This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2022.3214420

Deep Hierarchical Super Resolution for Scientific Data

Skylar W. Wurster, Hanqi Guo, *Member, IEEE,* Han-Wei Shen, Tom Peterka, *Member, IEEE,* and Jiayi Xu

Abstract—We present a novel technique for hierarchical super resolution (SR) with neural networks (NNs), which upscales volumetric data represented with an octree data structure to a high-resolution uniform grid with minimal seam artifacts on octree node boundaries. Our method uses existing state-of-the-art SR models and adds flexibility to upscale input data with varying levels of detail across the domain, instead of only uniform grid data that are supported in previous approaches. The key is to use a hierarchy of SR NNs, each trained to perform $2\times$ SR between two levels of detail, with a hierarchical SR algorithm that minimizes seam artifacts by starting from the coarsest level of detail and working up. We show that our hierarchical approach outperforms baseline interpolation and hierarchical upscaling methods, and demonstrate the usefulness of our proposed approach across three use cases including data reduction using hierarchical downsampling+SR instead of uniform downsampling+SR, computation savings for hierarchical finite-time Lyapunov exponent field calculation, and super-resolving low-resolution simulation results for a high-resolution approximation visualization.

1 Introduction

Computation resources such as node-hours, storage space, memory, and bandwidth are often limited in supply for scientific computing, which pushes scientists and researchers to develop new strategies to perform the desired tasks quicker and use a smaller storage footprint. Recently, approaches [1], [2], [3], [4], [5], [6], [7], have used machine learning (ML) based strategies to reduce computation resources using super resolution (SR) with trained neural networks (NNs). SR approaches use a curated set of training data to teach a neural network to upscale low-resolution (LR) volumes to their high-resolution (HR) ground truth, and may operate in either the spatial domain, temporal domain, or both [1], [2], [8], [9], [10]. With these trained networks, time and computation resources can be saved by running simulations at a lower spatial resolution [2], [3], [6] or temporal resolution [4], [5], [7], and performing SR to infer the missing spatial and/or temporal details by taking advantage of the transformations provided by trained neural networks. The trained neural network can also be used for data reduction by saving LR data and restoring the HR data via the neural network's upscaling abilities [5].

The goal of this work is to use SR techniques with hierarchical data representations for increased computation resource efficiency. Hierarchical data structures are another category of strategies designed for computation resource efficiency, yet there are no SR works that incorporate hierarchical methods. Hierarchical methods, such as octrees, k-d trees, and adaptive mesh refinement (AMR), use an adaptive resolution throughout the spatial domain, which will be more fine (higher resolution) in regions of interest, and more coarse (lower resolution) elsewhere. These methods have been used to speed up simulations [11], [12], [13], [14], [15], improve isosurface extraction and volume rendering speed [16], [17], [18], [19], speed up finite time Lyapunov exponent (FTLE) field computation [20], [21], improve implicit neural network training [22], and assist data reduction tools [23].

We propose a hierarchical SR method that combines the predictive ability of SR NNs with the resource efficiency of hierarchical methods to improve the performance of SR models across resource-constrained use cases. There are two primary challenges when incorporating adaptive hierarchical data with neural-network based SR. First, conventional SR neural networks assume the input will be a regular grid, which is incompatible with hierarchical data formats. Second, hierarchical methods often display *seams* between blocks of different levels of detail, which can cause distracting visual artifacts in tasks such as volume rendering or isosurface extraction [24], [25], [26].

We design a hierarchical SR algorithm that is compatible with existing regular grid SR models, allowing any state-of-the-art SR model to be used, such as ESRGAN [9], SSRTVD [2], and STNet [5]. Our hierarchical SR algorithm relies on a NN hierarchy, where each network in the hierarchy is responsible upscaling one level of detail to the next more fine level. To reduce the impact of seams, we first downscale the hierarchical data to a regular grid at the coarsest level, and then use the hierarchy of networks to upscale the coarse regular grid one level of detail at a time. As the data are upscaled level by level, any higher resolution voxels that are available from the hierarchical data are used to overwrite the upscaled approximated voxels. This upscaling-overwriting process is repeated until the data is at the desired resolution. Seams are avoided by performing regular grid super resolution on the entire domain, as opposed to upscaling separate chunks of data with varying levels of detail separately. To accompany our algorithm, we design a data format called an "SR-octree" to hold the hierarchical data in a format that is efficient for our hierarchical SR algorithm by keeping large chunks of adjacent data joined in a single node when possible.

- *S. Wurster, H. Guo, H.W. Shen, and J. Xu are with the Department of Computer Science and Engineering, The Ohio State University, Columbus, OH, 43210, USA.*
  Emails: {wurster.18, guo.2154, shen.94, xu.2205}@osu.edu
- *T. Peterka is with the Mathematics and Computer Science Division, Argonne National Laboratory, Lemont, IL 60439, USA.*
  E-mail: tpeterka@anl.gov

Authorized licensed use limited to: Argonne National Laboratory. Downloaded on September 14,2023 at 17:24:23 UTC from IEEE Xplore. Restrictions apply.
© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2022.3214420

In our experiments, we use three different architectures (ESRGAN [9], SSRTVD [2], and STNet [5]) within our hierarchies, and show they achieve better peak signal-to-noise (PSNR) ratio and SSIM [10] than bilinear/trilinear interpolation. We verify our hierarchical SR algorithm outperforms a baseline block-wise upscaling algorithm, and effectively minimizes the presence of seams between blocks with different refinement levels.

We also demonstrate our technique's usefulness for three resource-constrained use cases. The first use case is data reduction. As compute power grows, simulations are run at larger resolutions. However, storage and bandwidth cannot keep up, creating a bottleneck that leads to the need for data reduction [27]. SR algorithms are one proposed solution to reduce storage overhead by downscaling and using SR to recover details [2], [3], [5]. We show that our method applied to a state-of-the-art network STNet [5] improves data reduction capabilities. Our second use case is computational efficiency for hierarchical computation. Some expensive scientific and/or visualization algorithms will use hierarchical computation to focus computing resources on regions of interest. If those are allowed to run with an increased tolerated error, hierarchical super resolution can be used to upscale the results for saved computation time. In our evaluation, we use FTLE as the specific use case [20], [21]. Our last use case is upscaling low resolution simulation data for time savings. High resolution simulations take longer to compute compared to the same simulation at a lower resolution. Like other SR techniques, our method can be used to upscale low-resolution simulation output to preview what the high resolution simulation may look like. In our evaluation, we use the Nyx cosmological simulation [11] as the simulation we train our hierarchy to upscale.

In summary, our contributions are the following:

1) A hierarchical SR algorithm that upscales hierarchical data while minimizing seam artifacts for better reconstruction and less visual artifacts
2) A hierarchy of SR neural networks for use with our hierarchical SR algorithm
3) A hierarchical data structure called an SR-octree, which stores hierarchical data for use with our hierarchical SR algorithm.

2 Related Works

We review related works in super resolution methods for images and scientific data and hierarchical methods for scientific computing.

2.1 Super resolution

Super resolution techniques are used to increase the resolution of an image [9], [28], video [29], volume [2], [8], or time-varying volumetric data [4]. The related works below do not include image SR unless otherwise noted, and we refer readers instead to a survey by Wang et al. [30] for a comprehensive review on image SR. In general, there are two categories of SR: spatial super resolution (SSR) and temporal super resolution (TSR). TSR increases the temporal resolution of input data, such as TSR-TVD by Han et al. [4], which uses a convolutional LSTM to learn the recurrence between timesteps and recovers interpolated timesteps with higher accuracy than linear interpolation. TSR is also useful for video frame interpolation [29].

Our approach is an SSR technique, which increases the spatial resolution of input data. Convolutional neural networks (CNNs) were adopted for image SR, with specific techniques introduced to improve quality such as residual learning [29], [31], adversarial training [9], [32], and removing bias from interpolation techniques used in the networks [28], [33]. Similar to our approach, LapSRN [28] estimates multi-level upscaled output to allow $2\times, 4\times, 8\times, ...$ scale factor SR. However, their network trains end-to-end using a Charbonnier loss function and assumes the input is at the lowest resolution, whereas our hierarchy trains each model individually which allows SR from any starting resolution below the training data resolution. Weiss et al. use an SR network to increase the resolution of isosurfaces by upscaling the depth and normal maps [34]. Weiss et al. also use an SR approach for adaptive sampling SR for isosurface and volume rendering images [35].

Image-focused models were adapted to accommodate SSR of 3D scientific data as well, but no methods have yet been proposed for hierarchical data SSR. Zhou et al. [1] use a 3D convolutional neural network (CNN) to perform SSR on volumes with better feature reconstruction than trilinear interpolation or cubic-spline interpolation. Guo et al. [3] use three NNs to perform SSR on 3D vector fields by training a single network for each variable. Xie et al. create tempoGAN [36], which upscales fluid flows for temporally consistent HR output. Fukami et al. compare two ML-based SSR methods for 2D fluid flow [6]. Höhlein et al. [37] use CNNs to perform wind field downscaling, which learns the mapping from coarse input data to fine predicted output data. SSRTVD by Han and Wang [2] uses a generative adversarial network (GAN) for upscaling volumes such that the upscaled output frame is temporally coherent with adjacent timesteps. With the recent advances in SR, Jakob et al. have created a public 2D flow dataset with varying Reynolds number, citing that ML methods are a powerful method for interpolating flow maps and data reduction [38].

There are also works that use ML for spatiotemporal SR, wherein both space and time dimensions are upscaled [5], [7]. These networks use a combination of convolutional layers for spatial upscaling, and may use convolutional and/or recurrent layers for temporal upscaling.

2.2 Hierarchical data structures

Hierarchical data structures have been used in applications including rendering, scientific simulations, data reduction, and ML, to reduce computation and storage requirements by more efficiently allocating resources available. A survey of octree-based rendering methods is available by Knoll [18]. Rendering pipelines have utilized octrees [39] for efficient and scalable volume rendering by dynamically loading data from different octree depths depending on the current camera position and angle. Gobbetti et al. [40] organize large scalar fields into an octree structure and only feed the GPU information relevant to the current viewpoint and transfer function at each frame. Knoll et al. [16] create a technique for isosurface ray tracing for large octree volumes. Wilhelms

Authorized licensed use limited to: Argonne National Laboratory. Downloaded on September 14,2023 at 17:24:23 UTC from IEEE Xplore. Restrictions apply.
© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2022.3214420

and Van Gelder [17] use octrees for isosurface generation when the regions of interest are unknown. Hadwiger et al. [41] enable visualization of petascale volumes of microscopy data using an out-of-core multiresolution approach. Fogal et al. [42] use an out-of-core adaptive multi-grid rendering algorithm that samples densely only where needed to enable rendering of large scientific data on consumer graphics cards. A key problem for rendering hierarchical data is the seam artifact that can occur between data blocks. Methods for reducing the effect of this issue are presented by Santos et al. [24], Wald et al. [25], and Wang et al. [26].

