Make the Fastest Faster: Importance Mask Synthesis for Interactive Volume Visualization using Reconstruction Neural Networks

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

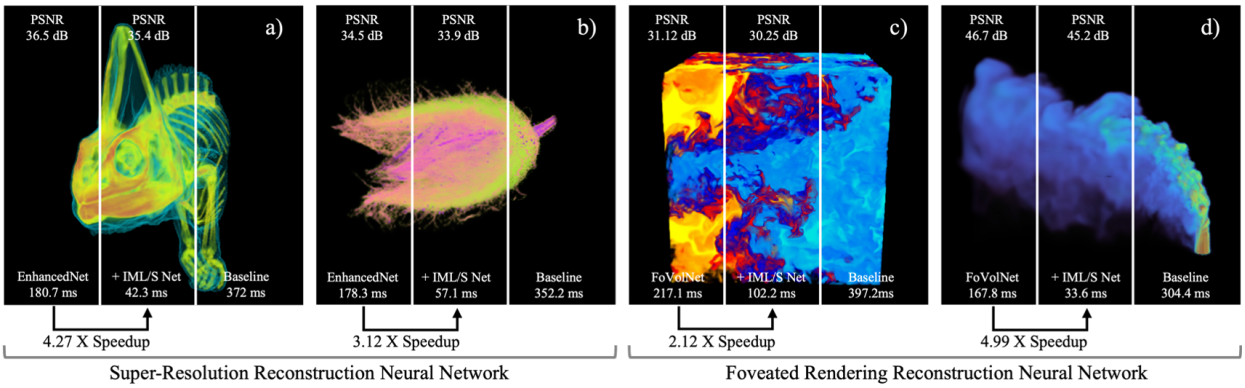


Fig. 1: The proposed IML and IMS Networks can further reduce the rendering latency of already fast volume visualization methods using reconstruction neural networks, like the super-resolution EnhanceNet (a and b) and foveated rendering FoVolNet (c and d), with similar rendering quality. For each subfigure, the rightmost portion is the baseline using the traditional ray casting method, the leftmost portion is the interactive visualization method using reconstruction neural network, and the middle portion is the proposed method using our IML/S Net that enhances the performance of respective reconstruction neural network.

Abstract— Visualizing a large-scale volumetric dataset with high resolution is challenging due to the substantial computational time and space complexity. Recent deep learning-based image inpainting methods significantly improve rendering latency by reconstructing a high-resolution image for visualization in constant time on GPU from a partially rendered image where only a portion of pixels go through the expensive rendering pipeline. However, existing solutions need to render every pixel of either a predefined regular sampling pattern or an irregular sample pattern predicted from a low-resolution image rendering. Both methods require a significant amount of expensive pixel-level rendering. In this work, we provide Importance Mask Learning (IML) and Synthesis (IMS) networks, which are the first attempts to directly synthesize important regions of the regular sampling pattern from the user's view parameters, to further minimize the number of pixels to render by jointly considering the dataset, user behavior, and the downstream reconstruction neural network. Our solution is a unified framework to handle various types of inpainting methods through the proposed differentiable compaction/decompaction layers. Experiments show our method can further improve the overall rendering latency of state-of-the-art volume visualization methods using reconstruction neural network for free when rendering scientific volumetric datasets. Our method can also directly optimize the off-the-shelf pre-trained reconstruction neural networks without elongated retraining.

Index Terms—Large-scale data, interactive visualization, deep learning, reconstruction neural network

1 Introduction

Interactive volume visualization techniques are crucial for enabling researchers across various disciplines to efficiently discover insightful features within scientific datasets from fields like medical imaging, geophysics, meteorology, materials science, and physical simulations. A visualization system that quickly responds to user interactions, adapting to data and viewing adjustments, can greatly enhance the efficiency and effectiveness of exploring complex scientific datasets. However, interactive volume visualization is challenging because of the following factors: First, the size of volumetric data generated nowadays is growing exponentially, necessitating an efficient storage and I/O system to facilitate large-scale data streaming between different computing units for generating comprehensive visualization results. Second, visualization techniques, such as ray casting, ray tracing, global illumination, and high-order rendering, often require significant computational resources to generate high-resolution, high-fidelity images.

Various strategies have been proposed to address this performance issue in large-scale volumetric data visualization. For example, distributed volume visualization methods [35, 48] harness the parallel machines to divide the workload and then composite the partially rendered results into the final rendering image. Multi-resolution methods [8, 9, 12, 36] optimize the arrangement of data segments with different levels of detail (LOD) to improve the rendering latency while maintaining a high rendering quality. To improve the smoothness of volumetric data exploration, visualization systems also utilize caching and prefetching [7, 23] to reduce out-of-core data movement across the system memory hierarchy, thereby decreasing input latency while rendering continuous frames along a user's exploratory trajectory. The user behavior while exploring volumetric data can be learned [13, 39] to better assist the prefetching through the prediction of view parameters.

Recent image synthesis techniques utilizing deep learning-based super-resolution in both spatial and temporal domains [10, 15, 19] are being adopted for visualizing large-scale volumetric datasets with responsive input latency. Given a partially rendered image from a view angle, where only a small percentage of the total pixels are rendered,

- *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

the idea is to predict or interpolate the missing pixels through a trained neural network in constant time by leveraging the parallelization of the GPU instead of going through expensive rendering algorithms to calculate color blending for each pixel of the final image. The network that reconstructs an incomplete image into a full image is called a reconstruction neural network, and we refer to such a network as *RecNN* in the rest of this paper. Those methods rely on training an implicit neural representation [33, 34] using the original large-scale dataset and a complex rendering pipeline, and once trained, the representation enables fast inference, making it well-suited for interactive visualization tasks. Volume visualization methods based on RecNNs have achieved state-of-the-art performance in rendering latency [3, 28], particularly when processing large-scale scientific datasets with complex lighting effects.

The bag of pixels to render for each partially rendered image is determined by a binary sampling pattern provided by the specific reconstruction neural network used (e.g., downsampling patterns for super-resolution RecNNs and foveal patterns for foveated rendering RecNNs). However, such a pattern is a predefined hyperparameter before training, and it does not adapt to any of the key visualization parameters like the user's views, transfer functions, or features of the underlying dataset. Recently proposed Learning Adaptive Sampling (LAS) [45] tries to predict an arbitrary selection of important pixels, called the Importance Mask (IM), from a low-res rendering image. The ratio of the important pixels to the total rendering pixels is called the Important Pixel Ratio (IPR). Although LAS reduces the number of pixels to render for the downstream RecNN, it suffers from several key drawbacks that limit its practicality and lead to suboptimal performance gains: 1) Rendering the low-res input image still takes a relatively long time. 2) The IM can only be learned from the 2D space with the same resolution of the final rendering image, which can be very high for scientific visualization (normally $\geq 512 \times 512$), resulting in a large number of samples to render for a specific IPR with a certain quality level. 3) LAS only supports the super-resolution type of RecNNs and needs a long time to retrain its weights. In this work, we provide a unified Importance Mask Learning Network (IML Net) to learn and distill the IM from a low-res intermediate embedding supporting various types of downstream RecNNs through the proposed compaction/decompaction layers. IML Net learns the IM directly from the sampling pattern by jointly considering the dataset, the user's view parameters, and the downstream RecNN. We also proposed an Importance Mask Synthesis Network (IMS Net), which will directly synthesize the IM from novel views during the user's exploration. Our neural rendering network (IML/S Net) combines the two networks with the downstream RecNN, achieving much faster rendering than LAS with similar rendering quality under the same IPR. We use two state-of-the-art image inpainting RecNNs for volume visualization, EnhanceNet [28] for super-resolution, and FoVolNet [3] for foveated rendering, to showcase how our method can further reduce the rendering latency for free without sacrificing noticeable rendering quality. We also perform a comprehensive comparison with LAS for both rendering quality and latency. The main contributions of this work include:

- Importance Mask Learning Network(IML Net), a unified neural network to learn the IM from low-res intermediate embedding derived from diverse sampling patterns by jointly considering the volumetric dataset, the user's view parameters, and the downstream reconstruction neural network.
- Importance Mask Synthesis Network(IMS Net), a regressor designed to efficiently predict the importance mask directly from view parameters instead of computationally expensive input.
- IML/S Net + RecNN, a visualization neural rendering network to further improve the already fast volume visualization methods using RecNN for free with similar rendering quality.
- Our networks can also be efficiently trained independently of the downstream RecNN to optimize the rendering latency for off-the-shelf pre-trained RecNN.

2 Related Work

2.1 Implicit Neural Representation

Implicit Neural Representation (INR) refers to a method of representing data, such as images, 3D shapes, or other types of continuous signals, using a neural network. For handling large-scale volume visualization, compression methods utilizing implicit neural representation [22, 41] have been proposed to reduce network size and optimize I/O-intensive operations. To improve the rendering latency of the visualization system, Wu et al. [46] utilize hash encoding-based INR to accelerate interactive volume visualization with high reconstruction quality. Yariv et al. [47] improve geometry representation and reconstruction in neural volume rendering by modeling the volume density as an implicit function of the geometry. Weiss et al. [44] introduce fV-SRN as a novel extension of SRN (Scene Representation Networks) to achieve significantly accelerated reconstruction performance of volume rendering. Despite the use of powerful GPUs, the training time for INR remains significantly long when processing complex, large-scale scientific datasets [42, 46]. Sun et al. [38] propose F-Hash to significantly speed up the convergence of training INR from large-scale time-varying scientific data through a feature-based multi-resolution Tesseract input encoding. Both the proposed IML and IMS Nets were trained as INRs with learned properties of the dataset, view parameters, and the downstream RecNN.

2.2 Reconstruction Neural Network

A Reconstruction Neural Network (RecNN) in visualization refers to a type of neural network used to reconstruct or generate detailed representations of data from incomplete, noisy, or compressed input [6, 20]. The goal of RecNN is to recover or restore a high-quality, detailed version of an object or dataset from a simplified or partial form. The most commonly used type of RecNN is super-resolution neural networks that reconstruct high-res data from low-res data. The sampling pattern of super-resolution RecNN is a downsampling pattern. The EnhanceNet, proposed by Sajjadi et al. [28], demonstrates high reconstruction quality in generating volume visualization images [45]. Weiss et al. [43] propose a volumetric isosurface rendering with deep learning-based super-resolution. Tang et al. [42] propose STSR-INR, which extends the super-resolution to both spatial and temporal domains. Another important RecNN is the foveated rendering RecNN, which does reconstruction from partial rendering near the region where the user is focusing their gaze. Together with the techniques in eye-tracking-related research [40, 50], foveated rendering provides a solution to balance the user experience and computational cost. The sampling pattern of foveated rendering RecNN is a foveal pattern. Kaplanyan et al. propose DeepFovea [17], the first foveated reconstruction method utilizing generative adversarial networks (GAN) to speed up the rendering frame rate for gaming. Then Bauer et al. propose FoVolNet [3] to improve the foveated rendering in the volume visualization domain and achieve state-of-the-art rendering latency. In this work, we select two RecNNs, EnhanceNet and FoVolNet, as representations of super-resolution RecNN and foveated rendering RecNN to demonstrate how much rendering latency can be further reduced using our method.

