Supplementary Materials: SurfsUp : Learning Fluid Simulation for Novel Surfaces (Paper ID 12656)

Contents

1 Videos for Rollouts in Paper

Please see the **rollout_videos** folder for full videos corresponding to all of the rollouts in the paper. We have included rollouts for complex scenes (room, coral, and notably multiple videos of the mountain scene in the `Mountain_videos` folder); also see several of the Shapenet objects. We have also included rollouts for objects in the primitives dataset and in the OOD test sets (e.g. primitives+unions). The website also includes several of the rollouts.

2 Comparing SurfsUp to the GNS Baseline – further analysis

Here we include further details of our comparison to the GNS baseline.

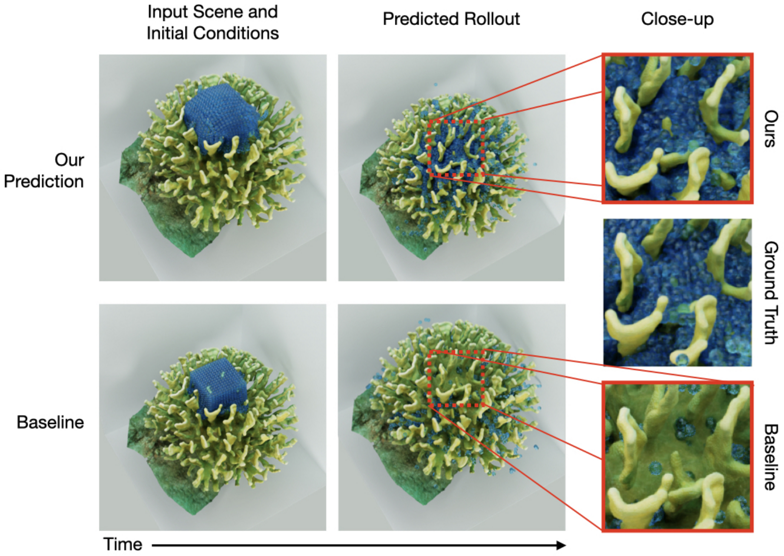


Figure 1: Comparing the baseline and our method on the coral object. Our method filters water through the fine structure of the coral while the baseline cannot account for the geometry.

2.1 Success modes of SURFSUP vs. GNS Baseline – qualitative results

We illustrate in Fig. 1 that GNS is unable to handle the resolution required to accurately model fluid interacting with the coral (discussed in main paper, Sec. 5). As a result the fluid falls through the finer tentacles and misses the geometry entirely. By contrast, our method endows surfaces with infinite resolution inside the fluid simulator, leading to accurate fine-grained fluid dynamics on the coral and other objects. This illustrates the limits of explicit methods which are transcended by our approach for handling complex geometry in larger scenes.

We also note that SURFSUP scales accurately to large scenes. As discussed in the paper, GNS scales up with the number of surface particles needed for the scene. Since particle size is constant at radius r (for a desired numerical accuracy of the simulation), an $N \times N \times N$ scene requires approximately $O(N^3)$ particles to represent at full resolution. By contrast, we can represent large scenes with reasonably small changes to the SDF network f_θ's capacity. Below is a reproduction of Fig. 7 in the main paper, showing that our method handles the large room scene accurately while the baseline's predictions cause fluid to leak into the furniture surfaces due to insufficient resolution. See the **videos** (folder `Complex_scenes (vs. baseline)`) for full rollouts on both the coral and room scenes.

