Learning to Generate 3D Training Data through Hybrid Gradient

Dawei Yang[1,2]
[1]University of Michigan
ydawei@umich.edu

Jia Deng[2]
[2]Princeton University
jiadeng@cs.princeton.edu

Abstract

Synthetic images rendered by graphics engines are a promising source for training deep networks. However, it is challenging to ensure that they can help train a network to perform well on real images, because a graphics-based generation pipeline requires numerous design decisions such as the selection of 3D shapes and the placement of the camera. In this work, we propose a new method that optimizes the generation of 3D training data based on what we call "hybrid gradient". We parametrize the design decisions as a real vector, and combine the approximate gradient and the analytical gradient to obtain the hybrid gradient of the network performance with respect to this vector. We evaluate our approach on the task of estimating surface normal, depth or intrinsic decomposition from a single image. Experiments on standard benchmarks show that our approach can outperform the prior state of the art on optimizing the generation of 3D training data, particularly in terms of computational efficiency.

1. Introduction

Synthetic images rendered by graphics engines have emerged as a promising source of training data for deep networks, especially for vision and robotics tasks that involve perceiving 3D structures from RGB pixels [7, 66, 59, 49, 39, 63, 8, 29, 55, 47, 46, 67, 33]. A major appeal of generating training images from computer graphics is that they have a virtually unlimited supply and come with high-quality 3D ground truth for free.

Despite its great promise, however, using synthetic training images from graphics poses its own challenges. One of them is ensuring that the synthetic training images are useful for real-world tasks, in the sense that they help train a network to perform well on real images. Ensuring this is challenging because a graphics-based generation pipeline requires numerous design decisions, including the selection of 3D shapes, the composition of scene layout, the application of texture, the configuration of lighting, and the placement of the camera. These design decisions can profoundly impact the usefulness of the generated training data, but have largely been made manually by researchers in prior work, potentially leading to suboptimal results.

In this paper, we address the problem of automatically optimizing a generation pipeline of synthetic 3D training data, with the explicit objective of improving the generalization performance of a trained deep network on real images.

One idea is black-box optimization: we try a particular configuration of the pipeline, use the pipeline to generate training images, train a deep network on these images, and evaluate the network on a validation set of real images. We can treat the performance of the trained network as a black-box function of the configuration of the generation pipeline, and apply black-box optimization techniques. Recent works [65, 50] have explored this exact direction. Yang and Deng [65] use genetic algorithms to optimize the 3D shapes used in the generation pipeline. In particular, they start with a collection of simple primitive shapes such as cubes and spheres, and evolve them through mutation and combination into complex shapes, whose fitness is determined by the generalization performance of a trained network. They show that the 3D shapes evolved from scratch can provide more useful training data than manually created 3D CAD models. Meanwhile, Ruiz et al. [50] use black box reinforcement learning algorithms to optimize the parameters of a simulator, and shows that their approaches converge to the optimal solution in controlled experiments and can indeed discover good sets of parameters.

The advantage of black-box optimization is that it assumes nothing about the function being optimized as long as it can be evaluated. As a result, it can be applied to any existing function, including advanced photorealistic renderers. On the other hand, black-box optimization is computationally expensive—knowing nothing else about the function, it needs many trials to find a reasonable update to the current solution. In contrast, gradient-based optimization can be much more efficient by assuming the availability of the analytical gradient, which can be efficiently computed and directly correspond to good updates to the current solution, but the downside is that the analytical gradient is

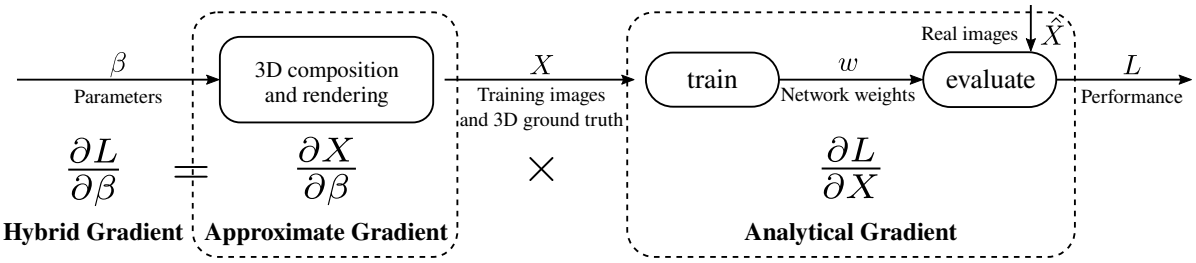

Figure 1. Our hybrid gradient method. We parametrize the design decisions as a real vector β and optimize the function of performance L with respect to β. From β to the generated training images and ground truth, we compute the approximate gradient by averaging finite difference approximations. From training samples X to L, we compute the analytical gradient through backpropagation with unrolled training steps.

often unavailable, especially for many advanced photorealistic renderers.

In this work, we propose a new method that optimizes the generation of 3D training data based on what we call "hybrid gradient". The basic idea is to make use of the analytical gradient where they are available, and combine them with black-box optimization for the rest of the function. We hypothesize that hybrid gradient will lead to more efficient optimization than black-box methods because it makes use of the partially available analytical gradient.

Concretely, if we parametrize the design decisions as a real vector β, the function mapping β to the network performance L can decompose into two parts: (1) from the design parameters β to the generated training images X, and (2) from the training images X to the network performance L. The first part often does not have analytical gradient, due to the use of advanced photorealistic renderers. We instead compute the approximate gradient by averaging finite difference approximations along random directions [36]. For the second part, we compute the analytical gradient through backpropagation—with SGD training unrolled, the performance of the network is a differentiable function of the training images. Then we combine the approximate gradient and the analytical gradient to obtain the hybrid gradient of the network performance L with respect to the parameters β, as illustrated in Fig. 1.

A key ingredient of our approach is representing design decisions as real vectors of fixed dimensions, including the selection and composition of shapes. Yang and Deng [65] represent 3D shapes as a finite set of graphs, one for each shape. This representation is suitable for a genetic algorithm but is incompatible with our method. Instead, we propose to represent 3D shapes as random samples generated by a Probabilistic Context-Free Grammar (PCFG) [21]. To sample a 3D shape, we start with an initial shape, and repeatedly sample a production rule in the grammar to modify it. The (conditional) probabilities of applying the production rules are parametrized as a real vector of a fixed dimension.

