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

Abstract

We introduce a novel approach for generative 3D modeling that explicitly encourages the physical and thus functional consistency of the generated shapes. To this end, we advocate the use of online physical simulation as part of learning a generative model. Unlike previous related methods, our approach is trained end-to-end with a fully differentiable physical simulator in the training loop. We accomplish this by leveraging recent advances in differentiable programming, and introducing a fully differentiable point-based physical simulation layer, which accurately evaluates the shape's stability when subjected to gravity. We then incorporate this layer in a signed distance function (SDF) shape decoder. By augmenting a conventional SDF decoder with our simulation layer, we demonstrate through extensive experiments that online physical simulation improves the accuracy, visual plausibility and physical validity of the resulting shapes, while requiring no additional data or annotation effort.

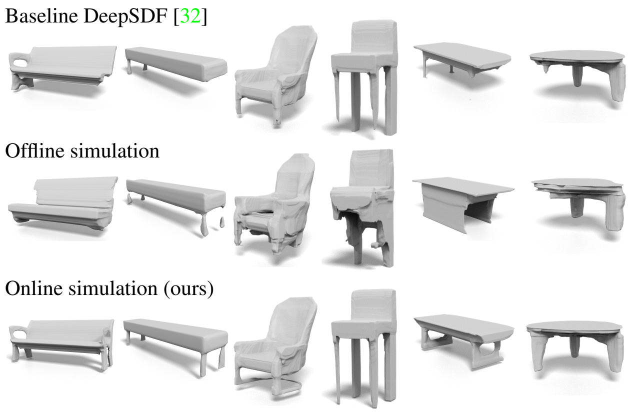

Figure 1. Qualitative comparison of online simulation with the SotA offline simulation [30] for the task of shape optimization. From top to bottom: physically invalid shapes sampled from baseline DeepSDF [32], results using [30] and our results. The optimized shapes reflect the accuracy and efficiency of online simulation compared to the offline setting in terms of physical quality and geometric consistency.

1. Introduction

Over the past several years, there has been a steady stream of works aimed at developing deep neural networks for 3D shape generation. A key challenge is to accommodate plausible and diverse content while preserving geometric and structural validity [1, 27, 42]. Though remarkable progress has been made in this direction, state-of-the-art approaches focus primarily on geometric or visual plausibility, while discarding a key purpose of 3D design: functionality [23]. Indeed, a designed 3D shape is often meant to serve a particular function in the real world. For instance, a chair is expected to be stable when subjected to gravity. Ignoring this crucial constraint leads the generated content to suffer from severe functional artifacts such as lack of connectivity and physical instability [30], which severely hinders its utility in real-world downstream tasks.

One way to address this challenge is by leveraging physical simulation to guide generative models to produce functionally valid shapes. Indeed, physical simulation is a common mechanism for verifying whether a candidate 3D shape fulfills certain functionality [4, 22, 41]. Although physical simulation is a powerful tool that is applicable without additional data annotation, incorporating it into generative modeling can be challenging, since it is typically non-differentiable and can be both complex and costly to be used at training time. Thus, existing approaches that attempt to combine generative networks with physical simulation are typically limited to *offline simulation*, which requires to either iteratively filter and update the training data [36] or to train a surrogate model that imitates the simulator while being suitable for learning [30]. While these works sidestep the aforementioned challenges, they achieve this by sacrificing the major benefits of end-to-end training. In particular, using offline simulation and surrogate models can hurt the generalization power of the trained networks and introduce data biases, implicitly promoting the seen valid geometric patterns instead of tackling generic physical failures.

At the same time, recent advances in building differentiable physical simulators [11,24,25] have opened the possibility of learning neural networks with online physical simulation. So far, these approaches have not been leveraged in 3D generative networks, in part due to the difficulty of integrating physical simulation, while ensuring efficiency and geometric accuracy in a single coherent framework.

In this paper, we make a first step towards this goal. Specifically, we show how to endow a deep generative network based on the DeepSDF decoder [32] with online simulation-based physical supervision. We focus on the generative modeling of man-made shapes that should naturally be stable when subjected to gravity, and we set *physical stability* as our target functional constraint.

To achieve this, we first design a simulation layer (SimL) based on a differentiable rigid body simulator that we implement using the recent efficient DiffTaichi framework [24]. The simulator computes rigid body dynamics of 3D shapes subjected to gravity in the presence of the ground plane. This allows to both accurately evaluate the physical behavior of the shapes in the forward stage, and to back-propagate simulation gradients for physical supervision during the backward stage. Then, we integrate SimL into the implicit shape decoder DeepSDF [32] and show how SDF-based generative modeling can be combined with physical simulation in a single coherent end-to-end trainable manner to create plausible and physically valid shapes.

Combining these contributions, we introduce a new model **Phys-DeepSDF**, which is the first end-to-end deep generative model endowed with *online physical simulation*. We illustrate the utility of our model on a range of challenging cases and demonstrate that it has significantly higher generalization power compared to methods trained with offline simulation, enabling several applications such as accurate shape generation and optimization.

Contributions. Our main contributions are three-fold:

(1) We build a point based simulation layer SimL based on a differentiable rigid-body simulator that can be incorporated into a 3D shape decoder.

(2) We show how our layer can be incorporated in a SDF-based generative model with a novel stability loss and accurate gradient back-propagation.

(3) We demonstrate the efficiency of 3D modeling with online physical simulation, by improving the generalization power and quality of the produced content.

2. Related Work

2.1. 3D shape generation of implicit fields

There have been efforts in building generative networks of 3D shapes based on a variety of representations such as voxels [16,18,37,42], octrees [35,39], point clouds [1,21], surface meshes [15,17,38,40], multi-view depth maps [2] and part-based composition [45]. Recently, learning implicit fields for generative shape modeling in the form of binary occupancies [8,28] and signed distances [13,26,32] has been widely studied. Such implicit representations proved to be computationally and memory efficient while allowing high resolution geometry decoding. In our paper, we focus on signed distance representation [32] since it yields additional information compared to binary occupancy, consisting of the distance to the shape surface as well as surface normals. In fact, previous work [34] demonstrated that 3D surface samples can be differentiated with respect to the underlying deep signed distance field. This enables us to combine deep generative networks of implicit fields with simulation based control that requires explicit representations. Differently from [34] that focused on latent code optimization using offline simulation, we build a novel differentiable *online simulation layer*, which ensures efficient, accurate and class-agnostic physical supervision. Finally, we show how online simulation can be integrated into end-to-end training of generative models.