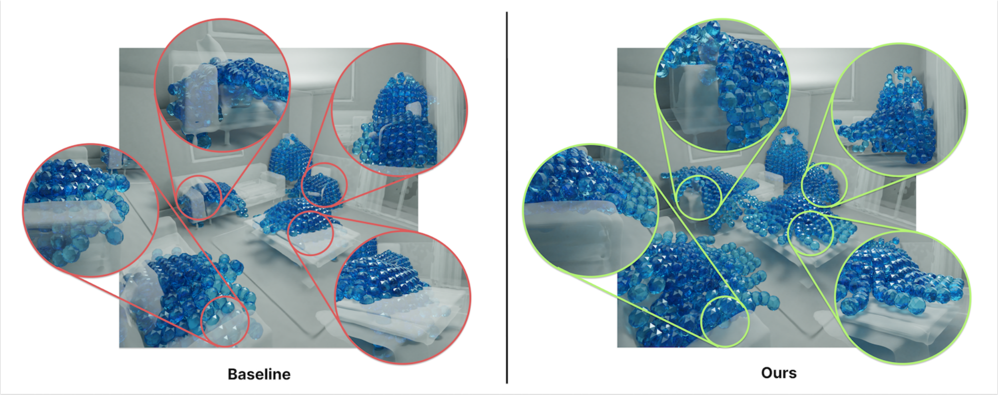


Figure 2: Comparing SURFSUP and the GNS baseline on the room scene, highlighting different areas of the scene. Reproduction of Fig. 7, main paper.

2.2 Quantitative Comparison to GNS on Complex Scenes

In the main paper we compare SurfsUp and GNS on several complex scenes ('Complex-Scenes' dataset, Ln **TODO**). The objects included the coral, room, and lion scene in the main paper, as well as four other objects (one other lion statue, and the armadillo and two dragon objects in the Stanford 3D Scanning Repository). We show the average metrics in the main paper (Table 1, Row 5), and show the results for each object in Table 1 below. SurfsUp significantly outperforms the baseline on *all* objects/scenes and *all* metrics, evidencing its ability to handle large scenes and complex object geometry.

Objects/Metrics	Chamfer	Chamfer Surface	Surface Penetration	Mean Inside Surface
Coral (Ours)	**0.096**	**119.79**	**6782**	**-0.0028**
Coral (Baseline)	0.380	602.66	102783	-0.051
Lion #1	**0.053**	**62.3233**	**215685**	**-0.0256**
	0.380	602.66	102783	-0.0507
Room Scene	**0.1153**	**125.3949**	**9526**	**-0.00654**
	1.1133	877.5509	36106	-0.0181
Lion #2	**0.0437**	**16.7388**	**16899**	**-0.0153**
	0.0645	74.6547	379329	-0.0273
Dragon #1	**0.046**	**9.6486**	**490**	**-0.000376**
	0.0593	39.5942	160354	-0.0219
Dragon #2	**0.0412**	**8.5804**	**202**	**-0.01488**
	0.0467	18.4627	36290	-0.0169
Armadillo	**0.0515**	**11.1678**	**34005**	**-0.0131**
	0.058	21.3429	60364	-0.0248

Table 1: Comparing performance of SurfsUp and the GNS baseline each object in 'Complex-Scenes'.

