F-Hash: Feature-Based Hash Design for Time-Varying Volume Visualization via Multi-Resolution Tesseract Encoding

Jianxin Sun, David Lenz, Hongfeng Yu, and Tom Peterka

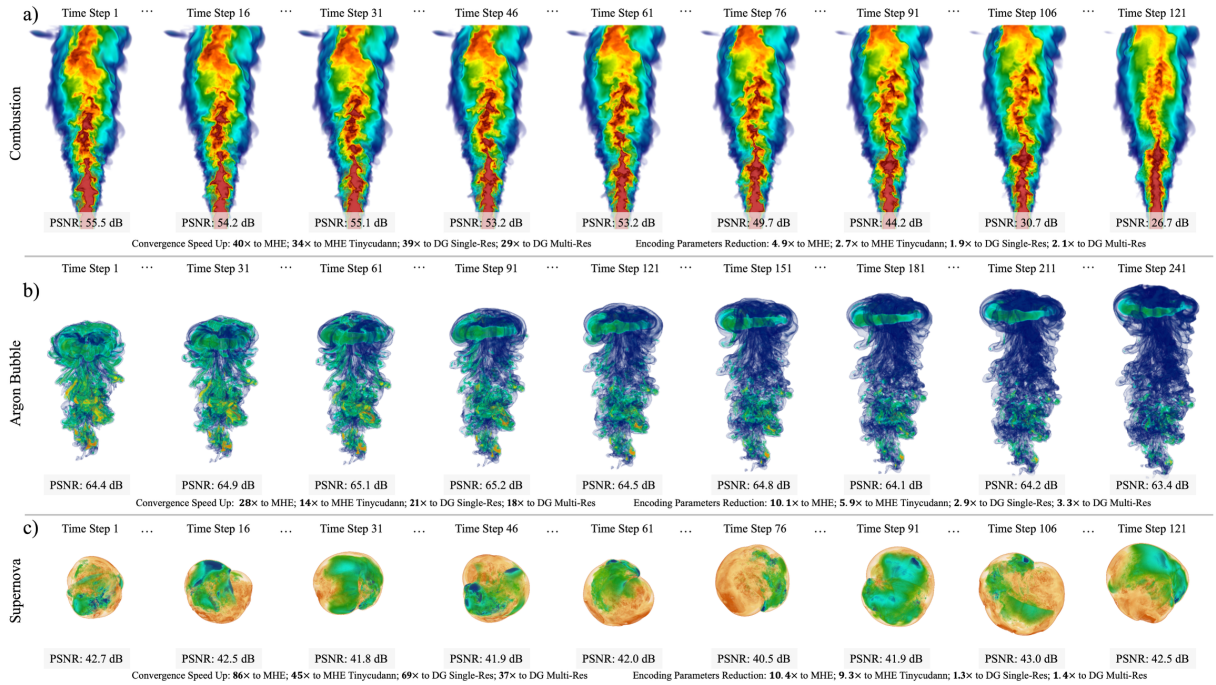


Fig. 1: Time-varying volume visualization via proposed F-Hash multi-resolution tesseract encoding on Combustion(a), Argon Bubble (b), and Supernova (c) datasets. F-Hash achieves high reconstruction accuracy with shorter convergence time by order of magnitude ($10\times$–$100\times$) and fewer encoding parameters compared to existing input encoding approaches.

Abstract—Interactive time-varying volume visualization is challenging due to its complex spatiotemporal features and sheer size of the dataset. Recent works transform the original discrete time-varying volumetric data into continuous Implicit Neural Representations (INR) to address the issues of compression, rendering, and super-resolution in both spatial and temporal domains. However, training the INR takes a long time to converge, especially when handling large-scale time-varying volumetric datasets. In this work, we proposed F-Hash, a novel feature-based multi-resolution Tesseract encoding architecture to greatly enhance the convergence speed compared with existing input encoding methods for modeling time-varying volumetric data. The proposed design incorporates multi-level collision-free hash functions that map dynamic 4D multi-resolution embedding grids without bucket waste, achieving high encoding capacity with compact encoding parameters. Our encoding method is agnostic to time-varying feature detection methods, making it a unified encoding solution for feature tracking and evolution visualization. Experiments show the F-Hash achieves state-of-the-art convergence speed in training various time-varying volumetric datasets for diverse features. We also proposed an adaptive ray marching algorithm to optimize the sample streaming for faster rendering of the time-varying neural representation.

Index Terms—Time-varying volume, volume visualization, input encoding, deep learning

1 Introduction

Numerous scientific disciplines involve intrinsically time-dependent phenomena. A substantial proportion of data produced by scientific simulations consists of time-varying volumetric data spanning fields such as computational fluid dynamics, combustion modeling, meteorological simulations, cosmological studies, and biomolecular simulations. However, the high resolution in both spatial and temporal domains results in extremely large-scale time-varying volumetric data, making interactive visualization significantly challenging.

Recent works leverage the neural network to model the discrete time-varying volumetric data into a continuous Implicit Neural Representation (INR), which can be accelerated on modern GPU, achieving state-of-the-art performance in compression, rendering, and super-resolution tasks. Multilayer perception (MLP)-based INR, as a universal approximator, provides an efficient way of representing high-dimensional data, enabling efficient compression [14, 39] of large-scale time-varying data for I/O intensive operations. The continuous nature of INR provides infinite super-resolution [11–13, 15, 40] on both spatial and temporal

- *Jianxin Sun and Hongfeng Yu are with the University of Nebraska-Lincoln. E-mail: jianxin.sun@huskers.unl.edu, yu@cse.unl.edu.*
- *David Lenz and Tom Peterka are with Argonne National Laboratory. E-mail: dlenz@anl.gov, tpeterka@mcs.anl.gov.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

domains. INR can also be trained for specific visualization tasks as generative models for fast rendering through novel view synthesis [5,34,48] or efficient out-of-core data management [33, 35].

Although INR gives outstanding performance in various visualization tasks, its core limitation lies in slow convergence, often requiring hours or even days of training for complex datasets. This makes it even more infeasible to directly model a large-scale dataset like time-varying volumetric data. Numerous works have been proposed to address this issue. SIREN (Sinusoidal Representation Network) [25, 30] uses periodic activation functions instead of traditional activations like ReLU or tanh to improve the convergence time. Meta-learning [16, 47] pretrains a warm-start INR with optimal initial weights trained from a sparse but representative subset of the whole training samples. Input encoding methods, which encode the input of INR into a higher-dimensional space, show advantages in both efficacy and simplicity. Frequency encoding [4, 36] was first proposed as a positional encoding method utilizing Fourier features mapping to better capture the high-frequency details in constructing neural radiance fields (NeRF) [26]. Parametric encoding [9, 32] was later introduced, which arranges additional trainable parameters (beyond weights and biases) in an auxiliary grid or tree data structure. Recent Multi-resolution Hash Encoding (MHE) [27] gives state-of-the-art convergence speed by combining the embedding grid, multi-resolution, and a hash function to enable "instant" reconstruction of various types of neural graphic primitives including static scientific volumetric data [46].

However, MHE suffers from three main problems. 1) Since the hash collisions were not handled, different entries in the embedding grid were forced to use the same embedding vector, adversely affecting learning precision and limiting the fitting capacity of the INR. Detailed supporting quantitative results can be found in the Appendix. 2) The embedding grid of each resolution level has the same size, leading to unused buckets in lower resolutions, causing inefficient use of the encoding parameters. This also produces an unnecessarily large INR with redundant weights. Although the first problem can be mitigated by increasing the hash table size, it will further intensify the second bucket waste problem. 3) The architecture of MHE can only process 3D spatial data. These limitations prevent MHE from effectively encoding and representing large-scale time-varying data at high resolutions. In this work, we proposed F-Hash, a novel feature-based hash design for multi-resolution input encoding, which greatly enhances the convergence speed compared to existing parametric encoding methods for modeling time-varying volumetric data. Our design consists of three components: A feature-based coreset selection method for Meta-learning, adaptive Tesseract embedding grids for spatiotemporal interpolation, and collision-free bijective hash functions with 100% bucket utilization. F-Hash is dynamically adjusted according to the feature of interest for the best convergence speed and encoding parameter reduction. F-Hash is also compatible with various visualization tasks, from feature tracking to evolution visualization. Our work is the first attempt to apply multi-resolution hash encoding to time-varying volumetric data with state-of-the-art convergence speed. While the convergence speed remains insufficient for online training demands and the compression ratio is suboptimal compared to recently proposed volume compressors [14, 22], our method achieves a substantial reduction in INR training time for large-scale time-varying volumetric data. We also introduce a rendering framework capable of directly visualizing time-varying neural representations encoded via F-Hash and the Adaptive Ray Marching (ARM) algorithm to dynamically optimize sample streaming to the GPU, enabling interactive visualization. The main contributions of this work include:

- A novel feature-based multi-resolution hash encoding, F-Hash, with state-of-the-art convergence speed for training neural representation from time-varying volumetric data.
- A Minimal Perfect Hash Function (MPHF) to efficiently map multi-resolution Tesseract embedding grid to hash table without collision and bucket waste, enabling effective while compact input encoding.
- A feature-based coreset selection method to select only informative samples as training data from the time-varying volumetric data.
- An adaptive ray marching algorithm to dynamically stream samples for interactive visualization of the time-varying neural representation.