2.3 Image Synthesis

Deep learning-based image synthesis in volume visualization refers to the use of deep learning techniques to generate or enhance visual representations of 3D volumetric data, such as medical scans (MRI, CT), scientific simulations, or geological models [21]. Volumetric data can be very large, making real-time rendering computationally expensive. Generative models can efficiently synthesize rendering results by learning the underlying structure of the data, thereby reducing the need for expensive pixel-level computation and enabling faster generation of visual results. Berger et al. [4] utilize a GAN framework to directly generate the volume visualization image from view parameters and transfer functions. He et al. [11] provide a regressor to directly generate visualization images from simulation and visualization parameters for fast parameter space exploration. Jun et al. propose VCNet, a new deep-learning approach for volume completion by synthesiz-

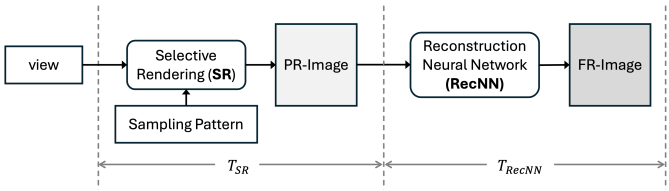


Fig. 2: Typical pipeline of interactive volume visualization using RecNN.

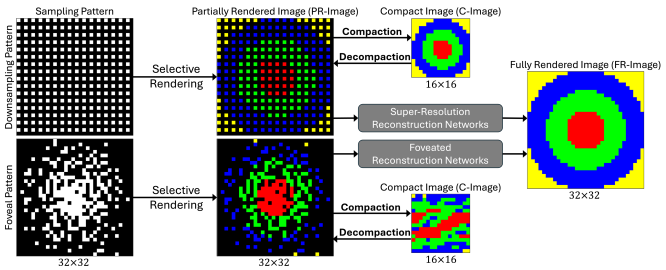


Fig. 3: Demonstration of compaction and decompaction for super-resolution RecNN and foveated rendering RecNN.

ing missing subvolumes. Our IMS Net also utilizes image synthesis techniques to generate an importance mask directly from view parameters. Although training generative models, especially for large-scale or high-dimensional data, requires large training datasets and significant computational resources, the importance masks learned by the proposed IML Net are low-res 2D binary matrices that can be efficiently learned by the IMS Net.

3 Methods

3.1 Overview

For a general volume visualization system, the rendering latency (RL) is the time difference between the moment the user makes a data- or view-dependent operation and the time when the rendering of the visualization image is finished. For interactive volume visualization using RecNN, as shown in Fig. 2, the rendering latency (T_{RL}) is the sum of two components, which are the time of Selective Rendering (SR) for generating the partially rendered image (PR-Image), T_{SR}, and the inferencing time of the RecNN to generate a fully rendered image (FR-Image) from PR-Image, T_{RecNN}:

$$T_{RL} = T_{SR} + T_{RecNN} \tag{1}$$

Once the RecNN is defined and trained, its inferencing time (T_{RecNN}) can be accelerated by GPU in constant time. We define the pixels covered by the sampling pattern as S_{sp}. For a specific sampling pattern, T_{SR} is the sum of time used to render each pixel in the sampling pattern defined by the specific RecNN.

$$T_{SR} = \sum_{k \in S_{sp}} T_k \tag{2}$$

The time used to render the kth pixel, T_k, is determined by the complexity of the chosen rendering technique (ray casting [26], ray tracing [24], global illumination [51] etc.) and the interpolation method (linear or high-order [37]). The time used to render the PR-Image, T_{SR}, is the main overhead of the overall rendering latency. Because of the high computational cost of pixel rendering, decreasing the size of S_{sp} can significantly reduce the rendering latency. However, the size of S_{sp} is predefined by the sampling pattern, which is a fixed hyperparameter for existing RecNNs. In order to further decrease the computational cost, a new set of important pixels S_{im} needs to be selected from S_{sp}, and the rendering of S_{im} can losslessly reconstruct the rendering of S_{sp}. We refer to the binary mask that filters out S_{im} from S_{sp} as the Importance Mask (IM) for the remainder of the paper. For a chosen rendering function $R()$ and reconstruction function $f()$. The optimal S_{im} can be derived from:

$$S_{im} = \arg \min_{|R(S_{sp}) - f(R(S_{im}))| < \varepsilon} |S_{im}| \tag{3}$$

where ε is a small predefined interval of error metric between the rendering of S_{sp} and the reconstruction from the rendering of S_{im}. $|S_{im}|$ is the number of important pixels. Our objective is to select an optimal subset from the sampling pattern with the fewest possible pixels while ensuring that the rendering of this subset can accurately reconstruct the full sampling pattern with minimal reconstruction error. We name the percentage of pixels when using optimal S_{im} as the Optimal Rendering Percentage (ORP).

The volumetric datasets, mostly collected from scientific domains, show strong spatial correlations in their 3D domains. Reflected on the 2D image domain, such spatial correlation can also be observed. This provides opportunities to compress the complex pixel-level rendering into compact representations. Although there are numerous existing works to extract superpixels directly from 2D datasets [2], S_{im} needs to be selected by not only considering the data itself but also jointly considering the users' input and downstream RecNN. Inspired by the saliency map [31] demonstrating which parts of the input data (e.g., pixels in an image) contribute most to the predictions of a neural network, we provide a method to automatically learn a selection of pixels that are the most important to reconstructing the rendering of the sampling pattern. Instead of calculating the gradient of output with respect to the input, as the saliency map does, we utilize an autoencoder architecture to learn the IM as a latent feature through backpropagation from the training dataset. In this work, we propose two neural networks: the Importance Mask Learning Network (IML Net) and the Importance Mask Synthesis Network (IMS Net). IML Net learns the importance mask from training datasets, which are visualization images generated from the respective user's view-dependent operations. Once the importance masks are learned, they will then be used as a training dataset for the IMS Net, which will learn the mapping from the user's view to the IM through novel view synthesis.

3.2 Differentiable Compaction and Decompaction

The goal of our work is to derive an optimal S_{im} from S_{sp}, however, the shape of the sampling pattern varies for different RecNN. For example, super-resolution RecNN uses a downsampling pattern while the foveated rendering RecNN uses a foveal pattern. The downsampling pattern is evenly distributed across the PR-Image. The foveal pattern generally refers to the region of an image where the eye's fovea (central vision) is focused, which is where human vision is most detailed. In foveated rendering, only the pixels in this high-resolution foveal area are rendered in full detail, while the surrounding areas corresponding to peripheral vision are rendered at a much lower resolution. To uniformly process sampling patterns with different distributions, we introduce a compaction layer to compact the pixels in the sampling pattern into a regular 2D matrix. Fig. 3 shows how the PR-Image is selectively rendered according to downsampling and foveal patterns, and how the PR-Image is compacted into a regular 2D matrix. We name this 2d matrix a compact image (C-Image). The compaction collects each pixel within the sampling pattern by following a straightforward Raster scan order on the PR-Image. If the C-Image is not filled after compaction, zeros are padded to its empty pixels. We can see that the C-Image of super-resolution RecNN is the corresponding low-res version of the FR-Image, while the C-Image of foveated rendering RecNN is more irregular. The compaction layer is a practice of dataset distillation where only informative pixels are gathered for efficient training. Decompaction is the reverse of compaction. The compaction and decompaction layers are 2D point transformations between PR-Image space and C-Image space. To make the compaction and decompaction layers differentiable, a transformation is implemented as a translation matrix multiplying each pixel in one space to map its value to the transformed location of another space. During compaction, the C-Image is calculated through the linear transformation:

$$C\text{-}Image = PR\text{-}Image \times C_{RecNN} \tag{4}$$

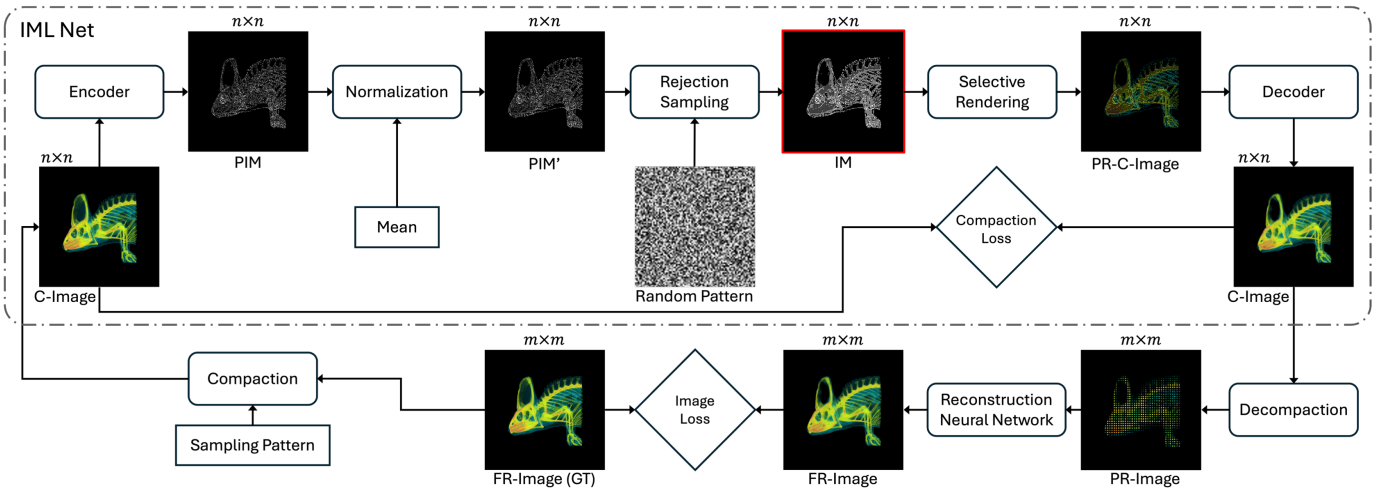


Fig. 4: Network architecture of IML Net. IML Net works on a compact 2D embedding domain (C-Image) with resolution of $n \times n$, which is much smaller than the full resolution output with a resolution of $m \times m$ ($m \gg n$).

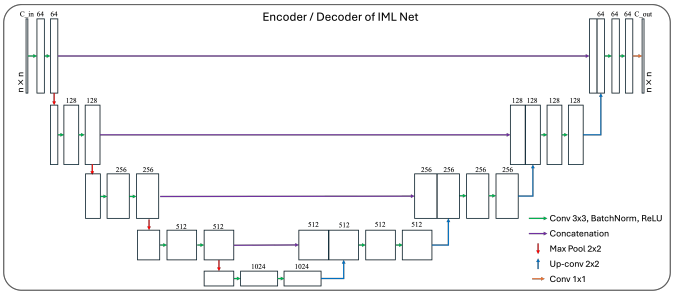


Fig. 5: Network architecture of the encoder and decoder in IML Net. The input image resolution is $n \times n$.

