Physical Primitive Decomposition

Zhijian Liu[1], William T. Freeman[1,2], Joshua B. Tenenbaum[1], and Jiajun Wu[1]

[1] Massachusetts Institute of Technology
[2] Google Research

Abstract. Objects are made of parts, each with distinct geometry, physics, functionality, and affordances. Developing such a distributed, physical, interpretable representation of objects will facilitate intelligent agents to better explore and interact with the world. In this paper, we study *physical primitive decomposition*—understanding an object through its components, each with physical and geometric attributes. As annotated data for object parts and physics are rare, we propose a novel formulation that learns physical primitives by explaining both an object's appearance and its behaviors in physical events. Our model performs well on block towers and tools in both synthetic and real scenarios; we also demonstrate that visual and physical observations often provide complementary signals. We further present ablation and behavioral studies to better understand our model and contrast it with human performance.

1 Introduction

Humans use a hammer by holding its handle and striking its head, not vice versa. In this simple action, people demonstrate their understanding of *functional parts* [33, 39]: a tool, or any object, can be decomposed into primitive-based components, each with distinct physics, functionality, and affordances [17].

How to build a machine of such competency? In this paper, we tackle the problem of *physical primitive decomposition (PPD)*—explaining the shape and the physics of an object with a few shape primitives with physical parameters. Given the hammer in Figure 1, our goal is to build a model that recovers its two major components: a tall, wooden cylinder for its handle, and a smaller, metal cylinder for its head.

For this task, we need a physical, part-based object shape representation that models both object geometry and physics. Ground-truth annotations for such representations are however challenging to obtain: large-scale shape repositories like ShapeNet [8] often have limited annotations on object parts, let alone physics. This is mostly due to two reasons. First, annotating object parts and physics is labor-intensive and requires strong domain expertise, neither of which can be offered by current crowdsourcing platforms. Second, there exist intrinsic ambiguity in the ground truth: it is impossible to precisely label underlying physical object properties like densities from only images or videos.

Let's think more about what these representations are *for*. We want our object representation to faithfully encode its geometry; therefore, it should be able to explain our visual observation of the object's appearance. Further, as
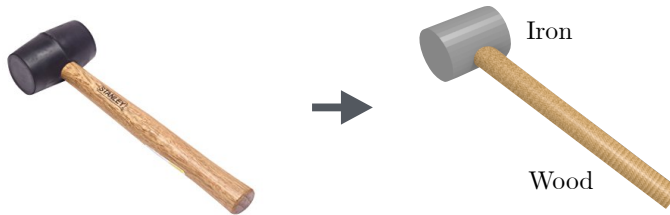
Fig. 1. A hammer (left) and its physical primitive decomposition (right).

the representation models object physics, it should be effective in explaining the object's behaviors in various physical events.

Inspired by this, we propose a novel formulation that learns a part-based object representation from both visual observations and physical interactions. Starting with a single image and a voxelized shape, the model recovers the geometric primitives and infers their physical properties from texture. The physical representation inferred this way is of course rather uncertain; it therefore only serves as the model's prior of this physical shape. Observing object behaviors in physical events offers crucial additional information, as objects with different physical properties behave differently in physical events. This is used by the model in conjunction with the prior to produce its final prediction.

We evaluate our system for physical primitive decomposition in three scenarios. First, we generate a dataset of synthetic block towers, where each block has distinct geometry and physics. Our model is able to successfully reconstruct the physical primitives by making use of both appearance and motion cues. Second, we evaluate the system on a set of synthetic tools, demonstrating its applicability to daily-life shapes. Third, we build a new dataset of real block towers in dynamic scenes, and evaluate the model's generalization power to real videos.

We further present ablation studies to understand how each source of information contributes to the final performance. We also conduct human behavioral experiments to contrast the performance of the model with humans. In a 'which block is heavier' experiment, our model performs comparably to humans.

Our contributions in this paper are three-fold. First, we propose the problem of physical primitive decomposition—learning a compact, disentangled object representation in terms of physical primitives. Second, we present a novel learning paradigm that learns to characterize shapes in physical primitives to explain both their geometry and physics. Third, we demonstrate that our system can achieve good performance on both synthetic and real data.

2   Related Work

Primitive-Based 3D Representations.. Early attempts on modeling 3D shapes with primitives include decomposing them into blocks [34], generalized cylinders [6], and geons [5]. This idea has been constantly revisited throughout the development of computer vision [11, 13, 2]. To name a few, Gupta *et al.* [11] modeled scenes as qualitative blocks, and van den Hengel *et al.* [13] as Lego blocks. More recently, Tulsaini *et al.* [40] combined the new and the old—using
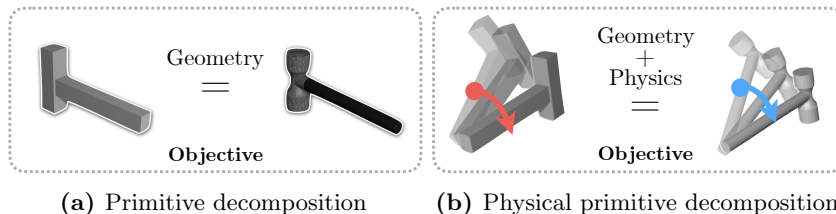
(a) Primitive decomposition   **(b)** Physical primitive decomposition

Fig. 2. Primitive decomposition **(a)** and physical primitive decomposition **(b)**. Both tasks attempt to convert an object into a set of primitives yet with different purposes: the former problem targets at shape reconstruction, while the latter one aims to recover both geometric and physical properties.

deep convolutional network to generate primitives of a given 3D shape; later, Zou *et al.* proposed 3D-PRNN [53], enhancing the flexibility of the system by leveraging modern advancement in recurrent generative models [41].

Primitive-based representations have profound impact that goes far beyond the field of computer vision. Scientists have employed this representation for user-interactive design [16] and for teaching robots to grasp objects [29]. In the field of computer graphics, the idea of modeling shapes as primitives or parts has also been extensively explored [50, 47, 27, 19, 21, 2]. Researchers have used the part-based representation for single-image shape reconstruction [15], shape completion [37], and probabilistic shape synthesis [14, 25].

