Predicting High-Resolution Turbulence Details in Space and Time

KAI BAI, ShanghaiTech University/SIMIT/UCAS
CHUNHAO WANG, ShanghaiTech University
MATHIEU DESBRUN, Inria Saclay/Ecole Polytechnique/Caltech
XIAOPEI LIU, ShanghaiTech University

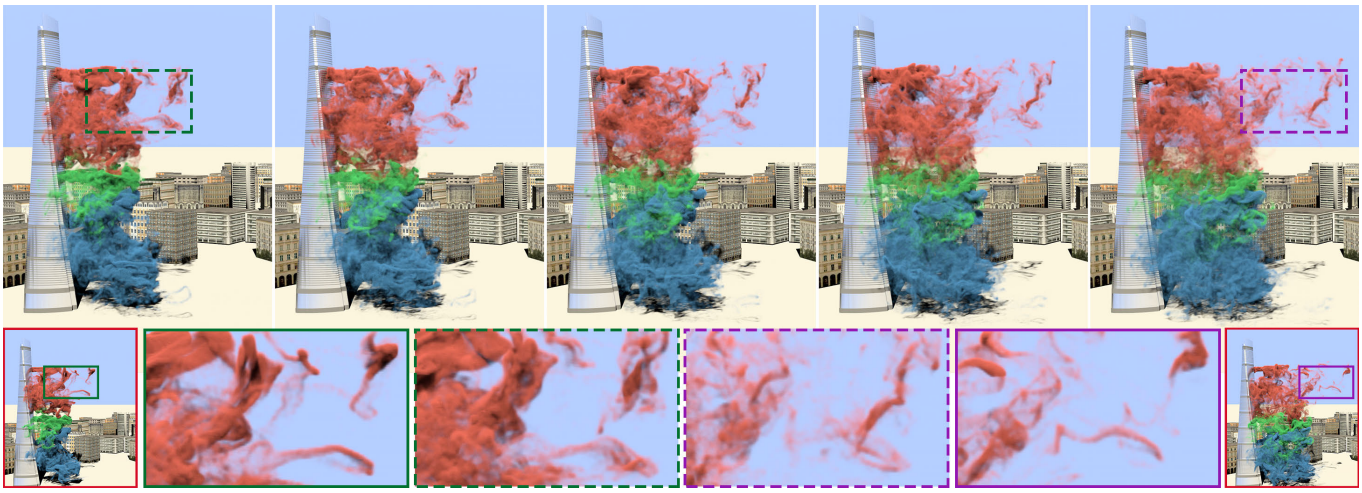

Fig. 1. **Spatio-temporal super-resolution of a downsampled wind turbulence simulation around a high-rise building**. Our new learning-based framework supports high-resolution and high-frame-rate turbulent flow synthesis, and achieves strong generalization. Trained with small local patches collected from the *flow simulation around a ball* in Fig. 3, our dictionary-based neural network can predict turbulent flow details from a low-rate, low-resolution input. Bottom row shows two consecutive frames of the low-resolution input simulation (far left, far right), with two respective zoom-ins (framed through solid lines); these same frames are spatially upsampled to the top leftmost and top rightmost frames, see also their corresponding zoom-ins (framed through dotted lines). Top row shows temporal upsampling, with new middle frames interpolating between the two upsampled frames. Note the refinement of smoke structures during spatial upsampling and their realistic temporal coherence, despite a very different training exemplar.

Predicting the fine and intricate details of a turbulent flow field in both space and time from a coarse input remains a major challenge despite the availability of modern machine learning tools. In this paper, we present a simple and effective dictionary-based approach to spatio-temporal upsampling of fluid simulation. We demonstrate that our neural network approach can reproduce the visual complexity of turbulent flows from spatially and temporally coarse velocity fields even when using a generic training set. Moreover, since our method generates finer spatial and/or temporal details through embarrassingly-parallel upsampling of small local patches, it can efficiently predict high-resolution turbulence details across a variety of grid resolutions. As a consequence, our method offers a whole range of applications varying from fluid flow upsampling to fluid data compression. We demonstrate the efficiency and generalizability of our method for synthesizing turbulent flows on a series of complex examples, highlighting dramatically better results in spatio-temporal upsampling and flow data compression than existing methods as assessed by both qualitative and quantitative comparisons.

CCS Concepts: • **Computing Methodologies** → **Neural Networks; Physical Simulation** .

Additional Key Words and Phrases: Fluid Simulation, Dictionary Learning, Neural Networks, Smoke Animation

ACM Reference Format:
Kai Bai, Chunhao Wang, Mathieu Desbrun, and Xiaopei Liu. 2021. Predicting High-Resolution Turbulence Details in Space and Time. *ACM Trans. Graph.* 40, 6, Article 200 (December 2021), 16 pages. https://doi.org/10.1145/3478513.3480492

Authors' addresses: B. Kai, C. Wang, X. Liu: School of Information Science and Technology (Shanghai Engineering Research Center of Intelligent Vision and Imaging) of ShanghaiTech University, Shanghai, China; B. Kai is also affiliated with the Shanghai Institute of Microsystem and Information Technology (SIMIT) and the University of the Chinese Academy of Sciences (UCAS); M. Desbrun: Inria Saclay, LIX/DIX, Institut Polytechnique de Paris, Palaiseau, France; M. Desbrun is on leave from Caltech.

0730-0301/2021/12-ART200 $15.00
https://doi.org/10.1145/3478513.3480492

1 INTRODUCTION

Be it in computer graphics (CG) or computational fluid dynamics (CFD), efficiently predicting physically-consistent high-frequency details from a spatially and/or temporally coarse (low-resolution) input is extremely desirable in a variety of contexts — but such a task is also acknowledged to be a very challenging problem, especially for turbulent flows. For instance, fluid simulation would be

drastically faster if one could compute a coarse version of the simulation followed by an efficient upsampling procedure that enriches the input with physically-plausible fine details at very little cost; a similar approach could also provide a viable solution for turbulence modeling in CFD [Wilcox et al. 1998]. Variants of this concept have been proven useful over the years: fluid super-resolution (upresing a spatially downsampled velocity field), fluid upsampling (upresing a spatially or temporally low-resolution simulation to a high-resolution one), fluid resimulation (slight editing to produce another similar simulation very efficiently), and fluid data compression (for efficient storage of time-varying velocity fields) can all benefit from a faster or better prediction of finely-detailed fluid motion from a coarse, low-resolution input. While a variety of techniques have been proposed in the literature to address some of these applications, very few approaches are available for temporal upsampling of fluid flows, and handling both spatial and temporal upsampling of varying fluid flow behaviors within the same computational framework is currently out of reach.

In this paper, we introduce a simple but effective learning-based approach, which not only outperforms state-of-the-art methods for spatial upsampling of velocity fields at high Reynolds numbers, but also offers high-quality temporal upsampling of arbitrary fluid flows using the same network structure. Building upon the recent work of Bai et al. [2020] which argued that fine spatial structures of turbulent flows are seemingly complex overall but locally simple, our method for predicting turbulent flow details also relies on a localized dictionary-based neural network that learns how to combine simple, local structures to obtain fine details globally. However, in a marked departure from [Bai et al. 2020], we show that additional filtering in the process can dramatically improve the prediction quality of the neural network, and that further augmenting the input with time stamps also leads to high-quality temporal interpolation between frames. As a result, our novel localized dictionary-based neural network can efficiently synthesize turbulent flows both spatially and temporally from coarse inputs as shown in Fig. 1. This approach can be similarly applied to produce visually plausible turbulent flows for a multitude of contexts in a highly generalizable manner, even when only trained on a single fluid simulation dataset — a key feature that none of the previous methods were able to offer. We test our method on a series of complex real-world applications, highlighting dramatically better results in spatio-temporal super-resolution/upsampling and flow data compression than existing methods as assessed by both qualitative and quantitative comparisons.

1.1 Related Work

We begin our exposition with a brief review of CG and CFD methods for producing fine details in fluid flows, to assess the different existing approaches and their varying levels of accuracy.

Turbulence modeling in CFD. Fluid simulation has been studied over decades in both CFD [Fadlun et al. 2000; Kim et al. 2001; Chessa and Belytschko 2003; Geier et al. 2006; Versteeg and Malalasekera 2007; Sierakowski and Prosperetti 2016; De Rosis 2017] and CG [Selle et al. 2008; Mullen et al. 2009; Zhu et al. 2013; de Goes et al. 2015; Zhang et al. 2015; Zehnder et al. 2018; Qu et al. 2019; Li et al. 2020]. Yet, accurately simulating turbulent flows is still resource- and time-demanding due to their chaotic, small dynamic features that require fine grid resolutions to resolve. Predicting from a simulation on a coarse grid the averaged high-frequency fluctuations using a simplified model is an attractive alternative used in turbulence modeling: Reynolds-averaged Navier-Stokes (RANS) models [Alfonsi 2009], large-eddy simulation (LES) models [John 2003] or the hybrid detached-eddy simulation (DES) models [Spalart 2009] are typical examples of turbulence models. However, all these models involve empirical parameters to adjust, and can still be quite slow if accuracy is called for — see [Cécora et al. 2015] for a recent example.

Physically-based reduced models in CG. In order to bypass the complexity of turbulence models, CG approaches to predicting fine fluid structures rely instead on physically-inspired, but heavily-simplified models. While they are often far from being physically accurate, they add enough plausible visual complexity to improve the overall flow motion details. Early models employed vorticity confinement [Steinhoff and Underhill 1994; Fedkiw et al. 2001] and vortex particles [Park and Kim 2005; Weißmann and Pinkall 2010; Golas et al. 2012; Pfaff et al. 2012] to help add fluid details. Later models include variants of noise-based methods [Bridson et al. 2007; Kim et al. 2008] as well as simplified turbulence models [Schechter and Bridson 2008; Pfaff et al. 2010] to more correctly capture flow details. Various other methods to capture fluid flow details have been formulated as well; e.g., [Wicke et al. 2009] constructed a set of composable reduced models or tiles, which capture spatially localized fluid behavior, while [Kim and Delaney 2013] presented a subspace integration method capable of efficiently adding/subtracting dynamics from an existing high-resolution fluid flow simulation. Recently, [Forootaninia and Narain 2020] suggested the addition of smoke details by simply combining the high-frequency components of a simulated fluid velocity with the low-frequency components of an input guiding field, and [Sato et al. 2021] proposed to formulate fluid guidance as a minimization problem in stream function space to add turbulence details to low-resolution fluid flows. However, due to physical inconsistency of the prediction induced by these models, unrealistic phenomena are often observed in their results.

Learning-based models. As machine learning techniques grew in popularity, they became useful to construct more accurate reduced models. In CG for instance, they have been explored as a means to synthesize either super-resolution or upsampling of fluid flows. For super-resolution, the input velocity field is a downsampled version of a high-resolution simulation, and generative neural networks [Xie et al. 2018; Werhahn et al. 2019] or local dictionary-based neural network [Bai et al. 2020] have been proposed to recover the missing fine details. For fluid upsampling, the input is a fluid simulation computed on a coarse grid, and methods to learn local patch descriptors [Chu and Thuerey 2017] or the formation of small-scale splashes [Um et al. 2018] have been formulated to enrich a simulation. The recent dictionary-based neural network of [Bai et al. 2020] can also handle fluid upsampling, but only for relatively restricted scenarios. Note that other learning-based models related to fluid simulation targeting the full synthesis of fluid flows rather than high-frequency flow details [Jeong et al. 2015; Tompson et al. 2017; Guo et al. 2016; Kim et al. 2019] or the prediction of specific

physical quantities such as aerodynamic drag [Umetani and Bickel 2018] have also been proposed: a solver-in-the-loop approach formulated correction functions to reduce numerical errors of iterative PDE solvers [Um et al. 2020], but training has to be performed on a case-by-case basis and is restricted to low Reynolds numbers (less than 10^3), making this approach valuable only in the case of re-simulation; Oh and Lee [2021] proposed a two-step temporal interpolation network using forward advection to generate smoke simulation efficiently, but cannot handle turbulent flows well. In CFD, machine learning methods have also been studied as a new alternative for turbulence modeling [Duraisamy et al. 2019]. For example, [Long et al. 2019] proposed a deep neural network to discover (time-dependent) PDEs from observed dynamic data with minor prior knowledge on the underlying mechanism; [Obiols-Sales et al. 2020] introduced a physical simulation and deep learning coupled framework to accelerate the convergence of Reynolds Averaged Navier-Stokes (RANS) simulations; More recently, [Fukami et al. 2021] proposed a convolutional neural network to perform spatio-temporal super-resolution, but only for flows with small Reynolds numbers (Re < 200) and with up to weeks of training time, while [Kochkov et al. 2021] used end-to-end deep learning to improve approximations for two-dimensional turbulent flows.

Style transfer. Fluid style transfer has also been targeted in CG, which at times requires predicting high-frequency flow details. For example, Sato et al. [2018] extended example-based image synthesis methods to handle velocity fields using a combination of patch-based and optimization-based texture synthesis that can transfer the turbulent style of an existing fluid simulation onto a new one; Kim et al. [2019] introduced the Transport-based Neural Style Transfer (TNST) algorithm, which can modify smoke simulations based on natural images by extending traditional image-based neural style transfer and reformulating it as a transport-based optimization. Most recently, Kim et al. [2020] presented a neural style transfer approach from images to 3D fluids which ensures temporal consistency of the optimized stylized structures.

