Physics-Informed Neural Corrector for Deformation-based Fluid Control

Jingwei Tang\textsuperscript{1}, Byungsoo Kim\textsuperscript{1}, Vinicius C. Azevedo\textsuperscript{2}, Barbara Solenthaler\textsuperscript{1}

\textsuperscript{1}ETH Zürich, Switzerland
\textsuperscript{2}DisneyResearchStudios, Switzerland

Abstract

Controlling fluid simulations is notoriously difficult due to its high computational cost and the fact that user control inputs can cause unphysical motion. We present an interactive method for deformation-based fluid control. Our method aims at balancing the direct deformations of fluid fields and the preservation of physical characteristics. We train convolutional neural networks with physics-inspired loss functions together with a differentiable fluid simulator, and provide an efficient workflow for flow manipulations at test time. We demonstrate diverse test cases to analyze our carefully designed objectives and show that they lead to physical and eventually visually appealing modifications on edited fluid data.

CCS Concepts

• Computing methodologies → Physical Simulation; Neural networks;
overlap — if the target smoke goes under extreme deformations — these methods cannot provide meaningful gradients for the optimization or heuristics computation, and artificially computed force fields will not be able to properly guide fluid simulations.

Alternatively, fluids can be controlled by direct manipulation of pre-simulated fluid data. For instance, the volumetric flow data can be deformed with the underlying grid [SDY’15, PM17a], various fluid scenes may be stitched [SDN18] or sculpted for resizing [FHM’21]. While these techniques offer an unprecedented level of post-editing functionality, they rely on computationally expensive optimizations or re-simulations.

To enable efficient artistic control that is less constrained by the type of the deformation, we propose a fluid post-processing pipeline, but with the key difference of enabling target matching in interactive environments while preserving physical constraints. In our pipeline, we simulate a base smoke configuration, deform it with accessible control tools for prototyping, and finally correct the motion to increase physical realism. This is enabled by a self-supervised neural corrector that projects the deformed simulations back onto the manifold of the Navier-Stokes (NS) equations with the help of differentiable physics simulation.

Our model is implemented by convolutional neural networks and coupled with a differentiable fluid solver for training without any reference correction samples. By considering various physical quantities explicitly, our method offers not only user-friendly control of fluids but also conservation of the original physical characteristics. Thanks to the high speed of feed forward models, there is no necessity of expensive simulations or optimizations except the computation of the baseline simulation for the following deformation. At inference time, our method is thus computationally more efficient than previous target-driven simulation methods and allows users to interactively art-direct fluids. Our contributions can be summarized as follows:

- A neural corrector that projects an artistically deformed velocity field back onto the physical manifold at interactive frame rates. (Sec. 3.1)
- An objective function implemented through a differentiable fluid solver for evaluating physics laws. (Sec. 3.2)
- A set of physics-based loss functions to train the neural corrector in a self-supervised way (Sec. 3.2, Sec. 3.3) and corresponding ablation studies. (Sec. 4)

### 2. Related Work

**Online Control for Fluids.** One set of fluid control methods aim at matching given coarse guidance in space or time during simulation. One category of this group is to compute control force fields by evaluating heuristics for guidance. These methods derive control forces based on the difference between current and desired density field [FL04], signed distance functions [SY05a, SDE05, YCZ11], geometric potentials [HK04, SY05b], control particles [REN’04, TKPR06, MM13] and low-resolution simulations [HMK11]. While these techniques are computationally efficient, they often require manual work for delicate control due to the sub-optimal results.

Another category is based on optimization to achieve higher quality matches. Analytic gradients are computed to optimize forces [TMAPS03] and are enhanced by the adjoint method [MTPS04], ADMM framework [GITH14, PM17b] and primal-dual algorithm [IETG17]. Nielsen et al. [NCZ10, NDB11] formulated a variational problem to guide high-resolution simulations with low-resolution counterparts. Alternative representations of guidance are utilized such as meshes [RTWT12], vortex
structure [WP10], spectral component [RLL*13, FN20, TCACS21] and stream function [SDK21].

**Offline Control for Fluids.** Unlike online control approaches, offline methods control fluids by post-processing with low computational cost and enhanced controllability at test time.

Reduced order models [TLP06, WST09] build basis functions from simulation snapshots by dimension reduction and span an approximation of the original full model for computational efficiency with a subspace integrator [KD13], Laplacian Eigenfunctions [DWLF12, CSK18] and learning-based models [KAT*19, WKA*20]. Those methods, however, do not provide additional controllability that goes beyond physical parameters.