2.2. Physics Simulation in Deep Learning

There has been increasing interest in improving object and scene understanding by exploiting physics supervision [46]. Particularly, leveraging physics simulators for neural network learning was envisaged for various topics including scene reconstruction [12], object understanding [29], contact points and physical forces inference [14] etc. Classical physics simulators such as PyBullet [10] are often non differentiable and it is not straightforward to use them in the context of neural networks learning. Consequently, employing gradient approximations [12,14,29] or replacing the simulator by a differentiable surrogate model that imitates the simulator output [3,34] prove necessary. Recently, there have been some works in building differentiable physics simulators [11, 24, 25] that provide gradients of simulation to overcome these limitations and pave the way for future work in this field.

The only prior deep generative approaches of 3D shapes informed with physics simulation were the works in [30, 36]. However, these methods are limited to offline simulation and require either training data update [36] or pre-training a surrogate network [30] to inform the learning network with the simulation feedback. In our paper, we advocate the use of online simulation gradient to learn the generative network by building a differentiable simulator. This makes our work advantageous in two key aspects: the physics supervision is class agnostic and does not require any pre-training, and the generative network can be efficiently optimized via end-to-end training.

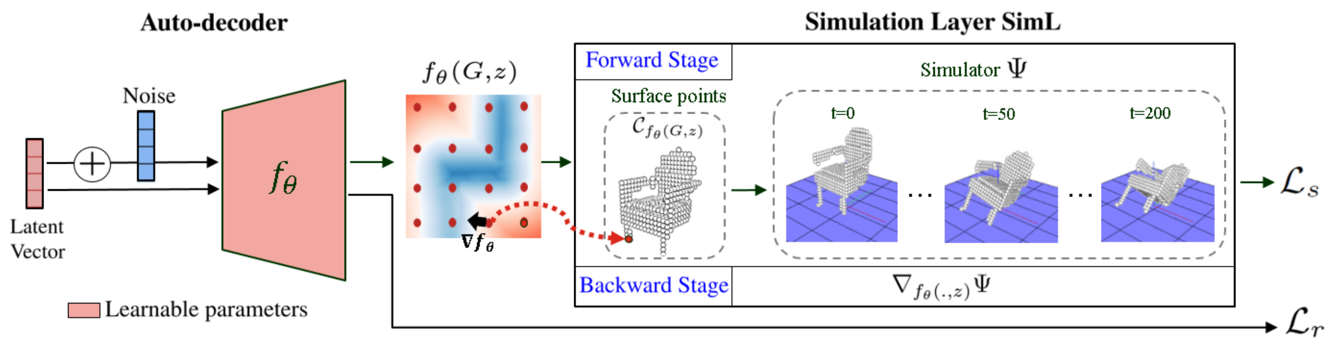

Figure 2. **PhysDeepSDF** overview. Our generative model consists of two modules a) shape auto-decoder that we design similarly to DeepSDF [32] network, and which maps a latent code to the SDF-based shape; and b) our custom simulation layer SimL that in the forward stage (Section 3.5.1) extracts the points from the shape's surface, (cf Eq. (3)) and simulates its trajectory when subjected to gravity (Section 3.4), and in the backward stage (Section 3.5.2) computes the simulation gradient with respect to the SDF parameters.

3. Proposed approach

3.1. Overview and Motivation

Our main goal is to combine the benefits from two major frameworks: implicit-based 3D shape modeling, as a powerful learning-based generative model, and physical simulation to guarantee the validity of the generated content beyond visual plausibility. Our architecture is depicted in Figure 2. It consists of an auto-decoder that learns the latent space of shapes, and a simulation layer SimL that ensures the physical stability of the generated content by informing the decoder with the supervisory physical signal. In this work we focus on the physical stability of shapes when subjected to gravity, with the ultimate goal that the generated content should be able to maintain a stable equilibrium on a ground plane. To this end, our simulation layer SimL performs a differentiable simulation of a given 3D shape, when dropped in the upright position from a small height above the ground and subjected to gravity. We then evaluate the shape orientation at the last simulation step to assess the shape's physical stability: it is stable if it maintains its upright position, and unstable otherwise. Figure 1 provides examples of stable and unstable shapes.

We focus on physical stability among other possible functional constraints since it is not only a common cause of functional failures [30], but also because it represents a shared functional requirement across different shape categories (chair, table, bench, etc). Moreover, physical stability has proved beneficial for boosting many computer vision tasks [12, 33, 43, 44] which broadens the potential scope of our approach beyond generative modeling.

The key difference of our approach compared to existing physically-aware generative models [30, 36], is that we use *online simulation* during the learning procedure, as part of the neural network. As we demonstrate below, this has several advantages. First, the physical module has no learnable parameters and is thus not tied to the training data, in contrast to using surrogate model pretraining [30]. Second, the online simulation gradient is accurate and allows to precisely and explicitly address physical failures without sacrificing the geometric diversity. This is in contrast to data-driven methods based on surrogate model pre-training [30] or training data filtering [36] that naturally inherit data biases and are likely to push towards the known valid geometries. As a result, the efficiency and accuracy of online simulation enables the training of a shape decoder without affecting its expressive power, whereas previous methods are limited to learning a mapping network [30] or updating training data [36].

Given these goals, two key challenges emerge: (1) building a scalable and time efficient physical simulator, and (2) combining explicit (point-based) physical simulation with an implicit function-based decoder. To address these, we first build an efficient differentiable sparse point cloud simulator, and then show how it can be incorporated in a generative model based on the signed distance function (SDF) representation [32], while enabling accurate gradient back-propagation between explicit and implicit representations.

The rest of this section is organized as follows: we start by introducing the SDF and the decoder architecture in Sections 3.2 and 3.3 respectively. Then, Sections 3.4 and 3.5 describe our differentiable physical simulator and our simulation layer SimL respectively. After that, we propose a novel physical stability objective function in Section 3.6, and discuss the generalization power of our approach in Section 3.7. Finally, we describe our training scheme along with target applications in Section 3.8.