Our approach is novel in multiple aspects. First, to the best of our knowledge, we are the first to propose the idea of hybrid gradient, i.e. combining approximate gradient and analytical gradient, especially in the context of optimizing the generation of 3D training data. Second, we propose a novel integration of PCFG-based shape generation and our hybrid gradient approach.

We evaluate our approach on the task of estimating surface normal, depth and intrinsic components from a single image. Experiments on standard benchmarks and controlled settings show that our approach can outperform the prior state of the art on optimizing the generation of 3D training data, particularly in terms of computational efficiency.

2. Related Work

Generating 3D training data Synthetic images generated by computer graphics have been extensively used for training deep networks for numerous tasks, including single image 3D reconstruction [54, 22, 39, 23, 65, 9], optical flow estimation [38, 7, 19], human pose estimation [59, 11], action recognition [48], visual question answering [25], and many others [45, 37, 63, 58, 46, 47, 62]. The success of these works has demonstrated the effectiveness of synthetic images.

To ensure the relevance of the generated training data to real-world tasks, a large amount of manual effort has been necessary, particularly in acquiring 3D assets such as shapes and scenes [9, 23, 13, 64, 22, 39, 55]. To reduce manual labor, some heuristics have been proposed to generate 3D configurations automatically. For example, Zhang et al. [67] design an approach to use the entropy of object masks and color distribution of the rendered images to select sampled camera poses. McCormac et al. [39] simulate gravity for physically plausible object configurations inside a room.

Apart from simple heuristics, prior work has also performed automatic optimization of 3D configurations towards an explicit objective. For example, Yeh et al. [66] synthesize layouts with the target of satisfying constraints such as non-overlapping and occupation. Jiang

et al. [24] learn a probabilistic grammar model for indoor scene generation, with parameters learned using maximum likelihood estimation on the existing 3D configurations in SUNCG [55]. Similarly, Veeravasarapu et al. [60] tune the parameters for stochastic scene generation using generative adversarial networks, targeting at making synthetic images indistinguishable from real images. Qi et al. [44] synthesize 3D room layouts based on human-centric relations among furniture, to achieve visual realism, functionality and naturalness of the scenes. However, these optimization objectives are different from ours, which is the generalization performance of a trained network on real images.

In terms of generating 3D training data, the closest prior works to ours are those of [65, 26, 50]. Specifically, Yang and Deng [65] use a genetic algorithm to optimize the 3D shapes used for rendering synthetic training images. Their optimization objective is the same as ours except that their optimization method is different: they leverage evolution-based approach as apposed to using gradient information. Similarly, Meta-Sim [26] also tries to optimize 3D parameters with REINFORCE towards better task generalization performance, and Ruiz et al. [50] learn a policy for simulator parameters also using REINFORCE. However, they do not backpropagate analytical gradient from the meta-objective, so their algorithms can be considered as black-box estimation by multiple trials, with an improved efficient sampling strategy (REINFORCE). In our experiments, we compared to an algorithm that has been shown competitive to REINFORCE in training deep policy networks [36, 51, 56].

Unrolling and backpropagating through network training One component of our approach is unrolling and backpropagating through the training iterations of a deep network. This is a technique that has often been used by existing work in other contexts, including hyperparameter optimization [35], meta-learning [1, 20, 40, 31, 16] and others [68, 12]. Our work is different in that we apply this technique in a novel context: it is used to optimize the generation of 3D training data, and the gradient with respect to the input images is integrated with approximate gradient to form hybrid gradient.

Hyperparameter optimization Our method is connected to hyperparameter optimization in the sense that we can treat the design decisions of the 3D generation pipeline as hyperparameters of the training procedure.

Hyperparameter optimization of deep networks is typically approached as black-box optimization [5, 4, 30, 6]. While Klatzer and Pock [28] propose a bi-level gradient-based approach for continuous hyperparameter optimization of Support Vector Machines, but it has not been applied to deep networks and 3D generation. Since black-box optimization does not make assumption of the function being optimized, it requires repeated evaluation of the function, which is expensive in this case because it contains the process of training and evaluating a deep network. In contrast, we combine the analytical gradient from backpropagation and the approximate gradient from generalized finite difference for more efficient optimization.

Domain Adaptation Researchers have also applied domain adaptation techniques to transfer the knowledge learned from synthetic data to real data. Like domain adaptation, our method involves data from two domains: synthetic and real. However, our setting is different: in domain adaptation, the distribution of training data is fixed; in our setting, we are concerned about *generating and changing* the distribution of training data in the source domain.

Differentiable Rendering Researchers have also explored differentiable rendering engines to obtain the gradient with respect to the input 3D content such as mesh vertices, lighting intensity *etc*. [34, 27, 61, 32, 10]. Generally, they obtain the gradient through backpropagation [27, 34] or sampling [61, 32, 10]. The differentiable renderers often assume simple surface reflectance and illumination model, and they are typically developed for a specific 3D input format (such as triangle meshes and directional lighting) or a specific rendering algorithm (such as path tracing). In fact, we are not aware of any photorealistic differentiable renderer that is differentiable over a shape parametrization that allows not only continuous deformation but also topology change. In our method, we assume nothing about the rendering engine and obtain the gradient with respect to the decision vector by approximation, bypassing the surface and illumination model or any rendering algorithms. So our method is flexible and not limited by choices of graphics engines of any kind.

3. Problem Setup

Suppose we have a probabilistic generative pipeline. We use a deterministic function, $f(\beta, r)$ to represent the sampling operation. This function f takes the real vector β and the random seed r as input. An image and its 3D ground truth are computed by evaluating the function $f(\beta, r)$. By choosing n different random seeds r, we obtain a dataset of size n for training:

$$X = (f(\beta, r^{(1)}), f(\beta, r^{(2)}), \cdots, f(\beta, r^{(n)})) \quad (1)$$

Then, a deep neural network with initialized weights w_0 is trained on the training data X, with the function $\mathrm{train}(w_0, X)$ representing the optimization process and generating the weights of the trained network.