2 Related Work

2.1 Time-Varying Volume Visualization

For the past two decades, time-varying volume visualization has remained a vibrant research domain, driven by its critical role in revealing temporal dynamics and evolving patterns [1, 24]. Time-varying volume visualization can be systematically classified into three aspects: feature tracking, evolution visualization, and rendering. 1) Feature tracking aims to acquire the temporal evolution of features, which is essential for understanding various phenomena. Widanagamaachchi et al. introduce an interactive system [45] to track pressure perturbations and a merge tree-based method [44] to track extinction holes in turbulent combustion simulations. Kumpf et al. [19] present a framework that tracks and evaluates ensemble forecast sensitivities using clustering and optical flow. Lukasczyk et al. [23] present a topological framework to detect, track, and analyze viscous fingers in fluid mixing simulations. 2) Evolution visualization visually represents the complex results of feature tracking across numerous events and time steps, requiring clear and aesthetically appealing designs. Dutta et al. [8] propose a distribution-based method using incremental Gaussian Mixture Models (GMMs) to track vaguely defined evolving features. Saikia et al. [29] present a global graph-based method to track and compare features in time-dependent scalar fields. 3) Rendering large time-varying volumetric data requires efficient encoding techniques for acceleration and coherent transfer functions for clear temporal feature observation. Ljung et al. [21] review transfer function techniques for direct volume rendering from the perspectives of data characteristics and user interactions. Wang et al. [42] introduce a ray-based proxy method using histograms and depth information to reconstruct time-varying volume data. Zhou et al. [51] propose an information-theoretic method to automatically select representative time steps from large-scale time-varying volume datasets. For scientific time-varying volumetric data, the primary focus is to design efficient and effective feature-centric visualization, which motivates us to prioritize the encoding of the spatial and temporal patterns toward specific features rather than processing the entire dataset.

2.2 Implicit Neural Representation

Implicit Neural Representation (INR) is a technique that encodes data, such as images, 3D shapes, or continuous signals, using neural networks. For large-scale time-varying volume visualization, recent studies have introduced INR-based compression methods [7, 14, 22, 37, 39] to reduce model complexity and mitigate I/O bottlenecks. INR-powered super-resolution frameworks have been developed to enable efficient rendering of time-varying volumetric data at high fidelity. Han et al. proposed SSR-TVD [11], a spatial super-resolution for time-varying data analysis and visualization, STNet [15], a generative framework for synthesizing spatiotemporal super-resolution volumes, and CoordNet [13], a coordinate-based deep learning framework that improves generalization in time-varying volumetric data visualization. Tang et al. proposed STSR-INR [40], a super-resolution for multivariate time-varying volumetric data on both spatial and temporal domains. Yariv et al. [49] enhance neural volume rendering by implicitly representing geometry, improving both shape reconstruction and volume density modeling. For accelerating rendering performance, Weiss et al. propose fV-SRN [43], an advanced adaptation of Scene Representation Networks (SRN) [31] that dramatically speeds up volumetric reconstruction. However, the training time for INR remains significantly long when processing complex, large-scale, time-varying datasets. Therefore, developing an optimized input encoding approach becomes essential to reduce convergence time when training an INR for time-varying volumetric data.

2.3 Network Input Encoding

Network input or positional encoding is an emerging technique used to map raw input coordinates into a higher-dimensional space before feeding them into an INR. This helps the network better represent

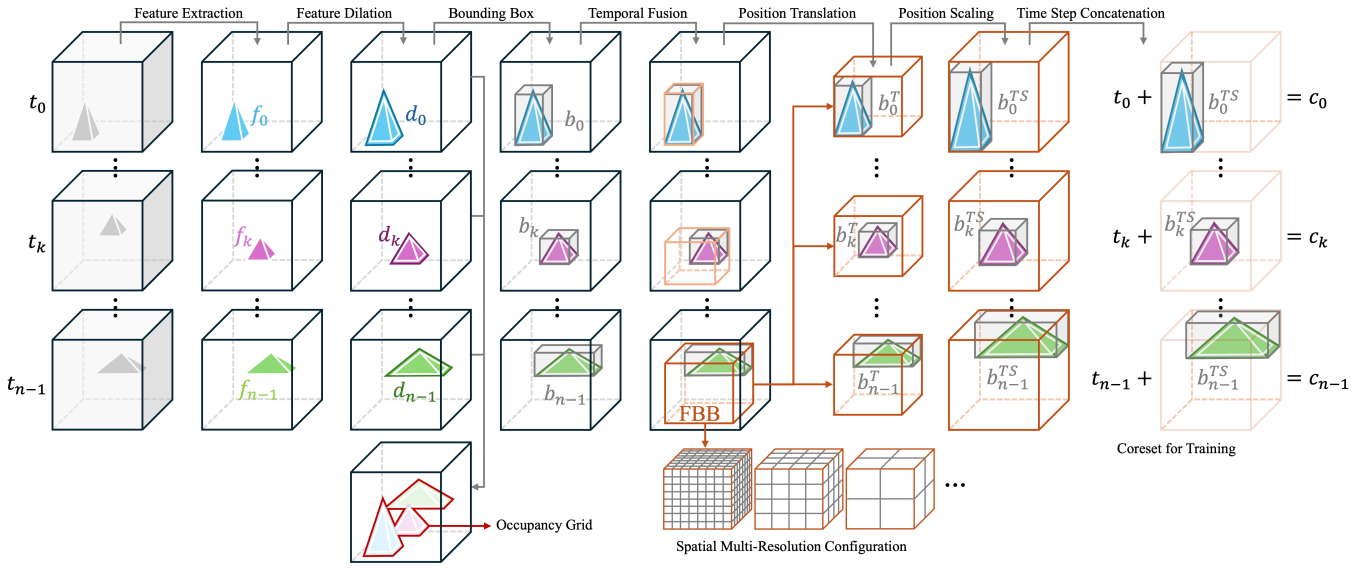


Fig. 2: Coreset selection process to retrieve the coreset for training, occupancy grid, and the FBB.

high-frequency details and converge faster. Input encoding can be classified into two main categories: frequency encoding and parametric encoding. 1) Frequency encoding was proposed as a positional encoding method utilizing Fourier features to map coordinates into high-frequency sine/cosine waves. Tancik et al. [36] demonstrate that applying Fourier feature mappings to input coordinates enables MLPs to learn high-frequency functions effectively in low-dimensional tasks. Barron et al. [4] replaces NeRF's ray sampling with cone tracing and integrated positional encoding to reduce aliasing and improve multiscale rendering. Mildenhall et al. [26] synthesizes photorealistic novel views by optimizing a 5D neural radiance field via differentiable volume rendering from input images. 2) Parametric encoding [9, 32] introduces supplementary trainable parameters organized in auxiliary grid or tree structures, extending beyond traditional network weights and biases. Sun et al. [32] optimize voxel grids directly for radiance fields, achieving NeRF-level quality with short training and sharp surface modeling. Fridovich-Keil et al. propose Plenoxels [9] that replaces neural networks with a sparse voxel grid for radiance fields, achieving NeRF-quality rendering 100× faster. The recent Multi-resolution Hash Encoding (MHE) [27] achieves state-of-the-art convergence speed by integrating an embedding grid architecture with multi-resolution and hash-based indexing, enabling near-instant reconstruction of diverse neural graphics primitives. Wu et al. [46] introduce MHE to model static scientific volumetric data and provide an interactive rendering pipeline to render visualization directly from the INR. While effective for static volumes, MHE's architectural limitations hinder direct application to time-varying volumetric data.

3 Methods

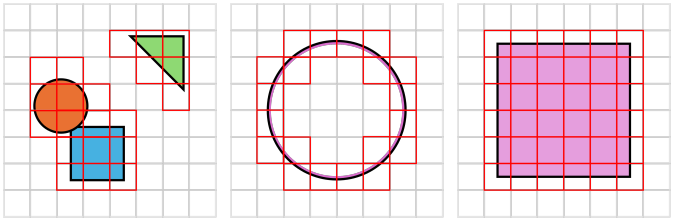
The design of F-Hash consists of three components: 1) A feature-based coreset selection method to select feature-related training samples for Meta-learning. 2) A group of adaptive Tesseract grids with various levels of resolution to hold the mapped higher-dimensional embedding vectors. 3) Collision-free bijective hash function for each resolution level with 100% bucket utilization. The Tesseract grids and hash functions are adaptive to the feature of interest for the best convergence speed up and encoding parameter reduction. The feature-centric design of our input encoding method makes it efficient and compatible with various time-varying features, enabling effective visualization from feature tracking to evolution visualization.

3.1 Feature-Based Coreset Selection

We adapt the idea of Meta-learning and provide a coreset selection method to collect only the feature-related subset of the entire time-varying volumetric data as the training data. The motivation is derived from two observations: 1) The analysis of time-varying volume data from various scientific domains primarily focuses on specific features of interest, such as dark matter halos in computational cosmology or flames in combustion science, as they hold key research significance. For the testing datasets we evaluated, only less than 20% of the entire volumetric region is related to the informative features of interest. Thus, this inspires us to selectively model only the relevant regions rather than the entire dataset for improved efficiency. 2) Due to the large number of training samples, universal INR modeling directly from the raw time-varying data results in an inevitably large batch size for a feasible training time. A large batch size results in a lower-variance gradient, making optimization smoother (less noisy updates) and helping in avoiding bad local minima since the loss landscape is averaged over more samples. However, modern GPUs are better optimized for smaller, parallel workloads than for a single computationally heavy one. Smaller batches fit entirely in the GPU cache, allowing frequent memory reuse and faster computing. Beyond a certain batch size, scheduling overhead increases, and some cores may remain underutilized due to memory bandwidth limits. This issue becomes more obvious when training an INR with a large number of training samples, like the time-varying volumetric data. Based on the above observations, to tackle this challenge and enhance convergence speed, we must minimize the size of the training data, thereby improving the convergence speed per epoch. The x, y, and z dimensions of each time step are normalized to range $[-1, 1]$. The key frame detection methods [17, 38, 50] are used to detect key frames that display critical evolutionary events in chronological order. Fig. 2 summarizes the key steps of processing the raw time-varying volumetric data with n key frames to select a representative coreset for training.