3.2. Signed Distance Function SDF.

In this work, we represent a closed shape S with SDF $f_S : \mathbb{R}^3 \rightarrow \mathbb{R}$ that attributes to each point $x \in \mathbb{R}^3$ its signed distance $f_S(x)$ to the closest surface point. $f_S(x)$ is positive if x is outside the shape and negative if it is inside. Compared to binary inside/outside implicit function representation [9], SDF yields additional information consisting of the *distance* to the surface $f_S(x)$ and also the direction for projecting x onto the surface $-\nabla_x f_S(x)$. This additional information is not only useful to generate a high quality surface, but it is also important to ensure the differentiability of the explicit shape representation, required for physical simulation, with respect to the implicit representation decoded by the network, as explained in Section 3.5.

3.3. Decoder architecture.

We design our shape decoder f_θ similarly to DeepSDF [32] where θ denotes the learnable parameters. We give a brief overview of f_θ and refer the reader to [32] for more details. Given a set of N training shapes $\{S_i\}$ paired with a set of points $\{X_i = \{x_i^j\}\}$ sampled around each shape, f_θ attempts to approximate SDF values $f_\theta(x_i^j, z_i) = f_{S_i}(x_j) = s_i^j$ for all $x_i^j \in X_i$ where $z_i \in R^d$ is a learnable latent representation of S_i. The loss function is defined by the L_1-norm between the predicted and ground truth SDF values:

$$L(f_\theta(x_j, z_i), s_i^j) = |clamp_\delta(f_\theta(x_i^j, z_i)) - clamp_\delta(s_i^j)|, \quad (1)$$

where $clamp_\delta(x) = min(\delta, max(-\delta, x))$ clamps SDF values with $\delta = 0.1$.

At training time, $\mathcal{Z}_{train} = \{z_i\}$ are randomly initialized from $\mathcal{N}(0, 0.01^2)$ and are optimized along with parameters θ via the following reconstruction loss:

$$\mathcal{L}_r(z_i, \theta; X_i) = \sum_j L(f_\theta(x_i^j, z_i), s_i^j) + \frac{1}{\sigma^2}\|z_i\|_2^2, \quad (2)$$

where $\sigma = 10^{-2}$ is a regularization parameter. Note that, although we fix the decoder architecture for f_θ, our method is generalizable and can use any shape decoder based on signed distance functions.

3.4. Physical simulator Ψ.

We aim to build a differentiable physical simulator that can record the gradient over the simulation steps, and replay them in a reversed order during the backward pass. To this end, we use the DiffTaichi [24] framework tailored for high-performance differentiable physical programming. For instance, a differentiable elastic object simulator written in DiffTaichi has been shown to be 188× faster than its counterpart TensorFlow implementation [24].

To assess the physical stability of a generated shape by f_θ, we implement a differentiable impulse-based rigid body simulator [5] detailed in Algorithm 1. Concisely, our simulator, denoted by Ψ, takes a shape represented by its surface points $\mathcal{C}$ along with the mass, center of mass and inertia matrix, and outputs its dropping trajectory from a height h simulated for T time steps: $\Psi(\mathcal{C}) = \{(p_t, r_t); t \in [1, T]\}$, where p_t and r_t are, respectively, the position of the center of mass and orientation at time step t. This is made

Algorithm 1 Physical simulator Ψ

```
    Variables  Point cloud shape $\mathcal{C}$, Gravitational force $F_g = m\mathbf{g}$, Position $p_0$, Velocity $v_0$, Inertia tensor $I_0$, Rotation $r_0$, Quaternion $q_0$, Angular velocity $w_0$, Angular momentum $L_0$
    end Variables
    for $t \to 1$ to $T$ do
        $\mathcal{C}_t^{col} \leftarrow \{p \in \mathcal{C}; p$ reaches the ground $\}$
        procedure Collision Handling
            for $p \in \mathcal{C}_t^{col}$ do
                Compute impulse $J_p$
            end for
            $J_t \leftarrow \frac{\sum_{p \in \mathcal{C}_t^{col}} J_p}{max(1, |\mathcal{C}_t^{col}|)}$
            $\tau_t^J \leftarrow \frac{\sum_{p \in \mathcal{C}_t^{col}} (p - x_t) \times J_p}{max(1, |\mathcal{C}_t^{col}|)}$
        end procedure
        procedure Linear Dynamics
            $v_t \leftarrow v_{t-1} + \frac{1}{m}(\Delta t\, F_g + J_t)$
            $p_t \leftarrow p_{t-1} + \Delta t\, v_t$
        end procedure
        procedure Rotational Dynamics
            $L_t \leftarrow L_{t-1} + \Delta t\, \tau_t^{F_g} + \tau_t^J$
            $I_t^{-1} \leftarrow r_{t-1}\, I_0^{-1}\, r_{t-1}^\intercal$
            $\omega_t \leftarrow I_t^{-1} L_t$
            $\hat{q}_t \leftarrow q_{t-1} + \frac{\Delta t}{2}(q_{t-1}.\omega_t)_{quat}$
            $q_t \leftarrow \frac{\hat{q}_t}{\|\hat{q}_t\|_2}$
            $r_t \leftarrow$ convert $q_t$ to rotation matrix
        end procedure
    end for
    return $\{p_t, r_t\}_{1 \le t \le T}$
```

computationally tractable thanks to our parallel implementation of the procedure Collision Handling in Algorithm 1 to compute physical impulses J_p following equation (8-18) in [5]. Here, $(.)_{quat}$ denotes quaternion product as in [5]. Note that we introduce small perturbations into the simulation process to penalize shapes in unstable equilibria by setting r_0 along the horizontal axes to random small values.

We highlight that Ψ contains no learnable parameters and that, by definition, Ψ is differentiable and $\nabla_{\mathcal{C}}\Psi$ is well-defined. We set the dropping height h equal to 0.1 times shape height and consider uniform and equal volumetric mass density for all shapes. The other hyperparameters including time steps T, simulation step Δt and initial simulation settings are determined empirically to optimize simulation accuracy and time efficiency. We provide more details and a comparison to PyBullet [10] in the supplementary.