The network is then evaluated on real data $\hat{X}$ with a validation loss l_{eval} to obtain a generalization performance L:

$$L = l_{\mathrm{eval}}(\mathrm{train}(w_0, X), \hat{X}) \quad (2)$$

Combining the above two functions, L is a function of β, and the task is to optimize this value L with respect to the parameters β.

As we mentioned in the previous section, black-box algorithms typically need repetitive evaluations of this function, which is expensive.

4. Approach

4.1. Generative Modeling of Synthetic Training Data

We decompose the function $f(\beta, r)$ into two parts: 3D composition and rendering.

3D composition Context-free grammars have been used in scene generation [24, 44] and in the parsing of the Constructive Solid Geometry (CSG) shapes [52] because they can represent shapes and scenes in a flexible and composable manner. Here, we design a probabilistic context-free grammar (PCFG) [21] to control the random generation of unlimited shapes [18].

In a PCFG, a tree is randomly sampled given a set of probabilities. Starting from a root node, the nodes are expanded by randomly sampling probabilistic rules repeatedly until all the leaf nodes cannot expand. Since multiple rules may apply, the parameters in a PCFG define the probability distribution of applying different rules.

In our PCFG, a shape is constructed by composing two other shapes through union and difference; this construction is recursively applied until all leaf nodes are a predefined set of concrete primitive shapes (terminals). The parameters include the parameters of primitive shapes as well as the probability of either expanding the node or replacing it with a terminal.

Given our PCFG model with the probability parameters β_S, a 3D shape S can be composed by computing a deterministic function f_S given β_S and a random string r_S as the input:

$$S = f_S(\beta_S, r_S) \tag{3}$$

Rendering training images we use a graphics renderer R to render the composed shape S. The rendering configurations P (e.g. camera poses), are also sampled from a distribution controlled by a set of parameters β_R (with a random string r_R):

$$P = f_{\mathcal{R}}(\beta_R, r_R) \tag{4}$$

Now that we have Eq. 3 and 4, The full function for training data generation can be represented as follows:

$$f(\beta, r) = R(S, P) = R(f_S(\beta_S, r_S), f_{\mathcal{R}}(\beta_R, r_R)) \tag{5}$$

where $\beta = (\beta_R, \beta_S)$ and $r = (r_R, r_S)$.

By sampling different random strings r, we obtain a set of training images and their 3D ground truth X.

4.2. Hybrid Gradient

After training deep network on synthetic training data X, the network is evaluated on a set of validation images $\hat{X}$ to obtain the generalization loss L.

Recall that to compute the hybrid gradient $\frac{\partial L}{\partial \beta}$ to optimize β, we multiply two types of gradient: the gradient of network training $\frac{\partial L_t}{\partial X}$ and the gradient of image generation $\frac{\partial X}{\partial \beta}$, as is shown in Fig. 2.

Analytical gradient from backpropagation We assume the network is trained on a set of previously generated training images $X^{(1)}, X^{(2)}, \cdots, X^{(n)}$. Without loss of generality, we assume mini-batch stochastic gradient descent (SGD) with a batch size of 1 is used for weight update. Let function g denote the SGD step and let l_{train} denote the training loss:

$$\begin{aligned} w^{(k+1)} &= w^{(k)} - \eta \frac{\partial l_{\text{train}}(w^{(k)}, X^{(k)})}{\partial w^{(k)}} \\ &= g(w^{(k)}, X^{(k)}; l_{\text{train}}, \eta) \end{aligned} \tag{6}$$

Note that the SGD step g is differentiable with respect to the network weights $w^{(k)}$ as well as the training batch $X^{(k)}$, if our training loss l_{train} is twice (sub-)differentiable. This requirement is satisfied in most practical cases. To simplify the equation, we assume the training loss l_{train} and the learning rate η do not change during one update step of β, so the variables can be safely discarded in the equation.

Therefore, the gradient from the generalization loss L to each sample $X^{(k)}$ can be computed through backpropagation. Given Eq. 6:

$$\begin{aligned} \frac{\partial L}{\partial X^{(k)}} &= \frac{\partial L}{\partial w^{(k+1)}} \cdot \frac{\partial w^{(k+1)}}{\partial X^{(k)}} = \frac{\partial L}{\partial w^{(k+1)}} \cdot g_2'(w^{(k)}, X^{(k)}) \\ \frac{\partial L}{\partial w^{(k)}} &= \frac{\partial L}{\partial w^{(k+1)}} \cdot \frac{\partial w^{(k+1)}}{\partial w^{(k)}} = \frac{\partial L}{\partial w^{(k+1)}} \cdot g_1'(w^{(k)}, X^{(k)}) \end{aligned} \tag{7}$$

with the initial value $\frac{\partial L}{\partial w^{(n+1)}}$ computed from the validation loss l_{eval}:

$$\frac{\partial L}{\partial w^{(n+1)}} = l_{\text{eval}}'(w^{(k+1)}, \hat{X}) \tag{8}$$

Approximate gradient from finite difference For the formulation in Eq. 5, the graphics renderer can be a general black box and non-differentiable. We can approximate the gradient of each rendered image with ground truth $X^{(1)}, X^{(2)}, \cdots$ with respect to the generation parameters β using generalized finite difference. We adopt the form of [36] because this gradient approximation algorithm in Random Search has been shown effective for training deep policy networks [36, 51, 56]. Concretely, we sample a set of noise from an uncorrelated multivariate Gaussian distribution:

$$\delta_1, \delta_2, \cdots, \delta_m \sim \mathcal{N}(\mathbf{0}, \sigma I) \tag{9}$$

