FNOPT: Resolution-Agnostic, Self-Supervised Cloth Simulation using Meta-Optimization with Fourier Neural Operators

Ruochen Chen, Thuy Tran, Shaifali Parashar

FNOPT: Resolution-Agnostic, Self-Supervised Cloth Simulation using Meta-Optimization with Fourier Neural Operators

Ruochen Chen* Thuy Tran* Shaifali Parashar

CNRS, École Centrale de Lyon, INSA Lyon, Université Claude Bernard Lyon 1, LIRIS, UMR5205, France

{ruochen.chen, dinh-vinh-thuy.tran, shaifali.parashar}@liris.cnrs.fr

Abstract

We present FNOPT*, a self-supervised cloth simulation framework that formulates time integration as an optimization problem and trains a resolution-agnostic neural optimizer parameterized by a Fourier neural operator (FNO). Prior neural simulators often rely on extensive ground truth data or sacrifice fine-scale detail, and generalize poorly across resolutions and motion patterns. In contrast,* FNOPT *learns to simulate physically plausible cloth dynamics and achieves stable and accurate rollouts across diverse mesh resolutions and motion patterns without retraining. Trained only on a coarse grid with physics-based losses,* FNOPT *generalizes to finer resolutions, capturing fine-scale wrinkles and preserving rollout stability. Extensive evaluations on a benchmark cloth simulation dataset demonstrate that* FNOPT *outperforms prior learning-based approaches in out-of-distribution settings in both accuracy and robustness. These results position FNO-based meta-optimization as a compelling alternative to previous neural simulators for cloth; thus reducing the need for curated data and improving cross-resolution reliability. Our code is publicly available at https://github.com/Simonhfls/FNOpt.*

1. Introduction

Physics-based cloth simulation has been a long-standing research topic in computer graphics. Traditional approaches model the cloth dynamics by discretizing the classical time-varying partial differential equation (PDE) of motion into an ordinary differential equation (ODE), and apply various numerical integration methods for simulation. Since the cloth deformation is stiff, traditional solvers require expensive computation for either small time steps in explicit methods [5] or additional techniques for implicit ones [1] in order to avoid numerical instability. This hinders real-time applications such as realistic animations in computer-aided engineering. To overcome these limitations, data-driven methods [22, 27] have emerged, where neural networks are trained on ground truth trajectories generated by traditional simulators [23]. These neural simulators predict the next state by a single forward pass and thus significantly improve the efficiency. However, they require large amounts of curated training data, which can be costly to generate. Moreover, these models often struggle to generalize beyond the distribution of motions and mesh resolutions seen during training.

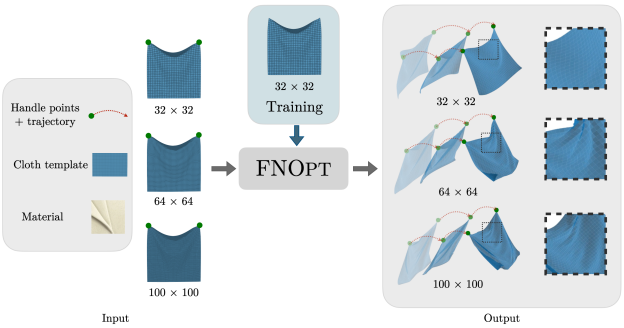

Figure 1. Being trained on 32 × 32 mesh, FNOPT generalizes to various mesh resolutions, cloth templates, motion speeds, handle points and trajectories without retraining.

Modern methods [30, 32] recast the implicit numerical integrators for the object motion's discretized ODE as an optimization problem, which allow learning neural garment and cloth simulators by minimizing the physics-based losses. However, their rollouts often suffer from stability issues. A novel development [36] shows that rather than a neural simulator, training a neural optimizer via meta-learning technique enables adaptive iterative update that can solve various PDEs and cloth simulations at higher precision. However, while this model generalizes to various motions, it still struggles to generalize across various mesh resolutions. Recently, neural operators such as [17, 18] are introduced that directly learn mappings between function spaces. Unlike conventional neural networks that can

*Equal contribution

only interpolate between the learned spaces, these operators allow a genuine evaluation across continuous spatial domains; consequently leading to a stronger generalization across various mesh resolutions. An important work in this direction is FNO [18], which performs a computationally efficient super-evaluation by transforming the computation onto a Fourier domain.

In this work, we present FNOPT, a novel self-supervised cloth simulation framework that combines meta-learning capabilities of [36] and efficient super-evaluation capabilities of FNOs. Therefore, the proposed framework enables a generalized meta-learning of various cloth motions across a diverse range of mesh resolutions. The super-evaluation capability of FNOs incorporated in cloth simulation allows a trained model on a lower resolution to perform a zero-shot generalization on a higher resolution (see Figure 1), capturing finer details that may not necessarily be present on the training data at lower resolution. Our experiments demonstrate a superior generalization and accuracy of FNOPT across a wide range of mesh resolutions, templates and boundary conditions compared to state-of-the-art methodologies in both supervised and self-supervised domains.

2. Related Work

Cloth Simulation. Early works [33] modeled cloth dynamics using a continuum formulation, in which the potential energy functions were derived based on elasticity theory. The cloth was then discretized into a rectangular mesh and the resulting ODE of motion was solved by applying semi-implicit integration method. [1] further developed an implicit integration formulation on triangular cloth and handled contact and geometric constraints directly, allowing larger step-size control while keeping the simulation stable. Although improving the efficiency and stability for real-time application, the method still fixes the cloth mesh topology, which prohibits realistic deformation with fine wrinkles and folds. To overcome this limitation, [23] presented a dynamic remeshing technique to capture high details of the cloth. It has been widely used and has become a standard framework, titled ARCSim, in cloth simulation community.

In order to accelerate simulation, data-driven methods train neural networks in a *supervised* manner using large-scale datasets containing ground truth cloth trajectories simulated by offline simulators. [2, 10, 24, 25, 28, 29, 34, 35, 39] focus on garments draped over articulated bodies and regress vertex-wise displacements from a reference template, conditioned on pose and shape parameters. In these settings, the cloth behavior is largely dominated by the underlying body geometry and motion. To fully capture the physical interactions arising between cloth vertices themselves, Libao [22] trains graph-based neural networks on trajectories generated by offline simulators. Their model builds on MeshGraphNets [27], which has shown strong performance across various mesh-based simulation tasks. However, collecting high-quality simulation data is costly and time-consuming. Beyond the data burden, the trained networks tend to overfit to specific mesh resolutions, boundary conditions and motion patterns seen during training, which limits their ability to generalize under distribution shifts such as finer meshes or faster motions. Because explicit physical constraints are not enforced, their predictions may deviate from physically accurate behavior in unseen scenarios.

Modern advances on garment and cloth simulation perform *self-supervised* learning by leveraging the key observation that the backward Euler solution to the discretized equations of motion can be recast as an optimization problem [3, 7, 30, 32]. Hence, by training a neural network that minimizes the physics-based energy functions, we obtain a differentiable neural simulator without the need to use data simulated a priori. This relaxes the requirement on simulating large-scale dataset by physics-based simulators for training supervised models. Moreover, the per-time-step numerical integration methods used in physics-based simulators are shifted to a forward network pass. Such novel formulation is remarkable for cloth simulation, especially in the contemporary era of deep learning and large models. While these methods inherit the efficiency of neural networks, their predictions can be unstable due to error accumulation over long sequences. [36] proposed to train a neural optimizer rather than a neural simulator, allowing for iterative updates to refine the prediction to higher precision. However, such self-supervised models are trained with a fixed rectangular mesh and can struggle when simulating cloth in significantly different mesh resolutions, limiting their usage in practical scenarios. In this paper, we overcome this limitation of self-supervised cloth simulation by incorporating resolution-agnosticism using the neural operators.

Neural Operators for solving PDEs.

Standard neural network architectures depend heavily on the discretization and have difficulty in generalizing to resolutions other than the training data. Recently, neural operators have been developed as a class of models that guarantee discretization invariance [14, 17, 18]. Such property is crucial in the context of solving PDEs, where the training data is often provided at varying resolutions and high-resolution data is expensive to generate. This property is also needed for resolution-agnostic cloth simulation, allowing for simulating fine-level details such as folds and wrinkles depending on the mesh-resolution.

Various architectures of neural operators have been studied. Graph neural operators (GNO) [17] performed kernel integration on graph structures and can handle irregular geometries. However, as other graph-based methods,

GNO is limited by computational complexity with long-range global interactions on the graph. To overcome this, FNO was introduced to represent the kernel integration in spectral domain by leveraging Fourier transform, enabling to use discrete fast Fourier transform (FFT) to improve efficiency. From these outstanding properties, FNO has become a standard in scientific computing and been applied into various domains including fluid and solid mechanics [8, 11, 19], geoscience [37, 38], weather forecasting [15, 26] and inverse-design problems [40]. Their advanced computational accuracy and widespread applicability motivates their usage in FNOPT.

3. Simulating Cloth via FNOPT

3.1. Mathematical Foundation

Our goal is to simulate cloth dynamics, which can be modeled by the following time-varying partial differential equation (PDE) of motion

$$\frac{\partial}{\partial t}\left(\mu\frac{\partial x}{\partial t}\right)+\gamma\frac{\partial x}{\partial t}+\frac{\delta\mathcal{E}}{\delta x}=f(x,t), \tag{1}$$

where $x(p,t)$ is the position of the particle p within the domain Ω at time t, $\mu(p)$ is the mass density, $\gamma(p)$ is the damping density, $\mathcal{E}(x)$ is the potential energy of elastic deformation and $f(x,t)$ represents external forces [33].