Note that although we focus on physical stability, which applies to many shape categories, our simulator can also be employed to promote other functional constraints as long as they operate on the simulated shape trajectory.

3.5. Simulation Layer SimL.

We implement a custom PyTorch layer SimL that takes as input SDF point-value pairs associated with a 3D shape and computes its dropping simulation trajectory using Ψ. Ultimately this allows SimL to be plugged on top of f_θ to assess the physical quality of the generated shapes.

3.5.1 Forward Stage:

SimL takes as input a decoded SDF-based shape for a given latent vector z evaluated on a regular grid G of resolution R_G: $\{f_\theta(x, z); x \in G\}$, and outputs its physical behavior using Ψ. For this, we start with computing the coarse-grained surface $\mathcal{C}$ required for the simulation module as follows:

$$\mathcal{C}_{f_\theta(G,z)} = \{x - f_\theta(x, z).\nabla_x f_\theta(x, z); x \in G, |f_\theta(x, z)| < \delta\}\,. \tag{3}$$

$\mathcal{C}_{f_\theta(G,z)}$ consists of a set of points x that are sufficiently close to the shape surface ($\delta = 4/R_G$), and that are projected onto the surface using the signed distance value $f_\theta(x, z)$ and the surface normal vector $\nabla_x f_\theta(x, z)$. $\mathcal{C}_{f_\theta(G,z)}$ is then fed to our simulator Ψ to compute its dropping trajectory $\Psi(\mathcal{C}_{f_\theta(G,z)}) = \{(p_t, r_t); t \in [1, T]\}$.

3.5.2 Backward Stage:

To ensure gradient back-propagation up to the shape decoder weights, we need to compute the following gradient:

$$\nabla_{f_\theta(.,z)}\Psi = \left(\frac{\partial \Psi}{\partial f_\theta}(x)\right)_{x\in G}. \tag{4}$$

We define $\nabla_{f_\theta(.,z)}\Psi$ as follows:

$$\nabla_{f_\theta(.,z)}\Psi = \begin{cases} \frac{\partial \Psi}{\partial x}\frac{\partial x}{\partial f_\theta} & \text{if } x \in \mathcal{C}_{f_\theta(G,z)} \\ 0 & \text{if } x \in G \setminus \mathcal{C}_{f_\theta(G,z)}. \end{cases}$$

Note that $\frac{\partial \Psi}{\partial x}$ for $x \in \mathcal{C}_{f_\theta(G,z)}$ is well defined since Ψ is differentiable by definition. It remains to compute $\frac{\partial x}{\partial f_\theta}$. Following Theorem 1 from [34], one can prove that

$$\frac{\partial x}{\partial f_\theta} = -\mathbf{n}(x) = -\nabla_x f_\theta, \tag{5}$$

where $\mathbf{n}$ denotes the surface normal. These two observations together provide the simulation gradient with respect to the decoded SDF-based shape.

To give an intuition about the physical gradient, for each $x \in \mathcal{C}_{f_\theta(G,z)}$, the SDF values gradient direction $-\frac{\partial \Psi}{\partial x}\nabla_x f_\theta$ is pushing $f_\theta(x, z)$ towards negative values if the inner product of the simulation gradient and normal vector is positive (tends to expand the shape), and towards positive values (tends to shrink the shape) otherwise. We visually illustrate this in our supplementary.

To summarize, our simulation layer is built as follows:

$$\begin{array}{lcccl} \text{SimL Forward} & : & f_\theta(., z) & \rightarrow & \Psi(\mathcal{C}_{f_\theta(.,z)}) \\ \text{SimL Backward} & : & \frac{\partial \mathcal{L}_s}{\partial \Psi} & \rightarrow & \frac{\partial \mathcal{L}_s}{\partial \Psi}.\nabla_{f_\theta(.,z)}\Psi, \end{array}$$

where $\mathcal{L}_s$ is the stability loss defined as a function of $\Psi(\mathcal{C}_{f_\theta(.,z)})$ in the section below, and $\frac{\partial \mathcal{L}_s}{\partial \Psi}$ is the upstream loss gradient.

3.6. Stability loss.

We are now ready to formulate our stability loss, which penalizes the physical instability of each decoded shape $f_\theta(., z)$ for a given latent vector z. Provided that all shapes are aligned in the upright orientation and assuming rigid-body motion (distances between points within the shape remain constant), enforcing physical stability boils down to having a null horizontal orientation at $t = T$. Observe that the 3^{rd} column and row of rotation matrix r_t denoted by $r_t^{3,3}$ is a function of orientations β_t and γ_t along the horizontal y and x axis respectively:

$$r_t^{3,3} = cos(\beta_t)cos(\gamma_t). \tag{6}$$

This implies that a shape is physically stable if $r_T^{3,3} = 1$. Consequently, we propose a stability loss that controls shape horizontal orientation at the final simulation step:

$$\mathcal{L}_s(z, \theta; G) = 1 - r_T^{3,3}. \tag{7}$$

In fact, we expect the shape to maintain its upright position during the dropping simulation.

3.7. Generalization to other shape representations.

Note that, even though we develop an approach that benefits from implicit shape representation as the current SotA method for generating high-quality geometry [7], our simulation layer can be also combined with other 3D shape representations. Specifically, our approach can be used directly with any 3D decoder f_θ if and only if, one can compute $\frac{\partial x}{\partial f_\theta}$ at the simulated $\mathcal{C}_{f_\theta}$, regardless of the architecture of f_θ. Note that computing $\frac{\partial x}{\partial f_\theta}$ amounts to assessing how the simulation gradient modifies the shape's geometry.

For demonstration, we provide additional detailed explanations along with examples for Point Cloud [31], Mesh [17] and Primitive [20] decoders, in the supplementary.

