Multiscale Spatiotemporal Super Resolution for 3D Turbulent Flow

Skylar Wurster,[1]* Han-Wei Shen,[1] Thomas Peterka,[2] and Hanqi Guo[2]

***Abstract*—**

I. INTRODUCTION

Scientists across many fields generate high resolution vector field data from large-scale fluid simulations of various phenomena. For example, Rayleigh-Bénard convection is a phenomenon that occurs when a fluid is heated from below which causes it to rise and be replaced with cooler fluid pulled down by gravity. This phenomenon is often simulated at high resolutions and studied due to its experimental accessibility. With supercomputers reaching exascale computing capabilities, scientists are afforded an increasingly large amount of compute power to generate more fine results for simulations like these quicker. However, they cannot afford to save the data at the resolution they are generated due to an I/O bottleneck, which forces scientists to discard some of the data. Spatial and/or temporal information are lost in this step that may be critical when analyzing the saved simulation results. This paper focuses on super-resolving high resolution (HR) turbulent flow vector field data from a low resolution (LR) input in both the space and time dimensions, allowing the scientist to recover the data that was thrown away.

Due to the I/O bottleneck, data reduction and reconstruction for scientific data is a popular research area. Recently, machine learning (ML) approaches for spatial and temporal super resolution of fluid flow have shown promising results. Current state-of-the-art ML approaches perform better than linear interpolation (LERP) for spatial, temporal, and spatiotemporal super resolution. However, there are a number of limitations to the current methods. First, these approaches use a static spatial upscaling ratio such as 4x or 8x. This limits the resolution that can be saved by the scientist to this one setting. A similar trend is found in the temporal domain, where networks are only trained for creating a predetermined k timesteps between two keyframes. Second, these ML approaches do not guarantee any preserved quality metric, such as PSNR or maximum relative error, for data inferred after training. Scientists may be more skeptical of ML methods when there is no guarantee of reconstruction quality. Third, visualizing streamlines and pathlines are crucial to understanding flow data, but current ML models do not explicitly incorporate any method to accurately reconstruct these features.

We aim to solve these three limitations with FlowSTSR, a CNN-based spatiotemporal super resolution model. To give scientists a choice in which data is thrown away, we design a hierarchy of networks for spatiotemporal super resolution. Each network in the hierarchy is responsible for 2x super resolution at a specific downscaling ratio. Using these models sequentially enables higher scale factor super resolution capabilities, such as 4x and 8x. This gives flexibility to the scientist to save data at whichever downsampling rate they would like for both space and time dimensions independently. For instance, the scientist may be more interested in the temporal evolution of the data than the spatial features. In this case, they could opt to use a small temporal downsampling rate like 2x, but use a large spatial downsampling rate like 8x. Next, we utilize this multi-scale architecture to super resolve multi-resolution data frames, which allows us to dynamically downsample the raw data in-situ such that FlowSTSR can reconstruct lost data with a maximum error criteria. This process maximally downsamples the raw data using an octree data structure, which saves different spatial regions at different resolutions, but ensures that when FlowSTSR super resolves the frame, some quality criteria is met. Lastly, we implement a novel streamline loss function that ensures the model is trained to reconstruct accurate streamlines.

II. RELATED WORK

A. Image/video super resolution

B. Flow super resolution

C. Deep learning for scientific visualization

III. METHOD

We approach multi-scale spatiotemporal super resolution as two distinct problems - spatial super resolution and temporal super resolution.

We model the multi-scale spatial super resolution with a set of functions $f = \{f_1, f_2, ...\}$ such that each f_i performs 2x super resolution, and given a full resolution size N, f_i will super resolve low resolution $\frac{N}{2^{|f|-i+1}}$ to high resolution $\frac{N}{2^{|f|-i}}$. Therefore, functions with smaller indices are responsible for performing super resolution on coarser inputs, such as 64^3 or 128^3, up to $f_{|f|}$, which outputs the full resolution. Since each function performs 2x super resolution, they can be performed sequentially. For example, $f_2(f_1(x))$ would perform 4x super resolution on x. We seek to estimate this set of functions with a hierarchy of generative adversarial neural networks (GANs) G. That is, we approximate each function in f using $|f|$ models. This approach is similar to works such as LapSRN, LapGAN, and SinGAN, where a hierarchy of models are used for various computer vision

[1]Computer Science and Engineering Department, 2015 Neil Ave, Columbus OH 43210, United States