Compression. A wide variety of efficient compression schemes for images and video sequences are available nowadays, see, e.g. [Joshi et al. 2014; Ma et al. 2019]. However, most of these techniques cannot be directly applied to the encoding of a time-varying velocity field, explaining the relative dearth of works in the literature for fluid data compression. One of the most frequently used techniques employs the discrete wavelet transform [Kang et al. 2003] for each frame, a direct extension of wavelet-based image compression. Quantization and entropy encoding can then be applied to the resulting wavelet coefficients [Sakai et al. 2013; Kolomenskiy et al. 2019], but this additional step can dramatically debase the vector field as it trades details in the flow for lower bitrates. There are also pure bitwise flow field compression techniques for lossless compression of floating-point data based on predictive coding [Lindstrom and Isenburg 2006] or for fixed-rate, near-lossless compression based on block transforms and embedded coding [Lindstrom 2014]. Recently, a few learning-based models have appeared which could potentially be adapted for compressing fluid datasets to reach far better compression ratios: Kim et al. [2019] proposed a generative CNN model which can synthesize fluid velocity fields based on scene setup parameters; Wiewel et al. [2019] proposed an LSTM-based approach to predicting changes of pressure fields over time. However, they are all *global* encoding models that restrict the resolution of the input to be the same as the training set; they are thus less flexible for compressing arbitrary flow fields, and are likely to be less efficient at compressing turbulent flow datasets.

1.2 Overview

Despite a large amount of related work, learning the chaotic and high-frequency structures of a turbulent flow both spatially and temporally from a coarse resolution input is still a major challenge. In this paper, we provide a unified learning-based approach to upsampling coarse inputs, such that both spatial *and* temporal upsampling can be performed efficiently. Moreover, we show that our approach can handle very turbulent smoke flows (i.e., flows at high Reynolds numbers) for a variety of contexts without generating visible artifacts even when using a very small training set, which opens up an opportunity for supporting a wide range of applications — from fast simulation to compression of turbulent flows — within the same computational framework.

2 SPATIO-TEMPORAL UPSAMPLING

We now describe our entire learning-based framework for predicting high-frequency flow structures, beginning with a brief summary of the work we built upon, followed by a discussion of the numerical experiments that set off our work, before detailing our various contributions to spatial and temporal upsampling.

2.1 Recap of the original approach by Bai et al. [2020]

The dictionary-based neural network recently proposed by Bai et al. [2020] lays the foundations of a very effective patch-based learning approach for predicting high-frequency flow structures — as often present in turbulent flows — from which we will build our own method. Their algorithm can be best understood as aiming to decompose a small patch $\boldsymbol{u}$ of a high-resolution velocity field into multiple components: $\boldsymbol{u}_L^* + \boldsymbol{u}_{H_1} + \boldsymbol{u}_{H_2} + ... + \boldsymbol{u}_{H_n}$, where $\boldsymbol{u}_L^*$ is a rough approximation of the flow patch obtained by first downsampling $\boldsymbol{u}$, and then upsampling this coarse patch back to its original resolution through trilinear interpolation (thus representing the input patch without fine details), while $\boldsymbol{u}_{H_1}$, ..., $\boldsymbol{u}_{H_n}$ are high-frequency *residual* patches at n different scales (frequency bands) that a neural network $\mathcal{H}$ based on sparse encoding with an over-complete dictionary predicts from $\boldsymbol{u}_L^*$. With such a dictionary-based neural network, given a low-resolution flow field, each of its local patches $\hat{\boldsymbol{u}}_L$ is first trilinearly upsampled to generate a high-resolution/low-frequency approximation $\hat{\boldsymbol{u}}_L^*$, on which high frequency residual patches are added; each residual patch $\hat{\boldsymbol{u}}_{H_i}$ of a given scale is then reconstructed by a sparse linear combination of the learned dictionaries, resulting in the prediction of flow details. More concretely, the architecture of the neural network, illustrated in Fig. 2, takes an input vector $\boldsymbol{y}_L$ (typically composed of a coarse patch at time t and at several previous time steps to help prediction), and consists in the serial application of sub-modules $\mathcal{H}_i$, each taking the residual patch of the previous scale as input: $\hat{\boldsymbol{u}}_{H_i} = \mathcal{H}_i(\Theta_i, \hat{\boldsymbol{u}}_{H_{i-1}})$, where $\hat{\boldsymbol{u}}_{H_0} = \boldsymbol{y}_L$.

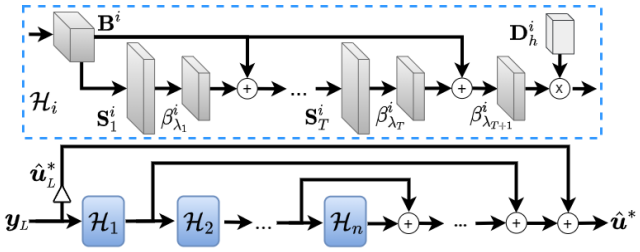

Fig. 2. **The original neural network proposed by [Bai et al. 2020]**. To synthesize high-resolution velocity patches, multiple network sub-modules are used to predict different residual patches from trilinear-upsampled input velocity patches (denoted by the delta symbol) that are extracted from the input vectors and the ground-truth high-resolution velocity patches, where each sub-module $\mathcal{H}_i$ is constructed by a dictionary-based neural network illustrated in the upper rectangular region enclosed by dotted blue lines, where $\mathbf{B}^i$, $\mathbf{S}^i$ and β^i are network parameters and $\mathbf{D}^i$ is the dictionary.

The vector Θ_i contains all the network parameters $\mathbf{B}^i$, $\mathbf{S}_j^i$ and $\beta_{\lambda_j}^i$ (for $j \in \{1, 2, ..., T\}$ indexing the T layers), as well as the learned dictionaries $\mathbf{D}_h^i$ that are used in each sub-module to synthesize the high-frequency residual patches. Note that it is easy to form $\hat{\boldsymbol{u}}_L^*$ by first picking low-resolution velocity patch $\hat{\boldsymbol{u}}_L$ out of the input $\boldsymbol{y}_L$, and then directly upsampling it by trilinear interpolation.

Training the whole network uses a loss function containing three distinct error metrics. The ℓ_2 synthesis error $\mathcal{E}_\ell$ measures the difference between the predicted patch (taking $\boldsymbol{y}_L$ as input) and the ground-truth patch $\hat{\boldsymbol{u}}$ from a training set containing K patches:

$$\mathcal{E}_\ell = \sum_{k=1}^{K} \left\| \hat{\boldsymbol{u}}(\boldsymbol{x}_k) - (\hat{\boldsymbol{u}}_L^*(\boldsymbol{x}_k) + \mathcal{H}(\boldsymbol{y}_L(\boldsymbol{x}_k); \Theta)) \right\|_2^2 ; \tag{1}$$

the Sobolev synthesis error $\mathcal{E}_g$ measures the Frobenius norm of the gradient error between synthesized and ground-truth patches, i.e.,

$$\mathcal{E}_g = \sum_{k=1}^{K} \left\| \nabla[\hat{\boldsymbol{u}}(\boldsymbol{x}_k)] - \nabla[\hat{\boldsymbol{u}}_L^*(\boldsymbol{x}_k) + \mathcal{H}(\boldsymbol{y}_L(\boldsymbol{x}_k); \Theta)] \right\|_F^2 , \tag{2}$$

where $\nabla[\cdot]$ is a component-wise gradient operator defined as $\nabla[\mathbf{x}] = [\nabla x_1, \nabla x_2, ..., \nabla x_n]^T$; and the divergence synthesis error $\mathcal{E}_d$ measures the divergence difference between synthesized and ground-truth patches through

$$\mathcal{E}_d = \sum_{k=1}^{K} \left\| \nabla \cdot [\hat{\boldsymbol{u}}(\boldsymbol{x}_k)] - \nabla \cdot \left[\hat{\boldsymbol{u}}_L^*(\boldsymbol{x}_k) + \mathcal{H}(\boldsymbol{y}_L(\boldsymbol{x}_k); \Theta) \right] \right\|_2^2 . \tag{3}$$

The loss function combines these three error measurements via:

$$L_T(\Theta) = \alpha_\ell \mathcal{E}_\ell + \alpha_g \mathcal{E}_g + \alpha_d \mathcal{E}_d + \alpha_\Theta \|\Theta\|_2^2 , \tag{4}$$

where the last ℓ_2 norm is used to prevent over-fitting during learning. Parameters α_ℓ, α_g, α_d and α_Θ help balance between training and test losses, see [Bai et al. 2020] for details. Once trained, the resulting dictionary-based neural network can perform spatial upsampling effectively, and even temporal upsampling by interpolating the intermediate frame patches that are taken as the network input. However, the authors acknowledged the importance of the training set to the success of the upsampling process due to limited generalizability; and indeed, our own tests demonstrate that spatially upsampling a high Reynolds number turbulent flow often exhibits blocking artifacts, dramatically lacking the typical crisp contours that fine simulations are known to produce, see the zoom-in view in Fig. 4 (i); additionally, temporal upsampling results are often poor as soon as one requires more than a two-fold increase in frame frequency compared to the input sequence, see Fig. 7 (a).

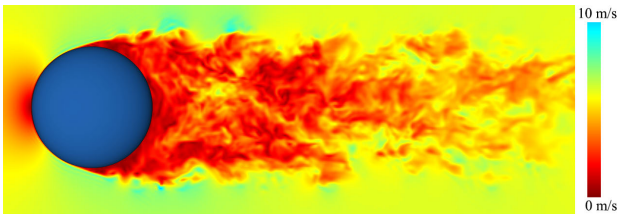

Fig. 3. **Turbulent wake flow behind a ball**. When air flows around a ball at a relatively high speed, turbulence will be generated behind the ball, creating fine chaotic structures visualized here through a cross-section displaying the velocity field's magnitude.

Note finally that Bai et al. [2020] proposed a lot of different variants of this basic approach. As an example, the input $\boldsymbol{y}_L$ can be composed of three patches from the same spatial location (at frames t, $t-1$, and $t-2$), and can even include vorticity patches to further enrich the context from which prediction is learned. In our work, we considerably simplify their approach by stripping out all these variants, ending up in a *simpler and more generic approach to spatio-tempoal upsampling*. For instance, we use only two frames as input (the current frame and the previous frame), i.e., $\boldsymbol{y}_L = [\hat{\boldsymbol{u}}_L^t, \hat{\boldsymbol{u}}_L^{t-1}]$, as our new approach gains little from including more context. In addition, we completely discard the space-time encoding proposed in [Bai et al. 2020] as it does not offer enough generalizability, and we only use $n = 2$ submodules as we found it to be usually sufficient to retain visual richness, even if the flow field is turbulent. We will also drastically simplify the various training scenarios that depend on the type of upsampling being sought after.

2.2 Motivations

Our novel contributions stem from experimenting with the learning-based framework proposed by Bai et al. [2020] to better delineate the issues that are limiting its applicability.

Characteristics of turbulent flows. Turbulent flows are ubiquitous in nature, exhibiting chaotic and multiscale spatio-temporal structures [Clark di Leoni et al. 2015] that usually require high resolution simulations to resolve accurately. Predicting these fine, fast-changing details from a low-frequency input in a physically consistent manner is thus extremely difficult due to the very nature of turbulence and its complex multiscale behavior, see Fig. 3 for a typical turbulent wake flow behind a ball. Consequently, many existing learning-based techniques such as convolutional neural networks (CNN) that were designed to predict fine details in images and videos may simply fail in this context. However, although the global structure of a turbulent flow is complex, it is locally rather simple due to the differential nature of its underlying dynamics: as pointed out by [Bai et al. 2020], it is the intricate *combination* of various locally simple structures from which global chaotic patterns emerge. Thus, designing a localized learning-based algorithm is *key* to the success of predicting high-frequency details in a *turbulent* flow, especially when the grid resolution is high.