Aside from rendering, hierarchical data formats have also been used during simulation to improve performance. Losasso et al. [43] create a method for simulating water and smoke on an unrestricted octree to capture small scale visual detail and allow for efficient solving of the Poisson equation. Popinet [44] uses quadtrees and octrees for a flexible and efficient approach to solving time-dependent incompressible Euler equations. Adaptive mesh refinement (AMR) schemes, a similar hierarchical data representation, have been utilized in numerous scientific software packages to improve performance by using a coarse mesh where there is little detail and fine meshes where precision is more important. Berger and Colella first design the AMR scheme for use with hyperbolic partial differential equations [45] and two dimensional shock dynamics [46]. Different implementations of AMR are implemented in modern simulation packages such as Enzo [15], Chombo [13], LAVA [14], Nyx [11], and AMReX [12].

Hierarchical representations can also be used for data reduction. Knoll et al. [16] use an octree representation that only uses a fraction of the original data's memory footprint, similar to the approach of Velasco and Torres [47], to accelerate volume rendering. Bhatia et al. [48] introduce AMM, an adaptive multilinear mesh framework that reduces the memory footprint of raw data using tensor products of linear B-spline wavelets and allow a tradeoff between numerical precision and spatial resolution. Hoang et al. [23] use a precision-resolution tree, and an encoding of the tree, to perform data reduction on scalar field data. Ainsworth et al. [49] create a multilevel technique to compress univariate data that allows users to select a level that minimizes memory use while meeting some tolerance level. Another multilevel technique is presented in MGARD [50], which uses adaptive error based coefficient quantization to enable different tolerances at different levels of detail, and proposes to use the multilevel data decomposition as a preconditioner to terminate at an appropriate level.

Hierarchical data representations have also been used in ML. However, our approach differs as it is a SR approach *for* hierarchical data representing scalar fields. Riegler et al. [51] create octree convolution, pooling, and upscaling layers for a NN to classify occupancy octree data representing geometric objects. This method is limited to only occupancy octrees, and not octrees for scalar fields, which we use. Tatarchenko et al. [52] create a deep convolutional decoder that learns to decode dense input to octree representations of 3D objects, which faces the same limitation as occupancy octrees. Martel et al. [22] and Takikawa et al. [53] use hierarchical data representations to assist with implicit neural representation.

3 Approach

There are three components to our approach - the NN hierarchy, hierarchical data structure, and hierarchical SR algorithm. Outlined in Section 3.1 is the construction and training of a SR hierarchy, which is composed of multiple neural networks trained individually. Next, we discuss the hierarchical data structure we use called an SR-octree in Section 3.2. This data structure is a hierarchical format equivalent to an octree [39], but constructed for our hierarchical SR method. We introduce a baseline approach for hierarchical SR in Section 3.3, and discuss our method for upscaling the hierarchical data with minimal seam artifacts in Section 3.4.

3.1 Spatial super resolution hierarchy

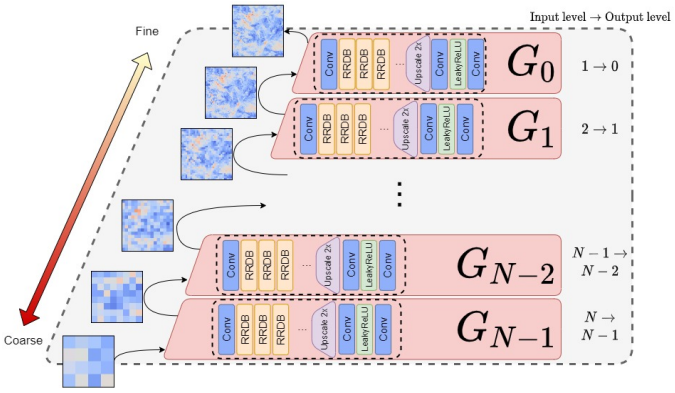


Fig. 1. An example of our SR hierarchy composed of N NNs, each trained to perform $2\times$ SR between different downscaling levels. In this example, each G_i is the generator from ESRGAN, but it can actually be any architecture capable of $2\times$ SR.

In this section, we detail the construction of an SR NN hierarchy. Since the hierarchical data may be at varying levels of detail, we require a design that can upscale various scales of data. For example, some regions may need $2\times$ SR, and others may need $16\times$. Hierarchical NN approaches, such as LapSRN [28] or EDSR [54] may offer multiple scale factors, but expect specific input/output levels of detail, which is incompatible with our SR method described in Section 3.4.

We design a hierarchy compatible with our hierarchical SR approach that is used for uniform scale factor SR at varying scale factors such as $2\times, 4\times, ...$, scaling with the number of NNs in the hierarchy. Our design works with our hierarchical SR algorithm outlined in Section 3.4 by allowing arbitrary input and output level of detail. Our design creates a hierarchy of SR networks that allows a volume of any level of detail as input, and produces an upscaled volume of any level of detail between the input and the original HR as output. Instead of level of detail, we define "downscaling level", such that downscaling level i corresponds to raw data that has been downscaled by a factor of 2^i, or simply data that is a factor of 2^i more coarse than the finest data in the volume. As the downscaling level increases, the data become coarser. We create a hierarchy G of N SR networks $G = \{G_0, G_1, ..., G_{N-1}\}$, where each network G_i is trained

Authorized licensed use limited to: Argonne National Laboratory. Downloaded on September 14,2023 at 17:24:23 UTC from IEEE Xplore. Restrictions apply.
© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2022.3214420

to perform $2\times$ SR from downscaling level $i+1$ to i. By using multiple networks in tandem, larger scale factors like $4\times, 8\times, ...$ up to $2^N\times$ are possible. An example of an SR hierarchy is shown in Figure 1, which shows how coarse input is super-resolved as we move up the hierarchy to the finer scales. Any SR architecture capable of $2\times$ SR can be used in our defined hierarchy for hierarchical SR, and every network within G is identical in terms of architecture, but no weights are shared between models. In this paper, we evaluate our hierarchy with ESRGAN [9], SSRTVD [2], and STNet [5].

3.2 Hierarchical data structure for super resolution

To model the hierarchical data, we define an octree data structure called an SR-octree. Our SR-octree is designed to minimize the number of nodes in the tree and keep data with the same downscaling level contiguous until spatially divided. Though everything will be defined in three spatial dimensions (for an octree), the same methods apply to 2D (for a quadtree). For brevity, we will use the terms "octree" and "SR-octree" regardless of the dimensionality of the data.

Like most octrees, children of a parent node in an SR-octree represent the 8 octants that compose the parent. The octree data structure only defines the spatial partitioning and *not* spatial averages, where the root node refers to the entire domain. Unique to the SR-octree is a `dwnscl_lvl` attribute on each node that represents the factor that data have been downscaled in each spatial dimension within that node. This attribute allows us to represent contiguous large regions with a uniform downscaling ratio with one node. Specifically, each leaf node in the SR-octree holds a volume of data defined by the node's spatial partition (position and size) and downscaling level. This is necessary to upscale the data properly in our hierarchical SR algorithm.

Figure 2 shows the concept of an SR-octree using a quadtree example in 2-D. Non-leaf nodes hold no data and only point to their child octants. Nodes are colored according to their downscaling level shown by the key on the left. We also show an equivalent "typical" octree (as defined by Meagher [39]) in the bottom right that has more dense nodes in the finer regions. Notice that the mesh density corresponds to the color (downscaling level) in the equivalent SR-octree on the left compared with the typical octree. Our SR-octree construction groups together octree nodes at the same downscaling level into a single node.

In some use cases and for testing purposes, it is necessary to downscale a uniform grid HR volume to an SR-octree, such as for our uses cases for data reduction (Section 4.4.1) and hierarchical SR for FTLE data (Section 4.4.2). To create an SR-octree from a uniform full resolution volume, we hierarchically reduce a full-resolution volume according to a user set error bound ϵ, as follows. Starting with the original volume, downscale the volume by $2\times$ in each spatial dimension, and see if the L^∞ norm of the downscaled version compared with the original volume is below the threshold ϵ. If so, keep the volume downscaled by $2\times$ and increment the downscaling level of the volume's octree node by 1. Otherwise, return the volume to the higher resolution version, split the volume into its octants, and repeat the above process for each suboctant if each dimension of the

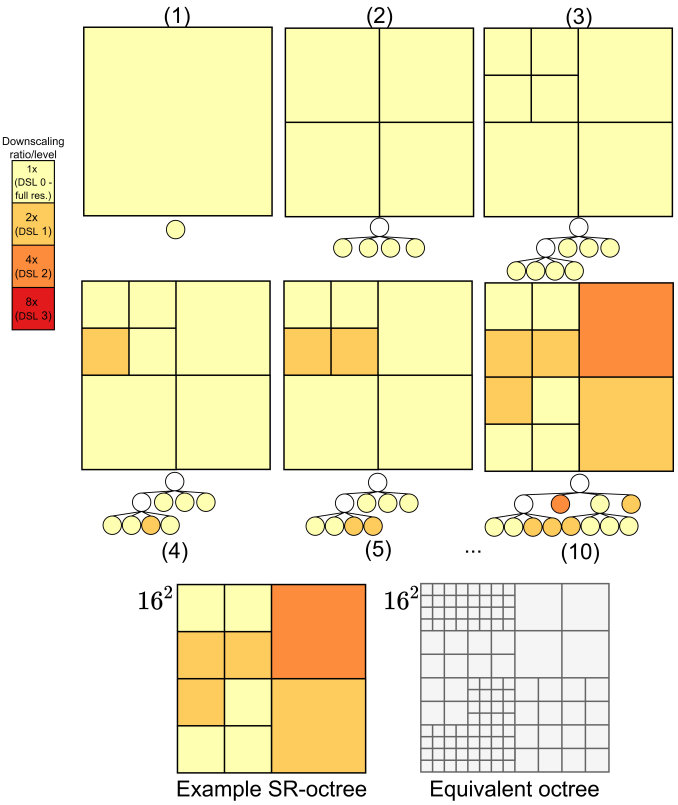


Fig. 2. A quadtree example of an SR-octree and visualization of each quadrant's `dwnscl_lvl`s while creating hierarchical data for experiments. In (1) data starts at full resolution (`dwnscl_lvl` $= 0$). In (2), a trial and error step for the ϵ error is not met, so the volume is split into quadrants, and the process is repeated starting in the top left corner in (3), which also fails, resulting in a second split for that quadrant. In (4), the bottom left quadrant was able to be downscaled while having error less than ϵ, shown by the color changing to orange, which represents `dwnscl_lvl` $= 1$. In (5), the same downscaling is successful for the sibling of the previous quadrant. By (10), the top right quadrant has been downscaled twice, and the bottom right quadrant once. The bottom shows the equivalent octree [39] (without the downscaling level attribute) which has a finer mesh over the regions which have a finer resolution.

suboctant is larger than some minimum size `min_chunk`. This process will maximally hierarchically downscale the volume according to the error ϵ. We add other downscaling parameters `max_dwnscl_lvl`, which controls the maximum downscaling factor, and `min_dwnscl_lvl`, which will automatically downscale the original volume to that scale factor before performing the hierarchical downscaling. After reduction, we join any adjacent nodes that have the same downscaling level so that the hierarchical data is represented with the fewest number of nodes as possible.

