Supplementary Material: Physical Simulation Layer for Accurate 3D Modeling

Mariem Mezghanni[1] Théo Bodrito Malika Boulkenafed Maks Ovsjanikov[1]

[1]LIX, Ecole Polytechnique, IP Paris
mezghanni,maks@lix.polytechnique.fr

In this document we assemble the supplementary materials that, due to the lack of space, we could not include in the main manuscript. The following sections are organized as follows:

- Section 1 gives a discussion of the societal impact of our work.
- Section 2 demonstrates the generalization power of our approach to different shape representations.
- Section 3 provides an interpretation of the simulation gradient derived in Section 3.5.2 in the main paper.
- Section 4 provides more visual results for the shape optimization task.
- Section 5 discusses our random sampling strategy.
- Section 6 describes our simulation process for stability annotation using Pybullet simulator [3] as well as the learning of baseline surrogate models based on offline simulation.
- Section 7 gives implementation details of our simulator Ψ and shows a comparison to Pybullet [3].
- Section 8 presents our experimental data statistics.
- Section 9 describes a topological regularization approach to address the limitation of our model induced by the rigid body assumption.

1. Societal Impact

In this work, we develop an approach to produce a physically realistic 3D content. We hope our work (or the principles behind it) will contribute in lowering the barrier to bringing the research efforts in 3D modeling to real-world fabrication. We also hope that our work will inspire more adjacent research fields, such as 3D object and scene understanding, to benefit from online physics as a rich and universal cue.

However, at the same time, we note that working on generative models always bears the risk of creating false and manipulating content. Besides, like all deep neural network models, our approach requires a relatively large amount of compute and hence may have a noticeable carbon footprint.

2. Generalization to other shape representations

The main focus of our work is physical plausibility and we view our contribution as *complementary* to that of recent papers that focus on improving geometric details. Our choice of DeepSDF [12] is based on its flexibility and its wide adoption as a building block in a generative pipeline. However, our approach can easily be combined with other methods. In the following, we combine our approach with different shape representations.

Point Cloud (StructureNet [11]): We plug our simulation layer SimL at the end of the point cloud shape decoder f_θ:

$$f_\theta(z) = \{x_i^j\}_{i,j} \text{ and } C_{f_\theta(z)} = \{x_i^j\}_{i,j}, \quad (1)$$

where $z \in \mathbb{R}^{256}$ is the shape latent code and $x_i^j \in \mathbb{R}^3$ is the i^{th} point of the j^{th} point cloud part.

Mesh (AtlasNet [7]): we plug our simulation layer SimL at the end of the mesh decoder f_θ generating a surface from a sphere:

$$f_\theta(z) = \{v_i\}_i \text{ and } C_{f_\theta(z)} = \{v_i\}_i, \quad (2)$$

where $z \in \mathbb{R}^{1024}$ is the shape latent code and $v_i \in \mathbb{R}^3$ is the i^{th} mesh vertex.

Spherical Primitive (DualSDF [9]): we plug our simulation layer SimL at the end of the spherical primitive decoder f_θ:

$$f_\theta(z) = \{\alpha_i = (r_i, c_i)\}_i \text{ and } C_{f_\theta(z)} = \{c_i + r_i.I_j\}_{i,j}, \quad (3)$$

where $z \in \mathbb{R}^{128}$ is the shape latent code, $r_i \in \mathbb{R}_+^*$ and $c_i \in \mathbb{R}^3$ are respectively the radius and center of primitive i, and $I_j \in \mathbb{R}^3$ is a normalized constant vector.