Influence of aliasing on learning. Given the specificity of fluid flows, our initial attempts at flow upresing quickly honed in on the

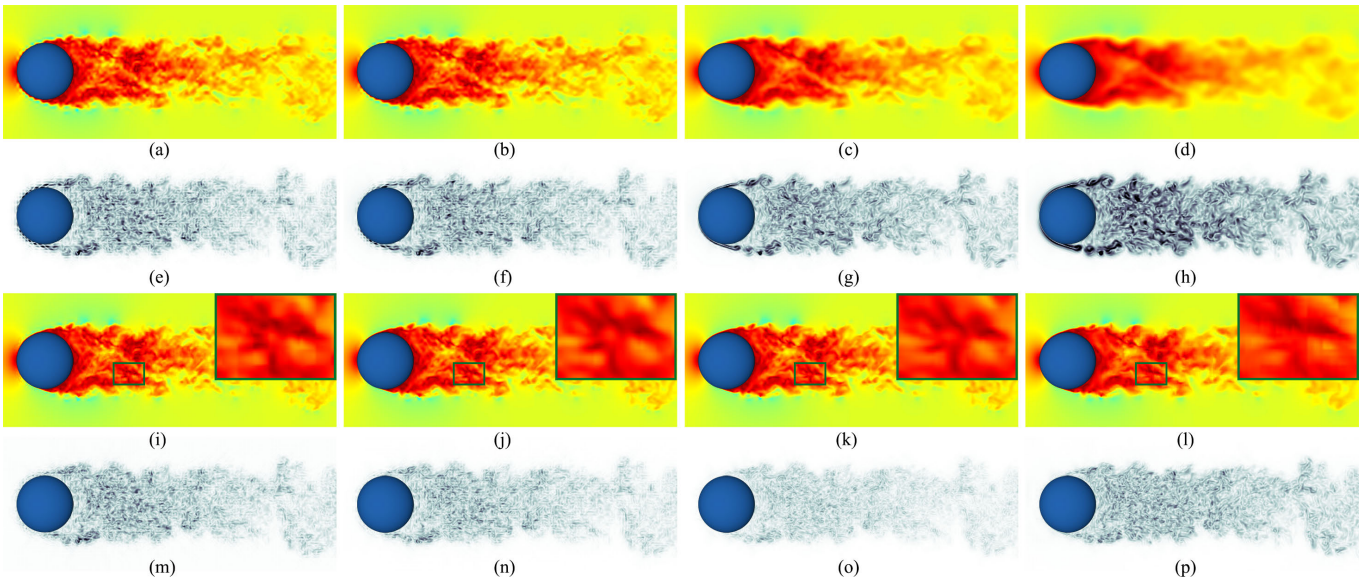

Fig. 4. **Input filtering for a turbulent flow around a sphere.** Top: downsampled input fields from a high-resolution simulation on which a Gaussian filter was applied, with (a) $\sigma = 0$, (b) $\sigma = 0.5$, (c) $\sigma = 2$, and (d) $\sigma = 4$, which were used to prepare coarse input patches together with the corresponding high-resolution goundtruth patches to train the neural network proposed by [Bai et al. 2020]. (e) to (h): the corresponding residual fields between the ground-truth high-resolution simulation and the fields of (a) to (d) upsampled using trilinear interpolation; note different amounts of aliasing. (i) to (l): synthesis results using the neural networks trained separately with the inputs from (a) to (d); note the blocking artifacts in (i) and over-smoothing in (l). (m) to (p): the residual fields between synthesized high-resolution fields and the ground-truth field; note that a minimal error is obtained when an appropriate σ value is selected (here, $\sigma = 2$ in (k) and (o)).

learning approach of [Bai et al. 2020] for its appealing properties mentioned earlier. While testing their method, we observed that the output quality of their neural network *depends heavily* on the nature of input patches: very small differences in the training sets can engender large variations in terms of both training time and prediction accuracy. Out of sheer curiosity, we tried to experiment with *filtering of the training set*: while their original method for super-resolution use coarse-fine pairs of patches for training where the coarse patches are obtained through direct downsampling of a fine-resolution flow (computed through trilinear interpolations of node-based velocities, followed by point-sampling at a coarser resolution), we modified their downsampling process by first applying Gaussian filtering to the fine-resolution flow *before* point-sampling the result on the coarse grid, for varying values of standard deviation σ — see Fig. 4, which identifies that too little or too much filtering both leads to larger prediction errors; instead, adding a *specific* amount of filtering is drastically better. This experiment helped us realize that an *aliasing issue* was happening: the downsampling employed by [Bai et al. 2020] introduces aliasing artifacts in the input that render learning of detail predictions more challenging than it should. Indeed, the spurious high-frequency artifacts generated by their unfiltered downsampling *complicates the learning process* due to the added non-physical, *incoherent* structures. However, a proper downsampling (involving filtering) that extracts the correct low-frequencies of the fine resolution removes this issue to a large extent, making the loss function easier to optimize. If too much filtering is used instead, patches become less salient (as they are all washed out), and the training becomes difficult again.

Impact of subgrid interactions. Another important observation we made concerns how the training set should be prepared to improve generalizability. In [Bai et al. 2020], the training pairs are mostly prepared by collecting patches from low-resolution simulations and their corresponding high-resolution simulations. As low-resolution simulations of turbulent flows cannot resolve subgrid features (assuming a very simple turbulence model is used), they may deviate quite significantly from the downsampled velocity fields of the corresponding high-resolution simulations. These deviations are usually difficult to predict, which is why a large number of turbulence models were proposed in the literature, aiming at achieving more accurate predictions. However, until now, there is no universal turbulence model which is sufficiently accurate for turbulent flow simulations. Thus, involving such deviations into the learning process can dramatically enlarge the function space that needs to be learned and increase the total number of required training pairs, making generalization much more difficult for a small training set and with limited computational resources.

Design consequences. Motivated from our experimentation and analysis described above, we can first trivially apply known signal processing tools to properly filter a fine simulation into a coarse (low-frequency) one in order to create better training sets: a Gaussian filter, for instance, will enforce that no spurious high frequencies are present in the result, while maintaining most of the low-frequencies of the initial flow — thus ensuring efficient training and more coherent outputs. But this realization also has further consequences: not only can we better prepare training sets for super-resolution, but we can also revise our neural network structure as well to learn an additional adaptive filtering process (not just a Gaussian one) that

helps differentiate low from high frequencies, and thus improve the accuracy of the sparse coding that our neural network performs. Moreover, our better understanding of the way training benefits from the relationship between coarse and fine patches as described in the previous paragraph suggests a *very different strategy* instead: it is, in fact, far preferable to always train the dictionary-based neural network with properly downsampled fine simulations rather than four different strategies as proposed in [Bai et al. 2020], leading to better convergence and smaller training sets. Indeed, a coarse simulator adds its own form of aliasing due to numerical errors and improperly-resolved subgrid-scale details, with both issues worsening as integration proceeds forward. Upresing should instead be trained on "clean" pairs of patches, free of such artificial deviations: training patches without spurious noise makes the training much easier, leading to better convergence; assuming the training set covers a sufficient space of simulation patches, upsampling predictions can be reliably achieved through sparse encoding even if small "extrapolations" are encountered. We will demonstrate that this observation brings significant improvements on predicting turbulent flow details, both spatially and temporally, especially when a low-resolution simulation input is fed into the network; it also drastically improves generalizability, and simplifies the use of the resulting neural network as its training always follows the same procedure, making our approach much faster and simpler to use than the approach upon which it is based.

2.3 Our New Learning Approach to Flow Upsampling

Based on our findings, we revisit the original dictionary-based neural network of Bai et al. [2020] and introduce two significant improvements as our contributions: the use of filtering (through a low-pass filtering of the input in both training and prediction, as well as an adaptive filtering within the neural network) to reduce spurious high-frequency structures which typically prevent the network from generating coherent predictions, and the introduction of an input extension (through time-stamp augmentation between neighboring frames) to provide accurate temporal upsampling. The resulting modified dictionary-based neural network, containing additional convolutional modules, provides a unified and effective neural network architecture that leads to significantly improved spatio-temporal upsampling of turbulent flows, with strong generalization ability to arbitrary fluid flow simulations even after training on a single exemplar sequence.

Low-pass filtering. As discussed in Section 2.2, low-pass filtering is needed before the input patches are processed by the dictionary-based neural network to avoid downsampling artifacts. A simple Gaussian filter is known to maintain low frequencies in the original flow field while removing high frequencies [Marr et al. 1980]. The key question is how to choose the actual value of the variance σ in the Gaussian filter, which controls the sharpness of the filter shape. To answer this question, we simply follow basic signal processing principles. When a high-resolution field is downsampled by a factor of r in each dimension ($r=4$ in most of our tests) the minimal cut-off frequency for the downsampled field is $f_c = 1/2r$ according to the Nyquist sampling theorem. Since a Gaussian filter has a theoretical cut-off frequency $f_c^g = 1/2\pi\sigma$, we slightly round up the desired

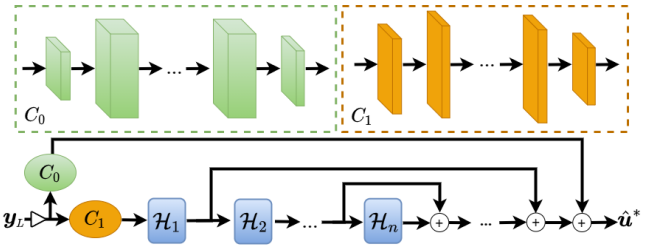

Fig. 5. **Convolutional dictionary-based neural network**. To enable better prediction, we add two convolutional modules to the original dictionary-based neural network proposed by [Bai et al. 2020]. The convolutional module C_0 uses the trilinear upsampled results from the input vector (denoted by the delta symbol) to produce $\hat{\boldsymbol{u}}_L^*$ in Fig. 2 in order to form a more representative low-frequency flow component, and the convolutional module C_1 provides a latent space that further filters the input and also makes the input more identifiable.

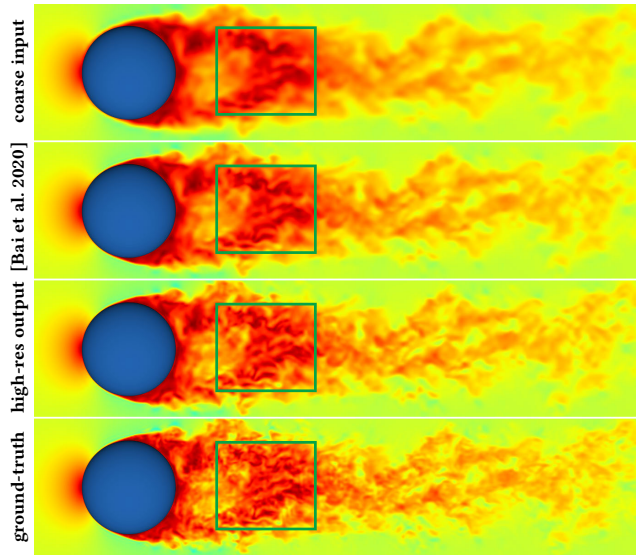

Fig. 6. **Learning with and without CNN modules**. Top row: an input velocity field, filtered by a Gaussian kernel with a variance σ determined as detailed in Sec. 2.3, is used for training. Second row: using filtered input patches, we can already predict more coherent velocity fields using the dictionary-based neural network in [Bai et al. 2020]. Third row: by augmenting their original neural network with two convolutional modules, the prediction of turbulent flow details is significantly improved, see green box. Bottom row: the corresponding ground-truth high-resolution simulation.

variance value in practice by selecting $\sigma = r/3$. By applying such a Gaussian filter before downsampling to create coarse input patches, the training patches are spatially more coherent, leading to better training convergence with improved prediction results. Note that Fig. 4 (c) uses the closest value to the optimal Gaussian variance, and indeed, it visually produces the best prediction result, with the minimal residual error out of the other three variances.

Adaptive filtering. Although very effective and straightforward, the above filtering applied to the input of the dictionary-based neural network is based on a Gaussian prior, and is quite limited. We can further improve the prediction quality by applying additional filtering inside the network which is, this time, learned from the

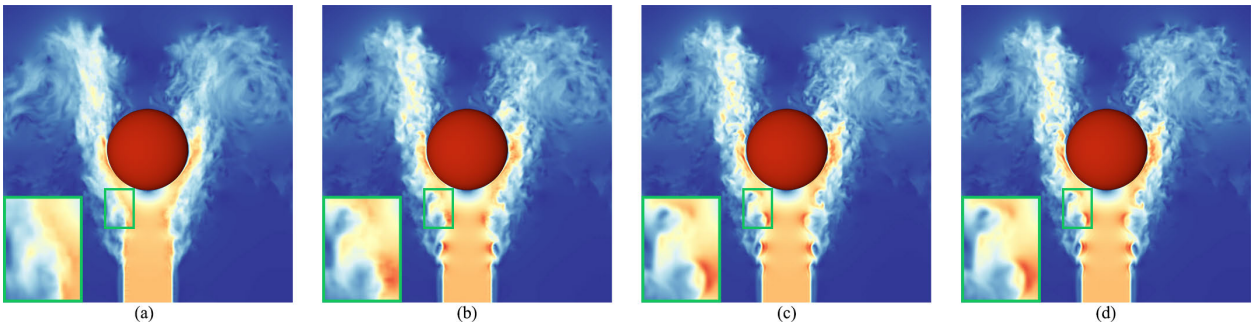

Fig. 7. **Temporal upsampling**. Using two consecutive frames of a low framerate sequence, we can properly interpolate the velocity field over time to produce intermediate frames coherently: (a) the neural network of [Bai et al. 2020] can produce in-between velocity fields through linear interpolation in time, but this results in poor prediction as compared to the ground-truth shown in (d); (b) by modifying the input structure of the neural network of [Bai et al. 2020] using the new design we propose, the prediction quality is dramatically improved, but some vortical structure are still not well preserved as compared to the ground-truth, see the green box region; (c) by using our new dictionary-based neural network with two additional convolutional modules, the prediction quality is further improved, now preserving small scale vortical structure well, see the green box region. Note that since this example is a vertical flow, we train our network using an example similar to Fig. 3 but with a *vertical* flow this time.

training set. First, we insert a convolutional neural network (CNN) module (denoted as C_0 in Fig. 5) to filter the coarse input, which is then taken as the new base flow upon which the final synthesized patch can be obtained. Note that the CNN module is a 3D CNN, where each element (a high-dimensional vector) contains the low-frequency, trilinearly-upsampled velocity fields at time t and t–1 that are derived from the input vector $\boldsymbol{y}_L$. Second, since a neural network based on a latent feature space usually leads to better prediction, we add another CNN module (with the same input construction as in C_0) to the coarse input (see C_1 in Fig. 5) in order to make the input more "identifiable;" by the neural network, leading to improved prediction result for the whole high-resolution output patch. These CNN parameters are learned along with the other network parameters during training. Fig. 6 shows an example of synthesis result for a turbulent wake flow behind a ball from a downsampled flow input (top row): while using the initial Gaussian filter described in Sec. 2.3 to the input patches already leads to realistic synthesized details (second row), the addition of CNN-based filters (third row) leads to a finer prediction that is closer to the ground-truth (bottom row); once it is used in rendering (e.g., by advecting smoke), visual richness is also enhanced.

