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.

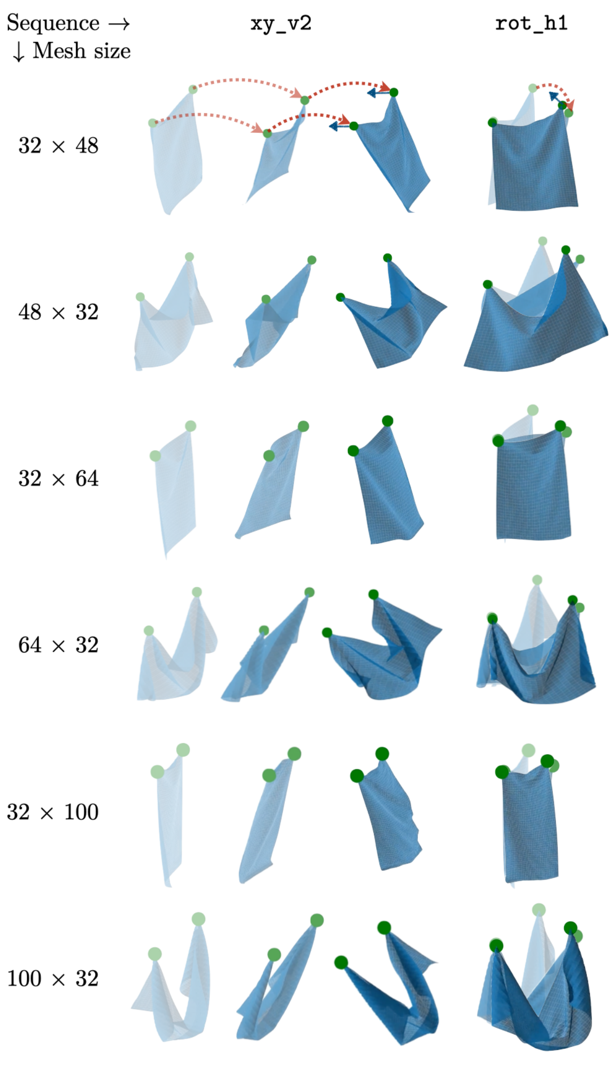

Figure 9. Generalization to non-square cloth meshes.

6.2. Ablation Study

6.2.1. Number of iterations per time step

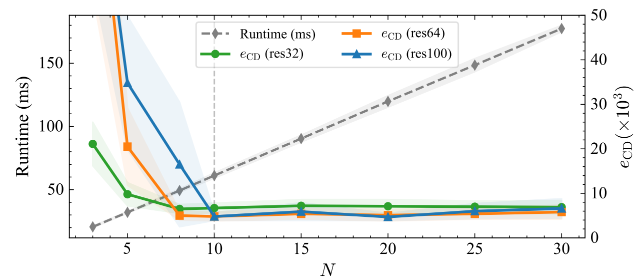

Figure 10. FNOpt runtime (dashed gray line, left axis) and Chamfer distance e_{CD} (solid colored lines, right axis) as functions of the number of iterations per time step. Green, orange, and blue curves correspond to mesh resolutions of 32×32, 64×64, and 100×100, respectively. Shaded bands denote one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

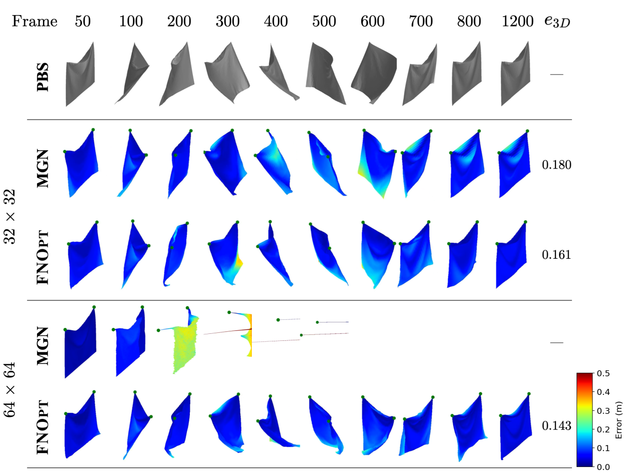

Figure 11. Vertex-wise error maps on the rot_h0 sequence, compared to MGN. Colors indicate the Euclidean distance between rollout and the PBS.