2.3 Improving Efficiency of Simulation

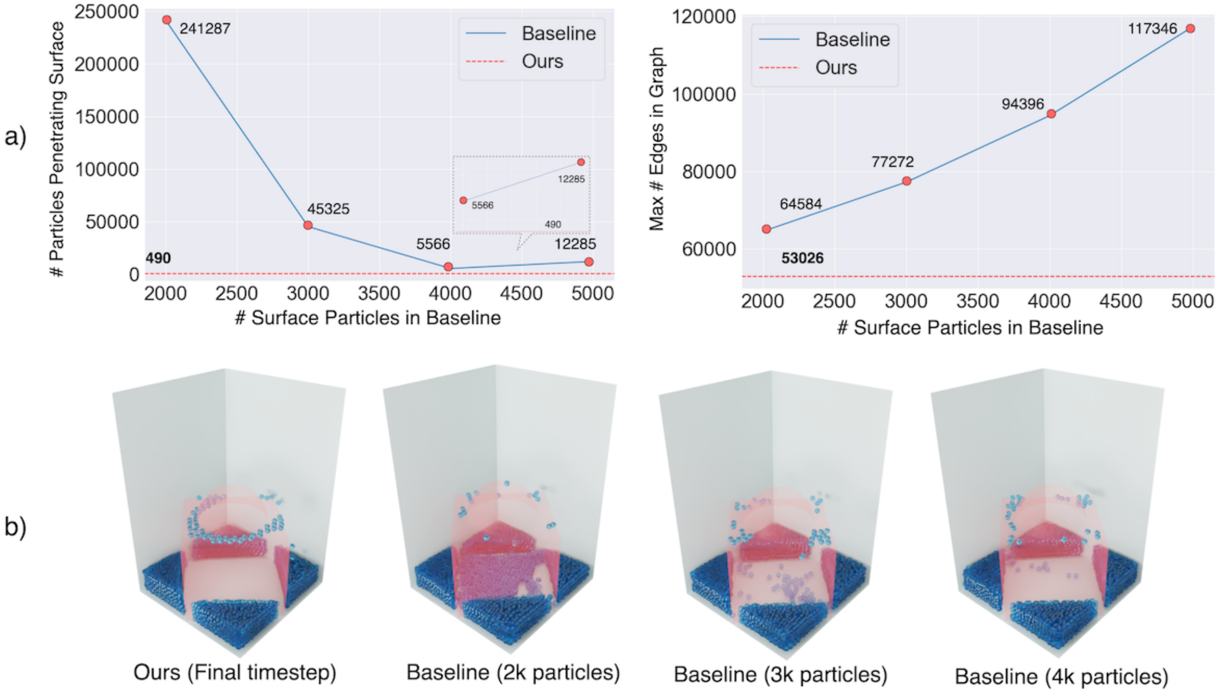
In the main paper we show how SurfsUp compares to a baseline with more surface particles. For the baseline (not for our method) there is an inherent tradeoff: more surface particles increases surface resolution and simulation accuracy, while also increasing the size of the particle graphs. Our method overcomes this tradeoff quite starkly: SurfsUp outperforms even a 10000-particle baseline on the room scene that would incur significant computational cost. Below we include a larger-size copy of Fig. 7 in the main paper (a scene in the Primitives+Unions test set). In Fig. 4 we show how SurfsUp on the room scene outperforms higher-resolution baselines (125.39 Chamfer Surface vs. 194.98 for the 10k-particle baseline). Note finally that SurfsUp successfully simulates a mountain scene (Fig. 8, main paper) that would be essentially impossible for GNS (requiring >1m particles), emphasizing our ability to scale where particles cannot.

2.3.1 Brief Study: Inference Times

Inference times can be highly variable, and we choose to report graph sizes in the main paper as a more reliable measure of computational cost. To provide a brief and rough timing comparison, we broadly find that SurfsUp 's speed for fluid rollouts is faster than the GNS baseline; experiments (benchmarked on a single A6000 GPU) suggest that we are 37% faster then the baseline on the PrimShapes test set and 13% faster on the Chairs test set (where SurfsUp involves evaluating neural SDFs). This can be attributed to the fewer particles and smaller graph sizes entailed by our method compared to the baseline. Naturally this advantage becomes more evident with the size and complexity of the scenes. We find both methods 2-3x faster than the classical simulator (evaluated on a 40-core CPU).

3 Rollouts on Other Test Sets

Figure 5 shows rollouts of our method on the other test sets we describe in the paper, such as the primitives test set (Fig. 5a, the primitives+unions test set (Fig. 5c), funnels (Fig. 5d), and ShapeNet chairs (Fig. 5b). For all examples, our



Ours (Final timestep) Baseline (2k particles) Baseline (3k particles) Baseline (4k particles)

Figure 3: Reproduction of Fig. 7 in the main paper, larger size.

.