[2]Argonne National Laboratory, 9700 S Cass Ave, Lemont, IL 60439

* wurster.18 at osu.edu

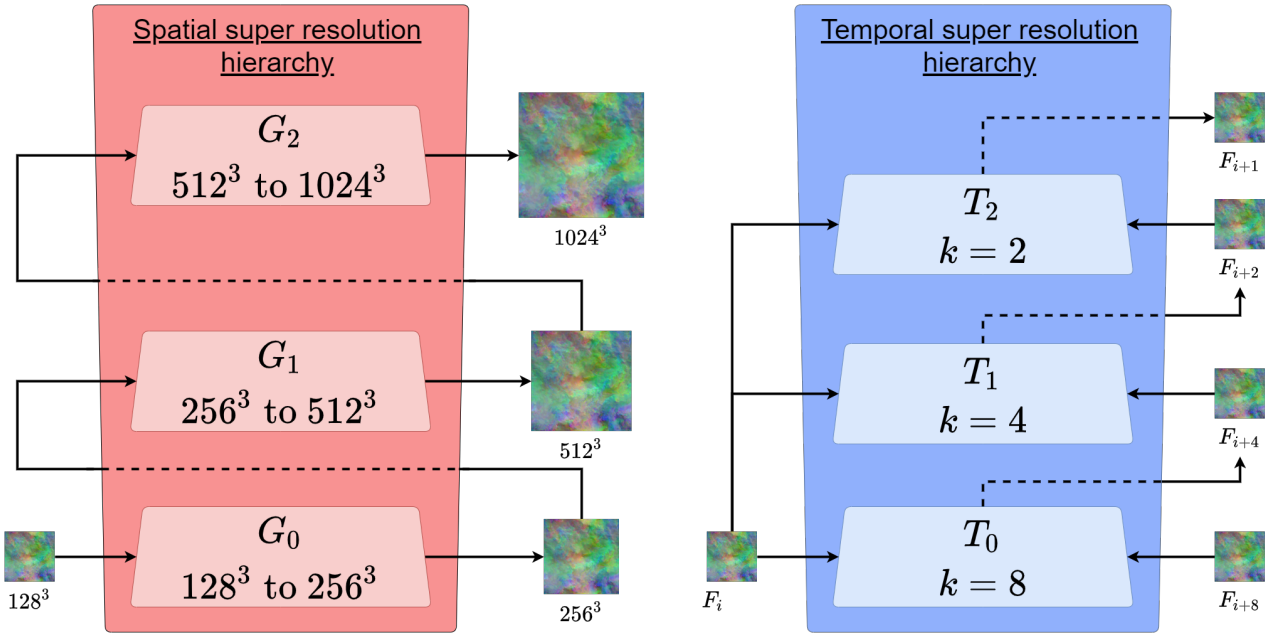

Fig. 1. Example of a FlowSTSR model with 3 networks in the spatial super resolution hierarchy and 3 networks in the temporal super resolution hierarchy. The spatial super resolution networks take an input of low resolution and super resolve a frame of 2x the size in each spacial dimension. The temporal super resolution networks take two frames of input that are k frames apart, and interpolate the middle frame.

tasks such as super resolution, image inpainting, and image synthesis.

For multi-scale temporal super resolution, we use a similar approach in the temporal domain. We model the problem as a set of functions $t = \{t_1, t_2, ...\}$ such that t_i interpolates the middle frame between timestep j and $j+2^{|t|-i+1}$. Therefore, functions with smaller indices interpolate larger timestep gaps, such as 16 or 8, while the largest index interpolates between timestep j and $j+2$. We estimate each function in t using a hierarchy of networks T.

Together, the spatial super resolution hierarchy $G = \{G_0, G_1, ...\}$ and the temporal super resolution hierarchy $T = \{T_0, T_1, ...\}$ allow arbitrary spatiotemporal super resolution from any 2^a spatial downscaling rate and 2^b temporal downscaling rate, for $0 \leq a \leq |G|$ and $0 \leq b \leq |T|$. Figure 1 shows an example of how frames are spatially super-resolved or interpolated using G and T, each a hierarchy of 3 networks, allowing a maximum of 8x super resolution in the space or time dimensions.

A. Spatial Super Resolution Hierarchy

The spatial super resolution hierarchy G is composed of $|G|$ generative adversarial networks, each trained independently to perform 2x super resolution on a specific downscaling ratio, and together enabling $2^{|G|}$x spatial super resolution.

1) Spatial super resolution generator: In our method, we use a super resolution generator architecture based on ESRGAN's generator, a state-of-the-art GAN for super resolving single images. Our network is a 3D adaptation of ESRGAN that takes a low resolution vector field F^{LR} and outputs a synthesized high resolution vector field F^{HR} that is twice as big in each spatial dimension. The full architecture is depicted in Figure 2. We use this architecture for 3 primary reasons: (1) learning is done in the low dimensional feature space, i.e. pre-upsample learning, (2) the use of voxel shuffling, and (3) the deep residual and use of Residual-in-Residual Dense Blocks (RRDB).