3.8. Training and Applications

Learning the latent space of shapes. Given a set of 3D shapes $\mathcal{S}$ paired with the ground truth SDF, we train our network using the reconstruction and our physical stability losses defined in Equations (2) and (7) respectively:

$$\mathcal{L}(z_i, \theta) = \mathcal{L}_r(z_i, \theta; X_i) + \alpha_s \mathcal{L}_s(z_i + \epsilon_\eta, \theta; G), \tag{8}$$

with $z_i \in \mathcal{Z}_{train}$ a learnable training vector, α_s a weighting factor and ϵ_η a d-dimensional noise depending on $\mathcal{Z}_{train}$

distribution. While $\mathcal{L}_r$ guarantees geometric relevance of the produced shapes, $\mathcal{L}_s$ aims not only to physically regularize training shapes ($\|\epsilon_\eta\|_2 = 0$), but also to make new realistic shapes' patterns appear in the latent space ($\|\epsilon_\eta\|_2 > 0$). To make sure that $z_i + \epsilon_\eta \sim \mathcal{Z}_{train} + \epsilon_\eta$ is relevant and lies within the latent space of shapes, we control ϵ_η norm consistently with $\mathcal{Z}_{train}$ vectors pairwise distances' norms. To this end, we use ϵ_η randomly sampled as follows:

$$\epsilon_\eta \sim \mathcal{U}(0, \eta). \frac{\mathcal{N}}{\|\mathcal{N}\|_2} . \mathbb{1}_{u \sim \mathcal{U}(0,1) < 0.1}$$

Here $\mathbb{1}$ is the indicator function, η equals 90% quantile of the $\mathcal{Z}_{train}$ vectors pairwise L_2 distances, and $\mathcal{U}$ and $\mathcal{N}$ are respectively the uniform and the d-dimensional standard normal distributions with d being the dimension of $\mathcal{Z}_{train}$ vectors. Please refer to our supplementary for more details.

Shape optimization. As a first application, we consider the task of single shape optimization to illustrate the functioning of our simulation layer SimL. We randomly sample $z_{in} \sim \mathcal{Z}_{train} + \epsilon_\eta$ and keep the z_{in} associated with unstable shapes. Our goal is to compute the optimal shape in terms of physical quality while incurring minimal shape variations to preserve the input shape structure. Formally, we compute:

$$\hat{z} = \arg\min_z \|z - z_{in}\|_2 + \alpha_s \mathcal{L}_s(z; \theta, G). \tag{9}$$

Shape reconstruction. We demonstrate that our physical loss preserves the latent space quality for the task of shape reconstruction thanks to our accurate and bias-free simulation gradient. For this, given a test shape S represented by its SDF points-values pairs $\{X_i, f_S(X_i)\}$, we compute:

$$\hat{z} = \arg\min_z \mathcal{L}_r(z; \theta, X_i). \tag{10}$$

In addition, we assess the impact of our physical module at the inference stage only, by considering the following reconstruction task:

$$\hat{z} = \arg\min_z \mathcal{L}_r(z; \theta_0, X_i) + \alpha_s \mathcal{L}_s(z; \theta_0, G), \tag{11}$$

where θ_0 denotes the baseline DeepSDF [32] decoder parameters, learned as in Equation 8 with $\alpha_s = 0$. Note that, in Equation 11, $\hat{z}$ is first computed with $\alpha_s = 0$ then finetuned using $\mathcal{L}_s$ with $\alpha_s > 0$. As such, if the shape is deemed stable by the physical module, then it remains unchanged.

4. Experiments

Data Preparation. To train the decoder f_θ, we use the 256^3 voxelized and flood-filled 3D models from ShapeNet Core dataset (v1) [6] provided in [19]. We start with converting shapes into meshes using Marching cubes from which we extract the ground truth SDF using the library proposed in [26]. Our motivation to start from voxeliezd shapes is to obtain watertight meshes that are convenient for SDF extraction. We consider the Chair, Bench and Table categories and use the train/validation/test splits similarly to [19]. Please refer to our supplementary for more statistics on the dataset.

Evaluation Metrics. All evaluations are conducted on meshes obtained by decoding shapes at a resolution of 256^3 and using Marching Cubes to recover the surface mesh. To evaluate the physical quality of the generated content, we report the **Stability Ratio (SR)** defined for a given set of shapes as the number of shapes that are physically stable divided by the total number of shapes. The stability assessment is performed using dropping simulation via the Bullet Physics Engine [10] to avoid bias in favor of our method. We provide more details on our evaluation procedure in our supplementary.

4.1. Shape optimization

We first compare our approach to several baselines for the task of shape optimization. As mentioned above, given a potentially unstable shape, our goal is to improve its stability while remaining close to the input geometry to preserve its structure.

As baselines, first we use DeepSDF+$\mathcal{L}_s$, which uses a fixed latent space pre-trained with the standard DeepSDF [32] and performs physical optimization using our differentiable simulator. Second, we also evaluate DeepSDF+ [30], introduced in [30], which uses a *surrogate model* to perform physical simulation, instead of our differentiable simulator. Finally, we evaluate our method, Phys-DeepSDF, that uses the latent space learned with our physical loss, and further exploits our differentiable simulator for shape optimization.

To train a surrogate model, we follow the approach in [30], and train a voxel-based neural stability predictor for each shape category to predict the physical stability of an input shape. To this end, we use the pre-trained baseline DeepSDF [32] to randomly sample N shapes following the distribution $\mathcal{Z}_{train} + \epsilon_\eta$. Then, we annotate each generated shape by stable $\{1\}$ or unstable $\{0\}$ using PyBullet [10]. The annotated dataset is used to train a surrogate model h (binary classifier) to predict the stability probability p. The baseline stability loss is hence $\mathcal{L}_s^{base} = max(0.5 - p, 0)$ (we preserve stable shapes with $p > 0.5$ as in [30]). Please refer to our supplementary for more details.

It is important to highlight that, in contrast to the surrogate model h, our SimL has no learnable parameters, and can therefore generalize to different shape categories with no annotation or pre-training effort.