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× 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× 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) 70	100	120	150	Ours 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	$\pm$25.2	$\pm$1.1	$\pm$0.9	$\pm$1.7	$\pm$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.

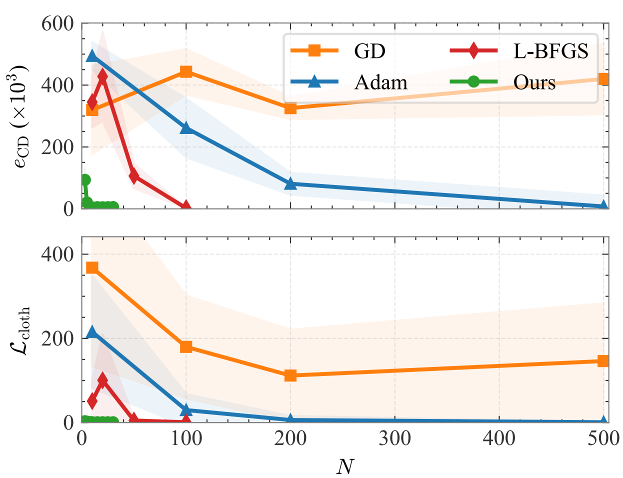

Figure 12. Comparison of e_{CD} and $\mathcal{L}_{\mathrm{cloth}}$ across classical optimizers and our method, versus the number of iterations per time step. Results on 64×64 resolution.

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.

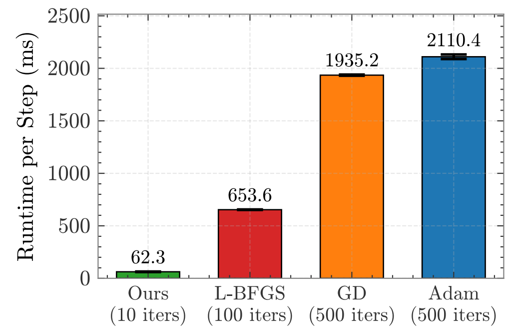

Figure 13. Runtime comparison across different optimizers. Each bar shows per-step runtime (ms); for every optimizer we use the minimum number of iterations that achieves convergence of $\mathcal{L}_{\mathrm{cloth}}$ (except GD, which did not converge). Results on 64×64 resolution.

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.

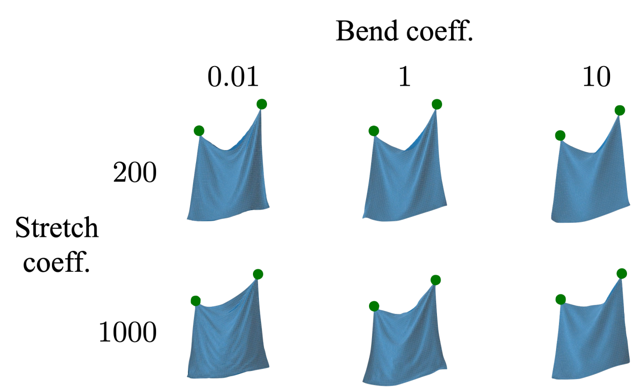

Figure 14. Simulation results of selected frame with different stretching and bending coefficients.

	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).