By defining the state $u(p,t) := \left(x(p,t), \frac{\partial x(p,t)}{\partial t}\right)^\top$, we can reformulate cloth dynamics to the PDE problem

$$\frac{\partial u}{\partial t}=\mathcal{R}(u,t), \quad \text{in } \Omega\times(0,\infty), \tag{2}$$

$$u=g, \quad \text{in } \partial\Omega\times(0,\infty), \tag{3}$$

$$u=a, \quad \text{in } \bar{\Omega}\times\{0\}, \tag{4}$$

where $\mathcal{R}$ is a possibly nonlinear partial differential operator, g is a known boundary condition on the boundary $\partial\Omega$ describing the handle points and their trajectories specified as input, and $a=u(\cdot,0)$, is the initial condition describing position and velocity of points within the closed domain, $\bar{\Omega}$. Such dynamical system can be solved by training a model u_θ to minimize the residual errors of Equations (2) to (4). In order to train it in a discretization-invariant way, we employ an FNO, $\mathcal{F}_\theta$ [18, 21], which maps from the initial condition a to the state $u=\mathcal{F}_\theta(a)$. In contrast to standard neural networks, this choice of architecture allows us to learn a mapping $a\mapsto u$ between function spaces, which is the key to generalize to perform super-evaluation on higher-resolution cloth to get finer details.

3.2. Training FNO

Because the dynamics in Equations (2) to (4) evolves over time, we must include the current time t as an input to the neural operator $\mathcal{F}_\theta$ so that it can produce at that time the corresponding cloth state $u(\cdot,t)=\mathcal{F}_\theta(a,\,t)$. However, this requires to train the network within a fixed time interval $(0,T]$ for some choice of large T, which prevents simulation for out-of-distribution $t>T$. Therefore, as suggested by [4], we instead train an autoregressive neural operator $\mathcal{F}_\theta$ that maps current state $u(\cdot,t)$ to next state $u(\cdot,t+\Delta t)$ for initial condition $u(\cdot,0)=a$, allowing us to simulate up to any time step.

As mentioned in Section 3.1, we must train the neural operator by minimizing the residual errors of Equations (2) to (4). Since the initial and boundary conditions associated with each sequence of cloth motion are known, the computation of the residual errors for Equations (3) and (4) is straightforward. However, the physics prior in Equation (2) requires an expensive computation of high-order derivatives of the network. To simplify the computation, we discretize the PDE in Equation (2) by using backward Euler method. Thus, we write

$$\mathbf{M}\frac{\mathbf{x}_{t+1}-\mathbf{x}_t-\Delta t\mathbf{v}_t}{\Delta t^2}+\frac{\partial\mathcal{E}_{\text{int}}}{\partial\mathbf{x}}=\mathbf{f}_{\text{ext}}\left(\mathbf{x}_{t+1},\frac{\mathbf{x}_{t+1}-\mathbf{x}_t}{\Delta t}\right), \tag{5}$$

where we have incorporated the damping term into external force $\mathbf{f}_{\text{ext}}$, $\mathbf{x}$ and $\mathbf{v}$ are the positions and velocities, respectively. As suggested by [9, 30, 32, 36], solving this system can be recast as minimizing the following loss function

$$\mathcal{L}_{\text{cloth}}=\mathcal{E}_{\text{int}}+\mathcal{E}_{\text{ext}}+\mathcal{E}_{\text{inertia}}. \tag{6}$$

The first term $\mathcal{E}_{\text{int}}(\mathbf{x}_t)$ corresponds to the internal potential energies of cloth, including stretching, shearing, and bending. The second term $\mathcal{E}_{\text{ext}}(\mathbf{x}_t)$ refers to the external forces such as gravity and wind. The last energy $\mathcal{E}_{\text{inertia}}$ imposes the inertia constraint, which makes wrinkles and dynamic behavior appear. We refer to [32] for more details on the energies.

3.3. Training Neural Optimizer via Meta-learning

Using the loss functions formalized above, we train a neural cloth simulator that predicts the acceleration $\mathbf{a}_{t+1}$ from the current position $\mathbf{x}_t$ and velocity $\mathbf{v}_t$, given some boundary conditions. We use the training cycle proposed by [32], where the data pool is initialized with random cloth states and then progressively augmented with the model's own predictions. This technique allows us to train the model in a self-supervised manner without using any data generated from traditional physics-based simulator (PBS). However, the predictions can be unstable due to error accumulation over frames when simulating long sequences. Hence, we follow [36] to train a neural optimizer via meta-learning technique. Concretely, rather than predicting the acceleration directly, we initialize the prediction as $\mathbf{a}_{t+1}^{(0)}=0$ and

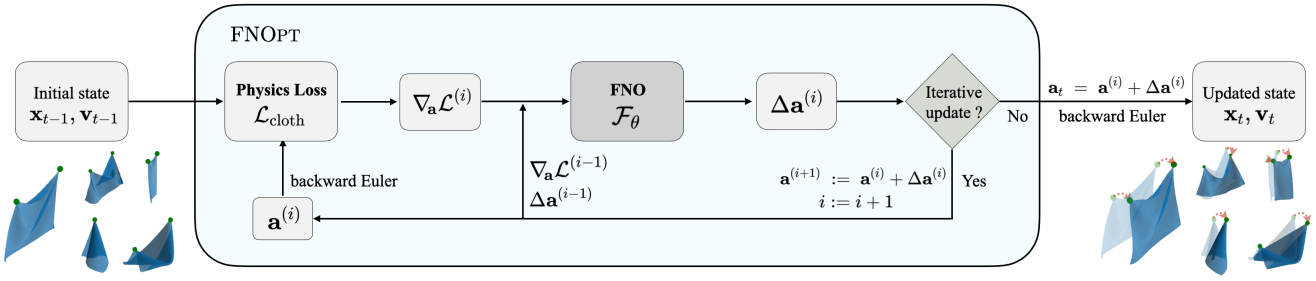

Figure 2. FNOPT pipeline. At each simulation time step t, an inner optimization loop uses an FNO-based optimizer to predict updates $\Delta\mathbf{a}^{(i)}$ from physics-based loss gradients and state information. After N iterations, backward Euler advances the state to $(\mathbf{x}_t, \mathbf{v}_t)$. Superscript i indicates the inner iteration index.

iteratively optimize it using the following update rule

$$\mathbf{a}_{t+1}^{(i+1)} = \mathbf{a}_{t+1}^{(i)} + \Delta\mathbf{a}_{t+1}^{(i)}, \qquad \text{for } i = 0, ..., N-1, \quad (7)$$

where N is the number of iterations used for each time step $t+1$, and the update direction $\Delta\mathbf{a}_{t+1}^{(i)}$ is given by the neural operator $\mathcal{F}_\theta$. The resulting acceleration is then used to evaluate the next cloth state using backward Euler method as

$$\mathbf{v}_{t+1} = \mathbf{v}_t + \Delta t\mathbf{a}_{t+1}, \quad (8)$$

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \Delta t\mathbf{v}_{t+1}. \quad (9)$$

Combining all features introduced so far, our FNOPT framework follows the pipeline in Figure 2.

3.4. Inference

At inference time, given initial position and velocity of the cloth as well as boundary conditions including handle points and their trajectories, FNOPT performs an autoregressive rollout. For each time step, we apply the learned optimizer $\mathcal{F}_{\theta^\star}$ for a fixed number N of inner updates to minimize the objective in Equation (6). Note that the same trained $\mathcal{F}_{\theta^\star}$ is used across all experiments, no fine-tuning is required.

Optionally, to handle self-collision between cloth, we can augment the objective with a repulsive term, $\mathcal{L}_{\text{rep}}$ [16] at inference time, which reduces interpenetration without retraining:

$$\mathcal{L}_{\text{rep}} = \lambda_{\text{rep}} \sum_{i=1}^{N} \sum_{j \in \mathcal{A}_i} -\log\big(\|\mathbf{v}_i - \mathbf{v}_j\|^2\big).$$

Where $\mathcal{A}_i := \{\, j \nsim i : \|\mathbf{v}_i - \mathbf{v}_j\| < \delta \,\}$. Here $j \nsim i$ denotes non-adjacent vertices $(i, j) \notin E$ for edge set E. We set δ to the local grid spacing; see Section 6.3.3 for ablations.

4. Experiments

We use the official implementation provided in the NeuralOperator library [13] which includes the FNO. Regarding the network hyper-parameters, we use 4 Fourier layers, with the number of Fourier modes set to 8 for both spatial dimensions (height and width). The number of hidden channels for intermediate Fourier layers, the lifting layer, the projection layer are set to 64, 256, 64, respectively. We train only at 32×32 resolution, with data pool of 1000 data points updated progressively during the self-supervised training. We set the batch size to 10 and learning rate to 10^{-3} and train the network for 100 epochs, which takes 8 hours on an NVIDIA H100 GPU. For experiments, we use the dataset from [22] which contains trajectories of a square cloth generated by ARCSim given sequences of handle point trajectory, and run the evaluation on NVIDIA A100 GPU. We use the evaluation set which contains 11 trajectories for translation and 4 trajectories for rotation run at 60 fps, which serves as ground truth. We selected various baselines for comparison: 1) the self-supervised cloth model used in PGSfT [32]; 2) Metamizer [36], the neural optimizer baseline with iterative refinement; and 3) MeshGraphNetRP (MGNRP) [22], the supervised baseline trained using PBS data. We use the official checkpoints released by the respective authors, except for Metamizer, which is re-trained exclusively with the cloth data pool to ensure a fair comparison. The physics-based losses in Equation (6) are different from ARCSim, which leads to different meaning of material coefficients in our setting. Hence, we apply grid search to find the corresponding material for PGSfT, Metamizer and FNOPT on the simulated sequences. Unless otherwise noted, the repulsive term is disabled. We refer to Section 6.3.3 for experiments with the repulsive loss.