Physical Shape and Scene Modeling. Beyond object geometry, there have been growing interests in modeling physical object properties and scene dynamics. The computer vision community has put major efforts in building rich and sizable databases. ShapeNet-Sem [36] is a collection of object shapes with material and physics annotations within the web-scale shape repository ShapeNet [8]. Material in Context Database (MINC) [4] is a gigantic dataset of materials in the wild, associating patches in real-world images with 23 materials.

Research on physical object modeling dates back to the study of "functional parts" [33, 39, 17]. The field of learning object physics and scene dynamics has prospered in the past few years [23, 1, 18, 3, 48, 30, 32, 7, 38, 20, 26]. Among them, there are a few papers that explicitly build physical object representations [30, 43, 45, 44, 49]. Though they also focus on understanding object physics [43, 45], functionality [51, 46], and affordances [22, 10, 52], these approaches usually assume a homogeneous object with simple geometry. In our paper, we model an object using physical primitives for richer expressiveness and higher precision.

3 Physical Primitive Decomposition

3.1 Problem Statement

Both primitive decomposition and physical primitive decomposition attempt to approximate an object with primitives. We highlight their difference in Figure 2.

Primitive Decomposition. As formulated in Tulsaini *et al.* [40] and Zou *et al.* [53], primitive decomposition aims to decompose an object O into a set of simple transformed primitives $x = \{x_k\}$ so that these primitives can accurately
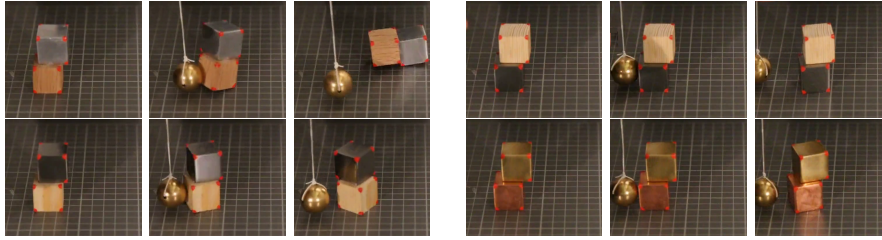
(a) **Above**: Aluminum and Wood;
  Below: Iron and Wood.

(b) **Above**: Wood and Iron;
  Below: Two Coppers.

Fig. 3. Challenges of inferring physical parameters from visual and physical observations: objects with different physical parameters might have **(a)** similar visual appearance or **(b)** similar physics trajectory.

approximate its geometry shape. This task can be seen as to minimize

$$\mathcal{L}_{\mathrm{G}}(x) = \mathcal{D}_{\mathrm{S}}\big(\mathcal{S}\big(\underset{k}{\cup}\, x_k\big), \mathcal{S}(O)\big), \tag{1}$$

where $\mathcal{S}(\cdot)$ denotes the geometry shape (*i.e.* point cloud), and $\mathcal{D}_{\mathrm{S}}(\cdot,\cdot)$ denotes the distance metric between shapes (*i.e.* earth-mover's distance [35]).

Physical Primitive Decomposition. In order to understand the functionality of object parts, we require the decomposed primitives $x = \{x_k\}$ to also approximate the physical behavior of object O. To this end, we extend the previous objective function with an additional physics term:

$$\mathcal{L}_{\mathrm{P}}(x) = \sum_{p \in \mathcal{P}} \mathcal{D}_{\mathrm{T}}\big(\mathcal{T}_p\big(\underset{k}{\cup}\, x_k\big), \mathcal{T}_p(O)\big), \tag{2}$$

where $\mathcal{T}_p(\cdot)$ denotes the trajectory after physics interaction p, $\mathcal{D}_{\mathrm{T}}(\cdot,\cdot)$ denotes the distance metric between trajectories (*i.e.* mean squared error), and $\mathcal{P}$ denotes a predefined set of physics interactions. Therefore, the task of physical primitive decomposition is to minimize an overall objective function constraining both geometry and physics: $\mathcal{L}(x) = \mathcal{L}_{\mathrm{G}}(x) + w \cdot \mathcal{L}_{\mathrm{P}}(x)$, where w is a weighting factor.

3.2   Primitive-Based Representation

We design a structured primitive-based object representation, which describes an object by listing all of its primitives with different attributes. For each primitive x_k, we record its size $x_k^{\mathrm{S}} = (s_x, s_y, s_z)$, position in 3D space $x_k^{\mathrm{T}} = (p_x, p_y, p_z)$, rotation in quaternion form $x_k^{\mathrm{R}} = (q_w, q_x, q_y, q_z)$. Apart from these geometry information, we also track its physical properties: density x_k^{D}.

In our object representation, the shape parameters, x_k^{S}, x_k^{T} and x_k^{R}, are vectors of continuous real values, whereas the density parameter x_k^{D} is a discrete value. We discretize the density values into $N_{\mathrm{D}} = 100$ slots, so that estimating density becomes a N_{D}-way classification. Discretization helps to deal with multi-modal density values. Figure 3a shows that two parts with similar visual appearance may have very different physical parameters. In such cases, regression with an $\mathcal{L}_2$ loss will encourage the model to predict the *mean* value of possible densities; in contrast, discretization allows it to give high probabilities to every possible density. We then figure out which candidate value is optimal from the trajectories.
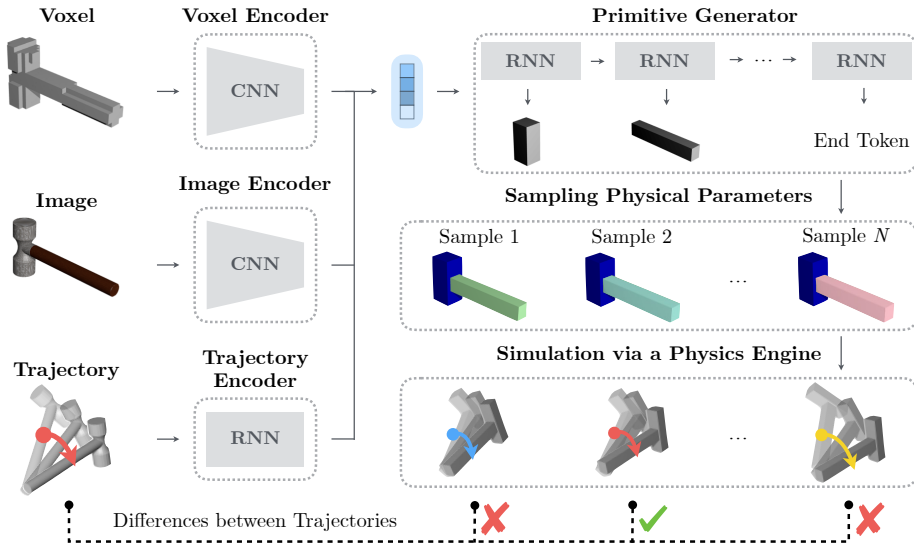
Fig. 4. Overview of our PPD model.