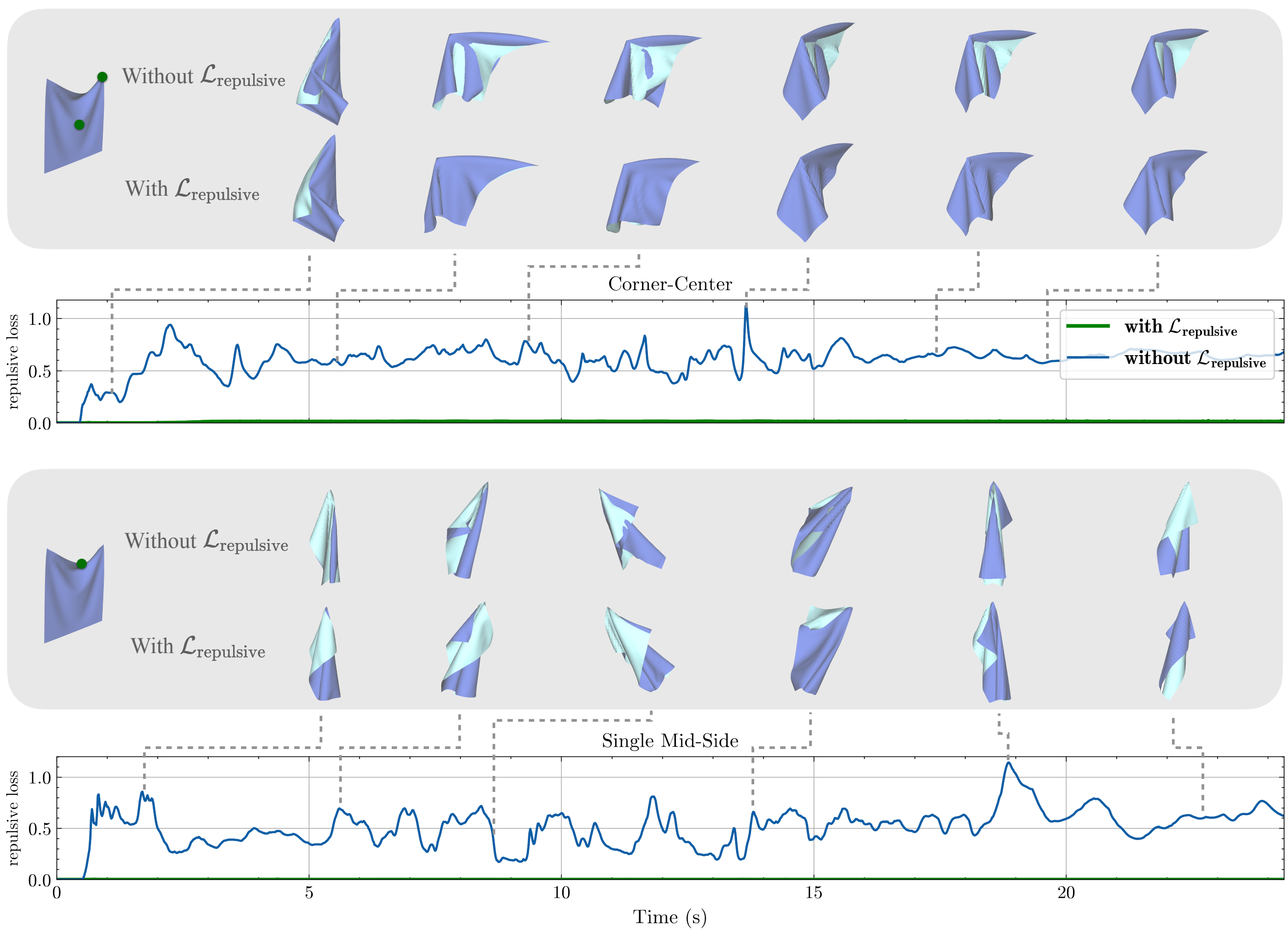

Figure 15. Framewise repulsive loss with and without $\mathcal{L}_{\text{rep}}$. A subset of frames is shown for visual comparison: results without $\mathcal{L}_{\text{rep}}$ (left) versus with $\mathcal{L}_{\text{rep}}$ (right).

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)			$e_{\mathrm{CD}} \downarrow$		
	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.