4.1. SOTA Comparison

Training resolution. We first evaluate FNOPT at the training resolution of 32×32. We measure the chamfer distance (e_{CD}) between the ground truth and the predicted point clouds, each of which is uniformly sampled 10^4 points from the corresponding mesh. As shown in Table 1, compared to other self-supervised methods, FNOPT significantly outperforms PGSfT and surpasses Metamizer in most of the

sequences. MGNRP is trained with supervision on domain-matched ground truth trajectories at the same resolution and therefore attains the lowest e_{CD} in-domain. However, as we will show in later sections, this advantage does not transfer to out-of-domain settings, making it less scalable.

Figure 3 shows the evaluated shapes and their Euclidean distance to the PBS ground truth sequence `xy_v2`. PGSfT suffers from noticeable drift over long horizons resulting in high global errors. In addition, it fails to recover high-frequency wrinkles, leading to unnaturally stiff cloth behavior. Metamizer and FNOPT perform quite well at training, showing good dynamics and wrinkles. Although achieving the best quantitative results, MGNRP exhibits subtle but noticeable temporal jitters (visible in supplementary video). Figure 5 (top) plots the per-frame chamfer distance e_{CD} on `xy_v2`; PGSfT produces significantly higher error than the other methods, showing limited dynamic fidelity and a lack of fine-scale detail.

Super-evaluation. To show our generalizability in multi-resolution, we perform simulation with higher resolutions 64×64 and 100×100. Although using graph-based network that can deal with different mesh topologies, MGNRP fails to simulate at higher resolutions. PGSfT's U-Net implementation supports inference only at the training resolution, cross-resolution results are unavailable. Metamizer also uses U-Net but their implementation can be run on different resolutions. As shown in Table 1, Metamizer is still able to get decent results at 64×64 but struggles at 100×100. Thanks to its FNO architecture, FNOPT can perform super-evaluation automatically at both 64×64 and 100×100. Since the wrinkles and details are simulated more precisely in these cases, it improves the average chamfer distance to 4.8 and 4.7 respectively, bridging the gap with respect to supervised method MGNRP. We show comparison for super-evaluation of sequence `xy_v2` in Figure 4. Metamizer exhibits large errors and temporal instabilities on a finer 64×64 mesh, and diverges at 100×100. MGNRP also results in divergence at both 64×64 and 100×100 resolutions. In contrast, FNOPT produces visually plausible cloth with realistic wrinkles and maintains stable, accurate rollouts across all tested resolutions, demonstrating superior fidelity and resolution-agnostic generalization (more details in supplementary video). Furthermore, the per-frame chamfer distance in the lower figure of Figure 5 demonstrates our stability when simulating higher resolutions. We also computed the relative 3D error between the ground truth (e_{3D}) and predicted meshes; more details can be found in Section 6.1 in supplementary.

Interpretation of the results. We analyze why the different approaches attain varying levels of motion realism, fine-scale detail, and temporal stability. PGSfT [32] is trained in a fully self-supervised fashion. Its one-shot architecture, however, cannot correct accumulated errors, leading to noticeable drift and a loss of high-frequency wrinkles in long rollouts. MGNRP [22] is trained in a supervised fashion on pre-computed ground truth trajectories, achieving high accuracy for motions, velocities, and mesh resolutions seen during training. It can handle irregular meshes due to its use of the MeshGraphNets architecture. However, the model extrapolates poorly, and its performance degrades significantly when tested on finer resolutions or faster motions outside the training distribution. Metamizer [36] improves visual fidelity and wrinkle detail over PGSfT by employing an iterative, meta-learned optimizer. Nevertheless, its performance drops sharply on finer meshes, and rollouts exhibit temporal instabilities. FNOPT produces stable, high-fidelity rollouts across all tested resolutions. We attribute this robustness to the resolution-agnostic FNO backbone, which supports zero-shot inference on previously unseen mesh resolutions, and to the meta-learned optimizer that optimizes per-step acceleration with only a few iterations.

Runtime performance. All timings were measured on an NVIDIA A100 GPU with a 1024-vertex mesh. With the default setting of 10 optimizer iterations, FNOPT runs at 61 ms/frame, and at 33 ms/frame with 5 iterations at the expense of a modest loss in accuracy. Enabling the repulsive term incurs an additional 3 ms/frame. In comparison, the supervised baseline MGNRP completes a frame in 40 ms using the authors' released model, while PGSfT, which requires only a single feed-forward pass, achieves 7 ms/frame but fails to capture fine-scale wrinkles. Metamizer with iterative refinement runs at 63 ms/frame. The offline finite-element solver ARCSim requires 404 ms/frame. Overall, FNOPT offers a compelling trade-off: its per-frame runtime is about $1.5\times$ that of the supervised baseline while delivering substantially higher accuracy on cross-resolution rollouts, and is roughly $6\times$ faster than the PBS.

4.2. Boundary Condition Generalization

Beyond scaling across resolutions, FNOPT remains robust under changes to boundary conditions (e.g. handle speed and placement, mesh sizes, etc), in contrast to baseline methods that struggle in such settings.

Speed generalization. We evaluate the speed generalization ability of FNOPT and compare with the supervised approach MGNRP. We create new motion sequences by accelerating the original ones using interpolation. We test speed factors $\alpha \in \{1.2, 1.5, 2\}$. The results are shown in Figure 6. Both methods are able to handle slight speed increase, as shown in $1.2\times$ with no noticeable artifacts. When speed factor goes to $1.5\times$, MGNRP shows overstretching artifacts around upper corners. With $2\times$ speed, severe artifacts are

Sequence name	MGNRP [22]			PGSfT [32]	Metamizer [36]			FNOpt[Ours]		
	res32	res64	res100	res32	res32	res64	res100	res32	res64	res100
xy_v2	**4.8**	–	–	25.3	7.1	33.6	–	7.1	**5.2**	**4.2**
xy_v2_opp	**5.0**	–	–	25.2	7.7	14.7	12.9	6.5	**5.3**	**4.8**
yz_v2	**2.6**	–	–	10.7	11.6	3.6	77.2	4.9	**3.4**	**5.5**
yz_v2_opp	**3.0**	–	–	13.5	4.9	4.2	–	4.3	**3.2**	**3.6**
xz_v2	**2.7**	–	–	22.5	6.5	**6.6**	–	8.9	**6.6**	**3.9**
xyz_v2	**5.4**	–	–	21.9	6.3	**4.9**	29.9	6.4	**4.9**	**4.3**
xyz_v2_opp	**4.3**	–	–	22.7	6.4	4.9	–	6.4	**4.1**	**5.2**
xyz_v3	**3.5**	–	–	21.4	7.2	10.2	–	6.4	**4.7**	**3.7**
xyz_v3_opp	**3.0**	–	–	20.6	29.2	12.1	–	6.1	**4.8**	**4.6**
xyz_v4	**3.7**	–	–	18.9	7.2	4.8	21.2	6.7	**3.7**	**3.5**
xyz_v4_opp	**3.3**	–	–	19.0	5.8	6.2	–	7.4	**4.2**	**4.7**
rot_h0	**5.4**	–	–	17.5	8.2	21.1	–	6.9	**5.2**	**6.2**
rot_h0_opp	**5.7**	–	–	17.5	9.3	6.8	–	7.6	**6.6**	**5.9**
rot_h1	**5.1**	8.3	–	18.6	7.8	7.6	–	7.6	**4.5**	**6.0**
rot_h1_opp	**5.3**	9.4	–	16.7	8.3	7.7	–	7.5	**5.7**	**6.2**
Avg ± σ	**4.2** ±1.1	–	–	19.5 ±3.9	8.9 ±5.6	10.0 ±7.8	–	6.7 ±1.1	**4.8** ±1.0	**4.7** ±0.9

Table 1. Comparison of the chamfer distance e_{CD} (scaled by 10^3) at different resolutions for each motion sequence. The best and second-best results are shown in **bold** and underline, respectively.

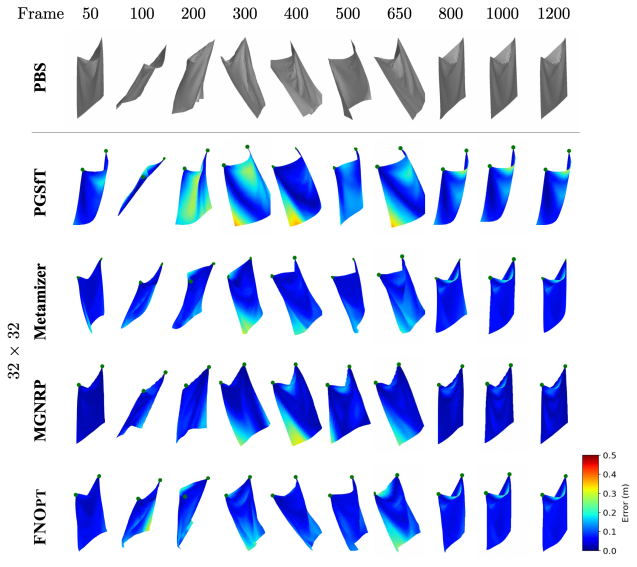

Figure 3. Vertex-wise error maps on the xy_v2 sequence on training resolution. Colors indicate the Euclidean distance between rollout and PBS.