The resulting SR-octree has volume data in leaf nodes, each having an associated `dwnscl_lvl` attribute for the volume it represents. This allows us to easily determine what SR scale factor is needed to restore an octree node to its original full resolution. We note that in visualizations of an SR-octree, the coarseness/fineness of a region is not determined by the density of the grid in a region (such as with typical octrees), but instead is represented with the `dwnscl_lvl` attribute, which we visualize with a discrete color scale in Figure 2.

Authorized licensed use limited to: Argonne National Laboratory. Downloaded on September 14,2023 at 17:24:23 UTC from IEEE Xplore. Restrictions apply.
© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2022.3214420

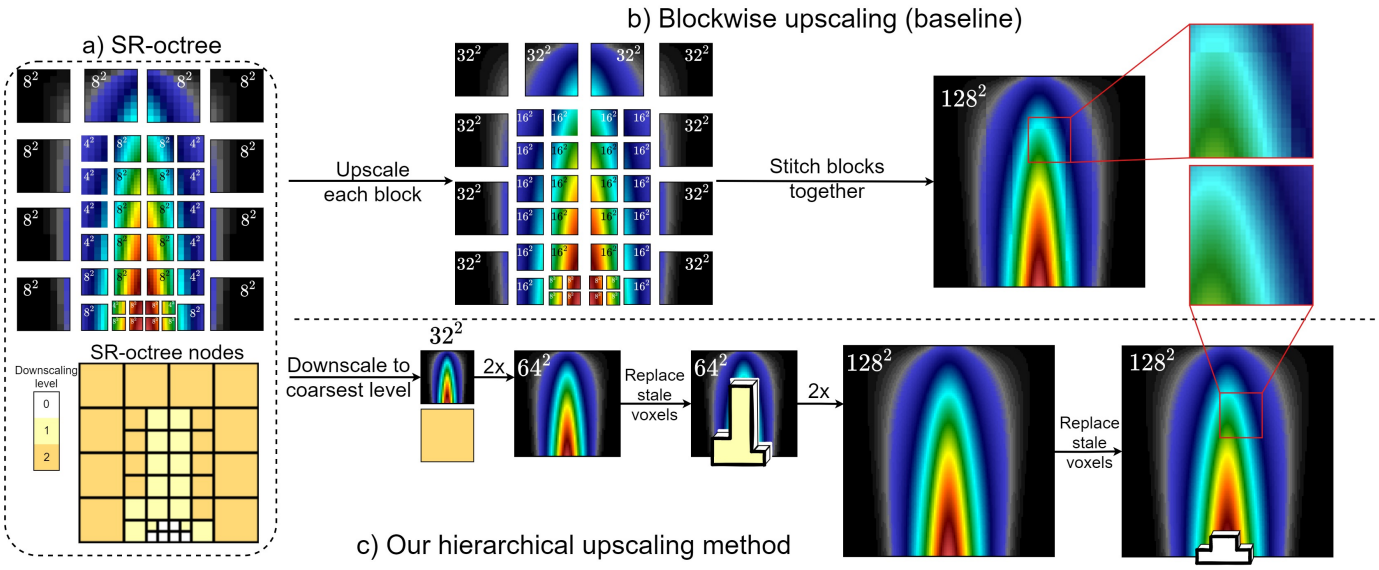


Fig. 3. For an SR octree (a), visualization of the baseline blockwise upscaling approach (b) and our hierarchical SR approach (c). In the baseline approach (b), each octree node is upscaled individually and stitched together. In our hierarchical SR approach (c), we follow the approach outlined in Section 3.4 that downscales data to the coarsest level first, then iteratively upscales with networks from the NN hierarchy (d) and overwrites stale voxels.

3.3 Limitation of an alternative hierarchical super resolution method

Since no methods currently exist for using SR NNs for upscaling hierarchical data, we contrast our algorithm with another baseline approach. The baseline solution to upscale an SR-octree upscales each octree node individually based on the `dwnscl_lvl` attribute each octree node is saved with (i.e. upscales by $2^{\texttt{dwnscl_lvl}}\times$), and then stitches together the resulting patches, illustrated in the top portion of Figure 3.

There are two drawbacks to this method that lead to poor performance with trained neural networks and the introduction of *seam* artifacts. The first problem is that convolutional neural networks use *padding* within convolution layers, which introduces incorrect neighborhood information to the trained neural network, causing error in the upscaled result. In convolutional neural networks, padding is a technique that will surround an input volume with a layer of values around the boundaries before a convolution operation. In practice, padding is used to ensure that the output resolution of the convolutional layer within the NN is the same as the input resolution. Without padding, a convolution layer with a stride of 1 would reduce the spatial resolution by $2 \times \lfloor \frac{\texttt{kernel_size}}{2} \rfloor$ voxels in each direction. The network's output is then incorrectly influenced by the padded values that are not actually there. When composing multiple convolutional layers in sequence, the effect is increased. Due to the error from padding, some evaluations for proposed SR NNs choose to ignore some number of border pixels when calculating the recovered image's PSNR, SSIM, or perceptual quality [55], [56], [57], [58], [59].

Closely tied to the padding problem is the second problem, which is the lack of shared neighbor information. Deep convolutional neural networks will often have large *receptive fields*, which indicate the window size that influences each output voxel. The larger the receptive field, the more global information is used within the neural network to determine the estimated output voxel value. The receptive field is determined by the neural network architecture (number of convolutions and the stride/kernel size/padding, etc.). Therefore, a hierarchical upscaling algorithm that shares or copies a static number of neighboring voxels/neighbor octree nodes, such as work by Ljung et al. [24] or Wang et al. [26], would only work for networks that have a receptive field smaller than the number of shared voxels.

In our setting specifically, padding and limited shared neighborhood information will have a pronounced impact for SR-octree nodes that have few voxels. For example, a 2^3 sized leaf node in an SR-octree would be dwarfed by the total number of padded voxels from up to 30 padding operations in our deep convolutional neural networks. Additionally, in this baseline approach, each node will use its own padding operations, so the upscaled results between each node will not align well, introducing *seams* between data chunks which can be distracting and/or misleading in visualizations.

3.4 Hierarchical super resolution for SR-octrees

In this section, we present our approach for hierarchical SR. We design an approach that uses the entire domain at each downscaling level during SR so that padding does not create seams within the volume, and so that the receptive field of the neural network is not limited when upscaling, both of which are limitations outlined in Section 3.3. Though data are distributed across many different downscaling levels, the only level that can *uniformly* represent the hierarchical data is the coarsest level. Converting the hierarchical data to this uniform representation is the first part of our algorithm and is called the downscaling process, outlined in Section 3.4.1. Next, our upscaling process (detailed in Section 3.4.2) will uniformly super resolve the uniform

Authorized licensed use limited to: Argonne National Laboratory. Downloaded on September 14,2023 at 17:24:23 UTC from IEEE Xplore. Restrictions apply.
© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2022.3214420

LR data without introducing seam artifacts and overwrite voxels that have more accurate information in the SR-octree, one level at a time, beginning with the coarsest level. In the end, we are left with an approximation of the HR uniform volume.

3.4.1 Downscaling process

Our downscaling process is a preprocessing step which operates on the SR-octree that downscales each node's data to the maximum downscaling level, which gives our upscaling process a uniform volume to begin SR. Downscaling all octree nodes to the maximum downscaling level, `MAXDSL`, and putting them together into a regular grid volume is not always possible however. As an example, a single voxel octree node cannot be downscaled by a factor of 8.

We resolve this issue with an iterative downscaling process. Instead of taking one large downscaling step for each node, we take many smaller global downscaling steps and combine data between each downscaling operation. Our SR-octree construction guarantees that single voxel octree nodes at `MINDSL`, the minimum downscaling level present in any octree node of an SR-octree T, must have siblings with the exact same size (a single voxel) and downscaling level. Therefore, these siblings can be joined to make a larger octree node that is 2^D and can be downscaled by $2\times$.

Algorithm 1 Our downscaling procedure that takes an octree T as input and returns a LR regular grid volume.

1: **procedure** HIERARCHICAL_DOWNSCALE(T)
2: `MINDSL` $\leftarrow 0$
3: **while** `MINDSL` $< T$.`MAXDSL` **do**
4: $N = \{n \in T \mid n$.`dwnscl_lvl` $=$ `MINDSL`$\}$
5: Combine adjacent single voxel data nodes in N
6: Downscale each $n \in N$ by $2\times$ in each dim.
7: `MINDSL` $\leftarrow$ `MINDSL` $+ 1$
8: **end while**
9: **return** T.`to_volume()`
10: **end procedure**

This motivates the downscaling process followed in Algorithm 1, which operates on an SR-octree T. At each level starting with the finest level, we combine single voxel nodes, and then downscale all nodes at that level by $2\times$ to the next level. The procedure is repeated until reaching `MAXDSL`, which is the maximum downscaling level on any node in T. The effect after finishing is that all data from the octree T is downscaled to a single regular grid volume at `MAXDSL`.

Since we downscale regions of data by a factor of 2 multiple times, we require the following property in the downscaling algorithm used. Given a full resolution volume V, scale factor S, and downscaling algorithm $\texttt{down}(V, S) : \mathbb{R}^{|V|} \rightarrow \mathbb{R}^{|V|/2}$, it must hold that $\texttt{down}(V, 2 \times S) = \texttt{down}(\texttt{down}(V, S), S)$. When this property holds, there is no effect from downscaling a region multiple times compared to downscaling it once at a large downscaling factor. For instance, downscaling a region by a factor of $2\times$ three times would be the same as downscaling it by $8\times$ once. We observe this property in all pooling algorithms (min, max, mean pooling) as well as subsampling, but not in linear interpolation downscaling.