Feature Extraction: First, the time-varying features for each key frame, from time step t_0 to t_{n-1}, are extracted from the respective normalized input volume by specific feature extraction methods. We considered three common types of time-varying features [1]: Spatial-first features, Temporal-first features, and 4D features. For time step t_k, the extracted feature, f_k, is in the form of a bag of vertices where each vertex is the corresponding sample in the volume.

Feature Dilation: During the visualization stage, volume rendering algorithms require value and gradient queries of samples on the rays shooting from pixels of the visualization image. The sample on the off-grid location needs to be interpolated from the eight corners of the cell containing the sample. Therefore, it is necessary to include the neighboring vertices of f_k as a comprehensive feature region d_k to ensure an accurate interpolation on the boundary. The feature dilation



(a) Interval Features (b) Isosurface Feature (c) Segmentation Feature

Fig. 3: Occupancy grid (red empty squares) extraction for three types of Spatial-first time-varying features, interval, isosurface, and segmentation. The black boundary around each feature is the region of dilation.

performs an efficient search for neighboring vertices through a hash-based dictionary data structure. An occupancy grid, which will be used during rendering for acceleration, also needs to be extracted in this step. The detailed usage of the occupancy grid is discussed in Sec. 3.3. Fig. 3 demonstrates how the occupancy grid is derived for various time-varying features. The occupancy grid has the same spatial dimension as the individual volume. The value of its voxel is set to 1 if d_k intersects with the voxel or 0 otherwise. The occupancy grid is stored as bits to minimize its memory footprint. Morton linearization is used to map the voxel to a 1D vector of bits for spatial locality. When rendering multiple features encoded by F-Hash, multiple occupancy grids can be merged to optimize the overall rendering performance.

Bounding Box and Temporal Fusion: The range of d_k in x, y, and z dimensions forms a bounding box b_k. Temporal fusion finds the minimal superset (coreset) of all the bounding boxes across time. We name this bounding box as Feature Bounding Box (FBB). FBB is critical in two aspects: 1) FBB contains the coreset of training samples for F-Hash. 2) The spatial multi-resolution configuration is derived from FBB. Detailed configuration policy is discussed in Sec. 3.2.

Position Translation and Scaling: Position translation shifts the sample coordinate in b_k from the original coordinate system to the coordinate system of FBB. The new x, y, and z coordinates of each vertex in b_k are calculated as:

$$b_k^T.x/y/z = b_k.x/y/z - Origin^{FBB}.x/y/z \tag{1}$$

where $Origin^{FBB}$ is the centroid coordinate of FBB in the original coordinate system. Then we do position scaling to scale the coordinates of samples within FBB to the range of $[-1,1]$:

$$b_k^{TS}.x/y/z = b_k^T.x/y/z \times (\frac{2}{Size^{FBB}.x/y/z}) - 1 \tag{2}$$

where $Size^{FBB}.x/y/z$ is the size of FBB on the x, y, or z dimension.

Time Step Concatenation: We concatenate the time step to corresponding samples in b_k^{TS} to construct 4D training samples c_k, which is composed of input $([t,x,y,z])$ and label (v) pair. The final coreset for training is constructed by:

$$Coreset = \bigcup_{k=0}^{n-1} c_k,\ \ c_k = [t_k,\ b_k^{TS}] \tag{3}$$

It is worth noticing that not all the samples in the FBB are in the coreset for training. A more aggressive method of defining the coreset is to only consider the regions where features are present. This results in irregularly shaped multi-resolution embedding grids instead of rectangular prism grids. Although this method generates an even smaller coreset and reduced model size through fewer encoding parameters, it introduces a significant computational bottleneck in looking up the hash table due to the arbitrary shape of the embedding grid and its large number of entries. We experiment with implementations using both a hash-based dictionary data structure and a parallel library of PyTorch tensor Argmax, and the results show a significant degradation in training time due to the hash lookup overhead.

3.2 Multi-Resolution Tesseract Encoding Design

In this section, we detail the design of the proposed multi-resolution Tesseract encoding, which has a grid-based parametric encoding structure. While most input encoding research papers use "feature" to name the high-dimensional vectors within the encoding grid, we instead use the term "embedding" to prevent confusion with the time-varying features.

3.2.1 Spatial Multi-Resolution Configuration:

We derive the spatial-only multi-resolution configuration for x, y, and z dimensions from the FBB because it provides a compact spatial distribution and a minimal upper bound of the resolution needed. We use the native resolution of FBB (S_x, S_y, S_z) as the level with the highest resolution (level 1) for all dimensions.

$$Res_x^1 = S_x,\ \ Res_y^1 = S_y,\ \ Res_z^1 = S_z \tag{4}$$

The native number of levels for each dimension is determined by:

$$L_x = \lceil log_f S_x \rceil,\ \ L_y = \lceil log_f S_y \rceil,\ \ L_z = \lceil log_f S_z \rceil \tag{5}$$

where f is the fold parameter with an integer value starting from 2. The shared number of resolutions L_s is the maximum of the three:

$$L_s = max(L_x,\ L_y,\ L_z) \tag{6}$$

The resolution with a level l greater than 1 is iteratively determined by the resolution of the previous level:

$$Res^l = \begin{cases} \lceil \frac{Res^{l-1}}{f} \rceil & \text{if } Res^{l-1} > f \\ Tail(l) & \text{otherwise} \end{cases},\quad l \in \{2,3,\ldots,L_s-1\} \tag{7}$$

$$Tail(l) = \begin{cases} 2 & \text{if } l \le L_s \\ stop & \text{otherwise} \end{cases} \tag{8}$$

A higher f gives a larger resolution difference between neighboring levels, resulting in a smaller number of resolution levels. Since the size of FBB on x, y, or z dimensions can be different, we pad the smaller dimension with the minimal resolution, which is 2, at the tail to save encoding parameters while still aligning the number of resolution levels as l_s. Compared with existing MHE-based methods using evenly partitioned resolutions across levels, our method has several advantages:

- Our method divides the resolution of the current level by a fold parameter to calculate the resolution of the next level. Our policy of configuring resolutions requires fewer encoding parameters when having the same number of resolution levels as MHE.
- For a specific resolution level, resolution remains identical across the dimensions in grid-based input encoding methods like MHE. F-Hash determines the resolution of each dimension independently, resulting in a smaller number of encoding parameters when handling rectangular cuboid volumetric frames with unequal edge lengths. Fig. 4 shows an example comparing the number of encoding parameters when handling 2D rectangle grids with a high aspect ratio.
- For various resolution levels, the MHE hash tables for different resolution levels share the same size, even though the embedding grid becomes smaller for lower resolution levels, resulting in bucket waste in those levels. The hash tables of F-Hash are resolution-dependent, scaling their sizes proportionally with the embedding grid size of the current resolution level.
- In MHE, the number of resolution levels and their respective resolutions are predefined hyperparameters that require manual tuning. In contrast, our approach automatically optimizes those parameters from the spatial information of the time-varying feature of interest.

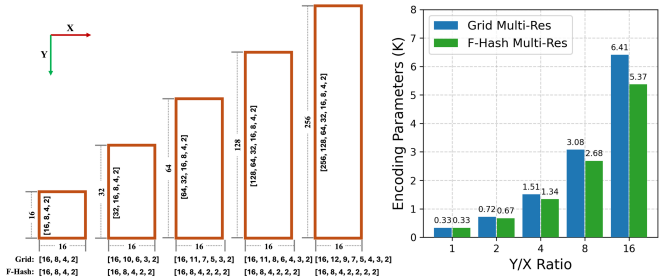


Fig. 4: Multi-resolution configuration of Grid-based methods and F-Hash on grids with high aspect ratio.

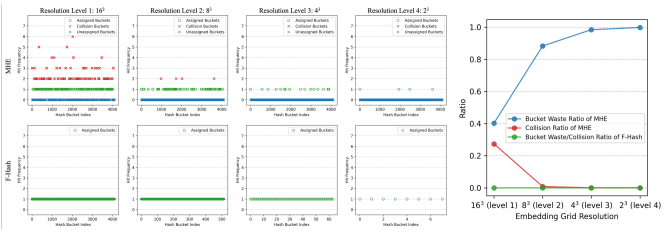


Fig. 5: Comparison between MHE and F-Hash on collision and bucket waste performance in an example with 4 resolution levels.

3.2.2 Temporal Multi-resolution Configuration

In order to expand the input encoding to higher dimensional data than 3D spatial volume, we propose multi-resolution Tesseract encoding with temporal dimension to directly model time-varying volumetric data. Due to the dynamic changes over temporal dimension and arbitrary length of time sequences, we need to treat temporal dimension independently to 3D spatial dimensions. In order to minimize the total number of encoding parameters, as mentioned in Sec. 3.1, we utilize key frame extraction methods to select the time steps of the informative frames where critical evolutionary events happen. We then apply a similar spatial resolution policy to configure the resolution on the temporal domain using fold. In real-world scenarios, compared to the size of spatial dimensions, the range of the temporal dimension of the key frames is much smaller, and the padding of 2 is normally needed at the tail of temporal resolution levels.