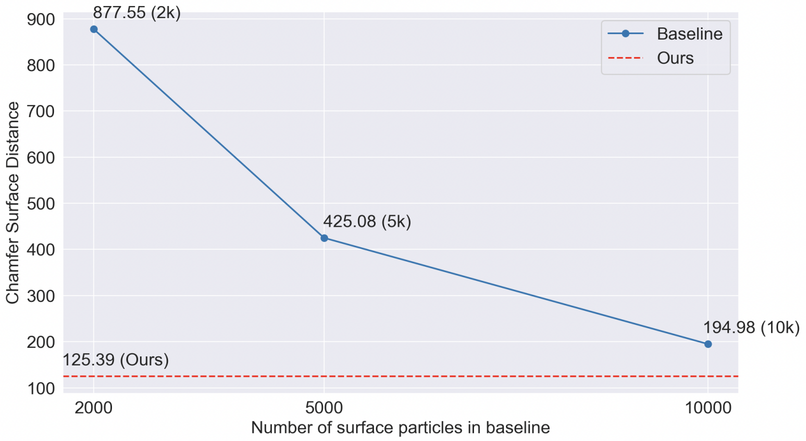


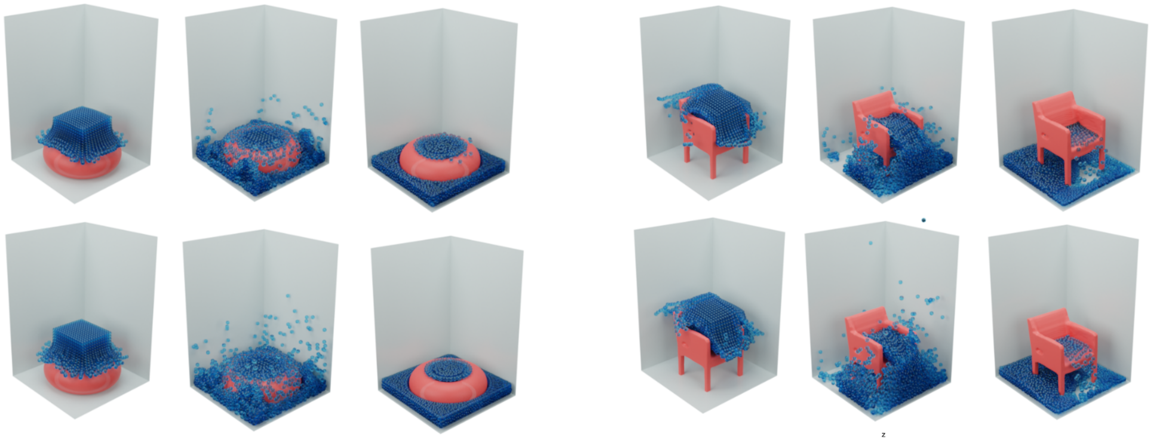
Figure 4: Comparing our method vs. the baseline on the room scene, as the number of surface particles increase from 2000 to 10000. The metric is the Chamfer Surface Distance.

.

predictions closely match the ground-truth fluid trajectory, indicating the accuracy and realism of our fluid rollouts.

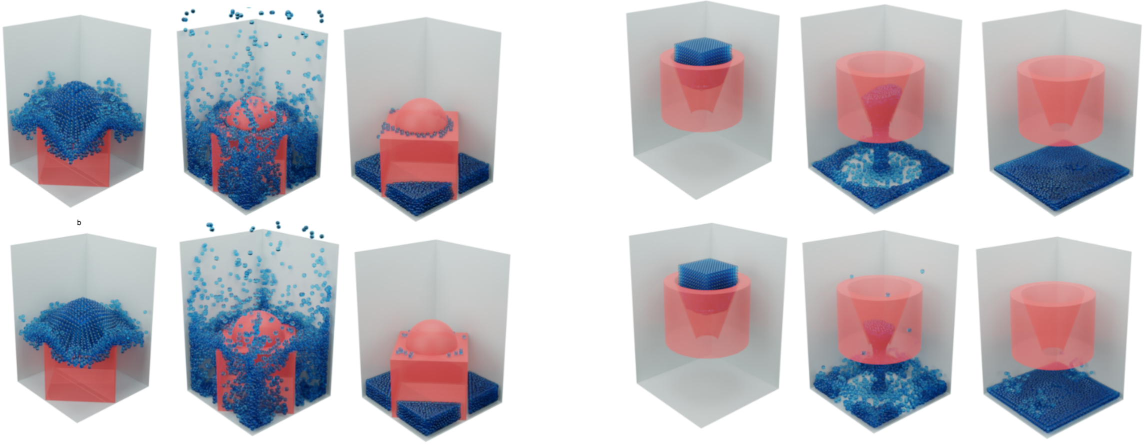
4 Ablation Study

We performed an ablation study on the SDF features provided to the model during training (Table 2). We found that providing only the SDF value and not the gradient (which conveys the particle's orientation w.r.t. the surface), degraded performance near the surface significantly; for example penetration went from 0 to > 30000 particles. Providing no SDF information at all and only fluid particles performs very poorly as expected.



(a) Rollout on a torus.