To conduct our experiments, we create a test set $\mathcal{T}_{in}$ of N=500 shapes by randomly sampling $z_{in} \sim \mathcal{Z}_{train} + \epsilon_\eta$ associated with non stable shapes (SR=0, cf Figure 1 first row). Note that we create an independent $\mathcal{T}_{in}$ for the

DeepSDF [32] and Phys-DeepSDF based experiments. We denote the optimized shapes by $\mathcal{T}_f$ and the training shapes corresponding to $\mathcal{Z}_{train}$ by $\mathcal{T}_{train}$. For evaluation, we use the Chamfer Distance CD (30K points) and Latent Distance LD defined as the mean L_2 distance of latent codes, to measure the proximity of the produced shapes $\mathcal{T}_f$ to the input ones $\mathcal{T}_{in}$. We also compute the minimum matching distance [1] with respect to the training set using Chamfer distance MMD($\mathcal{T}_f$,$\mathcal{T}_{train}$) to capture the novelty of the optimized content and to make sure that the optimization process leads to discover new shapes rather than to overfit to training examples. We report the MMD results divided by MMD($\mathcal{T}_{in}$,$\mathcal{T}_{train}$) since $\mathcal{T}_{in}$ and $\mathcal{T}_{train}$ are different for DeepSDF [32] and Phys-DeepSDF based experiments. Finally, we compute SR values for the physical stability assessment.

We report quantitative results in Table 1. First observe that the SimL based experiment DeepSDF + $\mathcal{L}_s$ beats the baseline DeepSDF + [30] for nearly all metrics across the different shape categories. This validates that SimL leads to superior physical quality (high SR) with superior geometric consistency (low CD and LD), while proposing novel and more diverse shapes than the training examples (high MMD). Moreover, visual examination in Figure 1 reveals that SimL produces more plausible and accurate shape variations than its h counterpart. This is primarily due to the accurate simulation gradient, compared to the h model gradient that is heavily biased by the learning data patterns (cf Figure 3). Added to this is the fact that h approximates simulator behavior and is still prone to errors, whereas SimL accurately detects instabilities. Second, Phys-DeepSDF based experiments further improve the numerical results. We attribute this to the fact that Phys-DeepSDF latent space accommodates superior patterns of plausible shapes compared to the baseline DeepSDF [32].

Qualitative results in Figure 4 support the relevance of Phys-DeepSDF in improving physical stability and geometric consistency. Our approach not only delivers shapes with sufficient physical quality, but it also produces visually plausible solutions. We provide more qualitative illustrations in our supplementary along with a comparison to the train shapes.

4.2. Shape reconstruction

We use the Chamfer Distance to measure the reconstruction quality by randomly sampling 30K points from shapes' surfaces. Table 2 and Figure 5 provide quantitative and qualitative evaluations respectively. Our approach yields sufficient performance in terms of CD for all shape categories. Particularly, although it induces only a slight improvement over baseline DeepSDF [32], note that our method overcomes the limitation of surrogate model-based approach, which degrades the reconstruction quality, thanks to our accurate and bias-free physical gradient illustrated in Figure 3. Besides, we observe that Phys-DeepSDF yields a particular accuracy improvement for the Bench category. We attribute this to the fact that the latter has a limited training set compared to Chair and Table (1K compared to about 5K). The physical supervision helps to address this data limitation by endowing the network with physical insights concerning the space of plausible geometric configurations.

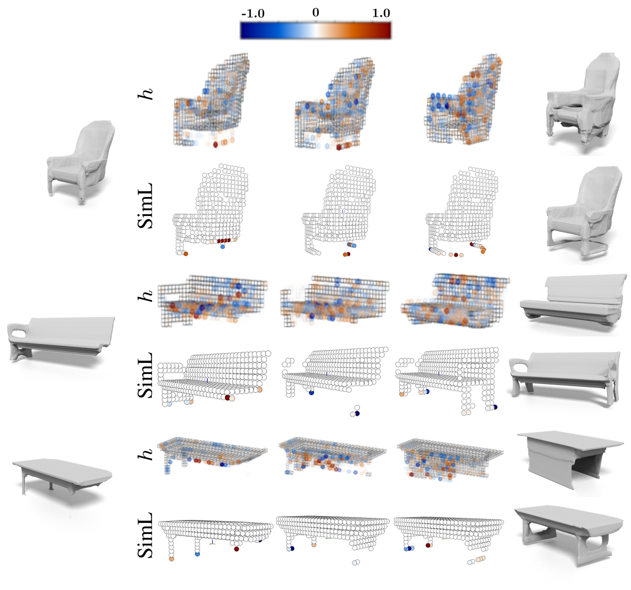

Figure 3. Visualization of shape optimization using our simulation layer SimL (bottom row of each example) and surrogate model h (top row of each example). We display normalized gradient values for intermediate iteration steps. We set gradients computed by h in voxels that remain outside the shape surface to 0 to avoid blur. Both approaches optimize the given examples. However, while SimL provides accurate gradient for relevant shape variations, h gradient is noisy and entails more severe shape changes.

	Net	CD↓	LD↓	SR↑	MMD↑
Chair	DeepSDF + [30]	1.40	0.45	59.8%	0.84
	DeepSDF + $\mathcal{L}_s$	0.69	0.34	62.8%	0.87
	Phys-DeepSDF (ours)	**0.68**	**0.10**	**71.8%**	**0.91**
Bench	DeepSDF + [30]	1.27	0.18	52.4%	**0.91**
	DeepSDF + $\mathcal{L}_s$	1.21	**0.12**	68.4%	0.88
	Phys-DeepSDF (ours)	**0.70**	0.13	**69.0%**	0.86
Table	DeepSDF + [30]	6.40	0.35	67.0%	0.80
	DeepSDF + $\mathcal{L}_s$	1.72	**0.09**	55.6%	0.82
	Phys-DeepSDF (ours)	**1.46**	0.10	**68.6%**	**0.86**

Table 1. Quantitative evaluation for shape optimization. Our physical module improves the quality of the latent space and delivers plausible solutions. CD is multiplied by 10^2.

4.3. Shape interpolation

Finally, we examine the relevance of our approach in the task of linear interpolation between training shapes. We

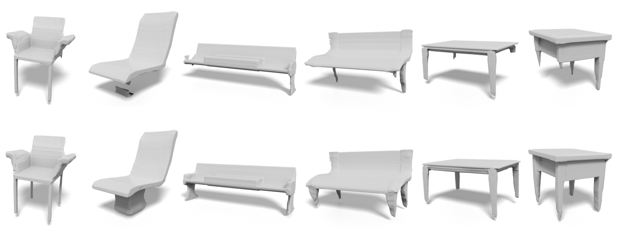

Figure 4. Visual results for 3D shape optimization using Phys-DeepSDF. Top row: initial shape. Bottom row: optimized shapes. We demonstrate the relevance of our approach on challenging examples randomly sampled from the Phys-DeepSDF latent space.