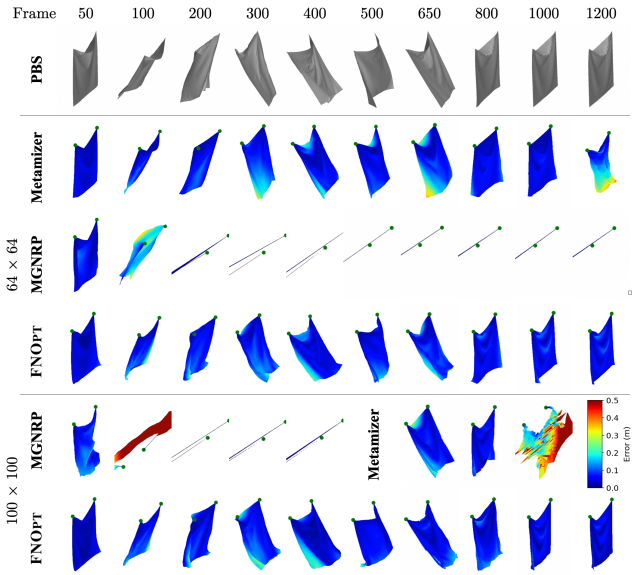

Figure 4. Vertex-wise error maps on finer resolutions on the xy_v2 sequence.

shown and MGNRP simulation becomes unstable. These behaviors happen because the accelerated sequences are not present in the training set, preventing MGNRP from generalizing to out-of-distribution motions. On the other hand, FNOpt remains stable at all tested motion speeds without artifacts around handles.

Flexible handle placement. We evaluate 5 different configurations of the handle points: *Diagonal*, *Mid-Edge*, *Corner-Center*, *Single Mid-Edge* and *Single Center*. *Diagonal* puts the two handles on the opposite corners. For *Mid-Edge*, we fix the handles on two midpoints of the opposite sides of the mesh. *Corner-Center* uses one corner and the center of the cloth as handles. Each of the two settings *Sin-*

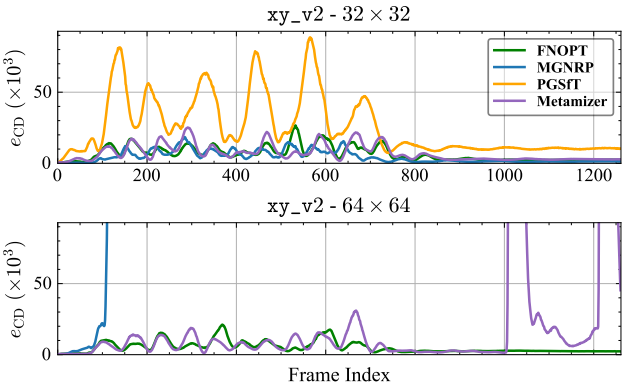

Figure 5. Per-frame e_{CD} ($\times 10^3$) on `xy_v2` sequence. All models are trained at 32×32. FNOPT achieves second lowest error on 32×32 resolution and generalizes to the finer 64×64 resolution.

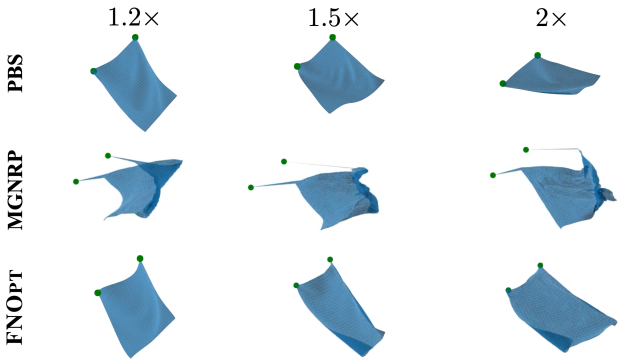

Figure 6. Generalization to speeds on a representative frame of the `xy_v2` rollout. MGNRP exhibits overstretching artifacts near the handles while FNOPT remains stable.

gle Mid-Edge and *Single Center* uses only a single handle point on the midpoint of the upper edge and the center of the cloth, respectively.

The qualitative comparison is illustrated in Figure 7. We can see that FNOPT supports arbitrary handle placements as boundary conditions, yielding stable rollouts with preserved fine-scale wrinkles without retraining or further fine-tuning. MGNRP only supports two-handle configuration with both handles at the top corners, and fails to simulate these 5 configurations. Metamizer utilizes the same formulation of self-supervised neural optimizer as FNOPT, but yields unstable simulation. In addition, FNOPT is generalizable to non-square cloth meshes, see Section 6.3 in supplementary.

4.3. Ablation Studies

Iterative Update. The iterative update discussed in Section 3.3 is crucial for our framework. The improvement is already demonstrated in Metamizer results (see Table 1, Figures 3 and 5), in which the iterative update is employed to achieve higher precision compared to PGSfT. We further train a baseline model using PGSfT formulation that directly predicts the acceleration $\mathbf{a}_{t+1}$ in a single forward pass, without using iterative update in Equation (7). This baseline is trained in a supervised fashion using the same loss in Equation (6), and uses FNO backbone rather than U-Net. As shown in Table 2, this approach yields significantly less accurate rollouts, and fails to capture high-frequency details such as wrinkles.

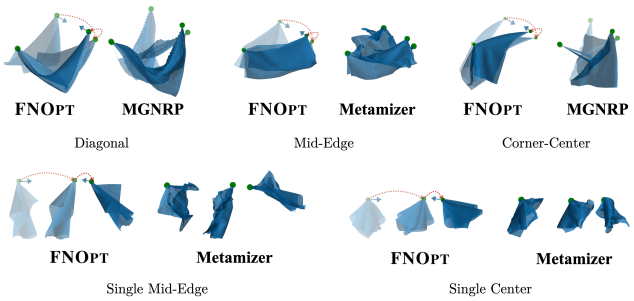

Figure 7. Qualitative comparison of generalization to various handle configurations. The top row shows configurations with two control points. The bottom row contains single-handle scenarios. FNOPT consistently produces plausible and stable cloth dynamics across all settings. Green points denote control handles. Red dashed arrows denote handle trajectories. Blue arrows indicate acceleration directions. Repulsive loss is activated.

Neural Optimizer. To evaluate the neural optimizer formulation in FNOPT framework, we replace the trainable neural optimizer by a classical optimizer using a step size and direction computed from the gradients with respect to the losses in Equation (6). This setup still performs iterative optimization, but lacks any data-driven adaptation of step length or search direction. We evaluate two classical optimizers, Adam [12] and vanilla gradient descent (GD) [6], each with a learning rate sweep, and report their best-performing results in Table 2. Even with a tuned step size, GD fails to minimize $\mathcal{L}_{\text{cloth}}$ effectively and yields poor rollouts. Adam converges given large enough iterations ($\sim$500), but remains $30\times$ slower (see Figure 13) than ours and less accurate in e_{CD}. This shows the necessity of an adaptive, learned optimization scheme. Additional detailed comparison, including runtime, error metrics and losses versus the number of iterations per time step, are provided in Section 6.2 in supplementary.

FNO Architecture. As discussed in Section 3.1, we adopt FNO as the backbone in our framework because its global spectral kernels capture long-range interactions and naturally support resolution-agnostic inference, producing realistic wrinkles and faithful high-resolution details. We have compared FNO with the U-Net backbone used by Metamizer in Section 4.1 and shown a clear advan-

tage in both accuracy and visual quality over various resolution. We further show the benefit of using FNO over MeshGraphNets[27], a popular graph-based simulator in cloth and fluid dynamics. Note that we keep the same formulation as the FNOPT framework and only change the network architecture. We refer to this variant as MGN. Despite using identical loss objectives, iteration counts, and training strategy, the MGN version exhibits visible high-frequency oscillations even at the training resolution and diverges rapidly on 64×64 meshes, see Figure 11 in supplementary.

We hypothesize that the instability arises from the autoregressive rollouts: with a fixed number of message-passing steps, MeshGraphNets can propagate information only within a limited neighborhood, so local errors are re-fed into the next step, propagate and amplify over time. Since the learned optimizer relies on accurate per-vertex gradients, even small per-step misestimations may accumulate and drive the system into unstable regimes. By contrast, the global spectral kernels of FNOs couple every vertex to the entire cloth surface, yielding smoother, more consistent updates and preventing long-horizon error growth.

Super-evaluation. We emphasize that FNOPT framework allows for super-evaluation that is trained on a lower resolution and generalizes directly to higher resolution, capturing finer details that were not presented in the low-resolution training data. This property is essential for physics simulation, especially when simulating cloth with folds and wrinkles. We clarify that such property is different from interpolation, which estimates high-resolution rollouts from lower-resolution prediction. To evaluate, we estimate the 64×64 rollouts by bilinearly interpolating the predicted 32×32 accelerations given by FNOPT. As can be seen in Figure 8, the resulting interpolation recovers fewer fine-scale wrinkles due to information loss during upsampling, showcasing the necessity of *super-evaluation* at the target resolution.

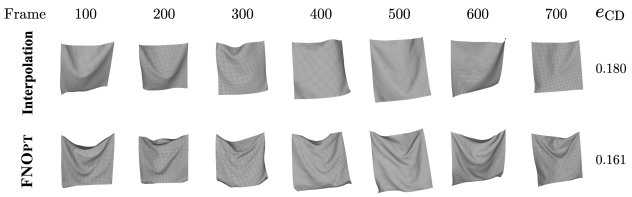

Figure 8. Visual comparison of 64×64 rollouts using bilinear interpolation (upper row) and ours (lower row).

Number of iterations per time step. To analyze the trade-off between iteration count and performance, we evaluate our method using different value of N (iterations per time step) ranging from 3 to 30. Figure 10 in supplementary plots the mean accuracy and runtime per frame curves at three resolutions across the tested values of N. We first observe that the runtime is quasi-linear with respect to the number of iterations. Meanwhile, increasing the number of iterations from 5 to 10 yields a significant accuracy gain. However, increasing to 20 iterations offers no further improvement while doubling the computational cost. The same trend is reflected in the e_{3D} error. See Table 7 in supplementary for the quantitative results of runtimes and Chamfer distances at each iteration count and resolution.

Method	$e_{CD} \downarrow$
FNOPT (Ours)	4.8 ± 0.99
No iterative update	24.7 ± 3.22
GD (lr=0.01, 500 iters)	419.9 ± 114.4
Adam (lr=0.1, 500 iters)	7.7 ± 2.6
Interpolation to hi-res	7.0 ± 1.27

Table 2. Ablation study at 64×64 resolution. We report e_{CD} over all evaluation sequences. Chamfer distances are multiplied by 10^3 for readability.