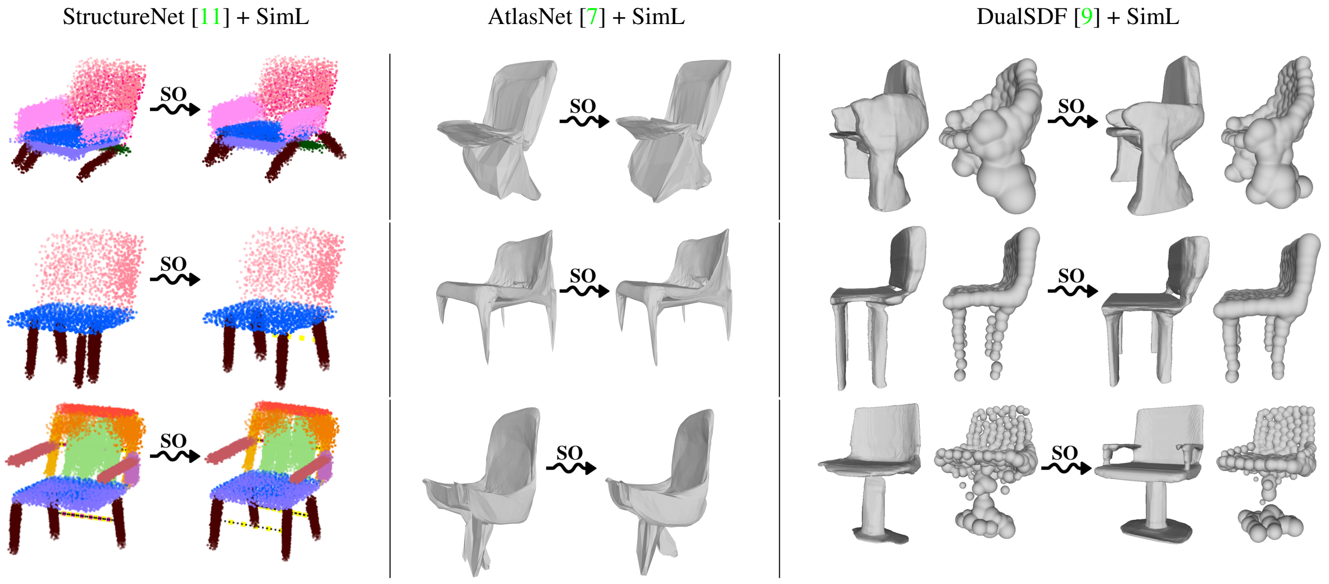


Figure 1. Qualitative results of shape optimization (SO) experiments using our simulation layer SimL. For each experiment, left and right shapes correspond to the input and optimized shapes respectively. Input shapes are randomly sampled from the learned latent spaces. First column StructureNet [11]: SO is performed on **point cloud** shapes. Second column AtlasNet [7]: SO is performed on **mesh** shapes. Third column DualSDF [9]: the left and right representations correspond to the high resolution and the spherical primitive shapes respectively. SO is performed on the **spherical primitive** shape. Our approach manages to improve the physical quality of the generated content.

Observe that in all of these cases $f_\theta(z) \mapsto C_{f_\theta(z)}$ has a closed-form expression and, hence, that $\frac{\partial x}{\partial f_\theta}$ is computed via automatic differentiation.

We conduct shape optimization experiments (cf Section 3.8 in the main paper) based on each of the mentioned decoders using pre-trained models released by the respective authors. Figure 1 illustrates qualitative results that reflect the efficiency of our approach in improving the physical quality of the generated content.

3. Simulation gradient interpretation

We recall the formulation of the Ψ gradient with respect to the decoded SDF value at $x \in \mathcal{C}_{f_\theta(G,z)}$ (cf Equations 4 and 5 in the main paper):

$$\frac{\partial \Psi}{\partial f_\theta}(x) = -\boldsymbol{g_x}.\boldsymbol{n_x}, \tag{4}$$

where $\boldsymbol{g_x} = \frac{\partial \Psi}{\partial x}$ and $\boldsymbol{n_x} = \nabla_x f_\theta$ are respectively the simulation gradient and the normal surface vector at point x. To give an intuition about the gradient value and the need for $\boldsymbol{n_x}$, we provide the example in Figure 2 illustrating the same simulation gradient $\boldsymbol{g_x}$ for opposite surface normal directions.

Given $\boldsymbol{n_x}$ direction, one can understand that in Figure 2a ($\boldsymbol{g_x}.\boldsymbol{n_x} > 0$), $\boldsymbol{g_x}$ points towards extending the shape surface, while in Figure 2b ($\boldsymbol{g_x}.\boldsymbol{n_x} < 0$), $\boldsymbol{g_x}$ points towards shrinking the shape surface. This is equivalent to saying that in Figure 2a $\boldsymbol{g_x}$ tends to decrease $f_\theta(z,x)$ $\left(\frac{\partial \Psi}{\partial f_\theta}(x) < 0\right)$, while in Figure 2b $\boldsymbol{g_x}$ tends to increase the $f_\theta(z,x)$ $\left(\frac{\partial \Psi}{\partial f_\theta}(x) > 0\right)$. This is is in line with the simulation gradient in Equation 4.