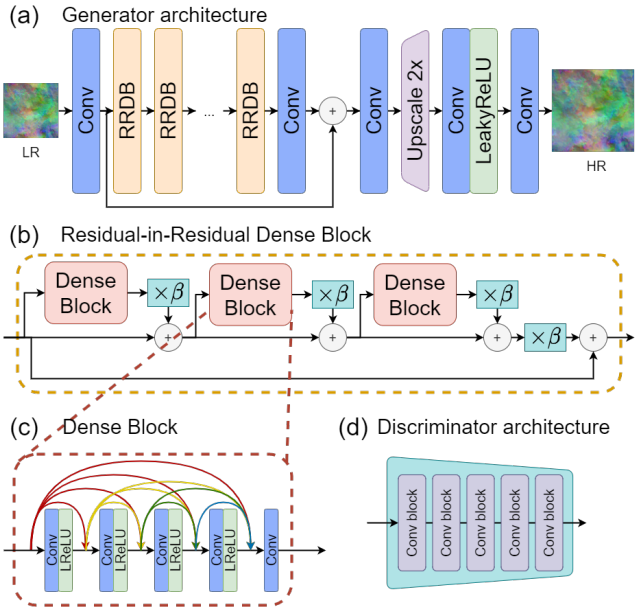

Fig. 2. (a) The spatial super resolution network architecture, utilizing Residual-in-Residual Dense Blocks (b) which contain dense blocks, shown in (c). (d) The spatial discriminator network.

Pre-upsample learning means that the network does its

learning in the low-resolution feature space *before* being upsampled to the final output resolution. This method is the opposite of *post-upsampling*, which first upsamples the data and then learns in the high-dimensional feature space. Training and inferring in this high dimensional space is much more computationally expensive than in the low dimensional space, and can also cause artifacts like blurring in super-resolved output due to biases in the upsampling method used (trilinear interpolation, bicubic interpolation). Given the limited memory budget and to increase training efficiency, we choose to learn in the low-dimensional feature space.

When using pre-upsampling learning in a network, the network must also learn how to upsample to the output resolution. This has been done with deconvolutional layers as well as pixel shuffling in computer vision tasks. Recent work for vector field super resolution has found success with a method called voxel shuffling, the 3D counterpart of pixel shuffling. Voxel shuffling permutes a volume of shape $[u^3C, L, W, H]$ to $[C, uL, uW, uH]$. In our case, $u = 2$, and each spatial dimension is increased by a factor of 2, while the channel dimension reduces by a factor of 8. For this reason, in our model, we add a convolution layer just before the voxel shuffle operation which has 8x more channels than the input, such that the upscaled output has the intended number of channels.

Finally, this architecture uses a deep residual as well as d sequential Residual-in-Residual Dense Blocks (RRDB). RRDB have proven to be powerful for super resolution tasks due to the skip connections in the dense blocks between shallow and deeper features, which allow deeper layers to use features from earlier in the network to influence learning. RRDBs also add two levels of residuals within them, shown in Figure 2(b). First, each dense block creates a residual which is added back to the input to the block. Second, the entire RRDB creates a residual which is added back to its input. Each residual is multiplied by a constant β between 0 and 1. This follows a technique called residual scaling which prevents instability in deep networks. After input has gone through the d RRDBs, that output is finally used as another residual and added back to a shallow feature map from before the first RRDB. All of the residuals used offer many pathways for gradients to be computed through, which helps work against the vanishing gradient problem.

2) Spatial super resolution discriminator: Each generator is coupled with a 3D patch discriminator, shown in Figure 2(d), which classifies overlapping patches as real or fake. This network is designed to be relatively shallow with a small receptive field. Each conv block(convolution, batch normalization, LeakyReLU) has a kernel size of 3 and a stride of 1. With only 5 layers there is a corresponding receptive field of 11 for each discriminator. This small receptive field is to prevent redundant learning across different scales. By keeping the receptive field small, the patch discriminator will only be judging the realness of *local* features in a small window, which were the responsibility of the paired generator to synthesize. Global features present in the data will either be ground truth low resolution data (which are already real), or are synthesized features from a previous generator. If the previous generators have learned to accurately reconstruct features at their scale, then there would be no reason to learn those features again in this discriminator.

