small overhead to each block, and is especially impactful when the data are split into many blocks, due to a small block size. We tried to mitigate this overhead by compressing the blocks stored to disk using the zstd compression [Met23] at the default compression level (see Appendix Table 1). Our measurements show that the block overhead is small in relation to the dataset size, but can become important in memory-constrained environments, where it is not feasible to increase the block size. When coupled with compression, this per block overhead is sometimes offset by the better compressibility of the encoded ensemble than the original input, like in the case of the relatively large SciVis Contest dataset, where our approach only requires 81.17% of the memory needed for storing the original compressed data. Our tests of the FWT approach indicate a similar behavior in the compressed case, but produce much larger outputs in some cases when the data is not compressed (see Appendix Table 2). This is likely due to the FWT requiring a block of data to be a power of two, and therefore padding smaller blocks to bigger sizes.

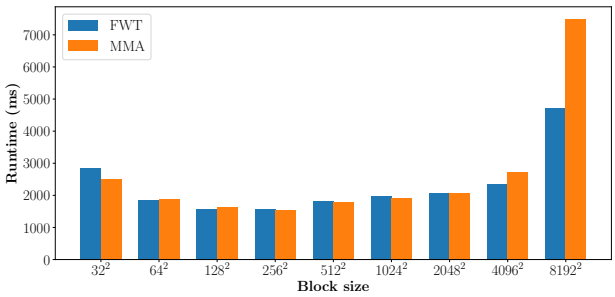

Figure 9: *Mean computation times for the reconstruction of the d_1 dataset using various block sizes. The times are measured using the FWT and MMA approaches and include the time required to load and decompress the blocks from disk.*

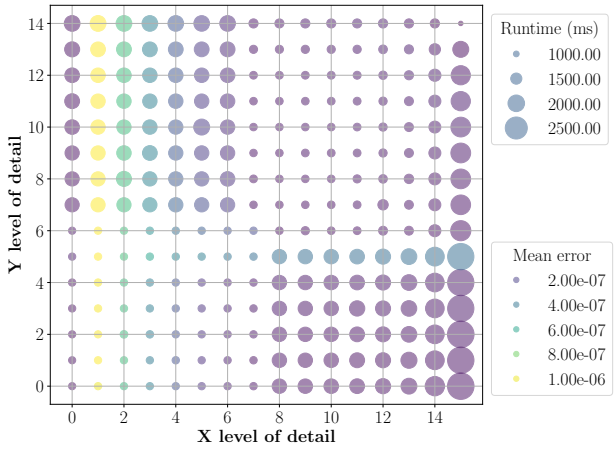

Figure 10: *Mean computation time for the reconstruction of the d_1 dataset using the MMA approach, with the color indicating the mean error of the reconstruction, for each possible level of detail. The reconstruction times include the time required to load and decompress the blocks from disk.*

8.3. Computation Time

Fig. 9 shows the reconstruction computation times of the d_1 dataset over multiple block sizes. A similar evaluation for the dataset d_2 can be found in the supplementary material. The measurements incorporate the time required to load the blocks from disk and decompress the blocks. The computation times of the two approaches are similar in most cases. The selected block size can have a big impact on the required computation time, forming a U-shape in the image. The optimal block size is dataset-dependent, but we observe that

the d_1 dataset provides many options for a good block size with a similar computation time as the optimal one. This observation may be caused by the increased size of the d_1 dataset, which allows for a higher amount of parallelism. As a guideline, one could start by choosing the block size in a way that the input can be split into, at least, 2 to 3 times the number of blocks as there are threads, while being small enough such that one block per thread fits into memory concurrently. We also depict the computation time required for each level of detail combination of the d_1 dataset with a block size of 256×256 (see Fig. 10). From the image, we can clearly see the transition between the two stages at $X = 7, Y = 6$. Past that point, it becomes very computationally expensive to include data from the previous stage, as it would necessitate the merging of multiple blocks. Besides that, the computation time generally increases with an increasing level of detail, with the optimum path being the diagonal, where no additional data adjustments have to take place. As expected, we observe increasing computation times with increasing distance from the main diagonal, as further distances require more adjustment steps. The computation time behaves similarly for d_3 (see ADDITIONAL RUNTIMES AND ERRORS in the supplemental materials). In the case of larger datasets like *SV*, the reconstruction time is entirely I/O bound, and therefore, a higher level of parallelism may be bottlenecked by the data retrieval bandwidth.