Where C_{RecNN} is the Compaction Matrix, which is a 2D constant matrix defined by the specific RecNN. During decompaction, the PR-Image can be retrieved:

$$PR\text{-}Image = C\text{-}Image \times D_{RecNN}, \quad D_{RecNN} = C_{RecNN}^{-1} \tag{5}$$

Where the Decompaction Matrix D_{RecNN} is the inverse of the C_{RecNN}. The benefit of incorporating compaction and decompaction layers includes: 1) Reduce the dimension of the redundant training data into a compact subset for efficient Meta Learning. 2) Decrease the number of parameters of the IML network. 3) Improve the speed of convergence during training. 4) Improve the speed of inferencing for interactive visualization. 5) Allow the proposed IML and IMS Networks to remain agnostic to diverse sampling patterns within a unified network architecture. Compared to the LAS, the proposed compaction/decompaction enables learning from a low-res intermediate embedding (C-Image), therefore, further reducing the expensive pixel calculation for faster rendering.

3.3 Importance Mask Learning Network

The functions of IML Net include: 1) For each rendered sampling pattern, learn an IM that selects only the important pixels from it to render for the downstream reconstructions. 2) Learn a decoder that reconstructs the full rendering of the C-Image from the partially rendered C-Image (PR-C-Image) based on the learned IM.

3.3.1 Network Architecture

Fig. 4 shows the network architecture of IML Net. IML Net is an autoencoder neural network that takes a C-Image after compaction as input and outputs an image as close as the C-Image. The number of pixels and their distribution on C-Image are defined by the specific sampling pattern selected. The IM is learned on top of it by optimizing the network parameters through backpropagation during training. Key components of the network are:

Encoder and Decoder: The first and the last subnetworks of the IML Net are encoder and decoder sharing the same network structure as shown in Fig. 5. We utilize a standard U-Net [27] to construct the encoder/decoder network. Our IML Net is invariant to the input C-image resolution due to the use of convolutional layers. For the encoder, C_in is set to 3 for taking the RGB C-Image after compaction as input, C_out is set to 1 for outputting a grayscale image where the value of each pixel correlates to the probability of being an important pixel. The U-Net structure of the encoder can effectively learn such pixel values to find the IM. After the last layer of the encoder, a softplus layer is used to convert the output value into the positive range for the following normalization layer. For the decoder, both C_in and C_out are set to 3 for taking the partially rendered C-Image (PR-C-Image) as input and outputting an image close to the input C-Image of the encoder. The decoder of our IML network performs a low-level in-painting to complete the missing pixels in the PR-C-Image to reconstruct the complete C-Image.

Normalization: The output of the encoder presents preliminary information about the importance of each pixel, and we name this output Preliminary Importance Mask (PIM). We apply a similar practice proposed by Weiss et al. [45] to normalize the PIM with a hyperparameter, called mean value, which will control the percentage of C-Image pixels selected as important pixels. The normalized PIM, can be computed:

$$PIM'_{ij} = l + PIM_{ij} \frac{u - l}{u_{PIM} + \delta} \tag{6}$$

where the i and j are pixel indices, u is the prescribed mean value, and l is the lower bound on the sample distribution, which is set to 0 in this work. u_{PIM} is the mean of PIM. δ is a small constant set as 10^{-7} to avoid zero division. All the above operations ensure that the normalization layer remains differentiable. It is worth noticing that while the normalization step can also be performed using a softmax layer, this approach may result in a loss of control over the proportion of C-Image pixels designated as important through the prescribed mean. Our normalization step enables the hyperparameter tuning to investigate the relationship between the sparsity of important pixels and reconstruction performance.

Rejection Sampling: The rejection sampling layer is to generate a binary IM from the PIM' with continuous values. The practice is filtering out the important pixel through a comparison between the pixel value in PIM' and a randomly sampled value. The rationale behind this practice is due to the spatial correlation found in the prediction from image generative neural networks. This fact makes the direct

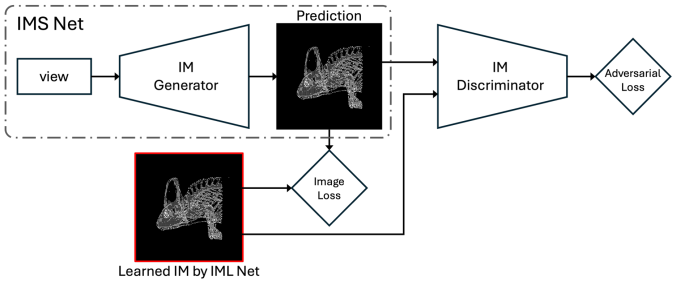


Fig. 6: Network architecture of IMS Net.

learning of the important pixel as a blob of connected pixels, which doesn't distinguish the underlying structure. Comparing the prediction with an uncorrelated random pattern can solve this issue and create isolated pixels that reflect the importance of the underlying structure. The random pattern P is a grayscale image with the same dimensions as C-Image, with its pixel value randomly sampled from a uniform distribution between 0 and 1. We use plastic sampling to generate P for the superior quality of the learned IM compared to other sampling strategies like random sampling, regular sampling, and Halton sampling. The IM can be retrieved through an independent Bernoulli process through rejection sampling.

$$IM_{ij} = \begin{cases} 1, & \text{if } PIM'_{ij} > P_{ij} \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

To make the rejection sampling layer differentiable, the IM can be calculated as:

$$IM_{ij} = Sigmoid(\alpha(PIM'_{ij} - P_{ij})), \quad Sigmoid(x) = \frac{1}{1+e^{-x}} \tag{8}$$

where α is a positive scalar determining the steepness of the Sigmoid function. The pixel value of each IM is in the range $[0,1]$. The above equation is only used for training the IML Net. When using IML Net during the inferencing, which will be discussed in Sec. 3.4.4, only the regular rejection sampling of Eq. (7) is used. In that case, a binary mask using 1 bit per pixel is enough to represent the importance mask.

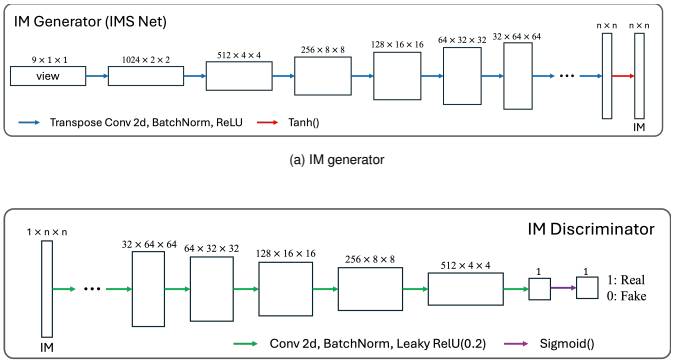
Selective Rendering: The selective rendering layer will generate the partially rendered C-Image (PR-C-Image) according to the IM. During training, this step is basically element-wise multiplication between the fully rendered C-Image (FR-C-Image) of the training dataset and IM:

$$PR\text{-}C\text{-}Image = IM \times FR\text{-}C\text{-}Image \tag{9}$$

During the inferencing stage, this layer is replaced by actual pixel-level rendering computation defined by the chosen volume rendering algorithm for learned important pixels.

3.3.2 Loss Functions

The IML Net can be trained by itself with a compaction loss function, as shown inside the IML Net block of Fig. 4, which is an image loss function between the input C-Image and reconstructed C-Image. This practice is suitable for optimizing the off-the-shelf pre-trained RecNNs without going through the long end-to-end training process resulting from adding a complex downstream RecNN. The IML Net can also be trained together with downstream RecNNs selected for various sampling patterns. In this case, the loss function is the weighted sum of the previous compaction loss and the image loss between the predicted reconstructed full-resolution image (FR-Image) and the ground truth (GT) image rendered using the baseline rendering algorithm without using RecNN. This end-to-end training will give better reconstruction results by jointly optimizing the parameters from both the IML Net and the downstream RecNN. In our work, we use the Mean Squared Error (MSE) as the image loss function and 0.5 as the weight for both image losses if trained end-to-end.



(a) IM generator

(b) IM discriminator

Fig. 7: Network architecture of the IM generator of IMS Net (a) and the IM discriminator for training the GAN (b).

3.4 Importance Mask Synthesis Network

Once the IML Network is learned, we collect the view that generates the C-Image and the learned IM as input-label pairs to train the IMS Net through supervised learning. The function of IMS Net is to quickly and directly predict the IM from a new set of view parameters.

3.4.1 Network Architecture

The focus of this paper is on visualizing scientific data, which is a time-sensitive application. Therefore, we need to keep the main steps of the pipeline, including training and inferencing, finished as fast as possible. Our training includes two steps: training the IML Net and training the IMS Net. We provide a method to speed up the training of IML network, as mentioned in Sec. 4.6.4, by training a standalone IML Net without connecting to the downstream RecNN. However, we cannot carry out the same optimization on training the IMS Net. As a result, we have to select a GAN with less training/inferencing complexity. DCGAN [25] provides a better trade-off between the complexity and the reconstruction quality than other GAN architectures like Conditional GAN (cGAN) [14] and StyleGAN [18] whose training/inferencing times are too long for scientific visualization. Fig. 6 shows the network architecture of IMS Net and the discriminator used for training. IMS Net is a generator or regressor that takes a sparse set of view parameters to generate a 2D binary image as the IM. The IMS Net is trained through the generative adversarial network (GAN). Key components of the network are:

Generator: The input to the generator is a set of view parameters. We adopted the OpenGL convention to define the view parameters as 9 float numbers:

$$view = (eye\_x/y/z,\ lookAt\_x/y/z,\ up\_x/y/z) \tag{10}$$

where the *eye* is the camera position, *lookAt* is the focal point where the camera is looking, and *up* is the "up" direction of the camera. IMS Net is a DCGAN with its detailed structure shown in Fig. 7a. It consists of several transposed convolutional layers with batch normalization and ReLU activation functions.

Discriminator: Fig. 7b shows the detailed structure of the binary classifier as a discriminator. The discriminator consisted of several convolutional layers with batch normalization and Leaky ReLU activation functions.

3.4.2 Loss Functions

As shown in Fig. 6, we consider two losses to train the IMS Net: image loss on the IM and adversarial loss. The adversarial loss is composed of two parts: the generator loss and the discriminator loss. The discriminator loss L_D is the cross-entropy of the binary classifier measuring how well the discriminator can distinguish between real and generated images.

$$L_D = -\mathbb{E}_{x\sim p_x(x)}[\log D(x)] - \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \tag{11}$$

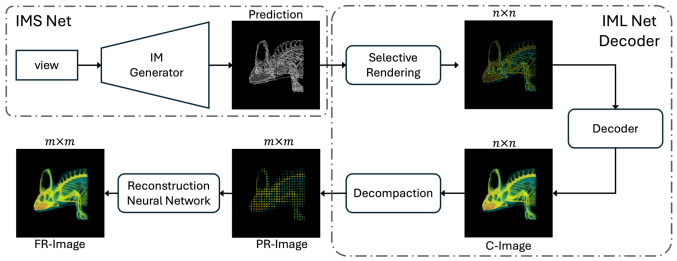


Fig. 8: Network architecture of the proposed visualization neural rendering network, IML/S Net + RecNN. IML/S is the trained IMS Net connected with the decoder of the trained IML Net.

Table 1: Volumetric datasets used in the experiments.

Dataset	Resolution	Size	Data Type
Chameleon	$1024^2 \times 1088$	2.1 GB	uint16
Beechnut	$1024^2 \times 1546$	3.0 GB	uint16
Rayleigh-Taylor	1024^3	4.0 GB	float32
Flame	$1408 \times 1080 \times 1100$	6.23 GB	float32

where $D(\mathbf{x})$ is the discriminator's output for a real image. $D(G(\mathbf{z}))$ is the discriminator's output for a fake image $G(\mathbf{z})$. The generator loss L_G used is the non-saturating loss measuring how well the generator can fool the discriminator.