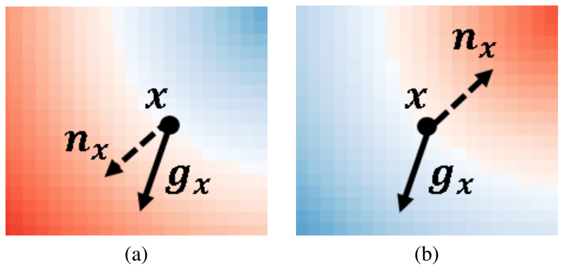


Figure 2. Example visualization of simulation gradient for opposite normal vector directions. Blue and Red colors correspond respectively to negative and positive SDF values.

Note that this observation motivates the use of SDF shape decoder in our work since it provides the normal direction: $\nabla_x f_\theta = \boldsymbol{n_x}$.

4. Shape optimization

We provide additional experimental results in Figure 3 for the task of shape optimization using Phys-DeepSDF model (cf Section 3.8 in the main paper) to illustrate the relevance of our stability loss.

The different visual examples illustrate a large variety of input stability failures. We observe that the optimization process converges to plausible solutions that a human might naturally suggest. This is due to constraining the search space to the latent space of shapes which embeds useful information about the target shape category. Besides, the comparison of the input shape to the closest training shape in terms of Chamfer Distance (for 10K randomly sampled surface points) proves that physical control leads to explore the learned latent space and to discover new and diverse shapes that go beyond training examples.

5. Random shape generation

We follow the *fixed generator* method to sample vectors from the latent space of shapes. In other words, a sampled vector z_s is the sum of a randomly selected training vector z_t to which we add a Gaussian noise vector ϵ:

$$z_s = z_t + \epsilon.$$

One can understand that $\|\epsilon\|_2$ should be big enough to explore the latent space of shapes and small enough to ensure that z_s is plausible and lies within the latent space of shapes. We control ϵ norm such that $\|\epsilon\|_2 \leq \eta$ with $\eta > 0$. Hence, η depends on $\mathcal{Z}_{train}$ vectors pairwise distances' L_2 norms:

$$\mathcal{Z}_{dists} = \{\|z_t - z_{t'}\|_2; z_t, z_{t'} \in \mathcal{Z}_{train}\}.$$

We consider η as an hyperparameter and we found empirically that setting η to the 90% quantile of $\mathcal{Z}_{dists}$ gives the best performance for the task of shape reconstruction. This observation motivates our sampling approach. Figure 4 shows a t-SNE visualization of 1K shapes' embeddings associated with training and sampled vectors for different shape categories. We observe that sampled vectors lie among the training shapes neighborhood and cover different zones of the learned latent space of shapes.

6. Stability annotation and surrogate model learning

6.1. Stability annotation using Pybullet [3]

To assess whether a given mesh shape is physically stable, we use the Pybullet simulator [3] to perform a dropping simulation that reveals whether the shape maintains its upright position when subjected to gravity and to trivial perturbations. To this end, we import the shape and the ground plane to the simulator with URDFs including the center of mass, the mass and the inertia matrix. We model the collision between the shape and the plane as perfectly inelastic, and consider the default Coulomb friction model. The dropping is then simulated from a height equal to 0.1 times shape height for $T = 3000$ simulation steps of $\Delta t = 0.008s$ time step (cf Figure 5). We set the initial horizontal orientation to random small values to penalize shapes in unstable equilibria. If the shape doesn't recover its initial upright position, then it is unstable (label=0), otherwise the shape is labeled stable (label=1).

Class	Accuracy
Chair	0.9307
Table	0.9203
Bench	0.9043

Table 1. Quantitative stability classification results.

6.2. Surrogate model h

Our paper provides a comparison of our online simulation-based approach to the standard offline simulation setting where a neural network is trained to imitate the behavior of a physical simulator while being suitable for learning [1, 10, 14].