Input structure design. For either spatial or temporal upsampling, interpolation of velocity fields is unavoidable if we directly use the original dictionary-based neural network of [Bai et al. 2020]. For spatial upsampling for instance, they first use a (tri)linear interpolation to upsample a coarse low-resolution input field to the desired high resolution before collecting patches to train the neural network for high-frequency detail prediction; in the case of temporal upsampling, they employ linear interpolation of two consecutive frames in time before collecting patches from these in-between frames and train the neural network to predict the temporal flow details. Note that this type of interpolation may not have significant influence on spatial upsampling since fluid structures remains fairly aligned between the original and upsampled fields, but can be arbitrarily bad for temporal upsampling as fluid structures can become highly distorted due to the nonlinear dynamics of fluid flows, particularly for turbulent flows. In order to obtain better temporal upsampling, we propose a new input structure for the dictionary-based neural network depicted in Fig. 5. Similar to the input augmentation proposed in [Bai et al. 2020], we also include another input parameter $\gamma \in [0, 1]$ that is *added* to the input vector $\boldsymbol{y}_L$, specifying the precise time code of any frame to be predicted between the two consecutive discrete frames; that is, the whole input vector is now changed to $\boldsymbol{y}_L = [\hat{\boldsymbol{u}}_L^t, \hat{\boldsymbol{u}}_L^{t-1}, \gamma]$. Once our new network is trained, we can then better predict an intermediate frame by specifying the corresponding γ. Note that $\boldsymbol{y}_L$ will be converted to the input of CNN modules, where each element will be augmented with the parameter γ. Note also that $\gamma = 0$ and $\gamma = 1$ correspond to a reproduction of $\hat{\boldsymbol{u}}^{t-1}$ and $\hat{\boldsymbol{u}}^t$, respectively; for a solely-spatial upsampling, we use $\gamma = 1$.

2.4 Velocity Field Upsampling

With our new learning-based approach described above, we are now ready to train our neural network to perform spatial and/or temporal upsampling of an input flow field as we describe next.

Spatial upsampling. For spatial velocity field upsampling, independent of whether we plan to use inputs that are downsampled from high-resolution simulations or directly from the output of low-resolution simulations, we always prepare the training set using patches sampled from high-resolution simulations and their filtered-then-downsampled versions, but the precise σ value in the Gaussian filter depends on the type of inputs, which will be clarified later in the description of our results. Once the training is achieved, we can now efficiently apply it to any low-resolution input field: we simply prepare an input vector for each output patch, from which our neural network will predict the added high frequency component, before assembling all the patches to form the final output velocity field. Fig. 6 shows a comparison of a 3D turbulent wake flow synthesized from a downsampled coarse velocity field (top row) that is 4 times smaller in grid samples along each dimension (effectively $4\times4\times4 = 64$ times smaller in the total number of grid nodes), where proper Gaussian filtering is applied before downsampling. It is visually striking that Bai et al. [2020] still lose details of the turbulent flow (second row) even if the input is better prepared, whereas our new model re-injects turbulence details (third row) in close agreement with the ground-truth (bottom row).

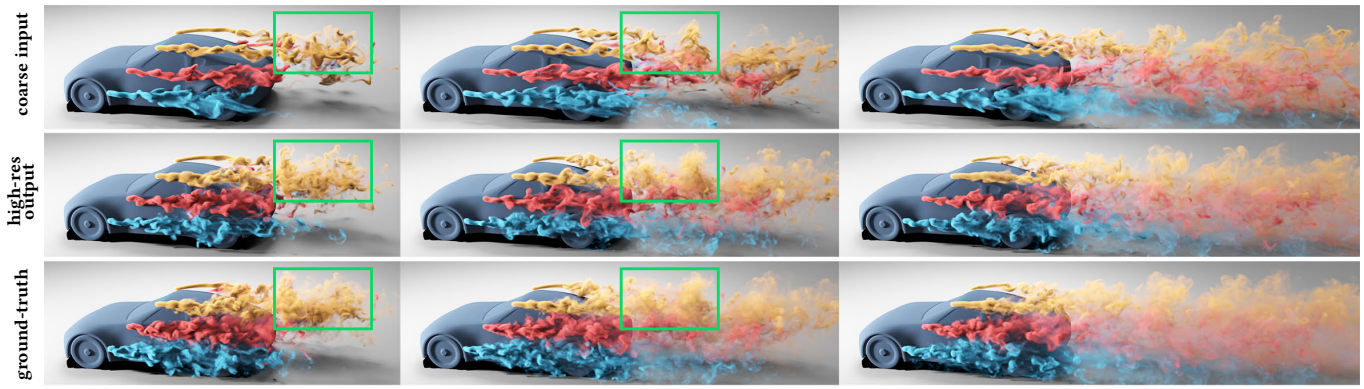

Fig. 8. **Spatial super-resolution**. In this example, we demonstrate turbulent flow spatial super-resolution, where the input flow field (top row) is a downsampled version of a high-resolution velocity field (bottom row). Training is performed using only the simulation shown in Fig. 3 and colored smoke particles are advected in the resulting vector fields for visualization purposes. Our synthesis result (middle row) shows close resemblance to the ground-truth (bottom row), indicating its ability for generalization — check out in particular the green box regions for detail preservation as compared to the ground-truth.

Temporal upsampling. For temporal velocity field upsampling, no Gaussian filtering is required since the spatial resolution is unaltered, but we must now train our network using a sequence of ground-truth velocity fields that have higher frame rates than the expected input velocity field sequence. Suppose that we aim at temporally upsampling by an integer factor k; i.e., we want to add $(k-1)$ frames between two input frames. From a high framerate ground-truth velocity field sequence (denoted $\mathcal{U}_h$) with a time interval Δt_h, we construct a low framerate velocity field sequence (denoted as $\mathcal{U}_l$) with a time interval of $\Delta t_l = k\Delta t_h$ by picking only one every k frames out of $\mathcal{U}_h$. Then, for any frame in $\mathcal{U}_h$ indexed by $i_h \in \{0, 1, 2, ...\}$, the input γ is computed as $\gamma = i_h/k - \lfloor i_h/k \rfloor$, where the two consecutive frames in $\mathcal{U}_l$ having an index of $\lfloor i_h/k \rfloor$ and $\lceil i_h/k \rceil$ are taken as the input patches $\hat{\boldsymbol{u}}_L^{t-1}$ and $\hat{\boldsymbol{u}}_L^{t}$ in $\boldsymbol{y}_L$. The network is then trained with this sequence. While the dictionary-based neural network of [Bai et al. 2020] proposed space-time encoding which could potentially be used for temporal prediction, it encodes the whole sequence and cannot be easily generalized to different input sequences. However, since our new input structure design only relies on nearby local frames, our visual results end up being far improved and with much stronger generalizability when compared to [Bai et al. 2020] as demonstrated in Fig. 7, where we show an example of a synthesized frame between two consecutive frames of an input. If we simply linearly interpolate the intermediate frame based on the γ value and take it as input, the network of Bai et al. [2020] returns the high frequency details in (a); comparing to the ground-truth frame in (d), many important vortical structures are missing (see the green box region). If we employ our new input structure design in the original network of Bai et al. [2020], we obtain much improved results as shown in (b), yet still imperfect (e.g., the vortices in green box region are not well captured when compared to the ground-truth). With adaptive filtering through a convolutional module within the new dictionary-based neural network, the prediction in (c) is further polished, preserving the evolution of small-scale vortices far better.

Spatio-temporal upsampling. Naturally, our learning framework can also support both spatial and temporal upsampling at once. To achieve this general case, we first decompose the upsampling process into 1) a spatial upsampling from the coarse (low spatial resolution) flow, followed by 2) temporal upsampling of the spatially-upsampled velocity fields. These two upsampling processes being independent, they can be decoupled to improve the efficiency of the learning phase and of the synthesis phase. Note that we have used two neural networks with the same architecture, but only differing in input and output patches as well as γ values that are used during training; these two networks are trained separately as described in the two paragraphs above, using training patches that are collected from the same high-resolution simulation. Fig. 1 shows a spatio-temporal upsampling result on a very complex geometry (a high-rise building) with an output grid resolution of 800×320× 320, which was achieved by directly applying the two networks after training on a single jet flow simulation through a ball obstacle (with a resolution of $600 \times 320 \times 320$) shown in Fig. 3: despite the vast difference in geometry between these two examples, our learning-based approach provides a very detailed upsampling of turbulent flows in both space and time of this particular input.

2.5 Fluid Data Compression

In addition of spatio-temporal upsampling, our learning-based approach to predicting high-resolution flow details is also very useful for the compression of time-varying flows, e.g., to efficiently store on disk time sequences of discrete velocity fields issued from large-scale numerical simulations of fluid flows: our proposed dictionary-based neural network can be seen as a patch-based encoder of high-frequency details over both space and time. To achieve spatial compression, one simply applies Gaussian filtering (without downsampling) to an input high-resolution simulation to turn the original velocity field into a low-frequency field, which can then be more efficiently compressed using the traditional wavelet-based approach; in addition, all the parameters Θ of our neural network serve as an encoding of the high-frequency components. Now for temporal compression, we can just downsample the whole fluid sequence to a lower frame-rate, e.g., keeping only one frame out of ten, and save all the network parameters for a decoder to be able to predict the intermediate frames based on the key frames that

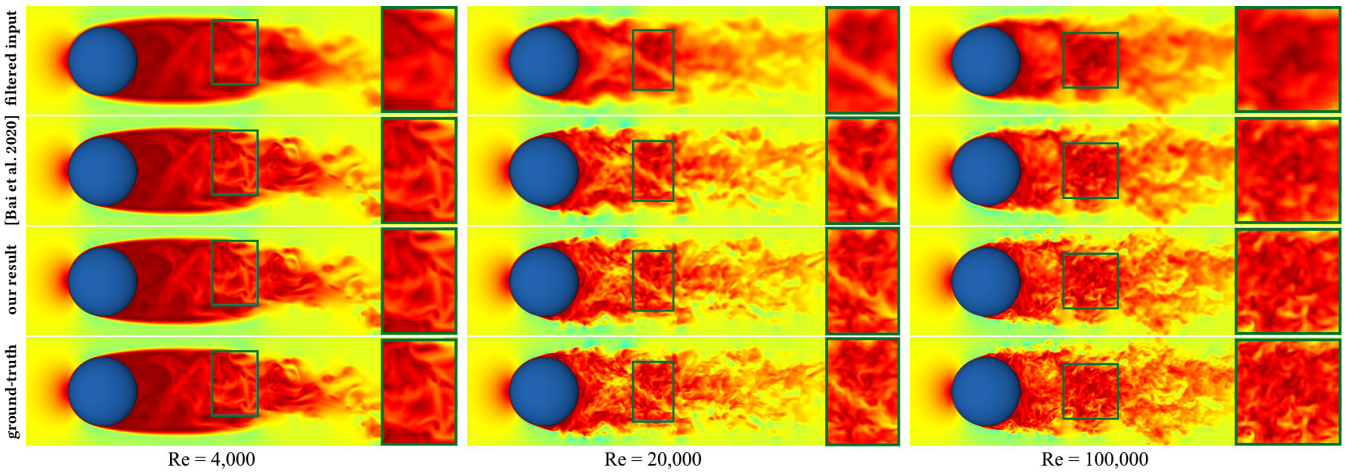

Fig. 9. **Fluid data compression at different Reynolds numbers**. Top row: the input velocity fields are filtered with a Gaussian kernel. Second row: the fluid data reconstruction using the original neural network proposed by [Bai et al. 2020] using these filtered fields as input. Third row: the fluid data reconstruction using our new proposed method for the same input. Bottom row: ground-truth frames from a fine simulation. Clearly, our new method preserves turbulence details very well, leading to better reconstruction especially for flows at higher Reynolds numbers. The compression ratios and relative reconstruction errors are as follows: for Re = 4,000 with a compression ratio of 160, the relative reconstruction error for [Bai et al. 2020] is 0.509% while our method's is 0.486%; for Re = 20,000 with a compression ratio of 80, the relative reconstruction error for [Bai et al. 2020] is 1.88% while our method's is 1.13%; for Re = 100,000 with a compression ratio of 50, the relative reconstruction error for [Bai et al. 2020] is 2.2% and ours is 1.63%. Note that the relative errors are measured with the ℓ_2-norm, where small differences may indicate large deviation in small-scale turbulence structures.

are spatially encoded. By saving only the wavelet coefficients of the filtered velocity field at low-frame rate along with the network parameters for both spatial and temporal upsampling, we effectively offer an efficient learning-based codec (encoder/decoder) for fluid flows. Note that the parameters Θ could have been trained using an exemplar simulation, different from the velocity field to encode; if high-quality compression is desired, one could also re-train these default parameters using the actual sequence to compress (thus increasing compression times) in order to improve the rate distortion function, as we will demonstrate in Sec. 3.