3.2.3 Hash Function Design

The proposed feature-based hash function (F-Hash) is to map each corner position of Tesseract embedding grid into a bucket in the hash table. The main goals of the design include no hash collision, no bucket waste, and fast looking up. Compared with existing multi-resolution hash encoding methods, the proposed F-Hash meets all the requirements, making it a more capable input encoding solution. Our F-Hash utilizes simple linearization to map 4D coordinates to the hash table index. This enables a single look-up to retrieve the mapping for all 16 corners of the Tesseract through simple shifting. The bucket index of an embedding grid corner (t,x,y,z) of resolution level l can be calculated by:

$$F\text{-}Hash^l(t,x,y,z) = t \times Res_x^l \times Res_y^l \times Res_z^l + z \times Res_x^l \times Res_y^l + y \times Res_x^l + x \quad (9)$$

MHE uses a simple spatial hash function [41], which presents inevitable collision on lower resolution levels and bucket waste on higher resolution levels as shown Fig. 5. F-Hash is a bijective minimal perfect hash function (MPHF) without collision and bucket waste, enabling more accurate modeling with more compact encoding parameters. Fig. 6 demonstrate the F-Hash mapping from the neighboring embedding cells to the hash table entries for each resolution level. Other linearizations like Morton and Z-order can also be used to construct F-Hash with preserved locality.

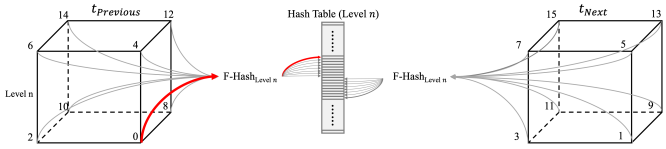


Fig. 6: Hash function design for mapping the multi-resolution Tesseract embedding grid to hash buckets of resolution level n.

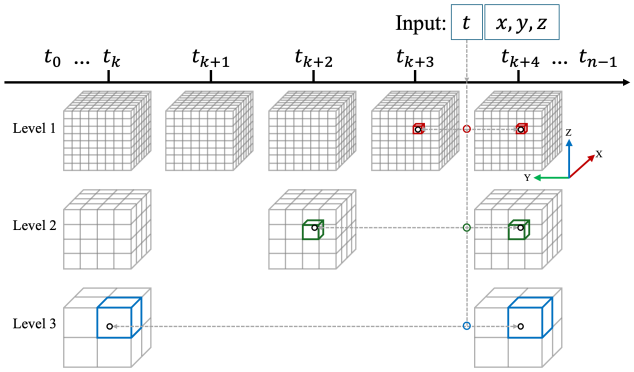


Fig. 7: Multi-Resolution Tesseract Encoding. For the given input (t,x,y,z), the neighboring time steps from level 1 to level 3 are, $[t_{k+3},t_{k+4}]$, $[t_{k+2},t_{k+4}]$, and $[t_k,t_{k+4}]$. Embedding cells are color-coded for each resolution level.

3.2.4 Input Encoding

The encoding steps for the proposed multi-resolution Tesseract encoding consist of the following steps:

1. Locate neighboring time steps: The input, (t,x,y,z), needs to find its neighboring time steps across the temporal dimension for each resolution level according to t. We name the time step before and after t as $t_{Previous}$ and t_{Next}. Fig. 7 shows the embedding grids of the proposed Tesseract encoding with three spatial resolution levels.

2. Locate embedding cells: Once the neighboring time steps are found, the embedding cells need to be located from the embedding grid at $t_{Previous}$ and t_{Next} for each resolution level. The embedding grid of each resolution level is a cube covering a range of $[-1,1]$ for each dimension. The embedding cell is the smallest unit for a specific resolution level containing the input location (x,y,z).

3. Look up embedding vectors: Each of the eight corners of the embedding cell is mapped to an embedding vector through the proposed F-hash described in Sec. 3.2.3. There are 16 embedding vector in total for both the embedding cells at $t_{Previous}$ and t_{Next}.

4. Interpolate embedding vectors: A quadrilinear interpolation across both spatial and temporal is performed for an aggregated embedding vector V for each resolution level. The spatial trilinear interpolation is performed first within the embedding cells, followed by the linear interpolation across the temporal dimension:

$$V^{Previous}(x,y,z) = \sum_{i=0}^{1}\sum_{j=0}^{1}\sum_{k=0}^{1} V_{ijk}^{Previous} \cdot x^i(1-x)^{1-i} \cdot y^j(1-y)^{1-j} \cdot z^k(1-z)^{1-k} \quad (10)$$

$$V^{Next}(x,y,z) = \sum_{i=0}^{1}\sum_{j=0}^{1}\sum_{k=0}^{1} V_{ijk}^{Next} \cdot x^i(1-x)^{1-i} \cdot y^j(1-y)^{1-j} \cdot z^k(1-z)^{1-k} \quad (11)$$

$$V(t) = \sum_{l=t_{Previous}}^{t_{Next}} V^l \cdot t^l(1-t)^{t_{Next}-l}. \quad (12)$$

All aggregated embedding vectors V from all resolution levels are concatenated to form the high-dimensional input for the downstream MLP. Fig. 8 demonstrates the interpolating process. To accelerate rendering in volumetric neural representations, a shallow network (with only one or minimal hidden layers) is used instead of a deep architecture. This reduces training and inferencing time significantly because the embedding grid encoding offloads the geometric complexity from the network, allowing a lightweight model to represent complex data. All

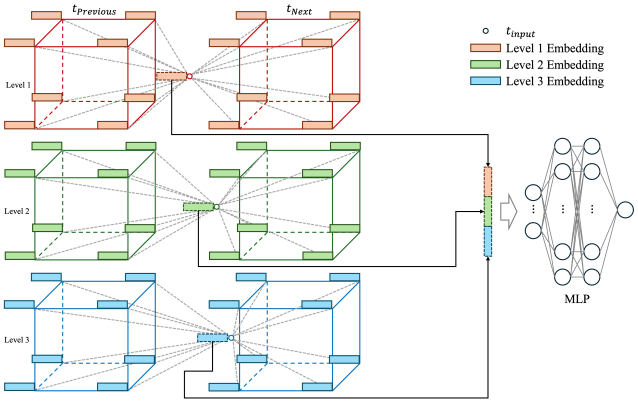


Fig. 8: Quadrilinear interpolation.

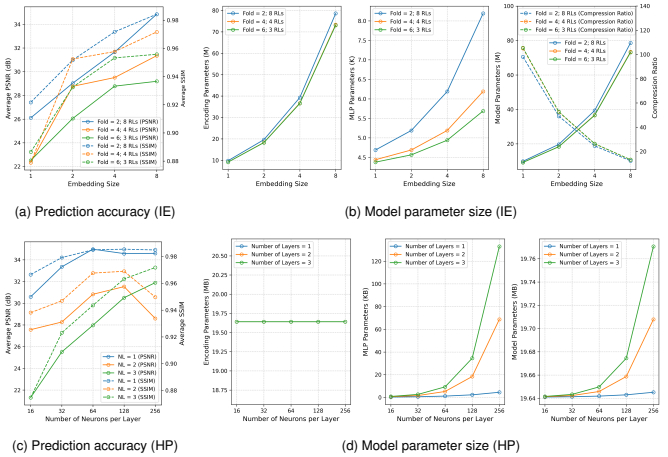


Fig. 9: Prediction accuracy (the 10th iteration, results of more iterations can be found in Appendix) and model parameter size evaluation. (a) and (b) are results of input encoding (IE) configuration with different folds/resolution levels (RL) and embedding size. (c) and (d) are results of Hyperparameter (HP) configuration with different numbers of layers and number of neurons per layer.

Algorithm 1 Time-varying volumetric neural representation rendering pipeline using sample streaming with Adaptive Ray Marching (ARM)

Input: Time Step (t), View, Transfer Functions(TF), and Trained INR
Output: Visualization image

$N_f = 0$ ▷ Finished samples
$R = getRays(view)$
if $N_f <= R_{max}$ **then**
  $N_a = getAliveRayNum()$
  if $N_a == 0$ **then** ▷ When all rays finished
    Break
  else
    $N_s = max(min(N_r/N_a, 64), 1)$ ▷ Adjust Marching Pace
    $N_f += N_s$
    $S = getSamples(N_s, OccupancyGrid)$ ▷ Sample streaming
    $S = t \| S$ ▷ Concatenate time step
    $V = model(S)$ ▷ Inference F-Hash INR
    $RGBA = applyTF(V, TF)$
    $compositing(RGBA)$
    $updateAlive()$
  end if
end if

the embedding vectors of Tesseract grids are initialized by uniform initialization before training.

3.3 Rendering

We implement a time-varying volume visualization algorithm to directly render the INR modeled through F-Hash. We design the rendering pipeline using a similar idea of sample streaming method [46] for visualizing volumetric neural representation. We further improve the algorithm by adding an adaptive ray marching (ARM) algorithm to optimize thread usage on GPU for reduced rendering latency. The idea of ARM is to efficiently utilize the GPU threads by increasing the ray marching pace over time as more rays get finished. Algorithm 1 details the rendering algorithm. Our renderer supports popular visualizations for time-varying volumetric datasets, including feature tracking and evolution visualization. The occupancy grid derived from the coreset selection step is used to accelerate the rendering by skipping the empty space. Super-resolution in the temporal domain is also supported. For a given unseen time step, a linearly interpolated occupancy grid for the specific time step is first calculated. Then, the trained INR can be directly inferenced for rendering. Since the INR is trained on the key frames, which cover all the critical evolutionary changes, the interpolated occupancy grid remains consistent with adjacent key frames. Although training INR with input encoding is fast, the convergence time of online training on time-varying volumetric data is still relatively long for interactive visualization. Therefore, our rendering pipeline needs a pre-trained INR before visualization.

Table 1: Training time under different input encoding configurations.