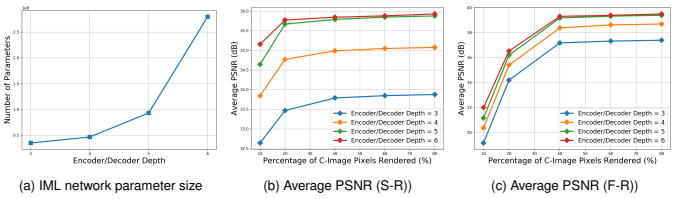
$$L_G = -\mathbb{E}_{z\sim p_z(z)}[\log D(G(z))] \tag{12}$$

$p_x(x)$ is the probability distribution of real training data and $p_z(z)$ is the probability distribution of the noise vector z. The loss function selection and configuration are crucial in training a GAN. Detailed results using different loss functions for training IMS Net are discussed in Sec. 4.5.2.

3.4.3 Training

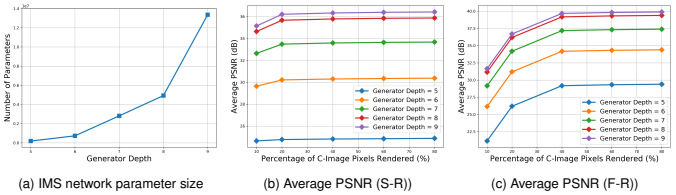
Training a generative model to directly generate volume visualization is challenging [1,4,5,29,49] for several reasons: 1) It is difficult for GAN to generate high-res images. 2) Training of GAN tends to be unstable because of the mode collapse, vanishing gradients, imbalanced learning rate, and data overlap issues. 3) While generative models can synthesize convincing volume visualization, they may struggle with very fine details or subtle structures. To prevent the aforementioned issues during the training of our generator, we utilize the following practices:

- Avoid learning the IM directly from the sparse view parameters but from the C-image, which presents more structural information, which is the motivation of the proposed IML Net. Our experiments show that the IM directly learned from view parameters is a binary mask whose important pixels are evenly distributed across the region of the volumetric object. In that case, important pixels couldn't properly capture the locations of informative regions of the rendered image.
- Train the generator to predict low-res simple patterns. The IM (as highlighted in Fig. 4) that the IMS Net tries to learn is a low-res 2D binary image with only a single channel and finite outputs (-1 or 1).
- Utilize batch normalization for both the generator and discriminator to stabilize the training.
- Using activation function ReLu for the generator and LeakyRelu(0.2) for the discriminator.
- Scale the input binary IM from $\{0,1\}$ to $\{-1,1\}$.
- Use the hyperbolic tangent function (Tanh) at the end of the generator to constrain the output between -1 and 1.
- Use adversarial loss together with advanced perceptual loss [16] instead of simple Binary Cross Entropy (BCE) or Mean Squared Error (MSE) loss. Perceptual Loss is a loss function that evaluates the similarity between images based on high-level image features, contents, and patterns, rather than pixel-wise differences. It leverages pre-trained deep neural networks (e.g., VGG [32]) to compare images at a feature level, focusing on perceptual quality and structural similarity.



(a) IML network parameter size (b) Average PSNR (S-R)) (c) Average PSNR (F-R))

Fig. 9: IML Networks with different depths of the U-Net architecture. Evaluation performed on the Chameleon dataset with a visualization image resolution of 512×512. (a) shows the total number of parameters of IML Net. (b) and (c) show the reconstruction quality of super-resolution (S-R) and foveated rendering (F-R) while increasing the percentage of C-Image rendered.



(a) IMS network parameter size (b) Average PSNR (S-R)) (c) Average PSNR (F-R))

Fig. 10: IMS network with different depth used in its generator. Evaluation performed on the Chameleon dataset with a visualization image resolution of 512×512. (a) shows the total number of parameters of the generator. (b) and (c) show the reconstruction quality of super-resolution (S-R) and foveated rendering (F-R) while increasing the percentage of C-Image rendered.

3.4.4 Inferencing

Once the IMS Net is trained, a visualization neural rendering network is constructed by combining three components: the trained IMS Net, the decoder of the trained IML Net, and the RecNN, as shown in Fig. 8. We name this network IML/S Net + RecNN. The new network directly generates a visualization image from the new view with improved rendering latency than the original RecNN. Detailed quality and latency evaluation of IML/S Net + RecNN is discussed Sec. 4.6.

4 Experiments and Evaluation

4.1 Datasets

4.1.1 Volumetric Datasets

For quality and performance evaluation, we select 4 large-scale volume datasets with distinct spatial features collected from diverse domains as detailed in Table 1. The Chameleon dataset is a CT scan of a chameleon. The Beechnut dataset is a microCT scan of a dried beechnut. The Rayleigh-Taylor dataset is a time step of a density field in a simulation of the mixing transition in Rayleigh-Taylor instability. The Flame dataset is a simulated combustion 3D scalar field. The spacings of all datasets are normalized within the spatial range [-1, 1] with values normalized within the range [0, 1].

4.1.2 Training Data

The training dataset used to train the networks is the view and fully rendered image (FR-Image) pairs. First, we randomly select 2000 views around the volume of each volumetric dataset. Second, an FR-Image is rendered using a specific volume rendering algorithm with lighting effect. In this work, we select the commonly used ray casting DVR as the renderer. Third, filter the FR-Image with the sampling pattern to generate the partially rendered image (PR-Image), where only pixels within the sampling pattern are kept. Fourth, apply compaction on PR-Image to generate C-Image. For training the IML Net standalone, the input is C-Image while the output is the predicted C-Image from the decoder.

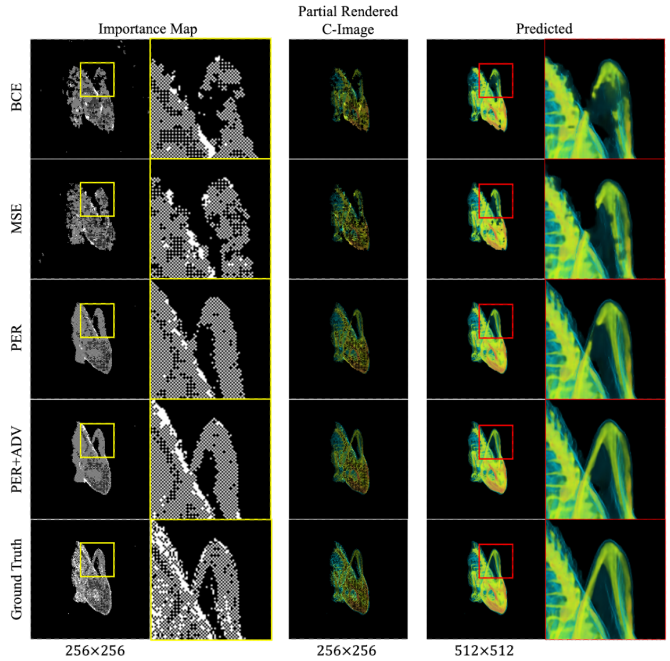


Fig. 11: IM prediction quality comparison using different configurations of loss functions on the Chameleon dataset.

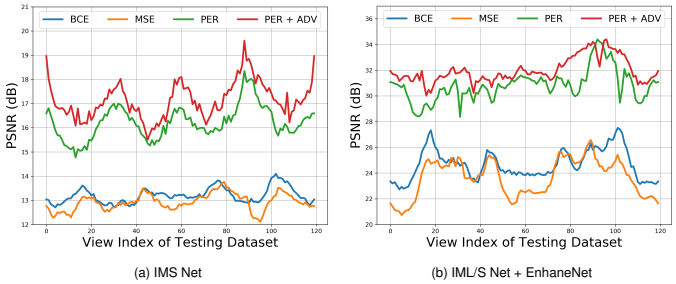


Fig. 12: Qualitative comparison of IM prediction from IMS Net and final reconstruction from IML/S Net + EnhanceNet using different configurations of loss functions on the Chameleon testing dataset.

For training the IML Net end-to-end with the RecNN, the input is the C-Image while the output is the predicted FR-image from RecNN. For training the IMS Net, the input is the view parameters while the output is the learned IM derived from the trained IML Net. The training and validation partition ratio is 9:1. Each volumetric dataset also generates a testing dataset consisting of 120 views, forming an exploratory trajectory to simulate real user exploration. The FR-Image is rendered with a resolution of 512×512 using the Visualization Toolkit (VTK) [30] with a customized timestamp to measure the rendering time for each pixel as detailed in the Appendix. The interpolation is set to trilinear. The sample distance is 0.02. The CPU thread is set as a single thread to faithfully measure the time duration when rendering individual pixels in sequence.

4.2 Experimental Setup

The experiments are designed to investigate the interactive volume visualization using RecNN. We are going to evaluate both the quality and performance using recent volume visualization leveraging RecNN and the proposed neural rendering pipeline, IML/S Net + RecNN. The input is a sequence of unseen views, and the output is the FR-image. We use PSNR as the main metric to evaluate rendering quality. The computing platform is a desktop featuring an Intel(R) Core(TM) i7-7700K CPU with 8 threads running at 4.20GHz, paired with 16GB of DDR4 DRAM clocked at 3200MHz, and operating on Ubuntu 20.04.4 LTS. The final visualization image has a resolution of 512×512 (262144 total pixels). The C-Image resolution is set as 256×256 (65535 total pixels).

4.3 RecNN Selection and Configuration

We select two state-of-the-art RecNNs used for interactive volume visualization, EnhanceNet and FoVolNet, to showcase how our method can help to further improve their rendering latencies. For the super-resolution RecNN using EnhanceNet, we also use $4\times$ super-resolution setting, the same as the experiment in its paper. Since our evaluation only focuses on one-shot image generation, for a fair comparison, we remove the recurrent connections of the FoVolNet and only keep its W-Net structure for static image prediction. Fixed foveated rendering, where the high-resolution region is static and fixed in the center of the screen, is used in the experiment rather than dynamic foveated rendering. The foveal pattern FP is generated by the following methods:

$$FP_{ij} = \begin{cases} 1, & \text{if } P_{ij} > E_{ij} \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

where P_{ij} is a random pattern generated by blue noise, which doesn't generate unevenly distributed samples from low-frequency energy spikes in the spatial domain [3]. The E_{ij} is a modulation surface to control the size of the foveated area through σ:

$$E_{ij} = e^{-0.5(f_x^2+f_y^2)\sigma} \tag{14}$$

where f_x and f_y are the center coordinates of the foveal pattern.

4.4 Training

The proposed neural networks 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 0.001 for training both IML and IMS Nets. A validation dataset is selected to detect overfitting on the training dataset and terminate the network parameters from updating through the early stop mechanism. The patience of the early stop is set as 20 epochs for IML Net and 1000 for IMS Net. The IML Net can be trained standalone or trained end-to-end with RecNN.

4.5 Ablation Study

In this section, we investigate how the configuration of the networks and hyperparameters would influence the model's performance.