Turbulence synthesis approaches, another group in offline fluid control, compose turbulent details dissipated during simulation using subgrid [KTJG08], solid boundary [PTS09] and convolutional neural networks [CT17, XCT18, UHT18].

On the other hand, Appearance transfer approaches [KEBK05, NKL*07] build on patch-based synthesis to change the appearance of a target object with a source texture by finding correspondences between local regions, and are enhanced to improve temporal coherency [BBRF14, JFA*15], extended to 3D velocity fields [SDKN18], and coupled with neural style transfer techniques [KAGS19, KAGS20]. The main challenge of these methods is to minimize the discontinuity between synthesized patches or frames over time and hence work at the cost of expensive optimization. Contrary to previous approaches, our neural corrector targets physically-corrupted fluid deformations rather than subtle feature re-injection or appearance matching on coarse simulations.

Other works in this direction aim at direct fluid authoring. Fluid simulations have been reconstructed from sparse 2D inputs [EHT18, EUT19, FST21, QLWQ21, KH*22], interpolated [RWTT14, Thu16] and resized [FHM19, FHM*21]. Pan and Manocha [PM17a] proposed a method for editing smoke simulations, where the underlying grid is deformed and applied in the fluid solver by altering the semi-Lagrangian advection operator. However, the deformed advection operator typically changes the appearance of original fluid details notably, and hence the deformation results of the density fields can also fail to match user-input controls. Sato et al. [SDY*15] presented a divergence-aware method for deforming velocity fields, but it only preserves incompressibility and requires the expensive solution of the Poisson equation to obtain vector potentials. A follow-up work [SDKN18] also requires an energy minimization for stitching velocity fields smoothly. In contrast, our method is inherently physics-informed by the coupling with a differentiable fluid solver, and thus can learn a variety of physical characteristics including incompressibility, kinetic energy and vorticity in a self-supervised way and reflect them into edited flows while still matching user control inputs in interactive environments.

**Differentiable Physics.** Differentiable simulation frameworks [HKT20, HLS*19, HAL*19] allow us to incorporate gradient-based methods with the help of automatic differentiation, and hence neural networks can be augmented for solving inverse problems more robustly. Our work shares the concept of incorporating differentiable features with [UBF*20], so-called “Solver-in-the-loop”, to naturally learn physics-informed corrections via augmented neural networks. We would like to highlight that, while previous work focused on reducing numerical errors compared to given high-resolution reference samples, our method focuses on learning the reference manifold directly with no samples for the rectification process of deformed fluid flows. Lastly, physics informed neural networks [RPK19] employ automatic differentiation to compute partial gradients in PDEs for training neural networks to solve fluid simulations in a self-supervised way.

### 3. Physics-Informed Neural Corrector

In this work, we focus on incompressible fluids simulated by solving the Navier–Stokes equations

\[
\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} = -\frac{1}{\rho} \nabla p + \mu \nabla^2 \mathbf{u} + \mathbf{f},
\]

\[
\nabla \cdot \mathbf{u} = 0,
\]

where \(\mathbf{u}\), \(\rho\) and \(\mathbf{f}\) denote the fluid velocity, pressure and external forces, respectively. External force can be represented through a force function \(\mathbf{f} = f(\rho, \mathbf{u})\) that depends on the marker density.
and velocity fields. Viscosity terms are omitted due to the inherent dissipation of velocity-pressure fractional stepping methods [ETK+07, MCP+09]. We further denote a fluid solver for multiple time integrations of \( n \) from the time frame \( t \) as

\[
\mathbf{u}_{t+\Delta t} = S^n(\mathbf{p}_t, \mathbf{u}_t)
\]

with a step size of \( \Delta t \). For simulating smoke, the density field \( \rho \) is passively advected \( \mathbf{p}_t = A(\rho_0, \mathbf{u}_0^{t-1}) \) where \( \mathbf{u}_0^{t-1} = \{\mathbf{u}_0, \ldots, \mathbf{u}_{t-1}\}_{t \geq 1} \) is the set of velocity fields computed from the solver.