To this end, we learn a surrogate model for physical stability prediction following the approach in [10]. Specifically, we train a voxel-based neural stability predictor for each shape category to predict the physical stability probability of an input shape. We first proceed with using the pre-trained baseline DeepSDF [12] to randomly sample N shapes following the distribution $\mathcal{Z}_{train} + \epsilon_\eta$ (cf Section 3.8 in the main paper). Each decoded shape is an evaluation of the decoder f_θ on a regular grid G of resolution $R_G = 32$. This results in N voxel shapes of signed distance field values of resolution R_G. Second, we convert each of the decoded shapes into a mesh using Marching Cubes algorithm, and annotate each mesh to be stable (1) or unstable (0) using the Pybullet simulator [3] as described in Section 6.1 above. Finally, the annotated data is used to learn a surrogate model h (binary classifier) that takes as input a voxel of resolution 32 and returns a stability probability p. In our implementation, h has a similar architecture to the voxel-based neural stability predictor in [10]. Specifically, h consists of 3 convolutional layers followed by two fully connected layers. All layers are followed by the ReLU activation except for the last layer where we use a Sigmoid activation to output a probability value. Moreover, we empirically found that applying a sigmoid function to the input SDF values improves the classification performance of h to a significant extent.

For each category we collect $N = 20K$ shapes equally distributed between stable and unstable shapes, and work with train/validation/test splits of a 80%-5%-15%. Table 1 reports the accuracy on the test set for each of the trained surrogate models employed in the current work experiments.

7. Implementation details about our simulator Ψ & Comparison to Pybullet simulator [3]

For Ψ simulation setting, we set time step $\Delta t = 0.01$ to accurately estimate the shape trajectory, and set simulation steps $T = 200$ so that the simulation process reaches

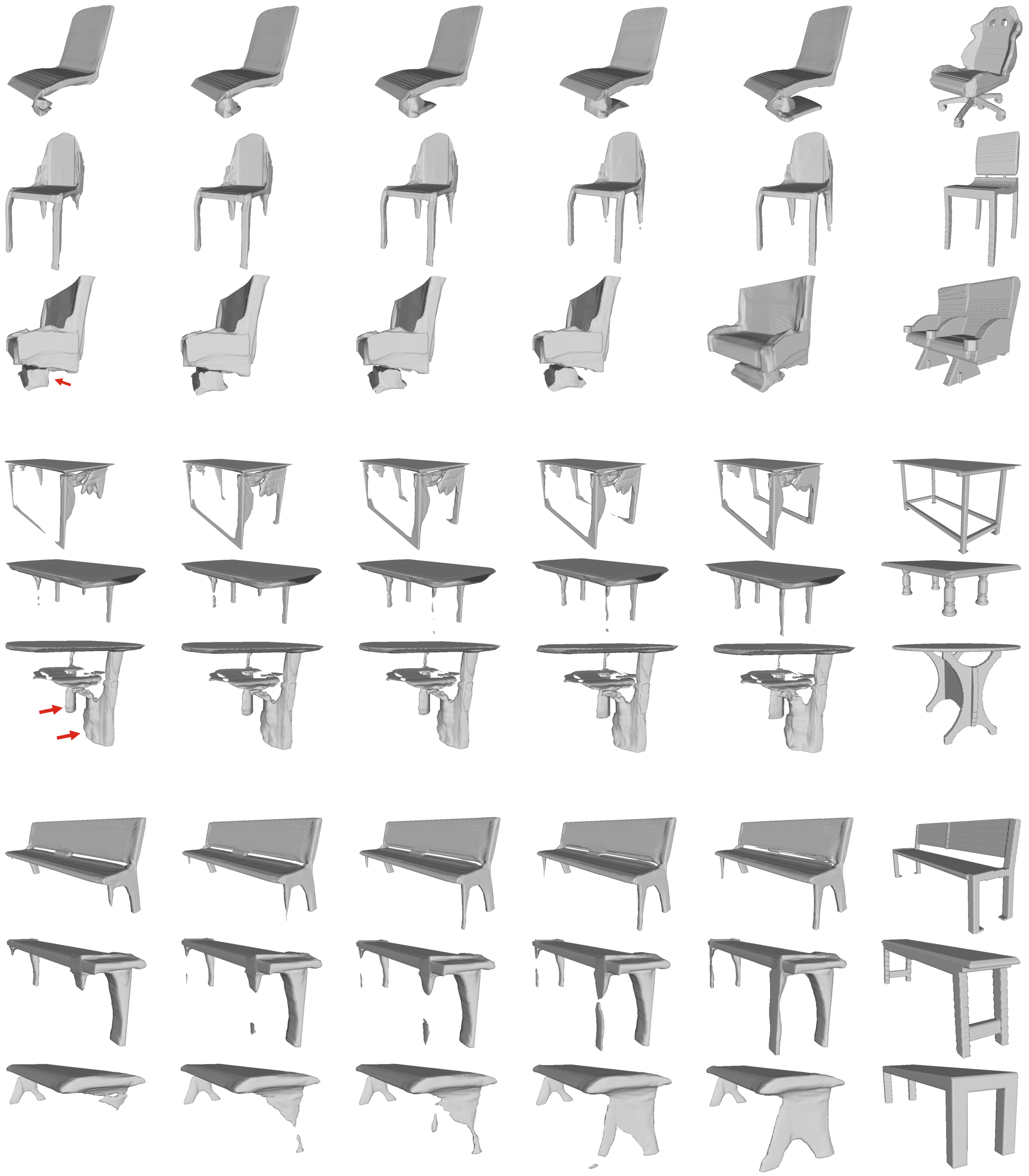
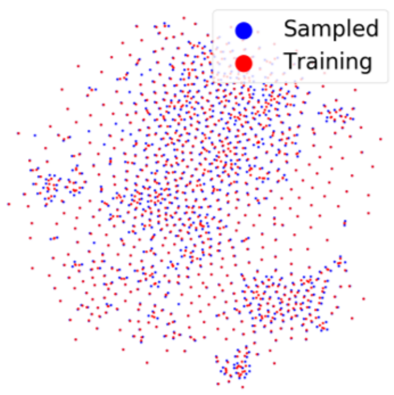
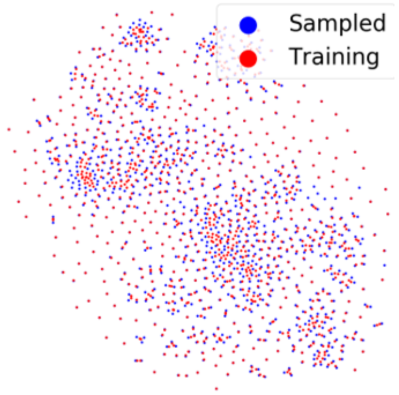