4   Approach

In this section, we discuss our approach to the problem of physical primitive decomposition (PPD). We present an overview of our framework in Figure 4.

4.1   Overview

Inferring physical parameters from solely visual or physical observation is highly challenging. This is because two objects with different physical parameters might have similar visual appearance (Figure 3a) or have similar physics trajectories (Figure 3b). Therefore, our model takes both types of observations as input:

1. **Visual Observation**. We take a voxelized shape and an image as our input because they can provide us with valuable visual information. Voxels help us recover object geometry, and images contain texture information of object materials. Note that, even with voxels as input, it is still highly nontrivial to infer geometric parameters: the model needs to learn to segment 3D parts within the object — an unsolved problem by itself [40].
2. **Physics Observation**. In order to explain the physical behavior of an object, we also need to observe its response after some physics interactions. In this work, we choose to use 3D object trajectories rather than RGB (or RGB-D) videos. Its abstractness enables the model to transfer better from synthetic to real data, because synthetic and real videos can be starkly different; in contrast, it's easy to generate synthetic 3D trajectories that look realistic.

Specifically, our network takes a voxel V, an image I, and N_{T} object trajectories $\boldsymbol{T} = \{T_k\}$ as input. V is a 3D binary voxelized grid, I is a single RGB image, and $\boldsymbol{T}$ consists of several object trajectories T_k, each of which records the response to one specific physics interaction. Trajectory T_k is a sequence of 3D object pose

$(p_x, p_y, p_z, q_w, q_x, q_y, q_z)$, where (p_x, p_y, p_z) denotes the object's center position and quaternion (q_w, q_x, q_y, q_z) denotes its rotation at each time step.

After receiving the inputs, our network encodes voxel, image and trajectory with separate encoders, and sequentially predicts primitives using a recurrent primitive generator. For each primitive, the network predicts its geometry shape (*i.e.* scale, translation and rotation) and physical property (*i.e.* density). More details of our model can be found in the supplementary material.

Voxel Encoder. For input voxel V, we employ a 3D volumetric convolutional network to encode the 3D shape information into a voxel feature f_V.

Image Encoder. For input image I, we pass it into the ResNet-18 [12] encoder to obtain an image feature f_I. We refer the readers to He *et al.* [12] for details.

Trajectory Encoder. For input trajectories $\boldsymbol{T}$, we encode each trajectory T_k into a low-dimensional feature vector h_k with a separate bi-directional recurrent neural network. Specifically, we feed the trajectory sequence, T_k, and also the same trajectory sequence in reverse order, T_k^{reverse}, into two encoding RNNs, to obtain two final hidden states: $h_k^{\rightarrow} = \text{encode}_k^{\rightarrow}(T_k)$ and $h_k^{\leftarrow} = \text{encode}_k^{\leftarrow}(T_k^{\text{reverse}})$. We take $[h_k^{\rightarrow}; h_k^{\leftarrow}]$ as the feature vector h_k. Finally, we concatenate the features of each trajectory, $\{h_k \mid k = 1, 2, \ldots, N_T\}$, and project it into a low-dimensional trajectory feature f_T with a fully-connected layer.

Primitive Generator. We concatenate the voxel feature f_V, image feature f_I and trajectory feature f_T together as $\hat{f} = [f_V; f_I; f_T]$, and map it to a low-dimensional feature f using a fully-connected layer. We predict the set of physical primitives $\{x_k\}$ sequentially by a recurrent generator.

At each time step k, we feed the previous generated primitive x_{k-1} and the feature vector f in as input, and we receive one hidden vector h_k as output. Then, we compute the new primitive $x_k = (x_k^D, x_k^S, x_k^T, x_k^R)$ as

$$x_k^D = \text{softmax}(W_D \times h_k + b_D), \quad x_k^S = \text{sigmoid}(W_S \times h_k + b_S) \times C_S,$$

$$x_k^T = \tanh(W_T \times h_k + b_T) \times C_T, \quad x_k^R = \frac{W_R \times h_k + b_R}{\max(\|W_R \times h_k + b_R\|_2, \epsilon)}, \tag{3}$$

where C_S and C_T are scaling factors, and $\epsilon = 10^{-12}$ is a small constant for numerical stability. Equation 3 guarantees that x_k^S is in the range of $[0, C_S]$, x_k^T is in the range of $[-C_T, C_T]$, and $\|x_k^R\|_2$ is 1 (if ignoring ϵ), which ensures that x_k will always be a valid primitive. In our experiments, we set $C_S = C_T = 0.5$, since we normalize all objects so that they can fit in unit cubes. Also note that, x_k^D is an $(N_D + 2)$-dimensional vector, where the first N_D dimensions indicate different density values and the last two indicate the "start token" and "end token".

Sampling and Simulating with the Physics Engine. During testing time, we treat the predicted x_k^D as a multinomial distribution, and we sample multiple possible predictions from it. For each sample, we use its physical parameters to simulate the trajectory with a physics engine. Finally, we select the one whose simulated trajectory is closest to the observed trajectory.

An alternative way to incorporate physics engine is to directly optimize our model over it. As most physics engines are not differentiable, we employ REIN-FORCE [42] for optimization. Empirically, we observe that this reinforcement

learning based method performs worse than sampling-based methods, possible due to the large variance of the approximate gradient signals.

Simulating with a physics engine requires we know the force during testing. Such an assumption is essential to ensure the problem is well-posed: without knowing the force, we can only infer the relative part density, but not the actual values. Note that in many real-world applications such as robot manipulation, the external force is indeed available.