Our deformation-based control consists of two parts: deformation \( \mathcal{D} \) and correction \( \mathcal{C} \). The deformation process takes user-defined deformations over time as the input control and deforms the density and velocity fields of the original fluid simulation. The deformation is not restricted to any specific operations, it is often performed by manipulating the underlying regular grid \( G \). For instance, when scaling the original simulation in the \( x \) direction by 1.5 times, physical quantities (divergence, velocity magnitude, etc.) vary substantially from 1.2 to 13.6 times compared to their original values. Moreover, naively deforming the velocity field can cause the velocity sequence to deviate significantly from Navier–Stokes Equations, as no specific constraints are applied on user inputs. Both factors eventually lead to visual artifacts in the density fields advected by the deformed velocities.

Figure 5: Ablation study on loss function terms, computed on simulations of resolution 256 \( \times \) 256 (a). (b) and (c) show the naive approaches using a deformed density and velocity, respectively. (d) and (e) show state-of-the-art methods that can be directly compared to ours (j). (f)-(i) show our model trained with different subsets of loss terms, which results in various artifacts in overall shape and details, as well as non-physical behaviors.

To deal with the issues created by naively deforming fields, Flynn et al. [FEHM19,FHM+21] uses a seam-carving technique to edit fluid data. To best determine which seam to add/remove, physical characteristics of the fluid such as kinetic energy \( \mathbf{u}^2 \), vorticity magnitude \( |\nabla \times \mathbf{u}| \) and the gradient magnitude of density \( |\nabla \rho| \) are considered and formulated into an energy function. Similarly, Sato et al. [SDN18] proposes an interpolation function for “copying and pasting” of fluids that is formulated by the spatial smoothness of the Dirichlet energy \( |\nabla \mathbf{u}|^2 \), which aims to preserve the divergence \( \nabla \cdot \mathbf{u} \) and vorticity magnitude \( |\nabla \times \mathbf{u}| \) of the original simulation.

Instead of directly constraining the deformer \( \mathcal{D} \) with energy minimization, we use a corrector \( \mathcal{C} \) to correct the unphysical motions from the deformation. Formulated as a neural network, the corrector can achieve faster inference at test time than energy minimization. When designing the objectives, we first take the Navier-Stokes equations (Eqns. (1) and (2)) into consideration through a differentiable fluid solver at training time. Thus, applying our corrector leads to deformed simulations that better satisfy temporal advection of quantities and the divergence conditions of the underlying fluid motion.
Moreover, we can also employ previous energy functions \[\text{[FEHM19,FHM21,SDN18]}\] for better detail preservation. The design choices of the proposed physically inspired corrector will be detailed in Sec. 3.1.

### 3.1. Neural Corrector

The corrector \(C\) takes a deformed velocity field \(\hat{u}\) and the deformation grid \(\hat{G}\) as input and outputs the velocity correction \(u'\). To differentiate between simulations with distinct properties, the corrector can optionally be conditioned on simulation parameters \(\alpha\) (e.g., buoyancy, vorticity confinement scale). The velocity correction term is added to the deformed velocity as the corrected velocity:

\[
\tilde{u}_t = \hat{u}_t + u'_t, \quad u'_t = C(\hat{u}_t, \hat{G}_t, \alpha).
\]  

Here the underline indicates the optional input. The correction is performed for all velocity fields in a deformed sequence. The final corrected density fields \(\hat{\rho}_t\) are obtained by \(r\)-recursive advections on the first deformed density frame with corrected velocity fields as

\[
\hat{\rho}_t = A(\hat{\rho}_0, \tilde{U}^{t-1}_0).
\]

We implement our fluid corrector as a convolutional encoder-decoder similar to Chu et al. \[\text{[CTS21]}\]. The encoder part downsamples the input field 4x (2D) or 16x (3D) with residual blocks \[\text{[HZRS16]}\] and strided convolutions. The deformation grid \(\hat{G}_t\) is restricted to a lower spatial resolution compared to the original velocities, since reducing the dimensionality of the deformations helps with generalization. This deformation grid is added as an input at the bottleneck of the network after a going through a convolutional layer (Subnet 1). The optional simulation parameter \(\alpha\), when available, goes through another set of convolution layers (Subnet 2) before being concatenated with the feature map in the bottleneck. Conditioning the architecture to varying simulation parameters allows the artist to have additional control over the final simulation appearance. The decoder part upsamples the bottleneck features back to the original resolution with transposed convolutions. The whole pipeline of our neural corrector can be seen in Fig. 2. We detail the architecture of the neural network in Table (2).

### 3.2. Physics-informed Loss Functions

As user-input deformations can be arbitrary, it is very difficult to obtain reference corrected velocity fields. We thus define all our loss functions for training the neural corrector in a self-supervised fashion.