(b) Rollout on chair (OOD)



(c) Rollout on a union of primitives (OOD)

(d) Rollout on funnels (OOD)

Figure 5: Predicted rollouts of SurfsUp for several examples in our test sets. For each rollout, the top row shows our *prediction* and the bottom row shows the ground-truth simulation.

	Chamfer	Cham-Surface	Num Inside
SurfsUp	0.0297	4.704	0
Value-only	0.0319	10.782	30433
No SDF	0.0815	45.463	193071

Table 2: Ablating SDF features on the PrimShapes test set.

5 Datasets: Details

We provide further details about our dataset creation. To sample trajectories for all datasets we use a "classical" (non-neural) particle-based fluid simulator. In particular, we use the SPlisHSPlasH simulator [1], a 3D particle-based fluid simulator based on the SPH technique. We extract 800 time-steps from each simulation (with $\Delta t = 0.005$); this corresponds to 4000 time-steps of the classical simulator ($\Delta t = 0.001$), which requires smaller time-steps for physical stability. In all our simulations we simulate water.

5.1 Prim-Shapes Dataset

The Prim-Shapes Dataset consists of five primitive shapes: spheres, boxes, cylinders, cones, and toruses. There are 1000 simulations in the training set (200 for each shape), and 100 in the test set. For each simulation, we initialize a $1 \times 1 \times 2$ m box containing the simulation, generate the primitive shape with random shape parameters, apply random rotations and translation, and initialize a block of fluid above the shape. The primitive shapes are parametrized as follows:

- Sphere: Parametrized by radius r, chosen uniformly in the range [0.25, 0.5].
- Box: Parametrized by side lengths s_x, s_y, s_z, each chosen uniformly in the range [0.4, 0.7].
- Cylinder: Parametrized by radius r chosen in range [0.15, 0.35] and height h chosen in range [0.4, 0.75].
- Cone: The 'capped cone' is parametrized by height h ([0.4, 0.7]), its bottom radius r_1 ([0.2, 0.4]) and the top radius r_2 (with r_1 > r_2). r_2 is fixed at 0.01 to approximate a true cone.
- Torus: The torus is constructed by revolving a circle with 'inner' radius' r_2 around an axis of revolution coplanar with the circle, where the 'outer radius' r_1 is the distance from the axis to the center of the torus. The ranges are [0.2, 0.4] for r_1 and $[r_1/4, 3r_1/4]$ for r_2, ensuring there is a hole in the torus.

See the excellent resource here for the expressions of these SDFs. Each shape is rotated randomly, then translated to the bottom of the container and randomly on the container bottom. When computing the SDF, these operations are encoded by inverse transformations on the query point. For the classical simulator (which takes in meshes and re-converts to SDFs), we apply these transformations to a mesh generated from the SDF via Marching Cubes. Finally a block of fluid (each dimension random in [0.4, 0.5]) is initialized at a random height directly above the shape. Since $r = 0.015$ is the particle radius, this generates 1900-3800 particles per simulation. Example initializations illustrating the dataset across the five shapes are shown in Figure 6.

5.2 Primitives-Unions

The Prim-Unions test set is generated by applying pairwise unions to all five shapes. There are a total of 5C2 = 10 combinations. For each combination, we choose one shape as the 'base shape' and union the other shape so that it sits on top of the base shape. See Fig. 5c for an example.

5.3 Complex-Scenes, SIREN training

The Complex-Scenes dataset consists of seven scenes decribed in Sec. 2.2. For each scene, we train a SIREN model [4], where the input is an xyz oriented point cloud of the object or scene. SIREN training produces fine-grained implicit representations with better detail and higher-quality gradients; SurfsUp can utilize these increasingly high-quality neural implicits for predicting dynamics. The SIREN scales the object into a cube with side length 2; we situate this

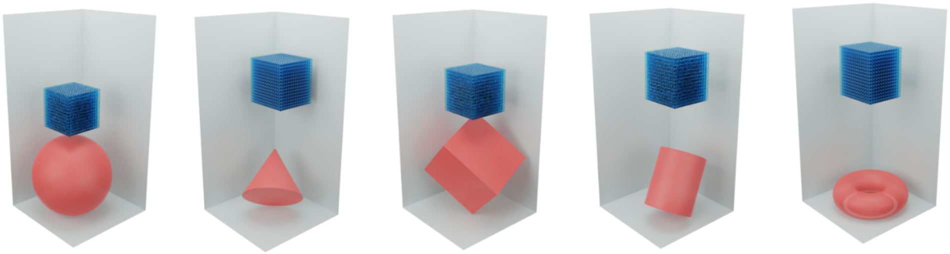


Figure 6: Examples of initial states in the PrimShapes dataset, for the five different shapes in our dataset.
.

object inside a $2 \times 2 \times 3$ container and drop fluid above the object. This is the setup for the six objects (Fig. 4 main paper, left); the exception is the room scene, where we drop fluid inside the room. Please see the github for SIREN training details.