4.2   Loss Functions

Let $x = (x_1, x_2, \ldots, x_n)$ and $\hat{x} = (\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_m)$ be the predicted and ground-truth physical primitives, respectively. Our loss function consists of two terms, geometry loss $\mathcal{L}_\mathrm{G}$ and physics loss $\mathcal{L}_\mathrm{D}$:

$$\mathcal{L}_\mathrm{G}(x, \hat{x}) = \sum_k \left(\omega_\mathrm{S} \cdot \|x_k^\mathrm{S} - \hat{x}_k^\mathrm{S}\|_1 + \omega_\mathrm{T} \cdot \|x_k^\mathrm{T} - \hat{x}_k^\mathrm{T}\|_1 + \omega_\mathrm{R} \cdot \|x_k^\mathrm{R} - \hat{x}_k^\mathrm{R}\|_1 \right), \quad (4)$$

$$\mathcal{L}_\mathrm{P}(x, \hat{x}) = -\sum_k \sum_i \hat{x}_k^\mathrm{D}(i) \cdot \log x_k^\mathrm{D}(i), \quad (5)$$

where ω_S, ω_T and ω_R are weighting factors, which are set to 1's because x^S, x^T and x^R are of the same magnitude (10^{-1}) in our datasets. Integrating Equation 4 and Equation 5, we define the overall loss function as $\mathcal{L}(x, \hat{x}) = \mathcal{L}_\mathrm{G}(x, \hat{x}) + w \cdot \mathcal{L}_\mathrm{P}(x, \hat{x})$, where w is set to ensure that $\mathcal{L}_G$ and $\mathcal{L}_P$ are of the same magnitude.

Part Associations. In our formulation, object parts (physical primitives) follow a pre-defined order (*e.g.*, from bottom to top), and our model is encouraged to learn to predict the primitives in the same order.

5   Experiments

We evaluate our PPD model on three diverse settings: synthetic block towers where blocks are of various materials and shapes; synthetic tools with more complex geometry shapes; and real videos of block towers to demonstrate our transferability to real-world scenario.

5.1   Decomposing Block Towers

We start with decomposing block towers (stacks of blocks).

Block Towers. We build the block towers by stacking variable number of blocks (2-5 in our experiments) together. We first sample the size of each block and then compute the center position of blocks from bottom to top. For the kth block, we denote the size as (w_k, h_k, d_k), and its center (x_k, y_k, z_k) is sampled and computed by $x_k \sim \mathcal{N}(x_{k-1}, w_{k-1}/4)$, $y_k \sim \mathcal{N}(y_{k-1}, h_{k-1}/4)$, and $z_k = z_{k-1} + (d_{k-1} + d_k)/2$, where $\mathcal{N}(\mu, \sigma)$ is a normal distribution with mean μ and standard deviation σ. We illustrate some constructed block towers in Figure 5. We perform the exact voxelization with grid size of $32 \times 32 \times 32$ by binvox, a 3D mesh voxelizer [31].

Materials. In our experiments, we use five different materials, and follow their real-world densities with minor modifications. The materials and the ranges of their densities are listed in Table 1. For each block in the block towers, we first assign it to one of the five materials, and then uniformly sample its density from possible values of its material. We generate 8 configurations for each block tower.
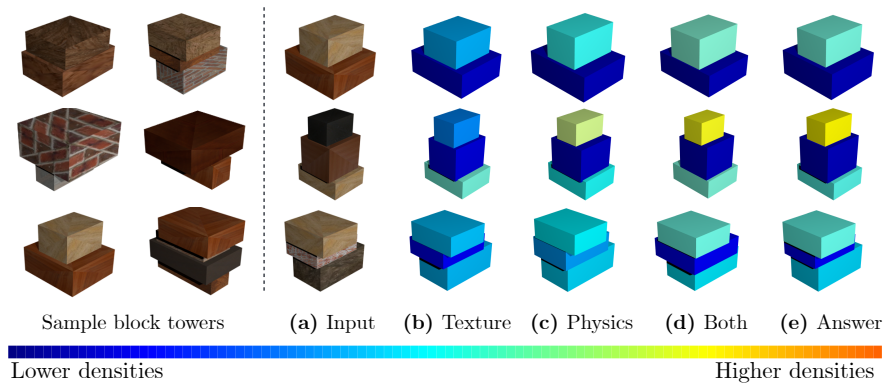
Sample block towers (a) Input (b) Texture (c) Physics (d) Both (e) Answer

Lower densities   Higher densities

Fig. 5. Sample objects in our block towers dataset (left) and qualitative results of our model with different combinations of observations as input (right).

Material	Wood	Brick	Stone	Ceramic	Metal
Density	$[1, 10]$	$[11, 20]$	$[21, 30]$	$[31, 60]$	$[21, 35] \cup [71, 100]$

Table 1. Materials and their real-world density values (unit: $\times 10^2 \cdot \text{kg/m}^3$). Objects made of similar materials (different types of metals) may have different physical properties, while different materials (*i.e.*, stone and metal) may have same physical properties.

Textures. We obtain the textures for materials by cropping the center portion of images from the MINC dataset [4]. We show sample images rendered with material textures in Figure 5. Since we render the textures only with respect to the material, the images rendered do not provide any information about density.

Physics Interactions. We place the block towers at the origin and perform four physics interactions to obtain the object trajectories ($N_\text{T} = 4$). In detail, we exert a force with the magnitude of 10^5 on the block tower from four pre-defined positions $\{(\pm 1, -1, \pm 1)\}$. We simulate each physics interaction for 256 time steps using the Bullet Physics Engine [9]. To ensure simulation accuracy, we set the time step for simulation to 1/300s.

Metrics. We evaluate the performance of shape reconstruction by the F_1 score between the prediction and ground truth: each primitive in prediction is labeled as a true positive if its intersection over union (IoU) with a ground-truth primitive is greater than 0.5. For physics estimation, we employ two types of metrics, i) density measures: top-k accuracy ($k \in \{1, 5, 10\}$) and root-mean-square error (RMSE) and ii) trajectory measure: mean-absolute error (MAE) between simulated trajectory (using predicted the physical parameters) and ground-truth trajectory.