Ablation tests show that our learning-based codec offers only marginal improvement compared to a direct wavelet encoding when a single frame is encoded: we get on average only 1.5 times better compression rates for equivalent reconstruction errors at the Reynolds number of 100,000. However, the situation is drastically different as soon as a time-varying velocity field sequence needs to be compressed. For turbulent flows, our compression rates are usually improved by an order of magnitude compared to a frame-by-frame wavelet compression for the same reconstruction error, with even higher rates for low Reynolds number flows. Although wavelet-based compression combined with motion compensation can improve compression ratio as frequently leveraged in video compression [Metkar and Talbar 2010], this improvement is typically small in the case of turbulent flows due to the existence of chaotic high-frequency structures that are poorly amenable to motion compensation. Note that the original approach by Bai et al. [2020] did not exploit their approach for compression, partly due to lacking sufficient accuracy and generalizablity; but because our new approach dramatically improves accuracy for both spatial and temporal upsampling while being generalizable, we will show in Sec. 3 that compression rates are much better with our new improved dictionary-based neural network than what they could have achieved — see Fig. 9 comparing flow field reconstructions at different Reynolds numbers (using only spatial compression since Bai et al. [2020] does not support temporal compression) for the same compression ratios, demonstrating the quality of our codec.

3 RESULTS AND DISCUSSIONS

We now discuss the specific implementation of our learning-based fluid flow synthesis algorithm, before reviewing results on a variety of application scenarios and comparisons.

3.1 Implementation Details

For reproducibility and clarity, we discuss a few crucial details first.

Training set. First, in order to show the superiority of our novel approach to upsampling through a dictionary-based neural network, most of the results shown in this paper were synthesized based on the training patches issued from a *single* high-resolution simulation of a flow around a ball (see Fig. 3), whose time-varying velocity field was provided by Li et al. [2020] and computed at a grid resolution of $600\times320\times320$. We use a downsampling ratio of 4 in each dimension to prepare the low-resolution input. While many previous works demonstate results based on training of very similar simulations, our method learns very fine turbulent flow details from a single exemplar simulation, and manages to properly reinject a similar type of high-resolution vortical details on coarse inputs representing very different flows with different obstacle geometries. Of course, our method can be trained with more training examples covering different types of fluid flows that are put altogether as an entire large training set, if needed, to further improve prediction accuracy and generalizability; it can also use any grid-based fluid flow solver as inputs as nothing in our approach is solver-specific, see Fig. 15. Once a training set is ready, we sample patches from equally-spaced frames (in our tests, we use 100 equally-spaced frames) over the

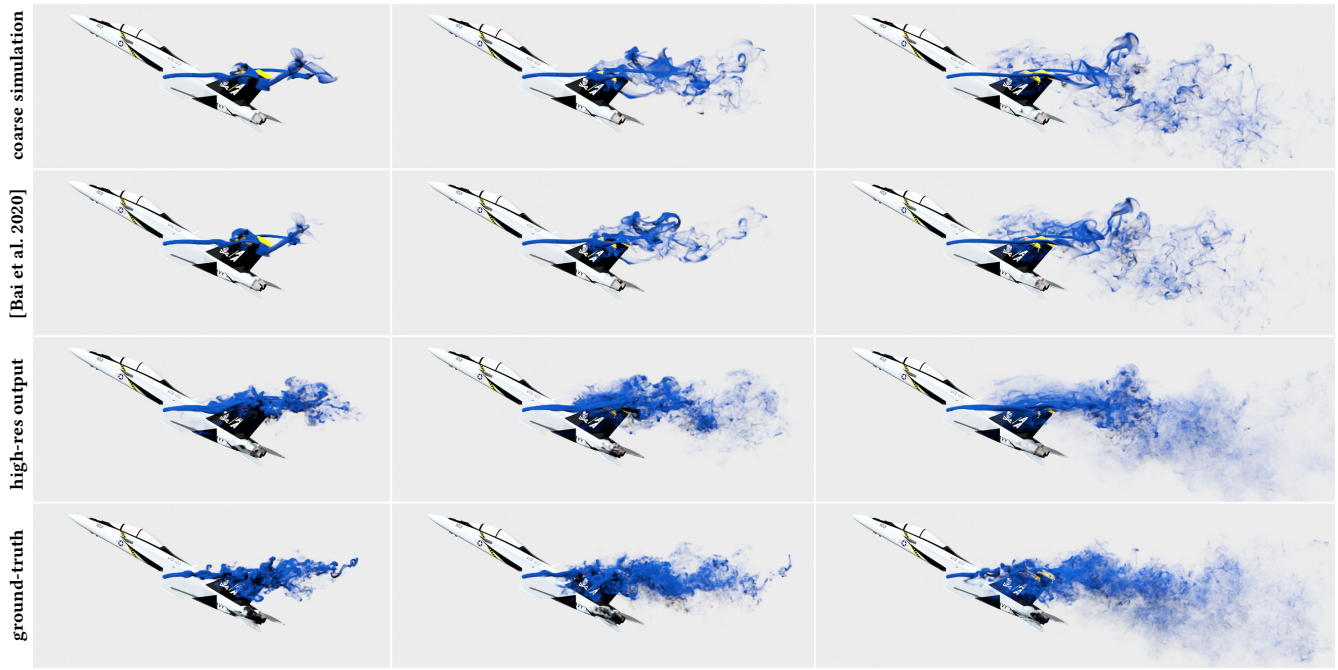

Fig. 10. **Spatial Upsampling**. In this example, we demonstrate turbulent flow spatial upsampling where the input flow field (top row) is a low-resolution simulation. Training is performed using only the simulation sequence shown in Fig. 3, and our synthesis result (third row) shows much closer resemblance to the ground truth (bottom row) compared to [Bai et al. 2020] (second row), indicating a strong capacity for generalization.

whole fluid flow sequence, and select 4M patches via importance sampling from those frames as used in [Bai et al. 2020] to train our new dictionary-based neural network for prediction.

Network setting and training. The convolutional module C_0 (see Fig. 5) has three layers, all using a kernel with a size of 3, and has the following specific settings: the first layer has an input/output dimension of 7/16 with a stride of 2; the second layer has an input/output dimension of 16/32, with a stride of 1; and the third layer has an input/output dimension of 32/3, with a stride of 1. The activation functions for the first two layers are both hyperbolic tangent (tanh) whereas the third layer has a linear activation function. The convolutional module C_1 has only two layers, which are constructed as follows: the first layer has an input/output dimension of 7/5 with a kernel size of 3, a stride of 2 and a linear activation function, while the second layer is a fully-connected network whose output dimension is 625, which is the latent space vector. Our hyper-parameters of the loss function are chosen differently from [Bai et al. 2020]: we pick $\alpha_\ell = 1.0$, $\alpha_g = 0.1$, $\alpha_d = 0.1$ and $\alpha_\Theta = 10^{-5}$. Note that the regularization parameter α_Θ is very small compared to theirs: we just need to ensure that no network parameters having large absolute values are learned, in order to prevent overshooting; larger regularization may, in fact, introduce smoothing in the result. Due to improved coherence, our new network does not require the type of strong regularization that Bai et al. [2020] used, hence preserving high-frequency details in the output. Finally, we initialize our network parameters with random numbers that are generated from a normal distribution, and the Adam optimization algorithm [Kingma and Ba 2014]is employed to train the network with a learning rate of 10^{-4}. We use 90% of the training patches for learning and the remaining 10% for validation. The network batch size is set to 4096, and we stop training when the validation loss no longer decreases during one epoch, which usually takes up to two million iterations.

System configuration. Our learning algorithm was implemented with TensorFlow [Abadi et al. 2016] and trained on a server with an Intel(R) Xeon(R) E5-2620 v4 @ 2.10GHz CPU, with 256GB RAM, and an NVIDIA P40 GPU with a total memory of 24GB. The synthesis process was implemented on a normal workstation equipped with an NVIDIA RTX3090 GPU with 24GB of memory. Spatial training takes approximately 3 days to finish, while temporal training takes approximately 5 days; note that prior work often report more than a week of training in comparison, see [Xie et al. 2018; Fukami et al. 2021] for instance. Synthesis usually takes 6 to 90 seconds per frame depending on the output resolution. Table 1 provides statistics for the various synthesis results shown in this paper.

Rendering. Aside from the 2D cross-section speed-colored visualizations of the high-resolution vector fields we generate, we also use these fields to advect smoke particles and use the GPU-based Redshift rendering software [Maxon 2021] for final rendering, which takes between 2 to 10 minutes per frame depending on the grid resolution and the complexity of the scene.

3.2 Synthesis Results

We now review a series of concrete results based on our learning approach for various applications, with smoke rendering or cross-section velocity field visualization to show our synthesis results.

Spatial super-resolution. Spatial super-resolution of a flow field refers to the spatial upsampling of a velocity field which was down-sampled from the original high-resolution counterpart (to reduce

storage size for instance). As mentioned in Sec. 2.4, training pairs for all of our super-resolution examples were prepared from the same high-resolution simulation of a jet flow blowing over a ball (for a Reynolds number of 20,000) as depicted in Fig. 3 and their downsampled versions after Gaussian filtering. Fig. 8 shows snapshots of our spatial super-resolution result for a simulation of the turbulent air flow around a car for a Reynolds number of 20,000, where our trained network was directly applied to the downsampled velocity field sequence (top row) to obtain the synthesized result (middle row). Very detailed smoke patterns statistically consistent with the ground-truth original simulation (bottom row) are observed.

Spatial upsampling. Spatial upsampling of a flow field takes a low-resolution fluid simulation sequence as input and returns a high-resolution output sequence with fine details added. As argued earlier in Sec. 2.2, we remove the impact of most-probably erroneous subgrid scale details in the training process by assembling training pairs from the high-resolution simulation in Fig. 3 and their downsampled versions, but using a Gaussian filter with a larger parameter σ (e.g., $\sigma = r$ in our experiment): as the mean flow is usually not perfectly resolved in a low-resolution simulation of a turbulent flow, it is better to prepare an input flow field that is more aggressively smoothed to mimic a directly downsampled velocity field using the Gaussian filter described in Sec. 2.3. This depends, however, on how accurate the turbulence model that the low-resolution simulation employs is. If the turbulence model can predict the downsampled velocity field accurately, σ can still be chosen to be $r/3$ or slightly above. Fig. 10 shows the spatial upsampling result (third row) with a Reynolds number of 50,000, where our trained network was directly applied to a flow passing through a very complex geometry (here, an F18 fighter airplane). Compared to [Bai et al. 2020] (second row), turbulent flow details are much better preserved as can be observed when compared to the ground-truth (bottom row).

Temporal upsampling. Our dictionary-based neural network excels at temporal upsampling, where a sequence of three-dimensional velocity fields with a relatively low frame-rate (e.g., 30 fps in our test) is refined into another sequence of velocity fields with a higher frame-rate (e.g., 450 fps). Here also, we only use the high-resolution simulation from Fig. 3 (computed at two different frame rates) to prepare the training set in the manner introduced in Sec. 2.4. Fig. 11 shows a temporal upsampling result for a plate falling in the center of a static smoke ring to generate a complex turbulent wake flow, where the snapshots marked by the red boxes are from the original low frame-rate sequence, while all the other snapshots are synthesized by our network; note the temporal coherence of the synthesized sequence. In Fig. 12, we compare our approach to the state-of-the-art temporal upsampling work of [Jiang et al. 2018], which is only a 2D upsampling method rather than a full 3D synthesis approach like ours. Our supplementary video shows very obvious temporal artifacts engendered by [Jiang et al. 2018] which are difficult to evaluate just from still frames; instead, our network predicts 3D intermediate fields with near perfect preservation of vortical structures. In addition, even though [Jiang et al. 2018] could technically be extended to 3D, it is a *global* learning approach, which considerably limits its applicability, while our patch-based approach scales nicely to high-resolution turbulent flow field synthesis.