Training Time (Minutes) ↓	Embedding = 1	Embedding = 2	Embedding = 4	Embedding = 8
Fold = 2 (Res lvl = 8)	8.4	16.8	33.7	67.3
Fold = 4 (Res lvl = 4)	7.9	15.7	31.4	62.9
Fold = 6 (Res lvl = 3)	7.8	15.6	31.3	62.6

4 Ablation Study

4.1 Input Encoding Configuration

The proposed embedding grids learn both the geometric and temporal dynamics of the time-varying data. We evaluate the following two key configurations of the input encoding: 1) **Fold Parameter:** The fold parameter determines the multi-resolution configuration. The default fold is set to 2, providing the most number of levels and the highest possible resolution on each level. Larger fold values give a smaller number of levels with lower resolution for subsequent levels, except for level 1. 2) **Embedding Size:** The embedding size determines the number of embedding weights assigned for each corner of the Tesseract grid. A larger embedding size results in a higher dimension of the input encoding for the downstream MLP. We fix the MLP parameters (number of MLP layers = 2, number of neurons per layer = 64) while exploring the input encoding parameter space for modeling the Argon Bubble dataset. As Fig. 9a shows, using a smaller fold or a larger embedding size gives better training accuracy per training iteration and improves convergence speed. Fig. 9b shows that changing the embedding size will change the size of both the encoding parameter and the MLP parameters more dramatically than changing the fold parameter, resulting in a larger INR. A similar correlation can be observed in Tab. 1 on training time, where increasing the embedding size significantly prolongs the training duration. This is because the embedding size will linearly increase the total number of trainable parameters of the INR, requiring more gradient computation and updates during backpropagation and therefore increasing the per-iteration training time. The last subfigure of Fig. 9b also shows the respective compression ratio using various folds and embedding sizes when encoding the Argon Bubble dataset with 9 key frames. Based on our observations, we set both the fold parameter and embedding size to 2 to achieve high accuracy while maintaining a moderate model size and reasonable training time.

4.2 Hyperparameter Tuning

Together with the embedding grids, the downstream neural network, as shown in Fig. 8, will predict the value at the query spatial-temporal location from the high-dimensional input encoding. We evaluate the

Table 2: Time-varying datasets information of variable, resolution, and size. Intensity, Angular, and MixFrac variables are selected as the volumetric scalar fields of each dataset for the experiment.

Dataset	Variable	Resolution $(X \times Y \times Z) \times T$	Data Type	Size
Combustion	CHI/HO2/OH/**MixFrac**	$(200 \times 172 \times 54) \times 136$	float32	0.94 GB
Argon Bubble	**Intensity**	$(128^2 \times 256) \times 241$	float32	3.77 GB
Supernova	**Angular**/Entropy	$(216^3) \times 136$	float32	5.11 GB

following two key hyperparameters of the network: 1) **Number of MLP Layers:** The number of MLP layers determines how deep the neural network is. A deeper network is better for hierarchical feature learning, but harder to train. 2) **Number of Neurons per Layer:** The number of neurons per layer determines the learning capability of the network. A wider network (more neurons per layer) can approximate simple functions efficiently. We fix the input encoding parameters (fold = 2, embedding size = 2) while exploring the MLP parameter space for modeling the Argon Bubble dataset. As shown in Fig. 9c, adding more layers to the MLP slightly improves training accuracy per iteration. While increasing the number of neurons in the lower range improves training accuracy, it diminishes in the higher range. As shown in Fig. 9d, although adding more neurons causes the MLP to grow exponentially, this doesn't substantially affect the total INR size since the input encoding accounts for the most parameters. Adjusting the MLP's depth and width has minimal impact on training time, with all configurations completed in approximately 39 minutes. The proposed Tesseract embedding grid effectively captures both spatial and temporal characteristics through a high-dimensional encoding, enabling a shallow MLP to accurately model the time-varying volumetric data. We select the number of MLP layers as 2 and the number of neurons per layer as 64 for our experiments.

5 Experiments and Evaluation

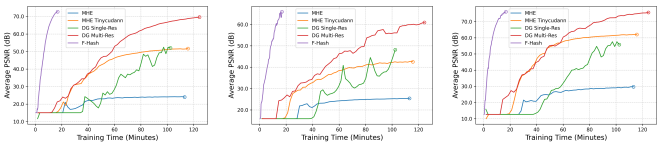
5.1 Experimental Setup

5.1.1 Datasets

For quality and performance evaluation, we select 3 large-scale time-varying volumetric datasets with distinct spatial and temporal features collected from diverse domains. Detailed information is listed in Tab. 2. The variables we used in our experiment are MixFrac from the Combustion dataset, Intensity from the Argon Bubble dataset, and Angular from the Supernova dataset. The spatial ranges of x, y, and z dimensions of each time step volume of all datasets are normalized within $[-1, 1]$. The values of each dataset are normalized across the 4D domain within the range $[0, 1]$. There are 10, 9, and 10 key frames extracted for the respective Combustion, Argon Bubble, and Supernova datasets.

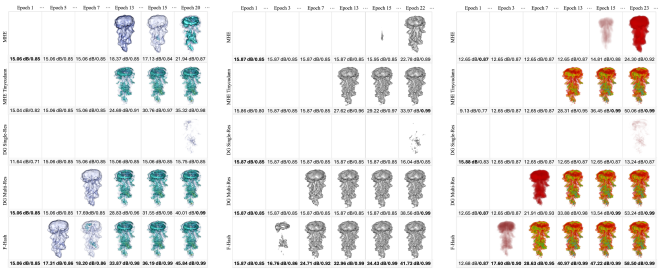
5.1.2 Evaluation

The experiments are designed to investigate input encoding methods from two aspects: convergence performance during training and model performance during rendering. PSNR and SSIM are used to evaluate the reconstruction accuracy. For convergence evaluation, we use convergence time and training time. We select representative time-varying visualization tasks from the literature reviews [2, 18] for a comprehensive evaluation of the practicality and viability of our method. We select the mainstream time-varying volume visualization techniques including Spatial-first Feature Tracking (Interval, Isosurface, and Segmentation features) and Evolution Visualization (Animation). The interval feature is extracted by selecting samples with values within a range of interest. The isosurface features represent the samples of constant value. The segmentation features represent foreground elements extracted from a 3D background space. Evolution visualization is to provide a meaningful overview through the visual dynamics over time. We compare our method with recent parametric-based input encoding works, which are Dense Grid Single-Resolution Encoding (DG Single-Res) [6], Dense Grid Multi-Resolution Encoding (DG Multi-Res) [10], a Pytorch implementation of Multi-resolution Hash Encoding (MHE) [27], and Multi-resolution Hash Encoding accelerated by fully fused MLP [28] through Tinycudann framework (MHE Tinycudann). We exclude the frequency-based encoding methods for their suboptimal performance.



(a) Interval (b) Isosurface (c) Segmentation

Fig. 10: Convergence speed for training the INR using different input encoding methods for various features of the Argon Bubble dataset.



(a) Interval (b) Isosurface (c) Segmentation

Fig. 11: Convergence visualization for the Argon Bubble dataset.

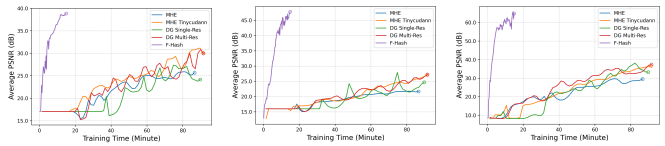
5.1.3 Training

The INRs with input encoding are trained using the PyTorch software stack to accelerate the training and inferencing performance on a single NVIDIA RTX A6000 GPU. The Adaptive Moment Estimation (Adam) optimizer is used with an initial learning rate of 0.01 for all the experiments. The training stops at the 60th epoch/iteration. Although using a relatively small batch size makes the training faster, the loss curve will dramatically oscillate during the training due to the frequent updates of local gradient descent, making the convergence comparison through accuracy very random. However, a very large batch size slows down the training dramatically. We use the batch size of 2^{21} for all the experiments to balance the training time and training accuracy.

5.2 Input Encoding Evaluation

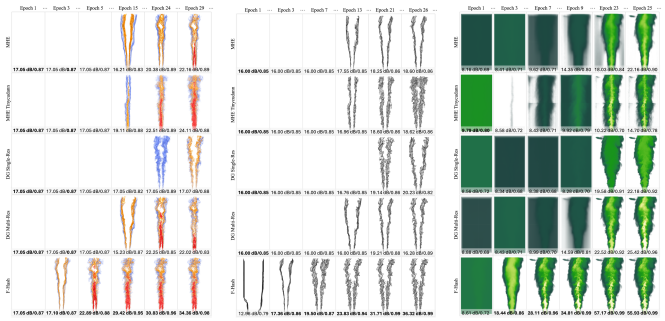
5.2.1 Convergence Time

We train the INR using different input encoding methods and evaluate convergence speed for visualizing three types of features: interval, isosurface, and segmentation. The embedded Tesseract grid design enables F-Hash to represent time-varying volumetric data as a unified neural representation, unlike other input encoding methods that require separate modeling of each key frame. We track the training time until the 60th iteration for each method and evaluate the convergence progress based on the average reconstruction accuracy of the visualization image generated from all key frames for each feature. Fig. 10a shows how fast each method converges in terms of PSNR and SSIM as the training goes on for the interval feature of the Argon Bubble dataset. F-Hash is significantly faster (reaching high PSNR/SSIM quickly) than other methods that finish the training roughly around the same time. Similar results can be observed in the isosurface (Fig. 10b) and segmentation (Fig. 10c) features. For the smaller Combustion dataset, all the methods can finish the training faster, as shown in Fig. 12. Due to the more complicated spatiotemporal structure of the dataset, other methods struggle to converge under 60 epochs, while F-Hash demonstrates superior efficiency in learning dynamic features. While the Supernova dataset, being both the largest and most dynamic, presents challenges for all methods in achieving high accuracy compared to the other two datasets, F-Hash still outperforms the other input encoding methods as shown in Fig. 14. The reasons for the observation are: 1) F-Hash's coreset selection reduces the total number of training samples, resulting in fewer batches for each training iteration. 2) The proposed multi-resolution Tesseract embedding grid is capable of learning complex spatiotemporal features with a smaller number of trainable encoding parameters, resulting in



(a) Interval (b) Isosurface (c) Segmentation

Fig. 12: Convergence speed for training the INR using different input encoding methods for various features of the Combustion dataset.