5.4 ShapeNet chairs/bowls, DeepSDF training

We also train a DeepSDF model [2] on the chair and bowl categories in ShapeNet. DeepSDF is a variational 'auto-decoder' model, which learns latent codes $\{z_i\}$ for the training shapes along with the weights of an SDF network $F_\theta(p, z)$. A Gaussian prior is applied to the latent codes so that they cluster around the origin. We utilize the training pipeline in [2] for extracting SDFs from triangle meshes and sampling points for training supervision (e.g. oversampling points near the surface). We train DeepSDF on a total of 5827 chairs from ShapeNet. We also train a separate DeepSDF model on the much smaller bowl category. We then create ground-truth simulation by initializing a block of fluid above the object in a $1 \times 1 \times 2m$ container.

5.5 Mountain scene

The mountain scene (Fig. 8, main paper) is obtained by training a SIREN on a triangle mesh of the Puncak Jaya mountains, Indonesia. The scene is scaled to within a side-2 cube for SIREN training. For scaling to the larger size in the paper, we apply a uniform scaling transform with a scale factor s to the SDF, i.e. $F_{sc} = sF(s^{-1}p)$. Note that SDFs can be scaled arbitarily unlike particles.

6 Experimental Details

Input features. The input to the model is the current particle positions, SDF shape name and parameters, and transformation parameters (rotation and translation). We also provide a short history of $T = 4$ previous time-steps, which is used in Sanchez-Gonzalez et al. [3]. In the 'encoder' step, we initialize the node embeddings v_i with the previous velocities and distance to the container (see [3] for details). We compute the SDF and its gradient on only the *current* particle positions, applying the inverse rigid-body transformation of the shape to the query points. The initial edge features e_{ij} contain the distance and displacement between v_i and v_j.

Model Details. We use a graph network architecture with ten message-passing steps (see Sec. 3.2, main paper for forward pass details). Each message-passing step consists of a node and edge MLP, which are shared across nodes/edges but *not* across steps. Finally, an MLP 'decoder' is applied to each node to predict accelerations. These accelerations are obtained via an inverse Euler update given the next positions $X^{(t+1)}$ [3]. Each MLP has two hidden layers with latent dimensions of 128 and ReLU activations, and a LayerNorm applied to the output. Please see [3] for more details which we leave mostly unchanged.

Loss and Training Details. We train all our models for 2M steps; we use Adam with a decaying learning rate schedule from 1e-4 to 1e-6. Noise is added to the input positions as is standard to improve stability over long rollouts [3]. For the weighting of particles near the surface (Eq. (1), main paper), we use $\lambda = 5$, and threshold $\alpha = 0.09$. This threshold is based on the classical simulator, which considers particle p_i's 'neighborhood radius' of relevant neighboring particles as $6r$, where $r = 0.015$ is the particle radius. Empirically we found that on aggregate no more than $\sim 25\%$ of the particles are considered near to the surface across a rollout.

6.1 Metrics: Additional details

Chamfer distance is computed *per-timestep* and averaged across time-steps. For each time-step we have a predicted point cloud P and ground-truth point cloud G. The Chamfer distance for each timestep is:

$$CD(P, G) = \frac{1}{|P|} \sum_{x \in P} \min_{y \in G} ||x - y||_2^2 + \frac{1}{|G|} \sum_{y \in G} \min_{x \in P} ||x - y||_2^2.$$