Table 8 reports the per-sequence accuracy for 5, 10, and 20 iterations per time step across three mesh resolutions. These results suggest that approximately 10 iterations are sufficient for the learned optimizer to converge across all tested mesh resolutions. We therefore adopt $N = 10$ as the default to balance accuracy and efficiency. Runtimes are similar across resolutions, indicating that FNOPT maintains efficient inference across resolutions.

5. Conclusion

We presented FNOPT, a novel resolution-agnostic cloth simulation framework that learns an optimizer from FNOs to simulate cloth dynamics. Extensive experiments demonstrate that FNOPT achieves stable, accurate and efficient cloth dynamics in comparison with previous cloth simulators. Moreover, by leveraging the discretization-invariant capabilities of neural operators, it exhibits significant generalizability across a wide range of mesh resolutions and boundary conditions.

Limitations and future work. While FNOPT scales across mesh resolutions, sizes and handle configurations, the current implementation is designed for rectangular grids. It is therefore not intended to operate directly on irregular or unstructured meshes. Extending the framework to handle arbitrary mesh topology using frameworks such as GINO [20] is left for future work.

References

[1] David Baraff and Andrew Witkin. Large steps in cloth simulation. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, page 43–54, 1998. 1, 2

[2] Hugo Bertiche, Meysam Madadi, and Sergio Escalera. Cloth3d: Clothed 3d humans. In *ECCV*, 2020. 2

[3] Hugo Bertiche, Meysam Madadi, and Sergio Escalera. Neural cloth simulation. *ACM TOG*, 41(6), 2022. 2

[4] Johannes Brandstetter, Daniel Worrall, and Max Welling. Message passing neural pde solvers. *arXiv preprint arXiv:2202.03376*, 2022. 3

[5] David E Breen, Donald H House, and Michael J Wozny. Predicting the drape of woven cloth using interacting particles. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 365–372, 1994. 1

[6] Augustin Cauchy. Méthode générale pour la résolution des systèmes d'équations simultanées. *Comp. Rend. Sci. Paris*, 25:536–538, 1847. 7

[7] Ruochen Chen, Shaifali Parashar, and Liming Chen. Gaps: Geometry-aware, physics-based, self-supervised neural garment draping. In *International Conference on 3D Vision (3DV)*, 2024. 2

[8] Abouzar Choubineh, Jie Chen, David A. Wood, Frans Coenen, and Fei Ma. Fourier neural operator for fluid flow in small-shape 2d simulated porous media dataset. *Algorithms*, 16(1), 2023. 3

[9] Theodore F Gast, Craig Schroeder, Alexey Stomakhin, Chenfanfu Jiang, and Joseph M Teran. Optimization integrator for large time steps. *IEEE transactions on visualization and computer graphics*, 21(10):1103–1115, 2015. 3

[10] Erhan Gundogdu, Victor Constantin, Shaifali Parashar, Amrollah Seifoddini, Minh Dang, Mathieu Salzmann, and Pascal Fua. Garnet++: Improving Fast and Accurate Static 3D Cloth Draping by Curvature Loss. *IEEE TPAMI*, 44(1):181–195, 2020. 2

[11] Mohammad S. Khorrami, Pawan Goyal, Jaber R. Mianroodi, Bob Svendsen, Peter Benner, and Dierk Raabe. A physics-encoded fourier neural operator approach for surrogate modeling of divergence-free stress fields in solids, 2025. 3

[12] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. 7

[13] Jean Kossaifi, Nikola Kovachki, Zongyi Li, David Pitt, Miguel Liu-Schiaffini, Robert Joseph George, Boris Bonev, Kamyar Azizzadenesheli, Julius Berner, and Anima Anandkumar. A library for learning neural operators, 2024. 4

[14] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 24(89):1–97, 2023. 2

[15] Thorsten Kurth, Shashank Subramanian, Peter Harrington, Jaideep Pathak, Morteza Mardani, David Hall, Andrea Miele, Karthik Kashinath, and Anima Anandkumar. Fourcastnet: Accelerating global high-resolution weather forecasting using adaptive fourier neural operators. In *Proceedings of the Platform for Advanced Scientific Computing Conference*, New York, NY, USA, 2023. Association for Computing Machinery. 3

[16] Dohae Lee, Hyun Kang, and In-Kwon Lee. Clothcombo: Modeling inter-cloth interaction for draping multi-layered clothes. *ACM Transactions on Graphics (TOG)*, 42:1 – 13, 2023. 4

[17] Zongyi Li, Nikola B. Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew M. Stuart, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations. *CoRR*, abs/2003.03485, 2020. 1, 2

[18] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations, 2021. 1, 2, 3

[19] Zhijie Li, Wenhui Peng, Zelong Yuan, and Jianchun Wang. Fourier neural operator approach to large eddy simulation of three-dimensional turbulence. *Theoretical and Applied Mechanics Letters*, 12(6):100389, 2022. 3

[20] Zongyi Li, Nikola Kovachki, Chris Choy, Boyi Li, Jean Kossaifi, Shourya Otta, Mohammad Amin Nabian, Maximilian Stadler, Christian Hundt, Kamyar Azizzadenesheli, and Anima Anandkumar. Geometry-informed neural operator for large-scale 3D PDEs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. 8

[21] Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations. *ACM/IMS Journal of Data Science*, 1(3):1–27, 2024. 3

[22] Emmanuel Ian Libao, Myeongjin Lee, Sumin Kim, and Sung-Hee Lee. Meshgraphnetrp: Improving generalization of gnn-based cloth simulation. In *Proceedings of the 16th ACM SIGGRAPH Conference on Motion, Interaction and Games*, New York, NY, USA, 2023. Association for Computing Machinery. 1, 2, 4, 5, 6, 11, 12

[23] Rahul Narain, Armin Samii, and James F. O'Brien. Adaptive anisotropic remeshing for cloth simulation. *ACM TOG*, 31 (6):147:1–10, 2012. 1, 2

[24] Xiaoyu Pan, Jiaming Mai, Xinwei Jiang, Dongxue Tang, Jingxiang Li, Tianjia Shao, Kun Zhou, Xiaogang Jin, and Dinesh Manocha. Predicting loose-fitting garment deformations using bone-driven motion networks. 2022. 2

[25] Chaitanya Patel, Zhouyingcheng Liao, and Gerard Pons-Moll. TailorNet: Predicting Clothing in 3D as a Function of Human Pose, Shape and Garment Style. In *CVPR*, 2020. 2

[26] Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, Pedram Hassanzadeh, Karthik Kashinath, and Animashree Anandkumar. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *arXiv preprint arXiv:2202.11214*, 2022. 3

[27] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. Learning mesh-based simulation with graph networks. In *ICLR*, 2021. 1, 2, 8, 11

[28] Igor Santesteban, Miguel A. Otaduy, and Dan Casas. Learning-Based Animation of Clothing for Virtual Try-On. *Comput. Graph. Forum*, 38(2):355–366, 2019. 2

[29] Igor Santesteban, Nils Thuerey, Miguel A Otaduy, and Dan Casas. Self-Supervised Collision Handling via Generative 3D Garment Models for Virtual Try-On. 2021. 2

[30] Igor Santesteban, Miguel A Otaduy, and Dan Casas. SNUG: Self-Supervised Neural Dynamic Garments. In *CVPR*, 2022. 1, 2, 3

[31] Vikramjit Sidhu, Edgar Tretschk, Vladislav Golyanik, Antonio Agudo, and Christian Theobalt. Neural dense non-rigid structure from motion with latent space constraints. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVI*, page 204–222, Berlin, Heidelberg, 2020. Springer-Verlag. 11

[32] David Stotko, Nils Wandel, and Reinhard Klein. Physics-guided shape-from-template: Monocular video perception through neural surrogate models. In *CVPR*, 2024. 1, 2, 3, 4, 5, 6, 12

[33] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. *SIGGRAPH Comput. Graph.*, 21(4):205–214, 1987. 2, 3

[34] Lokender Tiwari and Brojeshwar Bhowmick. Garsim: Particle based neural garment simulator. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 4472–4481, 2023. 2

[35] Raquel Vidaurre, Igor Santesteban, Elena Garces, and Dan Casas. Fully Convolutional Graph Neural Networks for Parametric Virtual Try-On. *Comput. Graph. Forum*, 2020. 2

[36] Nils Wandel, Stefan Schulz, and Reinhard Klein. Metamizer: a versatile neural optimizer for fast and accurate physics simulations. In *International Conference on Learning Representations (ICLR)*, 2025. 1, 2, 3, 4, 5, 6, 12

[37] Gege Wen, Zongyi Li, Qirui Long, Kamyar Azizzadenesheli, Anima Anandkumar, and Sally Benson. Real-time high-resolution co2 geological storage prediction using nested fourier neural operators. *Energy & Environmental Science*, 16, 2023. 3

[38] Yan Yang, Angela F. Gao, Jorge C. Castellanos, Zachary E. Ross, Kamyar Azizzadenesheli, and Robert W. Clayton. Seismic wave propagation and inversion with neural operators. *The Seismic Record*, 1(3):126–134, 2021. 3

[39] Meng Zhang, Tuanfeng Y. Wang, Duygu Ceylan, and Niloy J. Mitra. Dynamic neural garments. *ACM TOG*, 40 (6), 2021. 2

[40] Tingtao Zhou, Xuan Wan, Daniel Zhengyu Huang, Zongyi Li, Zhiwei Peng, Anima Anandkumar, John F. Brady, Paul W. Sternberg, and Chiara Daraio. Ai-aided geometric design of anti-infection catheters. *Science Advances*, 10(1): eadj1741, 2024. 3

6. Supplementary

6.1. SOTA Comparison

In the main paper, we reported only the Chamfer Distance (e_{CD}) metric due to space limitations. Here, we complement those results by reporting the per-sequence 3D error e_{3D} (defined in [31]) in Table 3, computed using the known vertex correspondences after re-meshing the ground truth. Both metrics show broadly consistent trends across methods, though some variations in relative rankings can be observed.