**Navier–Stokes (NS) Loss.** The core of our objective functions is implemented through a differentiable fluid solver for evaluating the network corrections. Under the arbitrary user-input deformations, the original fluid simulations usually go off course to the outside of the NS manifold (Fig. 2). The NS loss aims at projecting the velocity fields back onto the manifold. When advancing the corrected velocity field \(\tilde{u}\) by \(k\) solver steps, the resulted velocity field \(S^k(\hat{\rho}_t, \tilde{u}_t)\) should be similar to the corrected velocity field at frame \(t + k\):

\[
L_{NS} \left( \tilde{U}^{t+n}_t \right) = \sum_{k=1}^{n} \left\| S^k(\hat{\rho}_t, \tilde{u}_t) - \tilde{u}_{t+k} \right\|_2^2. \quad (6)
\]

Here \(\hat{\rho}_t\) denotes the deformed density. By minimizing the NS loss, we expect the corrected velocities in the time sequence to not only connect themselves through self-advection and external force integration, but also satisfy boundary conditions through the pressure projection. Note that no ground-truth corrected velocity field sequence is required in the NS loss. Only a set of snapshots of deformed velocity fields is present in training. More solver steps will result in a better evaluation of the NS loss, but makes the loss computations more expensive. We found in our experiments that \(n = 8\) for 2D and \(n = 4\) for 3D are good trade-offs between quality and complexity. We use such settings in all our experiments.

In addition, we emphasize the incompressibility constraint in Eqn. (2) by minimizing the divergence of the corrected velocity fields for \(n\) rollout steps:

\[
L_{\nabla} \left( \tilde{U}^{t+n}_t \right) = \sum_{k=0}^{n} \left\| \nabla \cdot \tilde{u}_{t+k} \right\|_2^2. \quad (7)
\]

**Physics Characteristic Losses.** While the NS loss implicitly takes the physical features of fluids into account, we additionally consider three extra quantities — kinetic energy, magnitude of vorticity and gradient magnitude of density — in our loss functions to better preserve the detailed characteristics of the original flows. We minimize the differences of the corrected physical fields compared to the original simulation fields:

\[
L_{KE} \left( \tilde{U}^{t+n}_t \right) = \sum_{k=0}^{n} \left\| \tilde{u}_{t+k}^2 - D \left( \tilde{u}_{t+k} \right) \right\|_2^2, \quad (8)
\]

\[
L_{\omega} \left( \tilde{U}^{t+n}_t \right) = \sum_{k=0}^{n} \left\| |\omega_{t+k}| - D \left( |\omega_{t+k}| \right) \right\|_2^2, \quad (9)
\]

\[
L_{\nabla\rho} \left( \tilde{U}^{t+n}_t \right) = \sum_{k=0}^{n} \left\| \nabla \rho_{t+k}^2 \right\|_2 - D \left( \left| \nabla \rho_{t+k} \right| \right) \right\|_2^2. \quad (10)
\]

Here \(\omega_{t+k} = \nabla \times \tilde{u}_{t+k}\) is the vorticity field, \(\rho'_{t+k} = A(\hat{\rho}_t, \tilde{U}^{t-1}_0)\) is the corrected density, i.e. deformed density recursively advected by corrected velocity sequence. When comparing the corrected and original simulation fields, we deform each original field (e.g., \(\omega\) in Eqn. (9)) to the deformed space through \(D\) so that both fields can be compared at a same grid location. Note that deforming a physical quantity does not necessarily imply physical accuracy and these losses are used to preserve features of the original simulation. Alternatively, one could compute features of the velocity and density fields after their mapping to the deformed grid, but this would require evaluating derivatives on curvilinear grids \[\text{[AO13]}\]. Similar to the NS loss, we evaluate our loss functions over \(n\)-rollouts.

**Density Guidance Loss.** The above losses push the corrected velocity sequence into a physically-correct direction. However, only incurring them can potentially undo the user-input deformations. Therefore we additionally define a density guidance loss to instruct the corrector to follow the general look of the given deformed den-
Figure 6: **Bending** of rising plume simulations in resolution $192 \times 256 \times 128$. The deformation grids for bending are shown as inset images. Our model can correct the nonphysical motions resulted from the bending.

Figure 7: **Examples from the 3D plume dataset.** The top row shows the original simulations of different simulation parameters at their last frames. The bottom row shows the deformed density field of the corresponding original simulation, with the deformation grid shown in the inset images.