3.4.2 Upscaling process

The upscaling process (illustrated in the bottom of Figure 3, and defined in Algorithm 2) begins with the uniform LR volume created by the downscaling process, and iteratively upscales and replaces specific voxels with non-approximated data from the SR-octree until we are left with a uniform HR volume approximating the ground truth data.

Algorithm 2 Our upscaling procedure that takes a LR volume V as input (the result of the downscaling procedure described in Section 3.4.1) and the corresponding SR-octree T. The procedure returns a HR regular grid volume.

1: **procedure** HIERARCHICAL_UPSCALE(V, T)
2: `MAXDSL` $\leftarrow T$.`MAXDSL`
3: **while** `MAXDSL` > 0 **do**
4: $V \leftarrow$ `upscale`(V)
5: `MAXDSL` $\leftarrow$ `MAXDSL` $- 1$
6: **for all** $n \in T$ **s.t.** n.`dwnscl_lvl` $\leq$ `MAXDSL` **do**
7: $V[n$.`region`$] \leftarrow n$.`data`
8: **end for**
9: **end while**
10: **return** V
11: **end procedure**

Beginning with the uniform LR volume (argument V in Algorithm 2), we uniformly upscale this volume to `MAXDSL` $- 1$ via a $2\times$ upscaling algorithm such as a neural network (line 4 in Algorithm 2). We do not need to use approximated voxels when non-approximated data exists in the SR-octree. If an octree node in the octree has `dwnscl_lvl` $\leq$ `MAXDSL`, we replace the approximated voxel values in our volume with the more accurate voxel values from the octree node representing them (line 7). We call these approximated values that should be overwritten *stale voxels*, and highlight these regions with an extrusion in Figure 3. We repeat our upscaling $\rightarrow$ stale voxel overwriting until we reach `dwnscl_lvl` $= 0$. Since we upscale the full volume instead of each octree node individually, we do not introduce seam artifacts on octree node boundaries. Additionally, since the entire domain is being used during upscaling, we are allowing as large a receptive field as possible to the neural network upscaling it.

3.5 SR model architecture design

In our proposed NN hierarchy, any spatial SR model capable of $2\times$ upscaling is compatible. In this paper, we use ESRGAN [9], SSRTVD [2] and STNet [5], which are all state-of-the-art neural networks for SR. Small changes were made to the models for compatibility and training stability. For all three models, we find they train more stably and with better reconstruction without the discriminator network(s) or losses, so each of the three architectures are only regressors that upscale input. Additionally, we adjust the models to only perform $2\times$ upscaling instead of $4\times$ upscaling. This is done by using only a single pixel/voxel shuffle [33] at the end of the model to upscale the features by a factor of 2 in each dimension instead of what the original architectures suggests. Regardless of the architecture chosen for the NN hierarchy, each generator network G_i in our constructed neural network hierarchy is responsible for

Authorized licensed use limited to: Argonne National Laboratory. Downloaded on September 14,2023 at 17:24:23 UTC from IEEE Xplore. Restrictions apply.
© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2022.3214420

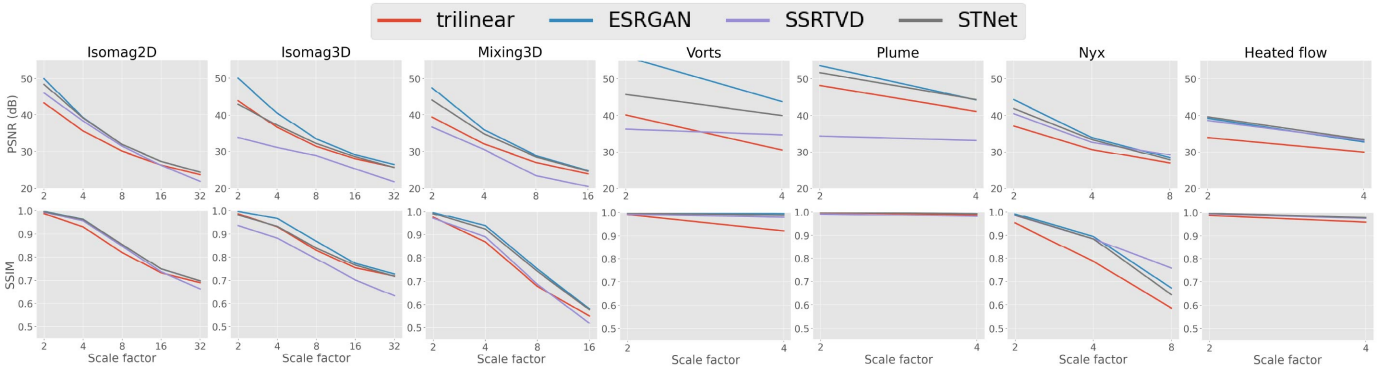


Fig. 4. Uniform SR PSNR (dB, top row) and SSIM (bottom row) for hierarchies trained with ESRGAN (blue), STNet (purple), and SSRTVD (gray) compared to bilinear/trilinear interpolation (red). The x-axis represents the scale factor, with scale factors larger than $2\times$ tested using multiple networks from the trained hierarchy in tandem.

learning $2\times$ upscaling on input data of downscaling level $i+1$, generating downscaling level i. Our loss function for any network during training is the L1 loss $\mathcal{L}_1(V, V^{GT})$, where V is the upscaled result from the model, and V^{GT} is the ground truth data.

4 Evaluation

We evaluate our method on seven datasets, described in Section 4.1. In Section 4.2, we describe hyperparameters and the training procedure. In Section 4.3, we compare our method against baselines for both uniform and hierarchical SR. In Section 4.3.2, we show that our hierarchical SR approach out-performs a baseline block-wise upscaling approach for hierarchical data, and that our upscaled volumes minimize seam artifacts. In Section 4.4, we apply our method for three use cases spanning data reduction, compute time reduction by upscaling adaptive resolution algorithm results, and compute resource reduction by upscaling low resolution simulations for a visualization preview of the high resolution result. All PSNR/SSIM metrics listed are calculated in the data space as opposed to image space. Additionally, whenever bilinear or trilinear interpolation is used, the values are assumed to be at cell centers instead of cell corners (`align_corners=False`), which consistently provides more accurate reconstruction.

4.1 Datasets

We experiment with seven scalar field datasets. Three are from Johns Hopkins Turbulence Databases (JHTDB) [60] that are time-varying results from a direct numerical simulation (DNS) for turbulent fluid flow. Information about each dataset is shown in Table 1. **Isotropic3D** is a 3D velocity magnitude dataset from a DNS of isotropic turbulent fluid flow at Reynolds number $R_\lambda \sim 433$, hosted in the "isotropic1024coarse" dataset in the JHTDB. We take the middle z-axis slice (z=512) of the Isotropic3D magnitude dataset to create an **isotropic2D** dataset for testing our approach with 2D data. The **mixing3D** dataset is the velocity magnitude field from the "mixing" dataset in JHTDB. The solar **plume** dataset is a velocity magnitude field from a simulation that examines the effect of the solar plume on the heat, momentum, and magnetic field of the sun. The **vorts** dataset is a 128^3 vorticity magnitude scalar field dataset from a pseudo-spectral simulation of vortex structures. **Nyx** is a cosmological simulation which is run with parameter settings uniformly sampled within suggested ranges by domain scientists: total matter density $OmM \in [0.12, 0.155]$, total density of baryons $OmB \in [0.0215, 0.0235]$, and Hubble constant $h \in [0.55, 0.85]$ [11], [61]. **Heated flow** is a 2D fluid simulation from the Gerris flow solver [62] created by Günther et al. [63] that simulates flow around a heated cylinder with Boussinesq approximation. All datasets are linearly scaled to [0.0,1.0] except for Nyx which is log scaled before being scaled to [0.0, 1.0]. All datasets are represented at single precision floating point accuracy to accommodate the architectures used.

TABLE 1
Dataset names, their sizes, the number of models $|G|$ in a hierarchy G for each dataset, the number of timesteps for training (N_{train}) and testing (N_{test}), and the number of iterations of training per model in the hierarchy.

Dataset	Volume size	$\lvert G\rvert$	N_{train}	N_{test}	iters.
Isomag2D	1024^2	5	400	100	20000
Isomag3D	1024^3	5	40	10	4000
Mixing3D	512^3	4	80	20	8000
Vorts	128^3	2	20	9	8000
Plume	512×128^2	2	20	9	15000
Nyx	256^3	3	170	100	17000
Heated flow	144×448	2	7530	7530	30120

4.2 Network training and hyperparameters

Three NN hierarchies are trained per dataset using the ESRGAN, STNet, and SSRTVD architectures. To train G_i in a hierarchy G, a ground truth volume V is downscaled to LR $V^{\mathbf{LR}}$ and HR $V^{\mathbf{HR}}$ (if necessary, i.e. i is not 0) by scale factors 2^{i+1} and 2^i, respectively. Then, the loss function is performed on $G_i(V^{\mathbf{LR}})$ and $V^{\mathbf{HR}}$.

The number of models trained in each hierarchy is shown in Table 1, column $|G|$, and the total number of iterations of training (with batch size 1) is listed in column "iters". We use Python 3.9 with PyTorch to train on an NVidia A100 40GB Tensor Core graphics card. The full

Authorized licensed use limited to: Argonne National Laboratory. Downloaded on September 14,2023 at 17:24:23 UTC from IEEE Xplore. Restrictions apply.
© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2022.3214420

resolution 3D data cannot fit in the GPU's memory during training, so we crop training volumes to 96^3 and use a random starting position for the cropping, augmented with random flipping in any subset of the spatial dimensions to further increase training diversity.

The SSRTVD models are constructed as implemented by the paper with no changes to the number of kernels per convolution. For our ESRGAN and STNet models, all layers (aside from output) use 96 kernels for a good mix between accuracy, GPU memory use, and training time. All convolution operations in all networks use reflection padding during training and inference.

The training time per iteration and the storage of each model are listed in Table 2. Since each architecture is the same size across all datasets, and the inputs are cropped to a fixed resolution, the models train at a similar time per iteration regardless of which level of detail they are training on. STNet trains the quickest, as it is the most lightweight architecture. ESRGAN trains quicker than SSRTVD, yet is a larger model with more parameters, likely due to SSRTVD's decision to be a wider network with more kernels per convolution instead of deeper network, which uses more convolution layers like ESRGAN does.

TABLE 2
Training time per iteration and storage cost for each the 2D/3D versions of the models tested with.