4.5.1 Network Configuration

Encoder and Decoder of IML: The encoder of the IML Net is critical to learning an accurate IM for it provides a comprehensive guess of the IM. Its U-Net architecture is capable of distinguishing fine-grained details through its skip connections and recommending candidates of important pixels. Compared with the encoder that serves as an information filter, the decoder serves as an information reconstruction that reconstructs the PR-C-Image to a full C-Image. We investigate how the network configuration of the IML encoder and decoder affects the performance of the proposed rendering pipeline. Since the decoder of the IML Net mirrors the encoder's configuration, we modify both together by adjusting the depth of their U-Nets. We fixed the architecture of IMS Net and only adjusted the IML Net. As the depth of the encoder and decoder increases, its reconstruction quality increases sublinearly (as shown in Fig. 9b and Fig. 9c) while the number of parameters of the network increases superlinearly (as shown in Fig. 9a). While a deeper IML Net can provide better reconstruction quality, its larger size slows down inference, leading to higher rendering latency. We select the depth of both the encoder and decoder as 5 for the balanced performance between rendering quality and inferencing latency.

Generator of IMS: The generator of IMS Net takes a novel view and generates the corresponding IM. The accuracy of the generated IM determines the quality of the input to the downstream IML decoder for

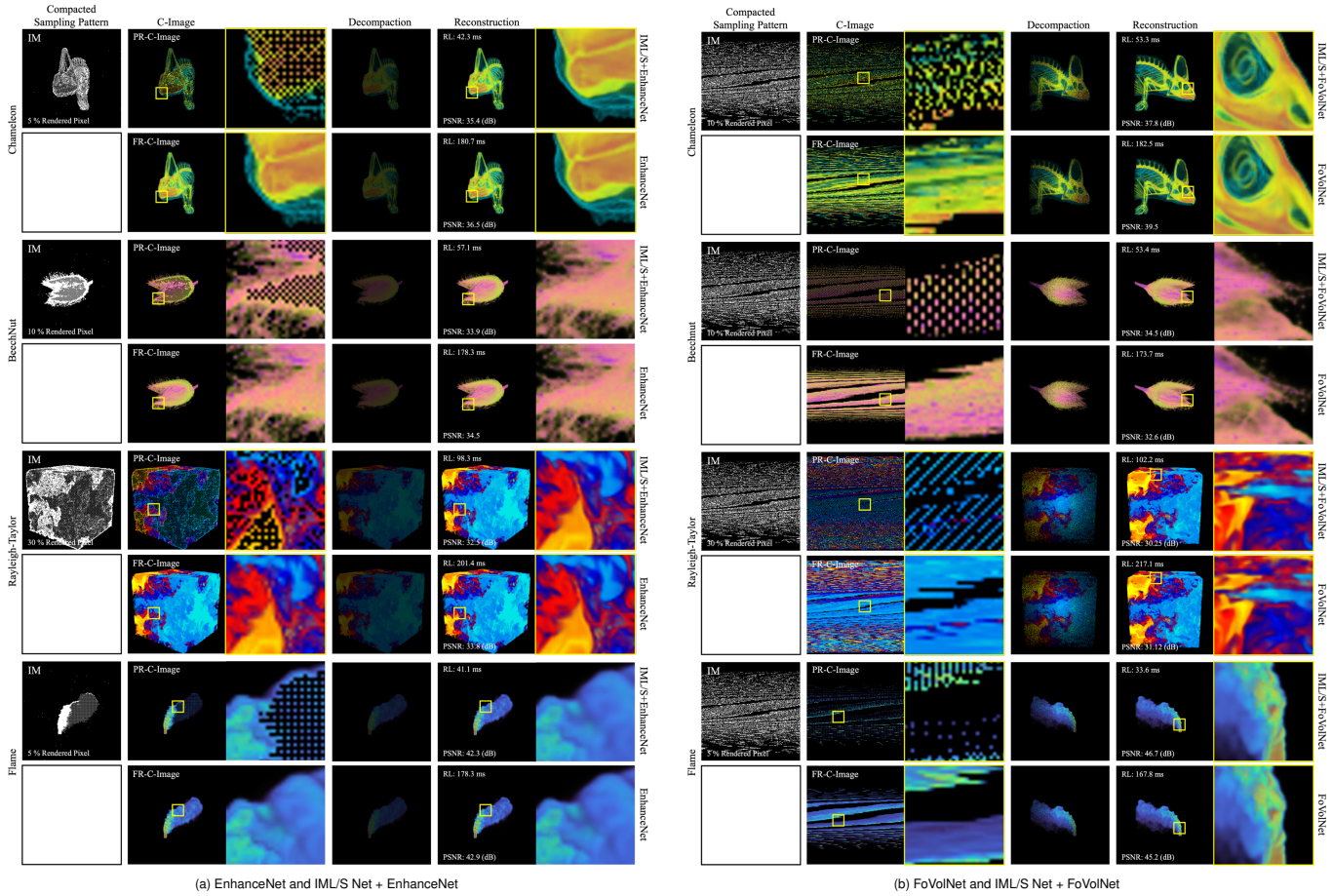


Fig. 13: Visual comparison of reconstruction quality.

reconstruction. We evaluate the reconstruction capability of IMS Net by using different numbers of transposed convolutional layers, which is also the depth of the generator. We fixed the architecture of IML Net and only adjusted the IMS Net. As the number of transposed convolutional layers increases, similarly to adjusting the depth of IML Net, its reconstruction quality increases sublinearly (as shown in Fig. 10b and Fig. 10c) while the number of parameters of the network increases superlinearly (as shown in Fig. 10a). We also observed that adjusting the number of transposed convolutional layers in IMS Net has a greater impact on the reconstruction quality than the depth of the U-Net in IML Net. We select the layer size as 8 for IMS Net to balance rendering quality and inferencing latency.

4.5.2 Hyperparameter Tuning

Mean Value for Rejection Sampling: The hyperparameter of the mean value plays an important role in determining the percentage of pixels in the C-Image as important pixels. The IML Net trained on a specific mean value, after the rejection sampling, will converge to have the percentage of important pixels of C-Image approximately equal to the $mean \times 10$. For a given reconstruction quality tolerance ε, we can find the optimal rendering percentage (ORP) through a binary search on the mean value within the range $[0.01, 1]$. The ε is set as 1 dB in our experiment to search for the optimal mean value and ORP. Using the ORP for IML/S Net will give the RecNN a free improvement on the rendering latency without noticeable quality degradation.

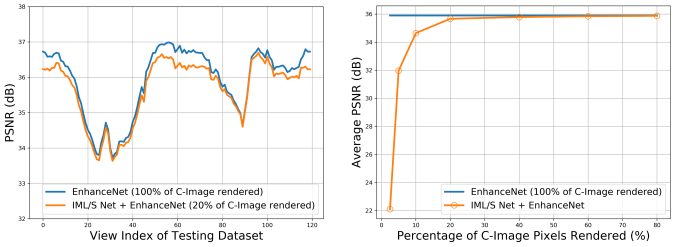
Loss functions: For training the IMS Net for the best results, we tried multiple image-based loss functions, including Binary Cross Entropy (BCE) loss, Mean Squared Error (MSE) loss, Perceptual Loss (PER) loss, and Adversarial (ADV) loss. We found that combining PER and ADV loss gives the best result, as shown in Fig. 11. BCE and MSE are not capable of learning an accurate IM. The PER loss is much better at accurately predicting the correct shape of the IM. However, PER loss struggles to predict the correct details of the IM. Using a loss that combines both the PER (weight = 1) and ADV (weight = 0.01) will give the closest result to the ground truth. The quantitative evaluation on the accuracy of the IM prediction from IMS Net and reconstruction from IML/S Net + EnhanceNet using various loss configurations is listed in Fig. 12.

4.6 Results

4.6.1 Quality Evaluation

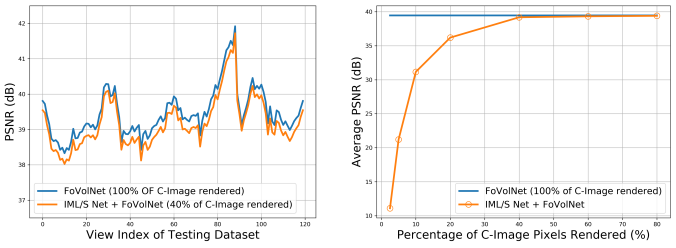
We compare the reconstruction quality between the visualization pipelines using RecNN alone and our proposed IML/S Net + RecNN.

Super-resolution: Fig. 13a shows the visual comparison between the EnhanceNet and our IML/S Net + EnhanceNet of all 4 volumetric datasets. For each dataset, integrating our networks with EnhanceNet delivers comparable rendering quality to using EnhanceNet alone, but with significantly reduced rendering latency (2 to 4 times faster). In the PR-C-Images, the importance mask is successfully learned to extract a subset of the C-Image as important pixels. From the IM, we can observe that image areas with more dynamic content, like the edges of the object and the boundaries between distinct regions, were assigned more important pixels by our IML Net. This observation demonstrates that our network is capable of distinguishing complex regions from simpler ones and automatically sampling more important pixels in the complex region to achieve optimal reconstruction quality. The zoomed-in view with a smaller background area requires a higher percentage of important pixels to be rendered in order to achieve a quality similar to the zoomed-out view. Complex datasets like Rayleigh-Taylor also needs more important pixels compared to simple datasets like the Flame. Fig. 14 shows the quantitative PSNR metric when rendering



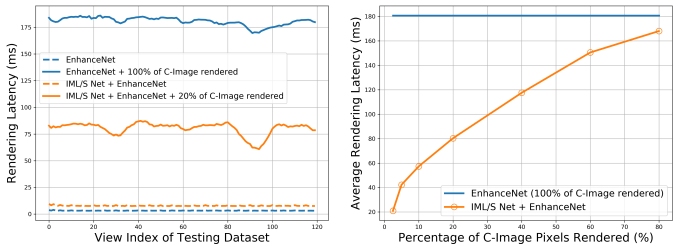
(a) PSNR of each view (b) Average PSNR

Fig. 14: Quantitative comparison of reconstruction quality between EnhanceNet and IML/S Net + EnhanceNet on the Chameleon dataset. (a) demonstrates the reconstruction quality from each view in the testing dataset using 20% of C-Image pixels rendered. (b) shows the trend of average reconstruction quality while increasing the percentage of C-Image rendered.



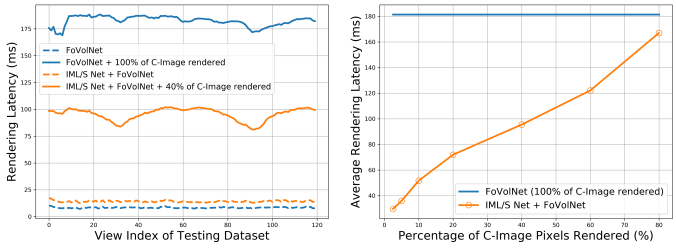
(a) PSNR of each view (b) Average PSNR

Fig. 15: Quantitative comparison of reconstruction quality between FoVolNet and IML/S Net + FoVolNet on the Chameleon dataset. (a) demonstrates the reconstruction quality from each view in the testing dataset using 40% of C-Image pixels rendered. (b) shows the trend of average reconstruction quality while increasing the percentage of C-Image rendered.