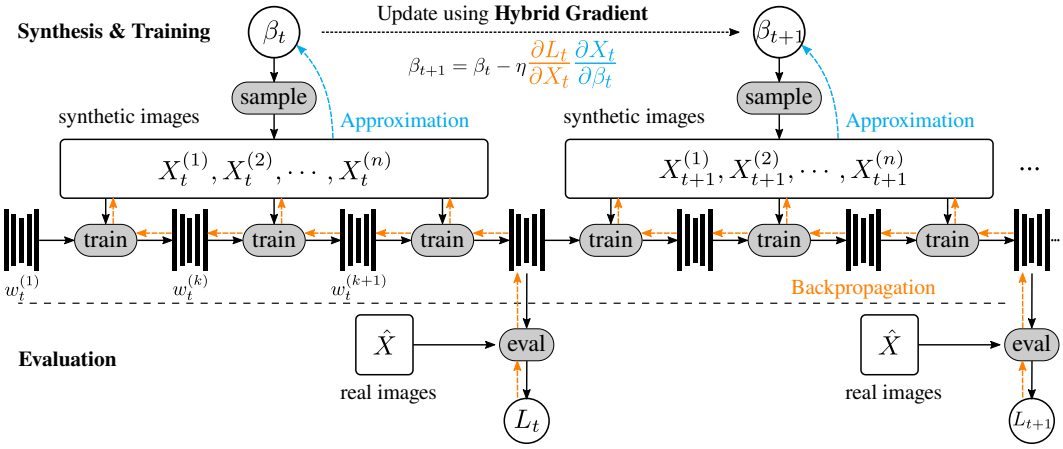

Figure 2. The details of using "hybrid gradient" to incrementally update β and train the network. The analytical gradient is computed by backpropagating through unrolled training steps (colored in orange). The numerical gradient is computed using finite difference approximation by sampling in a neighborhood of β_t (colored in cyan). Then β_t is updated using hybrid gradient, and the trained network weights are retained for the next timestamp $t+1$.

Next, we approximate the Jacobian for each sample ($\otimes$ denotes cross product):

$$\frac{\partial X^{(i)}}{\partial \beta} \approx \frac{1}{m} \sum_{j=1}^{m} \frac{f_{\mathcal{D}}(\beta + \delta_j, r_i) - f_{\mathcal{D}}(\beta - \delta_j, r_i)}{2\|\delta_j\|} \otimes \frac{\delta_j}{\|\delta_j\|} \tag{10}$$

Incremental training Following Yang and Deng [65], we incrementally train the network w along with the update of β, instead of initializing $w^{(1)}$ from scratch each time. At timestamp t, we update β_t with the hybrid gradient; for network weights, we keep the trained network in timestamp t for initialization in the next timestamp $t+1$:

$$\begin{aligned} \beta_{t+1} &= \beta_t - \gamma \frac{\partial L_t}{\partial \beta_t} = \beta_t - \gamma \sum_{i=1}^{n} \frac{\partial L_t}{\partial X_t^{(i)}} \cdot \frac{\partial X_t^{(i)}}{\partial \beta_t} \\ w_{t+1}^{(1)} &= w_t^{(n+1)} \end{aligned} \tag{11}$$

5. Experiments

Datasets We evaluate our algorithm on four different datasets, and three standard prediction tasks for single-image 3D. The input is an RGB image and the output is pixel-wise surface normal, depth, or albedo shading map.

Specifically, we experiment on the task of surface normal estimation on two real datasets: MIT-Berkeley Intrinsic Images Dataset (MBII) [3], which focuses on images of single objects and NYU Depth [53], which focuses on indoor scenes. For the other two datasets, we illustrate that our method can easily extend to other 3D setups. We experiment on the task of depth estimation on the renderings of the scanned human faces in the Basel Face Model dataset [43], and on the task of intrinsic image decomposition and evaluate on the renderings of ShapeNet [9] shapes.

Baselines For comparison, we implemented a black-box optimization method. Random search [2] has been extensively explored [17, 41, 36] as a derivative-free optimization method, and Mania et al. [36] have shown that their simple version, Basic Random Search, has comparable performance compared to typical reinforcement learning algorithms. Therefore, we re-implemented their Basic Random Search such that this baseline has the same setting as in our method, while the only difference is that the gradient from the validation loss is obtained through sampling instead of hybrid gradient. We also compare against baselines with a random β baseline in the following experiments. In these baselines, the networks are trained on a dataset generated using multiple random but fixed β, and the weight snapshots with the best validation performance are used to evaluate on the test set.

These two baselines, along with our hybrid gradient method, all use information from the validation set but in a different way: hybrid gradient backpropagates the gradient of the validation performance to update β; random search samples β to get the gradient from the validation performance; the random β baseline fixes the dataset and uses the validation performance to select the best network snapshot.

In all of our experiments, the network weights are updated using only synthetic images in the training iterations, and the generalization loss is computed only on the validation split of the datasets mentioned above. The decision vector β is updated using RMSprop [57] for hybrid gradient.

For MBII, we use pure synthetic shapes [65] to render training images. We first compare our method with ablation baselines, then show that our algorithm is better than the previous state of the art on MBII. For NYU Depth, we

base our generative model on SUNCG [55] and augment the original 3D configurations in Zhang et al. [67]. For Basel Face Model, we sample synthetic faces from a morphable model and evaluate on the renderings of scanned faces. For the intrinsic image decomposition task, we sample textures from a simple procedural pipeline and attach the synthetic textures to SUNCG shapes [55], and evaluate on renderings of ShapeNet shapes [9].

5.1. Normal Estimation on MIT-Berkeley Intrinsic Images

Following the work of Yang and Deng [65], we recover the surface normals of an object from a single image.

Synthetic shape generation In Yang and Deng [65], a population of primitive shapes such as cylinders, spheres and cubes are evolved and rendered to train deep networks. The evolution operators include transformations of individual shapes and the boolean operations of shapes in Constructive Solid Geometry (CSG) [18]. In our algorithm, we also use the CSG grammar for our PCFG:

```
S => E
E => C(E, T(E)) | P
C => union | subtract
P => sphere | cube | truncated_cone | tetrahedron
T => attach * rand_transl * rand_rotate * rand_scale
```

In this PCFG, the final shape `S` is generated by recursively composing (`C`) other shapes `E` with transformations `T`, until primitives `P` are sampled at all `E` nodes. The parameter vector β consists of three parts: (1) The probability of the different rules; (2) The means and variations of log-normal distributions controlling shape primitives (`P`), such as the radius of the sphere; (3) The means and variations of log-normal distributions controlling transformation parameters (`T`), such as scale values. Examples of sampled shapes are shown in Fig. 3. For the generalization loss L, we compute the mean angle error of predictions on the training set of the MIT-Berkeley dataset.