9. Discussion and Conclusion

In this work, we presented Multiresolution Aggregation (MA) and Multistage Multiresolution Aggregation (MMA) as its out-of-core extension to enable the creation of aggregations for large ensemble datasets. We evaluated our approach according to its memory requirement, its computation time, and the reconstruction error, where we showed that MMA is error-free except for minor numerical errors due to the limited precision of the datatypes, while not introducing any significant increase in the memory requirements. To show the utility of the aggregation scheme, we applied our method to the wildfire ensemble dataset and the MPI-GE dataset.

In contrast to other aggregation approaches which only aggregate along single facets, our approach allows for an efficient aggregation of multiple facets and at multiple levels of detail. Having multiple levels of detail enables the progressive rendering of ensembles, which is essential for explorations of large datasets. Further, the presented approach is not coupled to any specific visualization method, and can therefore be easily integrated into preexisting analytical workflows. This also allows an easy integration in visual analysis tools that already contain other visualizations, such as detail visualizations for single ensemble members. Hence, when considering the requirements R1-R5 introduced in Section 1, our approach is able to satisfy all five requirements.

We also present limitations and challenges. First, we have shown that finding the optimal block size is difficult, as it is affected by the amount of memory and level of parallelism available on the system, making the sharing of already aggregated ensembles suboptimal. However, our analysis indicates that there is some flexibility in choosing good block sizes that, even if not the optimum, do not significantly degrade the computation time. In future work, one could investigate the possibility of in-place re-encodings to optimize the representation for different machines. Another way to reduce the computation time of our approach is to investigate offloading the aggregation, reconstruction, and decompression of blocks to accelerators, like GPUs.

While the MMA enables multiple levels of detail, it does so on a logarithmic scale and is not continuous, i.e., each level roughly doubles the resolution of a prior level along the selected dimension. This is often sufficient when used to reduce the amount of data, or when used to render the ensemble progressively, but may become a limiting factor if the dimension contains some internal structure one wants to avail itself to. For example, if a dimension contains six elements, where the first three belong to one group and the last three to another group, it is impossible to access the aggregation of the first or last group. To mitigate this, one would be required to split the dimension in two.

Currently, MMA can only aggregate using the arithmetic mean, but other aggregation methods like the variance are also widely used. As presented, our approach can be used to aggregate using an arbitrary linear filter function and the weighted equivalent. In future work, we want to investigate other aggregation schemes for additional visual summarization.

Acknowledgments

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - 310876543.

References

[ATWK18] AINSWORTH M., TUGLUK O., WHITNEY B., KLASKY S.: Multilevel techniques for compression and reduction of scientific data—the univariate case. *Computing and Visualization in Science* (2018). doi:10.1007/s00791-018-00303-9. 2

[ATWK19] AINSWORTH M., TUGLUK O., WHITNEY B., KLASKY S.: Multilevel techniques for compression and reduction of scientific data—the multivariate case. *SIAM Journal on Scientific Computing 41*, 2 (2019), A1278–A1303. doi:10.1137/18M1166651. 2

[BHP15] BEYER J., HADWIGER M., PFISTER H.: State-of-the-art in GPU-based large-scale volume visualization. *Computer Graphics Forum 34*, 8 (2015), 13–37. doi:10.1111/cgf.12605. 2

[BHS*23] BORRELLI G., HAGEMANN L., STEINKÜHLER J., DERSTROFF A., EVERS M., HUESMANN K., LEISTIKOW S., RAVE H., SABBAGH GOL R., LINSEN L.: 2022 ieee scientific visualization contest winner: Multi-field analysis of vorticity-driven lateral spread in wildfire ensembles. *IEEE Computer Graphics and Applications* (2023), 1–10. doi:10.1109/MCG.2023.3310298. 7

[BMSK23] BARBARIOLI B., MERSY G., SINTOS S., KRISHNAN S.: Hierarchical residual encoding for multiresolution time series compression. *Proc. ACM Manag. Data 1*, 1 (2023). doi:10.1145/3588953. 2

[BRGIG*14] BALSA RODRÍGUEZ M., GOBBETTI E., IGLESIAS GUITIÁN J., MAKHINYA M., MARTON F., PAJAROLA R., SUTER S.: State-of-the-art in compressed GPU-based direct volume rendering. *Computer Graphics Forum 33*, 6 (2014), 77–100. doi:10.1111/cgf.12280. 2

[BRP16] BALLESTER-RIPOLL R., PAJAROLA R.: Compressing bidirectional texture functions via tensor train decomposition. In *Proceedings Pacific Graphics Short Papers* (Okinawa, 2016), The Eurographics Association, pp. 1–14. doi:10.2312/pg.20161329. 2