Methods. We evaluate our model with different combinations of observations as input: i) texture only (*i.e.*, no trajectory, by setting $f_\text{T} = 0$), ii) physics only (*i.e.*, no image, by setting $f_\text{I} = 0$), iii) both texture and physics but without the voxelized shape, iv) both texture and physics but with replacing the 3D trajectory with a raw depth video, v) full data in our original setup (image, voxels, and trajectory). We also compare our model with several baselines: i) predicting the most frequent density in the training set (*Frequent*), ii) nearest neighbor retrieval from the training set (*Nearest*), and iii) knowing the ground-truth material and

| Methods | Observations | | Density | | | | | Trajectory |
| | Texture | Physics | Accuracy | | | RMSE | | MAE |
			Top 1	Top 5	Top 10			
Frequent	−	−	2.0	9.7	13.4	25.4		74.4
Nearest	−	+	1.9	7.9	12.4	41.1		91.0
Oracle	+	−	6.9	35.7	72.0	18.5		51.3
PPD (no trajectory)	+	−	7.2	35.2	69.5	19.0		51.7
PPD (no image)	−	+	7.1	31.0	50.8	16.7		36.4
PPD (no voxels)	+	+	15.9	56.3	82.4	10.3		29.9
PPD (RGB-D)	+	+	11.6	50.5	79.5	12.8		30.2
PPD (full)	+	+	16.1	56.4	82.5	9.9		21.0
PPD (full)+Sample	+	+	**18.2**	**59.7**	**84.0**	**8.8**		**13.9**

Table 2. Quantitative results of physical parameter estimation on block towers. Combining appearance with physics does help our model to achieve better estimation on physical parameters, and our model performs significantly better than all other baselines.

guessing within its density value range (*Oracle*). While all these baselines assume perfect shape reconstruction, our model learns to decompose the shape.

Results. For the shape reconstruction, our model achieves 97.5 in terms of F1 score. For the physics estimation, we present quantitative results of our model with different observations as input in Table 2. We compare our model with an oracle that infers material properties from appearance while assuming ground-truth reconstruction. It gives *upper-bound* performance of methods that rely on only appearance cues. Experiments suggest that appearance alone is not sufficient for density estimation. From Table 2, we observe that combining appearance with physics performs well on physical parameter estimation, which is because the object trajectories can provide crucial additional information about the density distribution (*i.e.* moment of inertia). Also, all input modalities and sampling contribute to the model's final performance.

We have also implemented a physics engine–based sampling baseline: sampling the shape and physical parameters for each primitive, using a physics engine for simulation, and selecting the one whose trajectory is closest to the observation.

	1×	8×	64×	512×
Sample Phys.+Shape	142.2	87.1	70.8	58.7
Sample Phys.		89.7	60.1	38.7	22.7
PPD (ours)	**21.0**	**15.1**	**13.9**	**13.2**

Table 3. Comparison between our model and a physics engine based sampling baseline

We also compare with a stronger baseline where we only sample physics, assuming ground-truth shape is known. Table 3 shows our model works better and is more efficient: the neural nets have learned an informative prior that greatly reduces the need of sampling at test time.

5.2   Decomposing Tools

We then demonstrate the practical applicability of our model by decomposing synthetic real-world tools.

Methods	Observations		Density				Trajectory
	Texture	Physics	Accuracy			RMSE	MAE
			Top 1	Top 5	Top 10		
Frequent	−	−	2.5	10.2	13.6	25.9	348.2
Nearest	−	+	2.9	8.3	12.4	25.8	329.7
Oracle	+	−	7.4	35.2	72.0	19.1	185.8
PPD (no trajectory)	+	−	7.7	36.4	71.1	16.8	206.8
PPD (no image)	−	+	15.0	56.3	80.2	5.9	143.6
PPD (full)	+	+	35.7	**85.2**	95.8	2.6	103.6
PPD (full)+Sample	+	+	**38.3**	85.0	**96.1**	**2.5**	**74.4**

Table 4. Quantitative results of physical parameter estimation on tools. Combining visual appearance with physics observations helps our model to perform much better on physical parameter estimation, and compared to all other baselines, our model performs significantly better on this dataset.

Tools. Because of the absence of tool data in the ShapeNet Core [8] dataset, we download the tools from 3D Warehouse*** and manually remove all unrelated models. In total, there are 204 valid tools, and we use Blender to remesh and clean up these tools to fix the issues with missing faces and normals. Following Chang et al. [8], we perform PCA on the point clouds and align models by their PCA axes. Sample tools in our dataset are shown in Figure 6.

Primitives. Similar to Zou et al. [53], we first use the energy-based optimization to fit the primitives from the point clouds, and then, we assign each vertex to its nearest primitive and refine each primitive with the minimum oriented bounding box of vertices assigned to it.

Other Setups. We make use of the same set of materials and densities as in Table 1 and the same textures for materials as described in Section 5.1. Sample images rendered with textures are shown in Figure 6. As for physics interactions, we follow the same scenario configurations as in Section 5.1.

Training Details. Because the size of synthetic tools dataset is rather limited, we first pre-train our PPD model on the block towers and then finetune it on the synthetic tools. For the block towers used for pre-training, we fix the number of blocks to 2 and introduce small random noises and rotations to each block to fill the gap between block towers and synthetic tools.

Results. For the shape reconstruction, our model achieves 85.9 in terms of F1 score. For the physics estimation, we present quantitative results in Table 4. The shape reconstruction is not as good as that of the block towers dataset because the synthetic tools are more complicated, and the orientations might introduce some ambiguity (there might exist multiple bounding boxes with different rotations for the same part of object). The physics estimation performance is better since the number of primitives in our synthetic tools dataset is very small (≤ 2 in general). We also show some qualitative results in Figure 6.

*** `https://3dwarehouse.sketchup.com`

| Sample hand tools | **(a)** Input | **(b)** Texture | **(c)** Physics | **(d)** Both | **(e)** Answer |

Lower densities   Higher densities
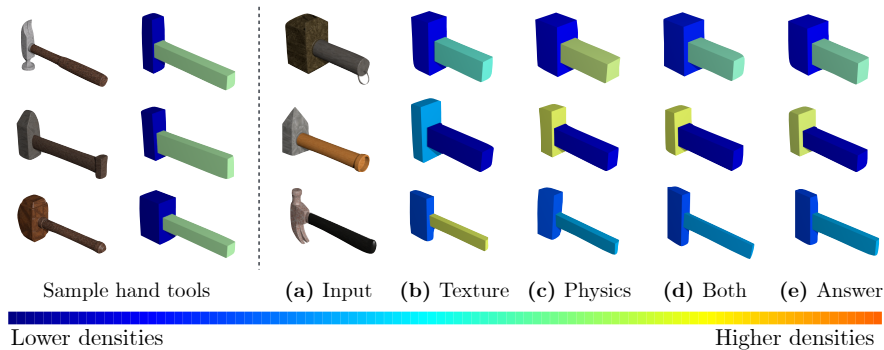
Fig. 6. Sample objects in synthetic tools dataset (left) and qualitative results of our model with different combinations of observations as input (right).