We use a GAN for two reasons. First, they have shown to be powerful for reconstructing important high-frequency features for both images as well as vector fields. Second, we expect that output from a GAN will more accurately match the distribution of target fluid vector fields within the domain of all vector fields. This is important when using multiple super resolution generators from the hierarchy sequentially for 4x or higher super resolution. When training, only ground truth data is used as both input and target output. As an example, we downscale a full resolution ground truth volume by a factor of 8, put that through a generator, and train it to match the same volume downscaled by a factor of 4. Consequently, each network is not conditioned to super-resolve output from previous generators as in works like LapSRN and SinGAN, but instead just ground truth data downscaled to it's specific input size. Therefore, the hierarchy is only learning to upscale data from within the ground truth distribution to high resolution vector fields. By constraining the output of each generator to a specific distribution, that of the real data, we should see less compounding errors as networks are used sequentially. This distribution matching is precisely the goal of GANs.

B. Temporal Super Resolution Hierarchy

C. In-situ Octree Downscaling

To take advantage the multi-scale super resolution available, we propose an in-situ octree-based downscaling method that downscales full resolution volumes of data at different ratios for different regions, thereby creating a multi-resolution data frame. The goal is to downscale regions of the volume in-situ where FlowSTSR is able to recover details well in order to relieve the I/O bottleneck while meeting a quality criteria, such as minimum PSNR or maximum relative error. Spatial regions that are the easiest for FlowSTSR to super resolve will be downscaled the most, while regions with features that FlowSTSR cannot super resolve will have little to no downscaling performed. When using this method, the scientist need not select a spatial downsampling rate ahead of time, but can instead specify a quality metric and let this algorithm compress where possible in-situ. We choose the octree data structure to represent the multiresolution output. Our approach is in fact agnostic to upscaling method, meaning that LERP or even other hierarchical super resolution models may benefit from this as well.

We face two challenges when designing the algorithm. First is related to the upscaling method. Many upscaling methods, such as LERP or super resolution neural networks, are subject to artifacts along the boundary of the upscaled region. LERP, for example, aligns corner pixel/voxel values and then will interpolate values in-between. Neural networks

Fig. 3. An example of how upscaling individual blocks separately can cause artifacts and seams in the upscaled result. The top left shows the ground truth image (512x512 pixels) with a 32x32 pixel grid superimposed on it, depicting blocks. The entire image is subsampled by 8 pixels to a size of 64x64 pixels, shown in the top right. Then, the image is upscaled using LERP in two ways. The bottom left shows the image when each block is individually upscaled by 8 and put into the correct position in the final image. The bottom right shows when the entire low resolution image is upscaled together. When blocks are not upscaled together, seams along block boundaries are visible.

may have their boundaries affected by 0-padding in convolutional layers, which can affect any pixels/voxels within the generator's receptive field. An example of these artifacts is visible in Figure 3. When the upscaling method is applied to smaller regions and then stitched together, seams are visible. To resolve this issue, we must first combine all low resolution data blocks before super resolving.

Given a multi-resolution data frame represented as an octree with data on each leaf node, we must first start upscaling from the largest downscaling rate. Since our hierarchy G performs $2\times$ super resolution at each level, the downscaling rate for each level's input is $\{2^{|G|}\times, ..., 4\times, 2\times\}$, meaning we begin the upscaling from $2^{|G|}\times$ downscaling rate.

We begin by creating an empty volume V with the size to be $\frac{1}{2^{|G|}}$th of the full resolution. Next, we fill this volume with the available data from the octree. We have two cases for every leaf node of the octree:

Case 1: The data at the octree leaf node is saved with the same downscaling factor. If this is the case, we directly fill in those voxel's data with the data in the octree leaf.

Case 2: The data at the octree leaf node is saved with a smaller downscaling factor. In this case, we must first downscale the available data in that region to match the downscaling factor for V. For example, if V is $8\times$ smaller than the full resolution, and the octree leaf node has data saved at a $4\times$ downscaling ratio, we must further downscale this leaf node by $2\times$ so that it occupies the correct number of voxels. After downscaling that saved data further, we fill V's corresponding voxels with that data.

Once all leaf nodes have had their data filled in to V, V will have all voxels occupied with $2^{|G|}\times$ downscaled data. With this foundation, we upscale by $2\times$ using some super resolution technique. Since the entire volume was filled, we will not have artifacts on octree node edges.

Now that we are at the next level up in the hierarchy, we may have data saved at this smaller downscaling rate stored in the octree. Since we are guaranteed that the octree saved data is correct, we once again traverse octree leaf nodes and overwrite any voxels in V which have the same or lower downscaling rate just as above. The only change now is that we ignore octree leaf nodes that have a larger downscaling rate than the current V, since that data is now too low resolution and has already been upscaled. After filling in available data, we again upscale V by $2\times$. This process is repeated until V is the full resolution of the ground truth data, at which point no more upscaling happens and the process is finished. Pseudocode for this algorithm is shown in Algorithm 2.