(a) Rendering latency for each view (b) Average rendering latency

Fig. 16: Quantitative comparison of rendering latency between EnhanceNet and IML/S Net + EnhanceNet on the Chameleon dataset. (a) demonstrates the rendering latency from each view in the testing dataset using 20% of C-Image pixels rendered. (b) shows the trend of average rendering latency while increasing the percentage of C-Image rendered.



(a) Rendering latency for each view (b) Average rendering latency

Fig. 17: Quantitative comparison of rendering latency between FoVolNet and IML/S Net + FoVolNet on the Chameleon dataset. (a) demonstrates the rendering latency from each view in the testing dataset using 40% of C-Image pixels rendered. (b) shows the trend of average rendering latency while increasing the percentage of C-Image rendered.

all the views in the Chameleon testing dataset. Fig. 14a shows our method gives similar PSNR when only using 20% of the C-Image pixels rendered. We observe from Fig. 14b that as the percentage of important pixels increases, our method infinitely approaches the quality of the standalone RecNN results, and this approximation converges quickly as the percentage of important pixels in the C-Image increases.

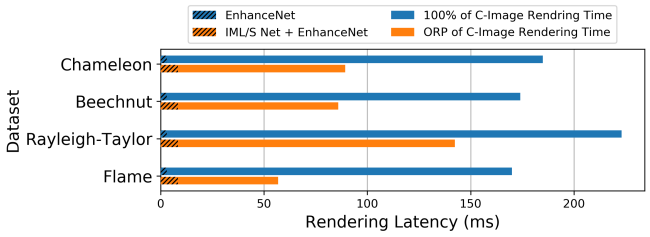
Foveated Rendering: Fig. 13b shows the visual comparison between the FoVolNet and our IML/S Net + FoVolNet of all 4 volumetric datasets. We can observe similar results from the super-resolution using RecNN. Although the C-Images compacted from the foveal pattern does not preserve human-readable features like the ones compacted from the downsampling pattern, the IML Net can still find the important pixel from it, and the IMS Net can still predict the correct IM successfully. Fig. 15a shows our method gives similar PSNR when only using 40% of the C-Image pixels rendered. A similar convergence in quality can be observed as the percentage of important pixels in the C-Image increases from Fig. 15b. We can also observe that the convergence in the foveated rendering of the same Chameleon testing dataset is slower compared to super-resolution rendering. This is due to the foveal pattern, which limits the sampling to the focal area, restricting the possible locations the IML Net can choose from.

4.6.2 Performance Evaluation

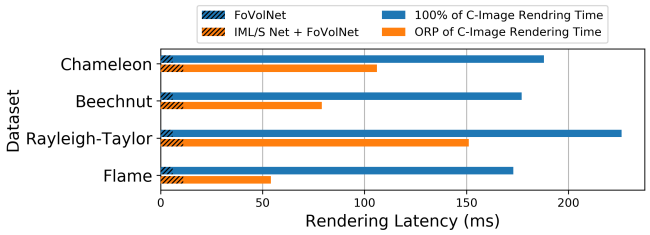
We compare the overall rendering latency between traditional RecNN rendering 100% of the C-Image and our proposed IML/S Net together with the RecNN rendering a portion of the C-Image. In this section, we present a detailed analysis of rendering performance using the Chameleon dataset as an example. A joint evaluation of both rendering quality and performance across all testing datasets will be provided in the next section.

Super-resolution: As we discussed before, the overall rendering latency is the sum of the time to partially render the input image to RecNN and the RecNN inferencing to reconstruct the FR-Image. Fig. 16a presents the quantitative measurement of time usage at each stage. We observe that the inferencing time for both methods is minimal, though the overhead introduced by the IML/S Net slightly increases the total inferencing time compared to using the standalone RecNN. Using the IML/S Net significantly reduces the time required to generate the partially rendered image, as fewer pixels undergo the costly rendering computations. Fig. 16b demonstrates that the growth of average input latency is close to linear with respect to the percentage of important pixels in C-Image.

Foveated Rendering: Fig. 17a presents the quantitative measurement of time usage at each stage for foveated rendering. Our IML/S Net demonstrates similar performance in foveated rendering RecNN as it does in super-resolution RecNN. Since FoVolNet is more complex than EnhanceNet, the inference time for both methods is higher compared to their counterparts in the super-resolution framework. Fig. 17b also demonstrates a linear relationship between the percentage of important pixels in C-Image and the average input latency. Considering both quality and performance results on the Chameleon dataset, we can conclude that, as the percentage of important pixels increases, the quality of reconstruction is more sensitive in super-resolution compared to foveated rendering. This suggests that, for the same reconstruction quality, super-resolution benefits more from our IML/S Net in terms of reducing rendering latency compared to foveated rendering. This observation is related to the nature of the distribution of the Chameleon dataset in the visualization image space. The contents of interest across views are more evenly distributed across the whole image domain rather than only distributed in the center. As a result, the downsampling pattern of super-resolution RecNN covers such a distribution better than the foveal pattern of foveated rendering RecNN.



(a) Super-resolution RecNN



(b) Foveated rendering RecNN

Fig. 18: Average rendering latency using ORP. The shaded regions are the time used to inference the network. The solid regions are the time used by the rendering algorithm to compute the pixel values in C-Image.

Table 2: Detailed ORP for free rendering latency improvement.

Dataset	ORP		Avg Rendering Latency (ms)		Avg Speed Up	
	Sup-Res ↓	Foveated ↓	Sup-Res ↓	Foveated ↓	Sup-Res ↑	Foveated ↑
Chameleon	20%	40%	89.4	106.1	2.07×	1.77×
Beechnut	18%	17%	86.0	79.1	2.02×	2.24×
Rayleigh-Taylor	40%	42%	142.4	151.1	1.75×	1.49×
Flame	11%	10%	56.9	54.1	2.98×	3.20×

4.6.3 Free Rendering Latency Improvement

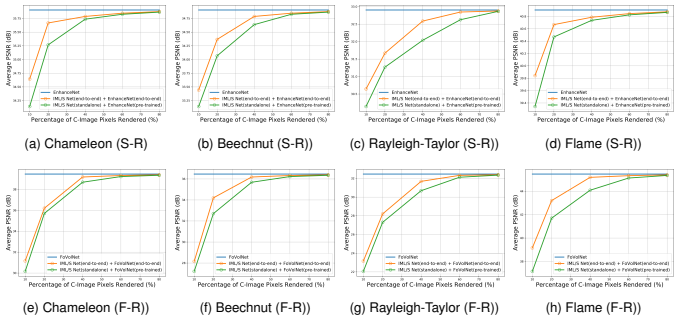
In this section, we further investigate the data-dependent characteristics of various testing datasets on rendering quality and performance using our method. We will also qualitatively evaluate how much rendering latency improvement can be achieved by our method without sacrificing perceivable rendering quality. We find the optimal rendering percentage (ORP) for our IML Net to make IML/S Net reconstruct perspective rendering within 1 dB PSNR difference (ε) to the rendering from the standalone RecNN. Fig. 18 shows the average rendering latency using ORP for both EnhanceNet and FoVolNet. On average, our method can improve the rendering latency for each volumetric dataset with similar rendering quality. The detailed results are listed in Tab. 2. We can see that for different datasets, ORP varies to achieve close reconstruction quality to the original RecNN. Rayleigh-Taylor requires the highest ORP to maintain quality due to its dynamic nature across the whole 3D space, which leads to the largest content area in the final rendering compared to other datasets. Flame is simpler and more compact in 3D space than others, resulting in the lowest ORP. Compared to super-resolution, foveated rendering favors datasets where the rendered object is primarily centered in the image, such as Beechnut and Flame, leading to a lower ORP.

4.6.4 Feasibility Study

In this section, we assess the practicality and viability of the proposed method. To take advantage of the rendering latency improvement provided by the proposed method, the overhead is the extra training step for the IML and IMS networks. The IMS Net is a simple network and its training dataset (view and IM) is also small, so its training process is relatively quick. Due to the complex structure of the RecNNs and their large training data (full-res images), a significant amount of time is dedicated to the end-to-end training of the IML Net and the downstream RecNN. To enhance the feasibility of our method, the proposed IML Net can be trained independently without connecting to the complicated downstream RecNN. Compared to RecNN, the network structure of IML Net is simpler, and its training data is smaller (low-res C-Image). The loss function for training this standalone IML Net is solely based on the compaction image loss. Once the IML Net is trained, its decoder can directly connect to an off-the-shelf pre-trained RecNN to improve the rendering latency. In this way, hours of training time can be saved. The total epochs and training times for training the IML and IMS Net are listed in Tab. 3. We can observe that training IML standalone will save a significant amount of training time compared to training IML end-to-end with RecNN. Due to the absence of joint optimization across the entire pipeline, using the independently trained IML Net will result in lower reconstruction quality compared to end-to-end training. However, as shown in Fig. 19, the quality degradation is still acceptable, especially when the IML Net is trained using a larger percentage of important pixels on C-Image.

Table 3: Training time and the number of epochs for training the IML and IMS Net. End-to-end trains the IML Net together with the downstream RecNN. Standalone only trains the IML Net.

Dataset	IML Net (epoches/hours ↓)				IMS Net (epoches/hours ↓)	
	EnhanceNet		FoVolNet			
	End-to-end	Standalone	End-to-end	Standalone	EnhanceNet	FoVolNet
Chameleon	83 / 5.1	92 / 2.0	107 / 7.2	117 / 2.5	1744 / 2.9	1687 / 3.1
Beechnut	95 / 6.3	102 / 2.0	43 / 3.3	73 / 1.5	1632 / 3.2	1722 / 3.5
Rayleigh-Taylor	136 / 8.2	141 / 2.9	161 / 10.2	180 / 3.5	2443 / 3.1	2901 / 3.8
Flame	152 / 9.1	120 / 2.5	255 / 15.0	163 / 2.8	1711 / 3.0	1804 / 3.2



(a) Chameleon (S-R) (b) Beechnut (S-R)) (c) Rayleigh-Taylor (S-R)) (d) Flame (S-R))

(e) Chameleon (F-R)) (f) Beechnut (F-R)) (g) Rayleigh-Taylor (F-R)) (h) Flame (F-R))

Fig. 19: Reconstruction quality comparison using RecNN between end-to-end trained IML Net + RecNN and standalone trained IML Net for both super-resolution RNN (S-R) and foveated Rendering RNN (F-R)

4.6.5 Comparison with LAS

We also compare the proposed IML/S rendering pipeline with the Learning Adaptive Sampling (LAS) method [45] on both rendering quality and latency. Since the LAS doesn't support foveated rendering, only the super-resolution RecNN is used for the comparison. In the experiment, we adopt the same EnhanceNet configuration detailed in the LAS paper, consisting of 10 residual blocks, as the downstream RecNN for both importance mask learning networks. The LAS baseline is the paper's open-source implementation[1]. The resolution of the rendering is 512×512. Both methods are tested at three levels of IPR for each testing dataset to ensure a thorough evaluation. All the time measurements are the average of 10 trials. Detailed measurements on rendering quality and latency across all the views of the testing datasets can be found in the Appendix.