(a) Interval (b) Isosurface (c) Segmentation

Fig. 13: Convergence visualization for the Combustion dataset.

faster backpropagation during training. 3) Our design of Tesseract embedding grid jointly considers both spatial and temporal information, providing a more informative high-dimensional representation for the MLP to learn. Fig. 11, Fig. 13, and Fig. 15 provide visualizations of convergence through the generated rendering image for each feature of the three datasets, which provides an informative insight about how each input encoding method prioritizes which region of data to converge first. The two MHE methods converge from the regions with a lower value range, and the MHE with Tinycudann is more efficient in capturing high-frequency details. The DG Multi-Res tends to converge on a more global region, while the DG Single-Res starts by converging regions with more random locations. F-Hash can quickly converge for the entire feature from global to detailed regions. The qualitative evaluation of the convergence time for a given reconstruction accuracy is detailed in the left-hand side of the Tab. 3. For all selected levels of PSNR accuracy (20, 30, and 40 dB), F-Hash is the fastest to converge.

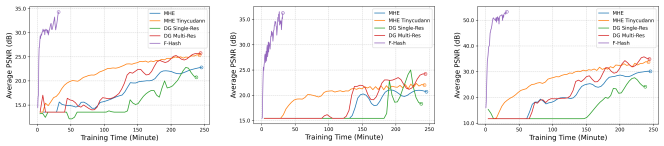
5.2.2 Number of Parameter

The configuration of each input encoding method while modeling different datasets is listed on the right-hand side of Tab. 3. The fold parameter of F-Hash is set to 2 for all the experiments for the highest modeling capacity. MHE-based methods adjust the hash table size to match the volume size of the frame. The number of resolution levels also increases when handling larger time-varying volumetric data. Since the FBB is adaptive to the specific feature, enabling efficient encoding through dynamic adjustment of the number of input encoding parameters when handling interval, isosurface, or segmentation features. As a result, F-Hash has the least number of input encoding parameters and achieves the shortest convergence time for all testing datasets.

5.3 Performance Evaluation

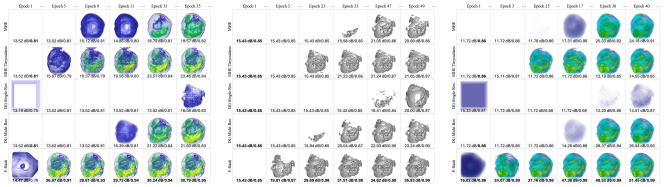
5.3.1 Reconstruction Accuracy

In this evaluation, we let all input encoding methods finish for the same number of iterations (30th) and compare their performances on reconstruction accuracy and training time. We use the popular evolution visualization as the time-varying visualization task on the segmentation feature. Fig. 16 demonstrates a visual comparison of reconstruction accuracy of frames over time for the three testing datasets. More frames are shown in Fig. 1 with convergence speed up for 30 dB PSNR and encoding parameters reduction. F-Hash achieves the highest reconstruction accuracy compared to other input encoding methods while maintaining a short training time. Quantitative measurements of average PSNR/SSIM across frames and training time are listed in



(a) Interval (b) Isosurface (c) Segmentation

Fig. 14: Convergence speed for training the INR using different input encoding methods for various features of the Supernova dataset.



(a) Interval (b) Isosurface (c) Segmentation

Fig. 15: Convergence visualization for the Supernova dataset.

Tab. 4. We can observe that, together with the model size, the size of the F-Hash feature region (coreset) plays a key role in determining the training time. A larger feature region tends to have a longer training time due to the large number of training samples in the coreset.

5.3.2 Compression

Since only a subset (key frames) of the total frames are considered to construct the INR. The trained INR also performs a compression to the original large-scale time-varying volumetric data. The compression ratio of F-Hash is the highest among all the input encoding methods as listed in the left-hand side of Tab. 5. However, compared with specialized volume compressors like the state-of-the-art learning-based compression methods, NeurComp [22][1], and lossy compression methods, SZ3 [20][2] and TTHRESH [3][3], F-Hash falls short of these approaches in compressing large-scale volumetric data.

5.3.3 Rendering

We also evaluate the rendering latency using our proposed ARM methods. We measure the average rendering time of generating a visualization image from a random view on a specific frame during an evolution visualization task. Since only the F-Hash encoded INR supports super-resolution, for a fair comparison, only the key frame is rendered. Tab. 6 shows the rendering latency across all input encoding methods. We can see that inferencing F-Hash encoded INR is faster with the help of an occupancy grid and a smaller model size. Using ARM can further reduce the rendering latency of the sample streaming algorithm, giving a better performance for interactive visualization.

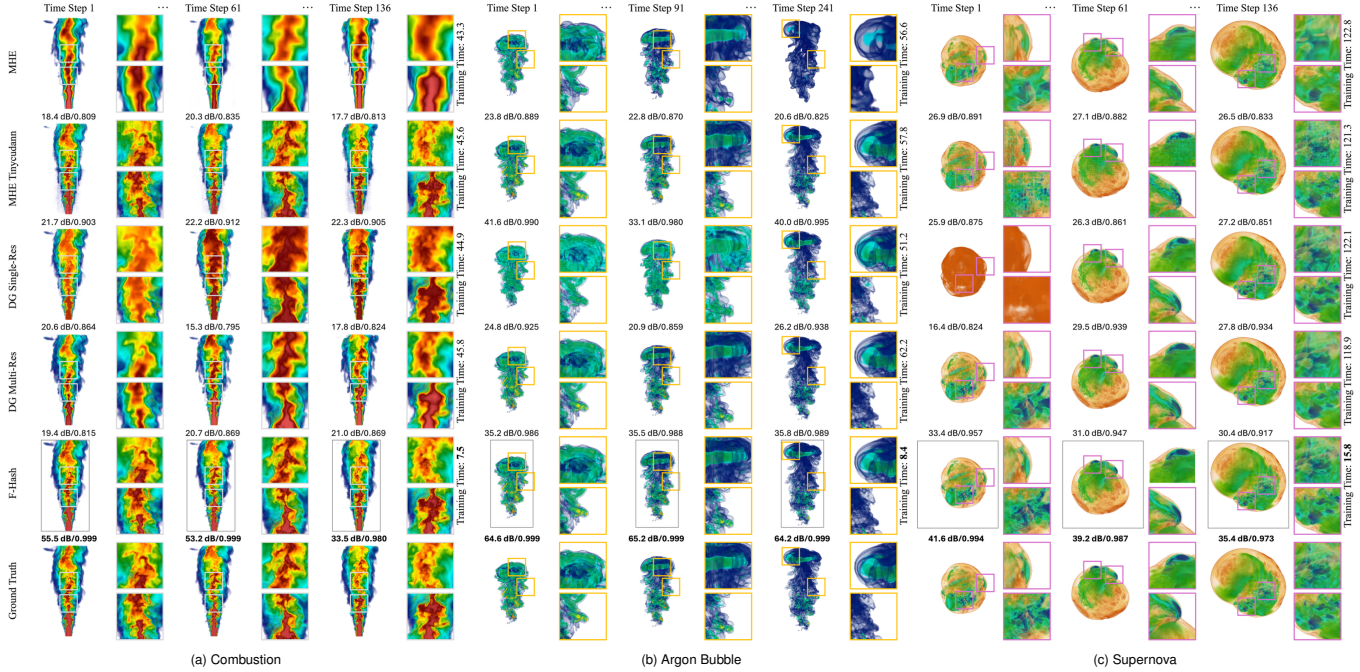
6 Limitations and Future Work

In this work, we propose an efficient encoding architecture that greatly enhances the convergence speed of training implicit neural networks for time-varying volumetric data. Our experiments demonstrate that F-Hash achieves the fastest convergence speed with more efficient use of encoding parameters compared with existing input encoding methods. The proposed input encoding design is easy to adapt for applications utilizing INR. Recent INR-based research in compression, visualization, and super-resolution can benefit from our multi-resolution Tesseract encoding to improve the convergence speed of modeling time-varying volumetric data. Our method can also be extended to other spatiotemporal data types, including video data, weather/climate data, and geospatial time series. The limitations of the work include: 1) While F-Hash achieves state-of-the-art convergence speed, its training time remains relatively slow to realize online training due to the large size of the time-varying data. 2) The current compression performance of F-Hash is not as good as existing learning-based compressors or other

[1]https://github.com/matthewberger/neurcomp
[2]https://github.com/szcompressor/SZ3
[3]https://github.com/rballester/tthresh

Table 3: Left is the convergence time measurement. The budget of the maximal training iteration allowed is set to 60. If a method can't reach a specific PSNR within the training budget, the time is marked as Not Reached (NR). On the right, the first two rows are the encoding configuration, while the last row is the number of parameters in mebi-unit (M).