Figure 8: **Non-uniform scaling** of rising plume simulations in $192 \times 256 \times 128$. The deformation grids are shown as inset images. Our model can create plausible corrections for spatially varying transformations that both stretch and twist the original simulation.
Our 2D plume dataset consists of buoyancy driven smoke simulations. The simulation is initialized by a rectangular density source on the bottom, an open boundary on the top, and closed boundaries on both sides. Vorticity confinement \cite{SU94} is used to reintroduce small scale details lost due to advection. We vary the buoyancy forces from $1.5 \times 10^{-4}$ to $3.0 \times 10^{-4}$ and vorticity confinement scale from 0.10 to 0.19, generating 24 simulation sequences of 150 frames. Both buoyancy and vorticity confinement are used as inputs to the network during training. In addition, we generate 30 deformation sequences of bending and uniform scaling. The parameters used to generate those deformations are randomized, in order to create a diverse set of deformation patterns that will enhance the network ability to generalize to unseen deformations. During training, each iteration randomly samples clips from both the simulation and deformation sequences, and deforms the simulation fields on-the-fly, effectively traversing all combinations of simulation parameters and deformation sequences in the dataset. The 3D plume dataset is likewise computed with a resolution of $192 \times 256 \times 128$ and examples of the 3D dataset can be seen in Fig. 7.

Besides plume simulations, we prepare a smoky character dataset to test our model on more complicated deformations in 3D. We first convert a static T-pose human body mesh from the AMASS dataset \cite{MGFT19} into a volume representing the smoke source and simulate for 100 steps in resolution of $208 \times 208 \times 112$. To conform the smoke motion to the source character animation, the density is dissipated with a rate of 0.9 in the region outside the character volume at each simulation step. Additionally, the velocity is initialized to follow the animated character surface and small perturbations are added to create turbulence patterns. Since only one simulation setting is used in this dataset, no simulation parameters are used as input to the neural network. Nine motion sequences of the same body mesh are extracted from AMASS dataset and used to deform the original smoke simulation. The deformation is done by tracking the differences of the corresponding mesh vertices using Point Deform Node in Houdini \cite{hou}. One example of the dataset can be seen in Fig. 11.

4. Experiments and Results

In this section, we show our results on test dataset and explore several smoke editing applications. We compare with the results from \cite{SDY15} and \cite{PM17}. These works are the most similar to ours as they directly use deformation grids to edit smoke sim-
Deformed $\hat{\rho} t$ Original
User Input
Deformation Grid
(a)

Figure 10: Editing plume simulation in screen space. (a) The original simulation is deformed with a 3D curve through the Houdini Curve Deformation Node. (b) Point handles are dragged in space (green to blue). The corresponding deformation grid is generated by 3D Moving Least Square method [ZG07].

4.1. Implementation Details

We implement our fluid simulation solver and the neural networks in Python with PyTorch [PGM∗19] for its embedded automatic differentiation tool as well as easy GPU deployment. We run all our 2D experiments on an NVIDIA GeForce RTX 2080Ti GPU with 11 GB of dedicated memory, and our 3D experiments on two NVIDIA GeForce RTX 3090 GPUs with 24 GB dedicated memory. The network is trained with an Adam Optimizer [KB15], with the learning rate fixed at $10^{-4}$. The training takes 24-48 hours for 2D models and 72-96 hours for 3D models.

For the differentiable fluid simulator, we use the orthogonal and equidistant staggered MAC grid [HW65] to discretize the density, pressure and velocity fields. The MacCormack method is employed to solve the advection step with a 3rd-order Runge-Kutta integrator. For the incompressibility constraint of the fluid solver in Eqn. (2), we implement pressure projection through a Preconditioned Conjugate Gradient (PCG) solver with the Incomplete Poisson Preconditioner [SKF11]. In particular, the backpropagation of the PCG solver is implemented by directly computing the adjoint through another linear solve [MTPS04], rather than a naive automatic differentiation, which can result in significant memory bottleneck. We use our simulator for both the data generation process and the Navier-Stokes loss in Eqn. (6).

4.2. 2D Test Cases for Ablation Studies

We first show 2D test results in Fig. 5 to illustrate the effect of each loss term and compare our method with previous works. Some of the sequences shown here are better represented on the accompanying video, since the Navier–Stokes loss can improve physical realism of transported quantities. The original density (a) is warped into its deformed configuration (b). We also show the result of just naively deforming velocities and advecting the original densities in (c). Simply warping density or velocity fields clearly creates visual artifacts. Our model trained with a loss function including all
terms (j) can better preserve the shape and physical properties of the original smoke.