Model	t_{train} per iter.	Storage size per model
ESRGAN 2D	0.033 sec.	11.56 MB
SSRTVD 2D	0.041 sec.	8.72 MB
STNet 2D	0.017 sec.	5.89 MB
ESRGAN 3D	0.514 sec.	38.42 MB
SSRTVD 3D	0.641 sec.	26.98 MB
STNet 3D	0.451 sec.	21.43 MB

4.3 Baseline comparison

We compare our proposed approach in two steps. In Section 4.3.1, we isolate the neural network hierarchies by comparing the uniform grid SR performance of the hierarchies against bilinear/trilinear interpolation. In Section 4.3.2, we isolate our hierarchical SR algorithm by comparing the results of hierarchical SR with our method against the baseline blockwise approach.

4.3.1 Improved SR with trained hierarchy vs. linear interpolation

We perform uniform SR using the trained neural network hierarchies and trilinear interpolation on the test sets for each dataset, and report the median PSNR and SSIM in Figure 4. Our results show that the trained hierarchies consistently outperform bilinear/trilinear interpolation across all scale factors for both PSNR and SSIM aside from a few cases with the SSRTVD models specifically. The low performance of SSRTVD is not due to our method, as even the $2\times$ scale SSRTVD experiments that do not use our neural network hierarchy do not perform well. More experiments for training routines may improve results for that model specifically. Aside from those under-performing models, the results verify that using multiple trained NNs in tandem for SR factors of $4\times$ or higher still performs well which is important because the networks were not trained conditional on the output of previous networks.

4.3.2 Block-wise baseline vs. hierarchical super resolution

To evaluate our hierarchical SR algorithm, we generate SR-octrees according to the process described in Section 3.2 and upscale them with the baseline blockwise approach as well as our proposed method using the ESRGAN hierarchies, as they performed best over all datasets. Figure 5 shows that blockwise upscaling fails to upscale the octree data properly due to the small size of some octree blockswhich limits the information the network can gather from neighboring voxels, as discussed in Section 3.3. Our approach does not have this problem because our upscaling process uses the global domain at each step, minimizing the ratio of padding voxels to actual input voxels.

4.4 Use cases

We provide examples of three use cases for our hierarchical SR approach. In Section 4.4.1, we show how our hierarchical SR method can be used to improve STNet's data reduction capabilities by decomposing the spatial domain based on the presence of features instead of uniform downscaling. Section 4.4.2 shows how computation resources can be saved on the expensive FTLE field computation by increasing tolerated error in hierarchical FTLE calculation algorithms and using our hierarchical SR algorithm to approximate the HR uniform FTLE. Lastly, Section 4.4.3, we demonstrate that a trained hierarchy can be used to upscale LR simulation output for visualization, which can be used to save computation resources before running the higher resolution simulation.

4.4.1 Data reduction with SR frameworks

As compute power grows, simulations are run at larger resolutions. However, storage and bandwidth cannot keep up, creating a bottleneck that leads to the need for data reduction [27]. Super resolution methods are one such proposed technique for data reduction [2], [3], [5]. One example of an SR approach for data reduction is STNet, which is a state-of-the-art spatiotemporal SR neural network by Han et al. [5] that shows promising performance for data reduction compared to TTHRESH [64], a state-of-the-art compressor. In the data reduction pipeline for STNet, spatiotemporal volumes are uniformly downscaled in both space and time dimensions to a LR before saved to storage. The data can then be reconstructed via inferring the HR data through the trained STNet model.

We show that by adding our hierarchical SR capability to STNet's spatial component, the reconstruction quality for the same data reduction factors increases. In STNet's approach, data is uniformly downscaled by $4\times$ in the spatial domain, which is a data reduction factor of $64\times$ before lossless compression. In our approach with a hierarchical STNet, we perform hierarchical downscaling as defined in Section 3.2 with settings $\epsilon = 0.02285$, $\texttt{min_dwnscl_lvl} = 1$, $\texttt{max_dwnscl_lvl} = 3$, and $\texttt{min_chunk} = 2$, which creates a hierarchical version that has an equivalent data reduction factor (before lossless compression) of $64.03\times$.

Authorized licensed use limited to: Argonne National Laboratory. Downloaded on September 14,2023 at 17:24:23 UTC from IEEE Xplore. Restrictions apply.
© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2022.3214420

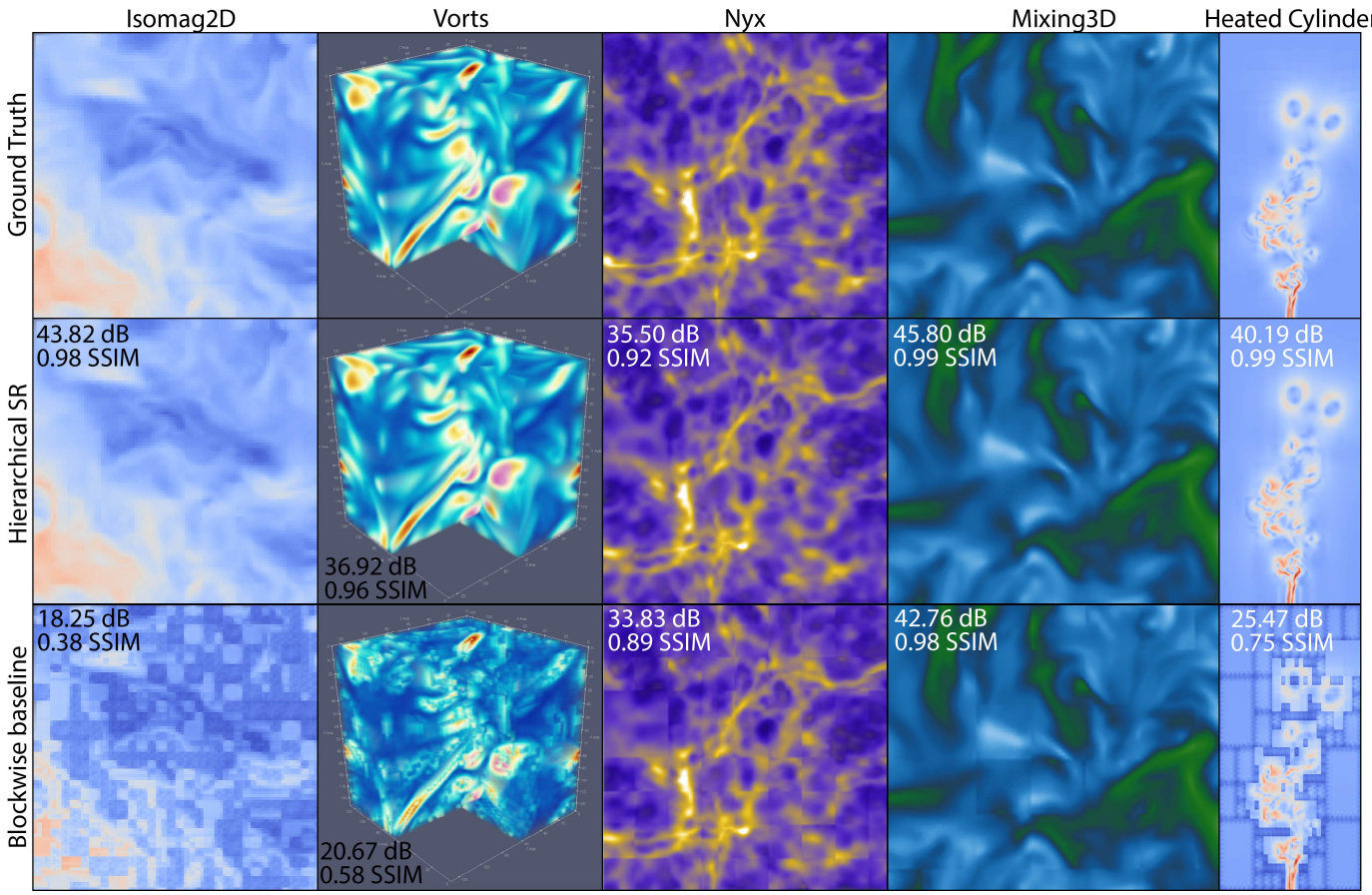


Fig. 5. Comparing the baseline blockwise upscaling against our hierarchical SR algorithm with a pre-trained ESRGAN hierarchy. Artifacts are visible in the blockwise baseline, creating seams along block boundaries.

We use the same STNet model trained on the plume dataset which is reported in Section 4.3.1. In Figure 6, we show the results of using the pre-trained hierarchy for either uniform ($4\times$) SR on the uniformly downscaled data in (a) versus hierarchical upscaling for the SR-octree of the same storage size in (c). We see that our hierarchical approach improves PSNR by roughly 5 dB for the same data reduction factor. We visualize the error of each method in (e), with the uniform upscaling error shown in orange and the hierarchical upscaling error shown in blue. The uniform upscaling has most of its error in the center more turbulent regions of the volume, whereas the hierarchical upscaling has its error further toward the homogeneous regions that are not of interest.

For further evaluation, we show that our approach can give higher data reduction at the same target quality. We identify an SR-octree that causes our hierarchical SR approach to have a similar reconstruction quality to the uniform SR from the last experiment with parameters $\epsilon = 0.068$, `min_dwnscl_lvl` $= 1$, `max_dwnscl_lvl` $= 3$, and `min_chunk` $= 2$. Shown in Figure 7, our hierarchical SR approach was able to achieve 44.13 dB PSNR on this octree, which is roughly the same quality as uniform SR (44.42 dB). However, our approach achieves this quality with a data reduction factor of $148.67\times$, which is $2.32\times$ higher than the uniform scale factor data reduction ratio of $64\times$. The error volume visualization shows a similar trend to the last experiment - uniform SR has a majority of its error (orange) toward the base of the plume with high turbulence, whereas our hierarchical approach has more error (blue) near the coarse regions of the plume.

4.4.2 *Upscaling hierarchically computed fields*

Some expensive algorithms can be computed in a hierarchical manner to adaptively allocate resources for a faster result than the same algorithm with a uniform grid. These algorithms will often use a metric for refining the spatial domain adaptively, such as tolerable error. Our approach can benefit these algorithms by allowing them to be run with a larger tolerable error to save more time, and performing hierarchical SR on the result for a high resolution approximation.