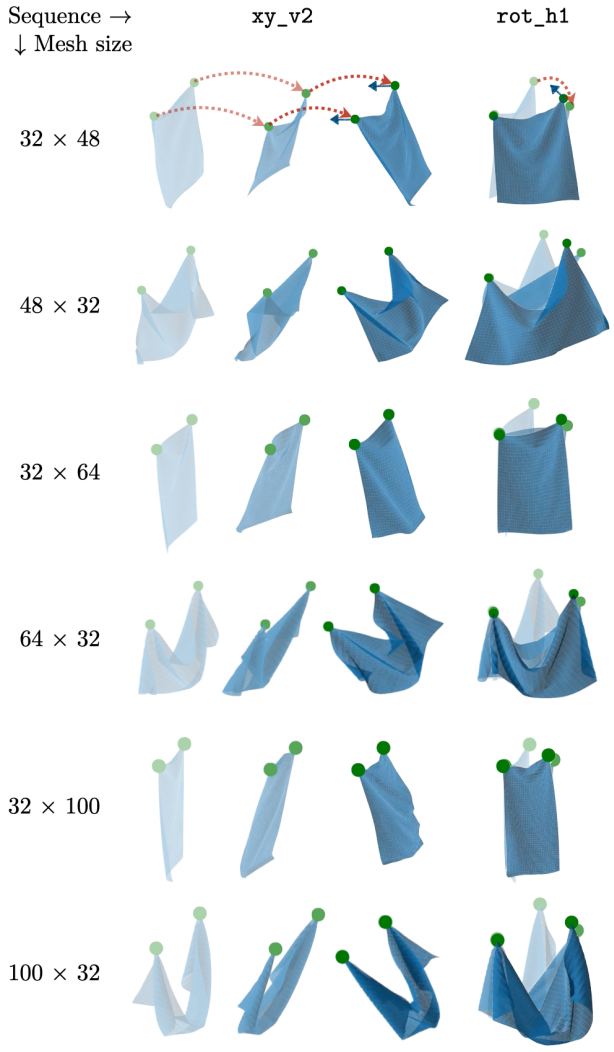

Figure 9. Generalization to non-square cloth meshes.

6.2. Ablation Study

6.2.1. Number of iterations per time step

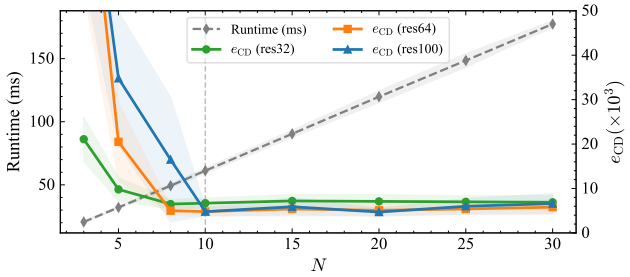

Figure 10. FNOPT runtime (dashed gray line, left axis) and Chamfer distance e_{CD} (solid colored lines, right axis) as functions of the number of iterations per time step. Green, orange, and blue curves correspond to mesh resolutions of 32×32, 64×64, and 100×100, respectively. Shaded bands denote one standard deviation over test sequences.

6.3. Boundary Condition Generalization

Non-square meshes. Trained on square grid of the cloth, FNOPT is able to perform rollouts on grids with non-square aspect ratios. We demonstrate rollouts on six different mesh sizes across two motion sequences in Figure 9. Without any retraining or parameter tuning, the results indicate that the learned optimizer transfers across domain shapes, preserving fine-scale wrinkles and long-horizon stability.

6.3.1. FNO Architecture

We embed MeshGraphNets [27] into the same meta-learning framework, while keeping the losses and training schedule unchanged. We refer to this variant as MGN. For hyper-parameters, we keep the default number of message-passing steps (15); the hidden size and the number of layers for the encoder, graph-net blocks, and decoder are set to 128 and 3, respectively. We train the network for 100 epochs, which takes around 40 hours on an NVIDIA H100 GPU. We evaluate on the evaluation sequences of the datasets from [22]; results are shown in Figure 11. The MGN variant exhibits visible high-frequency oscillations already at the training resolution and diverges rapidly on 64×64 meshes.

6.3.2. Neural Optimizer

We have compared FNOPT with classical first-order optimizers under varying numbers of inner iterations per time step: (a) vanilla gradient descent (GD) and (b) Adam. We additionally incorporate comparison with a quasi-Newton method, (c) limited-memory BFGS (L-BFGS). For each optimizer we run an independent learning rate sweep and report the best performing accuracy in Table 6. We report runtime in Figure 13. All experiments were run on a 64×64 mesh.

Sequence name	MGNRP [22]			PGSfT [32]	Metamizer [36]			FNOPT[Ours]		
	res32	res64	res100	res32	res32	res64	res100	res32	res64	res100
xy_v2	**11.0**	–	–	24.1	15.5	27.8	–	15.4	**12.7**	**11.0**
xy_v2_opp	**11.8**	–	–	23.2	16.2	17.9	16.5	14.5	**12.0**	**10.5**
yz_v2	**9.7**	–	–	20.1	17.6	12.1	34.3	15.0	**11.3**	**13.1**
yz_v2_opp	**8.6**	–	–	21.4	13.8	12.8	–	13.5	**10.3**	**10.9**
xz_v2	**7.9**	–	–	21.8	13.2	**11.9**	–	15.2	13.3	**9.9**
xyz_v2	**10.7**	–	–	21.7	14.7	**11.4**	18.9	14.6	12.0	**10.5**
xyz_v2_opp	**10.6**	–	–	22.7	15.4	11.5	–	14.7	**10.8**	**11.4**
xyz_v3	**9.1**	–	–	22.7	14.5	15.3	–	15.2	**12.2**	**10.6**
xyz_v3_opp	**9.7**	–	–	22.1	23.5	20.7	–	14.3	**11.8**	**10.9**
xyz_v4	**9.4**	–	–	21.9	14.8	11.4	22.1	14.4	**10.7**	**9.9**
xyz_v4_opp	**9.0**	–	–	21.4	13.4	13.3	–	15.2	**11.2**	**11.2**
rot_h0	**12.5**	–	–	23.5	17.0	22.4	–	16.1	**14.3**	**15.1**
rot_h0_opp	**12.8**	–	–	22.9	18.3	**14.8**	–	17.1	15.8	**14.8**
rot_h1	**11.4**	15.4	–	23.8	16.8	14.9	–	16.8	**13.9**	**14.6**
rot_h1_opp	**11.9**	16.4	–	23.2	17.8	**15.0**	–	17.0	15.8	**14.7**
Avg ± σ	**10.4** ±1.4	–	–	22.4 ±1.0	16.2 ±2.5	15.5 ±4.6	–	15.3 ±1.0	**12.6** ±1.7	**11.9** ±1.9

Table 3. Comparison of the 3D error e_{3D} ($\times 10^2$; lower is better) at different resolutions for each motion sequence. Within each resolution group, the best and second-best results are shown in **bold** and underline, respectively.

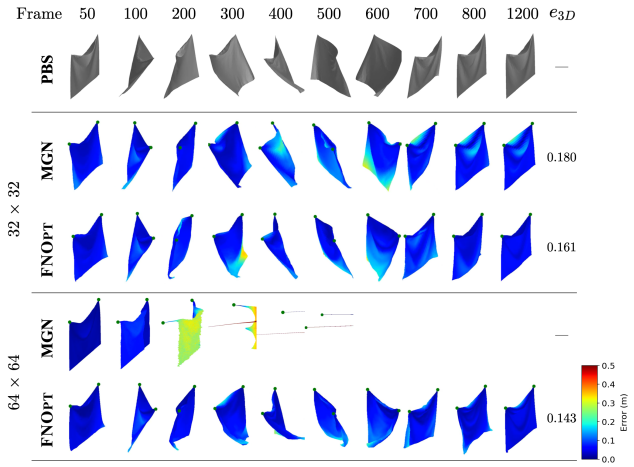

Figure 11. Vertex-wise error maps on the rot_h0 sequence, compared to MGN. Colors indicate the Euclidean distance between rollout and the PBS.

Even with a carefully tuned step size, GD fails to drive the physics loss $\mathcal{L}_{\text{cloth}}$ to a low value with even $N = 500$ iterations per time step, and the resulting roll-outs exhibit visible artifacts.

Thanks to its adaptive step size, Adam converges if a sufficiently large number of inner iterations (roughly 500 per time-step) is allowed. We use standard hyperparameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$. However, its final e_{CD} and e_{3D} are still higher than ours, while being $30\times$ slower.

For L-BFGS, we compute the search direction via the standard two-loop recursion. To equalize gradient evaluations with first-order baselines, we adopt a constant step size and use a history size of m=5; we refer to this baseline as L-BFGS (fixed step, no line search). This variant has $O(N^2)$ time complexity because of iteration over history, where N is the number of iterations per time step, but it turns out to converge in fewer iterations than GD and Adam and achieves the most accurate results with 100 iterations per time step. Yet, this is still $10\times$ slower than ours. Curiously, we evaluate its performance on a finer 100×100 mesh with 70, 100, 120, 150 iterations per time step and record the e_{CD} in Table 4. We find that it achieves the lowest metric at around 120 iterations per time step, and remains at low value for higher N. FNOPT achieves on par performance with only 10 iterations per time step.