Training setup For network training and evaluation, we follow Yang and Deng [65] and train the Stacked Hourglass Network [42] on the images, and use the standard split of the MBII dataset for the optimization of β and testing.

We report the performance of surface normal directions with the metrics commonly used in previous works, including mean angle error (MAE), median angle error, mean squared error (MSE), and the proportion of pixels that normals fall in an error range ($\leq N^\circ$). See supplementary material for detailed definitions.

Figure 3. Sampled shapes from our probabilistic context-free grammar, with parameters optimized using hybrid gradient.

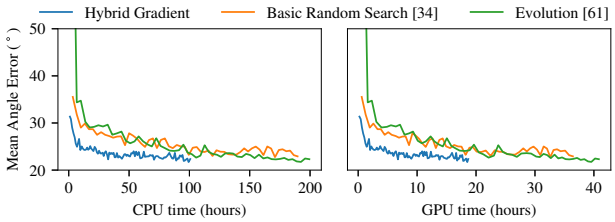

Figure 4. Mean angle error on the test images *vs*. computation time, compared to two black-box optimization baselines.

Ablation study We first sample 10 random values of β and fix those values in advance. Then, for each β, we sample 3D shapes and render images to train a network, with the same training and evaluation configurations as in our hybrid gradient, except that we do not update β. We then report the best, median and worst performance of those 10 networks, and label the corresponding β as β_{best}, β_{median} and β_{worst}. In hybrid gradient, we then initialize β_0 from these three values, run our algorithm, and report the performance on test images also in Table 1.

From the table we can observe that training with a fixed β can hardly match the performance of our method, even with multiple trials. Instead, our hybrid gradient approach can optimize β to a reasonable performance regardless of different initialization ($\beta_{\text{best/median/worst}}$). This simple diagnostic experiment demonstrates that our algorithm is working correctly: the optimization of β is necessary in order to generate useful synthetic images for training networks.

Comparison with the state of the art In addition to Basic Random Search as mentioned earlier, in this experiment we also compare with Yang and Deng [65], a state-of-the-art method on MIT-Berkeley Intrinsic Images.

In Shape Evolution [65], a population of shapes are evolved, and fitness scores for individual shapes are computed using a network trained on an incremental dataset and evaluated on the validation set. We compose our shapes in mesh representations, slightly different from the implicit functions in Yang and Deng [65]. Therefore, we re-implemented their algorithm with mesh representations for a fair comparison. We follow Yang and Deng [65] for the initialization of β, and train the networks and update β for the same number of steps. We then report the test performance of the network that has the best validation performance. The results are shown in Table 2.

We also run the experiments on the same set of CPUs and GPUs, sum the computation time, and plot the mean angle error (on the test set) with respect to the CPU time and GPU time Fig. 4). We see that our algorithm is more efficient than the above baselines. This is natural, because when computing $\partial L/\partial \beta^{(t)}$ in black-box algorithms, for each sample of $\beta^{(t)} + \delta_j$, one needs to train one network to evaluate the per-

		Summary Stats ↑			Errors ↓		
		$\leq 11.25°$	$\leq 22.5°$	$\leq 30°$	MAE	Median	MSE
Fixed β	$\beta = \beta_{\text{best}}$	19.9%	52.7%	70.5%	24.0°	21.5°	0.2282
	$\beta = \beta_{\text{median}}$	20.7%	50.9%	67.5%	24.8°	22.1°	0.2461
	$\beta = \beta_{\text{worst}}$	17.9%	46.7%	64.6%	25.6°	23.8°	0.2553
Hybrid gradient	$\beta_0 = \beta_{\text{best}}$	22.7%	58.5%	73.9%	22.5°	19.3°	0.2065
	$\beta_0 = \beta_{\text{median}}$	24.0%	60.1%	75.7%	21.8°	18.8°	0.1938
	$\beta_0 = \beta_{\text{worst}}$	26.0%	58.6%	73.9%	22.0°	19.1°	0.1998

Table 1. Ablation Study: the diagnostic experiment to compare with random but fixed β. We sample 10 values of β in advance, and then train the networks with the same setting as in hybrid gradient. The best, median and worst performance is reported on the test images, and the corresponding values of β are used to initialize β_0 for hybrid gradient for comparison. The results show that our approach is consistently better than the baselines with fixed β.

	Summary Stats ↑			Errors ↓		
	$\leq 11.25°$	$\leq 22.5°$	$\leq 30°$	MAE	Median	MSE
SIRFS [3]	20.4%	53.3%	70.9%	26.2°	—	0.2964
Evolution [65](Reported)	21.6%	55.5%	73.5%	23.3°	—	0.2204
Evolution [65](Our Impl.)	23.0%	58.3%	73.8%	22.5°	**18.8°**	0.2042
Basic Random Search [36]	21.9%	**59.6%**	74.0%	22.8°	19.2°	0.2106
Hybrid gradient	**24.5%**	59.3%	**74.3%**	**22.0°**	18.9°	**0.1984**

Table 2. Our approach compared to previous work, on the test set of MIT-Berkeley images [3]. The results show that our approach is better than the state of the art as reported in [65].

formance L, while in hybrid gradient, only a forward training pass and a backpropagation pass for a single network are required to compute $\partial L / \partial X$. Shapes sampled from our optimized PCFG are shown in Fig. 3.

5.2. Normal Estimation on NYU Depth

Scene perturbation We design our scene generation grammar as an augmentation of collected SUNCG scenes [54] with the cameras from Zhang et al. [67]:

```
S => E,P
E => T_shapes * R_shapes * E0
P => T_camera * R_camera * P0
T_shapes => translate(x, y, z)
R_shapes => rotate(yaw, pitch, roll)
```

For each 3D scene `S`, we perturb the positions and poses of the original cameras (`P0`) and shapes (`E0`) using random translations and rotations. The position perturbations follow a mixture of uncorrelated Gaussians, and the perturbations for pose angles (yaw, pitch & roll) follow a mixture of von Mises, *i.e.* wrapped Gaussians. The vector β consists of the parameters of the above distributions.