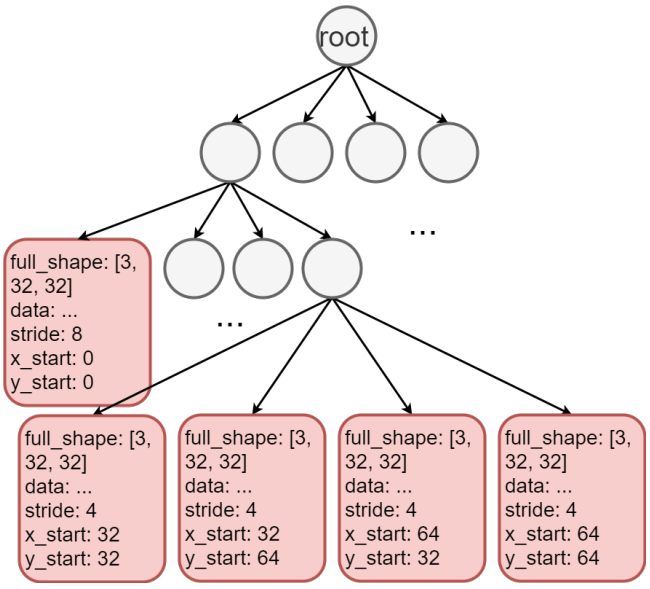

Fig. 4. An example quadtree representing an image. Note that this is analogous to the octree representation mentioned in this paper, except for 2D data. For 3D volumetric data, each leaf node would also have a "z_start" property, and nodes will either have either 0 or 8 children representing 8 subquadrants, instead of 0 or 4. Further, the "full_shape" property would be a [channel, x, y, z] tuple instead of [channel, x, y] as depicted. The gray nodes are body nodes that have the maximum number of children, and the red leaf nodes contain information about the data stored in that node necessary for upscaling it. The "full_shape" property is the size this data block originally occupied in the full resolution data frame. The "data" property holds the saved data. The "stride" property defines what downscaling rate this data block was saved using. For example, a stride of 8 means that the data stored in "data" was downscaled by $8\times$. The "*_start" properties are the absolute coordinates for where this data block exists in the full resolution volume.

The second challenge this method faces is a detail with downscaling. In the process above, note that when an octree leaf node has a downscaling ratio lower than V has, the leaf node's data is downscaled further to match the size it should be in V. For this reason, the downscaling method must have the property defined by $downscale(input, S) = downscale(downscale(input, \frac{S}{2}), \frac{S}{2})$. As an example, downscaling an input by $4\times$ should be equivalent to downscaling it by $2\times$ twice. This is important because of the impact on the distribution of data being upsampled. When training a single model G_i in FlowSTSR for spatial super resolution, we downscale a ground truth frame F to F^{LR} that has a downscaling rate $2^{|G|-i}\times$ and F^{HR} that has a downscaling rate $2^{|G|-i-1}$. We upscale F^{LR} with $G_i(F^{LR})$ and compare it to F^{HR} in our loss functions. Therefore, the model is learning to upscale data from a distribution of vector fields that are downscaled from our ground truth data. If the downscaling method does not hold this specific property, then downscaling a data frame multiple times will create a low resolution frame that is not in the same distribution of vector fields that the network was trained on, which may have adverse effects when super resolving.

This property is held by pooling techniques (min, max, average pooling) as well as uniform downsampling (taking voxels that are uniformly spaced by 2^i), but is not held when downsampling with a linear or cubic kernel.

Algorithm 1 `create_downsampled_octree_from_volume(vol, criteria, downscale, max_stride, min_chunk)`

Input:

vol: a high resolution volume

$criteria$: a method defining some criteria to meet, returns True when met

$downscale$: a method that performs 2x downscaling

max_stride: the maximum downscaling ratio

min_chunk_size: the minimum chunk size a node is allowed to represent

Output:

$octree$: an octree representing vol such that reconstruct_volume_from_octree(octree) satisfies criteria(octree)

```
octree = Octree(vol)
trees_to_check = [ octree ]
while trees_to_check.length > 0 do
    t = trees_to_check.pop()
    t.stride = ⌊t.stride / 2⌋
    original_data = t.data.clone()
    downscaled_data = downscale(t.data, 2)
    t.data = downscaled_data
    upscaled_volume = reconstruct_volume_from_octree(octree)
    if criteria(upscaled_volume) then
        if t.stride < max_stride then
            trees_to_check.append(t)
        end if
    else
        if t.data.size * t.stride > min_chunk_size then
            C = 8 quadrants of t
            for child ∈ C do
                child_octree = Octree(child)
                t.children.append(child_octree)
                trees_to_check.append(child_octree)
            end for
            t.data = null
        end if
    end if
end while
return octree
```

D. Training

E. Datasets

IV. RESULTS

Compare PSNR, angle difference, magnitude difference when applying the different physical constraints.