Spatio-temporal upsampling of a turbulent flow. Due to its improved treatment of spatial upsampling and its excellent ability to also interpolate between frames, our learning-based approach also excels at spatio-temporal upsampling, where a spatial network and a temporal network are trained separately but applied in concert to a space-and-time-coarse input: given a low-resolution input velocity field sequence with low frame-rate, we can first upsample the velocity field spatially using the spatial upsampling network, and then temporally based on the spatially upsampled low frame-rate sequence using the temporal upsampling network. Fig. 1 shows the spatio-temporal upsampling for a very complex turbulent flow through a large high-rise building. High-frequency turbulent flow details, well structured and far richer than typical wavelet-based noise due to their emergence from the actual fluid motion, can be observed both spatially and temporally; here again, still frames do not really do justice to the temporal coherence of our upsampled result, so we refer the reader to our supplementary video.

Compression of time-varying vector fields. Finally, our method handles fluid flow compression, and performs particularly well for lossy compression of vector fields corresponding to turbulent flow simulations containing high-frequency chaotic structures. While the previous applications we demonstrated thus far do not necessarily need very accurate synthesis result as visual plausibility is paramount, accuracy is usually important in fluid flow data compression: terabytes or even petabytes of raw data can be generated by a high-resolution direct numerical simulation (DNS) of fluid flows, requiring good codecs to store efficiently without losing important details. As a consequence, training using only the flow field shown in Fig. 3 may not be sufficient, especially for temporal prediction. In order to improve accuracy, we need either to increase the training set by involving more simulation scenarios (thus requiring more computational resources and time), or to refine the network for the specific compression task by *retraining* the network, using the existing trained network parameters as initialization. Note that we can either retrain both spatial and temporal upsampling networks or only retrain the temporal upsamling network; we adopt the second approach in our examples. Fig. 13 compares the fluid data compression using the traditional wavelet approach and our refined network, where we used an air flow simulation around a simpler car to retrain the network. (Note that we could use the actual F1 simulation to retrain the temporal upsampling process; we avoided using the same sequence to show how general our approach is in practice: we only need to retrain the model using a *similar type* of animation). The whole encoding, i.e., wavelet coefficients for Gaussian filtered fields and network parameters for spatio-temporal prediction, results in a compression ratio of 600 compared to the actual uncompressed input fluid flow sequence (last row), with a relative reconstruction error of only 3% in terms of ℓ_2-norm for our method (third row); a similar reconstruction error using lossy wavelet compression only achieves a compression ratio of 60 (second row), indicating a 10× improvement in compression. Note finally that wavelet encoding with a compression ratio of 600 results in a relative reconstruction error of 7% instead in terms of ℓ_2-norm, as it misses many turbulence details (top row), which clearly demonstrates the efficiency of our approach to compressing turbulent fluid flow datasets.

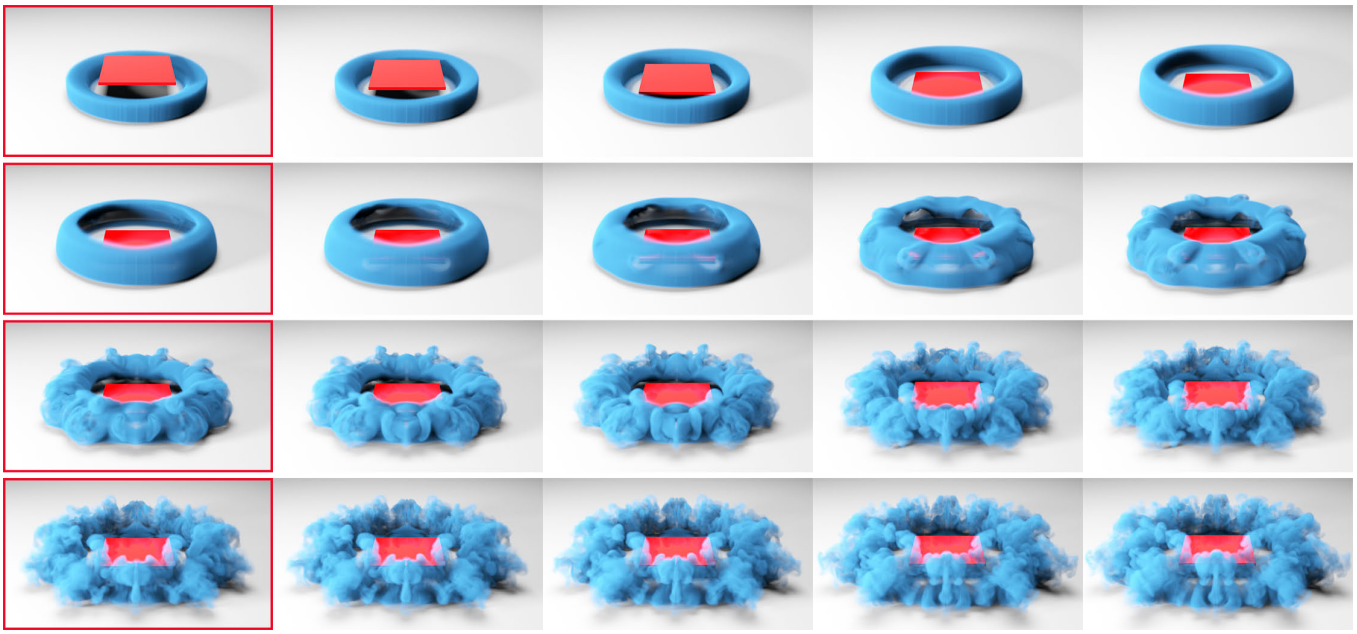

Fig. 11. **Temporal upsampling**. In this example, we demonstrate turbulent flow temporal upsampling, where the input flow field frames (left column, surrounded by red boxes) are a low-frame-rate velocity field sequence. Training is performed using only the simulation shown in Fig. 3, and our temporal synthesis results (all snapshots without a red border) show very smooth transition between the input frames. Our supplementary video shows a comparison to the ground-truth animation so that the synthesis quality can be better assessed.

3.3 Discussions

There are several important aspects related to our learning algorithm that deserve further discussions based on our practical experience.

Prediction accuracy. In order to further demonstrate higher accuracy compared to the original dictionary-based approach of Bai et al. [2020], we also plotted in Fig. 14 the energy spectra of the synthesized high-resolution flow fields compared to their ground-truth versions: it confirms that our approach predicts high-frequency details of turbulent flows far better. Note in particular that while their original method improves quite a bit when we properly filter its input (a fact that we reported in Sec. 2.2 as the motivation behind our novel approach), our other changes and addition to their approach brings about a spectrum that matches the ground-truth one much closer. After experimenting with our approach, we recommend the use of turbulence models such as [Alfonsi 2009; Liu et al. 2015] in low-resolution simulation of flows: since spatial upsampling relies on the "mean" flow provided as an input, the better the low-resolution simulation is able to stay close to a downsampled version of the equivalent high-resolution simulation, the more accurate the synthesized result will be. Of course, if visual plausibility is more important than physical accuracy, then our approach can upsample coarse simulations from any fluid solver. In addition, we found that Gaussian filtering together with CNN modules have higher prediction accuracy than CNN modules alone.

Generalization. Our new learning-based approach for predicting turbulent flow details exhibits far improved generalizability than prior work: one can train the network with a particular fluid flow simulation and apply the trained network to a very different style of simulation. Recall that Figs. 1, 8, 10 and 11 were all synthesized with the spatial and temporal training based on the dataset prepared from a single high-resolution fluid flow simulation, shown in Fig. 3; these are rather extreme cases of generalization compared to the reported results in many state-of-the-art methods, particularly for more physically consistent synthesis [Um et al. 2020; Fukami et al. 2021], for which a specific training must be performed for each different type of simulation scenarios. While these results show that any training can already enable plausible synthesis, training examples that are closer to the simulation to upsample are obviously able to further improve accuracy — in particular, for temporal upsampling: this is why we switch the training exemplar for Fig. 7 to another upward flow over an obstacle (the usual training exemplar from Fig. 3 does not fully capture the type of vertical flow present

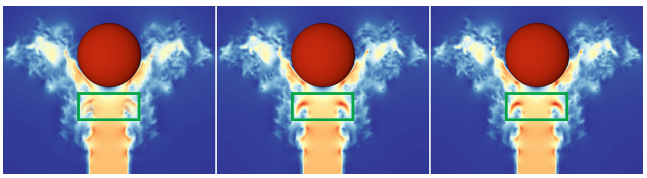

Fig. 12. **Comparison for temporal upsampling**. We apply the recent work of [Jiang et al. 2018] to upsample in time the rendered cross-section velocity field image in 2D (left) and compare the rendered cross-section velocity field from our 3D temporal synthesis (middle). Compared to the groundtruth (right), the method of [Jiang et al. 2018] show inconsistent temporal prediction, creating artifacts as shown in the green rectangle box, while our method always has consistent temporal prediction. Our supplementary video demonstrates these temporal artifacts more obviously.

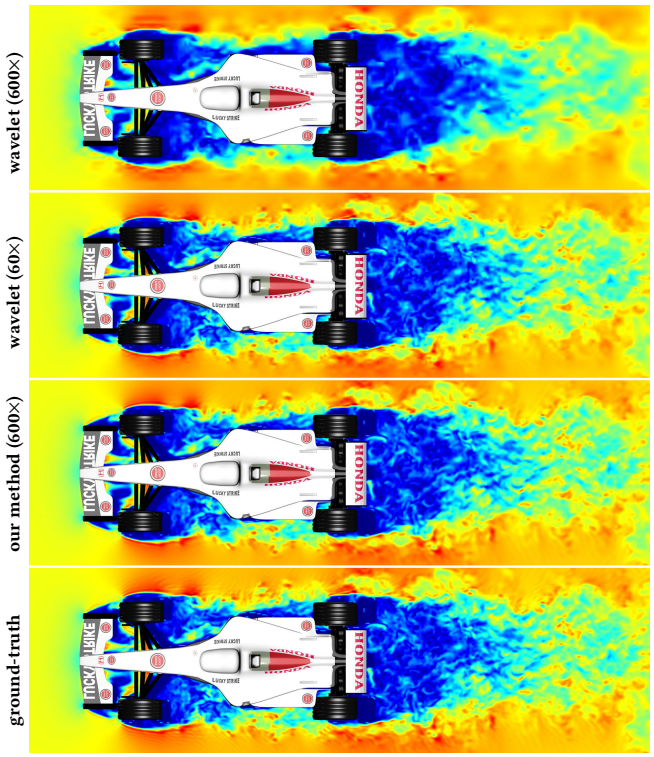

Fig. 13. **Fluid data compression**. We compare our learning-based compression with a traditional wavelet-based compression. Given a high-resolution velocity field sequence (bottom row), we can compress the whole sequence 600× with a relative reconstruction error of 3% (third row); a similar reconstruction error using wavelet compression only leads to a compression ratio of 60 (second row). To reach a compression ratio of 600 (first row), wavelet compression has now a relative reconstruction error of 7%. The error is measured in the ℓ_2-norm, and note that a small increase in this error measurement may dramatically degrade the reconstruction quality visually, as compared to the input ground-truth sequence.

in Fig. 7), and why we used a typical car simulation (using a simpler vehicle than the F1 car) to refine our temporal synthesis network in the compression example of Fig. 13. This possibility for added accuracy through more targeted training could be useful in a wide range of fluid flow synthesis applications, from editing turbulence details for artistic control to more accurate turbulence synthesis for fast computational design. Note that when the training examples are too different from the upsampling scenario, blocking artifacts may occur in the synthesized high-resolution velocity field. However, these issues are effectively filtered out when particle tracing for smoke visualization is performed: this is why the temporal synthesis of a vertical motion in Fig. 11 using the horizontal flow around a ball from Fig. 3 for training still leads to a visually pleasing smoke temporal interpolation.

Filter selection. Recall that the Gaussian filter parameter σ has different roles in different types of application scenarios: for spatial super-resolution, the variance σ should be selected in the way described in Sec. 2.3; for spatial upsampling, σ should be larger to compensate for inaccurate frequencies of low-resolution simulation, but its precise value depends on how accurate the turbulence model

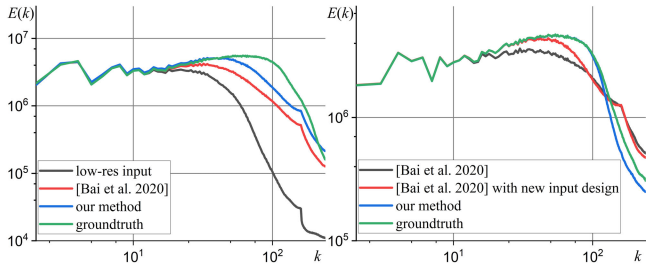

Fig. 14. **Comparison of energy spectra**. Here, we plot the energy spectra for both spatial super-resolution (left) shown in Fig. 6 and temporal upsampling (right) shown in Fig. 7. It is apparent that both spatial super-resolution and temporal upsampling show better spectrum distribution than [Bai et al. 2020], indicating more accurate high-frequency structure prediction. In temporal upsampling, even though the method of [Bai et al. 2020] is improved with our new input design to avoid direct linear interpolation, it does not show as good of a match in spectrum as our entire learning approach, which is a very close fit to the groundtruth spectrum.

used in low-resolution simulation (if any) is. In practice, we select the variance within the range $\sigma \in [r/3, r]$ for an upsampling factor of r in each dimension to prepare the training set, where higher values are used for very inaccurate inputs; this value can also be considered as a valuable tool for artistic control of fluid flow details. In addition to Gaussian filter, there are also other filters that could potentially be used. We experimented with some of them, such as Mitchell [Mitchell and Netravali 1988] and windowed sinc [Presti 2000] filters, as they are often better at preserving structures in the input in general, but our tests showed very limited improvement; so we kept a simple Gaussian filter.