Metric	Net	Chair	Bench	Table
CD	DeepSDF	1.72	5.09	2.54
	DeepSDF + [30]	1.94	5.97	3.90
	DeepSDF + $\mathcal{L}_s$	1.73	5.44	2.59
	Phys-DeepSDF (ours)	**1.68**	**4.74**	**2.48**

Table 2. Quantitative evaluation for shape reconstruction. Given the stochastic nature of shape reconstruction (shapes are reconstructed via gradient descent with a random initialization), the reported values are the result of performing two reconstructions of each shape and keeping the one with the lowest CD. Our physically-aware latent space allows to efficiently recover unseen shapes. CD is multiplied by 10^3.

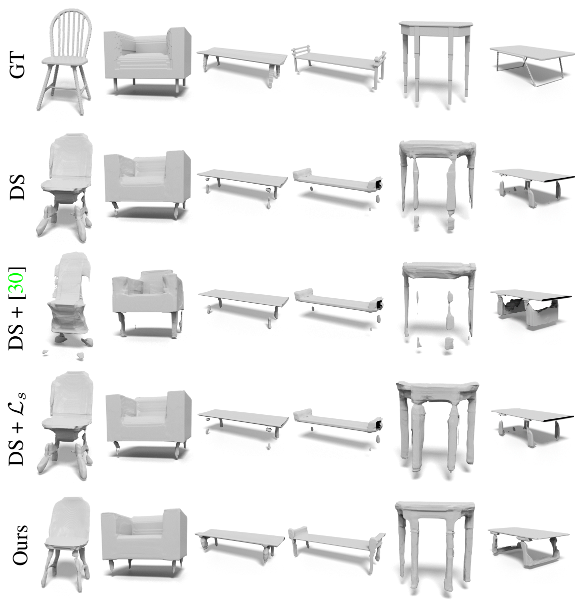

Figure 5. Qualitative results for 3D shape reconstruction. DS refers to baseline DeepSDF [32] and GT to ground truth. Our approach preserves geometric and physical plausibility.

select a challenging interpolation example from baseline DeepSDF [32] that we compare with its Phys-DeepSDF counterpart coupled with test time optimization using our

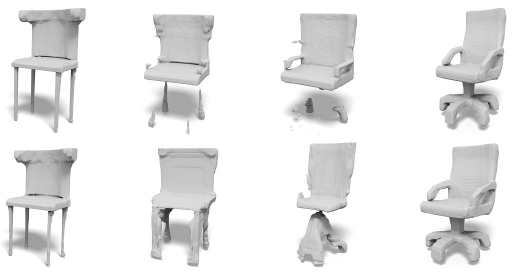

Figure 6. Qualitative results for 3D shape interpolation. We select a challenging interpolation example from baseline DeepSDF [32] (top row). Then, we visualize its counterpart using our Phys-DeepSDF coupled with optimization using SimL (bottom row).

physical loss $\mathcal{L}_s$ based on SimL. Figure 6 shows that physical regularization proves beneficial to ensure the validity of the generated content and the discovery of novel shapes.

5. Conclusion, Limitations & Future Work

We introduced a novel end-to-end trainable generative model with *online* differentiable physical simulation, which combines implicit generative modeling with explicit (point based) simulation. By using a differentiable simulator as part of training a generative model, and thus avoiding a surrogate model, our method helps to achieve better performance in terms of physical validity compared to baselines, while producing geometrically accurate and diverse shapes.

Our approach still has several limitations. First, although the rigid-body simulation is beneficial for the efficiency of our approach, the instabilities caused by *disconnected parts* are not addressed, since the simulator assumes that the distance between each two shape points is constant. For completeness, we give several illustrations in our supplementary and show how this limitation can be addressed by using a few iterations of topological regularization as post-processing. Second, we restrict the sampling resolution for physical evaluation to $R_G = 32$, which may mislead the physical evaluation. We address this limitation to a significant extent by using the projection step onto shape surface in Equation (3). Note that this limitation also applies to the discussed surrogate model-based approach.

In the future, it will be useful to extend our approach to other physical properties by exploiting differentiable simulation. It will also be interesting to exploit our approach in other applications, such as single-image 3D reconstruction, with limited training data. For completeness we provide a discussion of the societal impact in our supplementary.

Acknowledgements We thank the anonymous reviewers for their valuable feedback and suggestions. Parts of this work were supported by the ERC Starting Grant No. 758800 (EXPROTEA) and the ANR AI Chair AIGRETTE.

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas J Guibas. Learning representations and generative models for 3d point clouds. *arXiv preprint arXiv:1707.02392*, 2017. 1, 2, 7

[2] Amir Arsalan Soltani, Haibin Huang, Jiajun Wu, Tejas D. Kulkarni, and Joshua B. Tenenbaum. Synthesizing 3d shapes via modeling multi-view depth maps and silhouettes with deep generative networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 2

[3] 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. 2

[4] Ezer Bar-Aviv and Ehud Rivlin. Functional 3d object classification using simulation of embodied agent. In *British Machine Vision Conference*, 2006. 1

[5] David Baraff and David Baraff. Rigid body simulation. In *SIGGRAPH 95 Course Note 34. ACM SIGGRAPH*, 1992. 4

[6] 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. 6

[7] Siddhartha Chaudhuri, Daniel Ritchie, Kai Xu, and Hao (Richard) Zhang. Learning Generative Models of 3D Structures. In Wenzel Jakob and Enrico Puppo, editors, *Eurographics 2019 - Tutorials*. The Eurographics Association, 2019. 5

[8] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2

[9] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 4

[10] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016–2019. 2, 4, 6

[11] Filipe de Avila Belbute-Peres, Kevin Smith, Kelsey Allen, Josh Tenenbaum, and J. Zico Kolter. End-to-end differentiable physics for learning and control. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. 2