For the Chamfer Surface metric, at each time-step we consider $P_{\alpha'}$, the set of particles in P that have SDF value $F(p) < \alpha'$ in the predicted point cloud. We then measure the Chamfer distance of this 'near-surface' point cloud to the ground-truth point cloud G. Similarly, we compute the Chamfer distance of $G_{\alpha'}$ to all of P. The metric is:

$$CD(P, G) = \sum_{x \in P_{\alpha'}} \min_{y \in G} ||x - y||_2^2 + \sum_{y \in G_{\alpha'}} \min_{x \in P} ||x - y||_2^2$$

If P diverges significantly from G near the surface, then the average distance between a particle in $P_{\alpha'}$ and any particle in G should be high. By 'zooming in' only on near-surface particles, we can understand errors in fluid-surface interactions more effectively than overall Chamfer distance, which considers the entire rollout and averages out these errors. Note that in the Chamfer surface metric the distances are summed rather than averaged across particles.

Most of our metrics depend on access to the SDF value $f(p)$ (all except Chamfer distance). For complex scenes, neural SDFs have reliable zero-crossings but can have errors away from the surface. We convert the neural SDFs to meshes, and use an mesh → SDF library to obtain an SDF f_m which we use for computing metrics. Note that f_m is only approximately differentiable and is not scalable to large scenes (requires expensive mesh computation) compared to f_θ. Neural SDFs are more suitable for fluid prediction in large, complex scenes and also for solving inverse problems.

7 Design

7.1 Implementation Details

Please see the main paper for details on the tasks. Here we provide implementation details.

Bowl task. The reward is the log probability of a Gaussian centered at the bottom of the object, i.e. $N((0., 0., h), \Sigma)$, where h is the object height. We use a spherical covariance with $\sigma^2 = 0.8$. At each iteration the forward model is rolled out for 50 steps and the reward is measured on the final particle positions. If the filter radius r_2 reaches 0 during the optimization, it is fixed and only r_1 is optimized. We optimize for 100 iterations (likewise for the funnel task).

Funnel task. The reward is the log probability of a 2D Gaussian at the bottom of the container, centered at the origin; that is $N((0., 0.), \Sigma)$. This incentivizes the design to concentrate fluid onto the ground. We use a spherical covariance with $\sigma^2 = 0.75$. At each iteration the forward model is rolled out for 75 steps. Only particles that reaches the bottom of the container contribute to the reward. Once a particle reaches the ground it is removed from the simulation.

Latent Space Design. We use the bowl reward with $\sigma^2 = 0.6$, centered according to the unit sphere containing the chair. The DeepSDF model is a variational auto-decoder, so it learns latent codes z_i for each of the training shapes which are regularized by a Gaussian prior. A key result of [2] is that the resulting latent space is well-structured and interpolations represent valid shapes; we can thus use gradients to effectively search for a novel design (note that the bowl-shaped chair in Fig. 10, main paper certainly was not one of the training shapes, but design optimization with SURFSUP discovers it regardless).