Rendering Quality: For rendering quality with super-resolution RecNN, as shown in Tab. 4, LAS and IML/S have similar performance for each IPR. This is because, for both methods, the quality of the input to the RecNN is mainly determined by the IPR, although the important pixels are learned through different importance networks. A higher IPR provides both methods with a broader selection of important pixels, resulting in a more informative partially rendered image for the RecNN to reconstruct from. Consistent with Sec. 4.6.3, we observe that datasets featuring large 3D spaces and complex dynamics, such as

[1] https://github.com/shamanDevel/AdaptiveSampling

Table 4: Rendering and quality evaluation between Learning Adaptive Sampling (LAS) method and the proposed IML/S with RecNN for super-resolution and foveated rendering. Foveated Rendering is not supported (NS) by LAS. The speed up metric under average rendering latency is measured using IML/S compared to LAS.

Dataset	Average Rendering Latency of GPU Ray-caster (ms)	IPR	Average Rendering Latency (ms) ↓					Average Rendering Quality (Avg PSNR (dB) ↑ / Avg SSIM ↑)			
			Super-Resolution			Foveated Rendering		Super-Resolution		Foveated Rendering	
			LAS	IML/S	Speed Up	LAS	IML/S	LAS	IML/S	LAS	IML/S
Chameleon	326.5	5%	111.2	**42.0**	2.65×	NS	39.6	32.63 / 0.932	**32.86 / 0.940**	NS	29.11 / 0.883
		10%	191.4	**60.4**	3.17×	NS	59.3	**34.65 / 0.953**	34.64 / 0.951	NS	31.16 / 0.920
		15%	259.9	**70.9**	3.67×	NS	66.3	34.80 / 0.955	**35.25 / 0.976**	NS	34.21 / 0.951
Beechnut	517.7	5%	118.2	**50.3**	2.35×	NS	48.2	31.70 / 0.915	**32.98 / 0.940**	NS	26.03 / 0.859
		10%	213.6	**60.3**	3.54×	NS	57.5	32.86 / 0.937	**33.44 / 0.944**	NS	28.16 / 0.891
		15%	290.9	**75.0**	3.88×	NS	69.8	33.03 / 0.946	**34.01 / 0.949**	NS	31.05 / 0.908
Rayleigh-Taylor	354.0	10%	152.7	**44.9**	3.40×	NS	39.0	24.19 / 0.816	**30.64 / 0.910**	NS	23.16 / 0.803
		15%	220.5	**62.2**	3.55×	NS	58.9	29.32 / 0.894	**31.25 / 0.917**	NS	26.51 / 0.866
		20%	290.8	**77.2**	3.77×	NS	71.9	**32.45 / 0.942**	31.67 / 0.923	NS	28.20 / 0.889
Flame	438.7	2.5%	77.6	**22.3**	3.48×	NS	19.5	38.24 / **0.966**	**39.03** / 0.965	NS	35.33 / 0.980
		5%	124.3	**31.3**	3.97×	NS	27.1	38.31 / 0.957	**39.31 / 0.976**	NS	36.54 / 0.974
		7.5%	175.4	**43.4**	4.04×	NS	40.6	39.01 / 0.962	**39.57 / 0.985**	NS	38.02 / 0.975

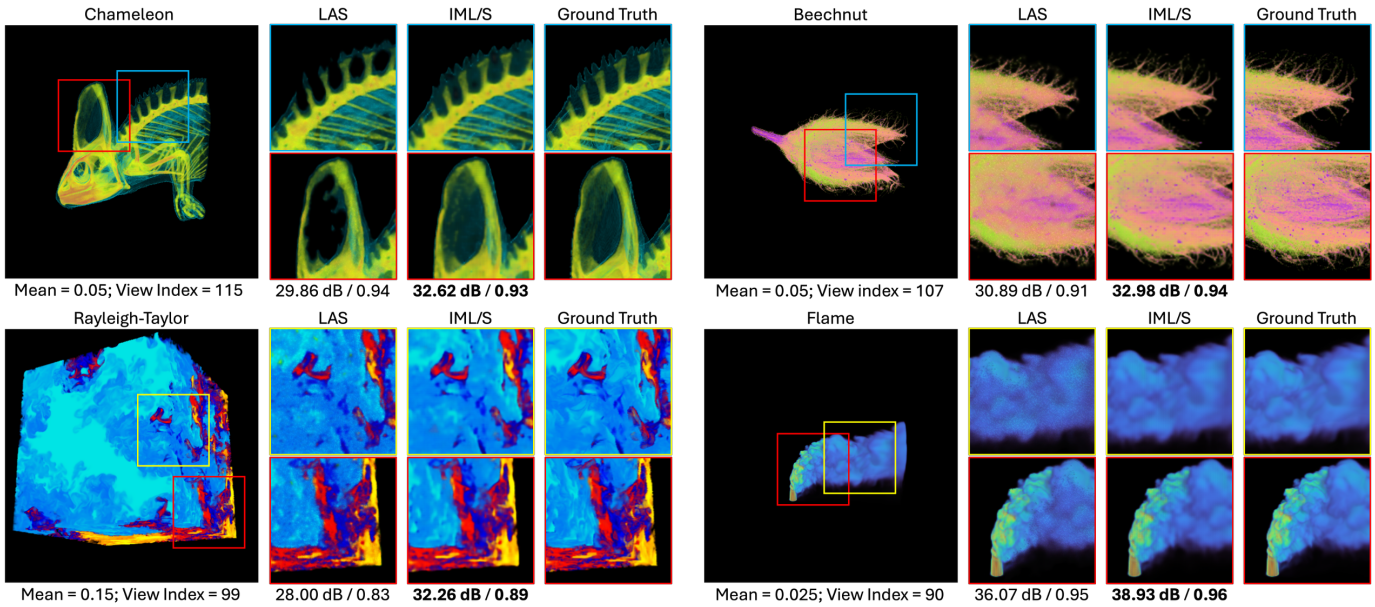


Fig. 20: Quantitative evaluation of rendering quality between LAS and IML/S + EnhanceNet from specific views of testing datasets.

Rayleigh-Taylor, require higher IPR to achieve high rendering quality. For spatially compact and simpler datasets such as Flame, relatively high rendering quality can be achieved with just a small set of important pixels from a lower IPR. A qualitative evaluation of rendering quality is shown in Fig. 20.

Rendering Latency: Tab. 4 also shows the rendering latency between the proposed method and LAS. We also measure the average rendering latency of a GPU Ray-caster[2] accelerated through the CUDA API as a baseline. Detailed time breakdowns of each rendering stage for all the testing datasets are shown in Fig. 21. IML/S is much faster than LAS for the following reasons: 1) The input to LAS is a low-res image (64×64), which needs expensive pixel calculation (blue portion of Fig. 21), while the input to IML/S is just a set of the view parameters, which is known from the user's exploration. 2) Both methods spend time on the three following stages: importance network inference, partial rendering, and RecNet inference. Both methods spend similar time on RecNet inference (red portion of Fig. 21) because they use the same network. Although IML/S spends more time on the important network inference (orange portion of Fig. 21) due to its more complicated network architecture, including both the IMS Net and the decoder of IML Net, it saves much more time during the partial rendering stage (green portion of Fig. 21). This is because IML/S extracts the important pixels from the C-Image, which has a lower resolution (256×256) than the full-res image (512×512), where LAS extracts important pixels from. Therefore, for the same IPR, IML/S results in a much smaller number of important pixels to render. Additionally, the C-Image's resolution can be scaled down as needed for the downstream RecNN.

[2]https://github.com/NVIDIA/cuda-samples/tree/master/Samples/5_Domain_Specific/volumeRender

5 Limitations and Future Work

In this paper, we provide a way to further optimize the already fast volume visualization method leveraging RecNN. We concentrate on the rationale and validation of the proposed method, as well as the evaluation of how much rendering latency can be reduced while maintaining similar rendering quality. We demonstrate that the network can successfully learn the important pixels from the C-Image by jointly considering various inputs and configurations of a volume visualization system.

One limitation of the current design is that the mean value used to determine the percentage of C-Image as important pixels is a predefined hyperparameter. While users can adjust the mean value during inference, our experiments indicate that the network trained with a specific mean value m achieves the best reconstruction quality when the inference mean value is also close to m. Using a mean value significantly higher than m, resulting in a large number of pixels being rendered as important pixels, does not improve reconstruction quality and may even lead to worse results. It would be highly beneficial to develop a method that enables the network to also learn an adaptive mean value,

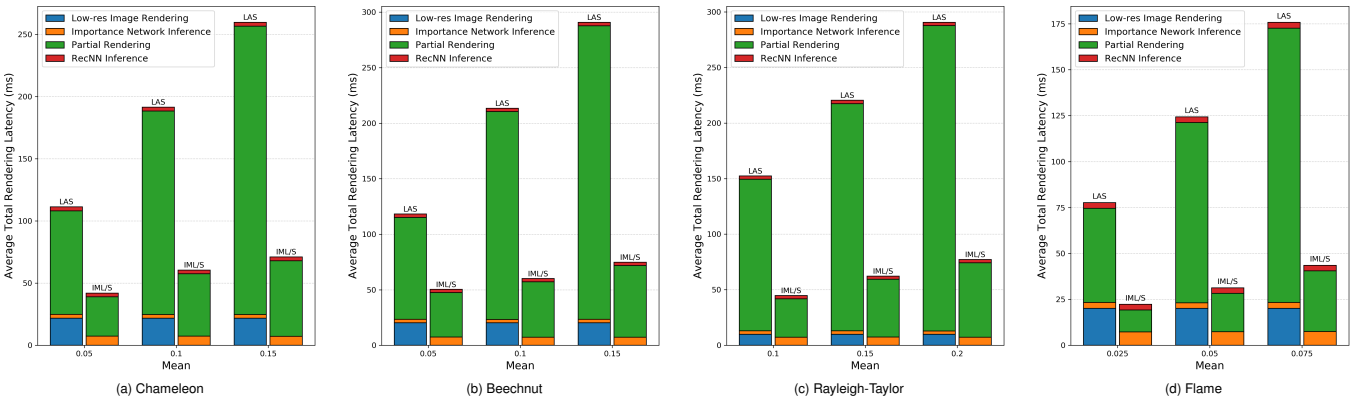


Fig. 21: Time breakdown of average total rendering latency for testing datasets.

resulting in a dynamic percentage of important pixels in the C-Image that adjusts based on the underlying rendering results. This approach can also assist in quickly identifying the ORP for a given reconstruction quality error tolerance. More comprehensive methods of identifying ORP, rather than setting a 1 dB threshold, are also worth exploring.

Various aspects of the pipeline need detailed investigation in the future. In this work, we only investigate the static fixed foveated rendering, it is worth investigating whether dynamic foveal rendering could impact reconstruction quality, rendering latency, and the pattern of the IM learned. IML Net is trained only using pixel-wise loss (MSE) in our work, other loss configurations, like structural similarity-based loss (SSIM), can be introduced to evaluate how the loss function affects the pattern of the learned IM. It is worth investigating which factors influence the distribution of important pixels and how many pixels are needed to render a specific frame. Factors such as image complexity, region size, and color dynamics may all contribute to determining which pixels are selected as important pixels. We also see the potential of utilizing the proposed IML Net to generate IMs with various levels of detail (LOD), enabling the construction of a multi-resolution representation to further optimize rendering performance.