Performance. Due to its use of local velocity field patches, our method is particularly amenable to the synthesis of high-resolution turbulent flows. Many of our synthesized velocity fields have very high resolutions — for instance, Fig. 10 where the grid is $1000 \times 400 \times 400$. Even with the fastest GPU-based turbulent flow solvers currently available [Li et al. 2020; Chen et al. 2021], a direct high-resolution numerical simulation requires about 11 minutes per frame, see Table. 1. With our learning-based approach, we achieve an equally compelling result for smoke synthesis in almost an order of magnitude shorter time. Moreover, this resolution is very difficult to achieve for many learning-based approaches currently available in the literature, as most of them adopted global learning approaches; ours is much more scalable to large grid sizes. Note that our current timings do not use specific GPU optimizations, so performance could be further improved with a more careful implementation.

Applicability to other fluid solvers. Although all the above results were obtained using training and input sequences computed via a kinetic-based fluid solver provided by [Li et al. 2020], our method is not restricted to any specific type of fluid flow solvers. In Fig. 15, we show an example where both the training sequence and the coarse low-res input (left) were generated by a recent Navier-Stokes solver (the reflection-advection MacCormack solver of [Zehnder et al. 2018]). The high-res turbulent smoke result (middle) exhibits the same type of plausible smoke details as the ground-truth simulation (right), proving that our approach is mostly agnostic to the solver used to generate simulation data.

Table 1. **Statistics and performance.** We provide in this table the parameters and timings per frame for various flow field synthesis examples shown in this paper. Note that all the simulations are computed using [Li et al. 2020] since their method currently offers the fastest and accurate GPU-based turbulent flow solver in graphics, to demonstrate our efficiency even when compared to their timings.

Figs	input type	σ	k	input resolution	output resolution	Re	low-res preparation time	high-res simulation time	prediction time	speed-up
Fig. 1	low frame rate downsampled input	1.4	10	200×80×80	800×320×320	50,000	n/a	256.9 sec.	44.3 sec.	5.8
Fig. 6	downsampled input	1.4	n/a	150×80×80	600×320×320	20,000	n/a	170.4 sec.	33.9 sec.	5.0
Fig. 7	low frame rate input	0.0	10	320×480×320	320×480×320	20,000	n/a	140.6 sec.	28.5 sec.	4.9
Fig. 9 (left)	Gaussian filtered input	2.0	n/a	600×320×320	600×320×320	4,000	n/a	170.4 sec.	33.9 sec.	5.0
Fig. 9 (middle)	Gaussian filtered input	2.0	n/a	600×320×320	600×320×320	20,000	n/a	170.4 sec.	33.9 sec.	5.0
Fig. 9 (right)	Gaussian filtered input	2.0	n/a	600×320×320	600×320×320	100,000	n/a	170.4 sec.	33.9 sec.	5.0
Fig. 8	downsampled input	1.4	n/a	180×60×60	720×240×240	20,000	n/a	154.1 sec.	20.4 sec.	7.6
Fig. 10	low-res simulation input	4.0	n/a	250×100×100	1000×400×400	50,000	6.1 sec.	678.3 sec.	93.2 sec.	6.8
Fig. 11	low frame rate input	0.0	15	320×160×320	320×160×320	30,000	n/a	59.7 sec.	6.8 sec.	8.8
Fig. 13	low frame rate Gaussian filtered input	2.0	10	600×200×250	600×200×250	100,000	n/a	93.5 sec.	13.7 sec.	6.8

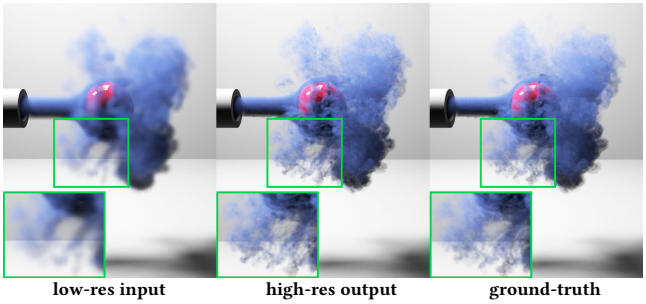

Fig. 15. **Applicability to semi-Lagrangian solver.** While most our results were generated and trained from the output of a kinetic fluid solver, we tested training and prediction using a Navier-Stokes solver (here, the MacCormack solver of [Zehnder et al. 2018]). Left: a low-resolution input; middle: our synthesis result using a training sequence still based on a turbulent flow passing through a ball depicted in Fig. 3; right: ground-truth simulation. Each image represents a simulation result produced by advecting smoke density using the corresponding velocity field.

Comparison with other fluid flow synthesis approaches. A few other fluid flow synthesis approaches related to our work in graphics deserve further discussions. For super-resolution, tempoGAN [Xie et al. 2018] is also a patch-based approach which can achieve relatively strong generalization. However, [Bai et al. 2020] showed that its ability to handle turbulent flows is very limited. In addition, it synthesizes smoke density rather than a velocity field, which was shown in [Forootaninia and Narain 2020] to lead to a smoke visual complexity no better than a direct trilinear velocity field upsampling followed by smoke advection. We argue that synthesizing velocity field for smoke animation is always better than directly synthesizing smoke density field as the velocity field is richer in details, enabling very complex smoke motions; smoke advection will in fact automatically filter small errors in the velocity field, producing a very coherent smoke synthesis result. Fig. 16 shows a comparison of smoke synthesis between tempoGAN and our neural network, proving that more faithful fine turbulent flow details can be achieved with our method; note that our training set is still based on the flow simulation from Fig. 3. Regarding smoke detail addition, [Forootaninia and Narain 2020] proposed a simple approach through frequency-domain synthesis. However, this recent work becomes very slow for high-resolution flow field synthesis. In addition, the authors point out that local editing of fluid details cannot be achieved. Finally, these two related works offer neither accuracy in their synthesis nor temporal upsampling, and are thus unlikely to be applicable to fluid data compression.

3.4 Limitations

Despite greatly improved upsampling results, our method still suffers several limitations. First, while we demonstrated that spatial upsampling can benefit even from a training set quite distinct from the type of simulation scenario to upscale, temporal synthesis still exhibits only relatively limited generalizability; even if we offer much improved results over previous works, there likely exists a better representation of the dynamics which would produce plausible temporal upscaling even from quite unrelated training data. Second, if physical accuracy is a must, a coarse solver that takes turbulence modeling into account is required: otherwise, the mean flow that we feed to our network is simply too off to expect that our upsampling approach would result in the correct physical behavior in practice. Third, since dictionary patches are learned from

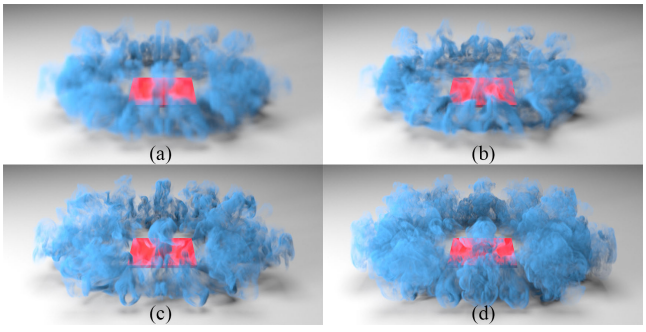

Fig. 16. **Comparison with tempoGAN.** Given a low-resolution one-way coupled fluid-solid simulation where a falling solid plate is creating a turbulent flow (a), we synthesize high-resolution fluid flow simulations using tempoGAN [Xie et al. 2018] (b) vs. our dictionary-based neural-network (c) trained from the simulation in Fig. 3. Compared to the ground-truth (d), our neural network captures better and finer structures.

the training patches selected via importance sampling, some input sequences may contain local regions that are far from a sparse interpolation of dictionary patches, hence leading to bad "extrapolation". Obviously, increasing the number and variety of training patches as well as the number of input sequences can remedy this issue, but at the price of extended training time and larger memory size. Finally, due to our added CNN modules, our flow synthesis time ends up being slightly slower than [Bai et al. 2020], but with significantly improved high-resolution flow synthesis.

4 CONCLUSION

In this paper, we have introduced a simple, yet effective learning-based approach to predicting high Reynolds number turbulent flow details. Our approach is not only more accurate than the state-of-the-art patch-based method by [Bai et al. 2020], but also much more generalizable for both space and time upscaling. Its core idea relies on a sparse dictionary-based encoding using small patches to make the upscaling very scalable, and from an explicit and adaptive filtering strategy that not only provides more coherent input data with reduced aliasing, but also produces a latent feature space that makes the input patches more identifiable. In addition, our new network design also facilitates temporal prediction of flow details by avoiding unnatural and aliasing-prone temporal linear interpolation. The two neural networks (a spatial upsampling network and a temporal upsampling network) of our approach were trained and applied to a variety of turbulent flow synthesis applications, hinting at the wide applicability of our technique. Finally, comparisons with state-of-the-art techniques in various applications were performed to highlight many advantages of our method.

We wish to pursue a number of future work. We believe that finding another approach to temporal prediction without time stamps is worthy of investigation: obtaining the same level of generalizability in time as in space would open up an even larger range of applications. Applying some of our compression ideas for videos or time-varying MRI datasets may also be fruitful. Finally, deriving a low-resolution solver which, once paired with our network, can produce high-resolution simulation accurately but at a fraction of the time that a DNS solution requires is another exciting direction of research to explore.

ACKNOWLEDGMENTS

The authors would like to thank all reviewers for their constructive comments on shaping this paper. This work was supported by the National Natural Science Foundation of China (No. 62072310 and No. 61976138) as well as ShanghaiTech University. M. Desbrun gratefully acknowledges generous support from Inria, Pixar Animation Studios and Ansys Inc. 3D meshes were provided by MFC at 3DWarehouse and shivakeswani at cgtrader (Fig. 1), printable_models at free3d (Fig. 8), pjedvaj at cgtrader (Fig. 10), and Romain Perera at TurboSquid (Fig. 13).

REFERENCES

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: a system for large-scale machine learning.. In *OSDI*, Vol. 16. 265–283.

Giancarlo Alfonsi. 2009. Reynolds-averaged Navier–Stokes equations for turbulence modeling. *Applied Mechanics Reviews* 62, 4 (2009).

Kai Bai, Wei Li, Mathieu Desbrun, and Xiaopei Liu. 2020. Dynamic Upsampling of Smoke through Dictionary-Based Learning. *ACM Trans. Graph.* 40, 1, Article 4 (Sept. 2020), 19 pages. https://doi.org/10.1145/3412360

Robert Bridson, Jim Houriham, and Marcus Nordenstam. 2007. Curl-Noise for Procedural Fluid Flow. *ACM Trans. Graph.* 26, 3.

René-Daniel Cécora, Rolf Radespiel, Bernhard Eisfeld, and Axel Probst. 2015. Differential Reynolds-stress modeling for aeronautics. *AIAA Journal* 53, 3 (2015), 739–755.

Yixin Chen, Wei Li, Rui Fan, and Xiaopei Liu. 2021. GPU Optimization for High-Quality Kinetic Fluid Simulation. *IEEE Trans. Vis. Comp. Graph.* (2021).

Jack Chessa and Ted Belytschko. 2003. An extended finite element method for two-phase fluids. *J. Appl. Mech.* 70, 1 (2003), 10–17.

Mengyu Chu and Nils Thuerey. 2017. Data-Driven Synthesis of Smoke Flows with CNN-Based Feature Descriptors. *ACM Trans. Graph.* 36, 4, Article 69 (July 2017), 14 pages.

P. Clark di Leoni, P. J. Cobelli, and P. D. Mininni. 2015. The spatio-temporal spectrum of turbulent flows. *Eur. Phys. J. E* 38, 136 (2015), 1–10.

Fernando de Goes, Corentin Wallez, Jin Huang, Dmitry Pavlov, and Mathieu Desbrun. 2015. Power Particles: An Incompressible Fluid Solver Based on Power Diagrams. *ACM Trans. Graph.* 34, 4, Article 50 (July 2015), 11 pages.

Alessandro De Rosis. 2017. Nonorthogonal central-moments-based lattice Boltzmann scheme in three dimensions. *Physical Review E* 95, 1 (2017), 013310.

Karthik Duraisamy, Gianluca Iaccarino, and Heng Xiao. 2019. Turbulence Modeling in the Age of Data. *Annual Review of Fluid Mechanics* 51, 1 (2019), 357–377. https://doi.org/10.1146/annurev-fluid-010518-040547

EA Fadlun, Roberto Verzicco, Paolo Orlandi, and J Mohd-Yusof. 2000. Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations. *J. Comput. Phys.* 161, 1 (2000), 35–60.

Ronald Fedkiw, Jos Stam, and Henrik Wann Jensen. 2001. Visual Simulation of Smoke. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*. 15–22.