5.3   Decomposing Real Objects

We look into real objects to evaluate the generalization ability of our model.

Real-World Block Towers. We purchase totally ten sets of blocks with different materials (*i.e.* pine, steel, aluminum and copper) from Amazon, and construct a dataset of real-world block towers. Our dataset contains 16 block towers with different configurations: 8 with two blocks, 4 with three blocks, and another 4 with four blocks.

Physics Interaction. The scenario is set up as follows: the block tower is placed at a specific position on the desk, and we use a copper ball (hang by a pendulum) to hit it. In Figure 7, we show some objects and their trajectories in our dataset.

Video to 3D Trajectory. On real-world data, the appearance of every frame in RGB video is used to extract a 3D trajectory. A major challenge is how to convert RGB videos into 3D trajectories. We employ the following approach:

1. **Tracking 2D Keypoints**. For each frame, we first detect the 2D positions of object corners. For simplicity, we mark the object corners using red stickers and use a simple color filter to determine the corner positions. Then, we find the correspondence between the corner points from consecutive frames by solving the minimum-distance matching between two sets of points. After aligning the corner points in different frames, we obtain the 2D trajectories of these keypoints.
2. **Reconstructing 3D Poses**. We annotate the 3D position for each corner point. Then, for each frame, we have 2D locations of keypoints and their corresponding 3D locations. Finally, we reconstruct the 3D object pose in each frame by solving the Perspective-n-Point between 2D and 3D locations using Levenberg-Marquardt algorithm [24, 28].

Training Details. We build a virtual physics environment, similar to our real-world setup, in the Bullet Physics Engine [9]. We employ it to simulate physics interactions and generate a dataset of synthetic block towers to train our model.

Results. We show some qualitative results of our model with different observations as input in Figure 8. In the real-world setup, with only texture or physics information, our model cannot effectively predict the physical parameters because
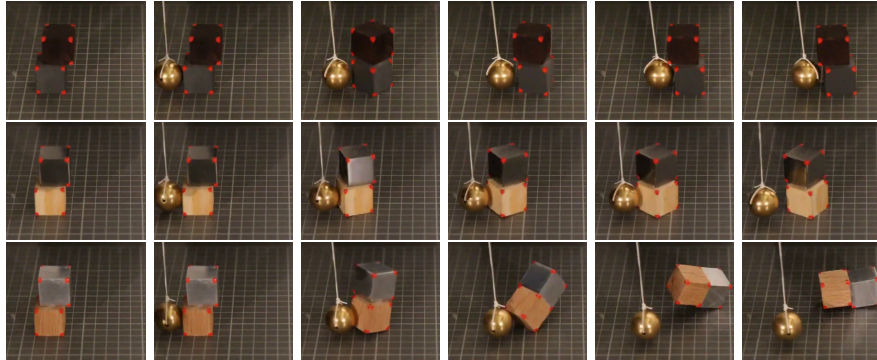
(a) Frame i_1 **(b)** Frame i_2 **(c)** Frame i_3 **(d)** Frame i_4 **(e)** Frame i_5 **(f)** Frame i_6

Fig. 7. Objects and their physics trajectories in six sampled frames from our real-world block towers dataset. As in the last two rows, objects with similar visual appearances may have distinct physical properties that we can only distinguish from their behaviors in physical events.
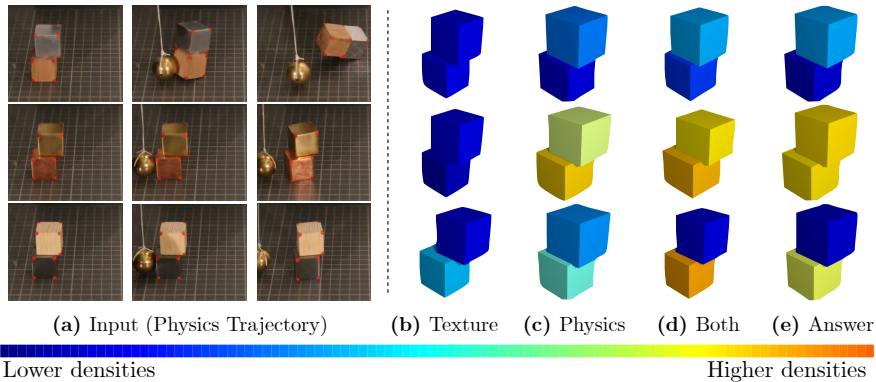

(a) Input (Physics Trajectory)   **(b)** Texture   **(c)** Physics   **(d)** Both   **(e)** Answer

Lower densities   Higher densities

Fig. 8. Qualitative results (on real-world block towers) of our model with different combinations of observations as input.

images and object trajectories are much noisier than those in synthetic dataset, while combining them together indeed helps it to predict much more accurate results. In terms of quantitative evaluation, our model (with both observations as input) achieves an RMSE value of 18.7 over the whole dataset and 10.1 over the block towers with two blocks (the RMSE value of random guessing is 40.8).

6 Analysis

To better understand our model, we present several analysis. The first three are conducted on synthetic block towers and the last one is on our real dataset.