6 Conclusion

In this work, we present IML Net and IMS Net to learn and synthesize an important mask that will help to further improve the already fast volume visualization using a reconstruction neural network. Our work presents the first attempt to directly synthesize an importance mask from a given view for predefined sampling patterns used in various reconstruction neural networks, via a unified interface that leverages the proposed compaction and decompaction layers. We showcase the quantitative improvement of rendering latency using our method on two state-of-the-art reconstruction neural networks, EnhanceNet and FoVolNet, on four distinct volumetric datasets, without noticeable degradation of reconstruction quality. Our proposed IML Net is also flexible to be trained independently from the downstream reconstruction neural network, allowing quick and practical application of our method on top of various off-the-shelf pre-trained reconstruction neural networks.

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 contract DE-AC02-06CH11357, program manager Hal Finkel. The authors would like to thank the anonymous reviewers for their insightful comments.

References

[1] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In D. Precup and Y. W. Teh, eds., *Proceedings of the 34th International Conference on Machine Learning*, vol. 70 of *Proceedings of Machine Learning Research*, pp. 214–223. PMLR, 06–11 Aug 2017. 6

[2] I. B. Barcelos, F. D. C. Belém, L. D. M. João, Z. K. G. D. Patrocínio, A. X. Falcão, and S. J. F. Guimarães. A comprehensive review and new taxonomy on superpixel segmentation. *ACM Comput. Surv.*, 56(8), apr 2024. doi: 10.1145/3652509 3

[3] 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, 7

[4] M. Berger, J. Li, and J. A. Levine. A generative model for volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 25(4):1636–1650, 2019. doi: 10.1109/TVCG.2018.2816059 2, 6

[5] A. Brock, J. Donahue, and K. Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018. 6

[6] J. Cao, Y. Li, K. Zhang, and L. Van Gool. Video super-resolution transformer. *arXiv preprint arXiv:2106.06847*, 2021. 2

[7] M. Cox, N. Bhandari, and M. Shantz. Multi-level texture caching for 3d graphics hardware. *SIGARCH Comput. Archit. News*, 26(3):86–97, apr 1998. doi: 10.1145/279361.279372 1

[8] E. Gobbetti and F. Marton. Far voxels: A multiresolution framework for interactive rendering of huge complex 3d models on commodity graphics platforms. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, p. 878–885. Association for Computing Machinery, New York, NY, USA, 2005. doi: 10.1145/1186822.1073277 1

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

[10] V. K. Ha, J. Ren, X. Xu, S. Zhao, G. Xie, and V. M. Vargas. Deep learning based single image super-resolution: A survey. In J. Ren, A. Hussain, J. Zheng, C.-L. Liu, B. Luo, H. Zhao, and X. Zhao, eds., *Advances in Brain Inspired Cognitive Systems*, pp. 106–119. Springer International Publishing, Cham, 2018. 1

[11] W. He, J. Wang, H. Guo, K. Wang, H. Shen, M. Raj, Y. G. Nashed, and T. Peterka. Insitunet: Deep image synthesis for parameter space exploration of ensemble simulations. *IEEE Transactions on Visualization & Computer Graphics*, 26(01):23–33, jan 2020. doi: 10.1109/TVCG.2019.2934312 2

[12] P. Heckbert and M. Garland. Multiresolution modeling for fast rendering. In *Graphics Interface*, pp. 43–43. Canadian Information Processing Society, 1994. 1

[13] F. Hong, J. Zhang, and X. Yuan. Access pattern learning with long short-term memory for parallel particle tracing. In *2018 IEEE Pacific Visualization Symposium (PacificVis)*, pp. 76–85, 2018. doi: 10.1109/PacificVis.2018.00018 1

[14] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 5

[15] J. Jiang, C. Wang, X. Liu, and J. Ma. Deep learning-based face super-resolution: A survey. *ACM Comput. Surv.*, 55(1), nov 2021. doi: 10.1145/3485132 1

[16] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In B. Leibe, J. Matas, N. Sebe, and M. Welling, eds., *Computer Vision – ECCV 2016*, pp. 694–711. Springer

International Publishing, Cham, 2016. 6

[17] A. S. Kaplanyan, A. Sochenov, T. Leimkühler, M. Okunev, T. Goodall, and G. Rufo. Deepfovea: neural reconstruction for foveated rendering and video compression using learned statistics of natural videos. *ACM Trans. Graph.*, 38(6), nov 2019. doi: 10.1145/3355089.3356557 2

[18] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(12):4217–4228, 2021. doi: 10.1109/TPAMI.2020.2970919 5

[19] H. Liu, Z. Ruan, P. Zhao, C. Dong, F. Shang, Y. Liu, L. Yang, and R. Timofte. Video super-resolution based on deep learning: a comprehensive survey. *Artificial Intelligence Review*, 55(8):5981–6035, 2022. 1

[20] H. Liu, Z. Ruan, P. Zhao, C. Dong, F. Shang, Y. Liu, L. Yang, and R. Timofte. Video super-resolution based on deep learning: a comprehensive survey. *Artif. Intell. Rev.*, 55(8):5981–6035, Dec. 2022. doi: 10.1007/s10462-022-10147-y 2

[21] G. Lochmann, B. Reinert, A. Buchacher, and T. Ritschel. Real-time novel-view synthesis for volume rendering using a piecewise-analytic representation. In *VMV'16 Proceedings of the Conference on Vision, Modeling and Visualization*, vol. 2016. Eurographics, 2016. 2

[22] Y. Lu, K. Jiang, J. A. Levine, and M. Berger. Compressive neural representations of volumetric scalar fields. In *Computer Graphics Forum*, vol. 40, pp. 135–146. Wiley Online Library, 2021. 2

[23] S. Mittal. A survey of recent prefetching techniques for processor caches. *ACM Comput. Surv.*, 49(2), aug 2016. doi: 10.1145/2907071 1

[24] S. Parker, M. Parker, Y. Livnat, P.-P. Sloan, C. Hansen, and P. Shirley. Interactive ray tracing for volume visualization. In *ACM SIGGRAPH 2005 Courses*, SIGGRAPH '05, p. 15–es. Association for Computing Machinery, New York, NY, USA, 2005. doi: 10.1145/1198555.1198754 3

[25] A. Radford. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. 5

[26] H. Ray, H. Pfister, D. Silver, and T. Cook. Ray casting architectures for volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 5(3):210–223, 1999. doi: 10.1109/2945.795213 3

[27] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, eds., *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pp. 234–241. Springer International Publishing, Cham, 2015. 4

[28] M. S. M. Sajjadi, B. Schölkopf, and M. Hirsch. Enhancenet: Single image super-resolution through automated texture synthesis. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 4501–4510, 2017. doi: 10.1109/ICCV.2017.481 2

[29] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, p. 2234–2242. Curran Associates Inc., Red Hook, NY, USA, 2016. 6

[30] W. Schroeder, L. Avila, and W. Hoffman. Visualizing with vtk: a tutorial. *IEEE Computer Graphics and Applications*, 20(5):20–27, 2000. doi: 10.1109/38.865875 7

[31] G. Sharma, F. Jurie, and C. Schmid. Discriminative spatial saliency for image classification. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3506–3513, 2012. doi: 10.1109/CVPR.2012.6248093 3

[32] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 6

[33] 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

[34] Y. Strümpler, J. Postels, R. Yang, L. V. Gool, and F. Tombari. Implicit neural representations for image compression. In S. Avidan, G. Brostow, M. Cissé, G. M. Farinella, and T. Hassner, eds., *Computer Vision – ECCV 2022*, pp. 74–91. Springer Nature Switzerland, Cham, 2022. 2

[35] J. Sun, D. Lenz, H. Yu, and T. Peterka. Scalable volume visualization for big scientific data modeled by functional approximation. In *2023 IEEE International Conference on Big Data (BigData)*, pp. 905–914, 2023. doi: 10.1109/BigData59044.2023.10386434 1

[36] 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 Symposium on Large-Scale Data Analysis and Visualization (LDAV)*, 2024. 1

[37] J. Sun, D. Lenz, H. Yu, and T. Peterka. Mfa-dvr: direct volume rendering of mfa models. *Journal of Visualization*, 27(1):109–126, 2024. 3

[38] J. Sun, D. Lenz, H. Yu, and T. Peterka. F-hash: Feature-based hash design for time-varying volume visualization via multi-resolution tesseract encoding. *arXiv preprint arXiv:2507.03836*, 2025. 2

[39] 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 1

[40] V. Sundstedt and V. Garro. A systematic review of visualization techniques and analysis tools for eye-tracking in 3d environments. *Frontiers in neuroergonomics*, 3:910019, 2022. 2

[41] 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

[42] 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 2

[43] 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*, 27(6):3064–3078, 2021. doi: 10.1109/TVCG.2019.2956697 2

[44] S. Weiss, P. Hermüller, and R. Westermann. Fast neural representations for direct volume rendering. In *Computer Graphics Forum*, vol. 41, pp. 196–211. Wiley Online Library, 2022. 2

[45] S. Weiss, M. IşIk, J. Thies, and R. Westermann. Learning adaptive sampling and reconstruction for volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 28(7):2654–2667, 2022. doi: 10.1109/TVCG.2020.3039340 2, 4, 10

[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*, pp. 1–14, 2023. doi: 10.1109/TVCG.2023.3293121 2

[47] 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

[48] H. Yu, C. Wang, and K.-L. Ma. Massively parallel volume rendering using 2–3 swap image compositing. In *SC '08: Proceedings of the 2008 ACM/IEEE Conference on Supercomputing*, pp. 1–11, 2008. doi: 10.1109/SC.2008.5219060 1

[49] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. N. Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 6

[50] L. Zhang, J. Sun, C. Peterson, B. Sharif, and H. Yu. Exploring eye tracking data on source code via dual space analysis. In *2019 Working Conference on Software Visualization (VISSOFT)*, pp. 67–77, 2019. doi: 10.1109/VISSOFT.2019.00016 2

[51] Y. Zhang and K.-L. Ma. Fast global illumination for interactive volume visualization. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '13, p. 55–62. Association for Computing Machinery, New York, NY, USA, 2013. doi: 10.1145/2448196.2448205 3

Jianxin Sun is a research assistant professor in the School of Computing at the University of Nebraska-Lincoln. He received a Ph.D. degree in Computer Science from the University of Nebraska-Lincoln and an MS degree in Electrical and Computer Engineering from Purdue University. His research concentrates on AI-driven scientific data modeling, analysis, and visualization through high-performance computing.

David Lenz is an assistant computer scientist at Argonne National Laboratory. He received his Ph.D. in Mathematics from University of California San Diego. His research focuses on scientific data visualization and analysis, with an emphasis on functional approximation, implicit neural representations, and lossy compression.

Hongfeng Yu is a Professor in the School of Computing and Director of Holland Computing Center at the University of Nebraska–Lincoln. He earned his Ph.D. in Computer Science from the University of California, Davis. His research focuses on developing theories and technologies for scalable data analysis and visualization, advancing discoveries across scientific and engineering fields through interdisciplinary collaborations.

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 150 peer-reviewed articles and papers since earning his Ph.D. in computer science from the University of Illinois at Chicago in 2007.