Dataset	PSNR	Convergence Time (min) ↓ MHE Int	MHE Iso	MHE Seg	MHE Tinycudann Int	MHE Tinycudann Iso	MHE Tinycudann Seg	DG Single-Res Int	DG Single-Res Iso	DG Single-Res Seg	DG Multi-Res Int	DG Multi-Res Iso	DG Multi-Res Seg	F-Hash Int	F-Hash Iso	F-Hash Seg	Encoding Configuration / Parameter Size (M) ↓ MHE	MHE Tinycudann	DG Single-Res	DG Multi-Res	F-Hash Int	F-Hash Iso	F-Hash Seg
Combustion	20 dB	33.2	51.9	31.7	24.3	54.7	45.6	37.4	46.3	35.9	29.0	39.7	25.9	**1.0**	**2.2**	**1.2**	Hash table size = 2^{20}	Hash table size = 2^{20}			Fold = 2	Fold = 2	Fold = 2
	30 dB	NR	NR	NR	83.6	NR	86.6	NR	NR	67.2	83.9	NR	58.0	**4.2**	**5.0**	**2.0**	Res level = 6	Res level = 6	Res level = 6	Res level = 6	Res level = 6	Res level = 6	Res level = 6
	40 dB	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	**7.4**	**2.9**	120.0 M	45.8 M	45.0 M	51.5 M	**16.1** M	**17.4** M	**24.2** M
Argon Bubble	20 dB	22.6	30.2	30.2	21.2	23.2	23.1	37.5	44.4	46.1	16.6	14.5	14.5	**2.2**	**1.1**	**1.1**	Hash table size = 2^{21}	Hash table size = 2^{21}			Fold = 2	Fold = 2	Fold = 2
	30 dB	NR	NR	NR	28.9	32.8	26.9	61.4	59.7	59.7	29.0	29.0	27.0	**3.4**	**3.1**	**2.2**	Res level = 7	Res level = 7	Res level = 7	Res level = 7	Res level = 6	Res level = 7	Res level = 7
	40 dB	NR	NR	NR	48.2	73.2	36.6	80.2	63.1	68.3	43.6	49.8	35.3	**4.8**	**5.9**	**3.4**	252.0 M	77.2 M	72.0 M	82.3 M	**16.7 M**	**19.5 M**	**19.7 M**
Supernova	20 dB	163.7	147.3	135.1	52.6	68.7	40.4	190.4	190.3	186.4	134.3	134.3	126.2	**2.1**	**2.1**	**1.6**	Hash table size = 2^{23}	Hash table size = 2^{23}			Fold = 2	Fold = 2	Fold = 2
	30 dB	NR	NR	237.4	NR	NR	137.5	NR	NR	NR	NR	NR	179.1	**7.4**	**10.0**	**2.6**	Res level = 7	Res level = 7	Res level = 7	Res level = 7	Res level = 8	Res level = 8	Res level = 8
	40 dB	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	**5.8**	1120.0 M	205.8 M	135.0 M	154.3 M	**108.2 M**	**86.9 M**	**107.7 M**



(a) Combustion (b) Argon Bubble (c) Supernova

Fig. 16: Evolution visualization using different input encoding methods. Reconstruction accuracy is labeled in the format of PSNR/SSIM for each rendering. For the F-Hash, 3D feature region (FBB) for each datasets are labeled as the gray bounding boxes with dimension of $80 \times 83 \times 204$, $200 \times 110 \times 54$, and $156 \times 183 \times 185$, which takes 63.95%, 32.30%, 52.41% of the Combustion, Argon Bubble, and Supernova frame volume.

Table 4: Measurements of average PSNR/SSIM across frames and training time.

Dataset	Avg PSNR (dB) ↑ / Avg SSIM ↑/ Training Time (min) ↓ MHE	MHE Tinycudann	DG Single-Res	DG Multi-Res	F-Hash
Combustion	18.9 / 0.818 / 43.4	23.7 / 0.906 / 45.7	20.2 / 0.829 / 44.8	20.5 / 0.850 / 45.9	**47.1 / 0.994 / 7.4**
Argon Bubble	22.3 / 0.862 / 56.5	28.3 / 0.987 / 57.9	23.9 / 0.908 / 51.1	35.6 / 0.987 / 62.3	**64.6 / 0.999 / 8.5**
Supernova	26.7 / 0.870 / 122.7	26.6 / 0.861 / 121.4	24.5 / 0.899 / 122.2	31.7 / 0.941 / 118.8	**38.7 / 0.986 / 15.7**

Table 5: Compression ratio of different input encoding methods, and specialized compression methods with $\geq 45dB$ average frame PSNR.

Dataset	MHE	MHE Tinycudann	DG Single-Res	DG Multi-Res	F-Hash Int	F-Hash Iso	F-Hash Seg	NeurComp	SZ3	TTHRESH
Combustion	1.9×	3.5×	5.1×	4.5×	14.3×	13.2×	9.5×	1407.3×	51.4×	1023.3×
Argon Bubble	3.7×	6.2×	12.8×	11.2×	55.2×	47.1×	46.8×	1223.8×	54.6×	1107.4×
Supernova	1.1×	2.4×	9.2×	8.1×	11.5×	14.3×	11.6×	1371.2×	61.7×	1240.3×

lossy compressors. 3) The dimension of the coreset (FBB) becomes closer to the input volume when the features of interest are spatially distant across frames, although this is rare in real-world scenarios, leading to longer training time. Future work can explore the various aspects of the architecture. The existing fold parameter is predefined, a data-adaptive approach could optimize these parameters, thereby further reducing the required encoding parameters. We also want to further optimize the coreset selection step to construct a more compact feature region with a fast look-up from the embedding entry to the hash bucket.

Table 6: Average rendering latency.

Dataset	Sample Streaming/ Sample Streaming + ARM (ms) MHE	MHE Tinycudann	DG Single-Res	DG Multi-Res	F-Hash
Combustion	385.4 / 253.5	211.1 / 136.6	144.5 / 97.1	165.2 / 110.9	**77.6 / 53.8**
Argon Bubble	898.5 / 592.4	531.5 / 351.9	256.9 / 168.8	293.6 / 192.4	**63.01 / 43.4**
Supernova	3592.5 / 2367.5	1686.1 / 1112.6	432.8 / 284.6	494.8 / 326.6	**345.3 / 228.1**

7 Conclusion

In this work, we present F-Hash, a novel feature-based multi-resolution Tesseract encoding achieving state-of-the-art convergence speed for training an INR for time-varying volumetric data. Our encoding method is a unified encoding solution for various time-varying features. We validate the proposed architecture through comprehensive qualitative and quantitative experiments on three large-scale time-varying volumetric datasets. The proposed adaptive ray marching algorithm improves the rendering latency of visualizing time-varying volumetric neural representation. Our input encoding method can empower a wide range of applications leveraging implicit neural representation for efficient training from complex and multi-dimensional data.

Acknowledgement

This research has been sponsored in part by the National Science Foundation grants IIS-1423487 and IIS-1652846, and Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under contracts DE-AC02-06CH11357, program manager Hal Finkel. The authors would like to thank the anonymous reviewers for their insightful comments.

References

[1] Z. Bai, Y. Tao, and H. Lin. Time-varying volume visualization: A survey. *Journal of Visualization*, 23:745–761, 2020. 2, 3

[2] Z. Bai, Y. Tao, and H. Lin. Time-varying volume visualization: a survey. *J. Vis.*, 23(5):745–761, 17 pages, Oct. 2020. doi: 10.1007/s12650-020-00654-x 7

[3] R. Ballester-Ripoll, P. Lindstrom, and R. Pajarola. Tthresh: Tensor compression for multidimensional visual data. *IEEE Transactions on Visualization and Computer Graphics*, 26(9):2891–2903, 2020. doi: 10.1109/TVCG.2019.2904063 8

[4] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. P. Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 5855–5864, October 2021. 2, 3

[5] D. Bauer, Q. Wu, and K.-L. Ma. Fovolnet: Fast volume rendering using foveated deep neural networks. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):515–525, 2023. doi: 10.1109/TVCG.2022.3209498 2

[6] E. R. Chan, C. Z. Lin, M. A. Chan, K. Nagano, B. Pan, S. De Mello, O. Gallo, L. J. Guibas, J. Tremblay, S. Khamis, T. Karras, and G. Wetzstein. Efficient geometry-aware 3d generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16123–16133, June 2022. 7

[7] S. Devkota and S. Pattanaik. Efficient neural representation of volumetric data using coordinate-based networks. *Computer Graphics Forum*, 42(7):e14955, 2023. doi: 10.1111/cgf.14955 2

[8] S. Dutta and H.-W. Shen. Distribution driven extraction and tracking of features for time-varying data analysis. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):837–846, 2016. doi: 10.1109/TVCG.2015.2467436 2

[9] S. Fridovich-Keil, A. Yu, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5501–5510, June 2022. 2, 3

[10] S. Hadadan, S. Chen, and M. Zwicker. Neural radiosity. *ACM Trans. Graph.*, 40(6), article no. 236, 11 pages, Dec. 2021. doi: 10.1145/3478513.3480569 7

[11] J. Han and C. Wang. Ssr-tvd: Spatial super-resolution for time-varying data analysis and visualization. *IEEE Transactions on Visualization and Computer Graphics*, 28(6):2445–2456, 2022. doi: 10.1109/TVCG.2020.3032123 1, 2

[12] J. Han and C. Wang. Tsr-vfd: Generating temporal super-resolution for unsteady vector field data. *Computers & Graphics*, 103:168–179, 2022. doi: 10.1016/j.cag.2022.02.001 1