One such algorithm is FTLE, which is a useful yet expensive computation for fluid dynamics researchers to visualize Lagrangian coherent structures in flow data. The FTLE field is computed by

$$\sigma_{t_0}^{T} = \frac{1}{T} \ln \left(\lambda_{\max} \left(\frac{\partial \phi_{t_0}^{T}(x)}{\partial x}' \cdot \frac{\partial \phi_{t_0}^{T}(x)}{\partial x} \right) \right), \qquad (1)$$

where x is a position, ϕ is the flow map for the fluid data that maps a particle at position x at start time t_0 to a position $\phi_{t_0}^{T}(x)$ after integration length T, and $\lambda_{\max}(\cdot)$ is the

Authorized licensed use limited to: Argonne National Laboratory. Downloaded on September 14,2023 at 17:24:23 UTC from IEEE Xplore. Restrictions apply.
© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2022.3214420

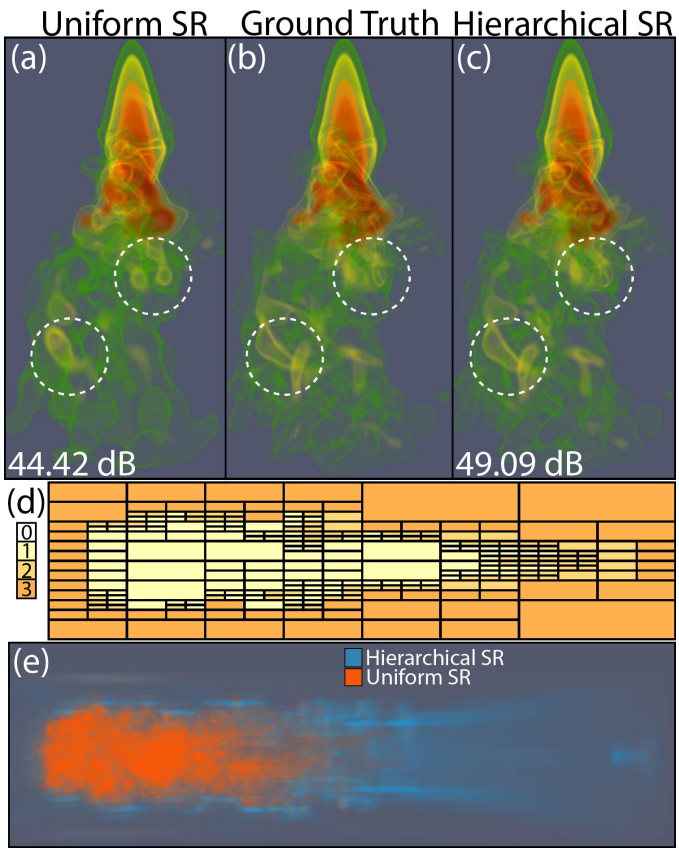


Fig. 6. Comparing $4\times$ uniform SR against hierarchical SR at the same data reduction factor ($64\times$). Our approach (c) performs better than uniform SR (a) at the same data reduction factor when comparing to the ground truth (b). The downscaling levels (d) show which regions are more fine (closer to white) and which are more coarse (closer to orange). The error volume (e) shows uniform SR has error near the more turbulent region of interest and our approach has more error in the coarse outer regions where it downscaled more.

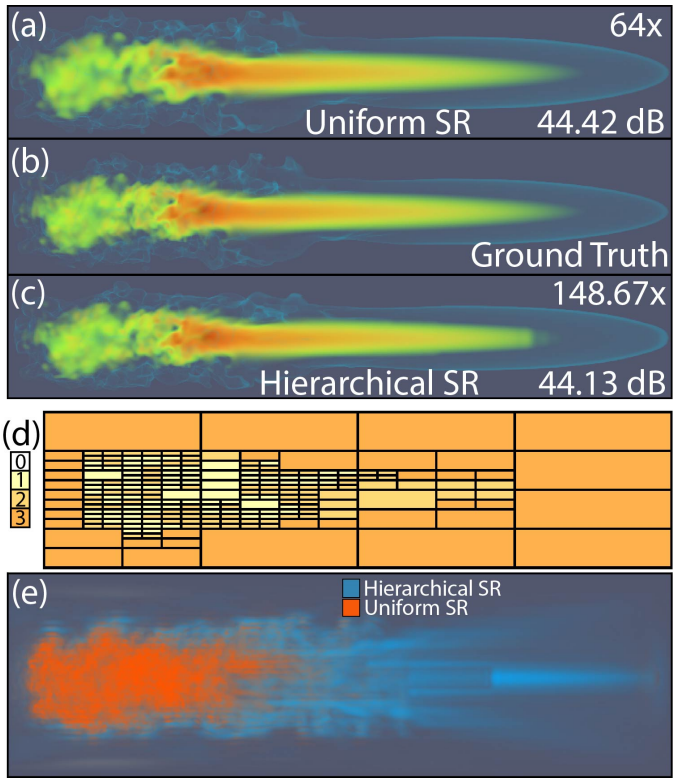


Fig. 7. Comparing $4\times$ uniform SR (a) with hierarchical SR (c) when holding the output quality constant. Reduction factors are listed in the top right corner of the image. Both methods achieve just above 44 dB PSNR. Hierarchical SR's reduction rate is $2.32\times$ larger than uniform SR at the same quality. The downscaling levels (d) show which regions are more fine (closer to white) and which are more coarse (closer to orange). The error volume (e) shows that hierarchical SR error is distributed in the coarse outer region of the plume, since it was downscaled further.

maximum eigenvalue for the matrix it operates on. Given time-varying vector field data, creating a flow map ϕ can be expensive, and so the FTLE field calculation can also be expensive. In addition, it can be difficult to know what integration length T will reveal interesting features about the flow data.

To accelerate FTLE calculation, researchers have created hierarchical methods that adaptively sample the domain to focus on FTLE ridgelines [20] or based on viewing direction [21]. We demonstrate that the FTLE calculation can be performed even quicker by making the subdivision requirement less strict and then using our hierarchical SR algorithm to approximate the HR result. When an approximated HR result has features of interest, the scientist may choose to expend computing resources and time to get the full resolution result for analysis.

In our test, we use a dataset from a simulation with a heated cylinder with Boussinesq approximation [62], [63], which is 2001 timesteps with dimensions 150×450. We compute at total of roughly 15,000 uniform grid HR FTLE fields over $T \in [1, 5, 10, 25, 50, 100, 250, 500]$ and all possible t_0 for each T. The dataset is randomly divided into 50% for training the network and 50% for a test set. We use the trained ESRGAN hierarchy, with qualitative results shown in Figure 4.

We use our downscaling approach as defined in Section 3.2 to create downscaled hierarchical data as a test case that is agnostic to the specific hierarchical FTLE computation method. In our experiment, we use two FTLE fields calculated with $t_0 = 730$, $T = 25$ and $t_0 = 805$, $T = 100$, which took 3.94 and 17.64 seconds to calculate on a uniform grid at full resolution, respectively. We use settings `min_downscaling_level=1`, `max_downscaling_level=5`, `min_chunk=2`, $\epsilon = 0.0193$ to generate a LR test case for the $T = 25$ example, and `min_downscaling_level=1`, `max_downscaling_level=5`, `min_chunk=2`, $\epsilon = 0.02$ for the $T = 100$ example, shown in Figure 8. The LR adaptive versions with our hierarchical super resolution speed up computation of the FTLE field by $16\times$ and $9.46\times$, for an estimated FTLE computation time of 0.25 seconds and 1.86 seconds to calculate each timestep, for $T = 25$ and $T = 100$, respectively. The upscaled results achieve higher reconstruction accuracy than the nearest neighbor resampled LR hierarchical FTLE field data, and show a strong resemblance to the ground truth HR FTLE data.

Authorized licensed use limited to: Argonne National Laboratory. Downloaded on September 14,2023 at 17:24:23 UTC from IEEE Xplore. Restrictions apply.
© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2022.3214420

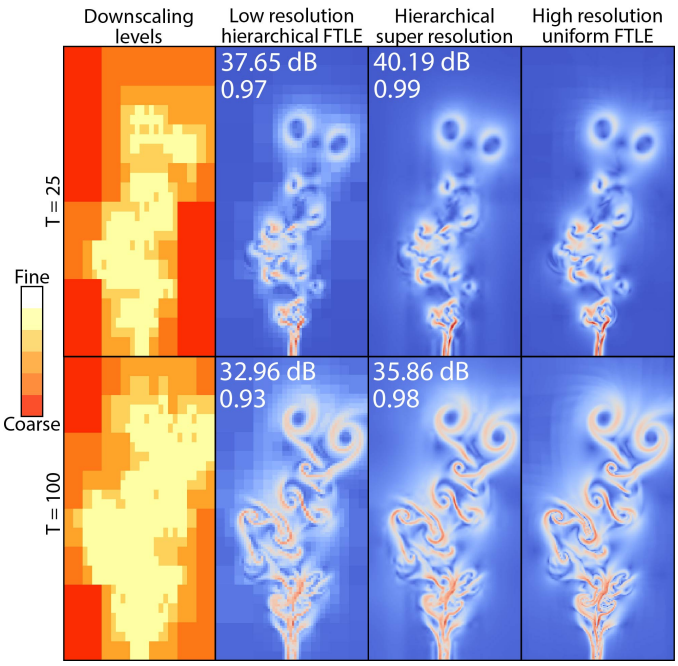


Fig. 8. Upscaling hierarchical FTLE data.

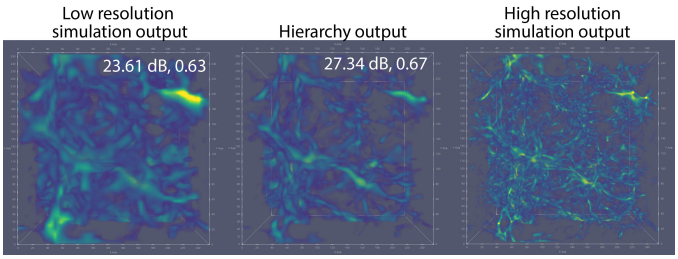


Fig. 9. Upscaling LR Nyx simulation output with trained neural networks for a visualization preview of what the HR simulation might look like.

4.4.3 *Visualizing upscaled simulation results*

Scientists run simulations of physical phenomena which may be analyzed and visualized for a deeper understanding of the science. The simulations may run on a fixed regular grid, in which case higher resolution simulations will take longer to finish computing than lower resolution simulations. To save computation resources, running a lower-resolution simulation may be preferable in order to get insight into the output, before spending computation resources on the higher resolution simulation. Super resolution methods can assist here by upscaling the low-resolution simulation output, for a visualization of what the high-resolution simulation may look like, as shown in state of the art super resolution work for scientific data [2], [3], [6].

To evaluate our work with this use case, we use the Nyx cosmological simulation [11], which uses input parameters (defined in Section 4.1) to generate volumes representing the log density of dark matter. Running the simulation at 256^3 for a single parameter setting takes roughly 65 minutes, whereas running at 64^3 takes only 110 seconds. Not all parameter settings may be of interest, so to avoid unnecessary computation, we use super resolution to visualize upscaled low-resolution simulation results of certain parameters before choosing to expend computation resources on running the high resolution simulation.