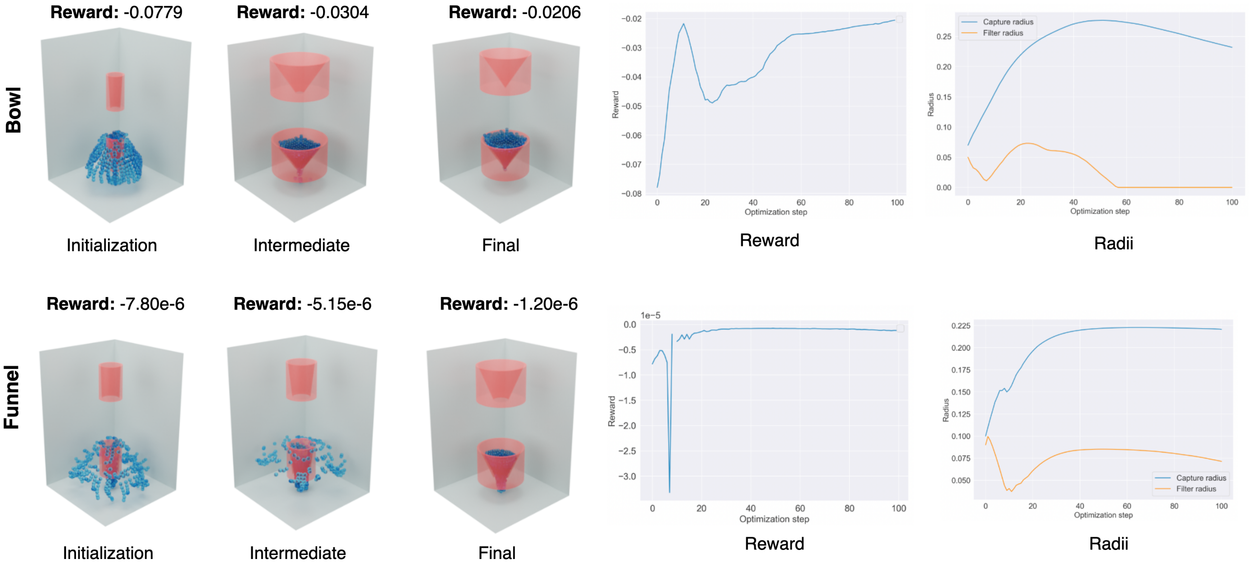
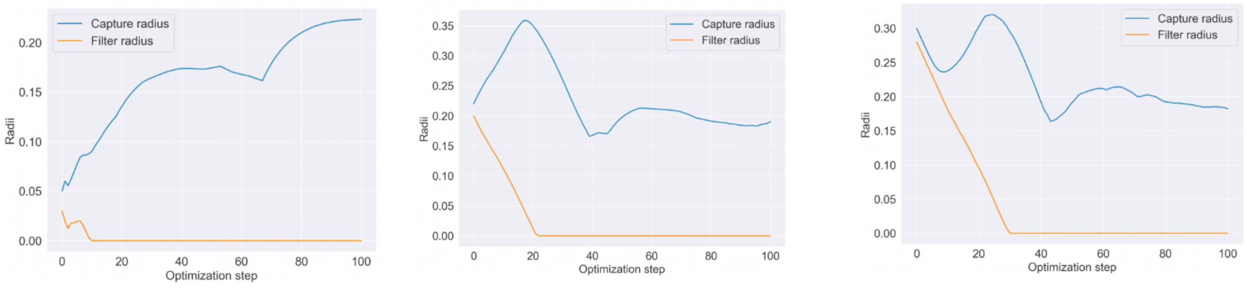


Figure 7: This shows the designs in Fig. 8 in the main paper, along with the evolution of the reward and the capture radius (blue) and filter radius (orange) over the optimization process.

7.2 Further Analysis of Design Solutions

Figure 7 shows further detail of Fig. 9 in the main paper. We additionally show the evolution of the reward and the design parameters during training. For the bowl, the reward increases throughout the optimization, while for the funnel it oscillates then converges quickly. For the bowl, we initialize $r_1 = 0.07$ (low capture radius) and $r_2 = 0.05$ (non-zero filter radius); the final solution is $r_1 = 0.232$ (high capture radius) and $r_2 = 0.0$ (perfect containing). For the funnel, we initialize $r_1 = 0.10$ (small) and $r_2 = 0.09$ (larger-than-desired); the final solution is $r_1 = 0.221$ (high capture radius) and $r_2 = 0.0714$ (narrower filter radius). For each of the six frames shown in Figure 7, we provide videos showing rollouts of exactly the length used to compute reward in our optimization (50 steps for the bowl, 75 steps for the funnel). See the `Design_videos` folder. Notably, we also find that the design optimization is robust to different initializations of r_1 and r_2 (Fig. 8).



$r_1 = 0.05, r_2 = 0.03$ $r_1 = 0.22, r_2 = 0.20$ $r_1 = 0.30, r_2 = 0.28$

Figure 8: This shows convergence of the bowl optimization for diverse initializations of the capture radius and filter radius. For all three initializations, the model converges to a sufficiently high capture radius r_1 (blue, approximately 0.20 for all initializations) and zero filter radius r_2 (orange), which is the optimal solution.

References

[1] Jan Bender and Dan Koschier. Divergence-free smoothed particle hydrodynamics. In *Proceedings of the 14th ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, SCA '15, page 147–155, New York, NY, USA, 2015. Association for Computing Machinery.

[2] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174, 2019.

[3] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning*, pages 8459–8468. PMLR, 2020.

[4] Vincent Sitzmann, Julien N.P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Proc. NeurIPS*, 2020.