Learning Speed with Different Supervisions. We show the learning curves of our PPD model with different supervision in Figure 9. Model supervised by physics observation reaches the same level of performance of model with texture supervision using much fewer training steps (500K *vs.* 2M). Supervised by both
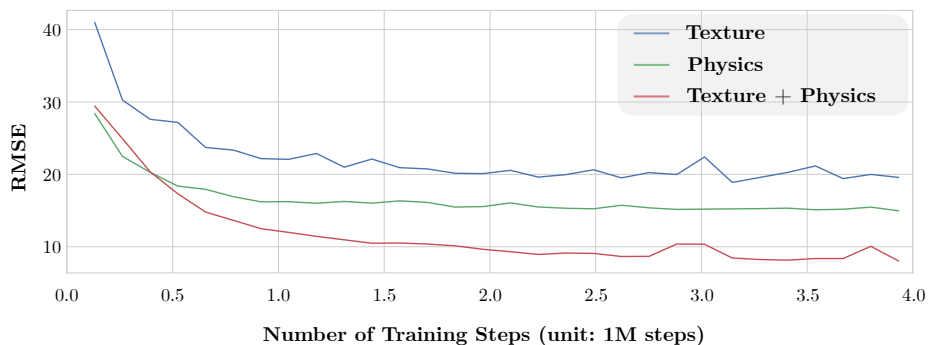
Fig. 9. Learning curves with different observations as input. Our model learns much better and faster when both texture and physics supervisions are available.
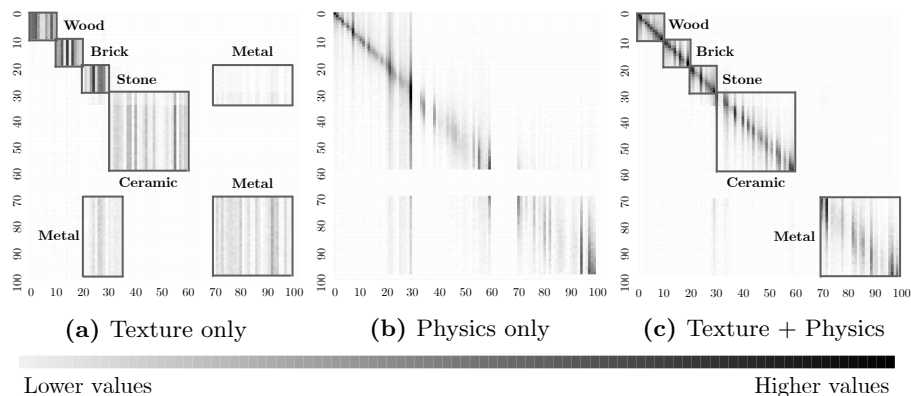

(a) Texture only   **(b)** Physics only   **(c)** Texture + Physics

Lower values   Higher values

Fig. 10. Confusion matrices of physical parameter estimation. The information provided by two types of observations are different: **(a)** with texture as input, our model tends to guess within the material's possible density values (see Table 1); **(b)** with physics as input, our model only makes errors between close values.

observations, our PPD model preserves the learning speed of the model with only physics supervision, and further improves its performance.

Preference over Possible Values. We illustrate the confusion matrices of physical parameter estimation in Figure 10. Although our PPD model performs similarly either with only texture as input or with physics as input, its preferences over all possible values turn out to be quite different. With texture as input (in Figure 10a), it tends to guess within the possible values of the corresponding material (see Table 1), while with physics as input (in Figure 10b), it only makes errors between very close values. Therefore, the information provided by two types of inputs is orthogonal to each other (in Figure 10c).

Impact of Primitive Numbers. As demonstrated in Table 5, the number of blocks has nearly no influence on the model with texture as input. With physics interactions as input, the model performs much better on fewer blocks, and its performance degrades when the number of blocks starts increasing. The degradation is probably because the physical response of any rigid body is fully characterized by a few object properties (*i.e.*, total mass, center of mass, and

Fig. 11. Human's, model's and ground-truth predictions on "which block is heavier". Our model performs comparable to humans, and its response is correlated with humans.

Observation	2 blocks	3 blocks	4 blocks	5 blocks	Overall
Texture	18.2	18.5	18.8	19.7	19.1
Physics	3.6	7.9	15.8	20.0	14.7
Texture + Physics	2.3	4.9	7.8	10.9	8.0

Table 5. Quantitative results (RMSE's) on block towers (with different block numbers): **(a)** with texture as input, our model performs comparably on different block numbers; **(b)** with physics as input, our model performs much better on fewer blocks.

moment of inertia), which provides us with limited constraints on the density distribution of an object when the number of primitives is relatively large.

Human Studies. We select the block towers with two blocks from our real dataset, and study the problem of "which block is heavier" upon them. The human studies are conducted on the Amazon Mechanical Turk. For each block tower, we provide 25 annotators with an image and a video of physics interaction, and ask them to estimate the ratio of mass between the upper and the lower block. Instead of directly predicting a real value, we require the annotators to make a choice on a log scale, *i.e.*, from $\{2^k \mid k = 0, \pm 1, \ldots, \pm 4\}$. Results of average human's predictions, model's predictions and the truths are shown in Figure 11. Our model performs comparably to humans, and its response is also highly correlated with humans: the Pearson's coefficient of "Human *vs*. Model", "Human *vs*. Truth" and "Model *vs*. Truth" is 0.69, 0.71 and 0.90, respectively.

7   Conclusion

In this paper, we have formulated and studied the problem of physical primitive decomposition (PPD), which is to approximate an object with a set of primitives, explaining its geometry and physics. To this end, we proposed a novel formulation that takes both visual and physics observations as input. We evaluated our model on several different setups: synthetic block towers, synthetic tools and real-world objects. Our model achieved good performance on both synthetic and real data.

References

1. Agrawal, P., Nair, A., Abbeel, P., Malik, J., Levine, S.: Learning to poke by poking: Experiential learning of intuitive physics. In: NIPS (2016)
2. Attene, M., Falcidieno, B., Spagnuolo, M.: Hierarchical mesh segmentation based on fitting primitives. The Visual Computer **22**(3), 181–193 (2006)
3. Battaglia, P.W., Hamrick, J.B., Tenenbaum, J.B.: Simulation as an engine of physical scene understanding. PNAS **110**(45), 18327–18332 (2013)
4. Bell, S., Upchurch, P., Snavely, N., Bala, K.: Material recognition in the wild with the materials in context database. In: CVPR (2015)
5. Biederman, I.: Recognition-by-components: a theory of human image understanding. Psychol. Rev. **94**(2), 115 (1987)
6. Binford, T.O.: Visual perception by computer. In: IEEE Conf. on Systems and Control (1971)
7. Brubaker, M.A., Fleet, D.J., Hertzmann, A.: Physics-based person tracking using the anthropomorphic walker. IJCV **87**(1-2), 140 (2010)
8. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al.: Shapenet: An information-rich 3d model repository. arXiv:1512.03012 (2015)
9. Coumans, E.: Bullet physics engine. Open Source Software: http://bulletphysics.org (2010)
10. Grabner, H., Gall, J., Van Gool, L.: What makes a chair a chair? In: CVPR (2011)
11. Gupta, A., Efros, A.A., Hebert, M.: Blocks world revisited: Image understanding using qualitative geometry and mechanics. In: ECCV (2010)
12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2015)
13. van den Hengel, A., Russell, C., Dick, A., Bastian, J., Pooley, D., Fleming, L., Agapito, L.: Part-based modelling of compound scenes from images. In: CVPR (2015)
14. Huang, H., Kalogerakis, E., Marlin, B.: Analysis and synthesis of 3D shape families via deep-learned generative models of surfaces. CGF **34**(5), 25–38 (2015)
15. Huang, Q., Wang, H., Koltun, V.: Single-view reconstruction via joint analysis of image and shape collections. ACM TOG **34**(4), 87 (2015)
16. Igarashi, T., Matsuoka, S., Tanaka, H.: Teddy: a sketching interface for 3d freeform design. In: SIGGRAPH (1999)
17. J. Gibson, J.: The theory of affordances. The Ecological Approach to Visual Perception **8.** (1977)
18. Jia, Z., Gallagher, A., Saxena, A., Chen, T.: 3d reasoning from blocks to stability. IEEE TPAMI **37**(5), 905–918 (2015)
19. Kalogerakis, E., Chaudhuri, S., Koller, D., Koltun, V.: A probabilistic model for component-based shape synthesis. ACM TOG **31**(4), 55 (2012)
20. Kim, M., Pons-Moll, G., Pujades, S., Bang, S., Kim, J., Black, M.J., Lee, S.H.: Data-driven physics for human soft tissue animation. In: SIGGRAPH (2017)
21. Kim, V.G., Li, W., Mitra, N.J., Chaudhuri, S., DiVerdi, S., Funkhouser, T.: Learning part-based templates from large collections of 3d shapes. ACM TOG **32**(4), 70 (2013)
22. Koppula, H.S., Saxena, A.: Physically grounded spatio-temporal object affordances. In: ECCV (2014)
23. Lerer, A., Gross, S., Fergus, R.: Learning physical intuition of block towers by example. In: ICML (2016)