[12] Yilun Du, Zhijian Liu, Hector Basevi, Ales Leonardis, Bill Freeman, Josh Tenenbaum, and Jiajun Wu. Learning to exploit stability for 3d scene parsing. In *Advances in Neural Information Processing Systems 31*. Curran Associates, Inc., 2018. 2, 3

[13] Yueqi Duan, Haidong Zhu, He Wang, Li Yi, Ram Nevatia, and Leonidas J. Guibas. Curriculum deepsdf, 2020. 2

[14] Kiana Ehsani, Shubham Tulsiani, Saurabh Gupta, Ali Farhadi, and Abhinav Gupta. Use the force, luke! learning to predict physical forces by simulating effects. In *Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[15] Lin Gao, Jie Yang, Tong Wu, Yu-Jie Yuan, Hongbo Fu, Yu-Kun Lai, and Hao(Richard) Zhang. SDM-NET: Deep generative network for structured deformable mesh. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH Asia 2019)*, 38(6):243:1–243:15, 2019. 2

[16] R. Girdhar, D.F. Fouhey, M. Rodriguez, and A. Gupta. Learning a predictable and generative vector representation for objects. In *ECCV*, 2016. 2

[17] 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. 2, 5

[18] Yanran Guan, Tansin Jahan, and Oliver van Kaick. Generalized autoencoder for volumetric shape generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020. 2

[19] Christian Hane, Shubham Tulsiani, and Jitendra Malik. Hierarchical surface prediction for 3d object reconstruction. *3DV*, 2017. 6

[20] 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. 5

[21] Amir Hertz, Rana Hanocka, Raja Giryes, and Daniel Cohen-Or. Pointgmm: A neural gmm network for point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2

[22] Lauren Hinkle and Edwin Olson. Predicting object functionality using physical simulations. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, November 2013. 1

[23] Ruizhen Hu, Manolis Savva, and Oliver van Kaick. Functionality representations and applications for shape analysis. *EUROGRAPHICS STAR 2018*, 37(2):603–624, 2018. 1

[24] Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Frédo Durand. Difftaichi: Differentiable programming for physical simulation. *ICLR*, 2020. 2, 4

[25] Y. Hu, J. Liu, A. Spielberg, J. B. Tenenbaum, W. T. Freeman, J. Wu, D. Rus, and W. Matusik. Chainqueen: A real-time differentiable physical simulator for soft robotics. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6265–6271, 2019. 2

[26] Marian Kleineberg, Matthias Fey, and Frank Weichert. Adversarial Generation of Continuous Implicit Shape Representations. In Alexander Wilkie and Francesco Banterle, editors, *Eurographics 2020 - Short Papers*. The Eurographics Association, 2020. 2, 6

[27] Jun Li, Kai Xu, Siddhartha Chaudhuri, Ersin Yumer, Hao Zhang, and Leonidas Guibas. Grass: Generative recursive autoencoders for shape structures. *ACM Transactions on*

Graphics (Proc. of SIGGRAPH 2017), 36(4):to appear, 2017. 1

[28] Gidi Littwin and Lior Wolf. Deep meta functionals for shape representation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 2

[29] Zhijian Liu, William T. Freeman, Joshua B. Tenenbaum, and Jiajun Wu. Physical primitive decomposition. In *The European Conference on Computer Vision (ECCV)*, September 2018. 2

[30] 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. 1, 2, 3, 6, 7, 8

[31] 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. 5

[32] 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, 2, 3, 4, 6, 7, 8

[33] Romain Prévost, Emily Whiting, Sylvain Lefebvre, and Olga Sorkine-Hornung. Make It Stand: Balancing shapes for 3D fabrication. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH)*, 32(4):81:1–81:10, 2013. 3

[34] 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. 2, 5

[35] Gernot Riegler, Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2017*, Piscataway, NJ, USA, July 2017. IEEE. 2

[36] Dule Shu, James Cunningham, Gary Stump, Simon W. Miller, Michael A. Yukish, Timothy W. Simpson, and Conrad S. Tucker. 3d design using generative adversarial networks and physics-based validation. In *Journal of Mechanical Design, 142(7)*, 2020. 1, 2, 3

[37] Edward J. Smith and David Meger. Improved adversarial systems for 3d object generation and reconstruction. *CoRR*, abs/1707.09557, 2017. 2

[38] Qingyang Tan, Lin Gao, Yu-Kun Lai, and Shihong Xia. Variational autoencoders for deforming 3d mesh models. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 5841–5850. IEEE Computer Society, 2018. 2

[39] M. Tatarchenko, A. Dosovitskiy, and T. Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *IEEE International Conference on Computer Vision (ICCV)*, 2017. 2

[40] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *ECCV*, 2018. 2

[41] Hongtao Wu, Deven Misra, and Gregory S. Chirikjian. Is that a chair? imagining affordances using simulations of an articulated human body. In *2020 IEEE International Conference on Robotics and Automation, ICRA 2020, Paris, France, May 31 - August 31, 2020*, pages 7240–7246. IEEE, 2020. 1

[42] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 82–90. Curran Associates, Inc., 2016. 1, 2

[43] Bo Zheng, Yibiao Zhao, Joey Yu, Katsushi Ikeuchi, and Song-Chun Zhu. Scene understanding by reasoning stability and safety. *IJCV*, 112(2):221–238, 2015. 3

[44] Bo Zheng, Yibiao Zhao, Joey C Yu, Katsushi Ikeuchi, and Song-Chun Zhu. Beyond point clouds: Scene understanding by reasoning geometry and physics. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 3127–3134. IEEE, 2013. 3

[45] Chenyang Zhu, Kai Xu, Siddhartha Chaudhuri, Renjiao Yi, and Hao Zhang. Scores: Shape composition with recursive substructure priors. *ACM Transactions on Graphics (SIGGRAPH Asia 2018)*, 37(6):to appear, 2018. 2

[46] Yixin Zhu, Tao Gao, Lifeng Fan, Siyuan Huang, Mark Edmonds, Hangxin Liu, Feng Gao, Chi Zhang, Siyuan Qi, Nian Ying Wu, B. Joshua Tenenbaum, and Song-Chun Zhu. Dark, beyond deep: A paradigm shift to cognitive ai with humanlike common sense. *Engineering*, 2020. 2