Finally, with only ten iterations per time step, FNOPT achieves low error and stable, high-fidelity rollouts while being at least an order of magnitude faster than all classical optimizers. Figure 12 compares of e_{CD} and $\mathcal{L}_{\text{cloth}}$ across classical optimizers and our method as a function of N. Unlike classical optimizers, which require at least 100 iterations to achieve low error, FNOPT reaches both low e_{CD} and $\mathcal{L}_{\text{cloth}}$ within just 10 iterations. Figure 13 shows the runtime comparison between different optimizers, evaluated on a node equipped with an AMD EPYC 7543 CPU and an NVIDIA A100 GPU. This experiment highlights the advantage of a learned, resolution-agnostic optimizer over classical update rules.

Sequence name	L-BFGS (step size = 1)				Ours
	70	100	120	150	10
xy_v2	73.8	7.9	5.3	7.3	4.2
xy_v2_opp	74.9	7.3	5.0	7.2	4.8
yz_v2	44.4	7.5	3.2	2.4	5.5
yz_v2_opp	40.9	7.3	3.3	2.4	3.6
xz_v2	40.6	4.9	5.7	8.1	3.9
xyz_v2	91.1	6.6	4.0	5.0	4.3
xyz_v2_opp	87.7	6.4	3.9	5.2	5.2
xyz_v3	85.7	6.7	3.8	5.1	3.7
xyz_v3_opp	88.1	6.0	3.8	5.3	4.6
xyz_v4	90.5	6.9	3.8	5.1	3.5
xyz_v4_opp	89.5	6.9	3.8	4.8	4.7
rot_h0	27.1	5.1	5.3	8.0	6.2
rot_h0_opp	27.3	5.0	5.6	7.9	5.9
rot_h1	26.9	4.9	5.4	7.8	6.0
rot_h1_opp	26.8	5.1	5.8	7.9	6.2
Avg	58.3	6.1	4.7	5.8	4.7
σ	±25.2	±1.1	±0.9	±1.7	±0.9

Table 4. Ablation on a 100×100 mesh comparing e_{CD} ($\times 10^3$) for L-BFGS and ours under varying *numbers of iterations* per time step.

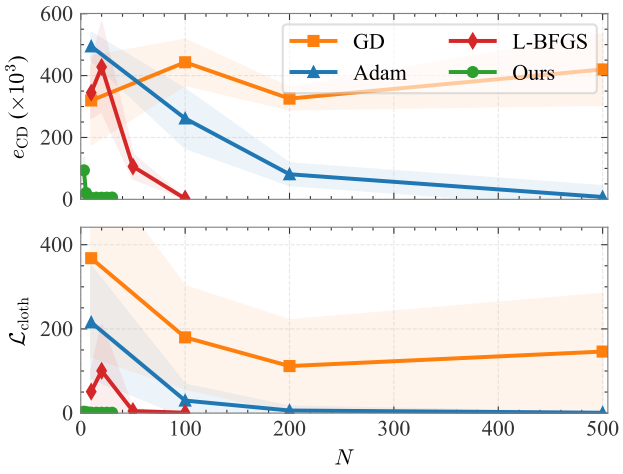

Figure 12. Comparison of e_{CD} and $\mathcal{L}_{\text{cloth}}$ across classical optimizers and our method, versus the number of iterations per time step. Results on 64×64 resolution.

6.3.3. Repulsive loss

To alleviate self-collision, we applied a repulsive loss that penalizes non-adjacent cloth vertices when they are in close proximity. To assess its effectiveness, we consider two challenging scenarios with severe self-collision: a) Corner-center handle placement with motion sequence `rot_h1`, and b) Single Mid-Edge with motion sequence `xy_v2`. Table 5 reports the effect of the repulsive loss: the left column shows the average repulsive loss over the sequence; the middle column reports the proportions of frames with repulsive loss higher than 0.1, and the right column uses a stricter threshold of 0.02. Figure 15 further provides the framewise loss curves together with representative qualitative results, illustrating that the repulsive loss substantially reduces self-collision.

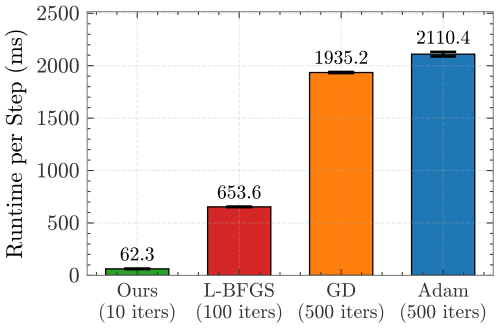

Figure 13. Runtime comparison across different optimizers. Each bar shows per-step runtime (ms); for every optimizer we use the minimum number of iterations that achieves convergence of $\mathcal{L}_{\text{cloth}}$ (except GD, which did not converge). Results on 64×64 resolution.

6.4. Simulation under Varying Material Coefficients

FNOPT can simulate cloth with different stretching, shearing and bending coefficients. We visualize the results under varying stretching and bending coefficients in Figure 14. The simulations are performed at 100×100 resolution with 10 iterations per time step.

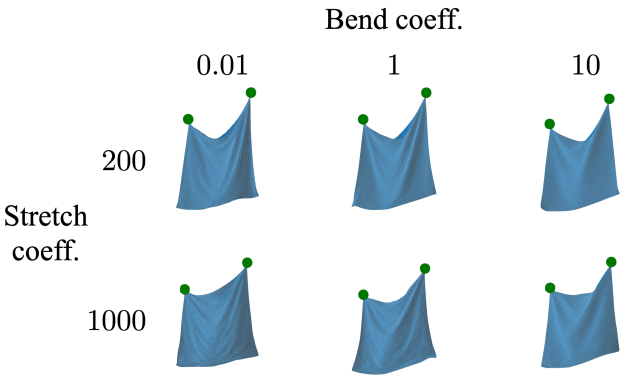

Figure 14. Simulation results of selected frame with different stretching and bending coefficients.

	Corner-Center			Single Mid-Edge		
	$\mathcal{L}_{\text{rep}}(\times 10^3) \downarrow$	% (0.1) $\downarrow$	% (0.02) $\downarrow$	$\mathcal{L}_{\text{rep}}(\times 10^3) \downarrow$	% (0.1) $\downarrow$	% (0.02) $\downarrow$
with $\mathcal{L}_{\text{rep}}$	13.12 ± 4.37	0.00	0.07	0.11±0.26	0.00	0.00
no $\mathcal{L}_{\text{rep}}$	598.53 ± 141.29	97.95	98.08	508.79 ± 182.38	97.60	97.74

Table 5. Ablation study on the repulsive loss. We report average repulsive loss (scaled by 10^3 for readability) and the percentage of frames with loss exceeding thresholds (0.02, 0.1).

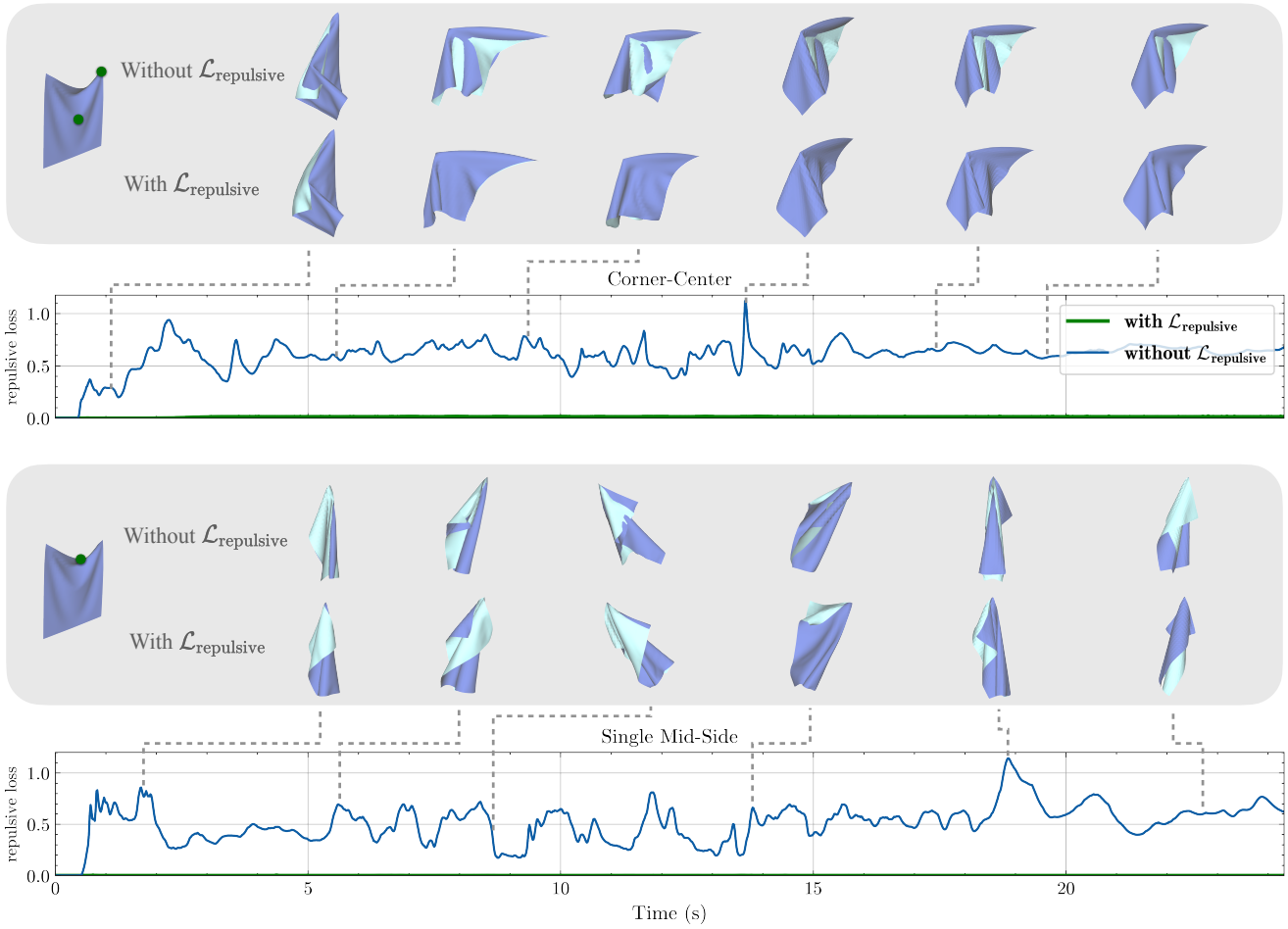

Figure 15. Framewise repulsive loss with and without $\mathcal{L}_{\text{rep}}$. A subset of frames is shown for visual comparison: results without $\mathcal{L}_{\text{rep}}$ (left) versus with $\mathcal{L}_{\text{rep}}$ (right).