24. Levenberg, K.: A method for the solution of certain non-linear problems in least squares. Quarterly of applied mathematics **2**(2), 164–168 (1944)
25. Li, J., Xu, K., Chaudhuri, S., Yumer, E., Zhang, H., Guibas, L.: Grass: Generative recursive autoencoders for shape structures. In: SIGGRAPH (2017)
26. Li, W., Leonardis, A., Fritz, M.: Visual stability prediction for robotic manipulation. In: ICRA (2017)
27. Li, Y., Wu, X., Chrysathou, Y., Sharf, A., Cohen-Or, D., Mitra, N.J.: Globfit: Consistently fitting primitives by discovering global relations. ACM TOG **30**(4), 52 (2011)
28. Marquardt, D.W.: An algorithm for least-squares estimation of nonlinear parameters. Journal of the society for Industrial and Applied Mathematics **11**(2), 431–441 (1963)
29. Miller, A.T., Knoop, S., Christensen, H.I., Allen, P.K.: Automatic grasp planning using shape primitives. In: ICRA (2003)
30. Mottaghi, R., Rastegari, M., Gupta, A., Farhadi, A.: "what happens if..." learning to predict the effect of forces in images. In: ECCV (2016)
31. Nooruddin, F.S., Turk, G.: Simplification and repair of polygonal models using volumetric techniques. IEEE TVCG **9**(2), 191–205 (2003)
32. Pham, T.H., Kheddar, A., Qammaz, A., Argyros, A.A.: Towards force sensing from vision: Observing hand-object interactions to infer manipulation forces. In: CVPR (2015)
33. Rivlin, E., Dickinson, S.J., Rosenfeld, A.: Recognition by functional parts. CVIU **62**(2), 164–176 (1995)
34. Roberts, L.G.: Machine perception of three-dimensional solids. Ph.D. thesis, Massachusetts Institute of Technology (1963)
35. Rubner, Y., Tomasi, C., Guibas, L.J.: The earth mover's distance as a metric for image retrieval. IJCV **40**(2), 99–121 (2000)
36. Savva, M., Chang, A.X., Hanrahan, P.: Semantically-enriched 3d models for common-sense knowledge. In: CVPR Workshop (2015)
37. Schnabel, R., Degener, P., Klein, R.: Completion and reconstruction with primitive shapes. CGF **28**(2), 503–512 (2009)
38. Soo Park, H., Shi, J., et al.: Force from motion: decoding physical sensation in a first person video. In: CVPR (2016)
39. Tenenbaum, J.B.: Functional parts. In: CogSci (1994)
40. Tulsiani, S., Su, H., Guibas, L.J., Efros, A.A., Malik, J.: Learning shape abstractions by assembling volumetric primitives. In: CVPR (2017)
41. Van Oord, A., Kalchbrenner, N., Kavukcuoglu, K.: Pixel recurrent neural networks. In: ICML (2016)
42. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. MLJ **8**(3-4), 229–256 (1992)
43. Wu, J., Lim, J.J., Zhang, H., Tenenbaum, J.B., Freeman, W.T.: Physics 101: Learning physical object properties from unlabeled videos. In: BMVC (2016)
44. Wu, J., Lu, E., Kohli, P., Freeman, W.T., Tenenbaum, J.B.: Learning to see physics via visual de-animation. In: NIPS (2017)
45. Wu, J., Yildirim, I., Lim, J.J., Freeman, W.T., Tenenbaum, J.B.: Galileo: Perceiving physical object properties by integrating a physics engine with deep learning. In: NIPS (2015)
46. Yao, B., Ma, J., Fei-Fei, L.: Discovering object functionality. In: ICCV (2013)
47. Yumer, M.E., Kara, L.B.: Co-abstraction of shape collections. ACM TOG **31**(6), 166 (2012)
48. Zhao, Y., Zhu, S.C.: Scene parsing by integrating function, geometry and appearance models. In: CVPR (2013)

49. Zheng, D., Luo, V., Wu, J., Tenenbaum, J.B.: Unsupervised learning of latent physical properties using perception-prediction networks. In: UAI (2018)
50. Zheng, Y., Cohen-Or, D., Averkiou, M., Mitra, N.J.: Recurring part arrangements in shape collections. CGF **33**(2), 115–124 (2014)
51. Zhu, Y., Zhao, Y., Zhu, S.C.: Understanding tools: Task-oriented object modeling, learning and recognition. In: CVPR (2015)
52. Zhu, Y., Fathi, A., Fei-Fei, L.: Reasoning about object affordances in a knowledge base representation. In: ECCV (2014)
53. Zou, C., Yumer, E., Yang, J., Ceylan, D., Hoiem, D.: 3d-prnn: Generating shape primitives with recurrent neural networks. In: ICCV (2017)