Zahra Forootaninia and Rahul Narain. 2020. Frequency-Domain Smoke Guiding. *ACM Trans. Graph.* 39, 6, Article 172 (Nov. 2020), 10 pages. https://doi.org/10.1145/3414685.3417842

Kai Fukami, Koji Fukagata, and Kunihiko Taira. 2021. Machine-learning-based spatio-temporal super resolution reconstruction of turbulent flows. *Journal of Fluid Mechanics* 909 (2021). https://doi.org/10.1017/jfm.2020.948

M Geier, A Greiner, and JG Korvink. 2006. Cascaded digital lattice Boltzmann automata for high Reynolds number flow. *Physical Review. E* 73, 6.2 (2006), 066705–066705.

Abhinav Golas, Rahul Narain, Jason Sewall, Pavel Krajcevski, Pradeep Dubey, and Ming Lin. 2012. Large-Scale Fluid Simulation Using Velocity-Vorticity Domain Decomposition. *ACM Trans. Graph.* 31, 6, Article 148 (Nov. 2012), 9 pages.

Xiaoxiao Guo, Wei Li, and Francesco Iorio. 2016. Convolutional neural networks for steady flow approximation. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 481–490.

SoHyeon Jeong, Barbara Solenthaler, Marc Pollefeys, Markus Gross, et al. 2015. Data-Driven Fluid Simulations Using Regression Forests. *ACM Trans. Graph.* 34, 6, Article 199 (Oct. 2015), 9 pages.

Huaizu Jiang, Deqing Sun, Varun Jampani, Ming-Hsuan Yang, Erik Learned-Miller, and Jan Kautz. 2018. Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In *IEEE Conference on Computer Vision and Pattern Recognition*. 9000–9008.

Volker John. 2003. *Large eddy simulation of turbulent incompressible flows: analytical and numerical results for a class of LES models*. Lecture Notes in Computational Science and Engineering, Vol. 34. Springer Science & Business Media.

Madhuri A Joshi, Mehul S Raval, Yogesh H Dandawate, Kalyani R Joshi, and Shilpa P Metkar. 2014. *Image and video compression: Fundamentals, Techniques, and Applications*. CRC press.

Hyungmin Kang, Dongho Lee, and Dohyung Lee. 2003. A study on CFD data compression using hybrid supercompact wavelets. *KSME international journal* 17, 11 (2003), 1784–1792.

Byungsoo Kim, Vinicius C. Azevedo, Markus Gross, and Barbara Solenthaler. 2020. Lagrangian Neural Style Transfer for Fluids. *ACM Trans. Graph.* 39, 4, Article 52 (July 2020), 10 pages. https://doi.org/10.1145/3386569.3392473

Byungsoo Kim, Vinicius C. Azevedo, Nils Thuerey, Theodore Kim, Markus Gross, and Barbara Solenthaler. 2019. Deep Fluids: A Generative Network for Parameterized Fluid Simulations. *Computer Graphics Forum* 38, 2 (2019).

Jungwoo Kim, Dongjoo Kim, and Haecheon Choi. 2001. An immersed-boundary finite-volume method for simulations of flow in complex geometries. *J. Comput. Phys.* 171, 1 (2001), 132–150.

Theodore Kim and John Delaney. 2013. Subspace Fluid Re-Simulation. *ACM Trans. Graph.* 32, 4, Article 62 (July 2013), 9 pages. https://doi.org/10.1145/2461912.2461987

Theodore Kim, Nils Thürey, Doug James, and Markus Gross. 2008. Wavelet Turbulence for Fluid Simulation. *ACM Trans. Graph.* 27, 3 (Aug. 2008), 1–6.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

Dmitrii Kochkov, Jamie A Smith, Ayya Alieva, Qing Wang, Michael P Brenner, and Stephan Hoyer. 2021. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences* 118, 21 (2021).

Dmitry Kolomenskiy, Ryo Onishi, and Hitoshi Uehara. 2019. Wavelet-based Data Compression for Three-dimensional Fluid Flow Simulations on Regular Grids. *Bulletin of the American Physical Society* (2019).

Wei Li, Yixin Chen, Mathieu Desbrun, Changxi Zheng, and Xiaopei Liu. 2020. Fast and Scalable Turbulent Flow Simulation with Two-Way Coupling. *ACM Trans. Graph.* 39, 4, Article 47 (July 2020), 20 pages. https://doi.org/10.1145/3386569.3392400

P. Lindstrom. 2014. Fixed-Rate Compressed Floating-Point Arrays. *IEEE Trans. Vis. Comp. Graph.* 20, 12 (2014), 2674–2683. https://doi.org/10.1109/TVCG.2014.2346458

P. Lindstrom and M. Isenburg. 2006. Fast and Efficient Compression of Floating-Point Data. *IEEE Trans. Vis. Comp. Graph.* 12, 5 (2006), 1245–1250. https://doi.org/10.1109/TVCG.2006.143

Beibei Liu, Gemma Mason, Julian Hodgson, Yiying Tong, and Mathieu Desbrun. 2015. Model-Reduced Variational Fluid Simulation. *ACM Trans. Graph.* 34, 6, Article 244 (2015).

Zichao Long, Yiping Lu, and Bin Dong. 2019. PDE-Net 2.0: Learning PDEs from data with a numeric-symbolic hybrid deep network. *J. Comput. Phys.* 399 (2019), 108925.

Siwei Ma, Xinfeng Zhang, Chuanmin Jia, Zhenghui Zhao, Shiqi Wang, and Shanshe Wang. 2019. Image and video compression with neural networks: A review. *IEEE Transactions on Circuits and Systems for Video Technology* 30, 6 (2019), 1683–1698.

D. Marr, E. Hildreth, and Sydney Brenner. 1980. Theory of edge detection. *Proceedings of the Royal Society of London. Series B. Biological Sciences* 207, 1167 (1980), 187–217.

Maxon. 2021. Redshift renderer. (2021). https://www.redshift3d.com/product

Shilpa P Metkar and Sanjay N Talbar. 2010. Fast motion estimation using modified orthogonal search algorithm for video compression. *Signal, Image and Video Processing* 4, 1 (2010), 123–128.

Don P. Mitchell and Arun N. Netravali. 1988. Reconstruction Filters in Computer-Graphics. In *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '88)*. Association for Computing Machinery, New York, NY, USA, 221–228. https://doi.org/10.1145/54852.378514

Patrick Mullen, Keenan Crane, Dmitry Pavlov, Yiying Tong, and Mathieu Desbrun. 2009. Energy-Preserving Integrators for Fluid Animation. *ACM Trans. Graph.* 28, 3, Article 38 (July 2009), 8 pages.

Octavi Obiols-Sales, Abhinav Vishnu, Nicholas Malaya, and Aparna Chandramowliswharan. 2020. CFDNet: A deep learning-based accelerator for fluid simulations. In *Proceedings of the 34th ACM International Conference on Supercomputing*. 1–12.

Young Jin Oh and In-Kwon Lee. 2021. Two-step Temporal Interpolation Network Using Forward Advection for Efficient Smoke Simulation. *Computer Graphics Forum* 40, 2 (2021), 355–365. https://doi.org/10.1111/cgf.142638 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.142638

Sang Il Park and Myoung Jun Kim. 2005. Vortex Fluid for Gaseous Phenomena. In *Symposium on Computer Animation*. 261–270.

Tobias Pfaff, Nils Thuerey, Jonathan Cohen, Sarah Tariq, and Markus Gross. 2010. Scalable Fluid Simulation Using Anisotropic Turbulence Particles. *ACM Trans. Graph.* 29, 6, Article 174 (Dec. 2010), 8 pages.

Tobias Pfaff, Nils Thuerey, and Markus Gross. 2012. Lagrangian Vortex Sheets for Animating Fluids. *ACM Trans. Graph.* 31, 4, Article 112 (July 2012), 8 pages.

L Lo Presti. 2000. Efficient modified-sinc filters for sigma-delta A/D converters. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal processing* 47, 11 (2000), 1204–1213.

Ziyin Qu, Xinxin Zhang, Ming Gao, Chenfanfu Jiang, and Baoquan Chen. 2019. Efficient and Conservative Fluids Using Bidirectional Mapping. *ACM Trans. Graph.* 38, 4, Article 128 (July 2019), 12 pages. https://doi.org/10.1145/3306346.3322945

Ryotaro Sakai, Daisuke Sasaki, and Kazuhiro Nakahashi. 2013. Parallel implementation of large-scale CFD data compression toward aeroacoustic analysis. *Computers & Fluids* 80 (2013), 116–127.

Syuhei Sato, Yoshinori Dobashi, and Theodore Kim. 2021. Stream-Guided Smoke Simulations. *ACM Trans. Graph.* 40, 4, Article 161 (July 2021), 7 pages. https://doi.org/10.1145/3450626.3459846

Syuhei Sato, Yoshinori Dobashi, Theodore Kim, and Tomoyuki Nishita. 2018. Example-Based Turbulence Style Transfer. *ACM Trans. Graph.* 37, 4, Article 84 (July 2018), 9 pages.

H. Schechter and R. Bridson. 2008. Evolving Sub-grid Turbulence for Smoke Animation. In *Symposium on Computer Animation*. 1–7.

Andrew Selle, Ronald Fedkiw, Byungmoon Kim, Yingjie Liu, and Jarek Rossignac. 2008. An Unconditionally Stable MacCormack Method. *Journal of Scientific Computing* 35, 2-3 (2008), 350–371.

Adam J Sierakowski and Andrea Prosperetti. 2016. Resolved-particle simulation by the Physalis method: enhancements and new capabilities. *J. Comput. Phys.* 309 (2016), 164–184.

Philippe R Spalart. 2009. Detached-eddy simulation. *Annual review of fluid mechanics* 41 (2009), 181–202.

John Steinhoff and David Underhill. 1994. Modification of the Euler equations for "vorticity confinement": Application to the computation of interacting vortex rings. *Physics of Fluids* 6 (1994), 2738–2744.

Jonathan Tompson, Kristofer Schlachter, Pablo Sprechmann, and Ken Perlin. 2017. Accelerating Eulerian fluid simulation with convolutional networks. In *International Conference on Machine Learning*, Vol. 70. 3424–3433.

Kiwon Um, Robert Brand, Yun (Raymond) Fei, Philipp Holl, and Nils Thuerey. 2020. Solver-in-the-Loop: Learning from Differentiable Physics to Interact with Iterative PDE-Solvers. In *NeurIPS*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Eds.), Vol. 33. 6111–6122.

Kiwon Um, Xiangyu Hu, and Nils Thuerey. 2018. Liquid splash modeling with neural networks. In *Computer Graphics Forum*, Vol. 37. 171–182.

Nobuyuki Umetani and Bernd Bickel. 2018. Learning Three-Dimensional Flow for Interactive Aerodynamic Design. *ACM Trans. Graph.* 37, 4, Article 89 (July 2018), 10 pages.

Henk Kaarle Versteeg and Weeratunge Malalasekera. 2007. *An introduction to computational fluid dynamics: the finite volume method*. Pearson education.

Steffen Weißmann and Ulrich Pinkall. 2010. Filament-based Smoke with Vortex Shedding and Variational Reconnection. *ACM Trans. Graph.* 29, 4, Article 115 (July 2010), 12 pages.

Maximilian Werhahn, You Xie, Mengyu Chu, and Nils Thuerey. 2019. A Multi-Pass GAN for Fluid Flow Super-Resolution. *Proc. ACM Comput. Graph. Interact. Tech.* 2, 2, Article 10 (July 2019), 21 pages. https://doi.org/10.1145/3340251

Martin Wicke, Matt Stanton, and Adrien Treuille. 2009. Modular Bases for Fluid Dynamics. *ACM Trans. Graph.* 28, 3, Article 39 (July 2009), 8 pages. https://doi.org/10.1145/1531326.1531345

Steffen Wiewel, Moritz Becher, and Nils Thuerey. 2019. Latent space physics: Towards learning the temporal evolution of fluid flow. In *Computer Graphics Forum*, Vol. 38. Wiley Online Library, 71–82.

David C Wilcox et al. 1998. *Turbulence modeling for CFD*. Vol. 2. DCW industries La Canada, CA.

You Xie, Erik Franz, Mengyu Chu, and Nils Thuerey. 2018. TempoGAN: A Temporally Coherent, Volumetric GAN for Super-Resolution Fluid Flow. *ACM Trans. Graph.* 37, 4, Article 95 (July 2018), 15 pages.

Jonas Zehnder, Rahul Narain, and Bernhard Thomaszewski. 2018. An Advection-Reflection Solver for Detail-Preserving Fluid Simulation. *ACM Trans. Graph.* 37, 4, Article 85 (July 2018), 8 pages.

Xinxin Zhang, Robert Bridson, and Chen Greif. 2015. Restoring the Missing Vorticity in Advection-projection Fluid Solvers. *ACM Trans. Graph.* 34, 4, Article 52 (July 2015), 8 pages.

Bo Zhu, Wenlong Lu, Matthew Cong, Byungmoon Kim, and Ronald Fedkiw. 2013. A New Grid Structure for Domain Extension. *ACM Trans. Graph.* 32, 4, Article 63 (July 2013), 12 pages.