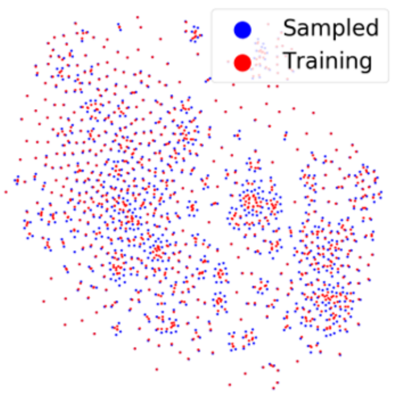
Figure 3. Visual results of shape optimization using Phys-DeepSDF model. From left to right: the initial shape, 3 intermediate shapes, the final shape and the closest training shape to the final shape in terms of Chamfer Distance (for 10K randomly sampled surface points). All shapes are obtained by randomly sampling from training vectors neighborhood using Phys-DeepSDF. Our optimization approach manages to correct stability failures while delivering visually plausible results and discovering novel geometries that go beyond the training samples.



(a) Chair



(b) Table



(c) Bench

Figure 4. We show a t-SNE [15] embedding of the training and the sampled latent vectors from the pre-trained DeepSDF [12] latent space. The sampling process follows the $\mathcal{Z}_{train} + \epsilon_\eta$ distribution explained in Section 3.8 from the main paper.

its end. Note that decreasing Δt would increase the accuracy of the trajectory estimation but would also require higher T and hence increase the simulation time (similarly to Pybullet [3] setting interpretation). Besides, we assume frictionless collision with the ground and consider uniform mass density by defining the shape mass m as the sum of masses of particles $\{x_p\}$, where $\{x_p\}$ are the G grid points such that $f_\theta(x_p, .) \leq 0$ (cf Section 3.5.1 in the main paper): $m = \sum_{x_p} m_p$ with constant m_p. However, one can also define $m_p(x_p)$ to account for the material at volume defined by x_p when considering shapes with non uniform density. The shape is simulated from a height equal to 0.1 times shape height, an initial velocity vector of $(0, 0, -1)$, a null initial torque and an initial rotation r_0 along the horizontal axes of small random values in [-5°, 5°].