Compare derived values such as "kinetic energy spectra" and "joint PDF of the second and third invariants of the velocity gradient tensor".

V. CONCLUSION AND FUTURE WORK

REFERENCES

[1] J. Wang, S. Hazarika, C. Li and H. Shen, "Visualization and Visual Analysis of Ensemble Data: A Survey," in IEEE Transactions on Visualization and Computer Graphics, vol. 25, no. 9, pp. 2853-2872, 1 Sept. 2019, doi: 10.1109/TVCG.2018.2853721.

[2] Y. Zhang, Y. Tian, Y. Kong, B. Zhong and Y. Fu, "Residual Dense Network for Image Super-Resolution," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, 2018, pp. 2472-2481, doi: 10.1109/CVPR.2018.00262.

[3] Wang, Xintao and Yu, Ke and Wu, Shixiang and Gu, Jinjin and Liu, Yihao and Dong, Chao and Qiao, Yu and Loy, Chen Change, "ESRGAN: Enhanced super-resolution generative adversarial networks," The European Conference on Computer Vision Workshops (ECCVW), Sept. 2018

[4] Ledig, C., Theis, L., Husz'ar, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., et al., "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network," 2017 IEEE/CVF Conference on Computer Vision and Pattern Recognition

[5] Blau, Y., Mechrez, R., Timofte, R., Michaeli, T., Zelnik-Manor, L., "The Pirm Challenge on Perceptual Super Resolution," https://www.pirm2018.org/PIRM-SR.html (2018)

[6] Assaf Shocher, Nadav Cohen, and Michal Irani. "Zero-Shot Super-Resolution using Deep Internal Learning," In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3118–3126, 2018

[7] T. R. Shaham, T. Dekel and T. Michaeli, "SinGAN: Learning a Generative Model From a Single Natural Image," 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea (South), 2019, pp. 4569-4579, doi: 10.1109/ICCV.2019.00467.

[8] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky."Deep image prior," IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

[9] Saeed Anwar, Salman Khan, and Nick Barnes. 2020. "A Deep Journey into Super-resolution: A Survey," ACM Comput. Surv. 53, 3, Article 60 (June 2020), 34 pages. DOI:https://doi.org/10.1145/3390462

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

Algorithm 2 `reconstruct_volume_from_octree(octree, upscale, max_stride)`

Input:

$octree$: an octree generated by $create\_downsampled\_octree\_from\_volume$ that represents a high resolution volume

$upscale$: a method for upscaling volumetric input by 2x

max_stride: the maximum downscaling ratio

Output:

vol: a high resolution volume upscaled with $upscale$

vol = torch.zeroes($\frac{octree.full\_shape}{max\_stride}$)
s = max_stride
while s > 0 **do**
  trees = [octree]
  while trees.length > 0 **do**
    t = trees.pop()
    if t.data is not null and t.stride $\leq$ s **then**
      vol[$\lfloor \frac{t.x\_start}{s} \rfloor : \lfloor \frac{t.x\_start}{s} + \frac{t.data.size[0]*t.stride}{s} \rfloor$, $\lfloor \frac{t.y\_start}{s} \rfloor : \lfloor \frac{t.y\_start}{s} + \frac{t.data.size[1]*t.stride}{s} \rfloor$, $\lfloor \frac{t.z\_start}{s} \rfloor : \lfloor \frac{t.z\_start}{s} + \frac{t.data.size[2]*t.stride}{s} \rfloor$] = t.data[::$\lfloor \frac{s}{t.stride} \rfloor$,::$\lfloor \frac{s}{t.stride} \rfloor$,::$\lfloor \frac{s}{t.stride} \rfloor$]
    else if t.children.length() > 0 **then**
      for child $\in$ t.children **do**
        trees.append(child)
      end for
    end if
  end while
  if s > 1 **then**
    vol = upsample(vol)
  end if
  s = $\lfloor \frac{s}{2} \rfloor$
end while
return vol

Jan. 2020, doi: 10.1109/TVCG.2019.2934255.