Training setup Our networks are trained on synthetic images only, and evaluated on NYU Depth V2 [53] with the same setup as in Zhang et al. [67]. For real images in our optimization pipeline, we sample a subset of images from the standard validation images in NYU Depth V2. We initialize our network from the synthetically trained model in Zhang et al. [67] and initialize β_0 using a small value. To compare with random β, we construct a dataset of 40k images with a small random β for each image. We then load the same pre-trained network and train for the same number of iterations as in hybrid gradient. We then evaluate the networks on the test set of NYU Depth V2 [53], following the same protocol. The results are reported in Table 3. Note that none of these networks has been trained on real images except for validation, and the validation subset of real images is only used to update the decision vector.

The numbers indicate that our parametrized generation of SUNCG augmentation exceeds the original baseline performance. Note that the network trained with random β is worse than original performance. This means without proper optimization of perturbation parameters, such random augmentation may hurt generalization, demonstrating that good choices of these parameters are crucial for generalization to real images.

	Summary Stats ↑			Errors ↓	
	$\leq 11.25°$	$\leq 22.5°$	$\leq 30°$	Mean	Median
Original [67]	24.1%	49.7%	61.5%	28.8°	22.7°
Random β + [67]	23.0%	48.8%	61.3%	29.2°	23.2°
Hybrid gradient + [67]	**27.3%**	**52.5%**	**63.8%**	**28.1°**	**21.1°**

Table 3. The performance of the finetuned networks on the test set of NYU Depth V2 [53], compared to the original network in [67]. The networks are trained only on the synthetic images. Without optimizing the parameters (random β), the augmentation hurts the generalization performance. With proper search of β using hybrid gradient, we are able to achieve better performance than the original model.

5.3. Depth Estimation on Basel Face Model

Synthetic face generation We exploit an off-the-shelf 3DMM morphable face and expression model [14, 70, 69] to generating human 3D models, with face and pose parameters randomly sampled from mixtures of Gaussians or von Mises. Since the parameters for 3DMM are PCA coefficients, we only include the first 10 principal dimensions each for geometry, texture and expression parameters in the decision vector β, and uniformly sample for the remaining dimensions to save disk usage.

Training setup We train a stacked hourglass network [42] from scratch with a single-channel output after a ReLU layer to predict the raw depth, and supervise using mean squared error. The learning rate for the network is 0.1 and the batch size is 8.

Evaluation We evaluate on the renderings of the scanned

human faces [43]. We split the 10 identities into two disjoint sets for validation and test, then use the rendering parameters provided in the dataset to recreate the renderings as well as depth images. For each scan, there are 3 lighting directions and 9 pose angles, creating 135 validation images and 135 test images. Example images are shown in Fig. 5. For depth evaluation, we use the standard metrics including the relative difference (absolute and squared) and root mean squared error (linear, log and scale-invariant log). The definitions are listed in Eigen et al. [15] and also detailed in the supplementary material.

Synthetic faces generated from the PCFG | Re-renderings of real scans

Figure 5. Training images generated using PCFG with 3DMM face model, and example test images.

	Relative Difference		RMSE		
	abs	sqr	linear	log	scale inv.
Random β	0.03718	9.701×10^{-3}	0.1395	0.1014	0.09717
Basic Random Search [36]	0.02330	1.728×10^{-3}	0.0581	0.0299	0.02700
Hybrid gradient	**0.02256**	$\mathbf{1.649 \times 10^{-3}}$	**0.0570**	**0.0293**	**0.02603**

Table 4. The results on the scanned faces of the Basel Face Model. Our method is able to search for the synthetic face parameters such that the trained network can generalize better.

The results in 4 show that our algorithm is able to search for better β so that the network trained on the synthetic faces and generalize better on the scanned faces.

5.4. Intrinsic Image Decomposition on ShapeNet

Texture generation and rendering We design a painter's algorithm as PCFG for generating the textures. To generate one texture image, we paint Perlin-noise-perturbed polygons sequentially onto a canvas, and then repeat the canvas as the final texture image. The number of repetitions and the number of polygons follow zero-truncated Poisson distributions, the vertex coordinates follow independent truncated Gaussian mixtures, and the number of edges in a polygon are also controlled by sampling probabilities. All the distribution parameters are concatenated to form the decision vector β. Example textures are shown in Fig. 6.

The texture is then mapped onto the SUNCG shapes [55]. We choose SUNCG shapes because they are well parametrized for texture mapping and we can easily apply our synthetic textures. We then render the textured shapes using random directional lights as training data. For validation and testing, we randomly render ShapeNet [9] shapes with their original textures, and randomly choose 50 as validation and 50 for test. The shapes used in validation or test are mutually exclusive.

Figure 6. Example textures generated using our procedural pipeline with parameters controlled by β.

Training We use the Stacked Hourglass Network [42] with a 4-channel output (3 for albedo, 1 for shading), and train with a learning rate of 10^{-4} and a batch size of 8. For supervision, we sum the mean squared error for both albedo and shading outputs as the loss.

Evaluation We also compare with our Basic Random Search implementation and with the random β baseline. We evaluate the performance using mean absolute error (abs), root mean squared error (rmse) and scale-invariant rmse for albedo and shading. We also evaluate the reconstruction error of the rendered image, even though we do not have any supervision for the reconstruction error of the image. The results are shown in Table 5.

		abs	rmse	rmse (scale inv.)
Random β	Albedo	0.157	0.198	0.175
	Shading	0.118	0.132	0.095
	Reconstruction	0.139	0.169	–
Basic Random Search [36]	Albedo	0.152	0.193	0.177
	Shading	**0.104**	**0.116**	**0.085**
	Reconstruction	0.134	0.166	–
Hybrid gradient	Albedo	**0.147**	**0.189**	**0.168**
	Shading	**0.104**	0.119	0.088
	Reconstruction	**0.118**	**0.150**	–

Table 5. The results of intrinsic image decomposition on the ShapeNet renderings.