We run Nyx with 30 parameters settings sampled randomly within the ranges listed in Section 4.1 at both 256^3 and 64^3 to generate 30 ensemble members at both low- and high-resolution. Of those, 10 are randomly sampled for training the ESRGAN hierarchy and the other 20 for testing.

Over the 20 test cases, the model scored an average 26.47 dB PSNR and 0.65 SSIM, while trilinear interpolation scored an average 23.64 dB PSNR and 0.62 SSIM. The hierarchy also achieves a lower maximum relative error (MRE) of 0.47 compared to trilinear interpolation's result of 0.57. Figure 9 displays volume rendered images of LR simulation output, the neural network's approximated HR output, and the HR simulation output for parameter settings $OmM = 0.14166$, $OmB = 0.02288$, $h = 0.61201$. The model achieves better PSNR and SSIM than low resolution output on the example, and the volume rendering image resembles the HR output closer than the low-resolution simulation output. Results may improve if more than only 10 simulation output pairs are used to train the models. Alternatively, GAN training may improve the upscaled result's visualization to more closely resemble the high resolution output's tendril-like features at the price of lower PSNR. Although this experiment does not use our hierarchical SR algorithm, the hierarchy of neural networks can still be useful. The simulation can be ran at any supported scale factor ($2\times, 4\times, ...$), allowing for a scalable time vs. accuracy trade-off.

5 Conclusion, Limitations, and Future Work

In this paper, we present a hierarchical SR algorithm for upscaling hierarchical data using NNs while minimizing seam artifacts on octree node boundaries. The hierarchical SR algorithm is comprised of a downscaling process followed by an upscaling process to reduce the effect of seam artifacts between nodes. We also present a hierarchy of SR NNs that can be used with our hierarchical SR algorithm to improve SR performance over bilinear/trilinear interpolation methods. We show that hierarchical SR has benefits over uniform SR, especially when there are coarse regions inside the volume. Over three use cases, we show how our hierarchical SR approach can benefit scientific computing.

One limitation of our method is that using $2\times$ SR NNs limits data shape, since we can only perform SR with a factor of 2^i, for $1 \leq i \leq |G|$. Therefore, data can only be downscaled (and upscaled) by up to the largest factor of two of a spatial dimension, which can limit data use. Resampling the data to a compatible size is a short term solution (as we did for the plume dataset), but to support arbitrary spatial dimension sizes, other kinds of NN SR architecture are needed. Another limitation is that using multiple networks in our NN hierarchies will increase the storage overhead of the saved networks. To reduce this overhead, a single network could be trained to perform super resolution for any scale, but it may not perform as well as the hierarchy. Lastly, we want to recognize that super resolution techniques may introduce errors and artifacts,

Authorized licensed use limited to: Argonne National Laboratory. Downloaded on September 14,2023 at 17:24:23 UTC from IEEE Xplore. Restrictions apply.
© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2022.3214420

which may be critical for specific use cases. We attempt to mitigate these drawbacks by using an error-bounded SR-octree and using ample training data so the network has learned how to upscale low-resolution features to the correct high-resolution feature. We observe our results tend to smooth out high-frequency features as opposed to generating false features.

In the future, our method could be improved to support non-factor-of-two volumes by using a different NN hierarchy structure, or by using continuous SR through models like LIIF [65]. Our approach may also see benefits by adapting to a more general hierarchical representation such as a k-d tree instead of an octree. Lastly, our approach may be more attractive in the data-reduction setting with additional research toward error-bounded super resolution models, similar to current state of the art compressors [64], [66], or physics informed neural networks [67].

ACKNOWLEDGMENTS

This work is supported in part by the US Department of Energy SciDAC program DE-SC0021360, National Science Foundation Division of Information and Intelligent Systems IIS-1955764, and National Science Foundation Office of Advanced Cyberinfrastructure OAC-2112606. This work is also supported by Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357, program manager Margaret Lentz.

REFERENCES

[1] Z. Zhou, Y. Hou, Q. Wang, G. Chen, J. Lu, Y. Tao, and H. Lin, "Volume upscaling with convolutional neural networks," in *Proc. 2017 Computer Graphics International Conference*, 2017, pp. 1–6.

[2] J. Han and C. Wang, "SSR-TVD: Spatial Super-Resolution for Time-Varying Data Analysis and Visualization," *IEEE Transactions on Visualization and Computer Graphics*, 2020, Early Access.

[3] L. Guo, S. Ye, J. Han, H. Zheng, H. Gao, D. Z. Chen, J. Wang, and C. Wang, "SSR-VFD: Spatial Super-Resolution for Vector Field Data Analysis and Visualization," in *Proc. 2020 IEEE Pacific Visualization Symposium*, 2020, pp. 71–80.

[4] J. Han and C. Wang, "TSR–TVD: Temporal Super-Resolution for Time-Varying Data Analysis and Visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 205–215, 2020.

[5] J. Han, H. Zheng, D. Z. Chen, and C. Wang, "STNet: An End-to-End Generative Framework for Synthesizing Spatiotemporal Super-Resolution Volumes," *IEEE Transactions on Visualization and Computer Graphics*, 2021, Early Access.

[6] K. Fukami, K. Fukagata, and K. Taira, "Super-Resolution Reconstruction of Turbulent Flows With Machine Learning," *Journal of Fluid Mechanics*, vol. 870, p. 106–120, 2019.

[7] ——, "Machine-learning-based spatio-temporal super resolution reconstruction of turbulent flows," *Journal of Fluid Mechanics*, vol. 909, p. A9, 2021.

[8] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a Deep Convolutional Network for Image Super-Resolution," in *Proc. of 2014 European Conference on Computer Vision*, 2014, pp. 184–199.

[9] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, and C. C. Loy, "ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks," in *Computer Vision – ECCV 2018 Workshops*, L. Leal-Taixé and S. Roth, Eds. Springer, 2019, pp. 63–79.

[10] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[11] A. S. Almgren, J. B. Bell, M. J. Lijewski, Z. Lukić, and E. V. Andel, "Nyx: A Massively Parallel AMR Code for Computational Cosmology," *The Astrophysical Journal*, vol. 765, no. 1, pp. 39–53, 2013.

[12] W. Zhang, A. Almgren, V. Beckner, J. Bell, J. Blaschke, C. Chan, M. Day, B. Friesen, K. Gott, D. Graves, M. Katz, A. Myers, T. Nguyen, A. Nonaka, M. Rosso, S. Williams, and M. Zingale, "AMReX: a framework for block-structured adaptive mesh refinement," *Journal of Open Source Software*, vol. 4, no. 37, pp. 1370–1374, 2019.

[13] M. Adams, P. Colella, D. T. Graves, J.N. Johnson, N.D. Keen, T. J. Ligocki. D. F. Martin. P.W. McCorquodale, D. Modiano. P.O. Schwartz, T.D. Sternberg and B. Van Straalen, "Chombo Software Package for AMR Applications – Design Document," lawrence Berkeley National Laboratory Technical Report LBNL-6616E.

[14] C. C. Kiris, M. F. Barad, J. A. Housman, E. Sozer, C. Brehm, and S. Moini-Yekta, "The LAVA Computational Fluid Dynamics Solver," in *52nd Aerospace Sciences Meeting*, 2014.

[15] B. W. O'shea, G. Bryan, J. Bordner, M. L. Norman, T. Abel, R. Harkness, and A. Kritsuk, "Introducing Enzo, an AMR cosmology application," in *Adaptive Mesh Refinement—Theory and Applications*, T. Plewa, T. Linde, and V. G. Weirs, Eds. Springer, 2005, pp. 341–349.

[16] A. Knoll, I. Wald, S. Parker, and C. Hansen, "Interactive Isosurface Ray Tracing of Large Octree Volumes," in *Proc. 2006 IEEE Symposium on Interactive Ray Tracing*, 2006, pp. 115–124.

[17] J. Wilhelms and A. Van Gelder, "Octrees for Faster Isosurface Generation," *ACM Transactions on Graphics*, vol. 11, no. 3, pp. 201–227, 1992.

[18] A. Knoll, "A Survey of Octree Volume Rendering Methods," 2006, Scientific Computing and Imaging Institute, University of Utah.

[19] B. Liu, G. J. Clapworthy, F. Dong, and E. C. Prakash, "Octree Rasterization: Accelerating High-Quality Out-Of-Core GPU volume rendering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 10, pp. 1732–1745, 2013.

[20] F. Sadlo and R. Peikert, "Efficient Visualization of Lagrangian Coherent Structures by Filtered AMR Ridge Extraction," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1456–1463, 2007.

[21] s. barakat, C. Garth, and X. Tricoche, "Interactive computation and rendering of finite-time lyapunov exponent fields," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 8, pp. 1368–1380, 2012.

[22] J. N. Martel, D. B. Lindell, C. Z. Lin, E. R. Chan, M. Monteiro, and G. Wetzstein, "ACORN: Adaptive Coordinate Networks for Neural Representation," *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 2021, Early Access.

[23] D. Hoang, B. Summa, H. Bhatia, P. Lindstrom, P. Klacansky, W. Usher, P. Bremer, and V. Pascucci, "Efficient and Flexible Hierarchical Data Layouts for a Unified Encoding of Scalar Field Precision and Resolution," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 02, pp. 603–613, 2020.

[24] P. Ljung, C. Lundström, and A. Ynnerman, "Multiresolution interblock interpolation in direct volume rendering," in *Proc. 2006 Eurographics/IEEE VGTC Symposium on Visualization*, 2006, pp. 259–266.

[25] I. Wald, C. Brownlee, W. Usher, and A. Knoll, "CPU volume rendering of adaptive mesh refinement data," in *Proc. SIGGRAPH Asia 2017 Symposium on Visualization*, 2017, pp. 1–8.

[26] F. Wang, I. Wald, Q. Wu, W. Usher, and C. R. Johnson, "CPU Isosurface Ray Tracing of Adaptive Mesh Refinement Data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, pp. 1142–1151, 2019.

[27] D. Hoang, P. Klacansky, H. Bhatia, P. Bremer, P. Lindstrom, and V. Pascucci, "A Study of the Trade-off Between Reducing Precision and Reducing Resolution for Data Analysis and Visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, pp. 1193–1203, 2019.

[28] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang, "Fast and Accurate Image Super-Resolution with Deep Laplacian Pyramid Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 11, pp. 2599–2613, 2019.

[29] H. Wang, D. Su, C. Liu, L. Jin, X. Sun, and X. Peng, "Deformable non-local network for video super-resolution," *IEEE Access*, vol. 7, pp. 177 734–177 744, 2019.

[30] Z. Wang, J. Chen, and S. C. H. Hoi, "Deep learning for image super-resolution: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020, to appear.