To illustrate the performance of our differentiable simulator Ψ, we provide a visual comparison to a reference physical engine Pybullet [3] for the example of dropping simulation under the gravity force. The Pybullet [3] simulation setting is described in Section 6.1 above.

Figure 5 shows the behavior across the simulation process for stable and unstable shapes. Pybullet [3] requires as input a mesh file that we obtain by decoding shapes at a resolution of 256 and using Marching Cubes to recover the surface. Whereas Ψ takes as input surface points computed following Equation (3) in the main paper. We observe that Ψ, as well as Pybullet [3], consistently detect instabilities, and also predict overall shape trajectories. Crucially, our differentiable simulator Ψ records the gradient over simulation steps, which is not possible using Pybullet directly.

Timing Furthermore, we note that our differentiable simulator Ψ takes 0.537 seconds for one simulation applied on $\mathcal{C}$ sampled from a grid G of resolution 32 (cf Equation(3) in our manuscript) on Intel Xeon 5220 Gold CPU. As for Pybullet, the simulation process takes 0.05 seconds *without producing gradient*, to which must be added the time for mesh and URDF file reconstruction, that are not required when using Ψ.

Class	Train		Val		Test	
	def	exp	def	exp	def	exp
Chair	4744	4718	678	673	1356	1350
Table	5956	5908	851	847	1702	1692
Bench	1271	1095	181	152	364	308

Table 2. Experimental data statistics. We report the size of the default (def) and the experimental (exp) datasets after the stability filtering step.

8. Experimental datasets

We use 3D models from ShapeNet Core dataset (v1) [2]. We consider the Chair, Bench and Table categories and use the default train/validation/test splits provided in [8]. Furthermore, we filter unstable shapes for each category to ensure that SR=100% for learning data. Table 2 reveals data size for each category before and after stability filtering. Figure 6 illustrates examples of filtered unstable shapes.

9. Limitation of the rigid shape assumption and topological regularization

Although the rigid-body assumption has proved beneficial to build a scalable and time efficient physical simulator and to improve the validity of the generated content, instabilities caused by disconnected parts are discarded. In fact, our simulator assumes that the distance between each two shape points is constant. We show how this limitation can be addressed using a topological regularization as a post-processing step to enforce shape connectivity.

We follow the approach described in [10] that proposes a topological loss operating on implicit 3D shape representation values $\{f(x); x \in G\}$ predicted by a generative network f and estimated on a voxel grid G; where the shape