6. Conclusion

In this paper, we have proposed hybrid gradient, a novel approach to the problem of automatically optimizing a generation pipeline of synthetic 3D training data. We evaluate our approach on the task of estimating surface normal, depth and intrinsic decomposition from a single image. Our experiments show that our algorithm can outperform the prior state of the art on optimizing the generation of 3D training data, particularly in terms of computational efficiency.

Acknowledgments This work is partially supported by the National Science Foundation under Grant No. 1617767.

References

[1] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, pages 3981–3989, 2016). 3

[2] N. Baba. Convergence of a random optimization method for constrained optimization problems. *Journal of Optimization Theory and Applications*, 33(4):451–461, Apr 1981). 5

[3] Jonathan T Barron and Jitendra Malik. Shape, illumination, and reflectance from shading. *TPAMI*, 2015). 5, 7

[4] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, NIPS'11, pages 2546–2554. Curran Associates Inc., 2011). 3

[5] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13(1):281–305, Feb. 2012). 3

[6] Eric Brochu, Vlad M. Cora, and Nando de Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *CoRR*, abs/1012.2599, 2010). 3

[7] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In A. Fitzgibbon et al. (Eds.), editor, *European Conf. on Computer Vision (ECCV)*, Part IV, LNCS 7577, pages 611–625. Springer-Verlag, Oct. 2012). 1, 2

[8] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017). 1

[9] 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). 2, 5, 6, 8

[10] Chengqian Che, Fujun Luan, Shuang Zhao, Kavita Bala, and Ioannis Gkioulekas. Inverse transport networks. *arXiv preprint arXiv:1809.10820*, 2018). 3

[11] Wenzheng Chen, Huan Wang, Yangyan Li, Hao Su, Zhenhua Wang, Changhe Tu, Dani Lischinski, Daniel Cohen-Or, and Baoquan Chen. Synthesizing training images for boosting human 3d pose estimation. In *3D Vision (3DV)*, 2016). 2

[12] Yunjin Chen, Wei Yu, and Thomas Pock. On learning optimized reaction diffusion processes for effective image restoration. In *ICCV*, 2015). 3

[13] Sungjoon Choi, Qian-Yi Zhou, Stephen Miller, and Vladlen Koltun. A large dataset of object scans. *arXiv:1602.02481*, 2016). 2

[14] Hang Dai, Nick Pears, William A. P. Smith, and Christian Duncan. A 3d morphable model of craniofacial shape and texture variation. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017). 7

[15] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2366–2374. Curran Associates, Inc., 2014). 8

[16] Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 9516–9527. Curran Associates, Inc., 2018). 3

[17] Abraham D. Flaxman, Adam Tauman Kalai, Adam Tauman Kalai, and H. Brendan McMahan. Online convex optimization in the bandit setting: Gradient descent without a gradient. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '05, pages 385–394, Philadelphia, PA, USA, 2005). Society for Industrial and Applied Mathematics. 5

[18] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics: Principles and Practice (2Nd Ed.)*. Addison-Wesley Longman Publishing Co., Inc., 1990). 4, 6

[19] Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. Virtual worlds as proxy for multi-object tracking analysis. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016). 2

[20] David Ha, Andrew Dai, and Quoc Le. Hypernetworks. In *ICLR*, 2017). 3

[21] M. A. Harrison. *Introduction to Formal Language Theory*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1978). 2, 4

[22] Binh-Son Hua, Quang-Hieu Pham, Duc Thanh Nguyen, Minh-Khoi Tran, Lap-Fai Yu, and Sai-Kit Yeung. Scenenn: A scene meshes dataset with annotations. In *International Conference on 3D Vision (3DV)*, 2016). 2

[23] A. Janoch, S. Karayev, , J. T. Barron, M. Fritz, K. Saenko, and T. Darrell. A category-level 3-d object dataset: Putting the kinect to work. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 1168–1174, Nov 2011). 2

[24] Chenfanfu Jiang, Siyuan Qi, Yixin Zhu, Siyuan Huang, Jenny Lin, Lap-Fai Yu, Demetri Terzopoulos, and Song-Chun Zhu. Configurable 3d scene synthesis and 2d image rendering with per-pixel ground truth using stochastic grammars. *International Journal of Computer Vision*, 126(9):920–941, 2018). 3, 4

[25] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017). 2

[26] Amlan Kar, Aayush Prakash, Ming-Yu Liu, Eric Cameracci, Justin Yuan, Matt Rusiniak, David Acuna, Antonio Torralba, and Sanja Fidler. Meta-sim: Learning to generate synthetic datasets. In *ICCV*, 2019). 3

[27] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018). 3

[28] Teresa Klatzer and Thomas Pock. Continuous hyper-parameter learning for support vector machines. In *CVWW*, 2015). 3

[29] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. Ai2-thor: An interactive 3d environment for visual ai, 2017. 1

[30] Alexandre Lacoste, Hugo Larochelle, Mario Marchand, and François Laviolette. Sequential model-based ensemble optimization. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*, UAI'14, pages 440–448. AUAI Press, 2914). 3

[31] Ke Li and Jitendra Malik. Learning to optimize. In *ICLR*, 2017). 3

[32] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. Differentiable monte carlo ray tracing through edge sampling. *ACM Trans. Graph.*, 37(6):222:1–222:11, Dec. 2018). 3

[33] Zhengqi Li and Noah Snavely. Cgintrinsics: Better intrinsic image decomposition through physically-based rendering. In *The European Conference on Computer Vision (ECCV)*, September 2018). 1

[34] Matthew M. Loper and Michael J. Black. Opendr: An approximate differentiable renderer. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 154–169, Cham, 2014). Springer International Publishing. 3

[35] Dougal Maclaurin, David Duvenaud, and Ryan Adams. Gradient-based hyperparameter optimization through reversible learning. In *International Conference on Machine Learning*, pages 2113–2122, 2015). 3

[36] Horia Mania, Aurelia Guy, and Benjamin Recht. Simple random search of static linear policies is competitive for reinforcement learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 1800–1809. Curran Associates, Inc., 2018). 2, 3, 4, 5, 7, 8