[31] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proc. 2016 IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

Authorized licensed use limited to: Argonne National Laboratory. Downloaded on September 14,2023 at 17:24:23 UTC from IEEE Xplore. Restrictions apply.
© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2022.3214420

[32] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network," in *Proc. 2017 IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 105–114.

[33] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network," in *Proc. 2016 IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1874–1883.

[34] S. Weiss, M. Chu, N. Thuerey, and R. Westermann, "Volumetric Isosurface Rendering with Deep Learning-Based Super-Resolution," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 6, pp. 3064–3078, 2021.

[35] S. Weiss, M. Işık, J. Thies, and R. Westermann, "Learning Adaptive Sampling and Reconstruction for Volume Visualization," *IEEE Transactions on Visualization and Computer Graphics*, 2021, Early Access.

[36] Y. Xie, E. Franz, M. Chu, and N. Thuerey, "tempoGAN: A Temporally Coherent, Volumetric GAN for Super-Resolution Fluid Flow," *ACM Transactions on Graphics*, vol. 37, no. 4, pp. 95:1–95:15, 2018.

[37] K. Höhlein, M. Kern, T. Hewson, and R. Westermann, "A comparative study of convolutional neural network models for wind field downscaling," *Meteorological Applications*, vol. 27, no. 6, p. e1961, 2020.

[38] J. Jakob, M. Gross, and T. Günther, "A Fluid Flow Data Set for Machine Learning and its Application to Neural Flow Map Interpolation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 2, pp. 1279–1289, 2021.

[39] D. Meagher, "Geometric Modeling Using Octree Encoding," *Computer Graphics and Image Processing*, vol. 19, no. 2, pp. 129–147, 1982.

[40] E. Gobbetti, F. Marton, and J. Iglesias Guitián, "A single-pass GPU ray casting framework for interactive out-of-core rendering of massive volumetric datasets," *The Visual Computer*, vol. 24, pp. 797–806, 2008.

[41] M. Hadwiger, J. Beyer, W.-K. Jeong, and H. Pfister, "Interactive volume exploration of petascale microscopy data streams using a visualization-driven virtual memory approach," *IEEE Transactions on Visualization & Computer Graphics*, vol. 18, no. 12, pp. 2285–2294, 2012.

[42] T. Fogal, A. Schiewe, and J. Krüger, "An Analysis of Scalable GPU-Based Ray-Guided Volume Rendering," in *In Proceedings of IEEE Symposium on Large-Scale Data Analysis and Visualization*, 2013, pp. 43–51.

[43] F. Losasso, F. Gibou, and R. Fedkiw, "Simulating Water and Smoke with an Octree Data Structure," in *Proc. 2004 ACM SIGGRAPH*, 2004, pp. 457–462.

[44] S. Popinet, "Gerris: A Tree-Based Adaptive Solver for the Incompressible Euler Equations in Complex Geometries," *Journal of Computational Physics*, vol. 190, no. 2, pp. 572–600, 2003.

[45] M. J. Berger and J. Oliger, "Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations," *Journal of Computational Physics*, vol. 53, no. 3, pp. 484–512, 1984.

[46] M. Berger and P. Colella, "Local Adaptive Mesh Refinement for Shock Hydrodynamics," *Journal of Computational Physics*, vol. 82, no. 1, pp. 64–84, 1989.

[47] F. Velasco and J. Torres, "Cell Octrees: A New Data Structure for Volume Modeling and Visualization," in *Proc. 2001 Vision Modeling and Visualization Conference*, 2001, pp. 151–158.

[48] H. Bhatia, D. Hoang, G. Morrison, W. Usher, V. Pascucci, P.-T. Bremer, and P. Lindstrom, "AMM: Adaptive Multilinear Meshes," arXiv:2007.15219 [cs.GR], 2020.

[49] M. Ainsworth, O. Tugluk, B. Whitney, and S. Klasky, "Multilevel techniques for compression and reduction of scientific data—The univariate case," *Computing and Visualization in Science*, vol. 19, no. 5, pp. 65–76, 2018.

[50] X. Liang, B. Whitney, J. Chen, L. Wan, Q. Liu, D. Tao, J. Kress, D. Pugmire, M. Wolf, N. Podhorszki, and S. Klasky, "MGARD+: Optimizing Multilevel Methods for Error-bounded Scientific Data Reduction," 2020.

[51] G. Riegler, A. O. Ulusoy, and A. Geiger, "OctNet: Learning Deep 3D Representations at High Resolutions," in *Proc. 2017 IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6620–6629.

[52] M. Tatarchenko, A. Dosovitskiy, and T. Brox, "Octree Generating Networks: Efficient Convolutional Architectures for High-resolution 3D Outputs," in *Proc. 2017 IEEE International Conference on Computer Vision*, 2017, pp. 2107–2115.

[53] T. Takikawa, J. Litalien, K. Yin, K. Kreis, C. Loop, D. Nowrouzezahrai, A. Jacobson, M. McGuire, and S. Fidler, "Neural Geometric Level of Detail: Real-time Rendering with Implicit 3D Shapes," in *Proc. 2021 Conference on Computer Vision and Pattern Recognition*, 2021.

[54] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced Deep Residual Networks for Single Image Super-Resolution," in *Proc. of 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 07 2017, pp. 1132–1140.

[55] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proc. 2017 IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 105–114.

[56] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a Deep Convolutional Network for Image Super-Resolution," in *Proc. 2014 European Conference on Computer Vision*, 2014, pp. 184–199.

[57] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced Deep Residual Networks for Single Image Super-Resolution," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 1132–1140.

[58] J. Kim, J. K. Lee, and K. M. Lee, "Accurate Image Super-Resolution Using Very Deep Convolutional Networks," in *Proc. 2016 IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1646–1654.

[59] C. Dong, C. C. Loy, and X. Tang, "Accelerating the Super-Resolution Convolutional Neural Network," in *Proc. of 2016 European Conference on Computer Vision*, 2016, pp. 391–407.

[60] Y. Li, E. Perlman, M. Wan, Y. Yang, C. Meneveau, R. Burns, S. Chen, A. Szalay, and G. Eyink, "A public turbulence database cluster and applications to study Lagrangian evolution of velocity increments in turbulence," *Journal of Turbulence*, vol. 9, 2008.

[61] W. He, J. Wang, H. Guo, K.-C. Wang, H.-W. Shen, M. Raj, Y. S. G. Nashed, and T. Peterka, "InSituNet: Deep Image Synthesis for Parameter Space Exploration of Ensemble Simulations," *IEEE transactions on visualization and computer graphics*, vol. 26, no. 1, pp. 23——33, 2020.

[62] S. Popinet, "Free computational fluid dynamics," *ClusterWorld*, vol. 2, no. 6, 2004. [Online]. Available: http://gfs.sf.net/

[63] T. Günther, M. Gross, and H. Theisel, "Generic objective vortices for flow visualization," *ACM Transactions on Graphics (Proc. SIGGRAPH)*, vol. 36, no. 4, pp. 141:1–141:11, 2017.

[64] R. Ballester-Ripoll, P. Lindstrom, and R. Pajarola, "TTHRESH: Tensor Compression for Multidimensional Visual Data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 9, pp. 2891–2903, 2020.

[65] K. Simonyan and A. Zisserman, "Learning Continuous Image Representation with Local Implicit Image Function," in *Proc. of 2021 Conference on Computer Vision and Pattern Recognition*, 2021, pp. 1–11.

[66] J. Liu, S. Di, K. Zhao, S. Jin, D. Tao, X. Liang, Z. Chen, and F. Cappello, "Exploring autoencoder-based error-bounded compression for scientific data," in *Proc. 2021 IEEE International Conference on Cluster Computing, Cluster 2021*, 2021, pp. 294–306.

[67] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations," *arXiv preprint arXiv:1711.10561*, 2017.



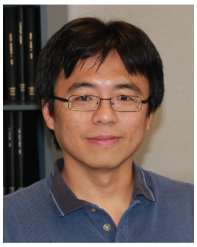
Skylar W. Wurster is a 4th year Ph.D. student studying under Professor Han-Wei Shen as part of his GRAVITY research group at The Ohio State University in Columbus, Ohio. He also collaborates with mentors Hanqi Guo at Ohio State University and Tom Peterka at Argonne National Lab in Lemont, Illinois. His research interests span deep learning, scientific data visualization, and computer games/graphics.

Authorized licensed use limited to: Argonne National Laboratory. Downloaded on September 14,2023 at 17:24:23 UTC from IEEE Xplore. Restrictions apply.
© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2022.3214420



Hanqi Guo received the BS degree in mathematics and applied mathematics from the Beijing University of Posts and Telecommunications in 2009 and the PhD degree in computer science from Peking University in 2014. He is an Associate Professor at the Department of Computer Science and Engineering in the Ohio State University. His research interests include data analysis, visualization, and machine learning for scientific data. He is an awardee of the DOE Early Career Research Program (ECRP) in 2022 and received multiple best paper awards in premiere visualization conferences.



Han-Wei Shen is a full professor at the Ohio State University. He received his B.S. degree from the Department of Computer Science and Information Engineering at National Taiwan University in 1988, his M.S. degree in computer science from the State University of New York at Stony Brook in 1992, and his Ph.D. degree in computer science from the University of Utah in 1998. From 1996 to 1999, he was a research scientist at NASA Ames Research Center in Mountain View California. His primary research interests are scientific visualization and computer graphics. He is a winner of the National Science Foundation's CAREER award and U.S. Department of Energy's Early Career Principal Investigator Award. He also won the Outstanding Teaching award twice in the Department of Computer Science and Engineering at the Ohio State University.



Tom Peterka is a computer scientist at Argonne National Laboratory, scientist at the University of Chicago Consortium for Advanced Science and Engineering (CASE), and fellow of the Northwestern Argonne Institute for Science and Engineering (NAISE). His research interests are large-scale parallel in situ analysis of scientific data. Recipient of the 2017 DOE early career award and five best paper awards, Peterka has published over 100 peer-reviewed articles and papers since earning his Ph.D. in computer science from the University of Illinois at Chicago in 2007.



Jiayi Xu is a research scientist at Meta AI. His research interests include high-performance data analysis, visualization, and machine learning. He is the recipient of the best paper award in the 14th IEEE Pacific Visualization Symposium. He received his Ph.D. degree in computer science and engineering from The Ohio State University in 2021 and his B.E. degree in computer science and technology from Chu Kochen Honors College of Zhejiang University in 2014.

Authorized licensed use limited to: Argonne National Laboratory. Downloaded on September 14,2023 at 17:24:23 UTC from IEEE Xplore. Restrictions apply.
© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.