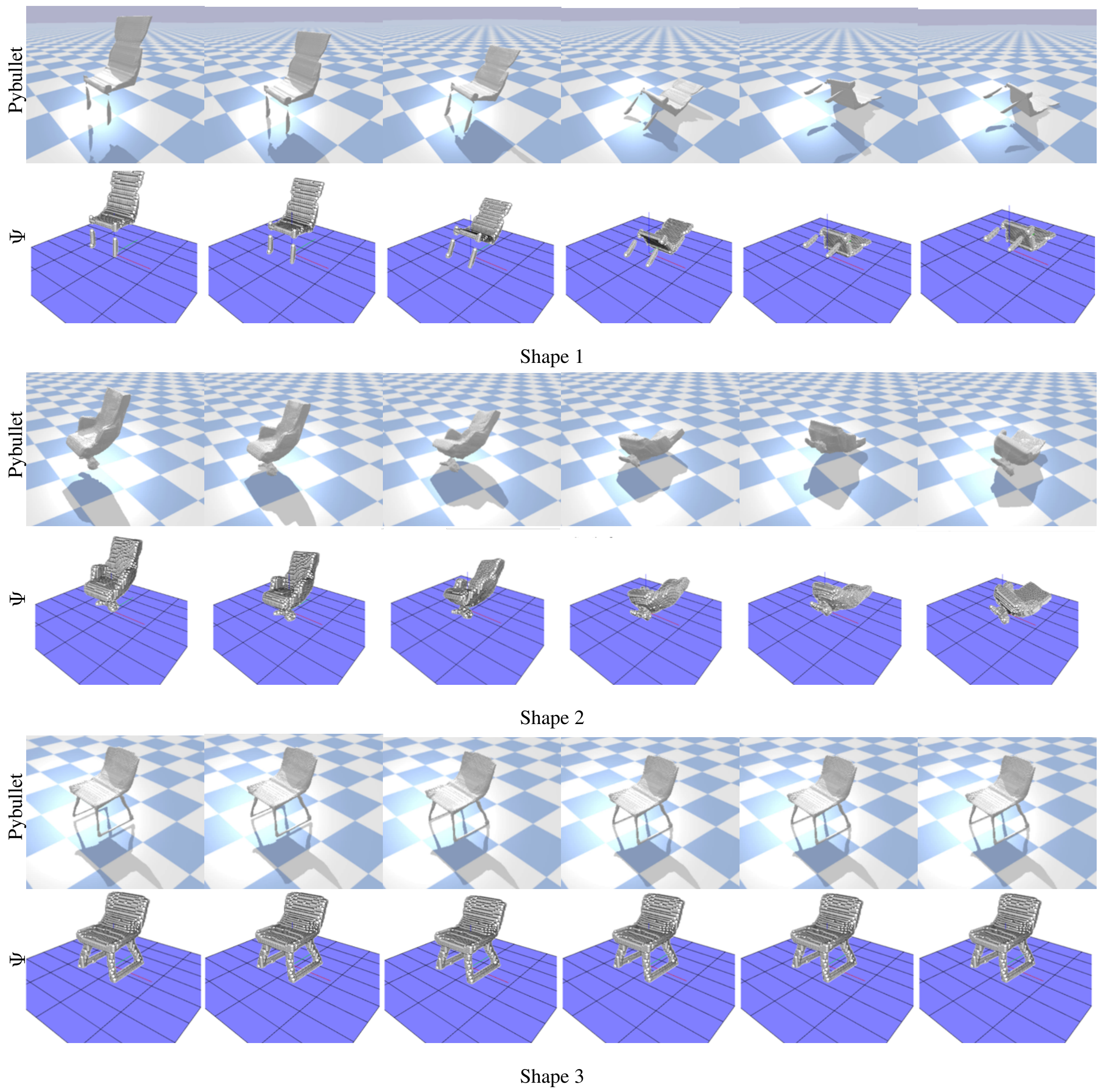


Figure 5. Comparison between results obtained using our differentiable rigid body simulator Ψ and simulations obtained using Pybullet [3] simulator. From left to right: initial state, 4 intermediate states, and the final state. The visual comparison demonstrates sufficient performance of our simulator Ψ for both: trajectory estimation and instability detection.

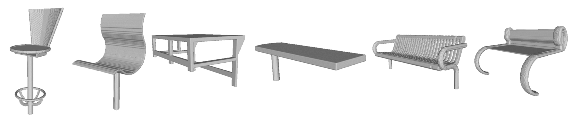


Figure 6. Visual examples of filtered unstable shapes. We show two examples of each category of (from left to right) Chair, Table and Bench.

surface corresponds to the λ-isosurface for $\lambda \in \mathbb{R}$. The idea is to explore tools from persistent homology [4–6, 13, 16] to build a loss that enforces f to have a single minimum (or maximum); or equivalently to produce a geometry with a single component. We briefly describe the loss formulation in our context and refer the interested reader to [10] for a more comprehensive overview. Here, f is $-f_\theta(.,z)$. We proceed with building the persistent diagram $P^{\lambda}_{f_\theta(.,z)} =$

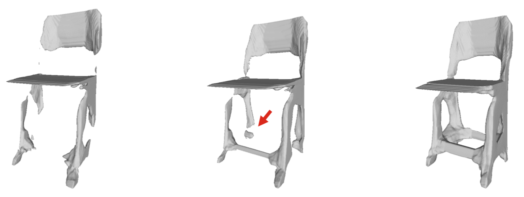


Figure 7. Visual examples of shape optimization experiments using our model Phys-DeepSDF. **(a)** The input unstable shape **(b)** The optimization result of Shape (a) using $\mathcal{L}_s$ **(c)** The optimization result of Shape (b) using $\mathcal{L}_c$. Stability loss $\mathcal{L}_s$ manages to recover missing parts and to stabilize shape under rigid body assumption, while topological regularization $\mathcal{L}_c$ addresses possible disconnections.