[11] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in Proc. Int. Conf. Medical Image Comput. Comput.-Assisted Intervention, 2015, pp. 234–241.

[12] He, W, Wang, J, Guo, H, et al. (2019) InSituNet: deep image synthesis for parameter space exploration of ensemble simulations. IEEE Transactions on Visualization and Computer Graphics 26(1): 23–33.

[13] Hazarika S., Li H., Wang K., Shen H., Chou C. NNVA: Neural network assisted visual analysis of yeast cell polarization simulation IEEE Trans. Vis. Comput. Graphics (2019) 1–1

[14] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. "Progressive growing of GANs for improved quality, stability, and variation." In ICLR, 2018.

[15] Chuan Li and Michael Wand. "Precomputed real-time texture synthesis with markovian generative adversarial networks." In European Conference on Computer Vision, pages 702–716. Springer, 2016.

[16] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. "Improved training of wasserstein GANs." In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17). Curran Associates Inc., Red Hook, NY, USA, 5769–5779.

[17] Bryan, G.H.; Fritsch, J.M. "A Benchmark Simulation for Moist Nonhydrostatic Numerical Models." Mon. Weather Rev. 2002, 130, 2917–2928.

[18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

[19] Arash Vahdat and Jan Kautz, "NVAE: A Deep Hierarchical Varational Autoencoder", arXiv preprint arXiv:2007.03898, 2020

[20] Xun Huang, Yixuan Li, Omid Poursaeed, John Hopcroft, and Serge Belongie. "Stacked generative adversarial networks." In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 5077–5086, 2017.

[21] Emily L Denton, Soumith Chintala, Rob Fergus, et al. "Deep generative image models using a laplacian pyramid of adversarial networks." In Advances in neural information processing systems, pages 1486–1494, 2015.

[22] M. Chu, Y. Xie, J. Mayer, L. Leal-Taixé, and N. Thuerey, "Learning temporal coherence via self-supervision for GAN-based video generation," 2018, arXiv:1811.09393. [Online]. Available: http://arxiv.org/abs/1811.09393

[23] J. van Amersfoort, W. Shi, A. Acosta, F. Massa, J. Totz, Z. Wang, and J. Caballero, "Frame interpolation with multi-scale deep loss functions and generative adversarial networks," arXiv preprint arXiv:1711.06045, 2017

[24] Xie Y, Franz E, Chu M, Thuerey N. 2018. "tempoGAN: a temporally coherent, volumetric GAN for superresolution fluid flow." ACM Trans. Graph. 37:95

[25] Sajjadi MS, Vemulapalli R, Brown M. "Frame-recurrent video super-resolution." In: IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp 6626–6634

[26] A. Kappeler, S. Yoo, Q. Dai, and A. K. Katsaggelos. "Video super-resolution with convolutional neural networks." In IEEE Transactions on Computational Imaging, 2016

[27] J. Caballero, C. Ledig, A. Aitken, A. Acosta, and Totz. "Realtime video super-resolution with spatio-temporal networks and motion compensation." In CVPR, 2017.

[28] X. Tao, H. Gao, R. Liao, J. Wang, and J. Jia. "Detail-revealing deep video super-resolution." In ICCV, 2017.

[29] D. Liu, Z. Wang, Y. Fan, X. Liu, Z. Wang, S. Chang, and T. Huang. "Robust video super-resolution with learned temporal dynamics." In CVPR, 2017.

[30] O. Makansi, E. Ilg, and T. Brox. "End-to-end learning of video super-resolution with motion compensation." In GCPR, 2017.

[31] Francesco Cardinale et al., "ISR", 2018 https://github.com/idealo/image-super-resolution

[32] Assaf Shocher, Nadav Cohen, and Michal Irani. "Zero-Shot Super-Resolution using Deep Internal Learning," 2018, https://github.com/assafshocher/ZSSR

[33] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky."Deep image prior," 2018, https://github.com/DmitryUlyanov/deep-image-prior

[34] Agustsson, E., Timofte, R. "Ntire 2017 challenge on single image super-resolution: Dataset and study." In: CVPRW. (2017)

[35] N. Ahn, B. Kang, and K.-A. Sohn, "Fast, accurate, and, lightweight super-resolution with cascading residual network," ECCV, 2018.

[36] Y. Fan, H. Shi, J. Yu, D. Liu, W. Han, H. Yu, Z. Wang, X. Wang, and T. S. Huang, "Balanced two-stage residual networks for image super-resolution," in CVPRW, 2017.

[37] Z. Wang, D. Liu, J. Yang, W. Han, and T. Huang, "Deep networks for image super-resolution with sparse prior," in ICCV, 2015.