[13] J. Han and C. Wang. Coordnet: Data generation and visualization generation for time-varying volumes via a coordinate-based neural network. *IEEE Transactions on Visualization and Computer Graphics*, 29(12):4951–4963, 2023. doi: 10.1109/TVCG.2022.3197203 1, 2

[14] J. Han, H. Zheng, and C. Bi. Kd-inr: Time-varying volumetric data compression via knowledge distillation-based implicit neural representation. *IEEE Transactions on Visualization and Computer Graphics*, 30(10):6826–6838, 2024. doi: 10.1109/TVCG.2023.3345373 1, 2

[15] 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*, 28(1):270–280, 2022. doi: 10.1109/TVCG.2021.3114815 1, 2

[16] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey. Meta-learning in neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):5149–5169, 2022. doi: 10.1109/TPAMI.2021.3079209 2

[17] C. Huang and H. Wang. A novel key-frames selection framework for comprehensive video summarization. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(2):577–589, 2020. doi: 10.1109/TCSVT.2019.2890899 3

[18] J. Kehrer and H. Hauser. Visualization and visual analysis of multifaceted scientific data: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 19(3):495–513, 2013. doi: 10.1109/TVCG.2012.110 7

[19] A. Kumpf, M. Rautenhaus, M. Riemer, and R. Westermann. Visual analysis of the temporal evolution of ensemble forecast sensitivities. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):98–108, 2019. doi: 10.1109/TVCG.2018.2864901 2

[20] X. Liang, K. Zhao, S. Di, S. Li, R. Underwood, A. M. Gok, J. Tian, J. Deng, J. C. Calhoun, D. Tao, Z. Chen, and F. Cappello. Sz3: A modular framework for composing prediction-based error-bounded lossy compressors. *IEEE Transactions on Big Data*, 9(2):485–498, 2023. doi: 10.1109/TBDATA.2022.3201176 8

[21] P. Ljung, J. Krüger, E. Groller, M. Hadwiger, C. D. Hansen, and A. Ynnerman. State of the art in transfer functions for direct volume rendering. *Computer Graphics Forum*, 35(3):669–691, 2016. doi: 10.1111/cgf.12934 2

[22] Y. Lu, K. Jiang, J. A. Levine, and M. Berger. Compressive neural representations of volumetric scalar fields. *Computer Graphics Forum*, 40(3):135–146, 2021. doi: 10.1111/cgf.14295 2, 8

[23] J. Lukasczyk, G. Aldrich, M. Steptoe, G. Favelier, C. Gueunet, J. Tierny, R. Maciejewski, B. Hamann, and H. Leitte. Viscous fingering: A topological visual analytic approach. In *Physical Modeling for Virtual Manufacturing Systems and Processes*, vol. 869 of *Applied Mechanics and Materials*, pp. 9–19. Trans Tech Publications Ltd, 9 2017. doi: 10.4028/www.scientific.net/AMM.869.9 2

[24] K.-L. Ma. Visualizing time-varying volume data. *Computing in Science & Engineering*, 5(2):34–42, 2003. doi: 10.1109/MCISE.2003.1182960 2

[25] I. Mehta, M. Gharbi, C. Barnes, E. Shechtman, R. Ramamoorthi, and M. Chandraker. Modulated periodic activations for generalizable local functional representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 14214–14223, October 2021. 2

[26] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: representing scenes as neural radiance fields for view synthesis. *Commun. ACM*, 65(1):99–106, 8 pages, Dec. 2021. doi: 10.1145/3503250 2, 3

[27] T. Müller, A. Evans, C. Schied, and A. Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4), article no. 102, 15 pages, July 2022. doi: 10.1145/3528223.3530127 2, 3, 7

[28] T. Müller, F. Rousselle, J. Novák, and A. Keller. Real-time neural radiance caching for path tracing. *ACM Trans. Graph.*, 40(4), article no. 36, 16 pages, July 2021. doi: 10.1145/3450626.3459812 7

[29] H. Saikia and T. Weinkauf. Global feature tracking and similarity estimation in time-dependent scalar fields. *Computer Graphics Forum*, 36(3):1–11, 2017. doi: 10.1111/cgf.13163 2

[30] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein. Implicit neural representations with periodic activation functions. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds., *Advances in Neural Information Processing Systems*, vol. 33, pp. 7462–7473. Curran Associates, Inc., 2020. 2

[31] V. Sitzmann, M. Zollhoefer, and G. Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds., *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019. 2

[32] C. Sun, M. Sun, and H.-T. Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5459–5469, June 2022. 2, 3

[33] J. Sun, D. Lenz, H. Yu, and T. Peterka. Adaptive multi-resolution encoding for interactive large-scale volume visualization through functional approximation. In *2024 IEEE 14th Symposium on Large Data Analysis and Visualization (LDAV)*, pp. 33–42, 2024. doi: 10.1109/LDAV64567.2024.00006 2

[34] J. Sun, D. Lenz, H. Yu, and T. Peterka. Make the fastest faster: Importance mask for interactive volume visualization using reconstruction neural networks. *arXiv preprint arXiv:2502.06053*, 2025. 2

[35] J. Sun, X. Xie, and H. Yu. Rmdncache: Dual-space prefetching neural network for large-scale volume visualization. *IEEE Transactions on*

Visualization and Computer Graphics, pp. 1–13, 2024. doi: 10.1109/TVCG.2024.3410091 2

[36] M. Tancik, P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. Barron, and R. Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds., *Advances in Neural Information Processing Systems*, vol. 33, pp. 7537–7547. Curran Associates, Inc., 2020. 2, 3

[37] D. Tang, S. Singh, P. A. Chou, C. Hane, M. Dou, S. Fanello, J. Taylor, P. Davidson, O. G. Guleryuz, Y. Zhang, et al. Deep implicit volume compression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1293–1303, 2020. 2

[38] H. Tang, L. Ding, S. Wu, B. Ren, N. Sebe, and P. Rota. Deep unsupervised key frame extraction for efficient video classification. *ACM Trans. Multimedia Comput. Commun. Appl.*, 19(3), article no. 119, 17 pages, Feb. 2023. doi: 10.1145/3571735 3

[39] K. Tang and C. Wang. Ecnr: Efficient compressive neural representation of time-varying volumetric datasets. *arXiv preprint arXiv:2311.12831*, 2023. 1, 2

[40] K. Tang and C. Wang. Stsr-inr: Spatiotemporal super-resolution for multivariate time-varying volumetric data via implicit neural representation. *Computers & Graphics*, 119:103874, 2024. doi: 10.1016/j.cag.2024.01.001 1, 2

[41] M. Teschner, B. Heidelberger, M. Müller, D. Pomerantes, and M. H. Gross. Optimized spatial hashing for collision detection of deformable objects. In *Vmv*, vol. 3, pp. 47–54, 2003. 5

[42] K.-C. Wang, T.-H. Wei, N. Shareef, and H.-W. Shen. Ray-based exploration of large time-varying volume data using per-ray proxy distributions. *IEEE Transactions on Visualization and Computer Graphics*, 26(11):3299–3313, 2020. doi: 10.1109/TVCG.2019.2920130 2

[43] S. Weiss, P. Hermüller, and R. Westermann. Fast neural representations for direct volume rendering. *Computer Graphics Forum*, 41(6):196–211, 2022. doi: 10.1111/cgf.14578 2

[44] W. Widanagamaachchi, J. Chen, P. Klacansky, V. Pascucci, H. Kolla, A. Bhagatwala, and P.-T. Bremer. Tracking features in embedded surfaces: Understanding extinction in turbulent combustion. In *2015 IEEE 5th Symposium on Large Data Analysis and Visualization (LDAV)*, pp. 9–16, 2015. doi: 10.1109/LDAV.2015.7348066 2

[45] W. Widanagamaachchi, A. Jacques, B. Wang, E. Crosman, P.-T. Bremer, V. Pascucci, and J. Horel. Exploring the evolution of pressure-perturbations to understand atmospheric phenomena. In *2017 IEEE Pacific Visualization Symposium (PacificVis)*, pp. 101–110, 2017. doi: 10.1109/PACIFICVIS.2017.8031584 2

[46] Q. Wu, D. Bauer, M. J. Doyle, and K.-L. Ma. Interactive volume visualization via multi-resolution hash encoding based neural representation. *IEEE Transactions on Visualization and Computer Graphics*, 30(8):5404–5418, 2024. doi: 10.1109/TVCG.2023.3293121 2, 3, 6

[47] M. Yang, K. Tang, and C. Wang. Meta-inr: Efficient encoding of volumetric data via meta-learning implicit neural representation. *arXiv preprint arXiv:2502.09669*, 2025. 2

[48] S. Yao, Y. Lu, and C. Wang. Visnerf: Efficient multidimensional neural radiance field representation for visualization synthesis of dynamic volumetric scenes. *arXiv preprint arXiv:2502.16731*, 2025. 2

[49] L. Yariv, J. Gu, Y. Kasten, and Y. Lipman. Volume rendering of neural implicit surfaces. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, eds., *Advances in Neural Information Processing Systems*, vol. 34. Curran Associates, Inc., 2021. 2

[50] Q. Zhong, Y. Zhang, J. Zhang, K. Shi, Y. Yu, and C. Liu. Key frame extraction algorithm of motion video based on priori. *IEEE Access*, 8:174424–174436, 2020. doi: 10.1109/ACCESS.2020.3025774 3

[51] B. Zhou and Y.-J. Chiang. Key time steps selection for large-scale time-varying volume datasets using an information-theoretic storyboard. *Computer Graphics Forum*, 37(3):37–49, 2018. doi: 10.1111/cgf.13399 2