[BRP19] BALLESTER-RIPOLL R., PAJAROLA R.: Tensor decompositions for integral histogram compression and look-up. *IEEE Transactions on Visualization and Computer Graphics 25*, 2 (2019), 1435–1446. doi:10.1109/TVCG.2018.2802521. 2

[DLJL22] DREES D., LEISTIKOW S., JIANG X., LINSEN L.: Voreen–an open-source framework for interactive visualization and processing of large volume data. *arXiv preprint arXiv:2207.12746* (2022). 6

[DS98] DAUBECHIES I., SWELDENS W.: Factoring wavelet transforms into lifting steps. *Journal of Fourier analysis and applications 4* (1998), 247–269. 4

[FBW16] FERSTL F., BÜRGER K., WESTERMANN R.: Streamline variability plots for characterizing the uncertainty in vector field ensembles. *IEEE Transactions on Visualization and Computer Graphics 22*, 1 (2016), 767–776. doi:10.1109/TVCG.2015.2467204. 2, 3

[GG16] GUTHE S., GOESELE M.: GPU-based lossless volume data compression. In *2016 3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON)* (2016), pp. 1–4. doi:10.1109/3DTV.2016.7548892. 2

[GS01] GUTHE S., STRASSER W.: Real-time decompression and visualization of animated volume data. In *Proceedings of the Conference on Visualization '01* (USA, 2001), IEEE Computer Society, p. 349–356. doi:0.1109/VISUAL.2001.964531. 2

[GS04] GUTHE S., STRASSER W.: Advanced techniques for high-quality multi-resolution volume rendering. *Computers & Graphics 28*, 1 (2004), 51–58. doi:10.1016/j.cag.2003.10.018. 2

[GWZ*22] GONG Q., WHITNEY B., ZHANG C., LIANG X., RANGARAJAN A., CHEN J., WAN L., ULLRICH P., LIU Q., JACOB R., RANKA S., KLASKY S.: Region-adaptive, error-controlled scientific data compression using multilevel decomposition. In *Proceedings of the 34th International Conference on Scientific and Statistical Database Management* (New York, NY, USA, 2022), SSDBM '22, Association for Computing Machinery. doi:10.1145/3538712.3538717. 2

[IP99] IHM I., PARK S.: Wavelet-based 3d compression scheme for interactive visualization of very large volume data. *Computer Graphics Forum 18*, 1 (1999), 3–15. doi:10.1111/1467-8659.00298. 2

[KMMG87] KRONLAND-MARTINET R., MORLET J., GROSSMANN A.: Analysis of sound patterns through wavelet transforms. *International Journal of Pattern Recognition and Artificial Intelligence 01*, 02 (1987), 273–302. doi:10.1142/S0218001487000205. 2, 3

[Lab22] LABORATORY L. A. N.: Ieee 2022 scivis contest, 2022. URL: https://www.lanl.gov/projects/sciviscontest2022/. 7

[LC87] LORENSEN W. E., CLINE H. E.: Marching cubes: A high resolution 3D surface construction algorithm. *SIGGRAPH Computer Graphics 21*, 4 (1987), 163–169. doi:10.1145/37402.37422. 6

[Lev89] LEVOY M.: *Display of Surfaces from Volume Data*. PhD thesis, University of North Carolina at Chapel Hill, USA, 1989. URL: http://www.cs.unc.edu/techreports/89-022.pdf. 6

[LGC*21] LIANG X., GONG Q., CHEN J., WHITNEY B., WAN L., LIU Q., PUGMIRE D., ARCHIBALD R., PODHORSZKI N., KLASKY S.: Error-controlled, progressive, and adaptable retrieval of scientific data with multilevel decomposition. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (New York, NY, USA, 2021), SC '21, Association for Computing Machinery. doi:10.1145/3458817.3476179. 2

[LKP03] LUO A., KAO D., PANG A.: Visualizing Spatial Distribution Data Sets. In *Eurographics / IEEE VGTC Symposium on Visualization* (2003), Bonneau G.-P., Hahmann S., Hansen C. D., (Eds.), The Eurographics Association. doi:10.2312/VisSym/VisSym03/029-038. 3

[LMG*18] LI S., MARSAGLIA N., GARTH C., WOODRING J., CLYNE J., CHILDS H.: Data reduction techniques for simulation, visualization and data analysis. *Computer Graphics Forum 37*, 6 (2018), 422–447. 2