The method of Sato et al. [SDY*15] (d) can preserve the incompressibility of the velocity field, but deviates from the original appearance of the plume, while the method of Pan et al. [PM17a] (e) does not match the user-input deformed density in (b). When our model is trained without Navier-Stokes Loss (Eqn. (6)) (f), fluid behaviors get lost, and visual artifacts appear. Divergence Loss (Eqn. (7)) is necessary as a helper term as the Navier-Stokes Loss usually does not get minimized to zero, and it enforces the incompressibility of the corrected velocity (g). Leaving out the physical characteristic losses (Eqns. (8), (9) and 10), less turbulent details from the original simulation are reconstructed (h). When Density Guidance Loss (Eqn. (11)) is removed (i), the final appearance of the plume is further away from the user-input deformation (b).

4.3 3D Applications

Plume Dataset. Fig. 6 shows the results on test examples similar to the training dataset. More extreme deformation examples are shown in Fig. 8. Our correction removes the unphysical motions such smoke sinks in the top-right corner (a), overly stretched densities (b), while also simultaneously matching the user-input deformation. The method of Sato et al. [SDY*15] can provide incompressible velocity sequences, but fails to match the small structures presented in the original simulation; lastly, the method of Pan et al. [PM17a] fails to match the deformation, and creates visual artifacts on the upper region of the smoke.

We additionally test the same model on higher resolution test examples of $288 \times 384 \times 192$. Fig. 9 shows our correction results on the same bending deformation as in Fig. 6(b). The deformation is used on simulations generated with different buoyancy and vorticity confinement coefficients that are not present in the training set. Our model can seamlessly handle all four simulation settings.

To test our model on fluid editing applications, we use the screen-space editing tools to generate deformation grids in Fig. 10. In (a), we use the Path Deform Node from Houdini to deform the simulation fields with a 3D curve. A more extreme deformation grid is generated from this method. In (b) Point handles are defined in 3D (green points), and dragged by the user to the target positions (blue points). A Moving Least Square method [SWW06, ZG07] is used to generate the deformation grid. Our method can correct the unphysical deformations in both cases while the methods of Sato et al. [SDY*15] and Pan et al. [PM17a] create visual artifacts or fail to match user inputs.

Smoky Character Dataset. To test our method on simulations other than plumes and more complicated deformations, we train the model on the smoky character dataset. We show model corrections on the training example in Fig. 11 and on higher resolution ($320 \times 320 \times 160$) test examples in Fig. 12. Our model can remove the clear visual artifacts from extreme deformations.

5. Conclusion and Discussion

We have presented a method for physics-informed correction on deformed fluid data. The recovery of deteriorated physical characteristics during deformation is promptly performed by learned neural networks with physics-inspired losses implemented with a differentiable simulation framework.

One limitation of our method is that once trained, it can only generalize to similar simulation setups. For example, the model trained on the 3D plume dataset cannot be used to predict corrections on the smoky character dataset. Subdividing the input fields into smaller patches during training can potentially help improve generalization, as the required receptive fields effectively decrease. Additionally, we empirically fix the magnitude of each loss term through a hyper-parameter search. The same set of loss weights might not be optimal for other datasets, but can serve as a good starting point for the tuning.

Several other aspects can be improved in the future work: we only tested our method on smoke simulations, but liquid simulations could be similarly corrected as our method it is not restricted to smoke by design. There are other types of corrections that we have not tested, for instance, detail enhancement as presented in [UBF*20] or flow interpolations as shown in [SDN18]. Other than our current version of linearly performing the correction on...
the deformed velocity fields, we tested using advection operator for non-linear corrections but didn't get satisfying results. More non-linear corrections could be tested for better corrections in extreme deformations. In terms of architectural choices, there is also room for improvement by using coordinate-based networks [SMB+20].

Appendix A: Neural Network Architecture

We detail our neural network architectures in Table 2. The table shows architectures in 2D, with a 256 × 256 input example to state the sizes. When using the network in 3D, we change the 2D layers to 3D counterparts. We also downsample the input 4× with strided convolutions before conv0 layer, and upsample the output 4× with transposed convolution after output layer.

References


Blending liquids. 