$(b_i^\lambda, d_i^\lambda)_{1\leq i\leq m_\lambda}$ with $m_\lambda \in \mathbb{N}^*$ that records the birth b_i^λ and death d_i^λ values of connected components such that the iso-surface value λ (here equals 0) lies in the birth-death interval. Intuitively, enforcing the shape associated with $f_\theta(.,z)$ to have a single connected component boils down to constraining $P^\lambda_{f_\theta(.,z)}$ to have a single element. Assuming that $b_j^\lambda - d_j^\lambda \geq b_i^\lambda - d_i^\lambda$ for $i > j$, the connectivity loss can be expressed as follows:

$$\mathcal{L}_c = \sum_{2\leq i\leq m_\lambda} b_i^\lambda - d_i^\lambda, \tag{5}$$

which is differentiable as demonstrated in [6, 10, 13].

We propose to relax the rigid body assumption by applying the topological regularization as post-processing step. Specifically, a produced disconnected shape undergoes an additional shape optimization experiment as described in Section 3.8 from the main paper where we replace our stability loss $\mathcal{L}_s$ by $\mathcal{L}_c$.

Figure 7 illustrates an example of shape optimization experiment using our Phys-DeepSDF model. The optimization of input shape in Figure 7a using our simulation-based physical loss $\mathcal{L}_s$ leads to shape in Figure 7b that is deemed stable under rigid body assumption only. By further optimizing the resulting shape in Figure 7b using $\mathcal{L}_c$ we efficiently address the disconnection limitation as shown in Figure 7c.

References

[1] P. Baqué, E. Remelli, F. Fleuret, and P. Fua. Geodesic convolutional shape optimization. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 472–481, 2018. 3

[2] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015. 5

[3] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2019. 1, 3, 5, 6

[4] H. Edelsbrunner and J. Harer. *Computational Topology: An Introduction*. Applied Mathematics. American Mathematical Society, 2010. 6

[5] Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. volume 28, pages 454 – 463, 02 2000. 6

[6] Rickard Brüel Gabrielsson, Bradley J. Nelson, Anjan Dwaraknath, Primoz Skraba, Leonidas J. Guibas, and Gunnar E. Carlsson. A topology layer for machine learning. *CoRR*, abs/1905.12200, 2019. 6, 7

[7] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan Russell, and Mathieu Aubry. AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2

[8] Christian Hane, Shubham Tulsiani, and Jitendra Malik. Hierarchical surface prediction for 3d object reconstruction. *3DV*, 2017. 5

[9] Zekun Hao, Hadar Averbuch-Elor, Noah Snavely, and Serge Belongie. Dualsdf: Semantic shape manipulation using a two-level representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020. 1, 2

[10] Mariem Mezghanni, Malika Boulkenafed, Andre Lieutier, and Maks Ovsjanikov. Physically-aware generative network for 3d shape modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9330–9341, June 2021. 3, 5, 6, 7

[11] Kaichun Mo, Paul Guerrero, Li Yi, Hao Su, Peter Wonka, Niloy Mitra, and Leonidas Guibas. Structurenet: Hierarchical graph networks for 3d shape generation. *ACM Transactions on Graphics (TOG), Siggraph Asia 2019*, 38(6):Article 242, 2019. 1, 2

[12] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 1, 3, 5

[13] Adrien Poulenard, Primoz Skraba, and Maks Ovsjanikov. Topological Function Optimization for Continuous Shape Matching. *Computer Graphics Forum*, 37(5):13–25, 2018. 6, 7

[14] Edoardo Remelli, Artem Lukoianov, Stephan Richter, Benoit Guillard, Timur Bagautdinov, Pierre Baque, and Pascal Fua. Meshsdf: Differentiable iso-surface extraction. In *Advances in Neural Information Processing Systems*, volume 33, 2020. 3

[15] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008. 5

[16] Afra Zomorodian and Gunnar Carlsson. Computing persistent homology. *Discrete and Computational Geometry*, 33:249–274, 02 2005. 6