Sequence name	GD (lr = 0.01)				Adam (lr = 0.1)				L-BFGS (step size = 1)				Ours
	10	100	200	500	10	100	200	500	10	20	50	100	10
xy_v2	390.2	492.8	313.3	463.0	479.1	338.1	96.8	9.4	405.1	580.2	120.2	3.1	5.2
xy_v2_opp	389.4	499.6	316.3	466.8	480.0	320.4	115.6	7.7	400.3	579.5	119.3	3.2	5.3
yz_v2	491.4	434.4	317.0	493.2	550.8	314.7	125.8	11.7	380.1	443.0	99.0	2.3	3.4
yz_v2_opp	130.0	247.5	213.0	401.5	454.7	282.6	105.5	8.0	270.0	438.3	89.7	2.4	3.2
xz_v2	282.9	417.9	364.0	475.9	527.1	224.8	5.8	6.8	435.0	553.7	126.0	2.5	6.6
xyz_v2	540.6	507.0	352.6	568.5	561.7	377.6	101.4	13.9	450.1	511.1	150.6	1.9	4.9
xyz_v2_opp	248.7	366.9	309.1	435.1	453.7	304.1	116.5	10.6	350.6	500.5	140.8	2.5	4.1
xyz_v3	251.2	367.6	309.3	433.8	448.4	327.7	98.6	5.1	350.8	501.3	134.9	2.5	4.7
xyz_v3_opp	535.0	505.7	353.0	566.4	585.4	355.0	107.6	5.4	430.2	505.5	147.7	2.8	4.8
xyz_v4	536.0	504.7	354.8	566.9	537.7	291.8	91.1	5.2	424.2	508.1	145.2	2.0	3.7
xyz_v4_opp	251.3	367.2	313.3	433.9	446.6	327.7	81.3	5.4	363.1	499.8	132.0	3.1	4.2
rot_h0	180.5	483.9	341.7	247.7	473.6	116.6	46.9	6.4	225.4	204.3	47.8	1.6	5.2
rot_h0_opp	193.2	486.5	344.0	250.3	471.5	147.0	51.6	5.9	228.5	206.3	46.8	1.6	6.6
rot_h1	180.6	481.4	340.2	248.7	463.2	91.8	42.8	8.2	223.4	186.1	47.1	1.7	4.5
rot_h1_opp	192.0	485.2	342.6	246.7	482.2	103.7	29.8	5.4	235.7	199.3	46.2	1.7	5.7
Avg	319.5	443.1	325.6	419.9	494.4	261.5	81.1	7.7	344.8	427.8	106.2	2.3	4.8
σ	±142.4	±73.3	±35.2	±114.4	±44.0	±95.0	±35.2	±2.6	±82.1	±143.1	±39.2	±0.5	±1.0

Table 6. Ablation study on a 64×64 mesh comparing e_{CD} ($\times 10^3$; lower is better) for different optimizers (GD, Adam, L-BFGS and ours) under varying *numbers of iterations* per time-step.

N	Runtime (ms)			$e_{\text{CD}} \downarrow$		
	res32	res64	res100	res32	res64	res100
3	20.6 ±0.7	20.6 ±1.1	21.9 ±0.9	21.1 ±4.9	93.7 ±38.9	93.1 ±35.9
5	32.1 ±0.7	32.2 ±1.1	32.4 ±0.4	9.8 ±2.6	20.5 ±8.7	34.8 ±15.0
8	49.3 ±1.3	52.3 ±2.2	50.8 ±4.9	6.5 ±1.0	5.0 ±1.0	16.5 ±14.0
10	61.2 ±2.1	62.3 ±2.3	61.7 ±1.3	6.7 ±1.1	4.8 ±1.0	4.7 ±0.9
15	90.4 ±2.7	91.7 ±3.2	90.8 ±1.3	7.2 ±1.4	5.4 ±1.2	5.9 ±2.0
20	119.8 ±4.5	119.3 ±2.2	120.4 ±3.6	7.1 ±1.3	5.1 ±1.3	4.7 ±0.8
25	148.5 ±5.2	147.3 ±1.8	149.6 ±1.8	7.0 ±1.3	5.4 ±1.2	6.0 ±1.7
30	177.4 ±2.5	177.5 ±3.5	178.7 ±3.6	6.9 ±1.5	5.8 ±1.6	6.6 ±2.2

Table 7. Ablation study with $N \in \{3, 5, 8, 10, 15, 20, 25, 30\}$ iterations per time-step across three mesh resolutions. We report the per-frame runtime (ms) and e_{CD} ($\times 10^3$).

Sequence name	32×32			64×64			100×100		
	5 iters	10 iters	20 iters	5 iters	10 iters	20 iters	5 iters	10 iters	20 iters
xy_v2	6.2 (16.1)	7.0 (15.2)	7.6 (15.5)	19.2 (24.4)	5.2 (12.7)	5.2 (12.4)	64.5 (46.5)	4.2 (11.0)	4.8 (10.5)
xy_v2_opp	5.9 (14.7)	6.5 (14.5)	7.7 (14.9)	24.7 (24.0)	5.3 (12.0)	5.6 (12.1)	32.8 (30.5)	4.8 (10.5)	5.5 (10.9)
yz_v2	11.3 (21.2)	4.9 (15.0)	4.0 (13.9)	8.3 (17.6)	3.4 (11.3)	3.3 (10.8)	19.9 (24.3)	5.5 (13.1)	3.4 (10.8)
yz_v2_opp	11.0 (19.4)	4.3 (13.5)	5.2 (13.8)	6.4 (15.1)	3.2 (10.3)	2.9 (09.8)	38.7 (27.7)	3.6 (10.9)	3.6 (09.5)
xz_v2	16.6 (21.6)	8.9 (15.2)	8.6 (14.9)	21.3 (22.7)	6.6 (13.3)	6.1 (10.7)	24.3 (23.6)	3.9 (9.9)	5.4 (10.3)
xyz_v2	10.0 (18.5)	6.4 (14.6)	6.4 (14.1)	14.0 (20.5)	4.9 (12.0)	4.3 (10.5)	44.4 (31.0)	4.3 (10.5)	4.0 (8.6)
xyz_v2_opp	9.1 (17.7)	6.4 (14.7)	5.3 (13.1)	16.5 (18.6)	4.1 (10.8)	4.5 (12.7)	27.4 (27.0)	5.2 (11.4)	4.4 (10.9)
xyz_v3	12.0 (19.4)	6.4 (15.2)	5.7 (13.2)	27.6 (24.4)	4.7 (12.2)	4.4 (12.1)	41.1 (31.8)	3.7 (10.6)	4.3 (10.7)
xyz_v3_opp	12.1 (19.6)	6.1 (14.3)	6.9 (14.8)	18.8 (21.3)	4.8 (11.8)	4.5 (10.7)	27.4 (23.8)	4.6 (10.9)	5.0 (10.0)
xyz_v4	6.5 (15.8)	6.7 (14.4)	8.3 (15.5)	21.4 (22.7)	3.7 (10.7)	4.8 (11.9)	29.5 (29.4)	3.5 (9.9)	4.0 (10.0)
xyz_v4_opp	8.7 (17.5)	7.4 (15.2)	7.2 (14.7)	25.3 (22.9)	4.2 (11.2)	4.0 (10.9)	31.1 (31.8)	4.7 (11.2)	4.1 (10.5)
rot_h0	9.1 (20.4)	6.9 (16.1)	7.7 (16.8)	14.3 (22.1)	5.2 (14.3)	7.5 (15.8)	18.3 (26.3)	6.2 (15.1)	4.6 (12.1)
rot_h0_opp	8.5 (19.6)	7.6 (17.1)	9.5 (18.3)	33.3 (22.3)	6.6 (15.8)	6.8 (14.2)	16.7 (25.0)	5.9 (14.8)	5.8 (13.3)
rot_h1	10.2 (21.0)	7.6 (16.8)	8.4 (17.5)	40.8 (31.1)	4.5 (13.9)	6.2 (14.5)	70.4 (54.3)	6.0 (14.6)	5.1 (13.2)
rot_h1_opp	9.2 (21.1)	7.5 (17.0)	8.7 (17.9)	16.3 (25.9)	5.7 (15.8)	7.1 (15.7)	36.1 (34.4)	6.2 (14.7)	6.0 (13.5)
Avg	9.8 (0.189)	6.7 (0.153)	7.2 (0.153)	20.5 (0.224)	4.8 (0.126)	5.1 (0.124)	34.8 (0.312)	4.7 (0.119)	4.7 (0.110)
σ	±2.6 (±0.021)	±1.1 (±0.010)	±1.5 (±0.016)	±8.7 (±0.036)	±1.0 (±0.017)	±1.3 (±0.019)	±15.0 (±0.083)	±0.9 (±0.019)	±0.8 (±0.014)

Table 8. Per-sequence e_{CD} ($\times 10^3$) and e_{3D} error ($\times 10^2$; values in parentheses) at 5, 10, and 20 iteration budgets, evaluated on three mesh resolutions.