[38] M. S. Sajjadi, B. Scholkopf, and M. Hirsch, "Enhancenet: Single image super-resolution through automated texture synthesis," in ICCV, 2017.

[39] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in CVPR, 2015.

[40] J. Kim, J. Kwon Lee, and K. Mu Lee, "Accurate image superresolution using very deep convolutional networks," in CVPR, 2016.

[41] Huaizu Jiang, Deqing Sun, Varun Jampani, Ming-Hsuan Yang, Erik Learned-Miller, and Jan Kautz. Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

[42] Simon Niklaus, Long Mai, and Feng Liu. Video frame interpolation via adaptive separable convolution. In IEEE International Conference on Computer Vision (ICCV), 2017.

[43] X. Tong, T. Lee and H. Shen, "Salient time steps selection from large scale time-varying data sets with dynamic time warping," IEEE Symposium on Large Data Analysis and Visualization (LDAV), Seattle, WA, 2012, pp. 49-56, doi: 10.1109/LDAV.2012.6378975.

[44] : Xin Liang, Sheng Di, Dingwen Tao, Zizhong Chen, Franck Cappello, "Error-Controlled Lossy Compression Optimized for High Compres-

sion Ratios of Scientific Datasets", in IEEE International Conference on Big Data (Bigdata 2018), Seattle, WA, USA, 2018.
[45] Dingwen Tao, Sheng Di, Franck Cappello, "Significantly Improving Lossy Compression for Scientific Data Sets Based on Multidimensional Prediction and Error-Controlled Quantization", in IEEE International Parallel and Distributed Processing Symposium (IPDPS 2017), Orlando, Florida, USA, 2017.
[46] Sheng Di, Franck Cappello, "Fast Error-bounded Lossy HPC Data Compression with SZ", in IEEE International Parallel and Distributed Processing Symposium (IPDPS 2016), Chicago, IL, USA, 2016.
[47] Xin Liang, Sheng Di, Dingwen Tao, Zizhong Chen, Franck Cappello, "An Efficient Transformation Scheme for Lossy Data Compression with Point-wise Relative Error Bound", in IEEE International Conference on Clustering Computing (CLUSTER 2018), Belfast, UK, 2018. (Best Paper)
[48] M. Burtscher and P. Ratanaworabhan. “FPC: A High-Speed Compressor for Double-Precision Floating-Point Data.” IEEE Transactions on Computers, Vol. 58, No. 1, pp. 18-31. January 2009.
[49] Hoang, D.; Klacansky, P.; Bhatia, H.; Bremer, P.T.; Lindstrom, P.; Pascucci, V. A Study of the Trade-off Between Reducing Precision and Reducing Resolution for Data Analysis and Visualization. IEEE Trans. Vis.Comp. Graph. 2019, 25, 1193–1203.
[50] N. Sasaki , K. Sato, T. Endo, and S. Matsuoka, “Exploration of Lossy Compression for Application-level Checkpoint/Restart, ” in 2015 IEEE 29th International Parallel and Distributed Processing Symposium, pp. 914–922, 2015
[51] A.H. Baker, H. Xu, J.M. Dennis, M.N. Levy, D. Nychka, and S.A. Mickelson, “A Methodology for Evaluating the Impact of Data Compression on Climate Simulation Data, ” in The ACM International Symposium on High-Performance Parallel and Distributed Computing (HPDC14), pp. 203-214, 2014.
[52] Simonyan, K. and Zisserman, A. "Very deep convolutional networks for large-scale image recognition." In ICLR 2015.
[53] Radu Timofte, Shuhang Gu, Jiqing Wu, and Luc Van Gool. "Ntire 2018 challenge on single image super-resolution: methods and results." In CVPRW, 2018
[54] Jianrui Cai, Shuhang Gu, Radu Timofte, and Lei Zhang. Ntire 2019 challenge on real image super-resolution: Methods and results. In CVPRW, 2019
[55] Radu Timofte, Eirikur Agustsson, Luc Van Gool, MingHsuan Yang, and Lei Zhang. Ntire 2017 challenge on single image super-resolution: Methods and results. In CVPRW, 2017
[56] Andreas Lugmayr, Martin Danelljan, Radu Timofte, et al. Aim 2019 challenge on real-world image super-resolution: Methods and results. In ICCV Workshops, 2019
[57] Y. Wang, L. Wang, J. Yang, W. An, and Y. Guo, “Flickr1024: A largescale dataset for stereo image super-resolution,” in Proc. IEEE Int. Conf. Comput. Vision Workshops, 2019, pp. 3852–3857

VI. SUPPLEMENTARY MATERIAL