[37] Pablo Martinez-Gonzalez, Sergiu Oprea, Alberto Garcia-Garcia, Alvaro Jover-Alvarez, Sergio Orts-Escolano, and Jose Garcia-Rodriguez. UnrealROX: An extremely photorealistic virtual reality environment for robotics simulations and synthetic data generation. *ArXiv e-prints*, 2018). 2

[38] Nikolaus Mayer, Eddy Ilg, Philipp Fischer, Caner Hazirbas, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. What makes good synthetic training data for learning disparity and optical flow estimation? *Int. J. Comput. Vision*, 126(9):942–960, Sept. 2018). 2

[39] John McCormac, Ankur Handa, Stefan Leutenegger, and Andrew J. Davison. Scenenet rgb-d: Can 5m synthetic images beat generic imagenet pre-training on indoor segmentation? In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017). 1, 2

[40] Tsendsuren Munkhdalai and Hong Yu. Meta networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 2554–2563. JMLR.org, 2017). 3

[41] Yurii Nesterov and Vladimir Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17(2):527–566, Apr 2017). 5

[42] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VIII*, volume 9912 of *Lecture Notes in Computer Science*, pages 483–499. Springer, 2016). 6, 7, 8

[43] Pascal Paysan, Reinhard Knothe, Brian Amberg, Sami Romdhani, and Thomas Vetter. A 3d face model for pose and illumination invariant face recognition. In Stefano Tubaro and Jean-Luc Dugelay, editors, *Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance, AVSS 2009, 2-4 September 2009, Genova, Italy*, pages 296–301. IEEE Computer Society, 2009). 5, 8

[44] Siyuan Qi, Yixin Zhu, Siyuan Huang, Chenfanfu Jiang, and Song-Chun Zhu. Human-centric indoor scene synthesis using stochastic grammar. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018). 3, 4

[45] Weichao Qiu, Fangwei Zhong, Yi Zhang, Siyuan Qiao, Zihao Xiao, Tae Soo Kim, Yizhou Wang, and Alan Yuille. Unrealcv: Virtual worlds for computer vision. *ACM Multimedia Open Source Software Competition*, 2017). 2

[46] Stephan R. Richter, Zeeshan Hayder, and Vladlen Koltun. Playing for benchmarks. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017). 1, 2

[47] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *Computer Vision – ECCV 2016*, pages 102–118. Springer International Publishing, 2016). 1, 2

[48] Cesar Roberto de Souza, Adrien Gaidon, Yohann Cabon, and Antonio Manuel Lopez. Procedural generation of videos to train deep action recognition networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017). 2

[49] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio Lopez. The SYNTHIA Dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *CVPR*, 2016). 1

[50] Nataniel Ruiz, Samuel Schulter, and Manmohan Chandraker. Learning to simulate. In *International Conference on Learning Representations*, 2019). 1, 3

[51] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning, 2017. 3, 4

[52] Gopal Sharma, Rishabh Goyal, Difan Liu, Evangelos Kalogerakis, and Subhransu Maji. Csgnet: Neural shape parser for constructive solid geometry. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018). 4

[53] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *Computer Vision – ECCV 2012*, pages 746–760. Springer Berlin Heidelberg, 2012). 5, 7

[54] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 567–576, 2015). 2, 7

[55] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. *Proceedings of 29th IEEE Conference on Computer Vision and Pattern Recognition*, 2017). 1, 2, 3, 6, 8

[56] Xingyou Song, Wenbo Gao, Yuxiang Yang, Krzysztof Choromanski, Aldo Pacchiano, and Yunhao Tang. Es-maml: Simple hessian-free meta learning. In *ICLR*, 2020). 3, 4

[57] T. Tieleman and G. Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012. 5

[58] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30, Sep. 2017). 2

[59] Gül Varol, Javier Romero, Xavier Martin, Naureen Mahmood, Michael J. Black, Ivan Laptev, and Cordelia Schmid. Learning from synthetic humans. In *CVPR*, 2017). 1, 2

[60] V. S. R. Veeravasarapu, Constantin A. Rothkopf, and Visvanathan Ramesh. Adversarially tuned scene generation. *CoRR*, abs/1701.00405, 2017). 3

[61] Jiajun Wu, Joshua B. Tenenbaum, and Pushmeet Kohli. Neural scene de-rendering. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017). 3

[62] Yi Wu, Yuxin Wu, Georgia Gkioxari, and Yuandong Tian. Building generalizable agents with a realistic and rich 3d environment. In *ICLR (Workshop)*. OpenReview.net, 2018). 2

[63] Fei Xia, Amir R. Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: Real-world perception for embodied agents. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018). 1, 2

[64] Yu Xiang, Wonhui Kim, Wei Chen, Jingwei Ji, Christopher Choy, Hao Su, Roozbeh Mottaghi, Leonidas Guibas, and Silvio Savarese. Objectnet3d: A large scale database for 3d object recognition. In *European Conference Computer Vision (ECCV)*, 2016). 2

[65] Dawei Yang and Jia Deng. Shape from shading through shape evolution. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018). 1, 2, 3, 5, 6, 7

[66] Yi-Ting Yeh, Lingfeng Yang, Matthew Watson, Noah D. Goodman, and Pat Hanrahan. Synthesizing open worlds with constraints using locally annealed reversible jump mcmc. *ACM Trans. Graph.*, 31(4):56:1–56:11, July 2012). 1, 2

[67] Yinda Zhang, Shuran Song, Ersin Yumer, Manolis Savva, Joon-Young Lee, Hailin Jin, and Thomas Funkhouser. Physically-based rendering for indoor scene understanding using convolutional neural networks. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017). 1, 2, 6, 7

[68] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip Torr. Conditional random fields as recurrent neural networks. In *ICCV*, 2015). 3

[69] Xiangyu Zhu, Zhen Lei, Xiaoming Liu, Hailin Shi, and Stan Z. Li. Face alignment across large poses: A 3d solution. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 146–155. IEEE Computer Society, 2016). 7

[70] Xiangyu Zhu, Zhen Lei, Junjie Yan, Dong Yi, and Stan Z. Li. High-fidelity pose and expression normalization for face recognition in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 787–796. IEEE Computer Society, 2015). 7