[Mal89] MALLAT S.: A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 11*, 7 (1989), 674–693. doi:10.1109/34.192463. 3

[Max95] MAX N.: Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics 1*, 2 (1995), 99–108. doi:10.1109/2945.468400. 2, 6

[Met23] META: Zstandard v1.5.5. https://github.com/facebook/zstd/releases/tag/v1.5.5, 2023. Accessed 22-November-2023. 9

[ML23] MAGRI V., LINDSTROM P.: A general framework for progressive data compression and retrieval, 2023. 2

[MMSG*19] MAHER N., MILINSKI S., SUAREZ-GUTIERREZ L., BOTZET M., DOBRYNIN M., KORNBLUEH L., KRÖGER J., TAKANO Y., GHOSH R., HEDEMANN C., ET AL.: The max planck institute grand ensemble: Enabling the exploration of climate system variability. *Journal of Advances in Modeling Earth Systems 11*, 7 (2019), 2050–2069. doi:10.1029/2019ms001639. 7

[Mur92] MURAKI S.: Approximation and rendering of volume data using wavelet transforms. In *Proceedings Visualization '92* (1992), pp. 21–28. doi:10.1109/VISUAL.1992.235230. 2

[MWK14] MIRZARGAR M., WHITAKER R. T., KIRBY R. M.: Curve boxplot: Generalization of boxplot for ensembles of curves. *IEEE Transactions on Visualization and Computer Graphics 20*, 12 (2014), 2654–2663. doi:10.1109/TVCG.2014.2346455. 2, 3

[Ose09] OSELEDETS I. V.: A new tensor decomposition. *Doklady Mathematics 80*, 1 (2009), 495–496. doi:10.1134/S1064562409040115. 2

[Ose11] OSELEDETS I. V.: Tensor-train decomposition. *SIAM Journal on Scientific Computing 33*, 5 (2011), 2295–2317. doi:10.1137/090752286. 2

[PWB*09] POTTER K., WILSON A., BREMER P.-T., WILLIAMS D., DOUTRIAUX C., PASCUCCI V., JOHNSON C. R.: Ensemble-vis: A framework for the statistical visualization of ensemble data. In *2009 IEEE International Conference on Data Mining Workshops* (2009), pp. 233–240. doi:10.1109/ICDMW.2009.55. 3

[RH19] RAHMAN M. A., HAMADA M.: Lossless image compression techniques: A state-of-the-art survey. *Symmetry 11*, 10 (2019). URL: https://www.mdpi.com/2073-8994/11/10/1274, doi:10.3390/sym11101274. 2

[SDS96] STOLLNITZ E. J., DEROSE T. D., SALESIN D. H.: *Wavelets for computer graphics: theory and applications*. Morgan Kaufmann, 1996. 3

[Sel92] SELF S.: Krakatau revisited: the course of events and interpretation of the 1883 eruption. *GeoJournal 28* (1992), 109–121. 8

[SG10] STEINBERGER M., GRABNER M.: Wavelet-based multiresolution isosurface rendering. In *IEEE/ EG Symposium on Volume Graphics* (2010), pp. 13–20. doi:10.2312/VG/VG10/013-020. 2

[TLB*11] THOMPSON D., LEVINE J. A., BENNETT J. C., BREMER P.-T., GYULASSY A., PASCUCCI V., PÉBAY P. P.: Analysis of large-scale scalar data using hixels. In *2011 IEEE Symposium on Large Data Analysis and Visualization* (2011), pp. 23–30. doi:10.1109/LDAV.2011.6092313. 2

[WHLS19] WANG J., HAZARIKA S., LI C., SHEN H.-W.: Visualization and visual analysis of ensemble data: A survey. *IEEE Transactions on Visualization and Computer Graphics 25*, 9 (2019), 2853–2872. doi:10.1109/tvcg.2018.2853721. 2

[WLW*17] WANG K.-C., LU K., WEI T.-H., SHAREEF N., SHEN H.-W.: Statistical visualization and analysis of large data using a value-based spatial distribution. In *2017 IEEE Pacific Visualization Symposium (PacificVis)* (2017), pp. 161–170. doi:10.1109/PACIFICVIS.2017.8031590. 2

[WMK13] WHITAKER R. T., MIRZARGAR M., KIRBY R. M.: Contour boxplots: A method for characterizing uncertainty in feature sets from simulation ensembles. *IEEE Transactions on Visualization and Computer Graphics 19*, 12 (2013), 2713–2722. doi:10.1109/TVCG.2013.143. 2, 3